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 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 the karyotype serves primarily as a tool of increasing diagnostic value, the karyotype has emerged as a tool of increasing diagnostic value. 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 30 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 operator; however the computer has the option to override the operator mode and/or to inhibit subsequent selection of the operator mode. The mode of operation is displayed by a stage display 26. A typewriter 27 serves to enable entry of data, instructions or queries into the computer or to receive data from the computer.

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

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

The computer 20 processes each digital image, as will be described and arranges each image in a karyotype format. A gray level display system 37 displays a karyotype image 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 a certain location 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 correct karyotype information received from the computer. This is applied to the hard copy interface circuit 40. This constitutes a circuit for converting the corrected digital karyotype digital signals to analog form and the associated timing required for enabling the hard copy output device 42, which is essentially a facsimile printer, to print out a karyotype picture from the analog signals.

FIG. 2 is a block schematic diagram illustrating the circuits used in controlling movements of the motorized stage of the microscope. The stage is driven in the X, Y and Z directions respectively by three motors 50, 52, and 54. The Z motor is driven in the +Z or −Z direction by signals from the computer in response to focus drive apparatus 56 in the Spread Detector and Automatic Unit 36 (FIG. 1). Both X and Y motors are driven in the +X or −X directions and +Y or −Y directions in response to outputs from a +X Or gate 58, or a −X Or gate 60 and a +Y Or gate 62, or a −Y Or gate 64. The inputs to these Or gates, come either from the computer 20, through logic circuits to be described or from the joystick pulse generator 66, as determined by whether the operation is to be in the manual mode or the computer mode.

In the manual mode, the direction in which the joystick is moved determines whether pulses are provided on a +X, or a −X or a +Y or a −Y line. +X and −X pulses are applied, in response to joystick motion, from the joystick oscillator to And gates 68 and 71 respectively. These And gates are enabled in response to the output of a mode flip flop 74. When flip flop 74 is set by means of switch 76 to its manual mode, then And gates 68 through 72 are enabled. The other output of the flip flop indicates that the system is in its computer mode. Flip flop 74 can be set in its computer mode by means of a manual switch 76, or the computer can override the switch 76 if a situation arises where computer operation must be maintained.

When in the computer mode, the output of flip flop 74 enables an And gate 78 to transfer pulses received from a pulse generator 81, within the computer 20, to another And gate 80, and also to four And gates 82, 84, 86 and 88 whose outputs respectively provide pulse inputs to the Or gates 58, 60, 62 and 64. And gates 82 and 84 are three input And gates. One of the inputs required to enable And gate 82 is the output of an X direction flip flop 90. The computer determines which direction the stage is to move, when the system is in the computer mode, and for one direction sets flip flop 90 with its +X output whereby And gate 82 receives one of its two required enabling inputs. The computer sets the flip flop with its −X output high for the opposite direction, whereby And gate 84 receives one of its two required enabling inputs. Similarly a Y direction flip flop 92 is provided which for one direction is set with its +Y output high thereby providing one of the two enabling inputs required by And gate 86, and for the opposite direction its −Y output is high whereby And gate 88 receives one of two required enabling inputs.

The second required enabling input to And gates 82 and 84 is provided by an inverter 94, which is driven by the output of a zero detector 96. The zero detector senses when an X motor drive register 98 has reached zero at which time it supplies an output to the inverter 94 whereby And gates 82 and 84 are disenabled.

A similar function is provided by a Y motor drive register 100. When it reaches zero, this is sensed by a zero detector 102. The zero detector output drives an inverter whereby And gates 86 and 88, which are otherwise enabled becomes disenabled.

The absolute X position of the microscope stage 12 is maintained by an X cumulative position register 106, which can provide this information to the computer. The X cumulative position register is a reversible counter which can receive +X and −X pulses which are applied to the X motor 50, whereby the X cumulative position register is driven to retain the exact X position of the microscope stage. The computer generates an X coordinate number representing the location to which it desires the microscope stage to move in the X direction. This number is subtracted from the number in the X cumulative position register. The difference
represents the distance along the X axis, the microscope stage must move and also represents the number of pulses required to complete such a move. This number is entered into the X motor drive register 98 by the computer. The pulse generator 81 is then instructed to commence supplying pulses, and if AND gate 78 is enabled, as it should be in the computer mode, then these pulses are applied through AND gate 80 to the X motor drive register 98, causing it to count down towards zero. These pulses from the pulse generator 81 are also applied to all of the 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 Y position of the microscope stage 12. It is driven in response to the same pulses which are applied to the +Y and -Y inputs to the Y motor 52. This cumulative position is supplied to the computer which determines the location along the Y axis to which it desires the microscope stage to move. This is subtracted from the Y cumulative position information and the difference is applied to the Y motor drive register by the computer. Pulses from the pulse generator 81 are applied to an AND gate 110, which is enabled when the Y motor drive register is not in its zero position. These pulses start to drive the Y motor drive register and are also applied to the +Y and -Y AND gates 62 and 64. The one of these two which is enabled is determined by the output of the Y flip flop 92.

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

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

FIG. 3 is a flow chart exemplifying the search mode of operation. When the computer starts a search the circuitry shown in FIG. 2 is instructed to move a predetermined amount in the X direction. In the embodiment of the invention which was constructed, each step of motion in either X or Y direction is 10 microns, however each move instruction from the computer in either the X or 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 built, this equipment, called the model 108 Computer Eye, was furnished by a company called Spatial Data Systems, Inc. The camera scanned the image at the standard television rate (525 lines, 60 frames per second, interlace 2:1) and produced a video signal which was monitored on the television display. The brightness value in the scan picture is sampled at each point of a 512 by 480 grid and converted to a 7-bit binary number. Points anywhere in the picture may be selected on the program control, or the digitization can proceed through all points in sequence. A cursor is displayed on the monitor to show the points to be digitized.

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

FIG. 4 represents a flow chart illustrating the steps in the analysis phase of operation which have just been described. SDS stands for scan data set. Each chromosome is tagged with a number. Where operator intervention is required, in the case of the spread 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 indentification technique, that is a general description of a method of chromosome identification which was implemented by the programming is described in a volume entitled Chromosome Identification edited by Torbjorn, Casperson and Lore Zech, which is a publication of Hughes Aircraft 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
#DKS: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 = 8 scan disk unit (DKS)
HDU = 3 hardcopy disk unit (DK3)
```

The scan data sets are named S1,S2,S3. The hardcopy data sets are named H1,H2. They are stored under UIC [2,2]. Allocation of data sets is done by:

```
NFSDS: .WORD -. # full scan data sets (F)
SDS: .BLKB NSDS if NSDFS 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 = full (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).

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

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

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

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

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

Purpose: Slide search edit, and focus. SEARCH partion 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

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 X 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 X 480 pixels is reduced to a 250 X 240 grid of sample points. The resulting binary image is surrounded with zero bits to provide a physical boundary to keep the perimeter walker used in the segmentation algorithm from wandering off the edge of the picture. The binary reduction is accomplished through multiple calls to the subroutines QTHR or STHR.

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

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

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

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

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

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

---

**Flow Chart**

---

**CALMS Segmentation Sequence**
SKIRT

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

SKIRT will increase the area of each chromosome by extending its boundary outward a uniform distance in all directions. (Flow Chart 6.) Each chromosome is thus skirted with marginal elements to prevent loss of data when the boundaries are recomputed during the 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.

Each chromosome record has the following format:

<table>
<thead>
<tr>
<th>WORD</th>
<th>意义</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NUMBER OF THE PREVIOUS RECORD WRITTEN FOR THE CHROMOSOME (4 FOR THE FIRST RECORD OF EACH CHROMOSOME)</td>
</tr>
<tr>
<td>2</td>
<td>NUMBER OF SEGMENTS IN THIS RECORD</td>
</tr>
<tr>
<td>3 AND 4</td>
<td>LINE AND SAMPLE COORDINATES FOR FIRST SEGMENT</td>
</tr>
<tr>
<td>5</td>
<td>NUMBER OF SAMPLE POINTS FOR FIRST SEGMENT (N1)</td>
</tr>
<tr>
<td>6 TO 4N1+15</td>
<td>GREY LEVELS FOR FIRST SEGMENT. SINCE EACH SAMPLE POINT REPRESENTS A 2 X 2 PIXEL AREA, 4N1 PIXELS ARE STORED, OR 2N1 PIXELS EACH FROM TWO CONSECUTIVE LINES.</td>
</tr>
<tr>
<td>4N1+16 TO ...</td>
<td>SIMILAR INFORMATION FOR THE SECOND SEGMENT</td>
</tr>
</tbody>
</table>

FLOW CHART 8 Chrome File Format
Purpose: To rethreshold and resegment the chromosomes and to store them in a chromosome file.

ROB requires as input a chromosome file containing segment end point coordinates and gray level information for each chromosome. Each chromosome is reassembled in core and its histogram is generated. Based on its display the picture on the gray scale, and calls SUPER (Son of Turtle), which records the perimeter of the object on a counterclockwise direction. The coordinates are sorted to correspond to segment end points by the subroutine SOTIN.

The segmentation process begins with a line by line scan for pixels above the computed threshold. This task is performed by the subroutine ISEG, which records the location of any objects that it finds. The task of tracking the perimeter of the object is performed by the subroutine SOT (Son of Turtle), which records the segment end point coordinates as it walks around the object on a counterclockwise direction. The coordinates are sorted to correspond to segment end points by the subroutine SORTIN.

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

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

The first record of the chromosome file contains the following:

```
WORD
1 Number of chromosomes (maximum of 60)
2-241 Chromosome directory containing the following four numbers for each chromosome:
FBW = First chromosome record written
YMIN = Minimum line coordinate
XMIN = Minimum sample coordinate
1 242-
302 Thresholds used for each chromosome
```

The second record contains the following:

```
WORD
1-60 Line coordinate of an internal point for each chromosome
61-120 Sample coordinate of an internal point
121-180 Perimeter measure for each chromosome
```

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

```
WORD
1 Number of segments in this records
2 Line coordinate for first segment
3 Sample coordinate for first segment
4 Number of samples (N) in segment
5 to N+4 N Gray levels for this segment
N+5 Line coordinate for second segment
```

Purpose: To number the objects found by SEGMENT or ROB and display the picture on the gray scale.

NOB reads 122 words of parameters into NK, LT(60), ST(60), and FL. If FL is zero, NOB does not display the picture on the gray scale, and calls APHASE (MASKPH).

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.

INTI

Purpose: To correct errors in object isolation.

INTI 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, INTI is called again to check for any remaining errors.

INTI communicates with CALMS by calling APHASE. It writes parameters to BINARY to indicate a quick count, and to NOB to tell it to call MASK and not display the picture when finishing up a count.

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. 1: Word (1) = # of input objects (integer)
Word (2,3) = Spread IOD (real)
Word (4,5) = Spread length (real)
Word (6,7) = Spread area (real)
Word (8) = Next available record in RCR (integer)
Word (9) = # of chromosomes
Words (16-465) = Chromosome directory
Rec. 2: Words (16-465) = Chromosome directory
Rec. 3: Reserved for CLASSFY results
Rec 4-7: Reserved 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>WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</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.

### CLASFY

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

CLASFY reads in the chromosome directory produced by MOB and classifies each object. Once classification is complete, CLASFY 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 CLASFY with the OS parameter, followed by the object numbers for the slots. In this case, CLASFY sets up the karyotype format but does not do any classifications.

### KTYPE

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

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

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

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

### INT2

**Purpose:** Provide operator interaction to fix karyotype errors.

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

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

### RESEL

**Purpose:** Save information on measured chromosomes.

RESEL uses the MOB output to rearrange the measurements in order of type and store them in KDATA (BDATA for banded spreads). It also writes the patient report line in PDATA.
FOUR

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

BAND recognizes the following parameters:

**B**: Store the waveform in a file for each chromosome image, regarding its normality. 

**A**: Mark the axis for each chromosome.

**W**: Store a representation of the waveform in place along the line.

**X**: Mark the axis for each chromosome.

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

**FOUR**

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

The inputs to FOUR are the classification table of means and variances, and the banded chromosome measurements. FOUR computes the likelihood that each chromosome belongs to each of the chromosome types, using 14 measurements — length, centromeric index by area, C(2) to C(8), and PHI (2) to PHI (6). If a chromosome's length or centromeric index differs by more than 6 S.D. for a particular chromosome type, that type is ruled out for that chromosome. Similarly, if the C sum or PHI sum exceeds 7 S.D. or the total sum exceeds 8 S.D., that type is excluded for that chromosome.

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

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

The classification results are written as parameters for CLASFY.

**PREP**

Purpose: To print the patient report

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

**ABNORM**

Purpose: To print information on abnormal chromosomes ABNORM is called only when console switch 10 is down. It reads the rotated chromosome file, RCR, and examines the profile of each chromosome to determine centromere information. It then prints a message for each chromosome, regarding its normality.
KFIX
Purpose: Syntactical classification within B,D,F, and G groups.
KFIX is the final phase of the hybrid classifier for banded chromosomes, and operates as follows:
(1) Take the 4 chromosomes in the G group. Measure the position along the length of the chromosome of the brightest band. The 2 chromosomes with the bright 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,BUFSIZ,MVB,MODE,LNAME)
CALL GET (MVB,LINE,INDEX,NORAJ)
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 MVIO 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:

- **MVB** 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>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disk Input</td>
</tr>
<tr>
<td>1</td>
<td>Disk Output</td>
</tr>
<tr>
<td>2</td>
<td>Disk Update</td>
</tr>
<tr>
<td>3</td>
<td>Tape Input</td>
</tr>
<tr>
<td>4</td>
<td>Tape Output</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MODE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Tape Output</td>
</tr>
</tbody>
</table>

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

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

**INDEX** is the offset in bytes from the start of the MVB to the requested line. In Fortran, when 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.
READ and WRITE can be used when RECLLEN = BUFSIZ and single buffering is specified. A line is read into LOC, or written from LOC, without any overlap.

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

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

The MVB format is shown below. Word numbers are in decimal, starting at 1 and byte numbers are in octal, starting at 0. TB = tran block, LB = link block, and FB = filename block.

<table>
<thead>
<tr>
<th>WORD</th>
<th>BYTE</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>TB</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Active Block #</td>
</tr>
<tr>
<td>3</td>
<td>+2</td>
<td>Active Buffer Address</td>
</tr>
<tr>
<td>4</td>
<td>+4</td>
<td>Word Count</td>
</tr>
<tr>
<td>5</td>
<td>+6</td>
<td>Function/Status</td>
</tr>
<tr>
<td>6</td>
<td>+8</td>
<td>Words Not Transferred</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>IBN</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>Inactive Block #</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td>IBA</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>Error Return</td>
</tr>
<tr>
<td>11</td>
<td>-2</td>
<td>Link Pointer</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>Logical Name</td>
</tr>
<tr>
<td>13</td>
<td>-4</td>
<td>Unit # Words To Follow</td>
</tr>
<tr>
<td>14</td>
<td>-6</td>
<td>Device Name</td>
</tr>
<tr>
<td>15</td>
<td>30</td>
<td>CRC</td>
</tr>
<tr>
<td>16</td>
<td>-2</td>
<td>Indicator (Unused)</td>
</tr>
<tr>
<td>17</td>
<td>34</td>
<td>FB</td>
</tr>
<tr>
<td>18</td>
<td>36</td>
<td>File Name (Word 1)</td>
</tr>
<tr>
<td>19</td>
<td>+2</td>
<td>File Name (Word 2)</td>
</tr>
<tr>
<td>20</td>
<td>+4</td>
<td>Extension</td>
</tr>
<tr>
<td>21</td>
<td>+6</td>
<td>UIC</td>
</tr>
<tr>
<td>22</td>
<td>44</td>
<td>WNT</td>
</tr>
<tr>
<td>23</td>
<td>46</td>
<td>SBN</td>
</tr>
<tr>
<td>24</td>
<td>50</td>
<td>NBF</td>
</tr>
<tr>
<td>25</td>
<td>52</td>
<td>NLR</td>
</tr>
<tr>
<td>26</td>
<td>54</td>
<td>RECLLEN</td>
</tr>
<tr>
<td>27</td>
<td>56</td>
<td>BLKSIZ</td>
</tr>
<tr>
<td>28</td>
<td>60</td>
<td>BUFSIZ</td>
</tr>
<tr>
<td>29</td>
<td>62</td>
<td>BPP</td>
</tr>
<tr>
<td>30</td>
<td>64</td>
<td>DBF/MODE</td>
</tr>
<tr>
<td>31</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, RECLLEN, NLR, and BPP 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.'

Label is the location of the label to be output.

Example:

```
INTEGER SPAR (5)
BYTE A (4200), B (4200)
CALL OPEN (A, 2648, 1, 'MVI')
CALL OPEN (B, 2648, 1, 'MV2')
CALL GLABEL (A, SPAR (IA + 1))
CALL PLABEL (B, SPAR, A (IA + 1))
```

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:

```
JSR R4, SAVER
```

To restore registers 0-4 and return via R5:

```
JMP RESTR
(SAVER is called via R4, and R5 is not saved.)
```
AFILE

Purpose: To assign a file to a dataset.

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

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

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

PARAM

Callable Entry Point Names: PARAM, PARBUF

Purpose: To read free-field parameters from the keyboard and convert them.

PARAM types an *, then reads up to 80 characters from the keyboard, terminated by a carriage return.
Parameters are separated by blanks or commas, and can be one of the following:
1. One word integer — negative integers are preceded by a minus sign.
2. Two word alphanumeric — 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 alphanumeric — the string is enclosed in apostrophes. If an odd number of characters, the last word is padded with a trailing blank. If an apostrophe is desired, type two consecutive apostrophes.
The raw character string is saved at 'PARBUF' which is accessible to MACRO programs, but not FORTRAN. However, the user can optionally supply his own PARBUF.
The calling sequence is:
CALL PARAM (NP, PAR, MAXNP [PARBUF])
NP is the number of parameter words that were stored in PAR
MAXNP is the maximum allowable number of parameter words (the size of PAR)
PARBUF is optional 83 byte user buffer for the raw character string.

Example:
INTEGER PAR (20)
CALL PARAM (NP, PAR, 20)
If the user types:
NH, PRINT, 42, -1, 'ABC' 'DEF'
The result is:
PAR (1) = NH
PAR (2) = AA
PAR (3) = PR
PAR (4) = IN
PAR (5) = 42
PAR (6) = -1
PAR (7) = AB
PAR (8) = C'
PAR (9) = DE
PAR (10) = FA
NP = 10
PARAM is reentrant if the user supplies PARBUF. However, an earlier keyboard input request must be satisfied before a later one can be done.
PARAM maintains a byte with the global name 'PBUSY'. It is set to one when PARAM is entered, and cleared when PARAM exits.

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).

QPRINT

Purpose: To print a message, with automatic buffering

QPRINT waits for completion of a previous print, moves the message to its own buffer, initiates a print with a TRAN request, and returns to the user.

CALL QPRINT (LOC [, NBYTES])
LOC is the first byte to be printed (carriage control).
NBYTES is the number of bytes to be printed. (maximum — 132).
NBYTES can be omitted if the message is terminated with a "zero" byte.

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

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

TYPE

Purpose: To type a message with automatic buffering

TYPE is similar to QPRINT, except that output is on the keyboard and the maximum message length is 80 bytes.

CALL TYPE (LOC [, NBYTES])
If NBYTES is omitted or zero, a null (0) terminates the message. TYPE inserts a <CR> at the end of the message unless NBYTES = 0.

IV

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

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

IV (byte) = Integer value (0-255) of a byte variable
CALL ITL (INT, L1) (INTEGER TO LOGICAL)
Move the low order byte of INT to L1
CALL MVL(L1, L2, N) (Move LOGICAL)
Move N bytes starting at L1 to L2
CALL MVW (I1, I2, N) (Move WORD)
Move N WORDS starting at I1 to I2
CALL ZIA (BUF, N) (Zero INTEGER array)
Zero N WORDS starting at IBUF
CALL ITLA (INT, L1, N) (INTEGER to LOGICAL array)
Store INT in N bytes starting at L1
CALL SWAP (I1, I2)
Interchange I1 and I2.
Note: RO is not saved by these routines!

TEXT

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

CALL TEXT(INBUF, INCHR, LINE, OUTBUF, SIZE)
INBUF is the location of the input ASCII characters
INCHR is the # of input characters
LINE is the line number (0-6) of the bit array
OUTBUF is the location for the output bytes
SIZE is the number of bytes to store for each bit (1,2,...)
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)
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.

PARAM
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, NPAME [PNUM]) To read parameters through the CALMS supervisor. See the PARAM writeup for details. PNUM is an optional phase number used to read another phase’s parameters.
CALL APhase (N) Normally, phases are loaded in sequence, but APhase is used to change the sequence. N is the number of the phase.

DLINE
Purpose: Subroutines used to access the gray scale and cursor, when running under the CALMS supervisor.
CALL 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,LINE,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 IXREL (XDEL) Initiates a relative X move
CALL IYABS (Y,Y) Initiates a relative Y move
CALL IYREL (YDEL) Initiates a relative Y move
CALL IXYABS (Y,Y) Initiates a relative X move
CALL IYABS (Y,Y) Initiates a relative Y move
CALL CFOC (F) Stores the sum of the two focus parameters
CALL LED (CODE, VAL) Puts VAL in the LED whose code is given
CALL MFST (MAGN, FPOS) Stores the magnification and filter position in MAGN and EPOS, and displays the magnification in the magnification LED. It also stores FSTEP, FLAST, XADJ, and YADJ for the lens in use.
CALL SORTQ Sorts the spread queue according to its rating.

ROACH
FUNCTION: ROACH (WI,BI,WORD,EWI,MASK)
ROACH will scan a binary line to locate the first set bit. The scan will begin at 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,NW2)
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.

**Segment End Point Decision Table**

FLOW CHART 10  Segment and Point Decision Table
SORTIN

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

CALL SORTIN(EDGE,N,IND)

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

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

ERASE

ERASE is invoked by SEGMENT to remove a chromosome from the binary spread image.

CALL ERASE(WORD,EDGE,AREA,N2,NW)

WORD points to the buffer area containing the binary image.
EDGE points to the end point coordinates for the chromosome.
N2 is the number of coordinates in EDGE (two per coordinate pair).
NW is the number of words per line in the binary image.
**ORIOB**

**Purpose:** Orient objects.

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

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

**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,SIN,COS,NL,EP,XMIN,XMAX,YMIN,YMAX)

CHROUT rotates the object in IBUF by the spec-

**Purpose:** Rotates chromosome endpoints.

**CALL** EPROT(SIN,COS,NL,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,YMZX).

**Purpose:** To rotate objects.

**CALL** OBROT(XMIN,XMAX,YMIN,YMAX,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.
CURRENT ORIGIN IS 30000

SOURCE WORD 6
MSCAN WORD 0
PHUM WORD .=

PHNUM ASCIl \12345679\ PATIENT IDENTIFICATION
SLNUM ASCIl \12341\ SLIDE IDENTIFICATION
SEX ASCIl / \ PATIENT SEX
SPH. WORD CURRENT SPREAD NUMBER
CSPQ. WORD SPQ+.- CURRENT LOC IN SPO
PR. BLKB 300 SPREAD RATINGS
SPO. BLKM 600 SPREAD QUEUE (X,Y)
LPHNUM WORD 10 LENGTH OF PNUM IN BYTES
LSLNUM WORD 6 LENGTH OF SLNUM
MPHASE. WORD NEXT PHASE TO BE CALLED
PAGE
CALMS CALL TYPE. (MSC2. ZER0) ASK ABOUT RESTART

MOV 16. R1
CLRS LINE
CLED. CALL LED. (LINE. ZERO) INITIALIZE LENS
INC LINE
S09 R1. CLED.
CALL OPEN.\{MVZ. PSIZE. ZERO. ZERO. LNAM\} OPEN OVERLAY DATASET
MOV PSIZE. R1X. RECLEN
CALL OPEN.\{PV10. PAT2Z. ZERO. TOO. PLN\} PARAMETER DS
MOV PARSZ. PMVB. RECLEN
MOV #4. DNPTSHB. INTPH+1NTPH-2. SET TO READ 4 CURSOR ADJANTS
MOV #512. HPTAB ALLOW RESTART
MOV #512. DNPTAB
MOV #125. R3B. BR SET UP THE START BLOCK NUMBERS FOR
MOV #RSHN. R1 ALL HARDCOPY DATA SETS
C05. MOVB R0. FILPEX+1
BIS #60. FILPEX+1 CONVERT TO ASCII
CALL AFILE. A. LHDU. FILPEX. TVG. TOO
CALL OPEN.4. TWELVE. ZERO. ZERO. LNAM)
MOV A+RSH. (R1)+ SAVE THE START BLOCK NUM
S09 R0. C05
CALL SEARCH INITIAL CALL TO SEARCH
JSR PC. PPAR PROCESS RESTART AND PHASE PARAMETERS
TST HSCAN
BNE S10 IF A SCAN RESTARTS
S06. JSP PC. THSCAN IF HSCAN=0 ALLOW SEARCH START

S10. TSB SFLAG 'IDLE LOOP' STARTS HERE
BNE S43 IF NO SCAN RUNNING
TST HSCAN
BEG S48 IF NO ONE TO SCAN
S13. MOV #MSDS. R0 LOOK FOR SCAN BS
S15. MOV #S0. R1
MOV #HFS0. R2
BEG S22 IF FULL SCAN DATASETS
BR #0. BR "10K TO USE THIS DATASET, STORE IT"
S22. MOVB R0. BR1 IF MORE TO CHECK
BR #0. S15 IF NO ONE AVAILABLE
BEG S40 IF NO TEST
S39. BIS #40. R0 CONVERT TO ASCII
MOV R0. FILPEX+1 STORE IN FILENAME
MOV #S. FILPEX IS SN
MOV #30. BMK1 STORE SCAN DISK UNIT
DEC HSCAN HSCAN=HSCA-1
S31. CALL PHASE. \{ONE\} SCAN
CALL LED. \{S0LED. HSCA\}
CALL TST HPHASE
TST HPHASE
BEG S05 IF SCAN WAS ABORTED
S315. CLR NPTSHB. HOPPHB. HOPPH-2 CLEAR NOB PARAMETERS FOR INT1
CLR NPTSHB. BIPH+2 CLEAR BINARY PARAMETERS
CALL PHASE. INTPH CALL INT1 FOR THE PRE-F03 ADJ
CNB HPHASE. #1 TEST FOR RESCAN OR ABORT
BEG S31 IF IF PE
BLT S05 IF SPREAD WAS ABORTED IN INT1
CNB HPHASE. #HOPPH
BNE S33 IF NOT A COUNT
CALL PHASE. HPHASE CALL MOD

4, 122, 518
MOVB #H.FILPEX, A

MOV #H.DUNIT, HCOPY DISK UNIT

INC -R1) SET DATASET BUSY

MOV R1, CMS SAVE LOC. OF CURRENT MASK DATASET

MOV R2, CSBH SAVE SBH LOCATION

CALL PHASE, <PHASE> ASK MASK

MOV #1, CMS SET DATASET FULL OF DATA

INC HCOPY

CALL LED, <HODE, HCOPY>

CALL LED, <AHL. LED, HCOPY>

JSR HCFLAG

BNE JS18 IF HCOPY IS OPERATING

INC HCFLG SET HCOPY OPERATING

BIS #11.6 LITES TURN ON HCOPY LITE

MOV CMS, CMS+1 SET UP START BLOCK NUMBER

MOV #PSY, -(SP) SAVE PSY TO SIMULATE TEMP

JSR HCOPY

JS18 JMP S10

PAGE

PLOC = 0008 PHASE START LOCATION

MOV R2, R2(PS) R1 STORE PHASE NUMBER

INC R1

MOV R1, NPHASE = NPHASE*PHASE+1

CALL ZIR, (15750H, L377) ZERO 510 BYTES OF UPPER CORE

CALL READ, <HCOPY, PHASE, INDEX, PLOC> READ PHASE PHASE INTO PLOC

ASL R1 ADD #PLITE-4, R1

BIS OR1, OR1.6 LITES TURN ON THE LITE FOR THIS PHASE

CALL PLOC, <DUNIT, FILPEX> CALL THE PHASE, GIVING IT A DATA SET NAME

BIC #11.6, OR1.6 LITES TURN OFF ANALYSIS LITES

RTS R5

PBR

CALL PARAM, <NP, PAR, MAXNP> JSR PC, PBR

DEC NP

BNE PFEXT BR IF HP WAS 0

BEQ PFEXT BR IF NO RESTART

CMPB PAR, R1 ADD OR1.6 LITES, OR1.6 LITES

BEQ RSTRT BR IF A RESTART

 JMP PP40 PHASE NUMBER AND PARAMETERS

PFEXT

RTS PC

RSTRT: TSTB SFLAG JSR PC, RSTRT

CALL TYPE, <M4, ZERO> ASK ABOUT SCAN RESTART

CALL PARAM, <NP, PAR, MAXNP>

CMPB PARBUF, R1

BNE PP05 BR IF NO SCAN RESTART

CALL RPARAM, <NP, SOURCE, L512, ONE> READ RESTART INFO

CALL TYPE, <M7, ZERO> ASK FOR FIRST SPREAD NUMBER

CALL PARAM, <NP, PAR, MAXNP>

MOV PAR, R0

CMP R0, 01 BR IF NUMBER LE 1

CMP R0, HSCAN BR IF CT HSCAN

BGT PP02 LEGAL NUMBER, DECREMENT IT

MOV R0, SPH STORE IT

SUB R0, HSCAN ADJUST HSCAN

ASL R0

ASL R0 N+4

ADD R8, CSPQ ADJUST CSPQ

PP02: CALL GHOSTS GET FIRST SPREAD TO SCAN AND ENABLE SPIRAL SEARCH

CALL LED, <SOLED, HSCAN>

PP05: CALL TYPE, <M5, ZERO> ASK ABOUT ANALYSIS RESTART

CALL PARAM, <NP, PAR, MAXNP>

CMPB PARBUF, R1

BNE PP15 BR IF NO ANALYSIS RESTART

MOV R8, R0

MOV #8, R8

MOV #8, R8
SET ALL SCAN DATA SETS AS FULL

CALL TYPE, (MB, ZERO) ;ASK HOW MANY
CALL PARAM, (HP, PAR, MAXHP)

TST PAR
BLE PPI2 ;BR IF N LE 0
CMP PAR, #SSDS ;BR IF H GE HSDS
MOV PAR, #HSDS ;STORE HSDS

CALL LED, (#AQUEL, HSDS) ;DISPLAY ANALYSIS QUEUE

TSTB, HCFLG
BNE PPEX ;BR IF HCOPY RUNNING
CALL TYPE, (MB, ZERO) ;ASK ABOUT HCOPY RESTART
CALL PARAM, (HP, PAR, MAXHP)
CMPB PARBUF, #Y
BNE PPEX ;BR IF NO HCOPY RESTART
MOV #HBS, B0

CALL HCFLG
BIS #1, #LITES ;TURN ON HCOPY LITE
MOV #PSY, #S (SP)
JSR PC, HCMD ;SIMULATE INTERRUPT TO HCOPY END RTE

PPEX RTS PC
P48 MOV PAR, #PHASE
BLE PERR
CMP B0, #MPH
BGT PERR
ASL B0
ADD #NPAT+2, B0 ;BNPAT+2,PKH=2
MOV #NPAT, #B0 ;STORE HP FOR THE PHASE
CALL MPARAM, (HP, PAR+2, PAR)

JPPAR JMP PPAR
PERR CALL TYPE, (PENSG, ZERO)
BR JPPAR

THSCAN TST MSCAN ;JSR PC, THSCAN
BNE CGHS ;BR IF MSCAN NOT ZERO
CALL TYPE, MSG1 ;TYPE 'OK TO START SEARCH'
CALL SEARCH ;ALLOW SEARCH START
RTS PC

CCNS CALL GNSTS ;GET NEXT SPREAD TO SCAN
RTS PC

MV: .BLK 1B ;MOVE FOR READING PHASE FROM CALMS.OVR
.BYTE 1,5 ;Dk
.RA250 /DK/
.BLK 2
.RA250 /CALMS.OVR/
.BYTE 2,2

L377 .WORD 377
PSIZE .WORD 32236.
TWO .WORD 2
INDEX .WORD 2
FPHASE .WORD 2 ;FIRST ANALYSIS PHASE (OR DEBUG PHASE)
MYPHASE .WORD MYPH;
PPHASE .WORD 16 ;PREP PHASE
EPHASE .WORD 17 ;TEXT EDITOR PHASE
FFASE .WORD 18 ;FORTRAN PHASE
DUNIT .WORD 2 ;DISK UNIT FOR SCAN OR HCOPY DATA SET
LHAM .ASCII 1/DV/
PLHAN .ASCII 1/PAR/
PMVS:  
BYTE 1.1  
RDMS /DK /  
RDMS /CALMS, PAR/  
BYTE 2.2  
BLK 10.  

PARSZ:
WORD 1024.  

PLITE:
WORD 1000, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 10, 10, 10, 10, 4, 2, 2, 2, 2, 2, 10010, 10010, 4000  

NPTAB:  
BLK 10.  

DNPTAB:  
BLK 10.  

