AUTOMATED CLINICAL SYSTEM FOR CHROMOSOME ANALYSIS

Inventors: Kenneth R. Castleman, Glendale; Howard J. Frieden, Pasadena; Elbert T. Johnson, La Crescenta; Paul A. Renale, Pasadena; Raymond J. Wall, Glendale, all of Calif.

Assignee: The United States of America as represented by the Administrator of the National Aeronautics & Space Administration, Washington, D.C.

Appl. No.: 687,251
Filed: May 17, 1976

Int. Cl. G06F 15/42

U.S. Cl. 364/300, 128/2 R; 364/120; 364/415; 364/900

Field of Search 444/1; 128/1 R, 2 R; 356/39; 235/151.3, 151; 364/200, 300, 900, 413, 415-417, 555, 496, 497, 120

Abstract

An automatic chromosome analysis system is provided wherein a suitably prepared slide with chromosome spreads thereon is placed on the stage of an automated microscope. The automated microscope stage is computer operated to move the slide to enable detection of chromosome spreads on the slide. The X and Y location of each chromosome spread that is detected is stored. At the conclusion of this searching operation, the computer directs the microscope to again sequence through the chromosome spread locations in response to the stored X and Y locations. At this time an operator can view these spreads to determine which ones are worthwhile and which ones are not. He is provided with an accept-reject switch. The microscope stage thereafter again sequences through only the accepted chromosome spreads, and this time a digital photograph of each of the chromosome spreads is made and entered into the computer storage. The computer thereafter measures the chromosomes in a spread, classifies them by group or by type and also prepares a digital karyotype image. This image is converted to analog form, displayed and printed out and constitutes a primary output of the system. Chromosome measurement data is filed in an interactive data base for subsequent statistical analysis. The computer system can also prepare a patient report summarizing the result of the analysis and listing suspected abnormalities.

15 Claims, 4 Drawing Figures

References Cited

U.S. PATENT DOCUMENTS
3,833,796 9/1974 Feter et al. 444/1

OTHER PUBLICATIONS
Golab, T.J.; MACDAC - An Inexpensive and Complete Biomedical Input and Output Display System; Proc. 23rd ACEMB; Nov. 1970.

Primary Examiner—Charles E. Atkinson
AUTOMATED CLINICAL SYSTEM FOR CHROMOSOME ANALYSIS

ORIGIN OF THE INVENTION

The invention described herein was made in the performance of work under a NASA contract and is subject to the provisions of Section 305 of the National Aeronautics and Space Act of 1958, Public Law 85-568 (72 Stat. 435; 42 USC 2457).

BACKGROUND OF THE INVENTION

This invention relates to automated medical analysis equipment and more particularly to improvements therein.

Since the introduction of a method allowing microscopic examination of individual human chromosomes, the karyotype has emerged as a tool of increasing diagnostic value. Under microscopic examination, the chromosomes, from a somatic cell in the metaphase stage of cell division, appear in scattered disarray. The karyotype is a systematic grouping of metaphase chromosomes from a single cell. This grouping was conceived to assist the geneticist in the identification of individual chromosomes. In normal humans, the 46 chromosomes can be reliably ordered into 24 types (seven groups). The diagnostic value of the karyotype is predicated upon the existence of a consistent pattern in normal patients and the correlation of certain chromosomal aberrations with specific clinical observations. There are two types of chromosomal irregularities: numerical and structural. Numerical aberrations exist when the number of chromosomes in one or more groups differs from the normal case. Structural aberrations manifest themselves in many forms, some presumably unobserved as yet. Those which presently merit nomenclature, amongst others, include variations in arm length and centromere position.

At present, manual karyotyping is so tedious and expensive that its general application is usually limited to those situations involving a suspected abnormality. In these circumstances, the clinical evidence is often so overpowering that the karyotype serves primarily as a corroborative tool. In addition, manual karyotyping offers little prospect of quantitative data. It is desirable to extend karyotype analysis to the clinically asymptomatic situation. For example, screening all newborns by karyotype may detect certain inherited disorders long before clinical symptoms appear. As the potential and reliability of the karyotype improves, fetal karyotyping through amniocentesis may become a routine part of prenatal care. Screening studies on large populations offer the potential of uncovering the effects of industrial and environmental poisons, aging, and long term low dosage ionizing radiations. These factors may manifest themselves in subtle structural aberrations requiring detailed analysis of the chromosome morphology. The ability to process cells rapidly and inexpensively would also aid in the detection of mosaicism, in which two or more cytogenetically distinct lines of cells exist in the individual.

There are certain functional requirements for an automated chromosome analysis system which should be met before widespread acceptance thereof can be anticipated. One of these is that the system should be compatible with current practice producing results compatible with those obtained with the present manual system of analysis. Further, the system should provide significant time savings in processing cytogenetic specimens without sacrificing accuracy. Its cost should not be prohibitive and it should be accurate.

OBJECTS AND SUMMARY OF THE INVENTION

An object of this invention is to provide a system for automating chromosome karyotyping.

Another object of this invention is to provide an automated chromosome analysis system which is compatible with current practice.

Still another object of this present invention is one which operates rapidly without sacrificing accuracy, and which reduces the cost.

The foregoing and other objects of the invention are achieved in a system wherein a previously prepared slide, which has a number of chromosome spreads is mounted on the stage of a microscope. Under instructions of a computer, a search of the slide is undertaken to locate the various chromosome spreads on the slide. The spread locations are stored. At the end of the slide search, it is again initiated with the stage of the microscope being stopped at each chromosome spread location, to enable a human observer to inspect the chromosome spread and determine whether it is acceptable for the purpose of analysis or not. If it is not acceptable the observer pushes a button whereby its location is removed from the spread storage list and the microscope stage is moved to the next location. The microscope automatically focuses at each location.

After all of the chromosome spread locations have been inspected, the microscope is again actuated to move its stage so that each one of the accepted chromosome spreads are passed under the optics of the microscope for the purpose of enabling digital pictures of the various spreads to be generated and stored in memory. The computer then proceeds to locate and analyze the chromosomes in each of the chromosome spreads by measuring the chromosomes, classifying them by group or by type and preparing a digital karyotype image format. This image is then converted to pictorial form and displayed on an image display tube to enable any corrections, if needed. A joystick control is provided so that a cursor, which is produced on the display tube screen can have its position moved whereby the operator points out to the computer objects in the image which he wants removed or chromosomes which need repositioning. Communication with the computer is made using a typewriter. The corrected karyotype image is then printed out. Provision is also made for a printout of the computer analysis of the optical image derived from a slide.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic drawing of a chromosome analysis system, in accordance with this invention.

FIG. 2 is a schematic drawing of the apparatus used to drive the microscope stage.

FIG. 3 is a flow chart illustrative of the searching and detecting operation, in accordance with this invention.

FIG. 4 is a flow chart illustrating the analysis operation in accordance with this invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. 1 is a block schematic diagram of the configuration of the system in accordance with this invention. The preparation of glass slides containing stained meta-
phase chromosome spreads suitable for use with this invention is known. The slides can be prepared by different techniques, such as the homogeneous Giemsa staining technique or by the Trypsin-Giemsa banding technique. Slides are produced with patient identification marks. A specimen slide 10 is placed on the stage 12 of a microscope 14. The stage 12 is motorized, that is it is capable of being driven in the X, Y and Z directions by three stage motors 16. The stage motors may either be controlled via a motor control interface circuit arrangement 18, from a computer 20, or by a joystick 22, when it is enabled to control stage position. A stage keyboard 24 enables an operator to select the mode of operation desired, that is, either computer mode or operator mode. Control of the selection is normally with the operator; however the computer has the option to override the operator mode and/or to inhibit subsequent selection of the operator mode. The mode of operation is displayed by a stage display 26. A typewriter 27 serves to enable entry of data, instructions or queries into the computer or to receive data from the computer.

The configuration of the optical imaging system of the microscope allows the specimen to be viewed simultaneously through a binocular eyepiece 28 and by a television camera 30. The output of the television camera is applied to a monitor 32, to an image digitizing system, 34 and also to circuitry designated as "Spread Detector and Auto Focus Unit" 36.

The Spread Detector and Auto Focus Unit 36 comprises circuitry which serves the function of detecting the presence of a chromosome spread and also for automatically focusing the television camera equipment for performing these functions is described in an article entitled "Metaphase Spread Detection and Focus Using Closed Circuit Television," by Johnson and Goforth, published in The Journal of Histochemistry and Cytochemistry in 1974, by the Histochemical Society, Inc. The Image Digitizing System 34 serves the purpose of digitizing the image being viewed by the television camera which is then entered into computer memory.

The computer 20 processes each digital image, as will be described and arranges each image in a karyotype format. A gray level display system 37 displays a karyotype image the information for which is supplied by the computer. A joystick 38 associated with the gray level display system is used for positioning a cursor at a location on the face of the display system for the purpose of pointing out certain locations or objects in the display to the computer. The computer can then be instructed by the typewriter to correct the object designated by the cursor, i.e. correct or erase, etc.

The hard copy printout of a karyotype image is produced in response to the corrected karyotype information received from the computer. This is applied to the hard copy interface circuit 40. This constitutes a circuit for converting the corrected digital karyotype digital signals to analog form and the associated timing required for enabling the hard copy output device 42, which is essentially a facsimile printer, to print out a karyotype picture from the analog signals.

FIG. 2 is a block schematic diagram illustrating the circuits used in controlling movements of the motorized stage of the microscope. The stage is driven in the X, Y and Z directions respectively by three motors 50, 52 and 54. The Z motor is driven in the +Z or -Z direction by signals from the computer in response to focus drive apparatus 56 in the Spread Detector and Autofo-
represents the distance along the x axis, the microscope stage must move and also represents the number of pulses required to complete such a move. This number is entered into the x motor drive register 98 by the computer. The pulse generator 81 is then instructed to commence supplying pulses, and if and gate 78 is enabled, as it should be in the computer mode, then these pulses are applied through and gate 80 to the X motor drive register 98, causing it to count down towards zero. These pulses from the pulse generator 81 are also applied to all of the Y gates 82, through 86.

Since the motor drive register is in its non-zero state, until it has counted down to zero, the one of the two X and gates 82, 84 which is enabled by the X direction flip flop 90 will be able to apply pulses to the X motor causing it to drive the microscope stage. When the X motor drive register reaches zero, this is detected, and the Y gate which has been applying pulses to the X motor is disabled.

The circuitry for driving the Y motor in the computer mode is the same as the circuitry used for driving the X motor in the computer mode. A Y cumulative position mode register 107 retains the absolute Y position of the microscope stage 12. It is driven in response to the same pulses which are applied to the +Y and -Y inputs to the Y motor 52. This cumulative position is supplied to the computer which determines the location along the Y axis to which it desires the microscope stage to move. This is subtracted from the Y cumulative position information and the difference is applied to the Y motor drive register by the computer. Pulses from the pulse generator 81 are applied to an and gate 110, which is enabled when the Y motor drive register is not in its zero position. These pulses start to drive the Y motor drive register and are also applied to the +Y and -Y and gates 62 and 64. The one of these two which is enabled is determined by the output of the Y flip flop 92.

Accordingly, the Y motor is driven until it reaches the position determined by the computer at which time the Y motor drive register will be at zero. This is detected by the zero detector with the consequent inverter output 104 disabling the one of the two and gates which was enabled by the Y flip flop 92. Also, no further pulses are applied to the Y motor drive register.

Both the X and Y cumulative position register outputs are applied to an X and Y display 112 to be visually displayed. The display comprises apparatus which converts the digital information in the X and Y cumulative position registers into visual information.

FIG. 3 is a flow chart exemplifying the search mode of operation. When the computer starts a search the circuitry shown in FIG. 2 is instructed to move a predetermined amount in the X direction. In the embodiment of the invention which was constructed, each step of motion in either X or Y direction is 10 microns, however each move instruction from the computer in either the X or Y direction is a 160 micron move. The field of view is 200 microns square, and a cell containing chromosomes is on the order of 80 microns in diameter. Accordingly, the instruction by the computer to the circuitry shown in FIG. 2 is to move the stage 160 microns in the X direction and then to institute a delay in order to focus and to permit the spread detector to determine if a spread is present in the field of view. If there is a spread the X-Y coordinates of that location are stored by the computer.

Next, a test is made to see if a limit of X travel which is the end of the X line has been reached. If it has not been reached, then the computer issues an instruction to move the next 160 microns in the X direction. If the end of the X line has been reached then a Y move instruction is issued. At the time of issuing the Y move instruction a test is made to see whether or not a limit of Y travel has been reached. If it has been reached, then the search mode is terminated. If it has not been reached, then an instruction to change the X direction is issued. The computer then moves in the Y direction and changes the X direction of motion by applying a signal to drive the X flip flop 90. Thereafter the microscope stage moves until it reaches the end of the new X line where the foregoing routine recurs.

To clarify the foregoing, in the search mode the microscope stage is moved from the beginning to the end of one line. It is then instructed to move in the Y direction to the end of the adjacent line, and then instructed to move in the X direction back to the beginning of that adjacent line. At the beginning of that adjacent line the microscope stage is moved again in the Y direction one line and then starts again in the X direction towards the end of the line to which it has been moved. The microscope slide is scanned in this manner from top to bottom over the entire area to be searched.

Automatic focusing is carried out each time a chromosome spread is detected.

The television camera scans a spread seen through the microscope and displays this on a monitor 32. The spread may also be viewed through the microscope viewing lens 28. Thus, during the editing mode, the operator can use either or both means for editing.

After the editing mode, the system goes into its scanning mode. It sequences the microscope stage to the locations of the chromosome spreads which have been accepted, automatically focuses the image, and digitizes the spread image into a scan data set (SDS). The digitization is accomplished by the television camera and the image digitizing system 36. This equipment is commercially available and, by way of example, in an embodiment of the invention which was built, this equipment, called the model 108 Computer Eye, was furnished by a company called Spatial Data Systems, Inc. The camera scanned the image at the standard television rate (525 lines, 60 frames per second, interlace 2:1) and produced a video signal which was monitored on the television display. The brightness value in the scan picture is sampled at each point of a 512 by 480 grid and converted to a 7-bit binary number. Points anywhere in the picture may be selected on the program control, or the digitization can proceed through all points in sequence. A cursor is displayed on the monitor to show the points to be digitized.

As soon as the computer detects the presence of a spread image in one of the scan data sets provided by the image digitizing apparatus, it enters into its analysis mode. If the spread image is not to be karyotyped, the chromosomes are located, counted and displayed to the operator for verification. However, if the cell is to be karyotyped, a sequence of programs are executed to effect the karyotype analysis of the cell. These programs locate the chromosomes in a cell, orient them, extract measurements, classify the chromosomes and compose the digital karyotype. After the chromosomes have been isolated, they are displayed on the gray level display device 37 along with sequence numbers to allow the operator to correct cases of chromosome touching.
and fragmentation. The karyotype is also displayed to allow the operator to verify correct classification. After the karyotype has been approved by the operator, it is formatted for output, combined with the spread image and copied into one of the output data sets. The analysis phase processes cell images one at a time from a raw spread image to digital karyotype.

FIG. 4 represents a flow chart illustrating the steps in the analysis phase of operation which have just been described. SDS stands for scan data set. Each chromosome is tagged with a number. Where operator intervention is required, in the case of the spread or karyotype interaction, where break ups are to be fixed or chromosomes are touching, or other problems. The operator moves the interactive joystick to the location of a picture element which required correction.

The operator then, by means of the typewriter 27, types an instruction to the computer in response to which the computer performs the required operation. The gray level display system presents a picture of the chromosome spread which has been organized into the standard karyotype format for ease of diagnosis. Similar pairs of autosomes are collected and numbered with homologous pairs being numbered from 1 - 22 and similar pairs being collected into groups lettered A through G based on similar morphology.

The gray level display device is a 1029 line television monitor driven by a scan converter called a Hughes 639 scan converter. This is provided by the Hughes Aircraft Co. with instructions for its use and its operation.

Hard copy printout is achieved by converting the digital elements into analog signals by the circuits 40, which are applied to the hard copy output device. This comprises essentially a facsimile recorder.

There has accordingly been shown and described above, a novel and useful system for scanning chromosome spreads detecting those spreads; editing the detected spreads. Thereafter digitizing the spread images, analyzing, classifying and placing a spread in a karyotype format. This format is displayed on a gray scale display for final correction. Then a hard copy printout of the corrected karyotype image is provided.

The Appendix that follows provides a specific description of the computer programs as well as a copy of the program, in the FORTRAN language, used on a DEC PDP-11 computer for performing the operations described. The computer is made by Digital Equipment Corporation, One Iran Way, Marlborough, Mass. This is to be considered as exemplary and not as limiting. The program can be translated for use on other types of general purpose computers, made by other manufacturers by those skilled in the art.

The chromosome indentification technique, that is a general description of a method of chromosome identification which was implemented by the programming is described in a volume entitled Chromosome Identification edited by Torbjorn, Casperson and Lore Zech, which is a publication of Hansens of the Twenty Third a Noble Symposia published by the Academic Press in 1973. The chromosome analysis procedure which was implemented by the programming is generally described in Perspectives in Genetics, edited by S. W. Wright et al., and published in 1972, by Charles C. Thomas.

**APPENDIX**

The computer has a core memory and the software fits into 64K bytes of core memory. It also has three disk drives each of which has two 2.5 million byte disk cartridges. The core memory is partitioned so that it can simultaneously perform three tasks; slide search, chromosome spread analysis on a recorded digital picture of a chromosome spread; and pictorial output generation of an analyzed chromosome spread. All software runs under a disk operating system monitor supplied by the manufacturer, as well as maintenance and interactive statistical analysis. Automatic karyotyping runs under a specially written supervisor (CALMS).

**CALMS**

Purpose: Clinical ALMS supervisor - Controls slide search, scanning, analysis and hardcopy output of karyotypes or counts.

The CALMS supervisor controls the three "partitionings", search, analysis, and hardcopy.

The supervisor occupies core from 30000 to 37777 and includes the hardcopy driver. The search partition is permanently resident at locations 40000 to 46777. System subroutines are permanently resident at locations 47000 to 57777.

Scan and analysis consist of 20 phases that reside on disk in core-image format. One phase at a time is loaded into locations 60000 to 156777 and called by the supervisor. (Locations 157000 to 157775 may be used for COMMON storage.)

Each analysis phase has a unique identifying number, from 1-20. Scan is phase 1, binary is phase 2, etc. The data set CALMS,OVR is used to store the phases. Each phase required 63 blocks. Program OVB is used to store a phase in CALMS,OVR after it has been linked with a bottom switch of 60000.

The following batch stream builds scan (phase 1):

```
SJOB [2,2]
SR LINK
#DK5:PHASE/CO.LP:/SH<CALMS,STB,<SCAN,EXIT,FTNLIB/B:60000/E
SR OVB
*1
SFI
```

Similar batch streams build the other phases.

An Analysis phase is loaded into core and then called as if it were a subroutine. The disk unit, file name and extension of the current scan data set are passed as parameters. The phase must return to the CALMS supervisor when it is finished. This can be done with a return statement in a subroutine or a call to the CALMSEXIT subroutine from a main program.

The following symbols are used to define the scan and hardcopy data sets.

