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
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 a 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 potency of karyotype may detect certain inherited disorders long before clinical symptoms appear. As the potency of karyotyping 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 can point 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 stage 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 computer, 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 Cytchemistry 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 driving 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, causing it to count down towards zero. These pulses from the pulse generator 81 are also applied to all of the And 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 And 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 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 disenabling 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 the 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 30 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 30 and the image digitizing system 36. This equipment is commercially available and, by way of example, in an embodiment of the invention which was constructed, 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 interaction 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 identification 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 Symposium 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 "partitions," 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 CALMS EXIT 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 NSDS contains F (F>0) the first F bytes contain the data set numbers (1=S1, 2=S2, etc) ordered by time of scan.
- **HDS**: BLKB NHDS each byte gives the status of a particular data set:
  - 0 = available
  - 1 = in use by mask
  - 1 = ready for hardcopy output

```
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).

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."

Flow Chart 1
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,
if search start is pressed. 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 focus 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 disk, display it on the gray scale, and calculate sector thresholds.

SCAN uses the SDS data camera to scan a picture onto a disk. The picture is scanned by columns, with alternating even and odd TV fields, to achieve the minimum scan time of 17 seconds. Each column is assembled in core and output as a line on disk and on the gray scale. The coils on the SDS monitor have been rotated so that the orientation is the same as on the gray scale monitor.

Sector histograms are accumulated as the picture is scanned, and sector thresholds are calculated at the end of each row of sectors. The thresholds are typed out if SW1 is up. The thresholds are written after the last picture line.

SCAN sets priority 6 during the pixel digitization loop to prevent interrupts from other devices which would cause it to lose "sync." Hardcopy interrupts (which are at priority 7) are the only ones allowed, due 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 2×2 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 500×480 pixels is reduced to a 250×240 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.
Fa.:

FIRST RECORD FOR OBJECT

NUMBER OF END POINTS

INTERRG DIR (512)

FBW1 N1 YMIN1 XMIN1 YMAX1 XMAX1
FBW2 N2 YMIN2 XMIN2 YMAX2 XMAX2
FBW3 N3 YMIN3 XMIN3 YMAX3 XMAX3

INTERRG EDGE (512)

\[
\begin{array}{cccccc}
Y_1 & X_{11} & Y_1 & X_{12} \\
Y_2 & X_{21} & Y_2 & X_{22} \\
Y_3 & X_{31} & Y_3 & X_{32} \\
\vdots & \vdots & \vdots & \vdots \\
Y_N & X_{N1} & Y_N & X_{N2} \\
\end{array}
\]

FLOW CHART 5 Edge File Format

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 rethresholding 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
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.

CHROME

FLOW CHART 8

Chrome File Format
Purpose: To rethreshold and resegment the chromosomes and to store them in a chromosome file. NOB 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 display the picture on the gray scale, and calls NOB reads 122 words of parameters into NK, LT(60), ST(60), and FI. If FI is zero, NOB does not display the picture on the gray scale, and calls A PHASE (MASKPHI).

NK is the number of chromosomes and LT and ST are the lines and samples where the numbers are to be placed.

NOB reads one line at a time and adds any numbers required on that line. If LT is in order, NOB runs a little faster since it doesn’t have to search the entire table for each line.

INT1

Purpose: To correct errors in object isolation.

INT1 is called immediately after each spread is scanned. The operator normally requests a quick count at this time. If the spread is only to be used for a count, the operator can correct for missing or extra objects and finish it.

If the spread is to be karyotyped, cuts, joins, and threshold changes are needed to correct for errors in object isolation. After the objects have been isolated, INT1 is called again to check for any remaining errors.

INT1 communicates with CALMS by calling A PHASE. It writes parameters to BINARY to indicate object isolation. After the objects have been isolated, INT1 is called again to check for any remaining errors.

MOB

Purpose: To orient and measure the chromosomes. MOB orients each input object, accumulates IOD, area and length and calculates centromeric index by length, IOD and area. The unrotated chromosome file (UCR) is MOB’s input (which is ROB’s output) and the rotated chromosome file (RCR) is MOB’s output. The format of RCR is as follows:


Rec. 2: Words (16-465) = Chromosome directory

Rec. 3: Reserved for CLASFY results

Rec 4-7: Reserves for BANDS results

Rec 8 & following records: Roasted chromosome images

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

<table>
<thead>
<tr>
<th>Integer</th>
<th>CHDIR (15, 60)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHDIR (1, 31)</td>
<td>CHDIR (15, 30) on Rec 1</td>
</tr>
<tr>
<td>CHDIR (15, 60)</td>
<td>on Rec 2</td>
</tr>
</tbody>
</table>

Each entry is as follows:
MOB will reject objects if they are too large or too small. The maximum allowable size before rotation is 88 \times 88. The maximum allowable size after rotation is 88 \times 48. The maximum area is 2000 points, and minimum area is 30 points. The minimum length and width are 5 lines and 5 samples. When an object is rejected, its area, length, and width are typed.

Originally, MOB had an option to use a skeleton method in locating centromeres. However, this method took 25 sec. longer with little or no improvement in centromere accuracy. This option was removed in order to save core space and allow a larger maximum chromosome size.

**CLASSFY**

**Purpose:** To classify the rotated chromosomes into 10 groups (conventional classifier).

CLASSFY reads in the chromosome directory produced by MOB and classifies each object. Once classification is complete, CLASSFY writes the classification tables into RCR, record 3. The format is:

| Word (1) | # of samples in karyogram |
| Word (2) | No object flag |
| Word (3-12) | Group ID's |
| Words (13-102) | Slot ID's |
| Words (103-108) | Initial slot for each row |
| Words (109-199) | Object found in each slot |
| Words (200-204) | Center line table for each row |
| Words (205-294) | Center sample for each slot |

This table is then used by KTYPE to build the output karyogram.

Classification is based on an internal table that gives the minimum and maximum allowable centromeric indexes for each length, for each of the ten groups of chromosomes. The table is called CT and is dimensioned (20, 51). The twenty entries for each length are minimum and maximum CI for A-1, A-2, A-3, B, C+X, D, E-16, E-17 and E-18, F and G+Y.

First, an initial classification is made. Then, moves are made from "heavy" groups to "light" groups. When there are several candidates for a move, the one with "maximum likelihood" is chosen.

After all possible moves have been made, the chromosomes within each group are ordered according to the slope table which describes the slope of a line that sweeps in from the right. Most groups are ordered by size. When the C+X group has 15 or 16 members, the third largest or third and fourth largest are placed in the X slots. When G+Y has five members, the chromosome with the maximum fit factor is placed in the Y slot.

The classification table can be supplied to CLASSFY with the OS parameter, followed by the object numbers for the slots. In this case, CLASSFY 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 CLASSFY 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 CLASSFY to reclassify on the basis of the new measurements. After the operator starts making moves, INT2 tells CLASSFY 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.

**Format of KDATA: RECL=1024**

| Line 1 | Directory Record 1 for Source 1 spreads |
| Bytes | 1-2 | NEXT 1*2 | Line # of next Dir Rec for Source 1, or 0 |
| | 3-4 | NUSED 1*2 | # of spreads in this Dir Rec max in 85 |
| | 5-85 | PID (85) | BYTE 10 byte patient ID for each spread, or 0 |
| | 85-1024 | LINE (85) | 1*2 | Corresponding data line # for each spread |
| Line 2 | Directory Record 1 for Source 2 spreads |
| : | : | : | : |
| Line 15 | Directory Record 1 for Source 15 spreads |

When "NUSED" becomes 85, next available line # will be inserted into NEXT. Then the line in NEXT will be the next directory record for this source.

Line 16-500 Data Records and Directory Records, if necessary.
FOUR file, RCR. Length and centromeric index are passed to and saved in records along the line is used. The waveform values for eight harmonics. From these, the slope of the parabola. C (Amplitudes) and PHI (Phase Angle) are computed to the boundary points and recalculates the chromosome. The output is written on disk, with an option to also write it on tape if switch is up. The picture on disk is written in hardcopy format, and the raw spread for counts. It checks for the KG parameter followed by the disk unit and filename for the raw spread.

MASK 2

Purpose: To combine two pictures and add a border. MASK2 combines the numbered spread and karyotype into a single picture formatted for the hardcopy. The output is written on disk, with an option to also write it on tape if switch 5 is up.

The picture on disk is written in hardcopy format, with four bits for each element, and the picture on tape is written with eight bits for each element. Gray scales, reference masks and annotation are added to the picture.

For bent chromosomes, BAND curve fits a parabola along the parabola, and using three points perpendicular to the slope of the parabola. For straight chromosomes, 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.

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.

BAND recognizes the following parameters:

WA: Store a representation of the waveform in place of the rotated chromosome images.

AX: Mark the axis for each chromosome

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.
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(DBFLAG,- 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 RK11 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 MVB.

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>Meaning</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</td>
<td>Tape Output</td>
</tr>
</tbody>
</table>

0,1 and 2 are functionally equivalent.

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 MVB is defined as a byte array, sample J of the requested line is at MVB(J+INDEX). In Marco, MVB+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.
**GLABEL 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 link 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:

```fortran
JSR R4, SAVER
```

To restore registers 0-4 and return via R5:

```fortran
JMP RESTR
```

(SAVER is called via R4, and R5 is not saved.)

---

### Word by Byte Table

<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>1</td>
<td>2</td>
<td>+2</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>+6</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>+8</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>IBN</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>IBA</td>
</tr>
<tr>
<td>5</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>+6</td>
</tr>
<tr>
<td>13</td>
<td>30</td>
<td>CRC</td>
</tr>
<tr>
<td>14</td>
<td>32</td>
<td>-2</td>
</tr>
<tr>
<td>15</td>
<td>34</td>
<td>FB</td>
</tr>
<tr>
<td>16</td>
<td>36</td>
<td>+2</td>
</tr>
<tr>
<td>17</td>
<td>40</td>
<td>+4</td>
</tr>
<tr>
<td>18</td>
<td>42</td>
<td>+6</td>
</tr>
<tr>
<td>19</td>
<td>44</td>
<td>WNT</td>
</tr>
<tr>
<td>20</td>
<td>46</td>
<td>SBN</td>
</tr>
<tr>
<td>21</td>
<td>50</td>
<td>NBF</td>
</tr>
<tr>
<td>22</td>
<td>52</td>
<td>NLR</td>
</tr>
<tr>
<td>23</td>
<td>54</td>
<td>RECLEN</td>
</tr>
<tr>
<td>24</td>
<td>56</td>
<td>BLKSIZ</td>
</tr>
<tr>
<td>25</td>
<td>60</td>
<td>BUFSIZE</td>
</tr>
<tr>
<td>26</td>
<td>62</td>
<td>BFP</td>
</tr>
<tr>
<td>27</td>
<td>64</td>
<td>DBF/MD</td>
</tr>
<tr>
<td>28</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 BLKSIZ, RECLEN, NLR, and BFP fields in the MVB. GLABEL and PLABEL must be called 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, # 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 MVB to the first byte of the label, as returned by 'get.'

**Example:**

```
INTEGER SPAR(5)
BYTE A(4200), B(4200)
CALL OPEN (A,2048,1,1,'MV1')
CALL OPEN (B,2048,1,1,'MV2')
CALL GLABEL (A,SPAR)
CALL PLABEL (B,SPAR,A(IA + 1))
```

Open A for input

Open B for output

Get label from A

Put label to B

---

**Example:**

```
READ and WRITE can be used when RECLEN = BUFSIZE 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, starting at 0. TB = tran block, LB = link block, and FB = filename block.

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

Example:

```
Label is the location of the label to be output.

Index is the offset from MVB to the first byte of the record to be updated. (The same index value will be returned by 'get.'

PLABEL transfers the data in SPAR to the label. Index is the offset from MVB to the first byte of the label, as returned by 'get.'

Label is the location of the label to be output.
```

---

**Example:**

```
WORD BYTE CONTENTS
1 0 TB Active Block 
2 2 +2 Active Buffer Address
3 4 +4 Word Count
4 6 +6 Function/Status
5 10 +8 Words Not Transferred
6 12 IBN Inactive Block 
7 14 IBA Inactive Buffer Address
8 16 -2 Error Return
9 20 LB Link Pointer
10 22 +2 Logical Name
11 26 +6 Device Name
12 30 CRC Current Record 
13 32 -2 Indicator ( Unused
14 34 FB File Name (Word 1)
15 36 +2 File Name (Word 2)
16 40 +4 Extension
17 42 +6 UIC
18 44 WNT Words Not Transferred
20 46 SBN Start Block 
21 50 NBF # Blocks In The File
22 52 NLR # Label Records
23 54 RECLEN Record Length (Multiple or Divisor of BLKSIZ)
24 56 BLKSIZ Block Size
25 60 BUFSIZE Buffer Size (Multiple of BLKSIZ and RECLEN)
26 62 BFP Blocks Per Buffer
27 64 DBF/MD Mode / Double Buffering Flag
28 66 WFLAG Write Flag (Last Block # To Be Written + 1)
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 AFILE (A1, 'PIC ', 5,5)
Assigns DK1:PIC[5,5] to MVB A

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.

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 RTI.
PARAM will then call the waiting routine with a simulated interrupt, when it has finished with the earlier request.

The PBUSY, PINT method is required under DOS; otherwise the KB driver gets confused.

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 'ASCIZ', in MACRO, also does this.

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

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.

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)
CALL ITLA (INT, L1, N) (INTEGER TO LOGICAL array)
Store INT in 0 bytes starting at L1
CALL SWAP (L1, 12) Interchange L1 and 12.

Note: RO is not saved by these routines.

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, ..., )

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)
CALL HCUPK (BUF4, BUF8, NE)
CALL MVW78 (BUF4, BUF8, NW)
BUF8 is the LOC of 8-Bit Data
BUF4 is the LOC of 4-Bit Data
BUF7 is the LOC of 7-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. NPNUM 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 DECLARE 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 register addresses and interrupt vector locations.

CALL IXYABS (Y,Y) Initiates a relative X move
CALL IPABS (Y,Y) Initiates a relative Y move
CALL IXREL (XDEL) Initiates a relative X move
CALL IYREL (YDEL) Initiates a relative Y move
CALL LED (CODE,VAL) Puts VAL in the LED

FUNCTION: ROACH (WI,BI,WORD,EWI,MASK)
ROACH will scan a binary line to locate the first set bit. The scan will begin at WORD (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,WOD,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.

<table>
<thead>
<tr>
<th>CURRENT DIRECTION</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PREVIOUS DIRECTION</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEGMENT END POINT DECISION TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOW CHART 10 Segment and Point Decision Table</td>
</tr>
</tbody>
</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.
AREA is returned as the number of sample points contained in the chromosome.

ORIOB

Purpose: Orient objects.
CALL ORIOB(IBUF,OBUF,EF,CHDIR,NL,OPROT,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. OPROT specifies any additional rotation desired by the operator.

EPROT

Purpose: Rotates chromosome endpoints.
CALL EPROT(SIN,COS,EP,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,IBUF,OBUF,COS,SIN,IBUF,OBUF)
OBROT rotates the object in IBUF into OBUF. COS and SIN specify the rotation angle and XMIN,XMAX,YMIN,YMAX give the enclosing rectangle of the object in unrotated coordinates.

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.

CHROUT

Purpose: To stand objects up and move them into the output buffer.
CALL CHROUT(IBUF,OBUF,NS,NL,LPB)
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 for the current cursor position.
CALL KURSOR(RY,XL,SLID,SLCL,SLCS,N)
Y,X,LS are the cursor coordinates returned by cursor. SLID is the slot ID indicated; SLCL and SLCS are the slot center coordinates; and N is the object ID of the object residing in the slot.

. TITLE CALMS - CALMS SUPERVISOR
. LIST_BEX
. CALL CALL, EXIT, PAUSE, INIT, TRAN, WAIT, RLSE
. GLOBAL RLS, CVT, CBT, MAKE EMTS RESIDENT
. SUPervisor FOR THE CLINICAL AUTOMATED LIGHT MICROSCOPE SYSTEM
. FOR CHROMOSOME ANALYSIS.
. THE SUPERVISOR CONTROLS SLIDE SEARCH, SCANNING, ANALYSIS, AND HARD-COPY
. OUTPUT OF KARYOTYPES OR COUNTS.
. SEARCH AND HARD-COPY ARE PERMANENTLY RESIDENT BELOW LOCATION 60000.
. SCAN AND ANALYSIS CONSISTS OF 16 PHASES THAT RESIDE ON DISK IN CORE-MEMORY FORMAT. ONE PHASE AT A TIME IS LOADED AT LOCATION 60000 AND
. CALLED BY THE SUPERVISOR.

LF=12
PSU=177776
NPH=24
BINPH=2
INTPH=7
MDPH=8
MASKPH=13
MDS=3
NSDS=2
SBU=5
HBU=3
MVB=56
LABS=12
IBH=12
JB=14
CRREC=50
SBN=46
RELCEN=54
BUFS=60
DPB=62
DBFLAG=64
SUR =175780
MOVB R1, 'M,FILPEX' ;AG HN
MOV R2, 'M,FILPEX' ;HARDCOPY DISK UNIT
INC R2, -(R1) ;SET DATASET BUSY
MOV R1, CMS ;SAVE LCN OF CURRENT MASK DATASET
MOV R2, CSBH ;SAVE SBH LOCATION
CALL PHASE, <PHASE> ;AMESK2
MOV R1, '07,CMDM' ;SET DATASET FULL OF DATA
INC NHCOPY ;CALL LED, <HQLED, NHCOPY>
CALL LED, <AOLED, NIFS>
TSTB HCFSLG
BNE JS18 ;BR IF HCOPY IS OPERATING
INC HCFSLG ;SET HCOPY OPERATING
BIS R1, '011#LITES' ;TURN ON HCOPY LITE
MOV CMS, CHDS
MOV QCSBH, A+SBH ;SET UP START BLOCK NUMBER
MOV QPSV, -(SP) ;SAVE PSV TO SIMULATE TEMP
JSR PC,HCOPY ;CALL HCOPY
JS18. JMP R1

PHASE, MOV R2, <R5>, R1 ;PHASE START LOCATION
MOV R1, PHNUM ;STORE PHASE NUMBER
INC R1

MOVB R1, 'NPHASE' ;PHASE=PHNUM+1
CALL ZIA, <157998L,377> ;ZERO 510 BYTES OF UPPER CORE
CALL READ, <MYB, PHNUM, INDEX.PLOC> ;READ PHASE PHNUM INTO PLOC
ASL R1
ADD #PLITE-4, R1
BIS R1, '011#LITES' ;TURN ON THE LITE FOR THIS PHASE
CALL PLOC, <DUNIT,FILPEX> ;CALL THE PHASE, GIVING IT A DATA SET NAME
BIC #141796#LITES ;TURN OFF ANALYSIS LITES
RTS R5

PPAR. ;CALL PARAM. <NP, PAR, MAXNP> ;JSR PC, PPAR
DEC NP

BNE PP16 ;BR IF NP WAS 0
CRPB PARBUF, #Y
BNE PP15 ;BR IF NO RESTART

PP16. ;CALL TYPE, <NP, SOURCE> ;ASK ABOUT SCAN RESTART
CALL PARAM, <NP, PAR, MAXNP>
CRPB PARBUF, #Y

BNE PP95 ;BR IF NO SCAN RESTART
CALL RPARAM, <NP, SOURCE, L512, ONE> ;READ RESTART INFO
CALL TYPE, <NP, ZERO> ;ASK FOR FIRST SPREAD NUMBER
CALL PARAM, <NP, PAR, MAXNP>
MOV PAR, R0
CMP R0, 61

BLE PP92 ;BR IF NUMBER LE 1
CMP R0, HSCAN

BGT PP02 ;BR IF GT HSCAN

DEC R0 ;LEGAL NUMBER, DECREMENT IT
MOV R0, SPH

INC R0, NSCAN

ASL R0

ASL R0, N=4
ADD R0, CSPQ

ADJUST CSPQ

CALL LED, <SOLED, NSCAN>

PP95. ;CALL TYPE, <M5, ZERO> ;ASK ABOUT ANALYSIS RESTART
CALL PARAM, <NP, PAR, MAXNP>
CRPB PARBUF, #Y
BNE PP15 ;BR IF NO ANALYSIS RESTART

MOVB R0, 'M,PHASE' ;PHASE=0
MOV *HSDS, R0
MOV *OSDS, R1
MOVI  R0, (RI)+
SUB  R0, PPI0
CALL TYPE, (R0, ZERO) ; ASK HOW MANY
CALL PARAM, (NP, PAR, MAXHP)
TST  PAR
BLE PPI2 ; BR IF N LE 0
CMP  PAR, 0HSDS ; BR IF N GE HSDS
MOV  PAR, HSDDS ; STORE HSDDS
PPI2: CALL LED, (AQ, LED, HSDDS) ; DISPLAY ANALYSIS QUEUE
PPI5: TSTB HFLG
BNE PPEX ; BR IF HCOPY RUNNING
CALL TYPE, (R0, ZERO) ; ASK ABOUT HARDCOPY RESTART
CALL PARAM, (NP, PAR, MAXHP)
CMPB PARBUF, 0'B
BNE PPEX ; BR IF NO HARDCOPY RESTART
SUB  R0, HSDDS, R0
MOV  @HBS, R0
CALL HFLG
BS 01, HLLITES ; TURN ON HARDCOPY LITE
MOV  @PSV, -5(SP)
JSR PC, HLED ; SIMULATE INTERRUPT TO HCOPY END RTE
PPEX: RTS  PC
MOV  PAR, R0, 0PHASE NUMBER
BLE PERR
CMP  R0, 0MPH
BGT  PERR
ASL  R0
ADD @HMTAB-2, R0 ; DBHTAB+2*PKUM-2
MOV  HP, R0 ; STORE HP FOR THE PHASE
CALL UPARAM, (NP, PAR+2, PAR)
JPPAR: JMP PPAR
PPAR: CALL TYPE, (PENMSG, ZERO)
BR  JPPAR
THSCN: TST MSGN ; JSR PC, THSCN
BNE CGHS ; BR IF MSGN HLT ZERO
CALL TYPE, MSG1 ; TYPE 'OK TO START SEARCH'
CALL SEARCH ; ALLOW SEARCH START
RTS  PC
CGHS: CALL GNSTS ; GET NEXT SPREAD TO SCAN
RTS  PC
MOVB: .BLKW 10 ; MOV FOR READING PHASE FROM CALMS.OVR
.BYTE 1,5 ; BK3.
.RAD50 /DK/
.BLKW 2
.RAD50 /CALMS OVR/
.BYTE 2,2
.L377: .WORD 377
.PSIZE: .WORD 32256.
.TWO: .WORD 2
INDEX: .WORD .
FFHASE: .WORD 2 ; FIRST ANALYSIS PHASE (OR DEBUG PHASE)
PHASE: .WORD MASKPH
PPHASE: .WORD 16 ; PREP PHASE
EHASE: .WORD 17 ; TEXT EDITOR PHASE
FFHASE: .WORD 18 ; FORTRAN PHASE
DUNIT: .WORD . ; DISK UNIT FOR SCAN OR HARDCOPY DATA SET
LHAM: .ASCII /D0/
PLHAN: .ASCII /PAR/
PMYS: BLK 10.
BYTE 1.1
RD50 /OK /
RD60 /CALMS_PAR /
RD61 /CALMS_PAR /
BYTE 2.2
BLK 10
PARSZ WORD 1024.
PLITE WORD 100.40.40.40.40.40.10.10.10.10.4.2.10010.10010.4000
WORD 4000.4000.4000.10010.0.0.0
NPTAB BLK MH
BYTE 12532
NPN 

tabcular

DNPTAB BLK MPH
BYTE 12532
HPN TABLE
PAGE
HCUC=172430
HCAD=172432
HCST=172434
HCMD=172434
HCIV1=440
HCIV2=442
INTPS=340
HCOPY: MOV $77777,A
CLRC ACTIVE BLOCK NUMBER
MOV $77777,A+IBN
CLRC INACTIVE BLOCK NUMBER
MOV $BLBSIZ2:A+BUFFSIZ
READ LABEL INTO SLAB
CLR A+DBFLAG
PREVENT READ-AHEAD
CLR A+CUREC
CLEAR CURRENT RECORD NUMBER
CALL GLABEL,(A,SPAR,1A)
:READ THE LABEL INFORMATION INTO CORE AND CALCULATE RECLEM.
THEN, SET UP THE MYS TO USE TWO 4096 BYTE BUFFERS IN CORE.
FOR THE DISK TO CORE TO HARD-COPY DMA TRANSFER.
MOV $77777,A
CLRC ACTIVE BLOCK NUMBER
MOV $100908.A+2
ONE BUFFER AT 16000
MOV $77777,A+IBN
CLRC INACTIVE BLOCK NUMBER
MOV $100908.A+IBA
ONE BUFFER AT 17000
MOV $100908.A+BUFFSIZ
BUFFSIZ=4096 BYTES
MOV $119,A+DBF
18 BLOCKS PER BUFFER
INC A+DBF
SET DOUBLE BUFFERING
CLR LINE
:INITIALIZE LINE COUNTER
MOV #ENTRY,0#HCIV1
JNT ENTRY ADDR TO HCOPY INT VEC
MOV #INTPS,0#HCIV2
JNT SVC PSW TO HCOPY INT VEC
INC BPL
IN CASE OF ODD BYTES
ASR BPL
WDT CNT FOR HCOPY LINE IS BPL/2
MOV BPL,BPL2
12*DBCHT
MOV A+54.RMB
RECLEM
SUB BPL,RMB
SUB BPL,RMB
MOV $100908.A+HCAD
SUB RMB,00HCAD
CLR #HCUC
CLEAN HCUC MNT REGISTER
BIT $2900,#HCST
BIT-10 IS HC RUNNING
BNE ONST
WAIT FOR PREVIOUS END-FRAME
BIT $4900,#HCST
BIT-11 IS HC READY
BNE REPL
BR IF HC READY
HCOFF: MOV #HCST,-(SP)
DISPLAY HCST
MOV $951,-(SP)
A191 MSG. HC OFF
TOT: MOV BPL,#HCAD
ACTION MSG. TO OPERATOR
SP ON ST?
CHECK HCOPY DEVICE AGAIN

HCOPY DEVICE INTERRUPT SERVICE ROUTINE
ENTRY: BIT #10090,A#SWR
BR IF SY.12 UP
DUE NXTLN
COM LXPB
BEQ NXTLN
BR IF NEW LINE
SUB BPL2,#HCAD
REPLACE HCAD
BEQ #GET
REPEAT THE LAST LINE
NXTLN: BEQ ML
DECREMENT LINE COUTN & TEST
REPL: INC LINE
ADD RMB,00HCAD
BIT $77777,#HCAD
BEQ #GET
CFL

RRDD
HOGED

R

BIT
BEQ
TSTB

BEQ
SCP

MOV

SCP, PINT
STORE INTERRUPT ADDRESS

R

CALL TYPE, (MSGJ, ZER0)
CALL PARAM, (NP, PAR, MAXNP, SPBUF)

CMPB SPBUF, Y

BNE HCOX

JMP HCOPI

HCOK, CLRBS BDHS
RELEASE THE HCOPY DATASET

DEC HICOPY

HCLED, CALL LED, (HLED, HCOPY)
ENTRY FOR RESTART

MOV R0, -(SP)

R1, -(SP)

MOV HND5, R0

MOV #HDS, R1

MOV #MSGN, REG2

H40:

TSTB (R1)+

BIT H59

AND H2, REG2

SOS P, H43

CLRBS HCFLG

SET HCOPY NO LONGER OPERATING

BIC H1, YLITES

TURN OFF HCOPY LITE

MOV (SP)+, R1

MOV (SP)+, R9

RTI

RETURN FROM INTERRUPT

H50:

MOV BREG2, 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

MAXNP, WORD MAXNP

LHGU, WORD HGU

ZERO, WORD 0

ONE, WORD 1

FOUR, WORD 4

FIVE, WORD 5

TWELVE, WORD 12

A, WORD 0

BLKB, WORD MBSSIZ

SLAD, BLKB LAPSSIZ

SPAR=SLAD+2

YL=SPAR+2

PL=SPAR+2

DONT=EPL

SPE=SPAR+4

LXPHD, WORD -

RMB, WORD -

SPL2, WORD -

LINE, WORD -

NFDS5, WORD -

CMD5, WORD -

CHDS5, WORD -

HDSN, BLKW HDSN

CHDS, WORD -

HOGED, BLKW HOGED

CSCN, WORD -

REG2, WORD -

XVAL, WORD 2080

YVAL, WORD 2080.

4,122,518
HHCOPY: WORD .-.
SFLAG: BYTE .-.
SDFS: .BLKB @MDS
HDFS: .BLKB @MDS
HCFLG: BYTE .-.
RFLAG: BYTE .-.
PFLAG: BYTE .-.
EFLAG: BYTE .-.
FFLAG: BYTE .-.
SIZE: .ASCIIZ /!
MSG1: .ASCIIZ /!
PEMSG: .ASCIIZ /!
MSG2: .ASCIIZ /!
M4: .ASCIIZ /!
M5: .ASCIIZ /!
M6: .ASCIIZ /!
M7: .ASCIIZ /!
NO: .ASCIIZ /!
SPBUF: @BLKB 3.
FILPEN: .ASCIIZ /!
LNAME: .ASCIIZ /!
END: .ASCIIZ /!

.HDBLS: source + 13908.

.END

.TITLE SEARCH - SEARCH PARTITION OF CALMS
.MCALL CALL. PAUSE
.LF=12
.CR=15
.FTIME=15
.PSY=177776
.SY=177776
.MACRO GETAS MSG,LOC
.TSTG PBUSY
.BEG LOC
.MOV @LOC.PINT+2
.RTI
.LOC.CLAR @PSU
.CALL TYPE, (MSG, ZER0)
.ENDM

.MACRO ENH CODE ENABLE INTERRUPT
.BIC CODE, @IEAPD
.ENDM

.MACRO DIS CODE DISABLE INTERRUPT
.BIC CODE, @IEAPD
.ENDM

.MACRO LON CODE TURN LIGHT ON
.BIC CODE, @LITES
.ENDM

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

.MACRO HINT KEY HANDLE INTERRUPT FOR KEYS 5-8 (AC, RJ, HX, LA)
.DSI 109
.LOFF 199999
.MOV RAX, (SP)
.CALL LED, (LKLED, KEY)
.ENDM

.CURRENT ORIGIN IS 48930

GHOSTS: JMP GNS
ENTRY POINT FOR GHOSTS

SEARCH: MOV @SRSTA, RA
INITIAL CALL FROM SUPERVISOR
.MOV @SRST, (RO)+
.SET UP SEARCH START INTERRUPT VECTO
.MOV @298, (RO)+
.PRIORITY 4
.MOV @IS, RI

4,122,518
IS20: MOV @UIINT,(RO)+  
NOV @200,(RO)+  
S0B R1,IS20

MOV @FOCA @FOC  
MOV @KYRB,.@KYRB  
MOV @2000.@KYRB+2  
.POSS MFST.<MAGN, FP0S>

E110  

INT: PAUSE E1APD  

NOV OY  

SRST: DSI 10  

CAL LLE. (KLED, SSKEY)  

GETKB MI.  

CALL PARM. (NP, PAR, MAXNP, SPBUF)  

CMP NP. 42  

BNE SR20  

BR IF NOT 2 WORDS  

CMP PAR. 3  

BNE SR20  

BR IF NOT ABORT  

CALL M YPE. M7  

SEARCH ABORTED, TYPE M7

SR20: INCB SFLAG  

SET SEARCH OPERATING  

LON LGSR99  

RETURN OFF EDIT LIGHTS  

LON 1000  

RETURN SEARCH LIGHT ON

TST NP  

BEQ SR30  

BR IF SAME PATIENT AND SLIDE

_CALL MVC R. (SPBUF, PARU, LPARU)  

MOVE PATIENT NUMBER

_CALL TYPE. (M2, ZERO)  

MOV @. PAR  

INITIALIZE SEX TO BLANK

_CALL PARAM. (NP, PAR, MAXNP, SPBUF)  

MOV PAR.PSEX  

STORE PATIENT SEX

_CALL TYPE. (M3. ZERO)  

_CALL PARAM. (NP, PAR, MAXNP, SPBUF)  

_CALL MVC R. (SPBUF, SNUM, LSLNUM)  

MOVE SLIDE NUMBER

_CALL TYPE. (M0, ZERO)  

_CALL PARAM. (NP, PAR, MAXNP, SPBUF)  

MOV PAR.SOURCE  

STORE SOURCE CODE

SR30: MOV @SP. CPSP  

INITIALIZE SEARCH QUEUE POINTER

CLR MSP  

MSP=0

MOV Rb. (SP)  

SAVE RO BEFORE CALLING ZIA

_CALL ZIA. (SPR, LSPR)  

ZERO THE SPREAD RATINGS

MOV (SP) + RO  

_CALL LED. (CMLED, MSP)  

_CALL MFST. (MAGN, FP0S)  

MOV XLCHR. XCTR  

INITIALIZE SEARCH PATTERN

MOV INDEL.XDEL  

_CALL SDR. @SDFPA  

_CALL SDFPA. @SDFPA  

_CALL SDFPA. @SDFPA  

_CALL SDR. @SDFPA

MOV @EDSt. @EDST  

EN1 74  

DISABLE INTERRUPTS

RTI

FOC:  

_CALL LED. (KLED, FKEY)  

FOCUS INTERRUPT FROM FOCUS KEY

FOCUB: DSI 400  

FOCUS SUBROUTINE

LOH 4000  

IT MAY BE ENTERED BY ANOTHER INTERRUPT

MOV @FIRE. FCNT  

RESET FOCUS COUNT

MOV @IEAPD. (IEAPD  

SAVE OLD IEPAD AND SDFPA

MOV @SDFPA. @SDFPA  

_CALL MFST. (MAGN, FP0S)  

FINO FSSTEP & FASL FOR THIS MAGNIFICATION

MOV @F41. @SDFPA  

SET UP INTERRUPT VECTOR

EN1 4

RTI

F410: _CALL CFQC. OLD  

_CALCULATE FOCUS

F420: MOV @F439. @SDFPA

RTI

F430: _CALL CFQC. F  

SEE IF WE ARE GOING IN THE RIGHT DIR

CMP F. OLD  

BGE F430  

BR IF SO

MOV FSTEP. N  

_CHANGE DIRECTION

ASL N  

MOV TWICE AS FAR IN REVERSE
CALL IFREL, H
MOV 0F440, 0SDFFPA
RTI

CALL CF0C, F
CMP F.O.LDF
BCT F440
CMP FSTEP, FLAST
BLE F400
MOV FSTEP, N
ASR FSTEP
ADD FSTEP, N
N=FSTEP+FSTEP/2
CALL IFREL, H
BR F420

F440:
NEG FSTEP
MOV F.O.LDF
CALL IFREL, FSTEP
JTAKE ANOTHER STEP
MOV 0F460, 0SDFFPA
RTI

F450:
CALL CF0C, F
CMP F.O.LDF
BCT F450
MOV FSTEP, N
BPL F470
NEG N
CALL CF0C, F
CMP F.O.LDF
BCT F450
MOV FSTEP, N
ASR FSTEP
BLE F400
BR F450

F470:
NEG FSTEP
CMP H, FLAST
JTEST FOR LAST STEP
BLE F400
BR F450

F480:
CALL IFREL, FSTEP
JTAKE LAST STEP BACK
MOV 0F490, 00SDFFPA
RTI

F490:
DSI 4
CALL RMC1
MOV 0SDFFPA, 00SDFFPA
RESTORE SDFFPA
BIT 24, OIEAPD
JSEE IF INTERRUPT WAS ENABLED
BEQ F500
BIT 01000, 01LITES
JBR IF NOT
BEQ F490
JRE-ENABLE IF IF NOT IN SEARCH
TST 00SPRD
JOR SPREAD NOT PRESENT
BPL F490
BIT 00000, 01SYR
JFOR SPREAD NOT PRESENT
BEQ F490
JOR SW II DOWN
LON 20000
JTURN ON SEARCH HALTED LIGHT.
