AUTOMATED CLINICAL SYSTEM FOR CHROMOSOME ANALYSIS

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

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

ABSTRACT

An automatic chromosome analysis system is provided wherein a suitably prepared slide with chromosome spreads thereon is placed on the stage of an automated microscope. The automated microscope stage is computer operated to move the slide to enable detection of chromosome spreads on the slide. The X and Y location of each chromosome spread that is detected is stored. At the conclusion of this searching operation, the computer directs the microscope to again sequence through the chromosome spread locations in response to the stored X and Y locations. At this time an operator can view these spreads to determine which ones are worthwhile and which ones are not. He is provided with an accept-reject switch. The microscope stage thereafter again sequences through only the accepted chromosome spreads, and this time a digital photograph of each of the chromosome spreads is made and entered into the computer storage. The computer thereafter measures the chromosomes in a spread, classifies them by group or by type and also prepares a digital karyotype image. This image is converted to analog form, displayed and printed out and constitutes a primary output of the system. Chromosome measurement data is filed in an interactive data base for subsequent statistical analysis. The computer system can also prepare a patient report summarizing the result of the analysis and listing suspected abnormalities.
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 and reliability of the karyotype improves, fetal karyotyping through amniocentesis may become a routine part of prenatal care. Screening studies on large populations offer the potential of uncovering the effects of industrial and environmental poisons, aging, and long term low dosage ionizing radiations. These factors may manifest themselves in subtle structural aberrations requiring detailed analysis of the chromosome morphology. The ability to process cells rapidly and inexpensively would also aid in the detection of mosaicism, in which two or more cytogenetically distinct lines of cells exist in the individual.

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

OBJECTS AND SUMMARY OF THE INVENTION

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

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

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

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

After all of the chromosome spread locations have been inspected, the microscope is again actuated to move its stage so that each one of the accepted chromosome spreads are passed under the optics of the microscope for the purpose of enabling digital pictures of the various spreads to be generated and stored in memory. The computer then proceeds to locate and analyze the chromosomes in each of the chromosome spreads by measuring the chromosomes, classifying them by group or by type and preparing a digital karyotype image format. This image is then converted to pictorial form and displayed on an image display tube to enable any corrections, if needed. A joystick control is provided so that a cursor, which is produced on the display tube screen can have its position moved whereby the operator 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 diagram of a chromosome analysis system, in accordance with this invention.

FIG. 2 is a schematic diagram of a computer used to drive the microscope in a preferred embodiment.

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

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

FIG. 2 is a block schematic diagram illustrating the circuits used in controlling movements of the motorized stage of the microscope. The stage is driven in the X, Y and Z directions respectively by three motors 50, 52, and 54. The Z motor is driven in the -Z or -Z direction by signals from the computer in response to focus drive apparatus 56 in the Spread Detector and Autofo-

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 disenabling the one of the two And gates which was enabled by the Y flip flop 92. Also, no further pulses are applied to the Y motor drive register.

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

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

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

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

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

The television camera 30 scans a spread seen through the microscope and displays this on a monitor 32. The spread may also be viewed through the microscope viewing lens 28. Thus, during the editing mode, the operator can use either or both means for editing.

After the editing mode, the system goes into its scanning mode. It sequences the microscope stage to the locations of the chromosome spreads which have been accepted, automatically focuses the image, and digitizes the spread image into a scan data set (SDS). The digitization is accomplished by the television camera 30 and the image digitizing system 36. This equipment is commercially available and, by way of example, in an embodiment of the invention which was built, this equipment, called the model 100 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
7

and fragmentation. The karyotype is also displayed to allow the operator to verify correct classification. After the karyotype has been approved by the operator, it is formatted for output, combined with the spread image and copied into one of the output data sets. The analysis phase processes cell images one at a time from a raw spread image to digital karyotype.

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

The operator then, by means of the typewriter 27, types an instruction to the computer in response to which the computer performs the required operation.

The gray level display system presents a picture of the chromosome spread which has been organized into the standard karyotype format for ease of diagnosis. Similar pairs of autosomes are collected and numbered with homologous pairs being numbered from 1 - 22 and similar pairs being collected into groups lettered A through G based on similar morphology.

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

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

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

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

The chromosome identification technique, that is a general description of a method of chromosome identification which was implemented by the programming is described in a volume entitled Chromosome Identification edited by Torbjorn, Casperson and Lore Zech, which is a publication of Hans Editions 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 3 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 supplied by the manufacturer, as well as maintenance and interactive statistical analysis. Automatic karyotyping runs under a specially written supervisor (CALMS).

CALMS

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

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

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

Scan and analysis consist of 20 phases that reside on disk in core-image format. One phase at a time is loaded into locations 60000 to 156777 and called by the supervisor. (Locations 157000 to 157775 may be used for COMMON storage.) Each analysis phase has a unique identifying number, from 1-20. Scan is phase 1, binary is phase 2, etc. The data set CALMS.OVR is used to store the phases. Each phase required 63 blocks. Program OVB is used to store a phase in CALMS.OVR after it has been linked with a bottom switch of 60000.

The following batch stream builds scan (phase 1):

```
SJOB [2,2]
SR LINK
*DK5:PHASE/CO.LP/15/SH<CALMS,STB,-
SCAN,EXIT,FTNLIB/B:60000/E
SR OVB
*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.

<table>
<thead>
<tr>
<th>NSDS</th>
<th>3 # of scan data sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>NHDS</td>
<td>2 # of hardcopy data sets</td>
</tr>
<tr>
<td>SDU</td>
<td>5 scan disk unit (DK3)</td>
</tr>
<tr>
<td>HDU</td>
<td>3 hardcopy disk unit (DK3)</td>
</tr>
</tbody>
</table>

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 NFSDS 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 partition of CALMS

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

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

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

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

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

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

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

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

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

The next key causes edit to move to the next spread without altering the accept-reject state. The last key causes edit to move to the previous spread. The edit finishes when the end of the queue is reached, or when the operator presses edit-end. The queue is then sorted according to rating. The highest rated spread is moved to the start of the queue, and the stage is moved to this spread in preparation for the first scan. At this point, the queue and other critical information is saved, by calling WPARAM for phase one. This allows a scan restart at a later time.

GNSTS is the entry point for the "get next spread to scan" subroutine. It initiates a motor move and auto-focus on the next spread in the scan queue. CSPQ contains the current location. It also enables a spiral search.

FLOW CHART 2 CALMS Data Flow
if search start is pressed. The spiral search is useful in locating a spread close to known coordinates. It takes steps of 50 microns in a spiral pattern and focuses when there is something to focus on. The operator can halt it or resume it with the search halt and resume keys.

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

(See Flow Chart 3).

FLOW CHART 3 Focus

SCAN

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

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

SCAN sets priority 6 during the pixel digitization loop to prevent interrupts from other devices which would cause it to lose "sync." Hardcopy interrupts (which are at priority 7) are the only ones allowed, due to their critical nature. When the hardcopy is running, the SCAN takes several seconds longer.

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

The operator may enter additional information as the SCAN ID.

### BINARY

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

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

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

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

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.
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.
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.

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

The segmentation process begins with a line by line scan for pixels above the computed threshold. This task is performed by the subroutine ISSEG, 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
   XM IN = Minimum sample coordinate
   LDW = Last chromosome record written

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:

```

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 (MASKPHI).

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

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

INT1
Purpose: To correct errors in object isolation.

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

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

INT1 communicates with CALMS by calling 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 CLASYST results
Rec 4-7: Reserved for BANDS results
Rec 8 & following records:
   Roated chromosome images
```

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