- **NSDS** = 3 # of scan data sets
- **NHDS** = 2 # of hardcopy data sets
- **SDU** = 5 scan disk unit (DK3:)
- **HDU** = 3 hardcopy disk unit (DK3:)
- **SDS** = BLKB NSDS if NSDFS contains F (F>0) the first F bytes contain the data set numbers (1=51, 2=52, etc) ordered by time of scan.
- **HDS** = BLKB NHDS each byte gives the status of a particular data set:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>available</td>
</tr>
<tr>
<td>1</td>
<td>in use by mask</td>
</tr>
<tr>
<td>-1</td>
<td>full (ready for hardcopy output)</td>
</tr>
</tbody>
</table>

```
When CALMS calls the MASK phase, it gives the current hardcopy data set as a parameter, instead of the current scan data set. When MASK returns, CALMS calls HCOPY, unless it is already operating. Whenever HCOPY finishes, it checks to see if another hardcopy data set is full, and if so, it starts to process it.

HCOPY and SEARCH are interrupt-driven and must not call any non-reentrant DOS routines, because the routine they call might be the one that was interrupted.

OPEN results in a call to the .INIT routine which is not re-entrant because it gets buffer space for a DDB. Therefore, HCOPY and SEARCH cannot call OPEN.

CALMS initially opens all the hardcopy data sets to find their start block numbers and saves them in the HSBN table.

The CALMS supervisor flow chart illustrates its operation. The ‘idle loop’ starts at S10 and the program will cycle until a spread can be scanned or analyzed, or a special request has been made (See Flow Chart 1).

After a spread is scanned, the program INT1 is called. If the operator only wants to do a count, the spread will not be placed in the analysis queue, and the counting and masking are done at this time.

If the spread is to be karyotyped, it will be added to the analysis queue, but if more spreads are to be scanned (and scan data sets are available), they will be scanned before analysis begins. When all spreads on a slide have been scanned, a new slide can be searched while spreads from the previous slide are karyotyped.

Flow Chart 2 details the flow of data through the various program and data sets during the karyotyping process.

The analysis loop begins at S75, and CALMS stays in this loop until the MASK phase is called (or a restart or abort is requested).

After MASK is called, CALMS starts the hard copy partition (unless it is already running), and returns to the "idle loop."
SEARCH

Purpose: Slide search edit, and focus. SEARCH partition of CALMS

SEARCH controls the microscope stage and handles the interrupts from the special-function keyboard and spread/focus data ready. Initially, CALMS calls SEARCH to set up some of the interrupts. Thereafter, SEARCH is interrupt-driven.

The operator presses SEARCH start on the special-function keyboard to start a search. Patient ID, sex, slide ID, and source are entered from the typewriter, before the SEARCH actually starts. The SEARCH pattern is a boustrophedon. Each step is 160 microns, and 60 horizontal steps are taken, before a vertical step is taken. Thus, the slide is searched in rows.

SEARCH is entered via the spread/focus data ready interrupt, after each step. If the "spread" bit is on, the X and Y values are saved in the spread queue, SPQ. SEARCH will perform an auto-focus whenever 15 steps have elapsed since the last auto-focus, and there is something to focus on.

The operator can manually halt the search with the SEARCH halt key. He can then move to another area of the slide and resume the search by pressing SEARCH resume. SEARCH disables the spread/focus data ready interrupt when it receives a SEARCH hlt interrupt, and re-enables it for SEARCH resume.

The SEARCH is terminated when 300 spreads have been found, or when the operator presses the edit-start key.

Normally, both SEARCH and edit are done at 63X, and there is only one edit. (There is also an option to search at 40X and do a "low-magnification" edit at 40X, followed by a "high magnification" edit at 100X).

Edit moves the stage to each spread location saved in the spread queue and initiates an auto-focus sequence. The operator can reject the spread before the auto-focus sequence finishes, and edit will immediately move to the next spread in the queue. If the auto-focus sequence finishes, edit turns on the operator action light and waits for the operator to press "accept," "reject," "next" or "last."

The operator normally centers the spreads he wishes to accept. When the accept key is pressed, edit saves the X and Y values in the same place in the queue and flags them by setting them negative. It also increments the rating for the spread by one each time the accept key is pressed. The ratings are stored in a byte table named SPR.

When the reject key is pressed, spread rating is zeroed and the X and Y values are made positive. The stage is then moved to the next spread.

The next key causes edit to move to the next spread without altering the accept-reject state. The last key causes edit to move to the previous spread.

The edit finishes when the end of the queue is reached, or when the operator presses edit-end. The queue is then sorted according to rating. The highest rated spread is moved to the start of the queue, and the stage is moved to this spread in preparation for the first scan. At this point, the queue and other critical information is saved, by calling WPARAM for phase one. This allows a scan restart at a later time.

GNSTS is the entry point for the "get next spread to scan" subroutine. It initiates a motor move and auto-focus on the next spread in the scan queue. CSPQ contains the current location. It also enables a spiral search.
The spiral search is useful in locating a spread close to known coordinates. It takes steps of 50 microns in a spiral pattern and focuses when there is something to focus on. The operator can halt it or resume it with the search halt and resume keys.

The focus routine can be entered via the focus key interrupt, or it can be called as a subroutine via a simulated interrupt. It operates by initiating motor moves and executing an RTI instruction, after setting up the spread/focus data ready interrupt. When the move is completed, it is reentered and compares the new focus value with the old one to see what the next move should be. Each lens has an initial and final step based on its magnification. The step size is decreased until the final step size is reached. The focus flow chart illustrates the algorithm.

See Flow Chart 3.

FLOW CHART 3  Focus

SCAN

Purpose: To scan a picture onto a disk, display it on the gray scale, and calculate sector thresholds.

SCAN uses the SDS data camera to scan a picture to their critical nature. When the hardcopy is running, the SCAN takes several seconds longer.

SCAN converts pixel values of 0 to 1, and values of 127 to 126. This allows cut and join lines to be differentiated from normal data values.
SCAN stores the source code, patient ID and sex, slide ID, X and Y values, date and time of scan into the label of the output data set. The operator may enter additional information as the SCAN ID.

**BINARY**

*Purpose:* To segment the chromosomes and generate an edge file containing the end point coordinates for each segmented chromosome.

BINARY will read in a scanned spread and assemble a reduced core image of binary sample points. Each sample point is obtained by averaging a $2 \times 2$ pixel area. If this average is above the sector threshold, then the sample point is recorded as a 1-bit. Otherwise, a zero bit is recorded. Thus a digitized spread of $500 \times 480$ pixels is reduced to a $250 \times 240$ grid of sample points. The resulting binary image is surrounded with zero bits to provide a physical boundary to keep the perimeter walker used in the segmentation algorithm from wandering off the edge of the picture. The binary reduction is accomplished through multiple calls to the subroutines QTHR or STHR.

After the binary image has been completely assembled in core, it is scanned line-by-line for chromosomes (1-bits). Each chromosome is "segmented" by recording its starting and ending coordinates on each scan line. Provision also exists for multiple segments to occur on any given line. (See Flow Chart 5)

The segmentation algorithm is implemented in the subroutine SEGMENT, which scans the binary image line-by-line for chromosomes. The search is performed by ROACH, which scans each line from left to right, stopping only when it encounters a chromosome. The starting bit location of any chromosome found is recorded and control is transferred to the subroutine TURTLE. TURTLE will walk counterclockwise around the perimeter of the chromosome, recording the segment end point coordinates as it proceeds. Since the coordinates are recorded in the order they are encountered along the perimeter, they must be rearranged so that they correspond to starting and ending segment coordinates. This is accomplished by sorting the coordinates with the integer sort routine SORTIN.

After a chromosome has been completely segmented, it must be removed from the image in order to prevent ROACH from re-encountering it while scanning the next line. The subroutine ERASE will use the segment coordinates to erase (set all 1-bits to zero) the chromosome from the image.

At this point, chromosomes may be accepted or rejected on the basis of length, width, area and perimeter measurements. All chromosomes thus accepted are recorded in an edge file, (see Flow Chart 5), in a format suitable for input to the phase SKIRT.

**PARAMETERS:** All parameters are optional and may appear in any order except where specified.

- **AREA** followed by two integers representing the minimum and maximum allowable cross sectional areas.
- **EP** followed by two integers representing the minimum and maximum number of perimeter points allowable.
- **LENGTH** followed by two integers representing the minimum and maximum allowable length.
- **SKIRT** followed by an integer representing the width of the skirt (in pixels) to be added to all the chromosomes during the SKIRT phase.

FLOW CHART 4  CALMS segmentation sequence
SKIRT

Purpose: To increase the cross sectional area of each segmented chromosome by recomputing the segment end points and storing the results in a segment file suitable for input to the CHROME phase.

SKIRT will increase the area of each chromosome by extending its boundary outward a uniform distance in all directions. (Flow Chart 6.) Each chromosome is thus skirted with marginal elements to prevent loss of data when the boundaries are recomputed during the rethre- shelving step (ROB phase). SKIRT requires an edge file as input and generates a segment file. (See Flow Chart 7)

The width of the skirt is controlled by the parameter B (number of boundary samples), which is introduced in the BINARY phase. The chromosomes are enlarged by recomputing the segment end points, adding new segments where necessary and merging segments that have grown together. The segment end points are stored in the segment file as triplets (line coordinate and starting and ending sample coordinates).

The background gray levels and thresholds for each chromosome are computed by estimating its center of mass and interpolating over the values for the four nearest sectors.
FLOW CHART 6  Segmented Chromosome Surrounded by Skirt

FLOW CHART 7  Segment File Format

3 = MEAN BACKGROUND GREY LEVEL
T = CHROMOSOME THRESHOLD
BSI = BEGINNING SEGMENT INDEX
ESI = ENDING SEGMENT INDEX
CHROME

Purpose: To gather the gray values of the segmented chromosome from the scanned spread and to store them in a chrome file.

CHROME requires as input a scanned spread and its corresponding segment file. Using the segment end point coordinates, CHROME gathers the gray values for each chromosome and stores them in a chrome file in a format suitable for input to the ROB phase (Flow Chart 8).

In order to avoid re-reading scan lines, chromosomes which appear on the same line are processed concurrently. For this reason, the gray values are stored in an intermediate buffer (CBUF) until an entire chromosome has been processed or the buffer becomes full. When all the gray values for a chromosome have been gathered in CBUF, the gray values and their associated segment end point coordinates are assembled in one or more chromosome records (see Flow Chart 8) and written onto the chrome file. This task is performed by the subroutine WBUF. If the intermediate buffer becomes full, the chromosome occupying the most space in the buffer will be assembled and written out with a call to WBUF.

The intermediate buffer is partitioned into sections of a fixed size and formatted into a list structure to facilitate the allocation and release of buffer areas. The first word of each section contains the buffer index of the next section in the list (= 0 for the last section in the list). The remaining words in each section is used to store gray level information.

Initially, all the sections are formatted into a single list representing all available sections. The next free section index (NFSI) points to the beginning of this list, and is updated whenever sections are removed from or added to the front of the list.

When sections are allocated to a chromosome, indexes pointing to the first and last sections of the chromosome list (FSI and LSI) are maintained in a chromosome directory.

EACH CHROMOSOME RECORD HAS THE FOLLOWING FORMAT:

<table>
<thead>
<tr>
<th>PBW</th>
<th>NSEG</th>
<th>Y1</th>
<th>X1</th>
<th>N1</th>
<th>2N1 SAMPLES</th>
<th>2N1 SAMPLES</th>
<th>Y2</th>
<th>X2</th>
<th>N2</th>
<th>2N2 SAMPLES</th>
<th>2N2 SAMPLES</th>
</tr>
</thead>
</table>

WORD

1. NUMBER OF THE PREVIOUS RECORD WRITTEN FOR THE CHROMOSOME (+0 FOR THE FIRST RECORD OF EACH CHROMOSOME)

2. NUMBER OF SEGMENTS IN THIS RECORD

3 AND 4. LINE AND SAMPLE COORDINATES FOR FIRST SEGMENT

5. NUMBER OF SAMPLE POINTS FOR FIRST SEGMENT (N1)

6 TO 4N1+5. GREY LEVELS FOR FIRST SEGMENT. SINCE EACH SAMPLE POINT REPRESENTS A 2 X 2 PIXEL AREA, 4N1 PIXELS ARE STORED, OR 2N1 PIXELS EACH FROM TWO CONSECUTIVE LINES.

4N1+6 TO . . . SIMILAR INFORMATION FOR THE SECOND SEGMENT

FAKE CHART 8 Chrome File Format
rob

purpose: to rethreshold and resegment the chromosomes and to store them in a chromosome file.

rob requires as input a chromosome file containing segment end point coordinates and gray level information for each chromosome. each chromosome is reassembled in core and its histogram is generated. based on its histogram, a new threshold is computed for the chromosome. using this new threshold, the chromosomes are resegmented by applying algorithms very similar to those used in the binary phase, except that the algorithms are applied on the gray values themselves.

the segmentation process begins with a line by line scan for pixels above the computed threshold. this task is performed by the subroutine iseg, which records the location of any objects that it finds. the task of tracking the perimeter of the object is performed by the subroutine sot (son of turtle), which records the segment end point coordinates as it walks around the object on a counterclockwise direction. the coordinates are sorted to correspond to segment end points by the subroutine sortin.

the segments are examined for pixels lying within the original boundary for the chromosome established in the binary phase. if this search fails (i.e., if the object lies completely in the skirt), the object is rejected. this is necessary to prevent fragments of neighboring chromosomes from being included in the chromosome file.

the object is then assembled into one or more chromosome records and the space it occupied in core is zeroed out. the task of moving the object is performed by the subroutine remove, which may optionally contrast stretch the gray values for maximal display. the object may be rejected at this point based on area measurements. the assembled chromosome records are written onto the chromosome file and the object entered into a chromosome directory.

the first record of the chromosome file contains the following:

<table>
<thead>
<tr>
<th>word</th>
<th>purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>number of chromosomes (maximum of 60)</td>
</tr>
</tbody>
</table>
| 2-241| chromosome directory containing the following four numbers for each chromosome:
|      | fbw = first chromosome record written |
|      | ymin = minimum line coordinate |
|      | xmin = minimum sample coordinate |
| 242-302| thresholds used for each chromosome |

the second record contains the following:

<table>
<thead>
<tr>
<th>word</th>
<th>purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-60</td>
<td>line coordinate of an internal point for each chromosome</td>
</tr>
<tr>
<td>61-120</td>
<td>sample coordinate of an internal point</td>
</tr>
<tr>
<td>121-180</td>
<td>perimeter measure for each chromosome</td>
</tr>
</tbody>
</table>

the remainder of the file is composed of one or more contiguous chromosome records for each chromosome. each such record will contain the following:

<table>
<thead>
<tr>
<th>word</th>
<th>purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>number of segments in this record</td>
</tr>
<tr>
<td>2</td>
<td>line coordinate for first segment</td>
</tr>
<tr>
<td>3</td>
<td>sample coordinate for first segment</td>
</tr>
<tr>
<td>4</td>
<td>number of samples (n) in first segment</td>
</tr>
<tr>
<td>5 to n+4</td>
<td>n gray levels for this segment</td>
</tr>
<tr>
<td>n+5</td>
<td>line coordinate for second segment</td>
</tr>
</tbody>
</table>

the chromosome directory (rec 1 and rec 2) has the following format:

<table>
<thead>
<tr>
<th>integer</th>
<th>purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>chdir (15, 60)</td>
<td>chdir (15, 30) on rec 1</td>
</tr>
<tr>
<td>chdir (13, 30)</td>
<td>chdir (15, 60) on rec 2</td>
</tr>
</tbody>
</table>

each entry is as follows:
The classification table can be supplied to CLASFY with the OS parameter, followed by the object numbers for the slots. In this case, CLASFY sets up the karyotype format but does not do any classifications.

**KTYPE**

**Purpose:** Builds KARYOGRAM and displays it on the gray scale.

KTYPE utilizes the chromosome directory, the classification tables and the rotated images on RCR to build the karyogram on KGM.

IBUF is a 20480 byte buffer that holds one record for each object on the current line. Since the record length is 1024 bytes, twenty objects can be accommodated.

KTYPE builds the karyogram one line at a time, inserting line segments from IBUF, object numbers, centromere marks and slot ID's at the appropriate time in the appropriate place.

**INT2**

Purpose: Provide operator interaction to fix karyotype errors.

INT2 is called after the chromosomes have been measured, classified, and displayed as a karyotype. The operator can correct rotation errors, centromere errors, and classification errors. The corrections are normally done in the above order, since a rotation error usually causes a centromere error and a centromere error usually causes a classification error.

INT2 uses WPARAM to write parameters to MOB and CLASFY and uses RPARAM to read them back to see what has previously occurred. If the operator corrects rotation and centromere errors and does not move any chromosomes around, INT2 will allow CLASFY to reclassify on the basis of the new measurements. After the operator starts making moves, INT2 tells CLASFY what the karyotype should look like and does not allow an automatic reclassification.

**RESEL**

Purpose: Save information on measured chromosomes.

RESEL uses the MOB output to rearrange the measurements in order of type and store them in KDATA (BDATA for banded spreads). It also writes the patient report line in PDATA.
FOUR file, RCR. Length and centromeric index are passed to and saved in records along the line is used. The window of eight points. The input to BAND is the rotated chromosome file, RCR, for each chromosome, regarding its normality.

For bent chromosomes, BAND curve fits a parabola, and using three points perpendicular to the parabola, and using three points perpendicular to the parabola. The classification results are written as parameters for CLASFY.

BAND Purpose: To find the waveform and calculate Fourier coefficients for banded chromosomes.

The input to BAND is the rotated chromosome file, RCR. For each chromosome, BAND first decides if the chromosome is straight or bent. For bent chromosomes, BAND curve fits a parabola to the boundary points and recalculates the chromosome length. The waveform is determined by sampling along the parabola, and using three points perpendicular to the slope of the parabola.

For straight chromosomes, the waveform is determined by sampling along each line, using a moving window of eight points. The maximum value found along the line is used.

A and B Fourier coefficients are then computed from the waveform values for eight harmonics. From these, C (Amplitudes) and PHI (Phase Angle) are computed and saved in records 4–7 of the rotated chromosome file, RCR. Length and centromeric index are passed to FOUR as parameters.

BAND recognizes the following parameters:

**A** and **B**: Fourier coefficients for banded chromosomes.

**C**: Amplitudes for types 1–24, 25, 26.

**PHI**: Phase angle for types 1–24, 25, 26.

**NC**: Number of chromosomes in the karyotype.

**LSUM**: Length sum (not normalized for 46).

**ISUM**: IOD sum (not normalized for 46).

**ASUM**: Area sum (not normalized for 46).

**UNUSED**: A placeholder for future use.

**LEN**: Length (adjusted for bend) un-normalized.

**IOD**: IOD/8 - Un-normalized.

**AREA**: Area.

**CIL**: CIL.

**CID**: CID.

**CIA**: CIA.

**ONL**: # lines for each object.

**ONS**: # samples.

**ORIV**: # objects.

**QB**: Quick BAND — process only the C group.

FOUR

QB: Quick BAND — process only the C group.

FOUR Purpose: To classify banded chromosomes using Fourier coefficients, length and centromeric index.

The inputs to FOUR are the classification table of means and variances, and the banded chromosome measurements. FOUR computes the likelihood that each chromosome belongs to each of the chromosome types, using 14 measurements — length, centromeric index by area, C(2) to C(8), and PHI (2) to PHI (6).

If a chromosome's length or centromeric index differs by more than 6 S.D. for a particular chromosome type, that type is ruled out for that chromosome. Similarly, if the C sum or PHI sum exceeds 7 S.D. or the total sum exceeds 8 S.D., that type is excluded for that chromosome.

The chromosomes are then classified in order of likelihood, subject to group membership rules. This procedure may leave some chromosomes unclassified, since certain types may have been excluded for certain chromosomes.

Using the unclassified chromosomes, the most likely classification is found, say chromosome i belongs to group j. The chromosomes in group j are then examined to see if one of them can be moved to another group that is not yet full. If so, the most likely move is made.

The classification results are written as parameters for CLASFY.

PREP Purpose: To print the patient report

PREP reads the patient report records that were written on disk by RESEL, and prints them on the line printer. If the parameter PL is used, the records for different patients are separated by a double space. Otherwise, each patient's report is on a separate page.

ABNORM Purpose: To print information on abnormal chromosomes ABNORM is called only when console switch 10 is down. It reads the rotated chromosome file, RCR, and examines the profile of each chromosome to determine centromere information. It then prints a message for each chromosome, regarding its normality.
KFIX

Purpose: Syntactical classification within B,D,F, and G groups.

KFIX is the final phase of the hybrid classifier for banded chromosomes, and operates as follows:

1. Take the 4 chromosomes in the G group. Measure the position along the length of the chromosome of the brightest band. The 2 chromosomes with the brightest band closest to the center correspond to the G-21 the other two are the G-22.

2. Take the 6 chromosomes in the D group. Measure the ratio of average IOD in the upper half of the chromosome so that of the lower half of the chromosome. The two smallest values correspond to the D-13, the two largest values correspond to the D-15 and the remaining two are the B-14.

3. Separate the F-19 and F-20 chromosome by IOD. The two chromosomes with the smallest integrated optical density are the F-19's.

4. Take the B group. Measure the average IOD between the centromere position and a distance along the long arm equal to the short arm length or to the midpoint of the chromosome, whichever is shorter. The two chromosomes with the largest value correspond the B-4.

MVIO

Purpose: To read and write contiguous files with automatic double buffering, blocking, and random or sequential access to lines of data.

The calling sequences for the six entry points are as follows:

CALL OPEN (MVB,BUFSIZE,MODE,LNAME)
CALL GET (MVB,LINE,INDEX[,NORA])
CALL PUT (MVB,LINE,INDEX)

CALL CLOSE (MVB)
CALL READ (MVB,LINE,INDEX,LOC)
CALL WRITE (MVB,LINE,LOC)

An OPEN call is required before any GET or PUT calls can be made to a dataset. The user must provide core space large enough to hold all the necessary control blocks and control information, as well as the data that is to be read or written. This allows MVIO to be re-entrant. In addition, no space is wasted on unused data sets, as would happen if MVIO contained storage for a fixed number of data sets.

The control information can be considered as a "mini-VICAR-block" or MVB. Each MVB is 56 bytes long and contains a tran block, link block, filename block and the information required by MVIO. (See Flow Chart 9)

The MVB is followed by one or two buffers to hold the data that is read or written. Each buffer is a multiple of the R1 disk block size (512 bytes). An entire buffer is normally read from or written onto the disk with a single access. This greatly increases the effective transfer rate. When two buffers are provided, MVIO allows the user to overlap computation with the disk input/output.

GET and PUT are called to obtain the index relative to the start of the MVB for the desired record in the data set. Thus the data does not have to be moved from one buffer to another. On most "get" calls, the requested line will already be in one of the buffers and MVIO simply returns the index without any physical I/O required.

PUT is called to obtain the index of where to store the line that is to be written. MVIO initiates physical I/O when a buffer has been completely filled. A "close" call is required to write any data left in a buffer by earlier "put" calls.
A logical record (or line) can be smaller than, larger than, or the same size as the physical block size (512 bytes). As previously noted, each buffer is a multiple of 512 bytes in length, but the buffer must also be large enough to hold a complete logical record.

MVIO (See Flow Chart 9) is normally used in conjunction with the label subroutines GLABEL and PLABEL, which are described in more detail elsewhere. They set up some of the fields in the MVB when the picture data is preceded by a label.

The fields in the calling sequences are defined as follows:

- MVIO is the location of the MVB for the data set. The user must reserve enough core for his buffers immediately following the MVIO.
- BUFSIZ is the size of each buffer (multiple of 512 bytes).
- DBFLAG is the flag for double buffering, 0 = no double buffering (one buffer), 1 = double buffer (two buffers).
- MODE is defined as:

<table>
<thead>
<tr>
<th>MODE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disk Input</td>
</tr>
<tr>
<td>1</td>
<td>Disk Output</td>
</tr>
<tr>
<td>2</td>
<td>Disk Update</td>
</tr>
<tr>
<td>3</td>
<td>Tape Input</td>
</tr>
<tr>
<td>4, 5</td>
<td>Tape Output, 4, and 5 are functionally equivalent</td>
</tr>
</tbody>
</table>

LNAME is a three character logical name for the data set that can be used to assign it to a file with the $AS command. (Subroutine AFILE can be used to make a default assignment).

LINE is the desired line number, positive for data lines, negative for label records, and zero for the "next" line.

INDEX is the offset in bytes from the start of the MVB to the requested line. In Fortran, when MVIO is defined as a byte array, sample J of the requested line is at MVIO(J+INDEX). In Marco, MVIO+INDEX is the location of the first sample of the requested line.

INDEX is set to zero for an end-of-file read from tape.

NORA is an optional parameter to prevent read-ahead.
READ and WRITE can be used when RECLLEN = BUFSIZ and single buffering is specified. A line is read into LOC, or written from LOC, without any overlap.

Updating is normally done with a GET and a PUT for the record to be updated. (The same index value will be returned on the GET and PUT.) This insures that other records in the block and other blocks in the buffer will not be changed.

The "get" can be omitted only if all records are "put" sequentially, starting with the first record of a block.

The MVB format is shown below. Word numbers are in decimal, starting at 1 and byte numbers are in octal, and byte numbers are in octal.

<table>
<thead>
<tr>
<th>WORD</th>