BR F300
JDON'T RE-ENABLE THE INTERRUPT
F499:
EN 4
JRE-ENABLE INTERRUPT
F500:
EN 480
JTURN OFF FOCUSING LITE
RTI

KY9:
DSI 1829
JDISABLE ANY FURTHER K80 REQ
CALL LED, (LKLED, KRKEY)
GETKB M4
CALL PARAM, (HP, PAR, MAXX, SPDUF)
TST HP
BEQ BKYEX
JBR IF NO INPUT
CMPB PAR, 'A'
BNE KY10
JBR IF NOT ABORT REQUEST
BEQ BKYEX
KY10:
CMPB PAR, 'C'
BNE KY20
JBR IF NOT CALIBRATE MICROSCOPE
CALL IKREL, BIG
CALL IYREL, BIG
JMOVE TO LIMITS
CALL IFREL, BIG
CALL IWAIT
CALL IWAIT
JWAIT FOR END OF MOVES
CALL IFCTR
JINITIALIZE F.X, AND Y
CALL IXCPR
CALL IYCR
CALL IKYABS, (YVAL, YVAL)
JMOVE STAGE TO 1000, 1000 OR LAST POSITION
CALL IFREL, FVBL
CALL XXWAIT
CALL YWAIT
CALL FUART
CALL RNCI
BR KYEX

KY20: CMPB PAR, #R
BNE KY30 BR IF NOT RESTART REQUEST
INCB FSLAG INFSET RESTART FLAG IN CALMS
KY30: CMPB PAR, #E
BNE KY23 BR IF NOT EDIT REQUEST
INCB EFLAG INFSET EDIT FLAG
BR KYEX

KY35: CMPB PAR, #F
BNE KY40 BR IF NOT FORTRAN REQUEST
INCB FSLAG INFSET FORTRAN FLAG
BR KYEX

KY40: CMPB PAR, #Z
BNE KY50 BR IF NOT ZERO SCAN QUEUE REQUEST
CLR NSCAN
CALL LED,(SQLED,NSCAN)
MOV #SRST, #SRSTA ALLOW SEARCH START
BR KYEX

KY50: CMPB PAR, #Q
BNE KY60 BR IF NOT QUEUE ADJUST
MOV @NCPRA, XOFF
SUB YVAL, XOFF INSTORE X OFFSET
MOV @YCPRA, YOFF
SUB YVAL, YOFF INSTORE Y OFFSET
BR KYEX

KY60: CMPB PAR, #S
BNE KY70 BR IF NOT SCAN REQUEST
KY65: INC NSCAN
CALL LED, (SQLED, NSCAN)
MOV #SPRCH, #RSRSTA INSET UP SPIRAL SEARCH
HPLP #44 INSEE IF X AND Y WERE GIVEN
BNE KYEX BR IF NOT
CALL IXYABS, (PAR+4, PAR+6)
ENI 1499
JMP FOCUSUB

KY70: CMPB PAR, #D
BNE KY80 BR IF NOT DEBUG REQUEST
MOV PAR+4, PPHASE INSTORE PHASE NUMBER
BR KY65 INCRE INCREMENT NSCAN
KY80: CMPB PAR, #B
BNE KY90 BR IF NOT BAHDED REQUEST
MOV #3, WSOURCE INFSET SOURCE OF #8
BR KYEX

KY90: CMPB PAR, #P
BNE KY10 BR IF NOT PATIENT REPORT REQUEST
INCB FFLAG INFSET PREP FLAG
CALL UPARSH, (ONE, PAR, PPHASE) INWRITE PREP PARAM
BR KYEX

KY10: ENI 1030
RTI

SDR: TST #SPRD INSPREAD DATA READY (SEARCH)
BNE SPR3S BR IF SPREAD PRESENT
MOV #1, FFLAG INRESET FOCUS FLAG
DEC FCHT INDEC INCREMENT FOCUS COUNT
BPL CHP BR IF NO NEED TO FOCUS
BIT #63999, #SPRD INBR IF NOTHING TO FOCUS ON
JMP FOCUSUB INGO FOCUS

SPFH3: ASL PFLAG INASLT
asl
JMP FOCUSUB INBR IF NO FOCUS THIS TIME

MOFOC: MOV R3, -(SP) INR3=CURRENT SPREAD QUEUE POINTER
MOV @NCPRA, (R0)+ INSTORE X
PLR XDR INBR IF X IS POSITIVE
XYNG: GETKB M6 INREQUEST CALIBRATION
MOV (SP)+, R0
XOK: MOV @ICYPR, (RB)+ ; STORE Y
BNI XNG ; BR IF Y IS NEGATIVE
CMP MAG, #63 ; SEE IF SEARCH IS AT HI-MAG
BLT LONAG ; IF AT HI-MAG, FLAG AS ACCEPTED IN ED-LO
LONAG: MOV RB, CSPQ ;
TST XCTR ; INCREMENT & SPREADS FOUND
CALL LED, (SNLED, HSP) ;
CALL HSP, HAXNSP
BGE ELDL ; IF ENOUGH SPREADS FOUND
CMP, TST XCTR ;
BGRT G50 ; BR IF X-MOVE IS OK
NEG XDDEL ; END OF ROW, REVERSE DIRECTION
CALL ITREL, (YDEL) ; AFTER MOVING DOWN
BR G50 ;
G48: MOV XLGTN, XCTR ; RELOAD XCTR
G50: CALL IREL, (YDEL) ; MOVE IN X DIRECTION
G60: DEC XCTR ;
BIS 04, 04IEAPD ; ENABLE INTERRUPT
RTI
SRHA: DSI 4 ; SEARCH OR FOCUS HALT
DTC 04, 04IEAPD ; PREVENT END OF FOCUS FROM RESUMING SEARCH
CALL LED, (XLKD, SHAKEY)
LOF 20399 ; TURN ON SEARCH HALTED LIGHT
CALL RNCI
RTI
SRRS: EMI 4 ; SEARCH OR FOCUS RESUME
CALL LED, (XLKD, RSKY)
LOF 20399 ; TURN OFF SEARCH HALTED LIGHT
RTI
EDST: DSI 184 ; EDIT START
LOF 120009 ;
CALL LED, (XLKD, ESKY)
CALL NFST, (MAG, FPPOS)
ENI 400 ; ENABLE FOCUS INTERRUPT
MOV @1 FFAG ; INITIALIZE FOCUS FLAG
BIT $1000, $0LITES
BNE EDLO ; BR TO EDIT-LO IF IN SEARCH
BIT $040, $0LITES
BNE ES40 ; BR IF 9N EDIT-LO
CMP MAG, #63
BLT ESLTT ; BR IF LO-MAG (IN EDIT-HI)
JMP EHRST ; HI-MAG (IN EDIT-HI)
ES40: CMP MAG, #63 ; JN EDIT-LO, TEST MAG
BLT EHRST ; BR IF LO-MAG
JMP EHRST ; HI-MAG. START EDIT-HI
EDLO: CLR @CSPQ ; MARK END OF QUEUE
DSI 20
CLR HSPAL
CMP MAG, #63
BLT ESLTT ; BR IF LO MAG
JMP EHRST ; START ED-HI IF HI MAG
ELSTT: CALL LED, (SOLED, HSPAL)
LOF 1200
LOF 400 ; TURN ON EDIT LO LIGHT
MOV $10AC, @ACSPQ ; ACCEPT SPRED
MOV $109J, @RJSPQ ; REJECT SPRED
MOV $10E1X, @NXSPQ ; NEXT SPRED
MOV $10LLA1, @LAST SPRED
ELAST: CLR SP ; SPREAD NUMBER
MOV @SP, CSPQ
MOV R0, -(SP) ; SAVE R0 ON THE STACK
EDNXT: MOV CSPQ, R0
LOF 40000 ; TURN OFF FOCUS LITE IN CASE FOCUS
ENI 400 ; WAS INTERRUPTED BY ARNL
INC SP
CALL LED, (SNLED, SPN)
LOAD 2000
MOV (R0)+, X
BEQ JEDHI ;BR IF END OF QUEUE
BPL EL38 ;BR IF NOT ALREADY ACCEPTED
LON 2000 ;INDICATE ALREADY ACCEPTED
EL38: MOV R8, C8PQ ;SAVE UPDATED Q POINTER
MOV (SP)+, R0 ;RESTORE R0
CALL IXYABS, (X,Y) ;INITIATE MOTON MOVE TO X,Y
MOV #ESDR, @SDFPA
MOV #1, FFAN ;ENABLE SPRD DATA READY INTERRUPT
RTI

JEDHI: JMP EDHI

ESDR: ENI 100 ;ALLOW RJ DURING FOCUS
NEG FFLAG ;EDIT SPRD DATA READY
BPL ES20 ;BR IF NO FOCUS THIS TIME
JMP FOCSUB
CALL RMC1 ;RELEASE MCI FROM COMPUTER CONTROL
LOH 100000 ;TURN ON OPERATOR ACTION LIGHT
RTI

ELAC: HINT ACKEY ;ACCEPT SPRD INTERRUPT
MOV C8PQ, R0
TST -4(R0)
BPL EL28 ;BR IF NOT PREVIOUSLY ACCEPTED
BEC HSPAL ;DON'T COUNT IT TWICE
EL28: MOV #CRP, -(R0) ;STORE UPDATED Y AND X
MOV #XCRP, -(R0)
NEG @R8
INC HSPAL
CALL LED, (SGLED, HSPAL)
CMP HSPAL, HLMAX
BGE EDHI ;BR IF ENOUGH FOUND
BR EDNXT ;GO TO NEXT SPRAD

ELRJ: HINT RJKEY ;REJECT SPRD INTERRUPT
MOV C8PQ, R0
TST -4(R0)
BPL JEDNXT ;BR IF NOT PREVIOUSLY ACCEPTED
BEC HSPAL
CALL LED, (SGLED, HSPAL)
BR JEDNXT
ELAH: HINT LHKEY ;LAST SPRAD
BR JEDNXT
ELLA: HINT LHKEY ;LAST SPRAD
SUB #4, C8PQ ;MOVE BACK TO CURRENT SPRAD
BEC SPH
BEQ JEDNXT ;BR IF FIRST SPRAD IN QUEUE
SUB #4, C8PQ ;BACK UP ANOTHER SPRAD
JEDNXT JMP EDNXT

EDHI: MOV (SP)+, R0 ;RESTORE R0
GETKB MS ;REQUEST HI MAGNIFICATION
EHSTT: LOFF 1400 ;TURN ON EDIT HI LIGHT
LON 2000 ;SET UP EDIT HI INTERRUPTS
MOV @EHS, #00CSPA
MOV #EHRS, @RJSPA
MOV #EHRS, #00HSPA
MOV #EHL, #00SPA
MOV #EDEH, #00HSAH
ENI 208
CLR HSCAN
CALL LED, (SGLED, HSCAN)
EHSTT: CLR SPH
MOV #SPQ, C8PQ
MOV R0, -(SP) ;SAVE R0
EHNXT: MOV C8PQ, R0
69

69

LOFF 4000

ENI 400

INC SPW

MOV (R0)+, X

BEQ JEDEND

BNI EH20

TST (R0)

BR EH10

EH10: LOFF 2000

MOV (R0)+, Y

BPL EH30

LON 2000

BR EH40

CALL MST.<MAGH,FPQ>

SUB X0DJ.X

ADD YADJ.Y

ADJ POSITIVE Y FOR CENTRALITY

EH20: MOV R0, CSPQ

TST -2(R0)

BPL EH20

BRR IF NOT ALREADY ACCEPTED AT HI MAG

DEC NSCAN: DON'T COUNT IT TWICE

EH30: MOV #0YCPR,-(R0)

NEG R8

MOV #0XCPR,-(R0)

INC NSCAN

ADD #2,R5

IND SP(R0)

MOV <SP>+30

LON 10000

BRI 100

;RETURN FOR ANOTHER AC OR HX

EH4: HINT RKEY

MOV CSPQ,R0

TST -(R0)

BPL EHHXT

NEG #83

MAKE IT POSITIVE IF PREVIOUSLY ACCEPTED

DEC NSCAN

TST -(R0)

GET LOC IN SPR FOR THE SPREAD

SUB #SPR,R0

ASH #2,R5

CLRB SPR(R0)

ZERO THE SPREAD RATING

CALL LED.<SOLED,NSCAN>

BR JEHNXH

EHX: HINT MXKEY

CMP NSCAN, MAXSC

BGE EDEND

BR JEHNXH

EHL: HINT LAKEY

SUB 4, CSPQ

DEC SPH

BEQ JEHNXH

BRI IF FIRST SPRD IN Q

EHX20: SUB 4, CSPQ

MOV SPH

BEQ JEHNXH

TST (R0)

BPL EHL20

BRR IF NOT -X

JEHNXT: JMP EHnxH

LAST -X

EHX: CALL LED.<LKLKE,EEKEY>

JMP
MOV R0,-(SP) ;GET NEXT SPREAD TO SCAN SRTE

MOV CPSQ.R0
MOV #SPSRCH.@SRSTA

CPSRCH CALL LED.(LED.KEY), SPIRAL_SEARCH

CPSRCH CLR XCTR ;INITIALIZE SPIRAL SEARCH
CLR YCTR
CLR SPLEM
MOV #SPSR, 03DFFA
MOV #SRHA, 03SRSA

SPSRCH ENI: 34 ;ENABLE SEARCH HALT & RESUME & 3DR

RTI

SPSRR CALL LED.(LED.KEY), SPIRAL_SEARCH

SPSRR CLR XCTR
CLR YCTR
CLR SPLEM
MOV #SPSR, 03DFFA
MOV #SRHA, 03SRSA

ENI: 34 ;ENABLE SEARCH HALT & RESUME & 3DR

RTI

SGNP CALL SPIR, XCTR

SGNP TST YCTR
EQU SXOK ;BR IF ANOTHER X MOVE IS OK
TST XCTR
EQU SYOK ;BR IF ANOTHER Y MOVE IS OK
INC SPLEN ;INCREMENT SPIRAL LENGTH
MOV SPLEM.XCTR ;RELOAD X AND Y COUNTERS
MOV SPLEM.YCTR
NEG SPSTEP ;CHANGE DIRECTION

SYOK CALL IXREL, SPSTEP

SYOK DEC XCTR
RTI

SYOK CALL IXREL, SPSTEP

SYOK DEC YCTR
RTI

SPLEN MOV WORD - , SPIRAL_LENGTH

SPSTEP MOV WORD 5 , 03R -5

XCTR MOV WORD -

YCTR MOV WORD 9980.

XOFF MOV WORD -

YOFF MOV WORD -
 TITLE DLINE - DISPLAY A LINE ON THE INTERACTIVE DISPLAY CALMS VERSION

/title DLINE - DISPLAY A LINE ON THE INTERACTIVE DISPLAY

;WORD COUNT REGISTER
;BUS ADDRESS REGISTER
;STATUS AND COMMAND REGISTER
;LINE REGISTER
;SAMPLE REGISTER
;GRAY SCALE STATUS/COMMAND REGISTER
;GRAY SCALE PIXEL COUNT REGISTER
;GRAY SCALE Y CURSOR POSITION
;GRAY SCALE X CURSOR POSITION
;WRITE COMMAND
;ERASE COMMAND
;ERASE/WRITE COMMAND
;REPLICATE BIT
;USER=177570
;LF=12
;MCALL .B1N2Q, CALL, .PAUSE
;
;CALL .DWAII

.DWAIIT .TST .C#GSSST, .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, SAVCR
        JSR R5, DWAIT
        MOV #4(R5), D1YREG ; STARTING LINE ON DISPLAY
        MOV #URCMD.R1
        BIS #490, @DRST ; ASSUME WRITE, SET CYCLE BIT
        TST #14(R5)
        BEQ NOER ; BR IF NO ERASE
        MOV #UCMD.R1 ; ASSUME ERASE/WRITE
        TST @14(R5)
        BPL NOER ; BR IF ERASE/WRITE
        MOV #ERCMD.R1 ; ERASE ONLY
        BIC #490, @DRST ; CLEAR CYCLE BIT
        NOER: MOV #1(R5), R3
        MOV #12(R5), R2
        BEQ NOREP ; BR IF NO REPLICATION
        MOV #RSIT.R1
        BIS #RST.R1 ; INSERT REPLICATE BIT
        ASL R3
        JSR R5, DWAIT ; WAIT
        INC $04YREG ; STEP TO NEXT LINE
        BR REP ; GO DO THE REPLICATION

EXIT: JMP RESTR

; CALL SC TO SET UP THE CURSOR

SC: JSR R4, SAVCR
SC10: MOV #3.R3
SC10: MOV #32, X
SC10: MOV $32, Y ; WRITE A MARK AT 32, 32
SC10: CALL DLINE, (ZERO, Y, X, TEN, ONE, ZERO)
SC10: MOV #9, R4
SC20: ADD $2, Y
SC20: CALL DLINE, (ZERO, Y, X, ONE, ONE, ZERO)
SC20: SUB #4, SC20
SC20: SUB R3, SC10 ; REPEAT TO MAKE IT WHITE ENOUGH
SC10: MOV #3, R3
SC30: MOV #973, Y
SC30: MOV #992, X ; WRITE A MARK AT 992, 992
SC30: MOV $9, R4
SC40: CALL DLINE, (ZERO, Y, X, ONE, ZERO)
SC40: ADD #2, Y
SC40: SUB #4, SC40
SC40: MOV #973, X
SC40: CALL DLINE, (ZERO, Y, X, TEN, ONE, ZERO)
SC10: MOV #3, R3
SC50: CALL TYPE, (M3.ZERO) ; TELL OPERATOR TO MOVE CURSOR
SC50: CALL PARAM, (MIP, PAR.FIVE)
SC50: JSR R5, UDR ; CHECK FOR REQUESTED CURSOR MOVE
SC50: BR SC50 ; CURSOR WAS MOVED
SC50: MOV #65CYCP, Y2 ; STOP Y ZERO ADJUSTMENT
MOV @0GSXCP.XZ  ;STORE X ZERO ADJUSTMENT
              CALL TYPE,<R4.ZERO>  ;MOVE TO LOWER RIGHT
SC60 CALL PARAM,<HP,PAR,FIVE>
JSR R5,ULDB
BR SC60  ;CURSOR WAS MOVED
MOV @0GSYCP.YD
SUB YZ.YD  ;STORE Y BELTA
MOV @0GSXCP.XD
SUB XZ.XD
BIC @176000.YZ
BIC @176000.XZ
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,SAEC)
CURSOR JSR R4,SAVER
MOV @0GSYCP.R0
BIC @176000.R0
SUB YZ.R0  ;ADJUST THE READING
MUL @960.R0
DIV YD.R0
ADD @32.R0  ;AY=(Y-YZ)*960/YD+32
ASL R1
CMP R1,YD
BLT C20
INC R0  ;ROUND UP
C10 MOV @0GSXCP.R2
BIC @176000.R2
SUB XZ.R2  ;ADJUST THE X READING
MUL @960.R2
DIV XD.R2
ADD @32.R2
ASL R3
CMP R3,XD
BLT C40
INC R2  ;ROUND UP
C40 MOV R0.02(R5)  ;STORE YCP
MOV R2.04(R5)  ;STORE YCP
BIT @4.R5
BEQ CEX  ;BR 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=YCP/2+1
CEX JMP RESTR
MCU SUB @2.00GSYCP  ;CALL MCU TO MOVE CURSOR UP
MCD ADD @2.00GSYCP  ;CALL MCU TO MOVE CURSOR DOWN
MCL SUB @2.00GSXCP  ;CALL MCL TO MOVE CURSOR LEFT
MCR ADD @2.00GSXCP  ;CALL MCR TO MOVE CURSOR RIGHT
ULBR MOV @2(R5),PAR  ;CALL ULBR(PAR) ... EXTERNAL CALL
BR UL20
ULDR TST HP  ;CHECK FOR U.L.D. OR A TYPED
BNE UL20  ;IF A LETTER WAS TYPED
TST <R5>+  ;SET FOR NO TYPEIN RETURN
RTS R5
UL20 CMPB PAR,0'U
BEQ UL30  ;BR IF NOT A U
CMPB PAR,0'L
INC @0GSYCP
MOV CURSOR UP ONE STEP
UL30 CMPB PAR,0'U
BEQ UL4B
INC @0GSYCP
MOV CURSOR DOWN ONE STEP
UL4B CMPB PAR,0'L
BEQ UL5B
CMPB PAR,0'U
INC UL4B
CMPB PAR,0'L
BEQ UL5B
CMPB PAR,0'U
INC UL5B
CMPB PAR,0'L
BEQ UL5B
DEC 0 GSXCP MOVE CURSOR LEFT ONE STEP
BNE UL50
UL50: CMPB PAR, 18
INC 0 GSXCP MOVE CURSOR RIGHT ONE STEP
BNE UL60
INC 0 GSXCP MOVE CURSOR RIGHT ONE STEP
UL60: RTS R5 RETURN

ZERO .WORD 0, 0, 0, 0, 0
ONE .WORD 1
FOUR .WORD 4
FIVE .WORD 5
TEN .WORD 10
INTPH .WORD 7 INT PHASE NUMBER
PAR .BLKW 5
Y=PAR 2
NP=PAP+4
XLST BEX
M3 .ASCIZ MOVE CURSOR TO UPPER LEFT MARK/
M4 .ASCIZ MOVE CURSOR TO LOWER RIGHT MARK/
YX .WORD 32
YB .WORD 960
XB .WORD 960.
XD .BLKB DWAIT+1500.

END

TITL MSCSUB - MCI SUBROUTINES CALNS VERSION
MCALL CALL. PAUSE
XCPFR=164039
YCPFR=164042
FCPR=164044
MSCR=164046
THR=164059
SFP=164073
FCPF=164076
LITES=164134
LEDS=164136
SRSTA=300 SEARCH START INTERRUPT ADDRESS
SRS=310 SEARCH RESUME
EDSTA=314 EDIT START
ACSPA=320 ACCEPT SPREAD
RJSPA=324 REJECT SPREAD
NXTSPA=330 NEXT SPREAD
LSPA=334 LAST SPREAD
EDERH=348 EDIT END 9
FOCA=344 FOCUS 16
KYPQ=355 KBD REQUEST 11
DSP=374 SPRD/FOCUS DATA READY INTERRUPT ADDR
CURREM ORIGIN IS 59500

MCI: BIS 48899.00MSC CAPTURE MCI
RTS R5
RNCI: BIC 48899.00MSC RELEASE MCI
RTS R5
XWAIT: BIT 48800.00SMSC
BNE XWAIT
RTS R5
YWAIT: BIT 4209.00SMSC
BNE YWAIT
RTS R5
FWAIT: BIT 110.00SMSC
BNE FWAIT
RTS R5
IXYABS CALL XWAIT CALL IXYABS(X,Y) TO INITIATE ABS MOVE
CALL YWAIT CALL CMC
MOV #2, 35.0(X)
PDL IXY20
81

NEG X
MOVE 04(R5), Y
BPL IXY30

NEG Y
MAKE Y POSITIVE

IXY30:
SUB X, #IXCYPR
INITIATE MOVE TO X, Y
SUB Y, #IXCYPR
RTS R5

IXCYREL:
CALL XWAIT
CALL XCYCLE RELATIVE TO INITIATE X MOVE

RTS R5

ICYREL:
CALL YWAIT
CALL CYCLE
MOVE 02(R5), A#IXCYPR
RTS R5

ICYREL:
CALL FWAIT
CALL CYCLE
MOVE 02(R5), A#ICYPR
RTS R5

CFOC:
MOVE 03FC, TEMP
CALL CFOC(T) TO CALCULATE FOCUS

CLR 02(R5)
MOVB TEMP+1.02(R5)
ADD TEMP, 02(R5)
FOCUS=F1+F2

RTS R5

ICYPR:
BI$ 0400.0#SHSC
INITIALIZE XCYPR

RTS R5

ICYPR:
BI$ 0200.0#SHSC
RTS R5

ICFC:
R8.0#SHSC
RTS R5

LED:
JSR R4. SAVR
CALL LED(CODE, VAL)
MOVE 02(R5), R0
CODE TO R0
ASH #13, .R0
SHIFT TO HL ORDER 3 BITS
MOVE 04(R5), R1
VALUE TO R1
BIC #177000, R1
ADD R1, R0
MOVE R8, B#LED
OUTPUT DISPLAY
JMP RESRT

MBST:
JSR R4, SAVR
CALL MBST(MAGN, FPPOS)
MOVE #81EAPD, R0
FLTR/MAGN POS REG TO R0
MOVE R8, R1
ALSO R1
ASH #1B, R0
MOVE #177776, R0
MASK OUT MAG
MOVB LEHS(R0), MAGN
STORE MAGN
CLR FSTEP
CLEAR TOP BYTE
MOVB FST3C(R0), FSTEP
STORE FOCUS STEP SIZE
MOVB FLTB(R0), FLAST
STORE LAST STEP VALUE
MOVB XTAB(R0), R2
ADJUSTMENT FOR NON-CENTRALITY
MOVE R2, XADJ
MOVE YTAB(R0), R2
ADJUSTMENTS ARE FOR 63 AND 100
MOVE R2, YADJ
RELATIVE TO 40
MOVE MAGN, 02(R5)
ASH #13, R1
SHIFT FILTER CODE
BIC #177776, R1
MOVE R1, 94(R5)
MOVE R1, FPPOS
CALL LED(COLLED, MAGN)
DISPLAY MAGNIFICATION
JMP RESRT

SORTQ:
JSR R4. SAVR
SUBROUTINE TO SORT THE QUEUE
MOVE #SPR, R3
R3 AND R4 POINT TO LOW QUEUE POSITIONS
MOVE #SPQ, R4
THE CURRENT HI RATING IS MOVED HERE
MOVE HSP, MSR
INITIALIZE # SPREADS REMAINING
MOVE R3, R1
USE R1 TO EXAMINE REMAINING SPREADS
MOVE MSR, R2
BEQ SDEX IF NO SPREADS REMAIN

MOVE (R1)+, R0
EXAMINE A RATING
BLE JBR IF NOT GREATER THAN MAX
MOVE - (R1), R0
STORE NEW MAX
MOVE R1, REG1
SAVE R1
I
NC
s O
BE
SQX
DEC
TST
BE
r
oy
I
J
E
rl
DD
HOYB
NOVB
t
r
y
.i
i
R
IF
NORE
TO
EXIRINE
.-
9
R
IF
ALL HAVE BEEN SORTED,
,PLLC
HkYEl
POSITIOIO
.....
.....
.....
SWlCH ReTINCS
..-..

;

JMP S102
NOTPQ: CMP *A, PAR
BNE NOTAB
.CALL APHASE, ZERO
JMP S388
NOTAB: CMP *Z, PAR
BNE S183
CLR NSCAN
.CALL APHASE, ZERO
JMP SEXIX
S183: CMP *EX, PAR
BNE S184
SCAN IF EX WAS NOT TYPED
JMP SEXIX
S184: CMP *DS, PAR
BNE S185
BR IF NOT DUMMY SCAN
INC DFLAG
SET DFLAG
S185: MOV *PBAR, R1
MOV @L+217, R2
ADD ANOTHER LABEL WITH MISC ID
HLAB: MOV R8, OCR
BEQ ELAB
BR IF END OF LABEL (CR)
S186: MOV R0, (R2)+
S188: JSR PC.PPAR
.JPROCESS PARAMETERS
.CALL OUTCON, (XPR, L+127, FOUR)
.CALL OUTCON, (YPR, L+135, FOUR)
.CALL MVL, (L+124, M2+15, FOUR)
.CALL MVL, (L+132, M2+22, FOUR)
.CALL TYPE, M2
CVTD $L+145,
.STORE DATE IN LABEL
CVTD $L+153,
.STORE TIME IN LABEL
TST DFLAG
BNE S188
NO OUTPUT LABEL IF DUMMY SCAN
.CALL PLABEL, (BUF,SPAR,L), OUTPUT LABEL
S188: CALL DCLEAR
S188: CALL DCLEAR
S189: CALL DCLEAR
S189: CALL DCLEAR
S190: CLR LINE
S190: CLR LINE
MOV SL, X
S190: INITIAL XCOUNT
CLR *COND
S190: DISABLE PIXEL INTERRUPT
CLR CY
S190: CLEAR GS LINE COUNTER
MOV *Ttab, TX
S190: PAD REST OF LABEL WITH BLANKS
MOV *Ttab, BX
S190: SET PRIORITY 6 FOR THE REST OF THE FIELD
JSR PC.THEAD
S190: TYPE HEADING
LSEC: CLR SHIRI
S190: SECTOR NUMBER=0
.CALL ZIA, (HIST, HDIN)
S190: ZERO HISTOGRAM COUNTS
EFLIN: MOV SS, Y
S190: INITIAL Y
MOV *X, REG
S190: SET LINE COUNT
INC LINE
S190: INCREMENT MV10 LINE COUNT
TST DFLAG
BNE S195
BR IF DUMMY SCAN
.CALL PUT, (BUF, LINDEX)
S190: PREPARE STORAGE OF NEXT TV LINE
S190: PUT BUFFER POINTER TO BUF+HS-1
0FLIN: ADD INDEX, R3
S190: SET POINTER FOR NEXT LINE
MOV RO, R4
S190: X4 HAS LOC OF HIST
MOV ISPS, R3
S190: INITIAL SAMPLES PER SECTOR
MOV Y, RO REG
S190: SET PIXEL COUNT
MOV #COND, R2
S190: (COND-STATUS REGISTER) TO R2
MOV *J, R2
S190: GET FIRST PIXEL OF THIS FIELD
TSTB OR2
BPL -2
S190: WAIT UNTIL READY
MOV #391, 00PSW
S190: SET PRIORITY 6 FOR THE REST OF THE FIELD
BR NOVY
SECTOR: MOV SPS, R3
S190: R3=SAMPLES PER SECTOR
PIXEL: MOV #3, OR2
S190: PIXEL COMMAND
TSTB OR2
SP -2
S190: WAIT UNTIL READY
NOVY: MOV #2, REG, R1
S190: MOVE PIXEL INTO BUFFER
BNE NOT2E
S190: BR IF NOT ZERO
INC R1

; CONVERT 0 TO 1

BNE NOT127

; BR IF NOT 127

MOV R1, #127.