```
Rec 1 fields:
   Integer CHDIR (15,60)
   CHDIR (1,1) - CHDIR (15,30) on Rec 1
   CHDIR (1,31) - CHDIR (15,60) on Rec 2
```

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 x 88. The maximum allowable size after rotation is 88 x 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:

| CHDIR (1,1) | 1st Rec. # of 1st rotated image, or zero if rejected |
| CHDIR (2,3) | # of lines in image |
| CHDIR (3,3) | # of samples in image |
| CHDIR (4,3) | Length of chromosome |
| CHDIR (5,3) | IOD of chromosome |
| CHDIR (6,3) | Centromeric Index by Length |
| CHDIR (7,3) | Centromeric Index by Density |
| CHDIR (8,3) | Centromeric Index by Area |
| CHDIR (9,3) | Centromere line |
| CHDIR (10,3) | Perimeter of chromosome |
| CHDIR (11,3) | Area |
| CHDIR (12,3) | #/Area |
| CHDIR (13,3) | Centromere Location Method |
| CHDIR (14,3) | Available for expansion |
| CHDIR (15,3) | Available for expansion |

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.

Format of KDATA: RECLEN = 1024

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

Line 2 | Directory Record 1 for Source 2 spreads |
| |
| Line 15 | Directory Record 1 for Source 15 spreads |

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

Line 16-500 Data Records and Directory Records, if necessary
FOUR file, RCR. Length and centromeric index are passed to and saved in records along the line is used. The window of eight points. The waveform values for eight harmonics. From these, the slope of the parabola. Using three points perpendicular along the parabola, and using three points perpendicular to the boundary points and recalculates the chromosome some length. The waveform is determined by sampling along each line, using a moving window of eight points. The output is written on disk, with an option to also write it on tape if switch is up.

The picture on disk is written in hardcopy format, and the raw spread for counts. It checks for the KG parameter followed by the disk unit and filename for the raw spread. MASK2 is also used to combine the numbered spread and karyotype of the rotated chromosome images. 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,BUFZI@BFLAG,- MODE,LNAME)
CALL GET (MVB,LINE,INDEX[,NORA])
CALL PUT (MVB,LINE,INDEX)

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:

MV$\text{B}$ is the location of the MVB for the data set. The user must reserve enough core for his buffers immediately following the MVB.

BUFSIZ$\text{Z}$ 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>
<tr>
<td>5</td>
<td>Tape Output</td>
</tr>
</tbody>
</table>

0, 1 and 2 are functionally equivalent.

0 and 0 are functionally equivalent.

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.

LNAME is a three character logical name for the data set that can be used to assign it to a file with the $\text{AS}$ 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.
READ and WRITE can be used when RECLEN = 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>4</td>
<td>Active Buffer Address</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>Word Count</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>Function/Status</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>Words Not Transferred</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>IBN</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>Inactive Block #</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>Link Pointer</td>
</tr>
<tr>
<td>10</td>
<td>22</td>
<td>Logical Name</td>
</tr>
<tr>
<td>11</td>
<td>30</td>
<td>Index</td>
</tr>
<tr>
<td>12</td>
<td>32</td>
<td>CRC</td>
</tr>
<tr>
<td>13</td>
<td>44</td>
<td>File Name (Word 1)</td>
</tr>
<tr>
<td>14</td>
<td>46</td>
<td>File Name (Word 2)</td>
</tr>
<tr>
<td>15</td>
<td>48</td>
<td>Extension</td>
</tr>
<tr>
<td>16</td>
<td>50</td>
<td>UIC</td>
</tr>
<tr>
<td>17</td>
<td>52</td>
<td>WNT</td>
</tr>
<tr>
<td>18</td>
<td>54</td>
<td>Words Not Transferred</td>
</tr>
<tr>
<td>19</td>
<td>56</td>
<td>Start Block #</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>Blocks In The File</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td># Label Records</td>
</tr>
<tr>
<td>22</td>
<td>5</td>
<td>RECLen</td>
</tr>
<tr>
<td>23</td>
<td>7</td>
<td>BLKSIZ</td>
</tr>
<tr>
<td>24</td>
<td>9</td>
<td>BUFSIZ</td>
</tr>
<tr>
<td>25</td>
<td>11</td>
<td>BPE</td>
</tr>
<tr>
<td>26</td>
<td>13</td>
<td>BPL</td>
</tr>
<tr>
<td>27</td>
<td>15</td>
<td>BPF/MD</td>
</tr>
<tr>
<td>28</td>
<td>17</td>
<td>Write Flag (Last Block # To Be Written + 1)</td>
</tr>
</tbody>
</table>

LABEL

Callable Entry Point Names: GLABEL, PLABEL

Purpose: To obtain and store label information on disk data sets.

These subroutines are used in conjunction with MVIO, when processing labeled data sets. They obtain or store the label parameters, and also set up the BLKSIZ, RECLen, NLR, and BPE 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) = NIL, # 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 (4000), B (4000)
CALL OPEN (A,2048,1,MBV')
CALL OPEN (B,2048,1,MBV'2)
CALL GLABEL (A,SPAR,A)
CALL PLABEL (B,SPAR,A(IA + 1))

GLABEL can also be used with an unlabeled data set. It will return N2 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. ERAA gives a single error message (A367), instead of the individual messages normally given by the FORTRAN error routine ERE.

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 RS 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 (MV, DUNIT, FILP2, GRP, USR)
MV is the mini-VICAR block for the dataset.
DUNIT is the disk unit (1 for DK1; 2 for DK2; etc.)
FILP2 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 (A,1, 'PIC ', 5,5)
Assigns DK1:PIC[5,5] to MV

PARAM

Callable Entry Point Names: PARAM, PARBUF
Purpose: To read free-field parameters from the key-
board 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 pre-
ceded by a minus sign.
2. Word alphameric — first character must be
alphanumeric. The character string is padded with trailing
blanks if less than 4 characters. It is truncated if more
than 4 characters.
3. Variable length alphameric — the string is en-
closed in apostrophes. If an odd number of characters,
the last word is padded with a trailing blank. If an apos-
trophe is desired, type two consecutive apostrophes.
The last 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 pa-
rameter 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 latera 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.

QPRINT

Purpose: To print a message, with automatic buffer-
ing
QPRINT waits for completion of a previous print,
blocks 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. (maxi-
— 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

IV is a function and the other entry points are subrou-
tines. These routines consider bytes to be positive num-
bers 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 (II, I2, N) (Move WORD)
Move N WORDS starting at I1 to I2
CALL ZIA (IBUF,N) (Zero INTEGER array)
Zero N WORDS starting at IBUF
CALL ITLA (INT,L1,N) (INTEGER to LOGI-
CAL 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 HCPK (BUF8, BUF4, NE)
CALL MVW78 (BUF4, BUF8, NW)
BUF8 is the LOC of 8-Bit Data
BUF4 is the LOC of 4-Bit Data
BUF7 is the LOC of 7-Bit Data
NE is the number of elements to pack or unpack. NW is the number of words to convert from 7-Bit format to 8-Bit format.

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

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

CALL RPARAM (NP, PAR, NP MAX [,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 A PHASE (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 DECL EAR to erase the entire gray scale.
CALL D LINE (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,L INE,S AMP) 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 motor move to X,Y CALL IXYREL (XDEL) Initiates a relative X move CALL IXYABS (Y,Y) Initiates a relative Y move CALL IFREL (FDEL) Initiates a relative F move CALL CPOC (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.
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 assending order, first by line coordinate, and second by sample coordinate.

CALL SORTIN(EDGE,N,IND)

N is the number of end points.

Upon return, IND=0 if the sort was successful, ≠ 0 if not.

EDGE is a buffer area 4N words long. The first 2N words contain the end point coordinates for the chromosome. The remaining 2N words is used as a work area for the bucket sort routine.

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

ERASE

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

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

WORD points to the buffer area containing the binary image.

EDGE points to the end point coordinates for the chromosome.

N2 is the number of coordinates in EDGE (two per coordinate pair).

NW is the number of words per line in the binary image.
AREA is returned as the number of sample points contained in the chromosome.

ORIOB
Purpose: Orient objects.
CALL ORIOB(IBUF, OBUF, EF, CHDIR, NL, OROT, 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.

ACCSUB
Purpose: Accumulate area and density by sample for rotated object.
CALL ACCSUB(BUF, NL, MS, AREAA, DENA, TAREA, TDEN)
The chromosome is located in BUF and is NL by NS. AREAA is the area accumulator by sample and DENA is the density by sample. TAREA and TDEN are the area and density totals for the object.

EPROT
Purpose: Rotates chromosome endpoints.
CALL EPROT(SIN, COS, 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, YMAX).

OBROT
Purpose: To rotate objects.
CALL OBROT(XMIN, XMAX, YMIN, YMAX, COS, SIN, IBUF, OBUF)
OBROT rotates the object in IBUF into OBUF. COS and SIN specify the rotation angle and XMIN, XMAX, YMIN, YMAX give the enclosing rectangle of the object in unrotated coordinates.

CHROUT
Purpose: To stand objects up and move them into the output buffer.
CALL CHROUT(IBUF, OBUF, NS, NL, LPB, IP, FLG, BUFSZ)
CHROUT rotates the object in IBUF (NS X NL) by ± 90° in to OBUF for output. LPB gives # of lines that will fit into OBUF, FLG tell which way to rotate object and BUFSZ is NS for IBUF.

KURSOR
Purpose: To locate information about the karyogram.
CALL KURSOR(X, Y, L, S, SLID, SLCL, SLCS, N)
X, Y, L, S are the cursor coordinates returned by cursor. SLID is the slot ID indicated; SLCL and SLCS are the slot center coordinates; and N is the object ID of the object residing in the slot.

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

LF=12
PSU=177776
MPH=24
BINPH=2
INTPH=7
MOBP=8
MASKPH=13
NSD=3
MDB=2
SDD=5
HDD=3
MVBS12=56
LABS12=12
IBH=12
JBS=14
CURRE=30
SBH=46
RECLN=54
BUFF12=60
DBP=62
DBFLAG=64
SUR =177570
CURRENT_ORIGIN IS 30000
SOURCE: .WORD 0.
MSCAN: .WORD 0.
PHUM: .WORD 0.
PANUM: ASCII /12345678/ .PATIENT IDENTIFICATION
SLNUM: ASCII /12341/ .SLIDE IDENTIFICATION
PHASE: ASCII /F/ .PATIENT SEX
SPN: WORD .CURRENT SPREAD NUMBER
CSPQ: WORD SPQ+. .CURRENT LOC IN SPQ
SPR: BLKB 300. .SPREAD RATING
SPQ: BLKW 600. .SPREAD QUEUE (X,Y)
LPNUM: WORD 10. .LENGTH OF PANUM IN BYTES
LSLNUM: WORD 8. .LENGTH OF SLNUM
NPHASE: WORD .NEXT PHASE TO BE CALLED
PAGE: .PAGE
CALMS: CALL TYPE,<MSC2,ZERO> .ASK ABOUT RESTART

MOV $6, RI

CLED: CALL LED,<LINE,ZERO> .INITIALIZE LEDS
INC LINE

S09 R0,CLED

.CALL OPEN,<MVB,PSIZE,ZERO,ZERO,LMAM> .OPEN OVERLAY DATASET
MOV $PSIZE,MVB+RECLEN

.CALL OPEN,<PASV,PARSIZ,ZERO,TWO,PLNM> .PARAMETER DS
MOV PASIZ,PMVB+RECLEN
MOV $14,DNPTAB+INTPH+INTPH-2 .SET TO READ 4 CURSOR ADJUANTS
MOV $512, HPTAB .ALLOW RESTART
MOV $512, DNPTAB
MOV $HHS.RB .SET UP THE START BLOCK NUMBERS FOR
MOV $HSSN.RI .CALL HARD COPY DATA SETS

C05. MOVB R0,FILPEX+1

BSB $60,FILPEX+1 .CONVERT TO ASCII
.CALL AFILE,(A,LHBU,FILPEX,TUG,TUG)

.CALL OPEN,(A,TWELVE,ZERO,ZERO,LNAME)

MOV A+SSN,(R1)+ .SAVE THE START BLOCK NUMBER

S06. MOV R0,C05

.CALL SEARCH .INITIAL CALL TO SEARCH
JSR PC,PBAR .PROCESS RESTART AND PHASE PARAMETERS

TST HSCAN
BNE S10 .IF A SCAN RESTART

S06. JSP PC,TNSCAN .IF HSCAN=0 .ALLOW SEARCH RESTART

S10. TSTB $FLAG ."IDLE LOOP" STARTS HERE
BNE S44 .IF SEARCH RUNNING

TST HSCAN

S13. MOV $NDSR,R0 .LOOK FOR SCAN BS

S15. MOV $NDSR,R2

MOV HFSDS,R2 .FULL SCAN DATASETS

S22. MOV S22 .IF ZERO

S20. CMPB R0,(R1)+ .CHECK R0 AGAINST ALL FULL DATASETS

BEG S25 .IF DATASET IS IN USE

S29. MOV S29 .IF MORE TO CHECK

S22. MOVB R0,0R1 .OK TO USE THIS DATASET, STORE IT
BR S30 .GO CALL SCAN

S25. MOV R0,S15 .IF MORE TO TEST

S39. MOV R0,FILPEX+1 .CONVERT TO ASCII

MOV R0,FILPEX+1 .STORE IN FILENAME

MVB $S,FILPEX .AS SN
MOV $SU,BUMIT .STORE SCAN DISK UNIT

S31. DEC HSCAN .HSCAN=HSCAN-1
DEC HSCAN

C51. CALL PHASE,<ONE> .SCAN

C51. CALL LED,<SLED,HSCAN>

TST HPHASE

S05. CALL PHASE

BEQ S06 .IF SCAN WAS ABORTED

S315. CALL PHASE,<INTPH>

CLRB HPTAB+HOBPH+HOBPH-2 .CLEAR HOB PARAMETERS FOR INT1

CLRB HPTAB+HOBPH+HOBPH-2 .CLEAR HOB PARAMETERS FOR INT1

CLRB HPTAB+B1PH+B1PH-2 .CLEAR BINARY PARAMETERS

C513. CALL PHASE,<INTPH> .CALL INT1 FOR THE PRE-F03 ADL

CMP HPHASE,.B1 .TEST FOR RESCAN OR ABORT

BEQ S31 .IF PFS3

BET S06 .IF SPREAD WAS ABORTED IN INT1

BNE S01 .IF NOT A COUNT

CALL PHASE,HPHASE .CALL MOD
C4H4,1223 18

N PHASE

A Y IL HS

K PH

I H S

PHASE, PPHASE

i CALL PHASE, PPHASE

i CALL BINARY

JR S86

INC HFSDS

i CALL LED, (AGLED, HFSDS)

UPDATE ANALYSIS QUEUE

BR S86

TSTB PFLAG

i SEE IF PATIENT REPORT WAS REQUESTED

BR S86

CLRB PFLAG

i CALL THE PATIENT REPORTER

TSTB EFLAG

i SEE IF TEXT EDITOR WAS REQUESTED

BR S86

CLRB EFLAG

i CALL THE EDITOR

TSTB FFLAG

i SEE IF FORTN WAS REQUESTED

BR S86

CLRB FFLAG

i CALL FORTN

JR PS18

GO BACK FOR RESTART

MOV #SDS.R1

i SEE IF READY FOR ANALYSIS

MOV HFSDS.R2

BR PS18

MOV #S.FILIP

MOV #S3U.DUMIT

MOV #R1.FILIP+1

MOV #R1.R0

MOV #R0.FILIP+1

i SAME DATA SET 0 IN R0

BIS R80

i CONVERT TO ASCII

DEC HFSDS

DEC R2

BEQ S86

i BR IF NO DATASET NUMBERS TO MOVE

INC R1

S86

BR R2.S86

MOV #PHASE, #PHASE

i START ANALYSIS WITH PHASE 2 OR DEBUG PHASE

MOV #DNPTAB.R0

MOV #DNPTAB.R1

MOV #PHL.R2

S86

MOV (R0)+, (R1)+

i MOVE DNPTAB TO HPTAB

S86

BR R2.S86

TSTB RFLAG

i TEST RESTART FLAG

BR R80

BR IF NOT SET

CLRB RFLAG

JR PS18

TST MHPHASE

i TEST FOR ABORT (OR ANALYSIS RESTART)

BR R80

BR IF NOT AN ABORT (MHPHASE=0)

CLRB LED, (AGLED, HFSDS)

JMP S86

S85

CMP MHPHASE, #MASKPH

BR R80

BR IF NOT THE MASK PHASE

END OF ANALYSIS LOOP

#LOOK FOR HACOPY DATA SET

#LOOK FOR HARDCOPY DATA SET

MOV #HDS.R0

i NO COPY DATA SETS

MOV #HDS.R1

MOV #HSBN.R2

S86

TSTB (R1)+

i BR IF DATASET IS EMPTY

TST (R2)+

STEP HSBH

S86

BR R8.C80

i BR IF MORE TO TEST

BR C15

i NOE AVAILABLE, KEEP LOOKING

MOV #60.R0

i CONVERT TO ASCII

MOV #R0.FILIP+1

i STORE IN FILENAME
MOV B *H,FILPEX
MOV #H,BUHIT
INC -(R1)
MOV R1,CMS
MOV R2,CBH
CALL PHASE,<PHASE>
INC HNCOPY
CALL LED,(HQLLED,HNCOPY)
CALL LED,(AQLED,NF5)
TEST HFLG
BEQ JS10
JS10 JMP S10

SECTION PLOC=G000

MOV R2,(R5),R1
MOV R1,PHN
STORE PHASE NUMBER
INC R1

BEQ JS10
JS10

CALL ZIA,(15700H,0377)
ZERO 510 BYTES OF UPPER CORE
CALL READ,(MYB,PHN,INDEX,PLOC)
READ PHASE PHN INTO PLOC
ASL R1
ADD #PLITE-4,R1
BIS ORI,0#LITES
TURN ON THE LITE FOR THIS PHASE
CALL PLOC,(DUNIT,FILPEX)
CALL THE PHASE, GIVING IT A DATA SET NAME
BIC #14176#LITES
TURN OFF ANALYSIS LITES
RTS R5

PPAR CALL PARAM,(NP,PAR,MAXNP)
JSR PC,PPAR

DECRH HIGH
BRN PFEXT
BR IF NP WAS 0
BR IF NO RESTART
CMPB PAR,"Y
BEQ RSTRT
BR IF A RESTART
CPL PP40
PHASE NUMBER AND PARAMETERS

PFEXT RTS PC
RSTRT TESTING
BR IF SEARCHING
CALL TYPE,(M4,ZERO)
ASK ABOUT SCAN RESTART
CALL PARAM,(NP,PAR,MAXNP)
CPL PARBF.0"Y
BR IF NO SCAN RESTART
CALL RRPARAM,(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,#1
BNE POP2
CMP R0,HS CAN
BGT POP2
BR IF NUMBER LE 1
DEC R0
LEGAL NUMBER, DECREMENT IT
MOV R0,SPH
STORE IT
SUB R0,HS CAN
ADJUST HSCANS
ASL R0
ASL R0
N=4
ADD R0,CSPQ
ADJUST CSPQ
PP02 CALL GHSTS
GET FIRST SPREAD TO SCAN AND ENABLE SPIRAL SEARCH
CALL LED,(SOLED,HSCANS)
PP05 CALL TYPE,(M5,ZERO)
ASK ABOUT ANALYSIS RESTART
CALL PARAM,(NP,PAR,MAXNP)
CML PB PARBF.0"Y
BEQ PP15
BR IF NO ANALYSIS RESTART
MOV #HSDS, R0
MOV #SERS, R1
51

4,122,518

52

MOV R0, (R1)+
SUB R0, PP10
CALL TYPE, (R0, ZERO) ; ASK HOW MANY
CALL PARAM, (NP, PAR, MAXNP)
TST PAR
BLE PP12 ; BR IF N LE 0
CMP PAR, #HSDB
JBR IF N GE HSDB
MOV PAR, #HSDB ; STORE HSDB

CALL LED, (#ALED, HSDB) ; DISPLAY ANALYSIS QUEUE
TSTB HCFLG
BNE PPEX ; BR IF HCOPY RUNNING
CALL TYPE, (R0, ZERO) ; ASK ABOUT HARDCOPY RESTART
CALL PARAM, (NP, PAR, MAXNP)
CMPB PARBUF, 'Y'
BNE PPEX ; BR IF NO HARDCOPY RESTART
MOV (HHSB), R0
CALL TYPE, (HDB, R0)
CALL TYPE, (R0, ZERO) ; ASK HOW MANY
CALL PARAM, (NP, PAR, MAXNP)
TST PAR
BLE PP18 ; BR IF N LE 0
CMP PAR, #HSBS
JBR IF N GE HSBS
MOV PAR, #HSBS ; LEGAL N

MOV R0, HHCOPY
SET N HARDCOPY DATA SETS AS FULL
MOV R0, PP20
CALL HCFLG ; SET HARDCOPY OPERATING
BIS 01, HITES ; TURN ON HARDCOPY LITE
MOV #PSYS, -(SP)
JSR PC.MCLED ; SIMULATE INTERRUPT TO HCOPY END RTE
PPEX RTS PC
MOV R0, PAR
JPHASE NUMBER
BLE PERR
CMP R0, #MPH
JBT PERR
ASL R0
ADD #MPTAB-2, R0
J DBPTAB+2*PKUM-2
MOV NP, #R0
ISTORE HP FOR THE PHASE
CALL MPARAM, (NP, PAR+2, PAR)

JPPAR: JMP PPAR
PPAR: CALL TYPE, (#PENS, ZERO)
BR JPPAR

THSCAN: TST NSECAN
BNE CGHS ; BR IF NSECAN NOT ZERO
CALL TYPE, MSG1 ; TYPE 'OK TO START SEARCH'
CALL SEARCH ; ALLOW SEARCH START
RTS PC
CGNS: CALL GNSTS
RTS PC

MVB: .BLKW 16 ; MOV FOR READING PHASE FROM CALMS.OVR
.BYTE 1, 5 ; DBK4:
.RAD50 /DK/
.BLKW 2
.RAD50 /CALMS OVR/
.BYTE 2, 2

L377: .WORD 377
PSIZE: .WORD 32256
.TOO: .WORD 2
INDEX: .WORD 2
.FPHASE: .WORD 2 ; FIRST ANALYSIS PHASE (OR DEBUG PHASE)
.MPHASE: .WORD MAPHPH
.PPHASE: .WORD 16 ; PREP PHASE
.EPHASE: .WORD 17 ; TEXT EDITOR PHASE
.FFAGE: .WORD 19 ; FORTRAN PHASE
.DUNIT: .WORD 0 ; DISK UNIT FOR SCAN OR HARDCOPY DATA SET
.LHAM: .ASCII /OVR/
.PLAN: .ASCII /PAR/
Parsize: WORD 1024.

PLITE: WORD 100, 40, 40, 40, 40, 40, 10, 10, 10, 10, 4, 2, 2, 10010, 10010, 4000

NPTAB: BLKWP NPH

DPNTAB: BLKWP NPH

NHC=172430

HCAD=172432

HCST=172434

HCIV1=448

HCIV2=442

INTPS=349

HCOPY: MOV $7777, A CLEAR ACTIVE BLOCK NUMBER

MOV $7777, A+IBN AND INACTIVE BLOCK NUMBER

MOV $SLAB, A+2 READ LABEL INTO SLAB

MOV $LABSZ: A+BUFSIZ READ LABELSZ BYTES

CLRB A+DBFLAG PREVENT READ-AHEAD

CALL GLABEL (A+SPAR, A)

:READ THE LABEL INFORMATION INTO CORE AND CALCULATE RECLEN.

THEN, SET UP THE NMD TO USE TWO 4096 BYTE BUFFERS IN CORE

FOR THE DISK TO CORE TO HARD-COPY DMA TRANSFER.

MOV $7777, A CLEAR ACTIVE BLOCK NUMBER

MOV $160000, A+2 ONE BUFFER AT 160000

MOV $7777, A+IBN AND INACTIVE BLOCK NUMBER

MOV $170000, A+1BA ONE BUFFER AT 170000

MOV $100000, A+BUFSIZ BUFSIZ=4096 BYTES

MOV $19, A+BPB 18 BLOCKS PER BUFFER

INCB A+DBFLAG SET DOUBLE BUFFERING

CLR LINE ;INITIALIZE LINE COUNTER

MOV $INTY, @HCV1 ;INIT ENTRY ADDR TO HCOPY INT VEC

MOV $INTPS, @HCV2 ;INT SVC PSW TO HCOPY INT VEC

INC DPL IN CASE OF ODD BYTES

MRR BPL WD CNT FOR HCOPY LINE IS DPL+2

ASL DPL2 ;2*8UCHT

MOV A+54, RMB ;RECLEH

SUB DPL, RMB

SUB DPL, RMB

MOV $160000, @HCAD

SUB RMB, @HCAD

ONST: CLR @HCST ;CLEAR HCOPY WD CNT REGISTER

BIT $296, @HCST ;BIT-10 IS HC RUNNING

BEQ ONST ;WAIT FOR PREVIOUS END-FRAME

BIT $1390, @HCST ;BIT-11 IS HC READY

BNE REPL ;BR IF HC READY

HCOFF: MOV @HCST, (SP) ;DISPLAY HCST

MOV $591, (SP) ;A191 MSG, HC OFF

TOT; TOT ;ACTION MSG TO OPERATOR

SP INST ;CHECK HCOPY DEVICE AGAIN

HCOPY DEVICE INTERRUPT SERVICE ROUTINE

INTRY: BIT $10000, @SWR ;BR IF SW 12 UP

DIE HXTLN ;BR IF NEW LINE

COM LPFH

BEQ HXTLN ;BR IF NEW LINE

SUB DPL2, @HCAD ;RESTORE HCAD

REP @HCAD ;REPEAT THE LAST LINE

NXTLN: BEQ ML ;DECREMENT LINE COUNT & TEST

REPL: INC LINE

ADD RMB, @HCAD

BIT $7777, @HCAD

BNE NOGET

INTP, INTR, 0@HCV1 ;INT ENTRY ADDR TO HCOPY INT VEC

MOV @INTPS, @HCV2 ;INT SVC PSW TO HCOPY INT VEC

INC DPL IN CASE OF ODD BYTES

MRR BPL WD CNT FOR HCOPY LINE IS DPL+2

ASL DPL2 ;2*8UCHT

MOV A+54, RMB ;RECLEH

SUB DPL, RMB

SUB DPL, RMB

MOV $160000, @HCAD

SUB RMB, @HCAD

ONST: CLR @HCST ;CLEAR HCOPY WD CNT REGISTER
57
4,122,518
58

HHCOPY: .WORD -170 IN HCOPY QUEUE
SFLAG: .BYTE 0 FLAG FOR SEARCH OPERATING
SDS: .BLKB MSDS ISCAN DATA SET FLAGS
HDS: .BLKB NHDS HARDCOPY DATA SET FLAGS
HCFLG: .BYTE 0 FLAG FOR HARDCOPY OPERATING
RFFLAG: .BYTE 0 FLAG FOR RESTART REQUEST
PFLAG: .BYTE 0 FLAG TO CALL THE PATIENT REPORTER
EFLAG: .BYTE 0 FLAG TO CALL THE TEXT EDITOR
FFLAG: .BYTE 0 FLAG TO CALL FORTRAN
SLZE: .ASCIZ /|J#
MSGL: .ASCIZ /| OK TO START SEARCH:<LF><LF>
MSG2: .ASCIZ /| IS THIS A RESTART?
PEMSS: .ASCIZ /| ILLEGAL PHASE NUMBER!
MSG3: .ASCIZ /| RERUN HCOPY?
MM: .ASCIZ /| RESTART SCAN?
MM: .ASCIZ /| RESTART ANALYSIS?
MM: .ASCIZ /| RESTART HARDCOPY?
MM: .ASCIZ /| WHICH SPREAD?
MM: .ASCIZ /| HOW MANY SPREADS?
SPOBF: .BLKB 03. HCOPY PARBF
FILPEX: .ASCII /| AT
LNAME: .ASCII /| XXX/ . EVEN
BLKB SOURCE+13900-. MAKE PROGRAM LENGTH CONSTANT
END CALMS

.TITLE SEARCH - SEARCH PARTITION OF CALMS
MCALL CALL. PAUSE
LF=12
CR=15
FTIME=15. MAX TIME BETWEEN FOCUS DURING SEARCH
PSU=177776
SY=177577
MCHRO GETKG MSG,LOC
TSTG PBUSY
BEG LOC
MOV @LOC,PINT+2
RTI
LOC CAL 0BPSU CALL TYPE,<MSG, ZERO>
ENDM

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

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

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

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

MCHRO HINT KEY HANDLE INTERRUPT FOR KEYS 3-8 (AC,RJ,NX,LA)
DSI 109
LOFF 109000
MOV RA, (SP)
CALL LED, (LKLED, KEY)
ENDM

CURRENT ORIGIN IS 40930
GHSTS: JMP GNS ENTRY POINT FOR GHSTS
SEARCH MOV @SRSTA, RA INITIAL CALL FROM SUPERVISOR
MOV @SRST, (RO)+ SET UP SEARCH START INTERRUPT VECT
MOV @298, (RO)+ PRIORITY 4
MOV @IS, RI
I520: MOV @UINT, (RD)+ ; UNEXPECTED INTERRUPT RTE TO OTHERS
MOV @200, (RD)+ ; PRIORITY 4
SBB R1, 1520
MOV @FOC, @FOCA
MOV @KYRG, @KYRGA
MOV @200, @KYRGA+2
CALL NSFST, <MAGN, FP03>

EN1 1410 ; ENABLE SRCH START, FOCUS AND KB3 REQUEST
DSI 357 ; DISABLE OTHERS

RTX R5 ; RETURN TO SUPERVISOR

UINT: PAUSE IEAP3 ; UNEXPECTED INTERRUPT
RTI

SRST, DSI 10
CALL LED, <KLLED, SSKEY>
GTRKB M1 ; GET KEYBOARD AND TYPE M1
CALL PARAM, <NP, PAR, MAXNP, SPBUF>
CMP NP, 42
BNE SR20 ; BR IF NOT 2 WORDS
CMP PAR, 99
BNE SR20 ; BR IF NOT ABORT
CALL TYPE, M7 ; SEARCH ABORTED, TYPE M7

EN1 10

SR20: INC B SFLAG ; SET SEARCH OPERATING
MOV L, @20599 ; RETURN OFF EDIT LIGHTS
LON 1000 ; RETURN SEARCH LIGHT ON
TST NP
BEQ SR30 ; BR IF SAME PATIENT AND SLIDE
CALL MVCN, <SPBUF, PANUM, LPNUM> ; MOVE PATIENT NUMBER
CALL TYPE, <M2, ZERO>
MOV @', PAR ; INITIALIZE SEX TO BLANK
CALL PARAM, <NP, PAR, MAXNP, SPBUF>
MOV PAR, PSEX ; STORE PATIENT SEX

CALL TYPE, <M3, ZERO>
CALL PARAM, <NP, PAR, MAXNP, SPBUF>
CALL MVCN, <SPBUF, SLNUM, LSLNUM> ; MOVE SLIDE NUMBER
CALL TYPE, <M0, ZERO>
CALL PARAM, <NP, PAR, MAXNP, SPBUF>
MOV PAR, SOURCE ; STORE SOURCE CODE

SR30: MOV @SPB, CSP0 ; INITIALIZE SEARCH QUEUE POINTER
CLR MSP
MOV R0, -<SP> ; SAVE R0 BEFORE CALLING ZIA
CALL ZIA, <SPR, LSPR> ; ZERO THE SPREAD RATINGS
MOV (SP)+, R0

CALL NSFST, <MAGN, FP03>
MOV XLCN, XCTR ; INITIALIZE SEARCH PATTERN
MOV IXDEL, XDEL
MOV @SBR, @SDFPA ; SEARCH DATA READY INTERRUPT
MOV @SRHA, @SARAA
MOV @SRB, @SARAA
MOV @BEDST, @BEDSTA

ENI 74 ; ENABLE INTERRUPTS
RTI

FOC: CALL LED, <KLLED, FOKEY> ; FOCUS INTERRUPT FROM FOCUS KEY
FOCUSUB: DSI 490 ; FOCUS SUBROUTINE

LOH 49000 ; IT MAY BE ENTERED BY ANOTHER INTERRUPT
MOV @FIRE, FCNT ; RESET FOCUS COUNT
MOV @IEAP, @IEAPD ; SAVE OLD IEAP AND SDFPA
MOV @SDFPA, @SDFPA
CALL NSFST, <MAGN, FP03> ; FIND FSTEP & FAST FOR THIS MAGNIFICATION
MOV @F413, @SDFPA ; SET UP INTERRUPT VECTOR

ENI 4
RTI

F410: CALL CFOC, OLD ; CALCULATE FOCUS
CALL IFREL, FSTEP

F420: MOV @F439, @SDFPA
RTI

F430: CALL CFOC, F
CMP F, 0.03F ; SEE IF WE ARE GOING IN THE RIGHT DIR
BGT F450 ; BR IF SO
MOV FSTEP, N ; CHANGE DIRECTION
ASL N
NEG N ; MOVE TWICE AS FAR IN REVERSE
.CALL IFREL.H
MOV $F440, 005DFPA
RTI

F440: .CALL CFHC.F
CMP $F.0LD.
BCT $F440, BR IF SO
CMP FSTEP, FLAST
BLE $F408, BR IF SO
MOV FSTEP, N
ASR FSTEP
ADD FSTEP, N
$N=FSTEP+FSTEP/2
.CALL IFREL.H
BR $F428

F440: .CALL CFHC.F
MOV $F.0LD.
.CALL IFREL, FSTEP
TO TAKE ANOTHER STEP
MOV $0F460, 005DFPA
RTI

F450: .CALL CFHC.F
CMP $F.0LD.
BCT $F450, BR IF SO
MOV FSTEP, N
BPL $F470, GET ABSOLUTE VALUE FOR TEST
NEG N

F470: NEG FSTEP
CMP H, FLAST
BLE $F490, BR IF SO
ASR FSTEP
DECREASE STEP SIZE
BR $F450

F480: .CALL IFREL, FSTEP
TO TAKE LAST STEP BACK
MOV $0F490, 005DFPA
RTI

F490: DSI 4
.CALL RNC1
MOV OSDFPA, 005DFPA
RESTORE SDFFPA
BIT $4.01EAPR
SEE IF INTERRUPT WAS ENABLED
BEQ $F590, BR IF NOT
BIT $01000, 00111
BEQ $F490, RE-ENABLE IT IF NOT IN SEARCH
TST $00SPRD
BPL $F490, OR SPREAD NOT PRESENT
BIT $48000, 01SYR
BEQ $F490, OR SW II DOWN
LON 29890
TURN ON SEARCH HALTED LIGHT
BR $F330, DON'T RE-ENABLE THE INTERRUPT
F499: ENI 4
F590: ENI 4
OFF 48830
TURN OFF FOCUSING LITE
RTI

KYR9: DSI 1839
DISABLE ANY FURTHER K80 REQ
.CALL LED. (LKLED, KRXKEY)
GETKB M4
.CALL PARAM. (NP, PAR, MAXNP, SPDUF)
TST NP
BEQ BKYEK, BR IF NO INPUT
CMPB PAR.8, "A"
BNE KY10, BR IF NOT REQUEST
.CALL APHASE. <ZERO>
BR BKYEK

KY10: CMPB PAR.9, "C"
BNE KY20, BR IF NOT CALIBRATE MICROSCOPE
.CALL IXREL. BIG
.CALL IYREL. BIG
MOVE TO LIMITS
.CALL IFREL. BIG
.CALL KWAIT
.CALL YWAIT
.CALL WAIT FOR END OF MOVES
.CALL IFCPGR, INITIALIZE F.X, AND Y
.CALL INCPGR
.CALL YCPGR
.CALL IXYABS. (XVAL, YVAL), MOVE STAGE TO 1000, 1000 OR LAST POSITION
63
CALL IFREL, FVLJ
CALL KWAIT
CALL YWAIT
CALL FMULT
CALL RMCI

KYEX: BR 6KYEX
KY20: CMPB PAR.0'R
BNE KY30: IF NOT RESTART REQUEST
INC CPRFLAG IFSET RESTART FLAG IN CALMS
KY30: CMPB PAR.0'E
BNE KY35: IF NOT EDIT REQUEST
INC CPRFLAG IFSET EDIT FLAG
BR KYEX
KY35: CMPB PAR.0'F
BNE KY40: IF NOT FORTRN REQUEST
INC PFLAG IFSET FORTRN FLAG
BR KYEX
KY40: CMPB PAR.0'Z
BNE KY50: IF NOT ZERO SCAN QUEUE REQUEST
CLR NSCAN
CALL LED. (SQLED.NSCAN)
MOV #SRSTL. #SRSTA SET ALLOW SEARCH START
BR KYEX
KY50: CMPB PAR.0'Q
BNE KY60: IF NOT QUEUE ADJUST
MOV @NCPRT.XOFF
SUB XVAL.XOFF STORE X OFFSET
MOV @NCPRT.YOFF
SUB YVAL.YOFF STORE Y OFFSET
BR KYEX
KY60: CMPB PAR.0'S
BNE KY70: IF NOT SCAN REQUEST
KY65: INC NSCAN
CALL LED. (SQLED.NSCAN)
MOV #SPRCH,#SRSTA SET UP SPIRAL SEARCH
IF X A4 AND Y WERE GIVEN
BNE KYEX IF NOT
CALL IXYABS. (PAR4,PAR8) 1999
ENI 1999
JMP FOCUSUB
KY70: CMPB PAR.0'B
BNE KY90: IF NOT DEBUG REQUEST
MOV PAR4, PHASE STORE PHASE NUMBER
BR KY65 IF INCREMENT NSCAN
KY80: CMPB PAR.0'B
BNE KY90: IF NOT BANDED REQUEST
MOV #-2, SOURCSET SOURCE OF -2
BR KYEX
KY90: CMPB PAR.0'P
BNE KY160: IF NOT PATIENT REPORT REQUEST
INC CPRFLAG IFSET PREP FLAG
CALL UPARSH. (ONE, PAR.PHASE) WRITE PREP PARAM
BR KYEX
KY160: ENI 1018
RTI

SDR: TST #SPRED SPREAD DATA READY (SEARCH)
BNI SPRES IF SPREAD PRESENT
MOV #1, FFLAG RESET FOCUS FLAG
DEC FCNT DECREMENT FOCUS COUNT
BPL GHP IF NOT NEED TO FOCUS
BIT #69080, #SPRD IF NOTHING TO FOCUS ON
JNP FOCUSUB GO FOCUS
JMP FOCUSUB IF NO FOCUS THIS TIME
NOFOC: MOV R8, -SP IF NO FOCUS THIS TIME
MOV CSP0, R8 CURRENT SPREAD QUEUE POINTER
MOV @NCPRT. (R8) + STORE X
STP XOK IF X IS POSITIVE
XYNG: GETKB M6 REQUEST CALIBRATION
MOV (SP) +, R8
LOFF  2008
  MOV  (RD)+, X
  BEQ  JEDHI:  CBR IF END OF QUEUE
  BPL  EL3B:  CBR IF NOT ALREADY ACCEPTED
  LON  2009  CINDICATE ALREADY ACCEPTED
  . .
  EL3B:  MOV  R8, CSPQ  CSAVE UPDATED Q POINTER
          MOV  (SP)+, R8  CRESTORE R8
  . .
  CALL  ITABS, (X,Y)  CINITIATE MOTOR MOVE TO X,Y
  MOV  #ESDR, #SDFPA  CMOVB
  MOV  #1, FF4L  CENABLE SPRD DATA READY INTERRUPT
  ENI  4
  .RTI
  . .
  JEDHI:  JMP  EDHI  CALLOW RJ DURING FOCUS
  . .
  ESDR:  ENI  108  CEDIT SPREAD DATA READY
          NEG  FFLAG  CBR IF NO FOCUS THIS TIME
          BPL  ES2B  CBR IF NO FOCUS THIS TIME
  JMP  FOCSUB  CNOW DISABLE THE SPRD INTERRUPT
  . .
  ES2B:  BSF  4  CMARK IT REJECTED X+
          CALL  RMC1  CRELASE MCI FROM COMPUTER CONTROL
          LON  100000  CTURN ON OPERATOR ACTION LIGHT
          .RTI
  . .
  ELAC:  HINT  ACKEY  CATCEPT SPRD INTERRUPT
          MOV  CSPQ, R0
          TST  -4(R0)  CTBR IF NOT PREVIOUSLY ACCEPTED
          BPL  ELA29  CDON'T COUNT IT TWICE
          . .
  ELA29:  MOV  @YCRP, -(R0)  CSTORE UPDATED Y AND X
          MOV  @XCRP, -(R0)  CMAKE X NEGATIVE TO FLAG THIS SPREAD
          NEG  R0  CBR IF NOT PREVIOUSLY ACCEPTED
          INC  HSPAL  CMARK IT REJECTED X+
          CALL  LED, <(SQed, Hspal)
          CMP  HSPAL, HM4X  CBR IF ENOUGH FOUND
          BR  EDHI  CBR IF ENOUGH FOUND
  . .
  ELRJ:  HINT  RKEY  CACCEPT SPRD INTERRUPT
          MOV  CSPQ, R0
          TST  -4(R0)  CBR IF NOT PREVIOUSLY ACCEPTED
          BPL  JEDNXT  CBR IF NOT PREVIOUSLY ACCEPTED
          . .
  JEDNXT:  MOV  #4, CSPQ  CMARK IT REJECTED X+
          BEQ  JEDNXT  CBR IF NOT PREVIOUSLY ACCEPTED
          DEC  HSPAL  CRMARK IT REJECTED X+
          . .
  JEDNXT:  JMP  EDHXT  CMARK IT REJECTED X+
  . .
  EDAH:  HINT  LAKEY  PAXTH SPREAD
          SUB  #4, CSPQ  CMOVB
          BGE  EDHXT  CMARK IT REJECTED X+
          BR  JEDNXT  CMARK IT REJECTED X+
  . .
  EDHI:  MOV  (SP)+, R0  CRESTORE R0
  GETKB  MS  CREQUEST HI MAGNIFICATION
  ENHST.  LOFF  1400  CREQUEST HI MAGNIFICATION
          LON  200  CREQUEST HI MAGNIFICATION
          MOV  @EHAC, @ACSPA  CREQUEST HI MAGNIFICATION
          MOV  @EHREJ, @RJSPA  CREQUEST HI MAGNIFICATION
          MOV  @EHNX, @NXSPA  CMARK IT REJECTED X+
          MOV  @ECHL, @ChASPA  CMARK IT REJECTED X+
          MOV  @ECHD, @EDSPA  CMARK IT REJECTED X+
          ENI  209  CREQUEST HI MAGNIFICATION
          CLR  NSCAN  CREQUEST HI MAGNIFICATION
          CALL  LED, <(SQed, NSCAN)
  . .
  EHRST.  CLR  SPH  CSAVE R0
          MOV  R8, (SP)  CRESTORE R8
  ENHXT:  MOV  CSPQ, R0
LOFF 4000 "TURN OFF FOCUS LITE IN CASE FOCUS
UNINTERRUPTED BY AMNL"

EH10 INC SPH
MOV (RB)+, X
BEQ JEDEND "IF END OF QUEUE"
BNI EH20 "IF - X (ACCEPTED IN EDIT LO)
TST (RB)+
BR EH10 "TRY THE NEXT SPREAD"

EH20 INC SPH
MOV (RB)+, Y
BPL EH30 "ABR IF NOT ALREADY ACCEPTED"
LON 2000 "TURN ON LITE"
BR EH40 "DO ADJUSTMENT IF ALREADY ACCEPTED"

EH30 CALL MFS1<MAXN, FPOS>
"SETUP XABJ AND YABJ"
SUB XABJ, X
"X IS NEGATIVE, ADJUST FOR CENTRALITY"
ADD YABJ, Y
"ADJUST POSITIVE Y FOR CENTRALITY"

EH40 MOV RO, CSPQ
MOV (SP)+, RO
"RESTORE RO"
CALL LEXX,<SWLED, SPH>
"INITIATE MOVE TO X, Y"
MOV #ESE0, #SDFP
MOV #1, FFLAG
ENI 4
JMP JEDEND "ENABLE SPRD DATA READY INTERRUPT"

JEDEND JMP JEDEND

EHAC: HINT ACKEY "IER HI ACCEPT"
MOV CSPQ, RO
TST -(RB)
BPL EH28 "ABR IF NOT ALREADY ACCEPTED AT HI MAG"
DEC NSCAN "DON'T COUNT IT TWICE"

EH28 MOV #YCPR, -(RB)
NEG RO
STORE -X AND -Y
MOV #XCPQ, -(RB)
NEG RO

INC NSCAN
CALL LED,<SOLED, NSCAN>
SUB #SPQ, RO
"FIND LOC IN SPR"
ASH #2, R7
INC SPQ,SP (RB)
"INCREMENT SPREAD RATING"
MOV (SP)+, RO
LON 10000
ENI 100
RTI "RETURN FOR ANOTHER AC OR HX"

EHJ: HINT RKJQ "IER HI REJECT"
MOV CSPQ, RO
TST -(RB)
BPL EH3X "MAKE IT POSITIVE IF PREVIOUSLY ACCEPTED"
DEC NSCAN
TST -(RB)
"GET LOC IN SPR FOR THE SPREAD"
SUB #SPQ, RO
ASH #2, R7
CLRO SPR,SP (RB)
"ZERO THE SPREAD RATING"
CALL LED,<SOLED, NSCAN>
BR JEHNXT

EHX: HINT MXKEY "IER MX FOUND"
CMP NSCAN, MAXSC
BGE EDEND "IF ENOUGH FOUND"
BR JEHNXT

ELA: HINT LAKJ "IER HI LAST"
SUB #4, CSPQ
DEC SPH
BEQ JEHNXT "IF FIRST SPRD IN Q"

EHL20: SUB #4, CSPQ
"INCREASE ANOTHER SPREAD"
DEC SPH
BEQ JEHNXT
TST RO, CSPQ
BPL EHJ20 "IF NOT - X"
JMP EHJ2X "LAST - X"
ENI 4

EDEN: CALL LED,<LKLEH, EEKEY>
DEEND:  LOFF  102200 ;TURN OFF ED HI AND ACC LIGHT
          RS  186 ;DISABLE AEML AND SDR
          TST  NSCN ;DISABLE EDIT START
          BEQ  EE30  ;BR IF NOF TO SCAN
          DSI  40  ;DISABLE EDIT START
          JSP  $5, SORTQ ;SORT THE QUEUE ACCORDING TO RATING
          MOV  *SPQ, C$PQ
          CLR  SPN
          CALL  WPARAM, <(512, SOURCE, ONE) ;SAVE INFO FOR RESTART
          CLR  XOFF
          CLR  YOFF
          CALL  CHSTS  ;GET FIRST SPREAD TO SCAN
          BR  EE40
          EE30:  MOV  B$RST, R$S$RST  ;ALLOW SEARCH START
          EE40:  MOV  (SP)$RH, R$  ;RESTORE R$
          CLR  SFLAG
          RTI

; CNS:  MOV  R8, -(SP)  ;GET NEXT SPREAD TO SCAN SB$TE
          MOV  C$PQ, R8
          MOV  #SPS$RCH, #RS$RST  ;ENABLE SPINAL SEARCH
          EEN:  INC  SPN  ;LOOK FOR FIRST Y
          EN:  MOV  (R8)$RVX$AL
          BEQ  GNEX  ;BR IF END OF QUEUE
          MOV  (R8)$RVY$AL
          BPL  EE10  ;BR IF NOT ACCEPTED
          MEC  $VX$AL  ;MAKE VX$AL AND Y$AL POSITIVE
          NSI  $VX$AL
          ADD  XOFF $VY$AL  ;ADD OFFSET
          ADD  YOFF $VY$AL
          MOV  R8, C$PQ
          CALL  XVABS, (XY$AL, YV$AL)  ;INITIATE MOTOR MOVE
          MOV  *PS$U, -(SP)
          JSP  PC, FOC$SUB  ;START FOCUSING
          GNEX:  CALL  LED, <(SNL$ED, SPN)
          MOV  (SP)$RH, R8
          RTI

; SPS$RCH:  CALL  LED, <(KL$ED, DSKEY) ;SPLRAL SEARCH
          CLR  XCT$R  ;INITIALIZE SPLRAL SEARCH
          CLR  YCT$R
          CLR  SPLE$N
          MOV  #SPS$DR, #DS$DFPA
          MOV  #S$RH$A, #S$RA$A
          MOV  #S$R$5, #S$RR$A
          EEN:  34  ;ENABLE SEARCH HALT & RESUME & 3DR
          RTI

; SP$RD:  BIT  #6,9,000, #SPRD ;SEE IF ANYTHING THERE
          BEQ  SGNP  ;BR IF NOT
          MEC  FFL$AC
          BPL  SGNP  ;BR IF NO NEED TO FOCUS
          SGNP:  C$T$U$  ;XCT$R
          BGT  SXOK  ;BR IF ANOTHER X MOVE IS OK
          TST  YCT$R
          BGT  SY$OK  ;BR IF ANOTHER Y MOVE IS OK
          INC  SC$LEN  ;INCREMENT SPINAL LENGTH
          MOV  SPLE$N, XCT$R  ;RELOAD X AND Y COUNTERS
          MOV  SPLE$N, YCT$R
          NEG  SP$STEP  ;CHANGE DIRECTION
          SXOK:  CALL  IX$REL, SP$STEP
          DEC  XCT$R
          RTI
          SYOK:  CALL  IY$REL, SP$STEP
          DEC  YCT$R
          RTI
          SP$LEN:  .WORD  1$84  ;SPLRAL LENGTH
          SP$STEP:  .WORD  5  ;DR  5
          YCT$R:  .WORD  1$84
          BX$G:  .WORD  9999
          XOFF:  .WORD  186
          YOFF:  .WORD  186
TITLE DLINE - DISPLAY A LINE ON THE INTERACTIVE DISPLAY

CALMS VERSION

BRUC=172410   ;WORD COUNT REGISTER
BRAC=172412   ;BUS ADDRESS REGISTER
BRAH=1724c+4  ;STATUS AND COMMAND REGISTER
YREC=164196   ;LINE REGISTER
ZREC=164104   ;SAMPLE REGISTER
CSSST=164180  ;GRAY SCALE STATUS/COMMAND REGISTER
GSPC=164118   ;GRAY SCALE PIXEL COUNT REGISTER
GSCP=164112   ;GRAY SCALE X CURSOR POSITION
GSCY=164114   ;GRAY SCALE Y CURSOR POSITION
WRCMD=4       ;WRITE COMMAND
ERCMD=6       ;ERASE COMMAND
ERCMD=2       ;ERASE/WRITE COMMAND
RBIT=10       ;REPLICATE BIT
WUP=177570    ;
LF=12
MCALL .BINNO, CALL, PAUSE

CALL DWAIT

DWAIT TST .GCSSST, .TEST READY.BIT

4,122,518

1:FLRG OSCILLATES FROM 1 TO -1
2:key.
3:UORK.
4:ACKE.
5:UORD.
6:UORD.
7:UORD.
8:UORD.
9:UORD.
10:UORD.
11:UORD.
12:UORD.
13:UORD.
14:UORD.
15:UORD.
16:UORD.
17:UORD.
18:UORD.
19:UORD.
20:UORD.
21:UORD.
22:UORD.
23:UORD.
24:UORD.
25:UORD.
26:UORD.
27:UORD.
28:UORD.
29:UORD.
30:UORD.
31:UORD.
32:UORD.
33:UORD.
34:UORD.
35:UORD.
36:UORD.
37:UORD.
38:UORD.
39:UORD.
40:UORD.
41:UORD.
42:UORD.
43:UORD.
44:UORD.
45:UORD.
46:UORD.
47:UORD.
48:UORD.
49:UORD.
50:UORD.
51:UORD.
52:UORD.
53:UORD.
54:UORD.
55:UORD.
56:UORD.
57:UORD.
58:UORD.
59:UORD.
60:UORD.
61:UORD.
62:UORD.
63:UORD.
64:UORD.
65:UORD.
66:UORD.
67:UORD.
68:UORD.
69:UORD.
70:UORD.
71:UORD.
72:UORD.
73:UORD.
74:UORD.
CALL DCLEAR

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

CALL BLINE(LOC,Y,X,NS,REP,ERASE) - 1 = ERASE ONLY

BLINE: JSR R4,SAVER
        JSR R5,DWAIT
        MOV #4(R5),@IVREG ;STARTING LINE ON DISPLAY
        MOV #URCMD,R1 ;ASSUME ERASE/WRITE
        BIS #409,R#RST ;ASSUME WRITE, SET CYCLE BIT
        TST #14(R5)
        BEQ NOER ;BR IF NO ERASE
        MOV #UCMD,R1 ;ASSUME ERASE/ WRITE
        TST #14(R5)
        BPL NOER ;BR IF ERASE/ WRITE
        MOV #ERCMD,R1 ;ERASE ONLY
        BIC #400,R#RST ;CLEAR CYCLE BIT
        NOER: MOV R1B,(R5),R3 ;NS
        MOV R12(R5),R2 ;REPL
        BEQ NOREP ;BR IF NO REPLICATION
        BIS #RST,R1 ;INSERT REPLICATE BIT
        ASL R3 ;2*NS
        NOREP: NEG R3
        MOV R3,@GSFC ;STORE PIXEL COUNT -N OR -2*N
        REP: MOV 2(R5),@DBBA ;LOC TO BUS ADDRESS REGISTER
        MOV #G(R5),@XREG ;STARTING SAMPLE ON DISPLAY
        MOV #R1B(R5),R2 ;SAMPLE COUNT
        INC R3
        ASR R3
        NEG R3
        MOV R3,@DRWC ;STORE NEGATIVE WORD COUNT
        MOV R1,@GSSST ;STORE WRITE OR ERASE/ WRITE COMMAND
        INC #RST ;ISSUE GO
        TST R2
        BLS EXIT ;BR IF DONE
        BIT #10,@SWR ;REPLICATE UNLESS SW 3 IS UP
        BNE EXIT ;EXIT IF SW 3 UP
        CLR R2
        JSR R5,DWAIT ;WAIT
        INC @IVREG ;STEP TO NEXT LINE
        BR REP ;GO DO THE REPLICATION
EXIT: JMP RESTR

; CALL SC TO SET UP THE CURSOR

SC: JSR R4,SAVER
SC10: MOV #3,R3
SC20: ADD #2,Y
        CALL BLINE, (ZERO,Y,X,ONE,ONE,ONE,ZERO)
        MOV #9,R4
SC25: ADD #2,Y
        CALL BLINE, (ZERO,Y,X,ONE,ONE,ONE,ZERO)
        MOV #4,SC20
        MOV R3,SC10 ;REPEAT TO MAKE IT WHITE ENOUGH
        MOV #5,R3
SC30: MOV #973,Y
        MOV #992,X ;WRITE A MARK AT 992,992
        MOV #9,R4
SC40: CALL BLINE, (ZERO,Y,X,ONE,ONE,ONE,ZERO)
        ADD #2,Y
        MOV #R4,SC40
        MOV #973,X
        CALL BLINE, (ZERO,Y,X,ONE,ONE,ONE,ZERO)
        MOV #3,SC30
SC50: CALL TYPE, (M3,ZERO) ;TELL OPERATOR TO MOVE CURSOR
        JMP PARAM, (MP,PAR,FIVE)
        JSR R5,ULDR ;CHECK FOR REQUESTED CURSOR MOVE
        BR SC50 ;CURSOR WAS MOVED
        MOV #65CYCP,Y2 ;STOP Y ZERO ADJUSTMENT
4,122,518

MOV @GSSCPXZ ;STORE X ZERO ADJUSTMENT
CALL TYPE: (R4,ZERO) ;MOVE TO LOWER RIGHT
SC60 JSR R5,ULDR;
BR SC60 ;CURSOR WAS MOVED
MOV @GSSCPYD ;STORE Y DELTA
SUB YZ,YB
MOV @GSSCPXD ;STORE Y DELTA
SUB XZ,XD
BIC @176900,YZ
BIC @176900,X2
CALL UPARAM: (FOUR,YZ,INTPH) ;SAVE THE ADJUSTMENTS ON DISK
JMP RESTR

CALL CURSOR (YCP,XCP,LINE,SAMP)
CURSOR JSR R4,SAVER
MOV @GSSCP,R0;
BIC @176300,R0
SUB YZ,R0
MUL @960,,R0 ;ADJUST THE READING
DIV YD,RO
ADD @32,,R0 ;AY=(Y-Z)*960/YD+32
ASL R1
CMP R1,YD
BLT C29
INC R0 ;ROUND UP
C29 MOV @GSSCP,R2
BIC @176300,R2
SUB XZ,R2 ;ADJUST THE X READING
MUL @960,,R2
DIV XD,R2
ADD @32,,R2
ASL R3
CMP R3,XD
BLT C40
INC R2 ;ROUND UP
C40 MOV R8,02(R5) ;STORE YCP
MOV R8,04(R5) ;STORE XCP
BIT @4,RS5
BEQ CEX ;JBR IF NOT 1 PARAMETER
ASR R8
INC R8
MOV R8,06(R5) ;LINE=YCP/2+1
ASR R2
INC R2
MOV R2,010(R5) ;SAMP=XCP/2+1
CEX JMP RESTR
:
MCU: SUB @2,@0GSSCP ;CALL MUC TO MOVE CURSOR UP
MCB: ADD @2,@0GSSCP ;CALL MCB TO MOVE CURSOR DOWN
MCL: SUB @2,@0GSSXP ;CALL MCL TO MOVE CURSOR LEFT
MCR: ADD @2,@0GSSCP ;CALL MCR TO MOVE CURSOR RIGHT
:
ULDR: MOV @2(RS),PAR ;CALL ULDR(PAR),EXTERNAL CALL
BR UL20
ULDR TST (R5) ;CHECK FOR U.L.D. OR A TYPED
BNE UL20 ;SET FOR NO TYPEIN RETURN
TST @R5+ ;JBR IF NO LETTER WAS TYPED
RTS R5
UL20: CMPB PAR,0'U
BNE UL38 ;JBR IF NOT A W
UL30: CMPB PAR,0'L
BNE UL40 ;MOVE CURSOR UP ONE STEP
INC @9GSSCP
UL40: CMPB PAR,0'L
BNE UL50 ;MOVE CURSOR DOWN ONE STEP
UL50
79
4,122,518

TITLE MCI SUB - MCI SUBROUTINES
CALNS VERSION

XIPR==164040
YCPR==164042
FCPR==164044
MSC==164046
IEHPD==164059
SPPD==164074
FOCP==164076
LITES==164134
LEDs==164136
SRSTK==328
SRHMT==384
SPPSta==319
SPPSt==320
EDSta==334
ACSPSt==336
REJSt==338
LASTSt==334
EDESta==348
FOCA==344
KYPSt==350
SPDSt==374
SFSt==386
SRSSt==387
TXBT==390
RXBT==392
XBT==394
YBT==396
FBT==398
CBT==400
FOCS==401
KYPQ==402
SUP==403
KBD REQUEST II
SPRD/FOCUS DATA READY INTERRUPT ADDR
CURRENT ORIGIN IS 50500

MCALL CALL, PAUSE

XIPR==164040
YCPR==164042
FOCP==164076
LITES==164134
LEDs==164136

FOCSt==344
FOCUS 10
KYPSt==350
KBD REQUEST II
SPRDS/FOCUS DATA READY INTERRUPT ADDR

CNCI: BIP 40000, 0/0MSC CAPTURE MCI
RTS R5
RCNCI: BIC 40000, 0/0MSC RELEASE MCI
RTS R5
XWAIT: BIT 40000, 0/0MSC
BNE XWAIT
RTS R5
YWAIT: BIT 4200, 0/0MSC
BNE YWAIT
RTS R5
FWAIT: BIT 10, 0/0MSC
BNE FWAIT
RTS R5

IXYABS CALL XWAIT CALL IXYABS(X, Y) TO INITIATE ABS MOVE
CALL YWAIT
CALL CNCI
MOV 20.39, X
PPL IXYTO
```
81  4,122,518

IDX20.  MOV  04(R5), Y   †MAKE X POSITIVE
       DPL IDX30
       MOV  Y, #0
       NEG   X
       NEG   Y   †MAKE Y POSITIVE

IDX30.  SUB  X, #0@CPR  †INITIATE MOVE TO X, Y
       SUB  Y, #0@CPR
       RTS  R5

IXREI.  CALL  XWAIT  †CALL IXREI(XDEL) TO INITIATE X MOVE
       CALL  CMI
       MOV  #02(R5), #00@CPR
       RTS  R5

IXREI.  CALL  YWAIT  †CALL CMI
       MOV  #02(R5), #00@CPR
       RTS  R5

IFREL.  CALL  FWAIT  †CALL IFREL(FDEL) TO INITIATE FOCUS MOVE
       CALL  CMI
       MOV  #02(R5), #00@CPR
       RTS  R5

CFOC.  MOV  @0F0C, TEMP  †CALL CFOC(F) TO CALCULATE FOCUS
       CLR  #02(R5)
       MOVB  TEMP+1, #02(R5)
       ADD  TEMP, #02(R5)
       RTS  R5

IXCPR.  BIS  #04@R5, #0@SHSC  †INITIALIZE XCPR
       RTS  R5

IXCPR.  BIS  #02@R5, #0@SHSC
       RTS  R5

IFCPR.  BIS  #01@R5, #0@SHSC
       RTS  R5

LED.   JSR  R4, SAV1R  †CALL LED(CODE, VAL)
       MOV  #02(R5), R0  †CODE TO R0
       ASH  #12, R0  †SHIFT TO HI ORDER 3 BITS
       MOV  #04(R5), R1  †VALUE TO R1
       BIC  #1177888, R1
       ADD  R1, R0
       MOV  R0, #0@LED5  †OUTPUT DISPLAY
       JMP  REST

MFST.  JSR  R4, SAV1R  †CALL MFST(MAGH, FPOS)
       MOV  #1@EAPB, R0  †FLTR/MAGH POS REG TO R0
       MOV  R5, R1  †ALSO R1
       ASH  #10, R0
       IC  #1177777, R0  †MASK OUT MAG
       MOVB  LEHS(R0), MAGH  †STORE MAGH
       CLR  FSTEP  †CLEAR TOP BYTE
       MOVB  FST3(R0), FSTEP  †STORE FOCUS STEP SIZE
       MOVB  FTA8(R0), FLAST  †STORE LAST STEP SIZE
       MOVB  XATAB(R0), R2  †ADJUSTMENT FOR NON-CENTRALITY
       MOV  #2, R0XDJ
       MOVB  YATAB(R0), R2  †ADJUSTMENTS ARE FOR 63 AND 100
       MOV  R2, YADJ  †RELATIVE TO 40
       MOV  R5, #0@RSR
       ASH  #13, R1
       MOV  R1, R4(R5)
       MOV  R1, FPOS
       CALL  LED(CODE, MAGH)  †DISPLAY MAGNIFICATION
       JMP  REST

SORT.  JSR  R4, SAV1R  †SUBROUTINE TO SORT THE QUEUE
       MOV  #0SPR, R3  †R3 AND R4 POINT TO LOW QUEUE POSITIONS
       MOV  #0SPR, R4  †THE CURRENT HI RATING IS MOVED HERE
       MOV  HSP, RSR
       MOV  R3, R4  †INITIALIZE # SPREADS REMAINING
       MOV  R3, R1  †USE R1 TO EXAMINE REMAINING SPRDS
       MOV  RSR, R2
       BEQ  S30X  †BR IF NO SPREADS REMAIN