<th>BYTE</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>TB</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>+2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>+4</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>+6</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>+8</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>IBN</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>IBA</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>-2</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>LB</td>
</tr>
<tr>
<td>10</td>
<td>22</td>
<td>+2</td>
</tr>
<tr>
<td>11</td>
<td>26</td>
<td>+4</td>
</tr>
<tr>
<td>12</td>
<td>30</td>
<td>CRC</td>
</tr>
<tr>
<td>13</td>
<td>32</td>
<td>-2</td>
</tr>
<tr>
<td>14</td>
<td>34</td>
<td>FB</td>
</tr>
<tr>
<td>15</td>
<td>36</td>
<td>+2</td>
</tr>
<tr>
<td>16</td>
<td>40</td>
<td>+4</td>
</tr>
<tr>
<td>17</td>
<td>42</td>
<td>+6</td>
</tr>
<tr>
<td>18</td>
<td>44</td>
<td>WNT</td>
</tr>
<tr>
<td>19</td>
<td>46</td>
<td>SBN</td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>NBF</td>
</tr>
<tr>
<td>21</td>
<td>52</td>
<td>NLR</td>
</tr>
<tr>
<td>22</td>
<td>54</td>
<td>RECLLEN</td>
</tr>
<tr>
<td>23</td>
<td>56</td>
<td>BLKSIZ</td>
</tr>
<tr>
<td>24</td>
<td>60</td>
<td>BUFSIZ</td>
</tr>
<tr>
<td>25</td>
<td>62</td>
<td>BPB</td>
</tr>
<tr>
<td>26</td>
<td>64</td>
<td>DBF/MD</td>
</tr>
<tr>
<td>27</td>
<td>66</td>
<td>WFLAG</td>
</tr>
</tbody>
</table>

LABEL

Callable Entry Point Names: GLABEL, PLABEL
Purpose: To obtain and store label information on disk data sets.

These subroutines are used in conjunction with MVIO, when processing labeled data sets. They obtain or store the label parameters, and also set up the SPAR table of system parameters. The SPAR table is set up after the OPEN call for the MVB.

CALL GLABEL (MVB, SPAR, INDEX) to get a label

CALL PLABEL (MVB, SPAR, LABEL) to put a label

MVB is the mini-VICAR block for the data set.
SPAR is a five word table of system parameters:
SPAR (1) = NL, # of Lines of picture data
SPAR (2) = BPL, Bytes per line
SPAR (3) = BPE, Bits per element
SPAR (4) = NLR, # of label records
SPAR (5) = BLKSIZ, Block size

GLABEL transfers the label information into SPAR. PLABEL transfers the data in SPAR to the label.

Index is the offset from MVIO to the first byte of the label, as returned by 'get'. Label is the location of the label to be output.

Example:

INTEGER SPAR (5)
BYTE A (4200), B (4200)
CALL OPEN (A,2048,1,IBM')
CALL OPEN (B,2048,1,IBM')
CALL GLABEL (A,SPAR,1)
CALL PLABEL (B,SPAR, (1A + 1))

Open A for input
Open B for output
Get label from A
Put label to B

DOS can also be used with an unlabeled data set. It will return NL as the # blocks in the data set
BPL = 512
BPE = 8
NLR = 0

These routines are re-entrant, except when GLABEL is used with an unlabeled data set.

EXIT

Purpose: To replace the FORTRAN exit and error subroutines and save 1260 bytes of core.
EXIT can be used to save core after a FORTRAN program has been checked out. ERRA gives a single error message (A367), instead of the individual messages normally given by the FORTRAN error routine ERRA.

When EXIT is called, it will either return to DOS via a .EXIT or return to CALMS via an RTS R5. It makes this decision by checking the value of R5 when EXIT was called. When DOS loads a program, it clears R5, but when CALMS loads a program, R5 is equal to a location within CALMS.

Therefore, CALL EXIT will work for programs running under DOS or under CALMS.

In order to select EXIT.OBJ over the FORTRAN exit subroutine, specify EXIT before FTNLIB in the linker command string. The /SU switch should be used in the FORTRAN command string to save additional core and time.
All FORTRAN modules of CALMS should be compiled with /SU or they may not fit in core.

SAVER

Purpose: To save and restore registers 0-4 on the stack, for subroutines called with an R5 calling sequence.
SAVER and RESTR provide a convenient way for MACRO subroutines to save and restore registers 0-4 on the stack.
To save registers 0-4:
JSR R4, SAVER
To restore registers 0-4 and return via R5:
JMP RESTR
(SAVER is called via R4, and R5 is not saved.)
AFILE

Purpose: To assign a file to a dataset.

CALL AFILE (MVB, DUNIT, FILPEX, GRP, USR)

MVB is the mini-VICAR block for the dataset.
DUNIT is the disk unit (1 for DK1; 2 for DK2; etc.)
FILPEX is the 6 character filename plus 3 character extension.
GRP is the group number from the UIC
USR is the user number from the UIC.

Example:
CALL AFFILE (A1, 'PIC ', 5,5)
Assigns DK1:PIC[8,5] to MVB A

PARAM

Callable Entry Point Names: PARAM, PARBUF

Purpose: To read free-field parameters from the keyboard and convert them.
PARAM types an *, then reads up to 80 characters from the keyboard, terminated by a carriage return. Parameters are separated by blanks or commas, and can be one of the following:
1. One word integer — negative integers are preceded by a minus sign.
2. Two word alphabetic — first character must be alphabetic. The character string is padded with trailing blanks if less than 4 characters. It is truncated if more than 4 characters.
3. Variable length alphabetic — the string is enclosed in apostrophes. If an odd number of characters, the last word is padded with a trailing blank. If an apostrophe is desired, type two consecutive apostrophes.

The raw character string is saved at 'PARBUF' which is accessible to MACRO programs, but not FORTRAN. However, the user can optionally supply his own PARBUF.

The calling sequence is:
CALL PARAM (NP, PAR, MAXNP [PARBUF])
NP is the number of parameter words that were stored in PAR
MAXNP is the maximum allowable number of parameter words (the size of PAR)
PARBUF is optional 83 byte user buffer for the raw character string.

Example:
INTEGER PAR (20)
CALL PARAM (NP, PAR, 20)
If the user types:
NH, PRINT, 42, -1, 'ABC' 'DEF'

The result is:
PAR (1) = NH
PAR (2) = AA
PAR (3) = PR
PAR (4) = IN
PAR (5) = 42
PAR (6) = -1
PAR (7) = AB
PAR (8) = C'
PAR (9) = DE
PAR (10) = FA
NP = 10

PARAM is reentrant if the user supplies PARBUF. However, an earlier keyboard input request must be satisfied before a later one can be done.

Any program which is entered by an interrupt (such as SEARCH or HCOPY) must check that PBUSY = 0 before calling PARAM. If PBUSY is not zero, the program must signal PARAM that it is waiting to call it, but could not because PARAM was busy. It does this by storing the location for PARAM to transfer control to in 'PINT' or PINT + 2, and then executing an RT1. PARAM will then call the waiting routine with a simulated interrupt, when it has finished with the earlier request.

(QBUSY, PINT method is required under DOS; otherwise the KB driver gets confused).

QPRINT

Purpose: To print a message, with automatic buffering

QPRINT waits for completion of a previous print, moves the message to its own buffer, initiates a print with a TRAN request, and returns to the user.
CALL QPRINT (LOC,[NBYTES])
LOC is the first byte to be printed (carriage control).
NBYTES is the number of bytes to be printed. (maximum — 132).

NBYTES can be omitted if the message is terminated with a "zero" byte.

FORTRAN automatically inserts a zero byte for literal strings and 'ASCII', in MACRO, also does this.

Example:
CALL QPRINT ('single space')
CALL QPRINT (' 0 double space')

TYPE

Purpose: To type a message with automatic buffering

TYPE is similar to QPRINT, except that output is on the keyboard and the maximum message length is 80 bytes.
CALL TYPE (LOC,[NBYTES])
If NBYTES is omitted or zero, a null (0) terminates the message. TYPE inserts a <CR> at the end of the message unless NBYTES = 0.

IV

Purpose: To convert and move logical *1 (byte) data and word data

IV is a function and the other entry points are subroutines. These routines consider bytes to be positive numbers from 0–255. (FORTRAN treats bytes as signed numbers from –128 to +127.)

IV (byte) = Integer value (0-255) of a byte variable
CALL ITL (INT, L1) (INTEGER TO LOGICAL)
Move the low order byte of INT to L1.
CALL MVL (L1, L2, N) (Move LOGICAL)
Move N bytes starting at L1 to L2
CALL MVW (I1, I2, N) (Move WORD)
Move N WORDS starting at I1 to I2
CALL ZIA (1BUF,N) (Zero INTEGER array)
Zero N WORDS starting at IBUF
CALL ITLA (INT,L1,N) (INTEGER to LOGICAL array)
Store INT in N bytes starting at L1
CALL SWAP (I1, I2) Interchange I1 and I2.

Note: RO is not saved by these routines!

TEXT

Purpose: To generate readable characters for labeling of pictorial output.
Each character is generated from a six sample by seven line array of BIT, where a one BIT represents black and a zero bit represents white. The left most of the six samples is always zero. CALL TEXT(INBUF, INCHR, LINE, OUTBUF, SIZE)

INBUF is the location of the input ASCII characters INCHR is the # of input characters LINE is the line number (0-6) of the bit array OUTBUF is the location for the output bytes SIZE is the number of bytes to store for each bit (1, 2, 3, ...) OUTBUF must be at least INCHR*SIZE*6 bytes long

HCPAK

Purpose: To pack and unpack data in hardcopy format (4 bits) and convert from 7 to 8 bit format.

CALL HCPAK (BUF8, BUF4, NE)

BUF8 is the LOC of 8-Bit Data
BUF4 is the LOC of 4-Bit Data

NE is the number of elements to pack or unpack. NW is the number of works to convert from 7-Bit format to 8-Bit format.

RPARAM

Purpose: To pass parameters to a CALMS phase and to assign the next phase to be loaded.

CALL WPARAM (NP, PAR, PNUM)
NP is the number of words of parameters to write
PAR is the location of the parameters
PNUM is the number of the phase which is to receive the parameters

CALL RPARAM (NP, PAR, NPMAX [,PNUM])

To read parameters through the CALMS supervisor. See the PARAM writeup for details. PNUM is an optional phase number used to read another phase's parameters.

CALL APHASE (N)

Normally, phases are loaded in sequence, but APHASE is used to change the sequence.
N is the number of the phase.

DLINE

Purpose: Subroutines used to access the gray scale and cursor, when running under the CALMS supervisor.

CALL DDECLAR to erase the entire gray scale.
CALL DLINE (LOC,Y,X,NS,REPL,ERASE) to write a line of data.
LOC is location of data in core.
Y is line on gray scale (0-1023)
X is starting sample on gray scale (0-1023)
NS is number of samples
REPL is non-zero to replicate samples and lines. (If switch 3 is up, only samples are replicated)
ERASE is positive to erase this line before writing negative to erase only zero to write only.

CALL DWAIT to wait for gray scale ready
CALL CURSOR (Y,X,LIN,SAMP) to read the cursor.
Y is the line on the gray scale (0-1023)
X is the sample on the gray scale (0-1023)

LINE is the picture line (1-512)
SAMP is the picture sample (1-512)
CALL SC sets up the cursor adjustments and saves them on disk as parameters for INT1.
CALL RCA reads in the cursor adjustments from disk.

CALL MCU moves the cursor up one line
CALL MCD moves the cursor down one line
CALL MCL moves the cursor left one sample
CALL MCR moves the cursor right one sample
CALL UDLR (PAR) moves the cursor one step according to whether PAR is a U,D,L, or R.

The cursor adjustment process is required to correct for cursor drift. Two marks are written on the gray scale, one at 32, 32 and one at 992, 992. The operator is requested to move the cursor to these reference positions and the readings are saved. Thereafter, CURSOR performs a linear interpolation on all cursor readings, using the saved values.

There is also a standalone version of DLINE in FTNLIB. It differs from the CALMS version in the way the cursor adjustments are saved and read back.

MCISUB

Purpose: Subroutines to control the MCI and sort the spread queue.

This module consists of miscellaneous subroutines used by SEARCH and CALMS. It also globally defines the MCI registry addresses and interrupt vector locations.

CALL IXYABS (Y,Y) Initiates a motor move to X,Y
CALL IXYREL (XDEL) Initiates a relative X move
CALL IYREL (YDEL) Initiates a relative Y move
CALL LED (CODE,VAL) Puts VAL in the LED whose code is given
CALL MFST (MAGN, FPOS) Stores the magnification and filter position in MAGN and EPOS, and displays the magnification in the magnification LED. It also stores FSTEP, FLAST, XADJ, and YADJ for the lens in use.

CALL UDLR (PAR) moves the cursor one step according to its rating.

ROACH

FUNCTION: ROACH (WI,BI,WORD,EWI,MASK)

ROACH will scan a binary line to locate the first set bit. The scan will begin at W (WI) and end at WORD (EWI). If no set bit is found, the return code is zero. If a set bit is found, WI is returned as the index of the word containing the bit. BI is the bit index within the word (numbered 0, 1, 2, ..., 15). The corresponding bit position within the MASK is also set.

TURTLE

TURTLE is invoked by SEGMENT to walk counterclockwise around the perimeter of objects in the binary picture.

CALL TURTLE (COMMON,EDGE,WORD,MASK,NW)

COMMON is composed of the following six words:
N = the number of segment end points found
PERIM = the number of perimeter points found
YMIN,XMIN, YMAX,XMAX, = the extreme coordinates of the object

EDGE is an integer array into which is placed the coordinates of the detected end points. On input, WORD points to the word containing the first detected bit of the object and MASK designates the bit positions within that word. NW2 is the number of bytes per line.

The TURTLE uses a four point connectivity algorithm in walking around the perimeter. At each step, the TURTLE will examine its four adjacent neighbors, numbered 0,1,2, and 3 (See Flow Chart 10) to determine its new direction. The order in which neighboring samples are examined is predetermined to ensure that the TURTLE will always proceed in a counterclockwise direction (See Flow Chart 11).

Since not all perimeter points are segment end points, the TURTLE uses a decision table to identify the end points. End point determination is based on the TURTLE's previous and current directions, and his conviction that since he is traversing the perimeter in a counterclockwise direction, the chromosome will always remain on his left. Line segments which contain only one sample are recorded twice so that all segments have starting and ending end point coordinates. The TURTLE's decision table is given in Flow Chart 10. The number of times perimeter points are recorded as end point coordinates appears in parentheses.

FLOW CHART 10 Segment and Point Decision Table
SORTIN

SORTIN is invoked by SEGMENT to sort the end point coordinates for a chromosome in the order that they would be encountered while scanning the chromosome line-by-line from left to right. Thus, the short is in ascending order, first by line coordinate, and second by sample coordinate.

CALL SORTIN(EDGE,N,IND)
N is the number of end points.
Upon return, IND=0 if the sort was successful, ≠ 0 if not.
EDGE is a buffer area 4N words long. The first 2N words contain the end point coordinates for the chromosome. The remaining 2N words is used as a work area for the bucket sort routine.

SORTIN uses a byte array to keep track of the ordering of the end points. Because of this, a maximum of 255 end points may be sorted.

ERASE

ERASE is invoked by SEGMENT to remove a chromosome from the binary spread image.
CALL ERASE(WORD,EDGE,AREA,N2,NW)
WORD points to the buffer area containing the binary image.
EDGE points to the end point coordinates for the chromosome.
N2 is the number of coordinates in EDGE (two per coordinate pair).
NW is the number of words per line in the binary image.
**ORIOB**

**Purpose:** Orient objects.

**CALL** ORIOB(IBUF,OBUF,EF,CHDIR,NL,O PROT,RCODE)

ORIOB finds the minimum enclosing rectangle for the object in IBUF using the endpoint table EP and 32 rotations from 0° to 90°. It then rotates the object into OBUF and sets the rotated NL and NS in CHDIR. OPROP specifies any additional rotation desired by the operator.

**ACCSUB**

**Purpose:** Accumulate area and density by sample for rotated object.

**CALL** ACCSUB(BUF,NL,MS,AREAA,DENA,- TAREA,TDEN)

The chromosome is located in BUF and is NL by NS. AREAA is the area accumulator by sample and DENA is the density by sample. TAREA and TDEN are the area and density totals for the object.

**EPROT**

**Purpose:** Rotates chromosome endpoints.

**CALL** EPROT(SIN,COS,XMIN,XMAX,-YMIN,YMAX)

EPROT rotates the endpoints (EP) by the angle specified by SIN and COS. It returns the limits of the enclosing rectangle (XMIN,XMAX, YMIN,YMAX).

**OBROT**

**Purpose:** To rotate objects.

**CALL** OBROT(XMIN,XMAX,YMIN,YMAX,- COS,SIN,IBUF,OBUF)

OBROT rotates the object in IBUF into OBUF. COS, SIN specify the rotation angle and XMIN,XMAX,YMIN,YMAX give the enclosing rectangle of the object in unrotated coordinates.

**CHROUT**

**Purpose:** To stand objects up and move them into the output buffer.

**CALL** CHROUT(IBUF,OBUF,NS,NL,LPB,IP,FLG,BUFSZ)

CHROUT rotates the object in IBUF (NS X NL) by ± 90° in to OBUF for output. LPB gives # of lines that will fit into OBUF, FLG tell which way to rotate object and BUFSZ is NS for IBUF.

**KURSOR**

**Purpose:** To locate information about the karyogram.

**CALL** KURSOR(X,L,S,SLID,SLCL,SLCS,N)

KURSOR is called by the supervisor for the current cursor position.

**CALMS**

**Purpose:** To locate information about the karyogram for the current cursor position.

**CALL** CALMS(IBUF,OBUF,NS,NL,LPB,IP,FLG,BUFSZ)

CALMS is called by KURSOR.

**NLIST**

**CALL** NLIST(BEX,RCALL,EXIT,PAUSE,INIT,TRAN,WAIT,RLSE)

NLIST is called by CALMS.

**SUPERVISOR**

**CALL** SUPERVISOR(FOR THE CLINICAL AUTOMATED LIGHT MICROSCOPE SYSTEM)

SUPERVISOR is called by the supervisor.
CURRENT ORIGIN IS 30000

SOURCE: WORD 8.
MSCAN: WORD 10.
PHUM: WORD -10.
PHNUM: ASCI1 /12345678/. PATIENT IDENTIFICATION
SLNUM: ASCI1 /12341/. SLIDE IDENTIFICATION
SEX: ASCI1 /F/. PATIENT SEX
SNGR: WORD. CURRENT SPREAD NUMBER
CSPQ: WORD. SPQ+.-. CURRENT LOC IN SPQ
SR: BLKb 300. SPREAD RATINGS
SPQ: BLKM 600. SPREAD QUEUE (X.Y)
LPNUM: WORD 10. LENGTH OF PANUM IN BYTES
OLSLNUM: WORD 8. LENGTH OF SLNUM
MPHASE: WORD . NEXT PHASE TO BE CALLED
PAGE

CALS: CALL TYPE(<MSC2. ZERO) . ASK ABOUT RESTART
MOV &S. R1
CLED: CALL LED((LINE. ZERO) . INITIALIZE LEDS
INC LINE
S0: R1. CLED
CALL OPEN, <MVBS. PSIZE. ZERO. ZERO. LMAM> . OPEN OVERLAY DATASET
MOV PSize. MVBS. RECLN
CALL OPEN. <PVBY. PSize. ZERO. TWO. PLNM> . PARAMETER DS
MOV PAR. PSize. MVBS. RECLN
MOV 4. DNPTAB+INTPH+INTPH-2 . SET TO READ 4 CURSOR ADJENTS
MOV 512. HPTAB . ALLOW RESTART
MOV 512. DNTPAB
MOV #HBS. R0 . SET UP THE START BLOCK NUMBERS FOR
MOV #MSNH. R1 . ALL HARD COPY DATA SETS
C05.
MOV R0. FILPEX+1
B13B #60. FILPEX+l . CONVERT TO ASCII
CALL AFILE. <A. IHBU. FILPEX. TG. TG>
CALL OPEN. <A. TWELVE. ZERO. ZERO. LMAME>
MOV A. SSNH. (RL)+ . SAVE THE START BLOCK NUMBER
SUB R0. G05
CALL SEARCH . INITIALIZE CALL TO SEARCH
JSR PC. PPAR . PROCESS RESTART AND PHASE PARAMETERS
TST HSCAN
BNE S10 . IF A SCAN RESTART.

S06. JSR PC. THSCAN . IF HSCAN=0 . ALLOW SEARCH START
/ 1
S10. TSTB SFLAG . 'IDLE LOOP' STARTS HERE
BNE S43 . IF SEARCH RUNNING
TST HSCAN
BEG S40 . IF NO SCAN
S13. MOV #SBD. R0
S15. MOV #SDB. R1
MOV HFSDB. R2 . FULL SCAN DATASETS
BEG S22 . IF ZERO
S20. CMPB R0. (RI)+ . CHECK R0 AGAINST ALL FULL DATASETS
BEG S25 . IF DATASET IS IN USE
SUB R2. S29 . IF MORE TO CHECK
S22. MOV R0. BR1 . OK TO USE THIS DATASET. STORE IT
BR S30 . GO CALL SCAN
BR S40 . IF MORE TO TEST
DR S15 . NO ERROR AVAILABLE
S39. B15 #60. R0 . CONVERT TO ASCII
MOV R0. FILPEX+l . STORE IN FILENAME
MOV #S. FILPEX . AS SN
MOV #SU. BUNI . STORE SCAN DISK UNIT
DEC HSCAN . INSCAN=HSCAN-1
S31. CALL PHASE. <ONE> . SCAN
CALL LED. <GOLED. HSCAN>
TST NPlllASE
BEG S06 . IF SCAN WAS ABORTED
S315. CLR HPTAB+MBPBO+MBPBO-2 . CLEAR MBP PARAMETERS FOR INT1
CALL PHASE+INTPH . CALL INT1 FOR THE PRE-FOB ADL
CMP HNPASE. #1 . TEST FOR RESCAN OR ABORT
BEG S31 . IF RESCAN
BF T S06 . IF SPREAD WAS ABORTED IN INT1
CNP HNPASE. #OBPH
BNE S33 . IF NOT A COUNT
CALL PHASE. HNPASE . CALL NOB
CMP NPHASE, #MASKPH
BNE S315 ; BR IF NOT MASK

S32: JSR PC,TMSCAN ; IF NSCAN = 0, ALLOW SEARCH START
JMP C15 ; GO CALL MASK2

S33: CMP NPHASE, #MASKPH
BNE S32 ; BR IF MASK
TST HPTAB+BINPH+BINPH-2
BEQ S39 ; BR IF NOT QUICK OPTION

CALL PHASE,NPHASE ; CALL BINARY
BR S315

S38: INC HFSDS ; INCREMENT ANALYSIS QUEUE
CALL LED,(#QLED,HFSBS) ; UPDATE ANALYSIS QUEUE
BR S06

S40: TSTB FFLAG ; SEE IF PATIENT REPORT WAS REQUESTED
BEQ S42 ; BR IF NOT
CLRB FFLAG

CALL PHASE,NPHASE ; CALL THE PATIENT REPORTER
TSTB EFLAG ; SEE IF TEXT EDITOR WAS REQUESTED
BEQ S45 ; BR IF NOT
CLPB EFLAG

CALL PHASE,NPHASE ; CALL THE EDITOR
TSTB RFLAG ; SEE IF FORTN WAS REQUESTED
BEQ S46 ; BR IF NOT
CLRB RFLAG

JR PC,RSTRT

S47: MOV $#SDS.R1 ; SEE IF READY FOR ANALYSIS
MOV HFSBS,R2
BEQ JPS10 ; BR IF NO FULL SCAN DATASETS
MOV #S.FILPEX
MOV $#SDU.DUMIT
MOV #R1,FILPEX+1
MOV #R1,R0
ISAVE DATA SET # IN R0
BIS #R0,FILPEX+1
ICMVR TO ASCII
DEC HFSDS
DEC R2
BEQ S60 ; BR IF NO DATASET NUMBERS TO MOVE
INC R1
S0B R2,S59

S60: MOV FPHASE,NPHASE ; START ANALYSIS WITH PHASE 2 OR DEBUG PHASE
MOV #DNPTAB.R0
MOV #DNPTAB.R1
MOV #NPH,R2

S65: MOV (R0)+,(R1)+ ; MOVE DNPTAB TO HPTAB
S03 R2,S55

S75: CALL PHASE,(NPHASE) ; CALL AN ANALYSIS PHASE
TSTB RFLAG ; TEST RESTART FLAG
BEQ S75 ; BR IF NOT SET
CLRB RFLAG
JSR PC,RSTRT

S80: TST NPHASE ; TEST FOR ABORT (OR ANALYSIS RESTART)
BNE S85 ; BR IF NOT AN ABORT (NPHASE=0)
CALL LED,(#QLED,HFSBS)
JPS10 JMP S10

S85: CMP NPHASE, #MASKPH
BNE S75 ; BR IF NOT THE MASK PHASE

C15: MOV #HDS.R0
MOV #HDS.R1
MOV #HSBN.R2

C20: TSTB (R1)+ ; BR IF DATASET IS EMPTY
BIS $60.R0

C30: MOV $R0,FILPEX+1 ; STORE IN FILENAME
PMVB:  BLK 10
BYTE 1,1
RNDS 1
BLK 2
.... 
RNDS ..CALMS_PAR/ 1CALMS_PAR
BYTE 2,2
BLK 10
PARSIZ 1024.
PLITE:  WORD 100, 100, 100, 100, 100, 100, 100, 100, 100
WORD 100, 100, 100, 100, 100, 100, 100, 100, 100
NPTAB:  BLK 10
HP TABLE
BPBLK NPH
HPTAB=172430
HCPY=172430
HCST=172430
HCIV=440
HCIV=440
INTPS=340
HCOPY: MOV #7777,A CLEAR ACTIVE BLOCK NUMBER
MOV #7777,A+IBN AND INACTIVE BLOCK NUMBER
MOV #SFLAG,A+2 READ LABEL INTO SLAB
MOV $LABSIZ,A+BUFFSIZ READ LABELSIZ BYTES
CLRB A+DBFLAG PREVENT READ-AHEAD
CLR A+CCUREC CLEAR CURRENT RECORD NUMBER
CALL GLABEL,(A,SPAR,1A) READ THE LABEL INFORMATION INTO CORE AND CALCULATE RECLEN.
JNE TRY, @RETRY SET UP THE MVG TO USE TWO 4096 BYTE BUFFERS IN CORE FOR THE DISK TO CORE TO HARD-COPY DMA TRANSFER.
MOV #7777,A CLEAR ACTIVE BLOCK NUMBER
MOV #1E0000,A+2 ONE BUFFER AT 160000
MOV #7777,A+IBN CLEAR INACTIVE BLOCK NUMBER
MOV #170000,A+IBA ONE BUFFER AT 170000
MOV #100000,A+BUFFSIZ 1BUFFSIZ 4096 BYTES
MOV #77,A+DBFLAG 18 BLOCKS PER BUFFER
INCB A+DBFLAG 1SET DOUBLE BUFFERING
CLR LINE : INITIALIZE LINE COUNTER
MOV #INTRY,0+HCIV CLEAR ENTRY ADDR TO HCPY INT VEC
MOV #INTPS,0+HCIV CLEAR SYG PSW TO HCPY INT VEC
INC DPL IN CASE OF ODD BYTES
MSR DPL AMD CNT FOR HCPY LINE IS DPL/2
ASL DPL DPL/2, 0+1DSCHT
MOV A+54,RMB : RECLEH
SUB DPL,RMB
SUB DPL,RMB
MOV $1E0000,0+HCAD
SUB RMB,0+HCAD
CLR 0+HCVC CLEAR HCPY UD CNT REGISTER
BIT #2900,0+HCST BIT-10 IS HC RUNNING
BNE ONST WAIT FOR PREVIOUS END-FRAME
BIT #4890,0+HCST BIT-11 IS HC READY
BNE REPL IHR IF HC READY
HCOFF: MOV 0+HCST,-(SP) DISPLAY HCST
MOV #591,-(SP) #A191 MSG, HC OFF
TOT. IDL ACTION MSG TO OPERATOR
SP ON ST CHECK HCPIY DEVICE AGAIN
HARDCOPY DEVICE INTERRUPT SERVICE ROUTINE
ENTRY: BIT #10000,#SWR IRR IF SY_12 UP
DIE NXML HI
LXPHD
BEQ NXML IIR IF NEW LINE
SUB DPL2,0+HCAD IRESTORE HCAD
REPL:  MO 0+HCAD
ADD RMB,0+HCAD
BIT #7777,0+HCAD
BNE NOGET
CALL GET,<A.LINE,INDEX>  GET LOCATION OF LINE
MOV R4,RHCD   AND LOAD INTO
ADD INDEX,RHCD   KNC BUS ADDR REG
HGET.  SUI RCON,0,RHCD   STORE NEGATIVE WORD COUNT
MOV R4L,RHCD   COMMAND LINE OUTPUT WITH PDATA=1, LXPHD=0
RTI   RETURN FROM INTERRUPT

EGDS.  CLR RHCST   STOP HCOPY DEVICE
CLR RPSW   LOWER PRIORITY
BIT R4,RPSW   BR IF SW 2 DOWN (NO RETURN CHECK)
TSTB PBUSY   TEST PARAM BUSY BYTE
BEQ SCP   BR IF NOT
MOV *SCP,PINT   STORE INTERRUPT ADDRESS
RTI

SCP.  CALL TYPE,<MSG3,ZER0>   CALL PARAM,<NP,PAR,MXNP,SPBUF>
CMPB SPBUF,VR   
BNE HCOX   BR IF OUTPUT WAS OK
JMP HCOPY   RPE-DO THE OUTPUT
HCOX.  CLRB RCDS   RELEASE THE HCOPY DATASET
DEC HCOPY
DECLED.  CALL LED,<HLED,HCOPY>   ENTRY FOR RESTART
MOV R0,-(SP)   
R1,-(SP)
MOV RMD5,R0
MOV RMD,R0   
MOV *RMD,REG2
H40.  TSIB (R1)+   
BNE H50   BR IF DATASET IS FULL
ADD H2,REG2
SOS P H40
CLRB HCFLO   SET HCOPY NO LONGER OPERATING
BIC H1,MULTS   TURN OFF HCOPY LITE
MOV (SP)+,R1
MOV (SP)+,R0
RTI   RETURN FROM INTERRUPT
H50.  MOV RREG2,A+SON   SET UP START BLOCK NUMBER
DEC R1
MOV R1,CHDS
MOV (SP)+,R1
MOV (SP)+,R0
JMP HCOPY   START NEW DATASET

PAGE
LINTPH  WORD LINTPH
NP  WORD -
PAR  BLKW 20
MXNP  WORD 20
LHUU  WORD HDU
ZERO  WORD 0
ONE  WORD 1
FOUR  WORD 4
FIVE  WORD 5
TWELVE  WORD 12
A  WORD -
BLEA  WORD MBSIZ
SLAD  BLK0 LABSIZ
SARL=SLAD+2
VL=SP+2
BPL=SP+2
DONT=EPJ
SPK=SPK+1
LXPHD  WORD -
RMB  WORD -
BPL2  WORD -
LINE  WORD -
NFDS  WORD -   NO OF FULL SCAN DATASETS
CMDS  WORD -   LOC OF CURRENT MASK DATASET FLAG
CHDS  WORD -   LOC OF CURRENT HCOPY DATASET FLAG
HDSN  BLKW MDS  HARDCOPY DATASET START BLOCK NUMBERS
CSGN  WORD -
REG2  WORD -
XVAL  WORD 2080
YVAL  WORD 2080.
57
58

HCOPY:  .WORD  -  # IN HCOPY QUEUE
SFLAG:  .BYTE  0  #FLAG FOR SEARCH OPERATING
SDS:   .BLKB  MSDS  #SCAN DATA SET FLAGS
HDS:   .BLKB  NHDS  #HCOPY DATA SET FLAGS
HCFLG: .BYTE  0  #FLAG FOR HCOPY OPERATING
RFLAG: .BYTE  0  #FLAG FOR RESTART REQUEST
PFLAG: .BYTE  0  #FLAG TO CALL THE PATIENT REPORTER
EFLAG: .BYTE  0  #FLAG TO CALL THE TEXT EDITOR
PFLAG: .BYTE  0  #FLAG TO CALL FORTRAN
SLZE:  .ASCIZ /
MSG1: .ASCIZ / OK TO START SEARCH/\s/\s/\nMSG2: .ASCIZ / IS THIS A RESTART? /
PEMSG: .ASCIZ / ILLEGAL PHASE NUMBER? /
M3G3: .ASCIZ / RERUN HCOPY? /
M4:   .ASCIZ / RESTART SCAN? /
M5:   .ASCIZ / RESTART ANALYSIS? /
M6:   .ASCIZ / RESTART HCOPY? /
M7:   .ASCIZ / WHICH SEARCH? /
M8:   .ASCIZ / HOW MANY SEARCHES?
SPBUF: .BLKB  03.  #HCOPY PARBUF
FILPEK: .ASCIZ /*
LNAME: .ASCIZ /XXX/
.EVEN
.BLKB  SOURCE+13980-  #MAKE PROGRAM LENGTH CONSTANT
.END  CALMS

.TITLE SEARCH - SEARCH PARTITION OF CALMS
.MCALL  CALL, PAUSE
.LF=12
.CR=15
.FTIME=15.  #MAX TIME BETWEEN FOCUS DURING SEARCH
.PSU=177776
.SVR=177579
.MACRO GETKS MSG,?LOC
.TSTG PBUSY
.BEG LOC
.MOV @LOC,PINT+2
.RI
.LOC CLEAR @PSU
.CALL TYPE,(MSG,ZERO)
.ENDM

.MACRO ENI  CODE  #ENABLE INTERRUPT
.BIS @CODE,@IEAPD
.ENDM

.MACRO DSI  CODE  #DISABLE INTERRUPT
.BIC @CODE,@IEAPD
.ENDM

.MACRO LON CODE  #TURN LIGHT ON
.BIS @CODE,@LITES
.ENDM

.MACRO LOFF CODE  #TURN LIGHT OFF
.BIC @CODE,@LITES
.ENDM

.MACRO MINT KEY  #HANDLE INTERRUPT FOR KEYS 3-8 (AC, RJ, NW, LA)
.DSI 109
.LOFF 169978
.MOV RA,-(SP)
.CALL LED, (LKLED, KEY)
.ENDM

.CURRENT ORIGIN IS 48330

JMP CNC  #ENTRY POINT FOR GNST

SEARCH, MOV #SRSTA,R9  #INITIAL CALL FROM SUPERVISOR
.MOV #SRST,(R0)+  #SET UP SEARCH START INTERRUPT VECTO
.MOV #298,(R0)+  #PRIORITY 4
.MOV #15,R1
IS20. MOV #UINT,(R0)+ ; UNEXPECTED INTERRUPT RTE TO OTHERS
MOV #200,(R0)+ ; PRIORITY 4
SBR R1, IS20
MOV #FOC, #FOCA
MOV #KYRG, #KYRG
MOV #200, #KYRG+2
CALL MFST, <MACH, FP00>
EHI 110 ; ENABLE SRCH START, FOCUS AND KB3 REQUEST
DSI 357 ; DISABLE OTHERS
RTR R5 ; RETURN TO SUPERVISOR
UINT : PAUSE IEAP3 ; UNEXPECTED INTERRUPT
RTR
SRST, DSI 10
CALL LED, (KLED, SSKEY)
GETKB M1 ; GET KEYBOARD AND TYPE M1
CALL PARAM, (NP, PAR, MAXNP, SPBUF)
CMP NP, #2
BNE SR20 ; BR IF NOT 2 WORDS
CMP PAR, #9
BNE SR20 ; BR IF NOT ABORT
CALL TYPE, M7 ; SEARCH ABORTED, TYPE M7
EHI 10
SR20: INCB SFLAG ; SET SEARCH OPERATING
LON #2599 ; RETURN OFF EDIT LIGHTS
LON 3000 ; RETURN SEARCH LIGHT ON
TST NP
BEQ SR30 ; BR IF SAME PATIENT AND SLIDE
CALL MVC, (SPBUF, PAR, MAXNP, SPBUF)
CALL TYPE, (M2, ZERO)
MOV #", PAR ; INITIALIZE SEX TO BLANK
CALL PARAM, (NP, PAR, MAXNP, SPBUF)
MOV PAR, PASEX ; STORE PATIENT SEX
CALL TYPE, (M3, ZERO)
CALL PARAM, (NP, PAR, MAXNP, SPBUF)
CALL MVC, (SPBUF, SLNUM, LSLNUM) ; MOVE SLIDE NUMBER
CALL TYPE, (M0, ZERO)
CALL PARAM, (NP, PAR, MAXNP, SPBUF)
MOV PAR, SOURCE ; STORE SOURCE CODE
SR30: MOV #SPB, CSPQ ; INITIALIZE SEARCH QUEUE POINTER
CLR MSP
MOV R0, (SP) ; SAVE R0 BEFORE CALLING ZIA
CALL ZIA, (SPR, LSPR) ; ZERO THE SPREAD RALTIONG
MOV (SP)+, R0
CALL LED, (SWLED, MSP)
CALL MFST, <MACH, FP05>
MOV XLCTR, XCTR ; INITIALIZE SEARCH PATTERN
MOV IXDEL, XDEL
MOV #SRH, #SDFPA ; SEARCH DATA READY, INTERRUPT
MOV #SRHA, #SDFPAA
MOV #SRBS, #SDFPSA
MOV #EDST, #EDSTA
EHI 74 ; ENABLE INTERRUPTS
RTI
FOC: CALL LED, (KLED, FOKEY) ; FOCUS INTERRUPT FROM FOCUS KEY
FOCUB: DSI 400 ; FOCUS SUBROUTINE
LON 48000 \ IT MAY BE ENTERED BY ANOTHER INTERRUPT
MOV #FIRE, FCNT ; RESET FOCUS COUNT
MOV #IEAP, IEAPD \ SAVE OLD IEAP AND SDFFA
MOV #SDFPP, SDFFPA
CALL MFST, <MACH, FP05> \ FIND FSTEP & FLAST FOR THIS MAGNIFICATION
MOV #F410, #SDFPA \ SET UP INTERRUPT VECTOR
EHI 4
RTI
F410: CALL CFOC, OLDF ; CALCULATE FOCUS
CALL IFREL, FSTEP
F420: MOV #F439, #SDFPA
F430: CALL CFOC, F
CMP F, #LDF ; SEE IF WE ARE GOING IN THE RIGHT DIR
BCT F450 ; BR IF SO
MOV FSTEP, N ; CHANGE DIRECTION
ASL N
NEG N ; MOVE TWICE AS FAR IN REVERSE
CALL INREL, H
MOV 0F440, 0DSFPA

F440
CALL CF0C, F
CMP F, 0LDF
BCT F440
CMP FSTEP, FLAST
BLE F480
MOV FSTEP, N
ASR FSTEP
ADD FSTEP, N
= FSTEP+FSTEP/2
CALL IFRSL, H

F443
NEG FSTEP
MOV F, OLFDF
CALL IFRSL, FSTEP
= TAKE ANOTHER STEP
MOV 0F460, 0DSFPA

F450
CALL CF0C, F
CMP F, 0LDF
BCT F450
MOV FSTEP, N
BPL F470
NEG N

F470
NEG FSTEP
CMP N, FLAST
= TEST FOR LAST STEP
BLE F490
BR F450

F480
CALL IFRSL, FSTEP
= TAKE LAST STEP BACK
MOV 0F490, 0DSFPA

F490
DSI 4
CALL RMC1
MOV 0DSFPA, 0DSFPA
= RESTORE SDFPA
BIT 64, OIEAPD
= SEE IF INTERRUPT WAS ENABLED
BEQ F590
BIT $1000, 01LITES
= BR IF NOT
BEQ F490
= RE-ENABLE IT IF NOT IN SEARCH
TST 0, 0SPRD
BPL F490
= OR SPREAD NOT PRESENT