; CHANGE 127 TO 126

MOV R1, #126

; STORE PIXEL IN BUFFER

ADDC R4, R1

; HIST + PIXEL VALUE (0, 2, 4, ... 62)

INC R1

; INC COUNT FOR THIS VALUE

SUB #2, R0

; DECREMENT POINTER

SBB R3, PIXEL

; ALL MORE PIXELS IN THIS SECTOR

MOV R1, #128

; STEP HISTOGRAM POINTER

HTEST: CMP R4, #HIST + 1024

; HIST + NC = 128

BLO SECTOR

; BR IF MORE SECTORS

BIT $1, Y

; TEST FOR EVEN OR ODD FIELD

BGT OFEND

; BR IF END OF ODD FIELD

EFEND: CLR @PSW

; RETURN TO PRIORITY 0 AT END OF FIELD

INC Y

; SET Y FOR ODD FIELD PIXELS

BEQ EFMOV

; BR IF FIRST LINE

EFMOV:

; CALL DLINE (<, -, +, #, GX, NS, REPL, ZERO) , REPLICATE PREVIOUS LINE

INC R0

; SET BUFFER POINTER TO BUF + MS - 2

JMP OFLIN

; GO GET ODD FIELD PIXELS

OFEND:

; CALL DLINE (<, -, +, #, GX, NS, REPL, ZERO) , WRITE TO GRAY SCALE

BIT $10, #5SR

; BR IF NOT QUICK LOOK

BEQ S5

; BR IF NOT QUICK LOOK

INC Y

; STEP Y AGAIN

INC X

; INCREMENT X (LINE.COUNT)

DEC SCTR

; BR IF END OF SECTOR

JSR PC.TRESH

; CALC THRESHOLDS FOR THESE SECTORS

CMP X, EL

; BR IF FINISHED

BSE FIN

; BR IF FINISHED

JMP LSEC

; GO START NEXT LINE SECTOR

JEFLIN:

; CALL DLINE (<, -, +, #, GX, NS, REPL, ZERO) , WRITE TO GRAY SCALE

JMP EFLIN

; TEST FOR END OF PICTURE

NOTES: CMP X, EL

; BR IF END OF SECTOR

JSR PC.TRESH

; CALC THRESH FOR LAST SECTORS

FIN:

; CALL DLINE (<, -, +, #, GX, NS, REPL, ZERO) , WRITE TO GRAY SCALE

MOV #TBATA, R1

TST DFLAG

; BR IF END OF SECTOR

BNE S202

CPUT:

; CALL PUT (.BUF-ZERO, INDEX)

MOV #BUF, R2

LDD INDEX, R2

MOV HS, R3

JMP S202

S202, S300:

SEXIT:

; CALL CLOSE.BUF

CLOSE.BUF

; STORE PIXEL IN BUFFER

P10:

; BR IF MORE PARAMETERS

CALL PPAR

; CALL PARAM. (@PAR, PAR, #PAR, #PAR, #PAR, #PAR)

MOV #PAR, R1

CLEAR R1

CLEAR HC

MOV @TPAR

JMP P15

; BR IF MORE PARAMETERS

P15:

MOV (R1) +, (R0)

MOV (R0), R2

MOV @R, R2

MOV @KEY, R3

P20:

CMP R0, (R3)
DEG P50: BR IF PARAMETER IS KEYWORD
DEG P20: BR IF MORE Keywords
CMP HPAR, R4: NOT A KEYWORD, MAYBE AREA IS SPECIFIED
BCE P30: BR IF 4 WORDS LEFT

P25: CALL TYPE, M4: PARmETER ERROR
BR PPAR

P30: CMP R0, #MAXXL
BCE P25: BR IF SL TOO LARGE
MOV R8, SL
BLT P25: BR IF SL TOO SMALL
MOV (R1)+, R0
BLE P25: BR IF EL TOO SMALL
CMP R8, #MAXXL
BGT P25: BR IF EL TOO LARGE
MOV R0, EL
SUB SL, R0

BLE P25: BR IF HL TOO SMALL
MOV R8, HL
INL=EL-SL
MOV (R1)+, R0
ISS
BLE P25: BR IF ES TOO SMALL
CMP R0, #MAXXL
BGT P25: BR IF ES TOO LARGE
SUB SS, R0
INS=ES-SS
BLE P25: BR IF HS TOO SMALL
MOV R0, HS
MOV #S12, R3
CLR R2
DIV R0, R2
MOV R2, NLR
INLR=S12/HS
ADD #BUFF, R0
DEC R0
MOV R0, BUFF
BUFF=BUF+HS-1
SUB #4, HPAR
HPAR=HPAR-4

P50: TST (R1)+: STEP OVER SECOND WORD
ADD #MK+MK-2, R3
LOC FOR THIS KEYWORD
MOV P23, PC
GO THERE

P60: MOV (R1)+, JPER
MOV #100, JPER

P63: MOV DPER, JPER
SUB DPER=PER-100-JPER
SUB #3, HPAR
HPAR=HPAR-3

P70: MOV OR1, DBEG
MOV (R1)+, BKSIZE
ASP BKSIZE
DEL BKSIZE
BKSIZE=DEL/2
MOV #62, BTKSZ
SUB BKSIZE=BTKSZ
BDSZ=62-BKSIZE
DR P63

P80: MOV (R1)+, NR
NR
BR P63

P90: MOV (R1)+, NC
NC
BR P63

P95: INC QFLAG
QUICK LOOK (WHILE SCANNING)
SUB #2, HPAR
BR P18

P100: MOV #YREG, JPAR
JOYSTICK
MOV #XREG, JPAR
STORE SL AND SS
BIC #177000, JPAR
CALL TYPE, (HS,-zero) REQUEST MOVE TO LOWER RIGHT
CALL PARAM, (HPAR, PAR, HPMAX)
MOV #YREG, JPAR+6
STORE ES
MOV #XREG, JPAR+2
EL
BIC #177000, JPAR+2
MOV #JPAR, R2
MOV (R1)+, R0
MOV #4, HPAR
JMP P30  \GO PROCESS AREA PARAMETERS

P159:  TST HR  \DONE WITH PARAMETERS
        BGT P160  \BR IF HR WAS A PARAMETER
        MOV HL,R0
        ADD #32,R0
        ASH 0-6,R0
        BGT P165  \BR IF HL GT 31

P155:  INC R0  \NR=1
        MOV R0,HR  \NR=(HL+32)/64
        TST HC
        BGT P170  \BR IF HC WAS A PARAMETER
        MOV NS,R0
        ADD #32,R0
        ASH 0-6,R0
        BGT P163  \BR IF NS GT 31
        INC R0  \NC=1
        MOV R0,HC  \NC=(NS+32)/64

P169:  MOV R0,HC
        ADD NR,R1
        DEC R1
        CLR R8
        DIV NR,R0
        INC R0
        BIC #1,R0  \MAKE LPS EVEN
        MOV R0,LPS  \LPS=(HL+NR-1)/NR
        MOV NS,R1
        CLR R8
        DIV NC,R9
        BIC #1,R9  \MAKE IT EVEN
        MOV R0,SPS2  \SPS2=NS/NC SPS FOR 2 FIELDS
        ASR R8
        MOV R0,SPS
        ADD #09,R0
        MOV HC,R1
        DEC R1
        MUL SPS,R1
        MOV NS,R0
        ASR R0
        SUB R1,R0  \SPS=NS/2-(NC-1)*SPS
        MOV HC,R1
        ASH #7,R1
        ADD #HIST,R1  \HIST+MC=128
        MOV R1,TEST+2
        SUB R128,R1
        MOV R1,TEST+2
        RTS

THRESH
        MOV #MSG+3,T50+10
        MOV #MSG+6,T60+10

T35:  MOV #HIST+995,-... R4  \HIST+(NC-1)*128
        JSR PC,SUB  \FIND BACK AND DATA
        MOV DPER,R1
        MUL DATA,R1
        MOV DPER,R3
        MUL BACK,R3
        ADD R3,R1
        ADD #50,R1
        CLR R8
        DIV #100,R8  \JTHR=(DPER*DATA+(100-DPER)*BACK*50)/100
        MOV DATA,R1
        SUB BACK,R1
        CMP R1,00  \CHECK FOR SMALL DIFFERENCE BETWEEN D AND B
        BGT T20  \BR IF OK

T20:  MOV BRESZ,R0  \JTHR=3BEC/2 (UNDIVIDED NUCLEUS)
        TST DATA
        BGT T30  \BR IF NO DATA IN SECTOR
        MOV #127,R0  \JTHR=127 IF NO DATA IN SECTOR
        T30:  TST BACK
        BGT T40  \BR IF BACK OK
        CHP BC,0 DTAB  \SEE IF FIRST SECTOR
        BEG T40
MOV BX, R1 ; USE BACK FROM PREVIOUS SECTOR
MOV -(R1), BX
T40:
MOV R0, DX
MOV R0, ITHR
TST DATA
BCT T45
CLR ITHR
T45:
MOV BACK, OBX ; STORE IN TTAB AND BTAB
ADD $2, TX
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 $4, THIST
BNIS T19 ; BR IF MORE HISTOGRAMS TO ANALYZE
SUB $6, T60+10
CLR $06+10
INC SHUM
CALL OUTCON, (SHUM, NO+2, THREE)
ADD NO, SHUM
BEC SHUM
MOV R0, M0+3
CALL OUTCON, (SHUM, M0+3, TWO)
BIT $2, 0000R
BEQ T70 ; BR IF SW 1 DOWN
CALL TYPE, M0
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)+(S03 R4, TH20) ; MOVE A SECTOR HEADING TO MSG
S03 R0, TH20 ;BR IF MORE SECTORS
CLR $02
MOV $06, R4
MOV $09, R2
MOV @THEAD, R3 ; FINAL HEADING
TH33:
MOVB (R3)+(, R2)+(S0B R4, TH39)
BIT $02, 0000R
BEQ T70 ; BR IF SW 1 DOWN
TH70:
RTS PC

SMOOTH: MOV R4, -(SP) ; JSR PC, SMOOTH SAVE R4
MOV @HIS+2, R3
SH23:
MOV $62, R2
MOV (SP)+(, R0)+(S0D R4, R0) ; HIST(I) = HIST(I-1) + HIST(I) + HIST(I+1)
ADD $04, R0
ADD 2(R4), R0
MOV R0, (R3)+
S0B R2, SH23
MOV (SP)+(, R4)
RTS PC

J SR PC, TSUB ; R0 = BACK = HIS+0
MOV @10, R1
MOV BK$12, R2 ; BACK RANGE IS NOMERICALLY 2-40
MOV @HIS+2, R3
TS40:
CMP R1, (R3)+
BGE TS45 ; $2 IF MAX1 GE HIS(I)
MOV -(R3), R1 ; MAX1 = HIS(I)
MOV R3, R8
J SR HJS, R1
TST (R3)+
TS45:
S0B R2, TS40 ; BR IF MORE TO SEARCH
SUB HC, R8
MOV R8, BACK ; STORE BACK
MOV @HIS, R0
MOV 84.R1
MOV DTSIZ.R2
; DATA RANGE IS NOMINALLY FROM 58-124
TS59: CMP R1, (R3)+
BGE TS60
CMP -2(R3), -4(R3)
BLT TS60
; ABORT IF MAX GE HIS(I)
MOV -R3, R1
TS61: SUB @HIS.R0
MOV R0, DATA
; STORE DATA

MAXH=512.
MAXWS=409.

SL: .WORD 8. ; STARTING LINE
EL: .WORD 59. ; ENDING LINE
SS: .WORD 0. ; STARTING SAMPLE
DBEG: .WORD 53. ; DATA BEGIN
SPS: .WORD 38. ; SAMPLES PER SECTOR OF A FIELD
ISPS: .WORD 30. ; INITIAL SPS
OKSIZ: .WORD 24. ; DBEG/2=1
DTSIZ: .WORD 38. ; 32-OOKSIZ
SIZE: .WORD 12288.; SIZE OF EACH 24-LINE BUFFER
STCR: .WORD -.
HIDIM: .WORD 512. ; HIST DIMENSION (512 WORDS)
SPACE: .WORD 49.
EJECT: .WORD 61.
UIC: .WORD 2. ; DEFAULT UIC IS 2,2
SYTV: .WORD 70.
ZERO: .WORD 0.
ONE: .WORD 1. ; CONSTANT
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. ; FLAG FOR BUMAX SCAN
INDEX: .WORD 56. ; REL BUFFER ADDR FROM PUT
HPAR: .WORD -.
JPAR: .BLKW 4. ; JOYSTICK PARAMETERS
PAR: .BLKW 80.
NMAX: .WORD 80.
SPAR: .BLKW 499. ; 10 LINES
NS: .WORD 472. ; 10 SAMPLES
NLR: .WORD 1.
HIST: .BLKW 512. ; 64 WORD HISTOGRAMS
HIS: .BLKW 64. ; SMOOTHED HISTOGRAM
TBATA: .WORD -.
MR: .WORD -.
NR: .WORD -.
LPS: .WORD 63. ; LINES PER SECTOR
SPSZ: .WORD 68. ; SAMPLES PER SECTOR
TTAB: .BLKW 64.
BTC: .BLKW 64.
BPER: .WORD 40. ; DATA PERCENT
TPER: .WORD 40. ; BACK PERCENT
TX: .WORD TTAB+..
BX: .WORD 0TAB+.. BACK: .WORD -.
DATA: .WORD -.
ITHK: .WORD -.
SHUN: .WORD -.
QY: .WORD -.
**Binary Picture Generator**

**SUBROUTINE BINARY(BUNIT.FILPEX)**

**BYTE FILPEX(9)**

**IMPLICIT INTEGER(A-Z)**

**COMMON/C1/NC,LPS,SFS,TTAB(64),BTAB(64),IBER**

**COMMON/C1/SLO,SSD,HLO,NSO,HOB,M,XHOH**

**COMMON/C1/MAREA,MNEPT,MXHPT,MXLEN,NUM,TONM,ISWI**

**COMMON/C1/ WORD(4128),PICT(12344),PBUF(512)**

**INTEGER PAR(230),KEYC(6):SPAR(5),CTAB(307),NTAB(30)**

**BYTE PIC,PMAG(22),PBUF**

**DATA NFILL/6/**

**CALL SWITCH(4. ISW4)**

**IF(ISW4.EQ.1) CALL TIMER**

**CALL SWITCH(1.ISW1)**

**IF(ISW1.EQ.1) CALL TYPE(‘BINARY’)**

**CALL SWAPAR(MPAR,PAR20)**

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

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

**CALL AFILE(PIC,BUNIT,FILPEX,2.2)**

**CALL OPENP(PIC,61441,0,’SCN’)**

**CALL GLABEL(PIC,INDEX)**

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

**C DEFAULT PARAMETERS**

**B = 3**

**MAXHO = 65**

**MXHAR = 32**

**MXGRA = 2000**

**MNEPT = 0**

**MXEPT = 299**

**SLO = 1**

**MAXLEN = 99**

**S30 = 1**


```
101
NLO = SPAR(1)
NSO = NHOB(SPAr(2),311)
CALL GET(PIC, NLO+1.I)
CALL MVW(PIC(I+1), NR, 1337)
C
CALL RPReM(HPAR, PAR, 20)
I = -2
1
J = I + 1
2
I = I + 2
IFI(I, GT, HPAR) GOTO 20
DO 5 K = 1, NK, Y
IFI(PAR(I).NE, KEY(K)) GOTO 3
GOTO(2, 12, 13, 14, 15, 16), K
5
CONTINUE
C
C PARAMETER ERROR
PAUSE 1
GOTO 20
12
TODN = PAR(I+2)
GOTO 1
13
I = I + 2
MNREAA = PAR(I)
MNREA = PAR(I+1)
GOTO 2
14
I = I + 2
MNEPT = PAR(I)
MNPT = PAR(I+1)
GOTO 2
15
MAXLEN = PAR(I+2)

16
B = PAR(I+2)
GOTO 1
20
CONTINUE
IFI(IS4.EQ, 1) CALL TIMER
B = (B+1)/2
SPS2 = (SPS+1)/2
MSAMPS = NLO - (CHR-1)*LPS
NW = NSO/32 + 1
HWT = (NSO-31)/32
HWT = HWT/16
C
**BINARY PICTURE AREA(WORD) IS ASSUMED TO BE ZERO INITIALLY
C
NW = NUMBER OF WORDS OUT PER LINE
C
HWT = 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,HWT
CTAB(W) = C
NTAB(W) = 16 -
IFI(C.EQ,0-HD-1) GOTO 50
S = S + 16
IFI(S.LT, SPS2) GOTO 50
S = S - SPS2
NTAB(W) = 16 - S
C = C + 1
50 CONTINUE
IFI(IS4.EQ, 1) CALL TIMER
C
S1 = 1
WI = NW
REC = 1
ILPS = LPS
C
DO 300 R=1,NR
IFI(R.EQ, NR) ILPS = MSAMPS
C
DO 290 L=2,ILPS,2
CALL GET(PIC, REC+PI)
REC = 0
CALL MVW(PIC(P1+1), PBUF, HUI)
CALL GET(PIC, 0.P1)
```
SUBROUTINE SEGNET

COMMON /C/ S1(73), S2(79), S3(70) /S1, S2, S3, ...

C DO 289 U=NRU
C CALL STH \( PBUF, S, YOR, D<, U, I + U \)   X=(Y)  YMAX
C CONTINUE

C WRITE PARAMETERS IN CASE OF READ
C CLOSE FILE
C RETURN
C END

C THE EDGE POINT DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJECT
C
C THE EDGE POINTS ARE COORDINATE PARTS DESCRIPTION SECTIONS
C
C READ FILE ( LIGHT ) CALL PARARM ( Nº, Nº, Nº )
C
C S = S + 1n
C CONTINUE
CONTINUE

REC = 3
NW = NW / 2
NW2 = NW2
NL = NLOD / 2
N = NS0 / 2

NOB = NUMBER OF OBJECTS
WI = WORD INDEX
EWI = ENDING WORD INDEX
EI = EDGE INDEX
DI = BIT INDEX (*0, 1, ..., 31)

EWI = NW

DI = 1
NOB = .1

SCAN BINARY PICTURE (WORD) LINE-BY-LINE

DO 200 L = 1, NL
   WI = EWI + 1
   EWI = EWI + NW
   LOOP = 0

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

140 IF DRAWH(WI, BI, WORD, EWI, MASK) EQ .01 GOTO 200
    LOOP = LOOP + 1
    IF LOOP EQ 200 PAUSE 1
    EDGE(1) = L
    EDG5(2) = (NW - (EWI - WI + 1)) * 16 + BI + 1
    CALL TURTLE(N, EDGE, WORD(WI), MASK, NW2)
    - IF N EQ .03 GOTO 140
    H2 = 2 * H
    IF (NOB, NE, TOSH) GOTO 141
    CALL L1LIST(NOB, H, YMIN, XMIN, YMAX)
    CALL PDPUMP(EDGE, EDGE(N2), 1)

141 CONTINUE
    IF (H, EQ .255) GOTO 142
    CALL SORTIN(EDGE, H, SIND)
    IF (NOB, EQ. TOSH) CALL PDPUMP(EDGE, EDGE(N2), 1)
    IF (SIND, EQ .0) GOTO 145

142 IF (H, EQ. MAXH) GOTO 143
    CALL TYPE(Too MANY EDGES!"
    CALL TYPE("

143 GOTO 200
    CONTINUE
    IF (TSWI, EQ .1) CALL TYPE("
    IF (ISW1, EQ .1) CALL STIME
    CALL SORTIN(EDGE, H)
    IF (ISW1, EQ .1) CALL TIMER

C

N = NUMBER OF EDGE POINTS

145 CALL ERASE(WI, EDGE, AREA, H2, NW)
    IF (AREA, LT, H2, OR. CT, MXEP3) GOTO 195
    IF (AREA, LT, HNAR, OR. AREA, CT, MXAR) GOTO 196
    IF (YMAX - YMIN, GE, MXLEN, OR. XMAX - XMIN, GE, MXLEN) GOTO 196

C

ENTER OBJECT INTO DIRECTORY

150 IF (HK, GE .0, OR. TOSH, GE .0) GOTO 175
    NK = HK + 1
    LIP(HK) = EDGE(1) + EDGE(1)
    SIP(HK) = EDGE(2) + EDGE(2)

C

STORE NGO INFORMATION FOR QUICK COUNT OPTION

151 Y = +EDGE(1)
    X = +EDGE(2) - 14
    IF (X LT 0) X = 0

C

WRITE AN ARROW BY THE OBJECT

CALL DLINE(WHITE, Y, X, 8, 1, 0)
    CALL DLINE(WHITE, Y, X, 8, 1, 0)
    CALL SSATCH(5, ISW6)
    IF (ISW6, EQ .1) GOTO 175
    CALL DLINEWHITE, Y, X, 8, 2, 1, 0
    CALL DLINE(White, Y, X, 8, 2, 1, 0)
    CALL DLINEWHITE, Y, X, 2, X, +10, 2, 1, 0
    CALL DLINEWHITE, Y + 2, X, +10, 2, 1, 0
    CALL DLINEWHITE, Y + 2, X, +10, 2, 1, 0
    CALL DLINEWHITE, Y + 4, X, 8, 2, 1, 0
    CALL DLINEWHITE, Y + 4, X, 8, 2, 1, 0

C
175 CONTINUE
DIRD(I) = REC
DIRD(I+1) = N
DIRD(I+2) = MAX0(YMIN-O.1)
DIRD(I+3) = MAX0(XMIN-O.1)
DIRD(I+4) = MIN0(YMAX+B.NL)
DIRD(I+5) = MIN0(XMAX+B.NS)
NOB = NOB + 1
I1 = I1 + 6
IF(NOBI.GT.MAXNOB) GOTO 201
C
IF(TODN.LT.0) GOTO 140
C DON'T WRITE IF QUICK COUNT OPTION
D0 195 1=1,N2,512
CALL WRITE(EBUF,REC,EDGE(I))
195 REC = REC + 1
GOTO 145
C
196 CALL SSUTCH(4,IS4)
IF(IS4.EQ.1) GOTO 140
CALL QPRINT( ' OBJECT REJECTED ' )
CALL ILIST(NOBI.N,AREA,YMIN,XMIN)
CALL ILIST(YMAX-XMIN,XMAX-XMIN,MAXR,MAXP,XLEN)
GOTO 140
C
200 MAXN = MAXN + NUM
C
201 NOBI=NOBI-1
IF(TODN.LT.0) GOTO 220
IF(DI.GT.1) CALL WRITE(EBUF,BREC,DIR)
CALL WRITE(EBUF,1,SPAR)
CALL CLOSE(EBUF)
220 IF(NOBI.GT.0) GOTO 200
C IF NO OBJECTS FOUND, TYPE MSG AND CALL INT1
CALL TYPE( ' NO OBJECTS FOUND; CHECK THRESHOLDS' )
CALL TYPE( ' B' )
CALL APHASE(7)
2000 CONTINUE
IF(TODN.EQ.0) RETURN
CALL MVL(' M=**',M.6)
CALL OUTCON(NK,M(6).2)
CALL TYPE(M.6)
CALL UPARAM(122,NK,6)
CALL APHASE(2)
C WRITE NOB PARAMETERS AND CALL INT1
END

SUBROUTINE SKIRT
IMPLICIT INTEGER(A-Z)
COMMON/CI/SPAR(133),SLG,SSO,MLD,NOB,B,SPER
COMMON/CI/DIR(512),SDIR(512)
COMMON/CI/EBUF(56),EDGE(1924),SEG6(56),SEG8(192)
COMMON/CI/SBUF(960).ESI(5)
INTEGER PAR(5)
BYTE EBUF,SEC
DATA MAXS/999/,MAXD/510/,.S/1/,.DI/1/
DATA BREC/2/,IDREC/2/
C THE SEGMENT DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJECT
C SDIR(DI) = BSIC(BEGINTING SEGMENT INDEX)
C SDIR(DI+1) = ESIC(ENDING SEGMENT INDEX)
C SDIR(DI+2) = YMIN
C SDIR(DI+3) = XMIN
C SDIR(DI+4) = YMAX
C SDIR(DI+5) = XMAX
C THE SEGMENTS ARE STORED AS TRIPLETS
C SEC(I) = Y
C SEC(I+1) = XI
C SEC(I+2) = X2
C THE ROB DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJECT
C DIR(DI) = BACKGROUND
C DIR(DI+1) = THRESHOLD
C DIR(DI+2) = YMIN
CALL SSOTCH(1, ISW1)
IF(ISW1 .EQ. 1) CALL TYPE(' SKIRT')
CALL AFIEL(EBUF, 1, 'PIC', .5, 5)
CALL OPEN(EBUF, 1024, 0, 'MV2')
CALL AFIEL(SEC, 5, 'PIC', .5, 5)
CALL OPEN(SEC, 1024, 0, 'MV3')
CALL GLABEL(EBUF, PAR, IND)
CALL MVL(' SEGMENT FILE', EBUF(IND+1), 13)
CALL PLABEL(SEC, PAR, EBUF(IND+1))
CALL PUT(SEC, 1, IND)

CALL SSOTCH(1, ISW1)
IF(ISW1 .EQ. 1) CALL TYPE(' SKIRT')
CALL AFIEL(EBUF, 1, 'PIC', .5, 5)
CALL OPEN(EBUF, 1024, 0, 'MV2')
CALL AFIEL(SEC, 5, 'PIC', .5, 5)
CALL OPEN(SEC, 1024, 0, 'MV3')
CALL GLABEL(EBUF, PAR, IND)
CALL MVL(' SEGMENT FILE', EBUF(IND+1), 13)
CALL PLABEL(SEC, PAR, EBUF(IND+1))
CALL PUT(SEC, 1, IND)

C
CALL SSOTCH(1, ISW1)
IF(ISW1 .EQ. 1) CALL TYPE(' SKIRT')
CALL AFIEL(EBUF, 1, 'PIC', .5, 5)
CALL OPEN(EBUF, 1024, 0, 'MV2')
CALL AFIEL(SEC, 5, 'PIC', .5, 5)
CALL OPEN(SEC, 1024, 0, 'MV3')
CALL GLABEL(EBUF, PAR, IND)
CALL MVL(' SEGMENT FILE', EBUF(IND+1), 13)
CALL PLABEL(SEC, PAR, EBUF(IND+1))
CALL PUT(SEC, 1, IND)

C
S = SEGMENT INDEX
SI = SEGMENT BUFFER INDEX
DO 200 08N=1, N8B
IF(DI .NE. 1) GOTO 30
CALL READ(EBUF, IREC, IND, DIR)
DO 30 IREC = (IDREC + 1)
CALL MVW(DIR, SDIR, 512)
I8RC = IDREC
SIDR(DI) = S
CALL INTERP(DI)
NL8 = H*H
NREC = (N8R+511)/512
DO 130 L=1, NREC
CALL READ(EBUF, IREC, IND, EDGE)
I8RC = IREC + 1
E8W = MIN0(N8R, 512)
ADD HORIZONTAL SKIRT TO SEGMENTS
DO 49 E=2, E8W + 1
EDGE(E) = MAX8(EDGE(E), B, NS)
49
EDGE(E+2) = MIN8(EDGE(E+2), B, NS)
C
IF(L .NE. 1) GOTO 60
Y0 = EDGE(1)
YBAR = Y0 - B
C
USE FIRST SEGMENT END POINT PAIR TO INITIALIZE SBUF
XI = EDGE(2)
X2 = EDGE(4)
C
SI = 1
I = 1
E1 = 5
DO 50 CONTINUE
C
IF(SI .GT. MAXSB) SI = 1
C
PROCESS REMAINING SEGMENT END POINTS
DO 120 E=E1, E8W, 4
Y = EDGE(E)
IF(Y .EQ. Y0) GOTO 90
C
ADVANCE SBUF ONE ROW
S1 = SI + SINC
I = I + 1
IF(S1 .GT. MAXSB) SI = 1
IF(I.GT.82) I=1
IF(YBAR.LT.0) GOTO 82
SEND = ESI(I)
DO 80 SIP=S1,SEND,2
SEG(S) = YBAR
SEG(S+1) = SBUF(SIP)
SEG(S+2) = SBUF(SIP+1)
80 S = S + 3
C
82 Y0 = Y0 + 1
YBAR = YBAR + 1
SBUF(S1) = EDGE(E+1)
SBUF(S1+1) = EDGE(E+3)
ESI(I) = S1
C
90 SIP = 1
X10 = EDGE(E+1)
X20 = EDGE(E+3)
C
DO 100 I1=1,02
SIP = SIP
SEND = ESI(I)
X1 = X10
X2 = X20
C
93 XP1 = SBUF(S1Q)
IF(X2.LT.XP1-1) GOTO 95
XP2 = SBUF(S1+1)
IF(X1.GT.XP2+1) GOTO 95
SEGMENT (X1,X2) IS CONNECTED TO A PREVIOUS SEGMENT
IF(S1Q.EQ.SEND) GOTO 94
X1 = MINB(X1,XP1)
X2 = MAXB(X2,XP2)
SBUF(S1Q) = SBUF(SEND)
SBUF(S1Q+1) = SBUF(SEND+1)
SEND = SEND - 2
GOTO 93
C
94 SBUF(SSEND) = MINB(X1,XP1)
SBUF(SEND+1) = MAXB(X2,XP2)
GOTO 98
C
95 S1Q = S1Q + 2
IF(S1Q.LE.SEND) GOTO 93
SBUF(S1Q) = X1
SBUF(S1Q+1) = X2
C
98 ESI(I) = S1Q
100 SIP = SIP + SINC
C
120 CONTINUE
C
EI = 1
130 KSEND = KSEND - EI
C
RELEASE REMAINING SEGMENTS
DO 150 I1=1,82
IF(YBAR.GT.KL) GOTO 180
I = I + 1
IF(S1.GT.MAXS3) S1=1
IF(I.GT.82) I=1
SEND = ESI(I)
DO 140 SIP=S1,SEND,2
SEG(S) = YBAR
SEG(S+1) = SBUF(SIP)
SEG(S+2) = SBUF(SIP+1)
140 S = S + 3
C
150 YBAR = YBAR + 1
C
180 SDIR(DI+1) = S - 1
C
C

DI = DI + 6
IF(DI.LT.MAXDI) GOTO 200
CALL WRITE(SEGB, DREC, DIR)
CALL WRITE(SEGB, DREC+1, SDIR)
DREC = DREC + 2
DI = 1
200 CONTINUE

C

IF(DI.EQ.1) GOTO 205
CALL WRITE(SEGB, DREC, DIR)
CALL WRITE(SEGB, DREC+1, SDIR)
C WRITE OUT SEGMENT

205 DO 210 I=1, 512
CALL WRITE(SEGB, DREC, SEG(I))
210 GREC = OREC + 1
CALL CLOSE(SEGB)
CALL CLOSE(EBUF)
END

SUBROUTINE CHROME(DUNIT, FILPEX)
BYTE FILPEX(9)
IMPLICIT INTEGER(A-Z)
COMMON/C1/SL0, S5O, HLO, HSD, HOB, B
COMMON/C1/DIR(512), CMVB(1852), C(512), PBUF(1388)
C THE FIRST 28 WORDS OF PBUF ARE THE MVWB, FOLLOWED BY 1824 WORDS OF
C BUFFER STORAGE, FOLLOWED BY 256 WORDS OF PICTURE STORAGE
COMMON/C1/SEGB(56), CBUF(560), CDIR(512), LBUF(100)
COMMON/C1/OREC, NSFI
BYTE SEGB, PB(2616)
EQUIVALENCE (PB, PBUF)
INTEGER PAR(5)
DATA SECSIZ/64, SSM1/63 /
DATA MAXDI/512, MAXBUF/5688, IDREC/2, PBUF/1852, DREC/2/
C LPBUF IS THE LENGTH OF PBUF MINUS 256 WORDS OF PICTURE STORAGE
C
CALL SSUTC(1, ISW1)
IF(ISW1.EQ.1) CALL TYPE(' CHROME')
CALL SSUTC(4, ISW4)
IF(ISW4.EQ.1) GOTO 3
CALL SPRINT(' CHROME')
CALL TIMER
MAXBUF=7328
Z CONTINUE
CALL AFIL(E(PBUF, DUNIT, FILPEX, 2, 2)
CALL OPEN(PBUF, LPBUF=50, 1, 0, 0, 'SCN')
CALL PLABEL(PBUF, IP)
N=PAR(2)+11/2
CALL AFIL(E(SEGB, 5, 'PIC ', 5, 5)
CALL OPEN(SEGB, 1824, 0, 0, 'MVWB')
CALL GLABEL(SEGB, PAR, INH)
CALL ZIA(CMVB, 28)
CALL AFIL(E(CMVB, 1, 'PIC ', 5, 5)
CALL OPEN(CMVB, 1824, 1, 1, 'MVWB')
CALL MVL(' CHROME FILE', PB(IP+188), 12)
CALL PLABEL(CMVB, PAR, PB(IP+1))
CALL READ(SEGB, 1, INH, SLO)
CALL PUT(CMVB, 1, IC)
CALL MV(L(SLO, CMVB, IP+188), 512)
NREC = (HOB+64)/85
IREC = 2*HREC + 2
DREC = HREC + 3
CALL READ(SEGB, IDREC, INH, DIR)
CALL PUT(CMVB, DREC, IC)
CALL MV(L(DIR, CMVB(2), 512)
CALL READ(SEGB, IDREC+1, INH, DIR)
IDREC = IDREC + 2
DREC = DREC + 1
C FORMAT CBUF
SEND = HOB+6
SEND = DIR(SEND-4)
DISP = MAXBUF-SEND
C
J = DISP + 1
DO 5 I=1,SEND+512
CALL READ(SEGB,IREC,IND,CBUF(J))
G = J + 512
5 IREC = IREC + 1
DO 6 DI=1,BEND,6
DIR(DI) = DIR(DI-1) + DISP
6 DIR(DI+1) = DIR(DI+1) + DISP
THE (DISP-1)/SECSIZ
N = THS/SECSIZ
DO 10 I=1,N,SECSIZ
CBUF(I) = I*SECSIZ
10 LFSI = N - SS1
CBUF(LFSI) = 0
C THE SEGMENT DIRECTORY CONTAINS THE FOLLOWING INFORMATION FOR EACH OBJ
C
C DIR(DI) = CSI = CURRENT SEGMENT INDEX
C DIR(DI+1) = ESI = ENDING SEGMENT INDEX
C DIR(DI+2) = YMIN
C DIR(DI+3) = XMIN
C DIR(DI+4) = YMAX
C DIR(DI+5) = XMAX
C
C C THE CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING INFORMATION FOR EACH OBJ.
C
C CDIR(DI) = BSI = BEGINNING SEGMENT INDEX
C CDIR(DI+1) = NS = TOTAL NUMBER OF SECTIONS USED UP BY OBJ
C CDIR(DI+2) = FSI = INDEX-OFFSET 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 CDIR(DI+5) = LWS = LAST HALFWORD OF SECTION USED BY OBJECT
C
C HFSI = NEXT FREE SECTION INDEX
C LFSI = LAST FREE SECTION INDEX
C
C DI = 1
C
50 DBEG = DI.
IF(ISW4.EQ.1) CALL TIMER
C INCREASE NUMBER OF SECTIONS IF POSSIBLE.
C CSI = DIR(DI)
C BSI = CDIR(DI)
C IF(BSI.EQ.0) BSI = CSI
C NSI = LFSI + SECSIZ.
