REPORT NO. DOT-TSC- FHWA-71-1

AUTOMATIC DATA REDUCTION FROM AERIAL PHOTOGRAPHS— PHASE 1 REPORT

JURIS G. RAUDSEPS
AND DAVID S. PRERAU
TRANSPORTATION SYSTEMS CENTER
55 BROADWAY
CAMBRIDGE, MA. 02142

AUGUST 1971
TECHNICAL REPORT

Prepared for
DEPARTMENT OF TRANSPORTATION
FEDERAL HIGHWAY ADMINISTRATION
WASHINGTON, D. C. 20546

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT-TSC-FHWA-71-1		

4. Title and Subtitle	5. Report Date
Automatic Data Reduction from Aerial Photographs - Phase I Report	8/15/71
	6. Performing Organization Code TCD
7. Author(s) Juris G. Raudseps, David S. Prerau	8. Performing Organization Report No.
9. Performing Organization Name and Address DOT/Transportation Systems Center 55 Broadway Cambridge, MA 02142	10. Work Unit No. R-1016
	11. Contract or Grant No. HW-05
12. Sponsoring Agency Name and Address Federal Highway Administration 800 Independence Avenue, S.W. Washington, D.C. 20546	13. Type of Report and Period Covered Technical Report 7/1/70 - 6/30/71
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Aerial photographs are useful in various studies of highway traffic behavior. From a timed sequence of aerial photographs of a fixed highway area, one can find for each vehicle crossing the area data on position, velocity, trajectory (i.e., entrancing, lane changing, and exiting) and type (i.e., car, truck, or bus). In this project, an interactive system consisting of a computer, a computer-controlled flying-spot scanner, and a graphics tablet is utilized to significantly automate the data reduction process. This report describes the current state-of-the-art of data reduction and the system being developed. The pertinent computer programs developed to date are documented in detail.

17. Key Words	18. Distribution Statement
	Availability is Unlimited. Document may be Released To the National Technical Information Service, Springfield, Virginia 22151, for Sale to the Public.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	119	

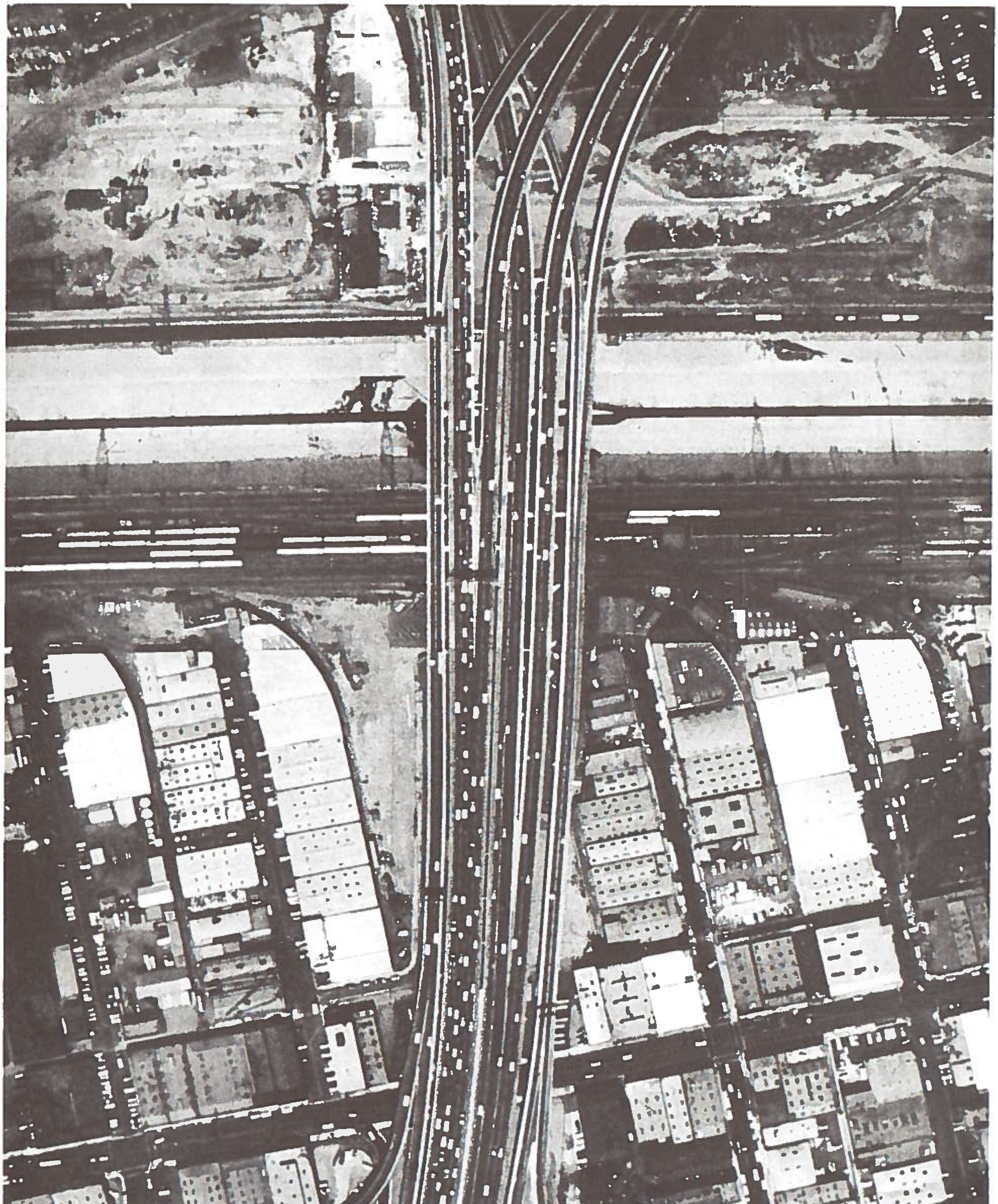


TABLE OF CONTENTS

Page

TABLE OF CONTENTS (CONT.)

Page

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (CONTINUED)

LIST OF TABLES

1.0 INTRODUCTION

This report deals with the problem of automatically extracting data about highway traffic flow from sequences of aerial photographs. The work reported is being performed at the Transportation Systems Center of the United States Department of Transportation for the Federal Highway Administration under General Work Agreement 71-HW-01 Project Plan Agreement No. HW-05.

The problem being attacked is to develop techniques to decrease the cost and time involved in reducing the traffic data contained in aerial photographs to usable form.

There is considerable interest in gathering data on traffic flow by means of aerial photography. The major reason for this is that an aerial photograph is the only practical medium for observing the location of every vehicle within the area covered at the same given instant in time. By extension, in successive frames of photography covering a given area one can observe all the changes in vehicle locations that have occurred during the interval between the instants that the photographs were taken. From these, average vehicle velocities over the interval can be calculated.

Both macroscopic and microscopic aspects of traffic flow can be observed from aerial photographs. The term "macroscopic" in this context denotes features pertaining to the flow of traffic as a whole - such things as mean velocity, density, flow rate in vehicles/hour, distribution of traffic by lanes, by vehicle types (cars, trucks), etc. "Microscopic" denotes those features defined in terms of the actions of individual vehicles or configurations of vehicles. Included among them are such things as headways between individual vehicles, lane-changing behavior, etc. Generally speaking, the people concerned with highway operation would be interested in the macroscopic aspects of flow - in such operationally meaningful quantities as the number of vehicles on a given stretch of road, the average velocity along that stretch, travel time between some pair of points, etc. The microscopic aspects of traffic are more of concern to designers, who must be able to predict the effects upon traffic flow of ramps, curves and grades, etc., and theoreticians who need data on which to base their models and with which to validate them.

Sequences of aerial photographs inherently contain a microscopic depiction of traffic behavior over the period they cover. This property makes aerial photography an essentially irreplaceable tool in those situations where the precise behavior of individual vehicles is the prime matter of interest, since the amount of instrumentation and recording equipment required to simultaneously follow precisely a large number of vehicles over an extended area is in general too formidable to consider.

It is less obvious that aerial photography is a good tool for observing the macroscopic aspects of traffic. Most of the parameters of traffic flow can be derived from readings of counters activated by the conceptual equivalents of tripwires stretched across the highway lanes (current technology favors electromagnetic induction loops). Furthermore, ground-based instrumentation, once installed, can operate largely independently of weather and lighting conditions and in some cases can be used for real-time control. In view of these advantages of ground instrumentation, it appears that aerial photography for observing gross traffic flow should be limited to areas where this would need to be done so infrequently that installing equipment on the ground would not be economical.

Whatever the merits of aerial observation in any given situation, it is an established fact that aerial photographs are being utilized quite widely for observing and studying traffic. This is so in spite of obvious and inherent problems in data collection, including the cost of flying and photographing and the limitations imposed by atmospheric affects, weather, and lighting, and in spite of the very considerable effort required to reduce the data to useable form from the photography.

Our survey of current techniques of data reduction shows that human operators do the work. The techniques appear slow, tedious, somewhat inaccurate, expensive, and limited in respect to the data ultimately available to the user. This is the "classical" description of a technique that "should be" in some way "automated" or "computerized."

Our approach is one of expanding on the work that has been done already in conjunction with manual systems. The goal is to eliminate, by stages, the tasks that must now be performed by a human operator of a film reader. We take advantage of interactive computer graphics techniques to facilitate human intervention and of the bookkeeping capability of an on-line computer to collect more complete data than a human film-reader can reasonably collect with both speed and accuracy.

This report describes the progress of the first year of a planned two-year effort. Included are a survey of current manual techniques of data reduction, a survey of film scanners, a description of our approach and our hardware, and documentation for the significant computer routines developed thus far.

2.0 STATE OF THE ART SURVEY

In this section, we shall present two state-of-the-art surveys: one on techniques presently used for data reduction of aerial photographs of highways, and the other on devices presently available for scanning film.

2.1 CURRENT TECHNIQUES OF DATA REDUCTION

At the present time, data reduction from aerial photographs of traffic is done manually. That is, the human eye and brain are the mechanism for recognizing vehicles as such, and human hands guide any measuring equipment used to define their location. The fact that a human being is doing the data reduction brings with it a number of advantages. The human is the best pattern recognizer known, being able to recognize vehicles under a wide range of conditions, adjusting automatically to changes in contrast, color, lighting and shadows, scale, orientation, etc., etc. Ideally, a sufficiently experienced, motivated, and attentive person using adequate equipment and diligence should be able to locate vehicles in a photograph more accurately and with lower likelihood of omission or false introduction than any automatic device.

It is reasonable to assume that the decision on whether or not there is a vehicle at a given location that is arrived at by a human photointerpreter devoting his full attention to that location will be the best possible, given the film input. In any practical case a human film reader will introduce errors beyond the irreducible minimum due to the limitations of the film medium itself. These errors will be of several kinds - false omissions and false introductions of vehicles due to too cursory an examination of the film, and measurement errors due to imprecise positioning of the measurement apparatus relative to the vehicle image. All these are errors in measuring the coordinates of points on the film. It is only these discrepencies between the true traffic situation and the traffic situation as finally recorded that can be treated as data reduction errors attributable to the film reader.

In comparing different film reading systems, the expected values of these errors must be matched against each other and against the errors in mapping points on the film into points on the ground.

The precise calculation of the ground coordinates corresponding to a point whose film coordinates are known is quite a complex process forming part of the science of photogrammetry.

The calculations are based on the measurements of the film coordinates of points whose ground coordinates have been precisely measured. Errors are introduced by measurement errors on the ground and the film, by round off in the calculations, by lens distortions, and by film warping. Generally speaking, all become interrelated. Ordinarily, a considerable number (~15) of ground reference points are considered per frame. As additional points are introduced into the calculation, they serve to both refine the mapping function and to validate the measurements of the other points.

There are in principle two classes of manual film reading systems now in use. The first class is used for extracting the macroscopic features of traffic flow only. A representative system of this type is that employed by the Freeway Operations Section of the California Division of Highways, concerned with monitoring freeway traffic flow in the Los Angeles area. The surveys have a limited aim - to monitor the degree of traffic congestion during weekday rush hours along selected sections of the freeway system.

The surveillance technique consists of overflying the section of freeway under consideration in a light airplane and photographing successive half-mile segments of the road with a standard 35 mm camera equipped with a large capacity film magazine. The data reduction process is rather crude - the photographs are projected on a screen and vehicles on fixed road segments are counted. The road segments considered are typically 800 feet long and are delineated by such landmarks as ramps, overpasses, etc. A surveillance flight may last some 2 1/2 hours, during which a six-mile section under study would be overflown some 25 times. The final output for consideration by the traffic engineer is a density chart showing the average number of vehicles per mile per lane as a function of time. From these, travel times can be deduced by utilizing empirically derived relationships between traffic density and average speed. Data gathered by surveillance flights are checked against speedomenter records of test cars driven along the sections under study during the time span covered by the photography. Test cars are also used on the days preceding and following the days of the flight, and the flight is repeated if the data from the test cars show that the traffic patterns on that day were atypical.

The requirements of film reading precision in this case are minimal, the only errors of significance being omissions and false introductions of vehicles. For all vehicles except those at the ends of the highway segments the assignment to the correct segment is obvious, and assignment in questionable cases is essentially arbitrary anyway. An inevitable degree of uncertainty about the car counts in some highway segments arises from the presence of overpasses that obscure parts of the road. In these cases the assumption is made that the vehicle density under the overpass is the same as the average density in its vicinity, and certainly on the average this must be true. The reading errors per se are estimated by the personnel of the California Division of Highways to vary between 2 - 10% omissions, the error rate varying with lighting conditions and increasing sharply as lighting becomes bad.

With current procedures, a typical 2 1/2 hour surveillance flight results in some 300 frames of photography, covering some 1100-1200 segments in which vehicles are to be counted. Film reading consists of counting vehicles within each segment and compiling a table of these values. That task currently takes a full man-week of effort. Density calculations and drafting to produce a finished chart take approximately again that long.

Currently, no data are being extracted regarding traffic composition by types (cars, trucks). Traffic density by lanes is recorded only in areas where this appears critical. Headway distributions are not recorded, and information concerning vehicle speed is not directly derivable from the photographs, since overlapping photographs closely spaced in time are not taken. These data are of varying degrees of interest, but are not now gathered because they would complicate the task of the human film reader. The speed data is of particular interest, but this can not be obtained without changing the entire mode of operation.

At the present time, the use of aerial photography in the inventory of freeway congestion in the Los Angeles area has been suspended because the personnel concerned are testing alternate methods of congestion inventory using lane occupancy recorders. As yet, this work has not progressed to a point where the results obtained by the alternate method can be compared to those obtained from aerial photography.

It appears to be the opinion of the personnel of the California Division of Highways that such comparison should be performed to validate the conclusions reached on the basis on ground instrument readings. Such validation would be of the nature of a calibration of the ground instrument system, probably necessary because presence detectors have a tendency to falsely introduce apparent vehicles, and their outputs depend on vehicle size, speed and perhaps spacing. Thus in addition to the initial comparison with aerial data necessary for calibration at the time a system of sensors is installed, subsequent such comparison for recalibration might be necessary if traffic patterns should over the course of time deviate significantly from those occurring at the time of system installation. Such variations in traffic patterns may be expected (in fact, hoped for) if various control measures, such as ramp metering, are installed.

In addition, the people at the California Division of Highways feel there will be occasion to fly aerial congestion survey flights even after ground instrumentation is installed and calibrated for the sections of freeway now surveyed. Currently, traffic survey flights are made only during morning and evening rush hours during midweek. Weekend congestion is not inventoried because the congestion patterns tend to vary with the time of year, weather, and various special events (e.g. fairs), so that it does not appear a good investment of resources to gather congestion inventories now. Such data would nevertheless be of considerable interest, particularly for weekends on which congestion is caused by various annual special events. In these, analysis of one year's problems might be used to improve flow the following year.

The improvements in data reduction techniques for macroscopic analysis of traffic that appear to be most desired are primarily a reduction in cost and in time. The additional data that are mentioned above as being of interest are not now gathered because of the clerical burden that obtaining them would impose on the human film readers. Were the film to be read by a computer-controlled system, these data would come essentially free, since a computer, unliked the human brain, is an excellent device for performing bookkeeping functions.

The details of the data gathering and data reduction process involved in observing the macroscopic behavior of traffic by using aerial photography are those pertaining to the work of the Freeway Operations Section, District 7, California Division of Highways. Very similar work has also been done in the San Francisco area and in St. Louis, Missouri, by the Missouri Highway Department, and apparently also in Houston, Texas. It was the strongly expressed opinion of the California highway engineers that continuous congestion inventory of the type that they were engaged in compiling would become recognized as a necessary program for essentially every highway department concerned with freeway system operation. It was their feeling that the availability of a central service bureau capable of returning fully reduced operations data within a turnaround time of approximately one week would be a welcome service for the potential users.

A number of organizations are engaged in the microscopic analysis of traffic from aerial photographs. It does not appear to be germane to a discussion of scanning techniques to discuss in detail the goals of their research except as these affect the accuracy requirements of the scanning process.

The System Development Corporation is engaged in a study of traffic flow through a diamond interchange. The interchange is photographed from a helicopter hovering at an altitude of some 2100'. A Maurer 70 mm camera is used, with a Zeiss Biogon 72° lens. This gives ground coverage of about 4000' on the diagonal, of which about 3000' are considered in the study. It is considered necessary to photograph an area extending some 500' on each side beyond the area of interest to assure full coverage, since the helicopter may experience difficulty in maintaining station and attitude. Photographs are taken at 1-second intervals. Faster repetition rates are not considered practical largely because of expected mechanical problems of wear, etc.

The model of a diamond interchange used by SDC involves dividing the interchange into functional blocks, such as ramps, approaches to ramps, segments between off and on ramps, etc. The ground reference points of interest are, therefore, the boundaries of these functional blocks. They are marked on the ground by 8' x 1' strips placed on the shoulders of the roadway.

The matters of interest to the researchers are the time each vehicle spends traversing each block, its speed in crossing block boundaries, and que lengths as a function of traffic signal states. The traffic signal state at any time can be seen in the photographs from special signal lights pointed vertically that have been installed for the purposes of the experiment.

The film is read by operators operating a Benson-Lehner Telereadex device. This is essentially a table on which the film is projected at either 10X or 20X magnification (the operators use 20X for this task). The table is equipped with a pair of crossed wires, one horizontal and one vertical, which the operator can move by twisting a handle with each hand. When she depresses a foot switch, a digital encoding of the positions of the wires at that instant is punched into IBM cards by a standard IBM card punch connected to the film reader. At the same time a card sequence number, advanced automatically, a film frame number, advanced automatically when the film is advanced, and various other codes, set by hand on a number of dials, are also punched. The reader is equipped with an elaborate film advance mechanism allowing foreward and backward advancing of the film as well as rotation of the projected image - convenient in that it is generally used to align the roadway with one of the cross-wires, so that only one wire need be moved when measuring the locations of successive cars in a line.