BIT $4000, 0ISYR
BEQ F490
= OR SW 11 DOWN
LON 28900
= TURN ON SEARCH HALTED LIGHT.
BR F330
= DON'T RE-ENABLE THE INTERRUPT
F499
ENI 4
= RE-ENABLE INTERRUPT
5F99
ENI 4
LOFF 48300
= TURN OFF FOCUSING LITE

KY9
DSI 1890
= DISABLE ANY FURTHER KBQ REQ
CALL LED, (LKLED, KRXKEY)
GETKB M4
CALL PARAM, (NP, PAR, MAXNP, SPBUF)
TST NP
BEQ BKYEX
= BR IF NO INPUT
CMPB PAR, 97A
BNE KY10
= BR IF NOT ABORT REQUEST
BR BKYEX
KY10
CMPB PAR, 97C
BNE KY20
= BR IF NOT CALIBRATE MICROSCOPE
CALL IXREL, BIG
CALL IYREL, BIG
= MOVE TO LIMITS
CALL IFRSL, BIG
CALL XIWAIT
CALL YWAIT
= WAIT FOR END OF MOVES
CALL IFCP3
= INITIALIZE F.X, AND Y
CALL IYCP3
CALL IXYABS, (XVAL, YVAL)
= MOVE STAGE TO 1000, 1000 OR LAST POSITION
63
CALL IFREL.PWCH
CALL XWAIT
CALL YWAIT
CALL FWAIT
CALL RMC1
BKYEX: BR KEX
KY20: CMPB PBR,0'RE
BNE K38 # BR IF NOT RESTART REQUEST
INCB PFLAG # SET RESTART FLAG IN CALLS
KY30: CMPB PBR,0'k
BNE K33 # BR IF NOT EDIT REQUEST
INCB EFLAG # SET EDIT FLAG
BR KEX
KY35: CMPB PBR,0'F
BNE K40 # BR IF NOT FORTRAN REQUEST
INCB FRLAG # SET FORTRAN FLAG
BR KEX
KY40: CMPB PBR,0'2
BNE KY50 # BR IF NOT ZERO SCAN QUEUE REQUEST
CLR NSCAN
CALL LED(5QLED.NSCAN)
MOV #SRST.#1SRSTA # ALLOW SEARCH START
BR KEX
KY50: CMPB PBR,0'q
BNE KY60 # BR IF NOT QUEUE ADJUST
MOV @NCPR.XOFF
SUB @YPL.XOFF # STORE X OFFSET
MOV @VYCP.YOFF
SUB @YVAL.YOFF # STORE Y OFFSET
BR KEX
KY60: CMPB PBR,0's
BNE KY70 # BR IF NOT SCAN REQUEST
KY65: INC NSCAN
CALL LED(5QLED.NSCAN)
MOV #SRSPRCH.#1SRSTA # SET UP SPIRAL SEARCH
IF X AND Y WERE GIVEN
BNE KYEX: BR IF NOT
CALL IXYABS.(PAR+4.PAR+6)
ENI 1090
JMP FOCUSUB
KY70: CMPB PBR,0'id
BNE KY80 # BR IF NOT DEBUG REQUEST
MOV PPAR.4.PHASE # STORE PHASE NUMBER
BR KY65 # GO INCREMENT NSCAN
KY80: CMPB PBR,0'b
BNE KY90 # BR IF NOT BANDED REQUEST
MOV .#12.SOURCE # SET SOURCE OF -8
BR KYEX
KY90: CMPB PBR,0'p
BNE K169 # BR IF NOT PATIENT REPORT REQUEST
INCB FFLAG # SET PREP FLAG
CALL UPARAH.(ONE.PAR.PHASE) # WRITE PREP PARAM
BR KEX
KY199: KEX: ENI 1090
RTI
SBR: TST #S1SPRD # SPREAD DATA READY (SEARCH)
BNE #S1SPRES #BR IF SPREAD PRESENT
MOV #1.FLRLAG # RESET FOCUS FLAG
DEC FCHT # DECREMENT FOCUS COUNT
BPL XHP # BR IF NO NEED TO FOCUS
BIT #6#D#R1#S#SPRD # BR IF NOTHING TO FOCUS ON
JNP FOCUSUB # GO FOCUS
JPF FOCUSUB # GO TO FOCUS SUBROUTINE
NOFOC: MOV R8,-(SP) #R8=CURRENT SPREAD QUEUE POINTER
MOV #1XCP.RR # STORAGE X
PPL XOK # BR IF X IS POSITIVE
XYNG: GETKB M6 # REQUEST CALIBRATION
MOV (SP)+.R8
4,122,518

DIABLE INTERRUPT

STORE Y

IF Y IS NEGATIVE

SEE IF SEARCH IS AT HI-MAG

IF AT HI-MAG, FLAG AS ACCEPTED IN ED-LO

INC HSP; INCREMENT # SPREADS FOUND

CALL LED. (SNLED, HSP)

CALL HSP. HAXNSP

BR IF ENOUGH SPREADS FOUND

BR IF ANOTHER X MOVE IS OK

END OF ROW, REVERSE DIRECTION

REPEAT MOVING DOWN

RELOAD XCTR

MOVE IN X DIRECTION

DISABLE INTERRUPT

SEARCH OR FOCUS HALT

PREVENT END OF FOCUS FROM RESUMING SEARCH

TURN ON SEARCH HALTED LIGHT

CALL RNCI

TURN OFF SEARCH HALTED LIGHT

EDIT START

DISABLE FOCUS INTERRUPT

INITIALIZE FOCUS FLAG

BR TO EDIT-LO IF IN SEARCH

BR IF 9TH EDIT-LO

BR IF LO-MAQ (IN EDIT-HI)

Z-EDIT-LO, TEST MAG

IF LO-MAQ (IN EDIT-HI)

MARK END OF QUEUE

ZMARK END OF QUEUE

ACCEPT SPRD

REJECT SPRD

NEXT SPRD

LAST SPRD

PREVIOUS NUMBER

SAVE R0 ON THE STACK

TURN OFF FOCUS LITE IF CASE FOCUS

TURN OFF FOCUS LITE IF CASE FOCUS

DISABLE INTERRUPT BY ARNL
LOFF 2000
MOV (R0)+, X
BEQ JEDHI ; BRK IF END OF QUEUE
BPL EL38 ; BRK IF NOT ALREADY ACCEPTED
LON 2000 ; INDICATE ALREADY ACCEPTED

EL38: MOV R0, CSPAQ ; SAVE UPDATED Q POINTER
MOV (SP)+, R0 ; RESTORE R0
CALL IXABS, (X, Y) ; INITIATE MOTOR MOVE TO X, Y
MOV $0ESDR, $0SEFFPA
MOV $11, FFAGL ; ENABLE SPRD DATA READY INTERRUPT
RTI

JEDHI: JMP EDHI

ESDR: ENI 100 ; ALLOW RJ DURING FOCUS
NEG FFLAG ; EDIT SPREAD DATA READY
BPL ES20 ; BRK IF NO FOCUS THIS TIME
JMP FOCSUB

ES20: DSI 4 ; NOW DISABLE THE SDR INTERRUPT
.CALL RMI ; RELEASE MCI FROM COMPUTER CONTROL.
LON 100000 ; TURN ON OPERATOR ACTION LIGHT
RTI

ELAC: HINT ACKEY ; ACCEPT SPRD INTERRUPT
MOV CSPAQ, R0
TST $0(R0)
BPL EL29 ; BRK IF NOT PREVIOUSLY ACCEPTED
BEC NSPAL ; DON'T COUNT IT TWICE

ELAC: MOV $0XCRP, $0(R0) ; STORE UPDATED Y AND X
MOV $0XCRP, $0(R0)
NEG $0R0
INC NSPAL
CALL LSLED, $0XLED, NSPAL
CNP NSPAL, HLMAX
BGE EDHI ; BRK IF ENOUGH FOUND
BR EBXT ; GO TO NEXT SPREAD

ELRJ: HINT RJKEY ; REJECT SPREAD INTERRUPT
MOV CSPAQ, R0
TST $0(R0)
BPL EL29 ; BRK IF NOT PREVIOUSLY ACCEPTED
BEC NSPAL ; MARK IT REJECTED X+

ELRJ: MOV $0XCRP, $0(R0) ; STORE UPDATED Y AND X
NEG $0(R0)
BEC NSPAL
CALL LSLED, $0XLED, NSPAL
BR JEDHXT

ELAK: HINT NSKEY ; NEXT SPREAD
BR JEDHXT

ELLA: HINT LAKEY ; LAST SPREAD
SUB $0, CSPAQ ; MOVE BACK TO CURRENT SPREAD
DEC SPH
BEQ JEDHXT ; BRK IF FIRST SPREAD IN QUEUE
SUB $0, CSPAQ ; BACK UP ANOTHER SPREAD
DEC SPH
JEDHKT JMP $32HXT

EDHI: MOV (SP)+, R0 ; RESTORE R0
GETKB NS ; REQUEST HI MAGNIFICATION
EHHST: LOFF 1400
LON 200 ; TURN ON EDIT HI LIGHT
MOV $0EHAC, $0ACSPA
MOV $0EHAC, $0ACSPA
MOV $0EHAC, $0ACSPA
MOV $0EHAC, $0ACSPA
MOV $0EHAC, $0ACSPA
ENO 209
CLR NSCAN
.CALL LSLED, $0XLED, NSCAN

EHHXT: MOV CSPAQ, R0
69

4,122,518

70

LOFF 4000	TURN OFF FOCUS LITE IN CASE FOCUS

EHN 400	WAS INTERRUPTED BY ARNL

INC SPH

MOV (RH)+, X

BEQ JEHNXT	BR IF END OF QUEUE

BNI EHN20	BR IF -X (ACCEPTED IN EDIT LO)

TST (RH)+

BR EHN10	TRY THE NEXT SPREAD

EHN20: 2000

MOV (RH)+, Y

BPL EHN30	BR IF NOT ALREADY ACCEPTED

LON 2000	RETURN ON LITE

BR EHN40	ADJUSTMENT IF ALREADY ACCEPTED

CALL MFT, (MAGN, FPQ)

SUB XADJ, X

ADD YADJ, Y	ADJUST POSITIVE Y FOR CENTRALITY

AND YADJ, Y	SETUP XADJ AND YADJ

CALL XAYABS, (X, Y)

MOV RESDR, RESPPA

MOV #1, FFLAG	ENABLE SPRD DATA READY INTERRUPT

RTI

JEHNXT: JMP JEHNXT

EHAC: HINT ACKEY	RED HI ACCEPT

MOV CSPQ, RH

TST -2(RH)

BPL EHN20	BR IF NOT ALREADY ACCEPTED AT HI MAG

DEC NSCAN	DON'T COUNT IT TWICE

EHN20: MOV @#CPR, -(RH)

NEG BR8

MOV @#CPR, -(RH)

NEG BR8

INC NSCAN

CALL LED, (SLED, NSCAN)

SUB @SP, RH	FIND LOC IN SPR

ASH @-2, RH

INC SP(RO)

MOV (SP)+, 30

LON 100000

EHI 100

RETURN FOR ANOTHER AC OR HX

EHJ: HINT RKEY	RED HI REJECT

MOV CSPQ, RH

TST -(RH)

BPL EHJNT

NEG @33

MAKE IT POSITIVE IF PREVIOUSLY ACCEPTED

DEC NSCAN

TST -(RH)

GET LOC IN SPR FOR THE SPREAD

SUB @SP, RH

ASH @-2, RH

CLR SP(RH)

CALL LED, (SLED, NSCAN)

BR JEHNXT

EHK: HINT HKKEY	RED HI LAST

CMP NSCAN, MAXSC

BGE EDNEND	BR IF ENOUGH FOUND

BR JEHNXT

EHL: HINT LKEY	RED HI LAST

MOV @4, CSPQ

DEC SPH

BEQ JEHNXT	BR IF FIRST SPRD IN Q

EHN20: MOV @4, CSPQ	MOVE BACK ANOTHER SPREAD

DEC SPH

BEQ JEHNXT

TST @CSPQ

BPL EHL20	BR IF NOT -X

JEHNXT: JMP EHJNT	LAST -X

EDEN: CALL LED, (LKEY, EEKEY)
TITLE: BLINE - DISPLAY A LINE ON THE INTERACTIVE DISPLAY

CALMS VERSION

RFDS=172410 WORD COUNT REGISTER
RFAD=RFDS+2 BUS ADDRESS REGISTER
RFCT=RFDS+4 STATUS AND COMMAND REGISTER
YREC=164196 LINE REGISTER
ZREC=164196 SAMPLE REGISTER
CSST=164180 GPAY SCALE STATUS/CMD REGISTER
GOSP=164180 GPAY SCALE PIX COUNT REGISTER
GCSC=164112 GPAY SCALE X CURSOR POSITION
GCSP=164114 GPAY SCALE Y CURSOR POSITION
WCMC=6 WRITE COMMAND
ERCMD=2 ERASE/Write COMMAND
RBTR=10 REPLICATE BIT
WTR=177570 WRITE BIT
LF=12

CALL BIN2O, CALL, PAUSE

CALL WAIT

WAIT TST XGSSST, TEST READY BIT
CALL DCLEAR

DCLEAR: JSR R5.DWAIT
INC @GSSST ;STORE CLEAR COMMAND
RTS R5

CALL DLINE(LOC,Y,X,NS,REPL,ERASE) -1=ERASE ONLY

DLINE: JSR R4.SAYER
JSR R5.DWAIT
MOV #4(R5),D1YREG ;STARTING LINE ON DISPLAY
MOV #URCMD.R1
BIS #499,#0DRST ;ASSUME WRITE, SET CYCLE BIT
TST 014(R5)
BEO NOER ;BR IF NO ERASE
MOV @UCMD.R1 ;ASSUME ERASE/WRITE
TST 014(R5)
PBL NOER ;BR IF ERASE/WRITE
MOV #ERCMD.R1 ;ERASE ONLY
BIC #499,#0DRST ;CLEAR CYCLE BIT
NOER: MOV 016(R5),R3 ;NS
MOV 012(R5),R2 ;REPL
BEO NOREP ;BR IF NO REPLICATION
BIS #0RST.R1 ;INSERT REPLICATE BIT
ASL R3 ;R2

NOREP: NEG R3
MOV #0,#SPC ;STORE PIXEL COUNT -1 OR -2*N
REP: MOV 2(R5),#0DBA ;LOC TO BUS ADDRESS REGISTER
MOV @G(RS),#0XREG ;STARTING SAMPLE ON DISPLAY
MOV @G(RS),#0XREG ;SAMPLE COUNT
INC R3

ASR R3
NEG R3
MOV R3,#DRWC ;STORE NEGATIVE WORD COUNT
MOV R1,#GSSST ;STORE WRITE OR ERASE/WRITE COMMAND
INC #0DRST ;ISSUE GO
TST R2
BLE EXIT ;BR IF DONE
BIT #10,#SWR ;REPLICATE UNLESS SW 3 IS UP
BNE EXIT ;EXIT IF SW 3 UP
CLR R2
JSR R5.DWAIT ;WAIT
INC @YREG ;STEP TO NEXT LINE
BR REP ;GO DO THE REPLICATION
EXIT: JMP RESTR

CALL SC TO SET UP THE CURSOR

SC: JSR P4.SAYER
MOV #3,R3
SC10: MOV #32,Y
MOV #32,Y ;WRITE A MARK AT 32,32
CALL DLINE,(ZERO,Y,X,ONE,ONE,ZERO)
MOV #9,R4
SC20: ADD #2,Y
CALL DLINE,(ZERO,Y,X,ONE,ONE,ZERO)
SBD #4,SC20
SBD R3.SC10 ;REPEAT TO MAKE IT WHITE ENOUGH
MOV #3,R3
SC30: MOV #973,Y
MOV #992,X ;WRITE A MARK AT 992,992
MOV #9,R4
SC40: CALL DLINE,(ZERO,Y,X,ONE,ONE,ZERO)
ADD #2,Y
SBD #4,SC40
MOV #973,X
CALL DLINE,(ZERO,Y,X,ONE,ONE,ZERO)
SBD R3,SC10 ;CALL TYPE,(M3,ZERO) ; TELL OPERATOR TO MOVE CURSOR
SC50: CALL PARAM,(MP,PAR,FIVE)
JSR R5,ULDR ;CHECK FOR REQUESTED CURSOR MOVE
BR SC50 ;CURSOR WAS MOVED
MOV #65YCP,Y ;STOP Y ZERO ADJUSTMENT
4,122,518

MOV 0@GSXCP, XZ    ;STORE X ZERO ADJUSTMENT
CALL TYPE, <R4, ZERO>    ;MOVE TO LOWER RIGHT
SC60 CALL PPARAM, <HP, PAR, FIVE>
JSR R5, ULDR
BR SC60    ;CURSOR WAS MOVED
MOV @GSYCP, YD
SUB YZ, YD    ;STORE Y BELTA
MOD YZ, XD
SUB XD, XD
BIC @176000, YZ
BIC @176000, XD
CALL UPARAM, <FOUR, YZ, INTPH>    ;SAVE THE ADJUSTMENTS ON DISK
JMP RESTR

RCA: CALL PPARAM, <HP, YZ, FOUR, INTPH>    ;READ ADJUSTMENTS FROM DISK
RTS R5

CALL CURSOR(YCP, XCP, LINE, SAMPL)

CURSOR: JSR R4, SAVER
MOV @GSXCP, R0
BIC @176000, R0
SUB YZ, R0    ;ADJUST THE READING
MUL @960, R0
DIV YD, R0    ;ROUND UP
ADD #32, R0    ;AY=(Y-ZY)*960/YD+32
ASL R1
CMP R1, YD
BLT C20
INC R0

C20: MOV @GSXCP, R2
BIC @176000, R2
SUB XD, R2    ;ADJUST THE X READING
MUL @960, R2
DIV XD, R2
ADD #32, R2
ASL R3
CMP R3, XD
BLT C40
INC R2

C40: MOV R0, #02(R5)    ;STORE YCP
MOV R0, #04(R5)    ;STORE XCP
BIT #4, R5
BEQ CEX    ;JBR IF NOT 4 PARAMETERS
ASR R0
INC R0
MOV R0, #06(R5)    ;LINE=YCP/2+1
ASR R2
INC R2
MOV R2, #01(R5)    ;SAMP=XCP/2+1
CEX: JMP RESTR

MCU: SUB R2, #000GSYCP    ;CALL MCU TO MOVE CURSOR UP
RTS R5

MCB: ADD R2, #000GSYCP    ;CALL MCB TO MOVE CURSOR DOWN
RTS R5

MCL: SUB R2, #000GSXCP    ;CALL MCL TO MOVE CURSOR LEFT
RTS R5

MCR: ADD R2, #000GSXCP    ;CALL MCR TO MOVE CURSOR RIGHT
RTS R5

ULLR: MOV &2(R5), PAR    ;CALL ULLR(PAR) ... EXTERNAL CALL
BR UL20

ULLR: TST HP    ;CHECK FOR U.L.R. OR A TYPED LETTER
BNE UL20    ;SET FOR NO TYPEIN RETURN
TST <R5>    ;JBR IF A LETTER WAS TYPED
RTS R5

UL20: CMPB PAR, O'U
BNE UL30    ;JBR IF NOT A W
DEC @GSYCP
MOV CURSOR UP ONE STEP

UL30: CMPB PAR, W'L
BNE UL50

UL40: CMPB PAR, W'L
BNE UL50
; MOVE CURSOR LEFT ONE STEP
DEC  @GSSXCP
BNE  U60

; MOVE CURSOR RIGHT ONE STEP
INC  @GSSXCP
RTS  R5

ZERO  .WORD  0,0,0,0
ONE  .WORD  1

FOUR  .WORD  4
FIVE  .WORD  5
TEN  .WORD  10

INTPHI  .WORD  7

PAR  .BLKW  5

X=PAR+2
NP=PAR+4

; MOVE CURSOR TO UPPER LEFT MARK
.MLIST BEX

; MOVE CURSOR TO LOWER RIGHT MARK

Y:  .WORD  32
X:  .WORD  32

XD  .WORD  960.
YD  .WORD  960.
BLKB  DWAIT+1500.

END

; TITLE MCI SUB - MCI SUBROUTINES
CALMS VERSION

MCALL  CALL, PAUSE

XCPP==164048
YCPP==164042
FCPP==164044
SMSC==164046
IHPPD==164059
SPDP==164074
FCPP==164076
LITES==164134
LEDSS==164136
SRSTAA==300
SSPA==310
ESDEAA==314
ACSPAA==320
RSSPA==324
MXSPA==334
LSPAA==334
EDPHAA==336
FOCAA==344
KYPGA==350
GDPFA==374

; SEARCH START INTERRUPT ADDRESS

; SEARCH RESUME

; EDIT START

; ACCEPT SPREAD

; REJECT SPREAD

; NEXT SPREAD

; LAST SPREAD

; EDIT END 9

; FOCUS 10

; ACCEPT SPREAD

; KBD REQUEST II

; SPREAD/FOCUS DATA READY INTERRUPT ADDR

; CURRENT ORIGIN IS 50590

; CAPTURE MCI

; RELEASE MCI

; DWAIT+1500.

; XWAIT

; YWAIT

; FWAIT

; CALL XWAIT

; CALL YWAIT

; CALL CHCI

; CALL IXYABS(X,Y) TO INITIATE ABS MOVE

; PPL  IXTO
81

82

4,122,518

IYX20.
MOV 04(R5), Y
BPL IYX30
NEG X

MAKE X POSITIVE

IYX30.
SUB X, #XCP
INITIATE MOVE TO X, Y
SUB Y, #YCP
RTS R5

IYREL.
.call XWAIT
.call XREL
MOV #2(R5), #XCP
RTS R5

IYREL.
.call YWAIT
.call YREL
MOV #2(R5), #YCP
RTS R5

IFREL.
.call FWAIT
.call FREL
MOV #2(R5), #FCP
RTS R5

CFOC.
clr #FOCF, TEMP
.call CFOC(F) TO CALCULATE FOCUS
CLR #2(R5)
MOVB TEMP+1, #02(R5)
CLRB TEMP+1
ADD TEMP, #2(R5)
FOCUS = F1 + F2
RTS R5

IXCP.
BIS #400, #SHSC
initialize XCP
RTS R5

IXCP.
BIS #200, #SNS
RTS R5

IFCP.
R5, #SHSC
RTS R5

LED.
JSR R4, SAVING
.call LFLED (CODE, VAL)
MOV #2(R5), R8
CODE TO RO
ASH #43, R8
SHIFT TO HIGH ORDER 3 BITS
MOV #4(R5), R1
VALUE TO R1
BIC #17000, R1
ADD R1, R0
MOV R8, #LEDSB
output display
JMP RESTR

MFST.
JSR R4, SAVING
.call MFST(MAG, FPPOS)
MOV #1IEAP, R0
FLTR/MAG POS REG TO RO
MOV R8, R1
ALSO R1
ASH #18, R8
MOV #177776, R8
MASK OUT MAG
MOVB LEHS(R8), MAG
.STORE MAG
CLR FSTEP
CLEAR TOP BYTE
MOVB FSTEP3(CR0), FSTEP
.STORE FOCUS STEP SIZE
MOVB FINTAB(CR0), FLAST
.STORE LAST STEP SIZE
MOVB XATAB(CR0), R2
.AJUSTMENT FOR NON-CENTRALITY
MOV R2, #AB1
MOVB YATAB(R0), R2
.AJUSTMENTS ARE FOR 63 AND 100
MOV R2, #ADJ
.RELATIVE TO 40
MOV MAG, #2(R5)
ASH #13, R1
SHIFT FILTER CODE
BIC #177776, R1
MOV R1, #4(R5)
MOV R1, FPPOS
.call LED, #MLLED, MAG, DISPLAY MAGNIFICATION
JMP RESTR

SORTG.
JSR R4, SAVING
.subroutine to sort the queue
MOV #SPP, R3
R3 AND R4 POINT TO LOW QUEUE POSITIONS
MOV #SPQ, R4
THE CURRENT HI RATING IS MOVED HERE
MOVB HSR, R3
HSP, HSR
initialize 8 spreads remaining
MOV R3, R1
USE R1 TO EXAMINE REMAINING SPRDS
NOV HSR, R2
BEQ SEX
BR IF NO SPREADS REMAIN
SQR3.
CMPB (R1)+, R0
.EXAMINE A RATING
BLE 
BR IF NOT GREATER THAN MAX
MOV R1, #R0
STORE NEW MAX
MOV R1, REG1
SAVE R1
MOV @R6.02(R5) ;STORE HP FOR THIS PHASE
BEQ RP49 ;SKIP THE READ IF HP=0
CMP @R6.06(R5) ;IF HP GT HMAX
BNE RPERR
MOV 4(R5), R20+14
MOV @R6.PMYB+BUF512 ;SET UP TO READ HP WORDS
ASL @R6.PMYB+BUF512
RP20. ;CALL READ(PMYB, PREC, INDEX, -) ;READ THE PARAMETER BS
RP40. ;NO REST
RPERR. MOV @R5.PNUM,-(SP) ;TOO MANY PARAMETERS
MOV @R5.01779,-(SP) ;PRINT F370

PREC. WORD -
INDEX. WORD -
END

TITLE SCAN = SDS DATA CAMERA DRIVER

CALLS VERSION CALL SCAN(DUNIT, FILPEX)
THIS VERSION USES MYIO IN DOUBBLE BUFFERING 24
VERTICAL LINES AT A TIME FROM CAMERA TO A DISK DATA SET
IN NON-INTERRUPT MODE.
THE PICTURE IS DISPLAYED ON THE GRAY SCALE.
SECTOR HISTOGRAMS ARE GENERATED AS THE PIXELS ARE SCANNED.
THRESHOLDS ARE FOUND AFTER EACH GROUP OF SECTORS AND TYPED
OUT WHILE THE NEXT ROW OF SECTORS IS SCANNED IF SW1 IS UP.
THE SCAN AREA IS VARIABLE WITH A MAXIMUM OF 512 LINES AND
1400 SAMPLES PER LINE.

GLOBAL OPEN.GT, PUT, CLOSE, PARAM, OPINT, LABEL, PARAM
GLOBAL FILE, SCAN
MCALL PARAM, EXIT, CVTBT, CALL, PAUSE
PARAM
HLIST, BEX
LF=-12
CR=-15
COMP=164920
STAT=164920
XREG=164922
YREG=164924
ZREG=164926
XJREG=164930
YJREG=164932
XJREG=164932

SCAN: MOV (R5), -<SP> ;SAVE R5
MOV 2<SP>, S109+10 ;CALL SCAN(DUNIT, FILPEX) STORE DUNIT
MOV 4<SP>, S109+12 ;STORE CALMS DATA SET NAME
MOV @R6.UIC,-(SP) IAND UIC OF [2, 2]
S101. ;CALL AFIL<SP>, -<SP>, -<SP>, UIC, UIC) ASSIGN DEFAULT FILE
S102. ;CALL OPEN<SP>, BUF, SIZE, ONE ONE, NAME<SP>, OPEN DISK DATA SET
S103. ;CALL TYPE<SP>, CM, ZERO<SP>, TEST IF STILL FOCUSING
TFICI. BIT #499. #410<SP> ;TEST IF STILL FOCUSING
BEQ TFICI IBR IF SO
.TFICI. ;RELEASE NCI TO OPERATOR
.CALL PARAM, <HPR, PHR, HMAX> ;READ LABEL
MOV SOURCE, L+12 ;MOVE INFO INTO LABEL
MOV PANUM, L+14
MOV SLNUM, L+16
MOV @R6.0C3F<SP>, L+18
MOV @R5<YCPK>, L+20
.CALL OUTCON<SP>, SOURCE, L+01, TWO
.CALL NVL<SP>, PANUM, L+21, LPanum<SP>, MOVE PANUM INTO LABEL
MOV PASEX, L+193
.CALL NVL<SP>, SLNUM, L+112, LSLNUM<SP>
TST HPR
BEQ S100 ;BR IF NO PARAMETERS (USE OLD LABEL)
CMP HPR, #2
BNE S104 ;BR IF NOT TWO PARAMETER WORDS
CMP =<PO> PAR
BNE NOTPO ;BR IF NOT PRINT Q REQUEST
.CALL PRINTQ<SP>, (L+72, SPQ, SPQ)
NOTPQ: JMP S102
CMP *A2, PAR
BNE NOTAB
.CALL APHASE.ZERO
JMP S30B

NOTAB: CMP *Z2, PAR
BNE S103
CLR NSCAH
.CALL APHASE.ZERO
BNE S194
.CALL NOTCIB.CHP
JMP S103

$103: CMP *EX, PAR
BNE S184
.SCAN IF EK WAS NOT TYPED
.JMP S68

$194: CMP *DS, PAR
BNE S185
.BR IF NOT DUMMY SCAN
INC DFLAG
.SET DFLAG

$105: MOV @PAR3UF.R1
MOV 4L+217., R2
.ADD ANOTHER LABEL WITH MISC ID
MOV #95, R3
.MAX OF 69 CHARACTERS IN LABEL
MOV (R1)+, R0
CRL
BEQ ELAB
.BR IF END OF LABEL (CR)
MOV R0, (R2)+
.STORE LABEL CHARACTER

$106: MOV *@BB, (R2)+
.ADD REST OF LABEL WITH BLANKS
SUB R3, MLAB
BLR S106

ELAB: JSR PC.PPAR
.PROCESS PARAMETERS
.CALL OUTCON.(<XCPR.L+127.,FOUR>)
.CALL OUTCON.(<YCPR.L+135.,FOUR>)
.CALL MVL.(L+124.,M2+12.,FOUR)
.CALL TYPE.M2
CLVTDT #0, 0L+145.
.STORE DATE IN LABEL
CLVTDT #0, 0L+135.
.STORE TIME IN LABEL
TST DFLAG
SUB S108
.NO OUTPUT LABEL IF DUMMY SCAN
.CALL PLABEL.(<BUF.SPAR.L>)
.OUTPUT LABEL
CLEAR THE CRAY SCALE

$110: MOV SL.X
.INITIAL XCOUNT
CLR #COND
.DISABLE PIXEL INTERRUPT
CLR CY
.CLEAR GS LINE COUNTER
MOV #ITAB, TX
MOV #ITAB, BX

LSCEC: CLR SHUM
.SECTOR NUMBER=0
.CALL ZIA.(<HIST,MBIN>)
.IZERO HISTOGRAM COUNTS
EFLIN: MOV SG,Y
.INITIAL Y
MOV #0, R3
.SET LINE COUNT
INC LINE
.INCREMENT MV10 LINE COUNT
TST DFLAG
SUB S115
.BR IF DUMMY SCAN
.CALL PUT.(<BUF.LINE,INDEX>)
.PREPARE STORAGE OF NEXT TV LINE
$115: MOV BUF.R0
.INIT BUFFER POINTER TO BUF+HS-1

0FLIN: ADD INDEX.R3
.SET POINTER FOR NEXT LINE
MOV #HIS.R4
.JR HAS LOC OF HIST
MOV ISP5.R3
.INITIAL SAMPLES PER SECTOR
MOV Y,#01REG
.SET PIXEL COUNT
MOV #COND.R2
.(COND-STATUS REGISTER) TO R2
MOV #0R2
.GET FIRST PIXEL OF THIS FIELD
TSTB #R2
BPL -2
.WAIT UNTIL READY
MOV #339, 00PSW
.SET PRIORITY 6 FOR THE REST OF THE FIELD
BR NOVBD

SECTOR: MOV SPS.R3
.R3=SAMPLES PER SECTOR
PIXEL: MOV #3, OR2
.PIXEL COMMAND
TSTB OR2
SP 2
.WAIT UNTIL READY
MOV #Z2REG.R1
.MOVE PIXEL INTO BUFFER
BNE NOTZE
.BR IF NOT ZERO
BRK IF PARAMETER IN KEYWORD
BRK IF MORE KEYWORDS
BRK IF 4 WORDS LEFT

CALL TYPE.N4
BR PPAR

CMP R0,8MMLXL
BCE P25
MOV R8,SL
BLT P25
BR IF SL TOO SMALL

MOV (R1)+,R0
BLE P25
BR IF EL TOO SMALL
CMP R8,8MMLXL
BGT P25
BR IF EL TOO LARGE
MOV R0,EL
SUB SL,R0

BLE P25
BR IF EL TOO SMALL
MOV R8,SL
INL=EL-SL
MOV (R1)+,R0
ISS
BLT P25
BR IF SS TOO SMALL
CMP R8,8MMLXL
BCE P25
BR IF SS TOO LARGE

BIC $1, R0
MAKE $S EVEN
MOV R6,SS
MOV (R1)+,R0
YES
BIC $1, R0
YES MUST BE EVEN
BLE P25
BR IF ES TOO SMALL
CMP R8,8MMLXL
BGT P25
BR IF ES TOO LARGE
SUB SS,R0
INS=ES-SS
BLE P25
BR IF HS TOO SMALL
MOV R8,HS
MOV $512,RJ
CLR R2
DIV R8,R2
MOV R2,HLR
INLR=$512/HS
ADD #BUF,R9
DEC R0
MOV R8,BUFF
BUFF=BUF+HS-1
SUB $4,HPAR
HPAR=HPAR-4

BCE P25
BR P10
STEP OVER SECOND WORD
ADD #MK+MK-2,R3
LOC FOR THIS KEYWORD
MOV P23,PC
GO THERE

MOV (R1)+,JPER
P50:
TST (R1)+
SUB DPER,BPER
ADD #100,BPER
MOV P10,BPER
SUB #5,HPAR
HPAR=HPAR-3

BR P10
MOV (R1)+,BEG
MOV (R1)+,BKSIZ
ASP BKSIZ
DEL BKSIZ
MOV $62,BTSIZ
SUB BKSIZ,BTSIZ
BTSIZ=$62-BKSIZ
DR P63

MOV (R1)+,HR
INC QFLAG
SUB $2,HPAR
BR P10

P199:
MOV #JREG,HPAR+4
JOYSTICK
MOV #JREG,HPAR
STORE SL AND SS
BIC $17086,HPAR
CALL TYPE.($5,ZERO)
REQUEST MOVE TO LOWER RIGHT
CALL PARAM.($HPAR,PAR,HPMAY)
MOV #JREG,HPAR+6
STORE ES
MOV #JREG,HPAR+2
ELS
BIC $17086,HPAR+2
MOV #JPAR,1
MOV (R1)+,R0
MOV #4,HPAR
JMP  P30  RG0 PROCESS AREA PARAMETERS

P159:   TST  HR   IDONE WITH PARAMETERS
        BGT  P160  IBR IF HR WAS A PARAMETER
        MOV  HL,R0
        ADD  $32,R0
        ASH  8-6,R0
        BGT  P155  IBR IF HL GT 31

P155:   INC  R0  INR=1
        MOV  R0,HR  INR=(NL+32)/64
        TST  HC
        BGT  P170  IBR IF HC WAS A PARAMETER
        MOV  NS,R0
        ADD  $32,R0
        ASH  8-6,R0
        BGT  P163  IBR IF NS GT.31
        INC  R0  INC=1
        MOV  NS,R0
        INC  (NS+32)/64

P163:   MOV  R0,HC
        ADD  HR,R1
        DEC  R1
        CLR  R0
        DIV  HR,R0
        INC  R0
        BIC  @1,R0  MAKE LPS EVEN
        MOV  R0,LPS  ILPS=(NL+HR-1)/NR
        MOV  NS,R1
        CLR  R0
        DIV  NC,R0
        BIC  @1,R0  MAKE IT EVEN
        MOV  R0,SPS2   ISPS2=NS/NC  SPS FOR 2 FIELDS
        ASR  R0
        MOV  R0,SPS  ISPS FOR ONE FIELD
        MOV  NC,R1
        DEC  R1
        MUL  SPS,R1
        MOV  NS,R0
        ASR  R0
        SUB  R1,R0  ISPS=NS/2-(NC-1)*SPS
        MOV  NC,R1
        ASH  @7,R1
        ADD  #HIST,R1  HIST+MC=128
        MOV  R1,HTEST+2
        SUB  #128,R1
        MOV  R1,T05+2
        RTS  PC

THRESH

MOV  #$56+3,T50+10
MOV  #$56+6,T60+10

T05:  MOV  #HIST+1995,+$R4    HIST+(HC-1)*128
T10:  JSR  PC,T5SB    :FIND BACK AND DATA
        MOV  DPER,R1
        MUL  DATA,R1
        MOV  DPER,R3
        MUL  BACK,R3
        ADD  R3,R1
        ADD  $50,R1
        CLR  R0
        DIV  #100,R0  JTHR=(DPER*DATA+(100-DPER)*BACK+50)/100
        MOV  DATA,R1
        SUB  BACK,R1
        CMP  R1,B0  :CHECK FOR SMALL DIFFERENCE BETWEEN D AND B
        BGT  T20  IBR IF OK

T20:  MOV  BK612Z,R0  JTHR=BBEC/2 (UNDIVIDED NUCLEUS)
        TST  DATA
        BGT  T20
        MOV  #127,R0  JTHR=127 IF NO DATA IN SECTOR
        TST  BACK
        BGT  T40  IBR IF BACK OK
        CMP  BX,38TAB  :SEE IF FIRST SECTOR
        BEG  T40
MOV BX,R1; USE BACK FROM PREVIOUS SECTOR
MOV -(R1),BACK
T40: MOV R0,R0
MOV R0,0
TST DATA
BCT T45 ;ITHR=0 IF NO DATA
CLR ITHR
T45: MOV BACK,0BX ;STORE IN TTAB AND BTAB
ADD #2,IX
ADD #2,BX ;INCREMENT PointERS
T50: ;CALL OUTCON,(ITHR,...,FOUR)
ADD #7,T50+10
T60: ;CALL OUTCON,(BACK,...,THREE)
ADD #7,T60+10
SUB #128,R4
CMP R4,#HIST
BNIS T19 ;BR IF MORE HISTOGRAMS TO ANALYZE
SUB #6,T60+10
CLR R6+10
INC SHUM
CALL OUTCON,(SHUM,NO+2,THREE)
ADD HC,SHUM
BEC SHUM
MOV R0,#NO+3
CALL OUTCON,(SHUM,NO+3,THO)
BIT #2,#0,SWR
BEG T70 ;BR IF SW I DOWN
CALL TYPE,NO
T70: RTS PC

THEAD: MOV HC,R1 ;JSR PC,THEAD SET UP HC HEADINGS
MOV #MSG,R2
TH10: MOV #THEAD,R3
MOV #7,R4
TH20: MOVB (R3)+(R2)+
SOB R4,TH20 ;MOVE A SECTOR HEADING TO MSG
SOB R1,TH10 ;BR IF MORE SECTORS
CLR R0
MOV #6,R4
MOV #NO,R2
MOV #THEAD,R3 ;FINAL HEADING
TH30: MOVB (R3)+(R2)+
SOB R4,TH30
BIT #2,#0,SWR
BEG T70 ;BR IF SW I DOWN
TH70: RTS PC

SMOOTH: MOV R4,-(SP) ;JSR PC,SMOOTH SAVE R4
MOV #HIS+2,R3
SH20: MOV #62,R2
MOV (R4)+,R0 ;HIS(I)=HIST(I-1)+HIST(I)+HIST(I+1)
ADD #R4,R0
ADD 2(R4),R0
MOV R0,(R3)+
S0B R2,SH20
MOV (SP)+,R4
RTS PC

TSUB: MOV #HIS,R0 ;R0=BACK=HIS+0
MOV #18,R1
MOV BK512,R2 ;BACK RANGE IS NOMINALLY 2-40
MOV #HIS+2,R3
TS40: CMP R1,(R3)+
BGE TS45 ;52 IF MAX1 GE HIS(I)
MOV -(R3),R1 ;MAX1=HIS(I)
MOV R3,R0 ;R0=HIS+1
TST (R3)+
TS45: SUB R2,TS40 ;BR IF MORE TO SEARCH
SUB #HIS,R0
MOV R0,BACK ;STORE BACK
MOV #HIS,R0
MOV #4, R1
MOV DTSIZ, R2
DTSIZE IS NOMINALLY FROM 58-124
TS50: CMP R1, (R3)+
BGE TS60
CMP -2(R3), -4(R3)  LOOK FOR RELATIVE MAXIMUM ON LEFT SIDE
BLO TS50
MOV -2(R3), R1
TS60: MOV R3, R0
TS60: CLR HIS, R0
MOV RB, @DATA
STOR DATA
RTS PC

MAXHL=512.
MAXXS=409.

KEY:

NK-6

SL: WORD 60. STARTING LINE
EL: WORD 586. ENDING LINE
SS: WORD 0. STARTING SAMPLE
DBEG: WORD 53. DATA BEGIN
SPS: WORD 38. SAMPLES PER SECTOR OF A FIELD
ISPS: WORD 39. INITIAL SPS
BKSIZ: WORD 24. dBEG/2-1
DTSIZ: WORD 38. BKSIZ
SIZE: WORD 12288. SIZE OF EACH 24-LINE BUFFER
SCTR: WORD 8.
HDIM: WORD 512. HIST DIMENSION (512 WORDS)
SPACE: WORD 49.
EJECT: WORD 61.
UIC: WORD 2. DEFAULT UIE IS 2.2
SYTY: WORD 70.
ZERO: WORD 0.
ONE: WORD 1. VARIABLE
TWO: WORD 2.
THREE: WORD 3.
FOUR: WORD 4.
FIVE: WORD 5.
X: WORD -.
Y: WORD -.
LINE: WORD -.
QFLAG: WORD 0. FLAG FOR QUICK LOOK (WHILE SCANNING)
BFLAG: WORD 0.
INP: WORD 56. REL BUFFER ADDR FROM PUT
HPAR: WORD -.
JPAR: BLK W 4. JOYSTICK PARAMETERS
PAR: BLK W 88.
NMAX: WORD 88.
NPAR: WORD 8.
NL: WORD 498. 10 LINES
NS: WORD 472. 10 SAMPLES
NLR: WORD 1.
HIST: BLK W 512.
BLKSIZ
HIS: BLK W 64.
SMOOTHED HISTOGRAM
TDATA: NR: WORD 8.
NR: WORD 16.
LPS: WORD 63.
SPS2: WORD 68.
TTab: BLK W 64.
BUTAB: BLK W 64.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
FY: WORD 48.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
BPER: WORD 40.
**C**

***** BINARY PICTURE GENERATOR *****

SUBROUTINE BINARY (BUNIT, FILPEX)

BYTE FILPEX(99)

IMPLICIT INTEGER (A-Z)

COMMON/C1/VR, NC, LPS, SFS, TTAB(64), BTAB(64), BPER

COMMON/C1/SLO, SSD, HLD, HSO, HOB, MAHNO3

COMMON/C1/ARED, MAREA, NWIDE, XWIDTH, NAME, MNEPT, MXEPT, MAXLEN, NUM, TDB, ISWI

COMMON/C1/ WORD(4128), PIC(12344), PBUF(512)

INTEGER PAR(29), KEY(6), SPAR(5), CTAB(307), NTAB(307)

BYTE PIC, PMSG(22), PBUF

DATA NAME/"BINARY", "NAME", "PAR", "LPS", "SFS", "TTAB(64)", "BTAB(64)", "BPER disadvantage"

CALL SSWITCH(4, ISW1)

IF(ISW4.EQ.1) CALL TIMER

CALL SSWITCH(1, ISW1)

IF(ISW1.EQ.1) CALL TYPE(‘BINARY’)

CALL SPARMAPAR(PAR, 20)

C HPAR WILL BE ZERO THE FIRST TIME BINARY IS RUN ON A SPREAD

IF(HPAR.EQ.0) CALL TYPE(‘STARTING ANALYSIS ON’)

CALL AFILE(PIC, DUNIT, FILPEX, 2, 2)

CALL OPEN(PIC, DUNIT, IF, ‘SCN’)

CALL CLABEL(PIC, SPAR, INDEX)

IF(HPAR.EQ.0) CALL TYPE(PIC(INDEX+73), 64)

C DEFAULT PARAMETERS

B = 3

MAXNO3 = 85

MAXAREA = 32

MAXLEN = 2000

MNEPT = 0

MXEPT = 200

SLO = 1

MAXLEN = 99

S30 = 1
NLO = SPAR(I)
NSO = MINB(SPAR(2),311)
CALL GET(PIC.NLO+1),I)
CALL NVW(PIC(I+1),NR,133)

CALL RPARAM(HPAR,PARI20)
I = -2
2  I = I + 2
IF(I .GT. HPAR) GOTO 20
DO 5 K = 1,10001
5 CONTINUE

C PARAMETER ERROR
PAUSE 1
GOTO 20
DO 12 I = PARI20
GOTO 3
12 TSNH = PARI12
NSHRE = PARI1
MNREA = PARI1
MNARE = PARI11
GOTO 2
14 I = I + 2
MNERT = PARI1
MNERE = PARI11
GOTO 2
15 MAXLEN = PARI12

C DO 70 R = HRH, LPS
DO 398 R = HRH
398 IF(R .LE. FP, NR) LPS = RSNPS.

C continued
IF(ISW4.EQ.1) CALL TIMER
N = (B+1)/2
SPS = (SPS+1)/2
MSAMPS = NLO - (CHR-1)*LPS
HW = (NSO+1)/2
HW = HW+16

C *** BINARY PICTURE AREA(WORD) IS ASSUMED TO BE ZERO INITIALLY
C HW = NUMBER OF WORDS OUT PER LINE
C HW = NUMBER OF WORDS IN PER LINE
S = 0
C = 0

C CTAB(30) = COLUMN TABLE (CONTAINS COLUMN INDEX FOR WORD)
C NTAB(30) = NUMBER OF SAMPLES OF WORD BELONGING TO THE
C CURRENT COLUMN
DO 50 W = 1, HW
50 CONTINUE

C continued
IF(S.LT.SPS) GOTO 50
S = S + 16
CONTINUE
IF(ISW4.EQ.1) CALL TIMER
C
SI = 1
WI = HW
REC = 1
ILPS = LPS

C DO 338 R = 1, NR
DO 338 R = 1, NR
338 IF(R .EQ. NR) ILPS = MSAMPS

C continued
DO 290 L = 2, ILPS, 2
290 CALL GET(PIC.REC.P1)
REC = 0
CALL NVW(PIC(P1+1),PBUF,HWI)
CALL GET(PIC,0,P1)
103  
C       S = 1
C       IF(IS4.U.EQ.1) CALL TIMER
C       IF(T0BH.EQ.0) GOTO 250
C       DO 200 W = 1, H UT
C       CALL OTHR(PBUF(S), W ORED(WI+W), TTAB(CTAB(W)+SI), PIC(P I+S), MTAB(W))
C       S = S + 32
C       IF(IS4.U.EQ.1) CALL TIMER
C       GOTO 290
C
C 250  CONTINUE
C       DO 200 W = 1, H UT
C       CALL OTHR(PBUF(S), W ORED(WI+W), TTAB(CTAB(W)+SI), PIC(P I+S), MTAB(W))
C       S = S + 32
C       IF(IS4.U.EQ.1) CALL TIMER
C 290  W I = W I + H V
C
C    300  SI = SI + NC
C    IF(NPAR.EQ.0) CALL TYPE(0)
C    PAR(1) = "LE"
C    PAR(3) = 90
C    CALL WPARAM(3, PAR, 3)
C    WRITE PARAMETERS IN CASE OF RERUN
C    CALL CLOSE(PI)
C    IF(IS4.U.EQ.1) CALL TIMER
C    CALL SEGNST
C    IF(IS4.U.EQ.1) CALL TIMER
C
C END

SUBROUTINE SEGNST
IMPLICIT INTEGER(4-2)
COMMON/C1/SPAR(133), SLO, SSO, HLO, HSO, HSB, HMAX, HMAX1
COMMON/C1/MMAREA, MAREA, MPERM, MMAX, MMAX1, NW, T0BH, ISVI
COMMON/C1/ WORK(128), NPERM, YMIM, HWIM, WMX, XM X MAX
COMMON/C1/DIR(512), EBUF(55), EEDGE(40, W), EMAX
COMMON/C1/NK, MLP(60), SIP(60), FI
INTEGER PAR(5)
BYTE LABEL(S12), M(6), WHITE(14)
DATA PAR/S12, 1924, 7, 1, 512/
DATA LABEL/ 79*, 'S', 'C', 79*, 'A', 'L, 368* '
DATA MAXH/1922/, DREC/2/
C THE EDGE POINT DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJECT
C DIR(B1) = FBN (FIRST BLOCK WRITTEN)
C DIR(B1) = N (NUMBER OF END POINTS)
C DIR(B1+2) = YM
C DIR(B1+3) = XM
C DIR(B1+4) = XM
C DIR(B1+5) = XM
C THE END POINTS ARE GROUPED IN COORDINATE PAIRS REPRESENTING SEGMENTS
C EDGE(E+1) = X1
C EDGE(E+2) = Y1
C EDGE(E+3) = X2
C READ OLD VALUE OF FI
C MARS = MAREA/4
C MAX = MMAREA/4
C MNFR = MPERM/2
C MAX = MPERM/2
C MAREA = MARE
C CALL SSWITCH(1, ISU1)
C IF(ISU1.EQ.1) CALL TYPE(' SEGNST')
C CALL SSWITCH(4, ISU4)
C IF(IS4.U.EQ.1) CALL WPRINT(' SEGNST')
C CALL ZIA(CBUF, 29)
C CALL ZIA(DIR, 510)
C CALL ZIA(EDGE, 1024)
C CALL MLA('EDGE FILE', LABEL(73), 19)
C IF(T0BH.LT.0) GOTO 14A
C CALL FILE(EDUF, 1, 'PIC', 5, 5)
C CALL OPEN(EDUF, 124, 0, 1, 'MV2')
C CALL FLAC(EDUF, PAR, LABEL)
C CALL PUT(EDUF, S, HUT)
CONTINUE

REC = 3
WU = WU / 2
WU2 = WU2 / 2
W2 = W2 / 2

N0 = NUMBER OF OBJECTS
WI = WORD INDEX EW1 = ENDING WORD INDEX
EI = EDGE INDEX DI = DIRECTORY INDEX
DI = BIT INDEX (*0, 1, 2, ..., 31)

L U = WU
DI = 1
N0B = 1

DO 200 L = 1, HL
WI = EW1 + 1
EW1 = EW1 + NU
LOOP = 0

DO 200 L = 1, HL
WI = EW1 + 1
EW1 = EW1 + NU
LOOP = 0

THE ROACH WILL SCAN LINE FOR OBJECTS (ONE-BITS)

140 IF (ROACH(WI, WI1, WORD, EW1, MASK), EO = 0) GOTO 200
LOOP = LOOP + 1
IF (LOOP, EO = 200) PAUSE 1.
EDGE(1) = L
EDGE(2) = (W2 - (W2 - WU1 + 1)) / 16 + 81 + 1
CALL TURTLE (N, EDGE, WORD(W1), MASK, W2)
IF (EO = 0), GOTO 140
W2 = 2 + H
IF (NOB = 0), GOTO 141
CALL LIST (NOB, W, YMIN, XMIN, YMAX)

142 IF (H2 = MAX, GOTO 143)
CALL TYPE (' TOO MANY EDGES')
CALL TYPE ('')
GOTO 200

CONTINUE

IF (SWI1 = 1), CALL TYPE (' SORT')
IF (ISW1 = 1, CALL STIMER
CALL SORT (EDGE, W)

N = NUMBER OF EDGE POINTS

145 CALL ERASE (WORD, EDGE, AREA, W, NW)
IF (HEIGHT, LT, H, GT, HMAX) GOTO 195
IF (AREA, LT, HMIN, OR, AREA, GT, HMAX) GOTO 196
IF (YMAX - YMIN, GE, XMEN, OR, XMEN - XMAX, GE, XMEN) GOTO 196

ENTER OBJECT INTO DIRECTORY

IF (H, GE, 60, OR, H, GE, 0), GOTO 175
NK = NK + 1
LIP (NK) = EDGE(1) + EDGE(1)
STP (NK) = EDGE(2) + EDGE(2)

STORE NOG INFORMATION FOR QUICK COUNT OPTION

Y = #EDGE(1)
X = #EDGE(2) - 14
IF (X, LT, 0) X = 0

WRITE AN ARROW BY THE OBJECT

CALL DLINE (WHITE, Y, X, 8.1, 0)
CALL DLINE (WHITE, Y, X, 8.1, 0)

CALL SSTORE (S, ISW6)
IF (ISW6, EO, 1), GOTO 175

CALL DLINE (WHITE, Y, 4, X, 8.2, 1.0)
CALL DLINE (WHITE, Y, 4, X, 8.2, 1.0)
CALL DLINE (WHITE, Y - 2, X, 10, 2.1, 0)
CALL DLINE (WHITE, Y, 2, X, 10, 2.1, 0)
CALL DLINE (WHITE, Y, 2, X, 10, 2.1, 0)
CALL DLINE (WHITE, Y, 2, X, 10, 2.1, 0)
CONTINUE

    DIR(DI) = REC
    DIR(DI+1) = N
    DIR(DI+2) = MAX(YM09.NB.B.1)
    DIR(DI+3) = MAX(0(NB.B.1)
    DIR(DI+4) = MIN0(0X(00.B.NL)
    DIR(DI+5) = MIN0(0X(00.B.NS)
    NOB = NOB + 1
    DI = DI + 6
    IF(NOBI,GT,MAXNOB) GOTO 201

C

    IF(TODN,LT,0) GOTO 140
C DON'T WRITE IF QUICK COUNT OPTION
    DO 195 1 = 1, NW2, 512
    CALL WRITE(EBUF, REC, EDGE(I))