C IF(NSI.GT.BSI-SECSIZ) GOTO 52
C CBUF(LFSI) = NSI
C LFSI = NSI
C GOTO 51.
C
52 CBUF(LFSI) = 0
Y0 = CBUF(CSI)
REC = 2*Y0 - 1
CALL GET(PBUF,REC,IND)
CALL MV(PBUF,IND+2,1),PBUF(LBUF+1),NU)
CALL GET(PBUF,0,IND)
IP=IND/2
C
C PROCESS OBJECTS 1 TO NOB.
C IF(ISW4.EQ.1) CALL TIMER
DO 100 DI=DBEG,BEND,6
CSI = DIR(DI)
C IF(CSI.EQ.0) GOTO 100
Y = CBUF(CSI)
C IF(Y.GT.Y0) GOTO 110
ESI = DIR(DI+1)
LSI = CDIR(DI+3)
NWS = CDIR(DI+5)
HSEC = 0.
IF(LSI.NE.0) GOTO 58
C FIRST SEGMENT OF OBJECT. ASSIGN A SECTION
C IF(LFSI.EQ.0) CALL TYPE( "WMBUF" )
C GOTO 58.
LOOP THROUGH EACH SEGMENT OF CURRENT OBJECT

C

DO 90 SI=CSI,CSI+3

Y = CBUF(SI)

IF(Y.CT.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(LHW.LT.SECNZ) GOTO 60

ADD ON A NEW SECTION

HFSI = HFS1

C

DO 50 CONTINUE

C

C

DO 100 CONTINUE

C

DO 118 CONTINUE

C

IF(ISW4.EQ.1) CALL TIMER

DO 120 DI=BEG, BEND+6

C

C

END
SUBROUTINE ROB
IMPLICIT INTEGER (A-Z)
COMMON / C1 / CMVB (56), C512
COMMON / C1 / CMVB (56), EDGE (1024), LBW (50), BUF, 6 TBL
COMMON / C1 / SLG, SGL, WGL, WGO, WGR, B, DPER
COMMON / C1 / PERIM, YMNI, YMNI, YMNI, YMNI
COMMON / C1 / ZDIR (60), TTAB (60), AREK (60), ID0 (60)
COMMON / C1 / NK, LINE (50), SLP (60), PERM (60), CIRCA (60)
LOGICAL = 1 BUF (1024)
BYTE CMVB, SNVB, STBL (128), MSG (20), MSGB (25), TMSG (128)
INTEGER DIR (510), PAR (100), HIS (60), OTTAB (50)
EQUIVALENCE (DIR, SDIR (145)), (HS, EDGE)
DATA MAXBUF / 100000, RECSIZ / 1024, NOBPH / 6
DATA HC / 10370, 1, HP1G, 0, MAXCMVB / 400, TOB, 8, MAXNP / 188
DATA ORC / 2, 2, DRC / 2, DRC / 2, RFLAG / 1, OTTAB / 60, 80, 80

C THE INPUT CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJ
C
C DIR (DI) = BACKGROUND
C
C DIR (DI+1) = THRESHOLD
C
C DIR (DI+2) = CNS (MINIMUM LINE)
C
C DIR (DI+3) = CMS (MINIMUM SAMPLE)
C
C DIR (DI+4) = CMXL (MAXIMUM LINE)
C
C DIR (DI+5) = CMXS (MAXIMUM SAMPLE)
C
C THE OUTPUT CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJ
C
C SDIR (CDI) = FBD (FIRST RECORD BLOCK WRITTEN TO DISK)
C SDIR (CDI+1) = MML (MINIMUM LINE)
C SDIR (CDI+2) = MS (MINIMUM SAMPLE)
C SDIR (CDI+3) = MLW (LAST RECORD BLOCK WRITTEN TO DISK)
C
C THE INPUT CHROMOSOMES ARE IN THE FOLLOWING FORMAT
C
C C(1) = PREVIOUS BLOCK WRITTEN
C C(2) = HSEG (NUMBER OF SEGMENTS IN THIS RECORD)
C C(3) = Y (LINE COORDINATE 0 - 255)
C C(4) = X (STARTING SAMPLE COORDINATE 0 - 255)
C C(5) = N (NUMBER OF SAMPLES IN SEGMENT)
C
C NOTE THAT TWO CONSECUTIVE LINES OF SEGMENT GREY VALUES ARE PRESENT
C ALSO THE DIMENSIONS ARE THOSE OF THE HALF-PICTURE AND MUST BE
C DOUBLED.
C
C THE OUTPUT CHROMOSOMES ARE IN THE FOLLOWING FORMAT
C
C C(1) = HSEG (NUMBER OF SEGMENTS IN THIS RECORD)
C C(2) = Y (LINE COORDINATE 1 - 512)
C C(3) = X (STARTING SAMPLE COORDINATE 1 - 512)
C C(4) = N (NUMBER OF SAMPLES IN SEGMENT)
C
C CALL SSWITCH (1, ISW1)
C IF (ISW1, EQ, 1) CALL TYPE (' ROB')
C CALL SSWITCH (4, ISW1)
C IF (ISW1, EQ, 1) CALL SPRINT (' ROB')
C CALL RPARAM (HP, PR, MAXNP)
C IP = 1
C IF (IP, GT, HP) GOTO 33
C IF (PAR (IP), EQ, ' 03') GOTO 30
C IF (PAR (IP), NE, ' OT') GOTO 35
C OTTAB (PAR (IP+2)) = PAR (IP+3)
C IP = IP+4
C GOTO 20
C TOB = PAR (IP+2)
C IP = IP+3
C GOTO 20
C CONTINUE
RECSIZ = RECSIZ / 2
C LBW (30) = LAST BLOCK WRITTEN FOR CURRENT CHROMOSOME
C NF = DIRECTORY INDEX
C NC = NUMBER OF CHROMOSOMES
C CALL AFFILE (CMVB, 1, ' PIC ', ' .5.5')
C CALL OPEN (CMVB, 1024, 6, ' MV2 ')
C CALL AFFILE (CMVB, 5, ' UCR ')
C CALL OPEN (CMVB, 1024, 6, ' UCR ')
C CALL GLABEL (CMVB, PAR, IND)
C CALL MVL (' ROB FILE ', CMVB (IND+139), 12)
C CALL GLABEL (MVIR, PAR, CMVB (IND+1))
C CALL SSWITCH (0, ISW1)
C IF (ISW1, NE, 1) GOTO 37
C CALL SPRINT (' 1 ')
C CALL SPRINT (CMVB (IND+73), 79)
CALL QPRINT('CMVB(IND=140).39')
CALL QPRINT('CMVB(IND=217).79')
CALL QPRINT('CMVB(IND=218).88')

121

<table>
<thead>
<tr>
<th>SL</th>
<th>SS</th>
<th>HL</th>
<th>HS</th>
<th>T1</th>
<th>T2</th>
<th>FPDM</th>
<th>AREA</th>
<th>109/8</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CALL ITLA(32, TMSG.128)
CALL PUT(SMVB, DREC.IND)
CALL READ(CMVB, 1, IND, SLO)
CALL READ(CMVB, DREC.IND, DIR)
DREC = DREC + 1
CALL READ(CMVB, DREC.IND, LBW)
DPER = 100 - DPER
B = 2*B
B1 = B + 1
B2 = B + 2
B3 = 2*B + 1
DI = 1

C
CBD = 2

LOOP THROUGH EACH OBJECT

DO 500 OBN=1, NOB
REC = LBW(OBN)
C1 = DIR(DI*2)+2 - 1
CNS = DIR(DI*3)+2 - 1
CMXL = DIR(B1+4)+2
CMXS = DIR(B1+5)+2
CNL = CMXL + CAL + 3
CMSL = CMS - CNS + 2
CMXLI = CAL - 1

IF(CNL+CNS+LT.MAXBUF) GOTO 50

CALL SWITCH(1, ISUI)

IF(ISUI.EQ.1) GOTO 500
CALL MVL('OBJECT TOO LARGE', MSG7.20)
CALL OUTCON(OBN, MSG7(10), 2)
CALL TYPE(MSG7.20)

GOTO 500

C
C READ IN CURRENT CHROMOSOME

50
CALL ZIA(BUF,MAXBUF/2)
CALL ZIA(HIS.64)

C

51
CALL READ(CMVB,REC,IND,C)
HSEC = C(2)
CL = 3

C

DO 60 NN=1, HSEC
B1 = 2+CSSCLI+CNS+CSSCL+1)+CNS+1
W = C(CI+2)
H2 = 2+W

IF(B1.LT.1) GOTO 60
C

IGNORE POSSIBLE BUF FROM CHROME OUTPUT

C

IF(B1.GT.MAXBUF) GOTO 45

CALL HI5(CI-3).BUF(BI).HIS.H2)
CALL NHI5(C(CI+N+3).BUF(BI+CNS).HIS.H2)

60

C

CI = CI + H2 + 3

C

PEC = C(1)

IF(REC.NE.0) GOTO 51

C

IF(REC.NE.0) GOTO 51

C

C

63
CONTINUE

C

BACK = DIR(BI+1/2)
ITHRES = DIR(BI+1/2)
C

RESET OBJECT

HIS(1) = 0
DATA = ITHRES
THRESH = ITHRES
IF(FLAG.EQ.0) GOTO 72
HIS(64) = 0
MAX = 0

C
B0 65 I=I+THRES/63
F = HIS(I-1) + HIS(I) + HIS(I+1)
IF(F.LT.MAX) GOTO 65
MAX = F

DATA = 1

65 CONTINUE
THRESH = (OPER+BACK + BPER+DATA)/100
ICFO=OTTAB(OBM).NE.0) THRESH=OTTAB(OBM)/2
72 CALL STRECH(HIS,STBL,THRESH,DMAK)

C
BI = BUFFER INDEX
EI = EDGE INDEX
DI = DIRECTORY INDEX
CDI = CHROMOSOME DIRECTORY INDEX
EBI = CHS
EI = 1
CHL2 = CHL - 2
CHS1 = CHS - 1
THRESH = THRESH-2
MLB = CHL2 - B
MS = CHS1 - B
BCNS = B+CHS

C
DO 389 L=1,CHL2
BI = EBI + 1
EBI = EBI + CHS

C
210 IF(ISEG(BI, BUF, THRESH, EBI).EQ.0) GOTO 388
SBIR(CD) = OREC+1
EDGE(I) = L
S = CHS (EBI - BI)
EDGE(2) = S
CALL GOTH(EDGE, BUF(BI), THRESH, CHS)
IF(HE.EQ.0) GOTO 210
IF(OBH.EQ.TOBM) CALL ILIST(N, YMIN, XMIN, XM, XMAX)
IF(OBM.EQ.TOBM) CALL PBUMP(EDGE, EDGE(2+H), 1)
110 IF(HE.333) GOTO 217
CALL SORTIN(EDGE, H, SIND)
IF(SIND.EQ.0) GOTO 215
IF(N.EQ.512) GOTO 214
CALL TYPE('EDGE OVERFLOW', THRESHOLD RAISED')
CALL TYPE('')
THRESH = THRESH + 4
GOTO 210
214 IF(ISW1.EQ.1) CALL TYPE('SORT')
IF(ISW4.EQ.1) CALL STIMER
CALL SORT(EDGE, H)
IF(ISW4.EQ.1) CALL TIMER
215 CONTINUE

IF(OBH.EQ.TOBM) CALL PBUMP(EDGE, EDGE(2+H), 1)
IF(OBM.EQ.TOBM) CALL ILIST(L, S, CHL, CHS, OBM)
C MAKE SURE THAT OBJECT CONTAINS AN INTERNAL POINT
INTPNT = 0
H2 = 2*N
DO 325 E=1, H2-4
Y = EDGE(E)
IF(Y.LT.B1.OA. Y.GT.NLB) GOTO 235
X1 = MAX(XEDGE(E+1), BI)
X2 = MINXEDGE(E+3), HBB
IF(X1.GT.X2) GOTO 235
BB = XCHNS + X1
I = BB - B

C
DO 228 LEFT=1, BI
IF(IV(BUF(LEFT)).NE.0) GOTO 221
220 CONTINUE

C
221 RIGHT = BB + X2-X1 + 0
C
DO 222 I=1, BI
IF(IV(BUF(RIGHT)).NE.0) GOTO 223
222 RIGHT = RIGHT - 1
125

| LEFT  = LEFT  + 1  |
| RIGHT = RIGHT - 1  |

IF(TOBH.EQ.0) CALL ILIST(X1,X2,LEFT,RIGHT,Y)
IF(LEFT.GT.RIGHT) GOTO 235

C

DO 230 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

225 CONTINUE

C THIS IS AN INTERNAL POINT
GOTO 236

C

CONTINUE

C

CONTINUE

C

DO 250 E=1,N2,4
   Y = EDGE(E)
   X1 = EDGE(E+1)
   X2 = EDGE(E+3)
   NS  =  X2 - X1 + 1
   IF(NS.GT.0) GOTO 230
   CALL TYPE('ROB ~ NS LE 0')
   CALL TYPE('0')
   CALL ILIST(X2,X1,X2,NS)
   PAUSE
GOTO 230

C

CONTINUE

MS2 = (NS+1)/2
BCI = ECI + 1
ECI = ECI + MS2 + 3
IF(ECI.LE.RECSZ2) GOTO 240

C

WRITE OUT CHROMOSOME LINE

C(1) = HSEG
OREC = OREC + 1
CALL WRITE(GMVB,OREC,C)
HSEG = 0
BCI = 2
ECI = MS2 + 4

C

240 C(BCI) = CMLI + Y
C(BCI+1) = CNSM1 + X1
C(BCI+2) = HS
CALL REMOVE(BUF(I*CHS+X1),C(BCI+3),STL,MS,IODN,THRESH)
NP = NP + HS
250 HSEG = HSEG + 1

C

IF(NP.GE.25 .AND. INTPTM.NS.0) GOTO 260
CALL SWITC(4,15544)
IF(CSWSW.NE.4) GOTO 210
CALL PRINT('OBJECT REJECTED')
CALL ILIST(OMN,N,NP,YNIN,XMIN)
CALL ILIST(YMAX-YMIN,XMAX-XMIN,XS,NC)
GOTO 210

C

WRITE OUT LAST LINE

260 OREC = OREC + 1
C(1) = HSEG
CALL WRITE(GMVB,OREC,C)

C

NC = NC + 1
HL = YMAX - YMIN + 1
HS = XMAX - XMIN + 1
IODN = IODN - BACK*NP/4
TTAB(NC) = .THRESH.
LIP(NC) = CMLI + L
SIP(HC) = CMSM1 + S
IOD(HC) = IODH
AREA(HC) = HP
PERM(HC) = PERM
CALL SWITCH(0. ISW0)
IF(ISW0.NE.1) GOTO 200

C PRINT IF SW0 IS UP
CALL OUTCON(NC, TMSG(4), 2)
CALL OUTCON(PERM(HC), TMSG(18), 3)
CALL OUTCON(BACK+2, TMSG(17), 3)
CALL OUTCON(DATA+2, TMSG(27), 3)
CALL OUTCON(MAX+2, TMSG(31), 3)
CALL OUTCON(CNL, TMSG(39), 4)
CALL OUTCON(CMS, TMSG(47), 4)
CALL OUTCON(CNL+1, TMSG(33), 5)
CALL OUTCON(CMS+1, TMSG(60), 5)
CALL OUTCON(CNL, TMSG(65), 5)
CALL OUTCON(NS, TMSG(78), 5)
CALL OUTCON(NTHRES+2, TMSG(69), 3)
CALL OUTCON(THRESH, TMSG(96), 3)
CALL OUTCON(CNL, TMSG(94), 2)
CALL OUTCON(NP, TMSG(182), 9)
CALL OUTCON(IDDH, TMSG(111), 9)
CALL OUTCON(IDDH/(NP/8), TMSG(116), 3)
CALL OUTCON(HC, TMSG(122), 2)
CALL QPRINT(TMSG, 120)

C ENTER OBJECT IN CHROMOSOME DIRECTORY
200 SDIR(CDI+1) = CNL + YMIN - 1
SDIR(CDI+2) = CMS + XMIN - 1
SDIR(CDI+3) = OREC
CDI = CDI + 4
IF(NC.EQ.MAXNOB) GOTO 510
GOTO 210

C 300 CONTINUE
350 D1 = D1 + 6

C 510 CONTINUE
SDIR(1) = CDI/4
NK=SDIR(1)
TEMP=PERM(1)
PERM(1)=64

C STORE NOB PARAMETER TEMPORARILY IN PERM(1)
CALL WPARAM(122, NK, NOBPH)
PERRM(1)=TEMP

C WRITE NOB PARAMETERS
CALL MVV(STAB, SDIR(241), 69)
CALL WRITE(SMYB, 1, SDIR)
CALL WRITE(SMYB, 2, LIP)
IF(SW0.EQ.1) CALL QPRINT('I')
CALL CLOSE(SMYB)
CALL CLOSE(CMV)
END.

C********
C* NOB = CALMS VERSION
C*
C********
C* NOB NUMBERS THE Raw SPREAD WITH THE OBJECT NUMBERS GENERATED
C* BY FOB
C
C SUBROUTINE NOB(BUHIT, FILPEX)
IMPLICIT INTEGER(*-Z)
COMMON/CL/ NK, LT, ST, FI
LOGICAL ORDER
BYTE FILPEX(2)

C I/O BUFFERS
BYTE 1 BUF2(12343), OBUF(12343)
C RANDOM AREAS
INTEGER SPAR(18)
BYTE CHAR(2)
C SAMPLE AND LINE TABLE FOR OBJECT NUMBERS

INTEGER ST(60), LT(60), TEMP(60)
DATA Y/0/ X/0/ REPL/1/ MASKPN/13/
C
C. INITIALIZE DATA SETS
C
CALL SSWICH(1, ISW1)
C
C. READ PARAMETERS
C
CALL RPARAM(H1, HK, 12)
C
C. CLEAR THE GRAY SCALE UNLESS FINISHING UP A COUNT.
CALL AFIL(CIBUF2, DUNIT, FILPEX, 2, 2)
CALL AFIL(OBUF, 1, 'PIC', 1, 'S', 'S')
CALL OPEN(CIBUF2, 6144, 1, 'SCN')
CALL OPEN(OBUF, 6144, 1, 'N0B')
CALL GLABEL(CIBUF2, SPAR, 12)
N1=SPAR(1)
N2=SPAR(2)
N=N1/2
N1=1
CALL FLABEL(OBUF, SPAR, CIBUF2(12+1))
C
C. SEE IF LT IS IN ORDER
ORDER=TRUE
IF(HK.LT.2) GOTO 45
DO 40 N=2,HK
IF(LT(NLT.H(N-1))) GOTO 42
40 CONTINUE
GOTO 45
42 ORDER=FALSE.
45 CONTINUE.
C
C. RUN THROUGH EACH LINE OF THE PICTURE
C
DO 500 L=1, N1
CALL GET(CIBUF2, L, 12)
CALL PUT(OBUF, L, 0)
CALL MV(CIBUF2(12+1), OBUF(0+1), NW)
C
C. GENERATE NUMBER FOR EACH OBJECT AT APPROPRIATE TIME
C
IF(HK EQ. 0 OR N1 GT HK) GOTO 210
DO 200 H=1,N1
LD=L-LT(H)
IF(LD GT 4) GOTO 200
IF(LD GE 4) GOTO 48
IF(ORDER) GOTO 210
GOTO 200
200 CONTINUE
SD=ST(N)-13
NCAR=2
IF(NC AR GE 10) GO TO 50
NCAR=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, CHAR(2), 2)
CALL TEXT(CHAR(3-NCAR), NCAR, LD+3, OBUF(0+SD), 1)
CALL ITC(0, OBUF(0+SD), 0, OBUF(0+SD+6+NCAR))
GO TO 260.
70 IF(ORDER) N1=N1+1
75 CALL ITLC(0, OBUF(0+SD), 6+NCAR+1)
260 CONTINUE
210 IF(FI.EQ.0) GOTO 500
C NO DISPLAY IF FINISHING UP A COUNT.
CALL DLINE(OBUF(0+1), Y, X, N1, REPL, 0)
Y=Y+REPL+1
500 CONTINUE
C
C. CLOSE DATA SETS.
INTERNAL I TO CHECK NOB OUTPUT AND CORRECT FOR ERRORS
SUBROUTINE INT1(DUNIT, FIPLEX)
IMPLICIT INTEGER (A-Z)
COMMON/C1/NK, LTAB, STAB, F1
INTEGER PAR(10), SPAR(5), SST(513), NST(513), TTA9(133), MPAR(5)
INTEGER XTAB(60), YTAB(60), XTAB(50), YTAB(50), TEMP(60), RPAR(100)
BYTE A(1234), BLACK(512), FIPLEX(9), BPAR(20), KEY(20), LIGHT(512)
BYTE MB(32), NM(2), MMSG(129), MK(0), WHITE(9), CHAR(2)
BYTE BELL(3)
EQUIVALENCE (PAR, BPAR), (HE, TTA9(2)), (LPS, TTA9(3)), (SPS, TTA9(4))
EQUIVALENCE (NL, SPAR(1)), (HS, SPAR(2))
DATA BELL, ' ', ' ', ' ' /0 /
DATA NSTIME/1000 /
DATA NOBNPH,-43, NOBNPH,-4, ERAGE, 110, KNOUNT, 118/, NOBNPH,S/
END
C ALWAYS SET NFLAG TO I
C FLAG TO CALL NOB FOR A FINISH COUNT
CALL TLLA(32, LIGHT, 312)
CALL ZLA(white, 4)
CALL MLV1(0, BACK, MB, 19)
CALL MLV1(0, THRESH, NT, 20)
CALL MLV1(0, NC, MK, 7)
MB(19) = 0
MK(0) = 0
RETURN
CALL TYPE(OCHECK SPREAD', 0)
NK = 0
CALL RPARAM(NP, RPAR, 199, NOBNPH)
CALL RPARAM(NP, NK, 122, NOBNPH)
IF(INQ.EQ.0) FI = 64
CALL DCOUNT(NK, FI)
C DISPLAY THE COUNT
CALL TLLA(32, GMSG, 129)
CALL RDA
C READ CURSOR ADJUSTMENTS
CALL AFILE(0, DUNIT, FILPEK, 2, 2)
C OPEN SCAN DATA SET AND READ TTAB
CALL OPENA(5144, 1, 2, 'CN')
CALL GLABELA, SPAR(1A)
CALL MLV(IA, 72), GMSG, 64)
C SAVE LABEL FOR PATIENT REPORT
CALL GETA(SPAR(1), IA)
CALL MVL(IA, 11, TTA9, 133)
C CONTINUE
CALL RPARAM(NP, PAR, 10)
30 IF(INQ.EQ.0) GOTO 1000
35 IF(PAR(I), EQ. R5) GOTO 60
IF(PAR(I), EQ. SH') GOTO 65
DO 50 J = 1, MKEY
IF(INQ(BPAR(I)) .NE. IVKEY(K))) GOTO 50
GOTO (90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1350, 1400, K)
50 CONTINUE
C TYPE (PARAMETER ERROR')
C CALL TYPE('0')
GOTO 25
C RS - RESCAN
C CALL AFNAME(1)
CALL CLOSE(A)
CALL EXIT
CALL APHASE(10)
CALL CLOSE(A)
CALL EXIT

CALL TYPE(' TYPE ONE OF THE FOLLOWING KEYWORDS TO SELECT AN OPTION')

CALL TYPE(' Q - QUICK COUNT')
CALL TYPE(' C - CUT APART A TOUCH (POSITION CURSOR FIRST)')
CALL TYPE(' J - 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(' R - RESTART THIS SPREAD')
CALL TYPE(' RS - RESCAN')
CALL TYPE(' F - FINISH THIS SPREAD (NO, KARIOTYPE)')
CALL TYPE(' P, P,N, - FINISH THIS SPREAD; IT HAS NN CHROMOSOMES')
CALL TYPE(' I - DISPLAY THE INITIAL (UN-NUMBERED) SPREAD')
CALL TYPE(' N - DISPLAY THE NUMBERED SPREAD')
CALL TYPE(' T - ERASE THE LAST CUT OR JOIN REQUEST')
CALL TYPE(' U, D, L, OR R - MOVE THE CURSOR UP, DOWN, LEFT, OR RIGHT')
CALL TYPE(' X - INTERACTIVE COUNT WITH CURSOR AND BELL')
CALL TYPE(' H - 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 - STOP MOB (USE OLD ZCP)')
CALL TYPE(' G-N, T - SET THRESH. OBJECT N TO T')
CALL TYPE(' . IF SPREAD IS OK, TYPE CARRIAGE-RETURN')
CALL TYPE(' ') GOTO 25
J -- JOIN TWO PIECES TOGETHER

CONTINUE

CALL CURSOR(Y1,X1,SL,SS)
CALL TYPE(N-HOME CURSOR TO END OF JOIN ') 0
DO 205 I=1,3

CALL DLINE(255,Y1-2+I,X1-1.3,0,0)

MARK END-POINT

CALL PARAM(NP,PAR,18)
IF(NP EQ 0) GOTO 220
P=P+1
IF(P.EQ.'U' .OR. P.EQ.'D' .OR. P.EQ.'L' .OR. P.EQ.'R') GOTO 215
IF(P.NE.(E-)) GOTO 228

ERASE THE MARK IF E WAS TYPED

DO 212 I=1,3

CALL DLINE(WHITE,Y1-2+I,X1-1.3,0,0)
GOTO 25

CALL UDLR(PAR)
GOTO 219

CONTINUE

CALL CURSOR(Y2,X2,EL,ES)
CALL ADL(SL,SS,EL,ES,SST,NSST,HL,NS)
DO 235 L=SL,EL

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

CALL DLINE(127,Y1+X1+MST(L),1,1)

ERASE ON THE GRAY SCALE TO INDICATE THE JOIN-LINE

CALL PARAM(NP,PAR,18)
T=IVC(PAR(I),IV.Erase).GOTO 255
DO 255 L=SL,EL
S=SST(L)
CALL GET(A,L,1)

CALL DLINE(A+(1+S),2*(L-1),2*(S-1),MST(L),1,0)

REINITIATE THE DATA

GOTO 25

RETURN

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

CALL ITL(V(127,A+(1+S)+SST(L)),MST(L))

WRITE 127 IN THE DATA SET FOR THE JOIN-LINE

GOTO 39

B - CHANGE BACKGROUND

CONTINUE

CALL CURSOR(Y,X,L,S)
CALL GET(A,L,1)
CALL OUTCONIV(A+(1+S),MB(7),3)
IT=(L/LPS)*HC+S+FPS+I
IF(S+EPS.EQ.NC) IT=IT-1
CALL OUTCON(ITAB(68+IT),MB(16),3)
CALL TYPE(HA,MP)
BGST=68

SET TO CHANGE BACK

IF(NP EQ 0) GOTO 25
IF(NP GT 1) GOTO 35
IF(PAR(1).GT.127 .OR. PAR(1).GE.3) GOTO 60
TH=T(B+1)+1+1+1+1
IF(A.SPARK(1)+1,L,1)
CALL PUT(A,SPARK(1)+1,L,1)
CALL NYU(ITAB,A+(1+S),133)
RETURN

GOTO 25

T -- CHANGE THRESHOLD

CONTINUE

T=IVC(MT(7)).IV.(IV.W+) GOTO 440

IF(T .GT. 1) GOTO 440

DISPLAY THE SECTOR BOUNDARIES

DO 419 L=LPS,HL,LPS

CALL DLINE(WHITE,L+LPS,HL,LPS)

CALL 418
CALL ILCA(127, BLACK, 512)
DO 415 S = SPS.HS, SPS
415 BLACK(S) = 0
DO 420 I = 1, 5
420 CALL BLHIC(BLACK, NL+NL+1+1-2, 6, HS, 1, 0)
440 CONTINUE
CALL CURSOR(Y, XL, S)
CALL GET(A, L, IA)
CALL OUTCONF IVG((1A+5)), MT(7), 3)
IT = (L/LPS)*MC+S/SPS*I
IF(S/SPS.EQ.MC) IT = IT-1
CALL OUTCONF(TAB(+IT), MT(10), 3)
CALL TYPE(MT, 0)
BORT = 4
C SET TO CHANGE THRESH
GOTO 320
C
C A - ABORT
C
500 CONTINUE
CALL APHASE(0)
RETURN
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 APHASE(MASKPH)
CALL CLOSE(A)
CALL APFILE(A, 1, 'PIC', 5, 5)
CALL OPENA(512, 0, 2, 'NOB')
C SET TO ADD NK TO NOB OUTPUT LABEL
GOTO 620
610 CONTINUE
FI = 0
CALL UPARAM(122, HK, NOBPH)
C SET NOB TO CALL MASK AND NOT PUT UP THE DISPLAY
CALL APHASE(HOBPH)
620 IF(HP, EQ, 3). HK = PAR(3)
C USE OPERATOR SUPPLIED COUNT
C CONVERT HK FOR GRSC
CACL
C
630 IF<HP, EPR3) HK = PHT(3&
C USE OPERATOR SUPPLIED COUNT
C CONVERT HK FOR GRSC
CACL
C
640 IF(HF .LT. 5) HK = 0
C INSERT LF LF
CALL TYPE(TEMP, 79)
CALL GLABEL(A, SPAR, IA)
CALL NVL(GMSG, 67, A(IA+290), 2)
CALL PLABEL(A, SPAR, A(IA+1))
C INSERT NK IN THE LABEL
650 CALL CLOSE(A)
CALL APFILE(A, 4, 'PBATA', 6, 6)
CALL OPENA(512, 0, 2, 'PRP')
CALL GLABEL(A, SPAR, IA)
NL = HL +1
CALL PLABEL(A, SPAR, A(IA+1))
CALL GET(A, NL, 14)
CALL NVL(GMSG, A(IA+1), 128)
CALL PUT(A, NL, IA)
PAR(1) = 'KG'
PAR(3) = BUNIT
CALL NVL(FILPEX, PAR(4), 18)
CALL UPARAM(0, PAR, MASKPH)
CALL CLOSE(A)
C CALL EXIT
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,IA)
  CALL BLINE(A(L+1),2*(L-1),0,HS,1,0)
  CALL BCOUNT(NK,FI)
GOTO 25
C
C DISPLAY THE MOB OUTPUT
C
  CALL UPARA(122,NK,MOBPH)
  CALL APHASE(MOBPH)
  CALL CLOSE(A)
  CALL EXIT
960 CALL MCU
GOTO 25
910 CALL MC
GOTO 25
920 CALL MC
GOTO 25
330 CALL MCR
GOTO 25
940 CONTINUE
C
C INTERACTIVE COUNT
C
  CALL CURSOR(Y0,X0,L,5)
  DO 947 I=1,NTIME
    CALL CPU(RY,X,L)
    IF(YX.NE.Y0.OR.XX.NE.X0) GOTO 945
  CONTINUE
C CURSOR HAS NOT MOVED IN A WHILE
  IF(YX.LT.24.AND.XX.GT.1000) GOTO 995
C IF UPPER RIGHT GOTO 995
  CALL CURSOR(YX,L)
  IF(YX.GT.1000.AND.XX.LE.1000) GOTO 970
  CALL TYPE(BELL0)
C RING THE BELL
NFLAG=1
  IF(NK.LT.60) NK=NK+1
  CALL BCOUNT(NK,FI)
  XTAB(NK)=X
  YTAB(NK)=Y
  STAB(NK)=S
  DO 950 J=1,2
  DO 950 I=1,5
  CALL BLINE(WHITE,Y-2+I,X-1,J,0,0)
  960 CALL CURSOR(Y2,X2,L)
  IF(Y2.GE.YX.AND.X2.LE.XX) GOTO 960
C WAIT FOR CURSOR TO BE MOVED
GOTO 945
970 CALL OUTCON(HK,NK(5),2)
  CALL TYPE(NK,8)
  IF(NK.LT.2) GOTO 25
C ORDER THE TABLE
  DO 990 H=2,NK
  CALL MVJ(LTAB(N),CE,LTAB(N-I)) GOTO 990
  LTAB(N)=LTAB(N)
  DO 990 J=1,N
  CALL MVJ(LTAB(J),LM,LTAB(J)) GOTO 985
  CONTINUE
  CALL MVJ(LTAB(N),CE,LTAB(N-I)) GOTO 990
  LTAB(N)=LTAB(N)
  DO 990 J=1,N
  CALL MVJ(LTAB(J),LM,LTAB(J)) GOTO 985
C ERASE THE LAST SPOT
  DO 997 J=1,NK
  DO 997 I=1,3

CALL DLINER(127.YTAB(NK)-2+I.XTAB(NK)-1.0.-1)
IF(NK.GT.0) NK=HK-1
CALL DCOUNT(HK,FI)
GOTO 960
1000 CALL CLOSE(A)
IF(PERUN.GT.0) CALL ASPACE(2)
CALL EXIT
C
1100 CONTINUE
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 DLINEM(WHITE,Y-3+I.X-2.50.0)
DO 1120 N=1,HK
IF(L.LT.LTAB(N)) GOTO 1130
1120 CONTINUE
LTHB(NK+1)=L
STAB(NK+1)=S
GOTO 1140
1130 NMOV=HK-N+1
CALL MVW(LTAB(H).TEMP.NMOV)
CALL MVW(TEMP.LTAB(N+I).NMOV)
LTHB(N+1)=L
CALL MVW(STAB(H).TEMP.NMOV)
CALL MVW(TEMP.STAB(N+I).NMOV)
STAB(N)=S
1140 HK=HK+1
CALL DCOUNT(HK,FI)
GOTO 25
1200 CONTINUE
C X PARMETER - 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 N=1,HK
DIFF=ABS(LTAB(N)-L)+ABS(STAB(N)-S)
IF(DIF.GT.MIN) GOTO 1220
MIN=DIFF
MAIN=HN
1220 CONTINUE
IF(MIN.EQ.20) GOTO 60
H=MIN
Y=LTABS(N)+2-2
X=STABS(H)+2-2
IF(X.LT.7) X=7
DO 1240 J=1,2
DO 1240 I=1,9
1240 CALL DLINER(127.Y-5+I.X-7.0.-1)
C ERASE THE OLD NUMBER
NMOV=HK-M
IF(NMOV.EQ.0) GOTO 1250
CALL MVW(LTAB(N+I).LTAB(H).NMOV)
CALL MVW(STAB(N+1).STAB(N).NMOV)
1250 HK=HK+1
CALL DCOUNT(HK,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 UPARM(3, PAR, 2)
CALL APHASE(2)
CALL UPARM(122, NK, MOBPH)
CALL CLOSE(A)
CALL EXIT
C QUICK COUNTER OPTION
C
C OT - SET OBJECT THRESHOLD
C
C 1400 CALL MKV(PAR, RPAR(NRP+1), 4)
NRP=NRP+4
CALL UPARM(NRP, RPAR, RCOPH)
RENUM=1
GOTO 25
END

***********
* MOB *
***********

MOB PROVIDES:
C A) THE ORIENTATION MECHANISMS EACH INPUT OBJECT
C IS FIRST ENCLOSED IN A MINIMUM ENCLOSING RECTANGLE, THEN, THE OBJE
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
C
SUBROUTINE MOB
C
COMMON/C1/NOBJ, IDIR
COMMON/C1/CHDIR, SMLBUF, LRGBUF, MS, NL, CUROLN
COMMON/C1/SST, EST
COMMON/C1/SPIOD, SPLTH, SPAREA
C SMALL BUFFER, FOR HOLDING ROTATED, UNMAGNIFIED OBJECT
BYTE SMLBUF(90, 50)
C LARGE BUFFER, FOR HOLDING UNROTATED OBJECT
BYTE LRGBUF(90, 90)
C BUFFER FOR I/O
BYTE BUF(2124)
BYTE OBUF(2124)
C NUMBER OF INPUT OBJECTS
INTEGER NRB
C SYSTEM PARAMETERS
INTEGER SPAR(5)
EQUIVALENCE (SPAR(1), NL), (SPAR(2), MS)
C PARAMETER BEY FOR DECODING
INTEGER NKEY, KEY(11)
C ENDPOINT TRACKERS (8 FRACTIONAL BITS)
INTEGER XMIN, XMAX, YMIN, YMAX, XDMIN, XDSM, YSMAX, YSMIN
C STARTING & ENDING SAMPLE TABLE
INTEGER=2, SST(00), EST(00)
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 OFLIP(60)
C OPERATOR SPECIFIED ROTATION
INTEGER OPROT(60)
C CENTROMERE LOCATION METHOD
INTEGER CMETH
C SKELETON FLAG
LOGICAL SKFLG
C SKELETON SWITCH
LOGICAL SWSK
C SKELETON ROUTINE PARAMETERS
INTEGER SKIN, SKDEL
C PI & PI/2
REAL PI, PIHALF
C DELTA THETA
REAL DELTA
LOGICAL L4
INTEGER SEDTH
INTEGER BK, BS, SL
C CURRENT OUTPUT LINE #
INTEGER CUOLNH, BL, CHRENT
C PARAMETER AREA
INTEGER PAR(500)
EQUIVALENCE (PAR, LRBUF)
C
C INPUT DIRECTORY
C
INTEGER IDIR(4, 60)
C FIRST BLOCK WRITTEN
INTEGER FBW
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 CIL, CID, CIA
C CENTROMERE LINE NUMBER
INTEGER CENLIN
C PERIMETER
INTEGER PERIM
C AREA
INTEGER AREA
C PERIMETER SQUARED DIVIDED BY AREA
INTEGER PSQA
C CENTROMERE LOCATION METHOD
INTEGER CEMETH
C MINIMUM AND MAXIMUM AREAS FOR CHROMOSOMES
INTEGER MAOA, MINDA
C DEGREES/ROTATION INCREMENT
C SAMPLE ACCUMULATORS
INTEGER DEMA(BB), AREAA(BB), ACC(176)
EQUIVALENCE (DEMA(1), ACC(1)), (AREAA(1), ACC(89))
C
C TOTAL AREA AND TOTAL DENSITY FOR NORMALIZING
C
INTEGER TAREA, TDEN
C
C ACCUMULATORS FOR CALCULATING CIA AND CID
C
INTEGER CDEN, CAREA
C
C SPREAD MEASUREMENTS
C
REAL SPID, SPAREA, SPLTH
C
C SHORT ARM INDICATOR
C
INTEGER SHRTA
C
C RANDOM AREAS
INTEGER CEN, CNPL, S, PER

REAL DPINC

BYTE EM(28)

C REMEASUREMENT CHROMOSOME SELECTOR
    LOGICAL SELFGL, SEL(60)
C
C DATA STATEMENTS
C
DATA DPINC/2.8125/
DATA MAKCHR/60/
DATA NKEY/11., KEY/'AR', 'RD', 'FL', 'CT', 'LV', 'SK', 'SP', 'SE',
    'LD', 'MV', 'MS'/
DATA SKMIN/16., SXDEL/2/
DATA ID/3., RP/0.017433., PT/3.14159., PIHALF/1.5708/
DATA FDW/1., HL/2., MS/3., LBW/4/
DATA DELTA/.85256/
DATA SKFLG/.FALSE./
DATA BLKNO/1., BIRNS/2., DIRM/3., LENGTH/4., IOD/5., CIL/6., CID/7/
DATA CIA/8., CENL/9., PERI/10., AREA/11., PSQDA/12., CEMETH/13/
DATA OPFLIP/60.+.FALSE./, OPCR/60+.OPROT/60+.8/
DATA MAXNL/88., MAXNS/88., CHRENT/15/
DATA SELFLG/.FALSE./, SEL/60+.FALSE./
C
CALL MVL('08U ** REJECTED ***** ** **', EM, 20)
C
C INITIALIZE 'SPREAD MEASUREMENTS'
C
SPIOD=0
SPARE=0
SPLTH=0
CUROLN=0
NC=0
C
C INITIALIZE CHROMOSOME DIRECTORY
C
CALL ITA(0, CHDIR, MAKCHR+CHRENT)
MINA=38
MAXA=2999
CLMETH=80
SWK=. + FALSE.