HCUC = 172430  
HCAD = 172432  
HCST = 172434  
HCIV = 440  
HCIV = 442  

INTPS = 340  

HCOPY:  
MOV $7777, A  
CLRF ACTIVE BLOCK NUMBER  
MOV $7777, A + IBN  
AND INACTIVE BLOCK NUMBER  
MOV $SLAB, A + 2  
READ LABEL INTO SLAB  
MOV $LABSIZ, A + BUFST  
READ LABSIZ BYTES  
CLR A + DBFLAG  
PREVENT READ-AHEAD  
CALL GLABEL, (A, SPAR, 1A)  
READ THE LABEL INFORMATION INTO CORE AND CALCULATE RECLEN.  
THEY: SET UP THE MYS TO USE TWO 4096 BYTE BUFFERS IN HIGH CORE  
FOR THE DISK TO CORE TO HARD-COPY DMA TRANSFER.  
MOV $7777, A  
CLRF ACTIVE BLOCK NUMBER  
MOV $160000, A + 2  
ONE BUFFER AT 160000  
MOV $7777, A + IBN  
CLRF INACTIVE BLOCK NUMBER  
MOV $170000, A + IBN  
ONE BUFFER AT 170000  
MOV $100000, A + BUFST  
BUFFERS = 4096 BYTES  
MOV $10, A + DBPS  
10 BLOCKS PER BUFFER  
INC A + DBFLAG  
SET DOUBLE BUFFERING

CLR LINE  
INITIALIZE LINE COUNTER  
MOV @INTRY, @HCIV  
INT ENTRY ADDR TO HCOPY INT VEC  
MOV @INTPS, @HCIV  
INT. SVC PSW TO HCOPY INT VEC  
INC BPL  
IN CASE OF ODD BYTES  
MSR BPL  
MD CNT FOR HCOPY LINE IS BPL/2  
MOV BPL, BPL2  
BPL = BPL2  
ADD 54, RMB  
RECLEN  
SUB BPL, RMB  
SUB BPL, RMB  
MOV $160000, @HCAD  
SUB RMB, @HCAD  
CLR @HCUC  
CLEAR HCOPY MD CNT REGISTER  
BIT $2B00, @HCST  
BIT-10 IS HC RUNNING  
BNE ONSL  
WAIT FOR PREVIOUS END-FRAME  
BIT $4999, @HCST  
BIT-11 IS HC READY  
BNE REP  
BR IF HC READY  
HCOFF:  
MOV @HCST, - (SP)  
DISPLAY HCST  
MOV $581, - (SP)  
A191 MSG. HC OFF  
TST, TOT.  
ACTION MSG TO OPERATOR  
SP UNST  
CHECK HCOPY DEVICE AGAIN

HCOPY DEVICE INTERRUPT SERVICE ROUTINE

INTRY:  
BIT $10838, @SWR  
BR IF SY_12 UP  
DIU  
X Plat  
BEQ HXTLH  
BR IF NEW LINE  
SUB BPL2, @HCAD  
RESTORE HCAD  
MOVE@HCAD  
REPEAT THE LAST LINE  
HXTLH:  
BEQ ML  
DECREMENT LINE COUNT & TEST  
REPL:  
INC LINE  
ADD RMB, @HCAD  
BIT $7777, @HCAD  
BNE MOVE@HCAD
CALL GET, <A, LINE, INDEX> ; GET LOCATION OF LINE
MOV @A, @HCAUI ; AND LOAD INTO

ADD INDEX, @HCAUI ; HC BUS ADDR REG

SBR HCON, @HCAUC ; STORE NEGATIVE WORD COUNT
MOV @HCAUL, @HCAUC ; COMMAND LINE OUTPUT WITH PDATA=1, LXPND=0
RTI

EBIS: CLR @HCST ; STOP HCOPY DEVICE
CLR @BPWU ; LOWER PRIORITY
BIT #4, @BSWR
BEQ HCOK ; BR IF SW 2 DOWN (NO RERUN CHECK)
TSTB @PBUSY ; TEST PARAM BUSY BYTE
BEQ SCP ; BR IF NOT
MOV @SCP, PINT ; STORE INTERRUPT ADDRESS
RTI

SCP: CALL TYPE, <MSG, 2ERD>
CALL PARAM, <NP, PAR, MAXNP, SPBUF>
CMPB SPBUF = "Y"
BNE HCOK ; BR IF OUTPUT WAS OK
JMP HCOPY ; JPE-DO THE OUTPUT

HCOK: CLRB HCDS ; RELEASE THE HCOPY DATASET
DEC HCOPY

HCLED: CALL LED, <HCLED, HCOPY> ; ENTRY FOR RESTART
MOV R0, -(SP)
R1, -(SP)
MOV @NHDS, R0
MOV @HIDS, R1
MOV @MSGN, REG2

H40: TSTB (R1) ; BR IF DATASET IS FULL
BNE @H50 ;
ADD #2, @REG2
SBS P, @H40

CLRB HCOFLC ; SET HCOPY NO LONGER OPERATING
BIC #1, @HLITES ; TURN OFF HCOPY LITE
MOV (SP)+, R1
MOV (SP)+, R9
RTI ; RETURN FROM INTERRUPT

H50: MOV @RECI, A+SEN ; SET UP START BLOCK NUMBER
DEC R1
MOV @RHCDS, R1
MOV (SP)+, R1
MOV (SP)+, R0
JMP HCOPY ; START NEW DATASET

PAGE

LINTPH: WORD LINTPH
NP: WORD -
PAR: WORD BLKW 20
MAXNP: WORD 20
LHDU: WORD HDU
ZERO: WORD 0
ONE: WORD 1
F0UR: WORD 4
FIVE: WORD 5
TWELVE: WORD 12
TA: WORD -
A: BLKW MVBSIZ
SLAD: BLKW LADSIZ
SPAR=SLAD+2
#SPAR+2
3PL=SPAR+2
WDCNT=EPL
SPE=SPAR+4
LXPND: WORD -
RMB: WORD -
PL2: WORD -
LINE: WORD -
NFSDS: WORD -
CMD: WORD -
HCON: BLKW HCON
CHDS: WORD -
REG2: WORD -
YVAL: WORD 2080
YYVAL: WORD 2080.
END CALMS

.TITLE SEARCH - SEARCH PARTITION OF CALMS
MCALL CALL PAUSE
LF=12
CR=15
FTIME=15
PSV=17776
SYD=17776
MCCRO GETK3 MSG,?LOC
TSTG PBUSY
BEQ LOC
MOV @LOC.PINT+2
RTI
LOC: 
CALL TYPE, (MSG, ZERO)

ENDM

.MACRO ENH CODE, #EAPD
BIS CODE, #EAPD
ENDM

.MACRO DSI CODE, #EAPD
BIC CODE, #EAPD
ENDM

.MACRO LON CODE
BIC CODE, @LITES
ENDM

.MACRO LOFF CODE
BIC CODE, @LITES
ENDM

MCCRO HINT KEY
HANDLE INTERRUPT FOR KEYS 3-8 (AC.RJ.HX.LA)
DSI 109
LOFF 199000
MOV RA, -(SP)
CALL LED, (LKLED, KEY)

CURRENT ORIGIN IS 199000
JMP GNS
ENTRY POINT FOR GNS

SEARCH
MOV SRSTA, RA
INITIAL CALL FROM SUPERVISOR
MOV SRST, (RO)+
SET UP SEARCH START INTERRUPT VECT
MOV #298, (RO)+
PRIORITY 4
MOV #15, RI
IS20. MOV  #UINT, (R0)+  ; UNEXPECTED INTERRUPT RTE TO OTHERS
        MOV  #200, (R0)+  ; PRIORITY 4
        SUB  R1, IS20
        MOV  #FOC, #FOCA
        MOV  #KYR0, #KYR0A
        MOV  #200, #KYR0A+2
        CALL  MFST, <MACH, FP00>
        ENS  1140  ; ENABLE SRCH START, FOCUS AND KB3 REQUEST
        DSI  257  ; DISABLE OTHERS
        RTS  R5  ; RETURN TO SUPERVISOR
        UINT  ; PAUSE IEAP  ; UNEXPECTED INTERRUPT
        RTI  
        SRST  ; DSI  10
        . CALL  LED,(<KLED, SKEY>)
        GETKB  M1  ; GET KEYBOARD AND TYPE M1
        . CALL  PARAN, (NP, PAR, MAXNP, SPBUF)
        CMP  NP, #2
        BNE  SR20  ; BR IF NOT 2 WORDS
        CMP  PAR#A
        BNE  SR20  ; BR IF NOT ABORT
        . CALL  TYPE, M7  ; SEARCH ABORTED, TYPE M7
        ENS  110  
        SR20: INC  SFLAG  ; SET SEARCH OPERATING
        LOD  162599  ; RETURN OFF EDIT LIGHTS
        LON  1000  ; RETURN SEARCH LIGHT ON
        TST  NP
        BEQ  SR30  ; BR IF SAME PATIENT AND SLIDE
        . CALL  MVCR, (SPBUF, PANUM, LPNUM)  ; MOVE PATIENT NUMBER
        . CALL  TYPE, (M2, ZERO)
        MOV  R6, #0  ; INITIALIZE SEX TO BLANK
        . CALL  PARAN, (NP, MAXNP, SPBUF)
        MOV  PAR#SEX  ; STORE PATIENT SEX
        . CALL  TYPE, (M3, ZERO)
        . CALL  PARAN, (NP, MAXNP, SPBUF)
        . CALL  MVCR, (SPBUF, SLNUM, LSLNUM)  ; MOVE SLIDE NUMBER
        . CALL  TYPE, (M0, ZERO)
        . CALL  PARAN, (NP, MAXNP, SPBUF)
        MOV  PAR#SOURCE  ; STORE SOURCE CODE
        SR30: MOV  #SPB, CSPQ  ; INITIALIZE SEARCH QUEUE POINTER
        CLR  MSP  ; MSP=0
        MOV  R0, (SP)  ; SAVE R0 BEFORE CALLING ZIA
        . CALL  ZIA, (SPR, LSPR)  ; ZERO THE SPREAD RATINGS
        MOV  (SP)+, R0
        . CALL  LED, (<MLED, MSP>)
        . CALL  MFST, <MACH, FP00>
        MOV  XLCHR, XCTR  ; INITIALIZE SEARCH PATTERN
        MOV  IXDEL, XDEL
        MOV  #SBR, #SSBFA  ; SEARCH DATA READY INTERRUPT
        MOV  #SRHA, #SSRMA
        MOV  #SSRFA, #SSRRA
        MOV  #EDST, #EDSTA
        ENS  74  ; ENABLE INTERRUPTS
        RTI  
        FOC: . CALL  LED, (<KLED, FOKEY>)  ; 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 IEAPD AND SDFPA
        MOV  #SSBFA, #SSSFA
        . CALL  MFST, <MACH, FP00>  ; FIND FSTEP & FLAST FOR THIS MAGNIFICATION
        MOV  #F410, #SSDFA  ; SET UP INTERRUPT VECTOR
        ENS  4  
        RTI  
        F410: . CALL  CFOC, OLD  ; CALCULATE FOCUS
        . CALL  IFREL, FSTEP
        F420: MOV  #F430, #SSDFA
        RTI  
        F430: . CALL  CFOC, F
        CMP  F, #0LDF
        BNE  F450  ; SEE IF WE ARE GOING IN THE RIGHT DIR
        BCT  F450  ; BR IF SO
        MOV  FSTEP, N  ; CHANGE DIRECTION
        ASL  N
        NEG  N  ; MOVE TWICE AS FAR IN REVERSE
CALL IFREL, N
MOV $F440, $05DFPA
RTI

CALL CFCOF, F
CMP F. OLD F
BCT F. OLD F
CMPS F. STEP, FLAST
BLE F. OLD F
MOV F. STEP, N
ASR F. STEP
ADD F. STEP, N
NEW F. STEP = F. STEP + F. STEP / 2
CALL IFREL, N
BR F. OLD F

CALL CFCOF, F
CMP F. OLD F
BCT F. OLD F
MOV F. OLD F
CALL IFREL, F. STEP
NEW TAKE ANOTHER STEP
MOV $F440, $05DFPA
RTI

CALL CFCOF, F
CMP F. OLD F
BCT F. OLD F
MOV F. OLD F
CALL IFREL, F. STEP
NEW TAKE LAST STEP BACK
MOV $F440, $05DFPA
RTI

DSI 4
CALL RMC1
MOV $05DFPA, $05DFPA
NEW RESTORE SDPPA
BIT 44, QIEAPD
NEW SEE IF INTERRUPT WAS ENABLED
BEQ F. 508
BIT $0100, $0100
BEQ F. 498
NEW RE-ENABLE IT IF NOT IN SEARCH
TST $00SPRD
BPL F. 490
NEW OR SPREAD NOT PRESENT
BIT $4000, $4000
BEQ F. 490
NEW OR SW II DOWN
JMP 28000
NEW TURN ON SEARCH HALTED LIGHT.
BR F. 530
NEW DON'T RE-ENABLE THE INTERRUPT
EMI 4
NEW RE-ENABLE INTERRUPT
F508: EMI 480
NEW OFF
F199: EMI 480
NEW TURN OFF FOCUSING LIGHT
RTI

KEY9: DSI 1830
NEW DISABLE ANY FURTHER K80 REQ
CALL LED, $0KLED, $0RKEY)
GETKB M4
CALL PARAM, $0P, $PAR, $MAXNP, $SPBUF)
TST $0P
BEQ $0K7
NEW BR IF NO INPUT
CMPB $PAR, 'A'
BNE KY10
CALL APHASE, $0ZERO)
BR $0K7
KY10: CMPB $PAR, 'C'
BNE KY20
CALL IXREL, BIG
CALL IYREL, BIG
NEW MOVE TO LIMITS
CALL IFREL, BIG
CALL IWAIT
CALL YWAIT
CALL HFWAIT
NEW WAIT FOR END OF MOVES
CALL IFCPR
NEW INITIALIZE F.X, AND Y
CALL IYCR
CALL IYCR
CALL 1XYABS, $0XVAL, $0YVAL)
NEW MOVE STAGE TO 1000, 1000 OR LAST POSITION
CALL IFREL, FVL.
CALL XWAIT
CALL YWAIT
CALL FUART
CALL RMC

BR KYEX

KY20: CMPB PAR.9'R
BNE KY38 BR IF NOT RESTART REQUEST
INCB FFLAG
SET RESTART FLAG IN CALLS

KY30: CMPB PAR.9'E
BNE KY23 BR IF NOT EDIT REQUEST
INCB EFLAG
SET EDIT FLAG

KY35: CMPB PAR.9'F
BNE KY40 BR IF NOT FORTRAN REQUEST
INCB FFLAG
SET FORTRAN FLAG

BR KYEX

KY40: CMPB PAR.9'Z
BNE KY50 BR IF NOT ZERO SCAN QUEUE REQUEST
CLR NSCAN
CALL LED,<SQLEB, NSCAN>
MOV @SRST1, @SRSTA ALLOW SEARCH START

BR KYEX

KY50: CMPB PAR.9'Q
BNE KY56 BR IF NOT QUEUE ADJUST
MOV @NCPRA.XOFF
SUB XVALL.XOFF
STORE X OFFSET

MOV @YCVPR.YOFF
STORE Y OFFSET

BR KYEX

KY60: CMPB PAR.9'S
BNE KY70 BR IF NOT SCAN REQUEST

KY65: INC NSCAN

CALL LED,<SQLEB, NSCAN>
MOV @SPRCH, @SRSTA SET UP SPIRAL SEARCH

HP. #44 IF X AND Y WERE GIVEN
BNE KYEX BR IF NOT

CALL IXYABS,<PAR+4, PAR+6>
ENI 1090
JMP FOCUSUB

KY70: CMPB PAR.9'D
BNE KY80 BR IF NOT DEBUG REQUEST
MOV PAR+4, FPHASE
STORE PHASE NUMBER
BR KY65 G0 INCREMENT NSCAN

KY80: CMPB PAR.9'B
BNE KY90 BR IF NOT BAHDED REQUEST
MOV @D, SOURCE
SET SOURCE OF -8
BR KYEX

KY90: CMPB PAR.9'P
BNE KY100 BR IF NOT PATIENT REPORT REQUEST
INCB PFLAG
SET PREP FLAG

CALL UPARBA, <ONE, PAR, PHASE> WRITE PREP PARAM
BR KYEX

KY100:

 KYEX: ENI 1090
  RTI

 SDR: TST @SPRD SPREAD DATA READY (SEARCH)
  BNI SPRES BR IF SPREAD PRESENT
  MOV @L, FFLAG
  RESET FOCUS FLAG
  DEC FCNT
  DECREMENT FOCUS COUNT
  BPL CHP BR IF NO NEED TO FOCUS
  BIT @68988, @SPRD
  BR IF NOTHING TO FOCUS ON
  JNP FOCUSUB G0 FOCUS

  NPC OF:
  A5I FLAG
  JNP FOCUSUB BR IF NO FOCUS THIS TIME
  ART OF:
  0PC X (SP)
  BR IF X IS POSITIVE
  XYN MB GETKB M6 REQUEST CALIBRATION
  MOV (SP)+, R0
4,122,518

X01: MOV @YCRF,(R0)+ 
BMP .XNG 
CMP MAG. #63. 
BLT LONH 
NEG -(<<R0) 
LONH: MOV R8,CSPQ 
MOV (SP)+,R8 
INC HSP 
CALL LED,(SNLED,HSP) 
CMP HSP,HAXNSP 
BGE EDLO 
BLT G30 
NEG XDEL 
JMP ON, REVERSE DIRECTION 
CALL IYREL,(YDEL) 
AFTER MOVING DOWN 
BR G60 
MOV XLTGH,XCTR 
CALL IYREL,(YDEL) 
MOVE IN X DIRECTION 
G00: DEC XCTR 
BIS 04.04IEAPD 
ENABLE INTERRUPT 
RTI 
SRN1: DSI 4 
JSEARCH OR FOCUS HALT 
DTC 04.04IEAPD 
PREVENT END OF FOCUS FROM RESUMING SRCH 
CALL LED,(LKLED,SHAKEY) 
LONH 20399 
TURN ON SEARCH HALTED LIGHT 
CALL RNCI 
RTI 
SRRS: ENH 4 
JSEARCH OR FOCUS RESUME 
CALL LED,(LKLED,RKEY) 
LOFF 20399 
TURN OFF SEARCH HALTED LIGHT 
RTI 
EDST: DSI 104 
EDIT START 
LOFF 122999 
CALL LED,(LKLED,ESKEY) 
CALL NFST,(MAG,FPPOS) 
ENH 400 
ENABLE FOCUS INTERRUPT 
MOV $1:FFLAG 
INITIALIZE FOCUS FLAG 
BIT $1000,06LITES 
BNE EDLO 
BR TO EDIT-LO IF IN SEARCH 
BIT 0408.06LITES 
BNE ES40 
BR IF 9N EDIT-LO 
CMP MAG. #63. 
BLT ELSTT 
JMP EHRST 
HI-MAG,(IN EDIT-HI) 
ES40: CMP MAG. #63. 
JMP-EDIT-LO, TEST MAG. 
BLT ELRST 
JBR IF LO-MAG 
JMP EHRST 
HI-MAG, START EDIT-HI 
EDLO: CLR @CSPQ 
MARK END OF QUEUE 
DSI 20 
CLR NSPAL 
CMP MAG. #63. 
BLT ELSTT 
JBR IF LO MAG 
JMP EHRST 
START ED-HI IF HI MAG 
ELSST: .CALL LED,(SOLED,NSPAL) 
LOFF 1200 
LONH 400 
TURN ON EDIT LO LIGHT 
MOV $ELAC,00ACSPAP 
ACCEPT SPRD 
MOV $ELRJ,00RJSPAP 
REJECT SPRD 
MOV $ELNX,00NXSPAP 
NEXT SPRD 
MOV $ELLA,0ILASFLAST SPRD 
ELAST: CLR SP 
SPREAD NUMBER 
MOV $SP0,CSPQ. 
SAVE R0 ON THE STACK 
EDNHX: MOV CSPQ,P0 
LOFF 40000 
TURN OFF FOCUS LITE IN CASE FOCUS 
ENH 400 
WAS INTERRUPTED BY ARNL 
INC SP 
CALL LED,(SNLED,SPN)
67

LOFF 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, CPSQ
; SAVE UPDATED Q POINTER
MOV (SP)+, R0
; RESTORE R0
CALL IXYABS, (X,Y)
; INITIATE MOTOR MOVE TO X,Y
MOV #4ESDR, #4SFPA
MOV $1, FFLAG
ENI 4
; ENABLE SPRD DATA READY INTERRUPT
RTI

JEDHI: JMP EDHI

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

ES20: DSI 4
; NOW DISABLE THE SPRD INTERRUPT
CALL RMCI
; RELASE MCI FROM COMPUTER CONTROL
LON 100000
; TURN ON OPERATOR ACTION LIGHT
RTI

ELAC: HINT ACKEY
; ACCEPT SPRD INTERRUPT
MOV CPSQ, R0
TST -4(R0)
BPL EL220
; BR IF NOT PREVIOUSLY ACCEPTED
BEC NSPAL
; DON'T COUNT IT TWICE
EL220: MOV @6CPR, -(R0)
; STORE UPDATED Y AND X
MOV @7XCR, -(R0)
NEG @R8
INC NSPAL
CALL LED, (<SLED, NSPAL)
CMP NSPAL, NLMAX
BGE EDHI
; BR IF ENOUGH FOUND
BR EDHXT
; GO TO NEXT SPREAD

ELRJ: HINT RJKEY
; REJECT SPREAD INTERRUPT
MOV CPSQ, R0
TST -4(R0)
BPL JEDHXT
; BR IF NOT PREVIOUSLY ACCEPTED
BEC NSPAL
; MARK IT REJECTED X+
NEG -4(R0)
INC NSPAL
CALL LED, (<SLED, NSPAL)
BR JEDHXT

ELAK: HINT KSKEY
; NEXT SPREAD
BR JEDHXT

ELLA: HINT LAKEY
; LAST SPREAD
SUB #8, CPSQ
; MOVE BACK TO CURRENT SPREAD
BEC SPH
BEQ JEDHXT
; BR IF FIRST SPREAD IN QUEUE
SUB #8, CPSQ
; BACK UP ANOTHER SPREAD
JEDHXT JMP E2HXT

EDHI: MOV (SP)+, R0
; RESTORE R0
GETKB M5
; REQUEST HI MAGNIFICATION
EHSTT: LOFF 1400
LON 200
; TURN ON EDIT HI LIGHT
MOV #4EHAC, #0ACSPA
; SET UP EDIT HI INTERRUPTS
MOV #4EHJ, #4RJSPA
MOV #4EHX, #4HXSPA
MOV #4EHN, #4HSPA
MOV #4EHL, #4LASPA
MOV #4EHD, #4EDH
ENI 200
CLR NSCAN
CALL LED, (<SLED, NSCAN)

EHNXT: CLR SPH
MOV #4SPC, CPSQ
MOV R8, -(SP)
; SAVE R0
EHSTT: MOV CPSQ, R0
; TURN OFF ED, HI AND ACC, LIGHT
; DISABLE AHNL AND SDR
; DISABLE EDIT START
; SORT THE QUEUE ACCORDING TO RATING
; SAVP INFO FOR RESTART
; GET FIRST SPREAD TO SCAN
; ALLOW SEARCH START
; SFLAG

MOV  R8,-(SP) ; GET NEXT SPREAD TO SCAN SBSTE
MOV  CSPQ,R8
MOV  *SPSRCH,*4rstA
; ENABLE SPIRAL SEARCH
INC  SPN
; LOOK FOR FIRST -Y
MOV  (R8),XVAL
BEQ  GXEX
; BR IF END OF QUEUE
MOV  (R8),YV AL
BPL  EE10
; BR IF NOT ACCEPTED
NEG  XV AL
; MAKE XV AL AND YVAL POSITIVE
ER  YVAL
ADD  OFFSET
ADD  YOFF,YVAL
MOV  R8,CSPQ
CALL  VARS,(XV AL,YVAL) ; INITIATE MOTOR MOVE
MOV  *#PSR.-(SP)
JSP  PC,FOCUSUB ; START FOCUSING

MOV  (SP)+,R8
RTI  25

CALL  LED,( sistem,SPN)

CLR  XCR
; INITIALIZE SPIRAL SEARCH
CLR  YCR
CLR  SPLM
MOV  #SPSBR,#SDFPA
MOV  #SRH#SRHAA
MOV  #SRR,#SRRSA

; ENABLE SEARCH HALT & RESUME & SDR
RTI  34

; SEE IF ANYTHING THERE
BEQ  SGNP
; BR IF NOT
NEG  FFLAG
BPL  SGNP
; BR IF NO NEED TO FOCUS

SGNP  FOCUSB
TST  XCR
BGT  SOK
; BR IF ANOTHER X MOVE IS OK
TST  YCR
BGT  SOK
; BR IF ANOTHER Y MOVE IS OK
INC  SPLEH
; INCREMENT SPIRAL LENGTH
MOV  SPLM,XCR
; RELOAD X AND Y COUNTERS
MOV  SPLM,YCR
NEG  SPST.EP
; CHANGE DIRECTION

SOK  CALL  IRREL,SPSTEP
DEC  XCR
RTI  184

SOK  CALL  IRREL,SPSTEP
DEC  YCR
RTI

SPLEH  : WORD -
; SPIRAL LENGTH
SPSTEP  : WORD 5
; OR -5
YCR  : WORD -
; YCR
YOFF  : WORD -
; YOFF
CALL DCLNEA

DCLNEA: JSR R5, DWAIT
INC @4GSST ;STORE CLEAR COMMAND
RTS R5

CALL DCLNE LOC, Y, X; NS, REPL, ERASE - 1*ERASE ONLY

DCLNE: JSR R4, SAVER
JSR R5, DWAIT
MOV @4(R5), 0DYREG ;STARTING LINE ON DISPLAY
MOV #URCMD.RI
BIS #409, 0DRST ;ASSUME WRITE, SET CYCLE BIT
TST 014(R5)
B EQ NOER ;BR IF NO ERASE
MOV #UCMD.R1 ;ASSUME ERASE/WRITE
TST 014(R5)
BPL NOER ;BR IF ERASE/WRITE
MOV #ERCMD.R1 ;ERASE ONLY
BIC #400, 0DRST ;CLEAR CYCLE BIT
NOER: MOV 016(R5), R3
LSR MOV 012(R5), R2 ;REPL
B EQ NOREP ;BR IF NO REPLICATION
BIS #DRST.RI ;INSERT REPLICATE BIT
ASL R3 ;1*NS
NOREP: MEG R3
MOV R3, 0*GSFC ;STORE PIXEL COUNT -H OR -2*N
REP: MOV 2*(R5), #DBPA ;LOC TO BUS ADDRESS REGISTER
MOV 06(R5), #XREG ;STARTING SAMPLE ON DISPLAY
MOV 016(R5), R3 ;SAMPLE COUNT
INC R3
ASR R3
MEG R3
MOV R3, 0*DRWC ;STORE NEGATIVE WORD COUNT
MOV R1, 0*GSST ;STORE WRITE OR ERASE/WRITE COMMAND
INC #DRST
TJSQ US GO
B LE EXIT ;BX IF DONE
BIT #10, 0*SWR ;REPLICATE UNLESS SW 3 IS UP
BNE EXIT ;EXIT IF SW 3 UP
CLR R2
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, SAVER
MOV #3, R3
SC10: MOV #32, X
MOV #32, Y ;WRITE A MARK AT 32,32
CALL DCLNE (ZERO, Y, X; ONE, ONE, ZERO)
MOV 09, R4
SC20: ADD #2, Y
CALL DCLNE (ZERO, Y, X; ONE, ONE, ZERO)
SDB R4, SC20
SDB R3, SC10 ;REPEAT TO MAKE IT WHITE ENOUGH
MOV #3, R3
SC30: MOV #973, Y
MOV #992, X ;WRITE A MARK AT 992,992
MOV #9, R4
SC40: CALL DCLNE (ZERO, Y, X; ONE, ONE, ZERO)
ADD #2, Y
SDB R4, SC40
MOV #973, X
CALL DLINE (ZERO, Y, X; ONE, ONE, ZERO)
SDB R3, SC10
CALL TYPE (M3.ZERO) ;TELL OPERATOR TO MOVE CURSOR
SC50: CALL PARAM (MP, PAR, FIVE)
JSR R5, ULDR ;CHECK FOR REQUESTED CURSOR MOVE
BR SC50 ;CURSOR WAS MOVED
MOV R6, SYCP, Y2 ;STORE Y ZERO ADJUSTMENT
MOV 0 @GSXCP,XZ ;STORE X ZERO ADJUSTMENT

SC60 CALL TYPE, <R4,ZERO> ;MOVE TO LOWER RIGHT

JMP R5, U5D
BR SC60 ;CURSOR WAS MOVED

MDIV 0 @GSXCP,YD ;STORE Y BELTA

MDIV XZ,XD

BIC 0 @176000,YZ

BIC 0 @176000,XZ

CALL UPPARM, <FOUR,YZ,INTPH> ;SAVE THE ADJUSTMENTS ON DISK

JMP REST

PDA CALL PPARM, <HP,YZ,FOUR,INTPH> ;READ ADJUSTMENTS FROM DISK

RTS R5

CALL CURSOR(YCP,XCP,LIN,SAMPL)

CURSOR ;JSR R4,SAVER

MOV 0 @GSXCP,R0

BIC 0 @176000,R0

SUB 0 @9600,R0

;ADJUST THE READING

DIV 0 @YD,X0

ADD 0 @32,R0

ADD 0 @AY+(Y-YZ)*960/YD+32

ASL 0 R1

CMP 0 R1,YD

BLT C28

INC 0 R0

;ROUND UP

C28 MOV 0 @GSXCP,P2

BIC 0 @176000,P2

SUB 0 @XZ,P2

;ADJUST THE X READING

MUL 0 @960,P2

DIV 0 @XZ,P2

ADD 0 @32,R2

ASL 0 R3

CMP 0 R3,XD

BLT C40

INC 0 @P2

;ROUND UP

C40 MOV 0 R.02(R5) ;STORE YCP

MOV 0 R.04(R5) ;STORE XCP

BIT 0 @4,R5

BEQ CEX ;JBR IF NOT 4 PARAMETERS

ASR 0 R8

INC 0 R9

MOV 0 R.06(R5) ;LINE=YCP/2+1

ASR 0 R2

INC 0 R2

MOV 0 R.010(R5) ;SAMP=XCP/2+1

CEX JMP RESTR

MCU ;SUB 0 @2,00GSXCP ;CALL MCU TO MOVE CURSOR UP

MCB ;ADD 0 @2,00GSXCP ;CALL MCB TO MOVE CURSOR DOWN

MCL ;SUB 0 @2,00GSXCP ;CALL MCL TO MOVE CURSOR LEFT

MCR ;ADD 0 @2,00GSXCP ;CALL MCR TO MOVE CURSOR RIGHT

UBLR ;MOV 0 @2(R5),PAR ;CALL UBLR(PAR)...EXTERNAL CALL

UBLR BR ULR0

UBLR TST 0 HP ;CHECK FOR U.L.R. OR A TYPED

BNE ULR0 ;JBR IF NO TYPEIN RETURN

TST 0 @R5

RTS R5

ULR0 CMPB 0 @U'U

BNE UL30 ;MOVE CURSOR UP ONE STEP

DEC 0 @GSXCP

CMPB 0 @R'L

BNE UL40 ;MOVE CURSOR DOWN ONE STEP

INC 0 @GSXCP

CMPB 0 @PAR'L

BNE UL50
79
4,122,518

UL50: DEC @9GSXCP MOVE CURSOR LEFT ONE STEP
      CMPB PAR, J'R
      BNE UL60
UL60: INC @9GSXCP MOVE CURSOR RIGHT ONE STEP
      RTS R5 ;TYPEIN RETURN

ZERO, .WORD 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

X=PAR+2
Y=PAR+4
NLIST BEX

M3, .ASCIZ / MOVE CURSOR TO UPPER LEFT MARK/
M4, .ASCIZ / MOVE CURSOR TO LOWER RIGHT MARK/

Y, .WORD 32
ZD, .WORD 960.
XD, .WORD 960.

BLKB DWAIT+1500.

END

.TITLE MCIUSER - MCI SUBROUTINES
CALMS VERSION

.MCALL CALL, PAUSE

XCPR==164044
YCPR==164044
FCPR==164044
MSC==164046
IEHPD==164055
SPDP==164074
FCOP==164076
LITER==164134
LEDS==164136
SRSTAIR==308
SPRAI==310
EDSTAIR==314
ASPA==322
RJSPA==324
MAXSPA==334
LASTSPA==334
EDEND==348
FOCO==344
FOCUS==350
KYPQA==374
SPRD/FOCUS DATA READY INTERRUPT ADDR

:CURRENT ORIGIN IS 59590

CHCI: BIS #48099.0@MSC ;CAPTURE MCI
      RTS R5

RNCHI: BIC #48099.0@MSC ;RELEASE MCI
      RTS R5

XWAIT: BIT #48000.0@MSC
      BNE XWAIT
      RTS R5

YWAIT: BIT #49000.0@MSC
      BNE YWAIT
      RTS R5

FWAIT: BIT #10.0@MSC
      BNE FWAIT
      RTS R5

IXYABS: CALL XWAIT ;CALL IXYABS(X,Y) TO INITIATE ABS MOVE
      CALL YWAIT
      CALL CHCI
      M0T #82.35.X
      DPL IXY20
NEG \$X

MAKE X POSITIVE

MOV \$4(R5), \$Y

DPL IXY30

MAKE Y POSITIVE

IXY30:

SUB \$X, 0; XCOPY

INITIATE MOVE TO X, Y

SUB \$Y, 0; YCOPY

RTS R5

IXREL:

CALL \$WAIT

CALL CMC1

MOV \$2(R5), 0; XCOPY

RTS R5

IYREL:

CALL YWAIT

CALL CMC1

MOV \$2(R5), 0; YCOPY

RTS R5

IFREL:

CALL FUWAIT

CALL CMC1

MOV \$2(R5), 0; FCOPY

RTS R5

CFOC:

CALL CFOC(\$) TO CALCULATE FOCUS

CLR \$2(R5)

MOVB TEMP+1, \$2(R5)

ADD TEMP, \$2(R5)

FOCUS=\$1+\$2

IXCOPY:

BIS \$400, \$0; HCSC

INITIALIZE \$COPY

RTS R5

IYCOPY:

BIS \$200, \$0; MSC

RTS R5

IFCOPY:

BIS 11, \$0; MSC

RTS R5

LED:

JSR R4, SAVER

CALL LEDX(CODE, VAL)

MOV \$2(R5), R0

CODE TO R0

ASH \$13, R0

SHIFT TO HI ORDER 3 BITS

MOV \$4(R5), R1

VALUE TO R1

BIC \$177800, R1

ADD R1, R0

MOV R8, 0; LEDS

OUTPUT DISPLAY

JMP RESTR

MFST:

JSR R4, SAVER

CALL MFST(MAGH, FPOS)

MOV \$1EAPB, R0

FLTR/MAGH POS REG TO R0

MOV R8, R1

ALSO R1

ASH \$13, R0

\$177776, R0

MASK OUT MAG

MOVB LEHS(R0), MAGH

STORE MAGH

CLR FSTEP

CLEAR TOPBYE

MOVB FST3(R0), FSTEP

STORE FOCUS STEP SIZE

MOVB FLTA8(R0), FLAST

STORE LAST STEP SIZE

MOVB \$X=(R0), \$R2

ADJUSTMENT FOR NON-CENTRALITY

MOV \$R2, \$XADJ

MOV \$Y=(R0), \$R2

ADJUSTMENTS ARE FOR 63 AND 100

MOV \$R2, \$YADJ

RELATIVE TO 40

MOV MAGH, \$2(R5)

ASH \$13, R1

\$177776, R1

SHIFT FILTER CODE

BIC \$177776, R1

MOV R1, \$4(R5)

MOV R1, FPOS

CALL LEDG(OMLEDMAG), DISPLAY MAGNIFICATION

JMP RESTR

SORTQ:

JSR R4, SAVER

SUBROUTINE TO SORT THE QUEUE

MOV \$SPR, R3

R3 AND R4 POINT TO LOW QUEUE POSITIONS

MOV \$SPR, R4

THE CURRENT HI RATING IS MOVED HERE

SQ23:

CLR R5

INITIALIZE 0 SPREADS REMAINING

MOV R3, R1

USE R1 TO EXAMINE REMAINING SPRDS

MOV HSR, R2

BR IF NO SPREADS REMAIN

SQ30:

CMPB \(R1)+, R0

EXAMINE A RATING

BIE \(R0)

BR IF NOT GREATER THAN MAX

MOV \(C1)+, R0

STORE NEW MAX

MOV R1, REG1

SAVE R1
SQ40: INC R1
SUB R2, S030 \#BR IF MORE TO EXAMINE
DEC HSR
TST R0
BEQ SQ4X \#BR IF ALL HAVE BEEN SORTED
MOV REG1, R1
MOV R1, R2 \#R2 WILL HAVE POSITION IN SQ0
SUB 1, R2
EQU *SPR.R2
ASH #2, R2
ADD *SPR.R2
MOV R0, R0
MOV OR3.R0 \#SWITCH RATINGS
MOV XOR3.R0
MOV R0, (R3)\*+1
MOV OR2.R0 \#SWITCH X AND Y VALUES
MOV OR3,R0
MOV OR4.R0 \#R3 AND R4 ARE UPDATED
BR SQ20 \#GO GET THE NEXT HIGHEST
S03X: CLR OR4 \#MARK END OF QUEUE
CLR OR2 \#MARK END OF RATINGS
JMP RESTR

MAGN: .WORD 0
FP0S: .WORD 0
LN0S: .BYTE 0.49, 64, 130, 181, 4.63, 0
FSTAB: .BYTE 1.15, 16, 0, 0, 25, 0, 1
FLTAB: .BYTE 18.5, 2.2, 15.4, 1
XATAB: .BYTE 0.0, 5.0, 8.9, 0 \#XADJ FOR 63 AND 106 REL TO 40
YATAB: .BYTE 0.0, 0.3, 0, 0, 2.0 \#YADJ
XADJ: \#WORD -
YADJ: \#WORD -
FLAST: \#WORD -
X: \#WORD -
Y: \#WORD -
SNLED: \#WORD 0 \#CODE FOR SPREAD NUMBER DISPLAY
SQLED: \#WORD 1 \#SCAN QUEUE
AILED: \#WORD 2 \#ANALYSIS QUEUE
OMLED: \#WORD 3 \#OBJ MGNH
HOLED: \#WORD 4 \#HARD COPY QUEUE
LKLED: \#WORD 5 \#LAST KEY
TEMP: \#WORD 0 \#LAST TEMP.
HNR: \#WORD -
REG1: \#WORD -
BLKB: CM11388 \#MAKE LENGTH CONSTANT

TITLE PARAM READ AND WRITE PARAMETERS UNDER CALMS
.MCALL CALL BUF12=60

APHASE: MOV OR2(R5), RPHASE \#ASSIGN NEXT PHASE CALL APHASE(H)
RTS R5
UPH0M: JSR R4, SAVER \#CALL UPH0M(NP, PAR.PNUM)
MOV OR2(R5), R0 \#HPR
MOV 4(R5), WP29+12 \#PAR LOCATION
MOV OR6(R5), R1 \#PNUM
MOV R1, PREC
ASL R1
ADD OR3 R1
UP29: CALL WRITE, (PNUM, PREC,...) \#WRITE THE PARAMETER DATASET
JMP RESTR

RPARAM: JSR R4, SAVER \#CALL RPARAM(NP, HPRMAXI, PHNUM)
MOV PHNUM, R0
BIT R4, OR5
BEQ RPIO \#BR IF NOT 4 PARAMETERS
MOV OR8(R5), R0 \#USE PHNUM FROM PARAMETER LIST
RPIO: MOV R0, PREC
ASL R0
ADD OR2 R0
; CALLS VERSION, CALL SCAN(DUNIT, FILEX)
; THIS VERSION USES MVIO IN DOUBLE BUFFERING 24
; VERTICAL LINES AT A TIME FROM CAMERA TO A DISK DATA SET
; IN NON- INTERRUPT MODE.
; THE PICTURE IS DISPLAYED ON THE GRAY SCALE.
; SECTOR_HISTOGRAMS ARE GENERATED AS THE PIXELS ARE SCANNED.
; THRESHOLDS ARE FOUND AFTER EACH GROUP OF SECTORS AND TYPED
; OUT WHILE THE NEXT ROW OF SECTORS IS SCANNED IF SYST IS UP.
; THE Scan AREA IS VARYING WITH A MAxIMUM OF 512 LINES AND
; 1480 SAMPLES PER LINE.