S30X.  CMPB  (R1)+, R0  †EXAMINE A RATING
       BGE  S30X  †BR IF NOT GREATER THAN MAX
       MOV  #00@CPR
       MOV  #01@CPR
       MOV  R1, REG1  †SAVE R1
```
; TITLE SCAN - SDS DATA CAMERA DRIVER
; CALMS VERSION CALL SCAN(DUNIT.FILPES)
; THIS VERSION USES WMI IN DOUBLE BUFFERING 24
; VERTICAL LINES AT A TIME FROM CAMERA TO DISK DATA SET
; IN NON-INTERRUPT MODE.
; THE PICTURE IS DISPLAYED ON THE GRAY SCALE.
; SECTOR HISTOGRAMS ARE GENERATED AS THE PIXELS ARE SCANNED.
; THRESHOLDS ARE FOUND AFTER EACH GROUP OF SECTORS AND TYPED
; OUT WHILE THE NEXT ROW OF SECTORS IS SCANNED IF SW1 IS UP.
; THE SCAN AREA IS VARIABLE WITH A MAXIMUM OF 512 LINES AND
; 488 SAMPLES PER LINE.
; GLOBAL OPEN, GET, PUT, CLOSE, PARAM, OPTRN, PLABEL, PARBUF
; GLOBAL AFIL, SCAN
; MCLL PARAM, EXIT, CVTBT, CALL, PAUSE
; PARAM
; HLIST, BEX
; LF=12
; CR=15
; CMP=164920
; STAT=164920
; XREG=164922
; YREG=164924
; ZREG=164926
; XJREG=164930
; YJREG=164932
; XJREG=164930
; /J JOYSTICK REGISTER
; YJREG=164932
; JY JOYSTICK REGISTER
; SCAN
; MOV R5,-(SP) ;SAVE R5
; MOV 2(R5),$10+10 ;CALL SCAN(DUNIT,FILPES) STORE DUNIT
; MOV $42(R5),$10+12 ;STORE CALMS DATA SET NAME
; MOV @2.UIC ;AND UIC OF [2, 2]
; S10A: CALL AFIL, Buf, - , - , - , UIC, UIC) ;ASSIGN DEFAULT FILE
; CALL OPEN, Buf, SIZE, OME, LNAME) ;OPEN DISK DATA SET
; S102: CALL TYPE, (NI, ZER0) ;RETURN
; TFCI, BIT #480, #81EAP ;TEST IF STILL FOCUSING
; BEG TFCI ;BR IF SO
; CALL RNCI ;RELEASE NCI TO OPERATOR
; CALL PARAM, <PAR, PAR, HMAX) ;READ LABEL
; MOV SOURCE.L+12 ;MOVE INFO INTO LABEL
; MOV PANUM, L+14
; MOV SLNUM, L+16
; MOV #3XEPR.L+18
; MOV #YCP.L+20
; CALL OUTCON, <SOURCE.L+51, TWO>
; CALL NLY, <PANUM.L+92, L PANUM>
; MOV NLY, L+183
; CALL NLY, <SLNUM.L+112, LSLNUM>
; TST #PFR
; BEG #100 ;BR IF NO PARAMETERS (USE OLD LABEL)
; CMP #PFR, #2
; BNE #104 ;BR IF NOT TWO PARAMETER WORDS
; CMP #*PO,FAR
; BNE NOTPO ;BR IF NOT PRINT Q REQUEST
; CALL PRINTQ, (L+72, SPO, SPQ)
NOTPQ: JMP S102
       CMP *A, PAR
       BNE NOTAB
       CALL APHASE, ZERO
       JMP S380
NOTAB: CMP *Z, PAR
       BNE S183
       CALL NSCAN
       JMP APHASE, ZERO
       AND ABORT THIS SCAN
$103: CMP *EX, PAR
       BNE S184
       CALL NSCAN
       IF EX WAS NOT TYPED
$104: CMP *DS, PAR
       BNE S185
       BR IF NOT DUMMY SCAN
       INC DFLAG
       SET DFLAG
$105: MOV *PAR3UF, R1
       MOV #L+217, R2
       ADD ANOTHER LABEL WITH_MISC_ID
$106: MOV #69R3
       MOV (R1)+, R8
       MOV R8, OCR
       BEQ ELAB
       BR IF END OF LABEL (CR)
SUB R3, HLAB
BR S106
ELAB: MOV #48, (R2)+
       PAD REST OF LABEL WITH BLANKS
SUB R3, ELAB
$107: JSR PC, PPAR
       PROCESS PARAMETERS
       CALL OUTCON, (XCPR.L+127..FOUR)
       CALL OUTCON, (YTZIP.L+135..FOUR)
       CALL MVL, (L+124..M+18..FOUR)
       CALL MVL, (L+132..M+22..FOUR)
       CALL TYPE, M2
       CVTDT #0L+145
       STORE DATE IN LABEL
       CVTDT #0L+155
       STORE TIME IN LABEL
       TST DFLAG
       BNE S108
       BR IF DUMMY SCAN
       CALL PLABEL. (BUF.SPAR.L)
       OUTPUT LABEL
$109: CALL DCLEAR
       CLEAR THE GRAY SCALE
$110: MOV SL, X
       INITIAL XCOUNT
       CLR #COND
       DISABLE PIXEL INTERRUPT
       CLR CY
       CLEAR GS LINE COUNTER
       MOV #TAB, TX
       MOV #TAB, BX
       JSR PC, THEAD
       TYPE HEADING
       CLR SHUM
       SECTOR NUMBER=0
       CALL ZIA, (HIST, MDIN)
       ZERO HISTOGRAM COUNTS
       EFLIN: MOV SS, Y
       INITIAL Y
       MOV #O, REG
       SET LINE COUNT
       INC LINE
       INCREMENT MV10 LINE COUNT
       TST DFLAG
       BNE S115
       BR IF DUMMY SCAN
       CALL PUT, (BUF.LINE.INDEX)
       PREPARE STORAGE OF NEXT TV LINE
$115: MOV BUF.R3
       LIMIT BUFFER POINTER TO BUF+HS-1
       0FLIN: ADD INDEX, R3
       SET POINTER FOR NEXT LINE
       MOV #HIST, R4
       R4 HAS LOC OF HIST.
       MOV IPS.R3
       INITIAL SAMPLES PER SECTOR
       MOV Y, #B1REG
       SET PIXEL COUNT
       MOV #CDMB.R2
       (COND-STATUS REGISTER) TO R2
       MOV #3R2
       GET FIRST PIXEL OF THIS FIELD
       TSTB OR2
       BPL -2
       WAIT UNTIL READY
       MOV #3080.00PSW
       SET PRIORITY 6 FOR THE REST OF THE FIELD
       BR MOV0B
       SECT0R: MOV SPS.R3
       R3=SAMPLES PER SECTOR
       PIXEL: MOV #3R2
       PIXEL COMMAND
       TSTB OR2
       SP -2
       WAIT UNTIL READY
       MOV #128EG, R1
       MOVE PIXEL INTO BUFFER
       BNE NOTZE
       BR IF NOT ZERO
The image shows a page of computer code written in assembly language, with various assembly instructions and comments. The page contains sections of code for different tasks, such as converting data and processing pixel values. The code includes labels, variables, and various instructions like MOV, CMP, INC, DEC, BNE, BGT, and others, which are typical in assembly language programming. The page also contains comments and placeholders for further processing steps, such as initialization and conditional checks.
## PROCESS AREA PARAMETERS

<table>
<thead>
<tr>
<th>JMP</th>
<th>F30</th>
<th>P</th>
<th>1GO</th>
<th>4,122,518</th>
</tr>
</thead>
</table>

**P159:** TST NR ; DONE WITH PARAMETERS
MOV HL, R0
ADD $32, R0
ASH $4-5, R0
BGT P155 ; IF HL GT 31

**P155:** MOV R0, NR ; NR = (NL+32)/64
TST NC
BGT P170 ; IF NC WAS A PARAMETER
MOV NS, R0
ADD $32, R0
ASH $6, R0
BGT P163 ; IF NS GT 31
INC R0 ; INC = 1
INC NS, R0
MOV R0, HC ; INC = (NS+32)/64

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

### THRESH

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

**T05:** MOV #HIST+96, +...24 ; HIST + (NC-1)*128

**T10:** JSR PC, TSUB ; FIND BACK AND DATA
MOV DPR, R1
MUL DATA, R1
MOV DPR, R3
MUL BACK, R3
ADD R3, R1
ADD #100, R1
DIV #100, R0 ; JTHR = (DPER+DATA+(100-DPER)*BACK+50)/100
MOV DATA, R1
SUB BACK, R1
CMP R1, #0 ; CHECK FOR SMALL DIFFERENCE BETWEEN D AND B
BGT T26 ; BR IF OK

**T20:** MOV BKSIZ, R0 ; JTHR = 3*BEQ/2 (UNDIVIDED NUCLEUS)
TST DATA
BGT T30 ; IF NO DATA IN SECTOR

**T30:** TST BACK
BGT T40 ; BR IF BACK OK
CMP #127, R0 ; JTHR = 127 IF NO DATA IN SECTOR
BEQ T40 ; SEE IF FIRST SECTOR
4,122,518

T40:
MOV BX, R1
JUSE BACK FROM PREVIOUS SECTOR
MOV -(R1), BACK
TST DATA
BCT T45
CLR ITHR
JITHR=0 IF NO DATA
T45:
MOV BACK, 0BX
JSTORE IN TTAB AND BTAB
ADD $2, TX
ADD $2, BX
JINCREMENT Pointers
T50:
.CALL OUTCOM,(ITHR, ..., FOUR)
ADD $7, T50+10
T60:
.CALL OUTCOM,(BACK, ..., THREE)
ADD $7, T60+10
SUB $120, R4
CMP R4, #HIST
BNHS T61
JBR IF MORE HISTOGRAMS TO ANALYZE
SUB $16, T60+10
CLR R6, T60+10
INC SHUM
.CALL OUTCOM,(SHUM, N0+2, THREE)
ADD NC, SHUM
BEC SHUM
MOVB $1, M0+3
.CALL OUTCOM,(SHUM, N0+3, TWO)
BIT $2, #8WR
BEQ T70
JBR IF SW 1 DOWN
T70:
RTS PC

THEAD:
MOV HC, R1
JSR PC, THEAD SET UP HC HEADINGS
MOV #MSG, R2
TH10:
MOV #HEAD, R3
MOV $7, R4
TH20:
MOVB (R3)+, (R2)+
SOB R4, TH20
MOVB R1, TH10
JSR PC, MSG
TH30:
MOVB (R3), (R2)+
BIT $2, #8WR
BEQ T30
JSR PC, SHOOT I SIG RE
TH70:
RTS PC

SMOOTH:
MOV R4, -(SP)
JSR PC, SMOOTH SAVE R4
MOV $HIS+2, R3
SH20:
MOV $62, R2
MOV (R4)+, R0
MOVB (R4)+, R0
ADD $4, R0
ADD (R4)+, R0
MOV R0, (R3)+
SUB R2, SH20
MOV (SP)+, R4
RTS PC

JSR PC, TSUB
TSUB:
MOV $HIS, R0
MOV $19, R1
MOV BK512Z, R2
JSR BACK RANGE IS NOMINALLY 2-48
MOV $HIS+2, R3
TS40:
CMP R1, (R3)+
BGE TS45
JSR PC, MSG
MOV -(R3), R1
JMAX=HIS(I)
MOV R3, R0
JHIS=I
TST (R3)+
SUB R2, TS40
JSR PC, MSG
MOV R0, BACK
JSTORE BACK
MOV $HIS, R0
DATA RANGE IS NOMINALLY FROM 50-124

; CHECK IF MAX GE HIS(I)
CMP -2(R3), -4(R3) ; LOOK FOR RELATIVE MAXIMUM ON LEFT SIDE
BLT TS60 ; JNS IF NOT

MOV R3.R0, R0+1 ; R0=HIS+1
TS60: SUB HIS, R0
MOV R0, DATA ; STORE DATA
RTS PC

MAXL=512.
MAXNS=499.
NK=6

KEY:

NK: .WORD /DPDBHKNQOLJ/
SL: .WORD 0. ; STARTING LINE
EL: .WORD 586. ; ENDING LINE
SS: .WORD 0. ; STARTING SAMPLE
DBEG: .WORD 53. ; DATA BEGIN
SPS: .WORD 38. ; SAMPLES PER SECTOR OF A FIELD
ISPS: .WORD 39. ; INITIAL SPS
BKX1Z: .WORD 24. ; DBEG/2-1
BKX2Z: .WORD 38. ; 162-BKSIZ
SIZE: .WORD 12288. ; SIZE OF EACH 24-LINE BUFFER
SCTR: .WORD 7. ; HIST DIMENSION (512 WORDS)
SPACE: .WORD 49.
EJECT: .WORD 61.
UIC: .WORD 2. ; DEFAULT UI: IS 2.2
SVTY: .WORD 70.
ZERO: .WORD 0.
ONE: .WORD 1. ; CONSTANT
TWO: .WORD 2.
THREE: .WORD 3.
FOUR: .WORD 4.
FIVE: .WORD 5.
X: .WORD -.
Y: .WORD -.
LINE: .WORD -.
QFLAG: .WORD 0. ; FLAG FOR QUICK LOOK (WHILE SCANNING)
BFLAG: .WORD 0. ; FLAG FOR BUTTON SCAN
BF: .WORD 56. ; BLOCK MONEY FROM PUT
HPAR: .WORD -.
JPAR: .BLKW 4. ; JOYSTICK PARAMETERS
PAR: .BLKW 80.
NPMAX: .WORD 80.
SPAR: .WORD 499. ; LINES
NS: .WORD 472. ; SAMPLES
NLR: .WORD 1.
HIST: .BLKW 512. ; 512 BLKSIZ HISTOGRAMS
HIS: .BLKW 64. ; SMOOTHED HISTOGRAM
TDATA: .WORD -.
HR: .WORD -.
NP: .WORD 1. ; ROWS OF SECTORS
NC: .WORD 1. ; COLUMNS OF SECTORS
LPS: .WORD 63. ; LINES PER SECTOR
SPS2: .WORD 68. ; SAMPLES PER SECTOR
TTab: .BLKW 64.
DTAB: .BLKW 64.
DPER: .WORD 68. ; DATA PERCENT
BPER: .WORD 40. ; BACK PERCENT
TX: .WORD TTab+.. ;
BX: .WORD 0Tab+.
BACK: .WORD -.
DATA: .WORD -.
ITHK: .WORD -.
SHUN: .WORD -.
QY: .WORD -.
C

****** BINARY PICTURE GENERATOR ******
SUBROUTINE BINARY(BUNIT, FILPESX)
BYTE FILPESX(?)

IMPLICIT INTEGER (A-Z)
COMMON/C1/NR, NC, LPS, SPS, TTAB(64), BTAB(64), BPER
COMMON/C1/SLO, SSD, HLO, HSO, HS, MAH09
COMMON/C1/NSM, MAREA, MANA, MNAN, MAE, ME, MXE, MAXLEN, NM, TOWN, ISW1
COMMON/C1/ WORD(4128), PIC(12344), PBUF(512)
INTEGER PAR(23), KEY(6), SPR(5), CTAB(307), MTAB(30)
BYTE PIC, PMSG(22), PBUF
DATA KEY/'BP', 'TD', 'AR', 'EP', 'LE', 'SK'/
CALL SSWITCH(4, ISW)

IF(ISW4.EQ.1) CALL TIMER
CALL SSWITCH(1, ISW1)
IF(ISW1.EQ.1) CALL TYPE(' BINARY')
CALL SPPARM(PAR, 20)
C
C WPARM WILL BE ZERO THE FIRST TIME BINARY IS RUN ON A SPREAD
IF(WPARM.EQ.0) CALL TYPE(' STARTING ANALYSIS ON?')
CALL AFILE(PIC, BUNIT, FILPESX, 2, 2)
CALL OPEN(PIC, 6144, 18, 'SCN')
CALL CLABEL(PIC, SPARM, IND)
101
NLO = SPAR(1)
NSO = NMPB(SPAR(2),311)
CALL GET(PIC,NLO+1,1)
CALL MVW(PIC(1+1),NR,1337)
C
CALL RPARAM(KP,PAR,20)
1 I = -2
2 I = I + 1
2 I = I + 2
IF(I.GT.HPAR) GOTO 20
DO 5 K=1,NKEY
IF(PAR(I).NE.KEY(K)) GOTO 3
GOTO(2,12,13,14,15,16,K)
5 CONTINUE
C
C PARAMETER ERROR
PAUSE 1
GOTO 20
12 TOSN = PAR(I+2)
GOTO 1
13 I = I + 2
MNAREA = PAR(I)
MNAREA = PAR(I+1)
GOTO 2
14 I = I + 2
MNEPT = PAR(I)
MNEPT = PAR(I+1)
GOTO 2
15 MAXLEN = PAR(I+2)

C
16 B = PAR(I+2)
GOTO 1
20 CONTINUE
IF(ISW4.EQ.1) CALL TIMER
B = (B+1)/2
SPS2 = (SPS+1)/2
MSAMPS = NLO - (CHR-1)*LPS
NW = NSO/32 + 1
HW = (NSO-2)/32
HWT = HWT+16
C 334 I
C ** Binary picture area (word) is assumed to be zero initially
C
C HW = number of words output per line
C HWT = number of words in per line
C
C S = 0
C 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.HW-1) GOTO 50
S = S + 16
IF(SLTSPS2) GOTO 50
S = S - SPS2
NTAB(W) = 16 - S
C = C + 1
50 CONTINUE
IF(ISW4.EQ.1) CALL TIMER
C
C SI = 1
WI = NW
REC = 1
ILPS = LPS
C
DO 300 R=1,HR
IF(R.EQ.NR) 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)
S = 1

IF(ISU4.EQ.1) CALL TIMER

IF(T0BN.EQ.0) GOTO 250
DO 200 W=1,H Ut
CALL STHR(PBUF(S).WORD(W+4).TTAB(CTAB(W)+1).PIC(P1+S).MTAB(W))

200 S=S+32

C IF(ISU4.EQ.1) CALL TIMER
GOTO 290

C CONTINUE
DO 289 W=1,H Ut
CALL STHR(PBUF(S).WORD(W+4).TTAB(CTAB(W)+1).PIC(P1+S).MTAB(W))

289 W=W+S+32

C IF(NPBR.EQ.0) CALL TYPE('0')
PAR(I)= 'L'
PAR(3)= 90
CALL WPARAM(3,PAR.2)

C WRITE PARAMETERS IN CASE OF RERUN
CALL CLOSE(PIC)
IF(ISU4.EQ.1) CALL TIMER
CALL SEGNT
IF(ISU4.EQ.1) CALL TIMER

C END

SUBROUTINE SEGNT
IMPLICIT INTEGER(A-Z)
COMMON/C1,SPAR(133),SL0,SSL,HLO,MSO,MOD,B,MAXMOD
COMMON/C1,MNAREA,MAREA,MPERM,MXPERM,MXXPLN,NOB,T0BN.ISWI
COMMON/C1,WORK(4126),NPERM,NPERR,NM,NP,NMAX,NMAX
CUMMON/C1,DIR(512),EDUF(512),EDGE(499)
COMMON/C1,NK,LIP(60),SIP(60),F1
INTEGER,PAR(5)
BYTE LABEL(S12),M(6),WHITE(14)
DATA PAR/12,1624,7,1,512/
DATA LABEL/79*8', 'S', 'C', '70*8', 'A', 'L', 368*8'/
DATA MAX/1022, DREC/2/ 
C THE EDGE POINT DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJECT
C DIR(BI) = FBW (FIRST BLOCK WRITTEN)
C DIR(BI+1) = N (NUMBER OF END POINTS)
C DIR(BI+2) = YMIN
C DIR(BI+3) = XMIN
C DIR(BI+4) = YMAX
C DIR(BI+5) = XMAX
C THE EDGE POINTS ARE GROUPED IN COORDINATE PAIRS REPRESENTING SEGMENTS
C EDGE(E)= X
C EDGE(E+1)= Y
CALL ITLA(0,WHITE,14)
IF(T0BN.LT.0) CALL RPARAM(NP,NK,122.6)
C READ OLD VALUE OF F1

MAXR = MNPERM/4
MAXL = MNAREA/4
MNEF = MNPERM/2
MNEX = MNPERM/2
MXL = MAXLEN/2
CALL SSWITCH(1,ISU1)
IF(ISU1.EQ.1) CALL TYPE('SEGNT')
CALL SSWITCH(4,ISU4)
IF(ISU4.EQ.1) CALL WPRINT('SEGNT')
CALL ZIA(EDUF,29)
CALL ZIA(DIR,510)
CALL ZIA(EDGE,400+)
CALL HVL('EDGE FILE',LABEL(73),19)
IF(T0BN.LT.0) GOTO 14A
CALL RFILE(EDUF,1,'PIC',5.5)
CALL OPENEDUF,1/24.3,1,'W2')
CALL PLAGH(EDUF,PAR,LABEL)
CALL FU(SGU,5,194)
105 CONTINUE
   .HE = NUMBER OF WORDS ON A BINARY LINE
   REC = 3
   NW = NW/2
   NW2 = NW2
   NL = NL0/2
   H = HSO/2

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

   SCAN BINARY PICTURE (WORD) LINE-BY-LINE

   DO 200 L=1, NL
   WI = EW1 + 1
   EW1 = EWI + NW
   LOOP = 0

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

   140 IF (ROACH(WI, BI, WORD, EWI, MASK).EQ.0) GOTO 200
   LOOP = LOOP + 1
   IF (LOOP .EQ. 200) PAUSE 1
   EDGE(1) = L
   EDGES(2) = (H-(EWI-WI+1)) + 16 + BI + 1
   CALL TURTLE(N, EDGE, WORD(WI), MASK, NW2)
   IF (N .EQ. 0) GOTO 148
   H2 = 2-H
   IF (NOB .NE. TSNH) GOTO 141
   CALL LIST(NOB, H, YMNIN, XMNIN, XMNAX)
   CALL PUMP(EDGE, EDGE(N2), 1)
   141 CONTINUE
   IF (N .GE. 255) GOTO 142
   CALL SORTIN(EDGE, H, SIND)
   IF (NOB .EQ. TSNH) CALL PUMP(EDGE, EDGE(N2), 1)
   IF (SIND .EQ. 0) GOTO 145
   142 IF (N .LE. MAXN) GOTO 143
   CALL TYPE(' TOO MANY EDGES')
   CALL TYPE(' ')
   GOTO 200

   143 CONTINUE
   IF (ISWI .EQ. 1) CALL TYPE(' SORT ')
   IF (ISW4 .EQ. 1) CALL STIMER
   CALL SORT(EDGE, W)
   IF (ISW4 .EQ. 0) CALL STIMER

   N = NUMBER OF EDGE POINTS

   145 CALL ERASE(WR, EDGE, AREA, N2, NW)
   IF (AREA .LE. XHR, OR. AREA .GT. XLMN) GOTO 196
   IF (YMAX-YMIN .GE. XMLEN .OR. XMEN-YMIN .GE. XMLEN) GOTO 196
   GOTO 143

   C ENTER OBJECT INTO DIRECTORY
   IF (NH .GE. 60) OR. TSNH .GE. 0) GOTO 173
   NH = NH+1
   LIP(NH) = EDGE(1)+EDGE(1)
   STP(NH) = EDGE(2)+EDGE(2)

   C STORE OBJECT INFORMATION FOR QUICK COUNT OPTION
   Y = Y*EDGE(1)
   X = X+EDGE(2)-14
   IF (X.GT. 0) X = 0

   C WRITE AN ARROW BY THE OBJECT
   CALL DLINE(WHIT, Y, X, 6.1, 0)
   CALL DLINE(WHIT, Y, X, 6.1, 0)
   CALL SSTCH(5, ISW6)
   IF (ISW6 .EQ. 1) GOTO 175
   CALL DLINE(WHIT, Y-4, X+8, 8, 2, 1, 0)
   CALL DLINE(WHIT, Y-4, X+8, 8, 2, 1, 0)
   CALL DLINE(WHIT, Y-2, X+10, 2, 1, 0)
   CALL DLINE(WHIT, Y-2, X+10, 2, 1, 0)
   CALL DLINE(WHIT, Y+2, X+10, 2, 1, 0)
   CALL DLINE(WHIT, Y+2, X+10, 2, 1, 0)
   CALL DLINE(WHIT, Y+4, X+8, 2, 1, 0)
   CALL DLINE(WHIT, Y+4, X+8, 2, 1, 0)
CONTINUE

DIR(DI) = REC

DIR(DI+1) = H
DIR(DI+2) = MAX0(YMIN-B.1)
DIR(DI+3) = MAX0(XMIN-B.1)
DIR(DI+4) = MINH(LMAX+B.0040)
DIR(DI+5) = MINH(MAX+B.0040)

NOF = NOF + 1
II = DI + 6
IF(NOF GT MAXNOB) GOTO 201

C

IF(TODN. LT. 0) GOTO 140
C DON'T WRITE IF QUICK COUNT OPTION
DO 195 1=1,N2,512
CALL WRITE(EBUF, REC, EDGE(1))

195 REC = REC + 1
GOTO 140
C

CALL SSITCH(4, ISW4)
IF(ISW4 NE.1) GOTO 140
CALL QPRINT(' OBJECT REJECTED')
CALL I LIST(NOB, N. AREA, YMIN, XMIN)
CALL LIST(KMAX-YMIN, XMAX-XMIN, MXAR, NKEPT, MXLEN)
GOTO 140
C

MAXH = MAXH + NUM

C

N O F = NO F - 1
IF(TODN, LT. 0) GOTO 220
IF(DI, GT. 1) CALL WRITE(EBUF, BREC, DIR)
CALL WRITE(EBUF, 1, SPAR)
CALL CLOSE(EBUF)

220 IF(NOB, GT. 0) GOTO 200
C IF NO OBJECTS FOUND. TYPE MSG AND CALL INTI
CALL TYPE(' NO OBJECTS FOUND: CHECK THRESHOLDS')
CALL TYPE(' B')
CALL APHASE(7)

200 C CONTINUE

IF(TODN, GE. 0) RETURN
CALL MVL(' NO OBJECTS FOUND: CHECK THRESHOLDS')
CALL TYPE(M.6)
CALL WPARAM(122, HK, 6)
CALL APHASE(2)
C

C WRITE NOB PARAMETERS AND CALL INTI
END

SUBROUTINE SKIRT

IMPLICIT INTEGER(A-Z)

COMMON/CT/SPAR(133), SLO, SSO, HLO, HSO, NOB, B, PEP
COMMON/CT/DIR(512), SDIR(512)
COMMON/CT/EBUF(56), EDGE(1924), SEG0(56), SEG(0192)
COMMON/CT/SBUF(9999), SSI(5)
INTEGER PAR(5)
BYTE EBUF, SECB
DATA MAXB/9999, MAXDI/5107, S/1, D/1/
DATA BREC/2, IBREC/2/

C

C THE SEGMENT DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJECT
C SDIR(DI) = BSIC(BEGINNING SEGMENT INDEX)
C SDIR(DI+1) = ESIC(ENDING SEGMENT INDEX)
C
C SDIR(DI+2) = YMIN
C SDIR(DI+3) = XMIN
C SDIR(DI+4) = YMAX
C SDIR(DI+5) = XMAX
C
C THE SEGMENTS ARE STORED AS TRIPLETS
C SEG(5) = Y
C SEG(5+1) = X
C SEG(5+2) = X
C
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 SS WITH (DI, ISWI)

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

CALL AFLIE (EBUF, 1, 'PIC', .5, 5)
CALL OPEN (EBUF, 1024, 0, 'MY2')
CALL AFLIE (SEC8, 5, 'PIC', .5, 5)
CALL OPEN (SEC8, 1024, 0, 'MV5')
CALL G LABEL (EBUF, PAR, IND)
CALL MV L ('SEGMENT FILE', EBUF (IND+73), 13)
CALL G LABEL (SEC8, PAR, EBUF (IND+1))

CALL PUT (SEC8, 1, IND)

CALL READ (EBUF, 1, IND, PAR)
DPER = SPAR (133) - 5
DREC = 2* (NB0 +94)/85 + 2
CALL WRITE (SEC8, 1, SLO)

NL = NLO/2
NS = NSO/2
B2 = 2*B + 1
SINC = MAXS/B2
MAXS = SINC + B2

S = SEGMENT INDEX
SI = SEGMENT BUFFER INDEX

DO 200 OB1 = 1, NOB
IF (DI .NE. 1) GOTO 30
CALL READ (EBUF, IDREC, IND, DIR)

CALL HEE (EBUF, IDREC, IND, S12)
IREC = IDREC + 1
CALL MV L (DIR, SDIR, 512)

CALL INTERP (DI)
NEMD = N*H
NREC = (NEMD+511)/512

DO 130 L = 1, NREC
CALL READ (EBUF, IREC, IND, EDGE)
IREC = IREC + 1
EWH = MINO (NEMD, 512)

ADD HORIZONTAL SKIRT TO SEGMENTS

EDGE (E) = MAXA (EDGE (E)-B, 1)

EWH = MINO (EDGE (E)-B, NS)

IF (L .NE. 1) GOTO 60
YB = EDGE (1)
YEAR = YB - 0

USE FIRST SEGMENT END POINT PAIR TO INITIALIZE SBUF

XI = EDGE (2)
X2 = EDGE (4)

SI = 1
I = 1
EI = 5

CONTINUE

DO 120 E = EI, EWH, 4
Y = EDGE (E)
IF (Y .EQ. YD) GOTO 90

ADVANCE SBUF ONE ROW

SI = SI + SINC
I = I + 1
IF (SI .GT. MAXS) SI = 10
IF(I.GT.B2) I=1
IF(YBAR.LT.0) GOTO 82
SEND = ESI(I)
DO 80 SIP=SI,SEND,2
SEG(S) = YBAR
SEG(S+1) = SBUF(SIP)
SEG(S+2) = SBUF(SIP+1)
80 S = S + 3
C
Y0 = Y0 + 1
YBAR = YBAR + 1
SBUF(SI) = EDGE(E+1)
SBUF(SI+1) = EDGE(E+3)
ESI(I) = SI
C
SIP = 1
X10 = EDGE(E+1)
X20 = EDGE(E+3)
C
DO 100 II=1,02
SIP = SIP
SEND = ESI(II)
X1 = X10
X2 = X20
C
XP1 = SBUF(SIQ)
IF(X2.LT.XP1-1) GOTO 95
XP2 = SBUF(SIQ+1)
IF(X1.GT.XP2+1) GOTO 95
C
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
C
SBUF(SSEND) = MINB(X1,XP1)
SBUF(SSEND+1) = MAXB(X2,XP2)
GOTO 98
C
SIP = SIP + 2
IF(SIP.LE.SEND) GOTO 93
SBUF(SIQ) = X1
SBUF(SIQ+1) = X2
C
ESI(II) = SIQ
100 SIP = SIP + SINC
C
120 CONTINUE
C
EI = 1
NEND = NEND - EI
C
DO 150 II=1,B2
IF(YBAR.GT.YL) GOTO 180
I = I + 1
IF(SI.GT.MAXSB) SI=1
IF(I.GT.B2) I=1
SEND = ESI(I)
DO 140 SIP=SI,SEND,2
SEG(S) = YBAR
SEG(S+1) = SBUF(SIP)
SEG(S+2) = SBUF(SIP+1)
140 S = S + 3
C
YBAR = YBAR + 1
C
150 SDIR(DI+1) = S - 1
C
130 NEND = NEND - EI
C
RELEASE REMAINING SEGMENTS
DO 150 II=1,B2
I:
IF(D1.EQ.1)
COT0 285
CALL WRITE(SECB, DREC, DIR)
CALL WRITE(SECB, DREC+1, SDIR)
C WRITE OUT SEGMENT
DO 210 I=I,6,512
OREC = OREC + 1
CALL CLOSE(SECB)
CALL CLOSE(EBUF)
END
SUBROUTINE CHROME(DUNIT, FILPEX)
BYTE FILPEX(9)
IMPLICIT INTEGER(A-Z)
COMMON/C1/SLO, S50, MLO, M50, N50, B
COMMON/C1/DIR(512), CMVB(1852), (S512), PBUF(1380)
C THE FIRST 28 WORDS OF PBUF ARE THE MV5. FOLLOWED BY 1824 WORDS OF
C BUFFER STORAGE. FOLLOWED BY 256 WORDS OF PICTURE STORAGE
COMMON/C1/SEGB(56), CBUF(56), SDIR(510), LBUN(100)
COMMON/C1/DREC, NFSI
BYTE SEGB,PB(2616)
EQUIVALENCE (PB, PBUF)
INTEGER PAR(5)
DATA SECSIZ/64/, SSNI/63/
DATA MAXBX/510/, MAXBUFF/SEGB, IDREC/2/, LBUF/1052/- DREC/2/
C LPBUF IS THE LENGTH OF PBUF MINUS 256 WORDS OF PICTURE STORAGE
C
CALL SSUTCH(1, ISUI)
IF(ISW. EQ.1) CALL TYPE(' CHROME')
CALL SSUTCH(4, ISW4)
IF(ISW4. NE. 1) GOTO 3
C CALL SPRTN(' CHROME')
CALL TIMER
MAXBUFF=7328
CONTINUE
CALL AFILE(PBUF, DUNIT, FILPEX, 2, 2)
CALL OPEN(PBUF, LBUF=20, l, 0, 0, 'SCN')
NI=(PAR(2)+1)/2
CALL AFILE(SEGB, 5, 'PIC', '5,5')
CALL OPEN(SEGB, 1824, 0, 0, 'MV5')
CALL GLABEL(SEGB, PAR, IND)
CALL ZIA(CMVB, 20)
CALL AFILE(CMVB, 1, 'PIC', '5,5')
CALL OPEN(CMVB, 1824, 1, 1, 'MV2')
CALL MV(' CHROME FILE', PB(IP+100), 12)
CALL PLABEL(CMVB, PAR, PB(IP+1))
CALL READ(SEGB, 1, IND, SLO)
CALL PUT(CMVB, 1, IC)
CALL MV(SLO, CMVB, PB(IP+1), 312)
NREC = (HOB+84)/85
IREC = 2*NREC + 2
OREC = NREC + 3
CAL READ(SEGB, IDREC, IND, DIRE)
CALL PUT(CMVB, DREC, IC)
CALL MV(DIR, CMVB, IC/2+1, 312)
CALL READ(SEGB, IDREC+1, IND, DIRE)
IDREC = IDREC + 2
DREC = DREC + 1
C FORMAT CBUF
BEND = N50+6
SEND = DIR(BEND-4)
DISP = MAXBUFF-SEND
C
J = DISP + 1
READ IN SEGMENT FILE

DO 5 I = 1, SEND - SI2
CALL READ(SEGB, IREC, IND, CBUF(I))
J = J + SI2
IREC = IREC + 1
5 CONTINUE

DO 6 DI = 1, BENG.6
DIR(DI) = DIR(DI) + DISP
6 CONTINUE

THS = (DISP - 1)/SECSIZ
N = THS*SECSIZ

DO 10 I = 1, N, SECSIZ
CBUF(I) = I*SECSIZ

THE SEGMENT DIRECTORY CONTAINS THE FOLLOWING INFORMATION FOR EACH OBJECT

DIR(DI) = BSI = CURRENT SEGMENT INDEX
DIR(DI + 1) = ESI = ENDING SEGMENT INDEX
DIR(DI + 2) = YMIN
DIR(DI + 3) = XMIN
DIR(DI + 4) = YMAX
DIR(DI + 5) = XMAX

THE CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING INFORMATION FOR EACH OBJECT

CDIR(DI) = BSI = BEGINNING SEGMENT INDEX

CDIR(DI + 1) = HSEC = TOTAL NUMBER OF SECTIONS USED UP BY OBJECT
CDIR(DI + 2) = FSI = INDEX OF FIRST SECTION USED BY OBJECT
CDIR(DI + 3) = LSI = INDEX OF LAST SECTION USED BY OBJECT
CDIR(DI + 4) = LWK = FIRST HALFWORD OF SECTION USED BY OBJECT

THE CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING INFORMATION FOR EACH OBJECT

CDIR(DI) = HFSI + 1 = NEXT FREE SECTION INDEX
CDIR(DI) = LFSI = LAST FREE SECTION INDEX

DI = 1

50 DBEG = DI
IF(ISW4.EQ.1) CALL TIMER

X = INCREASE NUMBER OF SECTIONS IF POSSIBLE

CSI = DIR(DI)
BSI = CDIR(DI)
IF(BSI.EQ.0) BSI = CSI
HSEC = LFSI + SECSIZ.
IF(HSI.GT.BSI-SECSIZ) GOTO 52
CBUF(LFSI) = NSI
LFSI = HSI
GOTO 51

52 CBUF(LFSI) = 0
Y = CBUF(CSI)
REC = 2*Y0 - 1
CALL GET(PBUF, REC, IND)
CALL MVW(PBUF, IND, Y0, PBUF(LBUF+1), NW)
CALL GET(PBUF, IND, IP1, Y0, IP1/2)

PROCESS OBJECTS 1 TO NOB
IF(ISW4.EQ.1) CALL TIMER

DO 100 DI = DBEG, BEND, 6
CSI = DIR(DI)
IF(CSI.EQ.0) GOTO 100
Y = CBUF(CSI)
IF(Y.GT.Y0) GOTO 110
ESI = DIR(DI + 1)
LSI = CDIR(DI + 3)
LW = CDIR(DI + 5)
HSEC = 0.
IF(LSI.NE.0) GOTO 58

FIRST SEGMENT OF OBJECT. ASSIGN A SECTION
IF(HFSI.EQ.0) CALL TYPE('WMBUF')
IF(HFSI.EQ.0) CALL TYPE('WMBUF')

4,122,518
117

C DIR(DI) = CSI
C DIR(DI+1) = 1
C DIR(DI+2) = NFSI
C DIR(DI+4) = 1
C DIR(DI+5) = 1
LSI = NFSI
NFSI = CBUF(NFSI)
LH = 1

C LOOP THROUGH EACH SEGMENT OF CURRENT OBJECT
58 DO 90 SI = CSI, CSI + 1
IF(Y GT Y0) GOTO 82
X1 = CBUF(SI + 1) + LPBUF
X2 = CBUF(SI + 2) + LPBUF
C
90 DO 70 L = 1, 2
C
DO 62 X = XI, X2
IF(LH(LT, SEC(SI)) .GT. 0) GOTO 60

C ADD ON A NEW SECTION
60 CBUF(LSI) = NFSI
LSI = NFSI
NFSI = CBUF(NFSI)
LH = 1

C
62 CBUF(LSI + LH) = PBUF(X).
62 LH = LH + 1
C
70 XI = CBUF(SI + 1) + IP
X2 = CBUF(SI + 2) + IP
C
90 CONTINUE
C
C DIR(DI+3) = LSI
C DIR(DI+5) = LH
C END OF OBJECT... WRITE OUT-TO-DISK
C
91 CALL WBUF(DI)
C
CSI = DIR(DI)
IF(CSI .EQ. 0) GOTO 91
C
IF(ISW(EQ. 0) CALL TIMER
GOTO 100
C
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) = LH
DIR(DI) = SI
100 CONTINUE
C
C NO MORE OBJECTS ON THIS LINE... UPDATE AND CONTINUE.
110 CONTINUE
C
IF(ISW(EQ. 1) CALL TIMER
DO 120 DI = BSEG, BEND, 6
CSI = DIR(DI)
IF(CSI .NE. 0) GOTO 50
120 CONTINUE
C
IF(ISW(EQ. 1) CALL TIMER
CALL PUT(CNVB, DEG, IC)
.CALL MVM(LAWCMBR(1/2+1), 102)
CALL CLOSE(CNVB)
CALL CLOSE(PBUF)
CALL CLOSE(SEGB)
IF(ISW(EQ. 1) CALL TIMER
END
SUBROUTINE ROB

IMPLICIT INTEGER(A-Z)

COMMON/C1/CMVGB(56),CS12
COMMON/C1/SHYB(56),EDGE(1024),LBW(68),BUF/6TB
COMMON/C1/SLD,SSO,WKO,WSO,HDG,BB,IPER
COMMON/C1/MIV,MVH,XMIX,XYMIX,XYMAX
COMMON/C1/SIR(68),TIR(68),AREA(68),IOD(68)
COMMON/C1/H1,L1,P1(68),S1,P1(68),PERM(68),CIRCA(68)
LOGICAL=1 BUF(1048)
BYTE CMVBSHVFSTBL(128),MSG7(20),MSG8(25),TMSG(120)
INTEGER DIR(512),PAR(184),HIS(64),OTTAB(60)
EQUIVALENCE (DIR,SDIR(145)),(HIS,EDGE)
DATA MA3XBUF,LOM100000,RECSIZ(1024),MODP/6,
      THC/103701/8,HP1,CMXMBG/30,TOOB/8,MAXHP/180
      DATA OREC/2/,UREC/2/,RFLAG/0/,OTTAB/60/8

THE INPUT CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJ

DIR(B1) = BACKGROUND

DIR(B1+1) = THRESHOLD

DIR(B1+2) = CMX (MINIMUM LINE)

DIR(B1+3) = CMS (MINIMUM SAMPLE)

DIR(B1+4) = CMXL (MAXIMUM LINE)

DIR(B1+5) = CMXS (MAXIMUM SAMPLE)

THE OUTPUT CHROMOSOME DIRECTORY CONTAINS THE FOLLOWING FOR EACH OBJ

SDIR(CDI) = FB0 (FIRST RECORD BLOCK WRITTEN TO DISK)

SDIR(CDI+1) = ML (MINIMUM LINE)

SDIR(CDI+2) = MS (MINIMUM SAMPLE)

SDIR(CDI+3) = LBW (LAST RECORD BLOCK WRITTEN TO DISK)

THE INPUT CHROMOSOMES ARE IN THE FOLLOWING FORMAT

C(1) = PREVIOUS BLOCK WRITTEN