C
DEFAULT IS LV = NO SKELETON AND WIDTH TO LOCATE CENTROMERE
C
C RETRIEVE PARAMETERS AND DECODE
C
95 CALL RPARAM(HP, PAR, 500, A)

10 IF(IPGTP) GO TO 50

11 IF(IP.GT.IP+2) GO TO 12

12 IF(IPH.LT.1.OR.IP.GT.MAKCHR) GO TO 13

12 CONTINUE
C
46 INVALID PARAMETER
C
13 CALL TYPE('*** PARAMETER ERROR')

GO TO 95
C
C PARAMETER AR - SET MINIMUM & MAXIMUM OBJECT AREA
C
15 MINA=PAR(IP+2)
MAXA=PAR(IP+3)

GO TO 10
C
C PARAMETER ROTA - ROTATE SPECIFIED OBJECT PRESCRIBED AMOUNT
C
18 IF(IPH.LT.1.OR.IP.GT.MAKCHR) GO TO 13

19 GO TO 10
C
C PARAMETER FLIP - FLIP SPECIFIED OBJECT
C
21 IF(IPH.LT.1.OR.IP.GT.MAKCHR) 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 OPFLIP(IPH)+PAR(IP+3)
GO TO 19

C PARAMETER LW - LOCATE CENTROMERE BY WIDTH
C
32 CLMETH=88
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 MOBBING ON RCR
C
47 K=PAR(IP+2)
IF(K.LT.1.OR.K.GT.60) GO TO 13
SELFFLG= 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 CLMETH=0
SWSK = .FALSE.
IP=IP+2
GO TO 10

C PARAMETER MW - LOCATE CENTROMERE USING WIDTH AND MODEL(SKELETON)
C
54 CLMETH = 88
IP=IP+2
SWSK=.TRUE.
GO TO 10

C PARAMETER MD - LOCATE CENTROMERE USING DENSITY AND MODEL(SKELETON)
C
57 CLMETH =0
IP=IP+2
SWSK=.TRUE.
GO TO 10

C OPEN DATA SETS
C
58 CALL AFILE(BUF,3,'UCR'..,'*3','*3')
CALL AFILE(BUF,1,'RCR'..,'13','*3')
CALL OPEN(BUF,1024,1,0,'UCR')
CALL OPEN(BUF,1024,1,1,'RCR')
CALL GLABEL(BUF,SPAR,10)
SPAR(1)=125
C PUT MAX NL IN LABEL SO COPY WILL NOT CAUSE F374
CALL PLABEL(BUF,SPAR,BUF(1B+1))
C READ IN JOB DIRECTORY
C
CALL GET(BUF, 1, IB)
CALL MLV(BUF(IB+1), NOB, MAXMR-@+2)
CALL GET(BUF, 2, IB)
DO 55 I=1, NOB
C HIR(FER, I)=IVZ(BUF(IB+2*1+(I-1)*2))
IF(.NOT. SEFLG) GO TO 60
C
C READ PREVIOUS JOB RESULTS IF SELECTIVELY RE-MEASURING
C
CALL GET(OBUF, 1, I,B)
CALL MLV(OBUF(IB+3), SPIOO, 12)
CURLN=IVZ(OBUF(IB+15))
NC=IVZ(OBUF(IB+17))
CALL MLV(OBUF(IB+31), CHIR(1, I), 2*15*30)
CALL GET(OBUF, 2, IB)
CALL MLV(OBUF(IB+31), CHIR(1, 31), 2*15*30)
C
55 CONTINUE
C
C MAIN PROGRAM LOOP - EXECUTED ONCE FOR EACH OBJECT TO BE MEASURED
C
DO 500 I=1, NOB
IF(.NOT. SEFLG) GO TO 70
IF(.NOT. SEL(I)) GO TO 500
SPIOO=SPIOO-CHIR(I, 0)
SPLIT=SPLIT-CHIR(LENGTH, I)
SPAREA=SPAREA-CHIR(AREA, I)
NC=NC-1
C CONTINUE
C
C INITIALIZE STORAGE FOR THE OBJECT
C
CALL ITIA(B9, SST, MAXHS)
CALL ITIA(0, EST, MAXHS)
CALL ITIA(0, LRGBUF, 90*90/2)
NL=0
NS=0
C
C READ & PROCESS OBJECT BLOCKS
C
KK=0
IL=0
BL=0
IFD=IDIR(FBU, H)
ILD=IDIR(LBW, N)
MINS=IDIR(MS, H)-1
C
C READ EACH BLOCK ONE BY ONE
C
DO 145 BLK=IFB, ILB
CALL GET(BUF, BLK, IB)
ASEC=IVZ(BUF(IB+I))
I=IB+3
C
C PROCESS EACH SEGMENT WITHLABEL THE BLOCK
C
DO 140 I1=1, NSEG
IF(I1.EQ. IVZ(BUF(I))) GO TO 110
I1=IVZ(BUF(I))
ML=BL+1
110 BS=IVZ(BUF(I+1))
IF(BS.GT. 88) GO TO 490
IF(SST<BL) GO TO 490
IF(HSAMP.GT. 88) GO TO 490
KK=KK+NSAMP
EST(BL)=BS+NSAMP-1
IF(EST(BL).GT. MAXHS) GO TO 490
C
C IGNORE POSSIBLE BUG IN FOB OUTPUT DATA **************
C
IF(SST(BL).EQ. 89) SST(BL)=BS
NSAMP=IVZ(BUF(I+1))
153
C TRANSFER SEGMENT INTO LRGBUFF
C
CALL MVL(BUF(I+6), LRGBUFF(GS.BL), HSAMP)
IF((HSAMP-<HSAMP/2)+2, HE, 0), HSAMP=HSAMP+1
140
II=II+HSAMP+6
145 CONTINUE
C
C REJECT OBJECT IF IT IS TOO SMALL OR TOO LARGE
C
IF(KK.GT MAXQ, OR.KK.LT MINQ), GO TO 490
C
C ORIENT THE OBJECT
C
CALL ORIG(BUFFER.SMBUFF, PST, CHIR(S1,H), BL, DIBOT(N), RCODE),
NS=CHIRDIR(DIRNS.N)
NL=CHIRDIR(DIRNL.N)
IF(RCODE.NE,0) GO TO 490
HC=HC+1
C
C BUILD ACCUMULATOR TABLES
C
CALL ACCSUB(SMBUFF, NL, NS, AREA, DEMA, 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(N)
IF(CEN.NE. 0) GOTO 170
C
C CHECK FOR OPERATOR SUPPLIED CENTROMERE
METH=0
C
C SET METH=0 FOR AUTOMATIC CENTROMERE
C
IFC NOT. SUBK) GO TO 150
C
CALL SKSUB(SMBUFF, NL, NS, LHRBUFF, CEN, DEMA, ERMIN, SKELT, CLMETH)
C
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
150
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
155
CONTINUE
NS2=NS-NS1+1
MINA=ACC(NS1+CLMETH)
CEN=NS1
DO 160 I=NS1, NS2
IF(ACC(I+CLMETH),GE.MINA) GOTO 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
165
IFC(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
179
IF(CHDIR(CENLIN, H), LT. 9) CEN=NS-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
IF(ACC(CEN+1+CLMETH), LT. MINA) GOTO 180
IF(ACC(CEN-1+CLMETH).LT.MINA) CEN=CEN-1
GOTO 250

C
C CHECK FOR DUPLICATE MINIMA AND TAKE AVERAGE IF SO
C
230 J=0
CEN1=CEN+1
DO 240 I=CEN1,NS2
IF(ACC(+CLMETH).GE.MINA) GOTO 245
240 J=J+1
245 CEN=CEN+J/2
C
C MEASURE THE ARMS
C
250 CALL DP((MLBUF,HL,CN-LD2))
CALL DP((MLBUF,HL,L3-LD1))
CALL DP((MLBUF,HL,NS-LD3))
ALTH=SQRT((CEN-.S)**2+(FLOAT(LB2-LB1))**2)
2ALTH=SQRT((NS-CEN+.S)**2+(FLOAT(LD3-LD2))**2)
ICR=100.*ALTH/2ALTH
SRTA=I
IF(OPCEN(M).EQ-.9) CEN=CEN-NS-1
STORE NEGATIVE CEN TO FLAG SHORT ARM ON RIGHT
CHDIR(CELMETH)=CEN
CHDIR(LENGTH,H)=CHRTH+.5
CHDIR(AREA, H)=TAREA
CHDIR(CDEN, H)=ICMR
CHDIR(CIA,H)=IAR
CHDIR(CICL,H)=IDR
FPSDA=(FPSDA+FPSDA)/(TAREA+.5*PER)
IF(FPSDA.GT.327.) FPSDA=327.
C
C CHECK FOR INTEGER OVERFLOW
CHDIR(FPSDA,H)=100.*FPSDA
CHDIR(CELMETH,H)=METH
C
C ADD INDIVIDUAL MEASUREMENTS TO SPREAD MEASUREMENTS
C
SPIOD=SPIOD+TARE
SPARE=SPARE+TAREA
SPLTH=SPLTH+CHRTH
C
C STORE CHROMOSOME UP AND WRITE IT INTO ROTATED FILE
C
IF(NS
LPE=1824/ML
CHDIR(BLKDNO,H)=CULON
CALL PUT(OBUF, CUROLH, IB)
CUROLH = CUROLH + 1
IF (SKFLG) GO TO 305
CALL CHROUT(SMLBUF, OBUF(IB+1), NS, HL, LPB, IP, SHRTA, 90)
GO TO 310
CALL CHROUT(LRBUF, OBUF(IB+1), NS, HL, LPB, IP, SHRTA, 90)
CONTINUE
IF (IP .NE. 0) GO TO 300
GO TO 500

C OBJECT BYPASSED

CONTINUE
CHDIR(BLKMD,H) = 0
CALL OUTCON(K, EM(7), 2)
CALL OUTCON(KK, EM(22), 4)
CALL OUTCON(HL, EM(25), 2)
CALL OUTCON(NS, EM(28), 2)
CALL TYPE(EM, 20)

C END OF MAIN LOOP

CONTINUE
C UPDATE CHROMOSOME DIRECTORY ON DISC
C
CALL PUT(OBUF, 1, IB)
CALL ITL2(HOB, OBUF(IB+1))
CALL ITL2(CUROLH, OBUF(IB+15))
CALL ITL2(NC, OBUF(IB+17))
CALL MVL(SP10D, OBUF(IB+3), 12)
CALL MVL(CHDIR(1,1), OBUF(IB+31), 2*30+15)
CALL PUT(OBUF, 2, IB)
CALL MVL(CHDIR(1,31), OBUF(IB+31), 2*30+15)
CALL CLOSE(OBUF)
CALL CLOSE(BUF)

C MEASUREMENTS COMPLETED

CONTINUE
C CALL EXIT END

***************
* CLASSFY = COMPIL WITH /CO.25  ***************
* = = = = = = = = =
C = = = =

C CLASSFY PROVIDES THE CLASSIFICATION MECHANISM FOR THE CLINICLA
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 CLASSFY
COMMON/C1,ITC,CIG,OFG,ICTB,ITLB
COMMON/C1/HOB,SP10D,SPITH,SPAREA,CHDIR,IOBUF
COMMON/C1/HOB,HFLAG,G1D,SID4,ISR,OFG,SLT,6CS

C COMMON REGION
C
C NUMBER OF CHROMOSOMES
C
INTEGER NOB
C
C CHROMOSOME DIRECTORY
C
INTEGER CHDIR(15,60)
C
C FORMAT OF DIRECTORY
C
C BLOCK NO. OF FIRST BLOCK OF AN OBJECT
159
INTEGRAL BLKNO
C NUMBER OF LINES IN OBJECT
INTEGRAL DIRNL
C NUMBER OF SAMPLES IN OBJECT
INTEGRAL DIRMS
C CHROMOSOME LENGTH
INTEGRAL LENGTH
C INTEGRATED OPTICAL DENSITY
INTEGRAL IOD
C CENTROMERIC INDEX BY LENGTH
INTEGRAL CIL
C CENTROMERIC INDEX BY DENSITY
INTEGRAL CID
C CENTROMERE INDEX BY AREA
INTEGRAL CIA
C CENTROMERE LINE NUMBER IN ORIGINAL PICTURE
INTEGRAL CEHLIN
C PERIMETER
INTEGRAL PERIM
C AREA OF CHROMOSOME
INTEGRAL AREA
C PERIM/2 AREA
INTEGRAL FSQDA
C # OF DIRECTORY ENTRIES FOR EACH CHROMOSOME
INTEGRAL CHRENT
C
C ID BUFFER
C
BYTE IOBUF(4288)
C
C RANDOM AREAS
C
INTEGER SPARK(10)
BYTE ICIL(27)
INTEGER SLSIZ.RECSIZ.OFSF.SEX
INTEGER SC55LC
INTEGER GNAM(28)
BYTE MSGC(29).HMSC(36).SMSC(39).PENC(22)
INTEGER KG(10).ICG(10).KFAIL(10).JFAIL(10)
INTEGER CT(20.51).ACT(20.51).MAC(10).MAL(10)
REAL ST(11)
INTEGER CID(99).GFX(273).ICTB(68).ILTB(68)
INTEGER OFS(91).PAR(90).KEYC(19)
LOGICAL CID(10)
INTEGER YINT(273).YID.BIR
LOGICAL DEBUG. DMR
C
C DATA STATEMENTS
C
DATA HGI/19/. HGI/11/. HGI/12/. HGI/28/. HGI/2MI/19/. YID/'Y'/
DATA CT/1290.19*0.66.99.19*0.66.99.16*0.
159.66.68.69.16*0.86.67.68.69.
116*0.59.67.68.69.140.62.68.57.68.69.140.62.68.58.67.68.
299.10*0.85.59.57.67.68.64.66.67.59.99.18*0.85.99.55.67.62.84.
350.63.01.99.19*0.99.54.67.62.64.69.59.12*0.88.69.59.54.67.
462.81.59.57.12*0.80.99.54.67.62.64.69.59.12*0.88.79.80.39.
554.67.62.81*0.80.99.54.67.62.64.69.59.12*0.88.79.80.39.16*0.
639.79.88.99.16*0.59.75.80.99.16*0.59.75.80.99.16*0.59.75.
799.99.14*0.70.79.50.79.99.12*0.50.53.70.79.79.79.14*0.
854.59.60.79.59.79.14*0.59.59.80.79.56.68.14*0.58.59.60.79.
956.68.14*0.59.59.80.79.56.68.12*0.58.67.59.79.56.68.10*0.
159.56.58.67.58.59.79.58.68.12*0.59.59.58.67.58.59.68.
179.58.69.10*0.59.56.57.58.59.69.79.12*0.58.56.55.67.59.
159.68.79.12*0.58.56.57.58.56.68.79.12*0.58.56.55.67.59.
136.68.79.12*0.59.55.56.57.56.56.71.16*0.58.56.55.57.67.56.
117.58.68.10*0.59.58.57.55.67.16*0.59.57.55.67.16*0.58.57.
135.57.61.16*0.59.57.55.67.16*0.58.57.57.39*0
DATA CT/190.0.18*0.70.99.18*0.70.99.18*0.70.99.18*0.70.99.
9 18*0.70.99.18*0.70.99.16*0.58.66.70.99.14*0.53.78.58.66.2*0.
3 18*0.70.99.18*0.70.99.16*0.58.66.70.99.14*0.53.78.58.66.2*0.
5 12*0.58.60.53.70.53.66.12*0.65.39.58.60.53.70.58.59.66.2*0.
C
25 CALL OUTCON(IP, PEN(22), 3)
   CALL QPRINT(PEN, 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 I = 1, NGT2
      1000 CT(CT.II) = A(I; P = P + 3
   1010 IF(IP.GT.NP) GOTO 70
   DO 1020 J = 1, MC
      IF(PAR(IP).EQ.GAM(J)) GOTO 1030
   1020 CONTINUE
   GOTO 10
   1030 JJ = J + J
      CT(JJ, IL) = PAR(IP + 3)
      CT(JJ - 1, IL) = PAR(IP + 2)
      IF(P = P + 4)
      GOTO 1010
C
C PARAMETER NN - NO OBJECT NUMBERS OR CENTROMERE FLAG
C
1100 NFLAG = 1
   IP = IP + 2
   GOTO 10
C
C PARAMETER SO - SET 0 OF OUTPUT SAMPLES
C
1200 NSO = PAR(IP + 2)
1203 IP = IP + 3
   GOTO 10

C
C PARAMETER SL - SET SLOT IB'S
C
1300 DO 1310 I = 1, 99
   1310 SID4(I) = PAR(IP + 1 + I)
   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 MYL(B.D (C + X)
      2 F F
      E-16 E E
      G C SID4(13).112
      GOTO 1410
C
C PARAMETER DE - DEBUG OPTION ON
C
1520 DEBUG = TRUE.
   GOTO 1410
C
C PARAMETER SZ - SET SLOT SIZE
C
1530 SLSZ = PAR(IP + 2)
   GOTO 1293
C
C PARAMETER MV - MOVE CHROMOSOMES AROUND FROM PREVIOUS KARYOTYPE
C
1540 IF(NSMOV.LT.50) NSMOV = NSMOV + 1.
      OLDNS(NSMOV) = PAR(IP + 2)
      HEAL(NSMOV) = PAR(IP + 3)
C PARAMETER ST - SET SLOPE TABLE
2 DO 2000 I=1,NG2M1
2000 ST(I)=PAR(IP+1+1)/PAR(IP+2+1)
   IP=IP+NG2M1+2
   COTO 19
C
C PARAMETER MI - SET MINIMUM # OF CHROMS. FOR EACH GROUP
C
3 DO 3300 I=1,NG
3300 MING(I)=PAR(IP+4+1)
   IP=IP+NG2
   COTO 18
C
C PARAMETER MX - SET MAXIMUM # OF CHROMS. FOR EACH GROUP
C
4 DO 4800 I=1,NG
4800 MAXG(I)=PAR(IP+5+1)
   IP=IP+NG2
   COTO 18
C
C PARAMETER CS - SET CENTER SAMPLES FOR EACH SLOT
C
5 DO 5000 I=1,NG2 
5000 SCS(I)=PAR(IP+4+1)
   IP=IP+92 
   COTO 18
C
C PARAMETER CL - SET CENTER LINE FOR EACH GROUP
C
6 DO 6000 I=1,5
6000 SCL(I)=PAR(IP+4+1)
   IP=IP+7
   COTO 18
C
C PARAMETER IR - SET INITIAL SLOT FOR EACH ROW
C
7 DO 7000 I=1,6
7000 ISR(I)=PAR(IP+5+1)
   IP=IP+92
   COTO 18
C
C PARAMETER IG - SET INITIAL SLOT FOR EACH GROUP
C
8 DO 8000 I=1,NG2
8000 IST(I)=PAR(IP+5+1)
   IP=IP+NG2+2
   COTO 18
C
C PARAMETER CR - MANUALLY INSERT CHROMOSOMES INTO SLOTS
C
9 OFSF=1
   NOFS=NP-IP-1
   IF(NOFS,MX.46.AND.NOFS,ME.46).COTO 0950
C SEE IF 46 OR 46 OFS PARAMETERS WERE USED
   DO 8900 I=1,NOFS
8900 OFS(XTBL(I))=PAR(IP+1+1)
C ALLOW FOR THE EXTRA SLOTS
   COTO 78
8950 CONTINUE
   DO 9800 I=1,90
9800 OFS(I)=PAR(IP+1+1)
   IF(IP+1=1.CH.NP).COTO 78
C
C PARAMETER CD - ADJUST COEFFICIENTS
C
15 DO 71 I=1,6
   CTHB(I)=PAR(IP+1+1)
C CONTINUE
IF(C1AB(0)*C1AB(2)*C1AB(3).NE.000) GO TO 900
IF(C1AB(4)*C1AB(5)*C1AB(6).NE.000) GO TO 900
GO TO 18

C CALL TYPE'*** COEFFICIENT SUM NOT = 100')
C CALL TYPE('0')
GO TO 900
C READ IN CHROMOSOME DIRECTORY
C
C DO 75 IS28
C IC(1 =C IC I)
C
C SET UP SLOT SIZE IF NOT INPUT
C
C CALL NVL(MAC.AC.10)
C MING(0)=8
MAXG(10)=1
IST(10)=78
C SET C-V GROUP FOR POSSIBLE 1
C
C CONTINUE
C CALL GET(IOBUF.1.1B)
C CALL NVL(IOBUF(1B+1),NOB.16)
NC=1212(IOBUF(1B+17))
SPID=SPID+46/NC
SPAR=SPAR+46/NC
CALL NVL(IOBUF(IOB+31),CHDIR(1.1).2*15*20)
CALL GET(IOBUF.2.1B)
CALL NVL(IOBUF(1B+33),CHDIR(1.31).2*15+30)
DO 72 1=1,NOB
CHDIR(LENGTH.1)=CHDIR(LENGTH.1)*10000/SPID
CHDIR(10.1)=CHDIR(10.1)*10000/SPID
CHDIR(AREA.1)=CHDIR(AREA.1)*10000/SPAR
72 CONTINUE
C
C INITIALIZE CURRENT INDEX AND INITIAL INDEX FOR EACH GROUP
C
C ICIG(1)=0
II(I)=1
DO 75 I=2,NGI
C ICIG(1)=ICIG(I-1)*25
75 II(I)=II(I-1)+25
C
C SET UP SLOT SIZE IF NOT INPUT
C
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
SCAL1=SSEP+SLSIZ/3
SCS(2)=SCS(2)+SLSIZ/3
30 752 I=2,17.2
SCS(1)=SCS(I-1)+SLSIZ+SSEP
752 SCS(I+1)=SCS(I)+SLSIZ/3
30 754 J=1.4
DO 754 J=1.4
754 SCS(18*J+1)=SCS(1)
756 CONTINUE
758 CONTINUE
C
C CHECK SEX
C
C IF(SEX.GT.0) GOTO 76
SEX=LSEX
76 IF(SEX.NE.1) GOTO 77
C MALE
MAXG(5)=15
MING(10)=MING(10)+1
C FEMALE

MNGR=16
MAXC(10)=MAXC(10)-1
IF(SID(78).EQ.10) SID(78)=SID(78)+1000

C CHANGE Y TO BLANK UNLESS SID WAS CHANGED BY USER

78 CONTINUE

IF(OFSF.EQ.1) GOTO 403

C

SKIP CLASSIFICATION IF OFS TABLE IS INPUT

C

DO INITIAL ASSIGNMENTS OF CHROMOSOMES TO GROUPS

DO 200 N=1,NOB
IF(CHDIR(BLKNO,N),EQ.0) GOTO 200
IC=CHDIR(CIL,N)+CTAB(4)+CHDIR(CID,N)+CTAB(5)+CHDIR(CIA,N)
1+CTAB(6))/100
IL((FLOAT(CHDIR(LNCHM,N)))+CTAB(4)+FLOAT(CHDIR(103,N))
1+CTAB(2)+FLOAT(CHDIR(AREA,N)))+CTAB(3))/1000.)*1.

IF(CIL.GT.51).IL=51
CALL OUTCOM(N,ICIL(5),-2)
CALL OUTCOM((IC+1),ICIL(5),-2)
CALL OUTCOM((IC+ICIL(7)),-2)

ICTB(N)=10
ILTB(N)=IL
IF(CIL.EQ.0 OR CHDIR(P5QKA,N),LT.1340 OR CHDIR(AREA,N),GT.690)
100 TO 82

C PUT IN UNKNOWN GROUP IF BLOB OR OVERLAP
DO 80 J=1,NHT2H1.2
IF(CIG.EQ.C(J-1,IL) AND IC-CT(J+1,IL)) GOTO 100

C IF IC IS WITHIN THE GROUP LIMITS FOR THIS VALUE OF IL GOTO 100
80 CONTINUE
82 J=4C;

C OBJECT IS IN THE FOEBDZEN ZONE
GOTO 110

85 CONTINUE

CALL TYPE(' ALL SLOTS FULL')
GOTO 200

100 CONTINUE

J=(J+1)/2

110 CONTINUE

CALL OUTCOM(J-1,ICIL(26),-2)
IF(NOQ(J),EQ.25) GOTO 85
NOQ(J)=NOQ(J)+1
CIG(J)=CIG(J)+1
OPG(CIG(J))=H

900 CONTINUE

C

IF ANY GROUP IS HEAVY OR LIGHT, TRY TO MOVE CHROMOSOMES AROUND

210 HFALL=0
220 CONTINUE
KH=0
HM=0
DO 3000 J=2,NC
IF(NOQ(J),GE.MXG(J)) GOTO 3000
K=(J-1)
HCN=0
DMIN=100000

C GROUP J IS LIGHT, SEE IF A PRECEDING GROUP IS HEAVY

DO 2500 K=1,K2
IF(NOQ(K),LE.MING(K)) GOTO 2500

C GROUP K IS HEAVY

IF(HFALL,EQ.0) GOTO 230
DO 225 N=1,HFALL
IF(K.EQ.KFAIL(N) AND J.EQ.JFAIL(N)) GOTO 235

C THIS J AND K HAS ALREADY FAILED

225 CONTINUE
230 KH=K
JL=J

C SAVE HEAVY AND LIGHT GROUP NUMBERS
235 CONTINUE

I1=IG(K)
I2=CIG(K)
DO 2400 I=11,12
M=OFG(I)
IC=ICTB(N)
IL=ILTBN(J)
JJ=J+J
IF(IC.LT.CT(JJ-1,IL).OR.IC.GT.CT(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 = (((IC-AC(JJ))**2+(IL-AL(JJ))**2)**2)*1
MAXB = (((IC-AC(K))**2+(IL-AL(K))**2)**2)**1
IF(D.GE.MAXB) GOTO 2460
C THIS IS THE MOST LIKELY CANDIDATE SO FAR
KSAV=K
ISAV=1
NCAN=1
DMIN=1
DO 2480 CONTINUE
2480 CONTINUE
IF(NCAN.EQ.0) GOTO 3000
HM0V=HM0V+1
C MOVE OBJECT N FROM GROUP KSAY TO GROUP J
CALL OUTCON(KSAV,MNSG(22),3)
CALL OUTCON(J,MNSC(29),3)
IF(DEBUG) CALL OPRINT(MNSG,29)
NOG(J)=NOG(J)+1
CIG(J)=CIG(J)+1
NGG(J)=NGG(J)+1
CIG(KSAV)=CIG(KSAV)+1
NOG(KSAV)=NOG(KSAV)+1
IF(CIG(KSAV).EQ.CIG(J)) I2=I2+1
DO 2600 I=ISAV,12
2600 CONTINUE
DO 2500=IFS1+1,11
C CLOSE UP THE REMAINING OBJECTS IN THE GROUP
3000 CONTINUE
IF(HM0V.GT.1) GOTO 220
C IF SOMETHING WAS MOVED, SEE IF MORE MOVES ARE POSSIBLE
IF(HM0V.EQ.0) GOTO 3100
KHN 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
3010 KG(L)=KH
3020 CALL CFO(KG(L),JG(L),IX,AC,AL,CT)
C CHECK FOR A CHROMOSOME IN THE KG-JG OVERLAP AREA
IF(IX.EQ.0) GOTO 3040
3040 IX(I)=IX
C SAVE INDEX NUMBER
IF(KG(L).EQ.KH) GOTO 3060
C WE NEED TO GO UP ONE LEVEL
L=L+1
JG(KG(L))=JG(L-1)
GOTO 3010
3060 KG(L)=KG(L+1)
C LOOK AT THE NEXT GROUP ON THIS LEVEL
IF(KG(L).LT.JG(L)) GOTO 3020
L=L-1
C GO DOWN ONE LEVEL
IF(L.GT.0) GOTO 3040
C.FAILED TO FIND A GOOD MOVE
NFALV=NFALV+1
IF(NFALV.GT.19) GOTO 3100
KFALV=KFALV+1
IF(KFALV.EQ.KH) GOTO 3200
IF(NFALV.EQ.JL) GOTO 3200
3200 CONTINUE
C MAKE THE L MOVES THAT HAVE BEEN FOUND
DO 3000 M=1,L
ISAV=1CM
JGM=JG(M)
NOG(JGM)=NOG(JGM)+1
CIG(JGM)=CIG(JGM)+1
GOTO 3000
4,122,518

173

OFC(CIC(JCM))=OFC(ISAY)
CIC(KCM)=CIC(KCM)-1
HOC(KCM)=HOC(KCM)-1
I2=CIC(KCM)
DO 3070 I=ISAY+12
3070 OFC(I)=OFC(I+1).

3090 CONTINUE
CALL OUTCON(IH,MSG(22),3)
CALL OUTCON(JL,MSG(23),3)
IF(DEBUG) CALL GPRINT(MSG.39)
GOTO 210
C LOOK FOR MORE MOVES
C
C DONE MOVING CHROMOSOMES BETWEEN GROUPS... NOW ORDER THE CHROMOSOMES
C WITHIN EACH GROUP
C
3100 CONTINUE
DO 320 J=1,HC1
IIF(HOC(J)) EQ.0) GOTO 300
I=IIC(J)
I2=CIC(J)
DO 240 I=11,12
H=OFC(I)
I=IIC84(J)
I=ILTB(i)
YINT(I)=-IC-ST(J)+IL
IF(ST(J).LT.0) YINT(I)=YINT(I)
240 CONTINUE
C STORE Y INTERCEPT
I=IYINT(I)

C 312 CONTINUE
L2=IST(J)+HOC(J)-1
IIF(L2.LT.IST(J+1)) GOTO 246
HOC(J)=HOC(J)-1
HOC(HGI),LT IST(HG2)-IST(HC1)) GOTO 244
CALL SWITCH(I.ISW1)
IIF(ISW1)=EQ.1) CALL TYPE('ALL SLOTS FULL')
GOTO 242
244 CONTINUE
CIC(HG1)=CIC(HG1)+1
HOC(HG1)=HOC(HG1)+1
OFC(CIC(HG1))=OFC(I2)
I2=I2-I
GOTO 242
246 CONTINUE
DO 260 L=71,L2
MINS=9999
DO 250 I=11,12
IIF(YINT(I).GE.MINY) GOTO 259
MINY=YINT(I)
MINY=I
259 CONTINUE.
OFS(L)=OFC(MINY)
YINT(MINY)=9999

260 CONTINUE
C STORE THE GROUP IN ORDER OF YINT.