    REC = REC + 1
    GOTO 140

C

    CALL SSST(4, IS4)
    IF(IS4 .NE. 1) GOTO 140
    CALL QPRINT( ' OBJECT REJECTED'
    CALL LIST( NOB, N, AREA, YMIN, XMIN)
    CALL LIST(YMAX - YMIN, XMAX - XMIN, NXEP.T, NXLEN)
    GOTO 140

C

    MAXH = MAXH + NW

C

    NOB = NOB - 1
    IF(TODN, LT, 0) GOTO 220
    IF(DI, GT, 1) CALL WRITE(EBUF, BREC, DIR)
    CALL WRITE(EBUF, 1, SPAR)
    CALL CLOSE(EBUF)
    GOTO 140

C

    CALL CSST(C, IS4)
    IF(IS4 .NE. 1) GOTO 140
    CALL QPRINT( ' OBJECT REJECTED'
    CALL LIST( NOB, N, AREA, YMIN, XMIN)
    CALL LIST(YMAX - YMIN, XMAX - XMIN, NXEP.T, NXLEN)
    GOTO 140

C

    IF(CT, GT, 0) RETURN
    CALL MVL( 'NC==', M, 6)
    CALL OUTCON(NH, M(6), 2)
    CALL TYPE(M, 6)
    CALL UPARAM(122, NH, 6)
    CALL APASE(7)

C

SUBROUTINE SKIRT

C THE SEGMENT DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJECT
C
    SDIR(DI) = BSIC(BEGINNING SEGMENT INDEX)
    SDIR(DI+1) = ESI(ENDING SEGMENT INDEX)
    SDIR(DI+2) = YMIN
    SDIR(DI+3) = XMIN
    SDIR(DI+4) = YMAX
    SDIR(DI+5) = XMAX
C
    THE SEGS ARE STORED AS TRIPLETS
    SEG(5) = Y
    SEG(5+1) = X
    SEG(5+2) = Y
C
    THE OBJ DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJECT
C
    DIR(DI) = BACKGROUND
    DIR(DI+1) = THRESHOLD
    DIR(DI+2) = YMIN

C DIR(D1+2) = XMIN
C DIR(D1+4) = YMAX
C DIR(D1+5) = XMAX
C CALL SSCH(C1, ISWI)
C IF(ISWI.EQ.1) CALL TYPE(' SKIRT')
C CALL AFILF(EBUF, .1, 'PIC', .5, 5)
C CALL OPEN(EBUF, 124, 0, 'HV2')
C CALL AFILF(SECB, .5, 'PIC', .5, 5)
C OPEN(SEGB, 1024, 0, 'HV3')
C CALL GLABEL(EBUF, PAR, IND)
C CALL MVL(' SEGMENT FILE', EBUF(INH+73), 13)
C CALL PLABEL(SEGB, PAR, EBUF(INH+1))
C CALL PUT(SEGB, 1, IND)
C GET SHAPE PARAMS
C CALL REAEBUF, IND, PAR)
C DPER = PAR(133) - 5
C DREC = 2*(XNOB+Y94)/85+2
C CALL WRITB(SECB, 1, SLD)
C HL = NLO/2
C NS = NSO/2
C B2 = 2*B + 1
C SIN = MAXB/B2
C MAXB = SIN*B2
C
C S = SEGMENT INDEX
C SI = SEGMENT BUFFER INDEX
C
C DO 200 OBX=1, WOB
C IF(DI NE 1) GOTO 30
C OPEN(EBUF, 1, IND, IND, DIR)
C IDREC = (DIR + 1)
C CALL MVW(DIR, SIR, 512)
C IREC = DIR(D1)
C NS = DIR(D1+1)
C SDIR = S
C CALL INTERP(D1)
C NEMD = 2**H
C NREC = (NEMD+511)/512
C
C DO 130 L=1, HREC
C CALL REAEBUF, IND, IND, EDGE)
C IREC = IREC + 1
C EIN = MINU(NEMB, 512)
C ADD HORIZONTAL SKIRT TO SEGMENTS
C DO 49 E=2, EIN, 1
C EDGE(E) = MAXV(EDGE(E)+B-1)
C EDGE(E+2) = MINU(EDGE(E+2)+B, NS)
C
C IF(L.NE.1) GOTO 60
C YB = EDGE(E)
C YBAR = YB - B
C USE FIRST SEGMENT END POINT PAIR TO INITIALIZE SBUF
C XI = EDGE(2)
C X2 = EDGE(4)
C SI = 1
C I = 1
C E1 = 5
C 60 CONTINUE
C PROCESS REMAINING SEGMENT END POINTS
C DO 120 E=E1, EIN, 4
C Y = EDGE(E)
C IF(Y.EQ.YB) GOTO 90
C ADVANCE SBUF ONE ROW
C SI = SI + SIN
C I = I + 1
C IF(SI GT MAXB) SI=1
IF(I.GT.82) I=1
IF(YBAR.LT.0) GOTO 82
SEND = ESL(I)
DO 60 SIP=SI, SEND,2
SEG(S) = YBAR
SEG(S+1) = SBUF(SIP)
SEG(S+2) = SBUF(SIP+1)
60 S = S + 3

82 Y0 = Y0 + 1
YBAR = YBAR + 1
SBUF(SI) = EDGE(E+1)
SBUF(SI+1) = EDGE(E+3)
ESI(I) = SI

90 SIP = 1
X10 = EDGE(E+1)
X20 = EDGE(E+3)
DO 110 II=1.02
S1Q = SIP
SEND = ESL(II)
XI = X10
X2 = X20

93 XP1 = SBUF(SIQ)
IF(X2.LT.XP1-1) GOTO 95
XP2 = SBUF(SIQ+1)
IF(X1.GT.XP2+1) GOTO 95
SEGMENT (X1,X2) IS CONNECTED TO A PREVIOUS SEGMENT
IF(SIQ.EQ.SEND) GOTO 94
X1 = MINB(X1,XP1)
X2 = MAXB(X2,XP2)
SBUF(SIQ) = SBUF(SEND)
SBUF(SIQ+1) = SBUF(SEND+1)
SEND = SEND - 2
GOTO 93
94 SBUF(SEND) = MINB(X1,XP1)
SBUF(SEND+1) = MAXB(X2,XP2)
GOTO 98
95 SIQ = SIQ + 2
IF(SIQ.LE.SEND) GOTO 93
SBUF(SIQ) = XI
SBUF(SIQ+1) = X2
98 ESL(II) = S1Q
100 SIP = SIP + SINC
120 CONTINUE

EI = 1
NEND = NEND - EI

RELEASE REMAINING SEGMENTS
DO 150 II=1.82
IF(YBAR.GT.NL) GOTO 180
I = I + 1
IF(SI.GT.MAXSB) SI=1
IF(I.GT.82) I=1
SEND = ESL(I)
DO 140 SIP=SI, SEND,2
SEG(S) = YBAR
SEG(S+1) = SBUF(SIP)
SEG(S+2) = SBUF(SIP+1)
140 S = S + 3
150 YBAR = YBAR + 1
180 SDLR(DI+1) = S - 1
SUBROUTINE CHROMEB(DUNIT, FILPEX)
   BYTE FILPEX(9)
   IMPLICIT INTEGER(A-Z)
   COMMON/C1, SLO, SS0, HLO, HSD, NO8, B
   COMMON/C1, G1, SR(X512), CMVB(512), C(512), PBUF(1389)
   C THE FIRST 24 WORDS OF PBUF ARE THE MVH, FOLLOWED BY 1024 WORDS OF
   C THE FIRST 28 WORDS OF PBUF ARE THE MVB, FOLLOWED BY 1024 WORDS OF
   C BUFFER STORAGE, FOLLOWED BY 256 WORDS OF PICTURE STORAGE
   COMMON/C1, SEG(56), CBUF(569), CDIR(S10), LBUF(108)
   COMMON/C1, OREC, MFSI
   BYTE SEG(B(216))
   EQUIVALENCE (PB, PBUF)
   INTEGER PAR(5)
   DATA SECSIZ/64, SSM1/63/1
   DATA MAXST/512, MAXBUF/5689 - IDREC/2, LBUFF/1032 - IDREC/2/
   C LPBUF IS THE LENGTH OF PBUF MINUS 256 WORDS OF PICTURE STORAGE
   C
   CALL SSWITC(1, ISW1)
   IF(ISW1.EQ.1) CALL TYPE('CHROMEB')
   CALL SSWITC(4, ISW4)
   IF(ISW4.EQ.1) GOTO 3
   CALL OPRINT('CHROMEB')
   CALL TIMER
   MAXBUF=7328

   CONTINUE
   CALL AFIL(F(0, DUNIT, FILPEX, 2), 2)
   CALL OPEN(CEB, LBUF, 29, 1, 0, 0, 'SCN')
   L=LABEL(PBUF, LBUF, IP)
   NW=(PAR(3)+1)/2
   CALL AFIL(CEB, 5, 'PIC ', 5, 5)
   CALL OPEN(CEB, 1024, 0, 0, 'MVH')
   CALL GLABEL(CEB, PAR, IND)
   CALL Z1A(CMVB, 20)
   CALL AFIL(CMV, 1, 'PIC ', 5, 5)
   CALL OPEN(CMV, 1024, 1, 0, 'MV2')
   CALL MVL('CHROMA FILE', PB(IP+188), 12)
   CALL PLABEL(CMV, PAR, PB(IP+1))
   CALL READ(CEB, 1, IND, SLO)
   CALL PUT(CMV, 1, IC)
   CALL MV(DCB(CMV, IC/2+1, 512)
   NREC=-(HOB+04)/85
   IREC = 2*NREC + 2
   OREC = NREC + 3
   CALL READ(CEB, IDREC, IND, DIR)
   CALL PUT(CMV, DRRC, IC)
   CALL MV(DIR, CMVB, IC/2+1, 512)
   CALL READ(CEB, IDREC+1, IND, DIR)
   IDREC = IDREC + 2
   OREC = OREC + 1
   C
   FORMAT CBUF
   BEND = NO8 + 6
   SEND = DIR(BEND-4)
   DISP = MAXBUF-SEND
   C
   J = DISP + 1
C READ IN SEGMENT FILE

DO 5 I=1,SEND+512
CALL READ(SEGB,IREC,IND,OBJ(J))
J = J + 512
5 IREC = IREC + 1

DO 6 DI=1,BEND,6
DIR(DI) = NDIR(DI) + DISP
6 DIR(DI+1) = DIR(DI+1) + DISP

THIS = (DISP-1)/SECSIZ
N = THIS+SECSIZ

DO 10 I=N,M,SECSIZ

CBUF(I) = I+SECSIZ

LFSI = N - SECSIZ
CBUF(LFSI) = 9

C THE SEGMENT DIRECTORY CONTAINS THE FOLLOWING INFORMATION FOR EACH OBJ
C
C DIR(DI) = CSI = CURRENT SEGMENT INDEX
C DIRECT над S1) = ESI = ENDING SEGMENT INDEX
C DIRECT над S2) = YMIN
C DIRECT над S3) = YMAX
C DIRECT над S4) = XMAX
C
C THE CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING INFORMATION FOR EACH
C
C CDIR(DI) = SSI = BEGINNING SEGMENT INDEX
C CDIR(DI+1) = NSI = TOTAL NUMBER OF SECTIONS USED BY OBJ
C CDIR(DI+2) = FSI = INDEX OF SECTION USED BY OBJECT
C CDIR(DI+3) = LSI = INDEX OF LAST SECTION USED BY OBJECT
C CDIR(DI+4) = FWS = FIRST HALFWORD OF SECTION USED BY OBJECT
C
C CDIR(DI+5) = LHY = LAST HALFWORD OF SECTION USED BY OBJECT
C
C LFSI = NEXT FREE SECTION INDEX
C
NFSI = 1
DI = 1

50 DBEG = DI.
   IF(ISW4.EQ.1) CALL TIMER
C INCREASE NUMBER OF SECTIONS IF POSSIBLE
   CSI = DIR(DI)
   BSI = CDIR(DI)
   IF(BSI.EQ.0) BSI = CSI
   NSI = LFSI + SECSIZ
   IF(NSI.GT.BSI-SECSIZ) GOTO 52
   CBUF(LFSI) = NSI
   LFSI = MNSI
   GOTO 51
52 CBUF(LFSI) = 0
   YB = CBUF(CSI)
   REC = 2*YB - 1
   CALL GET(PBUF,REC,IND)
   CALL MNV(PBUF,IND+2+(1).PBUF(LPBUF+1).NU)
   CALL GET(PBUF,0.IND)
   IP = IND/2
C
C PROCESS OBJECTS 1 TO NMB
C
   IF(ISW4.EQ.1) CALL TIMER
   DO 100 DI=DBEG,BEND,6
   CSI = DIR(DI)
   IF(CSI.EQ.0) GOTO 100
   Y = CBUF(CSI)
   IF(Y.GT.YB) GOTO 110
   ESI = DIR(DI+1)
   LSI = CDIR(DI+3)
   NU = CDIR(DI+5)
   HSEC = 0.
   IF(LSI.NE.0) GOTO 58
   FIRST SEGMENT OF OBJECT. ASSIGN A SECTION
   IF(NFSI.EQ.0) CALL TYPE(" WBUF")
   IF(NFSI.GE.0) CALL WBUF(DBEG,BEND,SI)
C LOOP THROUGH EACH SEGMENT OF CURRENT OBJECT

50 DO 90 SI=CSI,ESI,3

Y = CBUF(SI)

IF(Y.GT.YB) GOTO 82

X1 = CBUF(SI+1) + LPBUF
X2 = CBUF(SI+2) + LPBUF

C DO 70 L=1,2

C DO 62 X=X1,X2

IF(LH.W.LT.SECSIZ) GOTO 60

ADD ON A NEW SECTION

IF(NFS1.EQ.0) CALL TYPE(UMBUF2)

IF(NFS1.EQ.0) CALL WMBUF(3BEG,2END,DI)

NSEC = NSEC + 1
CBUF(LSI) = NFS1
LSI = NFS1
NFS1 = CBUF(NFS1)

LHY = 1

C 60 CBUF(LSI+LHY) = PBUF(X).

62 LHY = LHY + 1

C XI=CBUF(SI+1)+IP
70 X2=CBUF(SI+2)+IP

C DO 90 CONTINUE

C

CDIR(DI+3) = LSI
CDIR(DI+5) = LHY

C END OF OBJECT WRITE OUT TO DISK

C IF(ISW4.EQ.1) CALL TIMER

101 CALL WMBUF(DI)

C IF(CSI.NE.9) GOTO 91

C IF(ISW4.EQ.1) CALL TIMER

GOTO 100

C REMAINING PORTION OF OBJECT LIES BELOW CURRENT LINE

C UPDATE DIRECTORIES AND CONTINUE

82 CDIR(DI+1) = CDIR(DI+1) + NSEC
CDIR(DI+3) = LSI
CDIR(DI+5) = LHY
DIR(DI) = SI

C 100 CONTINUE

C

NO MORE OBJECTS ON THIS LINE UPDATE AND CONTINUE

C

110 CONTINUE

C IF(ISW4.EQ.1) CALL TIMER

DO 120 DJ=3BEG,2END,6

C SI = DIR(DI)

IF(CSI.NE.0) GOTO 50

120 CONTINUE

C

IF(ISW4.EQ.1) CALL TIMER

CALL PUT(CNVB,3BEG,IC)

CALL NUW(CNVB,CNVB(IC/2+1),102)

CALL CLOSE(CNVB)

CALL CLOSE(PBUF)

CALL CLOSE(SEGB)

IF(ISW4.EQ.1) CALL TIMER

END
SUBROUTINE ROB
IMPLICIT INTEGER(A-Z)
COMMON/C1/CNVB(56),CS512
COMMON/C1/CMVB(56),EDGE(1024),LBW(56),BUF(6TBL
COMMON/C1/SLID,SSR,RLO,RSV,MOD,B.DPER
COMMON/C1/PERI,YMIN,XYMIN,YMAX,YMAX
COMMON/C1/DSIR(60),TTRAB(60),AREA(60),IDR(60)
COMMON/C1/NK,LIP(60),SIP(60),PERM(60),CIRCA(60)
LOGICAL I = BUF(1000)
BYTE CMVB,SNVB,STBL(128),MSG(20),MSG(25),TMC(128)
INTEGER DIR(SIR),PAR(I00),HIS(64),OTTAB(I00)
EQUIVALENCE (DIR,DSIR(145))=(HIS.EDGE)
DATA MAXSUB/100000,RECSIZ/1024,NOBP/6,
DATA HGCR/1024,HPCTOL/6,MAXRNB/60,TOBH/8,MAXHP/180
DATA OREC/25,REC/25,RFLAG/1,OTTAB/60/0
C
C THE INPUT CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJ
C 1) DIR(B1) = BACKGROUND
C 2) DIR(B1+1) = THRESHOLD
C 3) DIR(B1+2) = CMS (MINIMUM LINE)
C 4) DIR(B1+3) = CMS (MINIMUM SAMPLE)
C 5) DIR(B1+4) = CMXL (MAXIMUM LINE)
C 6) DIR(B1+5) = CMXS (MAXIMUM SAMPLE)
C
C THE OUTPUT CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJ
C SDIR(CDI) = FBO (FIRST RECORD BLOCK WRITTEN TO DISK)
C SDIR(CDI+1) = NL (MINIMUM LINE)
C SDIR(CDI+2) = NS (MINIMUM SAMPLE)
C SDIR(CDI+3) = LBW (LAST RECORD BLOCK WRITTEN TO DISK)
C
C THE INPUT CHROMOSOMES ARE IN THE FOLLOWING FORMAT
C C(I) = PREVIOUS BLOCK WRITTEN
C C(I+2) = HSEG (NUMBER OF SEGMENTS IN THIS RECORD)
C C(I+3) = Y (LINE COORDINATE=0 - 255)
C C(I+4) = X (STARTING SAMPLE COORDINATE=0 - 255)
C C(I+5) = N (NUMBER OF SAMPLES IN SEGMENT)
C
C NOTE THAT TWO CONSECUTIVE LINES OF SEGMENT GREY VALUES MUST BE
C ALSO THE DIMENSIONS ARE THOSE OF THE HALF-PICUTURE AND MUST BE
C DOUBLSED.
C
C THE OUTPUT CHROMOSOMES ARE IN THE FOLLOWING FORMAT
C C(I) = HSEG (NUMBER OF SEGMENTS IN THIS RECORD)
C C(I+1) = Y (LINE COORDINATE=1 - 512)
C C(I+2) = X (STARTING SAMPLE COORDINATE=1 - 512)
C C(I+3) = N (NUMBER OF SAMPLES IN SEGMENT)
C
CALL SSWIF(1,ISW1)
IF(ISW1.EQ.1) CALL TYPE('ROB')
CALL SSWIF(4,ISW)
IF(ISW.EQ.1) CALL PRINT('ROB')
CALL RPARAM(HP,PAV.MAXHP)
IP = 1
28 IF(IP.GT.HP) GOTO 35
IF(PAR(IP,EQ.'TO') GOTO 30
IF(PAR(IP,H'C') GOTO 35
OTTAB(PAR(IP+2))=PAR(IP+3)
IP=IP+4
GOTO 20
30 TOBH=PAR(IP+2)
35 GOTO 20
35 CONTINUE
RECSIZ=RECSIZ/2
LBW(56) = LAST BLOCK WRITTEN FOR CURRENT CHROMOSOME
IF(DIR=N) DIR=NUMDIR
LC = NUMBER OF CHROMOSOMES
CALL AFILE(CNVB,1,'PIC',5.5)
CALL OPEN(CNVB,1024,6,'MV2')
CALL AFILE(SNVB,5,'UCR'
CALL OPEN(CMVB,1024,6,'UCR')
CALL GLABEL(CMVB,PAR,IND)
CALL NVL('RGB FILE',CMVB(INB+139),12)
CALL PLABEL(SNVB,PAR,CMVB(INB+1))
CALL SSWIF(0,ISW)
IF(ISW.EQ.1) GOTO 37
CALL OPEN('1')
CALL PPINT('1')
CALL QPRINT(CMVB IND=145), 30
CALL QPRINT(CMVB IND=217), 70
CALL QPRINT('009', PER=8, B MAX=LINE, SAMPL
1 SL SS HL HS T1 T2 FPOH AREA 10$9 AVG
2 OBJ)
37 CALL ILA (32, TMSG, 128)
CALL PUT (CMVB, DREC, IND)
CALL READ (CMVB, IND, SLO)
CALL READ (CMVB, DREC, IND, DIR)
DREC = DREC + 1
CALL READ (CMVB, DREC, LBW)
DPER = 100 - DPER
B = 2*B
B1 = B + 1
B2 = B + 2
B3 = 2*B + 1
D1 = 1
CDB = 2
C
   LOOP THROUGH EACH OBJECT
   DO 500 OBH=1, NOB
   REC = LBW(OBH)
   D1 = D1(D1*2) + 1
   CMS = D1(D1*3) + 1
   CNL = D1(D1*4) + 2
   CML = D1(D1*5) + 2
   CMS = CMS + CMS + 3
   CNL = CNL - 1
   CMSL = CMS - 1
   IF (CHL = CMS, LT, MAXBUF) GOTO 50
   CALL SSWITCH (1, ISU)
   IF (ISU(NE.1) GOTO 580
   CALL HVL ('OBJECT IS TOO LARGE', MSG7, 20)
   CALL OUTCOM (OBH, MSG7, 10, 2)
   CALL TYPE (MSG7, 20)
   GOTO 580
50 C
   READ IN CURRENT CHROMOSOME
   GOTO 21A (BUF, MAXBUF/2)
   CALL ZIA (BUF, MAXBUF/2)
   CALL ZIA (HSN, 64)
   C
   CALL READ (CMVB, REC, IND, C)
   HSEC = C(2)
   CI = 3
   C
   DO 60 HH=1, HSEC
   B1 = 2*CCCL(CI*1), CMS+CCCL(CI*1), CMS+1
   W = C(CI*2)
   N2 = 2*H
   IF (BI. LT. 1) GOTO 60
   C
   IGNORE POSSIBLE BUF FROM CHROME OUTPUT ******
   IF (BI. LT. MAXBUF) GOTO 45
   CALL (HSN, C(CI+3), BUF(BI), HIS, N2)
   CALL NHIS(C(CI+N2), BUF(BI+CNS), HIS, N2)
   C
   CI = CI + N2 + 3
   C
   REC = C(1)
   IF (REC. NE. 0) GOTO 51
   C
   IF (OBH. NE. 0) GOTO 51
   C
   CALL PRINT (I, BUF(BI), CNS)
   C
   B1 = B1 + CNS
   C
   CONTINUE
   60 C
   BACK = D1(BI/2)
   ITHRES = D1(BI+1)/2
   G
   RETRESHOLD OBJECT
   HIS(I) = 0
   DATA = ITHRES
   THRESH = ITHRES
123
---
IF(FLAG.EQ.0) GOTO 72
HIS(64) = 0
MAX = 0

C
D0 65 I=ITHRES-63
F = HIS(I-1) + HIS(I) + HIS(I+1)
IF(F.LE.MAX) GOTO 65
MAX = F

DATA = 1

65 CONTINUE
THRESH = (SOPER+BACK.*SPER+DATA)/100
IF(OTTAB(OBN).NE.0) THRESH=OTTAB(OBN)/2

72 CALL STRECH(HIS,STBL,THRESH,DIAM)

C
BI = BUFFER INDEX
EI = EDGE INDEX
DI = DIRECTORY INDEX
EBI = CHROMOSOME DIRECTIO INDEX

EBI = CHS
EI = 1

CHL2 = CHL - 2
CHS1 = CHS - 1

THRESH = THRESH + 2
MLB = CHL2 - B
MSB = CHS1 - B
BCNS = B + CHS

C
DO 309 L=1,CHL2
EBI = EBI + 1

309 EBI = EBI + CHS

C
210 IF(ISEG(BI,BUF,THRESH,EBI).EQ.0) GOTO 308
SBIR(CBI) = GREC+1
EDGE(1) = L
S = CHS - (EBI - BI)
EDGE(2) = S

CALL SORT(N,EDGE,BUF(BI),THRESH,CHS)

IF(HE.EQ.0) GOTO 210
IF(OBN,EQ.TOBH) CALL LILIST(L,YMIN,YMIN,YMAX,YMAX)
IF(OBN,EQ.TOBH) CALL PDUMP(EDGE,EDGE(2+H),1)

IF(HE.EQ.355) GOTO 217

CALL SORTIN(EDGE,H,SINH)

IF(SINH.EQ.0) GOTO 215

IF(HE.EQ.512) GOTO 214

CALL TYPE(EDGE,OVERFLOW,THRESH,RAISED)
CALL TYPE("")
THRESH = THRESH + 4

GOTO 210

211 IF(ISW1.EQ.1) CALL TYPE("SORT")
IF(ISW4.EQ.1) CALL TIMER

CALL SORT(EDGE,H)

IF(ISW4.EQ.1) CALL TIMER

215 CONTINUE

IF(OBN.EQ.TOBH) CALL PDUMP(EDGE,EDGE(2+H),1)
IF(OBN.EQ.TOBH) CALL LILIST(L,S,CHL,CHS,OBN)

C
MAKE SURE THAT OBJECT CONTAINS AN INTERNAL POINT

INTPNT = 0
H2 = 2*N

DO 325 E=1,H2,4

IF(Y.LT.BI.OR.Y.GT.NLB) GOTO 230
X1 = MAXX(EDGE(E+1),BI)
X2 = MINX(EDGE(E+3),HBB)

IF(X1.GT.X2) GOTO 235

B0 = XAMS + X1

I = B0 - B

C
DO 228 LEFT=1,B0
IF(IVC(BUF,LEFT)).NE.0) GOTO 221

220 CONTINUE

C
221 RIGHT = B0 + X2 - X1 + B

C
DO 222 I=1,B1
IF(IVC(BUF,RIGHT)).NE.0) GOTO 223

222 RIGHT = RIGHT - 1

C
223  LEFT = LEFT + 0
    RIGHT = RIGHT - 0
    IF(TOBH.EQ.0) CALL ILIST(X1,X2,LEFT,RIGHT,Y)
    IF(LEFT.GT.RIGHT) GOTO 235

C

230  DO 235 I=LEFT,RIGHT
       TP = I - BCNS - 1
       BP = I + BCNS + 1
       DO 225 LL=1,BCNS,CNS
       IF(TOBH.EQ.0) CALL ILIST(TP,BP,LL,IV(BUF(TP+LL)),IV(BUF(BP-LL)))
       IF(IV(BUF(TP+LL)).EQ.0.OR.IV(BUF(BP-LL)).EQ.0) GOTO 230
       CONTINUE
C

C  THIS IS AN INTERNAL POINT
C
235  GOTO 230

C

C

C

236  MSEG = 0
    HP = 0
    IODN = 0
    ECI = 1

C

250  DO 250 E=1,N2,4
       Y = EDGE(E)
       X1 = EDGE(E+1)
       X2 = EDGE(E+2)
       NS = X2 - X1 + 1
       IF(NS.GT.0) GOTO 230
       CALL TYPE('ROB ~ NS LE 0')
       CALL TYPE('0')
       CALL ILIST(E,N2,X1,X2,MS)
       PAUSE
       GOTO 230

C

C

C

250  CONTINUE
    MS2 = (NS+1)/2
    BCI = ECI + 1
    ECI = ECI + MS2 + 3
    IF(ECI.LE.RECS2) GOTO 240

C

C

C

240  C(BCI) = CMLM1 + Y
    C(BCI+1) = CMLM1 + X1
    C(BCI+2) = NS
    CALL REMOVE(BUF(Y*CNS*X1)*C(BCI+3),STBL,MS,IODN,THRESH)
    HP = HP + MS
    NSEG = NSEG + 1

C

C

C

260  IF(HP.GE.25.AND.INTPNT.NS,#) GOTO 260
    CALL SSNCH(4,184)
    IF(ISWA.EQ.4) GOTO 210
    CALL GPRINT('OBJECT REJECTED')
    CALL ILIST(ORH,N,HP,YMIN,XMIN)
    CALL ILIST(YMAX-YMIN,XMAX-XMIN,L,S,NC)
    GOTO 210

C

C

C

260  WRITE OUT LAST LINE
    OREC = OREC + 1
    C(1) = NSEG
    CALL WRITE(SMVB,OREC,C)

C

C

C

280  NC = NC + 1
    NL = YMAX - YMIN + 1
    NS = XMAX - XMIN + 1
    IODN = IODN + BACK*HP/4
    THRESH = THRESH
    LIP(NC) = CMLM1 + L
SIP(NC) = CSM1 + S
IOD(NC) = IODH
AREA(NC) = NP
PERM(NC) = PERIM
CALL SWITCH(0, ISW0)
IF(ISW0. NE. 1) GOTO 200
C PRINT IF SW1 IS UP
CALL OUTCON(NC, TMSG(4), 2)
CALL OUTCON(PERM(NC), TMSG(10), 3)
CALL OUTCON(BACK+2, TMSG(17), 3)
CALL OUTCON(DATA+2, TMSG(27), 3)
CALL OUTCON(MAX+2, TMSG(31), 3)
CALL OUTCON(CML, TMSG(39), 4)
CALL OUTCON(CMS, TMSG(47), 4)
CALL OUTCON(CML+1, TMSG(53), 5)
CALL OUTCON(CMS1+S, TMSG(60), 5)
CALL OUTCON(CML, TMSG(65), 5)
CALL OUTCON(NS, TMSG(78), 5)
CALL OUTCON(IITHRE+2, TMSG(69), 3)
CALL OUTCON(NDTHRE, TMSG(96), 3)
CALL OUTCON(INBH, TMSG(34), 2)
CALL OUTCON(NP, TMSG(122), 9)
CALL OUTCON(IDM, TMSG(111), 9)
CALL OUTCON(IDM, NP/8, TMSG(116), 3)
CALL OUTCON(NC, TMSG(122), 2)
CALL QPRINT(TMSG(120))
C ENTER OBJECT IN CHROMOSOME DIRECTORY
200 SDIR(CD1+1) = CML + YMIN - 1
SDIR(CD1+2) = CMS + XMIN - 1
SDIR(CD1+3) = UREC
CD1 = CD1 + 4
IF(NC .EQ. MAXNOB) GOTO 510
GOTO 210
C
300 CONTINUE
500 III = III + 6
C
510 CONTINUE
SDIR(1) = CD1/4
NK=SDIR(1)
TEMP=PERM(1)
PERM(1)=64
C STORE NOB PARAMETER TEMPORARILY IN PERM(1).
CALL WPARAM(122, NK, NOBPH)
PERM1=TEMP
C WRITE NOB PARAMETERS
CALL MV(ITTAB, SDIR(241), 69)
CALL WRITE(SMVB, 1, SDIR)
CALL WRITE(SMVB, 2, LIP)
IF(ISW0. EQ. 1) CALL QPRINT('I')
CALL CLOSE(SMNBD)
CALL CLOSE(CHMR)
END.

C*********
C*********
C* NOB = CALMS VERSION
C*
C*********
C NOB NUMBERS THE RAW SPREAD WITH THE OBJECT NUMBERS GENERATED
C BY FOB
C
SUBROUTINE NOB(BWHT, FILP0X)
IMPLICIT INTEGER(A-Z)
COMMON/CL/ NK, LT, ST, FI
LOGICAL ORDER
BYTE FILP0X(3)
C I/O BUFFERS
BYTE IBUF2(12344), OBUF(12344)
C RANDOM AREAS
INTEGER SPAR(10)
BYTE CHAR(2)
C SAMPLE AND LINE TABLE FOR OBJECT NUMBERS

INTEGER ST(68), LT(68), TEMP(68)
DATA Y, X, D, REPL, 1, MASKH, 13/
C
C. INITIALIZE DATA SETS
C
CALL SSWICH(1, ISW1)
IF(ISW1.EQ.1) CALL TYPE('N0B')
C
C. READ PARAMETERS
C
CALL RPARAM(HK, NK, 122)
IF(1.EQ.0) CALL APHAASE(MASKH)
C
C. CLEAR THE GRAY SCALE UNLESS FINISHING UP A COUNT
CALL AFIL(E(IFBUF2, DUNIT, FILPEX, 2, 2))
CALL AFIL(E(IFBUF, 'PIC ' 'SN',' 'SCn'))
CALL OPEN(I IFBUF, 6144, 1, 0, 'SCN')
CALL OPEN(OBUF, 6144, 1, 1, 'NOB')
CALL GLABEL(IFBUF2, SPAR, 12)
ML = SPAR(1)
NS = SPAR(2)
Ni = NS/2
Nh = 1
CALL FLABEL(IFBUF, SPAR, IFBUF2(12+1))
C
C. SEE IF LT IS IN ORDER
ORDER = .TRUE.
IF(NK.EQ.0) GOTO 45
DO 40 H = 2, NK
IF(LH .LT. LT(NH+1)) GOTO 42
CONTINUE
40 CONTINUE
GOTO 45
42 ORDER = .FALSE.
45 CONTINUE.
C
C. RUN THROUGH EACH LINE OF THE PICTURE
C
DO 500 L = 1, NLI
CALL GDIT(IFBUF2, L, 12)
CALL PUT(IFBUF, L, 0)
CALL WIT(IFBUF2(12+1), OBUF(O+1), NW)
C
C. GENERATE NUMBER FOR EACH OBJECT AT APPROPRIATE TIME
C
IF(NK.EQ.0.OR.NI.GT.NK) GOTO 210
DO 200 N = 1, NK
LD = L - LT(N)
IF(LD.GT.4) GOTO 280
IF(LD.LT.4) GOTO 200
IF(LD.EQ.-4) GOTO 48
IF(LD) GOTO 210
GOTO 200
CONTINUE
200 CONTINUE
SD = ST(N) - 13
NCHAR = 2
IF(N.LT.10) GOTO 50
NCHAR = 1
SD = SD + 6
50 IF(SD.LT.1) SD = 1
IF(LD.EQ.-4) GOTO 75
IF(LD.EQ.4) GOTO 70
CALL OUTCON(N, CHARD(2), 2)
CALL TEXT(CHAR(3-NCHAR), NCHAR, LD+3, OBUF(0+SD), 1)
CALL TBL(0, OBUF(0+SD+6+NCHAR))
GOTO 260.
70 IF(LD) NI = NI + 1
75 CALL TBL(0, OBUF(O+SD), 6+NCHAR+1)
200 CONTINUE
210 IF(LD.EQ.0) GOTO 500
C NO DISPLAY IF FINISHING UP A COUNT
CALL DLINE(IFBUF(O+1), Y, NI, REPL, 0)
Y = Y + REPL + 1
500 CONTINUE
C
C. CLOSE DATA SETS
CALL CLOSE(BUF2)
CALL CLOSE(BUF)
CALL EXIT

C

4,122,518

C

131

C INT 1 - INTERACTION 1 TO CHECK NOB OUTPUT AND CORRECT FOR ERRORS
SUBROUTINE INT1(COUNT, ILPEK)

IMPLICIT INTEGER (A-Z)

COMMON/C1/NK, LBLA, STAB, FI

INTEGER PAR(10), SPAR(5), SST(513), MST(513), TTab(133), MPar(5)

INTEGER XTab(69), YTab(68), STAB(68), TEMP(68), RPar(100)

BYTE A(1234), BLACK(124), FILPEK(9), BUF(20), KEY(20), LIGHT(512)

BYTE MB(99), HF(21), MMSG(129), MK(8), WHITE(9), CHAR(2)

BYTE BELL(3)

EQUIVALENCE (PAR, BPar), (NE, Ttab(2)), (LPS, Ttab(3)), (Sps, Ttab(4))

EQUIVALENCE (NL, SPAR(1)), (HS, SPAR(2))

DATA BELL, '4', '7', '0'

DATA NTIME, '10000'

DATA MKEY, '20', 'KEY', 'N', 'C', 'J', 'B', 'T', 'A', 'F', 'S', 'I', 'N'

DATA '1', '2', '3', '4', '2', '1', '0', '0', '1', '0', '1', '0'

DATA PARAM(192), ROBP(192)

DATA PARAM(5), NK, 122, ROBP(192)

MFLAG*1

C ALWAYS SET MFLAG TO 1

C FLAG TO CALL NBO FOR A FINISH COUNT

CALL ITLA('32, LIGHT, 512')

CALL ZIA('WHITE, 4')

CALL NVL('DH**** BACK**** ,MB, 19')

CALL NVL('DH**** THRESHHOLD ****,MT, 20')

CALL NVL('NC**** ,MK, 7')

MB(19)=0

NI(29)=0

MK(0)=0

READY=0

CALL TYPE('OCHECK SPREAD', 0)

NE=0

CALL PARAM (NRP, RPAR, 192, ROBP)

CALL PARAM (NK, NK+122, ROBP)

IF(CIP. EQ. 0) FI=64

CALL DCOUNT(NK, FI)

C DISPLAY THE COUNT

CALL ITLA('32, GMSG, 129')

CALL RCR

C READ CURSOR ADJUSTMENTS

CALL AFIL('A, BUNI, FILPEN, 2, 2')

C OPEN SCAN DATA SET AND READ TTAB

CALL OPENX('A, 5144, 1, 2, '5CN')

CALL GLABELA('SPAR, IA')

CALL NVL('A(IA+72), GMSG, 64')

C SAVE LABEL FOR PATIENT REPORT

CALL GET('A, SPAR, IA+72, TTAB, 133')

25 CONTINUE

CALL PARAM (HP, PAR, 10)

30 IF(CIP. EQ. 0) GOTO 1000

35 IF(PCR(1). EQ. 'R') GOTO 80

35 IF(PCR(1). EQ. 'M') GOTO 85

35 DO 50 K=1, MKEY

35 IF(K, PCR(1), NE, IV, KEY(K))) GOTO 50

35 GOTO (90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 950, 910, 920, 930, 940)

35 END (1100, 1200, 1300, 1350, 1400), K

50 CONTINUE

C TYPE ('PARAMETER ERROR')

60 CALL TYPE('0')

GOTO 25

80 CONTINUE

C RS - RESCAN

C

CALL AFILE('1')

CALL CLOSE(A)

CALL EXIT
CALL APhase(10)
CALL CLOSEA()
CALL EXIT

CALL TYPE('4.122,518')
CALL TYPE('3 18')

CALL TYPE('C - CUT APART A TOUCH (POSITION CURSOR FIRST)')
CALL TYPE('M - JOIN TWO PIECES TOGETHER (USE CURSOR)')
CALL TYPE('T - CHANGE THRESHOLD (USE CURSOR)')
CALL TYPE('B - CHANGE BACKGROUND (USE CURSOR)')
CALL TYPE('3 - SET UP THE CURSOR TO CORRECT FOR DRIFT')
CALL TYPE('A - ADAPT THIS SPREAD')
CALL TYPE('RS - RESCAN')
CALL TYPE('E - FINISH THIS SPREAD (NO. KARYOTYPE)')
CALL TYPE('I - DISPLAY THE INITIAL (UN-NUMBERED) SPREAD')
CALL TYPE('M - DISPLAY THE NUMBERED SPREAD')
CALL TYPE('F - ERASE THE LAST CUT OR JOIN REQUEST')
CALL TYPE('U, D, L, OR R - MOVE THE CURSOR UP, DOWN, LEFT, OR RIGHT')
CALL TYPE('K - INTERACTIVE COUNT WITH CURSOR AND BELL')
CALL TYPE('N - ADD A MISSING NUMBER ONLY FOR COUNTS')
CALL TYPE('X - REMOVE AN EXTRA NUMBER ONLY FOR COUNTS')
CALL TYPE('Z - ZERO THE CHROMOSOME COUNT')
CALL TYPE('SM - SKIP MOB (USE OLD RPC)')
CALL TYPE('OT N.T - SET THRESH HOLD FOR OBJECT N TO T')
CALL TYPE('E - IF SPREAD IS OK, TYPE CARRIAGE RETURN')
CALL TYPE('E')
GOTO 25
J - JOIN TWO PIECES TOGETHER

CALL CURSOR(Y1, XI, SL, SS)
CALL TYPE('H', MOV-CURSOR TO END OF JOIN ')

DO 205 I = 1, 3

CALL DLINE(255, Y1-2+I, XI-1.3, 0, -1)

C MARK END-POINT

CALL PARAM(NP, PAR-10)
IF(NP, EQ, 9) GOTO 220
P=PAR(I)
IF(P, NE, 'E') GOTO 220
C ERASE THE MARK IF E WAS TYPED
DO 212 I = 1, 3

CALL DLINE(WHITE, Y1-2+I, XI-1.3, 0, 0)

GOTO 25

CALL UDLR(PAR)
GOTO 210

C CONTINUE

CALL CURSOR(Y2, X2, EL, ES)
CALL ADD(SL, SS, EL, ES, SST, NST, NL, NS)

DO 235 L = SL, EL

Y = 2*(L-I)
X = 2*(SST(L)-1)

CALL DLINE(127, Y-X, NST(L), 1, -1)

C ERASE ON THE GRAY SCALE TO INDICATE THE JOIN-LINE

CALL PARAM(NP, PAR, 10)
IF(NP, EQ, PAR(I)), NE, ERASE) GOTO 250

DO 255 L = SL, EL
S = SST(L)
C GET(A, L, TA)

CALL DLINE(A(I+A), 2*(L-I), 2*(S-1), NST(L), 1, 0)
C REUSE THE DATA

GOTO 25

PERU=1
DO 275 L = SL, EL
CALL GET(A, L, TA)
CALL PUT(A, L, TA)

CALL ITL(A, A(I+1), SST(L), NST(L))
C WRITE 127 IN THE DATA SET FOR THE JOIN-LINE

GOTO 30

C

B - CHANGE BACKGROUND

C

CONTINUE

CALL CURSOR(Y, X, L, S)
C GET(A, L, TA)
CALL OUTCONF(A(I+A), MB(7), 3)
IT = (L/LPS)*HC = 5/SPS
IF(S, SP, EQ, HC) IT = IT-1
CALL OUTCONF(ITTAB(68+IT), MB(16), 3)
C TYPE(HR, A)
BOLT = 0

C SET TO CHANGE BACK

IF(NP, EQ, 9) GOTO 25

CALL PARAM(NP, PAR, 10)
IF(NP, EQ, 0) GOTO 25
IF(NP, GT, 1) GOTO 35

IF(PAR(I), GT, 127), OR, PAR(I), LT, 0), GOTO 60
TH = (B, BOLT, 1)*PAR
C GET(A, SPARK(I)+L, 1A)
C CALL PUT(A, SPARK(I)+L, 1A)
C CALL NYR, ITTAB, A(I+1), 133)
PERU=1

GOTO 25

C

T - CHANGE THRESHOLD

C

CONTINUE

IF(NP, EQ, 9) GOTO 440
C GOTO 440 IF T WAS ALREADY REQUESTED
C DISPLAY THE SECTOR BOUNDARIES

DO 419 L = LPS, NL, LPS

CALL DLINE(WHITE, L*4, NST, 6, 8, 1, 0)
CALL ILLA(127, BLACK, 512)
DO 415 N=SPS.NS.SPS
415 BLACK(S)=S
DO 420 I=1,5
DO 420 BLHM(BLACK, ML+NL+I+1,2, NS, L, 8)
CONTINUE
CALL CURSOR(Y, X, L, S)
CALL GET(A, L, IA)
CALL OUTCON(T(6+I+1), MT(7), 3)
IT=(LT*LP)+MC+S/SPS+1
IF(S/SPS.EQ.MC) IT=IT-L
CALL OUTCON(T(4+IT), MT(10), 3)
CALL TYPE(MT(4))
BOR=4
C SET TO CHANGE THRESH
GOTO 320
C
C A - ABORT
C
500 CONTINUE
CALL ABRASE(8)
RUN=0
GOTO 1000
600 CONTINUE
C
C F - FINISH KEYWORD
C
IF(NFLAG.NE.0) GOTO 610
C IF NOB WILL BE CALLED GOTO 610
CALL ABRASE(HASKHP)
CALL CLOSE(A)
CALL AFILE(A, 'PIC', 5, 5)
CALL OPENA(A, SPS, 0, 2, 'HOB')
C SET TO ADD HK TO NOB OUTPUT LABEL
GOTO 620
610 CONTINUE
FI=8
CALL UPARAM(122, HK, NOBPH)
C SET NOB TO CALL MASK AND NOT PUT UP THE DISPLAY
CALL ABRASE(NOBBPH)
620 IF(NP.EQ.3) HK=PAR(3)
C USE OPERATOR SUPPLIED COUNT
CALL OUTCOM(HK, GHSG(68), 4)
C CONVERT HK FOR GRSC
C CONVERT HK FOR CRAC
C SET TO ADD NK IN THE LABEL
630 CALL CLOSE(A)
CALL AFILE(A, 'PBATA', 6, 6)
CALL OPENA(A, SPS, 0, 2, 'PRP')
CALL GLABEL(A, SPAR, IA)
NL=NL+1
CALL GLABEL(A, SPAR, A(IA+1))
CALL PUT(A, NL, IA)
CALL NVL(GHSG, A(IA+1), 128)
CALL PUT(A, NL, IA)
PAR(I)=0 'KG'
PARE=3 'BUNIT'
CALL NVL(FILPHE, PAR(4), 18)
CALL UPARAM(9, PAR, MASKPH3)
CALL CLOSE(A)
C
700 CALL SC
GOTO 25
C
C DISPLAY THE RAW SCAN WITHOUT OBJECT NUMBERS
C
800 CALL DCLEAR
FI=64
MT(7)="*"
C RESET THRESH INDICATOR

DO 820 L=1, ML
CALL GET(A,L,1A)

820 CALL BLINE(A(L+1),2*(L-1),0, NS,1, 0)
CALL DCOUNT(NK,FI)
GOTO 25

C
C DISPLAY THE NOB OUTPUT
C

950 F1=64
CALL UPARAM(122, NK, NOBPH)
CALL APHASE(NOBNPH)
CALL CLOSE(A)
CALL EXIT

GOTO 25

C INTERACTIVE COUNT
C

945 CALL CURSOR(Y0,X0,L,S)
DO 947 I=1, NTIME
CALL CURSOR(Y,X,L,S)
IF(Y .LT. Y0, OR X.NE.X0) GOTO 945

947 CONTINUE
C CURSOR HAS NOT MOVED IN A WHILE

IF(Y.LT.24. AND.X.GT.1000) GOTO 995
C IF UPPER RIGHT GOTO 995

GOTO 970

C WAIT FOR CURSOR TO BE MOVED

GOTO 945

970 CALL OUTCON(NK, MK(6), 2)
CALL TYPE(NK, 6)
IF(NK.LE.2) GOTO 25

C ORDER THE TABLE
DO 990 J=2, NK
IF(LTAB(J), GE., LTAB(J-1)) GOTO 990
LTH=LTAB(J)
DO 980 J=1, N
IF(LTH.EQ.LTAB(J)) GOTO 985

985 CONTINUE

988 HMOV=J-1
CALL MVU(LTAB(J), TEMP, HMOV)
LTAB(J)=LTH
CALL MVU(TEMP, LTAB(J+1), HMOV)
CALL MVU(TEMP, STAB(J+1), HMOV)
STAB(J)=STAB(N)
CALL MVU(TEMP, STAB(J+1), HMOV)

990 CONTINUE
GOTO 25

995 CONTINUE

C ERASE THE LAST SPOT
DO 997 J=1, 2
DO 997 I=1, 3
141

CALL DLINE(127, YTAB(NK) - 2 + I, XTAB(NK) - 1, 3, 0, -1)

IF(NK.GT.0) NK=HK-1
CALL DCOUNT(NK, FI)
GOTO 960

142

CALL CLOSE(A)
IF(VERIFY.GT.0) CALL APHASE(2)
CALL EXIT
C

1120 CONTINUE
C
C M.PARAMETER - ADD A NUMBER FOR THE MISSING OBJECT
C
NFLAG=1
IF(NK.GE.60) GOTO 60
CALL CURSOR(Y, X, L, S)
DO 1110 J=1, 2
DO 1110 I=1, .5
1110 CALL DLINE(WHITE, Y - 3 + I, X - 2, 5, 0, 0)
DO 1120 H=1, HK
IF(L.LT.LTAB(H)) GOTO 1130
1120 CONTINUE
LTAB(H+1)=L
STAB(H+1)=S
GOTO 1140

1130 NMOV=HK-N+1
CALL MVW(LTAB(H), TEMP, NMOV)
CALL MVW(TEMP, LTAB(H+1), NMOV)
LTAB(H)=L
CALL MVW(STAB(H), TEMP, NMOV)
CALL MVW(TEMP, STAB(H+1), NMOV)
STAB(H)=S
NK=HK+1
CALL DCOUNT(NK, FI)
GOTO 25

1200 CONTINUE
C
C X PARAMETER - REMOVE ONE OF THE NUMBERED OBJECTS
C
NFLAG=1
IF(NK.LT.1) GOTO 60
CALL CURSOR(Y, X, L, S)
MIN=20
C FInd THE OBJECT CLOSEST TO L-S AND NO MORE THAN 20 AWAY
DO 1220 H=1, HK
DIF=ABS(LTAB(H)-L)+ABS(STAB(H)-S)
IF(DIF.GT.MIN) GOTO 1220
MIN=DIF
MAIN=H
1220 CONTINUE
IF(MIN.EQ.20) GOTO 60
N=NHKM
Y=LTAB(H)+2-2
X=STAB(H)+2-2
IF(X.LT.7) X=7
DO 1240 J=1, 2
DO 1240 I=1, 9
1240 CALL DLINE(127, Y-5+1, X-7, 5, 0, -1)
C ERASE THE OLD NUMBER
NMOV=HK-N
IF(NMOV.EQ.0) GOTO 1250
CALL MVW(LTAB(H+1), LTAB(H), NMOV)
CALL MVW(STAB(H+1), STAB(H), NMOV)
1250 HK=HK-1
CALL DCOUNT(NK, FI)
GOTO 25
C
C 2 - ZERO CHROMOSOME COUNT
C
1300 HK=0
GOTO 25
C
C 0 - QUICK COUNT
C
1350 PAR(1)="TO"
PAR(2)=1
CALL UPARAM(1,PAR,2)
CALL APHASE(2)
CALL UPARAM(122,NK,MOBPH)
CALL CLOSE(A)
CALL EXIT
C QUICK COUNT OPTION
C OT - SET OBJECT THRESHOLD
C 1400 CALL MVG(PAR,RPAR(NRP+1),4)
       NRP=NRP+4
CALL UPARAM(NRP,RPAR,ROBPH)
REUN=1
GOTO 25
END

************
*           *
*           *
*           *
************

C MOB PROVIDES:
C A) THE ORIENTATION MECHANISMS: EACH INPUT OBJECT
C IS FIRST ENCLOSED IN A MINIMUM ENCLOSING RECTANGLE, THEN, THE OBJECT
C IS ROTATED INTO THE INDICATED ORIENTATION, MEASURED
C AND THEN WRITTEN OUT. MOB ACCUMULATES A CHROMOSOME
C DIRECTLY CONTAINING THE RESULTS OF THE MEASUREMENTS

SUBROUTINE MOB

COMMON/C1/MOD, IDIR
COMMON/C1/CHDIR, SMLBUF, LRGBUF, NS, NL, CURDNLH
COMMON/C1/ SST, EST
COMMON/C1/SPIDR, SPLTH, SPAREA
C SMALL BUFFER, FOR HOLDING ROTATED, UNMAGNIFIED OBJECT
BYTE SMLBUF(90,90)
C LARGE BUFFER, FOR HOLDING UNROTATED OBJECT
BYTE LRGBUF(90,90)
C BUFFER FOR I/O
BYTE BUF(2124)
C NUMBER OF INPUT OBJECTS
INTEGER NO8
C SYSTEM PARAMETERS
INTEGER SPAR(5)
EQUIVALENCE (SPAR(1),NL), (SPAR(2),NS)
C PARAMETER BEY FOR DECODING
INTEGER NK, KEY(11)
C ENDPOINT TRACKERS (8 FRACTIONAL BITS)
INTEGER XMIN, XMAX, YMIN, YMAX, XSMIN, XSMAX, YSMIN, YSMAX
C STARTING & ENDING SAMPLE TABLE
INTEGER=2 SST(98), EST(98)
C RANDOM AREAS
C RETURN CODE FROM ORIENTATION SUBROUTINE
INTEGER RCODE
C RADIANS/DEGREE
PEAL RPD
C OPERATOR SPECIFIED CENTROMERE POSITION
INTEGER OPCEN(60)
C OPERATOR FLIP FLAG
LOGICAL O PfLP(60)
C OPERATOR SPECIFIED ROTATION
INTEGER OPROT(60)
C CENTROMERE LOCATION METHOD
INTEGER CLMETH
C SKELETON FLAG
LOGICAL SKFLG
C SKELETON SWITCH
LOGICAL SWK
C SKELETON ROUTINE PARAMETERS
INTEGER S xmin, S delt
C PI & PI/2
REAL PI, PI HALF
C DELTA THETA
REAL DELTA
LOGICAL I4
INTEGER SELTH
INTEGER BLK, BS, SL
C CURRENT OUTPUT LINE #
INTEGER CURolin, BL, CHRENT
C PARAMETER AREA
INTEGER PAR(500)
EQUIVALENCEx PAR, LRGBUF
C
C INPUT DIRECTORY
C
INTEGER IDIR(4,60)
C FIRST BLOCK WRITTEN
INTEGER F BW
C MINIMUM LINE
INTEGER ML
C MINIMUM SAMPLE
INTEGER MS
C LAST BLOCK WRITTEN
INTEGER LBW
C
C CHROMOSOME DIRECTORY AND ITS FORMAT
C
INTEGER CHDIR(15,60)
C BLOCK NUMBER OF 1ST LINE OF OBJECT
INTEGER BLKNO
C NUMBER OF SAMPLES IN OBJECT
INTEGER DIRNS
C NUMBER OF LINES IN OBJECT
INTEGER DIRNL
C LENGTH OF CHROMOSOME