An experienced operator can read car locations quite rapidly. Even so, it takes roughly 20 hours to read the 60 frames (each with some 500 cars) representing 1 minute of traffic. Reading inaccuracies are of two kinds - missed vehicles and positioning errors. No good statistics on the fraction of vehicles missed appeared to be available. SDC would like to keep it under 1%, but they have signed a contract with UCLA to perform film reading for them with a provision that only 95% of the vehicles in any one frame need be read.

The position errors are due to incorrect placement of the cross-wires relative to the vehicle image. This can come about for a number of reasons - fuzzy film, parallax effects, and simply the unsteadiness of the operator's hand. These errors are apparently under 3' in substantially all cases, and tend to be around ±1'. Errors in computed velocity tend to range around 1 1/2 m.p.h. The errors in position are not considered critical for most purposes of the study. However, during one phase of the work it was desired to obtain the trajectories of vehicles through the intersection in detail. It was discovered that this was impossible with data of the accuracy available, since slow vehicles appeared to move with an extremely syncopated rhythm, occasionally jumping backwards. To overcome these problems, the photographs were taken from an altitude of only 500 ft. (as opposed to 2000'), with the consequent enlargement in scale and improved reading accuracy.

The Institute of Transportation and Traffic Engineering at UCLA is engaged in film scanning both for their own research projects and on contract to SDC. They too use a Benson-Lehner Telereadex device, but in their case it is connected not to a card-punch but directly to an IBM 1800 computer. When the operator depresses the foot-switch that otherwise would activate the card punch, the computer is interrupted and the newest set of coordinates is read in. The computer immediately performs a validity check on the new data - i.e. it compares the newly read values with the projected location of the vehicle they probably represent, as obtained by extrapolating its trajectory computed from the previous frames. If the new value appears implausible, the computer types a message on a teletype next to the scanner operator to read the car locations in that vicinity again.

The advantages expected of this system when it becomes fully operational are severalfold. First is the minor advantage that data input to the computer is faster. More important are the following: Without the immediate interaction with the computer, operator blunders are not detected until the film data is analyzed. By then the film is no longer in place and can not be accurately repostitioned. Thus, if errors are in a frame are to be corrected, essentially the whole frame must be rescanned. In the interactive mode, only small areas need to be rescanned. Next, the problem of confusing vehicles is reduced. In tracing vehicle trajectories frame by frame, there is some danger that two vehicles may be confused (for instance, a car actually passing another may be recorded as having fallen in behind the other). In the interactive mode, the computer may ask for remeasurement when a clear-cut decision appears difficult. It may be hoped that the second set of measurements may give a less ambiguous set of vehicle locations. Finally, the computer may improve operator performance. One might expect the operator to become more careful if the computer asked for frequent rescanning, and alternatively to work faster, if less carefully, if the computer seemed satisfied with her performance.

The UCLA research project for which the data are being utilized is a study of the influence of highway exit ramps on lane changing behavior. The measurements must detect lane changes and their context - i.e. distance from the ramp, traffic density, etc. 70 mm photography covering 1 mile of road is being used. The vehicle locations are being obtained within ±3 ft., which appears adequate for the purpose of the study. The researchers feel that the 1 mile section being considered is probably too short - they might like as much as 5 miles. Such data are out of reach, however, more because of the inability of a helicopter to hover at sufficient altitude than because of film reading problems that might be encountered trying to cope with photography of that scale.

The UCLA system computes ground coordinates from film coordinates on the basis of observed ground marks. 10 to 15 such marks appear in a frame. They are marked on the ground as crosses of 1 1/2' x 16' marker strips.

Some error analyses on the scanner data have been performed. 3 frames of photography were scanned 9 times each. Repeatability of measurements was within ±1', but actual ground accuracy is estimated to be only ±3'. A UCLA student has also performed as a classroom project a study of reading errors of the omission-false introduction class. Taking his own very carefully obtained results as the standard, he found that the regular operators had performed as follows: 2064 vehicles detected correctly, 74 vehicles falsely introduced, 141 vehicles omitted.

As yet, the UCLA scanning procedures do not record vehicle types (cars or trucks), apparently because of the burden this would place on the operator, although it is planned to do so in the future. No provision exists or is planned for entering into the computer other characteristics of the vehicles, such as color, even though all the film used is color film and even though such information would be of obvious use in constructing vehicle trajectories, particularly in ambigous cases.

The UCLA workers would consider automatic entry of vehicle type and color information an operationally useful improvement on their system. However, the improvement that they would seek most is in speed. Their rate of data reduction is 1 frame every 15 or 20 minutes. This is roughly the same as the rate cited by SDC, although SDC engineers quoted 500 vehicles/ frame as a typical number, as opposed to 150-200 vehicles/ frame quoted as typical for the UCLA photography. It may well be that the smaller scale of the UCLA photographs has the effect of slowing down their scanner operators.

Work at the Ohio State University has involved study of traffic features of the most microscopic kind - the formation and dissipation of platoons of vehicles within a traffic stream. Photographs would be taken while flying above the platoon and staying with it. Some 2500' of road would be photographed on a 70 mm frame. Greater care than usual seems to have been taken in reading the film (e.g. - film placed between glass plates to prevent warping) and greater accuracies have been achieved. Errors in position of as little as 6" have been claimed, and velocity errors of less than one m.p.h. These results have been achieved at considerable cost in time. It appears that the films have been read with an analytical stereoplotter - an accurate device, but a slow one, since the average reading rate seems to have been 20 sec/point. Such performance appears beyond the resolution power of any but the ultra-precise and very slow automatic scanners.

The applications of aerial photography to traffic analysis that have been mentioned above appear to span the range of accuracy requirements. The list is not exhaustive, but probably sufficiently representative to show how widely automatic scanning techniques might be applied. It would appear that a computer-controlled flying-spot scanner system could be implemented to perform all the tasks described above except the last with accuracy comparable to that of the manual systems. The hardware capabilities of flying-spot scanners are described in some detail in the next section.

The application of the marvels of this hardware can now be done with the benefits of the published results of some 15 years of widely spread work in pattern recognition. This research was originally spurred by USAF interest in automatic target detection, but has since borne fruit in applications to automatic reading of printed text, interpretation of biomedical imagery, etc.

2.2 FILM SCANNERS A SURVEY

This section is principally concerned with the capabilities of devices known for historical reasons somewhat inaccurately as flying-spot scanners. A computer-controlled flying-spot scanner appears to be the only type of device suitable for automatic processing of aerial photographs of traffic. Other candidate devices are mentioned briefly only to indicate the reasons why they are considered unsuitable for this task.

2.2.1 FLYING-SPOT SCANNERS

A computer-controlled flying-spot scanner is in concept a fairly simple device. Its principal parts are, in addition to the computer, a cathode ray tube (CRT) with its associated electronics, optics, a film holder, and a light measuring device, normally a photo-multiplier tube (PMT). The system operates as follows: The computer outputs to the scanner the x and y coordinates of a point on the film to be read. The digital signals of the computer are converted by D/A converters and suitable amplifiers to electrical signals which deflect the electron beam of the cathode ray tube to the appropriate point on the tube face, creating there a point of light. This light is gathered by a lens and focused onto a corresponding point on the film. Depending on the density of the film at this point, a certain fraction is transmitted through the film. This is collected by another lens and directed at the photomultiplier tube, which produces a proportional output current. This current is measured, the measurement converted to digital form by an A/D converter, and read by the computer as input. The computer can then output the next set of x,y coordinates to read the film density at some other point. In particular, the computer may use the density values obtained at the set of points previously read to calculate the next point to examine. This is an important capability, since it may in certain cases greatly reduce the total number of points that need be read for the purpose at hand.

The outline of operation given above omitted a number of practical considerations that impose various limits on the operation of the device and restrict its realizable accuracy and speed. The discussion of these limitations that follows deals only with the case when both high accuracy and high speed are sought simultaneously.

The accuracy of a scanner is a concept that can not be characterized by any one number. Various deviations from ideal behavior occur, with different effects depending on circumstances. First, the resolution of the device is limited.

It appears that currently the "best" practice is to use 5-inch diameter CRT's. The "quality circle" on the face of such a tube, i.e. the area over which the scanning point can be accurately positioned and focused, is about 3-inches across. The best achievable spot size - i.e. the diameter of the light spot on the tube face measured between half-power points (a non-trivial thing to do) - is on the order of .0005" - .001". This means that a grid of 4000 x 4000 reasonably independent points is about the maximum that a scanner of this type can read. Scanners can be built with more addressable points on the grid (for instance, Information International, Inc. has built scanners with 16,000 x 16,000 addressable points), but the adjacent addressable points largely overlap. Additional degradation in resolution comes about because of the optics and even because of scattering in the film being read itself. Resolution of optical systems is conventionally expressed in terms of the modulation transfer function (MTF) of the system. The MTF is defined as the gain of the system as a function of spatial frequency (cycles/mm), normalized to one at frequency zero. Information International, Inc. claims that the as yet unpublished results of a survey conducted for NASA, Huntsville, of flying-spot scanners shows theirs to be the best. The performance curve of their device shows MTF of .95 at 30 cycles/mm, .50 at 50 cycles/mm, and .15 at 100 cycles/mm. Whether or not this is in fact the world's best scanner, these performance figures do represent the upper limit on achievable resolution. It should be appreciated that such performance is achieved by employing considerable sophistication. Dynamic focusing is used to focus the electron beam in the CRT (i.e. the magnetic focusing field for the beam is adjusted to compensate for changes in the length of the electron path when the beam is moved off axis). The objective lenses are specially designed to be color corrected for the particular phosphor used in the tube and to compensate for the light bending effects of the glass face-plate of the CRT (which is about 1/2 inch thick, for mechanical stability).

The positional accuracy of the spot is also limited. Despite "geometry correction" circuitry, a mathematically perfect rectangular grid of points is not achievable. Some "pincushion" or "barrel" distortion remains, and in addition the x and y axes may not be perfectly perpendicular to each other. These systematic effects can be kept well under 1%, however, and to the extent that they are still significant, can be lumped with the various other distortions that appear in the mapping from ground coordinates to perceived film coordinates and removed by the same set of calculations.

Somewhat more difficult to cope with is the problem of hysteresis. The demands of precision in beam deflection and focusing are such that they can not be met with electrostatic deflection plates and focus devices, but that deflection and focusing must be done by applied magnetic fields. Even though the magnetic core materials for the coils are chosen to have low residual magnetism and narrow hysteresis loops, hysteresis effects can not be avoided completely. Thus, the precise position to which a spot is deflected depends upon the previous scanning pattern as well as upon its nominal coordinates. The maximum possible positioning error due to hysteresis can apparently not be kept under 1 part in 1000 by design of the hardware (i.e. about 4 spot diameters). If care is taken in programming, however, its effects can probably be effectively eliminated. To this end, any given area of the film must always be approached from the same general direction.

Second-order effects limiting performance seem to abound with any precision equipment. With magnetically deflected scanners, the effects of heating also fall in this class. Since the deflection coils carry substantial currents (several amperes) at large deflections off axis and small currents when the beam is along the axis, they can undergo mechanical deformations due to changes in temperature. These change the effective magnetic fields and therefore the location of the scanning spot. No quantitative estimates for this effect are available. Careful design probably can reduce it. In any case, it is true of both this and the hysteresis effect that while they may affect the absolute accuracy of a measurement and its repeatability, they would not ordinarily affect significantly the measurements of relative placement of objects close together in the film.

In addition to a certain randomness in the positioning of the scanning spot, there is also some randomness inherent in the measurement of the film transmissivity at that spot. The transmissivity of film is the fraction of incident light that passes through it. Since there is considerable variability in the amount of light emitted by the CRT phosphor as a result of a perfectly constant electron beam bombardment ($\pm$ 10%, randomly distributed over the tube face, and changing as the tube ages), it is necessary in precision scanners to measure both the incident and the transmitted light and compare them to get the true transmissivity. To this end, a beam splitter is inserted in the light path between the CRT and the film. A fixed fraction (~ 10%) of the light emanating from the CRT is measured by a reference photo-multiplier tube (PMT). The rest is directed through the film and the transmitted part is measured by the main measuring PMT. The quotient of the two readings is the transmissivity of the film. The logarithm of this is the quantity known as the density. Frequently the measurement made is that of density because the human eye seems to perceive logarithmically equi-spaced steps in light as "equal steps of brightness", and because a better dynamic range can be obtained.

The values read as the intensity of light at either PMT are random variables in the true statistical sense of the term. At the low light levels that obtain with flying-spot scanners, the fact that light is a stream of discrete photons arriving at random intervals assumes practical signficance. One can no longer think in terms of the instantaneous value of light, but must think instead of an average value over some interval. From statistical theory it follows that the variance between the observed average and the "true average" value will decrease as the period over which the average is computed is increased.

The precise interval needed to achieve a certain mean square error varies with among other things the beam power of the CRT, the phosphor efficiency, the aperture and focal length of the objective lens , and the efficiency of the PMT. In practice it appears that an integration interval on the order of 5 μsec is necessary to get reasonably reliable gray scale resolution to about 32 levels. This is approximately the gray scale resolution capability of the human eye at normal ambient light levels.

The speed capability of the flying spot scanner can be defined in terms of the length of the interval between the instants at which the gray scale levels of successive points can be read by the computer. During this interval the computer must perform certain housekeeping chores: store the last light value read, obtain new value for the coordinates, output them, and read the new value of the gray level. The scanner must deflect the beam to the specified location, allow an appropriate interval for the integration of the light, and provide for the analog to digital conversion. Some of the functions of the computer and the scanner can overlap in time. For reading film in a raster scan mode with x incremented by fixed amounts and y constant, a DDP-516 computer (a machine falling in the upper part of the class referred to as mini-computers) requires a total of 14 memory access cycles of 1 μsec each per point. Within this cycle the interval between the output of the latest x-coordinate and the reading of the gray level is 9 μsec. This is adequate time for the scanner to perform its functions, allowing 2 μsec for beam deflection, 5 μsec for light integration, and 2 μsec for A/D conversion.

Thus, the data transfer rate to the computer in this mode is roughly 7×10^5 values/sec. This compares favorably with the highest speed of data input to the DDP-516 via magnetic tape. The highest-speed tape drive available for it operates at 80 i.p.s. with 800 b.p.i. tape, giving a character transfer rate of 6.4×10^5 characters (6 bits)/ sec - about 10% slower than the scanner.

The calculation above assumed that increments between adjacent spots were small. When the deflections between successive spots are large, substantially longer times than 2 μsec must be allowed for the spot position to stabilize. The increased settling time is due to larger oscillations in the deflection currents in response to a larger step input, and even more fundamentally due to oscillations within the magnetic material of the cores of the deflection yokes. The practitioners of the art of yoke design apparently can not fully eliminate this poorly understood phenomenon. In some cases, such as during retrace in raster scanning, the imposed delay may be as much as 200 μsec. In most cases, the actual time loss should not be significant, because the large changes in deflection would tend to occur at times at which the computer would perform somewhat lengthy data manipulations in any case.

Flying-spot scanners can read color film. Scanners designed to do this must use a CRT with a "white" phosphor (generally P24) and some system of colored filters. Generally, the design is like that of the TRIM scanner at TSC - a wheel with colored filters mounted in it is placed in the light path between the CRT and the film. In operation, a picture is read in turn with each of the three color filters in place. The filters must be moved mechanically whenever a different color is to be read. This arrangement has several advantages: First, only 2 PMT's (reference and reading) with their associated electronics are needed. More important, the device can also be used for output in color. If unexposed color film is inserted in place of the transparency, the color filters between the film and the CRT allow the film to be exposed to the three different colors in turn to produce a color picture. On the other hand, scanners of this configuration become impractically slow because of the required mechanical movement of the filters if one should want to obtain a full-color representation of an individual small area before proceeding to scan the next small area, again in full color. This, unfortunately, is precisely the scanning mode useful for scanning traffic imagery.

An alternate configuration is possible for flying-spot scanners which circumvents this problem. This is the following: A CRT with white phosphor is used. No filters are placed between the CRT and the film (Thus color output is ruled out.). The separation of light into component colors is performed upon the transmitted light by a set of dichroic mirrors. A dichroic mirror is essentially a beam splitter which reflects light in a certain spectral range and transmits the rest. Two different mirrors can be used to deflect the transmitted light in two spectral ranges to off-axis PMT's. A third PMT can be placed on axis to measure the remaining light. The operation of such a system then involves the reading by the computer of three different values representing the three different colors after each positioning of a scanning spot. Assuming the same equipment speeds and scanning mode as used before in the calculation of the reading cycle time, we find that to input and store 3-color data would require 22 μ-sec per point, as opposed to 14 μ-sec for only one transmissivity value per point.

2.2.2 OTHER TYPES OF SCANNERS

A number of various other devices can be used to scan film. Among them are the following:

1. TV cameras: These employ camera tubes such as vidicons, image orthicons, etc. All these have photosensitive surfaces on which the image to be scanned is projected. Electric charges gradually build up on such a surface as it is continuously illuminated. These charges are drained off point by point and the currents measured to obtain a measure of illumination at the point. Since the charge read off is a function of both light level and length of time since it was last read, random scanning under computer control is not possible, but instead an image must be read in a regular raster scan mode at a fixed rate. This is not a desirable mode of operation for the problem at hand.

2. Image disector cameras: An image disector tube has a photo-emissive front surface onto which the image to be examined is projected. The emitted electrons are drawn by an electrostatic field backward in essentially parallel paths, such that the current density at any cross-section of the beam is proportional to the distribution of illumination on the tube face. The back of the tube is in concept a conductive plate with a small hole in it. Most of the electrons in the beam hit the plate. Those corresponding to one small area on the photo-emissive pass through the hole. There they are collected and the resulting current is measured. The whole beam emanating from the front surface is deflected so as to cause the current from different spots on the tube face to hit the hole in the back. The deflection circuitry is of the same type as that in CRT's, and the needs to integrate the photo-emitted currents to get meaningful average values are the same as in the case of flying-spot scanners, so that operating speeds can be roughly the same for both devices. The disadvantage of image disectors versus flying-spot scanners

seems to be that currently it is possible to obtain resolution several times better with a flying-spot scanner than with an image disector, and that with image disectors there appears to be no way to avoid using mechanically moved color filters to obtain color information about the input image. Image disectors use light from an external source and can, therefore, analyze opaque images (rather than transparencies) as well as real-world scenes. Unlike flying-spot scanners, they can not be used as film output devices.