300 CONTINUE
310 CONTINUE
C
C NOW MAKE ANY OTHER ADJUSTMENTS REQUIRED
C
C IIF(HOX), GOTO 350
C CHECK FOR NO X SEPARATION FROM C GROUP DESIRED
C FIND THE X OR X-X IN GROUP C
II=IST(S)
I2=II+3
I3=II+2
I4=II+2
IIF(HOC5,EQ.16) GOTO 330
IIF(HOC(S).GE.15) GOTO 350
C CHAS 15. THIRD LARGEST IS X
OFS(I2)=OFS(I3)
DO 320 I=I3,14
320 OFS(I)=OFS(I+1)
OFS(I2-I)=0
GO TO 350
339 CONTINUE
C C H A S 16, 2 AND 4 ARE X-X
ITEM=0$S(I)
TAMP=0$S(I+1)
DO 244 I=13,14
340 OFS(I)=OFS(I+2)
OFS(I+1)=ITEM
OFS(I+2)=TAMP
350 CONTINUE
IF(NOSG(NC).NE.5.OR.SEX.EQ.2) GO TO 400
C IF 5 IN GROUP C, FIND THE Y CHROMOSOME
I=IST(NC)
I=I+4
IF(OFS(I+2).EQ.0) GO TO 403
MAX=1
DO 360 I=11,12
N=OFS(I)
N=OFS(I+2)/CHDIR(AREA,N)/CHDIR(DIRNL,N)/CHDIR(DIRNS,N))
IF(MAX.LE.MAX) GO TO 360
MAX=N
MAX=N
360 CONTINUE
I=I-2
DO 370 I=MAX+1
370 OFS(I)=OFS(I+1)
OFS(I+1)=0
OFS(I+2)=MAX
380 CONTINUE
IF(NOSG(NC).EQ.0) GO TO 493
DO 402 H=1.HSMOV
NEWSN=NEWSN(H)
OLDSN=OLDSN(H)
IF(NEWSN.NE.0) GO TO 401
C DELETE OBJECT IN OLDSN IF NEWSN = 0
OFS(OLDSN)=0
GO TO 482
491 CONTINUE
NSAVE=OFS(NEWSN)
OFS(NEWSN)=OFS(OLDSN)
OFS(OLDSN)=NSAVE
492 CONTINUE
493 CONTINUE
C ADJUST TABLE FOR OBJECTS WIDER THAN 24 SAMPLES
DO 410 J=1,HG
I=IST(J)
I=I+J+1
HDEL=0
DO 405 I=1,12
564=SG(C(I))=HDEL
C ADD ON ANY PREVIOUS DELTA
INP=OFS(I)
IF(INP.EQ.0) GO TO 494
HSTM24=CHDIR(DIRNS,INP)-SLSIZ
IF(HSTM24.LE.0) GO TO 494
HSTM24=HSTM24+HSTM24
C ADD DELTA FOR THIS OBJECT
HDEL=HDEL+HSTM24
494 CONTINUE
IF(SCS(I).GT.586) SCS(I)=586
405 CONTINUE
499 CONTINUE
IF(NEWSN.EQ.1.AND.NSO.EQ.2) GO TO 425
C IF NSO OF 0 WAS SPECIFIED. DO NOT CHANGE IT
C NOW FIND NSO
DO 420 J=2,6
DO 415 I=1,10
INP=OFS(I+GR(J)-1)
IF(INP.EQ.0) GO TO 415
C THIS IS THE LAST NON-VACANT SLOT ON THE ROW
ITEM=SCS(I+IR(J)-1)+SLSIZ/2+FPER
IF(ITEM.GT.NSO) NSO=ITEM
C ADJUST NSO IF REQUIRED
GO TO 420
415 CONTINUE
420 CONTINUE
IF(NSO.GT.MAXNSO) NSO=MAXNSO.
C
C OUTPUT RESULTS OF CLASSIFICATION.
C
425 CONTINUE
DO 500 I=1,ISRI
ISRI=ISRI+1
DO 500 J=1,ISJ
INP=OFF(J,J)
IF(J,INP.GT.MAX) INP=MAX
IF(INP.EQ.0) GO TO 500
ITEMP=CHDIR(DIRNL,INP)/2+28
IF(SCLT(I,L,ITEMP)) SCLT(I)=ITEMP
500 CONTINUE
C
CALL PUT(IOBUF,3,18)
CALL MYL(NSO,IOBUF(IB+1),2*294)
CALL CLOSE(IOBUF)
C
C JOHN RETURN
C
CALL EXIT
END
C
C*************
C
C* KTYPE *
C*
C*************
C
C KTYPE PRODUCES THE OUTPUT KARYOMAP FROM THE CLASSIFICATION TABLES
C PRODUCED BY CLASSFY AND THE ROTATED CHROMOSOME IMAGES
C PRODUCED BY ORIENT.
C
SUBROUTINE KTYPE
COMMON/C1,IMVB,IBUF,OBUF,MLT,NST,FBM,CEMLIN,CEMETH
COMMON/C1/NSO,HFLAG,GID,SID,ISR,OFFS,SCLT,SCS
INTEGER SPAR(10),CEMETH(40)
INTEGER ISR(16),GID(10),SID(98),R(39)
BYTE OBUF(3128),IBUF(20480)
INTEGER FBM(60),NST(60),MLT(60),CEMLIN(60),OFFS(91),SCS(98)
BYTE LABEL(216)
INTEGER SCLT(5)
CALL MYL('B=****'*R,78)
1 *** *** *** *** *** *** *** ***
C
C OPEN DATA SETS
C
CALL DCLEAR.
C CLEAR GRAY SCALE
CALL AFILE(INVB,1,'CR',13,13)
CALL AFILE(GBUF,5,'PIC',5,5)
CALL OPEN(INVB,1924,8,'RCA')
CALL OPEN(GBUF,1536,11,'KGM')
CALL GLABEL(INVB,SPAR,1B)
CALL MYL(INVB(IB+1),LABEL,216)
MLD=0
C
C READ CHROMOSOME DIRECTORY & BUILD TABLES
C
CALL READ(INVB,1,IB,IBUF(1))
NSO=IV2(IBUF(1))
DO 16 I=1,NSO
FBM(I)=IV2(IBUF(I+30+1))
MLT(I)=IV2(IBUF(3+30+I))
NST(I)=IV2(IBUF(5+30+I))
CEMLIN(I)=ABS(IV2(IBUF(17+30+I)))
16 CONTINUE
C
C*************
C
C* KTYPE *
C*
C*************
C
179
CEMETH(I)=IV(IBUTF(25+39*I))
IF (NOB.EQ.1) GO TO 40
CONTINUE
CALL READ(INVB,1B,IBUTF)
DO 28 I=1,30
FMN(I+30)=IV2(IBUTF(I+28+1))
NyT(I+30)=IV2(IBUTF(3+28*I))
HST(I+30)=IV2(IBUTF(5+30*I))
CMEN(I+30)=LSBS(IV2(IBUTF(17+30*1)))
CEMETH(I+30)=IV(IBUTF(25+38*I))
IF (NOB.EQ.1+30) GO TO 40
28 CONTINUE.
C READ IN CLASSIFICATION TABLES
C 40 CONTINUE
CALL READ(INVB,1B,IBUTF)
CALL MVL(IBUTF(1),MSO,2+294)
IF (MSO.GT.512) GO TO 908
SPEAR(I)=512
SPEAR(I+2)=MSO
SPEAR(3)=512
CALL PLABEL(obuf,SPAR, LABEL)
C MAIN LOOP.
C BUILD EACH ROW OF THE OUTPUT KARYOGRAM
C DO 500 I=1,5
ISRI=ISR(1)
IF (I.FT.5) GO TO 430
C SEE IF THERE ARE ANY OBJECTS ON ROW 5.
DO 420 J=1,18
IF (OFS(ISRI+J-1).NE.0) GO TO 430
420 CONTINUE
GOTO 500
430 CONTINUE
C NO OBJECTS ON ROW 5
C 430 CONTINUE
R(2)=GID(I+(I-1)*2)
R(3)=GID(2+(I-1)*2)
DO 450 J=1,18
INF=OFS(J)
IF (J.FG.ISR(I+1)) INF=0
CALL QUICH(INF,R(5+(J-1)*2),-2)
IF (INF.EQ.0) CALL ITL(32,R(5+(J-1)*2))
450 CONTINUE
CALL SWITCH(1,IBIT)
IF (IBIT.NE.1) GO TO 475
CALL TYPE(R,78)
C INVOKE THE ROW BUILDING ROUTINE
C 475 CONTINUE
CALL KROW(ISR(I)+ISRI,OFSC(ISRI),SCLT(I),SCS(ISRI),MSO,
ISIDC(ISRI),NFLAG)
NL0=NLO+2*SCLT(I)
500 CONTINUE
C CLOSE UP DATA SETS AND RETURN
C SPAR(I)=NLO
CALL PLABEL(obuf,SPAR, LABEL)
CALL CLOSE(INVB)
CALL CLOSE(obuf)
CALL EXIT
C C ERROR
C 999 CALL TYPE(' *** MSO 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

C COMMON/C, INV, IBUF, OBUF, MLT, NST, FBN, CENLIN, CEMETH
INTEGER CENLIN(60), CEMETH(60)
INTEGER X, Y, REPL
BYTE CHAR(4)
INTEGER SCL(90), IDT(60)
INTEGER OFS(90), SCS(90), MLT(60), NST(60), FBN(60)
INTEGER NSL, NSO, SCL2
BYTE OBUF(3128), IBUF(2048), LBO, LFF
INTEGER RECSIZ, MAXNSL

DATA L00/0/, LFF/127/, RECSIZ/1024/, MAXNSL/20/
DATA X/0/, Y/0/, REPL/1/

C C RETURN IF NSO=0
C IF(NSO.EQ.0) GO TO 910
IF(NSO.GT.MAXNSL) GO TO 900
C C SET UP BUFFER INDICES
C IDT(1)=1
DO 50 I=2, NSL
   50 IDT(I)=IDT(I-1)+RECSIZ
   NSO4(NSO+1)/2
   SCL2=SCL+2
C C MAIN ROW LOOP - PROCESSES EACH LINE WITHIN THE ROW
C DO 200 I=1, SCL2
   CALL PUT(OBUF, 0, 10)
   CALL ZIA(0BUF(1+I), NSO4)
C C SLOT LOOP - PROCESSES EACH SLOT WITHIN THE CURRENT LINE
C DO 150 I=1, NSL
   N=OF5(I)
   NS=OF5(I+1)
   IF(N.EQ.0) GO TO 80
C C CHECK FOR FIRST SLOT OF A PAIR EMPTY, BUT SECOND SLOT FULL
   IF(N.EQ.0) GO TO 150
C GOTO 150 IF BOTH SLOTS ARE EMPTY
   NL2=8
   GO TO 130
C C DECIDE IF OBJECT APPEARS ON THIS LINE
C 90 CONTINUE
   MLT=MLT(N)
   NL2=MLT/2
   IF(L.LT.SCL-NL2 OR L.GE.SCL+NL2) GO TO 120
   L0=L-SCL+NL2
C C CALCULATE INITIAL SAMPLE FOR OBJECT ON OUTPUT LINE
C NSIN=NST(N)
   MS2=NSIN/2
   IF(SCS(I)+MS2.GT.511) SCS(I)=511-MS2
   IS=SCS(I)-MS2
C C GET FIRST BLOCK # OF THE OBJECT
C IB=FBN(N)
   IF(IB.EQ.0) GO TO 120
C BYPASS OBJECT, IF NOT IN DIRECTORY.
LPBN=RECSIZ/MNSTN
MLD=MOD(LO,LPBN).
IF(MLD.GT.0) GO TO 98
CALL READ(MYB,IA+LO/LPBN,IA,IBUF(IDT(I)))
98 IDT(I)=MLD+MNSTN
C TRANSFER OBJECT SEGMENT INTO THE OUTPUT BUFFER
C CALL MVL(IBUF(JI),OBUF(IO+IS),MNSTN)
IS=IS+MNSTN
C C. SEE IF THIS IS THE CENTROMERE LINE.
C IF(NFLAG.NE.0).GO TO 158
IF(LO.NE.CENLIN(N)-1) GO TO 158
C CENTROMERE LINE, FIND WHERE TO PUT THE CENTROMERE ARROWS
C IT=IS+MNSTN
ITMAX=IT-1
DO 110 IU=IT,ITMAX
IF(OBUF(IO+IU).GT.08) GO TO 112
110 CONTINUE
GO TO 117.
112 CONTINUE
IT=IS
115 IT=IT-1
IF(OBUF(IO+IT).EQ.08) GO TO 119
117 CONTINUE
DO 118 K=3,5
OBUF(IO+IU-K)=LFF
118 OBUF(IO+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-ML2-2) GO TO 130
C LINE CONTAINS OBJECT #, SO PUT NUMER INTO OUTPUT BUFFER
C LO=L-SCL+ML2+9
IS=SCS(I)-11
IF(IS.GT.58) IS=58
IF(LO.LT.10) IS=IS-3
NN=NN+1
IF(CEMETH(N).EQ.1) NN=M
C FLAG OPERATOR CORRECTED CENTROMERE
C CALL OUTCOM(NN,CHAR(3),3)
C CALL TEXT(CHAR(3),LO,OBUF(IO+IS),1).
GO TO 150
C C CHECK TO SEE IF LINE CONTAINS GROUP ID
C 130 CONTINUE
IF(I.EQ.(I/2)+2) GO TO 150
IF(NL.EQ.0) GO TO 140
NL3=NL3(NL1)/2
140 IF(NL3.GT.NL2) NL2=NL3
CONTINUE
IF(L.LT.SCL+ML2+5.OR.L.GT.SCL+ML2+10) GO TO 150
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
IF(IS.GT.464) IS=464
CALL TEXT(IDH(N),4,LO,OBUF(IO+IS)-2)
150 CONTINUE
IF(LT.LT.1824) CALL DLINE(OBUF(IO+1),Y,Y,NSO,REPL,9)
Y=Y+REPL+1
200 CONTINUE
RETURN
C BUFFER TOO SMALL TO HANDLE THE ROW

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

SUBROUTINE INT2

C
C
C NSOR
C
C COMMON REGION:
COMMON/C1/NSO, NFLAG, C1D, S1D, TSN, OFS, SCLT, SCS
PUBLIC/CHDIR, HOB, SPTOS, SPLTH, SPAREA, CUROLON, NC
C
C I/O BUFFER:
BYTE I0GUF(2124)
C
C MOB PARAMETERS AND CURRENT PARAMETER HIGH WATER MARK
INTEGER NObPAR(S30), N.8N6PAR(4)
C
C CLSFL/KYTYPE PARAMETERS AND CURRENT PAM. HIGH WATER MARK
INTEGER K1TPAR(S30), K
C
C INT2 LOCAL PARAMETER BUFFER AND POINTER
INTEGER PAr(100), IF
C
C RE-MOB CHROMOSOME SELECTOR
LOGICAL SEL60
C
C CLASSIFICATION TABLES

INTEGER CURVED
C
C CHROMOSOME DIRECTORY AND ITS FORMAT
C
INTEGER CHDIR(15,60)
C
BLOCK # INTEGER BKHD
C NUMBER OF LINES IN OBJECT
INTEGER DIRNL
C NUMBER OF SAMPLES IN OBJECT
INTEGER DIRNS
C
C LENGTH
INTEGER LENGTH
C
C INTEGRATED OPTICAL DENSITY
INTEGER 100
C
C CENTROMERIC INDICES
INTEGER CIL, CID, CIA
C
C CENTROMERE LINE #
INTEGER CELIND
C
C PERIPHER
INTEGER PERIM
C
C AREA
INTEGER AREA
C
C PERIM SQUARED DIVIDED BY AREA
INTEGER PSQDA
C
C CENTROMERE LOCATION METHOD
INTEGER CMETH
C
C PHASE NUMBERS FOR MOB AND FOR CLASFY/KYTYPE
INTEGER MOBPH, KTYPHA, FOBPH, BANDPH, FOURPH
C
C RE-RUN FLAGS
LOGICAL MOBFLG, KTYFLG, ALLFLG, USFLG
C
C RANDOM AREAS
INTEGER SLCL, SCLS, SLID, ST, S1, SL2
INTEGER SPAR(18), SST(513), NST(513), S1, S2
INTEGER KEY(37)
BYTE LABEL(73), BLACK(72), PARBUF(90), Mobb(560)

C CLASSIFICATION TABLES
C
INTEGER GID(18), SID(90), ISR(6), OFS(91), SCLT(5), SCS(90)

C DATA STATEMENTS
C
DATA BLKNO/1/, DIRNL/2/, DIRMS/3/, LENGTH/4/, IDB/5/, CIL/6/, CID/7/
C CLR/8/, CENLIN/9/, PERM/10/, AREA/11/, POSDA/12/, CENETH/13/
C MAXPAR/500/, FOSPHA/6/, Mobph/6/, KYPHA/9/, BANPH/14/
C DEGPR/57, 2958/, FOURPH/15/
C
DATA LABEL/73/, ' ', BLACK/72, 127/
C
DATA MKEY/37/
C DATA MOBFILG/, FALSE/, KTYFLG/, FALSE/, ALEFLG/, FALSE/, OFSFLG/, FALSE/, CALL MVL('ARLWU B BCALSXMMKCFENXSEP SPNXIRGNCDSRF KX C F R M S LDBNML H WAKGBBG ', 'KEY', 74)
C
C INITIALIZE CHROMOSOME DATA SET
C CALL TYPE(' CHECK KARYOTYPE ', 0)
C CALL RCA
C READ CURSOR ADJUSTMENTS
C CALL AFille(IOBUF,), 'RCR ', '13', '13'
C CALL OPEN(IOBUF, 1024, 1, 'RCR')
C CALL GLABEL(IOBUF, SPAR, 1B)
C CALL MVL(IOBUF(IB+1+72), LABEL(2), 72)
C
C READ IN DIRECTORY AND CLASSIFICATION TABLES
C
C CALL GET(IOBUF, 1, IB)
C CALL MVL(IOBUF(IB+1), Mob, 10)
C CALL MVL(IOBUF(IB+1), CHER, 1, 1, 1, 215+30)
C CALL GET(IOBUF, 1, IB)
C CALL MVL(IOBUF(IB+1), CHER, 1, 1, 215+30)
C CALL GET(IOBUF, 3, IB)
C CALL MVL(IOBUF(IB+1), N50, 2, 294)
C
C READ PREVIOUS PARAMETERS
C
C CALL Rpam(M, MobPar, MaxPar, MobPha)
C CALL Rpam(K, KglyPar, MaxPar, KglyPha)
C IF(K.LT.92) GOTO 65
K=K-92
C IF(OSFILG=.TRUE.,
C DO NOT RECLASSIFY IF OFS PARAMETERS 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=1-I
C REQUEST OPERATOR INTERACTION
100 CONTINUE
110 CALL PARAM(NP, PAR, 100, PARBUF)
IF(NP.EQ.0) GO TO 500
IP=1
C
C SECODE PARAMETERS
C
DO 200 J=1, NKEY
IF(PAR(I).EQ. KEY(J)) GOTO (1800, 1650, 900, 910, 950, 1250, 1300, 1350
1, 1450, 1500, 1550, 1600, 1650, 1700, 1800, 1850, 1900, 1950, 2000, 2050, 2100, 2150, 2200
2 2250, 2300, 2350, 2400, 2450, 2500)
J
200 CONTINUE
C CALL TYPE(' PARAMETER ERROR')
C CALL TYPE(' ') GOTO 100
CALL TYPE(' CURSOR ERROR')
CALL TYPE('0')
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(' M - 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(' QB - QUICK BANB CALCULATION')
CALL TYPE(' VR - DISPLAY WAVEFORMS')
CALL TYPE(' AX - DISPLAY AXES')
CALL TYPE(' KC - CLEAR CLASSIFY PARAMETERS')
CALL TYPE(' HC - CLEAR HOB 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('B')
GOTO 100

PARAMETER U OR D - MOVE CURSOR UP OR DOWN

CALL MOB
GOTO 110

CALL MOB
GOTO 110

PARAMETER BC - CALL BANDED CLASSIFIER

CALL APHASE(SANDPH)
GOTO 709

PARAMETER AR - SET MAXIMUM AND MINIMUM AREAS OF VALID CHROMOSOMES

CALL HVL(PAR(IP),MOSPAR(M+1),0)
M=M+4
MOBFLG=.TRUE.
GO TO 110

PARAMETER LV - LOCATE CENTROMERE BY WIDTH RATHER THAN DENSITY

CALL HVL(PAR(IP),MOSPAR(M+1),4)
M=M+2
MOBFLG=.TRUE.
GO TO 110

PARAMETER CI - SET CENTROMERE OF CHROMOSOM SPECIFIED TO THAT INDICATED BY CURSOR POSITION

CALL KURSOR(Y,X,L,S,LID,SLCL,SLCS,M)
IF(N.EQ.0) GOTO 259
SEL(N)=.TRUE.

CALL HVL('CI',MOSPAR(M+1),2)
MOSPAR(N-3)=N
INL=CHDIR(INL,N)
MOSPAR(N+4)=L-(SLCL-INL/2)+1
CALL DLINE(BLACK,Y,X,4,1,-1)
CALL DLINE(BLACK,Y,X,4,1,-1)
IF(MOSPAR(N+4).GE.INL).OR.(MOSPAR(N+4).LE.1) GOTO 259
M=M+4
MOBFLG=.TRUE.
GO TO 110

PARAMETER FL - FLIP SPECIFIED CHROMOSOM AND SET CENTROMERE

CALL KURSOR(Y,X,L,S,LID,SLCL,SLCS,M)
191 IF(M.EQ.0) GOTO 250
SEL(N)=.TRUE.
CALL MVL('FL',MOBPAR(M+1),2)
MOBPAR(M+3)=N

192 INL=CHDIR(DIRNL,N)
MOBPAR(M+4)=N-(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
MOBFLG=.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,SLD,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(MP,PAR,190)
CALL KURSOR(Y,X,L,S,SLD,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) GO TO 1217
IF(FRSTY.EQ.SY) GO TO 1218

1217 THETA=0.
1215 CONTINUE
MOBPAR(M+4)=THETA*DEGPRD
M=M+4
MOBFLG=.TRUE.
GOTO 110

C PARAMETER AL - RE-NOB ALL CHROMOSONES
C 1250 ALLFLG=.TRUE.
GOTO 110
C PARAMETER SK - OUTPUT SKELETON PICTURES INSTEAD OF GREY LEVEL PICTURE
C 1300 MOBPAR(M+1)=PAR(IP)
M=M+2
MOBFLG=.TRUE.
GOTO 110
C PARAMETER NN - NO OBJECT NUMBERS
C 1350 KYPAR(X+1)=PAR(IP)
K=K+2
KYPFLG=.TRUE.
GOTO 110
C PARAMETER MA - MALE KARYOTYP
C 1450 KYPAR(K+1)=PAR(IP)
K=K+2
KYPFLG=.TRUE.
GOTO 110
C PARAMETER KC - CLEAR CLASY/KTYPE PARAMETERS
C 1500 K=0
CALL WPRA(K,KTYPE,KYPAR)
OSFLG=.FALSE.
KYPEL=.TRUE.
GO TO 110
C PARAMETER FE - FEMALE KARYOTYPE
C 1550 KYPAR(K+1)=PAR(IP)
K=K+2
KYPEL=.TRUE.
GO TO 110
C PARAMETER NX - HO X SEPARATION FROM C GROUP
C 1600 KYPAR(K+1)=PAR(IP)
K=K+2
KYPEL=.TRUE.
GO TO 110
C PARAMETER DE DEBUG OPTION ON
C 1650 KYPAR(K+1)=PAR(IP)
K=K+2
KYPEL=.TRUE.
GO TO 110
C PARAMETER P - PUSH A GROUP OF CHROMOSOMES RIGHT OR LEFT
C 1700 CALL KURSOR(Y,X,L1,S1,SL1,SLCL,SLCS,H)
CALL TYPE("*MOVE CURSOR TO END OF PUSH ",0)
CALL PARAM(RP,PAR.100)
CALL KURSOR(Y,X,L2,S2,SL2,SLCL,SLCS,H)
IF(S1.LT.S2) GOTO 1725
C GOTO 1725 FOR PUSH TO THE RIGHT
C PUSH LEFT
NLS=SL1-SL2+1
IF(NLS.LT.2.OR.NLS.GT.10) GOTO 250
Y=2*(L1-1)
CALL BLINE(127,Y,2*S2,S1-S2,1,-1)
X=2*(S2-1)
DO 1710 I=2,12,2
CALL BLINE(BLACK,Y+I-14,Y+I-14,1,2,1,-1)
1710 CALL BLINE(BLACK,Y+I,X+1,2,1,-1)
CALL NW(OFS(SL2),PAR.NSL)
C SAVE OFS IN PAR
CALL NW(PAR(2),OFS(SL2),NLS-1)
OFS(SL1)=PAR(1)
GOTO 1776
C PUSH RIGHT
S1 LT S2
1725 Y=2*(L1-1)
NLS=SL2-SL1+1
IF(NLS.LT.2.OR.NLS.GT.10) GOTO 250
CALL BLINE(127,Y,2*S2-S1,1,-1)
X=2*S2
DO 1730 I=2,12,2
CALL BLINE(BLACK,Y+I-14,Y+I-14,1,2,1,-1)
1730 CALL BLINE(BLACK,Y+I,X-1,2,1,-1)
CALL NW(OFS(SL1),PAR.NSL)
CALL NW(PAR(1),OFS(SL1+1),NLS-1)
OFS(SL1)=PAR(NSL)
GOTO 1776
C PARAMETER MY - MOVE INDICATED CHROMOSOME TO SLOT SHOWN BY CURSOR
C 1750 CALL KURSOR(Y,X,L1,S1,SLIB,SLCL,SLCS,H)
CALL BLINE(BLACK,Y,X-2,3,1,-1)
SXSIB=SLIB
CALL TYPE("*MOVE CURSOR TO OTHER SLOT ",0)
CALL PARAM(RP,PAR.100)
CALL KURSOR(Y,X,L2,S2,SLIB,SLCL,SLCS,H)
CALL AB(L1,S1,L2,S2,5ST,NST,312,512)
DO 1778 L=L1,L2
Y=2*(L-1)
CALL BLINE(127,Y,X,HST(L),1,-1)
SVI=OFS(SLID)
OFS(SLID)=OFS(SVSLID)
OFS(SVSLID)=SVID

CALL .DLINE(I??,YJ_X_.HST(t). 1:
X=2*(SST+C)
SYI=OFS(SLID)
OFS(SLID)=OFS(SYSL)

KTYFLG=. TRUE.
GO TO 110

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

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

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

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

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

PARAMETER SC - SET UP THE CURSOR
CALL SC
GOTO 110

SETUP THE CURSOR

PARAMETER DS - DISPLAY SPREAD

CALL A PHASE(FOBPHN)
GOTO 700

PARAMETER RF - RERUN FOB

CALL A PHASE(FOBPHN)
CALL WPHANO,MOBPAR,MOBPHN)
GOTO 700

RERUN FOB

PARAMETER AB - ABORT

CALL A PHASE(0)
GOTO 700

PARAMETER 'X' - REMOVE OBJECT FROM KARYOTYPE

CALL KURSOR(Y,X,L1,SLID,SLCL,SLCS,H)
DO 2260 I=2,26,2
CALL BLINE(BLACK,Y+I-14,Y+I-16,2,1,-1)
CALL BLINE(BLACK,Y+I-14,Y+I+16,2,1,-1)
CONTINUE

DRAW AN X OVER THE OBJECT TO BE DELETED
OFS(SLID)=0
4,122,518

C
C PARAMETER R - ADD A LABEL TO INDICATE DIAGNOSIS
C
2300 DO 2310 I=4,72

2310 CONTINUE
C
CALL TYPE( ' INCORRECT LABEL FORMAT' )
GOTO 190

2320 LABELN=1-3
C
CALL AFiLE(NOBB,1,'PIC ',5,5)
CALL OPEN(NOBB,512,6,2,'RDB')
CALL GET(NOBB, I,N)
CALL PUT(NOBB, I,N)
CALL MVL(PARBUF(3),NOBB(IN+290),LABELN)
CALL CLOSE(NOBB)
C
ALSO ADD TO THE NOB OUTPUT LABEL FOR RESEL'S BENEFIT
C
CALL GLABEL(TOBUF,SPAR,1B)
CALL MVL(PARBUF(3),TOBUF(IB+290),LABELN)
GOTO 110
C
C PARAMETER WAVE - DISPLAY WAVEFORMS FROM BAND
2350 CALL UPARAM(2,'WAVE',BANDPH)
GOTO 250
C
C PARAMETER AXIS - DISPLAY AXES FROM BAND
2400 CALL UPARAM(2,'AXIS',BANDPH)
GOTO 250
C
C PARAMETER BB - RE-RUN BANDED CLASSIFIER
C
2450 CALL APHASE(FOURPH)
GOTO 700
C
C PARAMETER BB - QUICK BAND CALCULATION (ONLY THE C GROUP)
C
2500 CALL SSWTCH(7,ISU7)
IF(ISU7.NE.1) CALL UPARAM(2,'BB ',BANDPH)
GOTO 250
C
C DOING LOOKING AT PARAMETERS, SEE IF A RE-RUN IS NECESSARY
C
3000 IF(.NOT.ALLFLG.AND..NOT.KTYFLG.AND..NOT.MOBFLG) GO TO 700
IF(.NOT.OSFLG) GOTO 550
C
C WRITE THE OFS PARAMETERS FOR CLASSFY
C
CALL MVL('OS',KTYPAR(K+1),2)
CALL MVU(OFS,KTYPAR(K+3),90)
K=K+92

5500 CONTINUE
IF(.NOT.ALLFLG.AND..NOT.MOBFLG) GO TO 650
IF(ALLFLG) GO TO 610
C
C MUST RE-RUN NOB ON SELECTED CHROMOSOMES
C
CALL MVL('SE',MOBPAR(N+1),2)
M=M+3
MM=M
NM=M
DO 600 N=1,NOB
IF(.NOT.SEL(N)) GO TO 600
N=M+1
MM=NM+1
MOBPAR(N)=N
CONTINUE
IF(M.EQ.0) GO TO 695
MOBPAR(N)=NM
CALL PUT(TOBUF,1,1B)
CALL MVL(NOB,TOBUF(IB+1),1B)
CALL MVL(CHDR(1,1),TOBUF(IB+31),2*15+30)
CALL PUT(ICSUF,2,1B)
CALL MVL(CHDIR(1,31),10BUF(1B+31),2*15+39)
G0 TO 610

605 M=M-3
C SELECT MOB AS NEXT PHASE TO BE RUN
C
610 CALL APHASE(MOBPHA)
CALL UPAR(A,M,MOBPAR,MOBPHA)
CALL WPAR(K,KYPAR,KYPHA)
G0 TO 700
C
C SELECT CLASSFY AS NEXT PHASE TO BE RUN
C
650 CALL APHASE(KYPHA)
CALL UPAR(A,K,KYPAR,KYPHA)
700 CONTINUE
CALL CLOSE(IOBUF)
CALL EXIT
END

SUBROUTINE RESEL
C RESEL IS USED TO ENTER INFORMATION ON MEASURED CHROMO
C SOMEK INTO KROMDATA. THE INPUT IS FROM MOB OUTPUT.