GLOBAL OPEN, GET, PUT, CLOSE, PARAM, OPINTER, PLABEL, PARSUB
GLOBAL AFFILE, SCAN
MCALL PARAM, EXIT, CYTBT, CALL, PAUSE
PARAM
HLST, BEX

LF=12
CR=15
COMD=164B20
STAT=164B20
XREG=164B22
YREG=164B24
ZREG=164B26
XJREG=164B30
YJREG=164B32

; JOYSTICK REGISTER

SCAN:
MOV R5, -(SP) ; SAVE R5
MOV 2(R5), $189+10 ; CALL SCAN(DUNIT, FILEX) STORE DUNIT
MOV 4(R5), $189+12 ; STORE CALMS DATA SET NAME
MOV $2.UIC ; AND UIC OF (2, 2)
S180A:
CALL AFFILE, BUF, -...,, UIC, UIC ; ASSIGN DEFAULT FILE
CALL OPEN, BUF, SIZE, ONE, ONE, NAME ; OPEN DISK DATA SET
S182:
CALL TYPE, (M.I.ZERO) ;
TFCI:
BIT #400, #1EAPP ; TEST IF STILL FOCUSING
BEG TFCI ; IF SO
; CALL RCI ; RELEASE RC TO OPERATOR
; CALL PARAM, (HPAR, PARA, HMAX) ; READ LABEL
MOV SOURCE, L+12 ; MOVE INFO INTO LABEL
MOV PANUM, L+14
MOV SLNUM, L+16
MOV $3XCPR, L+18
MOV $1YCPK, L+20
CALL OUTCON, (SOURCE, L+81, TWO)
CALL MVL, (PANUM, L+92, LPANUM) ; MOVE PANUM INTO LABEL
MOV BAPEN, L+83
CALL MVL, (SLNUM, L+112, LSNUM)
TST HPAR
BEG S180 ; BR IF NO PARAMETERS (USE OLD LABEL)
MPAR, #2
BNE S184 ; BR IF NOT TWO PARAMETER WORDS
MPAR, #2
BNE S186 ; BR IF NOT PRINT Q REQUEST
CALL PRINTQ, (L+72, SP, SPQ)
JMP S102
CMP #A, PAR BR IF NOT ABORT
BNE NOTAB
CALL APHASE, ZERO
JMP S300
NOTAB:
CMP #Z, PAR BR IF NOT ZERO QUEUE REQUEST
BNE S103
CLR NSCAN IZERO NSCAN
CALL APHASE, ZERO AND ABORT THIS SCAN
JMP SEXIT
S103:
CMP #EX, PAR BR IF EX WAS NOT TYPED
BNE S104
SCAN IF EX WAS NOT TYPED
JMP SEXIT
S104:
CMP #DS, PAR BR IF NOT DUMMY SCAN
BNE S105
INC DFLAG SET DFLAG
OR S175
S105:
MOV #PAR3UF, R1 ADD ANOTHER LABEL WITH MISC ID
MOV #L+217, R2 ADD ANOTHER LABEL WITH MISC ID
MOV #69, R3 MAX OF 69 CHARACTERS IN LABEL
CALAB:
MOV R1+2, R0 ADD ANOTHER LABEL WITH BLANKS
BNE ELAB BR IF END OF LABEL (CR)
MOV R0, (R2)+ STORE LABEL CHARACTER
SUB R3, HLAB ADD REST OF LABEL WITH BLANKS
BR S106
MOV #49, (R2)+ PAD REST OF LABEL WITH BLANKS
SUB R3, ELAB
JSR PC.PPAR PROCESS PARAMETERS
CALL OUTCON.<XCPR.L+127..FOUR>
CALL OUTCON.<YCPY.L+135..FOUR>
CALL MVL.<L+124..M+22..FOUR>
CALL MVL.<L+112..M+22..FOUR>
CALL TYPE.M2
CVTDT #.6L+145. STORE DATE IN LABEL
CVTDT #.6L+135. STORE TIME IN LABEL
TST DFLAG BNE S108 NO OUTPUT LABEL IF DUMMY SCAN
CALL PLABEL.<BUF, SPAR.L> OUTPUT LABEL
CALL DCLEAR CLEAR THE GRAY SCALE
CALL DCLEAR CLEAR THE GRAY SCALE
S118:
MOV SL.X INITIAL XCOUNT
CLR #COND DISABLE PIXEL INTERRUPT
CLR CY CLEAR GS LINE COUNTER
MOV #ITAB, TX MOVE PIXEL INTO BUFFER
MOV #ITAB, BX SET PRIORITY 6 FOR THE REST OF THE FIELD
JSR PC.PHEAD TYPE HEADING
LSEC:
CLR SHUM SECTOR NUMBER=0
CALL ZIA. (HIST.MBIN) IZERO HISTOGRAM COUNTS
EFLIN:
MOV SS.Y INITIAL Y
MOV #0X:REG SET LINE COUNT
INC LINE INCREMENT MVIO LINE COUNT
TST DFLAG BNE S115 BR IF DUMMY SCAN
CALL PUT.<BUF.LINE, INDEX> PREPARE STORAGE OF NEXT TV LINE
S115:
MOV BUF.R3 UPDATE BUFFER POINTER TO BUF+HS-1
0FLIN:
ADD INDEX.R3 SET POINTER FOR NEXT LINE
MOV #HIST.R4 R4 HAS LOC OF HIST
MOV ISPS.R3 INITIAL SAMPLES PER SECTOR
MOV H.0X:REG SET PIXEL COUNT
MOV #CDMB.R2 (COND-STATUS REGISTER) TO R2
MOV #3, OR2 GET FIRST PIXEL OF THIS FIELD
TSTB OR2 BPL -2 WAIT UNTIL READY
MOV #389, 0XPSW SET PRIORITY 6 FOR THE REST OF THE FIELD
BR NOPB
SECTOR:
MOV ISPS.R3 R3=SAMPLES PER SECTOR
PIXEL:
MOV #3, OR2 PIXEL COMMAND
TSTB OR2
SP -2 WAIT UNTIL READY
NOPB:
MOV #8, OR2 MOVE PIXEL INTO BUFFER
BNE NOTZE BR IF NOT ZERO
INM

BNE NOT127

MOV $126, R1

; CHANGE 127 TO 126

INC R1

; INC COUNTER TO THIS VALUE

ADD R4, R1

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

INC R1

; INC COUNTER TO THIS VALUE

SUB $2, R0

; DECREMENT POINTER

SBD R3, PIXEL

; ABK IF MORE PIXELS IN THIS SECTOR

HIST $128, R4

; STEP HISTOGRAM POINTERS

HTEST: CMP R4, #HIST+1024

; HIST+HC*120

BLO SECTOR

; ABK IF MORE SECTORS

BIT $10, Y

; TEST FOR EVEN OR ODD FIELD

BCT OFEND

; ABK 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

; ABK IF FIRST LINE

EFMOV:

CALL DLINE

<,-. , CY,GX,NS,REPL,ZERO

; REPLICATE PREVIOUS LINE

INC CY

; FOR QUICK LOOK, WRITE EVERY OTHER LINE

EFMOV:

MOV BUF, R0

DEC R0

; SET BUFFER POINTER TO BUF+HS-2

JMP OFLIN

; GO GET ODD FIELD PIXELS

OFEND:

CLR @PSW

; RETURN TO PRIORITY 0 AT END OF FIELD

ADD $2, R0

MOV R0, OFCALL+6

; STORE START OF CURRENT LINE

MOV R0, EFCALL+6

; SIM BOTH CALLS TO DLINE

OFCALL:

CALL DLINE

<,-. , CY,GX,NS,REPL,ZERO

; WRITE TO GRAY SCALE

INC CY

; IF NOT QUICK LOOK

BEQ $5

; STEP CY AGAIN

INC X

; INCREMENT X (LINE.COUNT)

DEC SCY

; FOR QUICK LOOK, WRITE EVERY OTHER LINE

JSR PC.TRESH

; CALC THRESHOLDS FOR THESE SECTORS

CMP X, EL

; FOR CURRENT LINE

JMP FIN

; ABK IF FINISHED

LSEC: JMP EFLIN

; GO START NEXT LINE SECTOR

NOS:

CMP X, EL

; TEST FOR END OF PICTURE

BLT JFLIN

; GO START NEXT LINE OF PICTURES

JSR PC.TRESH

; CALC THRESH FOR LAST SECTORS

FIN:

MOV #TDATA, R1

TST DFLAG

BNE $202

CPUT:

CALL CPUT (BUF, ZERO, INDEX)

MOV #BUF, R2

ADD INDEX, B2

MOV HS, R3

MTD: MOV # (R1)+, (R2)+

; MOVE TDATA TO BUF

SBD R3, MTD

CMP R1, #TDATA+66

LSEC: JMP EFLIN

; ABK IF END OF SECTOR

BNE #202

SEXT: CALL CLOSE.BUF

RTS: MOV (SP)+, R5

; RESTORE R5

RTS R5

; RETURN TO SUPERVISOR ***************

PPAR: CALL TYPE, (M3.ZERO)

; JSR PC.PPAR

CALL PARAMETERS (NPAR, PAR, NPMAX)

MOV #PAR, R1

CLEAR HR

CLEAR HC

P10: TST NPAR

BGT P15

; BR IF MORE PARAMETERS

JMP P15

MOV (R1)+, R0

; R0=PARAMETER WORD

MOV #NX, R2

MOV #KEY, R3

P20: CMP R0, (R3)+
4,122,518

DEG P50:  ABR IF PARAMETER EQ KEYWORD
LLR P50.P20: ABR IF MORE KEYWORDS
CMP HPAR, #4: NOT A KEYWORD, MAYBE AREA IS SPECIFIED
BCE P30: ABR IF 4 WORDS LEFT

P25: .CALL TYPE, M4: PARAMETER ERROR
BR PPAR
P30: CMP R0, #MAXNL
BCE P25: ABR IF SL TOO LARGE
MOV R0, SL
BLT P25: ABR IF SL TOO SMALL
MOV (R1)+, R0: SEL
BLE P25: ABR IF EL TOO SMALL
CMP R0, #MAXNL
BGT P25: ABR IF EL TOO LARGE
MOV R0, EL
SUB SL, R0
BLE P25: ABR IF NL TOO SMALL
MOV R0, NL
MOV (R1)+, R0
BLT P25: ABR IF SS TOO SMALL
CMP R0, #MAXNS
BCE P25: ABR IF SS TOO LARGE
BIC $1, R0
BSAVE SS
DEC R0
MOV R0, BUF
SUB $4, HPAR

P50: BR P10
TST (R1)+
ADD #MK+MK-2, R3
MOV P23, PC
GO THERE

P60: MOV (R1)+, DPER
MOV #100, DPER
SUB DPER, DPER
MOV DPER, DPER
SUB #3, HPAR
INC HPAR

P70: MOV R1, DBEG
MOV (R1)+, BKSIZE
ASP BKSIZE
BEG BKSIZE
MOV #62, #BSIZE
SUB BKSIZE, #BSIZE
SUB BKSIZE, #BSIZE
DR P63
P80: MOV (R1)+, NR
BR P63
P90: MOV (R1)+, NC
BR P63
P95: INC QFLAG
INC #2, HPAR

P100: MOV #YJREG, HPAR
MOV #XJREG, JPAR
MOV $177000, JPAR
CALL TYPE, (H5, ZERO): REQUEST MOVE TO LOWER RIGHT
CALL PARAM, (HPAR, PAR, HPMAX)
MOV #YJREG, JPAR+6
MOV #XJREG, JPAR+6
BIC $177000, JPAR+2
MOV #JPAR, R1
MOV (R1)+, R0
MOV #4, HPAR
<table>
<thead>
<tr>
<th>JMP</th>
<th>P30</th>
<th>(1)GO PROCESS AREA PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P159: TST</td>
<td>HR</td>
<td>IDONE WITH PARAMETERS</td>
</tr>
<tr>
<td>BGT</td>
<td>P160</td>
<td>IBR IF HR WAS A PARAMETER</td>
</tr>
<tr>
<td>MOV</td>
<td>HL, R0</td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>#32, R0</td>
<td></td>
</tr>
<tr>
<td>ASH</td>
<td>4-6, R0</td>
<td></td>
</tr>
<tr>
<td>BGT</td>
<td>P155</td>
<td>IBR IF HL GT 31</td>
</tr>
<tr>
<td>P155: INC</td>
<td>R0</td>
<td>INR=1</td>
</tr>
<tr>
<td>MOV</td>
<td>R0, HR</td>
<td>INR=(HL+32)/64</td>
</tr>
<tr>
<td>TST</td>
<td>NC</td>
<td></td>
</tr>
<tr>
<td>BGT</td>
<td>P170</td>
<td>IBR IF NC WAS A PARAMETER</td>
</tr>
<tr>
<td>MOV</td>
<td>NS, R0</td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>#32, R0</td>
<td></td>
</tr>
<tr>
<td>ASH</td>
<td>4-6, R0</td>
<td></td>
</tr>
<tr>
<td>BGT</td>
<td>P163</td>
<td>IBR IF NS GT 31</td>
</tr>
<tr>
<td>INC</td>
<td>R0</td>
<td>IN=1</td>
</tr>
<tr>
<td>MOV</td>
<td>R0, NC</td>
<td>INC=(NS+32)/64</td>
</tr>
<tr>
<td>ADD</td>
<td>HR, R1</td>
<td></td>
</tr>
<tr>
<td>DEC</td>
<td>R1</td>
<td></td>
</tr>
<tr>
<td>CLR</td>
<td>R0</td>
<td></td>
</tr>
<tr>
<td>DIV</td>
<td>HR, R0</td>
<td></td>
</tr>
<tr>
<td>INC</td>
<td>R0</td>
<td></td>
</tr>
<tr>
<td>BIC</td>
<td>#1, R0</td>
<td>IMAKE LPS EVEN</td>
</tr>
<tr>
<td>MOV</td>
<td>R0, LPS</td>
<td>ILPS=(NL+HR-1)/NR</td>
</tr>
<tr>
<td>MOV</td>
<td>NS, R1</td>
<td></td>
</tr>
<tr>
<td>CLR</td>
<td>R0</td>
<td></td>
</tr>
<tr>
<td>DIV</td>
<td>NC, R0</td>
<td></td>
</tr>
<tr>
<td>BIC</td>
<td>#1, R0</td>
<td>IMAKE IT EVEN</td>
</tr>
<tr>
<td>MOV</td>
<td>R0, SPS2</td>
<td>ISPS2=NS/NC SPS FOR 2 FIELDS</td>
</tr>
<tr>
<td>ASR</td>
<td>R0</td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>R0, SPS</td>
<td>ISPS FOR ONE FIELD</td>
</tr>
<tr>
<td>MOV</td>
<td>NC, R1</td>
<td></td>
</tr>
<tr>
<td>DEC</td>
<td>R1</td>
<td></td>
</tr>
<tr>
<td>MUL</td>
<td>SPS, R1</td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>NS, R0</td>
<td></td>
</tr>
<tr>
<td>ASR</td>
<td>R0</td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>R1, R0</td>
<td>ISPS=NS/2-(NC-1)*SPS</td>
</tr>
<tr>
<td>MOV</td>
<td>HC, R1</td>
<td></td>
</tr>
<tr>
<td>ASH</td>
<td>#7, R1</td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>#HIST, R1</td>
<td>#HIST+MC+128</td>
</tr>
<tr>
<td>MOV</td>
<td>R1, #TEST+2</td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>#128, R1</td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>R1, #J35+2</td>
<td></td>
</tr>
<tr>
<td>RTS</td>
<td>PC</td>
<td></td>
</tr>
</tbody>
</table>

\( \text{THRESH} \)

\begin{align*}
\text{MOV} & \quad \#MSG+3, #150+10 \\
\text{MOV} & \quad \#MSG+6, #160+10 \\
\text{T05.} & \quad \text{MOV} \quad \#HIST+895, \ldots, .24 \quad \text{HIST}+(NC-1)*128 \\
\text{T10.} & \quad \text{JSR} \quad \text{PC, TSUB} \quad \text{FIND BACK AND DATA} \\
\text{MOV} & \quad \text{DPER, R1} \\
\text{MUL} & \quad \text{DATA, R1} \\
\text{MOV} & \quad \text{DPER, R3} \\
\text{MUL} & \quad \text{BACK, R3} \\
\text{ADD} & \quad \#50, .R1 \\
\text{CLR} & \quad R0 \\
\text{DIV} & \quad #108, .R0 \quad \text{JTHR}=(\text{DPER}+\text{DATA}+(100-\text{DPER}) \times \text{BACK}+50) \times 100 \\
\text{MOV} & \quad \text{DATA, R1} \\
\text{SUB} & \quad \text{BACK, R1} \\
\text{CMP} & \quad R1, #88 \quad \text{CHECK FOR SMALL DIFFERENCE BETWEEN D AND B} \\
\text{BGT} & \quad \text{T20} \quad \text{IBR IF OK} \\
\text{T20.} & \quad \text{MOV} \quad \text{BSEGZ, RO} \quad \text{JTHR}=(\text{DBEG}/2 \quad \text{UNDIVIDED NUCLEUS}) \\
\text{T30.} & \quad \text{TST} \quad \text{DATA} \\
\text{BGT} & \quad \text{T30} \quad \text{IBR IF NO DATA IN SECTOR} \\
\text{T30.} & \quad \text{TST} \quad \text{BACK} \\
\text{BGT} & \quad \text{T40} \quad \text{IBR IF BACK OK} \\
\text{CAP} & \quad \text{BX, #DTAB} \quad \text{SEE IF FIRST SECTOR} \\
\text{BEG} & \quad \text{T40} \\
\end{align*}
4,122,518

95

MOV BX, R1
JUSE BACK FROM PREVIOUS SECTOR

T40:
MOV R0, 0
MOV R9, ITHR

T45:
TST DATA
BCT T45
CLR ITHR
JTHR=0 IF NO DATA

T50:
CALL OUTCOM, (ITHR, ..., FOUR)
ADD #7, T50+10

T60:
CALL OUTCOM, (BACK, ..., THREE)
ADD #7, T60+10
SUB #128, R4
CMP R4, #HIST
BHIS T19
JBR IF MORE HISTOGRAMS TO ANALYZE

SUB #6, T60+10
CLR B @T60+10
INC SHUM
CALL OUTCOM, (SHUM, M0+2, THREE)

ADD HC, SHUM
DEC SHUM

MOV B ' - . M0+3
CALL OUTCOM, (SHUM, M0+3, TWO)
BIT #2, @#SWR
BEQ T70
JBR IF SW I DOWN

CALL TYPE, M0

T70:
RTS PC

TH0:
MOV MC, R1
JSR PC, THEAD SET UP HC HEADINGS

MOV #MSG_R2

TH5:
MOV #SHEAD_R3
MOV #7, R4

TH8:
MOV (R3)+(, (R2)+
SOD R4, TH8
SOD R1, TH8
JBR IF MORE SECTORS

TH1:
MOV @R4, R4
TH2:
MOV @R4, R2
TH3:
MOV (R3)+, (R2)+
SOD R4, TH3

TH4:
MOV #R3, R2
TH5:
MOV #R3, R0
TH6:
ADD 2R4, R0
TH7:
MOV R0, R3+
TH8:
SOD R2, R2
TH9:
MOV (SP)+, R4
THA:
JSR PC, SMOOTH SAVE R4

SH20:
MOV @HIS+2, R3
SH21:
MOV #62, R2
SH22:
MOV (R4)+, R0

SH23:
ADD @R4, R0
SH24:
ADD 2R4, R0
SH25:
MOV R0, (R3)+
SH26:
SOD R2, R2
SH27:
MOV (SP)+, R4
SH28:
RTS PC

SH29:
JSR PC, TSUB

SH30:
MOV @HIS, R0
MOV #10, R1
MOV BK512, R2
BACK RANGE IS NOMINALLY 2-40

SH31:
MOV @HIS+2, R3
SH32:
CMP R1, (R3)+
BGE TS45
SH33:
MOV (R3), R1
SH34:
MOV R3, R0
SH35:
R9- HIS+1

SH36:
TST (R3)+
SH37:
SOD R2, TS40
SH38:
SUB @HIS, R0
SH39:
MOV R0, BACK
JSTORE BACK

SH40:
MOV @HIS, R0
MOV  #4, R1
MOV  DTSIZ, R2

DATA RANGE IS NOMICALLY FROM 58-124

TS59:  CMP  R1, (R3)+
BGE  TS60
CMP  -2(R3), -4(R3)
BLT  TS59

MOV  -4(R3), R1
MOV  R3, R0

TST  (R3)+

TS60:  SUB  #HIS, R0

MOV  R0, DATA

STOR DATA

RTS  PC

MAXH=512.
MAXHIS=499.

KEY:

EL:  .WORD  586.
SS:  .WORD  0.

DEBG:  .WORD  58.

SPS:  .WORD  38.

ISPS:  .WORD  36.

BKSIZ:  .WORD  24.

DTSIZ:  .WORD  38.

SIZE:  .WORD  12288.

SCTR:  .WORD  0.

HDTMN:  .WORD  512.

SPAC3:  .WORD  49.

EJECT:  .WORD  61.

UITC:  .WORD  2.

SYTY:  .WORD  70.

ZERO:  .WORD  0.

ONE:  .WORD  1.

TWO:  .WORD  2.

THREE:  .WORD  3.

FOUR:  .WORD  4.

FIVE:  .WORD  5.

X:  .WORD  0.

Y:  .WORD  0.

LINE:  .WORD  0.

QFLAG:  .WORD  0.

DFLAG:  .WORD  0.

NPCIR:  .WORD  0.

PBR:  .WORD  1.

NPS2:  .WORD  68.

DPE2:  .DCRP  69.

TX:  .WORD  1.

BX:  .WORD  9.

BACK:  .WORD  1.

DATA:  .WORD  0.

ITHK:  .WORD  0.

SHUN:  .WORD  0.

QY:  .WORD  0.

K=6

NK=6

PDBBMRMCQOL

P60, P70, P80, P90, P95, P100

WORD  P60, P70, P80, P90, P95, P100

/DPDBBMRMCQOL/

ASCII

WORD  P60, P70, P80, P90, P95, P100


EL:  .WORD  586.

SS:  .WORD  0.

DEBG:  .WORD  58.

SPS:  .WORD  38.

ISPS:  .WORD  36.

BKSIZ:  .WORD  24.

DTSIZ:  .WORD  38.

SIZE:  .WORD  12288.

SCTR:  .WORD  0.

HDTMN:  .WORD  512.

SPAC3:  .WORD  49.

EJECT:  .WORD  61.

UITC:  .WORD  2.

SYTY:  .WORD  70.

ZERO:  .WORD  0.

ONE:  .WORD  1.

TWO:  .WORD  2.

THREE:  .WORD  3.

FOUR:  .WORD  4.

FIVE:  .WORD  5.

X:  .WORD  0.

Y:  .WORD  0.

LINE:  .WORD  0.

QFLAG:  .WORD  0.

DFLAG:  .WORD  0.

NPCIR:  .WORD  0.

PBR:  .WORD  1.

NPS2:  .WORD  68.

DPE2:  .DCRP  69.

TX:  .WORD  1.

BX:  .WORD  9.

BACK:  .WORD  1.

DATA:  .WORD  0.

ITHK:  .WORD  0.

SHUN:  .WORD  0.

QY:  .WORD  0.
****** BINARY PICTURE GENERATOR ******
SUBROUTINE BINARY(BUNIT,FILPEX)
BYTE FILPEX(9)
IMPLICIT INTEGER (A-Z)
COMMON/C1/HR,NC,LPS,SPS,TTAB(64),BTAB(64),BPER
COMMON/C2/SLO,SSD,HLO,HSG,HOB,B,MNH03
COMMON/DS/HARMA,AREA,MNEPT,MXEP,T,MXLEN,NUM,TDN,ISWI
COMMON/C3/WORD(4128),PIC(12344),PBUF(512)
INTEGER PAR(23),KEY(6),SPAR(5),CTAB(307),MTAB(30)
BYTE PIC,PMGS(22),PBUF
DATA KEY/'BP','TD','AR','EP','LE','SK'/
call SSWTCH(4,ISWI)
if(ISWI.EQ.1) call TIMER
end
C
call SPARMAPAR,PAR,20
C HPAR WILL BE ZERO THE FIRST TIME BINARY IS RUN ON A SPREAD
if(HPAR.EQ.0) call TYPE(' BINARY')
call OPEN(PIC,6144,10,'SCN')
call STLABLE(PIC,SPAR,INDEX)
if(HPAR.EQ.0) call TYPE(PIC(INDEX+73),64)
C
default parameters
B = 3
MNH03 = 85
MHAEMA = 32
MXPER = 2000
MNEPT = 8
MXPT = 2000
SLO = 1
MXLEN = 98
S30 = 1
C
CALL RPARAM(NPAR,PAR(20))
I = -2
1 I = I + 1
2 I = I + 2
IF(I.GT.PAR) GOTO 20
DO 5 K = 1, NKEY
IF(PAR(1).NE.KEY(K)) GOTO 3
GOTO(2,12,13,14,15,16,K)
5 CONTINUE

C PARAMETER ERROR
PAUSE 1
GOTO 20
12 TOSH = PAR(I+2)
GOTO 1
13 I = I + 2
MHAREA = PAR(I)
MHAREA = PAR(I+1)
GOTO 2
14 I = I + 2
MNAPT = PAR(I)
MNAPT = PAR(I+1)
GOTO 2
15 MAXLEN = PAR(I+2)

DO 16 I = 1, MH
B = PAR(I+2)
GOTO 1

DO 20 I = 1, MH
CONTINUE
IF(ISW4.EQ.1) CALL TIMER
B = (B+1)/2
SPS2 = (SPS+1)/2
MSAMPS = NLO - (CH-1)*LPS
NW = NSO + 32 + 
HW = (NSO+31)/32
HWT = NW/16

C *** BINARY PICTURE AREA(WORD) IS ASSUMED TO BE ZERO INITIALLY
C HW = NUMBER OF WORDS OUT PER LINE
C HWI = NUMBER OF WORDS IN PER LINE
S = 0
C = 0
C
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 - 
IF(C.EQ.NH-1) GOTO 50
S = S + 16
IF(S.LT.SPS2) GOTO 50
S = S - SPS2
NTAB(W) = 16 - S
C = C + 1
50 CONTINUE

IF(ISW4.EQ.1) CALL TIMER

C
SI = 1
WI = NW
REC = 1
ILPS = LPS

DO 300 R=1,HR
IF(REQ,HR) ILPS = MSAMPS
C
DO 290 L=2,ILPS,2
CALL GET(PIC,REC,P1)
REC = 0
CALL MVW(PIC(P1+1),PBUF,HWT)
CALL GET(PIC,0,P1)
SUBROUTINE SEGMT
Implicit Integer(a-Z)
Common/C1/SPAR(133), SLO, SSO, HD, HDN, W, OB, OBX, OBXN
Common/C1/NNAREA, MAREA, MNPERM, XPERM, MAXLEN, NW, TOBN, ISWI
Common/C1/WORK(4126), NPERM, XMIN, XMAX, YMIN, YMAX, XMAX
Common/C1/DIR(012), EDUF(39), EDGE(40,14)
Common/C1/NK, LIP(60), SIF(60), F1
Integer • PAR(5)
Byte Label(S12), M(6), WHITE(14)
Data PAR/S12, 1024, 7, 1, 512/
Data Label/70*'A', 'B', 'C', 70* ', 'A', 'L', 360* ' '/
Data MAXH/1022/, DREC/2/
C THE EDGE POINT DIRECTORY CONTAINS THE FOLLO.
WING FOR EACH OBJECT
C DIR(1) = FBM (FIRST BLOCK WRITTEN)
C DIR(1) = N (NUMBER OF END POINTS)
C DIR(1) = YMIN
C DIR(1) = XMIN
C DIR(1) = YMAX
C DIR(1) = XMAX
C THE EDGE POINTS ARE GROUPED IN COORDINATE PAIRS REPRESENTING SEGMENTS
C EDGE(E) = X
C EDGE(E+1) = X
C EDGE(E+2) = Y
C EDGE(E+3) = Y
C CALL ITLA(0, WHITE, 14)
C READ OLD VALUE OF F1
C MAXR = NHAREA/4
C MAXR = NHAREA/4
C NNEP = MNPERM/2
C NNEP = MNPERM/2
C MAXLEN = MAXLEN - 2
C CALL SSWICH(1, ISWI)
C IF(ISWI.EQ.1) CALL TYPE(' SEGMNT')
C SSWICH(4, ISWI)
C IF(ISWI.EQ.1) CALL WPRINT(' SEGMNT')
C CALL ZIA(CBUF, 329)
C CALL ZIA(DIR, 510)
C CALL ZIA(EDGE, 1024)
C CALL MVLC(EDGE, LABEL, 1024, 3)
C CALL PUTF(EGRUP, 34, S14)
C
105
CONTINUE

REC = 3

NW = NW/2

H = H/2

N = H/2

NO = NUMBER OF OBJECTS

WI = WORD INDEX

EWI = ENDING WORD INDEX

EI = EDGE INDEX

DI = BIT INDEX (*0, 1, 2, ..., 31)

EWI = W

DI = 1

NOB = 1

C

SCAN BINARY PICTURE (WORD) LINE-BY-LINE

DO 200 L = 1, NL

WI = EW1 + 1

EWI = EW1 + NU

LOOP = 0

C

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

140 IF(DRACH(WI, B1, WORD, EW1, MASK), EQ. 0) GOTO 280

LOOP = LOOP + 1

IF(LR, EQ. 280) PAUSE 1.

EDGE(1) = L

EDGE(2) = (H - (EW1 - WI + 1)) + 16 + B1 + 1

CALL TURTLE(H, EDGE, WORD(WI), MASK, NW2)

IF(BN.EQ.0) GOTO 140

H2 = 2 + H

IF(NOB.NE.TOBN) GOTO 141

CALL LIST(NOBO.M, YMIN, XMIN, YMAX)

C

CALL PUMP(EDGE, EDGE(N2), 1)

141 CONTINUE

IF(EW.EQ.255) GOTO 142

CALL SORTIN(EDGE, H, SIND)

IF(NOB.EQ.TOBN) CALL PUMP(EDGE, EDGE(N2), 1)

IF(SIND.EQ.0) GOTO 145

142 IF(EW.EQ.MAXN) GOTO 143

CALL TYPE(' TOO MANY EDGES')

CALL TYPE(' )

GOTO 280

143 CONTINUE

IF(ISWI.EQ.1) CALL TYPE(' SORT')

IF(ISW4.EQ.1) CALL STIMER

CALL SORT(EDGE, W)

IF(ISW4.EQ.1) CALL STIMER

C

H = NUMBER OF EDGES POINTS

145 CALL ERASE(WORD, EDGE, AREA, N2, HN)

IF(AREA.LT.0.5 OR AREA.GT.N100) GOTO 195

IF(YMAX-YMIN.GE.MXLOR OR XMAX-XMIN.GE.MXLEN) GOTO 196

C

ENTER OBJECT INTO DIRECTORY

IF(HN.GE.60 OR TBN, GE.9) GOTO 175

NK = HN + 1

LIP(NK) = EDGE(1) + EDGE(1)

SPIP(NK) = EDGE(2) + EDGE(2)

C

STORE NOG INFORMATION FOR QUICK COUNT OPTION

Y4 = EDGE(1)

X4 = EDGE(2) + 14

IF(X.LT.0) X = 0

C

WRITE AN ARROW BY THE OBJECT

CALL DLIN(1,WHITE,Y,X,0,1,0)

CALL DLIN(WHITE, Y,X,0,1,0)

CALL STERIC(S, ISW6)

IF(ISW0.EQ.1) GOTO 175

C

CALL DLIN(WHITE, Y,-4,X,0,1,0)

CALL DLIN(WHITE, Y,-4,X,0,1,0)

CALL DLIN(WHITE, Y,2,X,0,2,1,0)

CALL DLIN(WHITE, Y,2,X,0,2,1,0)

CALL DLIN(WHITE, Y,2,X,10,2,1,0)

CALL DLIN(WHITE, Y,2,X,10,2,1,0)

CALL DLIN(WHITE, Y,4,X,0,2,1,0)

CALL DLIN(WHITE, Y,4,X,0,2,1,0)
CONTINUE

DIR(DI) = REC

DIR(DI+1) = N
DIR(DI+2) = MAX0(YMIN-B.I)
DIR(DI+3) = MAX0(XMIN-B.I)
DIR(DI+4) = MIN0(LYMAX+B.NL)
DIR(DI+5) = MIN0(XMAX+B.NS)

NOD = NOD + 1

DI = DI + 6

IF(NOD.GT.MAXNOD) GOTO 201

C

IF(T0DH.LT.0) GOTO 140

C DON'T WRITE IF QUICK COUNT OPTION

EQ.195.1=1.N2.S12

CALL WRITE(EBUF,REC,EDGE(I))

GOTO 140

C

NOD = NOD - 1

IF(T0DH.LT.0) GOTO 220

IF(DI.GT.1) CALL WRITE(EBUF,REC,DIR)

CALL WRITE(EBUF,1.FPAR)

CALL CLOSE(EBUF)

GOTO 140

C

MAXH = MAXH + HUN

C

IF(NOB.GT.0) GOTO 200

C IF NO OBJECTS FOUND, TYPE MSG AND CALL INT1

C CALL TYPE(' NO OBJECTS FOUND: CHECK THRESHOLDS')

C CALL TYPE('B')

CALL APHASE(7)

GOTO 200

CONTINUE

IF(T0DH.GE.0) RETURN

CALL MVL(' HC=**',M,6)

CALL OUTCON(NK,M,6,2)

CALL TYPE(M,6)

CALL UPARAM(122,NK,6)

CALL APHASE(2)

C WRITE NOD PARAMETERS AND CALL INTI

END

SUBROUTINE SKIRT

IMPLICIT INTEGER(A-Z)

COMMON/CI/SPAR(133),SGL,SSG,NDL,NBD,B,BPER

COMMON/CI/DIR(S12),SDIR(S12)

COMMON/CI/EBUF(S56),EDGE(1924),SEG0(56),SEG0(192)

COMMON/CI/SPAR(900),ESD(5)

INTEGER PAR(5)

BYTE EBUF,SEC

DATA MAXS/300/,MAXD/510/,S/1/,D/1/

DATA DREC/2/,IDREC/2/

C

THE SEGMENT DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJECT

C

SDIR(DI) = BSIC(BEGINNING SEGMENT INDEX)

SDIR(DI+1) = ESIC(ENDING SEGMENT INDEX)

SDIR(DI+2) = YMIN

SDIR(DI+3) = XMIN

SDIR(DI+4) = YMAX

SDIR(DI+5) = XMAX

C

THE SEGMENTS ARE STORED AS TRIPLET'S

SEC(S) = Y

SEC(S+1) = X

SEC(S+2) = X

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, ISWI)
IF(ISWI. EQ. 1) CALL TYPE(' SKIRT')
CALL AFIEL(EBUFF, 1, ' PIC', 5, 5)
CALL OPEN(EBUFF, ' TB24', 0, ' NV2')
CALL AFIEL(SECB, 5, ' PIC', 5, 5)
CALL OPEN(SECB, ' TB24', 0, ' NV2')
CALL GLABEL(EBUFF, PAR, IND)
CALL MVL(' SEGMENT FILE', EBUFF(INB+73), 13)
CALL PLABEL(SECB, PAR, EBUFF(INB+1))
CALL PUT(SECB, 1, IND)
CALL READ(EBUFF, 1, IND, PAR)
DPER = SPAR(133) - 5
DREC = 2*[(HOB+94)/85]+2
CALL WRITE(SECB, 1, SL0)

N4 = NLH/2
NS = NS0/2
B2 = 2*B + 1
SINC = MAXSB/B2
MAXSB = SINC*B2

S = SEGMENT INDEX
SI = SEGMENT BUFFER INDEX

DO 200 0BN=1,WOB
IF(DI, NE. 1) GOTO 30
CALL READ(EBUFF, IDREC, IND, DIR)

IDREC = (IDREC + 1)
CALL MVL(DIR, SDIR, 512)
IREC = DIR(D1)
NE = DIR(D1+1)
SDIR(D1) = S
CALL INTERP(D1)
NEMD = 1*11
NREC = (NEMD+511)/512

DO 130 L=1,HREC
CALL READ(EBUFF, IREC, IND, EDGE)
IREC = IREC + 1
EIN = MIN0(NEMD, 512)

ADD HORIZONTAL SKIRT TO SEGMENTS
DO 49 E=2, EIN, 1
EDGE(E) = MAX0(EDGE(E), B, 1)
EDGE(E+2) = MIN0(EDGE(E+2), B, NS)

IF(L, NE. 1) GOTO 60
Y0 = EDGE(1)

USE FIRST SEGMENT END POINT PAIR TO INITIALIZE SBUF
XI = EDGE(2)
X2 = EDGE(4)