3. Mechanical scanners: Various mechanical scanners are in use. These all use mechanical motion of either the image or a scanning head to position the scanning spot. Some light source is focused on the spot to be read and the light is then measured by a photomultiplier. The range of such equipment goes from facsimile news photo scanning equipment to micro-densitometers. The equipment can be made extremely precise in every way and can at least theoretically read images of any size. Again, it does not appear suitable for the purpose under consideration here because it is too slow for random accessing of points under computer control, and line-by-line raster scans are not a desired computer input.

4. Laser scanners: Various scanners using a laser as the primary light source have been proposed, and quite probably some have been built. They promise very high accuracy, but as yet it would appear that no satisfactory method of random point accessing is available.

3.0 AUTOMATIC DATA REDUCTION—SYSTEM OPERATION

It follows from the numbers quoted for the various presently operating data-reduction systems that to reduce the data for one hour of traffic requires approximately half a man year of film reading effort. Clearly this is bound to hinder any research effort dependent upon the data, and to make it prohibitively expensive. The solution we are seeking is to automate the film reading process.

We feel that the error bounds now achieved in measuring vehicle locations are within the capabilities of an automated system of the type we propose. The most significant errors from the point of view of the users of the data are those where vehicle trajectories are incorrectly joined. Because an automated film-reading system will be able to compare descriptive data of each vehicle detected in a frame with that from the previous frame, we believe that the rate of errors in trajectory matching will be lower in an automated system than in a manual one.

As an ideal solution, one might seek a system which would perform the following functions automatically:

1. Perform image enhancement and other transformations required to suppress background and noise.

2. Impose a suitably rectified reference grid on each picture frame.

3. Delineate the boundaries of highways, bridges, interchanges, intersections and other areas of interest.

4. Locate each vehicle, record its coordinates, and make measurements to be used for identification.

5. Apply automatic pattern recognition techniques to these sets of measurements to classify each vehicle as a car, bus, etc.

6. From the data on two successive frames compute the displacement and instantaneous velocity of each vehicle or group.

7. From data on several successive frames compute and verify vehicle trajectories, and calculate such traffic characteristics as vehicle separation, passing frequency, distribution of vehicle lane changes, and location of bottlenecks.

8. Generate suitable displays and printouts.

In specifying a system to actually implement, we must set our sights a little lower if the system is to be practical.

It is clear that one can not devise an automatic system that accepts pictures of an unidentified area containing no annotation, identifies the landmarks in the picture and scans those subareas that are of interest to the user. Since computers that intuitively sense intent are beyond the current state of the art, any automated system must be provided with a convenient means for operator entry of reference points and limits of the areas to be scanned.

Our approach is to use interactive computer graphics techniques to achieve these purposes, and our system configuration is designed to permit this. Our overall system consists of a computer controlling a flying-spot scanner and a CRT display, and receiving inputs from the flying-spot scanner and a graphics tablet (Figure 1).

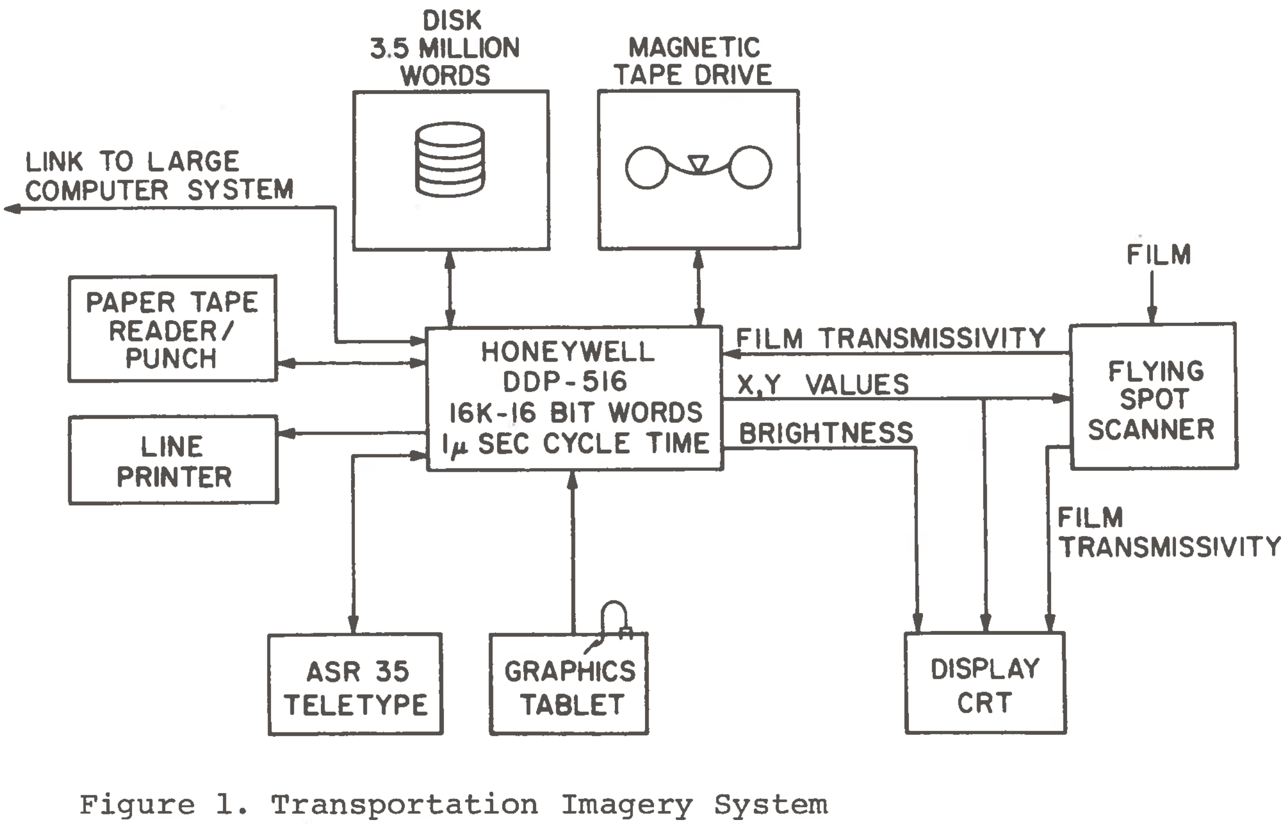


Figure 1. Transportation Imagery System

We seek improvements over the current systems by shifting onto the computer the largely mechanical tasks. We feel that it might be effort badly spent to attempt to shift onto the computer those tasks to which it is ill adapted and which a man can do easily. It appears that the best film data reduction system should involve man and machine in a symbiotic relationship. The division of labor should be such that the man makes the decisions requiring judgement and/or the ability to reason, whereas the computer performs all the routine data handling.

In the context of the problem at hand, this general philosophy seems to imply the following specific division of labor:

The human should decide what areas of a film frame should be scanned - i.e. locate and delineate for the computer the roads of interest. It would be quite substantial problem to program a computer to recognize a road as such against a background not known a priori. The landmarks on the film serving as ground reference points should also be pointed out to the computer by a human. The human can readily compare a map of the area that has been photographed with the photograph and identify the locations of the references. A computer program with such capability would be difficult to develop and would in general involve an enormous amount of calculation.

Several techniques could be employed to transfer such positional information to the computer. The simplest in the sense of requiring the least equipment would be to place appropriate markings directly on the film. It appears to be relatively easy for a computer to locate large opaque markings on the film. This approach has drawbacks. Marking the film is a somewhat tedious process and difficult to do with both precision and speed. Furthermore, the approach implies hands-off operation once the film is in the scanner. This is undesirable, since any error by the human in marking the film or any misinterpretation of the markings by the computer would go undetected, or, if detected, could not be conveniently corrected.

Our system is designed to allow input of road delineations and landmarks directly to the computer in a manner allowing immediate verification. It is implemented by a display on-line with the computer and a graphics tablet. We display the picture scanned at rather coarse resolution on the CRT display and superimpose on the display a marker corresponding to the location of the stylus on the graphics tablet. Routines are provided for selecting small sub areas of the photograph being scanned for display in greatly enlarged format at high resolution, and for selecting reference points within these high resolution sections.

The following steps are envisioned in the operation of the completed system in scanning a set of successive photographs of substantially the same area:

1. The first frame is displayed in coarse resolution. The operator causes the neighborhood of each ground reference to be displayed enlarged and at full resolution and marks the landmark precisely. The ground coordinates of the landmark are supplied to the computer via teletype or other convenient means.

2. The operator then traces the shoulders of any roads in the picture to be scanned. The computer scans in the immediate area of the indicated shoulder, locates the discontinuities in the film presumed to be the shoulder and computes and stores their ground coordinates.

3. The operator then marks the vehicles in the frame. The computer finds the objects marked, and checks that they satisfy appropriate sets of criteria regarding size, etc. before identifying them as vehicles and separating them into such classes as cars and trucks. The computer records each found vehicle's location and size, and its actual digitized representation, to be used in finding and identifying the same vehicle in subsequent frames.

   In future development, the computer may be able to perform these steps automatically by scanning along the road searching for vehicles.

4. On subsequent frames, the computer itself tries to locate the ground reference marks by searching in the general areas of the film where they were in the previous frame. It requests aid from the operator only if it fails.

5. With the location of the reference marks known, the computer finds the road to be scanned by applying coordinate transformation algorithms.

6. The computer then extrapolates the path of each vehicle found in the previous frame, and scans the road near the expected new location of the vehicle, (if this location is still within the picture) searching for the vehicle.

To extrapolate vehicle paths, we intend to use the algorithms already developed for trajectory matching in conjunction with the manual film reading systems. The positions of vehicles in a frame are predicted on the basis of their positions in the previous frame and their velocities (computed from their observed motion between the previous two frames). The velocity values are validated by comparison with the average traffic velocity which is deteriined from known empirical relationships between traffic density and speed.

The computer starts searching for a vehicle at the position which has been extrapolated for that vehicle. It locates possible vehicles by detecting deviations from the normal background color of the pavement. Digital spatial filtering techniques and nonlinear gray scale enhancement are used to "improve" the digitized images. When a possible vehicle is found, template matching techniques are used to determine if the vehicle being searched for has been found. As the template in this case, we use the digitized image in the previous frame of the vehicle sought. In matching, we allow for distortions due to turning of the vehicles, but this effect is generally small. If there is not a good match or if the vehicle physically could not have reached the position of the found vehicle, then the computer resumes its search until the vehicle is found. The capability to match vehicles by appearance from frame to frame should assist in trajectory matching in those cases where the present system finds this difficult on the basis of vehicle positions alone.

7. The operator marks all new (i.e. incoming) vehicles in the frame in a manner similar to step 3. After several frames, when the system has been able to deduce what the bare road looks like, the computer should be able to scan the road, locating the incoming vehicles automatically.

8. Finally, data of interest is abstracted and output in a form directly useful to the investigator. It is probable that in some cases routines can be developed that achieve savings in the amount of scanning that needs to be done by determining that some data is of no interest in the given application. For instance, one current study considers only data from periods of steady flow. The computer should be able to evaluate from observing the rate at which vehicles enter the area photographed whether such steady flow exists and

scan only those photographs in which it does. Similarly, one can envision that for studies evaluating the behavior of trucks, the computer would select for scanning only those sections of road in the vicinity of the trucks, etc.

A set of flow charts for the system is shown in the following pages. Figure 2 shows the overall system and Figures 3 to 10 show each of the parts of the system in greater detail. (On the flow charts, solid lines indicate flow and broken lines indicate subroutine calls.)