INTEGER LENGTH
C INTEGRATED OPTICAL DENSITY
INTEGER IOD
C CENTROMERIC INDICES
INTEGER CILO, CID, CIA
C CENTROMERE LINE NUMBER
INTEGER CENLI N
C PERIMETER
INTEGER PERIM
C AREA
INTEGER AREA
C PERIMETER SQUARED DIVIDED BY AREA
INTEGER PSQDA
C CENTROMERE LOCATION METHOD
INTEGER CENMETH
C MINIMUM AND MAXIMUM AREAS FOR CHROMOSOMES
INTEGER MAXOA, MINDO
C DEGREES/ ROTATION INCREMENT
C SAMPLE ACCUMULATORS
INTEGER DEHA(88), AREAA(88), ACC(176)
EQUIVALENCEx DEHA(1), ACC(1), AREAA(1), ACC(89)
C
C TOTAL AREA AND TOTAL DENSITY FOR NORMALIZING
C
INTEGER T AREA, TDEN
C
C ACCUMULATORS FOR CALCULATING CIA AND CID
C
INTEGER CIDEN, CAREA
C
C SPREAD MEASUREMENTS
C
REAL SPID, SPAREA, SPLTH
C
C SHORT ARM INDICATOR
C
INTEGER SHRTA
C
C RANDOM AREAS
C
INTEGER CEN, CENP1, S, PER
REAL DIPNC
BYTE EM(28)

C REEAMERGENT CHROMOSOME SELECTOR
LOGICAL SELFLG, SEL(60)
C
C DATA STATEMENTS
C
DATA DIPNC, 2.8125/
DATA MAXCHR, 60/
DATA MKET, 11, KEY, 'AR', 'RD', 'FL', 'CI', 'LW', 'SK', 'SP', 'SE',
= 'LD', 'MW', 'MD'/
DATA SKMIN, 15, SXDEL, 2/
DATA ID, 3, RPD, 0.17453, PL/7.14159, PIHALF, 1.5708/
DATA FW, 17, NL, 27, MS, 3, LBW, 4/
DATA DELTA, 8.5236/
DATA SKFLG, FALSE,/
DATA BLKNO, 1, BIRMS, 2, DIRNL, 3, LENGTH, 4, IOD, 5, CIL, 6, CID, 7/
DATA CIA, 8, CENL, 9, PERI, 10, AREA, 11, PSOBA, 12, SLMETH, 13/
DATA OPLIP, 60, FALSE, /, OPCEN, 60, 0, OPR0T, 60, 0/
DATA MAXNL, 88, MAXNS, 88, CHRENT, 13/
DATA SELFLG, FALSE, /, SEL, 60, FALSE, /
CALL MVLT, 'OBJ ** REJECTED **** ** * ', EM, 28)
C
C INITIALIZE SPREAD MEASUREMENTS
C
SPIOD = 0
SPLARE = 0
SPLTH = 0
CUROLN = 0
NC = 0
C
C INITIALIZE CHROMOSOME DIRECTORY
C
CALL ITIA(O, CHDIR, MAXCHR, CHRENT)
MINOA = 30
MAKOA = 2000
SWSK = FALSE.
C DEFAULT IS LW = NO SKELETON AND WIDTH TO LOCATE CENTROMERE
C
C RETRIEVE PARAMETERS AND DECODE
C
$5$ CALL RPARAM(HN, PAR, 500, 8)
$10$ IF (IP GT HN) GO TO 50
$18$ IF (IP) PAR(1P + 2)
$20$ 11 J = 1, HKEY
$12$ CONTINUE
C
C INVALID PARAMETER
C
$13$ CALL TYPE('*** PARAMETER ERROR')
$15$ GO TO 95
C
C PARAMETER AR - SET MINIMUM & MAXIMUM OBJECT AREA
C
$15$ MINOA = PAR(IP + 2)
$18$ MAXOA = PAR(IP + 3)
$19$ GO TO 10
C
C PARAMETER ROTA - ROTATE SPECIFIED OBJECT PRESCRIBED AMOUNT
C
$18$ IF (IP, LT, 1 OR IP, GT, MAXCHR) GO TO 13
$20$ K = PAR(IP + 3), DIPNC
$21$ OPR0T(IPN) = K +
$19$ GO TO 10
C
C PARAMETER FLIP - FLIP SPECIFIED OBJECT
C
$21$ IF (IP, LT, 1 OR IP, GT, MAXCHR) GO TO 13
C $e$ SKFLG=.TRUE.

C PARAMETER CL - SET CENTROMERIC INDEX
24 IF(IPH.LT.1 OR IPH.GT.MAXCHR) GO TO 13
OPFLIP(IPH)=.FALSE.
25 OPCH(IPH)=PAR(IP+3)
GO TO 19

C PARAMETER LW - LOCATE CENTROMERE BY WIDTH
C
32 CLNETH=0
SWSK = .FALSE.
IP=IP+2
GO TO 10

C PARAMETER SK - OUTPUT SKELETON PICTURES
C
40 SKFLG= TRUE.
IP=IP+2
GO TO 10

C PARAMETER SP - SET SKELETON PARAMETERS
C
45 SKHIN=PAR(IP+2)
SKDEL=PAR(IP+3)
IP=IP+4
GO TO 10

C PARAMETER SE - SELECT CHROMOSOMES FOR REMEASUREMENT
C NOTE - THIS PARAMETER ASSUMES THAT A CHROMOSOME DIRECTORY
C EXISTS FROM PREVIOUS MODDING ON RCR
C
47 K=PAR(IP+2)
IF(K.LT.1 OR K.GT.60) GO TO 13
SELFQ=TRUE.
IP=IP+3
DO 48 M=1,K
L=PAR(IP)
SEL(L)=TRUE.
48 CONTINUE
GO TO 10

C PARAMETER LD - LOCATE CENTROMERE BY DENSITY
C
51 CLNETH=0
SWSK = .FALSE.
IP=IP+2
GO TO 10

C PARAMETER LW - LOCATE CENTROMERE USING WIDTH AND MODEL(SKELETON)
C
54 CLNETH = 0
SWSK = .TRUE.
GO TO 10

C PARAMETER MD - LOCATE CENTROMERE USING DENSITY AND MODEL(SKELETON)
C
57 CLNETH = 0
SWSK = .TRUE.
GO TO 10

C OPEN DATA SETS
C
58 CALL AFIL(E(BUF.3,'UCR ','13,'13))
CALL AFIL(E(BUF.1,'RCR ','13,'13))
CALL OPEN(BUF.1024.1.0,'UCR')
CALL OPEN(BUF.1024.1.1,'RCR')
CALL GLABEL(BUF,SPAR,IB)
SPAR(1)=125
C PUT MAX NL IN LABEL SO COPY WILL NOT CAUSE F374
CALL PLABEL(BUF,SPAR,BUF(IB+1))
151

CALL GET(BUF,1,IB)
CALL MVL(BUF(IB+1),N00,MAXH=0+2)
CALL GET(BUF,2,IB)
DO 55 I=1,N00

55 CHDIR(Perim.)=TV2(BUF(IB+2X4+1)*1+2))
IF(.NOTSELFLOG ) GO TO 60

CALL GET(OBUF,1,IB)
CALL MVL(OBUF(IB+1),SPIOD,12)
CURLN=TV2(OBUF(IB+15))
NC=TV2(OBUF(IB+17))
CALL MVL(OBUF(IB+31),CHDIR(1,1),2*15*30)
CALL GET(OBUF,2,IB)
CALL MVL(OBUF(IB+31),CHDIR(1,31),2*15*30)

60 CONTINUE

C

CONTINUE

C

CALL ITIA(09,SST,MAXNS)
CALL ITIA(0,EST,MAXNS)
CALL ITIA(0,LRGBUF,99*99/2)

NL=0
NS=0

C

READ & PROCESS OBJECT BLOCKS

K=0
L=1
BL=0
IFD=IDIR(FBW,XX)
ILD=IDIR(LBW,XX)
MNS=IDIR(MS,XX)-1

C

READ EACH BLOCK ONE BY ONE

DO 145 BLK=IFBL,1IB
CALL GET(BUF,BLK,IB)
ASEG=TV2(BUF(IB+1))
I=IB+3

C

PROCESS EACH SEGMENT WITHIN THE BLOCK

DO 149 I=1,HSEG
IF(I.EQ.IV2(BUF(I))) GO TO 119
I=TV2(BUF(I))

119 MS=BL+1
IF(BL.GT.89) GO TO 490
IF(SST<MS) GT.89) SST(MS)=KB
IF(SST<BL) SST(BL)=KB
IF(SST<BL) SST(BL)=KB
IF(SST<BL) SST(BL)=KB

C

IGNORE POSSIBLE BUG IN FOB OUTPUT DATA

IF.SST(BL).EQ.89) SST(BL)=98

C

END

152
C TRANSFER SEGMENT INTO LRGBUF
C
CALL NWL(BUF(I)+6), LRGBUF(BS, BL), HSAMP
IF(NSAMP<(NSAMP/2)+2, HE. 0), HSAMP=NSAMP+1
I=I+NSAMP+6
CONTINUE
C
C REJECT OBJECT IF IT IS TOO SMALL OR TOO LARGE
C
IF(XX GT, MAXOA, OR, XX LT, MINOA), GO TO 490
C
C ORIENT THE OBJECT
C
CALL ORIOB(LRGBUF, SMLBUF, STI, CHJRS1, ML, DORBQ, RCODE)
ML=CHDIR(DIRNS, H)
IF(RCODE. HE. 0) GO TO 490
NC=NC+1
C
C BUILD ACCUMULATOR TABLES
C
CALL ACSRUB(SMLBUF, HL, HS, AREA, DEHA, TAREA, TDEN)
C
C BUILD AND ANALYZE THE SKELETON IF LOCATING
C
C THE CENTROMERE BY DENSITY
C
METH=1
C ASSUME OPERATOR SUPPLIED CENTROMERE (SET METH=1)
CEN=OPCEN(H)
IF(CEN. NE. 0) GOTO 170
C
C CHECK FOR OPERATOR SUPPLIED CENTROMERE
METH=0
C SET METH=0 FOR AUTOMATIC CENTROMERE
C
IFC. NOT. SWK) GO TO 150
CALL SKSUBL(SMLBUF, HL, HS, LRGBUF, CEN, BEND, SEmIN, SKBET, CLMETH)
C IF(CEN. NE. 0) GO TO 250
C
C THE PREVIOUS STATEMENTS WERE COMMENTED OUT TO MAKE MORE ROOM
C
C LOCATE CENTROMERE BY ALTERNATE METHOD
C
ASSIGN 165 TO LOOP
C
C LOOK FOR CENTROMERE IN MIDDLE HALF OF IMAGE
C
NS1=(NS-(NS/2)+1)/2
C
C FIND MINIMUM
C
CONTINUE
NS2=NS-NS1+1
MINA=ACC(NS1+CLMETH)
CEN=NS1
DO 160 I=NS1, NS2
IFC. ACC(I+CLMETH), GE, MINA) GO TO 160
MINA=ACC(I+CLMETH)
CEN=I
160 CONTINUE
GO TO LOOP
C
C CENTROMERE HAS BEEN TENTATIVELY LOCATED. IT IS CORRECT
C IF NOT ONE OF THE ENDPOINTS. IF IT IS AN ENDPOINT, LOOK
C FOR A CENTROMERE WITH AN INDEX BETWEEN .75 AND .87
C
IF(CEN. NE. NS1 AND CEN. NE. NS2) GO TO 230
ASSIGN 230 TO LOOP
NS1=(NS+4)/8+1
GO TO 155
C
C ADJUST CENTROMERE AS REQUIRED BY OPERATOR
C
IF(CHDIR(CENLIN, H), LT, 9) CEN=HS-CEN+1
C
C ADJUST FOR SHORT ARM ON RIGHT ORIGINALLY
MINA=ACC(CEN+CLMETH)
C
C LOOK ONE UP AND DOWN TO SEE IF THEY ARE SMALLER
IFC. ACC(CEN+1+CLMETH), LT, MINA) GOTO 180
IF (ACC(CEN+1+CLMETH).LT.MINA) CEN=CEN+1
GOTO 250

180 CEN=CEN+1
GOTO 250

C CHECK FOR DUPLICATE MINIMA AND TAKE AVERAGE IF SO
C
230 J=0
CEN1=CEN+1
DO 240 I=CEN1, NS2
IF (ACC(I+CLMETH).LT.MINA) GOTO 245
240 J=J+1
245 CEN=CEN+J/2
C

C MEASURE THE ARMS
C
250 CALL DPT(SMLBUF, HL, CEN, LD2)
CALL DPT(SMLBUF, HL, 1, LD1)
CALL DPT(SMLBUF, HL, NS, LD3)
A1TH=SQRT((CEN-.5)**2+FLOAT(LD2-LD1)**2)
A2LTH=SQRT((NS-CEN+.5)**2+(FLOAT(LD3-LD2)**2)
CHRLTH=A1TH+A2LTH
ICR=100.*A1TH/CHRLTH+.5
SHRTA=1.
IF (OPCEN(H).EQ.0) GOTO 260
C FIND SHORT ARM FOR OPERATOR SUPPLIED, CENTROMERE
IF (CH1(IF+CEN1,H).LT.0) SHRTA=-1
IF (OFFL(H)) SHRTA=-SHRTA
GOTO 265
260 IF (ICMR.GE.50) SHRTA=-1
265 IF (ICMR.LT.50) ICMR=100-ICMR
IF (ICMR.GT.99) ICMR=99
C
C CALCULATE CID AND CIA
C
CDEN=0
CAREA=9
DO 270 J=1,CEN
CDEN=CDEN+DENA(J)
270 CAREA=CAREA+AREA(J)
CAREA*=(CAREA+AREA(J))
IDR=((200.*CDEN)-((100.*DENA(CEN)))+FLOAT(TAREA))/2.*TDEN)
IF (IDR.LT.50) IDR=100-IDR
IF (IDR.GT.99) IDR=99
IAR=((200.*CAREA)-(100.*CAREA)+FLOAT(TAREA))/FLOAT(TAREA+TAREA)
IF (IAR.LT.50) IAR=100-IAR
IF (IAR.GT.99) IAR=99
C
C ADD MEASUREMENTS TO CHROMOSOME DIRECTORY
C
IF (SHRTA.EQ.-1) CEN=CEN-NS-1
C STORE NEGATIVE CEN TO FLAG SHORT ARM ON RIGHT
CHDIR(CLENTH,H)=CEN
CHDIR(LENGTH,H)=CHRLTH+.5
CHDIR(100,H)=TBN
CHDIR(CAREA,H)=TAREA
CHDIR(CIL,H)=ICMR
CHDIR(CIA,H)=IAR
CHDIR(CID,H)=IDR
FPF=CHDIR(PIER,H)
FPSDA=(FPF+FPF)/(TAREA*.5+FPF)
IF (FPSDA.GT.327.) FPSDA=327.
C CHECK FOR INTEGER OVERFLOW
CHDIR(FPSDA,H)=100.*FPSDA
CHDIR(CEMETH,H)=METH
C
C ADD INDIVIDUAL MEASUREMENTS TO SPREAD MEASUREMENTS
C
SPIOD=SPIOD+TBN
SPAREA=SPAREA+TAREA
SPLTH=SPLTH+CHRLTH
C
C STAND CHROMOSOME UP AND WRITE IT INTO ROTATED FILE
C
IP=NS
LPE=1924/HL
CHDIR(BLKN0,H)=CUROLN
CALL PUT(OBUF, CUROLN(I))
CUROLN=CUROLN+1
IF(SKFLG) GO TO 305
CALL CHROUT(SMLBUF, OBUF(IB+1), NS, NL, LPR, IP, SHRITA, 90)
GO TO 310
CALL CHROUT(LRGBUF, OBUF(IB+1), NS, NL, LPR, IP, SHRITA, 90)
CONTINUE
IF(IP.NE.0) GO TO 390
GO TO 500

C OBJECT BYPASSED
C
490 CONTINUE
CHDIR(BLKMD, N)=0
CALL OUTCON(N, EM(7), 2)
CALL OUTCON(KK, EM(22), 4)
CALL OUTCON(NL, EM(25), 2)
CALL OUTCON(NS, EM(28), 2)
CALL TYPE(EM, 20)
C
C END OF MAIN LOOP
C
500 CONTINUE
C
C UPDATE CHROMOSOME DIRECTORY ON DISC
C
CALL PUT(OBUF, 1, IB)
CALL ITL2(NOB, OBUF(IB+1))
CALL ITL2(CUROLN, OBUF(IB+15))
CALL ITL2(NC, OBUF(IB+17))
CALL MLV(SPIOD, OBUF(IB+3), 12)
CALL MLV(CHDIR(1, 1), OBUF(IB+31), 2*30+15)
CALL PUT(OBUF, 2, IB)
CALL MLV(CHDIR(1, 31), OBUF(IB+31), 2*30+15)
CALL CLOSE(OBUF)
CALL close(buf)
C
C MEASUREMENTS COMPLETED
C
999 CONTINUE
CALL EXIT
END

**************
C*       
C* CLASFY   COMPIL WITH /CO.25 **************
C*       
C*   **      
C
C CLASFY PROVIDES THE CLASSIFICATION MECHANISM FOR THE CLINICALA
C ALMS SYSTEM. ITS INPUT IS THE CHROMOSOME-DIRECTORY-PRODUCED-BY-MOB
C AND ITS OUTPUT CONSISTS OF THE RESULTS OF THE CLASSIFICATION PRO-
CEDURE WHICH IS UTILIZED BY KTYPE TO PRODUCE THE KARYOGRAM.
C
SUBROUTINE CLASFY
COMMON/C1, TIC, CIG, OFS, ICTB, ILTB,
COMMON/C1/MOB, SPIOD, SPLTH, SPAREA, CHDIR, IOBUF
COMMON/C1/NS, NFLAG, G1D, S1D4, ISR, OFS, SCLT, GCS
C
COMMON REGION
C
NUMBER OF CHROMOSOMES
C
INTEGER MOB
C
CHROMOSOME DIRECTORY
C
INTEGER CHDIR(15, 60)
C
FORMAT OF DIRECTORY
C
BLOCK NO. OF FIRST BLOCK OF AN OBJECT
INTEGER BLKNO
C NUMBER OF LINES IN OBJECT
INTEGER DIRNL
C NUMBER OF SAMPLES IN OBJECT
INTEGER DIRMS
C CHROMOSOME LENGTH
INTEGER LENGTH
C INTEGRATED OPTICAL DENSITY
INTEGER IOD
C CENTROMERIC INDEX BY LENGTH
INTEGER CIL
C CENTROMERIC INDEX BY DENSITY
INTEGER CID
C CENTROMERIC INDEX BY AREA
INTEGER CIA
C CENTROMERE LINE NUMBER IN ORIGINAL PICTURE
INTEGER CENLIN
C PERIMETER
INTEGER PERIM
C AREA OF CHROMOSOME
INTEGER AREA
C PERIM = 2 * AREA
INTEGER FSQDA
C OF DIRECTORY ENTRIES FOR EACH CHROMOSOME
INTEGER CHRENT
C
C IO BUFFER
C
BYTE I0BUF(4280)
C
C RANDOM AREAS
C
INTEGER SPARK(10)
BYTE ICIL(27)
INTEGER SSIZR, RECSIZ, OFSF, SEX
INTEGER SSFLG
INTEGER GNAM(26)
BYTE MSGS(29), MSGS(39), PMSG, PEM(22)
INTEGER KG(10), KG(19), JKG(1), JKG(18), JFAIL(10)
INTEGER CT(20, 51), ACT(20, 51), MAC(10), MAL(10)
REAL ST(11)
INTEGER IST(12), HG(11), OLDS(58), NEWS(58)
INTEGER SCS(99), SCLT(3), XTLB(46), ISR(6), CIG(11), IIG(11)
INTEGER CID(49), OFG(27), ICTG(68), ILTB(68)
INTEGER AC(10), AL(10), MING(10), MAXG(10), CTAB(6)
INTEGER OFS(91), PAR(900), KEY(19)
LOGICAL CID(19)
LOGICAL YINT(175), YID, BID
LOGICAL DEBUG, NUX
C
C DATA STATEMENTS
C
DATA HG/10/, NG1/11, NG2/12, NGT2/18, NG2/19, YID/'/'
DATA BID/'///ST/6*28.2m_25.-1.2*26.2m_27./'.
DATA CT/128=0, 19=0, 68=99, 10=0, 68=99, 16=0,
150.68.68.39=168, 60.68.68.99, 616=0, 68.68.68.99, 140=0, 62.68.68.99, 140=0, 62.68.68.68,
299 = 100, 85, 99, 52, 67, 62, 64, 56, 67, 291, 99, 100, 85, 99, 55, 67, 62, 84,
350.63.81.99=100, 61.99.54, 67, 62, 64, 56, 59.12=0, 80.99.54, 67,
462.81.50.57=128, 80.99.54, 67, 62, 64, 81, 50.56.16=0, 50.79.89.99,
534. 67.62.81.4=0, 80.99.54, 67, 62, 64, 81, 50.56.16=0, 50.79.89.99,
639.79.89.99=160, 50.79.89.99=160, 50.79.89.99=160, 50.79.89.99=160,
799.114=0, 70.73, 50.79, 89.99, 12=0, 50.59, 70.79, 53, 79, 114=0,
854.59.68.79.53, 79, 14=0, 58, 68.79.56.68, 14=0, 58, 59.68.79,
956.60.14=0, 59.59.68.79, 50.79.56.68, 12=0, 58, 68.79, 59.68.79, 56.68.14=0,
150.58.66.67.59.58.79.68=16=0, 58.65.56.67.59,
179.58.66.16=0, 58.56.67.59.68.79, 12=0, 58.65.56.67.59.
305.68.79.12=0, 58.56.67.58.68.79, 12=0, 58.65.56.67.59,
156.68.79.12=0, 58.56.67.61=0, 58.56.55.67.61=0, 58.57.55.67.61=0,
305.57.65.16=0, 50.57.56.67=16=0, 58.57.55.67.61=0, 58.57.
355.16=0, 50.57.55.67.61=0, 58.57.38=0 /
DATA MCT/180=0, 18=0, 70.99.10=0, 80.79.18=0, 70.99, 918=0, 70.99, 18=0, 70.99.99=16=0, 50.58.66.70.99, 14=0, 53.78, 58.66.2=0, 318=0, 57.99.16=0, 58.66.34=0, 58.66.2=0,
5 12=0, 58.66.34.58.33.68.58.34.33.68.58.33.58.66.2=0.
C
25 CALL OUTCON(IP, P=22), 3
CALL QPRINT(P=22)
GO TO 990
C
C PARAMETER CT - ADJUST THE DECISION TABLE
C
1 IL=PAR(IP+2)+1
IF(IL.LT.1 OR. IL.GT.51) GOTO 25
DO 1000 J=1, MGT2
CT(J)=CT(J)-A
IP=IP+1
1010 IF(IP.GT.NP) GOTO 70
DO 1020 J=1, MGC
IF(PAR(IP).EQ.CHAN(J)) GOTO 1030
1020 CONTINUE
GOTO 10
1030 JJ=J-J
CT(J,J)=PAR(IP+3)
CT(JJ-1, IL)=PAR(IP+2)
IP=IP+4
GOTO 10
C
C PARAMETER HH - NO OBJECT NUMBERS OR CENTROMERE FLAG
C
1100 HFLAG=1
IP=IP+2
GOTO 10
C
C PARAMETER SD - SET 0 OF OUTPUT SAMPLES
C
1200 NSD=PAR(IP+2)
NSDF=1
1203 IP=IP+3
GOTO 10
C
C PARAMETER SL - SET SLOT ID'S
C
1300 DO 1310 J=1, 36
1310 SID(J)=PAR(IP+1+J)
IP=IP+47
GOTO 10
C
C PARAMETER MA - MALE KARYOTYPE
C
1400 SEX=1
1410 IP=IP+2
GOTO 10
C
C PARAMETER FE - FEMALE KARYOTYPE
C
1500 SEX=2
GOTO 1410
C
C PARAMETER NX - NO X SEPARATION FROM C-GROUP
C
1510 NOX=TRUE.
CALL MVL(' B B ( C + X )
-16 E E
2 F F
G C SID4(13), 112)
1410
GOTO 1410
C
C PARAMETER DE - DEBUG OPTION ON
C
1520 DEBUG=TRUE.
GOTO 1410
C
C PARAMETER SZ - SET SLOT SIZE
C
1530 SLSD=PAR(IP+2)
GOTO 1293
C
C PARAMETER HV - MOVE CHROMOSOME AROUND FROM PREVIOUS KARYOTYPE
C
1540 IF(NSMOV.LT.50) NSMOV=NSMOV+1
OLDS(NSMOV)=PAR(IP+2)
... HEMO(NSMOV)=PAR(IP+3)
PARAMETER ST - SET SLOPE TABLE

DO 2980 IP=1,NCT2M1
2980 ST(IP)=PAR(IP+1)+PAR(IP+2)
       IP=IP+NCT2M1+2
       GOTO 10

PARAMETER M1 - SET MINIMUM # OF CHROMS. FOR EACH GROUP

3 DO 3300 IP=1,NG
3300 HING(IP)=PAR(IP+1)
       IP=IP+NG2
       GOTO 10

PARAMETER M2 - SET MAXIMUM # OF CHROMS. FOR EACH GROUP

4 DO 4800 IP=1,NG
4800 MAXG(IP)=PAR(IP+1)
       IP=IP+NG2
       GOTO 10

PARAMETER CS - SET CENTER SAMPLES FOR EACH SLOT

5 DO 5900 IP=1,NG
5900 SCS(IP)=PAR(IP+1)
       SCSL(IP)=1
       IP=IP+92
       GOTO 10

PARAMETER CL - SET CENTER LINE FOR EACH GROUP

6 DO 6900 IP=1,5
6900 SCL(IP)=PAR(IP+1)
       IP=IP+7
       GOTO 10

PARAMETER IR - SET INITIAL SLOT FOR EACH ROW

7 DO 7900 IP=1,6
7900 ISR(IP)=PAR(IP+1)
       IP=IP+92
       GOTO 10

PARAMETER IG - SET INITIAL SLOT FOR EACH GROUP

8 DO 8900 IP=1,NG2
8900 IST(IP)=PAR(IP+1)
       IP=IP+NGC+2
       GOTO 10

PARAMETER OS - MANUALLY INSERT CHROMOSOMES INTO SLOTS

9 OFSF=1
     NOFS=HP-1
     IF(NOFS.LE.48.AND.NOFS.GE.46).GOTO 8950
     C SEE IF 48 OR 46 OFS PARAMETERS WERE USED
     DO 8950 IN=1,NOFS
8950 OFS(XTBL(IP))=PAR(IP+1)
     GOTO 78
     C ALLOW FOR THE EXTRA SLOTS
     GOTO 78
     8950 CONTINUE
     DO 9800 IP=1,90
9800 OFS(IP)=PAR(IP+1)
       IP=IP+92
       GOTO 10

PARAMETER CO - ADJUST COEFFICIENTS

15 DO 71 IP=1,6
       CTW(IP)=PAR(IP+1)
DO 75 I=25
    IF(SLSIZ.EQ.8) SLSIZ=24
    IF(SLSIZ.GT.28) SLSIZ=28
    IF(SLSIZ.LT.12) SLSIZ=12
    PSEP=SLSIZ/3
C PAIR SEPARATION
    SCS(1)=PSEP+SLDIZ/2
    SCS(2)=SCS(1)+SLSIZ
    DO 752 I=3,17,2
        SCS(1)=SCS(1)+SCS(I-1)+SLSIZ+PSEP
     752
    DO 754 I=1,18
        SCS(1)=SCS(1)+SCS(I-1)+SLSIZ
     754
    DO 756 J=1,4
        SCS(18*J+1)=SCS(1)
     756
    CONTINUE
C C SETUP SLOT SIZE IF NOT INPUT
C IF(SLSIZ.EQ.0) SLSIZ=24
    IF(SLSIZ.GT.28) SLSIZ=28
    IF(SLSIZ.LT.12) SLSIZ=12
    PSEP=SLSIZ/3
C PAIR SEPARATION
    SCS(1)=PSEP+SLDIZ/2
    SCS(2)=SCS(1)+SLSIZ
    DO 752 I=3,17,2
        SCS(1)=SCS(1)+SCS(I-1)+SLSIZ+PSEP
     752
    DO 754 I=1,18
        SCS(1)=SCS(1)+SCS(I-1)+SLSIZ
     754
    DO 756 J=1,4
        SCS(18*J+1)=SCS(1)
     756
    CONTINUE
C C INITIALIZE CURRENT INDEX AND INITIAL INDEX FOR EACH GROUP
C C IG(1)=0
    IIG(1)=1
    DO 75 T=2,NG1
        C IG(I)=C IG(I-1)+25
     75
    IIG(I)=IIG(I-1)+25
C C SETUP SLOT SIZE IF NOT INPUT
C IF(SLSIZ.EQ.0) SLSIZ=24
    IF(SLSIZ.GT.28) SLSIZ=28
    IF(SLSIZ.LT.12) SLSIZ=12
    PSEP=SLSIZ/3
C PAIR SEPARATION
    SCS(1)=PSEP+SLDIZ/2
    SCS(2)=SCS(1)+SLSIZ
    DO 752 I=3,17,2
        SCS(1)=SCS(1)+SCS(I-1)+SLSIZ+PSEP
     752
    DO 754 I=1,18
        SCS(1)=SCS(1)+SCS(I-1)+SLSIZ
     754
    DO 756 J=1,4
        SCS(18*J+1)=SCS(1)
     756
    CONTINUE
C C CHECK SEX
C IF(SEX.GT.0) GOTO 76
    SEX=LSEX
    76 IF(SEX.NE.1) GOTO 77
C C MALE
    MAXG(5)=15
    MING(10)=MING(10)+1
DO 200 N=1,NOB
   IF(CHDIR(BKHO,N).EQ.0) GO TO 200
   ID=CHDIR(CIL,N)*CTAB(4)+CHDIR(CIL,N)*CTAB(5)+CHDIR(CIL,N)
   IF(ID.EQ.76/100)
   IL=FLOAT(CHDIR(LENC,N)) * CTAB(1) + FLOAT(CHDIR(10D,N))
   IF(IL.GT.51) IL=51
   CALL OUTCON(N,ICIL,N),,-2)
   CALL OUTCON(-ICIL,-N),,-2)
   CALL OUTCON(-1,ICIL(17),,-2)
   ICTBH(N)=IC
   ILTBH(N)=IL
   IF(IC.EQ.0 OR CHDIR(SQDA,N).LT.1348 OR CHDIR(AREA,N).GT.690)
   100 TO 22
   DO 88 J=1,NG2H,N,2
   IF(CIC.GE.(J+IL) AND IC.LE.(J+1+IL)) GOTO 100
   88 CONTINUE
   J=J+4;
   C GROUP IS THE FORBIDDEN ZONE
   GOTO 110
   85 CONTINUE
   CALL TYPE(' ALL SLOTS FULL')
   100 CONTINUE
   J=(J+1)/2
   110 CONTINUE
   CALL OUTCON(J,ICIL(26),,-2)
   IF(NOC(J).EQ.25) GOTO 85
   NOG(J)=NOG(J)+1
   CIC(J)=CIC(J)+1
   0FG(CIC(J))=N
   900 CONTINUE
   C IF ANY GROUP IS HEAVY OR LIGHT, TRY TO MOVE CHROMOSOMES AROUND
   C 210 NFAIL=0
   220 CONTINUE
   KX=0
   NMOV=0
   DO 300 J=2,NG
   IF(NOG(J).GE.MAX(J)) GOTO 300
   K=J-1
   NMOV=NMOV+1
   DMN=DMN-MIN
   C GROUP J IS LIGHT, SEE IF A PRECEDING GROUP IS HEAVY
   DO 2500 K=1,K2
   IF(NOG(K).LE.MING(K)) GOTO 2500
   C GROUP K IS HEAVY
   IF(NFAIL.EQ.0) GOTO 230
   DO 225 N=1,NFAIL
   IF(K.EQ.NFAIL(J) AND J.EQ.JFAIL(N)) GOTO 235
   C THIS J AND K HAS ALREADY FAILED
   225 CONTINUE
   230 KX=K
   JL=J
   C SAVE HEAVY AND LIGHT GROUP NUMBERS
   235 CONTINUE
   10=KX
   12=CIC(K)
DO 2400 I=11,12

M=OGF(I)

IC=ICBCH(N)

IL=ILTBC(N)

JJ=J+J

IF IC.LT.CIC(JJ-1,IL) OR IC.GT.CIC(JJ,IL) GOTO 2400

C OBJECT IS IN GROUP J. MOST LIKELY IF CLOSE TO CENTER OF J

C AND FAR FROM CENTER OF K

D=(CIC-AC(J))**2+(IL-AL(J))**2+2..(IL-AL(K))**2+2..1)

IF(D.GE.DIN) GOTO 2460

C THIS IS THE MOST LIKELY CANDIDATE SO FAR

KSAV=K

ISAV=1

NCAN=1

DIN=D

2460 CONTINUE

2500 CONTINUE

IF(NCAN.EQ.0) GOTO 3000

HMV=HMV+1

C MOVE OBJECT N FROM GROUP KSAV TO GROUP J

CALL OUTFCON(KSAV,MSG(22),3)

CALL OUTFCON(J,MSG(29),3)

IF(DEBUG) CALL OPRINT(MSG(29))

NOC(J)=NOC(J)+1

CIG(J)=CIG(J)+1

OGC(J)=OGC(J)

CIG(KSAV)=CIG(KSAV)-1

NOC(KSAV)=NOC(KSAV)-1

I2=CIG(KSAV)

DO 2600 I=ISAV,12

2600 OCF(I)=OGF(I)+1

C CLOSE UP THE REMAINING OBJECTS IN THE GROUP

3000 CONTINUE

IF(HMOV.GT.0) GOTO 220

C IF SOMETHING WAS MOVED, SEE IF MORE MOVES ARE POSSIBLE

IF(KH.EQ.0) GOTO 3100

C KH IS HEAVY AND JL IS LIGHT BUT NO SIMPLE MOVE WAS POSSIBLE

C SEE IF THERE IS A COMPLEX (MULTI-LEVEL) MOVE THAT CAN BE MADE

L=1

JG(J)=JL

3020 KG(J)=KH

3020 CALL CFO(KG(J),JG(J),IX,AC,AL,CT)

C CHECK FOR A CHROMOSOME IN THE KG-JG OVERLAP AREA

IF(IK.EQ.0) GOTO 3040

IK(I)=IX

3040 CONTINUE

C SAVE INDEX NUMBER

IF(KG(J).EQ.KH) GOTO 3060

C WE NEED TO GO UP ONE LEVEL

L=L+1

JG(J)=KG(J)

GOTO 3010

3040 KG(L)=KG(L-1)

C LOOK AT THE NEXT GROUP ON THIS LEVEL

IF(KG(J).LT.JG(J)) GOTO 3020

L=L-1

C GO DOWN ONE LEVEL

IF(JG(J).GT.0) GOTO 3040

C FAILED TO FIND A GOOD MOVE

NFAIL=NFAIL+1

IF(NFAIL.GT.10) GOTO 3100

NFAIL(NFAIL)=KH

JFAIL(JFAIL)=JL

CALL OUTFCON(KH,FMSG(22),3)

CALL OUTFCON(JL,FMSG(29),3)

IF(DEBUG) CALL OPRINT(FMSG(35))

GOTO 220

3060 CONTINUE

C MAKE THE MOVES THAT HAVE BEEN FOUND

DO 3080 M=1.I

ISAV=1GM

JGM=JGM

NOC(JGM)=NOC(JGM)+1

CIG(JGM)=CIG(JGM)+1

3080 CONTINUE
C LOOK FOR MORE MOVES

C DONE MOVING CHROMOSOMES BETWEEN GROUPS. NOW ORDER THE CHROMOSOMES WITHIN EACH GROUP

C

3100 CONTINUE
DO 240 J=1,NC1
IF(NOC(J) .EQ. 0) GOTO 300
I1=IC(J)
I2=CIG(J)
DO 240 I=11,12
H=OFS(I)
I=CIG(H1)
IL=IL(H1)
YINT(I)=IC-ST(J)*IL
IF(ST(J).LT.0) YINT(I)=YINT(I)
240 CONTINUE
C STORE Y INTERCEPT
I1=IST(Y1)

242 CONTINUE
L2=IST(J)*HOC(J)-1
IF(L2.LT.IST(J)+1) GOTO 246
HOC(J)=HOC(J)-1
IF(HOC(H2).LT.IST(H2)-IST(H1)) GOTO 244
CALL SSWCH(I.ISW1)
IF(ISW1 .EQ. 1) CALL TYPE( ' ALL SLOTS FULL' )
GOTO 242

244 CONTINUE
L1=CIG(H2)-HOC(H2)+1
HOC(H1)=HOC(H1)+1
OFS(CIG(H1))=OFS(I2)
I2=12-1
GOTO 242

246 CONTINUE
DO 260 L=L1,L2
MINY=9999
DO 258 I=11,12
IF(YINT(I).GE.MINY) GOTO 250
MINY=YINT(I)
MINI=I
250 CONTINUE
OFS(L)=OFS(MINI)
YINT(MINI)=MINY

260 CONTINUE
C STORE THE GROUP IN ORDER OF YINT

300 CONTINUE

310 CONTINUE
C

C CHECK FOR NO X SEPARATION FROM GROUP DESIRED
C FIND THE X OR X-X IN GROUP C

C

I1=IST(5)
I2=IST(15)
I3=11+2
I4=12-2
IF(HOC(5).EQ.16) GOTO 330
IF(HOC(5).NE.15) GOTO 350
C C HAS 15. THIRD LARGEST IS X
OFS(12)=OFS(13)
DO 320 I=13,14
320 OFS(I)=OFS(I+1)
OFS(12-1)=0
GO TO 350 CONTINUE

C IF X IS 16, 17 OR X IS X
ITEM = OFS(I)
ITEM = OFS(I+1)
DO 344 I = 13, 14

340 OFS(1) = OFS(I+2)
OFS(I+2-1) = ITEM
OFS(I+2) = ITEM

350 CONTINUE
IF (FNOG(NC), NE, .5 OR SEx, EQ, .2) GOTO 400

C IF 5 IN GROUP C, FIND THE Y CHROMOSOME
I = IST(NC)
I = I+1
IF (OFS(I) .EQ. 0) GOTO 403
MAX = 1
DO 360 N = 11, 12
N = OFS(I)
M = CHDIR(ARRAY(N), (CHDIR(DIRNL, N)*CHDIR(DIRNS, N)))
IF (M .LE. MAX) GOTO 360
MAX = M
M = MAX
MAX = N

360 CONTINUE
I = I-1
DO 370 J = 1, MAX+1

370 OFS(I) = OFS(I+1)
OFS(I+1) = 0
OFS(I+2) = MAX

400 CONTINUE
I = HC(NC) DO 402
HSMOV = NSHVN = 0
NSD = OFS(0)
IF (NSD = 0) GOTO 401
C DELETE OBJECT IF NSD = 0 IF NEWSN = 0
OFS(NSD) = 0
GOTO 402

401 CONTINUE
NSHVN = OFS(NEWSN)
OFS(NEWSN) = OFS(NSD)
OFS(NSD) = NSHVN

402 CONTINUE

403 CONTINUE
C ADJUST TABLE FOR OBJECTS WIDER THAN 24 SAMPLES
DO 410 J = 1, NJL
I = IST(J)
I = I+1
J = J+1
DO 420 I = 1, 12
GSL(N) = GSL(N) + IDEL

C ADD ON ANY PREVIOUS DELTA
I = OFS(I)
IF (I .EQ. 0) GOTO 404
HSTM24 = CHDIR(DIRNS, I)*CHDIR(DIRNL, I)
IF (HSTM24 .EQ. 0) GOTO 404
GSM(N) = GSM(N) + HSTM24

C ADJUST DELTA FOR THIS OBJECT
IDEL = IDEL + HSTM24

404 IF (SCL(I) .GT. 586) SCL(I) = 586

405 CONTINUE

409 CONTINUE
IF (FNOG Eq. 1 AND NS0, EQ, 0) GOTO 425
C IF NS0 OF 0 WAS SPECIFIED, DO NOT CHANGE IT
C NOW FIND NSD
DO 420 J = 2, 6
I = I+1
DO 415 I = 1, 10
I = OFS(10R(J))
IF (I .EQ. 0) GOTO 405

C THIS IS THE LAST NON-VACANT SLOT ON THE ROW
I = SCL(I)+SCL(R(J)) SLSIZ = 2*FPER
IF (I .EQ. 0) NSD = TEMP
C ADJUST NSD IF REQUIRED
GOTO 429

415 CONTINUE
179
CEMETH(1)=IV(1BUF(25+39*1))
IF(NEB.EQ.1) GO TO 40
CONTINUE
CALL READ(INVB,18,1BUF(1))
DO 20 I=1,10
FBN(I+30)=IV(1BUF(I+20*1))
MLY(I+30)=IV(1BUF(I+20*1))
NST(I+30)=IV(1BUF(I+30*1))
CELIN(I+30)=ABS(IV(1BUF(I+30*1)))-ABS(IY(BUF(I+30*1)))
IF(NOB.EQ.1+30) GO TO 40
CONTINUE
C
C READ IN CLASSIFICATION TABLES
C
DO 49
CONTINUE
CALL READ(INVB,18,1BUF(1))
CALL MVL(1BUF(1),NSO,2+294)
IF(NSO.GT.512) GO TO 900
SPAR(I)=S12
SPAR(2)=NSO
SPAR(3)=S12
CALL PLABEL(obuf,spar,label)
C
C MAIN LOOP:
C BUILD EACH ROW OF THE OUTPUT KARYOGRAM
C
DO 500 I=1,5
ISRI=ISR(I)
IFS(I,LT,5) GOTO 430
C SEE IF THERE ARE ANY OBJECTS ON ROW 5
DO 420 J=1,10
IF(OFS(ISRI+J-1).NE.0) GOTO 430
CONTINUE
GOTO 500
420
C NO OBJECTS ON ROW 5
CONTINUE
R(2)=GID(I+(1-I)*2)
R(3)=GID(2+(I-I)*2)
DO 450 J=1,10
INF=OFS(J)
IF(JJ.GE.ISRI(1)) INF=0
CALL QUIICH(INF,R(5+(J-1)*2),-2)
IF(INP.EQ.0) CALL ITL(32,R(5+(J-1)*2))
CONTINUE
CALL SSWITCH(1.IBIT)
IF(IBIT.NE.1) GO TO 475
CALL TYPE(R,78)
C
C INVOKE THE ROW BUILDING ROUTINE
C
CONTINUE
CALL KROW(ISRI(1)+1-ISRI(OFSC(ISRI).SCLT(I)).SCLC(ISRI).NSO.ISID(ISRI).NFLAG)
NLO=NLO+2*SCLT(I)
CONTINUE
C
C CLOSE UP DATA SETS AND RETURN
C
SPAR(1)=NLO
CALL PLABEL(obuf,spar,label)
CALL CLOSE(INVB)
CALL CLOSE(obuf)
CALL EXIT
C
C ERROR
C
990 CALL TYPE(' *** NSO TOO BIG')
CALL TYPE(' **')
CALL EXIT
END
SUBROUTINE KROW(NSL, OFS, SCL, SCS, NSO, SID, NFLAG)

C KROW CONSTRUCTS A ROW OF THE OUTPUT KARYOGRAM. IT IS INVOKED BY
C KTYPE.

C COMMON/CL, IMVB, IBUF, OBuf, MLT, NST, FBW, CEHIN, CEMETH
C INTEGER CEHIN(60), CEMETH(60)
C INTEGEX VY, REPL
C BYTE CHAR(4)
C INTEGER SCL, S13(98)
C INTEGER IDT(60)
C INTEGER OFS(91), SCS(99), MLT(98), NST(60), FBN(60)
C INTEGER NSL, NSO, SCL2
C BYTE OBzf(3120), IBUF(20488), L00, LFF
C INTEGER RECSIZ, MAXNSL
C DATA L00/0/, LFF/127/, RECSIZ/1824/, MAXNSL/20/
C DATA X/0/, Y/0/, REPL/1/

C RETURN IF NSO=0
C IF(NS0.EQ.0) GO TO 910
C IF(NSL.GT.MAXNSL) GO TO 900

C SET UP BUFFER INDICES
C
C IDT(1)=1
C DO 50 I=2, NSL
C 50 IDT(I)=IDT(I-1)+RECSIZ
C NSO4=NS0+1)/2
C SCL2=SCL+2

C MAIN ROW LOOP - PROCESSES EACH LINE WITHIN THE ROW
C
C DO 200 L=1, SCL2
C CALL PUT(0BUF, 0, 10)
C CALL ZIA(0BUF(10+1), NSO4)

C SLOT LOOP - PROCESSES EACH SLOT WITHIN THE CURRENT LINE
C
C DO 150 I=1, NSL
C NI=OFS(I)
C IF(NSL.GE.0) GO TO 80
C IF(NI.LE.0) GO TO 80
C GO TO 130

C CHECK FOR FIRST SLOT OF A PAIR EMPTY, BUT SECOND SLOT FULL
C IF(NL.GT.0) GO TO 100

C GOTO 150 IF BOTH SLOTS ARE EMPTY
C NL2=0
C GO TO 130

C DECIDE IF OBJECT APPEARS ON THIS LINE
C
C CONTINUE
C NL2+NLTH(N)
C NL2=NL2/2
C IF(L.LT.SCL-NL2 .OR. L.GE.SCL+NL2) GO TO 120
C NL2=SCL-MLT

C CALCULATE INITIAL SAMPLE FOR OBJECT ON OUTPUT LINE
C
C NSIN=NST(N)
C MS2=NS2/2
C IF(SCS(I)+MS2.GT.511) SCS(I)=511-MS2
C IS=SCS(I)-MS2

C GET FIRST BLOCK OF THE OBJECT
C
C IB=FBN(N)
C IF(IB.EQ.0) GO TO 120
C BYPASS OBJECT IF NOT IN DIRECTORY.
LPBN=RECSIZ/NSTN
MLD=MOD(LD,LPBN).
IF(MDL.GT.0) GO TO 98
CALL READ(MYB,IA+LD/LPBN,IA,IBUF(IDT(I)))
98 J=IDT(I)+MLD=NSTN
C TRANSFER OBJECT SEGMENT INTO THE OUTPUT BUFFER
C CALL NVL(IBUF(JI),OBUF(IQ+IS),NSTN)
IS=IS=NSTN
C
C SEE IF THIS IS THE CENTROMERE LINE.
C IF(NFLAG.NE.0) GO TO 158
IF(LQ.QU=CENLIN(N)-1) GO TO 158
C
C CENTROMERE LINE, FIND WHERE TO PUT THE CENTROMERE ARROWS
C IT=IS=NSTN
ITMAX=IT-1
GO 110 IT=IT,ITMAX
IF(OBUF(IQ+IU).EQ.100) GO TO 112
110 CONTINUE
GO TO 117
112 CONTINUE
IT=IS
115 IT=IT-1
IF(OBUF(IQ+IT).EQ.100) GO TO 119
117 CONTINUE
GO 110 K=3.5
OBUF(IQ+IU+K)=LFF
118 OBUF(IQ+IT+K)=LFF
GO TO 150
C
C CHECK TO SEE IF LINE CONTAINS OBJECT
C 120 CONTINUE
IF(NFLAG.NE.0.OR.L.LT.SCL-ML2-9.OR.L.GT.SCL-ML2-3) GO TO 130
C
C LINE CONTAINS OBJECT 0, SO PUT NUMBER INTO OUTPUT BUFFER
C LO=L-SCL-ML2+9
IS=SCS(I)-11
IF(IS.GT.500) IS=500
IF(N.LT.10) IS=IS-3
NNH=1
IF(INEMTH(N).EQ.1) NNH=N
C FLAG OPERATOR CORRECTED CENTROMERES
C CALL OUTCOM(NH,CHAR<3>,3)
C CALL TEXT(CHPR<3>,LO,OBUF(IQ+IS),1)
GO TO 150
C
C CHECK TO SEE IF LINE CONTAINS GROUP ID
C 130 CONTINUE
IF(I.EQ.(1/2)*2) GO TO 150
IF(N.L.EQ.0) GO TO 140
ML3=NL(N(N)-1)/2
140 IF(NL3.GT.NL2) NL2=NL3
CONTINUE
IF(L.LT.SCL+NL2+5.OR.L.GT.SCL+NL2+10) GO TO 150
C
C LINE CONTAINS GROUP ID, SO OUTPUT A LINE OF THE ID TO THE OUTPUTBUF
C LO=(L-SCL-ML2-5)/2
IS=((SCS(I)+SCS(I+1))/4)*2-23
150 IF(SCS(I).GT.454) IS=454
CALL TEXT(IDH(N)+HL.PO.OBUF(IQ+IS)+2)
CONTINUE
IF(N.L.T.1824) CALL SLINE(OBUF(IQ+1),Y.X.HSO,REPL-9)
RETURN
C
C BUFFER TOO SMALL TO HANDLE THE ROW

900 CONTINUE
CALL TYPE('**.BUFFER TOO SMALL FOR ROW')
CALL TYPE('Q')
RETURN
C
C NSO=0
C
910 CONTINUE
CALL TYPE('** NSO=0')
CALL TYPE('Q')
RETURN
END

SUBROUTINE INT2

C********
C
C* INT2 *
C* *
C********
C
C THIS ROUTINE IS THE SECOND OPERATOR INTERACTION PHASE. IT
C ALLOWS THE OPERATOR TO EITHER APPROVE THE KARYOTYPE WHICH IS
C BEING DISPLAYED ON THE GRAY SCALE DEVICE, FOR HARD COPY OUTPUT
C OR ELSE MAKE CHANGES VIA MOB AND/OR CLASSFY/KTYPE PARAMETERS.
C
C COMMON REGION
COMMON/C1/NSO,NFLAG,CHR,SID,ISR,OFF,SLCL,SCS
COMMON/C1/CHRDIR,MOB,SPINS,SPLEN,SPAREA,EROLE,REC
C
C I/O BUFFER
BYTE I0BUF(2124)
C MOB PARAMETERS AND CURRENT PARAMETER HIGH WATER MARK
INTEGER MOBPAR(590),MBHBPAR(4)
C CLASSFY/KTYPE PARAMETERS AND CURRENT PARAM. HIGH WATER MARK
INTEGER KTYPAR(590),K
C INT2 LOCAL PARAMETER BUFFER AND POINTER
INTEGER PAR(180),IP
C RE-MOB CHROMOSOME SELECTOR
LOGICAL SEL(60)
C CLASSIFICATION TABLES
C
C CHROMOSOME DIRECTORY AND ITS FORMAT
C
INTEGER CHRDIR(15,60)
C
C BLOCK #
INTEGER BLKRD
C NUMBER OF LINES IN OBJECT
INTEGER DIRNL
C NUMBER OF SAMPLES IN OBJECT
INTEGER DIRNS
C
LENGTH
INTEGER LENGTH
C INTEGRATED OPTICAL DENSITY
INTEGER IOO
C CENTROMERIC INDICES
INTEGER CIL,CIO,CIA
C CENTROMERE LINE #
INTEGER CEILIN
C PERIHERE
INTEGER PERIM
C AREW
INTEGER AREA
C PERIM SQUARED DIVIDED BY AREA
INTEGER PSQDA
C CENTROMERE LOCATION METHOD
INTEGER CENETH
C PHASE NUMBERS FOR MOB AND FOR CLASSFY/KTYPE
INTEGER MOBPHA,KTYPHA,FOBPHA,BANDPH,FOURPH
C RE-RUN FLAGS
LOGICAL MOBFLG,KYFLG,ALLFLG,DSFLG
C RANDOM AREAS
INTEGER SLCL,SLCS,SLID,S,Y,X,SL1,SL2
187
INTGCR SPAl1(18), SST(513), NST(513), S1, S2

188
INTEGER KEY(37)
BYTE LABLE(73), BLACK(72), PARBUF(90), MOB(560)

C CLASSIFICATION TABLES
C
INTEGER GID(18), SID(90), ISR(6), OFS(91), SCLT(5), SCS(90)
C
C DATA STATEMENTS
C
DATA BLKNO/I, DIRNL/2, DIRNS/3, LENGTH/4, 10B/5, CIL/6, CID/7
1, CL/8, CENLIN/9, PERIM/10, AREA/11, PSQDA/12, CENETH/13
DATA MAXPAR/50, FOSPHA/5, MOBPAR/5, KTYPHA/9, BANEPH/14
DATA DEGPR/57, 2958, /FOURPH/15
DATA LABEL/73, ' /, BLACK/72, 127
C
C INITIALIZE CHROMOSOME DATA SET
CALL TYPE('CHECK KARYOTYPE', 0)
CALL RCA
C READ CURSOR ADJUSTMENTS
CALL AFILE(IOBUF,1, 'RCR', '13, '13)
CALL OPEN(IOBUF,1024, 1, 'RCR')
CALL LABLE(IOBUF, SPAR, IB)
CALL NVL(IOBUF(IB+1+72), LABEL(2), 72)
C
C READ IN DIRECTORY AND CLASSIFICATION TABLES
C
CALL GET(IOBUF,1, IB)
CALL NVL(IOBUF(IB+1), MOB.10)
CALL NVL(IOBUF(IB+3), CHDIR(1,1), MAX=30)
CALL GET(IOBUF, 2, IB)
CALL NVL(IOBUF(IB+3), CHDIR(1,31), MAX=30)
CALL GET(IOBUF, 3, IB)
CALL NVL(IOBUF(IB+1), NSF, 2, 294)
C
C READ PREVIOUS PARAMETERS
C
CALL RPAM(M, MOBPAR, MAXPAR, MOBPHA)
CALL RPAM(K, KYTPAR, MAXPAR, KTYPHA)
19. L.T. 92) GOTO 65
K=K-92
OFSFLG=. TRUE.
C DO NOT RECLASSIFY IF OFS PAR4METERS ARE PRESENT
C ERASE OLD OFS PARAMETERS, IF PRESENT
C
65 IF(M.EQ.0) GOTO 100
DO 70 I=1,M
IF(MOBPAR(I).EQ. 'SE') GOTO 75
70 CONTINUE
GOTO 190
C REMOVE OLD SE PARAMETERS
75 N=I-1
C REQUEST OPERATOR INTERACTION
C
100 CONTINUE
110 CALL PARAM(NP, PAR, 100, PARBUF)
IF(NP.EQ.0) GOTO 500
NP=1
C
C DECODE PARAMETERS
C
DO 200 J=1,NKEY
IF(PAR(1P).EQ. KEY(1J)) GOTO (1808, 1859, 900, 910, 950, 1250, 1300, 1359
1, 1450, 1500, 1550, 1600, 1650, 1700, 1800, 1850, 1900, 1950, 2000, 2050, 2125, 2150
2, 2200, 2250, 2300, 2350, 1150, 1175, 2100, 1950, 1900, 1850, 2300, 3
3, 2350, 2400, 2450, 2500).J
200 CONTINUE
CALL TYPE('PARAMETER ERROR')
CALL TYPE('O')
GOTO 100
189