SI = 1
I = 1

60 CONTINUE

PROCESS REMAINING SEGMENT END POINTS
DO 120 E=E1, EIN, 4
Y = EDGE(E)
IF(Y, EQ, Y0) GOTO 90

ADANCE SBUF ONE ROW
SI = SI + SINC
I = I + 1
IF(SI, GT, MAXSB) SI = 1
4,122,518

111

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

112

82 Y0 = Y0 + 1
YBAR = YBAR + 1
SBUF(S1) = EDGE(E+1)
SBUF(S1+1) = EDGE(E+3)
ESI(I) = S1

90 SIP = 1
X10 = EDGE(E+1)
X20 = EDGE(E+3)
DO 100 II=1,82
SIQ = SIP
SEND = ESI(II)
X1 = X10
X2 = X20

93 XP1 = SBUF(SIQ)
IF(X2.LT.XP1-1) GOTO 95
XP2 = SBUF(SIQ+1)
IF(X1.GT.XP2+1) GOTO 95
SEGMENT (X1,X2) IS CONNECTED TO A PREVIOUS SEGMENT
IF(SIQ.EQ.SEND) GOTO 94
X1 = MINB(X1,XP1)
X2 = MAXB(X2,XP2)
SBUF(SIQ) = SBUF(SEND)
SBUF(SIQ+1) = SBUF(SEND+1)
SEND = SEND - 2
GOTO 93

94 SBUF(SEND) = MINB(X1,XP1)
SBUF(SEND+1) = MAXB(X2,XP2)
GOTO 98

95 S IQ = SIQ + 2
IF(SIQ.LE.SEND) GOTO 93
SBUF(SIQ) = X1
SBUF(SIQ+1) = X2

98 ESI(II) = SIQ
100 SIP = SIP + SINC
CONTINUE

110 E1 = 1
NEND = NEND - E1H

112

RELEASE REMAINING SEGMENTS
DO 150 II=1,82
IF(YBAR.GT.NL) GOTO 100
I = I + 1
IF(SIQ.GT.MAXSB) SI=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

150 YBAR = YBAR + 1

159 SDIR(DI+1) = S - 1

C
C
DI = DI + 6
IF(DI.LT.MAXDI) GOTO 280
CALL WRITE(SEGB, DREC, DIR)
CALL WRITE(SEGB, DREC+1, SDIR)
DREC = DREC + 2
DI = 1
GOTO 200
CONTINUE
C

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

DO 210 I=1,5,512
CALL CLOSE(SEGB)
CALL CLOSE(EBUF)
END

SUBROUTINE chrome(bunit, filpex)

BYTE FILPEX(9)
IMPLICIT INTEGER(A-Z)
COMMON/C1/SL0, SSC, HLO, HSD, N0B, B
COMMON/C1/DIR(S12), CMVB(152), (S512), PBUF(138)
C THE FIRST 28 WORDS OF PBUF ARE THE MV, FOLLOWED BY 1024 WORDS OF
C BUFFER STORAGE, FOLLOWED BY 256 WORDS OF PICTURE STORAGE
COMMON/C1/SEGB(56), CMVB(563), CDIR(S10), BBUF(100)
COMMON/C1/DREC, RFSI
BYTE SEGB, PBUF(216)
EQUIVALENCE (PB, PBUF)
INTEGER PAR(5)
DATA SECSIZ/64, SSM1/63/
DATA MAXBUF/512, MAXBUF/SEGB, IDREC/2, LBUF/1832, DREC/2/
C LPBUF IS THE LENGTH OF PBUF MINUS 256 WORDS OF PICTURE STORAGE
C
CALL SSWTCH(1, ISW1)
IF(ISW1.EQ.1) CALL TYPE(' CHROME')
CALL SSWTCH(4, ISW4)
IF(ISW4.EQ.1) GOTO 3
CALL SPRINT(' CHROME')
CALL TIMER
MAXBUF=7328
C
CONTINUE
CALL AFILE(PBUF, BUNIT, FILPEx, 2, 2)
CALL OPEN(PBUF, LBUF=20, 1, 0, 0, 'SCN')
CMVB=LABEL(PBUF, PAR, IP)
NW=(PAR(2)+1)/2
CALL AFILE(SEGB, 'PIC', '5,5')
CALL OPEN(SEGB, 5, 1, 0, 'MV,5')
CALL GLABEL(SEGB, PAR, INH)
CALL ZIA(CMVb, -20)
CALL AFILE(CMVb, 'PIC', '5,5')
CALL OPEN(CMVb, '1,1,1, 'MV,5')
CALL MLV(' CHROME FILE', PBI, IP, 180), 12
CALL PLABEL(CMVb, PAR, PB, (IP+1))
CALL READ(SEGB, 1, INH, SLO)
CALL PUT(CMVb, 1, IC)
CALL MV(SLO, CMVB, IC, 2, 1), S12
HREC = (H0B+64)/85
IREC = 24HREC + 2
DREC = HREC + 3
CALL READ(SEGB, IDREC, INH, DIR)
CALL PUT(CMVb, DREC-1C)
CALL MV(DIR, CMVB, IC, 2, 1), S12
CALL READ(SEGB, IDREC+1, INH, DIR)
IDREC = IDREC + 2
DREC = DREC + 1
C
FORMAT CBUF
BEND = NOB+6
SEND = DIR(BEND-4)
DISP = MAXBUF-SEND
C
J = DISP + 1
C READ IN SEGMENT FILE
DO 5 I=1,SEND+512
CALL READ(SEGB,IREC,IND,CBUF(J))
J = J + 512
IREC = IREC + 1
5 CONTINUE
C
DO 6 DI=1,BEND,6
DIR(DI) = DIR(LDI) + DISP
DIR(DI+1) = DIR(DI+1) + DISP
6 CONTINUE
THS = (DISP-1)/SECSIZ
N = THS*SECSIZ
C
DO 10 I=1,N,SECSIZ
CBUF(I) = I*SECSIZ
10 CONTINUE
C
LFSI = N - SSNI
CBUF(LFSI) = 0
C THE SEGMENT DIRECTORY CONTAINS THE FOLLOWING INFORMATION FOR EACH OBJ
C
DIR(DI) = BSI = CURRENT SEGMENT INDEX
C
DIR(DI+1) = ESI = ENDING SEGMENT INDEX
C
DIR(DI+2) = XMIN
C
DIR(DI+3) = XMAX
C
DIR(DI+4) = YMAX
C
DIR(DI+5) = XMAX
C
C THE CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING INFORMATION FOR EACH OBJ
C
CDIR(DI) = NSI = BEGINNING SEGMENT INDEX
C
CDIR(DI+1) = NSE = TOTAL NUMBER OF SECTIONS USED UP BY OBJEU
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) = FIM = FIRST HALFWORD OF SECTION USED BY OBJECT
C
CDIR(DI+5) = LIM = LAST HALFWORD OF SECTION USED BY OBJECT
C
C HFSI = NEXT FREE SECTION INDEX
C
LFSI = LFSI - SECSIZ
C
NFSI = NEXT FREE SECTION INDEX
C
DI = 1
C
50 DBEG = DI.
IF(ISV4.EQ.1) CALL TIMER
C INCREASE NUMBER OF SECTIONS IF POSSIBLE
CSI = DIR(DI)
BSI = CDIR(DI)
IF(BSI.EQ.0) BSI = CSI
51 NSI = LFSI + SECSIZ
IF(NSI.GT.BSI-SECSIZ) GOTO 52
CBUF(LFSI) = NSI
LFSI = NSI
GOTO 51
C
52 CBUF(LFSI) = 0
Y0 = CBUF(CSI)
REC = 2*Y0 - 1
CALL GET(PBUF,REC,IND)
CALL MVU(PBUF,IND+21),PBUF(LBUF+1),MVU
CALL GET(PBUF,0,IND)
IP=IND/2
C
C PROCESS OBJECTS 1 TO MOB
IF(ISV4.EQ.1) CALL TIMER
DO 100 DI=DBEG,BEND,6
CSI = DIR(DI)
IF(CSI.EQ.0) GOTO 100
Y = CBUF(CSI)
100 CONTINUE
C
IF(Y.GT.Y0) GOTO 110
ESI = DIR(DI+1)
LSI = CDIR(DI+3)
NU = CDIR(DI+5)
NSEC = 0.
IF(LSI.EQ.0) GOTO 58
C
FIRST SEGMENT OF OBJECT. ASSIGN A SECTION
IF(NFSI.EQ.0) CALL TYPE(' UMBUF1')
IF(NFSI.EQ.0) CALL UMBUF(0,BEG,BEND,BI)
117

C DIR DI = CSI  
C DIR: DI+1 = 1  
C DIR: DI+2 = NSF1  
C DIR: DI+4 = 1  
C DIR: DI+5 = 1  
LSI = NSF1  
NSF1 = CBUF(NSF1)  
LHW = 1

C LOOP THROUGH EACH SEGMENT OF CURRENT OBJECT

58 DO 90 SI = CSI, CSI + 1
  C V = CBUF(SI)
  IF(Y GT Y0) 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. SECSIZ) GOTO 60  
    IF(NSF1 .EQ. 0) CALL TYPE('WBUF2')
    IF(NSF1 .EQ. 8) CALL WBUF(BEG, END, DI)
    NSEC = NSEC + 1  
    CBUF(LSI) = NSF1  
    LSI = NSF1  
    NSF1 = CBUF(NSF1)
    LHW = 1
  C
  60 CBUF(LSI+LHW) = PBUF(X).
  62 LHW = LHW + 1
  C  
  XI = CBUF(SI+1) + IP  
  70 X2 = CBUF(SI+2) + IP
  C
  90 CONTINUE

C
  C DIR: DI+3 = LSI  
  C DIR: DI+5 = LHW  
C END OF OBJECT--WRITE OUT TO DISK

91 CALL WBUF(DI)  
  CSI = DIR(DI)
  IF(CSI .EQ. 0) GOTO 91  
  IF(ISW4 .EQ. 1) CALL TIMER  
  GOTO 100

C REMAINING PORTION OF OBJECT LIES BELOW CURRENT LINE
C UPDATE DIRECTORIES AND CONTINUE

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

100 CONTINUE

C NO MORE OBJECTS ON THIS LINE. UPDATE AND CONTINUE.

110 CONTINUE

C IF(ISW4 .EQ. 1) CALL TIMER  
C DO 120 DI = BEG, END, 6  
  CSI = DIR(DI)
  IF(CSI .EQ. 0) GOTO 50

120 CONTINUE

C  
  IF(ISW4 .EQ. 1) CALL TIMER  
  CALL PUT(CNVB, BSEG, IC)  
  CALL NW(IW, CNVB, IC/2+1), 102)  
  CALL CLOSE(CNVB)  
  CALL CLOSE(PBUF)  
  CALL CLOSE(SEGB)  
  IF(ISW4 .EQ. 1) CALL TIMER
END
SUBROUTINE ROB
IMPLICIT INTEGER(A-Z)
COMMON/C1/CMBV(56), (C512)
COMMON/C1/CMVB(56), EDGE([124]), LBW(60), BUF, 6TBL
COMMON/C1/SLOSSO, SSOL, MG, MSOL, MSG, MSG(52), MG(52)
COMMON/C1, PERIM, YM, XM, YM, XM
COMMON/C1/DIR(60), TLAB(60), AREA(60), ID(60)
COMMON/C1/BLUE, LIP(60), SIP(60), PERM(60), CIRCA(60)
LOGICAL = 1 BUF([124])
BYTE CMBV, CMVB, STBL([124]), MSG(52), MSG(52), TMSG([124])
INTEGER DIR(510), PAR(160), HIST(64), OTTAB(60)
EQUIVALENCE (DIR, SDIR([145]), HIS, EDGE)
DATA MAXBUF/[124], RECSZ/1024, HOBPH/6
DATA NCF/103701/9, NPILOT/6, MAXHOB/60, TOBN/6, MAXNP/188
DATA ORECZ/2, DRECZ/2, RFLAG/1, OTTAB/60=0
C
THE INPUT CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJ
C
	DIR(D1) = BACKGROUND
C	DIR(D1+1) = THRESHOLD
C	DIR(D1+2) = CMS (MINIMUM LINE)
C	DIR(D1+3) = CMXL (MAXIMUM LINE)
C	DIR(D1+4) = CNXL (MAXIMUM LINE)
C	DIR(D1+5) = CS (MAXIMUM SAMPLE)
C
THE OUTPUT CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJ
C
	SDIR(C1) = FB0 (FIRST RECORD BLOCK WRITTEN TO DISK)
C	SDIR(C1+1) = AL (MINIMUM LINE)
C	SDIR(C1+2) = MS (MINIMUM SAMPLE)
C	SDIR(C1+3) = LBW (LAST RECORD BLOCK WRITTEN TO DISK)
C
THE INPUT CHROMOSOME ARE IN THE FOLLOWING FORMAT
C
	C(1) = PREVIOUS BLOCK WRITTEN
C	C(2) = NSEG (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
NOTE THAT TWO CONSECUTIVE LINES OF SEGMENT GREY VALUES ARE PRESENT
C
ALL THE DIMENSIONS ARE THOSE OF THE HALF-PICTURE AND MUST BE
C
DOUBLED.
C
THE OUTPUT CHROMOSOME ARE IN THE FOLLOWING FORMAT
C
	C(1) = NSEG (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
CALL SSWCHR(1, 165)
IF(ISW1.EQ.1) CALL TYPE('ROB')
CALL SSWCHR(+1, 164)
IF(ISW4.EQ.1) CALL PRINT('ROB')
CALL RPARAM(HP, PR, MAXNP)
IP = 1
28 IF(IP.GT.HP) GOTO 33
IF(PAR(IP).EQ.T0) GOTO 30
IF(PAR(IP).NE.TD) GOTO 35
OTTAB(PAR(IP+2)) = PAR(IP+3)
IP = IP+4
GOTO 20
30 TOBN = PAR(IP+2)
IP = IP+3
GOTO 20
35 CONTINUE
RECSZ = RECSZ/2
LBW(60) = LAST BLOCK WRITTEN FOR CURRENT CHROMOSOME
NF = DIRECTORY IN MAP
NC = NUMBER OF CHROMOSOMES
C
CALL AFILE(CHYBV, 1, 'PIC', '5.5')
CALL OPEN(CHYBV, 1024, 5, 'NV2')
CALL AFILE(SMCMB, 5, 'UCR', '13, 13')
CALL OPEN(SMCMB, 1024, 5, 'UCR')
CALL GLABEL(CMBV, PAR, IND)
CALL MVL('RGB FILE', CHYBV(IND+100), 12)
CALL PLABEL(SMCMB, PAR, CHYBV(IND+1))
CALL SSWCHR(0, ISW1)
IF(ISW4.EQ.1) GOTO 37
CALL PRINT('1')
CALL PRINT(CHYBV(IND+73), 79)
CALL QPRINT(CMVB,IND=145,390).
CALL QPRINT(CMVB,IND=217,790).
CALL QPRINT('OBJ',PER=8,MAX=LINE,SAMP).

37 CALL ITLA(32,MSGB,128).
CALL PUT(CMVB,BREC,IND).
CALL READ(CMVB,IND,SLO).
CALL READ(CMVB,BREC,IND,DIR).
BREC = BREC + 1.
CALL READ(CMVB,BREC,IND,LBW).
DPER = 100 - DPER.
B = 2*B.
B1 = B + 1.
B3 = 2*B + 1.
DIL = 1.
C = 2.
LOOP THROUGH EACH OBJECT.
DO 500 OBS=1,NOB.
REC = LBK(OBN).
CRL = DIR(DI+2)+2 - 1.
CMS = DIR(DI+3)+2 - 1.
CMXR = DIR(BI+4)+2.
CMX5 = DIR(BI+5)+2.
CML = CMXR - CMR + 3.
CNX = CNX + CMS + 2.
CMXL = CMR - 1.
CMSM = CMS - 1.
IF(CML+CMSL.LT.MAXBUF) GOTO 50.
CALL SSWITCH(1,ISU).
IF(ISU,HE.1) GOTO 580.
CALL HMLC('OBJECT TOO LARGE',MSG7,20).
CALL OUTCOM(OBN,MSG7(10),2).
CALL TYPE(MSG7,20).
GOTO 580.
C = 50.
READ IN CURRENT CHROMOSOME.
CALL ZIA(BUF,MAXBUF/2).
CALL ZIA(HIS.64).
C = 51.
CALL READ(CMVB,RECD,IND,C).
HEG = C(2).
CI = 3.
C = 60.
HM=1,HSEC.
BI = 2*(C+1)*C + CMS + 1.
W = C(C+2).
H2 = 2+W.
IF(BI,LT.1) GOTO 60.
C = 51.
CALL HMLC('OBJECT TOO LARGE',MSG7,20).
CALL OUTCOM(OBN,MSG7(10),2).
C = 45.
IF(BI,GT.MAXBUF) GOTO 45.
CALL HMLC('BUF FROM CHROME OUTPUT','MSG7,45).
CALL HMLC('BUF(BI),HIS,H2).
CALL HMLC('BUF(BI,CMS),HIS,H2).
C = 60.
CI = CI + H2 + 3.
C = 51.
IF(REC,HE.0) GOTO 51.
C = 63.
GOTO 63.
C = 62.
B1 = B1 + CMS.
CONTINUE.
C = 63.
BACK = DIR(BI+2).
C = 64.
THRES = DIR(BI+1)/2.
RETHRESHOLD OBJECT.
HIS(1) = 0.
DATA = 1THRES.
THRESH = 1THRES.
123
IF(RFLAG.EQ.0) GOTO 72
HIS(64) = 0
MAX = 0

C
I = THRES + 1
F = HIS(I-I) + HIS(I) + HIS(I+I)
IF(FLE MAX) GOTO 65
MAX = F

65 CONTINUE
THRESH = (SOPER + BACKP + SPERDATA) / 100
IF(COTAB(OBN).NE.0) THRESH = OHTAB / 2
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
HSB = CHS1-B
BCNS = B+CHS

C
DO 369 L = 1, CHL2
BI = EBI - 1
EBI = EB1 + CHS

210 IF(ISEG(BI,BUF,THRESH,EBI).EQ.0) GOTO 300
SB1R(CDI) = ARE+1
EDGE(1) = L
S = CHS - (EBI - BI)
EDGE(2) = S
CALL SOTCH(EDGE,BUF(BI),THRESH,CHS)
IF(NH.EQ.0) GOTO 210
IF(OBN.EQ.TBHN) CALL ILIST(1, YMIN, XMIN, YMAX, XMAX)
IF(OBN.EQ.TBHN) CALL PDPUMP(EDGE,EDGE(2+N),1)
IF(CE.EQ.355) GOTO 301
CALL SORTIN(EDGE,H,SNH)
IF(SNH.EQ.0) GOTO 215
CALL TYPE(' EDGE OVERFLOW, THRESHOLD RAISED')
CALL TYPE('')
THRESH = THRESH + 4
GOTO 219

214 IF(ISWI.EQ.1) CALL TYPE(' SORT')
IF(ISWI.EQ.1) CALL STIMER
CALL SORT(EDGE,H)
IF(ISWI.EQ.1) CALL TIMER

215 CONTINUE
IF(OBN.EQ.TBHN) CALL PDPUMP(EDGE,EDGE(2+N),1)
IF(OBN.EQ.TBHN) CALL ILIST(L.B.CHL,CHE-0BM)
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 OR Y.GT.NLB) GOTO 238
X1 = MAXX(EDGE(1+B1)
X2 = MIN(EDGE(1+B1)
IF(X1.GT.X2) GOTO 235
BB = YABS + X1
I = 89 - B

C
DO 228 LEFT = 0
IF(LVFAT(BUF.LEFT)).NE.0) GOTO 221
CONTINUE

221 RIGHT = BO + X2-X1 + 0
C
DO 222 I = 1, 81
IF(LVFAT(BUF.RIGHT)).NE.0) GOTO 223
RIGHT = RIGHT - 1
LEFT = LEFT + B
RIGHT = RIGHT - B
IF(TOBH.EQ.0BN) CALL ILIST(X1,X2,LEFT,RIGHT,Y)
IF(LEFT.GT.RIGHT) GOTO 233
C
DO 230 I=LEFT,RIGHT
BP = I - BCNS + 1
DO 225 LL=1,BCNS.CNS
IF(TOBH.EQ.0BN) 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
C
THIS IS AN INTERNAL POINT
GOTO 236
C
CONTINUE
C
REMOVE OBJECT FROM BUF.
MSEG = 0
NP = 0
IODN = 0
ECI = 1
C
DO 250 E=1,N2.4
Y = EDGE(E)
X1 = EDGE(E+1)
X2 = EDGE(E+2)
NS = X2 - X1 + 1
IF(NS.GT.0) GOTO 230
CALL TYPE('ROB ~ NS LE 0')
CALL TYPE('0')
CALL ILIST(Y17,NS2,ECI)
PAUSE
GOTO 23B
C
CONTINUE
NS2 = (NS+1)/2
BCI = ECI + 1
ECI = ECI + NS2 + 3
IF(BCI.LE.RECS2) GOTO 240
C
WRITE OUT CHROMOSOME LINE
C(1) = MSEG
OREC = OREC + 1
CALL WRITE(GMVBC,OREC,C)
MSEG = 0
BCI = 2
ECI = NS2 + 4
C
240 C(BCI) = CMLM1 + Y
C(BCI+1) = CMSM1 + X1
C(BCI+2) = HS
CALL REMOVE(BUF(Y+CHS+X1),C(BCI+3),STBL,MS,IODN,TRESH)
HP = NP + NS
250 MSEG = MSEG + 1
C
IF(NP.GE.25 AND INTPT.MS.0) GOTO 260
CALL SLTCH(4,1644)
IF(CMVA.MS.1) GOTO 219
CALL GPRINT('OBJECT REJECTED')
CALL ILIST(ORH,HP,YMIN,XMIN)
CALL ILIST(YMAX-YMIN,XMAX-XMIN,L.S,NC)
GOTO 210
C
WRITE OUT LAST LINE
260 OREC = OREC + 1
C(1) = MSEG
CALL WRITE(GMVBC,OREC,C)
C
NC = NC + 1
HS = YMAX - YMIN + 1
IODN = IODN + BACK*NP/4
TABS(NC) = TRESH
LIP(NC) = CMLM1 + L
SIP(HC) = CMSM1 + S
ID0(NC) = ID0N
AREA(NC) = HP

PERM(NC) = PERM
CALL SWITCH(0, 1SW0)
IF(ISW0. NE. 1) GOTO 200

C PRINT IF SW0 IS UP
CALL OUTCON(INC, TMSG(4), 2)
CALL OUTCON(PERM(NC), TMSG(19), 3)
CALL OUTCON(BACK++2, TMSG(17), 3)
CALL OUTCON(DATA++, TMSG(27), 3)
CALL OUTCON(MAX++2, TMSG(31), 3)
CALL OUTCON(CNL, TMSG(39), 4)
CALL OUTCON(OMS, TMSG(47), 4)
CALL OUTCON(CNL+1, TMSG(33), 5)
CALL OUTCON(CMSM1+5, TMSG(60), 5)
CALL OUTCON(CNL, TMSG(65), 5)
CALL OUTCON(NS, TMSG(78), 5)
CALL OUTCON(I, TMSG(60), 3)
CALL OUTCON(THRESH, TMSG(86), 3)
CALL OUTCON(COB, TMSG(344), 2)
CALL OUTCON(NP, TMSG(182), 8)
CALL OUTCON(IDM, TMSG(111), 9)
CALL OUTCON(IOM, TMSG(116), 3)
CALL OUTCON(INC, TMSG(122), 2)
CALL QPRINTF(TMSG, 120)

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

C 300 CONTINUE
350 BI = BI + 6
C
510 CONTINUE
SDIR(1) = CDI/4
NK = SDIR(1)
TEMP = PERM(1)
PERM(1) = TEMP

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

C WRITE NOB PARAMETERS
CALL MVW(ITAB, SDIR(241), 69)
CALL WRITE(SMVN, 1, SDIR)
CALL WRITE(SMVB, 2, LIP)
IF(ISW0. EQ. 1) CALL QPRINTF('1')
CALL CLOSE(SMV0)
CALL CLOSE(CMV0)
END.

C*********
C* NOB = CALMS VERSION
C* 
C*********
C
C NOB NUMBERS THE RAW SPREAD WITH THE OBJECT NUMBERS GENERATED
C BY FOB
C
C SUBROUTINE NOB(DUMIT, FIPEX)
IMPLICIT INTEGER(A-Z)
COMMON/CL/ NK, LT, ST, FI
LOGICAL ORDER
BYTE FIPEX(2)
C I/O BUFFERS
BYTE IBUF2(12344), OBUF(12344)
C RANDOM AREAS
INTEGER SPAR(18)
BYTE CHAR(2)
C SAMPLE AND LINE TABLE FOR OBJECT NUMBERS
INTEGER ST(68),LT(68),TEMP(68)
DATA X/0/.X/0.+REPL/1./MASKN/13/

C INITIALIZE DATA SETS
C
CALL SSUINCH(1,1SW1)
C
IF(ISW1.EQ.1) CALL TYPE(‘HOB’)
C
CALL RPARAM(NP,NK,122)
C
IF(I.EQ.0) CALL DCLEAR
C
CLEAR THE GRAY SCALE UNLESS FINISHING UP A COUNT.
CALL AFILE(IBUF2,DUNIT,FIPEX,2,2)
CALL AFILE(OBUF,1,’PIC’)*S,’S’)
CALL OPEN(IBUF2,6144,1,0,’SCR’)
CALL OPEN(OBUF,6144,1,1,’HOB’)
CALL GLABEL(IBUF2,SPAR,12)
NLI=SPAR(1).
NSI=SPAR(2)
N=NSI/2
Ni=1
CALL PLABEL(OBUF,SPAR,IBUF2(12+1))
C
C SEE IF LT IS IN ORDER.
ORDER = TRUE.
IF(NK.LT.2) GOTO 45
DO 40 N=2,NK
IF(LT(N,LT(M-1))) GOTO 42
CONTINUE
GOTO 45
42 ORDER=.FALSE.
45 CONTINUE.
C
RUN THROUGH EACH LINE OF THE PICTURE
C
DO 500 L=1,NLI
CALL GET(IBUF2,L,12)
CALL PUT(OBUF,L,0)
CALL MVV(IBUF2(12+1),OBUF(0+1),NK)
C
GÈMEFAÈTE NUMBER FOR EACH OBJECT AT APPROPRIATE TIME
C
IF(NK.EQ.0.OR.NI.GT.NK) GOTO 210
DO 200 N=1,NK
LD=L-LT(N)
IF(LD.LT.4). GOTO 280
IF(LD GE -4) GOTO 48
IF(ORDER) GOTO 210
GOTO 200
CONTINUE
NCHAR=2
IF(X.GE.10) GO TO 50
NCHAR=1
SD=SD+6
50 IF(SD.LT.1). SD=1
IF(X.EQ.1) GOTO 75
IF(X.EQ.4) GOTO 70
CALL OUTCON(N,PCHAR(2),2)
CALL TEXT(CHAR(3-NCHAR),NCHAR,LD+3,OBUF(0+SD),1)
CALL ITL(B,OBUF(0+SD+6*NCHAR))
GO TO 260.
70 IF(ORDER) Ni=N1+1
75 CALL ITL(0,OBUF(0+SD),6*NCHAR+1)
200 CONTINUE
210 IF(X.EQ.0) GOTO 500
C NO DISPLAY IF FINISHING UP A COUNT.
CALL DLINE(OBUF(0+1),Y,X,NSI,REPL,0)
Y=Y+REPL+1
500 CONTINUE
C
C CLOSE DATA SETS.
CALL CLOSE(1BUF2)
CALL CLOSE(2BUF)
CALL EXIT

C

C INTI - INTERACTION I TO CHECK NOB OUTPUT AND CORRECT FOR ERRORS
SUBROUTINE INTI(DUNIT,FILEX)
IMPLICIT INTEGER (A-Z)
COMMON/C1/ NK,LAB,STAB,FI
INTEGER PAR(10),SPAR(5),SST(513),NST(513),TTAB(133),RPAR(5)
INTEGER XTAB(60),VTAB(60),LTAB(60),STAB(60),TEMP(60),RPAR(100)
BYTE X(1234),BLACK(512),FILEX(10),NPAR(20),KEY(20),LIGHT(512)
BYTE MB(49),MT(21),GMSG(129),MK(8),WHITE(8),CHAR(2)
BYTE BELL(3)
EQUIVALENCE (PAR,BPAR),(HE,TTAB(2)),(LPS,TTAB(3)),(SPS,TTAB(4))
EQUIVALENCE (NL,SPAR(1)),(HS,SPAR(2))
DATA BELL;/*4,7,0*/
DATA TIME/1000/
DATA MASKPH,44,NOBPH,4E,ERASE,110S,KOUNT,1113,ROBPH,S/
DATA KEY,28,KEY/'XH','C','J','B','T','H','F','S','I','N'
1/'U','O','L','A','R','N','O','/2'/10,'O'/0/1
NLAG=1
C ALWAYS SET NLAG TO 1
C FLAG TO CALL NOB FOR A FINISH COUNT
CALL ITLA(33,LIGHT,513)
CALL ZIA(WHITE,4)
CALL NL(124, BACK***,'MB-19')
CALL NL( 'DN*** THRESH****,'RT,28)
CALL NL( 'NC***,'MK,7)
MB(19)=1
NL(23)=1
NLAG=1
RETURN 0
CALL TYPE('OCHECK SPREAD',@)
NK=0
CALL RSPARH(NP,RPAR,198,ROBPH)
CALL RSPARH(NL,NK,122,NOBPH)
IF(NP.EQ.0) FI=64
CALL DCOUNT(NK,FI)
C DISPLAY THE COUNT
20 CALL ITLA(33,GMSG,129)
CALL RCG
C READ CURSOR ADJUSTMENTS
CALL AF(LA(DUNIT,FILEX,2,2))
C OPEN SCAN DATA SET AND READ TTAB
CALL OPEN(A,11,A,1,2,'5CM')
CALL GLABEL(A,SPAR,1A)
CALL NL(A,IA+72),GMSG,64)
C SAVE LABEL FOR PATIENT REPORT
CALL GET(A,SPAR('1,IA)
CALL NL(A,IA+1),TTAB,133)
25 CONTINUE
CALL PARM(HP,PAR,10)
30 IF(NP.EQ.0) GOTO 1000
35 IF(PAR('EQ.RO') GOTO 80
35 IF(PAR('EQ.5M') GOTO 65
DG 58 K=L+KEY
IF(CV(BPAR('1),NE,IVKEY(K))) GOTO 58
GOTO (50,180,200,300,400,500,600,700,800,900,100,110,120,130,1350,1400)
50 CONTINUE
C TYPE(' PARAMETER ERROR')
65 CALL TYPE('0')
GOTO 25
80 CONTINUE
C RS - RESCAN
CALL RL(A)
CALL CLOSE(A)
CALL EXIT
CALL APHASE(10)
CALL CLOSEA()
CALL EXIT

90 CALL TYPE(' TYPE ONE OF THE FOLLOWING KEYWORDS TO SELECT AN OPTION')
CALL TYPE(' 2 - QUICK COUNT')
CALL TYPE(' 0 - CUT APART A TOUCH (POSITION CURSOR FIRST)')
CALL TYPE(' - JOIN TWO PIECES TOGETHER (USE CURSOR)')
CALL TYPE(' T - CHANGE BACKGROUND (USE CURSOR)')
CALL TYPE(' 4 - SET UP THE CURSOR TO CORRECT FOR DRIFT')
CALL TYPE(' - ADAPT THIS SPREAD')
CALL TYPE(' - RESCAN')
CALL TYPE(' F - FINISH THIS SPREAD (NO. KARYOTYPE)')
CALL TYPE(' HH - FINISH THIS SPREAD. IT HAS HH CHROMOSOMES')
CALL TYPE(' I - DISPLAY THE INITIAL. (UN-NUMBERED) SPREAD')
CALL TYPE(' N - DISPLAY THE NUMBERED SPREAD')
CALL TYPE(' E - ERASE THE LAST CUT OR JOIN REQUEST')
CALL TYPE(' U.D.L.O.R - MOVE THE CURSOR UP, DOWN, LEFT, OR RIGHT')
CALL TYPE(' X - INTERACTIVE COUNT WITH CURSOR AND BELL')
CALL TYPE(' - ADD A MISSING NUMBER ONLY FOR COUNTS')
CALL TYPE(' - REMOVE AN EXTRA NUMBER ONLY FOR COUNTS')
CALL TYPE(' Z - ZERO THE CHROMOSOME COUNT')
CALL TYPE(' S - SKIP MOB (USE OLD MCP)')
CALL TYPE(' OT N.T - SET THRESHOLD FOR OBJECT N TO T')
CALL TYPE(' IF SPREAD IS OK. TYPE CARRIAGE RETURN')
CALL TYPE(' 0')
GOTO 25

C
C - CUT APART A TOUCH

100 CALL CURSOR(Y1,X1,SL,SS)
CALL TYPE('+MOVE CURSOR TO END OF CUT','0')
DO 105 J=1,2
DO 105 I=1,3
105 CALL BLINE(WHITE,Y1-2+I,X1-1,3,0,0)
CALL MARK THE FIRST ENDPOINT
110 CALL PARAM(NP,PAR,10)
IF(P.EQ.0) GOTO 120
P=P(RP(D))
IF(P.EQ.'U' OR P.EQ.'D' OR P.EQ.'L' OR P.EQ.'R') GOTO 115
IF(P.EQ.'L') GOTO 120
C ERASE THE MARK IF E WAS TYPED
DO 112 I=1,3
112 CALL BLINE(255,Y1-2+I,X1-1,3,0,-1)
GOTO 25
115 CALL UDLR(PAR)
C CHECK FOR UP, DOWN, LEFT, OR RIGHT
GOTO 110
120 CONTINUE