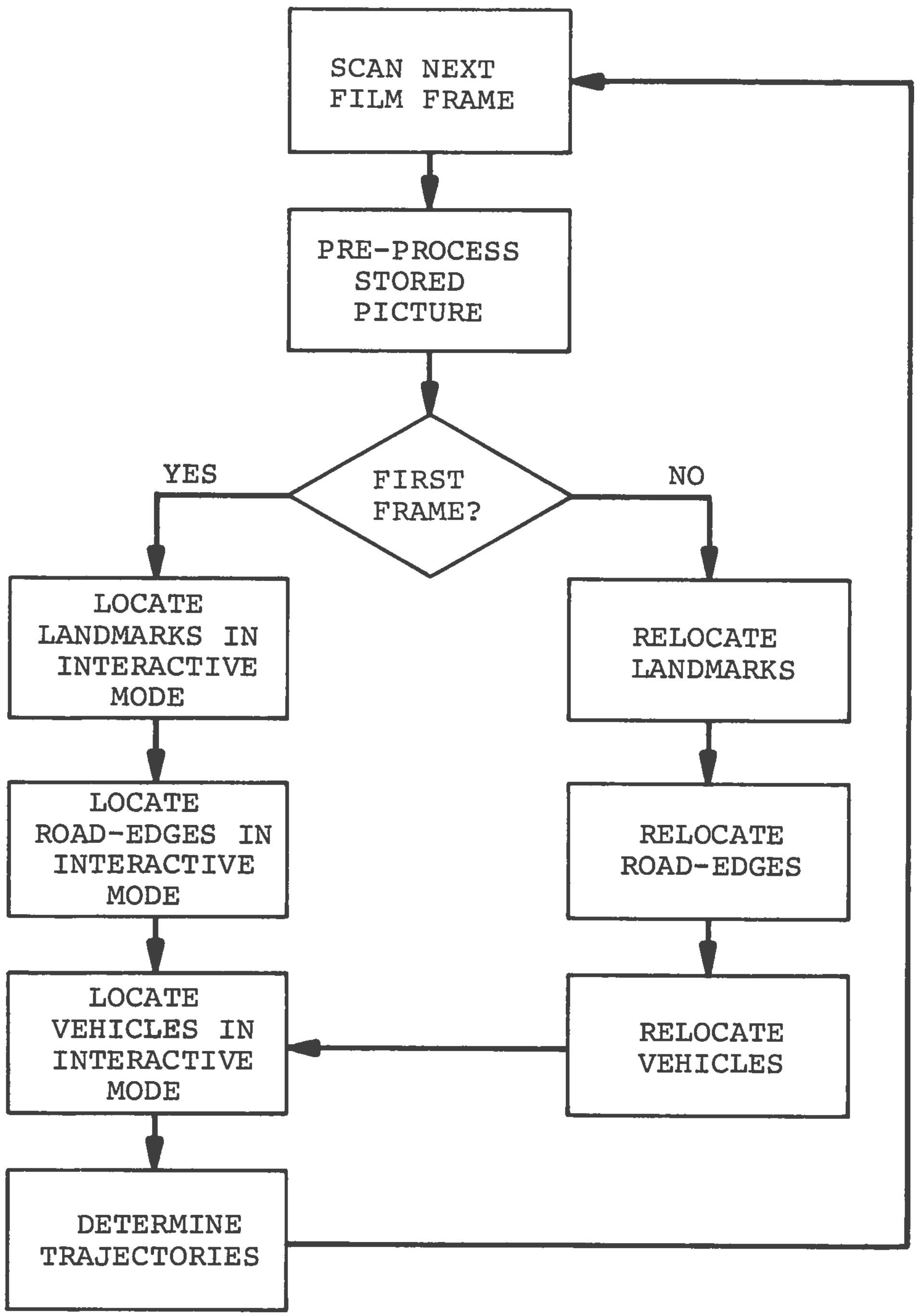


Figure 2. The Overall System

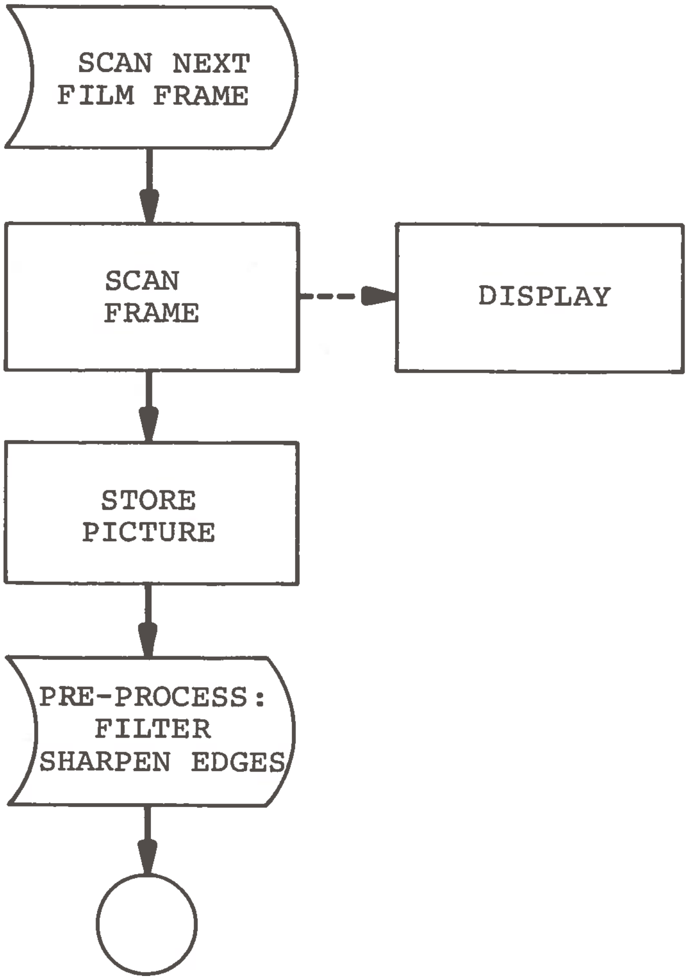


Figure 3. Scan Next Film Frame and Preprocess

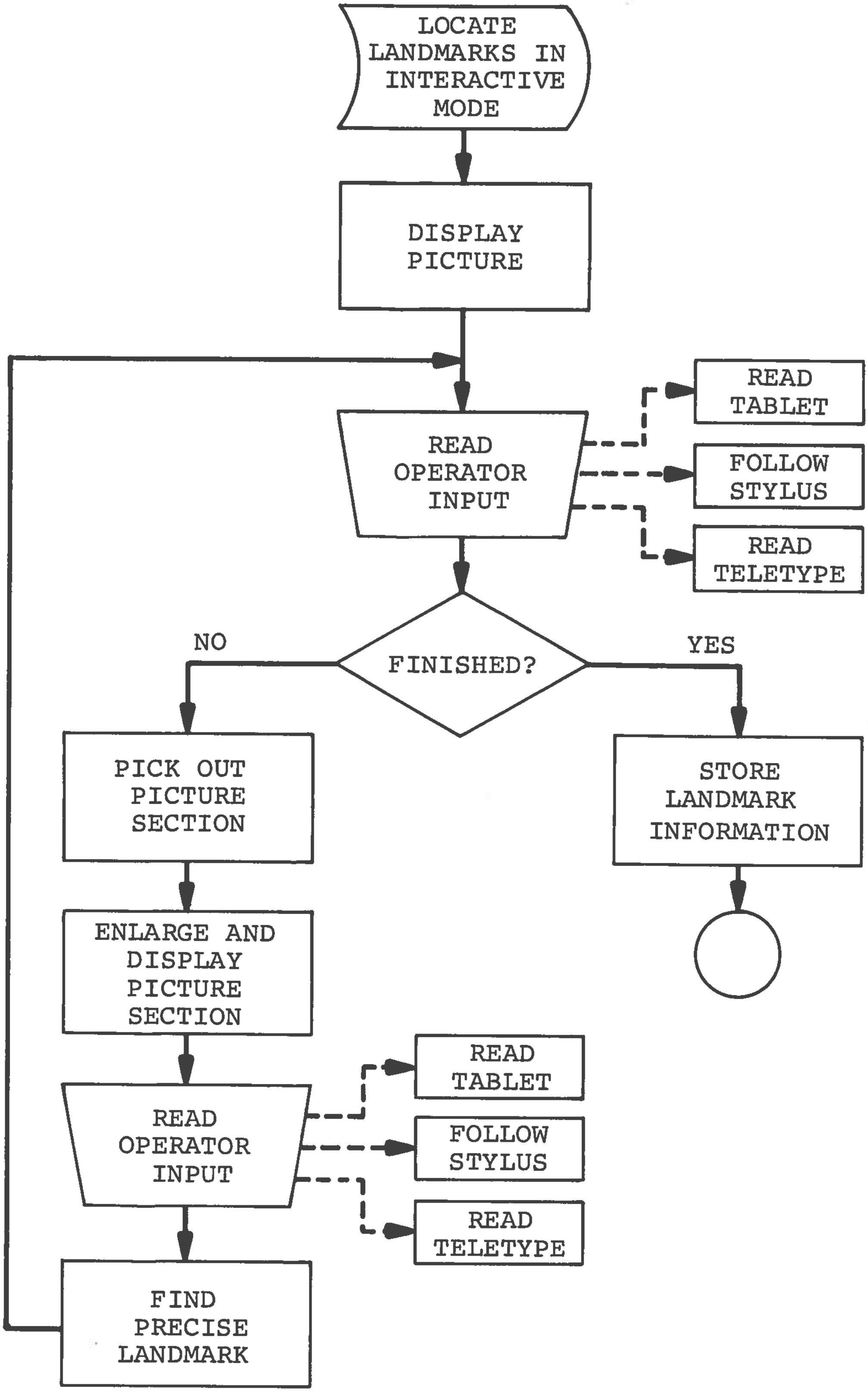


Figure 4. Locate Landmarks in Interactive Mode

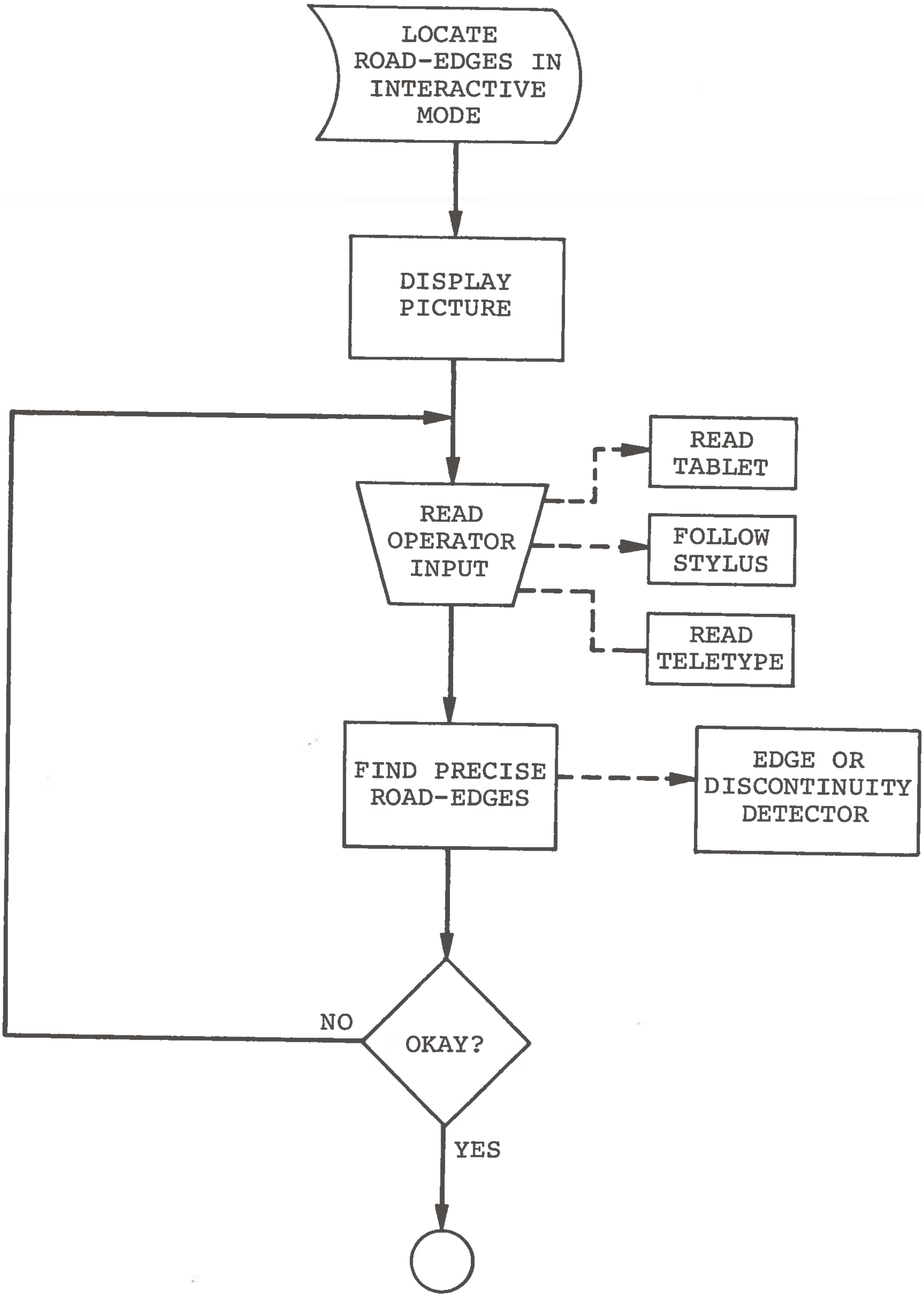


Figure 5. Locate Road-Edges in Interactive Mode

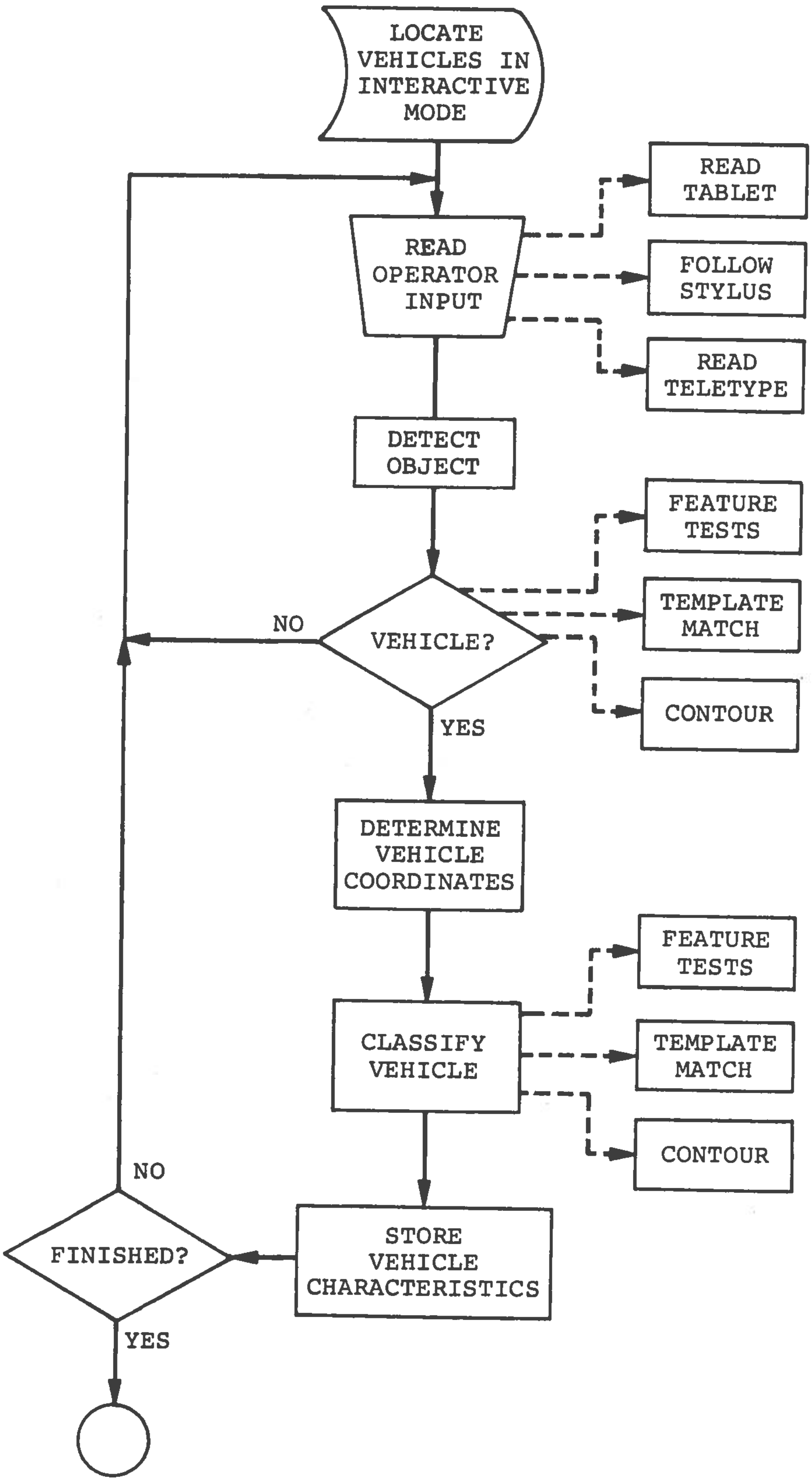


Figure 6. Locate Vehicles in Interactive Mode

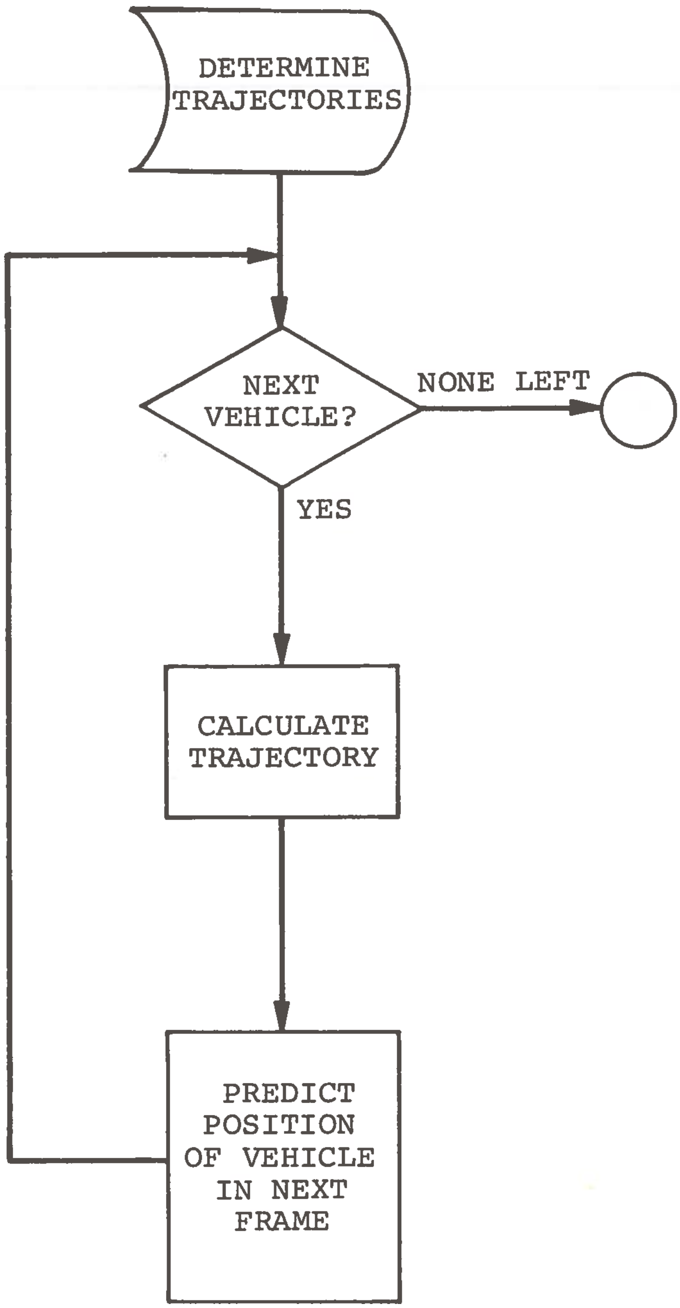


Figure 7. Determine Trajectories

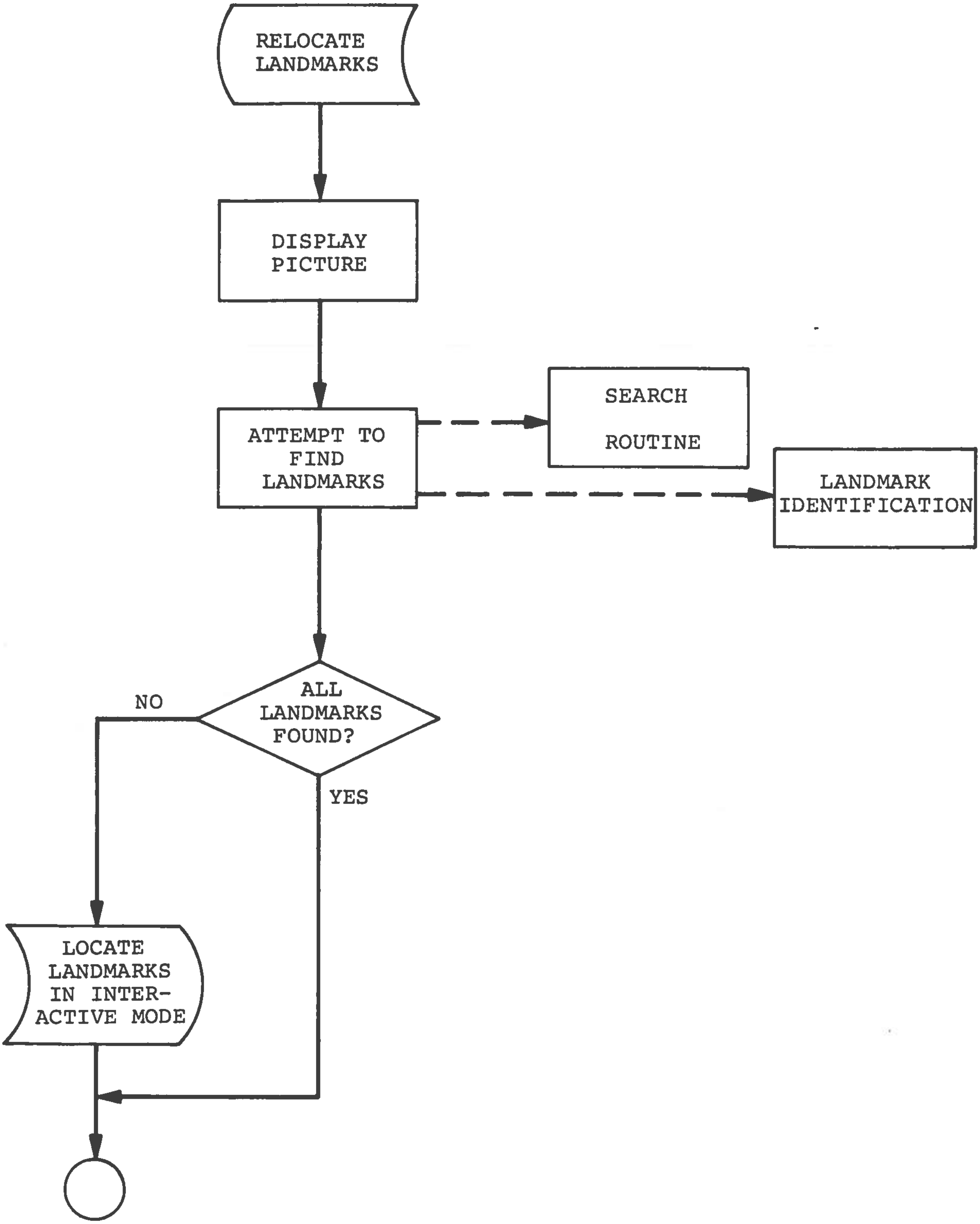


Figure 8. Relocate Landmarks

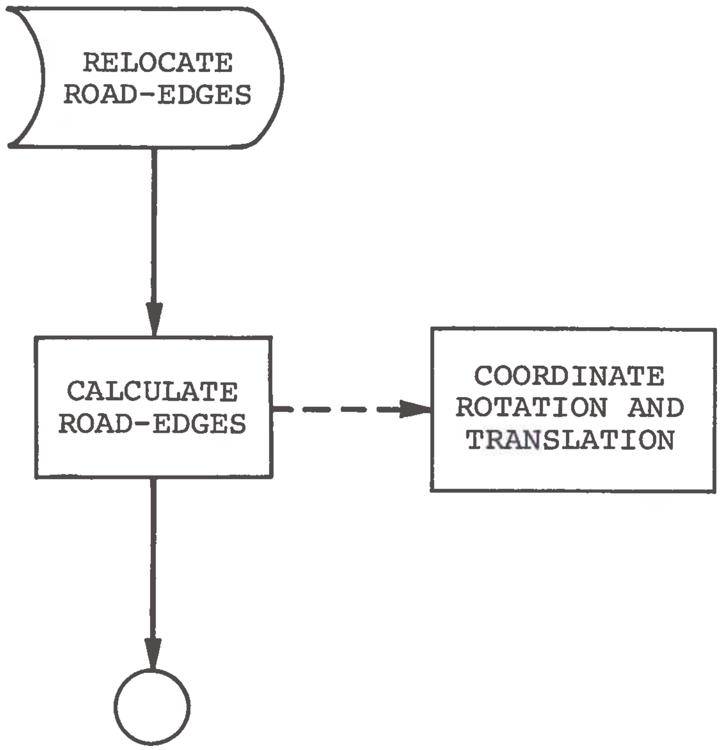


Figure 9. Relocate Road-Edges

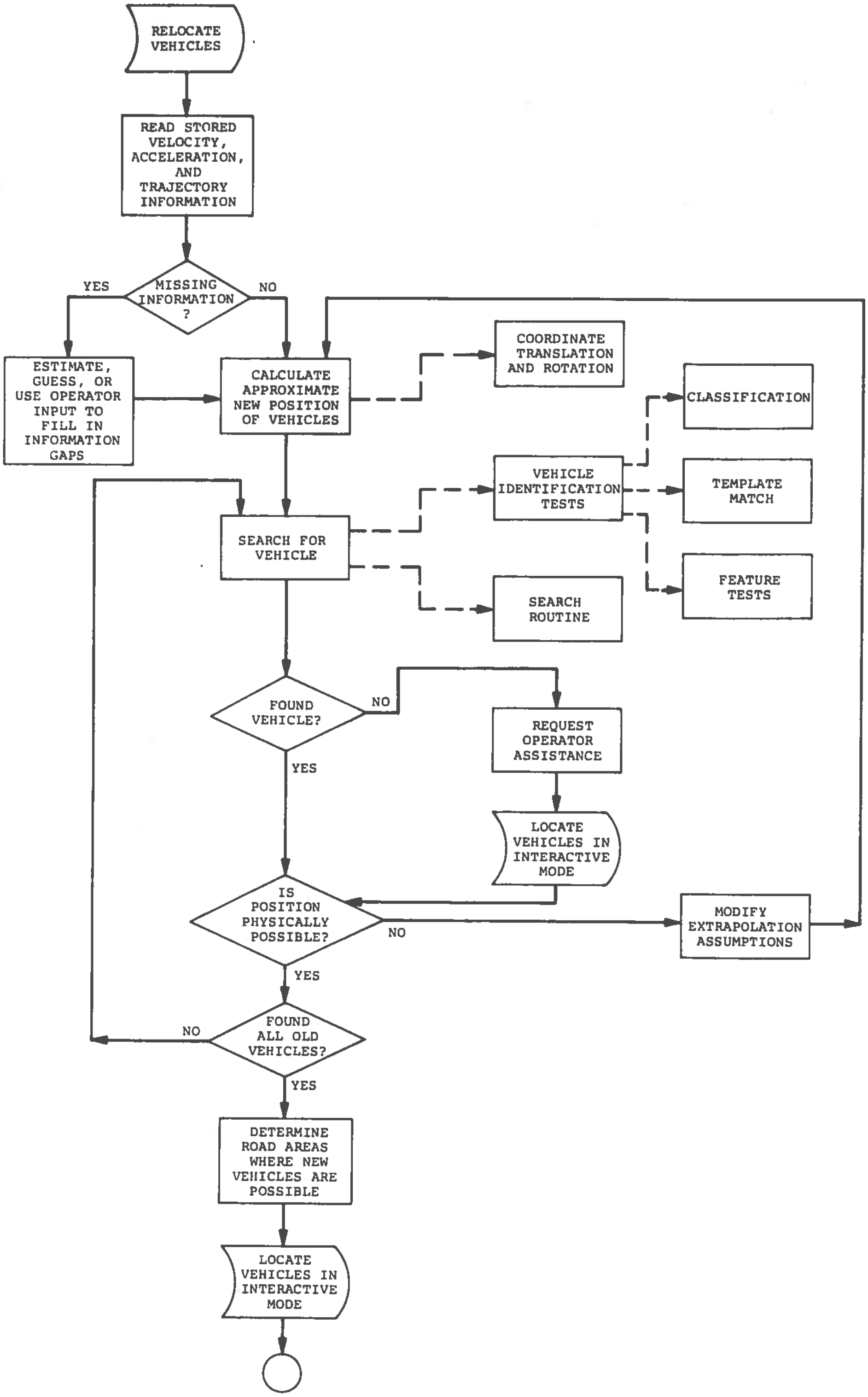


Figure 10. Relocate Vehicles

4.0 DESCRIPTION OF PROGRAMS DEVELOPED

This section contains a description of each of the programs developed thus far towards implementing the system described in Section 3. Appendix A contains the complete listings of the programs. All programs manipulating pictures assume that the digitized pictures are stored in the computer as sequences of words to be interpreted as follows: M,N I(1,1), I(1,2),...,I(1,N), I(2,1),...I(M,N) where M is the number of rows in the picture, N the number of points per row, and I(J,K) is the gray-level value of the K-th point in the J-th column (i.e. transmissivity of this point when the picture is scanned).

4.1 SCAN

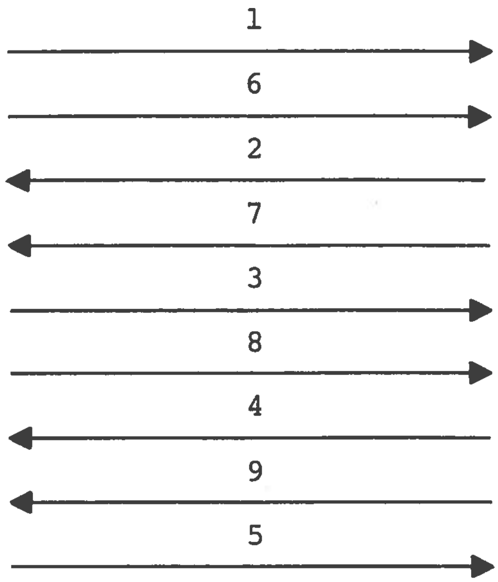
The following programs scan and store part of a frame of aerial highway film.

4.1.1 INTERLACED RASTER SCAN

CODE: : ILC

PURPOSE : Performs an interlaced zig-zag raster scan for display.

DISCUSSION: ILC will display a picture on the CRT as it scans it. Interlacing is used. An interlaced scan is one which displays odd-numbered lines and even-numbered lines on alternate sweeps. This method of scan is used to diminish visual fade and thus allow more points to be displayed without significant flicker. A zig-zag scan of the interlaced lines reduces the time for one sweep by decreasing the distance between two successive scan points when the end of a scan line is reached. The order of a typical scan is as follows:



CALLED BY :

CALL ILC,YTOP,YBOT,XLFT,XRGT,DELT
where
YTOP, YBOT,XLFT, XRGT = the scan boundaries
DELT = the spacing between scan points

METHOD : The zig-zag is accomplished by a "go left" and a "go right" loop. At the end of a scan line, the appropriate Y deflection is made to insure interlaced scanning.

SIZE : 145_8 locations

4.1.2 STORE PICTURE

CODE : STO

PURPOSE : To scan a picture segment using the TRIM scanner and store the digitized representation in core in standard form.

DISCUSSION: This is the basic routine for scanning and digitizing a picture. The maximum picture size that may be scanned is determined by the core memory available. The largest contiguous block possible is between octal locations 20,000 and 37,777, which accomodates a square picture of 90x90 elements.

CALLED BY : CALL STO,YTOP,YBOT,XL,XR,DELT,LOC
where
YTOP, YBOT, XL and XR denote the limits of the area to be scanned, DELT the spacing in terms of minimum TRIM deflection steps, and LOC the first core location for storing the picture.

METHOD : The area specified is scanned in conventional raster fashion. The minimum delay is allowed for deflection currents to settle between adjacent points (medium delay during the retrace). The intensity values are stored as read (i.e. without being shifted), or with arbitrary scaling or shifting. Such scaling must currently be provided for by modifying the program; the calling sequence does not provide for specifying any scale factors.

SIZE : 110_8 locations

4.1.3 PICTURE AVERAGING

CODE : AVG

PURPOSE : To scan a picture segment using the TRIM scanner and add the intensity values obtained to the current contents of memory.

DISCUSSION: Any digitized representation of a picture obtained by scanning will be corrupted by additive random noise. The signal-to-noise ratio may be improved by scanning repeatedly and averaging the values obtained. Calling STO once and AVG n-1 times will leave in core the unnormalized average values over n scans. The advantage of this procedure over repeating the reading n times during one raster scan is that it protects the CRT phosphor from overheating.

CALLED BY :
CALL AVG,YTOP,YBOT,XL,XR,DELT,LOC
where the calling parameters are the same as for STO.

METHOD : Essentially the same as for STO.

SIZE : 111_8 locations

4

4.2 DISPLAY AND PRINTOUT

DIS will display a stored picture, and PMD and PML will printout different representations of it on the line printer.

4.2.1 DISPLAY AND CALL

CODE : DIS

PURPOSE : Displays a stored picture, and calls the following functions of the displayed picture:

Contour trace (CRL, CRG), Oblique scan (OBL), Line-printer printout (PMD, PML), Road-edge contour trace (CRL, CRG), and Vehicle property determination (SZC).

DISCUSSION: After a selected area of the picture has been scanned and stored by STO and AVG, DIS will display it magnified to any desirable size. This enables close study of the picture.

It is used, for example, in landmark identification (LMI) where the neighborhood of the landmark is displayed enlarged to allow precise marking of the landmark.