C(2) = MSEC (NUMBER OF SEGMENTS IN THIS RECORD)

C(C1) = Y (LINE COORDINATE=0 - 255)

C(C1+1) = X (STARTING SAMPLE COORDINATE=-255)

C(C1+2) = N (NUMBER OF SAMPLES IN SEGMENT)

NOTE THAT TWO CONSECUTIVE LINES OF SEGMENT GREY VALUES ARE PRESENT

ALSO THE DIMENSIONS ARE THOSE OF THE HALF-PICTURE AND MUST BE

DOUBLED.

THE OUTPUT CHROMOSOMES ARE IN THE FOLLOWING FORMAT

C(1) = MSEC (NUMBER OF SEGMENTS IN THIS RECORD)

C(C1) = Y (LINE COORDINATE=1 - 512)

C(C1+1) = X (STARTING SAMPLE COORDINATE=1 - 512)

C(C1+2) = N (NUMBER OF SAMPLES IN SEGMENT)

CALL SSWAL(1,ISW1)
IF(ISW1.EQ.1) CALL TYPE('ROB')
CALL SSWAL(+1,ISW4)
IF(ISW4.EQ.1) CALL PRINT('ROB')
CALL RPRAHM(HP,PR,MAXHP)
IP=1

IF(IP.GT.HP) GOTO 32
IF(PAR(IP).EQ.'TO') GOTO 30
IF(PAR(IP).NE.'TO') GOTO 35
OTTAB(PAR(IP+2))=PAR(IP+3)
IP=IP+4
GOTO 20
TOSB=PAR(IP+2)
GOTO 20
CONTINUE
RECSIZ=RECSIZ/2
LBW(68) = LAST BLOCK WRITTEN FOR CURRENT CHROMOSOME
NFI = DIRECTORY INDEX
NC = NUMBER OF CHROMOSOMES

CALL AFILW('CHVMB','PIC','55')
CALL OPEN(CMVBSHVFSTBL(128),BUF/6TB)
CALL AFILW('SHVFSTBL)',UCR',13,'13')
CALL OPEN(CMVBSHVFSTBL(128),BUF/6TB,UCR')
CALL GLABEL(CMVB,PAR,IND)
CALL MVLV('RGB FILE','CMVBIND',12)
CALL PLAEOW(DIR,CMVB(IND+1))
CALL SSWAL(1,ISW1)
IF(ISW1.EQ.1) GOTO 37
CALL PRINT('-1')
CALL PRINT(CMVB(IND+73),78)
CALL QPRINT(CMVXH(IND+145),30)
CALL QPRINT(CMVXH(IND+217),70)
CALL QPRINT(CMVXH(IND+201),70)

121

<table>
<thead>
<tr>
<th>SL</th>
<th>SS</th>
<th>HL</th>
<th>HS</th>
<th>#</th>
<th>MAX</th>
<th>LINE</th>
<th>SAMPLE</th>
</tr>
</thead>
</table>
1  37 CALL ITLA(32,TMSGC,128)
    CALL PUT(SMVXH,DREC,IND)
    CALL READ(CMVXH,IND,SLO)
    CALL READ(CMVXH,DREC,IND,DIR)
    DREC = DREC + 1
    CALL READ(CMVXH,DREC,IND,LBW)
    DPER = 100 - DPER
    B = 2*B
    B1 = B + 1
    B2 = B + 2
    D3 = 2*D + 1
    DI = 1
    CBL = 2
    LOOP THROUGH EACH OBJECT
    DO 500 OBNO = 1, NOB
    REC = LBW(OBNO)
    CNL = DRL(DI+2)*2 + 1
    CNXL = DRL(DI+4)*2
    CNXS = DRL(DI+5)*2
    CNL = CNXL - CNL + 3
    CML = CNXL - CMS + 2
    CMS = CMS - 1
    IF(CML.CNS.LT.MAXBUF) GOTO 50
    CALL SWITCH(1,ISUL)
    IF(ISV1.ISN1) GOTO 580
    CALL MLK('OBJECT TOO LARGE',MSG7,20)
    CALL OUTCOM(98,MSG7(10),2)
    CALL TYPE(7,MSG7,20)
    GOTO 580
50

C

READ IN CURRENT CHROMOSOME
    CALL ZIA(BUF1,MAXBUF/2)
    CALL ZIA(HIS,64)

51

CALL READ(CMVXH,REC,IND,C)
    HSEG = C(2)
    CI = 3
C
    DO 60 HNO = 1, HSEC
    BI = 2*(C(CI+1)*CNLS+CCL(CI+1)).+CMS + 1
    W = C(CI+2)
    N2 = 2*H
    IF(BI.LT.1) GOTO 60
    C
    IGNORE POSSIBLE BUF FROM CHROM OUTPUT
    IF(BI.GT.MAXBUF) GOTO 45
    CALL HIS(C(CI+3),BUF(BI),HIS,N2)
    CALL MHIS(C(CI+3),BUF(BI+CNS),HIS,N2)
    CI = CI + N2 + 3
C
    REC = C(1)
    IF(REC.NE.0) GOTO 51

45

CC

IF(0.BN.NOB) GOTO 63
    BI = 1
    CALL PUT(1,BUF(BI),CNLS)
    CALL PRINT(BUF(BI),CNLS)
    BI = BI + CNS
    CONTINUE
C
    BACK = DIR(BI+1)/2
    ITHRES = DIR(BI+1)*2
    C
    REHEALTH OBJECT
    HIS(1) = 0
    DATA = ITHRES
    THRESH = ITHRES
IF(F. 
LE. 
EQ. 0) GOTO 72
HIS(64) = 0
MAX = 0

C

DO 65 I = THRES, 63
F = HIS(I-1) + HIS(I) + HIS(I+1)
IF(F. 
LE. 
MAX) GOTO 65
MAX = F
DATA = 1

65 CONTINUE
THRESH = (NSPER*BACK + NSPER*DATA)/100
IF(OTTAB(0BM). 
NE. 0) THRESH = OTTAB(0BM)/2

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

C

EBI = BUFFER INDEX
EI = EDGE INDEX
DI = DIRECTORY INDEX
CBI = CHROMOSOME DIRECTORY INDEX

C

EBI = CHS
EI = 1
CHL2 = CHL - 2
CHSI = CHS - 1
THRESH = THRESH + 2
NS = CHL2 - B
NSB = NS - B
BCHS = B + CHS

C

DO 399 L = 1, CHL2
EI = EBI - L
EBI = EBI + CHS

210 IF(ISEG(BI, BUF, THRESH, EBI). 
EQ. 0) GOTO 399
SBIR(BDI) = BRC + 1
EDGE(1) = L
S = CHS - (EBI - BI)
EDGE(2) = S
CALL SOTCN(EDGE, BUF(BI), THRESH, CHS)
IF(H.E. 
EQ. 0) GOTO 210
IF(SMBI.EQ.TOBM) CALL LIST(VM, VMX, VMN, VMIN, VMAX, VMAX)
IF(SMBI.EQ.TOBN) CALL PUMP(EDGE, EDGE(2*N), 1)
IF(CG.E. 355) GOTO 399
CALL SORTN(EDGE, H, SMIP)
IF(SMBI.EQ.0) GOTO 215
IF(SMBI.EQ.312) GOTO 214

213 CALL TYPE(' EDGE OVERFLOW... THRESHOLD RAISED')
CALL TYPE(' o')
THRESH = THRESH + 4
GOTO 219

214 IF(ISWI.EQ.1) CALL TYPE(' SORT')
IF(ISW4.EQ.1) CALL STIMER
CALL SORT(EDGE, H)
IF(ISW4.EQ.1) CALL TIMER

215 CONTINUE
IF(SMBI.EQ.TOBM) CALL PUMP(EDGE, EDGE(2*N), 1)
IF(SMBI.EQ.TOBN) CALL LIST(LS(CHL), CHS, 0BM)
C MAKE SURE THAT OBJECT CONTAINS AN INTERNAL POINT
INPUTM = 0
H2 = 2*N
DO 225 H = 1, H2, 4
IF(Y.EQ.EDGE(E)) GOTO 238

225 X1 = MAXX(EDGE(E+1), BI)
X2 = MINX(EDGE(E+3), 55B)
IF(X1, GT, X2) GOTO 235
BB = YACS + X1
I = B0 - B
C

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

228 CONTINUE

C

RIGHT = B0 + X2 - X1 + B

C

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

222 RIGHT = RIGHT + 1

C
C

DO 230 I=LEFT,RIGHT
TP = I - BCNS - I
BP = I + BCNS + 1
DO 225 LL=1,BCNS,CNS
IF(TOBH.EQ.0BN) CALL ILIST(TP,BP,LL,IV(BUF1(TP+LL)),IV(BUF(BP-LL)))
IF(IV(BUF(TP+LL)).EQ.0.OR.IV(BUF(BP-LL)).EQ.0)GOTO 230
CONTINUE

C

THIS IS AN INTERNAL POINT
GOTO 236

C

CONTINUE

C

CONTINUE

C

REMOVE OBJECT FROM BUF.

236

MSEG = 0
NP = 0
IODM = 0
ECI = 1

C

DO 250 E=1,N2+4
Y = EDGE(E)
X1 = EDGE(E+1)
X2 = EDGE(E+3)
NS = X2 - X1 + 1
IF(NS.GT.0) GOTO 230
CALL TYPE(0) GOTO 238
CALL TYPE(0) GOTO 238
CALL TYPE(0) GOTO 238
PAUSE
GOTO 238

C

CONTINUE

M2 = (NS+1)/2
BCI = ECI + 1
ECI = ECI + M2 + 3
IF(ECI.LE.RECSZ2) GOTO 240

C

WRITE OUT CHROMOSOME LINE
C(1) = MSEG
OREC = OREC + 1
CALL WRITE(SNYB,OREC,C)
MSEG = 0
BCI = 2
ECI = M2 + 4

C

240

C(BCI) = CMLM1 + Y
C(BCI+1) = CMLM1 + X1
C(BCI+2) = NS
CALL REMOVE(BUF1(Y=CHN+X1),C(BCI+3),STL,NS,IODM,THRESH)
HP = NP + NS
250

MSEG = MSEG + 1

C

IF(NP.GE.25.AND.INTPNT.NE.0) GOTO 260
CALL SSWITCH(4,1944)
IF(ISWA.NE.4) GOTO 210
CALL GPRINT('OBJECT REJECTED')
CALL ILIST(0BN,N,NP,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(SNYB,OREC,C)

C

NC = NC + 1
ML = YMAX - YMIN + 1
NS = XMAX - XMIN + 1
IODM = IODM - BACK+NP/4
TTAB(NC) = THRESH
LIP(NC) = CMLM1 + L
SIP(HC) = CMRM1 + S
IDON(HC) = IDON

AREA(HC) = HP
PERM(HC) = PERM
CALL SWITCH(0, ISW0)
IF(ISW0.EQ.1) GOTO 200
C PRINT IF SW0 IS UP
CALL OUTCON(NC, TMSG(4), 2)
CALL OUTCON(PERM(HC), TMSG(19), 3)
CALL OUTCON(BACK+2, TMSG(17), 3)
CALL OUTCON(BACK+DATA+2, TMSG(27), 3)
CALL OUTCON(BACK+MAX+2, TMSG(31), 3)
CALL OUTCON(CML, TMSG(39), 4)
CALL OUTCON(CMS, TMSG(47), 4)
CALL OUTCON(CML+1, TMSG(33), 5)
CALL OUTCON(CMS1+5, TMSG(60), 5)
CALL OUTCON(CML, TMSG(65), 5)
CALL OUTCON(NS, TMSG(78), 5)
CALL OUTCON(ITHRES+2, TMSG(69), 3)
CALL OUTCON(THRESH, TMSG(96), 3)
CALL OUTCON(NBO-THRESH, TMSG(94), 2)
CALL OUTCON(NP, TMSG(182), 8)
CALL OUTCON(IODM, TMSG(141), 9)
CALL OUTCON(IODM/(NP/8), TMSG(161), 3)
CALL OUTCON(NC, TMSG(121), 2)
CALL QPRINT(TMSG(120))

C ENTER OBJECT IN CHROMOSOME DIRECTORY
200 SDIR(CDI+1) = CML + YMIN - 1
SDIR(CDI+2) = CMS + XMIN - 1
SDIR(CDI+3) = OREC
CDI = CDI + 4
IF(NC .EQ. MAXNOB) GOTO 510
GOTO 210
C
300 CONTINUE
500 J1 = SI + 6
C
510 CONTINUE
SDIR(1) = CDI/4
NK=SDIR(1)
TEMP=PERM(1)
PERM(1)="64"
C STORE NOB PARAMETER TEMPORARILY IN PERM(1)
CALL WPARM(C122, NK, NOBPH)
PERM(1)=TEMP
C WRITE NOB PARAMETERS
CALL MVW(ITEM, SDIR(241), 60)
CALL WRITE(SMVB, 1, SDIR)
CALL WRITE(SMVB, 2, LIP)
IF(ISW0.EQ.1) CALL QPRINT('1')
CALL CLOSE(SMVB)
CALL CLOSE(CHVB)
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(DUNIT, FILPEx)
IMPLICIT INTEGER(A-Z)
COMMON/C1/ NK, LT, ST, FI
LOGICAL ORDER
BYTE FILPEx(2)
C I/O BUFFERS
BYTE 1, BUF2(12344), OBUF(12344)
C RANDOM AREAS
INTEGER SPAR(16)
BYTE CHAR(2)
C SAMPLE AND LINE TABLE FOR OBJECT NUMBERS

INTEGER ST(68), LT(68), TEMP(68)
DATA Y/0/ X/0/ REPL/1/ MASKN/13/

C INITIALIZE DATA SETS

C CALL SSwitch(1, ISW1)

IF(ISW1.EQ.1) CALL TYPE('NOB')

C READ PARAMETERS

C CALL RParam(NP,NK,128)

IF(FI.NE.0) CALL DClear

C CLEAR THE GRAY SCALE UNLESS FINISHING UP A COUNT

CALL AFFile(IBUF2,DUNIT,FILEX,2,2)
CALL AFFile(IBUF1,'PIC',*,S,*,S)
CALL OPEN(IBUF2,6144,1,0,'SCN')
CALL OPEN(QBUFF,6144,1,1,'NOB')
CALL GLabel(IBUF2,SPAR,I2)
NL=SPAR(1)
NS=SPAR(2)
NUM=N5/2
Ni=1
CALL FLabel(QBUFF,SPAR,IBUF2(12+1))

C SEE IF LT IS IN ORDER

ORDER=.TRUE.
IF(NK.LT.2) GOTO 45
DO 40 HI=2,NK
IF(LT(NK-LT(NK-1))) GOTO 42
40 CONTINUE
45 CONTINUE.

C RUN THROUGH EACH LINE OF THE PICTURE

DO 500 ML=1,ML
CALL GET(IBUF2,L,12)
CALL PUT(OBUFF,L,0)
CALL MVY(IBUF2(I2+1),OBUF(O+1),NK)

C GENERATE NUMBER FOR EACH OBJECT AT APPROPRIATE TIME

C IF(HK.EQ.0. OR W1.GT.NK) GOTO 210
DO 200 HI=NK+1
LD=L-LT(H)
IF(LD.GT.4) GOTO 200
IF(LD.EQ.-4) GOTO 48
IF(ORDER) GOTO 210
GOTO 200
200 CONTINUE

SD=ST(NK)-13
NCHAR=2
IF(HK.GE.19) GO TO 50
NCHAR=1
SD=SD+6
50 IF(SD.LT.1) SD=1
IF(LD.EQ.-4) GOTO 75
IF(LD.EQ.4) GOTO 70
CALL OUTCON(N1,NCHAR(2),2)
CALL TEXT(CHAR(3-NCHAR),NCHAR,LD+3,OBUF(0+SD),1)
CALL ITL(0,OBUF(0+SD)+6*NCHAR)
GO TO 200.
70 IF(ORDER) Hi=Hi+1
75 CALL ITL(0,OBUF(0+SD)+6*NCHAR+1)
200 CONTINUE

C NO DISPLAY IF FINISHING UP A COUNT

CALL DLine(OBUFF(0+1),Y,N5,REPL,0).

580 CONTINUE

C CLOSE DATA SETS
C
CALL CLOSE(1BUFF2)
CALL CLOSE(ODUF)
CALL EXIT
END

C
INT 1 - INTERACTION 1 TO CHECK NOB OUTPUT AND CORRECT FOR ERRORS
SUBROUTINE INT1(COUNT,FILEX)
IMPLICIT INTEGER (A-Z)
COMMON/C1/ NK,LTAB,STAB,FI
INTEGER PAR(10), SPAR(5), SST(513), HST(513), TTTAB(133), MPAR(5)
INTEGER XTAB(60), YTAB(60), LTTAB(60), STAB(60), TTTAB(60), RPAR(100)
BYTE A(1234), BLACK(512), FILEX(3), BPAR(20), KEY(20), LIGHT(512)
BYTE MB(10), HF(21), MMSG(129), MK(0), WHITE(0), CHAR(2)
BYTE BELL(3)
EQUIVALENCE (PAR,BPAR), (HE,TTTAB), (LPS,TTTAB), (SPS,TTTAB)
EQUIVALENCE (NL,SPAR), (HS,SPAR)
DATA BELL/*_tone_0_0*
DATA NTIME/1000/
DATA MKPH/,NOBPH/0,ERAGE/3.1538*,KOUNT/1131086,NOBPH/
DATA KEY/0B0B/, KEY'/H', 'C', 'J', 'B', 'T', 'A', 'F', 'S', 'I', 'N'
DATA /I'/, 'L', 'R', 'M', 'N', 'L', 'O', 'Q'/'0'/*_tone_0_0*
DATA NFLAG/1
C ALWAYS SET -NFLOG TO 1
C FLAG TO CALL NOB FOR A FINISH COUNT
C ITLLA(LIST,512)
CALL ZIA(\WHITE,4)
CALL NVL('/ DN**** BACK**** \MB,18)
CALL NVL('/ DN**** THRESH**** \RT,28)
CALL NVL('/ NC**** \MK,7)
MB(19)=0
MI(20)=0
MK(0)=0
RENUN=0
CALL TYPE('O,CHECK SPREAD',0)
HK=0
CALL RPARAM(NP,PAR,139,NOBPH)
CALL RPARAM(NP,NK,122,NOBPH)
IF(INP.EQ.0)FI=46
CALL DCOUNT(NK,FI)
C DISPLAY THE COUNT
C ITLLA(LIST,529,MSG,129)
CALL RCR
C READ CURSOR ADJUSTMENTS
CALL AFIL(A,BUN,K,FILPEN,2,2)
C OPEN SCAN DATA SET AND READ TTAB
CALL OPENA(A,5144,1,2,'5CH')
CALL GLABEL(A,SPAR,IA)
CALL NVL(A,IA+72,GMSG,64)
C SAVE LABEL FOR PATIENT REPORT
CALL GET(A,SPAR,1,I,IA)
CALL NVL(A,IA+1,TTTAB,133)
CONTINUE
CALL RPARAM(NP,PAR,10)
IF(INP.EQ.0)GOTO 1000
IF(PAR(1).EQ.85)GOTO 65
IF(PAR(1).EQ.84)GOTO 85
DO 59 K=1,1,NK
IF(INP(BPAR(1))*NEIVKEY(K))GOTO 59
GOTO (90,100,200,300,400,500,600,700,800,900,350,900,910,920,930,940,1100,1200,1300,1350,1400,K)
CONTINUE
C TYPE(' PARAMETER ERROR')
C GOTO 25
C RS - RESCAN
C
CALL AFILE(1)
CALL CLOSE(A)
CALL EXIT
CONTINUE

CALL AYPHASE(10)
CALL CLOSE(A)
CALL EXIT

CALL TYPE(' TYPE ONE OF THE FOLLOWING KEYWORDS TO SELECT AN OPTION')
CALL TYPE(' Q - QUICK COUNT')
CALL TYPE(' C - CUT APART A TOUCH (POSITION CURSOR FIRST)')
CALL TYPE(' J - JOIN TWO PIECES TOGETHER (USE CURSOR)')
CALL TYPE(' T - CHANGE THRESHOLD (USE CURSOR)')
CALL TYPE(' B - CHANGE BACKGROUND (USE CURSOR)')
CALL TYPE(' S - SET UP THE CURSOR TO CORRECT FOR DRIFT')
CALL TYPE(' A - ADAPT THIS SPREAD')

CALL TYPE(' T - RESCAN')
CALL TYPE(' F - FINISH THIS SPREAD (NO. KARYOTYPE)')
CALL TYPE(' P - FINISH THIS SPREAD; IT HAS NH CHROMOSOMES')
CALL TYPE(' I - DISPLAY THE INITIAL (UN-NUMBERED) SPREAD')
CALL TYPE(' N - DISPLAY THE NUMBERED SPREAD')
CALL TYPE(' R - ERASE THE LAST CUT OR JOIN REQUEST')

CALL TYPE(' U , D, L, OR R - MOVE THE CURSOR UP, DOWN, LEFT, OR RIGHT')
CALL TYPE(' X - INTERACTIVE COUNT WITH CURSOR AND BELL')
CALL TYPE(' H - ADD A MISSING NUMBER ONLY FOR COUNTS')
CALL TYPE(' X - REMOVE AN EXTRA NUMBER ONLY FOR COUNTS')
CALL TYPE(' Z - ZERO THE CHROMOSOME COUNT')
CALL TYPE(' S - SKIP MOB (USE OLD REC)')
CALL TYPE(' O - MOVE TO OBJECT N TO T)')
CALL TYPE(' I - IF SPREAD IS OK, TYPE CARRIAGE RETURN')
CALL TYPE('0')
GOTO 25

C

C - CUT APART A TOUCH

CALL CURSOR(Y1,X1,SL,SS)
CALL TYPE('MOVE CURSOR TO END OF CUT')
DO 105 J=1,2
DO 105 IT=1,3

105 CALL BLINE(WHITE,Y1-2+I,X1-1,3,I,0)
CALL MAX(LEFT,E1)
CALL PARAM(NP,PAR,10)
IF(NP.EQ.0) GOTO 120
P=PAR(I)

IF(P.EQ.0 OR P.EQ.1 OR P.EQ.2 OR P.EQ.3 OR P.EQ.4 OR P.EQ.5) GOTO 115
IF(P.EQ.1) 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 UCLR(PAR)

C CHECK FOR UP, DOWN, LEFT, OR RIGHT
GOTO 110

120 CONTINUE

CALL CURSOR(Y2,X2,EL,ES)
CALL ADL(SL,SS,EL,ES,SST,NST,ML,HS)
DO 135 L=SL,EL

135 CALL BLINE(LEDGE,Y,X,NST(L),1,0)
CALL PARAM(NP,PAR,10)
IF(Y=BPARI(N),NE,ERASE) GOTO 150

C RESTORE THE DATA IF E WAS TYPED
DO 155 L=SL,EL

155 CALL BLINE(127,Y,X,NST(L),1,-1)
GOTO 25

156 REP=1

170 CALL GET(A,L,1B)
CALL PUT(A,L,1A)
IF(A.L1B) PAUSE 55

C WRITE ZEROS IN THE DATA SET
GOTO 38
135 4,122,518

136

C  J - JOIN TWO PIECES TOGETHER

C 200  CONTINUE

CALL CURSOR(Y1, XI, SL, SS)
CALL TYPE('M)', MOVE CURSOR TO END OF JOIN ', 0)
DO 205 I=1, 3

235  CALL DLINE(255, Y1-2+1, XI-1, 3, 0, -1)

C  MARK END-POINT

210  CALL PARAM(NP, PAR, 10)
      IF(NP EQ 0) GOTO 220
      P=PAR(I)
      IF(P.EQ. 'U', OR. P_EQ. 'D', OR. P_EQ. 'L', OR. P_EQ. 'R') GOTO 215
      IF(P_NE.E('E')) GOTO 220

C  ERASE THE MARK IF E WAS TYPED

DO 212 I=1, 3

212  CALL DLINE(WHITE, Y1-2+1, XI-1, 3, 0, 0)
      GOTO 25

215  CALL UDLR(PAR)
      GOTO 210

220  CONTINUE

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

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

235  CALL DLINE(127, Y, HST(L), 1, -1)

C  ERASE ON THE GRAY SCALE TO INDICATE THE JOIN LINE

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

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

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

C  REUSE THE DATA

GOTO 25

250  RETURN

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

275  CALL ITLA(127, A(L+1), SST(L), HST(L))

C  WRITE 127 IN THE DATA SET FOR THE JOIN LINE

GOTO 30

C

B - CHANGE BACKGROUND

C 300  CONTINUE

CALL CURSOR(Y, X, L, S)
CALL GET(A, L, A)
CALL OUTCONIV(A(L+1), MB(7), 3)
IT=(L/LPS)=NC=S/SPS=I
IF(S/SPS, EQ, NC) IT=IT-1
CALL OUTCON(ITAB(68+IT), MB(16), 3)
CALL TYPE(HA, A)
BGR=69

C SET TO CHANGE BACK

320  CALL PARAM(NP, PAR, 10)
      IF(NP EQ 0) GOTO 25
      IF(NP GT 1) GOTO 35
      IF(PAR(I), GT, 127, OR. PAR(I), LT, 0) GOTO 60
      TH=B(0DRT)=IT+PAR
      CALL GET(A, SPARK(I)+1, 1, A)
      CALL PUT(A, SPARK(I)+1, 1, A)
      CALL NYV(ITAB, A(L+1), 133)
      RETURN

GOTO 25

C

T - CHANGE THRESHOLD

C 480  CONTINUE

IF(LV(MT(7)), NE, IV('W')) GOTO 440

C GOTO 440 IF T WAS ALREADY REQUESTED

C  DISPLAY THE SECTOR BOUNDARIES

DO 410 L=LPS, NL, LPS

410  CALL DLINE(WHITE, L+LPS+HSL, 8, 1, 0)
CALL ITLA(127, BLACK, 512)
DO 415 S=SPS, NS, SPS
415 BLACK(S)=0
DO 420 I=1, 5
420 CALL BLH(E(BLACK, NL+NL+1+1-2, 8, MS, 1, 0)
440 CONTINUE
CALL CURSOR(Y, X, LS)
CALL GET(A, L, IA)
CALL OUTCON(T(IW+IS)), NT(7), 3)
IT=(L/LPS)*MC+S/SPS*1
IF(S/SPS.EQ.MC) IT=IT-1
CALL OUTCON(T+TAB(IW+IS)), NT(10), 3)
CALL TYPE(NT, 0).
BORT=4
C SET TO CHANGE THRESH
GOTO 320
C
C A - ABORT
C
500 CONTINUE
CALL A_PHASE(8)
RUN=0
GOTO 1000
600 CONTINUE
C
C F - FINISH KEYWORD
C
IF(NFLAG.NE.0) GOTO 610
C IF NOB WILL BE CALLED GOTO 610
CALL A_PHASE(HASPH)
CALL CLOSE(A)
CALL AFILE(A, 'PIC ', 5, 5)
CALL OPENA(S, 512, 0, 2, 'NOBS').
C SET TO ADD NK TO NOB OUTPUT LABEL
GOTO 620
610 CONTINUE
F1=8
CALL UPARAM(122, NK, NOBPH)
C SET NOB TO CALL MASK AND NOT PUT UP THE DISPLAY
CALL A_PHASE(HOBPH)
620 IF(NP.EQ.3) NK=PAR(3)
C USE OPERATOR SUPPLIED COUNT
CALL OUTCOM(NK, GMSC(68), 4)
C CONVERT NK FOR GRSC
CALL NVL(GMSC, TEMP, 69).
TEMP(5) = *56.2
C INSEPT LF LF
CALL TYPE(TEMP, 79)
CALL GLABEL(A, SPAR, IA)
CALL NVL(GMSC(67), A(I=290), 2)
CALL PLABEL(A, SPAR, A(I=1).)
C INSEPT NK IN THE LABEL
630 CALL CLOSE(A)
CALL AFILE(A, 'PBTA ', 6, 6)
CALL OPENA(S, 512, 0, 2, 'PRP')
CALL GLABEL(A, SPAR, IA)
NL=NL+1
CALL PLABEL(A, SPAR, A(I=1))
CALL GET(A, NL, IA)
CALL NVL(GMSC, A(I=1), 128)
CALL PUT(A, NL, IA)
PAR(1)= 'K' 
PAR(3)= 'GMT'
CALL NVL(FILPEX, PAR(4), 18)
CALL UPARAM(A, PAR, MASKPH)
CALL close(A)
C
700 CALL SC
GOTO 25
C
C DISPLAY THE RAW-SCAN WITHOUT OBJECT NUMBERS
C
800 CALL DCLEAR
FI=64
MT(7)= 'A.'
C RESET THRESH INDICATOR
DO 820 L=1,ML
CALL GET(A,L,IA)
820 CALL DLINE(A(L+1),2*(L-1),0,HS,1,0)
CALL DCOUNT(NK,FI)
GOTO 25

C DISPLAY THE MOB OUTPUT
C
850 FI=64
CALL UPARAM(122,NK,MOBPH)
CALL APhASE(MOBPH)
CALL CLOSE(A)
CALL EXIT
900 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,NTIME
CALL CURSOR(Y,X,L,S)
IFY.YE.LE.78.OR.XE.LE.XE) GOTO 945
947 CONTINUE
C CURSOR HAS NOT MOVED IN A WHILE
IFY.Y.LT.24.AND.X.GT.1088) GOTO 995
C IF UPPER RIGHT GOTO 995
CALL CURSOR(Y,X,L,S)
IFY.Y.1088.AND.X.GT.1088) GOTO 970
CALL TYPE(BELL.0)
C RING THE BELL
NFLAG=I
IFY.NX.LT.60) NX=NX+I
CALL BCOUNT(NK,FI)
X1AB(NK)=X
X1AB(NK)=L
STAB(NK)=S
DO 950 J=1,2
DO 950 J=1,3
950 CALL DLINE(WHITE,Y-2+1-X-1,1,0,0)
960 CALL CURSOR(Y2,X2,L,S)
IFY.Y2.EQ.Y.AND.X2.EQ.X) GOTO 960
C WAIT FOR CURSOR TO BE MOVED
GOTO 945
970 CALL OUTCON(NK,MK(6),2)
CALL TYPE(NK,6)
IFY.NK.LT.2) GOTO 25
C ORDER THE TABLE.
DO 990 H=2,NK
IFY.LTAB(H).GE.LTAB(H-1)) GOTO 990
LTAB=LTAB(H)
DO 990 J=1,N
IFY.LT.LTAB(J)) GOTO 985
980 CONTINUE
NMV=X-J
CALL MVU(LTAB(J),TEMP,NMOV)
LTAB(J)=LTAB
CALL MVU(TEMP,LTAB(J+1),NMV)
CALL MVU(STAB(J),TEMP,NMOV)
STAB(J)=STAB(N)
CALL MVU(TEMP,STAB(J+1),NMV)
990 CONTINUE
GOTO 25
995 CONTINUE
C ERASE THE LAST SPOT
DO 997 J=1,2
DO 997 J=1,3
141

4,122,518

997 CALL DLINE(127, YTAB(NK)-2+I, XTAB(NK)-1, 3, 0, -1)
IF(NK.GT.0) HK=HK-1
CALL DCOUNT(NK, FI)
GOTO 960
1000 CALL CLOSE(A)
IF(PERUN.GT.0) CALL APARSE(2)
CALL EXIT
C
1100 CONTINUE
C
C M. PARAMETER - ADD A NUMBER FOR THE MISSING OBJECT
C
NFLAG=1
IF(NK.GE.60) GOTO 60
CALL CURSOR(Y, X, L, S)
DO 1110 J=1, 2
DO 1110 I=1, 5
1110 CALL DLINE(WHITE, Y-3+I, X-2, 5, 0, 0)
DO 1120 N=1, HK
IF(L.LT.LTAB(N)) GOTO 1130
1120 CONTINUE
LTAB(NK+1)=L
STAB(NK+1)=S
GOTO 1140
1130 NMOV=NK-N+1
CALL MVW(LTAB(N), TEMP, NMOV)
CALL MVW(TEMP, LTAB(N+1), NMOV)
LTAB(N+1)=L
CALL MVW(STAB(N), TEMP, NMOV)
CALL MVW(TEMP, STAB(N+1), NMOV)
STAB(N)=S
1140 HK=HK+1
CALL DCOUNT(NK, FI)
GOTO 25
1200 CONTINUE
C
C X PARAMETER - REMOVE ONE OF THE NUMBERED OBJECTS
C
NFLAG=1
IF(NK.LT.1) GOTO 60
CALL CURSOR(Y, X, L, S)
MIN=20
C FInd THE OBJECT CLOSEST TO L.S AND NO MORE THAN 20 AWAY
DO 1220 N=1, HK
DIF=ABS(LTAB(N)-L)+ABS(STAB(N)-S)
IF(DIF.GT.MIN) GOTO 1220
MIN=DIF
MAIN=H
1220 CONTINUE
IF(MIN.EQ.20) GOTO 60
N=NMIN
Y=LTAB(N)*2-2
X=STAB(N)*2-2
IF(X.LT.7) X=7
DO 1240 J=1, 2
DO 1240 I=1, 9
1240 CALL DLINE(127, Y+I, X-7, 5, 0, -1)
C ERASE THE OLD NUMBER
NMOV=NK-N
IF(NMOV.EQ.0) GOTO 1250
CALL MVW(LTAB(N+1), LTAB(N), NMOV)
CALL MVW(STAB(N+1), STAB(N), NMOV)
1250 HK=HK-1
CALL DCOUNT(NK, FI)
GOTO 25
C
C Z - ZERO CHROMOSOME COUNT
C
1300 HK=0
GOTO 25
C
C Q - QUICK COUNT
C
1350 PAR(1)=' T0'
PAR(2)=1
CALL UPARAM(I, PAR, 2)
CALL APHASE(2)
CALL UPARAM(122, NK, NOMP)
CALL CLOSE(A)
CALL EXIT
C QUICK COUNT OPTION
C
t - SET OBJECT THRESHOLD
C
CALL MVV(PAR, RPAR(NRP+1), RRP)
RRP=RRP+4
CALL UPARAM(NRP, RPAR, RCBPH)
REUN=1
GOTO 25
END

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

* MOD *

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

MOD PROVIDES:
C A) THE ORIENTATION MECHANISMS EACH INPUT OBJECT
C IS FIRST ENCLOSED IN A MINIMUM ENCLOSED RECTANGLE, THEN, THE OBJE
C IS ROTATED INTO THE INDICATED ORIENTATION, MEASURED
C AND THEN WRITTEN OUT. MOD ACCUMULATES A CHROMOSOME
C DIRECTLY CONTAINING THE RESULTS OF THE MEASUREMENTS
C
SUBROUTINE MOD
C
COMMON/C1-MOD, Dibri
COMMON/C1/CHDIR, SMLBUF, LRGBUF, WS, NL, CURDLN
COMMON/C1/ SST, EST
COMMON/C1/SPIOD, SPLTH, SPAREA
C SMALL BUFFER, FOR HOLDING ROTATED, UNMAGNIFIED OBJECT
BYTE SMLBUF(90,90)
C LARGE BUFFER, FOR HOLDING UNROTATED OBJECT
BYTE LRGBUF(90,90)
C BUFFER FOR I/O
BYTE BUF(2124)
BYTE OBUF(Z124)
C NUMBER OF INPUT OBJECTS
INTEGER NOB
C SYSTEM PARAMETERS
INTEGER SPAR(5)
EQUIVALENCE (SPAR(1), NL), (SPAR(2), WS1)
C PARAMETER BEY FOR DECODING
INTEGER NKEY, KEY(11)
C ENDPOINT TRACKERS (8 FRACTIONAL BITS)
INTEGER XMIN, XMAX, YMIN, YMAX, XSMIN, YSMIN, XSMAX, YSMAX
C STARTING & ENDING SAMPLE TABLE
INTEGER R2, SST(00), EST(00)
C RANDOM AREAS
C RETURN CODE FROM ORIENTATION SUBROUTINE
INTEGER RCODE
C RADIANS/DEGREE
REAL RPD
C OPERATOR SPECIFIED CENOTOMERE POSITION
INTEGER OPCEN(60)
C OPERATOR FLIP FLAG
LOGICAL OPLIP(60)
C OPERATOR SPECIFIED ROTATION
INTEGER OROT(60)
C CENTROMERE LOCATION METHOD
INTEGER CLMETH
C SKELETON FLAG
LOGICAL SKFLG
C SKELETON SWITCH
LOGICAL SKSW
C SKELETON ROUTINE PARAMETERS
INTEGER SKMIN, SKDELT
C PI & PI/2
REAL PI, PIHALF
C DELTA THETA
REAL DELTA
LOGICAL L4
INTEGER SEC/LTH
INTEGER BLK, BS, SL
C CURRENT OUTPUT LINE #
INTEGER CUROLH, BL, CHRENT
C PARAMETER AREA
INTEGER PAR(500)
EQUIVALENCE (PAR, LRBUF)
C
C INPUT DIRECTORY
C
INTEGER IDIR(4, 60)
C FIRST BLOCK WRITTEN
INTEGER FBW
C MINIMUM LINE
INTEGER ML
C MINIMUM SAMPLE
INTEGER MS
C LAST BLOCK WRITTEN
INTEGER LBM
C
C CHROMOSOME DIRECTORY AND ITS FORMAT
C
INTEGER CHDIR(13, 60)
C BLOCK NUMBER OF 1ST LINE OF OBJECT
INTEGER BLKNO
C NUMBER OF SAMPLES IN OBJECT
INTEGER DMRNS
C NUMBER OF LINES IN OBJECT
INTEGER DMRLN
C LENGTH OF CHROMOSOME
INTEGER LENGTH
C INTEGRATED OPTICAL DENSITY
INTEGER ID
C CENTROMERIC INDICES
INTEGER CID, 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 MAKO, MINOA
C DEGREES/ ROTATION INCREMENT
C SAMPLE ACCUMULATORS
INTEGER DEHA(B8), AREA(B8), ACC(176)
EQUIVALENCE(DENA(1), ACC(1), (AREA(1), ACC(89))
C
C TOTAL AREA AND TOTAL DENSITY FOR NORMALIZING
C
INTEGER TAREA, TDEN
C
C ACCUMULATORS FOR CALCULATING CIA AND CID
C
INTEGER CIDM, CAREA
C
C SPREAD MEASUREMENTS
C
REAL SPID, SPAREA, SPLTH
C
SHORT ARM INDICATOR
C
INTEGER SHRTA
C
RANDOM AREAS
INTEGER CEN, CHMPL, S, PER
REAL DPINC
BYTE EM(28)
C REMEASUREMENT CHROMOSOME SELECTOR
LOGICAL SELFGL, SEL(60)
C C DATA STATEMENTS
C
DATA DPINC/2.8125/
DATA MAXCHR/60/
DATA KEY/'11', 'KEY/2', 'AR', 'RD', 'FL', 'CI', 'LU', 'SK', 'SP', 'SE',
   'LD', 'MU', 'MD'/
DATA SKMIN/15/, SXLDT/2/
DATA ID/3/, RP/0.17453/, PI/3.14159/, PIHALF/1.5708/
DATA FW/17/, HL/2/, MS/3/, LBW/4/
DATA DELTA/.85236/
DATA SKFLG/.FALSE. /
DATA BLNKO/1/, BIRMO/2/, DIRL/3/, LENGTH/4/, ID5/5/, C10/6/, CID/7/
DATA CIA/8/, CENLIN/9/, PERIN/10/, AREA/11/, PSDY/12/, CEMETH/13/
DATA OPFLIP/60*.FALSE. /
DATA OPENO/600/, OPROT/600/
DATA MAXNL/25/, MAXNS/25/, CHRET/15/
DATA SELFGL/.FALSE. /
CALL MVL('99 J ** REJECTED **** ** **', EM, 28)
C C INITIALIZE SPREAD MEASUREMENTS
C
SPIOD=0
SPARE=0
SPLIT=0
C UROLN=8
NC=0
C C INITIALIZE CHROMOSOME DIRECTORY
C
CALL ITIA(0, CHDIR, MAXCHR+CHRET)
MINO=30
MAXO=2000
CLMETH=80
SWK=.FALSE.
C DEFAULT IS LN - NO SKELETON AND WIDTH TO LOCATE CENTROMERE
C C RETRIEVE PARAMETERS AND DECODE
C
95 CALL RPARAM(HP, PAR, 590, A)

10 IF(IPH.GT.HP) GO TO 58
   IPN=PAR(IP+2)
   GO TO 12
   J=1
   HKEY
   IF(PAR(IP), EQ, KEY(J)) GO TO (15, 16, 21, 24, 32, 40, 45, 47, 51, 54, 57), J
12 CONTINUE
C C INVALID PARAMETER
C
13 CALL TYPE('** ** PARAMETER ERROR**
   GO TO 95
C C PARAMETER AR - SET MINIMUM & MAXIMUM OBJECT AREA
C
15 MINO=PAR(IP+2)
   MAXO=PAR(IP+3)
   IP=IP+4
   GO TO 19
C C PARAMETER ROTA - ROTATE SPECIFIED OBJECT PRESCRIBED AMOUNT
C
18 IF(IPH.LT.1. OR. IPH.GT.MAXCHR) GO TO 13
   K=PAR(IP+3)/8
   OPRT(IPN)=K+4
   IP=IP+4
   GO TO 19
C C PARAMETER FLIP - FLIP SPECIFIED OBJECT
C
21 IF(IPH.LT.1. OR. IPH.GT.MAXCHR) GO TO 13
C

$e

SKFLG=.TRUE.

__-_

_. ~. ___~ __.. .~ ....__ ~... . ... . . . ... .... ~-----~---- --

C

C PkUAjiETER-

SP SET SKELETON PLLAIIETERS

n

48 K=PCtR(IP+2) ... IF(S LT. l.OR.K.CT.60) CO TO 13

.. .- C PUT

MAX NL IN LhBEL

SO COPY MILL-NOT

CaU-S€-_F3?4

-- .. . . . .

CALL PCABEL(OBUF, SPCIR. 6UF<) 16t 1)

C

C PARAMETER SK - OUTPUT SKELETON PICTURES

C

46 SKFLG= TRUE.

IP=IP+2

GO TO 10

C

C PARAMETER SP - SET SKELETON PARAMETERS

47 SKHIN=PAR(IP+2)

SKDEL=PAR(IP+3)

IP=IP+4

GO TO 10.

C

C PARAMETER SE - SELECT CHROMOSOMES FOR REMEASUREMENT

C NOTE - THIS PARAMETER ASSUMES THAT A CHROMOSOME DIRECTORY

C EXISTS FROM PREVIOUS MOBBING ON RCR

C

45 K=PAR(IP+2)

IF(K.LT.1.OR.K.GT.60) GO TO 13

SELFLG= TRUE

IP=IP+3

DO 48 M=1.K

L=PAR(IP)

SEL(L)= TRUE.

CONTINUE

GO TO 10

C

C PARAMETER LD - LOCATE CENTROMERE BY DENSITY

C

50 CLMETH=0

SWSK = .FALSE.

IP=IP+2

GO TO 10

C

C PARAMETER LW - LOCATE CENTROMERE USING WIDTH AND MODEL(SKELETON)

C

51 CLMETH=0

SWSK = .FALSE.

IP=IP+2

GO TO 10

C

C PARAMETER MD - LOCATE CENTROMERE USING DENSITY AND MODEL(SKELETON)