CALL CURSOR(Y2,X2,EL,ES)
CALL ADD(SL,SS,EL,ES,SST,HST,ML,MS)
DO 135 L=SL-EL
Y=Y2+L*(L-1)
x=Y-E(SST(L)-1)
135 CALL BLINE(LIGHT,Y,X,HST(L),1,0)
140 CALL PARAM(PARP,10)
IF(Y<PAR2(I).NE.E) ERASE) GOTO.150
C RESTORE THE DATA IF E WAS TYPED
DO 155 L=SL-EL
Y=Y2+L*(L-1)
x=Y-E(SST(L)-1)
155 CALL BLINE(127,Y,X,HST(L),1,-1)
GOTO 25
158 REPUN=1
DO 175 L=SL-EL
CALL GETA(L,1B)
CALL PUTA(L,1A)
IF(N.EQ.1B) PAUSE 55
175 CALL ITLA(210,N*(A+SST(L)),HST(L))
C WRITE ZEROES IN THE DATA SET
GOTO 38
C
C J - JOIN TWO PIECES TOGETHER
C
C 200 CONTINUE
C CALL CURSOR(Y1,X1,SL,SS)
C CALL TYPE('M' MOVE CURSOR TO END OF JOIN ')
DO 205 I=1,3
C MARK END-POINT
C CALL PARAM(HP,PAR,10)
IF(HP.EQ.0) GOTO 220
P=P+1
IF(P.EQ.'U'.OR.P.EQ.'D'.OR.P.EQ.'L'.OR.P.EQ.'R') GOTO 219
IF(P.EQ.'E') GOTO 220
C ERASE THE MARK IF E WAS TYPED
C CALL DLINE(WHITE,Y1-2+I,X1-1,3,0,0)
GOTO 25
C CALL UDLR(PAR)
GOTO 210
C CONTINUE
C CALL CURSOR(Y2,X2,EL,ES)
C CALL ADL(SL,SS,EL,ES,NST,HL,NS)
DO 235 L=SL,EL
Y=Y+(L-1)
X=X-(GOSL-1)
C CALL DLINE(127,Y,X,NST(L),1,1)
C ERASE ON THE GRAY SCALE TO INDICATE THE JOIN-LINE
C CALL PARAM(HP,PAR,10)
IF(IVP.EQ.0) GOTO 30
C WRITE 127 IN THE DATA-SET FOR THE JOIN-LINE
C REUSE THE DATA
GOTO 25
C 220 CONTINUE
C CALL CURSOR(Y2,X2,EL,ES)
C CALL ADL(SL,SS,EL,ES,NST,HL,NS)
DO 235 L=SL,EL
Y=Y+(L-1)
X=X-(GOSL-1)
C CALL DLINE(127,Y,X,NST(L),1,1)
C ERASE ON THE GRAY SCALE TO INDICATE THE JOIN-LINE
C CALL PARAM(HP,PAR,10)
IF(IVP.EQ.0) GOTO 30
C WRITE 127 IN THE DATA-SET FOR THE JOIN-LINE
C REUSE THE DATA
GOTO 25
C 235 CONTINUE
C CALL CURSOR(Y,X,L,S)
C CALL GET(A,L,1A)
C CALL OUTCONIV(A(1A+8)),MB(7),3)
IT=(L/LPS)*HC+SP*SFI
IF(SP.EQ.HC) IT=IT-1
C CALL OUTCON(ITAB(68+IT),MB(16),3)
C CALL TYPE('H',A)
BST=BST+1
C 275 CALL ITLH(127,A(1A+8)),NST(L))
C WRITE 127 IN THE DATA-SET FOR THE JOIN-LINE
GOTO 30
C 250 CONTINUE
C CALL CURSOR(Y,X,L,S)
C CALL GET(A,L,1A)
C CALL OUTCONIV(A(1A+8)),MB(7),3)
IT=(L/LPS)*HC+SP*SFI
IF(SP.EQ.HC) IT=IT-1
C CALL OUTCON(ITAB(68+IT),MB(16),3)
C CALL TYPE('H',A)
BST=BST+1
C SET TO CHANGE BACK
320 CALL PARAM(HP,PAR,10)
IF(HP.EQ.0) GOTO 25
IF(HP.EQ.1) GOTO 35
C SET TO CHANGE BACK
350 IF(PAR(1).GT.127.OR.PAR(1).LT.0) GOTO 60
TH=TH+(1)*PAR
C CALL GET(A,SPARK(1)+1,L,1A)
C CALL PUT(A,SPARK(1)+1,L,1A)
C CALL NYU(ITAB,A(1A+1),133)
REUN=1
GOTO 25
C 320 CONTINUE
C CALL CURSOR(Y,X,L,S)
C CALL GET(A,L,1A)
C CALL OUTCONIV(A(1A+8)),MB(7),3)
IT=(L/LPS)*HC+SP*SFI
IF(SP.EQ.HC) IT=IT-1
C CALL OUTCON(ITAB(68+IT),MB(16),3)
C CALL TYPE('H',A)
BST=BST+1
C 480 CONTINUE
C CALL IVMT(7),HE,IV('W') GOTO 440
C GOTO 440 IF T WAS ALREADY REQUESTED
C DISPLAY THE SECTOR BOUNDARIES
DO 419 L=LPS,HL,LPS
C 410 CALL DLINE(WHITE,L+1,LPS+1,8,1,0)
CALL ITLA(127, BLACK, 512)
DO 415 S=SPS.NS, SPS
415 BLACK(S)=0
DO 420 I=1, S
420 CALL BLHE(BLACK, NL+NL+I+1-2, 8, NS, 1, 0)
430 CONTINUE
CALL CURSOR(Y, X, L, S)
CALL GET(A, L, IA)
CALL OUTCONTV(6(L+4S)), MT(7), 3)
IT=(L/LPS)*MC+S/SPS+1
IF(S/SPS.EQ.MC) IT=IT+1
CALL OUTCON(TAB(4+IT), MT(10), 3)
CALL TYPE(MT, 0).
BORT=4
C SET TO CHANGE THRESH
GOTO 320
C
C A - ABDOT
C
500 CONTINUE
CALL APHASE(8)
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(HASKPH)
CALL CLOSE(A)
CALL AFILE(A, 'PIC', 5, 5)
CALL OPEN(A, 512, 0, 2, 'NOB')
C SET TO ADD NK TO NOB OUTPUT LABEL
GOTO 620
610 CONTINUE
FI=0
CALL UPARAM(122, NK, NOBPH)
C SET NOB TO CALL MASK AND NOT PUT UP THE DISPLAY
CALL APHASE(HEOBPH)
620 IF(HP.EQ.3) NK=PAR(3)
C USE OPERATOR SUPPLIED COUNT
C CONVERT NK FOR GMSG
CALL NVLV(GMSG, TEMP, 69)
TEMP(25)='56:2
C INSERT LF LF
CALL TYPE(TEMP, 79)
CALL GLABEL(A, SPAR, IA)
CALL NVLV(GMSG, 67), A(IA+290), 2
CALL PLABEL(A, SPAR, A(IA+1))
C INSERT NK IN THE LABEL
630 CALL CLOSE(A)
CALL AFILE(A, 'PBATA', 6, 6)
CALL OPEN(A, 512, 0, 2, 'FRP')
CALL GLABEL(A, SPAR, IA)
NL=NL+1
CALL PLABEL(A, SPAR, A(IA+1))
CALL GET(A, NL, 1A)
CALL NVLV(GMSG, A(IA+1), 128)
CALL PUT(A, NL-1A)
PAR(1)='KC'
PAR(3)=BUNIT
CALL NVLV(FILPEX, PAR(4), 18)
CALL UPARAM(0, PAR, HASKPH)
CALL CLOSE(A)
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,1A)
  CALL BLINE(A(L)+1),2*(L-1),0,NS,1,0)
  CALL BCOUNT(NK,FI)
  GOTO 25
C
C DISPLAY THE NOB-OUTPUT
C
  650   FI=64
  CALL WAPAM(122,NK,NOPBH)
  CALL APHASE(NOPBH)
  CALL CLOSEA)
  CALL EXIT
  980   CALL MCU
  GOTO 25
  910   CALL MCD
  GOTO 25
  920   CALL MCL
  GOTO 25
  930   CALL MCR
  GOTO 25
  940   CONTINUE
C
C INTERACTIVE COUNT
C
  945   CALL CURSOR(Y0,X0,L,S)
  DO 947 I=1,HTIME
  CALL CURSOR(Y,X,L,S)
  IF(Y .LT. 78 OR X .LT. X0) GOTO 945
  947   CONTINUE
C CURSOR HAS NOT MOVED IN A WHILE
C IF(Y.LT.24.AND.X.GT.1000) GOTO 995
C IF UPPER RIGHT GOTO 995
  CALL CURSOR(Y,X,L,S)
  IF(Y.LT.1000.AND.X.GT.1000) GOTO 970
  CALL TYPE(BELL,0)
C RING THE BELL
NFLAG=1
  IF(NK.LT.60) NK=NK+1
  CALL BCOUNT(NK,FI)
  XTAB(NK)=X
  LTAB(NK)=L
  STAB(NK)=S
  DO 950 J=1,2
  DO 950 J=1,3
  950   CALL BLINE(WHITE,Y-2+1,X-1,3,0,0)
  960   CALL CURSOR(Y2,X2,L,S)
  IF(Y2.EQ.Y.AND.X2.EQ.X) GOTO 960
C WAIT FOR CURSOR TO BE MOVED
  GOTO 945
  970   CALL OUTCON(HK,NK,G,2)
  CALL TYPE(NK,G)
  IF(NK.LT.2) GOTO 25
C ORDER THE TABLE.
  DO 990 H=2,NK
  IF(LTAB(N).GE.LTAB(N-1)) GOTO 990
  LTN=LTAB(N)
  DO 990 J=1,N
  IF(LTN.LE.LTAB(J)) GOTO 985
  980   CONTINUE
  HMOV=N-J
  CALL MVU(LTAB(J),TEM,HNMOV)
  LTAB(J)=LTN
  CALL MVU(TEM,LTAB(J),HNMOV)
  CALL MVU(STAB(J),TEM,HNMOV)
  STAB(J)=STAB(N)
  CALL MVU(TEM,STAB(J),HNMOV)
  990   CONTINUE
  GOTO 25
C
  995   CONTINUE
C ERASE THE LAST SPOT
  DO 997 J=1,2
  DO 997 J=1,3
CALL DLKNE(127, YTAB(NK)-2+1, XTAB(NK)-1, 3, 0, -1)
IF(NK.GT.0) NK=NK-1
CALL DCOUNT(NK, FI)
GOTO 960

CALL CLOSE(A)
IF(HERE.GT.0) CALL AIMAGE(2)
CALL EXIT

CONTINUE

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

CONTINUE
LTAB(NK+1)=L
STAB(NK+1)=S
GOTO 1140

1130 NMOV=NK-N+1
CALL MVW(LTAB(H), TEMP, NMOV)
CALL MVW(TEMP, LTAB(H+1), NMOV)
LTAB(H+1)=L
CALL MVW(STAB(H), TEMP, NMOV)
CALL MVW(TEMP, STAB(H+1), NMOV)
STAB(H)=S
NK=NK+1

CALL DCOUNT(NK, FI)
GOTO 25

CONTINUE

C X PARAMETER - REMOVE ONE OF THE NUMBERED OBJECTS
C
MFLAG=1
IF(NK.LT.1) GOTO 60
CALL CURSOR(Y,X,L,S)
MIN=20
C FIND THE OBJECT CLOSEST TO L-S AND NO MORE THAN 20 AWAY
DO 1220 H=1, NK
DIFF=ABS(LTAB(H)-L)+ABS(STAB(H)-S)
IF(DIFF.GT.MIN) GOTO 1220
MIN=DIFF
MAIN=H

CONTINUE
IF(MIN.EQ.20) GOTO 60
H=MAIN
Y=LTAB(H)+2-2
X=STAB(H)+2-2
IF(X.LT.7) X=7
DO 1240 J=1,2
DO 1240 I=1,9
1240 CALL DLKNE(127, Y+1, X-7, 5, 0, -1)

ERASE THE OLD NUMBER
NMOV=NMV
IF(NMOV.EQ.8) GOTO 1250
CALL MVW(LTAB(H), LTAB(H), NMOV)
CALL MVW(STAB(H), STAB(H), NMOV)

NK=NK-1
CALL DCOUNT(NK, FI)
GOTO 25

C C 2 - ZERO CHROMOSOME COUNT
C
4122,518

C 1398 NK=0
GOTO 25

C C 0 - QUICK COUNT
C
1358 PAR()='TO'
PAR(2)=-1
CALL UPARAM(3, PAR, 2)
CALL APHASE(2)
CALL UPARAM(122, HX, MOBPH)
CALL CLOSE(A)
CALL EXIT
C QUICK COUNT OPTION
C
C OT - SET OBJECT THRESHOLD
C
1400 CALL MVN(PAR, RPAR(HRP+1)+1)
HRP=HRP+1
CALL UPARAM(HRP, RPAR, RCSPH)
REUH=1
GOTO 25
END

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

C 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
C SUBROUTINE MOB
C
COMMON/C1/MOV, IDIR
COMMON/C1/CHDIR, SMLBUF, LRGBUF, NS, NL, CURDLN
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 MOB
C SYSTEM PARAMETERS
INTEGER SPAR(5)
EQUIVALENCE (SPAR(1),NL), (SPAR(2),NS)
C PARAMETER BEY FOR DECODING
INTEGER NKEY, KEY(11)
C ENDPOINT TRACKERS (8 FRACTIONAL BITS)
INTEGER XMIN, XMAX, YMIN, YMAX, XSMIN, XSMAX, YSMIN, YSMAX
C STARTING & ENDING SAMPLE TABLE
INTEGER=2, SST(88), EST(88)
C RANDOM AREAS
C RETURN CODE FROM ORIENTATION SUBROUTINE
INTEGER RCODE
C RADIANS/DEGREE
PEAL RPD
C OPERATOR SPECIFIED CENTROMERE POSITION
INTEGER OPCN(60)
C OPERATOR FLIP FLAG
LOGICAL OPFNPL(68)
C OPERATOR SPECIFIED ROTATION
INTEGER OPROT(60)
C CENTROMERE LOCATION METHOD
INTEGER CLMATH
C SKELETON FLAG
LOGICAL SKFLG
C SKELETON SWITCH
LOGICAL SWK
C SKELETON ROUTINE PARAMETERS
INTEGER SMIN, SKDelt
C PI & PI/2
REAL PI, PIHALF
C DELTA THETA
REAL DELTA
LOGICAL L4
INTEGER SKETH
INTEGER BLK, BS, SL
C CURRENT OUTPUT LINE #
INTEGER CUROLN, BL, CHRENT
C PARAMETER AREA
INTEGER PAR(500)
EQUIVALENCE (PAR, LRGBUF)
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 LBM
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 DIRS
C NUMBER OF LINES IN OBJECT
INTEGER DIRML
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 MAXOA, MINOA
C DEGREES/ ROTATION INCREMENT
C SAMPLE ACCUMULATORS
INTEGER DENA(88), AREA(88), ACC(176)
EQUIVALENCE (DENA(1), ACC(1)), (AREA(1), ACC(09))
C
C TOTAL AREA AND TOTAL DENSITY FOR NORMALIZING
C
INTEGER TAREA, TDEN
C
C ACCUMULATORS FOR CALCULATING CIA AND CID
C
INTEGER CIDEN, CAREA
C
C SPREAD MEASUREMENTS
C
REAL SPID, SAREA, SPLTH
C
C SHORT ARM INDICATOR
C
INTEGER SHRTA
C
C RANDOM AREAS
C
INTEGER CEN, CHP1, S.PER
REAL SPINC
BYTE EM(28)

C REMEASUREMENT CHROMOSOME SELECTOR
LOGICAL SELFLG, SEL(60)

C DATA STATEMENTS

DATA SPINC/2.8125/
DATA MAXCHR/60/
DATA NKEY/11, KEY/'AR', 'RO', 'FL', 'CI', 'LU', 'SK', 'SP', 'SE',
= 'L', 'NW', 'MD'/
DATA SKMIN/15, SKDEL/2/
DATA ID/3, RPD/0.17453, PT/3.14159, PIHALF/1.5708/
DATA FBD/17, HL/2, HS/3, LBW/4/
DATA DELTA/.85236/
DATA SKFLG/.FALSE./
DATA BLKNO/1, BIRMS/2, DIRHL/3, LENGTH/4, IOD/5, CIL/6, CID/7/
DATA CIA/8, CENLIN/9, PERIN/10, AREA/11, P50A/12, CEMETH/13/
DATA OIPFLIP/60* .FALSE., OPCTN/60* .OPROT/60* .
C INITIALIZE 'SPREAD MEASUREMENTS'

SPIOD=0
SPAREA=0
SPLIT=0
CUROLN=8
NC=0

C INITIALIZE CHROMOSOME DIRECTORY

CALL ITIA(0, CHDIR, MAXCHR*CHRENT)
MINOA=30
MAXOA=2080
CMETH=80
SWSK=.FALSE.

C DEFAULT IS LU - NO SKELETON AND WIDTH TO LOCATE CENTROMERE

C RETRIEVE PARAMETERS AND DECODE

CALL RPARAM(NP, PAR, 590, 8)

IF(IP GT NP) GO TO 58
IP=IP+2
DO 12 J=1, NKEY
IF(PAR(IP), EQ, KEY(J)) GO TO (15, 16, 21, 24, 32, 40, 45, 47, 51, 54, 57), J
12 CONTINUE

C INVALID PARAMETER

CALL TYPE('*** PARAMETER ERROR')
GO TO 95

C PARAMETER AR - SET MINIMUM & MAXIMUM OBJECT AREA

MINOA=PAR(IP+2)
MAXOA=PAR(IP+3)
GO TO 10

C PARAMETER ROTA - ROTATE SPECIFIED OBJECT PRESCRIBED AMOUNT

IF(IPH.LT.1 OR IPH.GT.MAXCHR) GO TO 13
K=PAR(IP+3)/SPINC
OPROT(IPN)=KE4
GO TO 10

C PARAMETER FLIP - FLIP SPECIFIED OBJECT

IF(IPH.LT.1 OR IPH.GT.MAXCHR) GO TO 13
C$eSKFLG=.TRUE.

C PARAMETER CLIL - SET CENTROMERIC INDEX
24 IF(IPH.LT.1.OR.IPH.GT.MAXCHR) GO TO 13
OPFLIP(IPH)=FALSE.

25 OPCHM(IPH)=PAR(IP+F3)
GO TO 19

C PARAMETER LW - LOCATE CENTROMERE BY WIDTH
C
32 CLNETH=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 SHOHL=PAR(IP+F2)
SHDEL=PAR(IP+F3)
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 NOBBING ON RCR
C
47 K=PAR(IP+F2)
IF(K.LT.1.OR.K.GT.60) GO TO 13
SELFLG= TRUE.
IP=IP+3
DO 45 M=1,K
L=PAR(IP)
SEL(L) = .TRUE.
CONTINUE
GO TO 10

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

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

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

C OPEN DATA SETS
C
58 CALL AFILF(BUF,5,'UCR','13,'13)
CALL AFILF (BUF,1,'RCR','13,'13)
CALL OPEN(BUF,1024,1,0,'UCR')
CALL OPEN(BUF,1024,1,1,'RCR')
CALL GLABEL (BUF,SPAR,10)
SPAR(I)=125
C PUT MAX NL IN LABEL SO COPY WILL NOT CAUSE F374
CALL PLABEL (BUF,SPAR,BUF(18+I))
C READ IN ROB DIRECTORY
CALL GET(BUF1, IB)
CALL MLV(BUF1(IB+1), NOB, MAXCHR=8+2)
CALL GET(BUF2, IB)
DO 55 I=1, NOB
C NDC(RF(N))=IV2(BUF1(IB+2*4*(I-1)+1))
IF(.NOT. SELFLG) GO TO 60

C READ PREVIOUS ROB RESULTS IF SELECTIVELY RE-MEASURING
CALL GET(BUF1, IB)
CALL MLV(BUF1(IB+3), SPIO, 8)
C NDC=IV2(BUF1(IB+10))
CALL MLV(BUF1(IB+31), CHDIR(1, 1), 2*15+30)
CALL GET(BUF2, IB)
CALL MLV(BUF1(IB+31), CHDIR(1, 31), 2*15+30)
60 CONTINUE

C MAIN PROGRAM LOOP — EXECUTED ONCE FOR EACH OBJECT TO BE MEASURED
DO 500 N=1, NOB
IF(.NOT. SELFLG) GO TO 70
IF(ON NOT. SELFLG) GO TO 500
SPIO=SPIO-CHDIR(10, N)
SPTH=SPTH-CHDIR(LENGTH, N)
SPA=SPA-CHA=CHDIR(AREA, N)
N=NC(I)
CONTINUE

C INITIALIZE STORAGE FOR THE OBJECT
CALL ITIA(B, SST, MANXS)
CALL ITIA(B, EST, MANXS)
CALL ITIA(B, LRGBUF, 90*90/2)
NL=0
NS=0

C READ & PROCESS OBJECT BLOCKS
KK=0
L=0
BL=0
IFD=IDIR(FBW, N)
ILD=IDIR(LBW, N)
MINS=IDIR(MS, N)-1

C READ EACH BLOCK ONE BY ONE
DO 145 BLK=IFB, IB
CALL GET(BUF, BLK, IB)
ASEC=IV2(BUF1(IB+1))
I=IB+3

C PROCESS EACH SEGMENT WITHIN THE BLOCK
DO 148 II=1, HSEG
IF(II.EQ. IV2(BUF1(I))) GO TO 110
II=IV2(BUF1(I))
WL=WL+1
IF(II.GT. 88) GO TO 490
110 BS=IV2(BUF1(I)+2)-MINS
IF(II.LT. 9) GOTO 490

C IGNORE POSSIBLE BUG IN FOB OUTPUT DATA ***************
IF(SST(BL).EQ. 89) SST(BL)=BS
NSAMP=IV2(BUF1(I+4))
IF(NSAMP.GT. 88) GO TO 490
KK=KK+NSAMP
EST(BL)=BS+NSAMP
IF(EST(BL).GT. MAXXS) GO TO 490
C TRANSFER SEGMENT INTO LRGBUF
CALL MVL(BUF(I+6), LRGBUF(BS, AL), HSAMP)
IF(NSAMP-(HSAMP/2)+2 NE 0) HSAMP=HSAMP+1
I=I+HSAMP+6
CONTINUE

C REJECT OBJECT IF IT IS TOO SMALL OR TOO LARGE
IF(KK_GT(MAXA OR KK_LT(MIMA)) GO TO 490

C ORIENT THE OBJECT
CALL ORIG(BL, LRGBUF, SMLBUF, GET, CHDIR(S1.M), AL, DUPQ2(N), RCODE),
NS=CHDIR(DIRNS, N)
NL=CHDIR(DIRNL, N)
IF(RCODE=NE 0) GO TO 490
NS=MC+1

C BUILD_ACCUMULATOR_TABLES
C CALL ACSSUB(SMLBUF, NL, HS, AREA, BENA, TAREA, TDEN)
C BUILD AND ANALYZE THE SKELETON IF LOCATING
C THE CENTROMERE BY DENSITY
METH=1
C ASSUME OPERATOR SUPPLIED CENTROMERE (SET METH=1)
CEN=OPCEN(N)
IF(CEN=NE 0) GOTO 170
C CHECK FOR OPERATOR SUPPLIED CENTROMERE
METH=0
C SET METH=0 FOR AUTOMATIC CENTROMERE
C IF NOT SUBK GO TO 150
CALL SKSUB(SMLBUF, ML, NS, LRGBUF, CEN, BENA, SMIN, SKDLT, CLMETH)
IF(CEN=NE 0) GO TO 250
C THE PREVIOUS STATEMENTS WERE COMMENTED OUT TO MAKE MORE ROOM
C LOCATE CENTROMERE BY ALTERNATE METHOD
C ASSIGN 165 TO LOOP
C LOOK FOR CENTROMERE IN MIDDLE HALF OF IMAGE
NS1=(NS-(NS/2)+1)/2
C FIND MINIMUM

C CONTINUE
NS2=NS-NS1+1
MINA=ACC(NS1+CLMETH)
CEN=NS1
DO 160 1=NS1, NS2
IF(ACC(I+CLMETH)=GE MINA) GO TO 160
MINA=ACC(I+CLMETH)
CEN=I
CONTINUE
GOTO 160
160
CONTINUE
GOTO LOOP
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 CONTINUE
I=NS1+1
MINA=ACC(NS1+CLMETH)
CEN=NS1
DO 165 1=NS1, NS2
IF(CEN=NE 0) AND (CEN=NE 0) GO TO 230
MINA=ACC(I+CLMETH)
CEN=I
CONTINUE
GOTO 165
165
CONTINUE
GOTO LOOP
C ADJUST CENTROMERE AS REQUIRED BY OPERATOR
C ADJUST FOR SHORT ARM ON RIGHT ORIGINALLY
MINA=ACC(CEN+CLMETH)
C LOOK ONE UP AND DOWN TO SEE IF THEY ARE SMALLER
IF(ACC(CEN+1+CLMETH)=LT MINA) GOTO 180

C ADJUST FOR SHORT ARM ON LEFT ORIGINALLY
MINA=ACC(CEN+CLMETH)
C LOOK ONE UP AND DOWN TO SEE IF THEY ARE SMALLER
IF(ACC(CEN+1+CLMETH)=LT MINA) GOTO 180

C LOOK ONE UP AND DOWN TO SEE IF THEY ARE SMALLER
IF(ACC(CEN+1+CLMETH)=LT MINA) GOTO 180

C LOOK ONE UP AND DOWN TO SEE IF THEY ARE SMALLER
IF(ACC(CEN+1+CLMETH)=LT MINA) GOTO 180

C LOOK ONE UP AND DOWN TO SEE IF THEY ARE SMALLER
IF(ACC(CEN+1+CLMETH)=LT MINA) GOTO 180

C LOOK ONE UP AND DOWN TO SEE IF THEY ARE SMALLER
IF(ACC(CEN+1+CLMETH)=LT MINA) GOTO 180
IF ACC(CEN-I+CLMETH).LT.MINA) CEN=CEN-I
GOTO 250

C CHECK FOR DUPLICATE MINIMA AND TAKE AVERAGE IF SO

230 J=0
CEN1=CEN+1
DO 240 I=CEN1,NS2
IF(ACC(I+CLMETH).LT.MINA) GOTO 245
240 J=J+1
245 CEN=CEN+J/2

C MEASURE THE ARMS

250 CALL DPT(SMLBUF,HL,CEN-LB2)
CALL DPT(SMLBUF,HL,L-LD1)
CALL DPT(SMLBUF,HL,NS-LD3)
ALTH=SQRT((CEN-5)**2+(FLOAT(LB2-LB1)**2)
AL2TH=SQRT((NS-CEN+5)**2+(FLOAT(LD3-LB2)**2)
ICR=100.*ALTH/CHRLTH+5
SHRTA=I
IF(OPCEN(N).EQ-.9) CCR10 CCR320
C CALCULATE CID AND CIA

CDEN=0
CAREA=9
DO 270 I=1,CEN
CDEN=CDEN+DENA(I)
270 CAREA=CAREA+AREA(I)
IDR=2*(200.*CDEN-(100.*DENA(CEN)))/2.*TEN
IF(IDR.LT.50) IDR=100-IDR
IF(IDR.GT.99) IDR=99
IAR=2*(200.*CAREA-(100.*AREA(CEN)))/2.*TAREA
IF(IAR.LT.50) IAR=100-IAR
IF(IAR.GT.99) IAR=99

C ADD MEASUREMENTS TO CHROMOSOME DIRECTORY

IF(SHRTA.EQ.-1) CEN=CEN-NS-1
C STORE NEGATIVE CEN TO FLAG SHORT ARM ON RIGHT
CHDIR(CMTHLH)=CEN
CHDIR(LENGTH.H)=CHRLTH+5
CHDIR(AREA.N)=TAREA
CHDIR(CIL.H)=ICMR
CHDIR(CIA.N)=IAR
CHDIR(CID.H)=IDR
FPSDA=(FPSDA+FPSDA)/(TAREA+TAREA)
CHDIR(CPSQA.H)=100.*FPSDA
CHDIR(CMETH.H)=METH

C ADD INDIVIDUAL MEASUREMENTS TO SPREAD MEASUREMENTS

C SPID=SPID+TEN
SAREA=SAREA+TAREA
SPLTH=SPLTH+CHRLTH

C STAND CHROMOSOME UP AND WRITE IT INTO ROTATED FILE

CIP=NS
LPE=1924/HL
CHDIR(BLKNH.N)=CURLN
CALL PUT(OBUF, CUROLN, IB)
CUROLN = CUROLN + 1
IF (SKFLG) GO TO 305
CALL CHROUT(SMLBUF, OBUF(IB+1), NS, ML, LPB, IP, SHRTA, 90)
GO TO 310
CALL CHROUT(LRGBUF, OBUF(IB+1), NS, ML, LPB, IP, SHRTA, 90)
CONTINUE
IF (IP .NE. 0) GO TO 300
GO TO 500

C OBJECT BYPASSED
C
CONTINUE
CHDIR(BLKNM, N) = 0
CALL OUTCON(N, EM(7), 2)
CALL OUTCON(KK, EM(22), 4)
CALL OUTCON(NS, EM(25), 2)
CALL TYPE(EM, 20)

C END OF MAIN LOOP
C
UPDATE CHROMOSOME DIRECTORY ON DISC
C
CALL PUT(OBUF, 1, IB)
CALL ITL2(NOB, OBUF(IB+1))
CALL ITL2(CUROLN, OBUF(IB+15))
CALL ML(C(SPIOD), OBUF(IB+3), 12)
CALL ML(CHDIR(1,1), OBUF(IB+31), 2*30+15)
CALL PUT(OBUF, 2, IB)
CALL ML(CHDIR(1,31), OBUF(IB+31), 2*30+15)
CALL CLOSE(OBUF)
CALL CLOSE!BUF)

C MEASUREMENTS COMPLETED
C
CONTINUE
CALL EXIT

C **********
C* CLASSFY =
C* Compile with /CG.25
C**********
CCLASSFY PROVIDES THE CLASSIFICATION MECHANISM FOR THE CLINICLA
C ALMS SYSTEM. ITS INPUT IS THE CHROMOSOME DIRECTORY PRODUCED BY MODB
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, ICT,C1G, OFG, ICTB, ILTB
COMMON/C1/MOB, SPIOD, SPLTH, SPAREA, CHDIR, 10BUF
COMMON/C1/MSO, NFLAG, G10, S134, ISR, OFG, SCLT, GCS
C COMMON REGION
C
NUMBER OF CHROMOSOMES
C
INTEGER MOD
C
CHROMOSOME DIRECTORY
C
INTEGER CHDIR(15, 60)
C
FORMAT OF DIRECTORY
C
BLOCK NO OF FIRST BLOCK OF AN OBJECT
C
25 CALL OUTCON(IP, PENV(22), 3)
CALL QPRINT(PENV = 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) GO TO 25
   DO 1000 I = 1, NGT2
     C (CT(I)) = A
       IP = IP + 3
1010 IF (IP.GT.NP) GO TO 70
   DO 1020 J = 1, MC
     IF (PAR(IP) .EQ. CHAM(J)) GO TO 1030
1020 CONTINUE
   GO TO 10
1030 JJ = J + J
   CT (JJ, IL) = PAR(IP + 3)
   CT (JJ - 1, IL) = PAR(IP + 2)
   IP = IP + 4
   GO TO 10
C
C PARAMETER NH - NO OBJECT NUMBERS OR CENTROMERE FLAG
C
1110 MFLAG = 1
   IP = IP + 2
   GO TO 10
C
C PARAMETER SD - SET # OF OUTPUT SAMPLES
C
1280 NSD = PAR(IP + 2)
   NSDF = 1
1283 IP = IP + 3
   GO TO 10
C
C PARAMETER SL - SET SLOT 1D'S
C
1380 DO 1310 I = 1, 90
1310 SID4(I) = PAR(IP + 1 + 1)
   IP = IP + 4
   GO TO 10
C
C PARAMETER MA - MALE KARYOTYPE
C
1400 SEX = 1
1410 IP = IP + 2
   GO TO 10
C
C PARAMETER FE - FEMALE KARYOTYPE
C
1580 SEX = 2
   GO TO 1410
C
C PARAMETER NX - NO X SEPARATION FROM C-GROUP
C
1510 NOX = TRUE.
   CALL MVL (' B B ( C + X )
             ( D B E-16 E E
             2 F F C C ', SID4(13), 112)
   GO TO 1410
C
C PARAMETER DE - DEBUG OPTION ON
C
1520 DEBUG = TRUE.
   GO TO 1410
C
C PARAMETER SZ - SET SLOT SIZE
C
1530 SLSZ = PAR(IP + 2)
   GO TO 1293
C
C PARAMETER HN - MOVE CHROMOSOMES AROUND FROM PREVIOUS KARYOTYPE
C
1540 IF (NSMOV.LT.58) NSMOV = NSMOV + 1
   OLD8(NSMOV) = PAR(IP + 2)
   ... HEMB(NSMOV) = PAR(IP + 3)
C PARAMETER ST - SET SLOPE TABLE
C
2 DO 2000 I=1,NGT2M1
2000 ST(I)=PAR(IP+1+1)+PAR(IP+2+1)
    IP=IP+NGT2M1+2
COTO 10
C
C PARAMETER NL - SET MINIMUM # OF CHROMS. FOR EACH GROUP
C
3 DO 3300 I=1,NG
3300 NLING(I)=PAR(IP+1+1)
    IP=IP+NG2
    COTO 10
C
C PARAMETER MX - SET MAXIMUM # OF CHROMS. FOR EACH GROUP
C
4 DO 4800 I=1,NG
4800 MAXG(I)=PAR(IP+1+1)
    IP=IP+NG2
    COTO 10
C
C PARAMETER CS - SET CENTER SAMPLES FOR EACH SLOT
C
5 DO 5900 I=1,998
5900 SCS(I)=PAR(IP+1+1)
    SCSCCLC=I
    IP=IP+92
    COTO 10
C
C PARAMETER CL - SET CENTER LINE FOR EACH GROUP
C
6 DO 6000 I=1,5
6000 SCL(I)=PAR(IP+1+1)
    IP=IP+7
    COTO 10
C
C PARAMETER IR - SET INITIAL SLOT FOR EACH ROW
C
7 DO 7000 I=1,6
7000 ISR(I)=PAR(IP+1+1)
    IP=IP+92
    COTO 10
C
C PARAMETER IG - SET INITIAL SLOT FOR EACH GROUP
C
8 DO 8000 I=1,NG2
8000 IST(I)=PAR(IP+1+1)
    IP=IP+NGC+2
    COTO 10
C
C PARAMETER CS - MANUALLY INSERT CHROMOSOMES INTO SLOTS
C
9 OFSF=1
    NOFS=NP-IP-1
    IF(NOFS, NE, 48, AND, NOFS, NE, 46). COTO 0950.
C SEE IF 48 OR 46 OFS PARAMETERS WERE USED
    DO 8950 1=1,NGFS
8950 OFS(XTBL(I))=PAR(IP+1+1)
C ALLOW FOR THE EXTRA SLOTS
    COTO 78
    8950 CONTINUE
    DO 9800 I=1,998
    9800 OFS(I)=PAR(IP+1+1)
        IF=IP+92
        COTO 10
C
C PARAMETER CO - ADJUST COEFFICIENTS
C
15 DO 71 I=1,6
    CTHE(I)=PAR(IP+1+1)
167 CONTINUE
IF(CTAB(1)+CTAB(2)+CTAB(3).NE.100) GO TO 990
IF(CTAB(4)+CTAB(5)+CTAB(6).NE.100) GO TO 990
GO TO 10

990 CALL TYPE('0')
CALL TYPE('COEFFICIENT SUM NOT = 100')
GO TO 990

C READ IN CHROMOSOME DIRECTORY

70 CALL AFIL(FIOBUF,1,RCR,13,13)
CALL OPENBUF(IOBUF,1024,1,0,RCR)
CALL GLABEL(IOBUF,SPAR.1B)
LSEX=0
IF(IJOBUF(B+14).EQ.1) LSEX=1
IF(IJOBUF(B+14).EQ.1) LSEX=2
IF(IJOBUF(B+18).EQ.1) LSEX=3
GOTO 777 IF NOT SOURCE 9, 19, 29, ... (MONKEY)
CALL MVU(MCT,CT,1820)
C REPLACE HUMAN TABLES WITH MONKEY TABLES FOR SOURCE 9
CALL MVU(MAL,AC, 10)
C SET C=0 IF GROUP FOR POSSIBLE 1
777 CONTINUE
CALL GET(IJOBUF,1,1B)
CALL MVU(IOBUF,B+11,NOB,16)
ALL=12(IJOBUF(B+17))
SP=<SPLTH+44/NC
SPID=SPID*46/NC
SPARE=SPARE*46/NC
CALL MVU(IJOBUF(B+13),CHDIR(1,1),15*80)
CALL GET(IJOBUF,2,1B)
CALL MVU(IOBUF(B+13),CHDIR(1,1),15*80)
DO 71 2=1,NOB
CHDIR(LENGTH,1)=CHDIR(LENGTH,1)+80000./SPLTH
CHDIR(100,1)=CHDIR(100,1)+80000./SPID
CHDIR(area,1)=CHDIR(area,1)+80000./SPARE
71 CONTINUE
C INITIALIZE CURRENT INDEX AND INITIAL INDEX FOR EACH GROUP
C
CICG(1)=0
CICG(1)=IC(1-1)+25
C 75 CICG(1)=IIC(1-1)+25
C
C SET UP SLOT SIZE IF NOT INPUT
C
IF(SLSI2.EQ.0) SLSI2=24
IF(SLSI2.GT.28) SLSI2=28
IF(SLSI2.LT.12) SLSI2=12
PSEP=SLSI2/3
C PAIR SEPARATION
SC(1)=RSEP+SLSI2/2
SC(1)=SC(1)+SLSI2
DO 752 I=1,17,2
SC(1)=SC(1)+SLSI2
DO 752 I=1,17,2
DO 752 J=1,1
754 SC(18+J)=SC(1)
756 CONTINUE
750 CONTINUE
C
C CHECK SEX
C
IF(SEX.GT.0) GOTO 76
C SEX=LSEX
C 76 IF(SEX.NE.1) GOTO 77
C MALE
MAXG(5)=15
MINC(10)=M INC(10)+1
169
4,122,518