DIS is also used to call other routines which perform functions on the stored picture. Printouts of the displayed picture can be generated. A rectangular area of the picture at any oblique angle can be picked out and redisplayed. An object in the redisplayed picture can be isolated by contour trace. A road-edge in the picture can be followed. A vehicle can be picked out and contour traced, and then its size properties and center can be found.

CALLED BY :

```
CALL  DIS,YDIS,XDIS,DDIS,LOC
where
(XDIS, YDIS) = the upper left point of the display
DDIS = the distance between display points
LOC = the location of the stored picture to be
      displayed
```

METHOD : DIS goes through the stored raster and displays the stored picture. To call other routines, Sense-switch 1 is set, whereupon "SWT=" is printed on the teletype. The operator types in a value which switches the program to any of five routines:

SWT	ROUTINE
2	Contour Trace (CRL, CRG)
1	Oblique Scan (OBL)
0	Line-Printer Print-out (PMD, PML)
-1	Display (DIS)
-2	Road-edge Contour Trace (CRL, CRG) and Size and Center (SZC)

On SWT = 2, the operator types in
(XO, YO) = the initial point of the contour trace
THSH = the threshold
and the contour trace of an object (in a rectangle found on the last oblique scan) is displayed. If (XO, YO) is not a contour point, a series of ten points directly above it are tested to find a contour point. If none is a contour point, another (XO, YO) can be tried.

On SWT = 1, the operator types in
(DXTP, DYTP) = the slope of the oblique scan lines
NROW, NCOR = the number of rows and column in the scan

and an oblique rectangle is picked out and redisplayed. The initial scan point of the oblique scan was previously chosen by stylus in the original mode (i.e. when SWT = 1).

On SWT = 0, the displayed picture is printed out in two forms: light-intensity values, and gray levels.

On SWT = -1, we return from any of the modes called by SWT to the original display mode (as when DIS was originally called). In this mode, an initial point can be set by stylus for the oblique scan or for the road-edge contour trace.

On SWT = -2, the operator types in
THSH = the contour trace threshold
LNTH = the number of points to be traced

and the computer will trace a contour starting with a point previously set by stylus in the original display mode (i.e. when SWT = -1). If the initial point is not on an edge, the twenty points above it are tested and if none is a contour point, another THSH can be chosen. The contour found will be displayed brightly superimposed on the original DIS display. If the initial point is near a road-edge, the road-edge will be displayed, with LNTH determining how many points in the edge will be displayed. If the initial point is on a vehicle, and if LNTH is greater then the number of points in the contour of the vehicle, then the vehicle contour will be displayed as a closed curve superimposed on the DIS display. At this juncture, if the stylus is depressed SZC will be called. The operator will type in

DELX, DELY = the slope of the center line of vehicle

and the teletype will print out

LNTH = the length of the vehicle
WDTH = the width of the vehicle
CNTR OF EXTENT = the center of extent point of the vehicle
CNTR OF GRAV = the center of gravity of the vehicle based on its contour points

The center of extent is defined as follows: the extreme points of the vehicle in the direction of the center line, and the extreme points of the vehicle in the direction perpendicular to the center line are found. These four points determine a rectangle which encloses the vehicle, and the center of the rectangle is defined to be the center of extent of the vehicle.

EXAMPLE : The following five photographs illustrate the calling of DIS by SFW, and the calls of DIS. This sequence uses several of the routines that have been written. Figure 11 shows a scanned area of highway (as produced when ILC is called by SFW). SFW calls STO and AVG to store the central section of the scanned area, and then SFW calls DIS. DIS displays the stored central section enlarged, as shown in Figure 12. The operator then chooses a vehicle, in this case the vehicle in the upper center of the enlarged section, by placing a cross on the vehicle by stylus (Figure 13). The operator calls the road-edge contour trace which traces the edge of the vehicle and shows it superimposed upon the display (Figure 14). The operator can also call

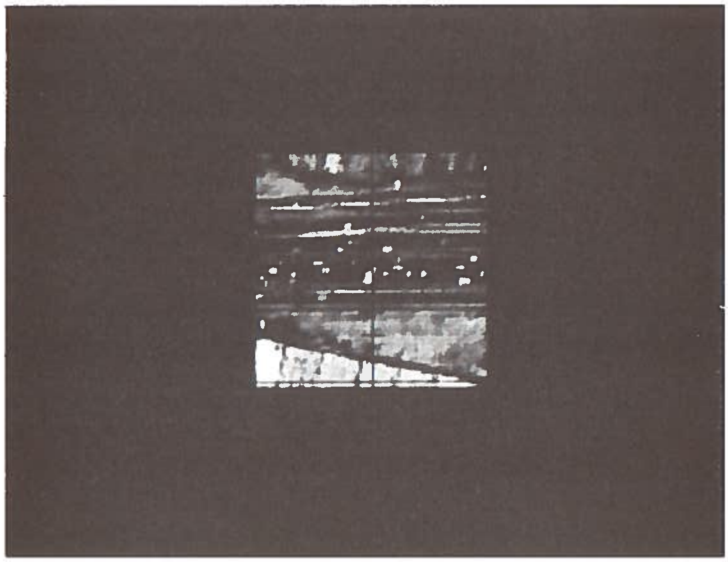


Figure 11. Area of a Highway Photograph Scanned by ILC when called by SFW

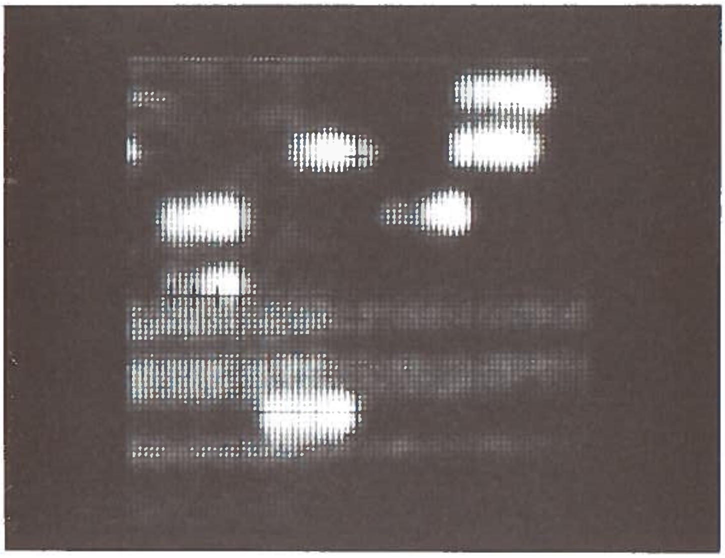


Figure 12. DIS Display of Stored Central Section of Scanned Area

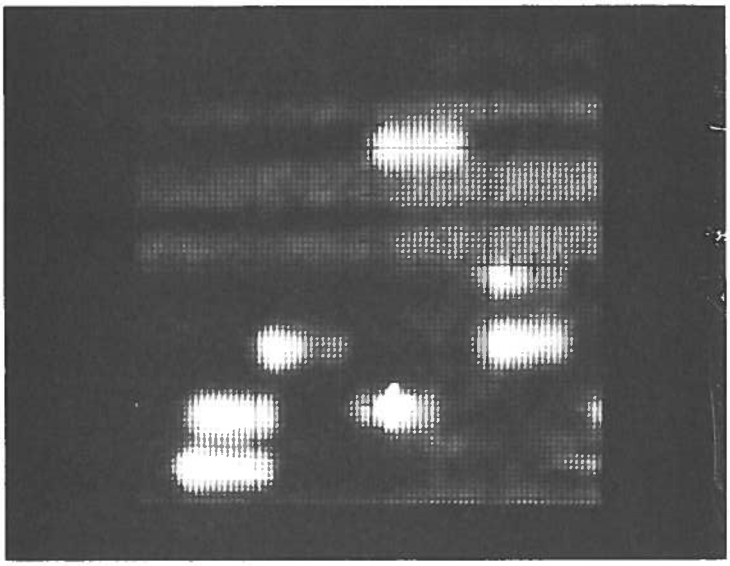


Figure 13. DIS Display with a Cross Placed on the Selected Vehicle

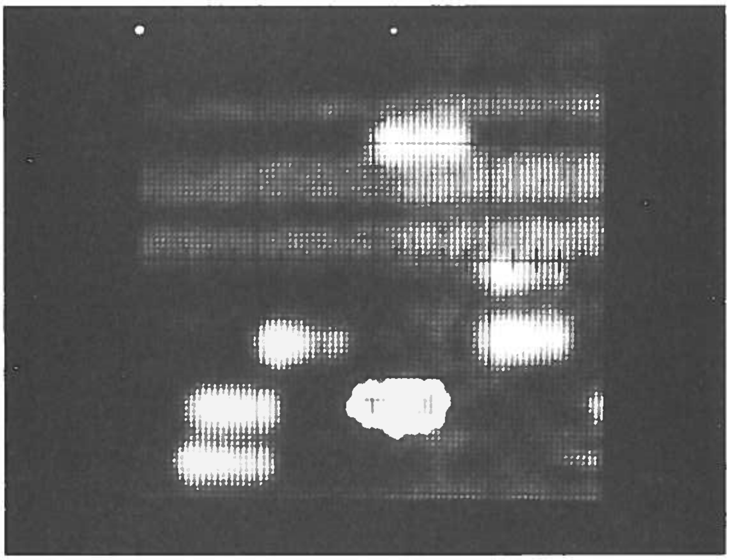


Figure 14. Contour of Selected Vehicle Determined and Displayed

the oblique scan which will pick out a small section of the display at any angle and redisplay it. Figure 15 shows the small section containing the vehicle from the upper center of the display being redisplayed in "normalized position" (i.e. with the vehicle facing upwards).