C

54 CLMETH= 0

IP=IP+2

SWSK= TRUE.

GO TO 10

C

C OPEN DATA SETS

C

55 CALL AFIELE(BUF,3,'UCR','','13','13)

CALL AFIELE(BUF,1,'RCR','','13','13)

CALL OPEN(BUF,1024,1,0,'UCR')

CALL OPEN(BUF,1024,1,1,'RCR')

CALL GLABEL(BUF,SPAR,1B)

SPAR(I)=125

C PUT MAX NL IN LABEL SO COPY WILL NOT CAUSE F374

C

CALL PLABEL(BUF,SPAR,BUF(1B+1))
C READ IN ROB DIRECTORIES
C
CALL GET(BUF.1,1B)
CALL MVL(BUF(1B+1).NOB, MAXCHR*8+2)
CALL GET(BUF.2,1B)
DO 55 I=1,NOB
C DIR(REM I)=IV2(BUF(B1B+241+(I-1)*2))
IF (.NOT. SELFLG) GO TO 60
55
C READ PREVIOUS ROB RESULTS IF SELECTIVELY RE-MEASURING
C
CALL GET(ORB.U.1,1B)
CALL MVL(ORB.U(1B+3).SPIOD,12)
CROU=XV2(ORB.U(1B+15))
NC=IV2(ORB.U(1B+17))
CALL MVL(ORB.U(1B+31),CHDIR(1,1),2*15*30)
CALL GET(ORB.U.2,1B)
CALL MVL(ORB.U(1B+31),CHDIR(1,31),2*15*30)
60 CONTINUE
C MAIN PROGRAM LOOP - EXECUTED ONCE FOR EACH OBJECT TO BE MEASURED
C
DO 500 N=1,NOB
IF (.NOT. SELFLG) GO TO 70
IF (. NOT. SEL(N)) GO TO 500
SPIOD=SPIOD-CHDIR(I0D.N)
SPLTH=SPLTH-CHDIR(LENGTH.N)
SPARE=SPARE-CHDIR(AREA.N)
NC=NC-1
CONTINUE
?
C INITIALIZE STORAGE FOR THE OBJECT
C
CALL ITIA(B9, SST, MAXNS)
CALL ITIA(B, EST, MAXNS)
CALL ITIA(B, LRGBUF.99*99/2)
NL=0
NS=0
C READ & PROCESS OBJECT BLOCKS
C
KK=0
1L=V
BL=0
IFD=IDIR(FUU,W)
IFL=IDIR(LOW,W)
MNS=IDIR(MS.N)-1
C READ EACH BLOCK ONE BY ONE
C
DO 145 BLK=IFB,1IB
CALL GET(BUF.BLK.1B)
ASEC=IV2(BUF(B1B+1))
1=IB+3
C PROCESS EACH SEGMENT WITHIN THE BLOCK
C
DO 140 II=1,HSEG
IF (II.EQ. IV2(BUF(II))) GO TO 110
II=IV2(BUF(II))
III=BL+1
IF (III - GT 88) GO TO 490
110 BS=IV2(BUF(II+2)) - MNS
IF (.LT. 0) GOTO 490
C IGNORE POSSIBLE BUG IN FOB OUTPUT DATA **********************
IF (SST<BL).EQ.(BS<BL).GO TO 496
C
C TRANSFER SEGMENT INTO LRGBUF

C CALL MVL(BUF(I+6),LRGBUF(05,BL),NSAMP)
IF(NSAMP-(NSAMP/2)+2,HE.0),NSAMP=NSAMP+1
I40 I=I+NSAMP+6
I45 CONTINUE

C REJECT OBJECT IF IT IS TOO SMALL OR TOO LARGE
C IF((X.K.GT.MAXOA.0R.KK.LT.MINOA)), GO TO 490

C ORIENTATION THE OBJECT
C CALL ORIG(LRGBUF,SMBUF,ASTU+CHIRSL(N),BL,OPROT(N),RMODE)
NS=CHIR(DIRNS.N)
NL=CHIR(DIRNL.N)
IF(RMODE.NE.0) GO TO 490
NC=NC+1

C BUILD ACCUMULATOR TABLES
C CALL ACCSUB(SMBUF,HL,NS,AREA,DEHA.TAREA,TDEN)

C BUILD AND ANALYZE THE SKELETON IF LOCATING
C THE CENTROMERE BY DENSITY
C METH=1
C ASSUME OPERATOR SUPPLIED CENTROMERE (SET METH=1)
CEN=OPCEN(N)
IF(CEN.NE.0) GOTO 170

C CHECK FOR OPERATOR SUPPLIED CENTROMERE
C METH=0
C SET METH=0 FOR AUTOMATIC CENTROMERE
C IF NOT PSUUK) GO TO 150
C CALL SKSUB(SMLBUF,HL,NS,LRGBUF,CEN,DEHA.SEMIN,SKDEL,CLMETH)
C IF(CEN.NE.0) GO TO 250

C THE PREVIOUS STATEMENTS WERE COMMENTED OUT TO MAKE MORE ROOM
C LOCATE CENTROMERE BY ALTERNATE METHOD
C150 ASSIGN 165 TO LOOP

C LOOK FOR CENTROMERE IN MIDDLE HALF OF IMAGE
C NS1=(NS-(NS/2)+1)/2

C FIND MINIMUM
C155 CONTINUE
NS2=NS-NS1+1
MINA=ACC(NS1+CLMETH)
CEN=NS1
DO 160 L=NS1.NS2
IF(ACC(L+CLMETH).GE.MINA) GO TO 160
MINA=ACC(L+CLMETH)
CEN=L
160 CONTINUE
GO TO 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
C165 IF(CEN.NE.NS1.AND.CEN.NE.NS2) GO TO 230
ASSIGN 230 TO LOOP
NS1=(NS+4)/8+1
GO TO 155

C ADJUST CENTROMERE AS REQUIRED BY OPERATOR
C170 IF(CHDIR(CENLIN,N).LT.9) CEN=NS-CEN+1

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
IF (ACC(CEN-1+CLMETH).LT.MINA) CEN=CEN-1
GOTO 250

180 CEN=CEN+1
GOTO 250
C CHECK FOR DUPLICATE MINIMA AND TAKE AVERAGE IF SO
C
230 J=0
CEN1=CEN+1
DO 240 I=CEN1.NS2
IF (ACC(I+CLMETH).NE.MINA) GOTO 245
240 J=J+1
245 CEN=CEN+J/2
C
C MEASURE THE ARMS
C
250 CALL DPT(SMLBUF.HL.CEN.LD2)
CALL DPT(SMLBUF.HL.L1.LD1)
CALL DPT(SMLBUF.HL.NS.LD3)
AILTH=SQRT((CEN-.5)**2+(FLOAT(LB2-LD2)**2)
ARLTH=SQRT((NS-CEN+.5)**2+(FLOAT(LD3-LB2)**2)
CHRTH=AILTH+ARLTH
ICHR=100.*AILTH/CHRTH+.5
SHRTA=1.
IF(CP(CEN).EQ.0.) GOTO 260
C FIND SHORT ARM FOR OPERATOR SUPPLIED CENTROMERE
IF (CHRIF(CEN).LT.0) SHRTA=-1
IF (OFFLIF(CN); SHRTA=-SHRTA
GOTO 265

260 IF (ICHR.GE.50) SHRTA=-1
265 IF (ICHR.LT.50) ICHR=100-ICHR
IF (ICHR.GT.99) ICHR=99
C
C CALCULATE CID AND CIA
C
CDEN=0
CAREA=0
DO 270 I=1,CEN
CDEN=CDEN+DENA(I)
CAREA=CAREA+AREA(I)
IDR=((200.*CDEN)-(100.*DENA(CEN))+FLOAT(TDEN))/(2.*TDEH)
IF(IDR.LT.80) IDR=100-IDR
IF(IDR.GT.99) IDR=99
IAR=((200.*CAREA)-(100.*AREA(CEN))+FLOAT(TAREA))/FLOAT(TAREA+TAREA)
IF(IAR.LT.80) IAR=100-IAR
IF(IAR.GT.99) IAR=99
C
C ADD MEASUREMENTS TO CHROMOSOME DIRECTORY
C
IF(SHRTA.EQ.-1) CEN=CEN-NS-1
C STORE NEGATIVE CEN TO FLAG SHORT ARM ON RIGHT
CHDIR(CELHLM.NH)=CEN
CHDIR(LENGTH.NH)=CHRTH+.5
CHDIR(LENGTH.MH)=TDEH
CHDIR(AREAH.NH)=TAREA
CHDIR(CELHCM.NH)=ICHR
CHDIR(CIAH.MH)=IAR
CHDIR(CIDM.MH)=IDR
FPAR=FPAR+FPER
IF(FPSD61.GT.327) FPSD61=327.
C CHECK FOR INTEGER OVERFLOW
CHDIR(FPSD66.NH)=100.*FPSD61
CHDIR(CENMTH.NH)=METH
C
C ADD INDIVIDUAL MEASUREMENTS TO SPREAD MEASUREMENTS
C
SPRDE=SPRDE+TDEH
SPAREA=SPAREA+TAREA
SPLTH=SPLTH+CHRTH
C
C STAND CHROMOSOME UP AND WRITE IT INTO ROTATED FILE
C
IP=NS
LPE=1824/HL
CHDIR(BLKNO.NH)=CUROLN
CALL PUT(OBUF, CUROLH, IB)
CALL CUROLH=CUROLH+1
CALL CHROUT(SMLBUF, OBUF(IB*1), NS, ML, LPR, IP, SHRTA, 90)
GO TO 310
CALL CHROUT(LRGBUF, OBUF(IB*1), NS, ML, LPR, IP, SHRTA, 90)
CONTINUE
IF(IP, NE, 0) GO TO 309
GO TO 299
C
OBJECT BYPASSED
C
GO TO 299
C
END OF MAIN LOOP
C
UPDATE CHROMOSOME DIRECTRY ON DISC
C
CALL PUT(OBUF, 1, IB)
CALL ITL2(MOD, OBUF(IB+1))
CALL ITL2(CUROLH, OBUF(IB+15))
CALL MVL(SP10D, OBUF(IB+3), 12)
CALL MVL(CHDIR(1, 1), OBUF(IB+31), 2*30*15)
CALL PUT(OBUF, 2, IB)
CALL MVL(CHDIR(1, 31), OBUF(IB+31), 2*30*15)
CALL CLOSE(OBUF)
C
MEASUREMENTS COMPLETED
C
CONTINUE
C
CALL EXIT
C
END

*******
C* CLASY = COMPILE WITH /CO:25 *******
C* ******
C
CLASY PROVIDES THE CLASSIFICATION MECHANISM FOR THE CLINICALA
C ALMS SYSTEM... ITS INPUT IS THE CHROMOSOME DIRECTRY PRODUCED BY MOB.
C AND ITS OUTPUT CONSISTS OF THE RESULTS OF THE CLASSIFICATION PRO-
CEDURE WHICH IS UTILIZED BY KTYPE TO PRODUCE THE KARYOGRAM.
C
SUBROUTINE CLASY
COMMON+C/ITL, CIG, OFG, ICTB, ILTB
COMMON+C/MOB, SP10D, SPLTH, SPAERA, CHDIR, IOBUF
COMMON+C/MOD, NGAC, GIA, S134, 1SR, OFG, SELT, GCS
C
COMMON REGION
C
NUMBER OF CHROMOSOMES
C
INTEGER MOD
C
CHROMOSOME DIRECTARY
C
INTEGER CHDIR(15, 60)
C
FORMAT OF DIRECTARY
C
BLOCK NO. OF FIRST BLOCK OF AN OBJECT
INTEGER BLKNO
C NUMBER OF LINES IN OBJECT
INTEGER DIRNL
C NUMBER OF SAMPLES IN OBJECT
INTEGER DIRNS
C CHROMOSOME LENGTH
INTEGER LENGTH
C INTEGRATED OPTICAL DENSITY
INTEGER IOD
C CENTROMERIC INDEX BY LENGTH
INTEGER CIL
C CENTROMERIC INDEX BY DENSITY
INTEGER CID
C CENTROMERIC INDEX BY AREA
INTEGER CIA
C CENTROMERE LINE NUMBER IN ORIGINAL PICTURE
INTEGER CENLIN
C PERIMETER
INTEGER PERIM
C AREA OF CHROMOSOME
INTEGER AREA
C PERIM2XAREA
INTEGER FSQDA
C OF DIRECTORY ENTRIES FOR EACH CHROMOSOME
INTEGER CHRENT
C C ID BUFFER
C BYTE IOBUF(4290)
C C RANDOM AREAS
C INTEGER SPARK(10)
INTEGER ICIL(27)
INTEGER SLSIZ, RECSIZ, OFSF, SEX
INTEGER SCSFLG
INTEGER CNRM(28)
BYTE MSGC(29), FMSC(36), SMSC(39), PEM(22)
INTEGER KG(10), JG(18), IC(18), KFAIL(18), JFAIL(18)
INTEGER CT(20, 51), ACT(20, 51), MAC(18), MAL(10)
REAL ST(11)
INTEGER IST(12), HG(11), OLDS(58), NEWS(58)
INTEGER SCG(98), SCLT(5), XTLB(48), ISR(6), CIG(11), IIG(11)
INTEGER CIB(498), OFG(273), ICBGB68, ILCGB68
INTEGER AC(10), AL(18), NING(10), MAXG(18), CTAG(6)
INTEGER OFS(91), PAR(9000), KEY(19)
LOGICAL CID(19)
INTEGER YINT(273), YID, BID
LOGICAL DEBU
C C DATA STATEMENTS
C DATA HG/I2B, HGI/I11, HCG/I2, HGT2/I28, HCTM/I19, YID/Y
DATA CT/1290, 130, 68, 59, 190, 68, 39, 160,
150, 68, 39, 160, 58, 67, 68, 59,
116, 8, 59, 67, 68, 99, 140, 62, 82, 58, 67, 68, 99, 140, 62, 84, 58, 67, 68,
350, 63, 81, 99, 108, 81, 99, 54, 57, 62, 84, 58, 59, 12, 88, 88, 59, 54, 67,
222, 50, 57, 12, 88, 88, 54, 67, 62, 81, 50, 56, 12, 88, 50, 79, 89, 99,
956, 68, 140, 58, 59, 88, 79, 79, 140, 58, 59, 89, 79, 140,
150, 58, 67, 58, 59, 79, 58, 67, 58, 59, 79, 140, 58, 59, 59, 69,
179, 58, 69, 140, 58, 59, 75, 67, 58, 59, 69, 79, 140, 58, 59, 59, 69,
159, 68, 79, 140, 58, 59, 59, 67, 58, 69, 76, 140, 58, 59, 59, 69,
135, 58, 59, 69, 140, 58, 59, 59, 67, 58, 59, 69, 79, 140, 58, 59, 59, 69,
155, 58, 67, 69, 140, 58, 59, 59, 67, 58, 59, 69, 79, 140, 58, 59, 59, 69,
DATA MCT/180, 180, 58, 70, 59, 180, 58, 70, 59, 180, 70, 59,
9 180, 70, 59, 180, 70, 59, 180, 70, 59, 140, 53, 78, 58, 67, 240,
3 180, 58, 67, 140, 53, 58, 58, 67, 58, 58, 67, 240,
5 12, 58, 60, 53, 78, 53, 66, 12, 66, 39, 58, 60, 53, 78, 59, 66, 240,
25 CALL OUTCON(IP, PNM(22), 3)
   CALL QPRINT(PNM, 22).
   GO TO 990

C PARAMETER CT - ADJUST THE DECISION TABLE

1 IL=PAR(IP+2)+1
   IF(IL.LT.1.OR.IL.GT.51) GO TO 25.
   DO 1000 I=1, NGT2
      CT(I)=CT(I)-1
   1000 CONTINUE
   GO TO 70.
1010 IF(IP.GT.NP) GO TO 70.
   DO 1020 J=1, MC
      IF(PAR(IP).EQ.CHAM(J)) GO TO 1930
   1020 CONTINUE
   GO TO 1319.
1030 JJ=J+I
   CT(JJ)=PAR(IP+3)
   IF(JJ.LT.IL) CT(JJ-1, IL)=PAR(IP+2)
   IF(JJ.LT.NP) GO TO 1810.

C PARAMETER NH - NO OBJECT NUMBERS OR CENTROMERE FLAG

1100 NFLAG=1
   IP=IP+2
   GO TO 10.

C PARAMETER NO - SET # OF OUTPUT SAMPLES

1200 NSO=PAR(IP+2)
   NSOF=1
   1203 IP=IP+3
   GO TO 10.

C PARAMETER SI - SET SLO. ID'S

1300 DO 1310 I=1, 90
   1310 SID4(I)=PAR(IP+1+I)
   IF(IP.GT.47) GO TO 10.

C PARAMETER MA - MALE KARYOTYPE

1400 SEX=1
   1410 IP=IP+2
   GO TO 10.

C PARAMETER FE - FEMALE KARYOTYPE

1500 SEX=2
   GO TO 1410.

C PARAMETER NX - NO X SEPARATION FROM C GROUP

1510 NOX=.TRUE.
   GO TO 1410.

C PARAMETER DE - DEBUG OPTION ON

1520 DEBUG=.TRUE.
   GO TO 1410.

C PARAMETER S2 - SET SLOT SIZE

1530 SLSZ=PAR(IP+2)
   GO TO 1203.

C PARAMETER NV - MOVE CHROMOSOMES AROUND FROM PREVIOUS KARYOTYPE

1540 IF(4SMOV.LT.50) NSMOV=NSMOV+1.
   OLD(SNSMOV)=PAR(IP+2)
   NV(SNSMOV)=PAR(IP+3)
C PARAMETER ST - SET SLOPE TABLE

2    DO 2000 I=1,NCT2M1
2000   ST(I)=PAR(IP+1)+PAR(IP+2)+IP=IP+NCT2M1+2
         COTO 18
C
C PARAMETER MI - SET MINIMUM # OF CHROMS. FOR EACH GROUP

3    DO 3300 I=1,NG
3300   MING(I)=PAR(IP+1+I)
         IP=IP+NG2
         COTO 18
C
C PARAMETER MX - SET MAXIMUM # OF CHROMS. FOR EACH GROUP

4    DO 4000 I=1,NG
4000   MAXG(I)=PAR(IP+1+I)
         IP=IP+NG2
         COTO 18
C
C PARAMETER CS - SET CENTER SAMPLES FOR EACH SLOT

5    DO 5000 I=1,99
5000   SCS(I)=PAR(IP+1+I)
         SCSCF=1
         IP=IP+92
         COTO 10
C
C PARAMETER CL - SET CENTER LINE FOR EACH GROUP

6    DO 6000 I=1,5
6000   SCL(I)=PAR(IP+1+I)
         IP=IP+7
         COTO 10
C
C PARAMETER IR - SET INITIAL SLOT FOR EACH ROW

7    DO 7000 I=1,6
7000   ISR(I)=PAR(IP+1+I)
         IP=IP+8
         COTO 10
C
C PARAMETER IC - SET INITIAL SLOT FOR EACH GROUP

8    DO 8000 I=1,NG2
8000   IST(I)=PAR(IP+1+I)
         IP=IP+NG2+2
         COTO 10
C
C PARAMETER CS - MANUALLY INSERT CHROMOSOMES INTO SLOTS

9    OFSF=1
     NOFS=NP-IP-1
     IF(NOFS .NE. 46 .AND. NOFS .NE. 46).GOTO.8950
C SEE IF 46 OR 46 OFS PARAMETERS WERE USED
     DO 8950 I=1,NOFS
8950   OFS(XTBL(I))=PAR(IP+1+I)
C ALLOW FOR THE EXTRA SLOTS
     COTO 78
     8950   CONTINUE
     DO 9800 I=1,99
9800   IF(IP+I-1.GT.NP).GOTO.78
         IP=IP+92
         COTO 18
C
C PARAMETER CC - ADJUST COEFFICIENTS

15    DO 71 I=1,6
9800   CTWH(I)=PAR(IP+I+1)
CONTINUE
IF(CTAB(1)+CTAB(2)+CTAB(3).NE.100) GO TO 900
IF(CTAB(4)+CTAB(5)+CTAB(6).NE.100) GO TO 900
GO TO 900
CALL TYPE('0')
CALL TYPE('0')
CALL TYPE('0')
GO TO 900
C READ IN CHROMOSOME DIRECTORY
C
CALL AFILIE(IOBUF.1,'RCA','RCA',"13","13")
CALL OPENR(IOBUF.1024,'RCA','RCA')
CALL LABELR(IOBUF,SPAR.1B)
LSEX=0
IF(IOBUF(1B+104).EQ.115) LSEX=1
IF(IOBUF(1B+104).EQ.116) LSEX=2
IF(IOBUF(1B+82)).NE.71 GOTO 777
C GOTO 777 IF NOT SOURCE 9, 19, 29, ..., (MONKEY)
C CALL MVU(MC.CT.1820)
C REPLACE HUMAN TABLES WITH MONKEY TABLES FOR SOURCE 9
C CALL MVU(MAL.AC.10)
C CALL MVU(MAL.MA.10)
MING(1B)=0
MAXC(1B)=1
IST(1B)=78
C SET CAY GROUP FOR POSSIBLE I
C
CONTINUE
C
C INITIALIZE CURRENT INDEX AND INITIAL INDEX FOR EACH GROUP
C
C IG(1)=0
C IG(1)=IG(1)-1
C IG(1)=IG(1)-1
C IG(1)=IG(1)-1
C
CONTINUE
C
C SET UP SLOT SIZE IF NOT INPUT
C
IF(SLSIZ.EQ.0) SLSIZ=24
IF(SLSIZ.GT.28) SLSIZ=28
IF(SLSIZ.LT.12) SLSIZ=12
PSSEP=SLSIZ/3
C PAIR SEPARATION
SCS(1)=PSSEP+SLSIZ/2
SCS(2)=SCS(1)+SLSIZ
DO 752 I=3,17,2
SCS(1)=SCS(1)+SLSIZ+PSSEP
DO 752 I=1,18
DO 754 J=1,4
754 SCS(18*J+1)=SCS(1)
756 CONTINUE
750 CONTINUE
C
C CHECK SEX
C
IF(SEX.GT.0) GOTO 76
SEX=LSEX
76 IF(SEX.NE.1) GOTO 77
C MALE
MAXC(5)=15
MING(1B)=MING(1B)+1
77 IF(SEX.NE.2) GOTO 70
C FEMALE
HING(5)=16
MAXC(16)=MAXC(18)-1
IF(SIGA(7B).EQ.YID) SIGA(7B)=BID
C CHAGE 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=I,NOB
IF(CHDIR(BKHO,N).EQ.0) GOTO 200
I=CHDIR(CIL,N)*CTAB(2)+CHDIR(CID,N)*CTAB(3)+CHDIR(CIA,N)
ITAB(6)*100
IL=((FLOAT(CHDIR(LENCH,N)))*CTAB(1)+FLOAT(CHDIR(102,N))
1+CTAB(2)*FLOAT(CHDIR(AREA,N)))*CTAB(3)/1000.*1.
IF(CIL.GT.51) IL=51
CALL OUTCON(N,ICIL(-2))
CALL OUTCON(0,IGIL(-2))
CALL OUTCON(L,ICIL(-2))
IF(CTB(N).NE.1)
ILTB=N
IF(CIL.EQ.0.OR.CHDIR(PSHA:N).LT.1340.OR.CHDIR(AREA,N).GT.600)
100 TO 82
C PUT IN UNKNOWN GROUP IF BLOB OR OVERLAP
DO 80 J=1,HTZTN,2
IF(CIG.GT(J)) AND IC-CT(J+1,IL)) GOTO 100
C IF IC IS WITHIN THE GROUP LIMITS FOR THIS VALUE OF IL GOTO 100
80 CONTINUE
82 J=HGT;
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(-2))
IF(NOJ(N).EQ.25) GOTO 85
NOJ(J)=NOJ(J)+1
CIG(J)=CIG(J)+1
OFG(CIG(J))=N
900 CONTINUE
C
C IF ANY GROUP IS HEAVY OR LIGHT, TRY TO MOVE CHROMOSOMES AROUND
C
210 NFIL=0
220 CONTINUE
KH=0
MDY=0
DO 300 J=2,NC
IF(NOJ(J).GE.MAXJ(J)) GOTO 300
K=J-1
HDN=0
DHM=190000
C GROUP J IS LIGHT, SEE IF A PRECEDING GROUP IS HEAVY
DO 2500 K=1,K2
IF(NOJ(K).LE.HING(K)) GOTO 2500
C GROUP K IS HEAVY
IF(NFIL.EQ.0) GOTO 230
DO 225 N=1,NFIL
IF(K.EQ.NFIL(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
I=IIG(K)
I2=IG(K)
DO 2400 I=11,12
H=OFG(I)
IC=ICTB(H)
IL=ILTBC(H)
JJ=J+J
IF (IC.LT.CT(JJ-1,1L)) OR. IC.GT.CT(JJ,1L)) 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)+1))
IF (D.GE.91) GOTO 2400
C THIS IS THE MOST LIKELY CANDIDATE SO FAR
KS=K
ISAV=1
NCAH=N
NIM=1
GOTO 2400
C MOVE OBJECT H FROM GROUP K TO GROUP J
CALL OUTCON(KSAV,MM(22),3)
CALL OUTCON(J,MM(29),3)
IF (DEBUG) CALL OPRINT(MMS(29))
NOC(J)=NOC(J)+1
CIG(J)=CIG(J)+1
LOGC(J)=LOGC(J)+1
CIG(KSAV)=CIG(KSAV)-1
NOC(KSAV)=NOC(KSAV)-1
ISAV=ISAV+1
IF (ISAV.GE.91) GOTO 3000
DO 2600 I=ISAV,12
C CLOSE UP THE REMAINING OBJECTS IN THE GROUP
3000 CONTINUE
IF (NM.GT.8) GOTO 220
C IF SOMETHING WAS MOVED, SEE IF MORE MOVES ARE POSSIBLE
IF (NM.EQ.8) GOTO 3100
C KH IS HEAVY AND JL IS LIGHT BUT NO SIMPLE MOVE WAS POSSIBLE
C SEE IF THERE IS A COMPLEX (MULTIPLE LEVEL) MOVE THAT CAN BE MADE
L=1
JG(J)=JL
3100 KG(J)=KH
3200 CALL CFO(KG(L),JG(L),IX,AC,AL,CT)
C CHECK FOR A CHROMOSOME IN THE KG-JG OVERLAP AREA
IF (IX.EQ.8) GOTO 3040
I(G(L))=IX
C SAVE INDEX NUMBER
IF (KG(L).EQ.KH) GOTO 3060
C WE NEED TO GO UP ONE LEVEL
L=L+1
JG(L)=KG(L+1)
GOTO 3010
3040 KG(L)=KG(L)+1
C LOOK AT THE NEXT GROUP ON THIS LEVEL
IF (KG(L).LT.JG(L)) GOTO 3020
L=L-1
C GO DOWN ONE LEVEL
IF (L.GT.8) GOTO 3040
C FAILED TO FIND A GOOD MOVE
NF=NF+1
IF (NF.GT.10) GOTO 3100
KF=KF+1
IF (KF.GT.10) GOTO 3100
C MAKE THE L MOVES THAT HAVE BEEN FOUND
DO 3060 M=1,L
ISAV=1CM
JG(M)=JG(M)
OG(M)=OG(JG(M)+1
CIG(JG(M))=CIG(JG(M)+1
3060 CONTINUE
173

174

IFC(CIG(JCM))=OFG(ISAY)
CIG(KCM)=CIG(KCM)-1
HOC(KCM)=HOC(KCM)-1
I2=CIG(KCM)
DO 3070 I=ISAY,12
3070 OFG(I)=OFG(I+1)

3080 CONTINUE
CALL OUTCON(KH,SMSG(22),3)
CALL OUTCON(JL,SMSG(23),3)
IF(DEBUG) CALL QPRINT(SMSG,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 309 J=1,HC1
IF(HOC(J) EQ.0) GOTO 309
I1=IIC(J)
I2=CIG(J)
DO 240 I=11,12
H=OFG(I)
IC=ICTB(H)
IL=ILT(B(H)
YINT(J)=IC-ST(J)*IL
IF(ST(J).LT.0) YINT(J)=YINT(I)
249 CONTINUE
C STORE Y INTERCEPT
I1=IF(ST(J))

242 CONTINUE
L2=IST(J)+HOC(J)-1
IF(L2.LT.1ST(J+1)) GOTO 246
HOC(J)=HOC(J)-1
IF(HOC(HG1).LT.1ST(HG2)-1ST(HG1)) GOTO 244
CALL SSWICH(1,ISW1)
IF(ISW1.EQ.1) CALL TYPE(' ALL SLOTS FULL')
GOTO 242

244 CONTINUE
L2=CIG(HG1)+1
HOC(HG1)=HOC(HG1)+1
OFG(CIG(HG1))=OFG(I2)
I2=12-1
GOTO 242

246 CONTINUE
DO 260 L=L11,L2
MINY=9999
DO 258 I=11,12
IF(YINT(I).GE.MINY) GOTO 259
MINY=YINT(I)
MINI=I
259 CONTINUE
OFS(L)=OF(G(MINI))
YINT(MINI)=9999

269 CONTINUE
C STORE THE GROUP IN ORDER OF YINT
300 CONTINUE
310 CONTINUE
C
C NOW MAKE ANY OTHER ADJUSTMENTS REQUIRED
C
IF(HOX). GOTO 350
C CHECK FOR NO X SEPARATION FROM C GROUP DESIRED
C FIND THE X OR-X-X IN GROUP C

I1=IST(S)
I2=11+12
I3=11+2
I4=12-2
IF(HOC(S).EQ.16) GOTO 330
IF(HOC(S).NE.15) GOTO 350
C C HAS 15. THIRD LARGEST IS X
OFS(I2)=OFS(I3)
DO 320 I=(3,14
320 OFS(I)=OFS(I+1)
OFS(I2-1)=0
GO TO 350
339 CONTINUE
C CLAS 16, J AND 4 ARE X-X
ITEM=OFS(J)
ITEM=OFS(J+1)
DO 344 I=1,13,14
340 OFS(I)=OFS(I+2)
OFS(I1)=ITEM
OFS(I2)=ITEM
350 CONTINUE
IF(ITEM .NE. 5. OR. SEX.EQ.2) GO TO 400
C IF 5 IN GROUP C, FIND THE Y CHROMOSOME
I1=IST(NCG)
I2=I1+4
IF(OFS(I1).EQ.0) GO TO 400
MAX=1
DO 360 I=1,11,12
N=OFS(I)
M=50*CHDIR(AREA,H)/(CHDIR(DIRNL,H)*CHDIR(DIRMS,H))
IF(M.LT.MAX) GO TO 360
MAX=M
MAX=N
360 CONTINUE
I1=I12-1
DO 370 I=MAX,14
370 OFS(I)=OFS(I+1)
OFS(I1)=0
OFS(I2)=MAX
380 CONTINUE
IF(N=0) GO TO 400
NEWSN=NEWSN+1
OLDSN=OLDSN(N)
IF(NEWSN .NE. 0) GO TO 401
C DELETE OBJECT IN OLDSN IF NEWSN = 0
OFS(OLDSN)=0
GO TO 402
401 CONTINUE
NSAVE=OFS(NEWSN)
OFS(NEWSN)=OFS(OLDSN)
OFS(OLDSN)=NSAVE
402 CONTINUE
403 CONTINUE
C ADJUST TABLE FOR OBJECTS WIDER THAN 24 SAMPLES
DO 410 J=1,NGL
II=IST(J)
I2=IST(J+1)-1
DEL=0
DO 405 I=1,12
SCS(I)=SCS(I)+DEL
405 CONTINUE
C ADD ON ANY PREVIOUS DELTA
inc=OFS(I)
IF(INC.EQ.0) GO TO 404
NSMT2=CHDIR(DIRMS,INC)-SLSIZ
IF(NSMT2 .LE. 0) GO TO 404
NSMT2=NSMT2+NSMT2
404 CONTINUE
C ADD DELTA FOR THIS OBJECT
DEL=DEL+NSMT2
405 CONTINUE
IF(SCS(I).GT.586) SCS(I)=586
406 CONTINUE
410 CONTINUE
C IF NSO OF 0 WAS SPECIFIED, DO NOT CHANGE IT
C NOW FIND NSO
DO 420 J=3,6
DO 415 I=1,10
INC=OFS(IOR(J-1))
415 CONTINUE
IF(INC.EQ.0) GO TO 415
C THIS IS THE LAST NON-VACANT SLOT ON THE ROW
ITEM=SCS(1R(J-1)+SLSIZ/2+PSEP
IF(ITEM GT NSO) NSO=ITEM
C ADJUST NSO IF REQUIRED
GO TO 420
415 CONTINUE
CONTINUE
IF(NSO.GT.MAXNSO) NSO=MAXNSO.

C
C OUTPUT RESULTS OF CLASSIFICATION
C
CONTINUE
DO 500 I=1,5
ISRI=ISRI(I).
DO 450 J=1,10
ISP=ISP(J).
IF(ISP.GE.I) TEMP=ISP+J
IF(ISP.EQ.0) GO TO 450
ITEMP=ITEMP+J
IF(IL.T.GT.ITEMP) SCLT(I)=ITEMP

500 CONTINUE
CALL PUT(IOBUF,3,18)
CALL MVL(NSO,IOBUF(1B+1),1B=1294)
CALL CLOSE(IOBUF)
C
C END

C***************************************************
C* KTYPE
C* ***************************************************
C KTYPE PRODUCES THE OUTPUT KARYOGRAM FROM THE CLASSIFICATION TABLES
C PRODUCED BY CLASFY AND THE ROTATED CHROMOSOME IMAGES
C PRODUCED BY ORIENT.
C
SUBROUTINE KTYPE
COMMON/C1/INVBO,IBUF,OBUF,MLT,NS,FBH,CNLMN,METH
COMMON/C1/NSO,NFLG,GID,SID,ISR,OF S,SCLT,SCS
INTEGER SPAR(10),CMETH(40)
INTEGER(ISR(56)
BYTE OBUF(3128),IBUF(2048).
INTEGER FBH(60),MLT(60),CNLMN(60),OS(91),SCS(91)
BYTE LABEL(216)
INTEGER SCLT(5)
CALL MVL('0,'....,...,...,...,...,...,...,'R,78)
C
C OPEN DATA SETS
C
CALL DCLEAR
C CLEAR GRAY SCALE
CALL AFILER(INVBO,1,\'RCR\','*13,'*13)
CALL AFILER(GID,5,\'PIC\','*5,'*5)
CALL OPEN(INVBO,1924,0,0,\'RCR\\
CALL OPEN(GBUP,1936,1,1,\'KGM\\
CALL GLABEL(INVBO,SPAR,1B)
CALL MVL(INVBO(1B+1),LABEL,216)
MLO=0
C
C READ CHROMOSOME DIRECTORY & BUILD TABLES
C
CALL READ(INVBO,1B,IBUF(1))
NSO=IV2(1BUF(1))
DO 16 I=1,30
FBH(I)=IV2(1BUF(1+30*I))
MLT(I)=IV2(1BUF(3+30*I))
NST(I)=IV2(1BUF(5+30*I))
CNLMN(I)=ABS(IV2(1BUF(17+30*I)))
CEMETH(I)=IVIBUF((25+39*I))
IF (NDB.EQ.1) GO TO 40
CONTINUE
CALL READ(INVB.2,IB,IBUF(1))
DO 20 J=1,70
FBN(I+30)=IVIBUF(I+20*I))
NLT(I+30)=IVIBUF(3+30*I))
HST(I+30)=IVIBUF(5+30*I))
CEMLIN(I+30)=TABS(IVIBUF(17+30*I))
CEMETH(I+30)=IVIBUF((25+30*I))
IF (NDB.EQ.1+30) GO TO 40
20 CONTINUE.
C READ IN CLASSIFICATION TABLES
40 CONTINUE
CALL READ(INVB.3,IB,IBUF(1))
CALL NLVL(IBUF(1),NSO.2+294)
IF(NSO.GT.512) GO TO 900
SPAR(1)=512
SPAR(2)=NSO
SPAR(3)=512
CALL PLABEL(OBUF,SPAR.LABEL)
C MAIN LOOP:
C BUILD EACH ROW OF THE OUTPUT KARYOGRAM
DO 500 I=1,5
ISR=ISR(1)
IF(I.GT.5) GOTO 439
C SEE IF THERE ARE ANY OBJECTS ON ROW 5
DO 420 J=1,10
IF(OSF(ISR+J-1).NE.0) GOTO 430
420 CONTINUE
GOTO 500
C NO OBJECTS ON ROW 5
439 CONTINUE
R(2)=GID(I+(I-1)*2)
R(3)=GID(2+(I-1)*2)
DO 450 J=1,10
J=ISR(J-1)
INF=OSF(J)
IF(J.JE.ISR(1)) INP=0
CALL QUICH(INP,R(5+(J-1)*2),-2)
IF(INP.EQ.0) CALL ITL(32,R(5+(J-1)*2))
450 CONTINUE
CALL SSUICH(1,IBIT)
IF(IBIT. NE.1) GO TO 475
CALL TYPE(R,78)
C INVOKE THE ROW BUILDING ROUTINE
475 CONTINUE
CALL KROW(ISR(I+1)-ISR,OFSC(ISR)).SCLT(I).SCS(ISR),NSO,
ISDC(ISR),MFLAG)
NLO=NLO+2*SCLT(I)
500 CONTINUE
C CLOSE UP DATA SETS AND RETURN
SPAR(1)=NLO
CALL PLABEL(OBUF,SPAR.LABEL)
CALL CLOSE(INVB)
CALL CLOSE(OBUF)
CALL EXIT
C ERROR
990 CALL TYPE('*** NSO TOO BIG')
CALL TYPE('**')
CALL EXIT
END
SUBROUTINE KROW(NSL, OFS, SCL, SCS, NSO, SID, NFLAG)

C KROW CONSTRUCTS A ROW OF THE OUTPUT KARYOGRAM. IT IS INVOKED BY
C KTYPE.
C
COMMON /CL, INV8, IBUF, OBUF, MLT, NLT, FB, CE, LIN, CMETH,
INTEGER CE, LIN(60), CMETH(60)
INTEGER X, Y, REPL
BYTE CHAR(4)
INTEGER SCL, S13(90)
INTEGER IDT(60)
INTEGER OFS(91), SCS(90), MLT(60), NLT(60), FB(60)
INTEGER NSL, NSO, SCL2
BYTE OBUF(3128), IBUF(2048), L00, LFF
INTEGER RECI, MAXNSL
DATA L00/0/, LFF/127/, RECI/1824/, MAXNSL/20/
DATA X/0/, Y/0/, REPL/1/

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

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

C MAIN ROW LOOP - PROCESSES EACH LINE WITHIN THE ROW
C
DO 200 L=1, SCL2

CALL PUT(08UFT,0,10)

CALL ZIA(08UFT,10+1), NSO4)

C SLOT LOOP - PROCESSES EACH SLOT WITHIN THE CURRENT LINE
C
DO 150 I=1, NSL
N=OFFS(I)
NI=OFFS(I+1)
IF(N.EQ.0) GO TO 80
C CHECK FOR FIRST SLOT OF A PAIR EMPTY, BUT SECOND SLOT FULL
IF(NL.EQ.0) GO TO 150
C GOTO 150 IF BOTH SLOTS ARE EMPTY
NL2=8
GO TO 130

C DECIDE IF OBJECT APPEARS ON THIS LINE
C
99 CONTINUE
MLT=MLT(N)
ML2=MLT/2
IF(L.LT.SCL-MT2 OR L.GE.SCL+ML2) GO TO 120
L=L-MT2

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

C GET FIRST BLOCK OF THE OBJECT
C
IB=FBK(N)
IF(IB.EQ.0) GO TO 120
C BYPASS OBJECT...IF NOT IN DIRECTORY.
   LPBN=RECSIZ/HSTN
   ML0=MOD(L0,LPBN).
   IF(ML0.GT.0) GO TO 98
   CALL READ(MYBP,IA+LO/LPBN,IA,IBUF(IBIT(I)))
98   I=IDT(I)+ML0+HSTN
C
C TRANSFER OBJECT SEGMENT INTO THE OUTPUT BUFFER
C
   CALL NVL(IBUF(JI),OBUF(IO*IS),HSTN)
   IS=IS+HSTN
C
C SEE IF THIS IS THE CENTROMERE LINE.
C
   IF(HFLAG.NE.0) GO TO 158
   IF(LO.NE.CENLIN(N)-1) GO TO 159
C
C CENTROMERE LINE, FIND WHERE TO PUT THE CENTROMERE ARROWS
C
   IT=IS-HSTN
   ITMAX=IS-4
   DO 110 IU=IT,ITMAX
   IF(OBUF(I+IU).GT.100) GO TO 112
110 CONTINUE
   GO TO 117
112 CONTINUE
   IT=IS
   115 IT=IT-1
   IF(OBUF(I+IT).EQ.100) GO TO 119
117 CONTINUE
   GO TO 117
118 CONTINUE
   IT=IT+1
   GO TO 110
C
C CHECK TO SEE IF LINE CONTAINS OBJECT #
C
   120 CONTINUE
   IF(HFLAG.NE.0.OR.L.GT.SCL-HL2-9.OR.L.GT.SCL-HL2+3) GO TO 130
C
C LINE CONTAINS OBJECT #, SO PUT NUMBER INTO OUTPUT BUFFER
C
   LO=L-SCL+NL2+9
   IS=SCS(I)-11
   IF(IS.GT.500) IS=500
   IF(H.LT.10) IS=IS-3
   NL3=NL2(RH)
C
   IF(CEMETH(N).EQ.1) HN=H
   C FLAG OPERATOR CORRECTED CENTROMER.
   CALL OUTCOM(HN,CHAR(3),3)
   CALL TEXT(CHAR(3),LO,OBUF(IO*IS),1)
   GO TO 150
C
C CHECK TO SEE IF LINE CONTAINS GROUP ID
C
   130 CONTINUE
   IF(I.EQ.(I/2)+2) GO TO 150
   IF(H.LT.10) GO TO 140
   ML3=NL3(NH)/2
   IF(NL3.GT.NL2) ML2=NL2
140 CONTINUE
   IF(LO-SCS+NL2+5.OR.L.GT.SCL+NL2+10) GO TO 150
C
C LINE CONTAINS GROUP ID, SO OUTPUT A LINE OF THE ID TO THE OUTPUTBUF
C
   LO=(L-SCL-NL2-3)/2
   IS=(SCS(I)+SCS(I+1))/4)*2-23
   IF(IS.GT.464) IS=464
   CALL TEXT(SID(I),4,LO,OBUF(IO*IS)-23)
150 CONTINUE
   IF(Y.LT.184) CALL DLNE(OBUF(IO*IS),Y,Y,HSD,REPL,9)
   Y=Y+REPL+1
200 CONTINUE
RETURN
C BUFFER TOO SMALL TO HANDLE THE ROW

900 CONTINUE
   CALL TYPE(’**.**.BUFFER_TOO_SMALL_FOR_ROW’)
   CALL TYPE(’@’)
   RETURN
C
C NS0=0
C
910 CONTINUE
   CALL TYPE(’*** NS0=0’)
   CALL TYPE(’@’)
   RETURN
END

SUBROUTINE INT2

C
C
C
C
C
C
C C THIS ROUTINE IS THE SECOND OPERATOR INTERACTION PHASE. IT
C ALLOWS THE OPERATOR TO EITHER APPROVE THE KARYOTYPE WHICH IS
C BEING DISPLAYED ON THE GRAY SCALE DEVICE, FOR HARD COPY OUTPUT
C OR ELSE MAKE CHANGES VIA MOB AND/OR CLASSFY/KTYPE PARAMETERS:
C
C C COMMON REGION:
C COMMON/CI/NS0,NFLAG,C10,SID,ISR.OFS,SCLEN,SCO
C COMMON/CI/CHDIR,MOB,SPTOS,SPLTN,SPAREA,SCCLST,NC
C I/O BUFFER
   BYTE I0BUFF(2124)
C MOB PARAMETERS AND CURRENT PARAMETER HIGH WATER MARK
   INTEGER NOBPAR(500),MOBPAR(4)
C CLASSFY/KTYPE PARAMETERS AND CURRENT PARAM. HIGH WATER MARK
   INTEGER KTPAR(500),K
C INT2 LOCAL PARAMETER BUFFER AND POINTER
   INTEGER PAR(100),IFP
C RE-MOB CHROMOSOME SELECTOR
   LOGICAL SEL(60)
C CLASSIFICATION TABLES
   INTEGER CURDEN