C
C IMPLICIT INTEGER(A-Z)
COMON/C/ MBUF,KBUF,NEXT,HUSED,PID,RECS,ON,ONL,ONS,
IOLEN,O10D,OHREN,OCLC,OC1D,OC1A,OF5,
RESL=0 D(3)
REAL+ 1C04<80,0),PHI(80,0),MU(10,24),SIGMA(10,24)
REAL+ ICMN,ISUM,LSUM,LEN,C1,HCC,SUMY,OCCOIR(51,8),OPHI(51,7)
INTEGER#NDO(18,3),CSPAR(5),PID(425),RECS(85)
BYTE ON(00),ONL(00),ONS(00),OLEN(00),OC1L(00),OC1D(00),OC1A(00)
BYTE PBUF(56a),LABEL(124),CT(3649),ODATA(3072)
BYTE MBUF(2164),KBUF(56),FNSG(40),TMSG(23),OMSG(128)
INTEGER#PSPAR(10)
INTEGER SPAR(10),KFAR(10),POID(88),GRERA(88),OFS(90),
2TFS(90),SCAND(90),TH(90),THN(60),GFS(90),NFS(10)
EQUIVALENCE (OBDATA,OCFOUR),(OBDATA(1633),OPHI)
EQUIVALENCE (C(57),MU),(C(I789),SIGMA)
EQUIVALENCE (SPAR(6),SCODE),(HLP,PSPAR(1))
DATA MB=3+DIB='465X---,46XY---',45XO--'
DATA HFD(6,4,6,6,4,4,0,6,6,4,6,4,0,6,6,4,0,6,6,4,4,0,6
DATA GNSG(129)
DATA TFS(1,0,0,4,0,4,0,4,4,0,4,0,4,4,5,4,0,4
DATA 26,6,6,7,6,6,9,9,10,10,11,12,12,23,23,40,40
DATA 5,13,13,14,7,15,4,0,4,0,16,16,4,0,4,0,17,17,10,17,4,0,4
DATA 4,4,19,29,29,40,4,0,26,26,25,25,21,21,22,22,24,24,40,40
DATA 5,10,40
DATA GFS(12)=1,6,2,16,3,9,4,10*5,6,2,0,19,7,5,0,10*9
DATA CMT(61)
C CURRENT MISCELLANEOUS INDEX = 61
C
C OPEN MOB OUTPUT AND COPY NECESSARY DATA FROM MOB LABEL
C
CALL SSCH(10,ISW10)
IF(ISW10.NE.1) CALL APHASE(19)
C IF ISW10 IS DON SET TO CALL #BNNORM
CALL AFILE(MBUF,1,'RCR',*13,*13)
CALL OPEN(MBUF,1024,1.0,'RCR')
CALL GLNGEL(MBUF,SPAR,M0)
CALL MVL(MBUF(ML+3),SCODE,1)
CODE=SCODE
IF(SCODE LE.8) SCODE=SCODE
IF(SCODE LE.8 AND SCODE.LE.15) GOTO 5
C CHECK FOR VALID CODE
CALL CLOSE(MBUF)
RETURN
5 CONTINUE
CALL MVL(MBUF(ML+73),GNSG(1),64)
CALL MVL(GNSG(ML+258),GNSG(107),22)
CALL MYL(MBUF(NL+73), LABEL, 64)
CALL MYL(MBUF(NL+113), LABEL, 20)
CALL MYL(MBUF(NL+217), LABEL, 20)
CALL MYL(MBUF(NL+289), LABEL, 20)

C SAVE LABEL INFORMATION
C
C OPEN KROMDATA AND GET NL
C
CALL MYL(HBUF), HL
CALL MYL(HBUF), HL+217

CALL AFILE(KBUF, 4, 'KDATA', 6, 6)
IF(Code.LT.0) CALL AFILE(KBUF, 4, 'KDATA', 6, 6)
CALL OPEN(KBUF, 1024, 0, 2, 'KRM')
CALL GLABEL(KBUF, KSPAR(KL))
NL=KSPAR(1)

C READ SOURCE DIRECTORY RECORD
C
10 CALL GET(KBUF, CODE, KD)
   IF(NUSED.LT.95) GOTO 22
C IF SPACE IN THIS RECORD GOTO 83
   IF(NEXT.EQ.0) GOTO 20
C IF LAST DIRECTORY RECORD GO MAKE A NEW ONE
   CODE=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, CODE, KD)
   CALL GET(KBUF, NL, KD)
   CODE=NL
   NUSED=0
22 CONTINUE
   DO 24 I=1, 1421, 5
      IF(PID(I).EQ.0) GOTO 26
5 CONTINUE
   CALL TYPE(' RESEL ERROR')
   CALL TYPE('0')
   PAUSE 12345
26 CALL MYL(LABEL(21), PID(I), 10)
C STORE PATIENT ID
   IF(REC.LE.8) GOTO 28
   NL=NL+1
C NEED TO MAKE A NEW RECORD
   REC=NL
   RECS(REC(REC))=REC
   IF(Code.LT.0) NL=NL+3
C ADD 3 MORE IF DAMNED
   28 CALL PUT(KBUF, CODE, KD)
   CALL GLABEL(KBUF, KSPAR(KL))
   KSPAR(I)=NL
C UPDATE NL
   CALL PLIEL(KBUF, KSPAR(KBUF(KL+1)))
C
C GET MGB OUTPUT DATA AND TRANSFER TO KROMDATA
C
C GET GFS FROM MGB OUTPUT AND CONVERT TO TYPE
C
   CALL GET(MBUF, 3, M6)
   II=M6+215
   DO 30 II=1, 190
      II=II+2
30 CALL MYL(MBUF(I), MFS(I), 1)
   CALL ZIA(COM, 40)
   CALL ZIA(CHM, 60)
   CALL ZIA(TH, 90)
   CALL ZIA(MFG, 10)
   DO 40 II=1, 30
   40 IF(MFSJ.EQ.0) GOTO 40
   C=GFS(J)
203 4,122,518
204

C NFNG(G)=NFNG(G)+1
C THDFS(J)=TFS(J)
C CONTINUE
C
C MOVE NECESSARY DATA FROM MGB LABEL AND LINE 1 TO KBUF
C CALL GET(MBUF,1,HD)
C CALL PUT(KBUF,REC,KB)
C CALL MVW(LABEL,KBUF(KD+1),62)
C CALL MVW(MBUF(HB+1),NOB,1)
C
N=1
50 IF(TN(N).EQ.0) GOTO 80
50 C GOTO 80 IF THIS OBJECT IS NOT TO BE ENTER
C T=TN(N)-1
C IF(T.LT.60 AND T.GT.0) GOTO 60
C T=CMT
C IF(T.GT.00) GOTO 80
C USE NEXT AVAILABLE SPACE IN MISCELLANEOUS AREA
C CMT=CMT+1
C CONTINUE
C IF(ON(T).EQ.0) GOTO 78
C
C GOTO 78 IF FIRST SLOT FOR THIS TYPE IS EMPTY
C T=T+1
C 78 ON(T)=N
C NNHM=N
C CONTINUE
C N=N+1
C IF(N.LE.NOB) GOTO 50
C
C TRANSFER CMDIR. 30 BYTES IS OFFSET OF OHL ON MBUF
C
II=30+HD
IST=1
IEND=30
DO 100 J=1,2
K=0
K=K+1
NNH=NNH(I)
IF(NNH.EQ.0) GOTO 90
IM=(K-1)*30+11
CALL MVW(MBUF(IM+3),OHL(NNH),1)
CALL MVW(MBUF(IM+5),OHL(NNH),1)
CALL MVW(MBUF(IM+7),OLEM(NNH),1)
CSUM=CSUM+1
LSUM=LSUM+OLEM(NNH)
CALL MVW(MBUF(IM+9),0103(NNH),1)
ISUM=ISUM+0103(NNH)
CALL MVW(MBUF(IM+21),OAREA(NNH),1)
ASUM=ASUM+OAREA(NNH)
CALL MVW(MBUF(IM+11),OCIL(NNH),1)
CALL MVW(MBUF(IM+13),OCLB(NNH),1)
CALL MVW(MBUF(IM+15),OCTA(NNH),1)
90 CONTINUE
IF(J.EQ.2) GOTO 100
CALL GET(KBUF,2,HD)
IST=31
IEND=NOB
II=30+HD
100 CONTINUE
CALL MVW(CSUM,KBUF(KD+129),1)
CALL MVW(LSUM,KBUF(KD+127),1)
CALL MVW(ISUM,KBUF(KD+129),2)
CALL MVW(ASUM,KBUF(KD+133),2)
IF(CODE.GT.0) GOTO 190
C CANCELled STREAM PROCESSING...
C CALL GET(MBUF,4,IN)
C CALL MVW(MBUF(IM+1),CFOUR(1,1),400)
C CALL GET(MBUF,5,IN)
CALL MYW(MBUF(I)+1), CFOUR(1.5, 480)
CALL GET(MBUF: 6, IN)
CALL MYW(MBUF(I)+1), PHI(1.1, 480)
CALL GET(MBUF, 7, IM)
CALL MYW(MBUF(I)+1), PHI(1.5, 480)
DO 110 N=1, NOB
T=M(N)
IF(T.EQ.0 OR. T.GT.51) GOTO 118
C GET THE FOURIER DATA
DO 102 I=1, 8
102 OFOUR(T(I))=CFOUR(N(I))
DO 104 I=1, 7
104 OPHI(T(I))=PHI(N(I)+1)
110 CONTINUE
C WRITE record on KronData
C CALL MYW(ON, KBUF(K+143), 448)
NC=0
DO 200 G=1, 9
200 NC=NC+1
CALL OUTCON(KFNC(N), 4)
200 NC=NC+1
CALL OUTCON(KM, GMSG(68+G*4), 4)
IFIV(GMSG(10?)), NE.32 GOTO 225
C IF OPERATOR SUPPLIED A DIAGNOSIS GOTO 250
DO 220 N=1, ND
DO 210 G=1, 9
IF(KFNC(N), NE, HFD(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 MYL(DID(N), GMSG(10?), B)
CALL MYL(DID(N), KBUF(K+186), B)
250 CALL AFIE(PBUF, 1, 'PIC', +3.5)
CALL OPEN(PBUF, .512, 8, 'NOB')
CALL GET(PBUF, 1, IP)
CALL OUTCON(REC, PBUF(IP+176), 4)
CALL MYL('RECORD', PBUF(IP+156), 6)
C STORE Record number in MOD output for HCOPT
IF(NC.NE.0) CALL MYL(DID(N), PBUF(IP+298), B)
CALL PUT(PBUF, 1, IP)
CALL CLOSE(PBUF)
C STORE diagnosis N in PREP and MOD output for HCOPT
CALL MYL(LABREC, 211), FMSG(31)
CALL OUTCON(REC, FMSG(29), 4)
CALL MYL(GMSG(107), FMSG(32), 8)
CALL TYPE(FMSG)
IFCODE GT. 8 GOTO 300
C WRITE THE BANDED RECORDS
DO 270 I=1, 3
CALL PUT(KBUF, REC+1, IK)
CALL MYL(ORBDATA(I+1024-1023), KBUF(IK+1), 312)
270 CONTINUE
C WRITE PATIENT REPORT RECORD
CALL AFIE(PBUF, 4, 'PDATA', +3.6)
CALL OPEN(PBUF, 512, 8, 'FRP')
CALL GLABEL(PBUF, PSFAR, IP)
NLP=NLIP+1
CALL PLABEL(PBUF, PSFAR, PBUF(IP+1))
CALL GET(PBUF, NLIP, IP)
CALL MYL(GMSG(PBUF(IP+1)), 128)
CALL PUT(PBUF, NLIP, IP)
CALL CLOSE(PBUF)
CALL CLOSE(KBUF)
CALL CLOSE(MBUF)
CALL TYPE('0')
RETURN
END
C  
C    -- COMBINE 2 PICTURES AND ADD A BORDER
C    SUBROUTINE MASK2(BUNIT,FILPEX)
C    IMPLICIT INTEGER(A-2)
C    REAL STATS(256)
C    BYTE *(6290), B(290), C(992), LE(12), RE(12)
C    COMMON STATS.*C.RE
C    INTEGER 2 LOW(I2), HIGH(SI1)
C    EQUIVALENCE (LOW.STATS), (HIGH(I1),LOW(2))
C    INTEGER 2 SPAR(5), SPARB(5), PAR(10), KEY(3)
C    BYTE FILPEX(12)
C    DATA MAXNS/992/
C    DATA NKEY/3/, KEY/'hi'/'ns', 'kc'/'hflag=0', HFR/128/, ISIZE/100/
C    CALL AFFILE(A-1,'PIC' '.5,5)
C    C ASSIGN DK1,PIC(5,5) AS DEFAULT FILE FOR HOB OUTPUT
C    CALL OPENA(A-1972-1,0,'NOB')
C    CALL GLABEL(A,SPAR.IA)
C    CALL AFFILE(0.5,'PIC').5.5)
C    C ASSIGN DK5,PIC(5,5) AS DEFAULT FILE FOR KGM
C    CALL Ryan,&PAR(10)
C    IF(IP.GT.NP) GOTO 30
C    DO 5 J=1,NKEY
C    IF(PAR(IP).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    GOTO 30
C    ISIZE=PAR(IP+2)
C    IP=IP+1
C    DO 10 IP=0, IP+10
C    IF(IP.PRO(IP+2).PAR(IP+3).2,2)
C    C ASSIGN A DIFFERENT KGM FILE
C    CALL OPENB(B,3072,1,0,'KGM')
C    CALL GLABEL(B,SPAR.BB)
C    HL=SPAR(1)
C    IF(SPAB(1).GT.NL) HL=SPARB(1)
C    NS=SPAR(2)+SPARB(2)+1
C    IF(NS.GT.MAXS) NS=MAXS
C    IF(SPAB(3).NE.?) HFR=256
C    CALL MSUBA(AIA+,I),H, DUNIT,FILPEX)
C    CALL TO MSUB
C    DO 100 I=1,NL
C    IF(LT.SPAB(1)) GOTO 90
C    CALL GET(A,1,1)
C    CALL MVL(A1A+,I,C SPAR2)
C    GOTO 55
C    CALL ITLA0(C,SPAR2))
C    C(SPAR2)+1)=127
C    IF(LT.SPAB(1)) GOTO 60
C    CALL GETB(.L1B)
C    CALL MVL(B,B+1,C(SPAR2)+2,: SPARB(2))
C    GOTO 65
C    CALL ITLA0(C,SPAR2)+2,.SPARB(2)
C    CALL MSUBC(C, NS)
C    IF(HFLAG.EQ.1) CALL LSTAT( SPAR2) ,((1)+1), STATS,1,1)
C    IF(STATS.I+1) CALL STAT(SPAR2, A(1+1), STATS.1,1)
C    CONTINUE
C    IF(HFLAG.EQ.1) GOTO 200
C    IF=1
C    C CONVERT THE FREQUENCY COUNTS TO REAL NUMBERS
C    DO 120 I=1,HFR
C    STATS(I)=LOW(I1)+HIGH(I1)*32768.
C    DO 120 I=1,2
C    CALL MSUB(STATS,-HFR.ISIZE)
C    GOTO 200
C    CONTINUE
C    CALL CLOSE(A)
C    END
SUBROUTINE NSUB(LOC+NSO, XSIPE, CSLPEX)

REAL STATS(256), MAXF, HSIZE
BYTE FILEPEX(12), DATA(1000)
COMMON STATS, DATA
INTEGER FREQ(256), SPAR(3)
EQUIVALENCE (FREQ, STATS)
INTEGER NAXNS(5), HAXF, HSIZE
BYTE FILEPEX-(12)*DATM(1896)

COB4HON STATSI DATA

IF(NAXNS) NS=NAXNS
IF(NAXNS GT. HAXNS) NS=NAXNS
DATA MAXNS/976/, MINHS/360/
DATA SPAR/0, 0, 4, 1, 512/
DATA MAXF/-1, HSIZE/-200/, -L/0, WHITE/15, STEP/15, LBW/0/
IF(LBW.GT.0) GOTO 100

C GOTO-100 IF NOT INITIAL CALL
CALL AFILE(DSRN, ISIZEIFILPEX, 2, 2)
CALL SSWRITE(5, ISW5)
BUFSIZE=3872
IF(ISW5.EQ.1) BUFSIZE=2048

C IF TAPE OUTPUT, DECREASE SIZE OF DSRN TO MAKE ROOM
CALL OPEN(DSRN, BUFSIZE, 1, "MSK")
CALL MVW(LOC, LAB, 256)
CALL LIO(LAB(?), BPE)

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 DATA
If(NS.LT. MINNS) NS=MINNS
C MINNS IS THE MINIMUM SIZE FOR THE DATE AND TIME LABEL AND HSTGRN
NSW=NS+24
NUW=(NSW+1)/2
SPAR(2)=NUW
SPAR(4)=512/NUW
IF(SPAR(4).EQ.0) SPAR(4)=1
CALL LABEL(1, SPAR). LAB_FINISH
IF(ISW5.EQ.1) GOTO 98
CALL MVW(T(21), 1)
CALL MVW(52140, T(23), 1)

C SET MYP FOR MTO:
CALL OPEN(T, 1024, 1, 3, "TAP")
CALL MVW(T(51), 1)

C BBP=1
CALL MVW(NSW, T(45), 1)
RECLE=NSW
CALL MVW(NSW, T(47), 1)

C BLOCKSIZE = NSW
50 CONTINUE
CALL ZIA(BUF, NUW)
CALL WLINE(DSRN, LBW, NSW, BUF, 0)

C WRITE A BLANK LINES
CMID=13+NS/2
BUF=NS/2

C GREY WEDGE WIDTH
BM=CN(-16*BUF-1)
D2=32*MUW+3
CALL ITLA(255, BUF(81), B2)
CALL WLINE(DSRN, LBW, NSW, BUF, 6)
CALL WEDGE(WHITE, STEP, BUF, NS)
CALL WLINE(DSRN, LBW, NSW, BUF, 12)
CALL UPWR(-PBM, -STEP, NSW, MK)
CALL WLINE(DSRN, LBW, NSW, BUF, 12)
CALL ITLA(255, BUF(81), B2)
CALL WLINE(DSRN, LBW, NSW, BUF, 0)

C WRITE GREY SCALE
CALL ZIA(BUF, NUW)
CALL WLINE(DSRN, LBW, NSW, BUF, 6)
DO 80 K=0, 6
IF(K.EQ.3) CALL REF(5, BUF, NSW)
DO 70 N=100, NSW, 100
CALL OUTC(N, LN(4), 4)
70 CALL TEXT(LH(2), 1, K, BUF(N+5), 1)
80 CALL WLINE(DSRN, LBW, NSW, BUF, 4)
CALL REF(255, NSW)
CALL ULINE(DSRH,LBU,NSW,BUF,63)
CALL ILLA(255,BUF(5),8)
CALL ILLA(255,BUF(13+NS),8)
CALL ULINE(DSRH,LBU,NSW,BUF,81)
RETURN
100 CONTINUE
   IF(NS0.LE.0) GOTO 200
C IF END OF PICTURE. GOTO 200
   IF(BPE.NE.7) GOTO 110
   CALL MAYA(LOC,DATA(13),NS2)
   GOTO 110
110 CALL MW(LOC,DATA(13),NS2)
115 L=L+1
C STEP INPUT LINE NUMBER
   CALL ILLA(0,DATA(1),12)
   CALL ILLA(0,DATA(13+NS),12)
C ZERO LINE REFERENCE MARKS
   LP3=L+3
   IF(MOD(LP3,100).NE.0) GOTO 130
C TIME FOR NUMBER OF HUNDREDS
   CALL OUTCPU(LP3,LM(4),4)
   CALL TEXT(20,1,ML,DATA(17+NS),1)
   GOTO 140
130 CONTINUE
   IF(MOD(L,5).NE.0) GOTO 150
   CALL ILLA(255,DATA(5),4)
   CALL ILLA(255,DATA(17+NS),4)
140 CONTINUE
   IF(MOD(L,25).NE.0) GOTO 150
   CALL ILLA(255,DATA(9),4)
   CALL ILLA(255,DATA(13+NS),4)
150 CONTINUE
   CALL ULINE(DSRH,LBU,NSW,BUF,1)
C WRITE ONE LINE OF DATA
RETURN
200 CONTINUE
C FINISH ENTRY WRITE BOTTOM OF PICTURE
DO 240 K=1,11
   IF(K.LT.5) CALL REF(25,BUF,NSW)
   IF(K.GE.5 .AND.K.LE.8) CALL REF(5,BUF,NSW)
   IF(K.EQ.9) CALL ZIA(BUF,NSW)
   IF(K.EQ.6) GOTO 228
C FINISH UP NUMBER THAT WAS STARTED ON LEFT AND RIGHT SIDE
   ML=ML+1
   CALL TEXT(LM(2),1,ML,BUF(2),1)
   CALL TEXT(LM(2),1,ML,BUF(17+NS),1)
220 CONTINUE
   IF(K.LT.5) GOTO 240
   IF(K.EQ.6) GOTO 228
   IF(LM(2).GE.100) GOTO 100
   CALL OUTCPU(LM,LOGN(4),4)
230 CALL TEXT(LOGN(2),1,1,K-5,BUF(H+9),1)
C WRITE BOTTOM REFERENCE MARKS
   CALL ZIA(BUF,NSW)
   CALL ULINE(DSRH,LBU,NSW,BUF,1)
   CALL ILLA(255,BUF(1),B2)
   CALL ULINE(DSRH,LBU,NSW,BUF,61)
   CALL WEDGE(255,-STEP,BUF,HS)
   CALL ULINE(DSRH,LBU,NSW,BUF,12)
   CALL WEDGE(WHITE,STEP,BUF,HS)
   CALL ULINE(DSRH,LBU,NSW,BUF,12)
   CALL ILLA(255,BUF(81),B2)
   CALL ULINE(DSRH,LBU,NSW,BUF,61)
C WRITE GREY SCALE
   MAXCHR=NS+11
   JMAX=72
   DO 250 J=1,JMAX-72
      IF(J.LT.JM) 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

I IF(LAB(J+70-I).NE.32) GOTO 243

C ENTIRE LABEL IS BLANK

GOTO 250

C NO BLANK FOUND IN 20 CHARACTERS. MAKE ARBITRARY SPLIT

I=1

C WRITE LABELS

J=J+CHR

LABLEN=LABEL-CHR

IF(LABLEN.GT.0) GOTO 244

C SKIPPED HISTOGRAM IF NSO EQ 0 GOTO 420

NFR=NSO

HSIZE=ISIZE(1)

CALL ZIA(BUF,NWW)

CALL WLINE(DSRH,LDW,NSW,BUF,.04)

DO 340 I=2,NFRM1

IF(STATS(I).LE.MAXF) GOTO 320

MAXF=STATS(I)

320 CONTINUE

IF(STATS(I).GT.MAXF) STATS(I)=MAXF

IF(STATS(NFR).GT.MAXF) STATS(NFR)=MAXF

DO 330 I=1,NFR

FREQ(I)=(STATS(I)/MAXF)*HSIZE+.9999

BARW=H+HFR

HSTART=(HS-BARW+NFR)/2+12

J=HSIZE

340 H=HSTART

DO 350 I=1,NFR

IF(FREQ(I).NE.0) GOTO 350

C TIME TO STORE BAR FOR THIS BN

CALL ITLA(255,BUF(H),BARW)

350 H=H+BARW

CALL WLINE(DSRH,LDW,NSW,BUF,1)

J=J-1

C THE FOLLOWING CODE WAS COMMENTED OUT TO MAKE ROOM FOR TAPE OUTPUT

C 30 370 J=J+64.56

C CALL ZIA(BUF,NWW)

C HSTEP=BARW

C HMAX=HSTART+NFR+BARW+1

C DO 360 H=HSTART,HMAX,HSTEP

C 360 CALL ITLA(255,BUF(H),BARW)

C CALL WLINE(DSRH,LDW,NSW,BUF,5)

C 370 CONTINUE

C CALL ZIA(BUF,NWW)

C CALL WLINE(DSRH,LDW,NSW,BUF,2)

C HSTEP=16*BARW

C DO 400 K=0,6

C H=HSTART

C CALL TEXT(19,1,K,BUF(H-2),1)

C DO 390 N=16,NFR,16

C H=H+HSTEP

C IF(BARW EQ 1 AND MOD(N,32) EQ 16) GOTO 390

C CALL OUTCON(N,LHC(4),4)
DO 19 I=1,60
     IF (OPTI.EQ. 'NC') GOTO 18
     IF (OPTI.EQ. 'MC') GOTO 12
     IF (OPTI.EQ. 'IS') GOTO 11
     IF (OPTI.EQ. 'AIR') GOTO 7
     IF (OPTI.EQ. 'AI') GOTO 20
     IF (OPTI.EQ. 'IB') GOTO 10
     OPTI=TRUE
     DO 10 I=1, 60
     SELECTED
     DO 10 CONTINUE
     CALL FILE(OBUF, 'CR', '13', '13')
     CALL CFILE(OBUF, SPAR, 19)
     CALL MYL(OBUF, SPAR, 19)
     NCHR=0
     CALL GET(OBUF, IB)
     CALL MLV(OBUF(1B+3), SPID, 12)
     NCHR=IV(OBUF(1B+17))
     CALL MLV(OBUF(1B+31), CHDIR(I, 1, 1), 2*15+30)
     CALL GET(OBUF, 2, IB)
     CALL MLV(OBUF(1B+31), CHDIR, 2*15+30)
     CALL GET(OBUF, 3, IB)
     CALL MMV(OBUF(217+1B), OFS, 72)
     IF (NOT.OPTI) Goto 38
     DO 20 I=19, 34
         IF (OPS(1).EQ. 0) GOTO 20
         SELECTED
         DO 20 CONTINUE
     20 CONTINUE
     DO 310 II=1, NCHR
         IF (NOT.SELECTI) GOTO 310
         CURROLN=CHDIR(I, II)
         NL=CHDIR(2, II)
         IF (CURROLN.EQ. 0) NL=0
         NS=CHDIR(3, II)
         IF (NL-LT. 1 .OR. NS.LT. 1) GOTO 310
         NG=0
         LPC=1024/NS
         CALL GET(CURROLN, IB)
         IF (I.EQ. 1) LASTIB=IB
         CURROLN=CURROLN+I
         DO 381 III=1, LPC
             NG=NG+1
             WRITE(5, 2888) (OBUF(1I+1B), I=1, NS)
         CALL MLV(OBUF(1B+1), LRGBUF(1, NG), NS)
         IB=IB+NS
         IF (NG.GE. NL) GO TO 311
     310 CONTINUE
     311 CONTINUE
     DO 380 I=1, NL
         CALL GET(CURROLN, IB)
         IF (I.EQ. 1) LASTIB=IB
         CURROLN=CURROLN+I
         DO 381 III=1, LPC
             NG=NG+1
             WRITE(5, 2888) (OBUF(1I+1B), I=1, NS)
         CALL MLV(OBUF(1B+1), LRGBUF(1, NG), NS)
         IB=IB+NS
         IF (NG.GE. NL) GO TO 311
     380 CONTINUE
     381 CONTINUE

TEST THE ORIENTATION
SCAN ALONG X (ROW) DIRECTION

JX2=NG/2
IY2=NL/2
NH2=0
NSECX=0
NSECY=0
DECIDE IF IT IS PROBABLY BENT OR NOT MAYBE

IF(NSECX.EQ.1.AND.NSEGY.EQ.1) GO TO 430
IF(NSECX.GT.1.AND.NSEGY.GT.1) GO TO 420
IF(NSECX.EQ.1.AND.NSEGY.GT.1) GO TO 430
GO TO 440

CONTINUE

C

STRAIGHT

DO 411 I=1,ML
LASTB=0
DO 405 J=1,NS
IF(LRGBUF(J,I).NE.0) GO TO 406
LASTB=J
405 CONTINUE
406 CONTINUE
FIRSTB=NS+1
DO 407 J=1,NS
J=NS+1-J
IF(LRGBUF(JL,1).NE.0) GO TO 408
FIRSTB=J1
407 CONTINUE
408 CONTINUE
IL=(LASTB+FIRSTB+1)/2
IBAND=0
IL=LASTB+1
IH=FIRSTB-1
IHML=IH+IL+1
IF(IHMIL.LE.LWIN) GOTO 420
C IF WIDTH IS LE WINDOW WIDTH GOTO 420
IK=LASTB+LWIN
DO 410 J=1,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(I)=MBAND/LWIN
GOTO 4895
4200 DO 409 J=IL,IH
409 IBAND=IBAND+LRGBUF(J,1)
BAND(I)=IBAND/IHMIL
4895 CONTINUE
IF(OPT3) LRGBUF(IL,1)=1
411 CONTINUE
C
C GO TO 440
C FLIP IT DIAGONALLY
C  CONTINUE
C  BENT AND PROPERLY ORIENTED
C  AVE Y (I)
C
HLL=((HL/3)*3)+1
NP=0
NSIGY=0
C
DO 431 I=1,HLL,3
DO 431 J=1,NS
IF(LRGBUF(J,I).EQ.0) GO TO 431
NP=NP+1
NSIGY=NSIGY+FLOAT(I)
C
431 CONTINUE...
IZERO=NSIGY/HP
C  AVE X (J)
C
NSL=((NS/3)*3)+1
NP=0
NSIGX=0
DO 422 J=1,NSL,3
DO 422 I=1,HL
IF(LRGBUF(J,I).EQ.0) GO TO 422
NP=NP+1
NSIGX=NSIGX+J
C
422 CONTINUE
IZERO=NSIGX/HP
C
A11=0.0
A21=0.0
A12=0.0
A22=0.0
A13=0.0
A23=0.0
A33=0.0
S=0.0
T=0.0
C
C  HSM2=HS-2
DO 500 J=1,HSM2,3
DO 500 I=1,HL
C
IF(LRGBUF(J,I).EQ.0) GO TO 500
XPM=J-IZERO
YP=I-IZERO
A32=A32+1.0
XTERM=XP
YTERM=YP
A23=A23*YTERM
U=U+XTERM
T=T+XTERM*YTERM
YTERM=YTERM*YP
S=S+XTERM*YTERM
A22=A22*YTERM
YTERM=YTERM*YP
A21=A21*YTERM
YTERM=YTERM*YP
A11=A11+YTERM
C
500 CONTINUE
C
A12=A12
A13=A23+A23+(A12+U+A13+7)
PPP=U*A22+A13*A23*S*A33+T*A12

4,122,518  221

222
PPP = A11 + A22 + A33 + A12 * A23 + A23 * A12
PPP = A22 + A22 * A13 + A23 * A12 + A12 + A13
P = (PPP - PPP) / (PPP - PPP)
Q = A23 * (S - A11 + P) - A13 * (T - A12 + P)
Q = A12 + A23 - A13 * A22
Q = 0 / Q
R = (S - A11 + P - A12 * Q) / A13

NDIV = 3

DO 780 I = 1, NL
Y = I - ZERO
DXY = 2 * P * YP + Q
DXY = 1.0 / DXY
XP = P * (I - ZERO) ** 2 + Q * (I - ZERO) + R + JZERO
MJ = XP + 8.5
MI = YP + JZERO + 0.5

IF (OPT ==) LRGBUF(MJ, MI) = 1
BAND(I) = LRGBUF(MJ, MI)
MI = YP + JZERO + 0.5 + DY
MJ = XP + 1.5

IF (MJ .LE. 0) GO TO 781
IF (MJ .GT. NL) GO TO 781
BAND(I) = BAND(I) + LRGBUF(MJ, MI)
MI = YP + JZERO + 0.5 + DY
MJ = XP + 0.5

IF (MJ .LE. 0) GO TO 781
IF (MJ .GT. NL) GO TO 781
BAND(I) = BAND(I) + LRGBUF(MJ, MI)

GO TO 782

781 CONTINUE
NBIT = 2
BAND(I) = BAND(I) / NBIT
NBIT = 3
782 CONTINUE
BAND(I) = BAND(I) / NBIT
780 CONTINUE
C
CALCULATE LENGTH
C
PT2 = P * P

YSTAR = 0.0 / PT2
YP = (Y) / YEND
PYE = TPE + YPE
PPYE = TPE + PYE
EL = LOG(YPE + YPE + 1.0) + LOG(YPE + YPE + 1.0)
CDIR = 4.0 / EL / PT2 + 0.5
C
USE MOB LENGTH DUE TO UNUSUAL RESULTS!
C
440 CONTINUE
C
FOURIER COEFFICIENTS
C
BNX = BAND(I)
DO 710 I = 2, NL
IF (BAND(I) .GT. BMAX) BMAX = BAND(I)
CONTINUE

C
DO 711 I=1, NL
AHI(S(I))=(BANK(I)+100)/BMAX
711 CONTINUE
C
NIR=8
FNLO=FLOAT(NL-1)
PIL=6.2302/FNLO
C
CALL SSUCHCH(4, ISW4)
IF(ISW4.EQ.1) CALL TIMER
C
NLH=FLOAT(NL+1)/2
DO 5850 J=I, NLH
RM=FLOAT(J-1)+PIL
5850 CONTINUE
C
DO 5850 I=1, NIR
AF=0.0
BF=0.0
IM1=I-1
RM=FLOAT(IM1)+PIL
5850 CONTINUE
C
DO 5810 J=1, NL
JM1=J-1
RM=R+FLOAT(J-1)
LRMJI=JM1+1
LRMJI=MOD(LRMJI, NL-1)+1
C
AF=AF+AHIS(J)*COSHE(LRMJI)
BF=BF+AHIS(J)*SINHE(LRMJI)
5810 CONTINUE
C
AF=(AF+AF)/FNLO
BF=(BF+BF)/FNLO
C
CFOUR(HUMOB.I)=SORT(AF*AF+BF*BF)
PHI(HUMOB.I)=ATAN2(BF, AF)
C
5850 CONTINUE
C
IF(ISW4.EQ.1) CALL TIMER
C
IF (.NOT. OPT2) GO TO 810
DO 811 I=1, NL
811 JUV=AMURC(I)+100.0-NLMO(CHR2-2)
MAX WIDTH OF WAVEFORM IS 20 SAMPLES
JUJ=JUV-JUV/2+1
JUL=JUV-JUV/2
DO 812 J=1, JUJ
LRGBUF(J, I)=0
812 DO 813 J=JUJ, JUL
LRGBUF(J, I)=127
814 DO 814 J=JUL, JUV
LRGBUF(J, I)=0
814 CONTINUE
810 CONTINUE
C
IF (.NOT. OPT3 .AND. .NOT. OPT2) GO TO 310
C
CUROLM=CHR4(1,11)
NG=0
LPC=1624/NS
C
DO 888 I=1, NL
CALL GET(OBUF, CUROLM, IB)
DO 831 III=1, LPC
831 NG=NG+1
CALL MVU(6, IB+1, OBUF(1B+1), NS)
IB = IB + MS
IF(NC.GE.NL) GO TO 842
CONTINUE
CALL PUT(obuf, CURSQL, IB)
CURSQL = CURSQL + 1
IF(NC.GE.NL) GO TO 841
CONTINUE
C
C 310 CONTINUE
C
C OUTPUT FOURIER COEFS TO FILE RCR
C
CALL PUT(obuf, 4, IB)
C
CALL MVU(CF0UR(1, 1), OBUF(IB+1), 400)
CALL PUT(obuf, 5, IB)
C
CALL MVU(CF0UR(1, 1), OBUF(IB+1), 400)
CALL PUT(obuf, 6, IB)
C
CALL MVU(PIK(1, 1), OBUF(IB+1), 400)
CALL PUT(obuf, 7, IB)
C
CALL MVU(PIK(1, 5), OBUF(IB+1), 400)
C
CALL CLOSE(obuf)
C
C CREATE PARAMETERS FOR PHASE 14 (CLAS)
C
DO 4000 I = 1, NCHAR
CLPARK(9+2*I) = CHDIR(41, 1)
CLPARK(18+2*I) = CHDIR(81, 1)
IF(CHDIR(1, 1), EQ, 0) CLPARK(9+2*I) = 0
IF(CHDIR(1, 1), EQ, 0) CLPARK(18+2*I) = 0
C IF BLKNO IS ZERO, THE PHASE IS REJECTED BY MOD < 4000. CONTINUE
C
C SET UP PART OF CHROMOSOME DIRECTORY FOR NEXT PHASE
C
CLPARK(1) = NCHAR
C
C POSITIONS 2 THRU 10 MAY BE REGARDED AS SPARE.