SIZE : 1154_8 locations

4.2.2. PRINT MATRIX

CODE : PMD, PML

PURPOSE : The line-printer prints a matrix from a stored picture, showing either an alphabetic digit (PMD) or a gray-level character (PML) for each point of the picture.

DISCUSSION: To get a hard-copy version of a stored picture, it is convenient to be able to print-out via line-printer a matrix corresponding to the stored picture. PMD will print a matrix where each position represents a picture point and where the character at each position represents the light intensity of the picture point. PML, a similar routine, can be used for a gray-level line-printer print-out, where the character at each matrix position has a visual gray-level proportional to the light intensity of the point. In this case, the print-out will visually recreate the stored picture.

EXAMPLE : An example of each print-out is shown in the Figures 16 and 17.

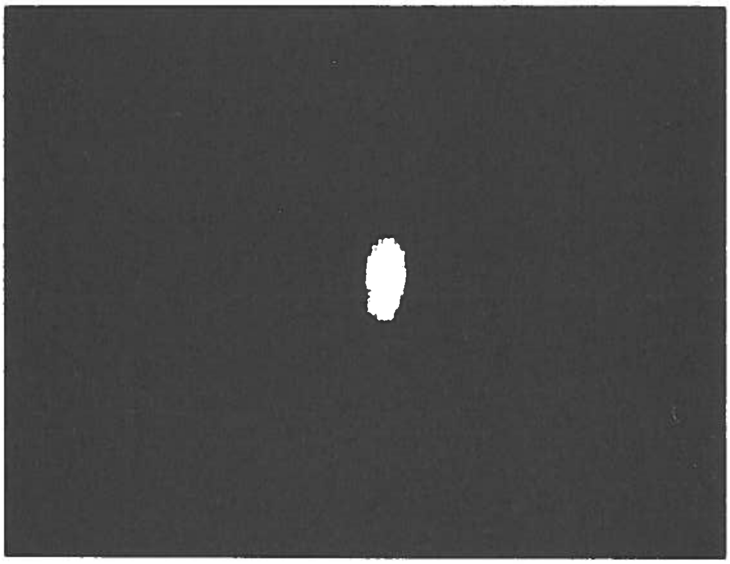


Figure 15. Selected Oblique-Angled Picture Section Redisplayed in "Normalized Position"

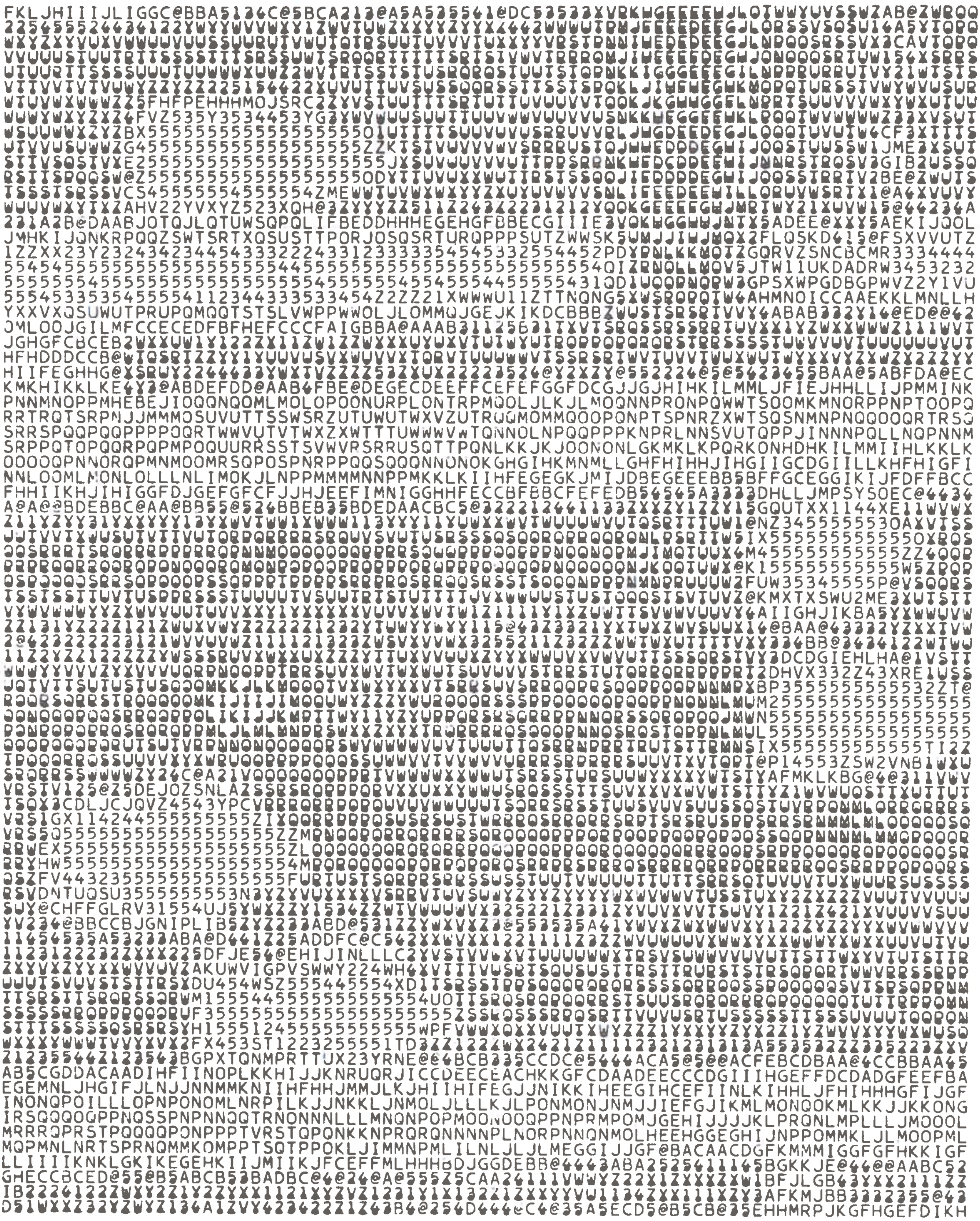


Figure 16. Light Intensity Print-out: The Symbol at each Point Signifies the Numerical Light-Intensity of the Point

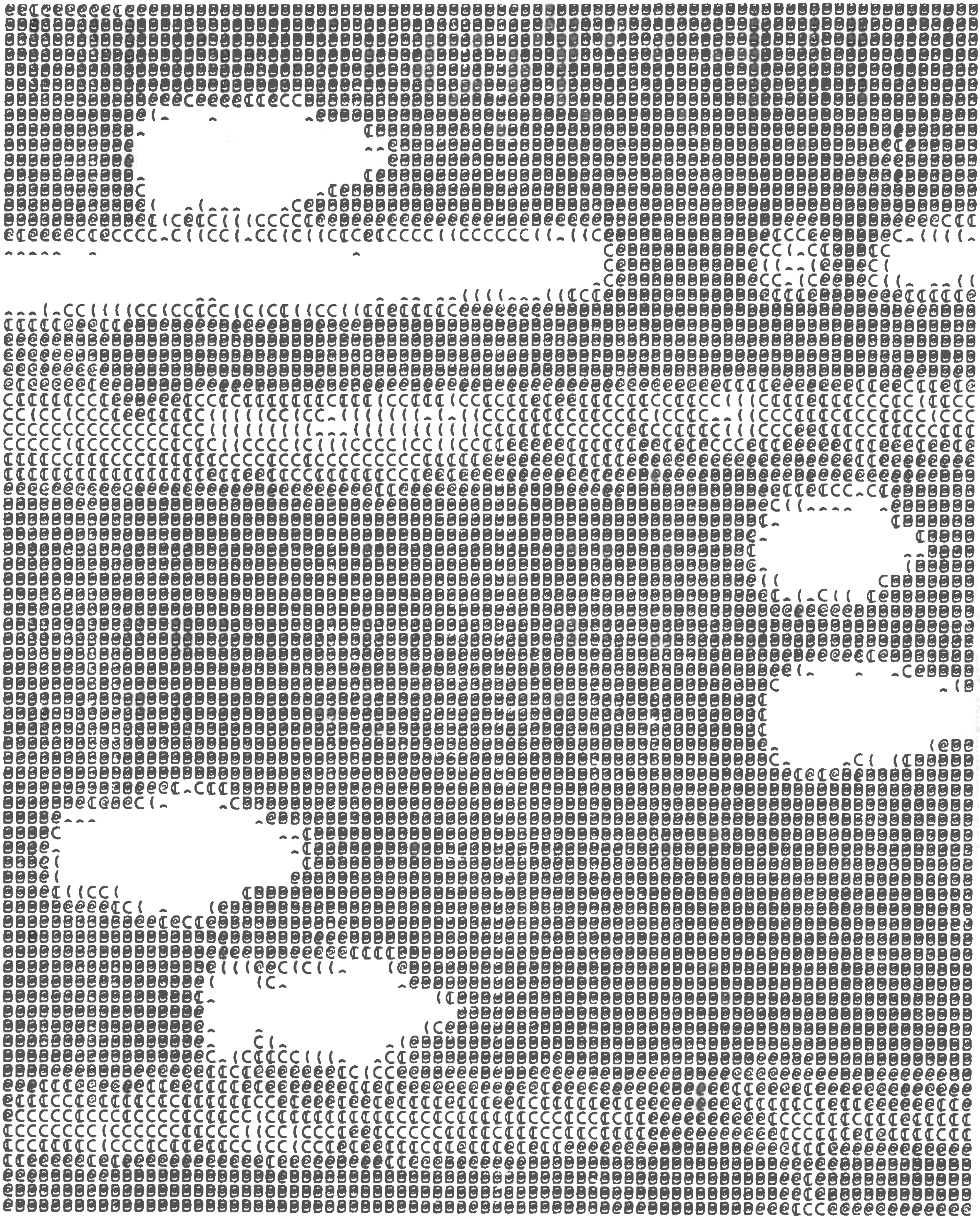


Figure 17. Gray-Level Print-out: The Symbol at Each Point has a Visual Gray-Level Approximating the Light Intensity of the Point

CALLED BY :
CALL PMD,PICT
or
CALL PML,PICT
where
PICT is the first location of the stored picture.

METHOD: : PMD will print an over-printed character corresponding to 64 light intensity levels using the following coding for the 64 levels = @, A, B,..., Z, 1, 2,..., 5, @, A, B,..., Z, 1, 2,..., 5. Note that the first group of 32 levels is the same as the second but the first has an over-printed period.

PML will print an over-printed character corresponding to selected ranges of light intensities to give a gray-level appearance. The over-printed characters pairs used are @B, @@, @C, CI, CC, ((, ^_, and __ (where _ here stands for blank). This gives the gray-level characters B,@,@,C,C,(,^, and .

When either routine is called, it reads through the stored picture row by row. For each point in a row, it computes or finds in a table the first character to be printed of the two over-printed characters that correspond to the light intensity level of the point. That character is stored in a buffer, and when the characters for the whole row have been found, the contents of the buffer is printed by the line-printer. A line-feed command is not given, and the process is repeated on the same row for the second of the two characters per position. After the second character is printed over the first, a line-feed instruction is executed and the next row of the stored picture is set to be printed.

SIZE : 516_8 instructions

4.3 FILTER

The following routines are used to sharpen a picture by filtering.

4.3.1 LAPLACIAN CALCULATOR

CODE : LAP

PURPOSE : To compute the Laplacian of a picture segment stored in core in standard form.

DISCUSSION: Computation of the Laplacian is one of the standard edge enhancing operations that can be performed on a picture. The Laplacian of a picture I(x,y) is

$$\nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

The numeric approximation of the Laplacean for a picture sampled over a square grid with spacing h is

$$\nabla^2 I(i,j) = \frac{1}{h^2}[I(i+1,j)+I(i-1,j) + I(i,j+1) + I(i,j-1)-4I(i,j)]$$

CALLED BY :

CALL LAP,LOCP,LOCL

where LOCP is the first storage location in core of the picture to be operated upon, and LOCL is the first storage location in core for storing the Laplacian, also as a picture in standard form, but with 2 fewer rows and columns. LOCL may be the same as LOCP.

METHOD : The Laplacian is evaluated by the application of the equation given, but without the division by h^2.

SIZE : 70_8 locations

4.3.2 COMBINE PICTURES ON DISK AND IN CORE

CODE : CDC

PURPOSE : To compute the weighted sum of segments of two pictures, of which one is stored on disk and the other in core, and to replace the picture in core with the computed result. If $I_1(x,y)$ and $I_2(x,y)$ are the source pictures, the resulting picture will be an NX x NY segment

$$I_3(x,y) = (C_1 I_1(x+dx_1, y+dy_1) + C_2 I_2(x+dx_2, y+dy_2))/C_3$$

DISCUSSION: This routine was written to be utilized in constructing filtering functions - e.g. to add a picture and its Laplacean, etc.

CALLED BY :

```
CALL  CDC,U,C1,DY1,DX1,LOC,C2,DY2,DX2
MDAC  C3,NY,NX
```

where
U is the literal unit number associated with the disk file on which picture I_1 is stored.
LOC is the symbolic first address of the picture I_2 in core (and becomes the address of the first location of the resulting picture I_3). The other calling parameters are defined by the equation above.

METHOD : The program loops through line by line, and point by point within lines. The computed values are stored in place of the picture stored in core. The dimensions of the source pictures and the offsets and dimensions specified as calling parameters are checked for compatability. The routine types ERROR 700 and returns to TOP if they are incompatible. The routine assumes that a buffer BUFA of sufficient length to hold a complete line of the picture on disk has been defined externally.

SIZE : 272_8 locations

4.4 INTERACTIVE OPERATION

The following programs involve interactive operation of the system.

4.4.1 MAIN PROGRAM (STYLUS FOLLOWER)

CODE : SFW

PURPOSE : This is the main program of a demonstration package. It follows the graphics tablet stylus with a cross and an interlaced scan, moving window and calls several other routines.

DISCUSSION: SFW will display a cross (CSS, CSD) at the position corresponding to the stylus and will display an interlaced scan (ILC) of part of the frame. When the stylus is depressed, the interlaced scan will be centered at the stylus point. This is used by the operator to pick out the part of the frame (i.e. the sector of highway) that he is interested in. Then, on sense switch 2 set, STO, AVG, and DIS are called in succession. STO and AVG store and average a picture section, and DIS displays it enlarged. DIS then can call routines to print-out (PMD, PML), oblique scan (OBL), contour trace (CRL, CRG), and determining properties (SZC).

CALLED BY : Executed directly.

METHOD : With all sense switches down, SFW displays a small section of the frame (by calling ILC). The display is centered at the last point where the stylus was depressed, and this is continually checked. Thus, the displayed section can follow the stylus, giving a "moving window" effect. With sense switch 4 set, the display is enlarged three times in width and in height by calling ILC with different parameters. To prevent excess blinking of the image, this larger section is displayed at coarse resolution. An example of such a scanned section is shown in Figure 2 of Section 4.2.1. With sense switch 3 up, the large and small displays are alternated, thus showing a detailed inner square surrounded by a coarse outer square.

On sense switch 2 set, STO, AVG, and DIS are called, and the smaller displayed section is stored and redisplayed enlarged.

SIZE : 402_8 locations

4.4.2 SET CROSS, DISPLAY CROSS

CODE : CSS, CSD

PURPOSE : Displays a cross (i.e. a "+") at a given location on the CRT.

DISCUSSION: Display of a cross is useful for tracking the data tablet stylus, for choosing a desired point on a picture, and for displaying a chosen point or points superimposed on a picture.

CALLED BY :

```
CALL  CSS,N,DELT,Z
where
  N = The number of displayed points in the
      horizontal (or vertical) line of the cross.
```

DELT = the separation between cross points.
Z = the intensity of the cross display.

<u>and</u>

```
CALL  CSD,XMID,YMID
```

where
(XMID, YMID) = the mid point of the cross.

EXAMPLE : If CSS is called with parameter values:
N = 5
DELT = 2
Z = '3777
and CSD is then called with parameter values:
XMID = 25
YMID = 53,
the display will be centered at (25, 53) and will look as follows:

```
O   O   O   O   X   O   O   O   O

O   O   O   O   O   O   O   O   O

O   O   O   O   X   O   O   O   O

O   O   O   O   O   O   O   O   O

X   O   X   O   X   O   X   O   X

O   O   O   O   O   O   O   O   O

O   O   O   O   X   O   O   O   O

O   O   O   O   O   O   O   O   O

O   O   O   O   X   O   O   O   O
```

where
X = bright point
O = dark point

METHOD : CSS sets up the size and brightness for a cross. CSD is then called and displays this cross at the desired position.

CSS need be called only once, and it will set up a standard cross for use with CSD throughout the program.

The cross cannot have equal length legs if both DELT is odd and N is even, and therefore this should be avoided.

SIZE : 105_8 locations

4.4.3 READ MESSAGE

CODE : RMR

PURPOSE : To read alphanumeric input from teletype and store it in core packed 2 characters/word.

CALLED BY : CALL RMR,-N,LOC
where N is the number of words reserved for storage beginning at LOC. RMR will read up to 2N characters.

METHOD : The system routine for reading characters via teletype is invoked. RMR returns to the calling program after either 2N characters or a carriage return have been typed. Any unused storage locations after the final character are filled with blanks.

SIZE : 50_8 locations

4.4.4 READ NUMBERS

CODE : RNR

PURPOSE : This routine accepts via teletype and leaves in the accumulator either decimal or octal integers. Octal numbers must be identified as such by preceding apostrophe.

DISCUSSION: The system routines provide for reading either decimal or octal numbers. RNR allows the operator to select the input format at run time.

CALLED BY : JST RNR

METHOD : RNR normally invokes the system routine to read a decimal number. If the first character typed is an apostrophe, RNR invokes the system routine to read an octal number.

SIZE : 15_8 locations

4.4.5 TEXT MATCHING

CODE : MCH

PURPOSE : To compare two strings of alphanumeric characters, one given explicitly in calling sequence, the other stored in some specific location in core. Return is to instruction immediately following calling sequence if texts match exactly, to next succeeding instruction if they do not.

DISCUSSION: MCH is intended for use in conjunction with RMR for interpreting operator instruction via teletype during program execution.

CALLED BY : The macro-instruction

MATCH n,LOC [or LOC*],TEXT

which expands to

```
  JST*  MCH=
  DEC   -n
  DAC   LOC [or DAC* LOC]
  BCI   N,TEXT
```

where n is the number of words used to store each string of text (of up to 2n characters) and LOC is the first core location of the unknown string.

SIZE : 27_8 locations

4.4.6 PARAMETER READING

CODE : PRR

PURPOSE : This routine accepts parameters from an operator via teletype in an interactive way.