C CHROMOSOME DIRECTORY AND ITS FORMAT
C
   INTEGER CHDIR(15,60)
C BLOCK 
   INTEGER BLKNO
C NUMBER OF LINES IN OBJECT
   INTEGER DIRNL
C NUMBER OF SAMPLES IN OBJECT
   INTEGER DIRNS
C LENGTH
   INTEGER LENGTH
C INTEGRATED OPTICAL DENSITY
   INTEGER IO0
C CENTROMERIC INDICES
   INTEGER CIL,C10.C1A
C CENTROMERE LINE 
   INTEGER CEHLEN
C PERIPHER
   INTEGER PERIM
C AREA
   INTEGER AREA
C PERIM SQUARED DIVIDED BY AREA
   INTEGER PSoDA
C CENTROMERE LOCATION METHOD
   INTEGER CMETH
C PHASE NUMBERS FOR MOB AND FOR CLASSFY/KTYPE
   INTEGER MOBPH,KTYPHA,FOBPH,FOBPH,BANDPH,FORPH
C RE-RUN FLAGS
   LOGICAL MOBFLG,KTFLAG,ALLFLG,OSFLG
C RANDOM AREAS
   INTEGER SLCL,SLCS,SLID,S,Y,S,L,S,L2
INTEGER SPAR(18), SST(513), NST(513), S1, S2
INTEGER KEY(37)
BYTE LABEL(73), BLACK(72), PARBUF(90), MOBB(568)

CLASSIFICATION TABLES

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

DATA STATEMENTS

DATA BLKNO/1/, DIRNL/2/, DIRNS/3/, LENGTH/4/, IDB/5/, CSL/6/, CID/7/
1. CLR/B/, CENL/N/., PERI/10/, AREA/11/, PSQDA/12/, CENETH/13/
DATA MAXPAR/300/, FOSPHA/6/, MOBPH/6/, KTYPE/9/, BAMPH/14/
DATA DECPB/57.2958/, FOURPH/15/
DATA LABEL/73*, 'BLACK/72*127/
DATA NKEY/37/
DATA MOBFLG/., FALSE/., KTYFLG/., FALSE/., ALLFLG/., FALSE/., OFSFLG/., FALSE/.,
CALL MYLC('ARLWU 2 BCALEXMMXMACFEXXSEP, SPXIRIGCDSRF
CALL TYPE('CHECK KARYOTYPE', B)
CALL RCA
CALL CURSOR ADJUSTMENTS
CALL AFIL((IOBUF, I, 'RCR', '*13,'*13)
CALL OPEN(IOBUF, 1024, 1, 'RCR')
CALL LABEL(IOBUF, SPAR, 1B)
CALL MYLC(IOBUF, IB+1+72), LABEL(2), 72

READ IN DIRECTORY AND CLASSIFICATION TABLES

CALL GET(IOBUF, I, 1B)
CALL MYLC(IOBUF(IB+1), MOB, 10)
CALL MYLC(IOBUF(IB+3), CHDIR(1, 1), 2*15*30)
CALL GET(IOBUF, 2, 1B)
CALL MYLC(IOBUF(IB+3), CHDIR(1, 31), 2*15*30)
CALL GET(IOBUF, 3, 1B)
CALL MYLC(IOBUF(IB+1), N50, 2*294)

READ PREVIOUS PARAMETERS

CALL RPRA(N(M, MOBPAR, MAXPAR, MOBPH)),
CALL RPRA(K, KTPAR, MAXPAR, KTYPH),
IF(K < 92) GOTO 65
K=K-92

OSFLG = TRUE.
DO NOT RECLASSIFY IF OFS PARAMETERS ARE PRESENT
ERASE OLD OFS 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 150
C REMOVE OLD SE PARAMETERS
75 N=1-1
C REQUEST OPERATOR INTERACTION
100 CONTINUE
110 CALL PARAM(NP, PAR, 100, PARBUF)
IF(NP.EQ.0) GO TO 500
IP=1

SECODE PARAMETERS

DO 200 J=1, NKEY
IF(PAR(1P).EQ.KEY(J)) GOTO (1000, 1050, 900, 910, 950, 1250, 1300, 1350
1, 1450, 1500, 1550, 1600, 1650, 1700, 1800, 1850, 1900, 1950, 2000, 2050, 2125, 2150
2 2220, 2025, 1190, 1150, 1100, 1250, 1750, 2100, 1950, 1850, 1800, 1750, 2300, 2350
3, 2350, 2400, 2450, 2500), J
200 CONTINUE
CALL TYPE('PARAMETER ERROR')
CALL TYPE('0')
GO TO 100
CALL TYPE('CURSOR ERROR')
CALL TYPE('E')
GOTO 100
1 CALL TYPE('ONE OF THE FOLLOWING KEYS TO SELECT AN OPTION')
CALL TYPE('C - CHANGE CENTROMERE (USE CURSOR)')
CALL TYPE('F - FLIP AND CHANGE CENTROMERE (USE CURSOR)')
CALL TYPE('R - ROTATE CHROMOSOME (USE CURSOR)')
CALL TYPE('H - MOVE CHROMOSOME TO ANOTHER SLOT')
CALL TYPE('X - REMOVE CHROMOSOME FROM KARYOTYPE')
CALL TYPE('P - PUSH A GROUP OF CHROMOSOMES RIGHT OR LEFT')
CALL TYPE('L - ADD A LABEL (TYPE LABEL ON SAME LINE)')
CALL TYPE('A - ABORT')
CALL TYPE('S - SET UP CURSOR TO CORRECT FOR DRIFT')
CALL TYPE('U OR D - MOVE CURSOR UP OR DOWN')
CALL TYPE('R - RERUN FOB')
CALL TYPE('S - DISPLAY SPREAD')
CALL TYPE('B - CALL THE BANDED CLASSIFIER')
CALL TYPE('R - RERUN THE BANDED CLASSIFIER')
CALL TYPE('Q - QUICK BAND CALCULATION')
CALL TYPE('W - DISPLAY WAVEFORMS')
CALL TYPE('X - DISPLAY AXES')
CALL TYPE('C - CLEAR CLASSIFY PARAMETERS')
CALL TYPE('G - CLEAR MOD PARAMETERS')
CALL TYPE('M - MALE OR FEMALE')
CALL TYPE('X - NO X SEPARATION FROM Y GROUP')
CALL TYPE('IF KARYOTYPE IS OK, TYPE CARRIAGE RETURN')
CALL TYPE('B')
GOTO 100

C
C PARAMETER U OR D - MOVE CURSOR UP OR DOWN

C
C
90 CALL MCD
GOTO 110
91 CALL MCD
GOTO 110
C
C PARAMETER BC - CALL BANDED CLASSIFIER

C
C
950 CALL APHASE(BANDPH)
GOTO 709
C
C PARAMETER AR - SET MAXIMUM AND MINIMUM AREAS OF BAND CHROMOSOMES

C
C
100 CALL HVL(PAR(IP),MOBPAR(M+1),0)
M=M+4
MOBFLG=.TRUE.
GOTO 110
C
C PARAMETER LV - LOCATE CENTROMERE BY WIDTH RATHER THAN DENSITY

C
C
1050 CALL HVL(PAR(IP),MOBPAR(M+1),4)
M=M+2
MOBFLG=.TRUE.
GOTO 110
C
C PARAMETER CI - SET CENTROMERE OF CHROMOSOME SPECIFIED TO THAT
C INDICATED BY CURSOR POSITION

C
C
1100 CALL KURSOR(Y,X,L,S,LID,SLCL,SLCS,N)
IF(N.EQ.0)GOTO 250
SEL(N)=.TRUE.
CALL HVL('C1',MOBPAR(M+1),2)
MOBPAR(N-3)=N
INL=CHDIR(INL,N)
MOBPAR(M+4)=L-(SLCL-INL/2)+1
CALL DLINE(BLACK,Y,X,4.1,-1)
CALL DLINE(BLACK,Y,X,4.1,-1)
IF(MOBPAR(M+4).GE.INL.OR.MOBPAR(M+4).LE.1)GOTO 250
M=M+4
MOBFLG=.TRUE.
GOTO 110
C
C PARAMETER FL - FLIP SPECIFIED CHROMOSOME AND SET CENTROMERE

C
C
1150 CALL KURSOR(Y,X,L,S,LID,SLCL,SLCS,N)
IF(M.EQ.0) GOTO 250
SEC(N) = TRUE.
CALL ML('FL', MOBPAR(M+1), 2)
MOBPAR(N+3) = N
INL = CHDIR(DIRNL, H)
MOBPAR(N+4) = - ((SLCL - INL/2) + 1)
CALL DLINE(BLACK, Y, X, 4, 1, -1)
CALL DLINE(BLACK, Y, X, 4, 1, -1)
IF(MOBPAR(M+4).GE.INL.OR.MOBPAR(M+4).GE.INL) GOTO 250
MOBFLG = TRUE.
GO TO 110

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

1200 CALL KURSOR(Y, X, L, SL, SLCL, SLCS, N)
IF(M.EQ.0) GOTO 250
CALL DLINE(BLACK, Y, X - 2, 3, 1, -1)
CALL DLINE(BLACK, Y, X - 2, 3, 1, -1)
SEC(N) = TRUE.
CALL ML('RO', MOBPAR(M+1), 2)
MOBPAR(N+3) = N
FRSTY = Y
FRSTX = X
CALL TYPE('MOVE CURSOR TO END OF AXIS':0)
CALL PARAM(MP, PAR, 198)
CALL KURSOR(Y, X, L, SL, SLCL, SLCS, N)
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
FRSTY = Y
FRSTX = X

1210 THETA = ATAN2((SX-FRSTX), (SY-FRSTY))
GO TO 1215

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

C PARAMETER AL - RE-MOB ALL CHROMOSOMES
C
1250 ALLFLG = TRUE.
GO TO 110

C PARAMETER SK - OUTPUT SKELETON PICTURES INSTEAD OF GREY LEVEL PICTURE
C
1300 MOBPAR(M+1) = PAR(IP)
M = M + 2
MOBFLG = TRUE.
GO TO 110

C PARAMETER NN - NO OBJECT NUMBERS
C
1350 KTYPE = PAR(IP)
K = K + 2
KTYPEFLG = TRUE.
GO TO 110

C PARAMETER MA - MALE KARYOTYPE
C
1450 KTYPE = PAR(IP)
K = K + 2
KTYPEFLG = TRUE.
GO TO 110

C PARAMETER KC - CLEAR CLASY/KTYPE PARAMETERS
C
1500 K = 0
CALL WPARAM(K,KTPAR,KTPHA)
OSFLG=.FALSE.
KTYFLG=.TRUE.
GO TO 110

C PARAMETER FE - FEMALE KARYOTYPE

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

C PARAMETER PX - HO X SEPARATION FROM C GROUP

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

C PARAMETER DE DEBUG OPTION ON

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

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

C 1700 CALL KURSOR(Y,X,L1,S1,S1,L1,SLCL,SLCS,N)
CALL TYPE('+MOVE CURSOR TO END OF PUSH ','0')
CALL PARAM(KP,PAR,100)
CALL KURSOR(Y,X,L2,S2,SL2,SLCL,SLCS,N)
IF(S1.LT.S2) GOTO 1725
C GOTO 1725 FOR PUSH TO THE RIGHT

C PUSH LEFT

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

C PUSH RIGHT S1 LT S2

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

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

C 1750 CALL KURSOR(Y,X,L1,S1,S1,L1,SLCL,SLCS,N)
CALL BLINE(BLACK,Y,2,3,1,-1)
S1SLID=SL1
CALL TYPE('+MOVE CURSOR TO OTHER SLOT ','0')
CALL PARAM(KP,PAR,100)
CALL KURSOR(Y,X,L2,S2,SL2,SLCL,SLCS,N)
CALL ABP(L1,S1,L2,S2,SST,NST,SL1,SL2)
GO 1770 L=L1,L2
Y=2*(L-1)
4,122,518

195

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

CALL .DLINE(I??.YJ_X_.HST(t). 1:
-11.

X=2*(SST
C
L) -1 1
SYI
D=OFS
(SL
D)
OFS
(SLID) =OFS
(SYSL
ID)
OFS
(SVSLID)

KTYFLG=. TRUE.

196

CALL .DLINE(I??.YJ_X_.HST(t). 1:
-11.

X=2*(SST
C
L) -1 1
SYI
D=OFS
(SL
D)
OFS
(SLID) =OFS
(SYSL
ID)
OFS
(SVSLID)

KTYFLG=. TRUE.

GO TO 110

C PARAMETER SP - SET SKELETON PARAMETERS
C

1900

CALL MVL(PAR(IP), MOBPAR(M+1), 8)
K=K+4
MOBFLG=. TRUE.

GO TO 110

C PARAMETER MX - SET MAXIMUM # OF CHROMOSOMES PRE GROUP
C

1950

CALL MVL(PAR(IP), KYPAR(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), KYPAR(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), KYPAR(K+1), 28)
K=K+14
KTYFLG= TRUE.

GO TO 110

C PARAMETER MC - CLEAR MOB PARAMETERS
C

2050

M=0
MOBFLG= TRUE.

GO TO 110

C PARAMETER SC - SET UP THE CURSOR

2100

CALL SC

GO TO 110

C SETUP THE CURSOR
C

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, MOBPHA)

CALL WPARAM(0, KYPAR, KYPHA)

GOTO 700

C RERUN FOB
C

C PARAMETER AB - ABORT
C

2200

CALL APHASE(0)

GOTO 700

C PARAMETER X - REMOVE OBJECT FROM KARYOTYPE
C

2250

CALL KURSOR(Y, X, LI, SLID, SLCL, SLCS, K)
DO 2250 I=2, 26, 2

CALL BLHE(BLACK, Y+I-14, Y+I-16, 2, 1, -1)

CALL BLHE(BLACK, 1, Y-I+14, X+I-16, 2, 1, -1)

2260

CONTINUE

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

GOFLG=.TRUE.
KTYFLG=.TRUE.
GOTO 110

C PARAMETER L - ADD A LABEL TO INDICATE DIAGNOSIS

300 DO 2310 1=4,72

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

LABLEN=1-3
CALL AFILE(NOBB,1,'PIC ',5,5)
CALL OPEN(NOBB,512,6,2,'NOB')
CALL GET(NOBB,1,IN)
CALL PUT(NOBB,1,IN)
CALL AFILE(NOBB,3,NOBB(IN+299),LABLEN)
CALL CLOSE(NOBB)

C ALSO ADD TO THE NOB OUTPUT LABEL FOR RESEL's BENEFIT
CALL GLABEL(IOBUF,SPAR,1B)
CALL MLV(PARBUF(3),IOBUF(IB+299),LABLEN)
GOTO 110

C PARAMETER WAVE - DISPLAY WAVEFORMS FROM BAND

2350 CALL UPKRMAN(2,'WAVE',BANDPH)
GOTO 950

C PARAMETER AXIS - DISPLAY AXES FROM BAND

2400 CALL UPARAM(2,'AXIS',BANDPH)
GOTO 950

C PARAMETER RB - RERUN BANDED CLASSIFIER

2450 CALL APHASE(FOURPH)
GOTO 700

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

2500 CALL SSWITCH(7,ISU7)
IF(ISU7 .NE. 1) CALL UPARAM(2,'QB ',BANDPH)
GOTO 950

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

300 IF(.NOT.ALLFLG.AND..NOT.KTYFLG.AND..NOT.MOBFLG) GO TO 700
IF(.NOT.OSFLG) GOTO 550

C WRITE THE OFS PARAMETERS FOR CLASYFY

C CALL MLV('OS',KTPAR(K+1),2)
CALL MLV(OSKS,KTPAR(K+3),90)
K=K+92

558 CONTINUE
IF(.NOT.ALLFLG.AND..NOT.MOBFLG) GO TO 650
IF(ALLFLG) GO TO 610

C MUST RE-RUN NOB ON SELECTED CHROMOSOMES

C CALL MLV('SE',MOBPAR(M+1),2)
M=M+3
M=M+M
M=M+M
DO 600 N=1,NOB
IF(.NOT.SEL(N)) GO TO 600
M=M+1
M=M+M+1
MOBPAR(N)=N
CONTINUE
IF(M EQ.0) GO TO 695
MOBPAR(M)=N
C CALL PUT(IOBUF,1,1B)
CALL MLV(NOB,IOBUF(IB+1),10)
CALL MLV(CHDIR(1,1),IOBUF(IB+31),2*15+30)
CALL MLV(MBUF(ML+73), LABEL, 64)
CALL MLV(MBUF(ML+143), LABEL, 64)
CALL MLV(MBUF(ML+217), LABEL, 85)
CALL MLV(MBUF(ML+289), LABEL, 105)
C SAVE LABEL INFORMATION
C
C OPEN KROM DATA AND GET NL

CALL AFILE(KBUF, 4, 'KDATA', 6, 6)
IF (CODE.LT. 0) CALL AFILE(KBUF, 4, 'KDATA', 6, 6)
CALL OPEN(KBUF, 1024, 0, 2, 'KRA')
CALL GLABEL(KBUF, KSPAR(KL))
NL=KSPAR(1)
C READ SOURCE DIRECTORY RECORD
C
10 CALL GET(KBUF, SCODE, KD)
IF (HUSED.LT. 95) GOTO 22
C IF SPACE IN THIS RECORD GOTO 83
C IF NEXT EQ. 8) GOTO 29
C IF LAST DIRECTORY RECORD GO MAKE A NEW ONE
SCODE=NEXT
GOTO 10
C GO READ NEXT DIRECTORY RECORD
NL=NL+1
NEXT=NL
C STORE NEXT RECORD IN THIS DIRECTORY RECORD
CALL PUT(KBUF, SCODE, KD)
CALL GET(KBUF, NL, KD)
HUSED=0
22 CONTINUE
DO 24 I=1, 421, 5
C IF THIS SLOT IS FREE GOTO 26
C CALL TYPE(' RESOL ERROR')
C CALL TYPE(' 0')
PAUSE 12345
26 CALL MLV(LABEL(21), PID(I), 10)
C STORE PATIENT ID
HUSED=HUSED+1
IR=(I+4)/5
REC(RECST(I))=IR
IF(REC.NE. 0) GOTO 26
C GOTO 29 IF THIS RECORD WAS DELETED
NL=NL+1
C NEED TO MAKE A NEW RECORD
REC=NL
RECST(REC)=REC
C ADD 3 MORE IF BANNED
29 CALL PUT(KBUF, SCODE, KD)
CALL GLABEL(KBUF, KSPAR(KL))
KSPAR(I)=NL
C UPDATE NL
CALL PLABEL(KBUF, KSPAR(KBUF(KL+1)))
C
C GET MEB OUTPUT DATA AND TRANSFER TO KROM DATA
C
C GET OFS FROM MEB OUTPUT AND CONVERT TO TYPE
C CALL GET(MBUF, J, MD)
II=MD+215
DO 30 I=1, 98
II=II+2
30 CALL MLV(MBUF(I), OFS(I), 1)
C CALL ZIA(OM, 40)
C CALL ZIA(HM, 60)
C CALL ZIA(TM, 90)
C CALL ZIA(MFG, 10)
30 40 J=1, 30
IF (OFSTJ.EQ. 0) GOTO 40
C=GF(SJ)
MOVE NECESSARY DATA FROM NOG LABEL AND LINE 1 TO KBUF

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

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

GOTO 50 IF THIS OBJECT IS NOT TO BE ENTER

T=2=TN(N)+1
IF(T.LT.60, AND T.GT.0) GOTO 60
T=CMT
IF(T.GT.00) GOTO 80

USE NEXT AVAILABLE SPACE IN MISCELLANEOUS AREA

CMT=CMT+1
60 CONTINUE

IF(ON(T), EQ.0) GOTO 70

GOTO 70 IF FIRST SLOT FOR THIS TYPE IS EMPTY

T=T+1

70 ON(T)=N

N=N+1

80 CONTINUE

W=W-1

IF(W.LE.NO) GOTO 50

TRANSFER CHDIR. 30 BYTES IS OFFSET OF OHL ON MBUF

II=30+MD
IST=I
IEND=30
DO 100 J=1,2
K=0

DO 90 I=IST, IEND
K=K+1

NHI=NHI(I)
IF(NHI.EQ.0) GOTO 90

IM=(K-1)+30+11

CALL MVV(MBUF(IM+3), OHL(NHI), 1)
CALL MVV(MBUF(IM+3), OLEH(NHI), 1)

CSUM=CSUM+I1
LSUM=LSUM+OLEH(NHI)

CALL MVV(MBUF(IM+9), OLEH(NHI), 1)

ISUM=ISUM+OLEH(NHI)

CALL MVV(MBUF(IM+21), OAREA(NHI), 1)

ASUM=ASUM+OAREA(NHI)

CALL MVV(MBUF(IM+11), OCLL(NHI), 1)

CALL MVV(MBUF(IM+13), OCLH(NHI), 1)

CALL MVV(MBUF(IM+15), OCLH(NHI), 1)

90 CONTINUE

IF(J.EQ.2) GOTO 100

CALL GET(MBUF, 2, MD)

IST=31
IEND=NOB
II=30+MD

100 CONTINUE

CALL MVV(CSUM, KBUF(KB+123), 1)
CALL MVV(LSUM, KBUF(KB+127), 1)

CALL MVV(ISUM, KBUF(KB+129), 2)

CALL MVV(ASUM, KBUF(KB+133), 2)

IF(CODE.GT.0) GOTO 190

C CANCELED SEPERATE PROCESSING

CALL GET(MBUF, 4, IN)

CALL MVV(MBUF(IM+1), CFORT(1, 1), 400)

CALL GET(MBUF, 5, IN)
C GET THE FOURIER DATA
DO 102 I=1,8
102 OCFOUR(T.I)=CFOUR(N.I)
DO 104 I=1,7
104 OPHI(T.I)=PHI(N.I+1)
110 CONTINUE
190 CONTINUE
C WRITE RECORD ON KROMDATA
C
CALL MVW(MBUF(M+1),CFOUR(1.5),400)
CALL GET(MBUF,6,IN)
CALL MVW(MBUF(M+1),PHI(1.1),400)
CALL GET(MBUF,7,IN)
CALL MVW(MBUF(M+1),PHI(1.5),400)
DO 110 N=1,N0B
T=NN(N)
IF(T.EQ.0 OR T.GT.51) GO TO 118
C GET THE FOURIER DATA
DO 102 I=1,8
102 OCFOUR(T.I)=CFOUR(N.I)
DO 104 I=1,7
104 OPHI(T.I)=PHI(N.I+1)
110 CONTINUE
C WRITE RECORD ON KROMDATA
C
CALL MVW(MBUF(KD+143),448)
NC=0
DO 200 G=1,9
CALL OUTCON(NFG(C):GMSG(69+G*4):4)
200 NC=NC+NFG(G)
CALL OUTCON(NC:GMSG(69):4)
IF(IY(GMSG(107)) NE.32) GO TO 225
C IF OPERATOR SUPPLIED A DIAGNOSIS GO TO 250
DO 220 N=1,N0B
DO 210 G=1,18
IF(NFG(C,NC.HFG(G.H))) GO TO 220
210 CONTINUE
C NFC MATCHES THIS DIAGNOSIS
GOTO 250
220 CONTINUE
C NO MATCH
225 N=0
GOTO 250
230 CALL MVL(DID(N),GMSG(187),8)
CALL MVL(DID(N),KBUF(KB+186),8)
250 CALL OPEN(PBUF,512,2,"NOB")
CALL GET(PBUF,1,IP)
CALL OUTCON(REC,PBUF(IP+176),4)
CALL MVL('RECORD',PBUF(IP+166),6)
C STORE RECORD NUMBER IN N FOR HCOPY
IF(N.NE.0) CALL MVL(DID(N),PBUF(IP+250),8)
CALL PUT(PBUF,1,IP)
CALL CLOSE(PBUF)
C STORE DIAGNOSIS N IN PREP AND NOB OUTPUT FOR HCOPY
CALL OUTCON('** XXXXXXXXXX ON RECORD XXXX',FMSG,31)
CALL MVL(TAGREC(22):FMSG(37):107)
CALL OUTCON(REC,FMSG(29),4)
CALL MVC(GMSG(107):FMSG(327:8)
CALL TYPE(FMSG)
IF(CODE GT.0) GO TO 300
C WRITE THE BANDED RECORDS
DO 270 I=1,3
CALL PUT(KBUF,REC+1,IK)
CALL MVW(KBUFF,DATA(I+1024-1023),KBUF(IK+1),312)
270 CONTINUE
300 CONTINUE
C WRITE PATIENT REPORT RECORD
CALL AFILE(PBUF,4,'MDATA ',6,6)
CALL OPEN(PBUF,512,0,2,'FRP')
CALL GLABEL(PBUF,PSPAR,IP)
NLP=NLIP+1
CALL PLABEL(PBUF,PSPAR,PBUF(IP+1))
CALL GET(PBUF,NLP,IP)
CALL MVC(GMSG,PBUF(IP+1),128)
CALL PUT(PBUF,NLP,IP)
CALL CLOSE(PBUF)
CALL CLOSE(KBUF)
CALL CLOSE(MBUF)
CALL TYPE('8')
RETURN
END
C
MASK2 - COMBINE 2 PICTURES AND ADD A BORDER

CALLS VERSION
SUBROUTINE MASK2(BUNIT,FILPEX)
IMPLICIT INTEGER(A-Z)
REAL STATS(256)
BYTE *(6295), B(6299), C(992), LE(12), RE(12)
COMMON STATS, LE, RE
INTEGER 2, LOW(5), HIGH(5)
EQUIVALENCE (LOW, STATS), (HIGH, LOW)
INTEGER 2, SPAR(S), SPAR(B), PAR(10), KEY(3)
BYTE FILPEX(12)
DATA MAXNS/992/
DATA NKEY/3/, KEY/'H', 'N', 'K', HFLAG/0/, NFR/128/, ISIZE/100/
CALL AFIELD(A, .PIC ', 5.5)
C ASSIGN DK1,PIC(5,5) AS DEFAULT FILE FOR NOB OUTPUT
CALL OPENA(A-3072, I, 'NOB')
CALL GLABEL(A, SPAR, IA)
CALL AFIELD'B.5, .PIC ', 5.5)
C ASSIGN DK3,PIC(5,5) AS DEFAULT FILE FOR KGM
CALL RPARAMNP,FAR(10)

11111
IP=1
3 IF(IP. GT. NP) GOTO 30
DO 5 J=1, NKEY
5 CONTINUE
5 CALL TYPE( 'PARAMETER ERROR')
PAUSE
10 HFLAG=1
CALL ZIA(STATS, 512)
IP=IP+2
GOTO 2
12 ISIZE=PAR(IP+2)
IP=IP+1
GOTO 10
15 CALL AFIELD(B,PAR(IP+2), PAR(IP+3), 2, 2)
C ASSIGN A DIFFERENT KGM FILE
10 CALL OPEN(B, 3072, 1, 0, 'KGM')
CALL GLABEL(B, SPAR, IB)
NL=SPAR(1)
IF(SPAR(1), GT, NL) NL=SPAR(1)
NS=SPAR(2)+SPAR(2)+1
IF(NS, GT. MAXNS) NS=MINS
IF(SPAR(3), NE. 7) NFR=256
CALL MSUB(A, IA+1), NS, DUNIT, FILPEX)
C LABEL CALL TO MSUB
DO 100 L=1, NL
IF(L, GT, SPAR(1)) GOTO 50
CALL GET(A, L, IN)
CALL MVL(A, IN+1), C, SPAR(2)
GOTO 55
50 CALL ITC(0, C, SPAR(2))
55 CALL ITC(0, C, SPAR(2)+1)=127
CALL GET(B, L, IB)
CALL MVL(B, IB+1), C, SPAR(2)+2, SPAR(2)
GOTO 65
60 CALL ITC(0, C, SPAR(2)+2, SPAR(2))
65 CALL MSUB(B, NS)
IF(HFLAG, EQ, 1) CALL LSTAT(SPAR(2), A, IN+1), STATS, I, I
C IF HISTOGRAM ISN'T REQUESTED, CALL LSTAT FOR STATISTIC
100 CONTINUE
IF(HFLAG, EQ, 9) GOTO 200
100 CONTINUE
I=1
C CONVERT THE FREQUENCY COUNTS TO REAL NUMBERS
DO 120 I=1, NFR
STATS(I)=LOW(I)+HIGH(I)*32768.
120 II=I+2
CALL MSUB(STATS, NFR, ISIZE)
GOTO 220
200 CALL MSUB(0, 0)
CONTINUE
CALL CLOSE(A)
END
SUBROUTINE NSUB(LOC+NSO, XSIPE, CILPEX)

IMPLICIT INTEGER(A-Z)

REAL STATS(256), MAXF, HSIZE
BYTE FILPEX(12), DATA(1000)
COMMON STATS, DATA
INTEGER2 FREQ(256), SPAR(3)
EQUIVALENCE (FREQ, STATS)
INTEGER I SIZE(5)
BYTE DSRN(6258), T(2104), LMSG(20)
EQUIVALENCE (DSRN(4153), T, DSRN(6257), LMSG(3))
BYTE LAB(512), LOC(976), BUF(1000), LH(4), LOGM(4)
DATA MAXNS=976, MNNS=360
DATA SPRI(1896)
DATA 1896)
DATA MAX=4/ (HSIZE=200, L/0/ WHITE/15, STEP/10, LBW/0/)
IF(LBW.GT.9) GOTO 100
C GOTO 100. IF NOT INITIAL CALL
CALL AFILE(DSRN, ISIZEIFILPEX, 282)
IF(ISWS.EQ.1) BIFSIZ=3872
C IF TAPE OUTPUT, DECREASE SIZE OF DSRN TO MAKE ROOM
CALL OPEN(DSRN, BUFSIZ-1, 'MSK')
CALL MVW(LOC, LAB, 256)
CALL LTI2(LAB(?), BPE)
C SAVE LABEL AND BPE
NS NSO
TF(NS.GT.MAXNS) NS=MAXNS
NS2=(NS+1)/2
C NSD2 IS THE NUMBER OF WORDS TO MOVE TO DATA
IF(NS.LT. MNNS) NS=MNNS
C MINNS IS THE MINIMUM SIZE FOR THE DATE AND TIME LABEL AND HSTGRMN
NSW=NS+24
NWW=(NSW+1)/2
SPAR(2)=NWW
SPAR(4)=512/NWW
IF(SPAR(4).EQ.0) SPAR(4)=1
CALL LABEL(DSRN, SPAR, LAB)
IF(ISW5.NE.1) GOTO 38
CALL MVW(1, T(21), 1)
CALL MVW=52140, T(32), 1)
C SET MV FOR MTO:
CALL OPEN(1, 1024, 1, 3, 'TAP')
CALL MVW(1, T(51), 1)
C BPP=1
CALL MVW(HSW, T(45), 1)
RECLE=NSW
CALL MVW(HSW, T(47), 1)
C BLOCKSIZE = NSW.
50 CONTINUE
CALL ZIA(BUF, NWW)
CALL WLINE(DSRN, LBW, HSW, BUF, 8)
C WRITE A BLANK LINE.
GMID=13+NB/2
GW=NS-32
C GREY WEDGE WIDTH
BW = GMIN-16+GW-1
D2=32+GW+3
CALL ITL(255, BUF(81), 82)
CALL WLINE(DSRN, LBW, HSW, BUF, 81)
CALL WEDGE(WHITE, STEP, BUF) NS)
CALL WLINE(DSRN, LBW, HSW, BUF, 12)
CALL UFND(TRANS, -STEP, HSW, NS)
CALL WLINE(DSRN, LBW, HSW, BUF, 12)
CALL ITL(255, BUF(81), 82)
CALL WLINE(DSRN, LBW, HSW, BUF, 81)
C WRITE GREY SCALE
CALL ZIA(BUF, HSW)
CALL WLINE(DSRN, LBW, HSW, BUF, 1)
DO 80 K=0, 6
IF(K.EQ.3) CALL REF(5, BUF, HSW)
DO 70 N=100, NS, 100
CALL OUTCON(N, LH(4), 4)
70 CALL TEXT(LH(2), L.K, BUF(H+5), 1)
80 CALL WLINE(DSRN, LBW, HSW, BUF, 1)
CALL REF(25, BUF, HSW)
CALL WRITE_TOP_REFERENCE_MARKS
CALL ITLA(255, BUF(5), 8)
CALL ITLA(255, BUF(13 + HS), 8)
CALL WRITE(DSRH, LBW, NSW, BUF, 81)
RETURN
100 CONTINUE
IF(NS0.LE.0) GOTO 200
C IF END OF PICTURE, GOTO 200
IF(BP.NE.7) GOTO 110
CALL WRITE(NS0, DATA(13), HS2)
GOTO 110
110 CALL WRITE(NS0, DATA(13), HS2)
115 L = L + 1
C STEP INPUT LINE NUMBER
CALL ITLA(L, DATA(1), 12)
CALL ITLA(L, DATA(13 + HS), 12)
C ZERO LINE REFERENCE MARKS
LP3 = LP3 + 3
IF(LP3.GT.100) GOTO 130
C TIME FOR NUMBER OF HUNDREDS
CALL OUTCON(LP3, LH(4), 4)
CALL TEXT(LH(2), 1, ML, DATA(17 + NS), 1)
GOTO 140
130 CONTINUE
IF(MOD(LP3, 5).NE.0) GOTO 150
CALL ITLA(255, DATA(5), 4)
CALL ITLA(255, DATA(17 + HS), 4)
C INSERT EVERY 5 MARK
140 CONTINUE
IF(MOD(LP3, 25).NE.0) GOTO 150
CALL ITLA(255, DATA(9), 4)
CALL ITLA(255, DATA(13 + HS), 4)
150 CONTINUE
CALL WRITE(DSRH, LBW, NSW, DATA, 1)
C WRITE ONE LINE OF DATA
RETURN
200 CONTINUE
C FINAL ENTRY WRITE BOTTOM OF PICTURE
DO 240 K = 1, 11
IF(K.LE.5) CALL REF(25, BUF, NSW)
IF(K.GT.5.AND.K.LE.8) CALL REF(5, BUF, NSW)
IF(K.EQ.9) CALL ZIA(BUF, NSW)
IF(K.EQ.6) GOTO 228
C FINISH UP NUMBER THAT WAS STARTED ON LEFT AND RIGHT SIDE
ML = ML + 1
CALL TEXT(LH(2), 1, ML, BUF(2) - 1)
CALL TEXT(LH(2), 1, ML, BUF(17 + HS), 1)
220 CONTINUE
IF(K.LT.5) GOTO 240
ML = ML * 10000
CALL OUTCON(ML, LOGN(4), 4)
230 CALL TEXT(LOGN(2), 1, K - 5, BUF(H + 9), 1)
240 CALL WRITE(DSRH, LBW, NSW, BUF, 1)
C WRITE BOTTOM REFERENCE MARKS
CALL ZIA(BUF, NSW)
CALL WRITE(DSRH, LBW, NSW, BUF, 1)
CALL ITLA(255, BUF(81), B2)
CALL WRITE(DSRH, LBW, NSW, BUF, 81)
CALL WRITE(255, -STEP, BUF, HS)
CALL WRITE(DSRH, LBW, NSW, BUF, 12)
CALL WRITE(WHITE, STEP, BUF, HS)
CALL WRITE(DSRH, LBW, NSW, BUF, 12)
CALL ITLA(255, BUF(81), B2)
CALL WRITE(DSRH, LBW, NSW, BUF, 81)
C WRITE GREY SCALE
MAXCH = (NS + 11) / 12
JMAX = 43
DO 250 J = 17, JMAX - 1
IF(J.LT.(J - 1).NE.67) GOTO 260
C FINISHED WITH LABELS, IF NO C IN COLUMN 72
J = J + 1
C SKIP THE FIRST LABEL CHARACTER WHICH IS ALWAYS BLANK
C FIND THE LENGTH OF THIS LABEL
30 242 I=1,78
IF(LAB(J+78-I).NE.32) GOTO 243
242 CONTINUE
C ENTE LABEL IS BLANK
GOTO 250
243 LABELN=78-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+MAXCHR-I).EO.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=LABELN-CHR
IF(LABELN.GT.0) GOTO 244
250 CONTINUE
C SKIP HISTOGRAM IF NSO EQ 8
IF(NSO.EQ.8) GOTO 420
C SKIP HISTOGRAM IF NSO EQ 8
HSIZE=ISIZE(1)
CALL ZLINE(BUF,NW)
CALL WLINE(DSRH,LDW,NS,BUF,04)
30 240 K=0,6
CALL TEXT(LAB(J),CHR,K,BUF(13),2)
CALL WLINE(DSRH,LDW,NS,BUF,02)
249 CONTINUE
C WRITE LABELS
J=1+CHR
LABELN=LABELN-CHR
IF(LABELN.GT.0) GOTO 244
250 CONTINUE
C SKIP HISTOGRAM IF NSO EQ 8
IF(NSO.EQ.8) GOTO 420
C SKIP HISTOGRAM IF NSO EQ 8
HSIZE=ISIZE(1)
CALL ZLINE(BUF,NW)
CALL WLINE(DSRH,LDW,NS,BUF,1)
HFRM1=HFR-1
DO 320 I=2,HFRM1
IF(STATS(I).LE.MAXF) GOTO 320
MAXF=STATS(I)
320 CONTINUE
IF(STATS(I).GT.MAXF) STATS(I)=MAXF
IF(STATS(HFR).GT.MAXF) STATS(HFR)=MAXF
DO 330 I=1,HFR
330 DO 330 FREQ(I)=(STATS(I)/MAXF)*HSIZE+.9999
BARW=HS/HFR
HSTART=(HS-BARW*HFR)/2+12
J=HSIZE
340 DO 350 I=1,HFR
IF(FREQ(J).NE.J) GOTO 339
350 H=H+BARW
CALL WLINE(DSRH,LDW,NS,BUF,1)
J=J-1
480 CONTINUE
C THE FOLLOWING CODE WAS COMMENTED OUT TO MAKE ROOM FOR TAPE OUTPUT
C
30 370 J=0.64,56
C CALL ZLINE(BUF,NW)
C HSTEP=J+BARW
C HMAX=HSTART+HFR+BARW+1
C DO 360 H=HSTART,HMAX,HSTEP
C 360 CALL ITLA(255,BUF(H),BARW)
C CALL WLINE(DSRH,LDW,NS,BUF,5)
C 370 CONTINUE
C CALL ZLINE(BUF,NW)
C CALL WLINE(DSRH,LDW,NS,BUF,2)
C HSTEP=16*BARW
C DO 400 K=0,6
C CALL ZLINE(BUF,NW)
C CALL TEXT(10,1,BUF(H-2),1)
C DO 390 H=H+HSTEP
C IF(BRM.EQ.1.AND.MOD(H,32).EQ.16) GOTO 390
C CALL OUTCON(H,LH(4),4)
C       IF(N.GT.99) GOTO 300
C       COMMON/C1/ HOB, IBI
C       COMMON/C1/ CHDIR, SMLBUF, LRGBUF, NS, HL, CUORL
C       COMMON/C1/ SST, ET
C       COMMON/C1/ SPID, SPLTH, SPAREA
C       COMMON/C1/ HCHR
C       COMMON/C1/ CFOUR, PHI
C       COMMON/C1/ BAND, AHIS
C       COMMON/C1/ OBUF, BUF
C       BYTE LRGBUF(75, 8)
C       BYTE OBUF(4152)
C       BYTE TEMPB
C       LOGICAL OPT, OPT2, OPT3, SEL(68)
C       INTEGER TEMPI
C       INTEGER SPAR(5), OFS(72)
C       INTEGER BNX
C       INTEGER BAND(168)
C       INTEGER KPAR(200)
C       INTEGER APL
C       INTEGER CHDIR(15, 68)
C       INTEGER CUORL
C       INTEGER PAR(18)
C       INTEGER CLPARM(130)
C       REAL CFOUR(69, 8), PHI(69, 8)
C       REAL AHIS(158)
C       REAL COSINE(158), SINE(158)
C       REAL AFOUR(69, 8), BFOUR(69, 8)
C       EQUIVALENCE(XP, P0, QQ, SGN, JX2, YSTAR, AF, JDIF)
C       EQUIVALENCE(YP, P0, QQ, SGY, IY2, YEND, BF, JUY)
C       EQUIVALENCE(XTERM, P0PP, NSL, NNZ, LPG, FL, RM, JUL)
C       EQUIVALENCE(YTERM, P0PP, ML, LL, KG, HDIV, RMJ, JLL)
DATA LNIN=5, OPT1=.FALSE., OPT2=.FALSE., OPT3=.FALSE.
C LENGTH OF WINDOW FOR WAVEFORM CALCULATION ON STRAIGHT CHROMOSOMES
DATA SEL=69, TRUE.
C
PAR(1)=0
CALL RPARN(HPAR.PAR.10)
HP=1
DO 10 HP=1, HPAR.2
IF(PAR(HP).EQ. 'WA') OPT2=.TRUE.
IF(PAR(HP).EQ. 'AX') OPT3=.TRUE.
IF(PAR(HP).NE. 'QB') GOTO 10
OPT1=.TRUE.
DO 8 I=1, 69
SEL(I)=.FALSE.
10 CONTINUE
C
CALL AFILE(obuf, 'RCR', '*13', '*13')
CALL CLAGEL(obuf, SPAR.18)
IB=0
C
CALL GET(obuf, IB)
CALL MVL(obuf(IB+3), SPID, IB)
NCHR=IV(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(1,1), 2*15+33)
CALL GET(obuf, 3, IB)
CALL MVW(obuf(217+IB), OFS, 72)
IF(NOT.OPT1) GOTO 30
DO 20 I=19, 34
IFSSEL(I).EQ.0) GOTO 20
SEL(OFs(I))=.TRUE.
20 CONTINUE
C
DO 310 II=1, NCHR
IF(.NOT.SEL(II)) GOTO 310
CUROLN=CHDIR(1, II)
NL=CHDIR(2, II)
IF(NUROLN.EQ.0) NL=0
NS=CHDIR(3, II)
IF(NL.LT.1.OR.NS.LT.1) GOTO 310
NG=0
LPG=1624/NS
C
LINES=GET
C
DO 390 I=1, NL
CALL GET(obuf, CUROLN, IB)
IFS(I.EQ.1) LASTIB=IB.
CUROLN=CUROLN+1
DO 391 III=1, LPG
NG=NG+1
C
WRITE(5, 3000) (obuf(I+IB), I=1, NS)
CALL MVL(obuf(IB+1), LRGBUF(1, NG), NS)
IB=IB+NS
IF(NG.GE. NL) GOTO 311
381. CONTINUE
390 CONTINUE
311 CONTINUE
C
TEST THE ORIENTATION
C
SCAN ALONG X (ROW) DIRECTION
C
JX2=NS/2
IY2=NL/2
NH2=0
NSEGX=0
NSEGY=0
C
```
219
4,122,518
220