C
CALL UPARIIK(1150, CLPARK, 15)
C
IF(ISO2, EQ, 1) CALL APhASE(i19)
C IF SUB IS UP, CALL KTYPE INSTEAD OF FOUR
IF(OPT2 OR OPT3) CALL APhASE(10)
C
CALL EXIT
C
END
C
SUBROUTINE FOUR
COMMON / CI / CHDIR, SMUF, LRGBUF, MS, NL, CURSQL
COMMON / CI / SP10D, SPLIT, SPARE
COMMON / CI / NCHAR
COMMON / CI / CF0UR, PH1
COMMON / CI / 66UF, OBUF
INTEGER JV(50), I1TAB(100), J1TAB(100)
BYTE MSEC(127)
BYTE OBUF(2124)
BYTE C(3640)
REAL LTMH(68), CTAB(68)
INTEGER START(25), IHNC(25), NHC(25)
INTEGER LOFS(91)
INTEGER DEY(112), KPAR(92)
INTEGER CLPARK(138)
INTEGER CPAR(5)
INTEGER SPAR(5)
INTEGER CURSQL
INTEGER PFL
REAL CF0UR(CF0UR, 19), PHI(68, 10)
INTEGER MCROM(24)
INTEGER CLAS(60)
REAL DIST(18)
REAL VECTOR(18)
REAL ITLKL(68, 24)
REAL N(U(18, 24))
REAL SIGMA(18, 24)
REAL MBUF(18, 24)
REAL SBUF(18, 24)

C
EQUIVALENCE(NBUF(1, 1), C(23, 23))
EQUIVALENCE(SBUF(1, 1), C(1785))
EQUIVALENCE(SBUF(1, 1), SIGMA(1, 18))
EQUIVALENCE(NBUF(1, 1), NU(1, 18))
EQUIVALENCE(KPAN(3), OFS(1))

C
DATA IMEAS/16/, IMEAS/11/, IMEAS/5/, JIMEAS/12/
DATA VMX1/36/, VMX2/36/, VMX3/50/, VMX4/50/, VMX5/64/
DATA JS1/19/, JS2/24/
DATA J/6/, JC2/13/
C SLOTS TO BE CLASSIFIED WITH HYBRID CLASSIFIER
C DATA NEXTPH/20/
C
C NEXT PHASE-10 CALL FOR HYBRID CLASSIFIER
C DATA KIMEAS/2/, K2MEAS/3/, KXMEAS/1/, NGROUP/24/
DATA START/1, 5, 9, 13, 15, 19, 21, 23, 25, 1
DATA 1, 27, 29, 31, 37, 39, 41, 45, 49, 51,
DATA 1, 55, 57, 65, 67, 74, 70, 74, 1
DATA ICR/22/, 1, 1, 1,
DATA MNC/22/, 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
C CALL RPARAM(KPAN, CLPAN, 130)
C CALL SSUCC(4, ISU4)
C IF(ISU4.EQ.1) PAUSE 15
C
C PAUSE IF SW4 IS UP
C
C NCHR=CLPAN(1)
C
C READ BACK FOURIER COEF'S PRODUCED BY BAND (PHASE 14)
C
C CALL AFILB(OFUB, 1, 'CARD', '*13', '*13')
C CALL AFILB(OFUB, 1, 'CARD', '*13', '*13')
C CALL GLABEL(OFUB, 50, 18)
C IF(OFUB(18+184).EQ.*115) MNC(23)=1
C IF(OFUB(18+104).EQ.*166) MNC(24)=0
C
C CHECK FOR SEX OF M OR F AND CHANGE X AND Y MAX
C
C CALL GET(OFUB, 3, 18)
C CALL MYL(OFUB(217+18), LOMS(1), 182)
C
C CALL GET(OFUB, 4, 18)
C CALL MYL(OFUB(1+18), CF0UR(1, 1), 960)
C
C CALL GET(OFUB, 5, 18)
C CALL MYL(OFUB(1+18), CF0UR(1, 1), 960)
C
C CALL GET(OFUB, 6, 18)
C CALL MYL(OFUB(1+18), PHI(1, 1), 960)
C
C CALL GET(OFUB, 7, 18)
C CALL MYL(OFUB(1+18), PHI(1, 1), 960)
C
C C C C C
C
C THE HYBRID CLASSIFIER USES RESULTS OF
C THE CONVENTIONAL CLASSIFIER
C
C CALL SSUCC(7, ISU7)
C IF(ISU7.EQ.1) GOTO 800
C
C GOTO 800 FOR THE HYBRID CLASSIFIER
C SW7 UP IS FOR A FULL FOURIER CLASSIFIER
ISI=1
IS2=90
JCJ=1
JC2=24
NEXTPHN=9
GOTO 800

800 CONTINUE
CALL MV((LOFS, OFS. 90)
CALL ZIA((LOFS, ISI), 32-ISI)

C FOUR - WILL CLASSIFY THE C GROUP STARTING WITH ISI
CALL ZIA((LOFS(13), 4)
CALL ZIA((LOFS(37), 6)
CALL ZIA((LOFS(55), 4)
CALL ZIA((LOFS(65), 4)

C KY7 WILL CLASSIFY X.B.D.E. AND C
C
C 820 IF (CHR.LE. 46) GOTO 830
C IF MORE THAN 46, ALLOW FOR 2 X AND 2 Y

MNC(23)=2
830 CONTINUE
C GET MEANS AND SIGMAS
C
CALL AFIL(C, 4, 'CDATA ', '6, 96)
CALL OPEN(C, 3506, 6, 6, 'CDATA')
CALL NLABEL(C, CPAR, IC)
CALL GET(C, 1, IC)
995 CONTINUE

C ITOL=6
NRELCH=CHR.
DO 899 II=1, NCHR
IF (CLPARM(9+2*II).EQ. 0) NRELCH=NRLECH-1
ITOL=ITOL+CLPARM(9+2*II)
899 CONTINUE
FTOL=46. * ITOL/ NRELCH
DO 4000 I=1, NCHR
LTAB(I)=(CLPARM(9+2*I)*10000 )/FTOL
CTAB(I)=CLPARM(19+2*I)
C STORE REAL LENGTH AND CI
4000 CONTINUE
C
DO 896 II=1, NCHR
DO 896 JI=1, NGROUP
LIKLE(I,JJ)=999.
896 CONTINUE
C
CALL SS uterus(0, ISW)
IF (ISW, EQ. 1) CALL OPRINT('"
1 OBJ GROUP DLEN DCIA. VLEN VCIA
2 VCSUM VPSUM TVAR"
C
DO 900 IS=IS1, IS2
II=LOFS(IS)
IF (II, EQ. 0) GOTO 900
C
IF (ISW, EQ. 1) CALL OPRINT('"
C
DO 903 JJ=JC1, JC2
JJ=JJ
IF (JJ, EQ. 13, AND, ISW, NE. 1) JJ=23
C
SIGII2=SIGMA(K1MEAS, JJ)
SIG222=SIGMA(K2MEAS, JJ)
IF (SIGII2, LT. 0.00001) SIGII2= 0.00001
IF (SIG222, LT. 0.00001) SIG222= 0.00001
C
W1=LTAB(IJ)-MUK(KMEAS, JJ)
W2=CTAB(IJ)-MUK(K2MEAS, JJ)
C
VECTOR(1)=DIST(1)/SIGMA(JJ)+DIST(2)
IF VECTOR(1) GT VMAX(1) GOTO 903
VECTOR(2)=(DIST(2)/SIGMA(JJ)+DIST(2)
IF VECTOR(2) GT VMAX(2) GOTO 903
C
FSUM=0.0
DO 907 JJ=1,NMEAS
DISTI=CFOUR(I(I,J,J-J)-MU(J,J)
FSUM=FSUM+DISTI*DISTI*SIGMA(I,JJ)
CONTINUE
IF FSUM GT VMAX(1) GOTO 903
PSUM=0.0
IF JMEAS_EQ .0) GOTO 9088
C
DO 908 I=1,NMEAS
DISTI=PHI(I,I-I-10)-MU(I,JJ)
IF DISTI GT 3.1416 DISTI=6.2832-DISTI
IF DISTI LT -3.1416 DISTI=-6.2832-DISTI
PSUM=PSUM+DISTI*DISTI/NSIGMA(I,JJ)
CONTINUE
IF(PSUM GT VMAX(4) GOTO 903
C
IF J LIKE (1,JJ)=VECTOR(1)+VECTOR(2)+FSUM+PSUM
CALL SSNTPG(0,ISW8)
IF(ISW8 .NE. 1) GO TO 9010
J(5)=I
J(6)=J
J(4)=DIST(1)
J(2)=DIST(2)
J(3)=VECTOR(1)
J(4)=VECTOR(2)
J(5)=FSUM
J(6)=PSUM
J(7)=LIKLI(I,JJ)
DO 9099 I=1,9
CALL OUTCON(JV(1),JMSG(18*1)),18)
CALL ORPRINT(JMSG,99)
9010 CONTINUE
C
903 CONTINUE
C
DO 952 I=1,NCHR
ELMIN=VMAX(4)
IASAY=0
C
DO 950 J=JCI,YCJ
JJ=J
IF(JJ .EQ. 13 AND ISW7 .NE. 1) JJ=23
IF(1CHROM(JJ).GE.MNC(JJ))) GO TO 950
C
C IGNORE GROUPS THAT ARE FULL
C LOOK FOR UNECLASIFIED CHROMOSOMES
C
DO 951 IS=ISL.IS2
II=LOFS(IS)
IF(I.I .EQ. 0) GOTO 951
IF(CLASS(I.I,N) .GE.1) GO TO 951
IF(LIKLI(I,JJ).GE.ELMIN) GOTO 953
ELMIN=LITL(I,JJ)
IASA=II
JJSAY=JJ
CONTINUE
951 CONTINUE
C
950 CONTINUE
C
IF(IASAY .EQ. 0) GO TO 968
CLASS(IISAY)=JJSAY
NCHROM(JJSAY)=NCHROM(JJSAY)+1
OFS(STTR(JJSAY))=IASAY
STRJ(JJSAY)=STTR(JJSAY)+INCR(JJSAY)
952 CONTINUE
C
968 CONTINUE
C
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=1,180
ELMIN=VMAX5
DO 250 IS=1,181,182
II=LOFS(IS)
IF(II.EQ.0) GOTO 250
IF(CLASS(II).NE.0) GOTO 250
DO 240 JJ=JCI,JC2
IF(IJ.EQ.0) GOTO 240
IF(JJ.EQ.12.AND.ISW7.NE.1) JJ=23
IF(MNC(JJ).EQ.0) GOTO 240
IF(LIKE(IJ,JJ).GE.ELMIN) GOTO 240
ELMIN=LIKE(IJ,JJ)
IISAV=II
JJSAV=JJ
240 CONTINUE
250 CONTINUE
IF(ELMIN.EQ.VMAX5) GOTO 320
IITAB(IK)=IISAV
JJTAB(IK)=JJSAV
LIKE(IISAV,JJSAV)=990.
C RAISE LIKE
320 CONTINUE
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)
JS2=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
IF(IJ.EQ.12.AND.ISW7.NE.1) JJ=23
IF(MNC(JJ).GE.MNC(JJ)) GOTO 440
IF(LIKE(IJ,JJ).GE.ELMIN) GOTO 440
BLIK=LIKE(IJ,JJ)-LIKE(IJ,JJ)
C BLIK IS A MEASURE OF THE LIKELIHOOD THAT I BELONGS TO GROUP JJ
C AND DOESN'T BELONG TO GROUP J
IF(BLIK.GE.ELMIN) GOTO 440
ELMIN=BLIK
ISAV=I
JJSAV=JS
440 CONTINUE
450 CONTINUE
IF(ELMIN.EQ.VMAX5) GOTO 500
C IF NO MOVE OUT OF J CAN BE MADE, GOTO 500
C OTHERWISE, MOVE OBJECT ISAV FROM SLOT JJSAV TO GROUP JJSAV
C THEN MOVE OBJECT J1 TO SLOT JJSAV
C
CLASS(ISAV)=JJSAV
MCHROM(JJSAV)=MCHROM(JJSAV)+1
OFS(JJSAV)=ISAV
START(JJSAV)=START(JJSAV)+INCR(JJSAV)
C
CLASS(J1)=J
OFS(JJSAV)=J1
500 CONTINUE
C
520 CONTINUE
C C CHECK FOR STILL UNCLASSIFIED AND PUT AT BOTTOM
C
DO 990 IS=181,182
I=LOFS(IS)
IF(I.EQ.0) GOTO 990
990 CONTINUE
IF (CLASS(I).EQ.0) GOTO 980
CLASS(I) = 25

965 IF (START(25).GE.90) GOTO 980
START(25) = START(25) + 1
IF (OFST(START(25)).NE.0) GOTO 965

980 CONTINUE

C
CALL MYL('05 ',KPAR,4)
CALL WPARAM(92,KPAR,NEXTPH)
C
CURLOM=7
CALL GET(obuf,CURLOM,1B)
CALL MYL(CLASS(I),obuf(951+1B),60)
CALL PUT(obuf,CURLOM,1B)
C
CALL CLOSE(obuf)
CALL APHASE(NEXTPH)
C
IF(ISSUB.EQ.1) CALL QPRINT('1')
RETURN
END

C************
C
C KFIX
*
C
C************
C
CHROMOSOME CLASSIFIER PART 3
C
SYNTACTICAL CORRECTION OF CLASSIFICATION
C

SUBROUTINE KFIX
CALL /C1/HOB, IDIR
COMMON /C1/ CHDIR, SMLBUF, LRCBUF, MS, NL, CURLOM
COMMON /C1/ SST, EST
COMMON /C1/ SPIQD, SPLTH, SPAREA
COMMON /C1/ NCHR
COMMON /C1/ CFOUR, RHI
COMMON /C1/ BAND, AIHIS
COMMON /C1/ OBUF, BUF
BYTE LRCBUF(78,88)
BYTE BUF(124)
LOGICAL BGROUP(60)
LOGICAL DGROUP(60)
LOGICAL CGROUP(60)
LOGICAL FGROUP(60)
LOGICAL EGROUP(60)
INTEGER ICHCHR(6)
INTEGER DPOINT(6)
INTEGER IIDCHR(6)
INTEGER IIFCHR(6)
INTEGER IIFCHR(6)
INTEGER KPAR(92)
INTEGER OFS(91)
INTEGER OFS(91)
INTEGER LOFS(90)
INTEGER IPROF(90)
INTEGER JPROF(90)
INTEGER KGBAHR(6)
INTEGER JFETUR(20)
INTEGER SPAR(8)
INTEGER BMAX
INTEGER BPL
INTEGER CHDIR(15,60)
INTEGER CURLOM
INTEGER PK(1B)
REAL BB(6)
REAL CD(6)
REAL RATIO(12)
REAL CFOUR(60,15), PHI(60,15)
REAL AFOUR(60,15), BFOUR(60,15)
EQUIVALENCE (KPAR(3), OFS(1))
EQUIVALENCE (SPAR(2), BPL)
EQUIVALENCE (CFour(1, 1), AFFour(1, 1))
EQUIVALENCE (PHI(1, 1), BFOUR(1, 1))
EQUIVALENCE (H1, NLO)
EQUIVALENCE (HUMOR, IT)

DATA HD/0/
DATA NCHR/0/
DATA DGROUP/60*, FALSE. /
DATA GGROUP/60*, FALSE. /
DATA EGROUP/60*, FALSE. /

CALL AFILL(GBUF, 1, ’RCR ’, ’*13’)
CALL OPEN(GBUF, 1, 1024, 1, 0, ’RCR ’)
CALL GLABEL(GBUF, SPAR, IB)

CALL RPARAM(KPAR, KPAR, 32)

READ RESULTS OF CONVENTIONAL CLASSIFICATION AND
EXTRACT GROUP CLASSIFICATIONS

D-GROUP

CALL GET(GBUF, 3, IB)
CALL MVL(GBUF(217+IB), LOFS(1), 182)

DO 184 I=37, 42
IIBCHR(I-36)=0
LOFS=LOFS(I)
IF(LOFS.EQ.0) GO TO 184
DGROUP(IL0FS)=.TRUE.
IIBCHR(I-36)=IL0FS
184 CONTINUE

B-GROUP

DO 182 I=13, 16
IIBCHR(I-12)=0
LOFS=LOFS(I)
IF(LOFS.EQ.0) GO TO 182
DGROUP(IL0FS)=.TRUE.
IIBCHR(I-12)=IL0FS
182 CONTINUE

E-GROUP

DO 1050 I=45, 46
LOFS=LOFS(I)
IF(LOFS.EQ.0) GO TO 1050
EGROUP(IL0FS)=.TRUE.
1050 CONTINUE

DO 1051 I=49, 52
LOFS=LOFS(I)
IF(LOFS.EQ.0) GO TO 1051
EGROUP(IL0FS)=.TRUE.
1051 CONTINUE

F-GROUP

DO 186 I=55, 58
IIFCHR(I-54)=0
LOFS=LOFS(I)
IF(LOFS.EQ.0) GO TO 186
FGROUP(IL0FS)=.TRUE.
IIFCHR(I-54)=IL0FS
186 CONTINUE

G-GROUP, INCLUDING Y
DO 107 I=65,70
IINCHE(1-64)=0
ILOFS=LOFS(I)
IF(ILOFS.EQ.0) GO TO 107
IGROUP(ILOFS)=.TRUE.
IINCHE(1-64)=ILOFS
107 CONTINUE
C GET CHROMOSOME DIRECTORY
C CALL GET(OBUF,1,IB)
CALL NVL(OBUF(IB+3),SPIOS,12)
NCHR=TVZ(OBUF(IB+17))
CALL NVL(OBUF(IB+31),CHDIR(1,1),2*15+30)
CALL GET(OBUF,2,IB)
CALL NVL(OBUF(IB+31),CHDIR(1,31),2*15+30)
C JAY=0
DO 310 IX=1,6
II=IINDCHR(IX)
C IF(DGROUP(II)) GO TO 304
GO TO 310
304 CONTINUE
C JFETUR(11)=0
C CURROLN=CHDIR(1,II)
NL=CHDIR(2,II)
NS=CHDIR(3,II)
IF(NS.LT.1.OR.NS.LT.LT.1) GO TO 310
NG=0
LPG=1024/NS
C LINES=GET.
C DO 308 I=1,NL
CALL GET(OBUF, CURROLN, IB)
IF(I.I.EQ.1).LASTIB=IB
CURROLN=CURROLN+1
DO 381 III=1,LPG
NG=NG+1
CALL SSCHUR(0,JAY)
DO 330 J=1,NS
LRBUF(J,NG)=OBUF(J+IB)
330 CONTINUE
IB=IB+NS
IF(NG.GE.NL) GO TO 311
331 CONTINUE
308 CONTINUE
311 CONTINUE
C ... MAIN LOGIC
C IF(.-DGROUP(II) ) GO TO 401
C ML2=ML/2
LTOP1=2
LTOP2=ML2-1
LBOT1=ML2+1
LBOT2=ML-1
C ISUMT=0
ISUMB=0
MTOP=0
NBOT=0
C DO 411 J=LTOP1,LTOP2
DO 410 I=1,NS
IVAL=LRBUF(I,J)
IF(I.I.EQ.0) GO TO 410
ISUMT=ISUMT+IVAL
MTOP=MTOP+1
410 CONTINUE
411 CONTINUE
CONTINUE
ITOP=ISUMT/NTOP

DO 421 J=LBOT1, LBOT2
DO 420 I=1, NS
IVAL=LRGBUF(I, J)
IF (IVAL.EQ.0) GO TO 420
ISUMB=ISUMB+IVAL
NBOT=NBOT+1

420 CONTINUE
421 CONTINUE
I BOT=ISUMB/NBOT

NB=NB+1
RATIO(NB)=FLOAT(ITOP)/FLOAT(IBOT)
DPOINT(NB)=II

481 CONTINUE
318 CONTINUE

DO FOR B-GROUP

DO 350 IX=1,4
II=1IBCHR(IX)
IF (II.EQ.0) GOTO 350
CUROLN=CHDIR(1,II)
NL=CHDIR(2,II)
NS=CHDIR(3,II)
IBC=CHDIR(8,II)
MC=0
LPC=1024/NS

DO 1880 I=1,NL
CALL GET(OUF, CUROLN, IB)
CUROLN=CUROLN+1
DO 1881 III=1, LPC
NG=NS+1
DO 1830 J=1, NS
LRGBUF(J, NG)=OBUF(J+IB)

1830 CONTINUE
ID=1B+NS
IF (NG.GE.NL) GO TO 1811

1880 CONTINUE
1881 CONTINUE

EXTRACT THE SO-CALLED B-BAND

ISTART=FLOAT(IBCT+NL)/100.0
ISTART=NL-ISTART
IEND=ISTART+1
NL2=NL/2

IF (IEND.GT.HL2) IEND=HL2

NPB=0
BIOD=0.0
DO 351 I=ISTART, IEND
DO 352 J=1, NS
IVAL=LRGBUF(J, I)
IF (IVAL.EQ.0) GO TO 352
D1X=BIOD+FLOAT(IVAL)
NPB=NPB+1

352 CONTINUE
351 CONTINUE

BX(IX)=BIOD/FLOAT(NPB)

CLASSIFY THE B-GROUP
DO 360 I=1,4
DENMAX=0.0
DO 361 J=1,4
IF(DJ(J).LE.DENMAX) GO TO 361
IOBJ=1IBCHR(J)
IF(IOBJ.LT.1) GO TO 361
DENMAX=DB(J)
JS=J
361 CONTINUE
OFS(12+I)=IOBJ
BB(J)J)=0.0
360 CONTINUE
C C DO FOR F-GROUP
C C DO 450 IX=1,4
IF(II.EQ.0) GOTO 450
C C CUROLN=CHDIR(1. II)
NL=CHDIR(2. II)
NG=0
LPC=1024/NS
C C DO 2380 I=1,NL
CALL GET(0BUF..CUROLN. 1B)
CUROLN=CUROLN+1
DO 2301 III=1,1,LPC
NG=NG+1
DO 2330 J=1,NS
L_BUF(J,NG)=0BUF(J+IB)
2330 CONTINUE
IB=1B+NS
IF(NG.GE.NL) GO TO 2311
2381 CONTINUE
239 CONTINUE
2311 CONTINUE
C C EXTRACT INTEGRATED OPTICAL DENSITY
C C G10=0.0
NPG=0
DO 451 I=1,NL
DO 452 J=1,NS
IVAL=L_BUF(J, I)
IF(IVAL.EQ.0) GO TO 452
G10=G10+FLOAT(IVAL)
G10=G10/FLOAT(NPG)
452 CONTINUE
451 CONTINUE
450 CONTINUE
C C CLASSIFY THE F-GROUP
C C DO 460 I=1,4
DENMAX=0.0
DO 461 J=1,4
IF(CJ(J).LE.DENMAX) GO TO 461
IOBJ=1IFCHR(J)
IF(IOBJ.LT.1) GO TO 461
DENMAX=CD(J)
JS=J
461 CONTINUE
OFS(59+J)=IOBJ
CD(J)J)=0.0
460 CONTINUE
C C DO FOR G-GROUP
C C DO 510 IX=1,4
II=1IGCHR(IX)
IF(II.EQ.0) GOTO 510
4
C

CUROLN=CHDIR(1,II).
NL=CHDIR(2,II).
HS=CHDIR(3,II).
MG=0.
LPC=1824/NS.

C

DO 1300 I=1,NL
CALL CET(0BUF,CUROLN,IB)
CUROLN=CUROLN+1.
DO 1301 I=1,LPC
MG=MG+1.
DO 1330 J=1,HS
LRGBUF(J,NS)+=0BUF(J+IB).
1330 CONTINUE
IB=IB+NS
IF(NG,CE,HL) GO TO 1311
1300 CONTINUE
1311 CONTINUE.

C

EXTRACT DENSITY PROFILES.

C

DO 590 K=1,NL
NP=1.
IPROF(K)=0.
DO 595 L=1,HS
IVAL=LRGBUF(L,K)
IF(IVAL.EQ.0) GO TO 591
IPROF(K)=IPROF(K)+IVAL.
NP=NP+1
591 CONTINUE
IPROF(K)=IPROF(K)/NP
590 CONTINUE.

C

ANALYZE THE BANDS IN THE C-GROUP.

C

DO 511 K=1,NL
511 JPROF(K)=0.

C

NL=NL-2.
DO 513 K=3,NL+2
IV=0.
DO 512 KK=1,5
IV=IV+IPROF(K+KK-3)
513 CONTINUE
JPROF(K)=IV.
512 CONTINUE.
MAX=JPROF(J).
DO 515 K=1,NL+2.
IF(MAX.GE.JPROF(K)) GO TO 514
MAX=JPROF(K).
514 CONTINUE.
KBAND=K.
515 CONTINUE.

C

NSAME=0.
DO 516 K=KBAND,NL+2.
IF(JPROF(K).NE.MAX) GO TO 516
NSAME=NSAME+1.
516 CONTINUE.
KBAND(K)=KBAND+NSAME/2.

C

CLASSIFY TH C-GROUP.

C

DO 521 I=1,4
MAXBP=0.
DO 522 J=1,4
IF(KBAND(J).LE.MAXBP) GO TO 522
108J=ICHR(J).
IF(108J.LT.1) GO TO 522
MAXBP=KBAND(J).
521 CONTINUE.
DO 601 I=1,6
   RMAX=0.0
   DO 602 J=1,6
      IF(J .LE. RMAX) GO TO 602
   IDB=BDPOINT(J)
   IF(IDB.LT.1) GO TO 602
   RMAX=RATIO(J)
   JS=J
   CONTINUE
602 CONTINUE
601 CONTINUE
600 CONTINUE
C
1000 FORMAT(110)
2000 FORMAT(3X,3913)
3000 FORMAT('ICHROMOSOME ','12)
4000 FORMAT(10F10.2)
C
CALL WPARAN(92,KPAR,9)
CALL CLOSE(OBUF)
CALL APHASE(9)
C
CALL EXIT
END
C**********
C
AEBNORM =
C**********
C
ROUTINE TO ASSESS NORMALITY OF CHROMOSOMES IN A SPREAD
C
"SRoutine AEBNORM
COMMON /CI/ NOB, IDIR
COMMON /CI/ CHDIR, SMLBUFF, LRGBUF, HS, ML, CUIROLN
COMMON /CI/ SST, EST
COMMON /CI/ SPFIO, SPLTH, SPAREA
COMMON /CI/ MCHR
COMMON /CI/ CFOUR, PHI
COMMON /CI/ BAND, AHIS
COMMON /CI/ GBUFF, BUFI BYTE, LRGBUF(50,90)
BYTE OBUFF(2124)
BYTE TEMP8
LOGICAL OPT1, OPT2, OPT3
INTEGER IPROF(90)
INTEGER UFETUR(20)
INTEGER TEMPI
INTEGER SPFAR(5)
INTEGER MAX
INTEGER BAND(180)
INTEGER BPL
INTEGER CHB1R(15,60)
INTEGER CUIROLN
INTEGER$^2$ SST(76), EST(76)
INTEGER PAR(18)
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( SPFAR(2),BPL)
EQUIVALENCE(CFOUR(1,1), AFOUR(1,1))
EQUIVALENCE (PHI(1,1), BFDR(1,1))
EQUIVALENCE (NL,NL0)
EQUIVALENCE (HUMO. II)
DATA L0P1//"TI"/
DATA L0P2//"T2"/
DATA L0P3//"T3"/
DATA L0P//"0P"/

CONTINUE

CALL ARFILE(obuf, 1, "RCE", "13", "13")
CALL OPEN(obuf, 1024, 1, 0, "RCA")
CALL GLABEL(obuf, 9, 10)
CALL GET(obuf, 1, 1B)
CALL ML(obuf, 1B, SPOD, 12)
NCB=10(obuf, 1B)
CALL ML(obuf, 1B+31), CHDIR(1, 1), 2*15*30
CALL ML(obuf, 1B+31), CHDIR(1, 31), 2*15*30

DO 310 II=1, NCHR
JFETUR(1)=0
WRITE(5, 3000) II
CONTINUE

LASTIB=IB
CURLIN=CHDIR(1, 1)
NL=CHDIR(2, II)
NS=CHDIR(3, II)
IF(NL.LT.1. OR. NS.LT.1) GO TO 310
NG=0
LPG=1024/NS
LINES=GET

DO 300 I=1, NL
CALL GET(obuf, CUROLN, IB)
IF(NL.LT.1. OR. NS.LT.1) GO TO 310
DO 300 III=1, LPG
NG=NG+1
CALL SSWITCH(9, JAY)
IF(JAY.EQ.1) WRITE(5, 3000) (obuf(J+IB), J=1, NS)
CONTINUE
DO 330 J=1, NS
LRGBUF(J, NG)=dbuf(J+IB)
CONTINUE
IF(CN. GE. NL) GO TO 311
CONTINUE
CONTINUE
CONTINUE

MAIN LOGIC

NSM=NS-1
JTH = LINMAX / 2
IPROF(K) = NS

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

IPROF(K) = 0
CONTINUE

IPROF(K) = IPROF(K) + 1
CONTINUE

IPROF(K) = IPROF(K) + 1
CONTINUE

IF (IPROF(K) .GE. NS) IPROF(K) = B
CONTINUE

DO 630 K = 1, NLH

P=IP(TOT) / NLH
CONTINUE

DO 620 K = 1, HL

IF (IPROF(K) .LT. 0) HL80 = HL80 + 1
CONTINUE

IPROF(K) = IPROF(K) + 1
IPROF(NL) = 1
IPROF(NL - 1) = 1

CONTINUE

IF (JAY .EQ. 1) WRITE (5, 1308) (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-
SIXTED, LOWEST DIP IN PROFILE IS FIRST OR LAST ONLY IN SEQUENCE OF NEG NUMBERS THAT 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.LT.NL-4) GO TO 750
IF(IPROF(K).GE.B) GO TO 760
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

SCAN TO POS NUMBER K=K+4
K=K+1
IF(IPROF(K+1).LT.0) GO TO 701
GO TO 760

CONTINUE

MINUS=1
DO 720 K=1,6
K=K+2
IF(K.LT.NL) GO TO 721
IF(IPROF(K).GE.0) GO TO 720
MINUS=MINUS+1
720 CONTINUE

MINUS=MINUS+1
GO TO 700

IFETUR=IFETUR+1
KFETUR=2
JFETUR(IFETUR)=KFETUR
KFETUR=K+4
K=K+1
IF(IPROF(K+1).LT.0) GO TO 731
GO TO 760

CONTINUE

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

FOROM(' FEATURES; ',2013)
257 4,122,518

C
IF(NL.T.0.GT.NL.) GO TO 940
GO TO 941
CONTINUE

C
WRITE(5,1094)
FORMAT(1,1894)
GO TO 950
CONTINUE

C
IF(IFETUR.EQ.0) GO TO 908
IF(IFETUR.EQ.1) GO TO 910
IF(IFETUR.EQ.2) GO TO 926
IF(IFETUR.EQ.3) GO TO 930

C
GO TO 940
CONTINUE
GO TO 950
CONTINUE

C
920 CONTINUE
IABIND=IFETUR(1)+IFETUR(2)
IF(IABIND.EQ.2) WRITE(5,1292)
IF(IABIND.EQ.3) WRITE(5,1392)
IF(IABIND.EQ.4) WRITE(5,1492)
1292 FORMAT(9,1292)
1392 FORMAT(9,1392)
1492 FORMAT(9,1492)
CONTINUE
GO TO 950

C
910 CONTINUE
IF(IFETUR(1).EQ.1) GO TO 911
WRITE(5,1292)
GO TO 912
CONTINUE
WRITE(5,1191)
911 FORMAT(9,1191)
1191 FORMAT(9,1191)
CONTINUE
GO TO 950

C
900 CONTINUE
IF(NL.LT.30) GO TO 901
WRITE(5,1992)
GO TO 903
CONTINUE
WRITE(5,1901)
901 FORMAT(9,1901)
903 FORMAT(9,1903)
CONTINUE

C
1901 FORMAT(9,1901)
1902 FORMAT(9,1902)
CONTINUE

C
950 CONTINUE

C
CALL EXIT

C
WRITE(5,7030)
CALL APhase(13)
CALL CLOSE(obuf)

C
1000 FORMAT(10)
2000 FORMAT(3X,3913)
3000 FORMAT(9,129)
3030 FORMAT(9,12)
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 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,
   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 programmed 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.
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 moving said stage under said microscope to successively present said plurality of dispersed metaphase chromosome spreads to the optics of said microscope includes:
   moving said stage with a scanning motion in successive predetermined motion increments underneath the optics of said microscope,
   detecting whether or not a chromosome spread is present within the distance of the predetermined increment of motion, and
   storing the location data for each detected 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 chromosomes 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.