DISCUSSION: At various times during the development of programs it has been necessary to try different combinations of values for various parameters in order to achieve the desired performance. The routine PRR provides for convenient input of such values in the various circumstances that have arisen in practice. Alternate sets of parameters can be entered, to be used as the occasion arises. The parameters are stored in a core array as follows: Assuming that there are n sets of m different named parameters, the block is defined by the assembly language statements

```
  BLOK DEC  n
       DEC  m
       BCI  2, PAR1
  PAR1 BSS  m
       BCI  2, PAR2
  PAR2 BSS  m
       ...
  PARN BSS  m
```

This sequence of statements can be created by the macro
BLOK PARAMETERS, n, m, PAR1, PAR2...PARn
where m and n are (literal) integers, and PARj may be any symbolic variable name of 4 or fewer alphanumeric characters.

CALLED BY :

CALL PRR,BLOK,k
BCI k,MESSAGE TO BE TYPED
where BLOK identifies the parameter block

METHOD : When called, PRR

1. Types the message specified in the calling sequence
2. Types "SET NR."
3. Reads number input by operator. If this is greater than is specified in defining BLOK, it types ? and accepts another value.
4. Given a valid set number, PRR types "WHICH" and lets operator type option selected. If operator types
   a. "ALL", the routine types the variable names in order, and accepts a numeric value (decimal, or octal if preceded by ') for each, then returns to calling program.
   b. "NOW", the routine types the variable names and their current (decimal) values, then types "WHICH" and awaits new operator input.
   c. "COPY", the routine types "SET NR.", accepts a number and sets the value of each parameter in the set under consideration to the value in the set given, then types out the variable names and values, awaits new operator input.
   d. Any of the parameter names, routine acknowledges by typing "=". and accepts numeric value, awaits next input.
   e. "NEXT", routine goes to step 2.
   f. "NONE", routine exits to calling program.
   g. Carriage return or carriage return following arbitrary number of spaces, routine exits to calling program.
   h. Anything else, routine types question mark, awaits new imput. PRR invokes the subroutines RMR, RNR, AND MCH.

SIZE : 325_8 locations

4.5 TEMPLATE MATCHING AND LANDMARK IDENTIFICATION

LMI allows the operator to identify landmarks. PDC is used to find the template match between two pictures. PAR is called by PDC.

4.5.1 LANDMARK IDENTIFICATION PROGRAM

CODE : LMI

PURPOSE : To allow the operator to interactively identify landmarks in picture currently being scanned by TRIM.

DISCUSSION: LMI permits the operator working with the stylus on the graphics tablet to select sub-areas of the picture being scanned for display enlarged, and to select and identify by number points within these sub-areas. Their coordinates are stored and remain available for use by other routines. LMI is intended for use in marking landmarks in any given frame, and for interactively marking vehicles.

CALLED BY : JST LMI

METHOD : The routine provides the following sequence of steps for marking a point:

1. The full frame of imagery in the scanner is displayed on the monitor scope. Points marked already are shown by crosses. The scanner during this stage is in the READ mode. The operator may move the stylus to select the sub-area to be displayed in detail. The position of the graphics stylus is indicated by a small intensified area about it. An example is shown in Figure 18.
2. When the operator depresses the stylus, the selected sub-area is scanned and digitized at maximum scanner resolution. It is then displayed enlarged,(see Figure 19), with the scanner in the DISPLAY mode.
3. The operator may select the point to be marked by moving the stylus on the graphics tablet. The position of the stylus is marked by a cross superimposed on the image as shown in Figure 19. When the stylus is depressed, its location in terms of the full-frame co-ordinates is computed and assumed to be the location of a landmark. The routine will

request its identifying number by typing LNDMK NR = on the teletype and accept an integer $0 \leq i \leq 20$. The landmark coordinates will then be stored in the symbolic locations LMIX + i and LMIY + i. If the integer typed in is outside the proper range, the routine will type a question mark and accept another value. After a valid number has been read, the routine returns to step 1, with the new landmark now shown as a cross (as in Figure 20).

Raising sense switch 1 during the display of the full frame causes a return to the calling program.

The routine LMI invokes routines CSS, CSD, ILC, STO, AVG, DIS, AND RNR.

SIZE : 344_8 locations.

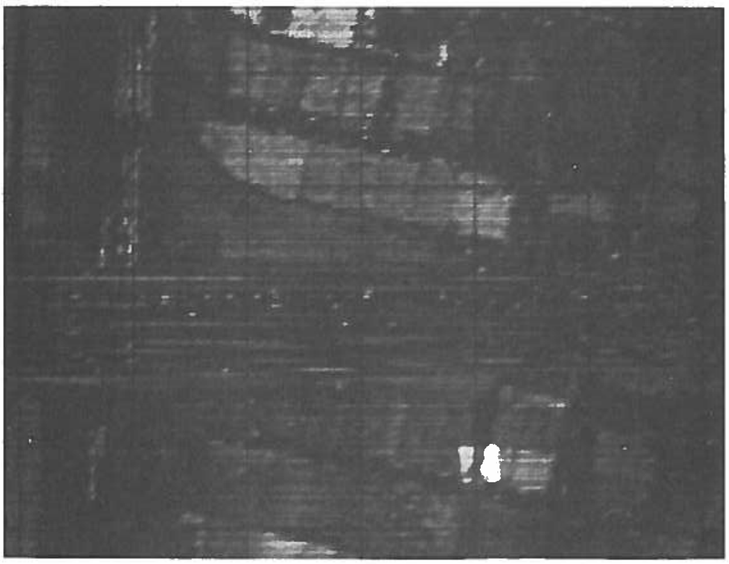


Figure 18. A Full Frame is Scanned and a Subarea is Selected by the Brightened Square

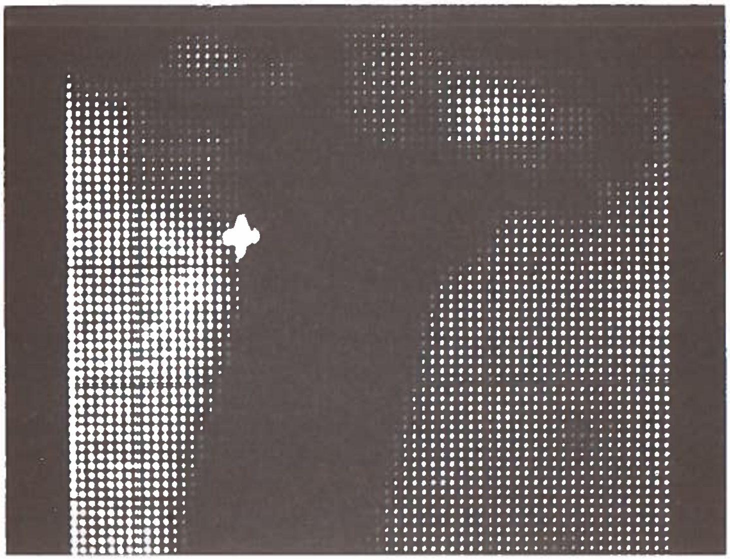


Figure 19. The Selected Subarea is Displayed Enlarged, and the Selected Landmark Point is Chosen with the Cross

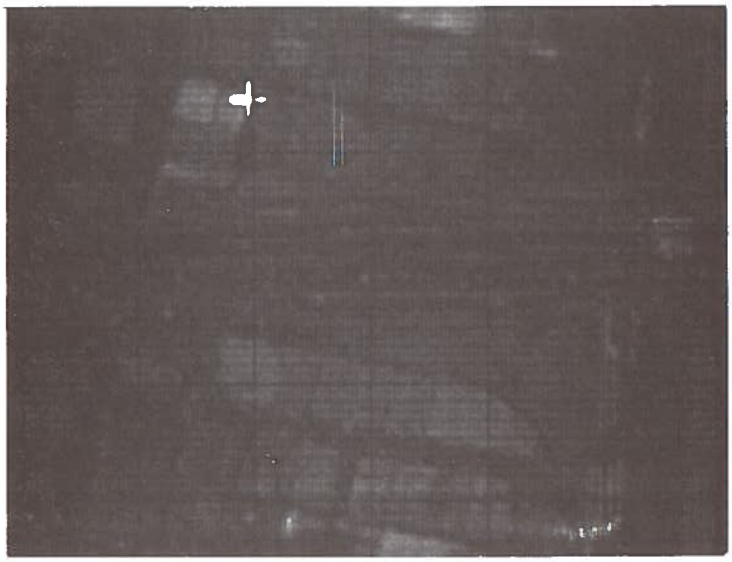


Figure 20. The full Frame is Displayed Again with the Cross Indicating the Previously Determined Landmark

request its identifying number by typing LNDMK NR = on the teletype and accept an integer $0 \leq i \leq 20$. The landmark coordinates will then be stored in the symbolic locations LMIX + i and LMIY + i. If the integer typed in is outside the proper range, the routine will type a question mark and accept another value. After a valid number has been read, the routine returns to step 1, with the new landmark now shown as a cross (as in Figure 20).

Raising sense switch 1 during the display of the full frame causes a return to the calling program.

The routine LMI invokes routines CSS, CSD, ILC, STO, AVG, DIS, AND RNR.

SIZE : 344_8 locations.

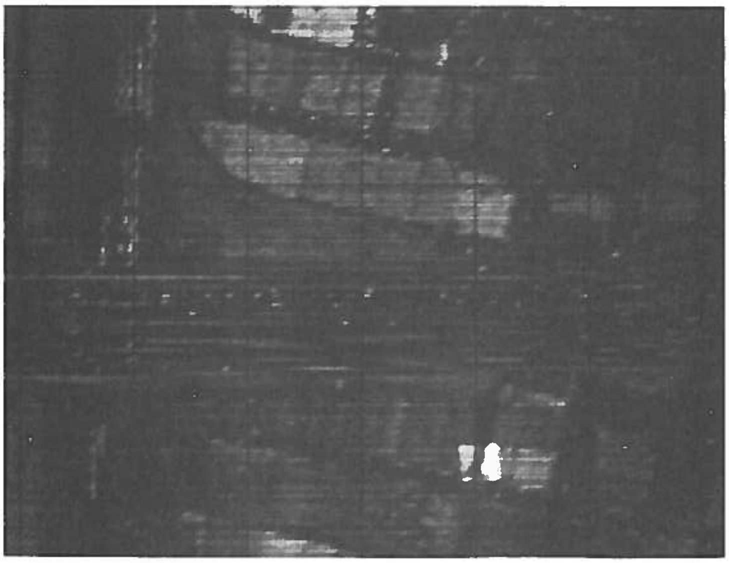


Figure 18. A Full Frame is Scanned and a Subarea is Selected by the Brightened Square

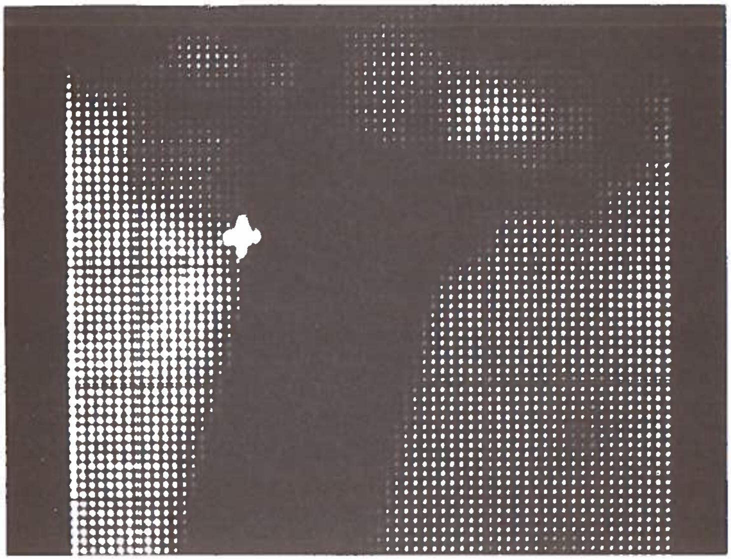


Figure 19. The Selected Subarea is Displayed Enlarged, and the Selected Landmark Point is Chosen with the Cross

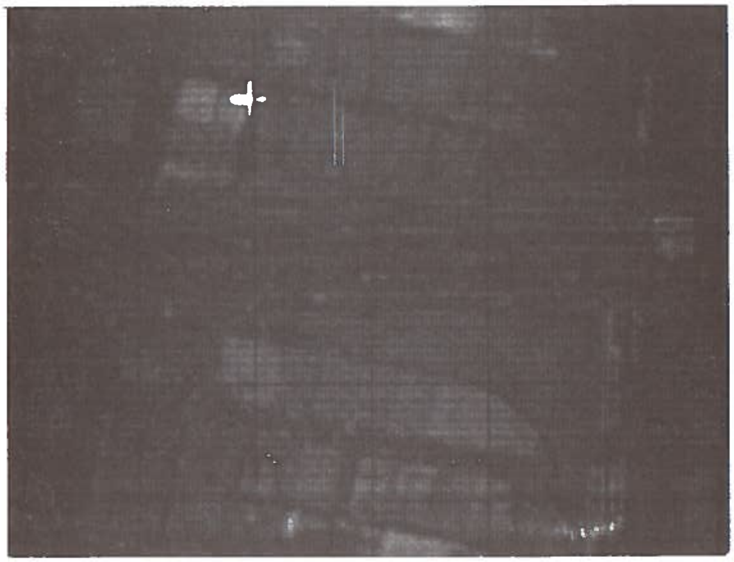


Figure 20. The full Frame is Displayed Again with the Cross Indicating the Previously Determined Landmark

4.5.2 PICTURE COMPARISON (DISK AND CORE)

CODE : PDC

PURPOSE : To compute the cross-correlation of two picture segments, one on disk and one in core. The quantity computed is

$$\sum_{j=1}^{NY} \sum_{k=1}^{NX} D(DY1+j, DX1+k)\ C(DY2+j, DX2+k)$$

where D(i, j) is the intensity of the (i, j) point of a picture stored on the disk, and C(i, j) is the intensity of the (i, j) point of the picture in core.

DISCUSSION: Given a picture C and another picture D such that

$$D(x, y) = a\ C(x, y) + b$$

where a and b are constants, and letting R be a region of the x, y plane: the quantity

$$M(\xi,\zeta) = \frac{\iint_R C(x, y)\ D(x+\xi, y+\zeta)\ dxdy}{\iint_R D^2(x+\xi), y+\zeta\ )\ dxdy}$$

will have its maximum at $\xi=\zeta=0$ for all a and b ($a\neq 0$). This relation is well known. The proof is based on the Schwartz inequality.

If the picture D(x, y) also includes additive, uncorrelated noise, then the expected value $E[M(0,0)] \geq E[M(\xi,\zeta)]$ for $\xi,\zeta\neq 0$. The routine PDC computes the numerator of the function $M(\xi,\zeta)$, replacing integration by summing.

CALLED BY :

CALL PDC,U,DY1,DX1,LOC,DY2,DX2,NY,NX
where
U is the literal number of the disk file containing picture D
LOC is the first storage location of the picture C stored in core in standard form.
The significance of the other calling parameters is clear from the description above.

METHOD : The program accumulates the double sum, looping first row by row and within rows point by point.

The answer is returned as a double precision number in the combined A,B register. The C bit is set if an overflow occurs during computation. Checks are performed to verify that the calling parameters are compatible with the dimensions of the stored pictures. If not, the routine goes to TOP, typing ERROR 701. The routine requires that a buffer BUFA of sufficient length to contain a full line of the picture on the disk be externally defined.

SIZE : 146_8 locations

4.5.3 PICTURE AUTOCORRELATION ROUTINE

CODE : PAR

PURPOSE : To compute the quantity

$$S(PIC) = \sum_{i=DY+1}^{DY+NY} \sum_{j=DX+1}^{DX+NX} [I(i, j)]^2$$

for a picture, where I(i,j) is the gray level value at the j-th point of the i-th row of a picture stored in core beginning at location PIC.

DISCUSSION: $\sqrt{S}$ is a normalizing factor in computing the degree of match between two picture segments by means of cross-correlation. The result is stored as a double-precision number in ANS and ANS+1.

CALLED BY :

```
CALL  PAR,PIC,NY,NX,DY,DX,ANS
```

METHOD : The routine accumulates the sum, going through the picture row by row, and point by point within each row. Locations '500 and '501 are used for temporary storage. No check is made as to whether DY, NY, DX, and NX are compatible with the size of the picture segment stored.

SIZE : 100_8 locations

4.6 OBLIQUE SCAN

The following programs provide the facility to scan an oblique rectangle from a stored picture and to draw a line at any oblique angle.

4.6.1 SET DRAW LINE, DRAW LINE

CODE : DLS, DLN

PURPOSE : Draws the best staircase approximation to a line at a given slope from a given point.

DISCUSSION: The ability to draw a line at a given slope is needed in oblique scanning (see OBL) and other applications. Since the scanned picture points are points on a square grid of finite resolution, this involves finding a discrete approximation to a straight line. It was decided to use a series of points to approximate the line such that each point in the series would be the one of the eight points adjacent to the previous point in the series that is closest to the line's continuation. This is equivalent to drawing the real line across the digitizing grid, and choosing the closest grid point whenever the real line crosses a grid line.

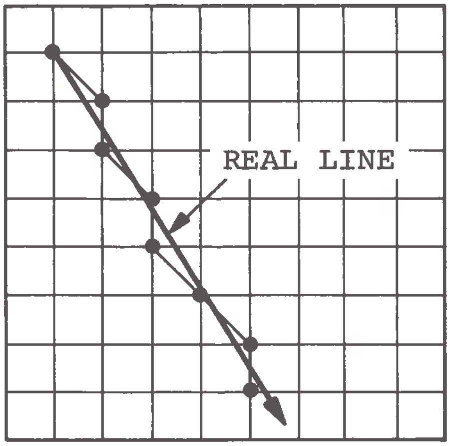
CALLED BY :

```
CALL  DLS,X1,Y1,DELX,DELY
where
   (X1, Y1) = The initial point on the line
   DELY = The signed change in Y for each signed
          change in X of DELX
and
   CALL  DLN
```

EXAMPLE : If DLS is called with

```
   (X1, Y1) = (1,8)
   DELY/DELX = -5/+3
```

Then if DLN is called repeatedly seven times, it will successively return the points as indicated below:



METHOD : DLS sets up the parameters for a new line. Then DLN is called repeatedly, and each time returns (in the A and B registers respectively) the X and Y values of the next point of the digitized line. The following algorithm was devised to compute the digitized line:

Given: ΔX and ΔY, the signed slope components, and (X1, Y1), the initial point.