77 IF(SEX.NE.2) GOTO 79
C FEMALE

HNG(5)=16
MNXC(10)=MAXC(10)-1
IF(SID4(70).EQ.YID) SID4(70)=BID
C CHANGE Y TO BLANK UNLESS SID WAS CHANGED BY USER
78 CONTINUE
IF(OFSF.EQ.1) GOTO 403
C
C SKIP CLASSIFICATION IF OFS TABLE IS INPUT .
C
C DO INITIAL ASSIGNMENTS OF CHROMOSOMES TO GROUPS
C
DO 200 N=1,NOB
IF(CHDIR(BLBK0,N).EQ.0) GO TO 200
I=(CHDIR(CIL,N)*CTAB(4)+CHDIR(CID,N)*CTAB(5)+CHDIR(CIA,N)
I*CTAB(6))/100
IL=((FLOAT(CHDIR(LENCM,N)))*CTAB(1)+FLOAT(CHDIR(LEN2.N))
1+CTAB(2)+FLOAT(CHDIR(AREA.N))/CTAB(3))/1000.+1.
IF(CIL.GT.51).IL=51
CALL OUTCON(N,ICIL,N,-2)
CALL OUTCON(IC1,ICIL(-2),-2)
CALL OUTCON(IL,ICIL(Il),-2)
ICIL(N)=IC
ILTB(N)=-IL
IF(CIL.EQ.0.OR.CHDIR(PSQDA,N).LT.1349.OR.CHDIR(AREA.N).GT.699)
100 TO 82
C PUT IN UNKNOWN GROUP IF BLOB OR OVERLAP
DO 80 J=1,NHT2HI,2
IF(CIL.GE.CT(J,IL) AND IC1.LE.CT(J,IL)) GOTO 100
C IF IC IS WITHIN THE GROUP LIMITS FOR THIS VALUE OF IL GOTO 100
80 CONTINUE
82 J=IC1;
C OBJECT IS IN THE FORBIDDEN ZONE
GOTO 110
85 CONTINUE
CALL TYPE(' ALL SLOTS FULL')
GOTO 200
100 CONTINUE
J=(J+1)/2
110 CONTINUE
CALL OUTCON(J,ICIL(26),-2)
IF(NOC(J).LE.25) GOTO 85
NOC(J)=NOC(J)+1
CIC(J)=CIC(J)+1
OFG(CIC(J))=N
200 CONTINUE
C
C IF ANY GROUP IS HEAVY OR LIGHT, TRY TO MOVE CHROMOSOMES AROUND
C
210 HFAIL=0
220 CONTINUE
KH=0
NM=0
DO 300 J=2,NG
IF(NOC(J).GE.MAXC(J)) GOTO 300
K=J-1
HKG=0
HMIN=199999
C GROUP J IS LIGHT, SEE IF A PRECEDING GROUP IS HEAVY
DO 2500 K=1,K2
IF(NOC(K).LE.HING(K)) GOTO 2500
C GROUP K IS HEAVY
IF(HFAIL.EQ.0) GOTO 230
DO 225 N=1,HFAIL
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=I1G(K)
I2=I2G(K)
DO 2400 I=11,12
H=OFG(I)
IC=ICTBC(N)
IL=ILTBC(N)
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
MAX=(((IC-AC(K))**2+(IL-AL(K))**2)**2)**1
IF(D.GE.MAX) GOTO 2460
C THIS IS THE MOST LIKELY CANDIDATE SO FAR
KSAV=K
IASAV=I
NCAN=N
DRIN=D
2460 CONTINUE
2500 CONTINUE
IF(NCAN.EQ.0) GOTO 3000
HM=HM+1
C MOVE OBJECT N FROM GROUP KSAS TO GROUP J
CALL OUTCON(KSAV,MMSG(22),3)
CALL OUTCON(J,JMSG(29),3)
IF(DEBUG) CALL OPRINT(MMSG,29)
NOG(J)=NOG(J)+1
CIG(J)=CIG(J)+1
DEGJJG(J)=NCAV
CIG(KSAV)=CIG(KSAV)-1
NOG(KSAV)=NOG(KSAV)-1
II=2*CIG(KSAV)
DO 2600 I=IASAV,12
2600 OKEG(I)=OFG(I+1)
C CLOSE UP THE REMAINING OBJECTS IN THE GROUP
3000 CONTINUE
IF(HMOV.GT.0) GOTO 220
C IF SOMETHING WAS MOVED, SEE IF MORE MOVES ARE POSSIBLE
IF(HN.EQ.0) GOTO 3100
C KH IS HEAVY AND JL IS LIGHT BUT NO SIMPLE MOVE WAS POSSIBLE
C SEE IF THERE IS A COMPLEX (MULTI-LEVEL) MOVE THAT CAN BE MADE
L=1
JG(J)=JL
3010 KG(J)=KH
3020 CALL CFO(KG(J),JG(L),IX,AC,AL,CT)
C CHECK FOR A CHROMOSOME IN THE KG-JG OVERLAP AREA
IF(IX.EQ.0) GOTO 3040
IXG(J)=IX
3040 CALL CFO(KG(J),IXG(J),KH) GOTO 3060
C WE NEED TO GO UP ONE LEVEL
L=L+1
JG(J)=KG(L-1)
GOTO 3010
3040 KG(J)=KG(J)+1
C LOOK AT THE NEXT GROUP ON THIS LEVEL
IF(KG(J).LT.JG(L)) GOTO 3020
L=L-1
C GO DOWN ONE LEVEL
IF(JG(L).GT.0) GOTO 3040
C FAILED TO FIND A GOOD MOVE
NFAIL=NFAIL+1
IF(NFAIL.GT.10) GOTO 3100
KFAIL=NFAIL
JFAIL=J
CALL OUTCON(KH,FMSG(22),3)
CALL OUTCON(JL,FMSG(29),3)
IF(DEBUG) CALL OPRINT(FMSG,36)
GOTO 220
3060 CONTINUE
C MAKE THE L MOVES THAT HAVE BEEN FOUND
DO 3000 I=1,L
IASAV=IC(JM)
JG(M)=JM
NOG(JG(M))=NOG(JG(M))+1
CIG(JG(M))=CIG(JG(M))
173

FOC(CIG(JCM))=FOC(ISAY)
CIG(KCM)=CIG(KCM)-1
HOG(KCM)=HOG(KCM)-1
I2=CIG(KCM)
DO 3070 I=ISAY+1
3070 FOG(I)=FOG(I+1)
3080 CONTINUE
CALL OUTCON(KH,SMSC(22),3)
CALL OUTCON(JL,SMSC(23),3)
IF (DEBUG) CALL QPRINT(SMSC,39)
GOTO 218
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,HGI
IF (HOG(J)<EQ.0) GOTO 300
I1=IC(J)
I2=CIG(J)
DO 240 I=II,12
H=FOC(I)
I2=ICTB(H)
IL=ILT(B(H)
YINT(I)=IC-ST(J)*IL
IF (ST(J).LT.0) YINT(I)=-YINT(I)
240 CONTINUE
C STORE Y INTERCEPT
I1=IC(J)
C
242 CONTINUE
L2=IST(J)+HOG(J)-1
IF (J<LT.1) GOTO 246
HOG(J)=HOG(J)-1
IF (HOG(HGI)).LT.1ST(HG2).LT.1ST(HG1)) GOTO 244
CALL SSQCHG(J,ISWJ)
IF (ISWI.EQ.1) CALL TYPE(' ALL SLOTS FULL')
GOTO 242
244 CONTINUE
CIC(HG1)+CIC(HG1)-1
HOG(HG1)=HOG(HG1)+1
FOC(CIC(HG1))=FOC(I2)
I2=12-1
GOTO 242
246 CONTINUE
DO 260 L=L1,L2
MINY=9999
DO 258 I=II,12
IF (YINT(I)).GE.MINY) GOTO 259
MINY=YINT(I)
MINI=I
259 CONTINUE
OFS(L)=FOC(MINI)
YINT(MINI)=MINY
25A CONTINUE
C STORE THE GROUP IN ORDER OF YINT
300 CONTINUE
310 CONTINUE
C
C NOW MAKE ANY OTHER ADJUSTMENTS REQUIRED
C
C IF(NXO). GOTO 358
C CHECK FOR NO X SEPARATION FROM C GROUP DESIRED
C FIND THE X OR X-X IN GROUP C
I1=IST(S)
I2=II+15
I3=II+2
I4=II-2
IF (HOG(S)).EQ.16) GOTO 330
IF (HOG(S)).NE.15) GOTO 358
C C HAS 15. THIRD LARGEST IS X
OFS(I2)+OFS(I3)
DO 320 I=I3,14
320 OFS(I)=OFS(I+1)
OFS(I2-1)=0
GO TO 350
CONTINUE
C C HR = 16. J AND 4 ARE X-X
ITEM=0$S(I3)
ITEM=0$S(I3)
DO C44=1,I3,I4
340 OFS(1)=OFS(1)+2
OFS(I3(I3(I3))-ITEM
OFS(I3(I3(I3)))=ITEM
350 CONTINUE
IF(NOG(NG),NE,5.OR.SEX.EQ.2)GO TO 480
C IF 5 IN GROUP C, FIND THE Y CHROMOSOME
I=1ST(NG)
I=I+4
IF(OFS(I3(I3(I3(I3(I3(I3)))))),GO TO 403
MAX=1
DO 360 I=1,11,12
N=OFS(I)
M=$CHDIR(ARE.X(N)/CHDIR(DIRNL(N))*CHDIR(DIRMS(N)))
IF(M.LE.MAX)GO TO 360
MAX=M
360 CONTINUE
I=I2-1
DO 370 I=MAX,I4
370 OFS(I)=OFS(I)*1
OFS(I3)=0
OFS(I3(I3))+NMAX
CONTINUE
IF(NDSX=20,11)GO TO 493
DO 402 N=1.HSMOV
NEWSN=NEWSN(N)
OLDSN=OLDSN(N)
IF(NEWSN(NEWS)GO TO 480
C DELETE OBJECT IN OLDSN IF NEWSN = 0
OFS(OLDSN)=0
GO TO 482
490 CONTINUE
NSAVE=OFS(NEWSN)
OFS(NEWSN)=OFS(OLDSN)
OFS(OLDSN)=NSAVE
492 CONTINUE
493 CONTINUE
C ADJUST TABLE PRO-OBJECTS WIDER THAN 24 SAMPLES
DO 410 J=1,NGL
N=1ST(J)
I=I+4
J=I3(I3(I3(I3(I3(I3))))
DO 405 J=1,1,12
SCS(J)=SCS(J)+1
C ADD ON ANY PREVIOUS DELTA
INP=OFS(I)
IF(INP.EQ.0)GO TO 404
GSTM24=$CHDIR(DIRNS,INP)-SLSIZ
GSTM24(LES,0)GO TO 404
GSTM24(ENGLE)+GSTM24(ENGLE)
C ADD DELTA FOR THIS OBJECT
INP=INP+GSTM24
404 IF(SCS(I),GT.586)SCS(I)=586
405 CONTINUE
400 CONTINUE
IF(NOGF.EQ.1.AND.NSNO.EQ.0)GO TO 425
C IF NSO OF 0 WAS SPECIFIED, DO NOT CHANGE IT
C NOW FIND NSO
DO 420 J=1,6
I=1ST(J)
INP=OFS(I)
IF(INP.EQ.0)GO TO 415
C THIS IS THE LAST NON-VACANT SLOT ON THE ROW
ITEM=SCS(I)+SCS(I)-1)+SLSIZ/2+FSER
IF(ITEM.GT.NSNO)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=1,5
      ISRI=ISR(I).
      DO 450 J=1,15
         J=ISR(J-1)
         TNP=OFF(J)
         IF(JJ.GT.TMPR(I)+1) TNP=0
         IF(INP.EQ.0) GO TO 450
         ITEMP=CHAIN(DIRNL(INP)/2+28)
         IF(SCLT(I).LT.ITEMP) SCLT(I)=ITEMP
   450 CONTINUE
500 CONTINUE
   CALL PUT(IOBUF,3,18)
   CALL VML(NSO,IOBUF(IB+1),2*294)
   CALL CLOSE(IOBUF)
C
C  RETURN
C
   CALL EXIT
C
END

C

*/
C  KTYPE PRODUCES THE OUTPUT KARYOGRAM FROM THE CLASSIFICATION TABLES
C PRODUCED BY CLASSFY AND THE ROTATED CHROMOSOME IMAGES
C PRODUCED BY ORIENT

SUBROUTINE KTYPE
COMMON/C1/INVB,IOBUF,OBUF,MLT,HST,FNM,CENLIN,CEMETH
COMMON/C1/NSO,NFLAG,GID,SID,ISR,OFF,SCLT,SCS
INTEGER SPAR(10),CEMETH(40)
BYTE INV(56)
INTEGER ISR(45),GID(10),SID(98),I(39)
BYTE OBUF(3128),IOBUF(20480)
INTEGER FNM(56),HST(10),MLT(10),CENLIN(60),OFF(91),SCS(98)
BYTE LABEL(216)
INTEGER SCLT(5)
CALL VML('O=...***...***...***...***...',R,78)
C
C  OPEN DATA SETS
C
   CALL DCLEAR
C CLEAR GRAY SCALE
   CALL AFILE(INVB,1,'RCR'/*13,'fill'*/)
   CALL AFILE(OBUF,5,'PIC'/*5,5'*/)
   CALL OPEN(INVB,1924,9,'RCR')
   CALL OPEN(IOBUF,1936,11,'KCM')
   CALL GLABEL(INVB,SPAR,1B)
   CALL VML(INVB(IB+1),LABEL,216)
   NLO=0
C
C  READ CHROMOSOME DIRECTORY & BUILD TABLES
C
   CALL READ(INVB,1,IB,IOBUF(1))
   HSF=IV2(IOBUF(1))
   DO IB=1,15
      HSF=IV2(IOBUF(IB))
   NLO=0
   MLO=0
   FNM(1)=IV2(IOBUF(1+30))
   MLT(1)=IV2(IOBUF(3+30))
   HSF=IV2(IOBUF(5+30))
   CENLIN(1)=ABS(IV2(IOBUF(17+30)))
   CONTINUE
      INPUT CONTINUE
CEWETH(I)=IY~IBUF<2Jt3~*1))

CONTINUE

CALL READ(1MBB.18.1BUF(1)))

DO 26 I=1,70
FBN(I+30)=IY2(IBUF(1+28*1))
MLY(I+30)=IY2(IBUF(3+28*1))
NST(I+30)=IY2(IBUF(5+28*1))
CEMN(I+30)=TABS(IY2(IBUF(17+30*1)))
CEWETH(I+30)=IY(IBUF(25+30*1))

CONTINUE

C READ IN CLASSIFICATION TABLES

CONTINUE

CALL READ(1MBB.3.IB.1BUF(1)))
CALL MLV(IBUF(1),HSG.2+294)
IF(NSG.GT.512) GO TO 980
SPAR(I)=512
SPAR(2)=HSG
SPAR(3)=512
CALL PLABEL(GBUF.SPAR.LABEL)

C MAIN LOOP:
C BUILD EACH ROW OF THE OUTPUT KARYOCRAN

DO 500 I=1,5
ISR=ISR(I)
JFI=LT.5) GOTO 420
C SEE IF THERE ARE ANY OBJECTS ON ROW 5
DO 420 J=1,10
IF(OF$ISRI+J-1).NE.0) GOTO 430
CONTINUE

GOTO 500

C NO OBJECTS ON ROW 5
CONTINUE

R(2)=GID(I+(I-1)*2)
R(3)=GID(2+(I-1)*2)
DO 450 J=1,10
J=ISR(I)+J-1
INF=GS(IJ)
IF(JJ.GE.ISRI(I))) INF=0
CALL QUICK(2,INF,R(5+(J-1)*2).-2)
IF(INP.EQ.0) CALL ITL(32.R(5+(J-1)*2))
CONTINUE

CALL SSUICH(1,IBIT)
IF(IBM.HE.1) GO TO 475
CALL TYPE(R.78)

C INVOKE THE ROW BUILDING ROUTINE

CONTINUE

CALL KROW(ISRI(I)+1)-ISRI..OFSCISRI).SCLT(I).SEC(ISRI).HSG.
.ISIDCISRI).NFLAG)
NCG=NLG.2*SCLT(I)

CONTINUE

C CLOSE UP DATA SETS AND RETURN

SPAR(I)=NLG
CALL PLABLEL(GBUF.SPAR.LABEL)
CALL CLOSE(INVB)
CALL CLOSE(GBUF)
CALL EXIT

C ERROR

CALL TYPE(' *** HSG TOO BIG')
CALL TYPE(' ** ')
CALL EXIT

END
**Routine Description**

Routine `KROW` constructs a row of the output karyogram. It is invoked by the `KTYPE` routine.

**Variables and Parameters**

- `NSL`: Number of slots in the current line.
- `OFS`: Offset of the current slot.
- `SCS`: Number of objects in the current slot.
- `NSO`: Number of slots in the current row.
- `SID`: Side information.
- `MFLAC`: Marked for linear access.
- `CEMLIN`: Column length.
- `CEMETH`: Column method.
- `BYTE CHAR(4)`
- `BYTE IMBV(56)`
- `INTEGER IDTI(60)`
- `INTEGER OFS(91), SCS(98), MLT(60), NST(60), FBN(60)`
- `BYTE OBUF(3128), IBUF(20480), LBO, LFF`
- `INTEGER RECSIZ, MAXNSL`...and more

**Main Routine Loop**

1. **Slot Loop**
   - Processes each slot within the current line.
   - `DO 158 I=1, NSL, 1`
   - `CALL PUT(OBUF, 0, IO)`
   - `CALL ZIA(OFU(10+1), NSO4)`
2. **Slot Loop**
   - Processes each slot within the current line.
   - `DO 159 I=1, NSL
   - OFFS(I)
   - NF=OFFS(I+1)
   - IF(NF.EQ.0) GO TO 80`
3. **Check for First Slot of a Pair Empty, But Second Slot Full**
   - `IF(NSL.EQ.0) GO TO 100`
   - `GO TO 130`
4. **Internal Loop**
   - `CONTINUE
   - MLT=MLT(N)
   - NL2=MLT/2
   - IF(L.LT.SCL-M2 .OR. L.GE.SCL+NL2) GO TO 128
   - L0=L-SCL+NL2`
5. **Calculate Initial Sample for Object on Output Line**
   - `HSIN=NST(N)
   - HS2=HSIN/2
   - IF(SCS(I)+HS2.GT.511) SCS(I)=511-HS2
   - IS=SCS(I)-HS2`
6. **Get First Block of the Object**
   - `IB=EBN(N)
   - IF(IB.EQ.0) GO TO 120`
C BYPASS OBJECT IF NOT IN DIRECTORY.

LPB=RECSIZ/MSTN
MLD=MOD(LO,LPB).
IF(MLD GT 0) GO TO 98

CALL READ(NMB,IA+LO/LPB,IA,IBUF(IDT(I)))

98 J=IDT(I)+MLD+MSTN

C TRANSFER OBJECT SEGMENT INTO THE OUTPUT BUFFER

CALL NVL(IBUF(J),OBUF(IO+IS),MSTN)
IS=IS+MSTN

C SEE IF THIS IS THE CENTROMERE LINE

C IF(NFLAG.NE.0) GO TO 150
IF(LO.NE.CENLIN(N)-1) GO TO 150

C CENTROMERE LINE, FIND WHERE TO PUT THE CENTROMERE ARROWS

IT=IS-MSTN
ITMAX=IS-1
DO 110 IU=IT,ITMAX
IF(OBUF(IO+IU).GT.00) GO TO 112

110 CONTINUE
GO TO 117.

112 CONTINUE
IT=IS
115 IT=IT-1
IF(OBUF(IO+IT).EQ.100) GO TO 119

117 CONTINUE
GO 118 K=3,5
OBUF(IO+IU+K)=LFF
118 OBUF(IO+IT+K)=LFF
GO TO 150

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 0, SO PUT NUMBER INTO OUTPUT BUFFER

LO=L-SCL-ML2+9
IS=SCS(I)-11
IF(IS.GT.500) IS=500
IF(LH.LT.10) IS=IS-3

IF(CEMETH(N).EQ.1) NN=N
C FLAG OPERATOR CORRECTED CENTROMERES
CALL OUTCOM(NN,CHAR<3>,3)
CALL TEXT(CHAR<3>,LO,OBUF(IO+IS),1)
GO TO 150

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=NL(NL1)/2
IF(NL3.GT.NL2) NL2=NL3

140 CONTINUE
IF(L.SCL+NL2+5.OR.L.GT.SCL+NL2+10) GO TO 150

C LINE CONTAINS GROUP ID, SO OUTPUT A LINE OF THE ID TO THE OUTPUTBUF

LO=(L-SCL-ML2-5)/2
IS=(SCS(I)+SCS(I+1))/4)*2-23
IF(IS.GT.64) IS=64
CALL TEXT(IDH(N),4,LO,OBUF(IO+IS)-23

150 CONTINUE
IF(LH.LT.1824) CALL DLINE(OBUF(IO+1),Y,Y,NO.REPL-0)

200 CONTINUE
RETURN
C BUFFER TOO SMALL TO HANDLE THE ROW

900  CONTINUE
     CALL TYPE( '...', BUFFER_TOO_SMALL_FOR_ROW')
     CALL TYPE( 'B')
     RETURN

C NSD=0

910  CONTINUE
     CALL TYPE( '*** NO=')
     CALL TYPE( 'B')
     RETURN

END

SUBROUTINE INT2

**********************************
C
C* INT2 *
C*
C**
C
C THIS ROUTINE IS THE SECOND OPERATOR INTERACTION PHASE. IT
C ALLOWS THE OPERATOR TO EITHER APPROVE THE KARYOTYPE WHICH IS
C BEING DISPLAYED ON THE Gray SCALE DEVICE, FOR HARD COPY OUTPUT
C OR ELSE MAKE CHANGES VIA Mob AND/OR CLASYFY/KTYPE PARAMETERS.
C
C COMMON REGION:
C COMMON/CH/N50, NFLAG, C10, S1D, ISD, OFS, SQLT, SCS
C COMMON/CHND/CHDIR, NOB, SPTOS, SPLIT, SPAREA, CURCOLN, NC
C I/O BUFFER
C  BYTE I0GUEF(2124)
C Mob PARAMETERS AND CURRENT PARAMETER HIGH WATER MARK
C  INTEGER MobPAR(538), MobPAR(4)
C Clasyfi/Ktype PARAMETERS AND CURRENT PARAM. HIGH WATER MARK
C  INTEGER KTypePAR(538), K
C INT2 LOCAL PARAMETER BUFFER AND POINTER
C  INTEGER PAR(100), IF
C Re-mob CHROMOSOME SELECTOR
C LOGICAL SEL(60)
C CLASSIFICATION TABLES
C  INTEGER CURDEN
C
C CHROMOSOME DIRECTORY AND ITS FORMAT
C
C  INTEGER CHDIR(15, 60)
C
C BLOCK
C  INTEGER BLKHD
C NUMBER OF LINES IN OBJECT
C  INTEGER DIHR
C NUMBER OF SAMPLES IN OBJECT
C  INTEGER DIRS
C LENGTH
C  INTEGER LENGTH
C INTERGREATED OPTICAL DENSITY
C  INTEGER IOO
C CENTROMERIC INDICES
C  INTEGER CIL, CID, CIA
C CENTROMERE LINE
C  INTEGER CENLIN
C PERIMERE
C  INTEGER PERIM
C AREA
C  INTEGER AREA
C PERIM SQUARED DIVIDED BY AREA
C  INTEGER PSQDA
C CENTROMERE LOCATION METHOD
C  INTEGER CMETH
C PHASE NUMBERS FOR MOB AND FOR ClASYFY/KTYPE
C  INTEGER MobPAR, KypPar, F0SPHA, BANDPH, FOURPH
C Re-run Flags
C  LOGICAL MobFLG, KypFLG, ALLFLG, USFLG
C Random Areas
C  INTEGER SLCL, SLCS, SLID, S, Y, X, SL1, SL2
INTEGER SPAR(18), SST(513), NST(513), S1, S2
INTEGER KEY(37)
BYTE LABEL(73), BLACK(72), PARBUF(90), NOBUF(568)

CLASSIFICATION TABLES

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

DATA STATEMENTS

DATA BLKNO/1/, DIRNH/2/, DIRMS/3/, LENGTH/4/, 10B/5/, CIL/6/, CID/7/
1.CHA/8/, CENLIN/9/, PERIM/10/, AREA/11/, PSQDA/12/, CENETH/13/
DATA MAXPAR/59/, FOSPHA/6/, MOBPHA/6/, KTPH/9/, GANDPH/14/
DATA DECR/37./2958/, FORURP/15/
DATA LABEL/73*/' ', BLACK/72*127*

INITIALIZE CHROMOSOME DATA SET

CALL TYPE(' CHECK KARYOTYPE ', 0)
CALL RCA

READ CURSOR ADJUSTMENTS

CALL AFLE(IDBUF, 1), ' RCR '*, "13", "13"
CALL OPEN(IDBUF, 1024, 1, ' RCR ')
CALL GLABEL(IDBUF, SPAR, 1)

CALL NIO(IDBUF, 1B + 172), LABEL(2), 72

READ IN DIRECTORY AND CLASSIFICATION TABLES

CALL GET(IDBUF, 1, 1B)
CALL NIO(IDBUF, 1B + 1, NOB, 10)
CALL NIO(IDBUF, 1B + 1), CHD(1, 1), 215*30
CALL GET(IDBUF, 2, 1B)
CALL NIO(IDBUF, 1B + 1), CHD(1, 3), 215*30
CALL GET(IDBUF, 3, 1B)
CALL NIO(IDBUF, 1B + 1), NSO, 2*294

READ PREVIOUS PARAMETERS

CALL RPARAM(M, MOBPAR, MAXPAR, MOBPHA)
CALL RPARAM(K, KTPAR, MAXPAR, KTPH)
IF (K.LT.92) GOTO 65
K = K - 92
OSFLG. = TRUE.

DO NOT RECLASSIFY IF OSF PARAMETERS ARE PRESENT
ERASE OLD OSF PARAMETERS, IF PRESENT

65 IF (M.EQ.0) GOTO 100
DO 70 I = 1, M
IF (MOBPAR(I).EQ. 'SE') GOTO 75
70 CONTINUE
GOTO 100
C REMOVE OLD SE PARAMETERS
M = 1 - 1
C REQUEST OPERATOR INTERACTION

100 CONTINUE

CALL RPARAM(NP, PAR, 100, PARBUF)
IF (NP.EQ.0) GOTO 500
IP = 1

CODE PARAMETERS

DO 200 J = 1, NKEY
IF (PAR(J).EQ. KEY(J)) GOTO (1009, 1059, 908, 910, 950, 1250, 1550, 1359
1, 1450, 1508, 1558, 1650, 1758, 1800, 1858, 1950, 2000, 2058, 2125, 2150
2, 2250, 2250, 2350, 2450, 2500, 2650, 2750, 2850, 2950, 3050, 3150
3, 3250, 2450, 2450, 2500, 2650, 2750, 2850, 2950, 3050, 3150, 3250)
J
200 CONTINUE

CALL TYPE(' PARAMETER ERROR')
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 CENTOMERE (USE CURSOR)')
CALL TYPE(' F - FLIP AND CHANGE CENTOMERE (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(' Q - QUICK BANB CALCULATION')
CALL TYPE(' VA - DISPLAY WAVEFORMS')
CALL TYPE(' AX - DISPLAY AXES')
CALL TYPE(' KC - CLEAR CLASSIFY PARAMETERS')
CALL TYPE(' MO - CLEAR MOD PARAMETERS')
CALL TYPE(' MA OR FE - MALE OR FEMALE')
CALL TYPE(' NX - NO X SEPARATION FROM C GROUP')
CALL TYPE(' IF KARYOTYPE IS OK. TYPE CARRIAGE RETURN')
CALL TYPE('0')
GOTO 100

C PARAMETER U OR D - MOVE CURSOR UP OR DOWN
C
910 CALL MCD
GOTO 110

C PARAMETER BC - CALL BANDED CLASSIFIER
C
950 CALL A PHASE(BANDPH)
GOTO 700

C PARAMETER AR - SET MAXIMUM AND MINIMUM AREAS OF VALID CHROMOSOMES
C
1099 CALL MVL(PAR(IP),MOBPAR(M+1),0)
M=M+4
MOBFLG=.TRUE.
GO TO 110

C PARAMETER LW - LOCATE CENTOMERE BY WIDTH RATHER THAN DENSITY
C
1050 CALL MVL(PAR(IP),MOBPAR(M+1),4)
M=M+2
MOBFLG=.TRUE.
GO TO 110

C PARAMETER CI - SET CENTOMERE OF CHROMOSOME SPECIFIED TO THAT
C INDICATED BY CURSOR POSITION
C
1199 CALL KURSOR(Y,X,L,S,SLID,SLCL,SCS,M)
IF(N.EQ.0) GOTO 258
SEL(M)=.TRUE.
CALL TNL('C1',MOBPAR(M+1),2)
MOBPAR(M+3)=N
INL=CHDIR(INL,N)
MOBPAR(M+4)=INL-INT(INL/2)+1
CALL DLNE(BLACK,Y,X,4.1,-1)
CALL DLNE(BLACK,Y,X,4.1,-1)
IF(MOBPAR(M+4).GE.INL.OR.MOBPAR(M+4).LE.1) GOTO 258
M=M+4
MOBFLG=.TRUE.
GO TO 119

C PARAMETER FL - FLIP SPECIFIED CHROMOSOME AND SET CENTOMERE
C
1159 CALL KURSOR(Y,X,L,S,SLID,SLCL,SCS,M)
IF(N.EQ.0) GOTO 250
SEL(N)=.TRUE.
CALL MVL('FL',MOBPAR(M+1),2)
MOBPAR(M+4)=N
INL=CHDIR(DIRNL,N)
MOBPAR(M+4)=NL-(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.NIL.OR.MOBPAR(M+4).LE.1) GOTO 250
M=N+4
MOBFIL=.TRUE.
GO TO 110

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

1200 CALL KURSOR(Y,X,L,S,SLID,SLCL,SLCS,N)
IF(N.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+4)=N
FRSTY=Y
FRSTX=X
CALL TYPE('MOVE CURSOR TO END OF AXES:'),
CALL PARAM(MP,PAR,190)
CALL KURSOR(Y,X,L,S,SLID,SLCL,SLCS,N)
CALL DLINE(BLACK,Y,X-2,3,1,-1)
CALL DLINE(BLACK,Y,X-2,3,1,-1)
SY=Y
SX=X
IF(FRSTX.EQ.SX) GO TO 1217
IF(FRSTY.EQ.SY) GO TO 1218
SY=FRSTY
SX=FRSTX
FRSTX=X
FRSTY=Y

1218 THETA=ATAN2((SX-FRSTX),(SY-FRSTY))
GO TO 1215
1217 THETA=0
1215 CONTINUE
MOBPAR(M+4)=THETA*DEGPRD
M=N+4
MOBFIL=.TRUE.
GOTO 110

C PARAMETER AL - RE-MOB ALL CHROMOSOMES
C

1250 ALFLG=.TRUE.
GO TO 118
C
C PARAMETER SK - OUTPUT SKELETON PICTURES INSTEAD OF GREY LEVEL PICTURE
C

1300 MOBPAR(M+1)=PAR(IP)
M=N+2
MOBFIL=.TRUE.
GO TO 110
C
C PARAMETER NN - NO OBJECT NUMBERS
C

1350 KYPAR(Y+1)=PAR(IP)
K=K+2
KYPFL=.TRUE.
GO TO 110
C
C PARAMETER MA - MALE KARYOTYPE
C

1450 KYPAR(K+1)=PAR(IP)
K=K+2
KYPFL=.TRUE.
GO TO 110
C
C PARAMETER KC - CLEAR CLASY/KTYPE PARAMETERS
C

1500 K=0
CALL WPARAM(K,KTPAR,KYPMA)
OSFLG=.FALSE.
KYPFLG=.TRUE.
GO TO 110
C
C PARAMETER FE - FEMALE KARYOTYPE
C
1550 KTPAR(K+1)=PAR(IP)
K=K+2
KYPFLG=.TRUE.
GO TO 110
C
C PARAMETER HX - HO X SEPARATION FROM C GROUP
C
1600 KTPAR(K+1)=PAR(IP)
K=K+2
KYPFLG=.TRUE.
GO TO 110
C
C PARAMETER DB - DEBUG OPTION ON
C
1650 KTPAR(K+1)=PAR(IP)
K=K+2
KYPFLG=.TRUE.
GO TO 110
C
C PARAMETER P - PUSH A GROUP OF CHROMOSOMES RIGHT OR LEFT
C
1700 CALL KURSOR(Y,X,L1,S1,SL1,SLCL,SLCS,N)
CALL TYPE(",MOVE CURSOR TO END OF PUSH ",Q)
CALL PARAM(HP,PAR.100)
CALL KURSOR(Y,X,L2,S2,SL2,SLCL,SLCS,N)
IF($1.LT.$2) GOTO 1725
C
GOTO 1725 FOR PUSH TO THE RIGHT
C
C PUSH LEFT
N$=SL1-SL2+1
IF(N$ LT.2 OR N$ .GT.10) GOTO 250
Y=2*(L1-1)
CALL BLINE(127,Y,2*S2,S1-S2,1,-1)
X=2*(S2-1)
B1:1710 I=2,12,2
CALL BLINE(BLACK,Y+I-14,Y+I-2,1,2,1,-1)
1710 CALL BLINE(BLACK,Y+I,X+1,2,1,-1)
CALL MW(OF$SL2,PAR,N$)
C SAVE OF$ IN PAR
CALL MW(PAR(2),OF$SL2,N$-1)
OF$(SL1)=PAR(1)
GOTO 1776
C PUSH RIGHT S1 LT S2
1725 Y=2*(L1-1)
N$=SL2-SL1+1
IF(N$ LT.2 OR N$ .GT.10) GOTO 250
CALL BLINE(127,Y,2*S1,S2-S1,1,-1)
X=2*S2
B0:1730 I=2,12,2
CALL BLINE(BLACK,Y+I-14,Y+I-14,2,1,-1)
1730 CALL BLINE(BLACK,Y+I,X+1,1,-1)
CALL MW(OF$SL1,PAR,N$)
CALL MV(PAR(1),OF$SL1,N$-1)
OF$(SL1)=PAR(N$)
GOTO 1776
C
C PARAMETER MV - MOVE INDICATED CHROMOSOME TO SLOT SHOWN BY CURSOR
C
1750 CALL KURSOR(Y,X,L1,S1,SLIB,SLCL,SLCS,N)
CALL BLINE(BLACK,Y,2.3,1,-1)
S$VSLIB=SLIB
CALL TYPE("+MOVE CURSOR TO OTHER SLOT "*,Q)
CALL PARAM(NP,PAR.100)
CALL KURSOR(Y,X,L2,S2,SLIB,SLCL,SLCS,N)
CALL AB(L1,S1,L2,S2,SST,NST,512,512)
B0:1770 L=L1,L2
Y=2*(L-1)
1770 CALL DLINE(127,Y,X,HS(T(L)),1,-1)
SV(D)=OFS(SLID)
OFS(SLID)=OFS(SVSLID)
OFS(SVSLID)=SVID
1775 OSFLG=.TRUE.
KTYFLG=.TRUE.
GO TO 110
C
PARAMETER SP - SET SKELETON PARAMETERS
C
1800 CALL MVL(PAR(IP),MOBPAR(M+1),8)
K=K+4
MOBFLG=.TRUE.
GO TO 110
C
PARAMETER MX - SET MAXIMUM # OF CHROMOSOMES PER GROUP
C
1950 CALL MVL(PAR(IP),KTYPE(K+1),24)
K=K+12
KTYFLG=.TRUE.
GO TO 110
C
PARAMETER IR - SET INITIAL SLOT FOR EACH ROW
C
1950 CALL MVL(PAR(IP),KTYPE(K+1),14)
K=K+7
KTYFLG=.TRUE.
GO TO 110
C
PARAMETER IG - SET INITIAL SLOT FOR EACH GROUP
C
2000 CALL MVL(PAR(IP),KTYPE(K+1),28)
K=K+14
KTYFLG=.TRUE.
GO TO 110
C
PARAMETER MC - CLEAR MOB PARAMETERS
C
2050 M=0
MOBFLG=.TRUE.
GO TO 110
C
PARAMETER SC - SET THE CURSOR
C
2100 CALL SC
GOTO 110
C
SETUP THE CURSOR
C
PARAMETER DS - DISPLAY SPREAD
C
2125 CALL APHASE(FDBPHA)
GOTO 700
C
PARAMETER RF - RERUN FOB
C
2150 CALL APHASE(FDBPHA)
CALL WPARAM(0,MOBPAR,MOBPAR)
CALL WPARAM(0,KTYPE,KTYPE)
GOTO 700
C
RERUN FOB
C
PARAMETER AB - ABORT
C
2200 CALL APHASE(0)
GOTO 700
C
PARAMETER 'X' - REMOVE OBJECT FROM KARYOTYPE
C
2250 CALL KURSOR(Y,X,L1,SLID,SLCL,SLCS,H)
DO 2260 I=2,26,2
CALL DLINE(BLACK,Y+1-14,Y+1-16,2,1,-1)
CALL DLINE(BLACK,Y+1-14,Y+14-1,2,1,-1)
CONTINUE
C
DRAW AN X OVER THE OBJECT TO BE DELETED
OFS(SLID)=0
OSFLC=. TRUE.
KTYFLC=. TRUE.
GOTO 110