DO 400 J=1,NS
IF(LRGBUF(J,1Y2).NE.0) NHZ=NHZ+1
IF(LRGBUF(J,1Y2).EQ.0) NHZ=0
IF(NHZ.EQ.3) NSEGX=NSEGX+1
CONTINUE

HOW SCAN IN THE Y (COLUMN) DIRECTION

NHZ=0

DO 401 I=1,ML
IF(LRGBUF(JX2, I).NE.0) NHZ=NHZ+1
IF(LRGBUF(JX2, I).EQ.0) NHZ=0
IF(NHZ.EQ.3) NSEGY=NSEGY+1
CONTINUE

DECIDE IF IT IS PROBABLY BENT OR NOT
MAYBE

IF(NSEGX.EQ.1.AND.NSEGY.EQ.1) GO TO 410
IF(NSEGX.GT.1.AND.NSEGY.GT.1) GO TO 420
GO TO 444

CONTINUE

STRAIGHT

DO 410 I=1,ML
LASTB=0
DO 405 J=1,NS
IF(LRGBUF(J,1).NE.0) GO TO 406
LASTB=J
CONTINUE

406 CONTINUE

FIRSTB=NS+1
DO 407 J=1,NS
J=NS+1-J
IF(LRGBUF(J,1).NE.0) GO TO 408
FIRSTB=J
CONTINUE

408 CONTINUE

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

IK=LASTB+LWIN
DO 4100 J=1L,IK
IBAND=IBAND+LRGBUF(J,1)
MBAND=IBAND
C MBAND IS MAXIMUM VALUE OF LWIN SAMPLES

4150 IK=IK+1

IBAND=IBAND-LRGBUF(IL,1)+LRGBUF(IK,1)
C MOVE ONE PLACE OVER
IL=IL+1
IF(IBAND.GT.MBAND) MBAND=IBAND
IF(IK.LT.IH) GOTO 4150