```
250 CALL TYPE(' CURSOR ERROR')
CALL TYPE(' @')
GOTO 100

CALL TYPE(' TYPE ONE OF THE FOLLOWING KEYWORDS TO SELECT AN OPTION')
CALL TYPE(' C - CHANGE CENTROMERE (USE CURSOR)')
CALL TYPE(' F - FLIP AND CHANGE CENTROMERE (USE CURSOR)')
CALL TYPE(' R - ROTATE CHROMOSOME (USE CURSOR)')
CALL TYPE(' H - MOVE CHROMOSOME TO ANOTHER SLOT')
CALL TYPE(' X - REMOVE CHROMOSOME FROM KARYOTYPE')
CALL TYPE(' P - PUSH A GROUP OF CHROMOSOMES RIGHT OR LEFT')
CALL TYPE(' L - ADD A LABEL (TYPE LABEL ON SAME LINE)')
CALL TYPE(' A - ABORT')
CALL TYPE(' S - SET UP CURSOR TO CORRECT FOR DRIFT')
CALL TYPE(' U OR D - MOVE CURSOR UP OR DOWN')
CALL TYPE(' RF - RERUN FOB')
CALL TYPE(' DS - DISPLAY SPREAD')
CALL TYPE(' BC - CALL THE BANDED CLASSIFIER')
CALL TYPE(' RB - RERUN THE BANDED CLASSIFIER')
CALL TYPE(' OB - QUICK BAND CALCULATION')
CALL TYPE(' VA - DISPLAY WAVEFORMS')
CALL TYPE(' AX - DISPLAY AXES')
CALL TYPE(' KC - CLEAR CLASSIFY PARAMETERS')
CALL TYPE(' MK - CLEAR MOD PARAMETERS')
CALL TYPE(' MA OR FE - MALE OR FEMALE')
CALL TYPE(' NX - NO X SEPARATION FROM C GROUP')
CALL TYPE(' IF KARYOTYPE IS OK TYPE CARRIAGE RETURN')
CALL TYPE(' @')
GOTO 100