C PARAMETER L - ADD A LABEL TO INDICATE DIAGNOSIS
C
DO 2310 I=4,72
2310 IF((VCPARBUF(I)).EQ. 'IS') GOTO 2320
CONTINUE
CALL TYPE(' INCORRECT LABEL FORMAT')
GOTO 190
LABLEN=1-3
CALL AFIL(E(MOB, 1, 'PIC',5,5)
CALL OPEN(MOB512, 0, 2, 'NOB')
CALL GET(MOB, 1, IN)
CALL PUT(MOB, 1, IN)
CALL HVL(PARBUF(3), MOB(IN+290), LABEL)
CALL CLOSE(MOB)
C ALSO ADD TO THE MOB OUTPUT LABEL FOR RESEL'S BENEFIT
CALL GLABEL(TOBUF, SPAR, IB)
CALL MV(L(PARBUF(3), TOBUF(IB+290), LABEL)
GOTO 110
C
C PARAMETER L - DISPLAY WAVEFORMS FROM BAND
C
2350 CALL UPW(RMH(2, 'WAVE', BANDPH)
GOTO 250
C
C PARAMETER AXIS - DISPLAY AXES FROM BAND
C
2400 CALL UPPAR(M(2, 'AXIS', BANDPH)
GOTO 950
C
C PARAMETER RB - RE-RUN BANDED CLASSIFIER
C
2450 CALL A PHASE(FOURPH)
GOTO 700
C
C PARAMETER QB - QUICK BAND CALCULATION (ONLY THE C GROUP)
C
2500 CALL SSW TCH(7, ISU7)
IF(ISU7, HE, I) CALL UPPAR(2, 'QB ', BANDPH)
GOTO 750
C
C DOIE LOOKING AT PARAMETERS. SEE IF A RE-RUN IS NECESSARY
C
300 IF (. NOT. ALLFLG.AND. . NOT. KTYFLG.AND. . NOT. MOBFLG) GOTO 700
IF (. NOT. OSFLG) GOTO 550
C
C WRITE THE OFS PARAMETERS FOR CLASSFY
C
CALL MV(L('OS', KTPAR(K+1), 2)
CALL MV(L('OF.S', KTPAR(K+3), 99)
K=K+92
CONTINUE
IF (. NOT. ALLFLG.AND. . NOT. MOBFLG) GOTO 650
IF(ALLFLG) GOTO 610
C
C MUST RE-RUN MOB ON SELECTED CHROMOSOMES
C
CALL MV(L('SE', MOBPAR(M+1), 2)
M=M+3
MM=M
MM=M+1
DO 600 N=1, MOB
IF (. NOT. SEL(N)) GO TO 600
M=M+1
MM=MM+1
MOBPAR(N)=N
CONTINUE
IF (M EQ. 0) GO TO 695
MOBPAR(MM)=NN
CALL PUT(TOBUF(1, IB)
CALL MV(L(NOB, TOBUF(IB+1), 10)
CALL MV(L(CHDIR(1, IB), TOBUF(IB+31), 2*15*30)
CALL PUT(IOBUF, 2, 1B)
CALL MVL(CHDIRC(1.31), IOBUF(1B+31), 2*15+39)
GO TO 610

695 M=M-3
C
C SELECT MOB AS NEXT PHASE TO BE RUN
C
610 CALL APHASE(MOBPH)
CALL UPARAM(N, MOBPAR, MOBPHA)
CALL WPARAM(K, KTYPAR, KTYPHA)
GO TO 700
C
C SELECT CLASSFY AS NEXT PHASE TO BE RUN
C
650 CALL APHASE(KTYPHA)
CALL UPARAM(K, KTYPAR, KTYPHA)
C
700 CONTINUE
CALL CLOSE(IOBUF)
CALL EXIT
END

SUBROUTINE RESEL
C
RESSEL IS USED TO ENTER INFORMATION ON MEASURED CHROMO
C
SOMES INTO KROMDATA. THE INPUT IS FROM MOB OUTPUT.
C
IMPLICIT INTEGER(A-Z)
COMMON/C1/MBUF, KBUF, NEXT, HUSED, PID, RECS, ONL, ONS,
IOLEN, O10D, OHREA, OCIL, OClD, OCIA, OFS
R=3 D(3)
REAL*4 CFOUR(60, 0), PHI(60, 0), MU(10, 24), SIGMA(18, 24)
REAL*4 TSUM, SUML, SUMH, SUMX, OCFOUR(51, 0), OPHI(51, 7)
INTEGER NF0(18, 3), PSPAR(3), PDI(425), RECS(85)
BYTE O6N(80), ONL(80), ONS(80), OLEH(80), OCIL(80), OClD(80), OCIA(80)
BYTE PBUF(563), LABEL(124), CT3640, ODBDATA(3072)
BYTE MBUF(2184), KBUF(36), FNSG(44), TMSG(23), OMSGC(128)
INTEGER PSMAR(18), PSTP(18), P10D(80), OARREA(80), OFS(90),
2TSF(90), SCAND(90), TH(90), WN(60), GF9(90), NF0(10)
EQUVALENCE (OBDATA, OCFOUR), (OBDATA(1633), OPHI)
EQUVALENCE (C1(57), MU), (C(128), SIGMA)
EQUVALENCE (SPAR(6), SCODE), (HLP, PSPAR(1))
DATA MD/3, DIB='/46XX----/46XX--'/45XO--'
DATA NF0/6.4, 16.6, 6.4, 0.8, 0.6, 4.15, 6.6, 6.4, 0.8, 0.8,
1.6, 4.15, 6.6, 6.4, 0.8, 0.8/
DATA GSNG/128*
DATA TFS/1, 1, 0.4, 8.2, 2.8, 4.0, 3.5, 4.0, 4.4, 5.0, 4.0, 0.8
2.6, 6.7, 7.8, 8.9, 9.8, 10.8, 11.12, 12.23, 23.40, 40
5.10, 40
DATA GS/M/12=1.6*2.10*3.8*4.10*5.6*6.2*8.10*7.6*8, 10*9/
DATA CM/61/
C CURRENT MISCELLANEOUS INDEX = 61
C
OPEN MOB OUTPUT AND COPY NECESSARY DATA FROM MOB LABEL
C
CALL SSUCH(10, ISW10)
IF(ISW10 NE.1) CALL APHASE(19)
C IF SUI0 IS DOWN SET TO CALL #BNORM
CALL AFLEX(MBUF, 1), RCR, RCR
CALL OPEN(MBUF, 1024, 1), RCR
CALL OLWSEL(MBUF, SPAR, HL)
CALL MVW(MBUF(ML-13), SCODE, 1)
CODE=SCODE
IF(SCODE LT. 0) SCODE=-SCODE
IF(SCODE LT. 0 AND SCODE.LE. 15) GOTO 5
C CHECK FOR VALID CODE
CALL CLOSE(MBUF)
RETURN
5 CONTINUE
CALL MVL(MBUF(ML-73), GSNG(1), 64)
CALL MVL(MBUF(ML-296), GSNG(107), 22)
CALL MV(L(MBUF(ML+73), LABEL, 64)
CALL MV(L(MBUF(ML+143), LABEL, 65), 28)
CALL MV(L(MBUF(ML+217), LABEL, 65), 20)
CALL MV(L(MBUF(ML+289), LABEL, 105), 28)
C SAVE LABEL INFORMATION
C
C OPEN KROMDATA AND GET NL
C
CALL AFIL(KBUF, 4, 'KDATA', 6.6)
IF(CODE. LT. 6) CALL AFIL(KBUF, 4, 'KDATA', 6.6)
CALL OPEN(KBUF, 1024.6.2, 'KRN')
CALL GLABEL(KBUF, KSPAR(KL))
NL=KSPAR(1)
C READ SOURCE DIRECTORY RECORD
C
IF(SPACE IN THIS RECORD GOTO 83)
IF(NEXT. EQ. 0) GOTO 20
C IF LAST DIRECTORY RECORD GO MAKE A NEW ONE
SCORE=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, SCORE, KDB)
CALL GET(KBUF, NL.KD)
SCORE=NL
NUSED=0
22 CONTINUE
DO 24 I=1, 1421, 5
IF(PIC(I). EQ. 0) GOTO 26
C IF THIS SLOT IS FREE GOTO 26
24 CONTINUE
CALL TYPE(' RESOL ERROR')
CALL TYPE(' 0')
PAUSE 12345
26 CALL MV(L(LABEL(21), PIC(I), 10)
C STORE PATIENT ID
NUSED=NUSED+I
IR=(I+4)/5
REC(REC(IR))=REC
IF(REC. NE. 0) GOTO 26
C GOTO 20 IF THIS RECORD WAS DELETED
NL=NL+1
C NEED TO MAKE A NEW RECORD
REC=NL
REC(IR)=REC
IF(CODE. LT. 6) NL=NL+3
C ADD 3 MORE IF BANDED
20 CALL PUT(KBUF, SCORE, KDB)
CALL GLABEL(KBUF, KSPAR(KL))
KSPAR(1)=NL
C UPDATE NL
CALL PLIIEEL(KBUF(KL), KSPAR, KBUF(KL+1))
C
C GET ODS FROM ODS OUTPUT AND TRANSFER TO KROMDATA
C
C GET ODS FROM ODS OUTPUT AND CONVERT TO TYPE
C
CALL GET(MBUF, 3, MD)
II=MD+215
DO 30 I=1, 90
II=I+2
30 CALL MV(L(MBUF(I)), ODS(I), 1)
CALL ZIA(M, 40)
CALL ZIA(M, 60)
CALL ZIA(M, 90)
CALL ZIA(M, 10)
DO 40 J=1, 30
40 IF(ODS(J). EQ. 0) GOTO 40
G=ODS(J)
203  
MFG(G)=MFG(G)+1  
THOFSC(J)=TFS(J)  
CONTINUE

C  MOVE NECESSARY DATA FROM NOG LABEL AND LINE 1 TO KBUF

C  CALL GET(MBUF,1,HD)  
CALL PUT(KBUF,REC,KB)  
CALL MVW(LABEL,KBUF(KD+1),62)  
CALL MVW(MBUF(KB+1),NOB,1)

C  
N=1  
50  IF(TN(N).EQ.0) GOTO 80  
C  
C  GOTO 80 IF THIS OBJECT IS NOT TO BE ENTER

T=T+TN(M)=1  
IF(T.LT.60. AND. T.GT.0) GOTO 60  
T=CNT  
IF(T.GT.00) GOTO 80

C  USE NEXT AVAILABLE SPACE IN MISCELLANEOUS AREA

C  CNT=CNT+1  
CONTINUE

C  
60  GOTO 70 IF FIRST SLOT FOR THIS TYPE IS EMPTY

T=T+1  
70  OCN(T)=N  
N(NH)+T  
80  CONTINUE

C  CNT=N.LE.NOBJ GOTO 50

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

II=30+MD  
IST=1  
IEND=30  
DO 100 J=1,2  
K=0  
DO 90 I=IST,IEND  
K=K+1  
MNI=NHI(I)  
IF(MNI.EQ.0) GOTO 90  
IM=(K-1)+30+11

C  CALL MVW(MBUF(IM+3),OML(MNI),1)  
CALL MVW(MBUF(IM+5),OHL(MNI),1)  
CALL MVW(MBUF(IM+7),OLEN(MNI),1)  
CSUM=CSUM+1  
LSUM=LSUM+OLC(MNI)  
CALL MVW(MBUF(IM+9),01O3(MNI),1)  
ISUM=ISUM+0103(MNI)  
CALL MVW(MBUF(IM+11),OAREA(MNI),1)  
ASUM=ASUM+OAREA(MNI)  
CALL MVW(MBUF(IM+13),OCIL(MNI),1)  
CALL MVW(MBUF(IM+15),OCIB(MNI),1)  
CALL MVW(MBUF(IM+17),OCIA(MNI),1)  
90  CONTINUE

C  IF(J.EQ.2) GOTO 100  
CALL GET(MBUF,2,MD)  
IST=31  
IEND=NOB  
II=30+MD

100  CONTINUE

C  CALL MVW(CSUM,KBUF(KD+29),1)  
CALL MVW(LSUM,KBUF(KD+27),1)  
CALL MVW(ISUM,KBUF(KD+29),2)  
CALL MVW(ASUM,KBUF(KD+133),2)  
IF(CODE.GT.0) GOTO 190

C  CANCELED STREAM PROCESSING

C  CALL GET(MBUF,4,IN)  
CALL MVW(MBUF(IM+1),CFour(1,1),480)  
CALL GET(MBUF,5,IN)
CALL MW(MBUF(1+1), CFOUR(1.5), 400)
CALL GET(MBUF: 6, IM)
CALL MW(MBUF(1+1), PHI(1.1), 400)
CALL GET(MBUF: 7, IM)
CALL MW(MBUF(1+1), PHI(1.5), 400)
DO 110 N=1, NOB
T=N(NK(N))
TF(N. E0 . OR. T GT. 51) GOTO 118
C GET THE FOURIER DATA
DO 102 I=1, NOB
102 CFOUR(T, I)=CFOUR(N, I)
DO 104 I=1, NOB
104 PHI(T, I)=PHI(N, I+1)
110 CONTINUE
199 CONTINUE
C WRITE RECORD ON KRONDATA
C CALL MW(ON, KBUF(KD+143), 448)
NC=0
DO 200 G=1, 9
CALL OUTCON(NFC(G), GMSG(69+G*4), 4)
200 NC=NC+NFC(G)
CALL OUTCON(NC, GMSG(48), 4)
IF(N(IP(N-17))), NE. 32) GOTO 225
C IF OPERATOR SUPPLIED A DIAGNOSIS GOTO 250
DO 220 N=1, ND
DO 210 G=1, 10
IF(NFC(G), NE, HFB(G, T)) GOTO 220
210 CONTINUE
C HFC MATCHES THIS DIAGNOSIS
GOTO 250
220 CONTINUE
C NO MATCH
225 N=0
GOTO 250
230 CALL MW(BID(N), GMSG(107), 8)
CALL MW(BID(N), KBUF(KD+1067), 8)
250 CALL AFIEL(PBUF, 1, 'PIC', 5, 5)
CALL OPEN(PBUF, 512, 0, 2, 'NOB')
CALL GET(PBUF, 1, IP)
CALL OUTCON(REC, PBUF(IP+176), 4)
CALL MW('RECORD', PBUF(IP+166), 6)
C STORE RECORD NUMBER IN NOB OUTPUT FOR HCOPY
IF(N. NE. 0) CALL MW(BID(N), PBUF(IP+258), 8)
CALL PUT(PBUF, 1, IP)
CALL CLOSE(PBUF)
C STORE DIAGNOSIS IN PREP AND NOB OUTPUT FOR HCOPY
CALL "** XXXXX ON RECORD XXX ** FMSG, 31"
CALL MW(LABELREC(217), FMSG(37), 107)
CALL OUTCON(REC, FMSG(29), 4)
CALL MW(GMSG(107), FMSG(327), 8)
CALL TYPE(FMSG)
IFCODE GT. 0) GOTO 380
C WRITE THE BANDED RECORDS
DO 270 I=1, 3
CALL PUT(KBUF, REC+1, IK)
CALL MW(OBDATA(I+1024-1023), KBUF(1K+1), 312)
270 CONTINUE
380 CONTINUE
C WRITE PATIENT REPORT RECORD
CALL AFIEL(PBUF, 4, 'PDATA', 6.6)
CALL OPEN(PBUF, 512, 0, 2, 'FRP')
CALL LABEL(PBUF, PSPAN, IP)
NLP=NLH+1
CALL PLABEL(PBUF, PSPAN, PBUF(IP+1))
CALL GET(PBUF, NLH, IP)
CALL MW(GMSG, PBUF(IP+1), 128)
CALL PUT(PBUF, NLH, IP)
CALL CLOSE(PBUF)
CALL CLOSE(KBUF)
CALL CLOSE(MBUF)
CALL TYPE('8')
RETURN
END
C MASK2 - COMBINE 2 PICTURES AND ADD A BORDER

CALNS_VERSION

SUBROUTINE MASK2(DUNIT,FILPEX)
IMPLICIT INTEGER(A-Z)
REAL STATS(256)
BYTE (G(208),B(209),C(952),LE(12),RE(12))
COMMON STATS,LE,RE

INTEGER 2,LOW(512),HIGH(511)
EQUIVALENCE (LOW,STATS), (HIGH(1),LOW(2))
INTEGER 2 SPAR(5),SPARB(5),PAR(10),KEY(3)
BYTE FILPEX(12)
DATA MAXHS/992/
DATA NKEY/3/KEY/'HI','HS', 'KG'/'.$HFLAG/0,NFR/128/ISIZE/100/
CALL AFILE(A,1,'PIC',5,5)

C ASSIGN DK1,PIC(5,5) AS DEFAULT FILE FOR NOB OUTPUT
CALL OPENA(3982-1,0,'NOB')
CALL LABEL(A,SPAR,1A)
CALL AFILE(A,5,'PIC',5,5)

C ASSIGN DK5,PIC(5,5) AS DEFAULT FILE FOR KGM
CALL RPARAMWR.PAR(10)

IP=1
3 IF(IP.GT.NP) GOTO 30
DO 5 J=1,NKEY
IF(FPAR(IP),EQ.KEY(J)) GOTO (10,12,15),J
5 CONTINUE
8 CALL TYPE(0, ** PARAMETER ERROR**)
PAUSE
CALL ZIA(STATS,512)
IP=IP+1
GOTO 7

12 ISIZE=PAR(IP+2)
IP=IP+1
GOTO 10

15 CALL AFILE(A,PAR(IP+2),PAR(IP+3),2,2)
C ASSIGN A DIFFERENT KGM FILE
CONTINUE
CALL OPENB(3982-1,0,'KGM')
CALL LABELB(0,SPARB,1B)
NL=SPARB(1)
IF(SPARB(1),GT.NL) NL=SPARB(1)
NS=SPAR(2)+SPARB(2)+1
IF(NS.GT.MAXNS) NS=MAXNS
IF(SPAR(3),NE,7) NFR=256
CALL MSUB(A,1A+1),NS,DUNIT,FILPEX)
C LABEL CALL TO MSUB
DO 100 L=1,N
IF(LT.SPAR(1)) GOTO 50
CALL GET(A,L,IN)
CALL MVL(A,IA+1),C,SPAR(2)
GOTO 55
50 CALL ITLA(0,C,SPAR(2))
55 CXSPAR(2)+1=127
IF(LT.SPARB(1)) GOTO 60
CALL GETB(L,IB)
CALL MVL(B,IB+1),C,SPAR(2),SPARB(2)
GOTO 65
60 CALL ITLA(0,C,SPAR(2),SPARB(2))
65 CALL MSUBC(NC,NS)
IF($HFLAG.EQ,1) CALL LSTAT(SPAR(2),A,IA+1),STATS,1,1)
C IF HISTOROGM TAB REQUESTED, CALL HFR THE STATISTICS
100 CONTINUE
IF($HFLAG.EQ,0) GOTO 200
II=1
C CONVERT THE FREQUENCY COUNTS TO REAL NUMBERS
DO 120 I=1,NFR
STATS(I)=LOW(I)+HIGH(I)*32768
120 II=II+2
CALL MSUB(STATS,-NFR,ISIZE)
GOTO 220
220 CALL MSUB(0,0)
CONTINUE
CALL CLOSE(A)
END
SUBROUTINE MSUB(LOC, MSO, ISIZE, FILEP)

IMPLICIT INTEGER(A-Z)

REAL STAT$(256), MXIF, HSIZ
BYTE FILEP$(12), DATA(1000)
COMMON STATS, DATA
INTEGER FREQ(256), SPAR(5)
EQUIVALENCE (FREQ, STATS)
INTEGER ISIZE(5)
BYTE DSRN(6258), T(2104), LMSG(29)
EQUIVALENCE (DSRN(4153), T(4153), DSRN(6257), LMSG(29))
BYTE LAB(512), LOC(976), BUF(1000), LH(4), LOGN(4)
DATA MAXXS/976/, MINNS/360/
DATA SPAR/0, 0, 4, 1, 512/
DATA MAXF/1, HSIZ/200, L/G, WHITE/15, STEP/15, LBW/8/
IF(LBW.GT.0) GOTO 100
C GOTO 100 IF NOT INITIAL CALL
CALL AFILE(DSRN, ISIZEFILEP, 2, 2)
CALL SWITCH(5, IS5)
BUFF1Z=3872
IF(ISW.S.EQ.1) BUFF1Z=3848
C IF TAPE OUTPUT, DECREASE SIZE OF DSRN TO MAKE ROOM
CALL OPEN(DSRN, BUFSIZ-1, 'MSK')
CALL MVW(LOC, LAB, 256)
C SAVE LABEL AND BPE
NS=NSO
IF(NS.GT.MAXXS) HS=MAXXS
NS2=NS+1)/2
C NS2 IS THE NUMBER OF WORDS TO MOVE TO DATA
IF(NS.LT.MINNS) HS=MINNS
C MINNS IS THE MINIMUM SIZE FOR THE DATE AND TIME LABEL AND HSTRGM
NS4=NS+24
NW=(NSW+1)/2
SPAR(2)=NW
SPAR(4)=512/NW
IF(SPAR(4).EQ.0) SPAR(4)=1
CALL LABEL(DSRN, SPAR, LAB)
IF(ISW5.NE.1) GOTO 50
CALL MVW(1, T(21), 1)
CALL MVW='S2140',T(23),1)
C SET MVW FOR MTO:
CALL OPEN(T, 1024, 1, 3, 'TAP')
CALL MVW(1, T(51), 1)
C BPE=1
CALL MVW(NSW, T(45), 1)
C RECLE=NSW
CALL MVW(NSW, T(47), 1)
C BLOCKSIZE = NSW.
50 CONTINUE
CALL ZIA(BUF, NW)
CALL WLINE(DSRN, LBW, HSU, BUF, 04)
C WRITE A BLANK LINES
CMD=13+MSW/2
GWW=MSW/32
C GREY WEDGE WIDTH
BGW=16+GWW
D2+32+GWW+3
CALL ITLA(255, BUF(B1), B2)
CALL WLINE(DSRN, LBW, HSU, BUF, 01)
CALL WEDGE(WHITE, STEP, BUF, NW)
CALL WLINE(DSRN, LBW, HSU, BUF, 12)
CALL UWPEF(PPP, STEP, NW).
CALL WLINE(DSRN, LBW, HSU, BUF, 12)
CALL WLINE(DSRN, LBW, HSU, BUF, 01)
C WRITE GREY SCALE
CALL ZIA(BUF, NW)
CALL WLINE(DSRN, LBW, HSU, BUF, 01)
DO 90 K=0, 6
IF(K.EQ.3) CALL REF5(BUF, NW)
DO 70 N=100, MSW
CALL OUTCUN(LH(4), 4)
70 CALL TEXT(LH(2), 1, K, BUF(H5), 1)
90 CALL WLINE(DSRN, LBW, HSU, BUF, 1)
CALL REF5(BUF, NW)
CALL ULLNE(MRN, L&J, HWrWFc@.&..)

CALL ITLA(255, BUF(5), 8)
CALL ITLA(255, BUF(13+NS), 8)
CALL WLINE(DSRH, LBW, NSW, BUF, 81)

RETURN

CONTINUE

IF(NS0.LE.B) GOTO 260

CONTINUE

CALL ITLA(0, DATA(1), 12)
CALL ITLA(0, DATA(13+NS), 12)

C ZERO LINE REFERENCE MARKS
LP3=L=3

ML=MOD(LP3, 100)

C TIME FOR NUMBER OF HUNDREDS
CALL OUTCON(LP3, LH(4), 4)
CALL TEXT(LH(2), 1, ML, DATA(17+NS), 1)
GOTO 140

CONTINUE

IF(MOD(L, 5).NE.0) GOTO 150
CALL ITLA(255, DATA(5), 4)
CALL ITLA(255, DATA(17+NS), 4)

C INSERT EVERY 5 MARK
140 CONTINUE

IF(MOD(L, 25).NE.0) GOTO 150
CALL ITLA(255, DATA(9), 4)
CALL ITLA(255, DATA(13+NS), 4)

150 CONTINUE

CALL WLINE(DSRH, LBW, NSW, DATA, 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.GE.10) GOTO 228

228 CONTINUE

CALL TEXT(LH(2), 1, ML, BUF(2-1), 1)
CALL TEXT(LH(2), 1, ML, BUF(17+NS), 1)

C FINISH UP NUMBER THAT WAS STARTED ON LEFT AND RIGHT SIDE
ML=ML+1
CALL TEXT(LH(2), 1, ML, BUF(2-1), 1)
CALL TEXT(LH(2), 1, ML, BUF(17+NS), 1)

230 CONTINUE

IF(K.LT.5) GOTO 240
CALL TEXT(LOGN(2), 1, K-5, BUF(H+9), 1)

240 CALL WLINE(DSRH, LBW, NSW, BUF, 1)

C WRITE BOTTOM REFERENCE MARKS
CALL ZIA(BUF, NSW)
CALL WLINE(DSRH, LBW, NSW, BUF, 1)
CALL ITLA(255, BUF(81), B2)
CALL WLINE(DSRH, LBW, NSW, BUF, 81)
CALL WEDGE(255, -STEP, BUF, HS)
CALL WLINE(DSRH, LBW, NSW, BUF, 12)
CALL WEDGE(WHITE, STEP, BUF, HS)
CALL WLINE(DSRH, LBW, NSW, BUF, 12)
CALL ITLA(255, BUF(81), B2)
CALL WLINE(DSRH, LBW, NSW, BUF, 81)

C WRITE GREY SCALE
MAXCHAR=(NS+11)/12
JMAX=43

DO 250 J=12, JMAX-72
IF(MOD(J-1), 67) GOTO 260

C FINISHED WITH LABELS IF NO C IN COLUMN 72
J=J+1

C SKIP THE FIRST LABEL CHARACTER WHICH IS ALWAYS BLANK
C FIND THE LENGTH OF THIS LABEL
30 242 I=1,78
242 CONTINUE
C ENTIRE LABEL IS BLANK
GOTO 250
243 LABELN=79-I
244 CHR=LABELN
IF(LABELN.LE.MAXCHR) GOTO 247
C SPLIT UP LABEL--IF IT IS TOO LONG
30 245 I=1,20
IF(LAB(J+70-I).EQ.32) GOTO 246
245 CONTINUE
C NO BLANK FOUND IN 20 CHARACTERS--MAKE ARBITRARY SPLIT
I=1
246 CHR=MAXCHR-I+1
247 CONTINUE
C WRITE LABELS
J=1
CHR=LABELN
LABEL=LABELN-CHR
IF(LABELN.GT.0) GOTO 244
250 CONTINUE
260 CONTINUE
IF(NSO.EQ.0) GOTO 240
C SKIP HISTOGRAM IF NSO EQ 0
NFR=NSO
HSIZE=ISIZE(1)
CALL Z1A(BUF,NW)
CALL WLINE(DSRH,LDW,NSW,BUF,04)
DO 320 I=2,NFR
320 H=H+HSIZE
CALL ZIAC(BUF,NW)
CALL WLINE(DSRH,LDW,NSW,BUF,1)
NFRM1=NFR-1
DO 320 I=2,NFRM1
320 IF(STATS(1).LE.MAXF) GOTO 320
MAXF=STATS(1)
330 CONTINUE
IF(STATS(1).GT.MAXF) STATS(1)=MAXF
IF(STATS(NFR).GT.MAXF) STATS(NFR)=MAXF
DO 330 I=1,NFR
330 FREQ(I)=(STATS(I)/MAXF)*HSIZE+.9999
BARW=H$HFR
HSTART=(H$BARW$HFR)/2+12
J=HSIZE
340 H=H$HSTART
DO 360 I=1,NFR
350 H=H$BARW
CALL WLINE(DSRH,LDW,NSW,BUF,1)
360 H=H$BARW
350 H=H$BARW
CALL WLINE(DSRH,LDW,NSW,BUF,1)
360 H=H$BARW
C THE FOLLOWING CODE WAS COMMENTED OUT TO MAKE ROOM FOR TAPE OUTPUT
C
DO 370 J=0.64,56
C CALL Z1A(BUF,NW)
C H$HSTART=J$BARW
C HMAX=H$HSTART$HMAX$HSTEP
C 380 CALL ITLA(255,BUF(H),BARW)
C CALL WLINE(DSRH,LDW,NSW,BUF,5)
C 370 CONTINUE
C CALL Z1A(BUF,NW)
C CALL WLINE(DSRH,LDW,NSW,BUF,2)
C H$HSTEP=16$BARW
C 390 CALL ITLA(255,BUF(H),BARW)
C CALL WLINE(DSRH,LDW,NSW,BUF,3)
C 370 CONTINUE
C CALL Z1A(BUF,NW)
C CALL WLINE(DSRH,LDW,NSW,BUF,1)
C H$HSTEP=16$BARW
C 390 CALL OUTCON(H,LH(4),4)
C ***********
C BAND *
C ***********
C
C FIND WAVEFORM AND FOURIER COEFFICIENTS FOR BANDED CHROMOSOMES
C
COMMON /CL/HOB,IBIR
COMMON /C1/CHDIR,MLBUF,LRGBUF,NW,HL,COURLH
COMMON /C1/SPID,SPAR,SPAREA
COMMON /C1/HCHR
COMMON /C1/COUPH
COMMON /C1/BANDAHIS
COMMON /C1/QLBUF,QLBUF,QLRGBUF(5,152)
BYTE LRBGBUF(75,8)
BYTE QLBUF(4,152)
BYTE TEMPB
LOGICAL OPT1,OPT2,OPT3,SEL(68)
INTEGER TEMPI
INTEGER SPAR(5),SP(256)
INTEGER BNX
INTEGER BAND(15,68)
INTEGER KPAR(200)
INTEGER BNL
INTEGER CHDIR(15,68)
INTEGER COURLH
INTEGER PAR(150)
INTEGER CLPAR(130)
REAL CFOUR(69,8),PHIC(69,8)
REAL AHIS(150)
REAL COSINE(150),SINE(150)
REAL AFOUR(68,8),BFOUR(68,8)
C EQUIVALENCE(XP,PP,GG,SIGN,JX2,YSTAR,AF,JDIF)
EQUIVALENCE(YPP,QQ,SIGY,LY2,YEND,BF,JU)
EQUIVALENCE(XTERM,PPPP,NSL,NNZ,LPC,PL,RR,JUL)
EQUIVALENCE(YTERM,PPPPP,HL,LL,KG,HDIV,RMJ,JLL)
DATA LIN/B, OPT1/FALSE, OPT2/FALSE, OPT3/FALSE.
C LENGTH OF WINDOW FOR WAVEFORM CALCULATION ON STRAIGHT CHROMOSOMES
DATA SEL=60, TRUE.
C
PAR(1)=0
CALL RPARAN(NPAR, PAR, 18)
N=1
DO 10 NP=1, NPAR, 2
IF(PAR(NP).EQ. 'WA') OPT2=TRUE.
IF(PAR(NP).EQ. 'AX') OPT3=TRUE.
IF(PAR(NP).NE. 'QB') GOTO 10
OPT1=TRUE
10 CONTINUE
C
CALL AFILE(OBUF, 'RCR', '13', '13')
CALL CLAGEL(OBUF, SPAR, 18)
IB=0
C
CALL GET(OBUF, IB)
CALL MVLC(OBUF(IB+3), SPID, 12)
NCHR=IV(OBUF(IB+17))
CALL MVLC(OBUF(IB+31), CHDIR(1, 1), 2*15+30)
CALL GET(OBUF, 2, IB)
CALL MVLC(OBUF(IB+31), CHDIR(1, 31), 2*15+30)
CALL GET(OBUF, 3, IB)
CALL MWV(OBUF(217+IB), OFS, 72)
IF(NOT OPT1) GOTO 30
IF(OFS).EQ.0 GOTO 20
SEL(OFS(1))=.TRUE.
20 CONTINUE
30 CONTINUE
C
DO 310 II=1, NCHR
IF(.NOT. SEL(II)) GOTO 310
CUROLN=CHDIR(1, II)
NL=CHDIR(2, II)
IF(CUROLN.EQ. 0) NL=0
NS=CHDIR(3, II)
IF(NS.LT.1 OR NL.LT.1) GOTO 310
NL=0
LPG=1624/NS
C
LINES=GET
C
DO 380 I=1, NL
CALL GET(OBUF, CUROLN, IB)
IF(I.EQ.1) LASTIB=IB
CUROLN=CUROLN+1
DO 381 II=1, LPG
NG=NG+1
C
WRITE(5, 2888) (OBUF(I+1A), M1, M1, NS)
CALL MVLC(OBUF(IB+1), LRGBUF(1, NG), NS)
IB=IB+NS
IF(NG.GE. NL) GOTO 311
381 CONTINUE
380 CONTINUE
311 CONTINUE
C
TEST THE ORIENTATION
C SCAN ALONG X (ROW) DIRECTION
C
JX2=NS/2
IY2=NL/2
NH2=0
MSEGX=0
MSEGY=0
C
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.EQ.1) GO TO 410
IF(NSECX.GT.1.AND.NSEGY.GT.1) GO TO 420
GO TO 440

CONTINUE

STRAIGHT

DO 411 I=1, NL
LASTB=0
DO 405 J=1, NS
IF(LRGBUF(J,J)).NE.0 GO TO 406
LASTB=J
CONTINUE

CONTINUE

IF(HSECX.LE.1.AND.HSEGY.LE.1) GO TO 420
IF(HSECX.GT.1.AND.HSEGY.GT.1) GO TO 430
GO TO 440

CONTINUE

MOVE ONE PLACE OVER

IL=IL+1
IF(IBAND.GT.MBAND) MBAND=IBAND
IF(IL.LT.IH) GO TO 4150

CONTINUE

FLIP IT DIAGONALLY
CONTINUE
BENT AND PROPERLY ORIENTED

DO 431 I=1, NLL
DO 431 J=1, NSIGY
IF(LRGBUF(J, I).EQ.0) GO TO 431
HP=HP+1
NSIGY=NSIGY+FLOAT(I)
CONTINUE
IZERO=NSIGY/HP

DO 422 J=1, MSL
DO 422 I=1, HL
IF(LRGBUF(J, I).EQ.0) GO TO 422
HP=HP+1
NSIGX=NSIGX+J
CONTINUE
IZERO=NSIGX/HP

A11=0.0
A21=0.0
A12=0.0
A22=0.0
A13=0.0
A31=0.0
A32=0.0
A33=0.0
S=0.0
T=0.0
U=0.0

HSM2=HSM-2
DO 500 J=1, HSM2
DO 500 I=1, HL
IF(LRGBUF(J, I).EQ.0) GO TO 500
XP=J-IZERO
YP=I-IZERO
XTERM=XP
YTERM=YP
A23=A23+YTERM
U=U+XTERM
T=T+XTERM*YTERM
UTERM=UTERM*YP
S=S+XTERM*YTERM
A22=A22+YTERM
YTERM=YTERM*YP
A21=A21+YTERM
YTERM=YTERM*YP
A11=A11+YTERM
CONTINUE

A12=A21
A13=A22
A31=A32
A23=A31
A12=A23+A33*(A12+U+A13+S)
PPP=U*A22+A13*A23+5*A33+T*A12

CONTINUE
PPP=AI1*A22*A33*A12+A23*A22+AI1
PPPP=A22*A22+A13*A23*A23+A11+A12*A12
P=(PP-PPP)/(PPP-PPP)
Q=(A23*(S-A11+P)-A13*(T-A12+P))
QO=A12*A23-A13*A22
Q=QQ/QO
R=(S-A11+P-A12+Q)/A13

NDIV=3

DO 700 I=1, NL
YP=1-I-ZERO
DY=2.*P*YP+Q
DYX=1.8/DY
XP=P*(I-I-ZERO)**2+Q*(I-I-ZERO)+R+ZERO
MJ=XP+0.5
MI=YP+I-ZERO+0.5

ENCORE) LRGBUF(MJ, MI) =1

BAND(I)=LRGBUF(MJ, MI)
MJ=XP+1.5
MI=YP+I-ZERO+0.5-DY

IF(MI.LE.0) GO TO 701
IF(MI.GT.NL) GO TO 701
BAND(I)=BAND(I)+LRGBUF(MJ, MI)

GO TO 702

701 CONTINUE
NDIV=2
BAND(I)=BAND(I)/NDIV
NDIV=3

702 CONTINUE

BAND(I)=BAND(I)/NDIV

CALCULATE LENGTH

PTZ=P*P
YSTAR=0.*PTZ
YENDFLOAT(NL)/2.*YSTAR
PYE=P*YEND
PPYE=PYE+PYE
EL=PPYE+SORT(PPYE+PPYE+1.0)+ALOG(PPYE+SORT(PPYE+PPYE+1.0))
CHDIR=4./T1=EL/PTZ+0.5

USE MOD LENGTH DUE TO UNSUAL RESULTS!