.CHECK IF ALL SAMPLES PROCESSED

BAND(I)=MBAND/LWIN
GOTO 4095

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

4095 CONTINUE

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

CONTINUE

GO TO 440

FLIP IT DIAGONALLY
```
CONTINUE

BENT AND PROPERLY ORIENTED

AVE Y (I)

HLL=((HLL/3)*3)+1
HP=0
NSIGY=0

DO 431 I=1,HLL,3
DO 431 J=1,NS
IF(LRBUF(J,I).EQ.0) GO TO 431
HP=HP+1
NSIGY=NSIGY+FLOAT(I)

CONTINUE

IZERO=NSIGY/HP

AVE X (J)

NSL=((NS/3)*3)+1
HP=0

DO 422 J=1,NSL,3
DO 422 I=1,HL
IF(LRBUF(J,I).EQ.0) GO TO 422
HP=HP+1
NSIGN=NSIGN+J

CONTINUE

JZERO=NSIGN/HP

REAL A11=0.0
A21=0.0
A12=0.0
A22=0.0
A13=0.0
A31=0.0
A32=0.0
A23=0.0
S=0.0
T=0.0
A33=0.0
U=0.0

HSM2=HS-2
GO 500 J=1,HSM2,3
GO 500 I=1,HL

IF(LRBUF(J,I).EQ.0) GO TO 500
XP=J-JZERO.
YP=I-IZERO.
A23=A23+1.0
XTERM=YP
YTERM=XP
A23=A23+YTERM
U=U+XTERM
YTERM=YTERM+YP
S=S+XTERM+YP
A22=A22+XTERM
YTERM=YTERM+YP
A21=A21+XTERM
YTERM=YTERM+YP
A11=A11+XTERM

CONTINUE

500 CONTINUE

A12=A12
A12=A22
A31=A13
A23=A32
P=P+S-A22*A33+A23*(A12+U+A13*T)
PPP=U-A22*A13+A23*S-A33*T+A12
PPP = A11 + A22 + A23 + A12 + A13
PPP = A22 + A22 + A23 + A23 + A12 + A13
P = (PP - PPP) + (PPP - PPP)
Q = A23 * (S - A11 * P) - A13 * (T - A12 * P)
Q = A12 + A23 - A13 + A22
Q = Q / QB
R = (S - A11 * P - A12 * Q) / A13

MDIV = 3

DO 700, I = 1, NL
YP = I - ZERO
DXY = 2.0 * P * YP + Q
DYDX = 1.0 / DXY
XP = P * (I - ZERO) * 2 + Q * (I - ZERO) + R + JZERO
MJ = XP + 0.5
MI = YP + ZERO + 0.5
IF (COP3) LRGBUF(MJ, MI) = 1
BAND(I) = LRGBUF(MJ, MI)
MI = YP + ZERO + 0.5 - DY
MJ = XP + 1.5
IF(MI, LE, 0) GO TO 701
IF(MI, GT, NL) GO TO 781
BAND(I) = BAND(I) + LRGBUF(MJ, MI)
MI = YP + ZERO + 0.5 + DY
MJ = XP + 0.5
IF(MI, LE, 0) GO TO 701
IF(MI, GT, NL) GO TO 781
BAND(I) = BAND(I) + LRGBUF(MJ, MI)
GO TO 702
CONTINUE
701
NBIV = 2
BAND(I) = BAND(I) / NBIV
NBIV = 3
702
CONTINUE
BAND(I) = BAND(I) / NBIV
780
CONTINUE
CALCULATE LENGTH
PTZ = E * P
YSTAR = 0 / PTZ
YEND = FLOA(NL) / 2.0 - YSTAR
PYE = PYE * YEND
PPYE = PYE * PYE
EL = PPEY + SQR(PPYE + PYE + 1.0) + ALOC(PPYE + SQR(PPYE + PYE + 1.0))
CBR = 4.11 * EL / PTZ + 0.5
USE MOB LENGTH DUE TO UNUSUAL RESULTS!
CONTINUE
FOURIER COEFFICIENTS

BMAX = BAND(I)
DO 710 I = 2, NL
IF (BAND(I), GT, BMAX) BMAX = BAND(I)
DO 711 I=1, NL
AHIS(I) = (BANK(I) * 100) / BMAX

711 CONTINUE
C
RHM=0
PNL=FLOAT(NL-1)
PIL=6.2832/FNLNO
C
CALL SSWITCH(4, ISW4)
IF(ISW4. EQ. 1) CALL TIMER
C
NLHALF=(NL+1)/2
DO 5920 J=1, NLHALF
R=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
IMI=I-1
RM=FLOAT(IMI)+PIL
C
DO 5010 J=1, NL
JM=J-1
C
RMJ=RM*FLOAT(J-1)
LRMJ=JM+1
LRMJ=MOD(LRMJ, NL-1)+1
C
AF=AF+AHIS(J)*COSINE(LRMJ)
BF=BF+AHIS(J)*SINE(LRMJ)

5010 CONTINUE
C
AF=(AF+AF)/FNLNO
BF=(BF+BF)/FNLNO
C
CFOUR(NUMDB. I)+SORT(AF*AF+BF*BF)
PHI(NUMDB. I)+ATAN2(BF, AF)
C
5050 CONTINUE
C
IF(ISW4. EQ. 1) CALL TIMER
C
IF (.NOT. OPT2) GO TO 810
DO 811 I=1, NL
JUV=(AHIS(I) < 1.0) = AHISM.CH< 2.0)
MAX WIDTH OF WAVEFORM IS 20 SAMPLES
JDF=HS-JUV
JLL=JDF+2/I
JUL=HS-JDF/2
DO 812 J=1, JLL
LRBUF(J, I)=0
DO 813 J=JLL, JUL
LRBUF(J, I)=127
DO 814 J=JUL, HS
LRBUF(J, I)=0
811 CONTINUE
810 CONTINUE
C
IF (.NOT. OPT3, AND, .NOT. OPT2) GO TO 310
C
CURREN=CHDIR(I, II)
NG=0
LPX=1024/WS
C
DO 889 I=1, NL
CALL GET(OBUF, CURREN, IB)
DO 831 III=1, LPG
NG=NG+1
CALL MUL((1,CHRBF(1,NS),obuf(IB+1),NS))
IB=IB+NS
IF (MC.GE.ML) GO TO 842.
831 CONTINUE
842 CONTINUE
CALL PUT(obuf,CHRF(1
CHRF=CHRF+1
IF (MC.GE.ML) GO TO 841
888 CONTINUE
841 CONTINUE
C 310 CONTINUE
C OUTPUT FOURIER COEF'S TO FILE RCR
C CALL PUT(obuf,4,IB)
CALL MUL(CDUR(1,1,obuf(IB+1),400)
CALL PUT(obuf,5,IB)
CALL MUL(CDUR(1,5,obuf(IB+1),400)
CALL PUT(obuf,6,IB)
CALL MUL(PIH(1,1),obuf(IB+1),400)
CALL PUT(obuf,7,IB)
CALL MUL(PIH(1,5),obuf(IB+1),400)
C CALL CLOSE(obuf)
C CREATE PARAMETERS FOR PHASE 14 (CLAS)
C DO 4000 I=1,NCHR
CLPARM(9+2*I)=CDHR(4,1)
CLPARM(10+2*I)=CDHR(8,1)
IF (CDHR(1,1).EQ.0) CLPARM(9+2*I)=0
IF (CDHR(1,1).EQ.0) CLPARM(10+2*I)=0
C IF BLNO IS ZERO, THE PHASE IS REJECTED BY MOD
C 4000 CONTINUE
C SET UP PART OF CHROMOSOME DIRECTORY FOR NEXT PHASE
C CLPARM(1)=NCHR
C POSITIONS 2 THRU 10 MAY BE REGARDED AS SPARE
C CALL UPARM(139,CLPARM,15)
C C IF SUB IS UP, CALL KYTE INSTEAD OF FOUR
C IF (OPT2 OR OPT3) CALL APHAS(10)
CALL EXIT
END
SUBROUTINE FOUR
COMMON /CI/ CHDR, SM1BF, LSRBF, MS, NL, CRHOF
COMMON /CI/ SP10D, SPLIT, SPARE
COMMON /CI/ NCHR
COMMON /CI/ CDUR, PHI
COMMON /CI/ GBUF, BUF
INTEGER JV(50), 1, JTAB(190), JTAB(190)
BYTE JMSC(127)
BYTE OBUF(2124)
BYTE C(3640)
REAL LTIR(60), CTAR(60)
INTEGER START(25), HIR(25), NHC(25)
INTEGER LOFS(91)
INTEGER DEF(10), KPAR(92)
INTEGER CLPARM(139)
INTEGER CPAR(5)
INTEGER SPAR(5)
INTEGER CUSCH(5)
INTEGER PPAR(5)
REAL CDUR(60,18), PHI(60,18)
INTEGER NCROM(24)
INTEGER CLASS(60)
BYTE MSC(30)
REAL DIST(18)
REAL VECTOR(18)
REAL LIXL(60,24)
REAL NNU(10,24)  
REAL SIGMA(10,24)  
REAL NBUF(10,24)  
REAL SBUF(10,24)  

C EQUIVALENCE (NNU(1,1), C185))
EQUIVALENCE (SBUF(1,1), C185))
EQUIVALENCE (SBUF(1,1), SIGMA(1,1))
EQUIVALENCE (NBUF(1,1), NU(1,1))
EQUIVALENCE (KPAR(3), OFS(1))

C DATA VMAS/16/, VMAS/12/, VMAS/12/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/, VMAS/5/
C SW7 UP IS FOR A FULL FOURIER CLASSIFIER

IS1=1
IS2=90
JCl=1
JCJ=24
NEXTPH=9
GOTO 820

800 CONTINUE
CALL MV(LOFS.OFS.90)
CALL ZIA(LOFS(ISI),32-IS1)

C FOUR WILL CLASSIFY THE C GROUP STARTING WITH ISI
CALL ZIA(OFS(13),4)
CALL ZIA(OFS(37),6)
CALL ZIA(OFS(55),4)
CALL ZIA(OFS(63),4)

C KFIX WILL CLASSIFY X.B.D.E. AND C

C
820 IF(NCHR.LE.46) GOTO 830
C IF MORE THAN 46, ALLOW FOR Z.X.AND. 2 Y
MNC(24)=2

830 CONTINUE
C
C GET MEANS AND SIGMAS
C
CALL AFILE(C,4,'CBATA /",*6)
CALL OPEN(C,3504.0.0,'CBAT')
CALL GLOB(C,CLPAR.IC)

C GET(C.I.IC)
C
C
C ITOTL=0
NRELCH=NCHR.
DO 899 I=1,NCHR.
IF(CLPARM(9+2*I).EQ.0) NRELCH=NRELCH-1
ITOTL=ITOTL+CLPARM(9+2*I)

899 CONTINUE
FTOTL=46*ITOTL/NRELCH
DO 4000 I=1,NCHR.
LTAB(I)=(CLPARM(9+2*I)*10000./FTOTL)
CTAB(I)=CLPARM(19+2*I)

C STORE REAL LENGTH AND CI

4000 CONTINUE
C
C DO 896 II=1,NCHR.
DO 896 JJ=1,NGROUP
LIKEX(II,JJ)=999.

896 CONTINUE
C
C CALL SSWTC(H(0,IS80).

IF(ISW0.EQ.1) CALL QPRINT('1
1 OBJ GROUP BLH. 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 II=JC1,JC2
JJ=II
IF(JJ.EQ.13.AND.ISW7.NE.1) JJ=23
C
SIG112=SIGMA(KIMEAS.JJ)
SIG222=SIGMA(K2MEAS.JJ)
IF(SIG112.LT.0.00001) SIG112=.00001
IF(SIG222.LT.0.00001) SIG222=.00001
C
DIST(I)=LTAB(IJ)-MU(KIMEAS.JJ)
DIST(2)=CTAB(IJ)-MU(K2MEAS.JJ)
VECTOR(1) = (DIST(1)/SIGMA1) * BIST(1)
IF (VECTOR(1) > VMAX1) GOTO 983
VECTOR(2) = (DIST(2)/SIGMA2) * BIST(2)
IF (VECTOR(2) > VMAX2) GOTO 983
C
FSUM = 0.0
DO 987 I = JMEAS, LMEAS
   DISTI = FOUR(I, I - 1) - MU(1, J)
   FSUM = FSUM + (DISTI * DISTI * SIGMA1(I, J))
987 CONTINUE
IF (FSUM > VMAX3) GOTO 983
PSUM = 0.0
IF (JMEAS EQ 0) GOTO 988
DO 988 I = JMEAS, LMEAS
   DISTI = PHI(I, I - 10) - MU(I, J)
   IF (DISTI > 3.1416) DISTI = 6.2832 - DISTI
   IF (DISTI < -3.1416) DISTI = -6.2832 + DISTI
   PSUM = PSUM + DISTI * SIGMA1(I, J)
988 CONTINUE
IF (FSUM > VMAX4) GOTO 983
L = JMEAS, LMEAS
IF (VECTOR(1) + VECTOR(2) + FSUM + PSUM > 0.0)
   CALL SSNCHG(0, ISW0)
   IF (ISW0 .NE. 1) GO TO 9010
   JV(1) = II
   JV(2) = JI
   JV(3) = DIST(1)
   JV(4) = DIST(2)
   JV(5) = VECTOR(1)
   JV(6) = VECTOR(2)
   JV(7) = FSUM
   JV(8) = PSUM
   JV(9) = LIKLE(I, J)
   DO 9089 I = 1, 9
9089 CALL OUTCON(JV(1), JMSG(18*1), 18)
   CALL QPRINT(JMSG, 99)
9010 CONTINUE
C
903 CONTINUE
903 CONTINUE
C
DO 952 IK = 1, NCHR
   ELMIN = VMAX4
   ISAY = 0
C
DO 950 IJ = JC1, JC2
   JJ = JI
   IF (IJ .EQ. 13 AND ISW7 .NE. 1) JJ = 23
   IF (NCHROM(JJ) .GT. NNC(JJ)) GO TO 950
C
IGNORE GROUPS THAT ARE FULL
C LOOK FOR UNCLASSIFIED CHROMOSOMES
C
DO 951 IS = IS1, IS2
   II = LOFS(IS)
   IF (II .EQ. 0) GOTO 951
   IF (CLASS(I1, II) .NE. 0) GO TO 951
   IF (LIKLE(I2, JJ).GE. ELMIN) GOTO 953
   ELMIN = LIKLE(I2, JJ)
   ISAY = II
   JJSAY = JJ
953 CONTINUE
951 CONTINUE
C
950 CONTINUE
C
IF (ISAY .EQ. 0) GO TO 968
CLASS(IISAY) = JJSAY
NCHROM(JJSAY) = NCHROM(JJSAY) + 1
OFS(START(JJSAY)) = ISAY
STHR(T(JJSAY)) = START(JJSAY) + INCR(JJSAY)
952 CONTINUE
C
960 CONTINUE
C LOOK FOR UNCLASSIFIED AND SEE IF MOVES CAN BE MADE
C ORDER UP TO 100 LIKELIHOODS IN ITAB AND JTAB
C
DO 300 IK=1,188
ELMIN=VMAX5
DO 250 IS=1, IS2
II=LOFS(IS)
IF(II.EQ.0) GOTO 250
IF(CLASS(I).NE.0) GOTO 250
DO 240 JJ=JC1.JC2
JJ=JJ
IF(JJ.EQ.13.AND.ISW7.NE.1) JJ=23
IF(INC(JJ).EQ.0) GOTO 240
IF(LIKE(JJ).GE.ELMIN) GOTO 240
ELMIN=LIKE(JJ)
II=SAV=II
JJ=JJ
CONTINUE
DO 250 CONTINUE
IF(ELMIN.EQ.VMAX5) GOTO 320
II=SAV=I
JJ=JJ
CONTINUE
IF(NO MORE LIKELY CLASSIFICATIONS GOTO 320
II=SAV=I
JJ=JJ
CONTINUE
IF(II.EQ.0) GOTO 320
DO 500 K=1, IK
II=II=ITAB(K)
J=JJ=JTAB(K)
IF(CLASS(J).NE.0) GOTO 500
C SEE IF WE CAN MOVE SOMEONE OUT OF GROUP J INTO A LIGHT GROUP
JSI=START(JJ)=HCHROM(JJ)+INC(JJ)
JS2=START(JJ)-INC(JJ)
IF(JS1.GT.JS2) CALL SWAP(JS1,JS2)
ELMIN=VMAX5
DO 450 JS=JS1, JS2
I=OFS(JS)
DO 440 JJ=JC1.JC2
JJ=JJ
IF(JJ.EQ.13.AND.ISW7.NE.1) JJ=23
IF(INC(JJ).GE.MINC(JJ)) GOTO 440
IF(LIKE(JJ).GE.VMAX5) GOTO 440
LIKE(JJ)=LIKE(JJ)-1
DIK=LIKE(JJ)
C BLIK IS A MEASURE OF THE LIKELIHOOD THAT I BELONGS TO GROUP JJ
C AND DOESN'T BELONG TO GROUP J
IF(DLIK.GE.ELMIN) GOTO 440
ELMIN=BLIK
ISAV=I
JSAV=JS
CONTINUE
440 CONTINUE
IF(ELMIN.EQ.VMAX5) GOTO 500
C IF NO MOVE OUT OF J CAN BE MADE, GOTO 500
C OTHERWISE, MOVE OBJECT ISAV FROM SLOT JSAV TO GROUP JJSAV
C THEN, MOVE OBJECT JJ TO SLOT JSAV
CLASS(ISAV)=JSAV
MCHROM(JJSAV)=MCHROM(JJSAV)+1
OFS(JJSAV)=OFS(JJSAV)+1
START(JJSAV)=START(JJSAV)+INC(JJSAV)
C
CLASS(JJ)=J
OFS(JJSAV)=JJ
CONTINUE
500 CONTINUE
C
C CHECK FOR STILL UNCLASSIFIED AND PUT AT BOTTOM
C
DO 520 IS=1, IS2
II=LOFS(IS)
IF(II.EQ.0) GOTO 390
CONTINUE
520 CONTINUE
C
IF(CLASS(I).NE.0) GOTO 980
CLASS(I) = 25

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

980 CONTINUE
C
CALL MYL('OS ',KPAR,4)
CALL WPARM(92,KPAR,NEXTPH)
C
C
CUROLN = 7
CALL GET(obuf,CUROLN,18).
CALL MYL(CLASS(I),obuf(961+18),60)
CALL PUT(obuf,CUROLN,18)
C
CALL CLOSE(obuf)
CALL APHASE(NEXTPH)
C
IF(ISUB.EQ.1) CALL QPRINT('1')
RETURN
END

C*************
C
C KFIX
C
C*************
C
C CHROMOSOME CLASSIFIER PART 3
C SYNTACTICAL CORRECTION OF CLASSIFICATION
C
SUBROUTINE KFIX
CALL HOBIDIR
COMMON /C1/ CHDIR,SMBUF,LRBUF,NS,NL,CUROLN
COMMON /C1/ SST,EST
COMMON /C1/ SPQR,SPLTH,SPAREA
COMMON /C1/ NCHM
COMMON /C1/ CFOURKHI
COMMON /C1/ BAND,AKIS
COMMON /C1/ OBUF,obuf
BYTE LRGBUF(78,88)
BYTE OBUF(1224)
LOGICAL BGROUP(60)
LOGICAL BGROUP(60)
LOGICAL GGROUP(60)
LOGICAL EGROUP(60)
LOGICAL FGROUP(60)
LOGICAL EGROUP(60)
INTEGER ICHK(6)
INTEGER DPINT(6)
INTEGER IIK(10)
INTEGER IIK(10)
INTEGER KPAR(92)
INTEGER OFS(81)
INTEGER LGFS(98)
INTEGER IPROF(90)
INTEGER JPROF(90)
INTEGER KBDBH(6)
INTEGER JFETUR(20)
INTEGER SPAR(8)
INTEGER BMAX
INTEGER BPL
INTEGER CHDIR(15,60)
INTEGER CUROLN
INTEGER PX(18)
REAL BD(6)
REAL CD(6)
REAL RATIO(12)
REAL CFOUR(60,15),PHI(60,15)
REAL AFOUR(60,15),SFOUR(60,15)
CALL AFILE(OBUF, 'RCR', '*13.*13')
CALL OPEN(OBUF, 1024, 1, 'RCR')
CALL GLABEL(OBUF, SPAR, IB)

CALL RPARM(KPARN, KPAR, 32)

READ RESULTS OF CONVENTIONAL CLASSIFICATION AND
EXTRACT GROUP CLASSIFICATIONS

D-GROUP

CALL GET(OBUF, 3, IB)
CALL MVL(OBUF, 217+18), ILOFS(1), 182...

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

B-GROUP

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

E-GROUP

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

DO 1851 I=49,52
ILOFS=LOFS(I)
IF(ILOFS.EQ.0) GO TO 1851
EGROUP(ILOFS)=.TRUE.
1851 CONTINUE

F-GROUP

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

G-GROUP, INCLUDING Y
DO 107 I=65,70
IICCHR(I-64)=0
ILOFS=LOFS(I)
IF(ILOFS.EQ.0) GO TO 107
GROUP(ILOFS)=.TRUE.
DO 107 CONTINUE
C
GET CHROMOSOME DIRECTORY
C
CALL GET(OBUF, 1, IB)
CALL MLV(OBUF(IB+3), GIPC+12)
CHCHR=IV2(OBUF(IB+17))
CALL MLV(OBUF(IB+31), CHDIR(1, 1), 2*15+30)
CALL GET(OBUF, 2, IB)
CALL MLV(OBUF(IB+31), CHDIR(1, 31), 2*15+30)
C
JMY=0
DO 310 IX=1,6
II=IICCHR(IX)
C
IF(DGROUP(II)) GO TO 304
GO TO 310
304 CONTINUE
C
JFETUR(1)=0
C
CUROLH=CHDIR(1, II)
NL=CHDIR(2, II)
NS=CHDIR(2, III)
IF(NL.LT.1 OR NS.LT.1) GO TO 310
NG=0
LPG=1024/NS
C
DO 380 I=1, NL
CALL GET(OBUF, CUROLH, IB)
IF(I.EQ.1) LASTIB=1B
CUROLH=CUROLH+1
DO 381 III=1, LPG
NG=NG+1
CALL SSWITCH(0, JMY)
DO 330 J=1, NS
LRGBUF(J, NG)=OBUF(J+IB)
330 CONTINUE
IB=IB+NS
IF(NG.LT.1) GO TO 311
311 CONTINUE
300 CONTINUE
C
.. MAIN LOGIC
C
IF(NOT DGROUP(II)) GO TO 401
C
ML2=ML/2
LTOP1=2
LTOP2=ML2-1
LBOT1=NL2+1
LBOT2=NL-1
C
ISUMT=0
ISUMB=0
MTOP=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
MTOP=MTOP+1
CONTINUE
ITOP=ISUMT/NTOP

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

CONTINUE
IBOT=ISUMB/NBOT

MD=MH+1
RAT(INB)=FLOAT(ITOP)/FLOAT(IBOT)
DPNTY(MH)=11

CONTINUE

DO FOR B-GROUP

DO 350 IX=1,4
II=16CHR(IX)
IF(II.EQ.0) GO TO 350
CURONL=CHDIR(1,II)
HL=CHDIR(2,II)
NS=CHDIR(3,II)
IBC=CHDIR(8,II)
M=0
LPC=1024/NS

DO 1800 I=1,HL
CALL GET(LOBF,CURONL,IB)
CURONL=CURONL+1
DO 1801 I=1,LPC
MH=MH+1
DO 1830 J=1,NS
LRGBUF(J,MH)=OBUF(J+IB)
1830 CONTINUE
ID=IB+NS
IF(MH.GE.HL) GO TO 1811
1800 CONTINUE
1801 CONTINUE

EXTRACT THE SO-CALLED B-BAND

ISTART=FLOAT(IBCT+NL)/100.0
ISTART=HL-ISTART
IEND=ISTART+ISTART-1
NL2=NL/2
IF(IEHD.GT.HL2) IEHD=HL2

HPB=0.
BIOB=0.0
DO 351 I=ISTART,IEND
DO 352 J=1,NS
IVAL=LRGBUF(J,I)
IF(IVAL.EQ.0) GO TO 352
DIX=BIOD*FLOAT(IVAL)
HPB=HPB+1
352 CONTINUE
351 CONTINUE
350 CONTINUE

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(IOBJ.LT.1) GO TO 361
DENMAX=BD(J)
JS=J
CONTINUE
OFS((12+1)*IOBJ)
BD(JS)=0.0
CONTINUE

DO FOR F-GROUP
DO 450 IX=1,4
II=IFCHR(IX)
IF(II.EQ.0) GOTO 450
CUROLN=CHDIR(I,II)
NL=CHDIR(2,II)
NG=9
LPG=1024/NS
DO 2388 J=1,NL
CALL GET(obuf,CUROLN,IB)
CUROLN=CUROLN+1
DO 2301 J=1,LPG
NG=NG+1
DO 2330 J=1,NS
LGBUF(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=LGBUF(J,I)
IF(IVAL.EQ.0) GO TO 452
G10D=I10D+FLOCIT(I1VAL)
MPG=MPG+1
CONTINUE

CONTINUE
GB(IX)=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(IOBJ.LT.1) GO TO 461
DENMAX=GB(J)
JS=J
CONTINUE
OFS((59+1)*IOBJ)
GB(JS)=0.0
CONTINUE

DO FOR G-GROUP
DO 510 IX=1,4
II=IFCHR(IX)
IF(II.EQ.0) GOTO 510
C
CUROLN=CHDIR(1, II).
NL=CHDIR(2, II).
HS=CHDIR(3, II).
NC=0.
LPC=1824/HS.

C DO 1300 I=1, NL.
CALL GIT(CBUF, CUROLN, I).
CUROLN=CUROLN+1.
DO 1301 I=1, LPC.
NC:NC=1.
DO 1330 J=1, NS.
LRGBUF(J, NC)+=OBUF(J+IB).
1330 CONTINUE.
IB=IB+NS.
IF(NC, CE, HL) 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, HS.
IVAL=LRGBUF(L, K).
IF(Ival, EQ, 0) GO TO 501.
IPROF(K)=IPROF(K+IVAL).
NP:NP=1.
591 CONTINUE.
IPROF(K)=IPROF(K)/NP.
500 CONTINUE.

C ANALYZE THE BANDS IN THE C-GROUP.
C DO 511 K=1, NL.
511 JPROF(K)=0.

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

C NSAME=-9.
DO 515 K=KBAND+NLM2.
IF(JPROF(K), NE, MAX) GO TO 516.
NSAME:NSAME+1.
515 CONTINUE.
516 CONTINUE.
KBAND(K)=KBAND+NSAME/2.

C
CLESSIFY THE C-GROUP.
C DO 521 I=1, 4.
MAXBP=0.
DO 522 J=1, 4.
IF(KBAND(J)<, LE, MAXBP) GO TO 522.
1061 ICCHR(J).
IF(I0BJ, LT, 1) GO TO 522.
MAXBP=KBAND(J).
JS=1.
CONTINUE

OFS(44-1)=10BJ
KGBAND(JS)=0

CONTINUE

CLASSIFY. THE. D. GROU.

DO 601 I=1,6
RMAX=0.0
DO 602 J=1,6
IF(RATIO(J).LE.RMAX) GO TO 602
IBJ=BDPOIN(T(J))
IF(IBJ.LT.1) GO TO 602
RMAX=RATIO(J)
JS=J

CONTINUE

JS=JS(J)
BCHR=BCHR(J)+1
RbT(IO(J))=4.8

CONTINUE

DO 6W,J4,4,6

CALL WPARAN(92, KPAR, 9)
CALL CLOSE(OBUF)
CALL APHASE(9)

CALL EXIT
END

**********

**********

**********

ROUTE TO ASSESS NORMALITY OF CHROMOSOMES IN A SPREAD

SUBROUTINE ABNORM

COMMON /C1/ NOB, IDIR
COMMON /C1/ CHDIR, SMOBUF, LRGBUF, MS, ML, CROLM
COMMON /C1/ SST, EST
COMMON /C1/ SPF, SPITH, SPAREA
COMMON /C1/ MCHR
COMMON /C1/ CFOUR, PHI
COMMON /C1/ BAND, AHIS
COMMON /C1/ OBUF, BUUF
BYTE LRGBUF(50,90)
BYTE OBUF(2124)
BYTE TEMP0

LOGICAL OPT1, OPT2, OPT3
INTEGER IPROF(90)
INTEGER UFETUR(20)
INTEGER TEMPI
INTEGER SPARE(5)
INTEGER BMX
INTEGER BAND(180)
INTEGER BPL
INTEGER CHDIR(15,60)
INTEGER CROLM

INTEGER *2, SST(76), EST(76)
INTEGER PAR(10)
REAL CFOUR(60,15), PHI(60,15)
REAL AHIS(180)
REAL COSINE(150), SINE(150)
REAL AFOUR(60,15), BFOUR(60,15)

EQUIVALENCE(SPAR(2), BPL)
EQUIVALENCE (CFOUR(1,1), AFOUR(1,1))
EQUIVALENCE (PHI(1,1), BFOUR(1,1))
EQUIVALENCE (NL, NL0)
EQUIVALENCE (HUMOB, II)
DATA LOPT1//"TI"/
DATA LOPT2//"T2"
DATA LOPT3//"T3"/
DATA LOP //"OP"/

CONTINUE

CALL ARFILE(OBUF, 1, 'RCE', '*13', '*13')
CALL OPEN(OBUF, 1024, 1, 0, 'RCA')
CALL GLABEL(OBUF, SPAR, IB)

CALL GET(OBUF, 1, IB)
CALL MYL/OEU(IB+3), SPIOB, 12)
    NCHR=IV(OBUF(IB+17))
    CALL MYL(OBUF(IB+31), CHDIR(1,1), 2*15+30)
    CALL GET(OBUF, 2, IB)
    CALL MYL(OBUF(IB+31), CHDIR(1,31), 2*15+30)

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

333 CONTINUE

    LASTIB=IB
    CUROLN=CHDIR(1,II)
    NL=CHDIR(2,II)
    NS=CHDIR(3,II)

    IF(NL.LT.1 .OR. NS.LT.1) GO TO 310
    NG=0
    LPG=1024/NS

    DO 330 I=1, NL
        CALL GET(OBUF, CUROLN, IB)
        IF(I.EQ.1) LASTIB=IB
        CUROLN=CUROLN+1
        DO 301 III=1, LPG
            NG=NG+1
            CALL SSWITCH(0, JAY)
            IF(JAY.EQ.1) WRITE(5, 3000) (OBUF(J+IB), J=1, NS)

302 CONTINUE
    DO 330 J=1, NS
        LRGBUF(J, NG)=OBUF(J+IB)

330 CONTINUE

    IB=IB+NS
    IF(NC.< GE. NL) GO TO 311

301 CONTINUE

311 CONTINUE

-----------

CONTINUE

NSM1=NS-1

MAIN LOGIC
C 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 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 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
C C C
IF(IPROF(K), GE, NS) IPROF(K)=0
C C 500 CONTINUE
C C C
IPTOT=0
NLM2=NL-2
DO 600 K=3, NLM2
IPTOT=IPTOT+IPROF(K)
600 CONTINUE
C C C
IPAVE=IPTOT/NLM2
C C DO 620 K=3, NLM2
IPROF(K)=IPROF(K)-IPAVE
IF(IPROF(K), LT, 0) HLTO=HLTO+1
620 CONTINUE
C C C
IPROF(1)=1
IPROF(2)=1
IPROF(NL)=1
IPROF(NL-1)=1
C C C
IF(JAY.EQ.1) WRITE(*, 1386) (IPROF(L), L=1, NL)
C C C C
TYPE 1 DETECTION-
AT LEAST FIVE NEG NUMBERS IN A ROW IN PROFILE
C C 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-
SEIZED, LOWEST DIP IN PROFILE IS FIRST OR LAST ONLY IN SEQUENCE OFNEG 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+4).GE.B) GO TO 710
IF(IPROF(K+3).GE.B) GO TO 710
IF(IPROF(K+2).GE.B) GO TO 710
IF(IPROF(K+1).GE.B) GO TO 710

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

SCAN TO POS NUMBER

701
K=K+4
K=K+1
IF(IPROF(K+1).LT.B) GO TO 701
GO TO 760
710 CONTINUE

NMINUS=1
DO 720 KL=1,6
K=KL+2
IF(K.GT.NL) GO TO 721
IF(IPROF(K+K).GE.B) GO TO 720
NMINUS=NMINUS+1
720 CONTINUE

721 CONTINUE

IF(NMINUS.LT.4) GO TO 700

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

731
IF(IPROF(K+1).LT.B) GO TO 731
GO TO 760

750 CONTINUE

WRITE(5.4800) (JFETUR(JF), JF=1, IFETUR)
430
FORMAT(' FEATURES: ',203)
IF (NL < GTNL*3) GO TO 940
GO TO 941

CONTINUE

WRITE(S, 1094)
FORMAT(' NOT A CHROMOSOME')
GO TO 950

CONTINUE

IF (IFETUR .EQ. 0) GO TO 988
IF (IFETUR .EQ. 1) GO TO 910
IF (IFETUR .EQ. 2) GO TO 920
IF (IFETUR .EQ. 3) GO TO 930

GO TO 940

CONTINUE
GO TO 950

CONTINUE

IABIND = IFETUR(1) + IFETUR(2)
IF (IABIND .EQ. 2) WRITE(S, 1292)
IF (IABIND .EQ. 3) WRITE(S, 1392)
IF (IABIND .EQ. 4) WRITE(S, 1492)

1292 FORMAT(' CHROMOSOME IS BICENTRIC, OR BADLY TWISTED')
1392 FORMAT(' CHROMOSOME MAY BE BICENTRIC')
1492 FORMAT(' CENTROMERE LOCATION DOUBTFUL - PROBABLY NORMAL')

CONTINUE

IF (IABIND .EQ. 1) WRITE(S, 1191)
GO TO 912
WRITE(S, 1191)

CONTINUE

FORMAT(' SINGLE CENTROMERE IS WELL DEFINED - NORMAL')
1291 FORMAT(' SINGLE CENTROMERE POORLY DEFINED - PROBABLY NORMAL')

CONTINUE

IF (NL .LT. 30) GO TO 901
WRITE(S, 1932)
GO TO 903
WRITE(S, 1901)

CONTINUE

FORMAT(' NO CENTROMERE FOUND, COMMON IN SMALL CHROMOSOMES - PROBABLY NORMAL')
1292 FORMAT(' NO CENTROMERE FOUND - POSSIBLE ACENTRIC')

CONTINUE

CONTINUE

WRITE (5, 3030)
CALL AGRAVE(13)
CALL CLOSE(0BUFF)

CONTINUE

FORMAT(118)
2003 FORMAT(3X, 3913)
3000 FORMAT('CHROMOSOME ', 12)
3030 FORMAT('1')
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 operator to view each successive chromosome spread to inspect the chromosome spread and determine whether it is acceptable for the purpose of analysis or not,
   deleting under operator control the location of each spread determined not to be acceptable thereby establishing a list by location of which the chromosome spreads are acceptable,
   again moving said stage under instructions of said programmed computer for generating a digital picture of each of said acceptable chromosome spreads,
   arranging the chromosomes in the digital picture of an acceptable chromosome spread in a karyotype format, and
   producing a visible image of the karyotype format from the classified and oriented chromosomes.
2. A system as recited in claim 1 wherein said step of producing a visible image of the karyotype format for each acceptable chromosome spread includes
   displaying said karyotype format on a gray scale display,
   correcting any errors to provide a corrected karyotype format, and
   printing said corrected karyotype format.
3. A system as recited in claim 1 wherein said step of moving said stage under said microscope to successively present said plurality of dispersed metaphase chromosome spreads to the optics of said microscope includes
   moving said stage with a scanning motion in successive predetermined motion increments underneath the optics of said microscope,
   detecting whether or not a chromosome spread is present within the distance of the predetermined increment of motion, and
   storing the location data for each detected chromosome spread.
4. A system as recited in claim 3 wherein said step of generating a digital picture of each of said accepted chromosome spreads includes,
   successively moving said stage to the undeleted locations of accepted chromosomes spreads to position each such accepted chromosome spread under the microscope optics,
   automatically focusing the optics of said microscope on an accepted chromosome spread positioned thereunder,
   photographing each said accepted chromosome spread through said focused microscope optics in a manner to generate successive picture elements, and
   digitizing each picture element generated in photographing a chromosome spread.
5. A system as recited in claim 1 wherein said step of arranging the chromosomes in the digital picture of an acceptable chromosome spread in a karyotype format includes
   locating each chromosome in said digital picture of an acceptable chromosome spread,
   orienting each chromosome in a predetermined direction,
   measuring each oriented chromosome,
   classifying the oriented chromosome on the basis of measured length and centromeric index, and
   composing the karyotype format from the classified and oriented chromosomes.
6. A system as recited in claim 5 wherein after the step of locating each chromosome in said digital picture of an acceptable spread there is included the step of assigning a different number to each chromosome which has been located.
7. A system as recited in claim 8 wherein said step of composing the karyotype format from the classified and oriented chromosomes includes
   displaying said composed karyotype format on a gray scale display, and
   correcting any chromosome orientation and classification errors.
8. A system as recited in claim 4 wherein the step of producing a visible image of the karyotype format includes
   converting the digitized picture elements into analog picture elements, and
   printing a picture responsive to said picture elements.
9. An operator-assisted system using a programmed computer for producing karyotype images from a slide mounted on the stage of a microscope, said slide having a plurality of dispersed metaphase chromosome spreads, comprising
   means for moving said stage under instructions of said programmed computer connected to control said microscope to successively present said plurality of dispersed metaphase chromosome spreads to the optics of said microscope,
   means for storing the location of each spread, thereby automatically prescanning said slide for chromosome spreads,
   means for moving said stage under instructions of said programmed computer to place the slide motionless on said stage at each successive chromosome spread location, thereby enabling an operator to view each successive chromosome spread to inspect the chromosome spread and determine whether it is acceptable for the purpose of analysis or not,
   means for determining under operator control the location of each spread determined not to be acceptable thereby establishing which of said chromosome spreads are acceptable,
   means under instructions of said programmed computer for generating a digital picture of each of said acceptable chromosome spreads,
   means for arranging the chromosomes in a digital picture of an acceptable chromosome in a karyotype format, and
   means for producing a visible image of the karyotype format.
format for each acceptable chromosome spread.
10. A system as recited in claim 9 wherein said step of producing a visible image of the karyotype format includes
   means for displaying said karyotype format on a gray scale display,
   means for correcting any errors to provide a corrected karyotype format, and
   means for printing out said corrected karyotype format.
11. A system as recited in claim 9 wherein means for moving said stage under said microscope to successively present said plurality of dispersed metaphase chromosome spreads to the optics of said microscope includes
   means for moving said stage with a scanning motion in successive predetermined motion increments underneath the optics of said microscope,
   means for detecting whether or not a chromosome spread is present within the distance of the predetermined increment of motion, and
   means for storing the location data for each detected chromosome spread.
12. A system as recited in claim 9 wherein means for generating a digital picture of each of said accepted chromosome spreads includes,
   means for successively moving said stage to the undeleted locations of accepted chromosome spreads to position each such accepted chromosome spread under the microscope optics,
   means for automatically focusing the optics of said microscope on an accepted chromosome spread positioned thereunder,
   means for photographing each said accepted chromosome spread through said focussed microscope optics in a manner to generate successive picture elements, and
   means for digitizing each picture element generated in photographing a chromosome spread.
13. A system as recited in claim 9 wherein the means for locating each chromosome in said digital picture of an acceptable spread includes
   means for assigning a different number to each chromosome which has been located.
14. A system as recited in claim 9 wherein means for producing a visible image of the karyotype format includes
   means for converting the digitized picture elements into analog picture elements, and
   means for printing a picture responsive to said picture elements.
15. A system as recited in claim 9 wherein said means for arranging the chromosomes in a digital picture of an acceptable chromosome spread in a karyotype format comprises
   computer means programmed for locating each chromosome image in an acceptable digital spread, for orienting each chromosome image in a predetermined direction, for measuring each oriented chromosome image for classifying the oriented chromosome images, and for composing the karyotype format from the classified and oriented chromosome images.