C PARAMETER U OR D - MOVE CURSOR UP OR DOWN
C
910 CALL MOB
goto 110
910 CALL MOB
goto 110
C PARAMETER BC - CALL BANDED CLASSIFIER
C
950 CALL A PHASE(BANDPH)
goto 709
C PARAMETER AR - SET MAXIMUM AND MINIMUM AREAS OF VALID CHROMOSOMES
C
1099 CALL HNL(PAR(IP),MOBPAR(M+1),0)
M=M+4
MOBFLG=TRUE.
goto 110
C PARAMETER LV - LOCATE CENTROMERE BY WIDTH RATHER THAN DENSITY
C
1050 CALL HNL(PAR(IP),MOBPAR(M+1),4)
M=M+2
MOBFLG=TRUE.
goto 110
C PARAMETER CI - SET CENTROMERE OF CHROMOSOME SPECIFIED TO THAT
C INDICATED BY CURSOR POSITION
C
1199 CALL KURSOR(Y, X, L, S, SLID, SLCL, SLC, S).H)
IF(N EQ. 0) GOTO 259
SEL(N)=TRUE.
CALL HNL('CI',MOBPAR(M+1),2)
MOBPAR(N-3)=N
INL=CHDIR(INL,N)
MOBPAR(N+4)=L-(SLCL-INV/2)+1
CALL DLINE(BLACK,Y,X,4,1,-1)
CALL DLINE(BLACK,Y,X,4,1,-1)
IF(MOBPAR(N+4).GE.INL.OR.MOBPAR(N+4).LE.1) GOTO 259
M=M+4
MOBFLG=TRUE.
goto 119
C PARAMETER FL - FLIP SPECIFIED CHROMOSOME AND SET CENTROMERE
C
1159 CALL KURSOR(Y, X, L, S, SLID, SLCL, SLC, S).H)
```
IF(M.EQ.0) GOTO 250
SEL(N) = TRUE.
CALL MVL('FL', MOBPAR(M+1), 2)
MOBPAR(M+3) = N

INL = CHDIR(DIRNL, N)
MOBPAR(M+4) = (SLCL- INL/2)+1
CALL DLINE(BLACK, Y, X, 4, 1, -1)
CALL DLINE(BLACK, Y, X, 4, 1, -1)
IF(MOBPAR(M+4).GE.INL.OR.MOBPAR(M+4).LE.1) GOTO 250
M = M+4
MOBFGL = TRUE.
GOTO 110

C PARAMETER RO - ROTATE INDICATED CHROMOSOME SO THAT THE 2 CURSOR
C SPTS BECOME VERTICAL

1200 CALL KURSOR(Y, X, L, S, SLID, SLCL, SLCS, N)
IF(M.EQ.0) GOTO 250
CALL DLINE(BLACK, Y, X-2, 3, 1, -1)
CALL DLINE(BLACK, Y, X-2, 3, 1, -1)
SEL(N) = TRUE.
CALL MVL('RO', MOBPAR(M+1), 2)
MOBPAR(M+3) = N
FRSTY = Y
FRSTX = X
CALL TYPE('MOVE CURSOR TO END OF AXIS', 0)
CALL PARAM(NP, PAR, 190)
CALL KURSOR(Y, X, L, S, SLID, SLCL, SLCS, N)
CALL DLINE(BLACK, Y, X-2, 3, 1, -1)
CALL DLINE(BLACK, Y, X-2, 3, 1, -1)
SY = Y
SX = X
IF(FRSTX.EQ.SX) GOTO 1217
IF(FRSTY.LE.SY) GOTO 1218

SY = FRSTY
SX = FRSTX
FRSTX = Y
FRSTY = X
1210
THETA = ATAN2((SX-FRSTX), (SY-FRSTY))
GO TO 1215
1217
THETA = 0.
1215
CONTINUE
MOBPAR(M+4) = THETA*DEGPRD
M = M+4
MOBFGL = TRUE.
GOTO 110

C PARAMETER AL - RE-MOB ALL CHROMOSOMES
C
1250 ALLFLG = TRUE.
GOTO 110
C
C PARAMETER SK - OUTPUT SKELETON PICTURES INSTEAD OF GREY LEVEL PICTURE
C
1300 MOBPAR(M+1) = PAR(IP)
M = M+2
MOBFGL = TRUE.
GOTO 110
C
C PARAMETER NN - NO OBJECT NUMBERS
C
1350 KTYPE(IP+1) = PAR(IP)
K = K+2
KTYPEFLG = TRUE.
GOTO 110
C
C PARAMETER MA - MALE KARYOTYPE
C
1450 KTYPE(IP+1) = PAR(IP)
K = K+2
KTYPEFLG = TRUE.
GOTO 110
C
C PARAMETER KC - CLEAR CLASY/KTYPE PARAMETERS
C
1500 K = 0
CALL WPROM(K,KTYPAR,KTYPAR)
OSFLG=.FALSE.
KTYFLG=.TRUE.
GO TO 110

C PARAMETER FE - FEMALE KARYOTYPE

1550 KTYPAR(K+1)=PAR(IP)
K=K+2
KTYFLG=.TRUE.
GO TO 110

C PARAMETER HX - HO X SEPARATION FROM C GROUP

1600 KTYPAR(K+1)=PAR(IP)
K=K+2
KTYFLG=.TRUE.
GO TO 110

C PARAMETER DE DEBUG OPTION ON

1650 KTYPAR(K+1)=PAR(IP)
K=K+2
KTYFLG=.TRUE.
GO TO 110

C PARAMETER P - PUSH A GROUP OF CHROMOSOMES RIGHT OR LEFT

1700 CALL KURSOR(Y,L,L1,S1,SL1,SLCL,SLCS,N)
CALL TYPE("*MOVE CURSOR TO END OF PUSH ",0)
CALL PARAM(NP,PAR,100)
CALL KURSOR(Y,L,L2,S2,SL2,SLCL,SLCS,N)
IF(S1.LT.S2) GOTO 1725
C GOTO 1725 FOR PUSH TO THE RIGHT

C PUSH LEFT

NSL=SL1-SL2+1
IF(NSL.LT.2.OR.NSL.GT.10) GOTO 250
Y=2*(L1-1)
CALL BLINE(127,Y,2*S2,S1-S2,1,-1)
X=2*(S2-1)
30 1710 X=2,12,2
CALL BLINE(BLACK,Y+I-14,Y+14-1,2,1,-1)
1710 CALL BLINE(BLACK,Y+I,X+1,2,1,-1)
CALL MW(OFS(SL2),PAR,NSL)
C SAVE OFS IN PAR
CALL MW(PAR(2),OFS(SL2),NSL-1)
OFS(SL1)=PAR(1)
GOTO 1770

C PUSH RIGHT S1 LT S2

1725 Y=2*(L1-1)
NSL=SL2- SL1+1
IF(NSL.LT.2.OR.NSL.GT.10) GOTO 250
CALL BLINE(127,Y,2*S1,S2-S1,1,-1)
X=2*S2
30 1730 X=2,12,2
CALL BLINE(BLACK,Y+I-14,Y+14-1,2,1,-1)
1730 CALL BLINE(BLACK,Y+I,X+1,2,1,-1)
CALL MW(OFS(SL1),PAR,NSL)
CALL MW(PAR(1),OFS(SL1+1),NSL-1)
OFS(SL1)=PAR(NSL)
GOTO 1770

C PARAMETER MY - MOVE INDICATED CHROMOSOME TO SLOT SHOWN BY CURSOR

1750 CALL KURSOR(Y,L1,S1,SLIB,SLCL,SLCS,N)
CALL BLINE(BLACK,Y-2.3,1,-1)
SVSLIB=SLIB
CALL TYPE("*MOVE CURSOR TO OTHER SLOT ",0)
CALL PARAM(NP,PAR,100)
CALL KURSOR(Y,L,L2,SLIB,SLCL,SLCS,N)
CALL ABT(L1,S1,L2,S2,SS1,1ST,312,512)
30 1770 L=L-L2
Y=2*(L-1)
X=2*(SST(L)-1)

CALL DLINE(127,Y,X,HST(L),1,-1)
SVI(OFS(SLID))
OFS(SLID)=OFS(SVSLID)
OFS(SVSLID)=SVID

OSFLG=.TRUE.
KTYFLG=.TRUE.
GO TO 110

C PARAMETER SP - SET SKELETON PARAMETERS
C
1900 CALL MYL(PAR(IP),MOBPAR(M+1),8)
K=K+4
MOBFLG=.TRUE.
GO TO 110

C PARAMETER MX - SET MAXIMUM OF CHROMOSOMES PER GROUP
C
1850 CALL MYL(PAR(IP),KYPAR(K+1),24)
K=K+12
KTYFLG=.TRUE.
GO TO 110

C PARAMETER IR - SET INITIAL SLOTC FOR EACH ROW
C
1850 CALL MYL(PAR(IP),KYPAR(K+1),14)
K=K+7
KTYFLG=.TRUE.
GO TO 110

C PARAMETER IG - SET INITIAL SLOTC FOR EACH GROUP
C
2000 CALL MYL(PAR(IP),KYPAR(K+1),28)
K=K+14
KTYFLG=.TRUE.
GO TO 110

C PARAMETER MC - CLEAR MOB PARAMETERS
C
2050 M=0
MOBFLG=.TRUE.
GO TO 110

C PARAMETER SC - SET UP THE CURSOR
C
1780 CALL SC
GO TO 110

C SETUP THE CURSOR
C
C PARAMETER DS - DISPLAY SPREAD
C
2125 CALL APHASE(FDBPHA)
GO TO 700

C PARAMETER RF - RERUN FOB
C
1500 CALL APHASE(FDBPHA)
CALL UPARAM(0,MOBPAR,MOBPHA)
CALL UPARAM(0,KYPAR,KYPHA)
GO TO 700

C RERUN FOB
C
C PARAMETER AB - ABORT
C
2200 CALL APHASE(0)
GO TO 700

C PARAMETER X - REMOVE OBJECT FROM KARYOTYPE
C
2250 CALL KURSOR(Y,X,L1,SLID,SLCL,SLCS,R)
DO 2250 I=2,26,2
CALL BLINE(BLACK,Y+1-14,Y+1-16,2,1,-1)
CALL BLINE(BLACK,Y+1-14,Y+14-1,2,1,-1)
CONTINUE

C DRAW AN X OVER THE OBJECT TO BE DELETED
OFS(SLID)=0
C PARAMETER L - ADD A LABEL TO INDICATE DIAGNOSIS

230 DO 2310 I=4,72

2310 IF((VPARBUF(I)).EQ.."IS") GOTO 2320

2320 LABELM=1-I

CALL AF9LE(MOB,1,'PIC ','5,5)

CALL OPEN(MOB,512,6,2,'MOB')

CALL GET(MOB,1,IN)

CALL PUT(MOB,1,IN)

CALL HVL(PARBUF(2),MOB(IN+290),LABLEN)

CALL CLOSE(MOB)

C ALSO ADD TO THE MOB OUTPUT LABEL FOR RESEL'S BENEFIT

CALL GLABEL(10BUF,SPAR,IB)

CALL MVL(PARBUF(3),10BUF(IB+290),LABLEN)

GOTO 110

C PARAMETER WAVE - DISPLAY WAVEFORMS FROM BAND

2350 CALL UPAR9AN(2,'WAVE',BANDPH)

GOTO 950

C PARAMETER AXIS - DISPLAY AXES FROM BAND

2400 CALL UPAR9AN(2,'AXIS',BANDPH)

GOTO 950

C PARAMETER RB - RE-RUN BANDED CLASSIFIER

2450 CALL A9HASE(FOURPH)

GOTO 700

C PARAMETER QB - QUICK BAND CALCULATION (ONLY THE C GROUP)

2500 CALL SSWTCH(7,IS?)

IF(IS?,HE,1) CALL UPAR9AN(2,'QB ','BANDPH)

GOTO 950

C DOING LOOKING AT PARAMETERS, SEE IF A RE-RUN IS NECESSARY

300 IF(.NOT.ALLFLG.AND...NOT.KTYFLG.AND...NOT.MOBFLG) GOTO 700

IF(.NOT.OSFLG) GOTO 550

C WRITE THE OFS PARAMETERS FOR CLASSFY

CALL MVL('OS',KYPAR(K+1),2)

CALL MV9('OFS',KYPAR(K+3),90)

K=K+92

CONTINUE

IF(.NOT.ALLFLG.AND...NOT.MOBFLG) GOTO 650

IF(ALLLFLG) GOTO 610

C MUST RE-RUN MOB ON SELECTED CHROMOSOMES

CALL MVL('SE',MOBPAR(M+1),2)

M=M+3

MM=MM+3

NM=NM+3

DO 600 M=1,NOB

IF(.NOT.SEK(N)) GOTO 600

M=M+1

MM=MM+1

MOBPAR(N)=N

CONTINUE

IF(M NE.0) GOTO 695

MOBPAR(MM)=NM

CALL PUT(10BUF,1,IB)

CALL MVL(MOB,10BUF(IB+1),10)

CALL HVL(CHSTR(1,1),10BUF(IB+31),2*15*30)
CALL MVL(MBUF(ML+73), LABEL, 64)
CALL MVL(MBUF(ML+114), LABEL, 85)
CALL MVL(MBUF(ML+217), LABEL, 105)
CALL MVL(MBUF(ML+289), LABEL, 105)
C SAVE LABEL INFORMATION
C OPEN KROMDATA AND GET ML
C
CALL AFILE(KBUF, 4, 'DATA', 6, 6)
IF(CODE.LT.0) CALL AFILE(KBUF, 4, 'DATA', 6, 6)
CALL OPEN(KBUF, 1024, 0, 2, 'RKH')
CALL GLABEL(KBUF, KSPAR(KL))
ML=KSPAR(1)
C READ SOURCE DIRECTORY RECORD
C
10 CALL GET(KBUF, SCODE, KD)
IF(HUSED.LT.95) GOTO 22
C IF SPACE IN THIS RECORD GOTO 83
IF(NEXT.EQ.8) GOTO 20
C IF LAST DIRECTORY RECORD GO MAKE A NEW ONE
SCODE=NEXT
GOTO 10
C GO READ NEXT DIRECTORY RECORD
20 NL=NL+1
NEXT=NL
C STORE NEXT RECORD IN THIS DIRECTORY RECORD
CALL PUT(KBUF, SCODE, KD)
CALL GET(KBUF, NL, KD)
HUSED=0
C CONTINUE
DO 24 I=1,1421,5
IF(PID(I).EQ.0) GOTO 26
C IF THIS SLOT IS FREE GOTO 26
24 CONTINUE
CALL TYPE(' RESEL ERROR')
CALL TYPE('0')
PAUSE 12345
C STORE PATIENT ID
HUSED=HUSED+1
IR=(I+4)/5
REC=REC(IR)
IF(REC.NE.0) GOTO 28
'C GOTO 28 IF THIS RECORD WAS DELETED'
NL=NL+1
C NEED TO MAKE A NEW RECORD
REC=NL
RECS(IR)=REC
IF(CODE.LT.0) NL=NL+3
C ADD 3 MORE IF BANDED
20 CALL PUT(KBUF, SCODE, KD)
CALL GLABEL(KBUF, KSPAR(KL))
KSPAR(1)=NL
C UPDATE ML
CALL PLABEL(KBUF, KSPAR, KBUF(KL+1))
C GET MOB OUTPUT DATA AND TRANSFER TO KROMDATA
C
C GET OFS FROM MOB OUTPUT AND CONVERT TO TYPE
C
CALL GET(MBUF, J, MD)
II=MD+215
DO 30 I=1, 190
II=II+2
30 CALL MVL(MBUF(I), OFS(I), 1)
CALL ZIA(OM, 40)
CALL ZIA(HM, 60)
CALL ZIA(TM, 90)
CALL ZIA(MFG, 10)
30 40 DJ=1,30
IF(OFSTJJ.EQ.0) GOTO 40
C=GF(S(J)
203 4,122,518

204

40 CONTINUE

CALL GET(MBUF,1,HD)
CALL PUT(KBUF,REC,KB)
CALL NVW(LABEL,KBUF(KB+1),62)
CALL NVW(MBUF(HB+1),NOB,1)

N=1
50 IF(TN(N).EQ.0) GOTO 80

C GOTO 80 IF THIS OBJECT IS NOT TO BE ENTER

T=2+TN(N)-1
IF(T.LT.60. AND. T.GT.0) GOTO 60
T=CMT
IF(T.GT.00) GOTO 60

C USE NEXT AVAILABLE SPACE IN MISCELLANEOUS AREA
C CMT=CMT+1
60 CONTINUE
IF(ON(T).EQ.0) GOTO 70

C GOTO 70 IF FIRST SLOT FOR THIS TYPE IS EMPTY
T=T+1
70 ON(T)=N
N=NH(N)+1

80 CONTINUE
C IF(N.LE.NOB) GOTO 50

C TRANSFER CHDIR. 30 BYTES IS OFFSET OF ONH ON MBUF

II=30+MD
IST=1
MEND=30
DO 100 J=1,2
K=0
DO 90 I=IST,MEND
K=K+1
NH(N)=NH(N+1)
IM=(K+1)*30+11
CALL NVL(MBUF(IM+1),ONH(NHI),1)
CALL NVL(MBUF(IM+7),OLEN(NHI),1)
C SUM=SUM+1
LSUM=LSUM+OLEN(NNI)
CALL NVL(MBUF(IM+9),010B(NHI),1)
SM=SUM+010B(NNI)
CALL NVL(MBUF(IM+21),SAREA(NNNI),1)
ASUM=ASUM+SAREA(NNI)
CALL NVL(MBUF(IM+11),OCL(NNNI),1)
CALL NVL(MBUF(IM+13),OCIL(NHI),1)
CALL NVL(MBUF(IM+15),OCIA(NHI),1)
90 CONTINUE
IF(J.EQ.2) GOTO 100
CALL GET(KBUF,2,MD)
IST=31
MEND=NOB
II=30+MD

100 CONTINUE
CALL NVW(CSUM,KBUF(KB+129),1)
CALL NVW(LSUM,KBUF(KB+127),1)
CALL NVW(ISUM,KBUF(KB+129),2)
CALL NVW(ASUM,KBUF(KB+133),2)
IF(CODE.CT.0) GOTO 190

C CANCELLED PROCESSING

CALL GET(MBUF,4,IN)
CALL NVW(MBUF(IM+1),CFOUR(1,1),400)
CALL GET(MBUF,5,IN)

D
CALL MYE(MBUF(1M+1),CFour(1.5),408)
CALL GET(MBUF,6,IM)
CALL MYW(MBUF(1M+1),PHI(1.1),408)
CALL GET(MBUF,7,IM)
CALL MYW(MBUF(1M+1),PHI(1.5),408)
DO 110 N=1,NOB
T=NH(N)
IF(CF(ED<0 OR 1 GT 51)) GOTO 118
C GET THE FOURIER DATA
DO 102 I=1,S
102 QCFour(T,)*=CFour(N,1)
DO 104 I=1,7
104 OPHI(T,)*=PHI(N,1+I)
110 CONTINUE
C WRITE RECORD ON KRONDATA
C
CALL MYW(ON,KBUF(KD+143),448)
NC=0
DO 200 G=1,9
CALL OUTCON(NHC(G),GMSG(69+G*4),47)
NC=NC+HFG(G)
CALL OUTCON(NHC,GMSG(69),4)
IF(V(GMSG(107)) .LT.32) GOTO 225
C IF OPERATOR SUPPLIED A DIAGNOSIS GOTO 250
DO 220 N=1,NOC
DO 210 G=1,10
IF(NFG(G).LT.HFB(G,N)) GOTO 220
210 CONTINUE
C NFC MATCHES THIS DIAGNOSIS
GOTO 250
220 CONTINUE
C NO MATCH
225 N=0
GOTO 250
230 CALL NYL(DID(N),GMSG(I07),8)
CALL NYL(DID(N),KBUF(KB+1067),8)
250 CALL AFILP(PBUF,512,0,2,'NOR')
CALL GET(PBUF,1,IP)
CALL OUTCON(REC,PBUF(IP+176),4)
CALL NYL('RECORD',PBUF(IP+166),6)
C STORE RECORD NUMBER IN NOC OUTPUT FOR HCOPT
IF(NC .NE. 0) CALL NYL(DID(N),PBUF(IP+259),8)
CALL PUT(PBUF,1,IP)
CALL CLOSE(PBUF)
C STORE DIAGNOSIS N IN PREP AND NOC OUTPUT FOR HCOPT
CALL OUTCON(REC,FMSG(31),4)
CALL MYE(GMSG(107),FMSG(327),8)
CALL TYPE(FMSG)
IF(CODE GT 0) GOTO 380
C WRITE THE BANDED RECORDS
DO 270 T=1,3
CALL PUT(KBUF,REC+1,IK)
270 CONTINUE
C WRITE PATIENT REPORT RECORD
CALL AFILP(PBUF,4,'PDATA','.6,6)
CALL OPEN(PBUF,512,0,2,'FRP')
CALL GLABEL(PBUF,PSVAR,IP)
NLP=NLIP+1
CALL PLABEL(PBUF,PSVAR,PBUF(IP+1))
CALL GET(PBUF,NLP,IP)
CALL MYE(GMSG,PBUF(IP+1),128)
CALL PUT(PBUF,NLP,IP)
CALL CLOSE(PBUF)
CALL CLOSE(KBUF)
CALL CLOSE(MBUF)
CALL TYPE('O')
RETURN
END
C MASK2 - COMBINE 2 PICTURES AND ADD A BORDER
C
C SUBROUTINE MASK2(BUNIT, FILPEX)
C IMPLICIT INTEGER(A-Z)
C REAL STATS(256)
C BYTE *(2560), B(2590) + C(992), LE(12), RE(12)
C COMMON STATS, LE, RE
C INTEGER 2-LOW(512), HIGH(511)
C EQUIVALENCE (LOW.STATS), (HIGH(1), LOW(2))
C INTEGER 2 SPAR(5), SPARB(5), PAR(10), KEY(3)
C BYTE FILPEX(12)
C DATA MANS/992/
C DATA NKEY/3., KEY/'HI', 'HS', 'KG', HFLAG/0, HFR/128, ISIZE/100/
C CALL AFILE(A, 'PIC', '.5, 5)
C CALL AFILE(B, 'PIC', '.5, 5)
C CALL SCALE(A, 'PIC', '.5, 5)
C CALL SCALE(B, 'PIC', '.5, 5)
C CALL AFILE(C, 'PIC', '.5, 5)
C CALL SCALE(C, 'PIC', '.5, 5)
C ASSIGN DK1, PIC(5, 5) AS DEFAULT FILE FOR NOB OUTPUT
C CALL OPENA(A, 3872, 1, 1, 'NOB')
C CALL LABLEL(A, SPAR, IA)
C ASSIGN DK2: PIC(5, 5) AS DEFAULT FILE FOR KGM
C CALL PARAMETER.PAR(10)
C
C IP=1
C IF(IP.GT.10) GOTO 30
C DO 5 J =1, MKEY
C IF(PAR(IP, 1).EQ.KEY(J)) GOTO (10, 12, 15), J
C CONTINUE
C CALL TYPE('** PARAMETER ERROR')
C CALL TYPE('0')
C PAUSE
C HFLAG=1
C CALL ZIA(STATS, 512)
C IP=IP+2
C GO TO 7
C ISIZE=PAR(IP+2)
C IP=IP+1
C GO TO 10
C CALL AFILE(0, PAR(IP+2), PAR(IP+3), 2, 2)
C ASSIGN A DIFFERENT KG M FILE
C CALL OPENB(3872, 1, 0, 'KGM')
C CALL LABLEL(B, SPARB, 1B)
C NL=SPAR(1)
C IF(SPARB(1).GT.NL) NL=SPARB(1)
C NS=SPAR(2)+SPARB(2)+1
C IF(NS.GT.MAXNS) NS=MANS
C IF(SPARB(3).GT.7) HFR=256
C CALL MSUB(A, 1A, 1), MUNIT, FILPEX)
C CALL SCALE(A, 'PIC', '.5, 5)
C ASSIGN A DIFFERENT KG M FILE
C CALL OPENB(3872, 1, 0, 'KGM')
C CALL LABLEL(B, SPARB, 1B)
C NL=SPAR(1)
C IF(SPARB(1).GT.NL) NL=SPARB(1)
C NS=SPAR(2)+SPARB(2)+1
C IF(NS.GT.MAXNS) NS=MANS
C IF(SPARB(3).GT.7) HFR=256
C CALL MSUB(A, 1A, 1), MUNIT, FILPEX)
C CALL SCALE(A, 'PIC', '.5, 5)
C IF(HFLAG.EQ.1) CALL LSTAT(SPARB(2), M1A, 1), STATS, 1.1
C IF(HFLAG.EQ.0) CALL LSTAT(SPARB(2), A, B, 1), STATS, 1.1
C C CONVERT THE FREQUENCY COUNTS TO REAL NUMBERS
C DO 120 I=1, HFR
C STATS(I)=LOW(I)+HIGH(I)*32768.
C 120 CONTINUE
C CALL MSUB(STATS, HFR, ISIZE)
C CALL CLOSE(A)
SUBROUTINE NSUB(LOC-NSO, XSIPE, CILPEX)

REAL STATS(256), MAXF, HSIZE

COMMON STATS, DATA

INTEGER 2, FRED(256), SPAR(5)

EQUIVALENCE (FRED, STATS)

INTEGER 2, ISIZE(5)

BYTE DSRN(6258), T(2104), LMSG(20)

EQUIVALENCE (DSRN(4153), T, DSRN(6257), ISW5)

BYTE LAB(512), LOC(976), BUF(1000), LH(4), LOGN(4)

DATA MAXNS=976, MINNS=360

DATA SPAR/0, 0, 4, 1, 512/

DATA MAXF=1, HSIZE=200, L/G, WHITE-IS, STEP/IS, LBW/0/

IF(LBW.GT.0) GOTO 100

C GOTO. 100 IF NOT INITIAL CALL

CALL AFILE(DSRN, ISIZEIFILPEX, 282)

BUFSIZE=3872

C IF TAPE OUTPUT, DECREASE SIZE OF DSRN TO MAKE ROOM

CALL OPEN(DSRN, BUFSIZE-1, 'MSK')

CALL MVW(LOC, LAB, 256)

C SAVE LABEL AND BPE

NS=NSO

IF(NS.GT.MAXNS) NS=MAXNS

NSD2=(NS+1)/2

C NSD2 IS THE NUMBER OF WORDS TO MOVE TO DATA

IF(NS.LT.MINNS) NS=MINNS

C MINNS IS THE MINIMUM SIZE FOR THE DATE AND TIME LABEL AND HSTGRM

NSW=NS-24

NWW=(NSW+1)/2

SPAR(2)=NWW

SPAR(4)=512/NWW

IF(SPAREND, EQ, 0) SPAR(4)=1

CALL LABEL(DSRN, SPAR, LAB)

IF(ISW5.NE.1) GOTO 58

CALL MVW(1, T(21), 1)

CALL MVW(52140, T(23), 1)

C SET MVB FOR MTO:

CALL OPEN(T, 102, 1, 3, 'TAP')

CALL MVW(1, T(51), 1)

C BPB=1

CALL MVW(NSW, T(45), 1)

C RECLE=NSW

CALL MVW(NSW, T(47), 1)

C BLOCKSIZE = NSW

50 CONTINUE

CALL ZIA(BUFNWW)

CALL WLINE(DSRN, LBW, HSW, BUF, 04)

C WRITE 4 BLANK LINES

CMID=13+NS/2

QWW=NS/32

C GREY WEDGE WIDTH

BUFFER=16+QWW-1

D2=QWW+QWW+3

CALL ITLA(255, BUF(81), B2)

CALL WLINE(DSRN, LBW, HSW, BUF, 01)

CALL WEDGE(WHITE, STEP, BUF, NS).

CALL WLINE(DSRN, LBW, HSW, BUF, 12)

CALL UPNAME(SBWW, -STEP, HSW, NS)

CALL WLINE(DSRN, LBW, HSW, BUF, 12)

CALL ITLA(255, BUF(81), B2)

CALL WLINE(DSRN, LBW, HSW, BUF, 01)

C WRITE GREY SCALE

CALL ZIA(BUFNWW)

CALL WLINE(DSRN, LBW, HSW, BUF, 0)

DO 80 K=0, 6

IF(K.EQ.0) CALL RECF5BUF, HSW)

DO 70 N=100, NS, 100

CALL OUTCON(N, LH, 4)

70 CALL TEXT(LH(2), 1, K, BUF(N+5), 1)

80 CALL WLINE(DSRN, LBW, HSW, BUF, 01)

CALL RECF5BUF, HSW)
CALL ULINE(DSRH, LBW, NSW, BUF, 03)
C WRITE TOP REFERENCE MARKS
CALL ITLA(255, BUF(5), 8)
CALL ITLA(255, BUF(13+HS), 8)
CALL ULINE(DSRH, LBW, NSW, BUF, 01)
RETURN
100 CONTINUE
IF(NS0.LE.0) GOTO 200
C IF END OF PICTURE, GOTO 200
IF(EPE.NE.7) GOTO 110
CALL NUTA(LOC, DATA(13), NSW2)
GOTO 110
110 CALL MVW(LOC, DATA(13), NSW2)
115 L=L+1
C STEP INPUT LINE NUMBER
CALL ITLA(LOC, DATA(1), 12)
CALL ITLA(LOC, DATA(13+HS), 12)
C ZERO LINE REFERENCE MARKS
LP3=LP3+3
ML=MOD(LP3, 100)
IF(ML.GT.6.OR.L.LT.4) GOTO 130
C TIME FOR NUMBER OF HUNDREDS
CALL OUTCON(LP3, LM(4), 4)
CALL TEXT(LM(2), 1, ML, DATA(17+HS), 1)
GOTO 140
130 CONTINUE
IF(MOD(L, 5).NE.0) GOTO 150
CALL ITLA(255, DATA(5), 4)
CALL ITLA(255, DATA(17+HS), 4)
C INSERT EVERY 5 MARK
140 CONTINUE
IF(MOD(L, 25).NE.0) GOTO 150
CALL ITLA(255, DATA(9), 4)
CALL ITLA(255, DATA(13+HS), 4)
150 CONTINUE
CALL WLINE(DSRH, LBW, NSW, DATA, 1)
C WRITE ONE LINE OF DATA
RETURN
200 CONTINUE
C FINAL ENTRY, WRITE BOTTOM OF PICTURE
DO 240 K1=1, 11
IF(K1.LT.5) CALL REF(25, BUF, NSW)
IF(K1.GE.5.AND.K1.LE.8) CALL REF(5, BUF, NSW)
IF(K1.EQ.9) CALL ZIA(BUF, NSW)
IF(K1.GE.10) GOTO 230
C FINISH UP NUMBER THAT WAS STARTED ON LEFT AND RIGHT SIDE
ML=ML+1
CALL TEXT(LM(2), 1, ML, BUF(2), 11)
CALL TEXT(LM(2), 1, ML, BUF(17+HS), 11)
220 CONTINUE
IF(K1.LT.5) GOTO 240
ML=ML+100
H=H+100
HS=100
CALL OUTCON(H, LOGN(4), 4)
230 CALL TEXT(LOGN(2), 1, K-5, BUF(H+9), 1)
CALL WLINE(DSRH, LBW, NSW, BUF, 1)
C WRITE BOTTOM REFERENCE MARKS
CALL ZIA(BUF, NSW)
CALL WLINE(DSRH, LBW, NSW, BUF, 1)
CALL ITLA(255, BUF(01), B2)
CALL WLINE(DSRH, LBW, NSW, BUF, 01)
CALL WEDGE(255, -STEP, BUF, HS)
CALL WLINE(DSRH, LBW, NSW, BUF, 12)
CALL WEDGE(WHITE, STEP, BUF, HS)
CALL WLINE(DSRH, LBW, NSW, BUF, 12)
CALL ITLA(255, BUF(01), B2)
CALL WLINE(DSRH, LBW, NSW, BUF, 01)
C WRITE GREY SCALE
MAXCHR=(NS+11)/12
JMAX=433
DO 250 J=J+1, JMAX-72
IF(J.LT.(J-1).NE.67) GOTO 260
C FINISHED WITH LABELS, IF NO C IN COLUMN 72
J=J+1
C SKIP THE FIRST LABEL CHARACTER WHICH IS ALWAYS BLANK.
C FIND THE LENGTH OF THIS LABEL
20 242 I=1,76
242 CONTINUE
C ENTIRE LABEL IS BLANK
GOTO 259
243 LABELN=78-1
244 CHR=LABELN
IF(LABELN.LE.MAXCHR) GOTO 247
C SPLIT UP LABEL IF IT IS TOO LONG
20 245 I=1,20
IF(LAB(J+70-I).EQ.32) GOTO 246
245 CONTINUE
C NO BLANK FOUND IN 20 CHARACTERS. MAKE ARBITRARY SPLIT
I=1
246 CHR=MAXCHR-I+1
247 CONTINUE
C WRITE LABELS
JI=J+CHR
LABELN=LABELN-CHR
IF(LABELN.GT.0) GOTO 244
250 CONTINUE
C NO = 8 GO TO 420
260 CONTINUE
IF(NSO.EQ.8) GOTO 420
C SKIP HISTOGRAM IF NSO EQ 0
30 320 HFR=NSO
HSIZE=ISIZE(1)
CALL ZIAR(BUF,NW)
CALL WLINE(DSRH,LDW,NSU,BUF,0)
IFR=HFR-1
DO 320 I=2,HSFR1
IF(STATS(J).LE.MAXF) GOTO 320
MAXF=STATS(J)
320 CONTINUE
IF(STATS(J).GT.MAXF) STATS(J)=MAXF
IF(STATS(HFR).GT.MAXF) STATS(HFR)=MAXF
DO 330 I=1,1
330 FREQ(I)=(STATS(I)/MAXF)*HSIZE+.9999
BARY=HS-HFR
HSTART=(HS-BARY+HFR)/2+12
J=HSIZE
340 H=HSTART
DO 350 I=1,1
350 H=H+HFR
CALL ZJAR(BUF,NW)
CALL WLINE(DSRH,LDW,NSU,BUF,1)
J=J-1
360 CONTINUE
C THE FOLLOWING CODE WAS COMMENTED OUT TO MAKE ROOM FOR TAPE OUTPUT
C DO 370 J=8.64.56
370 CONTINUE
C CALL ZIAR(BUF,NW)
C HSTEP=J BARW
C HMAX=HSTART+HFR+BARW+1
C DO 360 H=HSTART HMAX HSTEP
C CALL ZJAR(BUF,NW)
C CALL WLINE(DSRH,LDW,NSU,BUF,5)
C CONTINUE
C CALL ZIAR(BUF,NW)
C CALL WLINE(DSRH,LDW,NSU,BUF,2)
C HSTEP=16*BARW
C DO 400 K=0.6
400 H=HSTART
C CALL TEXT(10,1,K,BUF(H-2),1)
C DO 390 H=H,HSTEP
C IF(BARW.EQ.1 AND MOD(H,32).EQ.16) GOTO 390
C CALL OUTCON(N,LH(4),4)
C**       **
C**
C   BAND
C** *
C FIND WAVEFORM AND FOURIER COEFFICIENTS FOR BANDED CHROMOSOMES
C
COMMON /C1/  HOB,I1BER
COMMON /C1/  CHDIR,SM1BUST,RG2BUF,NSW,HL,C2R0LTH
COMMON /C1/  SST,EST
COMMON /C1/  SPIOB,SP1TH,SPARE
COMMON /C1/  HCHR
COMMON /C1/  CFORH,PHI
COMMON /C1/  BAND,ANIS
COMMON /C1/  OB1,BUFF,BYTE
BYTE LRG2BUF(25,8)
BYTE OB1(4,52)
BYTE TEMPB
LOGICAL OPT1,OPT2,OPT3,SEL(68)
INTEGER TEMPI
INTEGER SPAR(5),OFS(72)
INTEGER BMX
INTEGER BAND(160)
INTEGER KPAR(200)
INTEGER BPI
INTEGER CHDIR(15,68)
INTEGER C2R0LTH
INTEGER PAR(10)
INTEGER CLPARM(130)
REAL CF0UR(69,8),PHI(69,8)
REAL ANIS(150)
REAL COSINE(150),SINE(150)
REAL AFOUR(69,8),BF0UR(69,8)
C EQUIVALENCE(XP,PP,QQ,SIGX,JX2,YSTAR,AF,JDIF)
C EQUIVALENCE(YPP,QQQ,SIGY,LY2,YEND,BF,JUV)
C EQUIVALENCE(XTERM,PPPPP,NSL,NHZ,LPG,FZ,RR,JUL)
C EQUIVALENCE(YTERM,PPP,HL,LL,NG,HDIV,RMJ,JLL)
DATA LIN=0, OPT1=.FALSE., OPT2=.FALSE., OPT3=.FALSE.
C LENGTH OF WINDOW FOR WAVEFORM CALCULATION ON STRAIGHT CHROMOSOMES
DATA SEL=60., TRUE.

DO 18 I=1, 60
SEL(I)=.FALSE.
10 CONTINUE

CALL AFIL(E(OBUF, 'RCR', '13', '13')
CALL GLAGEL(OBUF, SPBIB, IB)
CALL CET(OBUF, IB, 18)
CALL MYL(OBUF(18+3), SPIOD, IB)
 CONTINUE
DO 310 II=1, NCHR
IF (.NOT. SEL(II)) GOTO 310
CUROLN=CHDIR(I, II)
NL=CHDIR(2, II)
IF (CUROLN .EQ. 0) NL=0
NS=CHDIR(3, II)
IF (NL.LT.1.AND.NS.LT.1) GOTO 310
NG=0
LPG=1024/NS
CALL GETCOBUF, CUROLN, IB)
311 CONTINUE
30 CONTINUE

DO 310 II=1, NCHR
IF (.NOT. SEL(II)) GOTO 310
CUROLN=CHDIR(I, II)
NL=CHDIR(2, II)
IF (CUROLN .EQ. 0) NL=0
NS=CHDIR(3, II)
IF (NL.LT.1.AND.NS.LT.1) GOTO 310
NG=0
LPG=1024/NS
CALL GETCOBUF, CUROLN, IB
311 CONTINUE
30 CONTINUE

DO 310 II=1, NCHR
IF (.NOT. SEL(II)) GOTO 310
CUROLN=CHDIR(I, II)
NL=CHDIR(2, II)
IF (CUROLN .EQ. 0) NL=0
NS=CHDIR(3, II)
IF (NL.LT.1.AND.NS.LT.1) GOTO 310
NG=0
LPG=1024/NS
CALL GETCOBUF, CUROLN, IB
311 CONTINUE
30 CONTINUE

WRITE(5, 2000) (OBUF(I+1), I=1, NS)
CALL MYL(OBUF(I+1), LRGBUF(1, NG), NS)
IB=IB+NS
IF (NG.GE. NL) GOTO 311
311 CONTINUE
30 CONTINUE

TEST THE ORIENTATION
C SCAN ALONG X (ROW) DIRECTION
JX=NS/2
IY=NL/2
NHZ=0
MSX=0
MSY=0
C
DECIDE IF IT IS PROBABLY BENT OR NOT MAYBE

IF (NSX.EQ.1 .AND. NSEHY.EQ.1) GO TO 410
IF (NSX.EQ.1 .AND. NSEGY.EQ.1) GO TO 420
IF (NSX.EQ.1 .AND. NSEGY.EQ.1) GO TO 430
GO TO 440

CONTINUE

STRAIGHT

DO 411 I=1, NL
LASTB=0
DO 410 J=1, NS
IF (LRGBUF(J,I).NE.0) GO TO 406
LASTB=J
CONTINUE

CONTINUE

IL=(LASTB+FIRSTB+1)/2
IBAND=0
IL=LASTB+1
IH=FIRSTB-1
ILHIL=IH-IL+1
IF (IHIL.LE.LWIN) GOTO 4200

C IF WIDTH IS LE WIDTH Window GOTO 4200
IK=LASTB+LWIN
DO 4100 J=IK, IK
IBAND=IBAND+LRGBUF(J,1)
MBAND=IBAND
C MBAND IS MAXIMUM VALUE OF LWIN SAMPLES
4150 IK=IK+1
IBAND=IBAND-LRGBUF(IL,1)+LRGBUF(IK,1)
C MOVE ONE PLACE OVER
IL=IL+1
IF (IBAND.GT.MBAND) MBAND=IBAND
IF (IK.LT.IH) GOTO 4150
C CHECK IF ALL SAMPLES PROCESSED
BAND(IBAND)=MBAND/LWIN
GOTO 4095

4200 DO 409 J=1L, IH
409 IBAND=IBAND+LRGBUF(J,1)
BAND(IJ)=IBAND/IHIL
CONTINUE

IF (OPT3) LRGBUF(IL,1)=1
CONTINUE

GO TO 440
CONTINUE

FLIP IT DIAGONALLY
CONTINUE

BENT AND PROPERLY ORIENTED

AVE Y (1)

\[ \text{MNL = } ((\text{NL/3}) \times 3) + 1 \]
\[ \text{NP = 0} \]
\[ \text{NSIGY = 0} \]

DO 431 \( I = 1, \text{NL/3} \)
DO 431 \( J = 1, \text{MNL} \)
IF (LRGBUF(J, I).EQ.0) GO TO 431
NP = NP + 1
NSIGY = NSIGY + FLOAT(I)

CONTINUE.....
IZERO = NSIGY/HP

AVE X (J)

\[ \text{MSL = } ((\text{MS/3}) \times 3) + 1 \]
\[ \text{NP = 0} \]
\[ \text{NSIGX = 0} \]

DO 422 \( J = 1, \text{MS/3} \)
DO 422 \( I = 1, \text{HL} \)
IF (LRGBUF(J, I).EQ.0) GO TO 422
NP = NP + 1
NSIGX = NSIGX + J

CONTINUE
IZERO = NSIGX/HP

A11 = 0.0
A21 = 0.0
A12 = 0.0
A22 = 0.0
A13 = 0.0
A31 = 0.0
A23 = 0.0
S = 0.0
T = 0.0
A33 = 0.0
U = 0.0

NSM2 = MS - 2
DO 500 \( J = 1, \text{NSM2/3} \)
DO 500 \( I = 1, \text{HL} \)
IF (LRGBUF(J, I).EQ.0) GO TO 500
XP = J - IZERO
YP = I - IZERO
A23 = A23 + 1.0
XTERM = XP
YTERM = YP
A23 = A23 + YTERM
U = U + XTERM
YTERM = YTERM + YP
S = S + XTERM + YTERM
A23 = A23 + YTERM
YTERM = YTERM + YP
A23 = A23 + YTERM
YTERM = YTERM + YP
A11 = A11 + YTERM

CONTINUE

A12 = A12
A13 = A13
A31 = A31
A23 = A23
PP = S*A22 + A33 + A23 + (A12 + U + A13 + 7)
PPP = U + A22 + A13 + A23 + S + A33 + T + A12
DO 700 I=1, NL
YP=I-ZERO
DPY=2.0+P*YP+Q
DYDXY=1.0/DXDY
XP=P*(I-ZERO)*2+Q*(I-ZERO)+R+JZERO
MJ=XP+0.5
MI=YP+ZERO+0.5
I=0
C IF (OPT3) LRBUF (MJ, MI) = I
C BAND(I) = LRBUF (MJ, MI)
C MI=YP+ZERO+0.5-DY
MJ=XP+1.5
C IF (MI, LE, 0) GO TO 701
IF (MI, GT, NL) GO TO 701
BAND(I) = BAND(I) + LRBUF (MJ, MI)
C MJ=XP+0.5
MI=YP+ZERO+0.5+DY
C IF (MI, LE, 0) GO TO 701
IF (MI, GT, NL) GO TO 701
BAND(I) = BAND(I) + LRBUF (MJ, MI)
C GO TO 702
C 701 CONTINUE
NBIV=2
BAND(I) = BAND(I) / NBIV
NBIV=3
702 CONTINUE
BAND(I) = BAND(I) / NBIV
700 CONTINUE
C CALCULATE LENGTH
C PT2=P*P
YSTAR=0./PT2
YPY=PY+YPY
EPY=EPY+EPY
EL=EL+EPY+SORT (PPYE*PPYE+1.0)+ALOG (PPYE+SORT (PPYE*PPYE+1.0))
CHDIR=411.1+EL/PT2+0.5
C USE L2D LENGTH DUE TO UNUSUAL RESULTS!
C CONTINUE
C 440 CONTINUE
C FOURIER COEFFICIENTS
C BMAX=BAND(I)
DO 710 I=2,NL
IF (BAND(I), GT, BMAX) BMAX=BAND(I)
710 CONTINUE
CONTINUE
C
DO 711 I=1, NL
AHIS(I)=(BANK(I)*100)/AMAX
711 CONTINUE
C
NHARM=0
FNLO=FLOAT(NL-1)
PIL=6.2832/FNLO
C
CALL SSWTCH(4, ISW4)
IF(ISW4.EQ.1) CALL TIMER
C
NLHALF=(NL+1)/2
DO 5820 J=1, NLHALF
RMJ=FLOAT(J-1+PIL)
LRMJ(J)
COSINE(LRMJ)=COS(RMJ)
SINE(LRMJ)=SIN(RMJ)
NLJ=LJ-1
COSINE(NLJ)=COSINE(LRMJ)
SINE(NLJ)=SINE(LRMJ)
5820 CONTINUE
C
DO 5850 I=1, NHARM
AF=0.9
BF=0.9
IM1=I-1
RM=FLOAT(IM1)*PIL
C
DO 5810 J=1, NL
JM1=J-1
RMJ=RM*FLOAT(J-1)
LRMJ=JM1*IM1
LRMJ=MOD(LRMJ, NL-1)+1
C
AF=AF+AHIS(J)*COSINE(LRMJ)
BF=BF+AHIS(J)*SINE(LRMJ)
5810 CONTINUE
C
AF=(AF+AF)/FNLO
BF=(BF+BF)/FNLO
C
CFOUR(HMVR(0, I)=SORT(AF*AF+BF*BF)
PHI(HMVR(0, I)=ATAN2(BF, AF)
C
C
5850 CONTINUE
C
IF(ISW4.EQ.1) CALL TIMER
C
IF (.NOT. OPT2) GO TO 810
DO 811 I=1, NL
C
MAX WIDTH OF WAVEFORM IS 20 SAMPLES
JDF=HS-JUV
JLL=JDF/2+1
JUL=HS-JDF/2
DO 812 J=1, JLL
LRBUF(J)=0
DO 813 J=JLL, JUL
LRBUF(J)=127
DO 814 J=JUL, HS
LRBUF(J)=0
811 CONTINUE
810 CONTINUE
C
IF (.NOT. OPT3 .AND. .NOT. OPT2) GO TO 310
C
CUROLN=CHDIR(1, II)
NG=0
LPC=1024/NS
C
DO 888 I=1, NL
CALL GET(ODBUF, CUROLN, IB)
DO 831 III=1, LPC
NG=NG+1
C
CALL MVU(LGBUF(1), IB), OBUF(IB+1), NS)
 IB=IB+NS
 IF(NC.GE.NL) GO TO 842
 831 CONTINUE
 842 CONTINUE
 CALL PUT(obuf, CROSLN, IB)
 CROSLN=CROSLN+1
 IF(NC.GE.NL) GO TO 841
 841 CONTINUE
 C 310 CONTINUE

C OUTPUT FOURIER COEF'S TO FILE RCR
 CALL PUT(OBUF, 4, IB)
 CALL MVU(CFOUR(1,1), OBUF(IB+1), 400)
 CALL PUT(OBUF, 5, IB)
 CALL MVU(CFOUR(1,5), OBUF(IB+1), 400)
 CALL PUT(OBUF, 6, IB)
 CALL MVU(PHI(1,1), OBUF(IB+1), 400)
 CALL PUT(OBUF, 7, IB)
 CALL MVU(PHI(1,5), OBUF(IB+1), 400)
 C CALL CLOSE(OBUF)

C CREATE PARAMETERS FOR PHASE 14 (CLAS)
 DO 4000 I=1, NCHR
 CLPARM(9+2*I)=CHDIR(4, I)
 CLPARM(10+2*I)=CHDIR(8, I)
 IF(CHDIR(1, I), EQ, 0) CLPARM(9+2*I)=0
 C IF BLIND IS ZERO, THE PHASES ARE REJECTED BY MOD
 <4000. CONTINUE
 C SET UP PART OF CHROMOSOME DIRECTORY FOR NEXT PHASE
 C CLPARM(1)=NCHR
 C POSITIONS 2 THRU 10 MAY BE REGARDED AS SPARE
 C CALL UPARM(130, CLPARM, 15)
 C IF 1902, EQ, 1: CALL APHASE(10)
 C IF SUB IS UP, CALL KTYPE INSTEAD OF FOUR
 C IF(OPT2 OR OPT3) CALL APHASE(10)
 C CALL EXIT
 ENDO
 SUBROUTINE FOUR
 COMMON /CI/ CHDIR, SMILBUF, LGBUF, NS, NL, CROSLN
 COMMON /CI/ SP10D, SPLTH, SPAREA
 COMMON /CI/ NCHR
 COMMON /CI/ CFOUR, PHI
 COMMON /CI/ GBUF, BUF
 INTEGER JV(50), 1TAB(100), JTAB(100)
 BYTE JNSEC(127)
 BYTE OBUF(212)
 BYTE C(364)
 REAL LTHM(6B), CTAB(6B)
 INTEGER START(25), INHC(25), MHC(25)
 INTEGER LOFS(91)
 INTEGER DES(118), KPARE(92)
 INTEGER CLPARM(130)
 INTEGER CLPAR(5)
 INTEGER SPAR(5)
 INTEGER CROSLN
 INTEGER PPL
 REAL CFOUR(68, 18), PHI(68, 18)
 INTEGER MCHROM(24)
 INTEGER CLASS(64)
 BYTE NSC(30)
 REAL DIST(18)
 REAL VECTOR(18)
 REAL LTRK(68, 24)
REAL NUC(10,24)
REAL SIGM(18,24)
REAL MBUF(18,24)
REAL SBUF(18,24)

C
EQUIVALENCE(NBUF(1,1),C(57))
EQUIVALENCE(SBUF(1,1),C(1785))
EQUIVALENCE(SBUF(1,1),SIGMA(1,1))
EQUIVALENCE(NBUF(1,1),NU(1,1))
EQUIVALENCE(KPAR(3),DFS(1))

C
DATA VMEAS/16/,LMEAS/11/,UMEAS/3/,UMEAS/12/
DATA VMAX1/36/,VMAX2/36/,VMAX3/50/,VMAX4/50/,VMAX5/64/
DATA IS1/19/,IS2/24/
DATA JC1/6/,JC2/13/
C SLOTS TO BE CLASSIFIED WITH HYBRID CLASSIFIER
C
DATA NEXTPH/20/
C
C
NEXT-PHASE TO CALL FOR HYBRID CLASSIFIER
DATA KMEAS/2/,K2MEAS/3/,KXMEAS/1/,NGROUP/24/
DATA START/1.5,9.13,15,19,21,23,25/
1 27,29,31,37,39,41,45,49,51/
C M5.57,65,67,74.70,74/
DATA NCR/22+1,-1,-1,1/
DATA MNC/23+2,1,46/
DATA OFS/118+0/
DATA NCHROM/24+0/
DATA CLASS /60+0/
C
C
OBTAIN PARAMETERS, LENGTH AND CENTROMERE INDEX,
C AS WELL AS THE NUMBER OF CHROMOSOMES IN THE SPREAD
C
CALL RPARM(KPARR,CLPARN,130)
CALL SSITCH(4,ISU4)
IF(ISU4.EQ.1) PAUSE 15
C
C
PAUSE IF SW4 IS UP
C
NCVR=CLPARN(1)
C
READ BACK FOURIER COEF'S PRODUCED BY BAND (PHASE 14)
C
CALL AFILF(OBUF,1,'RCR ','*13.*13')
CALL OPEN(OFBUF,1824,1,8,'RCR')
CALL GLABEL(OFBUF,SPAR,18)
IF(OBUF(1B=104),EQ.'*105') MNC(23)=1
IF(OBUF(1B=104),EQ.'*106') MNC(24)=0
C
C
CHECK FOR SEX OF M OR F AND CHANGE X AND Y MAX
C
CALL GET(OBUF,3,1B)
CALL MVL(OFBUF(217+18),LOFS(1),182)
C
CALL GET(OBUF,4,1B)
CALL MVL(OFBUF(1+18),CFOUR(1,1),968)
C
C
CALL GET(OBUF,5,1B)
CALL MVL(OFBUF(1+1B),CFOUR(1,5),968)
C
CALL GET(OBUF,6,1B)
CALL MVL(OFBUF(1+1B),PHI(1,1),968)
C
CALL GET(OBUF,7,1B)
CALL MVL(OFBUF(1+1B),PHI(1,5),968)
C
C
C
*****************************************************************************
C
THE HYBRID CLASSIFIER USES RESULTS OF
THE CONVENTIONAL CLASSIFIER
C
CALL SSITCH(7,ISU7)
IF(ISU7.EQ.1) GOTO 880
C
GOTO 880 FOR THE HYBRID CLASSIFIER
C SW7 UP IS FOR A FULL FOURIER CLASSIFIER
IS1=1
IS2=90
JC1=1
JC2=24
NEXTPH=9
GOTO 820
800 CONTINUE
CALL MV((LOFS, OFS, 90)
CALL ZIA((OFS, IS1), 32-1S1)
C FOUR WILL CLASSIFY THE C GROUP STARTING WITH IS1
CALL ZIA((OFS, 13), 4)
CALL ZIA((OFS, 37), 6)
CALL ZIA((OFS, 55), 4)
CALL ZIA((OFS, 65), 4)
C KFX WILL CLASSIFY Y.B.D.E. AND C
C
820 IF(NCHR.LE.46) GOTO 830
C IF MORE THAN 46, ALLOW FOR 2.X AND 2 Y
MNC(24)=2
830 CONTINUE
C C GET MEANS AND SIGMAS
CALL AFILC((C, 4,'CDATA ' ,"6.6")
CALL OPEN((C, 3500.0.0,'CDATA ')
CALL ILABEL((C, CLPAR, IC)
CALL GET((C, 1, IC)
895 CONTINUE
C C ITOTL=0
NRELCH=CHR
DO 899 II=1,NCHR
IF(CLPARM(9+2*II).EQ.0) NRELCH=NRELCH-1
ITOTL=ITOTL+CLPARM(9+2*II)
899 CONTINUE
FTOTL=46.*ITOTL/NRELCH
DO 4000 I=1,NCHR
LTAB(I)=(CLPARM(9+2*I)*10000./FTOTL)
CTAB(I)=CLPARM(19+2*I)
C STORE REAL LENGTH AND CI
4000 CONTINUE
C C DO 896 II=1,NCHR
DO 896 JJ=1,NGROUP
LIKEX(I,JJ)=999.
896 CONTINUE
C C CALL SSWSCH((0, ISW)), ISW)
IF(ISW0, EQ.1) CALL QPRINT('1
1 OBJ GROUP BLEN DCIA VLEN VCIA
2 VCSUM VPSUM TVAR')
C DO 900 IS=IS1, IS2
II=LOFS(IS)
IF(I1, EQ.0) GOTO 900
C IF((ISW0, EQ.1) CALL QPRINT('"
C DO 903 JJ=JC1, JC2
JJ=I1
IF(JJ, EQ 13, AND. ISW7, NE. 1) JJ=23
C SIG12=SIGMA(AKMEAS, JJ)
SIG222=SIGMA(BKMEAS, JJ)
IF(SIG12, LT 0.0001) SIG12= 0.0001
IF(SIG222, LT 0.0001) SIG222= 0.0001
C W=(LTAB(I1)-MU(KMEAS, JJ))
DIST(I1)=CTAB(I1)-MU(KMEAS, JJ)
DIST(2)=CTAB(I1)-MU(KMEAS, JJ)
VECTOR(1) = (DIST(1)/SIGMA1) * BIST(1)
IF(VECTOR(1) > VMAX1) GOTO 983
VECTOR(2) = (DIST(2)/SIGMA2) * BIST(2)
IF(VECTOR(2) > VMAX2) GOTO 983
C
FSUM = 0.0
DO 987 I = NJEAS, NJEAS
DIST = FOUR(I, I - 3) * MUL(I, JJ)
FSUM = FSUM + (DIST - MUL(I, JJ))
987 CONTINUE
IF(FSUM > VMAX3) GOTO 983
PSUM = 0.0
IF(JMEAS_EQ_0) GOTO 983
DO 988 I = NJEAS, NJEAS
DIST = PHI(I, I - 10) - MUL(I, JJ)
IF(DIST > 3.1415) DIST = 6.2832 - DIST
IF(DIST < -3.1415) DIST = 6.2832 - DIST
PSUM = PSUM + DIST * SIGMA(I, JJ)
988 CONTINUE
IF(FSUM > VMAX4) GOTO 983
DO 989 I = NJEAS, NJEAS
PSUM = FSUM
DO 951 I = IS, ISZ
EU = -LAM(I, JJ)
C
DO 952 I = 1, NCHR
ILES = VMAX4
IISAV = 0
C
DO 958 I = JC1, JC2
JJ = IJ
IF(JJ_EQ_13) AND ISW7 = 1) JJ = 23
IF(HCROM(JJ), HE, MNC(JJ)) GOTO 958
C
C IGNORE GROUPS THAT ARE FULL
C LOOK FOR UNCLASSIFIED CHROMOSOMES
C
DO 951 IS = ISL, IS2
II = LOFES(IS)
IF(IJ_EQ_0) GOTO 951
IF(CLASS(IJ, HE, 0) GOTO 951
IF(LIKE(I, JJ), HE, ELMIN) GOTO 953
ELSE = LIKE(I, JJ)
IISAV = II
JSAY = JJ
953 CONTINUE
951 CONTINUE
C
950 CONTINUE
C
IF(IISAV_EQ_0) GOTO 960
CLASS(IISAV) = JSAY
HCROM(JJSAY) = HCROM(JJSAY) + 1
OFFST(RET(JJSAY)) = IISAV
START(JJSAY) = START(JJSAY) + INCX(JJSAY)
952 CONTINUE
C
960 CONTINUE
C LOOK FOR UNCLASSIFIED AND SEE IF MOVES CAN BE MADE
C ORDER UP TO 100 LIKELIHOODS IN IITAB AND JJTAB
C
DO 388 IK=1,188
ELMIN=VMAX5
DO 250 IS=1,IS2
II=LOFS(IS)
IF(II.EQ.0) GOTO 250
IF(CLASS(II).NE.0) GOTO 250
DO 240 JJ=JC1,JC2
JJ+JJ
IF(JJ.EQ.13.AND.ISW7.NE.1) JJ=23
IF(MNC(JJ).EQ.0) GOTO 240
IF(LIKLE(JJ).GE.ELMIN) GOTO 240
ELMIN=LIKLE(JJ)
II=ISAV
JJSAV=JJ
CONTINUE
250 CONTINUE
IF(ELMIN.EQ.0) GOTO 320
LITAB(IIK)=ISAV
JJTAB(IIK)=JJSAV
LIKLE(ISAV, JJSAV)=998
C IF NO MORE LIKELY CLASSIFICATIONS GOTO 320
C RAISE LIKLE
300 CONTINUE
C
IK=IK+1
IF(IK.EQ.0) GOTO 320
DO 500 K=1,IK
II=IITAB(K)
J=JJTAB(K)
IF(CLASS(J).NE.0) GOTO 500
C SEE IF WE CAN MOVE SOMEONE OUT OF GROUP J INTO A LIGHT GROUP
JS1=START(J)-MCHROM(J)*INCR(J)
J=START(J)-INCR(J)
IF(JS1.GT.JS2) CALL SWAP(JS1,JS2)
ELMIN=VMAX5
DO 450 JS=JS1,JS2
I=OFS(JS)
DO 440 JJ=JC1,JC2
JJ+JJ
IF(JJ.EQ.13.AND.ISW7.NE.1) JJ=23
IF(MNC(JJ).GE.MNC(JJ)) GOTO 440
IF(LIKLE(JJ).GE.ELMIN) GOTO 440
BLIK=LIKLE(JJ)
LIKLE(JJ)=1
JS=JS1
CONSINUE
450 CONTINUE
IF(ELMIN.EQ.0) GOTO 500
C IF NO MOVE OUT OF J CAN BE MADE, GOTO 500
C OTHERWISE, MOVE OBJECT JSAV FROM SLOT JJSAY TO GROUP JJSAY
C THEN, MOVE OBJECT II TO SLOT JJSAY
C
CLASS(ISAV)=JSAV
MCHROM(JSAV)=MCHROM(JJSAY)+1
OFS(JSAV)=I
CLASS(JJSAY)=START(JJSAY)+INCR(JJSAY)
C
CLASS(IJ)=J
OFS(JJSAY)=II
500 CONTINUE
C
520 CONTINUE
C C CHECK FOR STILL UNCLASSIFIED AND PUT AT BOTTOM
C
DO 900 IS=IS1,IS2
II=LOFS(IS)
IF(II.EQ.0) GOTO 900
C
IF(CLASS(II).NE.0) GOTO 980
CLASS(II)=25
965 IF(START(25).GE.90) GOTO 980
START(25)=START(25)+1
IF(OFS(START(25)).NE.0) GOTO 965
OFS(START(25))=I1
980 CONTINUE
C
CALL MYL('OS ', KPAR, 4)
CALL XPARAM(92, KPAR, NEXTPH)
C
CURREM=7
CALL GET(0BUF, CURREM, IB)
CALL MYL(CCLASS(IJ), OBUF(951+IB), 40)
CALL PIP(0BUF, CURREM, IB)
C
CALL CLOSE(0BUF)
CALL PIPASE(NEXTPH)
C
IF(ISUB. EQ. 1) CALL QPRINT('1')
RETURN.
END

********
C
KFIX
C
********
C
CHROMOSOME CLASSIFIER PART 3
C
SYNTACTICAL CORRECTION OF CLASSIFICATION
C
SUBROUTINE KFIX
CALL /C1/ HOB, IDIR
COMMON /C1/ CHDIR, SBMIBF, LRBUF, LS, NL, CURREM
COMMON /C1/ SST, EST
COMMON /C1/ SPLL, SPLL, SPARE
COMMON /C1/ NCHR
COMMON /C1/ CBOR, RHI
COMMON /C1/ SBW, AANIS
COMMON /C1/ OBS, BUF
BYTE LRBUF(78, 88)
BYTE 0BUF(12124)
LOGICAL BGROUP(60)
LOGICAL DGROUP(60)
LOGICAL GGROUP(60)
LOGICAL FGROUP(60)
LOGICAL EGROUP(60)
INTEGER IICHR(6)
INTEGER IDCNCHR(6)
INTEGER IIICHR(6)
INTEGER IICHR(6)
INTEGER KPAR(92)
INTEGER OFS(91)
INTEGER LOFS(90)
INTEGER IPRF(90)
INTEGER JPRF(90)
INTEGER KBHSH(6)
INTEGER JFETUR(20)
INTEGER SPARE(5)
INTEGER BMAX
INTEGER BPL
INTEGER CHDIR(15, 60)
INTEGER CURREM
INTEGER PK(18)
REAL BB(6)
REAL CC(6)
REAL RATIO(12)
REAL CFOUR(60, 15), PHI(60, 15)
REAL AFOUR(60, 15), BFOUR(60, 15)
C EQUIVLAENCE (KPAR,3),OFS(1))
C EQUIVALENCE (SPAR(2),BPL)
C EQUIVALENCE (CFour(l-1),AFour(l,l))
C EQUIVALENCE (PHI(l,l),BFOUR(l,1))
C EQUIVALENCE (NL,NLO)
C EQUIVALENCE (NUMOB,11)
C
DATA ND/0/
DATA NDCHR/0/
DATA DGROUP/60.*.FALSE./
DATA DGROUP/60.*.FALSE./
DATA FGROUP/60.*.FALSE./
C CALL AFILC(0BUF,1,'RCR',13,13)
CALL OPEN(0BUF,1024,1,0,'RCR')
CALL GLABEL(0BUF,SPAR,IB)
C CALL RPARAM(KP.1,KPAR,32)
C READ RESULTS OF CONVENTIONAL CLASSIFICATION AND
C EXTRACT GROUP CLASSIFICATIONS
C D-GROUP
C CALL GET(0BUF,3,IB)
CALL MVL(0BUF(127+IB),LOFS(1),182)
C C
C DO 184 I=37,42
IIDCHR(I-36)=O
LLOFS=LOFS(I)
IF(LLOFS.EQ.0) GO TO 184
DGROUP(ILOFS)=.TRUE.
IIDCHR(I-36)=ILOFS
184 CONTINUE
C C B-GROUP
C DO 182 I=13,16
IIDCHR(I-12)=O
LLOFS=LOFS(I)
IF(LLOFS.EQ.0) GO TO 182
BGROUP(ILOFS)=.TRUE.
IIDCHR(I-12)=ILOFS
182 CONTINUE
C C E-GROUP
C DO 1850 I=45,46
LLOFS=LOFS(I)
IF(LLOFS.EQ.0) GO TO 1850
EGROUP(ILOFS)=.TRUE.
1850 CONTINUE
D 1851 I=49,52
LLOFS=LOFS(I)
IF(LLOFS.EQ.0) GO TO 1851
EGROUP(ILOFS)=.TRUE.
1851 CONTINUE
C C F-GROUP
C DO 186 I=55,58
IFCHR(I-54)=O
LLOFS=LOFS(I)
IF(LLOFS.EQ.0) GO TO 186
FGROUP(ILOFS)=.TRUE.
IFCHR(I-54)=ILOFS
186 CONTINUE
C C G-GROUP, INCLUDING Y
DO 107 I=65,70
   IFICHR(I-64)=0
   ILOFS=LOFS(I)
   IF(ILOFS.EQ.0) GO TO 107
   GROUP(ILOFS)=.TRUE.
   IF(1) CONTINUE
C GET CHROMOSOME DIRECTORY
CALL GET(OBUF, 1, IB)
   CALL MVL(OBUF(IB+3), SP10, 12)
   MCHR=T2(OBUF(IB+17))
   CALL MVL(OBUF(IB+31), CHD(I, 1), 2*15+30)
   CALL GET(OBUF, 2, IB)
   CALL MVL(OBUF(IB+31), CHD(I, 31), 2*15+30)
C JAY=0
   DO 310 IX=1.6
   II=I1ICHRI<IX>
   IF(DGROUP(II)) GO TO 304
   GO TO 310
304 CONTINUE
C JFETUR(1) = 0
C CUROLN=CHD(I, II)
   NL=CHD(2, II)
   NS=CHD(3, II)
   IF(NL.LT.1 OR. NS.LT.1) GO TO 310
   NG=0
   LPG=1024/NS
C LINES=GET.
   DO 300 J=1, NL
   CALL GET(OBUF, CUROLN, IB)
   IF(L.J.EQ.1) LASTIB=IB
   CUROLN=CUROLN+1
   DO 380 III=1, LPG
   NG=NG+1
   CALL SS(III, JAY)
   DO 330 J=1, NS
   LRBUF(J, NG)=OBUF(J+IB)
330 CONTINUE
   IB=IB+NS
   IF(NG.GE.MH)GO TO 311
300 CONTINUE
311 CONTINUE
C MAIN LOGIC
C IF(NOT DGROUP(I)) GO TO 401
C ML2=NL/2
   LTOP1=2
   LTOP2=ML2-1
   LBOT1=ML2+1
   LBOT2=NL-1
C ISUMT=0
   ISUMB=0
   NTOP=0
   HBOT=0
C DO 411 J=LTOP1.LTOP2
   DO 410 I=1, NS
   IVAL=LRBUF(I+1)
   IF(I.EQ.0) GO TO 410
   ISUMT=ISUMT+IVAL
   NTOP=NTOP+1
410 CONTINUE
411 CONTINUE
CONTINUE
ITOP=ISUMT/NTOP
C
DO 421 J=LBOT1,LBOT2
DO 420 I=1,NS
IVAL=LRGBUF(I,J)
IF(IVAL .EQ. 0) GO TO 420
ISUMB=ISUMB+IVAL
NBOTI=NBOTI+1
420 CONTINUE
C
NBOT=NBOT+1
QWT(ND)=FLOAT(ITOP)/FLOAT(NBOT)
DPOINT(ND)=11
C
401 CONTINUE
318 CONTINUE
C
DO FOR B-GROUP
C
C
DO 350 IX=1,4
II=10.CHRI(IX)
IF(II .EQ. 0) GO TO 350
C
C UR0LN=CHDIR(1,II)
NL=CHDIR(2,II)
NS=CHDIR(3,II)
IBC=CHDIR(8,II)
NC=0
LPC=1824/NS
C
DO 1800 I=1,NL
CALL GET(obuf, CUR0LN, IB)
CUR0LN=CUR0LN+1
DO 1801 III=1,LPC
NG=NG+1
DO .1320 J=1,NS
LRCBUF(J,NG)=OBUF(J+IB)
1820 CONTINUE
ID=IB+NS
IF(NG.GE.NL) GO TO 1811
C
1801 CONTINUE
1800 CONTINUE
1811 CONTINUE
C
C EXTRACT THE SO-CALLED B-BAND
C
C
ISTRT=FLOAT(IBCT+NL)/100.0
ISTRT=NL-ISTART
IEND=ISTART+IEND-1
NL2=NL/2
IF(IEND.GT.NL2) IEND=NL2
C
NB=0
BIOB=0.0
DO 351 I=ISTART, IEND
DO 352 J=1,NS
IVAL=LRGBUF(J, I)
IF(IVAL .EQ. 0) GO TO 352
DIX=0.BIOB+FLOAT(IVAL)
NB=NB+1
352 CONTINUE
351 CONTINUE
350 CONTINUE
C
CLASSIFY THE B-GROUP
C DO 360 I=1,4
DENMAX=0.0
DO 361 J=1,4
IF(BD(J).LE.DENMAX) GO TO 361
I0BJ=IBCHR(J)
IF(I0BJ.LT.1) GO TO 361
DENMAX=DB(J)
JS=J
361 CONTINUE
OFS(12+1)=I0BJ
BD(J)=0.0
360 CONTINUE
C DO FOR F-GROUP
C DO 450 IX=1,4
IF(II.EQ.0) GOTO 450
C
CUROLN=CHDIR(1,II)
NL=CHDIR(2,II)
NC=CHDIR(3,II)
NG=0
LPG=1024/NS
C DO 2389 I=1,NL
CALL GET(obuf,CUROLN,IB)
CUROLN=CUROLN+1
DO 2381 III=1,LPG
NG=NG+1
DO 2389 J=1,NS
LRCBUF(J,NC)=obuf(J+IB)
2389 CONTINUE
IB=IB+NS
IF(NG.LE.NL) GO TO 2311
2381 CONTINUE
2389 CONTINUE
2311 CONTINUE
C EXTRACT INTEGRATED OPTICAL DENSITY
G10D=0.0
MPG=0
DO 451 I=1,NL
DO 452 J=1,NS
IVAL=LRCBUF(J,I)
IF(<IVAL.EQ.0) GOTO 452
GIOB=GIOB+FLOAT(IVAL)
G10D=G10D+FLOAT(IVAL)
452 CONTINUE
451 CONTINUE
GB(IX)=GIOB/FLOAT(NPG)
C 458 CONTINUE
C CLASSIFY THE F-GROUP
C DO 460 I=1,4
DENMAX=0.0
DO 461 J=1,4
IF(GB(J).LE.DENMAX) GO TO 461
I0BJ=IFCHR(J)
IF(I0BJ.LT.1) GO TO 461
DENMAX=GB(J)
JS=J
461 CONTINUE
OFS(59+1)=I0BJ
GB(J)=0.0
460 CONTINUE
C DO FOR G-GROUP
C DO 510 IX=1,4
II=1GCHR(IX)
IF(II.EQ.0) GOTO 510
C
C

CUROLN=CHDIR(1,II)
NL=CHDIR(2,II)
HS=CHDIR(3,II)
NG=0
LPC=1024/HS

C DO 1330 J=1,NS
CALL CET(OBUF, CUROLN, IB)
CUROLN=CUROLN+1
DO 1301 II=1,LPC
NG=NG+1
DO 1330 J=1,NS
LRGBUF(J,NG)=OBUF(J+IB).
1330 CONTINUE
IB=IB+NS
IF((NG,CE,HL)) GO TO 1311
1300 CONTINUE
1311 CONTINUE
C
C EXTRACT DENSITY PROFILES
C
DO 500 K=1,NL
HP=0
IPROF(K)=0
DO 504 L=1,HS
IVAL=LRGBUF(L,K)
IF(IVAL.EQ.0) GO TO 501
IPROF(K)=IPROF(K)+IVAL
501 HP=HP+1
500 CONTINUE
IPROF(K)=IPROF(K)/HP
C
C ANALYZE THE BANDS IN THE C-GROUP
C
DO 511 K=1,NL
511 JPROF(K)=0

C

NLM2=NL-2
DO 512 K=3,NLM2
IV=0
DO 513 KK=1,5
IV=IV+IPROF(K+KK-3)
513 CONTINUE
JPROF(K)=IV
512 CONTINUE
MAX=JPROF(K)
DO 514 K=4,NLM2
IF(MAX.GE.JPROF(K)) GO TO 514
MAX=JPROF(K)

C
KBAND=K
514 CONTINUE
C
NSAME=0
DO 515 K=KBAND,NLM2
IF(JPROF(K).GT.MAX) GO TO 516
NSAME=NSAME+1
515 CONTINUE
NSAME=NSAME+1
516 CONTINUE
KBAND(X)=KBAND+NSAME/2
C

C CLASSIFY THE C-GROUP
C
DO 521 I=1,4
MAXBP=0
DO 522 J=1,4
IF(KGBAND(J).LE.MAXBP) GO TO 522
I=ICHR(J)
IF(I.UB.LT.1) GO TO 522
MAXBP=KGBAND(J)
JS=JS+1
522 CONTINUE
521 CONTINUE

522 CONTINUE
OFS(43-I)=IOBJ
KGBAND(JS)=0

521 CONTINUE
C
C CLASSIFY THE D-GROUP
C
DO 681 I=1,6
RMAX=0.0
DO 682 J=1,6
IF(RATIO(J).LE.RMAX) GO TO 682
IBO=IBPOINT(J)
IF(IBO.LT.1) GO TO 682
RMAX=RATIO(J)
JS=J

682 CONTINUE
WCHRP=HIPHR+1
OFS(43-I)=IOBJ
RATIO(JS)=O.0

681 CONTINUE
680 CONTINUE
C
1000 FORMAT(110)
2000 FORMAT(3X,3F9.3)
3000 FORMAT(1CHROMOSOME ',12)
4000 FORMAT(1F16.2)
C
CALL WPARAN(92,KPAR,9)
CALL CLOSE(0BUF)
CALL APhase(9)
C
CALL EXIT
END
C************
C
C ACHROM
C
C************
C
ROUTE TO ASSESS NORMALITY OF CHROMOSOMES IN A SPREAD
C
SUBROUTINE ACHROM
COMMON /C1/ MOB.IDIR
COMMON /C1/ CHDIR,SMDBUF,LRGBUF,HS,ML,CUROLN
COMMON /C1/ SST,EST
COMMON /C1/ SPLIT1,SPLIT2,SPLAREA
COMMON /C1/ FCHR
COMMON /C1/ CFOUR,PHI
COMMON /C1/ BAND,AHIS
COMMON /C1/ OBUF,OBUF
BYTE LRGBUF(50,90)
BYTE OBUF(2124)
BYTE TEMPO
LOGICAL OPT1,OPT2,OPT3
INTEGER IPROF(90)
INTEGER UFLOW(20)
INTEGER TEMPI
INTEGER SPAR(5)
INTEGER BMX
INTEGER BAND(180)
INTEGER BPL
INTEGER CMBIR(15,60)
INTEGER CUROLN
C
INTEGER SST(76),EST(76)
INTEGER PAR(10)
REAL CFOUR(60,15),PHI(60,15)
REAL AHIS(180)
REAL COSINE(150),SINE(150)
REAL AFOUR(60,15),BFOUR(60,15)
C
C
EQUIVALENCE(SPAR(2),BPL)
EQUIVALENCE(CFOUR(1,1),AFOUR(1,1))
EQUIVALENCE (PHI(1,1), BFQUR(1,1))
EQUIVALENCE (NL, NL0)
EQUIVALENCE (HUM0B, II)
DATA LOP1/'T1'/
DATA LOP2/'T2'/
DATA LOP3/'T3'/
DATA LOP4/'OP'/
C
DO 10 CONTINUE
C
CALL FILE(OBUF,1,'RCE ','*13,*13)
CALL OPEN(OBUF,1024,1,0,'RCE')
CALL GLABEL (OBUF, SPAR, IB)
C
CALL GET (OBUF, IB, 12)
CALL MYL/OBUF(IB+3), SPI0B.12
NCHR=IV(OBUF(IB+17))
CALL MYL/OBUF(IB+31), CHDIR(1,1), 2*15+30
CALL GET (OBUF,2, IB)
CALL MYL/OBUF(IB+31), CHDIR(1,31), 2*15+30
C
DO 310 II=1,NCHR
JFETUR(1)=0
C
WRITE(5, 3000)II
C
CONTINUE
C
LASTIB=IB
CURLBN=CHDIR(1, II)
NL=CHDIR(2,II)
HS=CHDIR(3,II)
IF (NL.LT.1 .OR. HS.LT.1) GO TO 310
NG=0
LPG=1024/NS
C
DO 392 III=1,NPG
C
CONTINUE
C
CALL GET (OBUF, CUR0LH, IB)
IF (III.EQ.1) LASTIB=IB
CUR0LH=CUR0LH+1
DO 392 III=1,LPG
NG=NG+1
CALL SSITCH(9, JAY)
C
IF (JAY.EQ.1) WRITE(5, 3000) (OBUF(J+IB), J=1, NS)
C
CONTINUE
DO 330 J=1,NS
LRGBUF(J, NG)=OBUF(J+IB)
C
CONTINUE
IB=IB+NS
IF (NG.GE. NL) GO TO 311
C
CONTINUE
C
CONTINUE
C
CONTINUE
C
NSM1=NS-1
DO 500 K=1, NL

MAXLOC=1
LINMAX=LRGBUF(1, K)

DO 510 L=2, NS
IF(LRGBUF(L,K).LE.LINMAX) GO TO 511
LINMAX=LRGBUF(L,K)
MAXLOC=L
511 CONTINUE
510 CONTINUE

JTH=LINMAX/2
IPROF(K)=NS

DO 530 L=1, NS
IF(LRGBUF(L,K).GE.JTH) GO TO 532
IPROF(K)=IPROF(K)-1
530 CONTINUE
532 CONTINUE

IF(IPROF(K).GE:NS) IPROF(K)=0

CONTINUE

IPTOT=0
MLM2=NL-2
DO 600 K=3, MLM2
IPTOT=IPTOT+IPROF(K)
600 CONTINUE

IPAVE=IPTOT/MLM2

DO 620 K=3, MLM2
IPROF(K)=IPROF(K)-IPAVE
IF(IPROF(K).LT.0) HLTO=HLTO+1
620 CONTINUE

IPROF(1)=1
IPROF(2)=1
IPROF(NL)=1
IPROF(NL-1)=1

IF(JAY.EQ.1) WRITE(5, 1306) (IPROF(L), L=1, NL)

TYPE 1 DETECTION-
AT LEAST FIVE NEG NUMBERS IN A ROW IN PROFILE

TYPE 2 DETECTION-
AT LEAST FOUR OUT OF A CONTIGUOUS SEVEN NUMBERS IN PROFILE ARE NEGATIVE BUT NOT FIVE CONTIGUOUS, OR THREE CONTIGUOUS.

NOTE THAT THE SEARCH FOR TYPE 1 TAKES PRECEDENCE, EXCEPT THAT TYPE 2 SPANS A WIDER INTERVAL. THE SEARCH IS STRICTLY ONE DIRECTIONAL, FROM THE TOP OF THE PROFILE ARRAY TO THE BOTTOM. THIS SEARCH PATTERN FEATURES SOME AMBIGUITY, BUT NOTHING SERIOUS.

TYPE A CENTROMERE-
SKEJED, LOWEST DIP IN PROFILE IS FIRST OR LAST ONLY IN SEQUENCE OF NEG NUMBERS THAT INDICATE THE CENTROMERE. THIS IS EXPECTED TO OCCUR NEAR THE END (TOP) OF THE CHROMOSOME.

TYPE B CENTROMERE-
SYMMETRIC, LOWEST DIP OCCURS CORRESPONDINGLY POSITIONED IN THE PROFILE TO THE CENTER OF THE CENTROMERE IN THE CHROMOSOME.

K=0
IFETUR=0
K=K+1

IF(K.GT.NL-4) GO TO 750
IF(IPROF(K).GE.B) GO TO 720
IF(IPROF(K+1).GE.B) GO TO 710
IF(IPROF(K+2).GE.B) GO TO 710
IF(IPROF(K+3).GE.B) GO TO 710
IF(IPROF(K+4).GE.B) GO TO 710

IFETUR=IFETUR+1
KFETUR=1
JFETUR(IFETUR)=KFETUR

K=K+4
K=K+1

IF(IPROF(K+1).LT.0) GO TO 701
GO TO 750

CONTINUE

MINUS=1
DO 720 KL=1,6
IF(K.GT.KL-4) GO TO 721
IF(IPROF(K+K).GE.B) GO TO 720
MINUS=MINUS+1
720 CONTINUE

721 CONTINUE

IF(MINUS.LT.4) GO TO 700

IFETUR=IFETUR+1
KFETUR=2
JFETUR(IFETUR)=KFETUR
K=K+4
K=K+1

IF(IPROF(K+1).LT.0) GO TO 731
GO TO 750

CONTINUE

WRITE(5,4900) (JFETUR(JF),JF=1,IFETUR)

FORMATT(' FEATURES: ',1,2031)
C IF(NLTB.GT.NL.3) GO TO 940
GO TO 941
940 CONTINUE
C WRITE(5,1894)
FORMAT(' NOT A CHROMOSOME')
C GO TO 950
941 CONTINUE
IF(IFETUR.EQ.0) GO TO 988
IF(IFETUR.EQ.1) GO TO 910
IF(IFETUR.EQ.2) GO TO 920
IF(IFETUR.EQ.3) GO TO 930
C GO TO 940
930 CONTINUE
GO TO 950
920 CONTINUE
IABIND=JFETUR(1)+JFETUR(2)
IF(IABIND.EQ.2) WRITE(5,1292)
IF(IABIND.EQ.3) WRITE(5,1352)
IF(IABIND.EQ.4) WRITE(5,1492)
1292 FORMAT(' CHROMOSOME IS BICENTRIC, OR BADLY TWISTED')
1392 FORMAT(' CHROMOSOME MAY BE BICENTRIC')
1492 FORMAT(' CENTROMERE LOCATION DOUBTFUL - PROBABLY NORMAL')
GO TO 950
C 910 CONTINUE
IF(JFETUR(1).EQ.1) GO TO 911
WRITE(5,1291)
GO TO 912
911 WRITE(5,1191)
912 CONTINUE
1191 FORMAT(' SINGLE CENTROMERE IS WELL DEFINED - NORMAL')
1291 FORMAT(' SINGLE CENTROMERE POORLY DEFINED - PROBABLY NORMAL')
GO TO 950
C 900 CONTINUE
IF(NLTB.GT.30) GO TO 901
WRITE(5,1932)
GO TO 903
901 CONTINUE
WRITE(5,1901)
903 CONTINUE
1901 FORMAT(' NO CENTROMERE FOUND. />
1 ' COMMON IN SMALL CHROMOSOMES - PROBABLY NORMAL')
1292 FORMAT(' NO CENTROMERE FOUND - POSSIBLE ACENTRIC')
C C C
C 950 CONTINUE
C C C
C C C
C C C
C C
C 310 CONTINUE
WRITE (5,7030)
CALL APHASE(+)
CALL CLOSE(0BUFF)
C C 1000 FORMAT(1X)
2000 FORMAT(1X,3913)
3000 FORMAT(1X,13)
3030 FORMAT(1X)
CALL EXIT
END
We claim:

1. An operator assisted system using a programmed computer for producing karyotype images from a slide mounted on the stage of a microscope, said slide having a plurality of dispersed metaphase chromosome spreads, comprising:
   - moving said dispersed metaphase chromosome spreads to the optics of said microscope,
   - automatically prescanning said slide for chromosome spreads,
   - arranging the chromosomes in the digital picture of an acceptable chromosome spread, and
   - producing a visible image of the karyotype format for each acceptable chromosome spread.

2. A system as recited in claim 1 wherein said step of producing a visible image of the karyotype format for each acceptable chromosome spread includes:
   - displaying said karyotype format on a gray scale display,
   - correcting any errors to provide a corrected karyotype format, and
   - printing said corrected karyotype format.

3. A system as recited in claim 1 wherein said step of generating a digital picture of each of said accepted chromosome spreads includes:
   - moving said stage under instructions of said programed computer to place the slide motionless on said stage at each successive chromosome spread location, thereby enabling an operator to view each successive chromosome spread to inspect the chromosome spread and determine whether it is acceptable for the purpose of analysis or not,
   - automatically focusing the optics of said microscope on each successive chromosome spread to inspect the chromosome spread and determine whether it is acceptable for the purpose of analysis or not,
   - deleting under operator control the location of each spread determined not to be acceptable thereby establishing a list by location of which the chromosome spreads are acceptable,
   - again moving said stage under instructions of said programed computer for generating a digital picture of each of said acceptable chromosome spreads,
   - arranging the chromosomes in the digital picture of an acceptable chromosome spread in a karyotype format, and
   - producing a visible image of the karyotype format for each acceptable chromosome spread.

4. A system as recited in claim 3 wherein said step of generating a digital picture of each of said accepted chromosome spreads includes:
   - successively moving said stage to the undeleted locations of accepted chromosome spreads to position each such accepted chromosome spread under the microscope optics,
   - automatically focusing the optics of said microscope on an accepted chromosome spread positioned thereunder,
   - photographing each said accepted chromosome spread through said focused microscope optics in a manner to generate successive picture elements, and
   - digitizing each picture element generated in photographing a chromosome spread.

5. A system as recited in claim 1 wherein said step of arranging the chromosomes in the digital picture of an acceptable chromosome spread in a karyotype format includes:
   - locating each chromosome in said digital picture of an acceptable chromosome spread,
   - orienting each chromosome in a predetermined direction,
   - measuring each oriented chromosome,
   - classifying the oriented chromosome on the basis of measured length and centromeric index, and
   - composing the karyotype format from the classified and oriented chromosomes.

6. A system as recited in claim 5 wherein after the step of locating each chromosome in said digital picture of an acceptable spread there is included the step of assigning a different number to each chromosome which has been located.

7. A system as recited in claim 8 wherein said step of composing the karyotype format from the classified and oriented chromosomes includes:
   - displaying said composed karyotype format on a gray scale display, and
   - correcting any chromosome orientation and classification errors.

8. A system as recited in claim 4 wherein the step of producing a visible image of the karyotype format includes:
   - converting the digitized picture elements into analog picture elements, and
   - printing a picture responsive to said picture elements.

9. An operator-assisted system using a programmed computer for producing karyotype images from a slide mounted on the stage of a microscope, said slide having a plurality of dispersed metaphase chromosome spreads, comprising:
   - means for moving said stage under instructions of said programmed computer connected to control said microscope to successively present said plurality of dispersed metaphase chromosome spreads to the optics of said microscope,
   - means for storing the location of each spread, thereby automatically prescanning said slide for chromosome spreads,
   - means for moving said stage under instructions of said programmed computer to place the slide motionless on said stage at each successive chromosome spread location, thereby enabling an operator to view each successive chromosome spread to inspect the chromosome spread and determine whether it is acceptable for the purpose of analysis or not,
   - means for determining under operator control the location of each spread determined not to be acceptable thereby establishing which of said chromosome spreads are acceptable,
   - means under instructions of said programmed computer for generating a digital picture of each of said acceptable chromosome spreads,
   - means for arranging the chromosomes in a digital picture of an acceptable chromosome in a karyotype format, and
   - means for producing a visible image of the karyotype format.
format for each acceptable chromosome spread.

10. A system as recited in claim 9 wherein said step of producing a visible image of the karyotype format includes
means for displaying said karyotype format on a gray scale display,
means for correcting any errors to provide a corrected karyotype format, and
means for printing out said corrected karyotype format.

11. A system as recited in claim 9 wherein means for moving said stage under said microscope to successively present said plurality of dispersed metaphase chromosome spreads to the optics of said microscope includes
means for moving said stage with a scanning motion in successive predetermined motion increments underneath the optics of said microscope,
means for detecting whether or not a chromosome spread is present within the distance of the predetermined increment of motion, and
means for storing the location data for each detected chromosome spread.

12. A system as recited in claim 9 wherein means for generating a digital picture of each of said accepted chromosome spreads includes,
means for successively moving said stage to the undeleted locations of accepted chromosome spreads to position each such accepted chromosome spread under the microscope optics,
means for automatically focusing the optics of said microscope on an accepted chromosome spread positioned thereunder,
means for photographing each said accepted chromosome spread through said focussed microscope optics in a manner to generate successive picture elements, and
means for digitizing each picture element generated in photographing a chromosome spread.

13. A system as recited in claim 9 wherein the means for locating each chromosome in said digital picture of an acceptable spread includes
means for assigning a different number to each chromosome which has been located.

14. A system as recited in claim 9 wherein means for producing a visible image of the karyotype format includes
means for converting the digitized picture elements into analog picture elements, and
means for printing a picture responsive to said picture elements.

15. A system as recited in claim 9 wherein said means for arranging the chromosomes in a digital picture of an acceptable chromosome spread in a karyotype format comprises
computer means programmed for locating each chromosome image in an acceptable digital spread, for orienting each chromosome image in a predetermined direction, for measuring each oriented chromosome image for classifying the oriented chromosome images, and for composing the karyotype format from the classified and oriented chromosome images.