CONTINUE

FOURIER COEFFICIENTS

BMAX=BAND(I)
DO 710 I=2, NL
IF(BAND(I).GT.BMAX) BMAX=BAND(I)

END
CONTINUE
C
DO 711 I=1,NL
AHIS(I)=<BAND(I)+180>/BMAX
711 CONTINUE
C
NHARM=8
FNL0=FLOAT(NL-1)
PIL=6.2832/FNL0
C
CALL SSWTCH(4,ISW4)
IF(ISW4.EQ.1) CALL TIMER
C
NLHALF=(NL+1)/2
DO 5020 J=1,NLHALF
RMJ=FLOAT(J-1)+PIL
LRMJ=J
COSINE(LRMJ)=COS(RMJ)
SINE(LRMJ)=SIN(RMJ)
NLMJ=NL-J+1
COSINE(NLMJ)=COSINE(LRMJ)
SINE(NLMJ)=SINE(LRMJ)
5020 CONTINUE
C
DO 5050 I=1,NHARM
AF=0.0
BF=0.0
IM1=I-1
RM=FLOAT(IM1)+PIL
C
DO 5080 J=1,NL
JM1=J-1
RMJ=RM+FLOAT(J-1)
LRMJ=JM1+IM1
LRMJ=MOD(LRMJ,NL-1)+1
C
AF=AF+AHIS(J)*COSINE(LRMJ)
BF=BF+AHIS(J)*SINE(LRMJ)
5080 CONTINUE
C
AF=(AF+AF)/FNL0
BF=(BF+BF)/FNL0
C
CFOUR(NHMOB,1)=SORT(AF+BF)
PHI(NHMOB,1)=ATAN2(BF,AF)
C
5080 CONTINUE
C
IF(ISW4.EQ.1) CALL TIMER
C
IF(.NOT.OPT2) GO TO 610
DO 811 I=1,NL
JUV=AM(1)(I,1)10+AM(1)(I,2)
MAX WIDTH OF WAVEFORM IS 20 SAMPLES
JDIF=JUV-JUL
JLL=JDIF/2+1
JUL=JS-JDIF/2
DO 812 J=1,JLL
LRBUF(J,I)=0
DO 813 J=JLL,JUL
LRBUF(J,I)=127
DO 814 J=JUL,NS
LRBUF(J,I)=0
811 CONTINUE
810 CONTINUE
C
IF(.NOT.OPT3.AND..NOT.OPT2) GO TO 310
C
CUROLN=CHDIR(1.11)
NG=0
LPG=1824/NS
C
DO 888 I=1,NL
CALL GET(ODBUF,CUROLN,IB)
DO 831 III=1,LPG
NG=NG+1
CALL MUV(1, OBUF(1), OBUF(16), NS)
1B = IB + MS
IF(NG.GE.NL) GO TO 842.
831 CONTINUE
842 CONTINUE
CALL PUT( OBUF, CUROLN, IB)
CUROLN = CUROLN + 1
IF(NG.GE.NL) GO TO 841
841 CONTINUE
C 310 CONTINUE

C OUTPUT FOURIER COEF'S TO FILE RCR
C CALL PUT(OBUF, 4, IB)
CALL MUV(CFOUR(1), OBUF(10+1), 480)
CALL PUT(OBUF), 5, IB)
CALL MUV(CFOUR(1), OBUF(10+1), 480)
CALL PUT(OBUF, 6, IB)
CALL MUV(PHI1, OBUF(10+1), 480
CALL PUT(OBUF, 7, IB)
CALL MUV(PHI1, OBUF(10+1), 480)

C CALL CLOSE( OBUF)
C
C CREATE PARAMETERS FOR PHASE 14 (CLAS)
C DO 4000 I = 1, NCHR
CLPARM(9 + 2 * I) = CHDIR(4, I)
CLPARM(10 + 2 * I) = CHDIR(18, I)
IF(CHDIR(I1, EQ. 0) CLPARM(I2 + 2 * I) = 0
C IF BLKND IS ZERO, THE PHASE WAS REJECTED BY MOD
C 4000 CONTINUE
C
C SET UP PART OF CHROMOSOME DIRECTORY FOR NEXT PHASE
C CLPARM(I) = NCHR
C
C POSITIONS 2 THRU 10 MAY BE REGARDED AS SPARE
C CALL UPARM(I0, CLPARM, 19)
C IF .EQ. 1, CALL APHASE(I0)
C IF SUB IS UP, CALL KTYPE INSTEAD OF FOUR
C IF(OPT2 OR OPT3) CALL APHASE(I0)
C CALL EXIT
C END

SUBROUTINE FOUR
COMMON /C1/ CHDIR, SMLBUF, LRCBUF, MS, NL, CUROLN
COMMON /C1/ SPF10D, SPLTH, SPARE
COMMON /C1/ NCHR
COMMON /C1/ CFOUR, PHI
COMMON /C1/ GBUF, BUF
INTEGER JV(50), J1TAB(193), J2TAB(100)
BYTE JMSC(127)
BYTE OBUF(212)
BYTE C(364)
REAL LTAB(60), CTAB(60)
INTEGER START(25), INKRC(25), MRC(25)
INTEGER LOFS(91)
INTEGER DES(118), KPAR(92)
INTEGER CLPARM(139)
INTEGER CPAR(5)
INTEGER SPAR(5)
INTEGER CUROLN
INTEGER PH1
REAL CFOUR(69, 18), PHI(60, 18)
INTEGER MCHR(24)
INTEGER CLASS(60)
BYTE MGC(30)
REAL DIST(18)
REAL VECTOR(18)
REAL LKLE(69, 24)
REAL N(10, 24)
REAL SIGMA(10, 24)
REAL MBUF(10, 24)
REAL SBUF(10, 24)

C
EQUIVALENCE(MBUF(1, 1), C(57))
EQUIVALENCE(SBUF(1, 1), C(1785))
EQUIVALENCE(SBUF(1, 1), SIGMA(1, 1))
EQUIVALENCE(MBUF(1, 1), NU(1, 1))
EQUIVALENCE(KPAR(3), OFS(1))

C
DATA VMAX1/80, VMAX2/80, VMAX3/58, VMAX4/58, VMAX5/64, / 
DATA IS1/19, IS2/24 / 
DATA JC1/6, JC2/13 /
C SLOTS TO BE CLASSIFIED WITH HYBRID CLASSIFIER
DATA NEXTPH/20 /
C
NEXT-PHASE 10.CALL FOR HYBRID CLASSIFIER
DATA KMEAS/2, K2MEAS/3, KXMEAS/1, NGROUP/24, / 
DATA START/1, 5, 9, 13, 15, 19, 21, 23, 25, / 
1 27, 29, 31, 37, 39, 41, 45, 49, 51, / 
C 55, 57, 65, 67, 74, 78, 84, / 
DATA INCR/22, 24, -1, -1, -1 /
DATA CNC/23*2, 1, 46 /
DATA OFS/118*0 /
DATA NCHROM/24*0 /
DATA CLASS / 60*0 /
C
C
OBTAIN PARAMETERS, LENGTH AND CENTROMERE INDEX
C AS WELL AS THE NUMBER OF CHROMOSOMES IN THE SPREAD
C
CALL RPARN(KPARN, CLPARN, 130)
CALL SSWTCH(4, ISW4) IF(ISW4. EQ. 1) PAUSE 15
C
PAUSE IF SW4 IS UP
C
NCHR=CLPARN(1)
C
READ BACK FOURIER COEF'S PRODUCED BY BAND (PHASE 14)
C
CALL AFILE(OBUF, 1, 'RCR ', '13', '13')
CALL OPEN(OBUF, 1824, 1, B, 'RCR')
CALL GLLABEL(OBUF, 18, IB)
IF(OBUF(18+184), EQ. '115') MNC(23)=1
IF(OBUF(18+104), EQ. '106') MNC(24)=0
C
C
CHECK FOR SEX OF M OR F AND CHANGE X AND Y MAX
C
CALL GET(OBUF, 3, IB)
CALL MVC(OBUF, 18, LOFS(1), 182)
C
CALL GET(OBUF, 4, IB)
CALL MVC(OBUF, 18+184, CFOUR(1, 1), 960)
C
CALL GET(OBUF, 5, IB)
CALL MVC(OBUF, 18+184, CFOUR(1, 1), 960)
C
CALL GET(OBUF, 6, IB)
CALL MVC(OBUF(1+184), PHI(1, 1), 960)
C
CALL GET(OBUF, 7, IB)
CALL MVC(OBUF(1+184), PHI(1, 1), 960)
C
THE HYBRID CLASSIFIER USES RESULTS OF
THE CONVENTIONAL CLASSIFIER
C
CALL SSWTCH(7, ISW7)
IF(ISW7. EQ. 1) GOTO 880
C
GOTO 880 FOR THE HYBRID CLASSIFIER
C SW7 UP IS FOR A FULL FOURIER CLASSIFIER
IS1=1
IS2=90
JC1=1
JC2=24
NEXTPH=9
GOTO 820

800 CONTINUE
CALL MUV(LOFS, OFS, 90)
CALL ZIA(OFS(IS1), 32-IS1)
C FOUR - WILL CLASSIFY THE C GROUP STARTING WITH IS1
C CALL ZIA(OF5(I3), 4)
CALL ZIA(OF5(37), 6)
CALL ZIA(OF5(55), 4)
CALL ZIA(OF5(63), 4)
C KFiW WILL CLASSIFY X, Y, E., AND C
C
820 IF(NCHR.LE.46) GOTO 830
C IF MORE THAN 46, ALLOW FOR 2 X AND 2 Y
MNC(23)=2
GOTO 830
C GET MEANS AND SIGMAS
C CALL AFFILE(C, 6,'CBATA', '6.*6)
CALL OPEN(C, 3504.0, 0, 'CBAT')
CALL NLABEL(G, CPAR. IC)
CALL GET(C, 1, IC)
C
950 CONTINUE
C
ITOTL=0
NRELCH=CHCR.
DO 999 II=1, NCHR
IF(CLPARM(9+2*II).EQ.0) NRELCH=NRELCH-1
ITOTL=ITOTL+CLPARM(9+2*II)
999 CONTINUE
FTOTL=46.*ITOTL/NRELCH
DO 4000 I=1, NCHR
LTAB(I)=(CLPARM(I+2*1)*10000)/FTOTL
CTAB(I)=CLPARM(I+2*1)
C STORE REAL LENGTH AND CI
C 4000 CONTINUE
C
DO 896 II=1, NCHR
DO 896 JJ=1, NGROUP
LIKLI(II, JJ)=999.
896 CONTINUE
C
CALL SSWITCH(J, ISW).
IF(ISW0, EQ.1) CALL QPRINT('1
1 OBJ GROUP BLEN ICIA VLEN VCIA
2 VCSUM VPSUM TVAR')
C
DO 900 IS=IS1, IS2
II=LOFS(IS)
IF(II.EQ.0) GOTO 900
C
IF(ISW0, EQ.1) CALL QPRINT('')
C
DO 903 JJ=JC1, JC2
JJ=1J
IF(JJ.EQ 13.AND. ISW7, NE.1) JJ=23
C
SIG112=SIGMA(KIMEAS, JJ)
SIG222=SIGMA(K2MEAS, JJ)
IF(SIG112. LT. 0.000001) SIG112=.00001
IF(SIG222. LT. 0.000001) SIG222=.00001
C
DIST(1)=LTAB(JJ)-MUL(KIMEAS, JJ)
DIST(2)=CTAB(JJ)-MUL(K2MEAS, JJ)
233

VECTOR(1)=DIST(1)/SIGMA(2) BIST(1)

IF VECTOR(1) > VMAX(1) GOTO 903

VECTOR(2) = (DIST(2)/SIGMA(2) BIST(2)

IF VECTOR(2) > VMAX(2) GOTO 903

C

FSUM=0.0

DO 907 I = JMEAS, JMEAS

DISTI=FOUR(11, I-1)-MU(1, JJ)

FSUM=FSUM+(DISTI*DISTI*SIGMA(1, JJ))

907 CONTINUE

IF FSUM GT VMAX(2) GOTO 903

PSUM=0.0

IF JMEAS EQ 0 GOTO 900

DO 908 I = JMEAS, JMEAS

DISTI=MJU(11, I-1)-MU(1, JJ)

IF DISTI GT 3.1416 DISTI = 6.2832 DISTI

IF DISTI LT -3.1416 DISTI = -6.2832 DISTI

PSUM=PSUM+DISTI*DISTI/SIGMA(1, JJ)

908 CONTINUE

IF FSUM GT VMAX(4) GOTO 903

JEME(1, JJ)+=VECTOR(1)+VECTOR(2)+FSUM+PSUM

CALL SSNCH(9, ISW(0))

IF ISW(9, NE, 1) GO TO 9010

J(1)=I

J(2)=JJ

J(3)=DIST(1)

J(4)=DIST(2)

J(5)=VECTOR(1)

J(6)=VECTOR(2)

J(7)=FSUM

J(8)=PSUM

J(8)=LIK(I, JJ)

DO 989 I=1, 9

989 CALL OUTCOM(JV(1), JMSG(18*1), 18)

CALL GPRINT(JMSG, 99)

9810 CONTINUE

C

903 CONTINUE

C

DO 952 IX = 1, NCHR

ELMIN=VMAX

IISAY=0

C

DO 950 IJ=JCI, JCE

JJ=IJ

IF(JJ .EQ. 13 AND ISW(9, NE, 1) JJ=23

IF(HCHROM(JJ), GE, MNC(JJ)) GO TO 950

C

C IGNORE GROUPS THAT ARE FULL

C LOOK FOR UNECLASIFIED CHROMOSOMES

C

DO 951 IS=ISI, IS2

II=LOES(IS)

IF(II, EQ, 0) GOTO 951

IF(CLASS(IS), NE, 0) GO TO 951

IF(LIKE(II, JJ), GE, ELMIN) GOTO 953

ELMIN=LIK(I, JJ)

IISAY=II

JJSAY=JJ

953 CONTINUE

951 CONTINUE

C

950 CONTINUE

C

IF(IISAY, EQ, 0) GO TO 958

CLASS(IISAY)=JJSAY

HCHROM(JJSAY)=HCHROM(JJSAY)+1

OPT(START(JJSAY))=IISAY

STRT(JJSAY)=START(JJSAY)+INCR(JJSAY)

952 CONTINUE

C

960 CONTINUE

C
C LOOK FOR UNCLASSIFIED AND SEE IF MOVES CAN BE MADE
C ORDER UP TO 100 LIKELIHOODS IN ILTAB AND JJTAB
C
DO 300 IK=1,188
ELMIN=VMAXS
DO 250 IS=1,IS2
II=LOFS(IS)
IF(II.EQ.0) GOTO 250
IF(CLASS(II).NE.0) GOTO 250
DO 240 JJ=JJL,JJH
JJS=JJ
IF(JJ.EQ.13.AND.ISW7.NE.1) JJ=23
IF(INC(JJ).EQ.0) GOTO 240
IF(LIKLE(JJ,IS),GE.ELMIN) GOTO 240
ELMIN=LIKLE(JJ,IS)
IJSAY=II
JJSAY=JJ
240 CONTINUE
250 CONTINUE
IF(ELMIN.EQ.VMAXS) GOTO 320
ILTAB(IIK)=IJSAY
JJTAB(IK)=JJSAY
Likel(IJSAY,JJSAY)=998.
C IF NO MORE LIKELY CLASSIFICATIONS GOTO 320
C
300 CONTINUE
C
C IF=IK+1
320 IK=IK+1
IF(IK.EQ.0) GOTO 520
DO 500 K=1,IK
II=ILTAB(K)
J=JJTAR(K)
IF(CLASS(II).NE.0) GOTO 500
C SEE IF WE CAN MOVE SOMEONE OUT OF GROUP J INTO A LIGHT GROUP
JS1=START(JJ)-MCHROM(JJ)*INCR(JJ)
JS2=START(JJ)-INCR(JJ)
IF(JS1.GT.JS2) CALL SWAP(JS1,JS2)
ELMIN=VMAXS
DO 450 JS=JS1,JS2
I=OFS(JS)
DO 440 1J=JC1,JC2
J=JJ
IF(JJ.EQ.13.AND.ISW7.NE.1) JJ=23
IF(INC(JJ).GE.MINC(JJ)) GOTO 440
IF(LIKLE(JJ,IS),GE.ELMIN) GOTO 440
BLIK=LIKLE(JJ,JS)
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
IJSAY=I
JJSAY=JS
440 CONTINUE
450 CONTINUE
IF(ELMIN.EQ.VMAXS) GOTO 500
C IF NO MOVE OUT OF J CAN BE MADE GOTO 500
C OTHERWISE, MOVE OBJECT IJSAY FROM SLOT JJSAY TO GROUP JJSAY
C THEN MOVE OBJECT I1 TO SLOT JJSAY
C
CLASS(IJSAY)=JJSAY
MCHROM(JJSAY)=MCHROM(JJSAY)+1
OFS(START(JJSAY))=IJSAY
START(JJSAY)=START(JJSAY)+INCR(JJSAY)
C
CLASS(I1)=J
OFS(JJSAY)=I1
500 CONTINUE
C
520 CONTINUE
C C CHECK FOR STILL UNCLASSIFIED AND PUT AT BOTTOM
C
DO 980 IS=1,IS2
II=LOFS(IS)
IF(I1.EQ.0) GOTO 980
DO 980 980 CONTINUE
IF(CLASS(I), NE, 0) GOTO 980
CLASS(I) = 25

965 IF(START(25), NE, 90) GOTO 980
START(25) = START(25) + 1
IF(OFS(START(25), NE, 0) GOTO 965
OFS(START(25)) = 1

980 CONTINUE

CALL MYL ('OS ', KPAR, 4)
CALL WCARD(92, KPAR, NEXTPH)

CUROLM = 7
CALL GET(OBUF, CUROLM, 18)
CALL MYL(CLASS(I), OBUF(961+18), 60)
CALL PUT(OBUF, CUROLM, 18)

CALL CLOSE(OBUF)
CALL APHASE(NEXTPH)

IF(ISUB.EQ.1) CALL QPRINT('1')
RETURN
END

C**************
C KFIX
C**************
C
C CHROMOSOME CLASSIFIER PART 3
C SYNTACTICAL CORRECTION OF CLASSIFICATION
C
C SUBROUTINE KFIX
C COMMON /C1/ CHDIR, SMPLBUF, LRGBUF, MS, NL, CUROLM,
C COMMON /C1/ SST, EST,
C COMMON /C1/ SPTD, SPTH, SPTA,
C COMMON /C1/ NCHR,
C COMMON /C1/ CFOUR, RHI,
C COMMON /C1/ BAND, AHIS,
C COMMON /C1/ OBUF, BUF,
C BYTE LRGBUF(78, 68)
C BYTE OBUF(2124)
C LOGICAL BGROUP(60)
C LOGICAL DGROUP(60)
C LOGICAL CGROUP(60)
C LOGICAL FGROUP(60)
C LOGICAL EGROUP(60)
C INTEGER ICHCHR(6)
C INTEGER BPINT(6)
C INTEGER ICHCHR(6)
C INTEGER IFCHR(6)
C INTEGER ICHCHR(6)
C INTEGER KPAR(92)
C INTEGER DFSC(61)
C INTEGER LOFS(90)
C INTEGER IPROF(90)
C INTEGER JPROF(90)
C INTEGER KBPH(16)
C INTEGER JFETUR(28)
C INTEGER SPAR(8)
C INTEGER BMNS
C INTEGER BPL
C INTEGER CHDIR(15, 60)
C INTEGER CUROLM
C INTEGER PIR(18)
C REAL BBG(6)
C REAL CD(6)
C REAL RATIO(12)
C REAL CFQW(60, 15), PHI(60, 15)
C REAL AFQW(60, 15), SFQW(60, 15)
C EQUIVALENCE (KPAR(3), OFS(1))
EQUIVALENCE (SPAR(2), BPL)
EQUIVALENCE (CFOUR(1,1), AFour(1,1))
EQUIVALENCE (PHI(1,1), BFour(1,1))
EQUIVALENCE (NL, NLO)
EQUIVALENCE (NUMOB, IT)

C DATA ND/0/
DATA NDCHR/0/
DATA DGROUP/60*, FALSE./
DATA FGROUP/60*, FALSE./
DATA EGROUP/60*, FALSE./

C CALL AFIL(UBUF, 'RCR', '*13,*13')
CALL OPEN(UBUF, 1324, 1, 0, 'RCR')
CALL GLABEL(UBUF, SPAR, IB)

C CALL RPARAM(KPAR, KPAR, 32)

C READ RESULTS OF CONVENTIONAL CLASSIFICATION AND
EXTRACT GROUP CLASSIFICATIONS

D-GROUP

C CALL GET(UBUF, 3, IB)
CALL MVL(UBUF(217*IB), LOFS(1), 182)

C

C DO 184 I=37,42
IIDCHR(I-36)=0
ILDFS=LOFS(I)
IF(ILDFS EQ 0) GO TO 184
DGROUP(ILOFS)=.TRUE.
IIDCHR(I-36)=ILOFS
184 CONTINUE

C

B-GROUP

C DO 182 I=13,16
IIDCHR(I-12)=0
ILDFS=LOFS(I)
IF(ILDFS EQ 0) GO TO 182
BGROUP(ILOFS)=.TRUE.
IIDCHR(I-12)=ILOFS
182 CONTINUE

C

E-GROUP

C DO 1850 I=45,46
ILDFS=LOFS(I)
IF(ILDFS EQ 0) GO TO 1850
EGROUP(ILOFS)=.TRUE.
1850 CONTINUE

1851 CONTINUE

C

F-GROUP

C DO 186 I=55,56
IFCHR(I-54)=0
ILDFS=LOFS(I)
IF(ILDFS EQ 0) GO TO 186
FGROUP(ILOFS)=.TRUE.
186 CONTINUE

C

G-GROUP, INCLUDING Y
DO 107 I=65,70
   I=IDCHR(1-64)=0
   ILOFS=LOFS(I)
   IF(ILOFS.EQ.0) GO TO 107
   GGROUP(ILOFS)=.TRUE.
   I=IDCHR(1-64)=ILOFS
107 CONTINUE

C GET CHROMOSOME DIRECTORY
C
   CALL GET(OBUF, 1, IB)
   CALL MVL(OBUF(IB+IB), SPOS=12)
   N Chr=IV2(OBUF(IB+17))
   CALL MVL(OBUF(IB+31),CHDIR(1,1),2*15+30)
   CALL GET(OBUF, 2, IB)
   CALL MVL(OBUF(IB+31),CHDIR(1,31),2*15+30)

C
   JAY=0
   DO 310 IX=1,6
   II=IDCHR(IX)
   C IF(DGROUP(II)) GO TO 304

C
   IF(DGROUP(II)) GO TO 304
   GO TO 310

C
C JFETUR(1)=0
C
   CUROLN=CHDIR(1,II)
   NL=CHDIR(2,II)
   NS=CHDIR(3,II)
   IF(NL.LT.1.OR.NS.LT.1) GO TO 310
   MG=0
   LPG=1824/NS
   C LINES/GET.
C
   DO 388 I=1,NL
   CALL GET(OBUF, CUROLN, IB)
   IF(I.EQ.1) LASTID=IB
   CUROLN=CUROLN+1
   DO 381 III=1,LPG
   CALL SSWITCH(O, JAY)
   DO 338 J=1,NS
   LRGBUF(J, MG)=OBUF(J+IB)
   338 CONTINUE
   IB=IB+NS
   IF(MG.GE.NL) GO TO 311

C
   IF(NOT DGROUP(II)) GO TO 401

C
   NL2=NL/2
   LTOP1=2
   LTOP2=NL2-1
   LBOT1=NL2+1
   LBOT2=NL-1

C
   ISUMT=0
   ISUMB=0
   NTOP=0
   NBOT=0

C
   DO 411 J=LTOP1,LTOP2
   DO 410 I=1,NS
   IVAL=LRGBUF(I, J)
   IF(I.EQ.0) GO TO 410
   ISUMT=ISUMT+IVAL
   NTOP=NTOP+1
CONTINUE
ITOP=ISUMT/NTOP

DO 421 J=LBOT1,LBOT2
DO 420 I=1,NS
IVAL=LRBUF(I,J)
IF(IVAL.EQ.0) GO TO 429
ISUMB=ISUMB+IVAL
NBOT=NBOT+1

420 CONTINUE
421 CONTINUE
IBOT=ISUMB/NBOT

NB=NB+1
RATIO(NB)=FLOAT(ITOP)/FLOAT(IBOT)
DPOINT(NB)=11

C
C DO FOR B-GROUP
C
C DO 350 IX=1,4
II=IBCHR(IX)
IF(II.EQ.0) GO TO 350

C CUROLN=CHDIR(1,II)
NL=CHDIR(2,II)
NS=CHDIR(3,II)
IBCI=CHDIR(8,II)
NG=0
LPC=1024/NS

C DO 1800 I=1,HL
CALL GET(OBUF,CUROLN,IB)
CUROLN=CUROLN+1
DO 1801 III=1,LPC
NG=NG+1
DO 1830 J=1,NS
LRBUF(J,NG)=OBUF(J+IB)

C
C EXTRACT THE SO-CALLED B-BAND
C
C ISTART=FLOAT(IBCI+HL)/100.0
ISTART=HL-ISTART
IEND=ISTART+1
NL2=NL/2

C
C HPB=0
BIOD=0.0
DO 351 I=ISTART,IEND
DO 352 J=1,NS
IVAL=LRBUF(J,1)
IF(IVAL.EQ.0) GO TO 352
IVAL=FLOAT(IVAL)
DIX=BIOD+FLOAT(IVAL)

352 CONTINUE
351 CONTINUE
BD(IX)=BIOD/FLOAT(HPB)

C CLASSIFY THE B-GROUP
DO 360 I = 1, 4
DENMAX = 0.0
DO 361 J = 1, 4
IF(BD(J), LE, DENMAX) GO TO 361
IOBJ = IBCHR(J)
IF(I0BJ, LT, 1) GO TO 361
DENMAX = BD(J)
JS = J
CONTINUE
OFS((I2 + 1) * IOBJ)
BD(JS) = 0.0
CONTINUE
DO FOR F-GROUP
DO 450 IX = 1, 4
IF(II, EQ, 0) GOTO 450
CUROLN = CHDIR(I.II)
NL = CHDIR(2.II)
NG = 9
LPG = 1024/NS
DO 2360 I = 1, NL
CALL GET(OBUF, CUROLN, IB)
CUROLN = CUROLN + 1
DO 2301 III = 1, LPG
NG = NG + 1
DO 2330 J = 1, NS
LBUF(J, NG) = OBUF(J + IB)
CONTINUE
IB = IB + NS
IF(NG, GE, NL) GO TO 2311
CONTINUE
EXTRACT INTEGRATED OPTICAL DENSITY
G10D = 0.0
MPG = 0
DO 451 I = 1, NL
DO 452 J = 1, NS
IVAL = LBUF(J, I)
IF(IVAL, EQ, 0) GO TO 452
G10D = G10D + FLOA T(IVAL)
MPG = MPG + 1
CONTINUE
CONTINUE
CONTINUE
G10D = G10D / FLOAT(MPG)
CONTINUE
CLASSIFY THE F-GROUP
DO 460 I = 1, 4
DENMAX = 0.0
DO 461 J = 1, 4
IF(GB(J), LE, DENMAX) GO TO 461
IOBJ = IFCHR(J)
IF(I0BJ, LT, 1) GO TO 461
DENMAX = GB(J)
JS = J
CONTINUE
OFS(59 * JS + 1) = IOBJ
GB(JS) = 0.0
CONTINUE
DO FOR G-GROUP
DO 510 IX = 1, 4
II = IIGCHR(IX)
IF(II, EQ, 0) GOTO 510
C 4,122,518

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

C DO 1330 I=1,NL.
CALL CET(0BUF, CUROLN, IB).
CUROLN=CUROLN+1.
DO 1381 II=1,LPC.
NG=NG+1.
DO 1330 J=1,NS.
LRGBUF(J, NG)+=0BUF(J+IB).

1330 CONTINUE.
IB=IB+NS.
IF(NG, CE, NL) GO TO 1311.

1300 CONTINUE.
1311 CONTINUE.

C EXTRACT DENSITY PROFILES.

C DO 500 K=1, NL.
NP=0.
IPROF(K)=0.
DO 500 L=1, NS.
IVAL=LRGBUF(L, K).
IF(IVAL.EQ.0) GO TO 501.

500 CONTINUE.

C ANALYZE THE BANDS IN THE C-GROUP.

C DO 511 K=1, NL.
IPROF(K)=0.

511 CONTINUE.

C NLM2=NL-2.
DO 512 K=3, NLM2.

512 CONTINUE.

C IV=0.
DO 513 KK=1, 5.
IVAL=IV+IPROF(K+KK-3).

513 CONTINUE.

C IPROF(K)= IV.

514 CONTINUE.

C MAX=IPROF(J).
DO 514 K=4, NL.
IVAL=IVAL+IPROF(K).
IF(IVAL.GE.MAX) GO TO 514.
MAX=IVAL.
KBAND=K.

515 CONTINUE.

C KBAND=KBAND+NSAME/2.

516 CONTINUE.

C NSAME=0.
DO 515 K=KBAND, NLM2.
IVAL=IVAL+IPROF(K).
IF(IVAL.LE.MAX) GO TO 516.

515 CONTINUE.

C CLASSIFY THE C-GROUP.

C DO 521 I=1, 4.
MAXBP=0.
DO 522 J=1, 4.

522 CONTINUE.

C IF(KGBAND(J).LE.MAXBP) GO TO 522.

512 CONTINUE.

C MAXBP=KGBAND(J).
JS=1.
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>522</td>
<td>CONTINUE</td>
<td>OFS(43-1)=IBJ</td>
</tr>
<tr>
<td>521</td>
<td>CONTINUE</td>
<td>KGAND(JS)=0</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>CLASSIFY THE D-GROUP</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>DO 601 I=1,6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMAX=0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DO 602 J=1,6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IF(RATIO(J).LE.RMAX) GO TO 602</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I0BJ=I0J(J)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IF(I0BJ.LT.1) GO TO 602</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMAX=RATIO(J)</td>
</tr>
<tr>
<td></td>
<td>602</td>
<td>CONTINUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFS(43-1)=IBJ</td>
</tr>
<tr>
<td></td>
<td>601</td>
<td>CONTINUE</td>
</tr>
<tr>
<td></td>
<td>600</td>
<td>CONTINUE</td>
</tr>
<tr>
<td>1899</td>
<td>FORMAT</td>
<td>(118)</td>
</tr>
<tr>
<td>2000</td>
<td>FORMAT</td>
<td>(13,3,39I3)</td>
</tr>
<tr>
<td>3000</td>
<td>FORMAT</td>
<td>(ICHROMOSONE ' ,12)</td>
</tr>
<tr>
<td>4000</td>
<td>FORMAT</td>
<td>(19F10.2)</td>
</tr>
<tr>
<td></td>
<td>CALL</td>
<td>WPARE(92,KPAR,9)</td>
</tr>
<tr>
<td></td>
<td>CALL</td>
<td>CLOSE(0BUF)</td>
</tr>
<tr>
<td></td>
<td>CALL</td>
<td>APHASE(9)</td>
</tr>
<tr>
<td></td>
<td>CALL</td>
<td>EXIT</td>
</tr>
<tr>
<td></td>
<td>END</td>
<td>END</td>
</tr>
</tbody>
</table>

**SUBROUTINE ABNORM**

COMMON /C1/ NCH,CHIR,SMOBUF,LRGBUF,HG,ML,CUROLN
COMMON /C1/ SST,EST
COMMON /C1/ MAP,SMIT,SPAREA
COMMON /C1/ MCHR
COMMON /C1/ CFOUR,PHI
COMMON /C1/ BAND,AHIS
COMMON /C1/ OBUR,BUF
BYTE LRGBUF(50,90)
BYTE DBUF(124)
BYTE TEMP8
LOGICAL OPT1,OPT2,OPT3
INTEGER IPROF(90)
INTEGER UFETUR(20)
INTEGER TEMPI
INTEGER SPARE(5)
INTEGER BMAX
INTEGER BAND(180)
INTEGER BPL
INTEGER CHIR(15,60)
INTEGER CUROLN
INTEGER SST(76),EST(76)
INTEGER PAR(18)
REAL CFUOR(60,15),PHI(60,15)
REAL AHIS(180)
REAL COSINE(150),SINE(150)
REAL AFOUR(60,15),BFUOR(60,15)

**EQUIVALENCE** (SPARE(2),BPL)
EQUIVALENCE (CFUOR(1,1),AFOUR(1,1))
EQUIVALENCE (PHI(1,1), BFOUT(1,1))
EQUIVALENCE (NL, NL0)
EQUIVALENCE (HUMOB, II)
DATA L0PT1,'T1'/
DATA L0PT2,'T2'/
DATA L0PT3,'T3'/
DATA LOP,'OP'/

CONTINUE

CALL AFILE(UBUF, 1, 'RCE', *13, 'I3')
CALL OPEN(UBUF, 1, 6, 'RCA')
CALL GLABEL(UBUF, SPAR, IB)

CALL GET(UBUF, 1, IB)
CALL MYL(UBUF(IB+3), SPIH, 12)
CHCR=IV(UBUF(IB+17))
CALL MYL(UBUF(IB+31), CHDIR(1,1), 2*15+30)
CALL GET(UBUF, 1, IB)
CALL MYL(UBUF(IB+31), CHDIR(1,31), 2*15+30)

DO 310 II=1, NCHR
  JFETUR(1)=0
  WRITE(5,3000)II
  CONTINUE
DO 330 I=1, NL
  CALL GET(UBUF, CURLH, IB)
  IF (I.EQ.1) LASTIB=IB
  CURLH=CURLH+1
  DO 301 III=1, LPG
  NG=NG+1
  CALL SSWITCH(0, JAY)
  IF (JAY.EQ.1) WRITE(5,3000) (UBUF(J+IB), J=1, NS)
  CONTINUE
DO 330 J=1, NS
  LRGBUF(J, NG)=UBUF(J+IB)
  CONTINUE
  ID=IB+NS
  IF (NG.EQ. NL) GO TO 311
  CONTINUE
CONTINUE

MAIN LOGIC

NS=NS-1
C C
DO 500 K=1, NL
C MAXLOC=1
LINMAX=LRGBUF(1, K)
C DO 510 L=2, NS
IF(LRGBUF(L, K), LE, LINMAX) GO TO 511
LINMAX=LRGBUF(L, K)
MAXLOC=L
511 CONTINUE
510 CONTINUE
C C
JTH=LINMAX/2
IPROF(K)=NS
C DO 530 L=1, NS
IF(LRGBUF(L, K), GE, JTH) GO TO 532
IPROF(K)=IPROF(K)-1
530 CONTINUE
532 CONTINUE
C C
DO 540 L=1, NS
LL=NS+1-L
IF(LRGBUF(LL, K), GE, JTH) GO TO 542
IPROF(K)=IPROF(K)-1
540 CONTINUE
542 CONTINUE
IF(IPROF(K), GE, NS) IPROF(K)=0
C 500 CONTINUE
C C
IPTOT=0
MLM2=NL-2
DO 600 K=3, MLM2
IPTOT=IPTOT+IPROF(K)
600 CONTINUE
C IPAVE=IPTOT/MLM2
C DO 620 K=3, MLM2
IPROF(K)=IPROF(K)-IPAVE
IF(IPROF(K), LT, 0) HL0=HL0+1
620 CONTINUE
C IPROF(1)=1
IPROF(2)=1
IPROF(NL)=1
IPROF(NL-1)=1
C C
IF(JAY.EQ.1) WRITE(5, 1306) (IPROF(L), L=1, NL)
C C
TYPE 1 DETECTION-
AT LEAST FIVE NEG NUMBERS IN A ROW IN PROFILE
C C
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-
SEXED, 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
700 IF(K.GT.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

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

SCAN TO POS NUMBER

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

CONTINUE

M:MINUS=1
DO 728 KL=1,6
K=KL+2
IF(K.GT.KL) GO TO 721
IF(IPROF(K+1).GE.B) GO TO 720
M:MINUS=M:MINUS+1
720 CONTINUE
721 CONTINUE

IF(M:MINUS.LT.4) GO TO 700

IFETUR=IFETUR+1
KFETUR=2
JFETUR(IFETUR)=KFETUR
K=K+4
K=K+1
731 IF(IPROF(K+1).LT.0) GO TO 731
GO TO 700

CONTINUE

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

FORMAT(' FEATURES: ',2013)
C
IF (HL0 .GT. NL) GO TO 940
GO TO 941
CONTINUE
C
WRITE(5,1094) FORMA(' NOT A CHROMOSOME')
C
GO TO 950
C
CONTINUE
IF (IFETUR.EQ.0) GO TO 950
IF (IFETUR.EQ.1) GO TO 950
IF (IFETUR.EQ.2) GO TO 950
IF (IFETUR.EQ.3) GO TO 950
GO TO 940
C
CONTINUE
GO TO 950
C
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)
FORMAT(' CHROMOSOME IS BICENTRIC OR BADLY TWISTED')
1292 FORMAT(' CHROMOSOME MAY BE BICENTRIC')
1492 FORMAT(' CENTROMERE LOCATION DOUBTFUL - PROBABLY NORMAL')
GO TO 950
C
CONTINUE
IF (IFETUR(1).EQ.1) GO TO 951
WRITE(5,1291)
GO TO 951
C
CONTINUE
WRITE(5,1191)
C
CONTINUE
FORMAT(' SINGLE CENTROMERE IS WELL DEFINED - NORMAL')
1291 FORMAT(' SINGLE CENTROMERE POORLY DEFINED - PROBABLY NORMAL')
GO TO 950
C
CONTINUE
IF (HL .LT. 30) GO TO 952
WRITE(5,1932)
GO TO 952
C
CONTINUE
WRITE(5,1901)
C
CONTINUE
FORMAT(' NO CENTROMERE FOUND /.'
1292 FORMAT(' NO CENTROMERE FOUND - POSSIBLE ACENTRIC')
C
CONTINUE
C
CONTINUE
C
WRITE (5,7030)
CALL APHASE(13)
CALL CLOSE(80BUF)
C
CONTINUE
1000 FORMAT(I10)
2003 FORMAT(3X,3913)
3000 FORMAT('CHROMOSOME ',12)
3030 FORMAT('I')
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 connected to control said microscope to successively present said plurality of dispersed metaphase chromosome spreads to the optics of said microscope,
   storing the location of each spread, thereby automatically prescanning said slide for chromosome spreads,
   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 operating 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:
   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 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.

* * * * *