Let: $\delta X_k = X_{k+1} - X_k$

$\delta Y_k = Y_{k+1} - Y_k$

(Note: by definition of the digitized line both δX_k and δY_k can only have values +1, 0, or -1.)

then:

(1) For $|\Delta X| \geq |\Delta Y|$, i.e. Octants 1, 4, 5, and 8:

$T_1 = c$

$T_k \geq 0 \rightarrow \quad \delta X_k = \text{sgn } (\Delta X)$

$\delta Y_k = \text{sgn } (\Delta Y)$

$T_{k+1} = T_k + b$

$$T_k < 0 \rightarrow \quad \delta X_k = \text{sgn}(\Delta X), \quad \delta Y_k = 0, \quad T_{k+1} = T_k + a$$

where:

$$a = 2|\Delta Y|$$
$$b = 2|\Delta Y| - 2|\Delta X|$$
$$c = 2|\Delta Y| - |\Delta X|$$

(2) For $|\Delta X| < |\Delta Y|$, i.e. Octants 2, 3, 7, 8

$$T_1 = c'$$

$$T_k > 0 \rightarrow \quad \delta X_k = \text{sgn}(\Delta X), \quad \delta Y_k = \text{sgn}(\Delta Y), \quad T_{k+1} = T_k + b'$$

$$T_k < 0 \rightarrow \quad \delta X_k = 0, \quad \delta Y_k = \text{sgn}(\Delta Y), \quad T_{k+1} = T_k + a'$$

where:

$$a' = 2|\Delta X|$$
$$b' = 2|\Delta X| - 2|\Delta Y|$$
$$c' = 2|\Delta X| - |\Delta Y|$$

In the above

$$\text{sgn } Z = \begin{cases} +1 & \text{for } Z \geq 0 \\ -1 & \text{for } Z < 0 \end{cases}$$

DLS computes a, b, c (or a', b', c') and initializes the algorithm. The algorithm allows very fast computation, and when DLN is called, it returns the next point of the digitized line in 22 μsec.

SIZE : 170_8 locations

4.6.2 OBLIQUE SCAN

CODE : OBL

PURPOSE : "Scans" an oblique rectangular area of a picture in core, and stores the rectangle in standard form. The "scan" lines are at the oblique angle of the rectangle.

DISCUSSION: In successive frames, a vehicle may be oriented at different angles in the picture due to its following along the curvature of the road, due to its lane-changing, entering, and exiting maneuvers, and due to camera movement. To make valid pictorial comparisons of vehicles in different frames, it is necessary to normalize the position of the vehicle. This can be done by scanning the vehicle in each frame using scan lines which are perpendicular to the vehicle's direction of motion in that frame. The scanned points are stored in a standardized format independent of the oblique angle at which they were scanned. The vehicle's images in different frames can then be compared.

The above process introduces comparison inaccuracies due to the digitization. For example, if the scan lines are at 45° angle, the distance between available points is $\sqrt{2}$ units. Thus the nth point on a scan line corresponds to a position on the vehicle which is $n\sqrt{2}$, units from the first point of that scan line. For a 0° vehicle image, the nth point corresponds to a position n units from the first point of the scan line.

Thus, in matching a 0° vehicle image against a 45° vehicle image, there will be inherent inaccuracies, since rather than matching points that correspond to the same vehicle-surface point, we will be matching points which correspond to vehicle-surface points $n\sqrt{2} - n$ units apart.

This kind of inaccuracy can be reduced by interpolation, if necessary. However, a study of the errors inherent in matching vehicles scanned at different oblique angles revealed that interpolation may not be necessary for the data considered in this problem. Design standards for highway curvature vs. design speed shows that the angle through which a vehicle following the highway curvature could turn in one second is less than 7°, and that the large factors of safety used make it this maximum very rarely encountered in practice.

The following test was made: A vehicle 21 points by 11 points was assumed to be oriented at angles 0°, 5°, 10°, 15°, and 20°. At each orientation angle the vehicle was scanned, using as scan-lines the oblique line approximations. The actual vehicle points hit were recorded. These are compared in Figures 21 to 25, for orientation angle differences of 5° and 10°. Five different scan rows are shown for each orientation. The distances between corresponding points is shown in the Figures to be no more than two units (where a unit distance is the distance between two successive scan points), when one of the upper corner points of the vehicle-images were superimposed. In practice, this is a worst-case condition, as the vehicle images will usually be centered with respect to each other. Thus, if we use the points on scan-row 10, we see that the distance between vehicle-image points for vehicle images separated by 5° is never more than one unit.

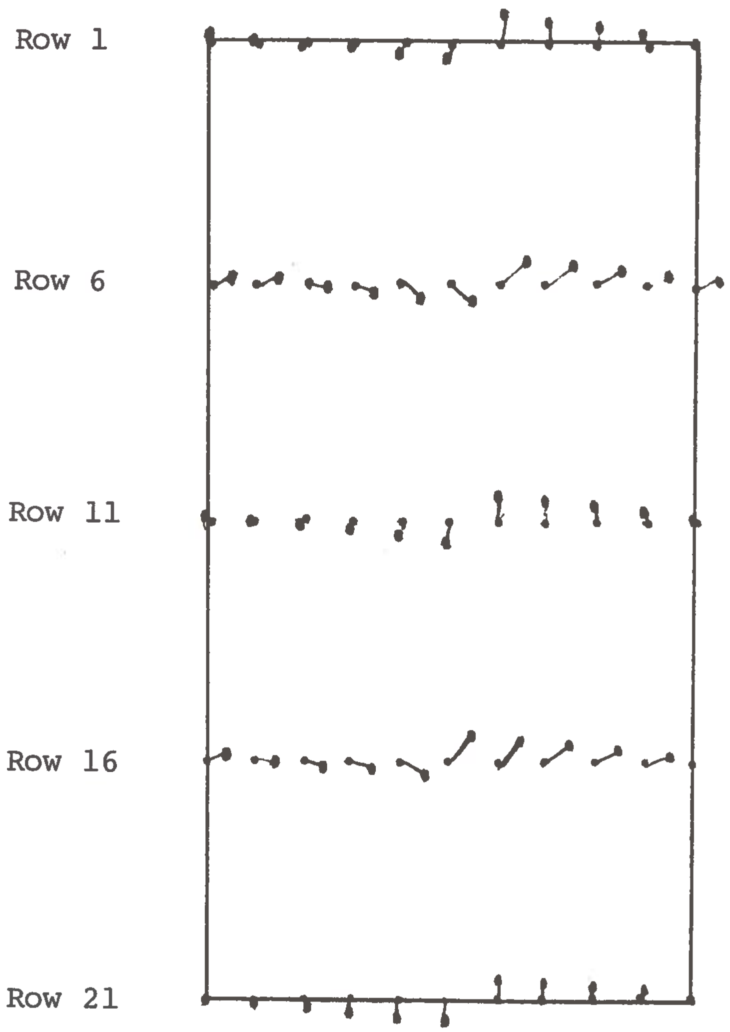


Figure 21. A 0° Vehicle vs. a 5° Vehicle

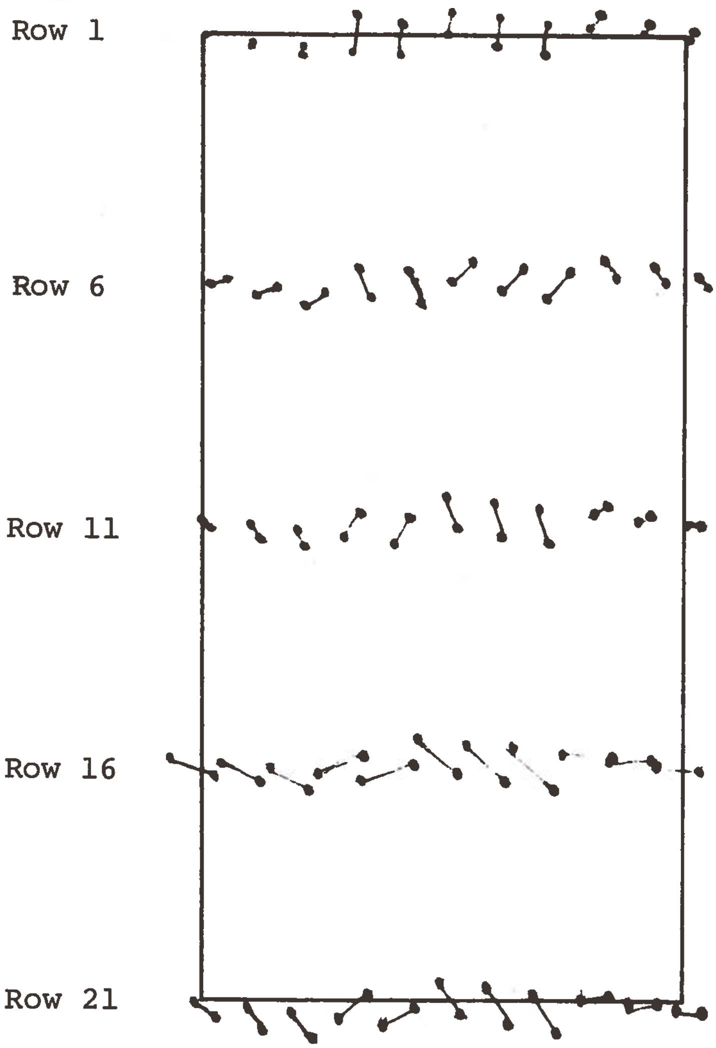


Figure 22. A 5° Vehicle vs. a 10° Vehicle

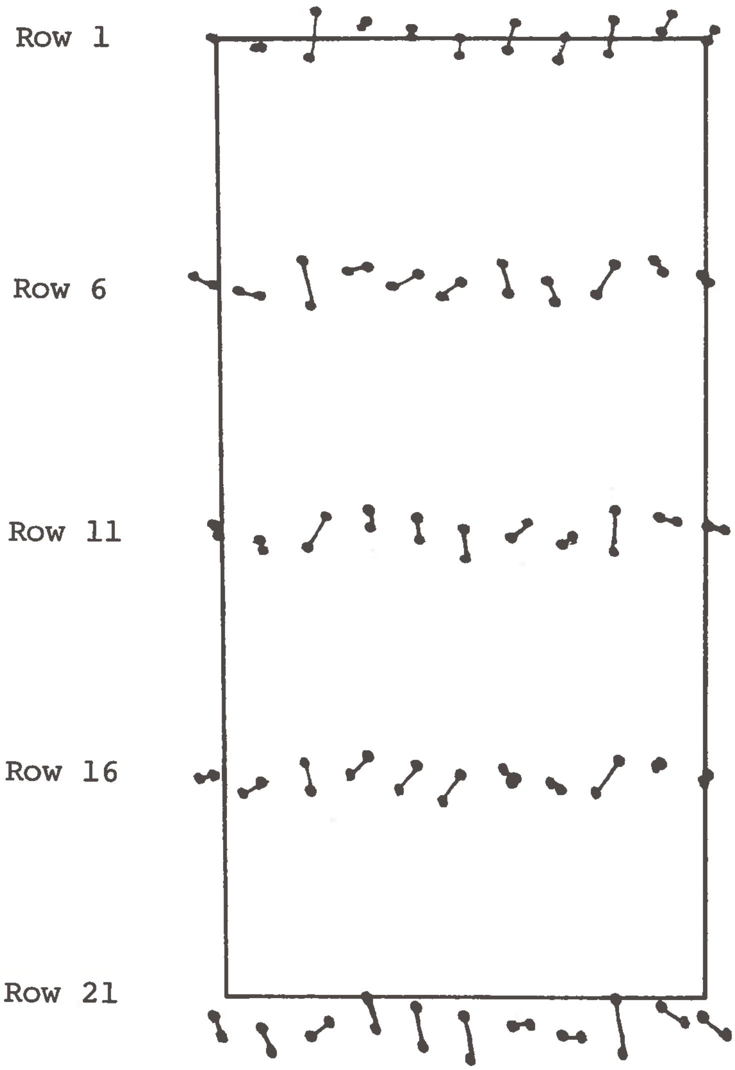


Figure 23. A 10° Vehicle vs. a 15° Vehicle

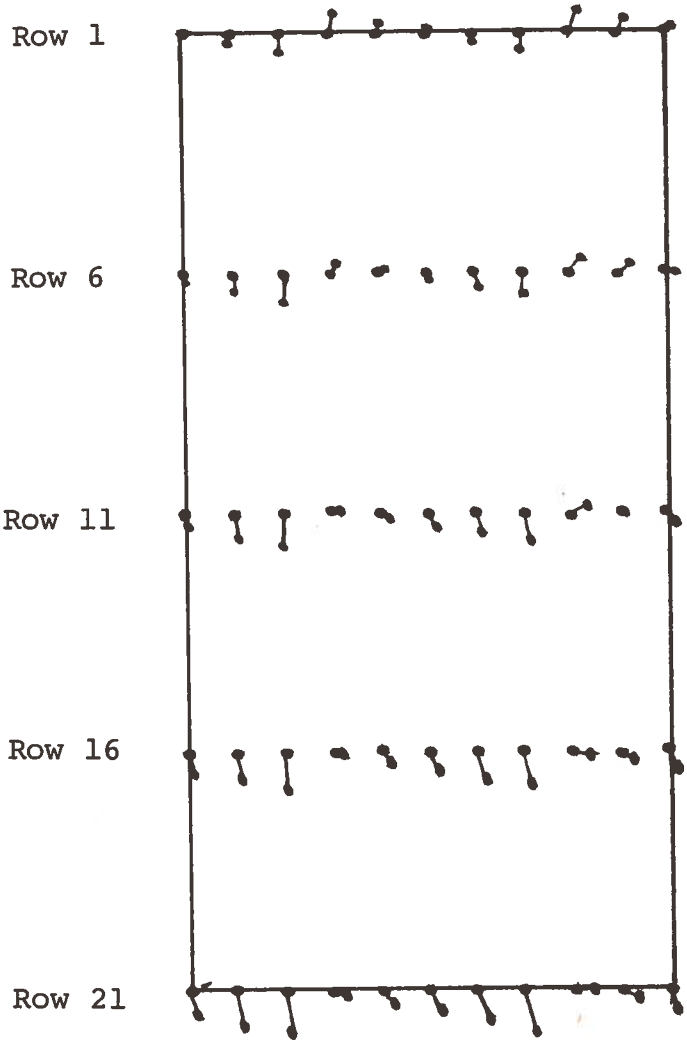


Figure 24. A 0° Vehicle vs. a 10° Vehicle

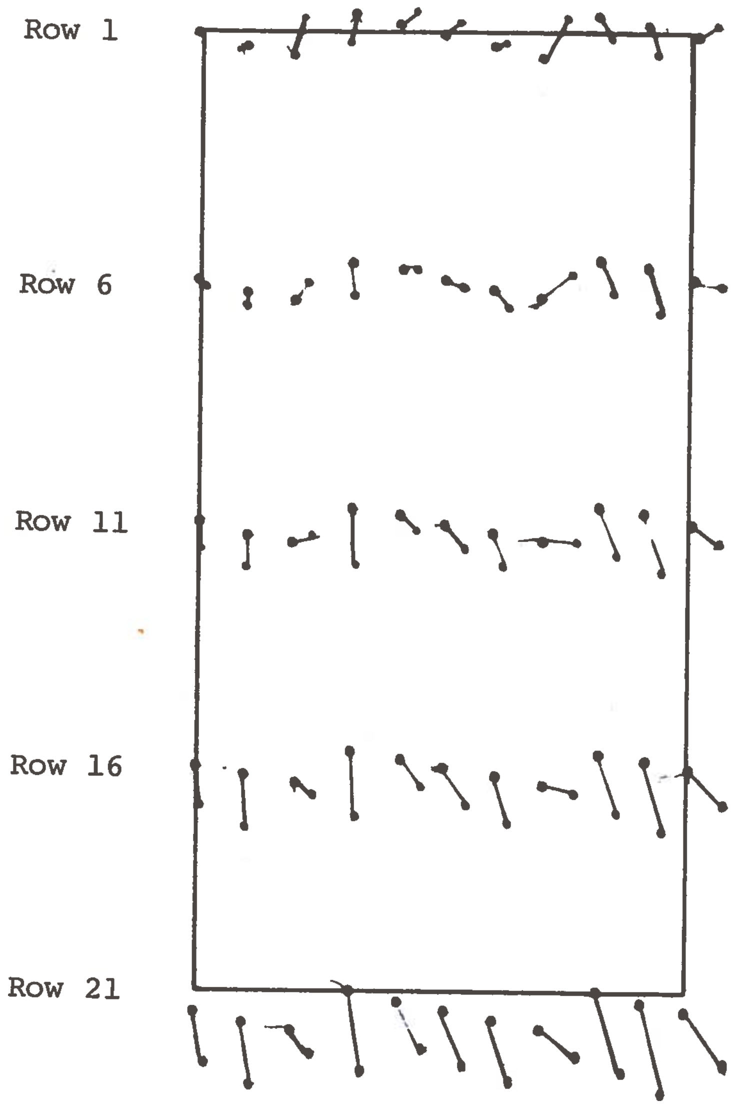


Figure 25. A 10° Vehicle vs. a 20° Vehicle

CALLED BY :

```
CALL  OBL,XC,YC,DXTP,DYTP,NROW,NCOL,RECT,PICT
```

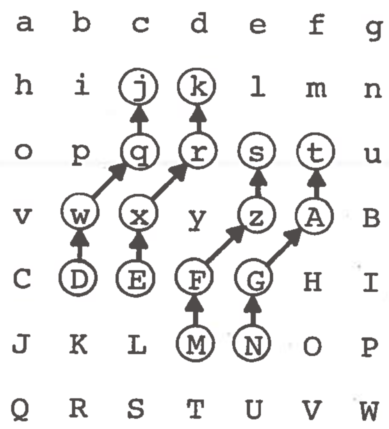
where
(XC, YC) = That corner point of the oblique rectangle which will be stored as the first point of the scan, i.e. the upper left point if the rectangle was along the coordinate axes.
DYTP/DXTP = The directioned slope, starting from (XC, YC), of the "top" of the rectangle (i.e. the first scan line). The signs of DXTP, DYTP give the quadrant. XC, YC, DXTP, DYTP are expressed in the stored pictures coordinates.

NROW = The number of rows to be scanned.
NCOL = The number of columns to be scanned.
RECT = Is the first location of the array where the rectangle will be stored.
PICT = Is the first location of the stored picture.

EXAMPLE : For the picture matrix:

a	b	c	d	e	f	g
h	i	j	k	l	m	n
o	p	q	r	s	t	u
v	w	x	y	z	A	B
C	D	E	F	G	H	I
J	K	L	M	N	O	P
Q	R	S	T	U	V	W

if OBL is called with parameters
XC, YC, = 1, 2 (i.e. point "D")
DYTP/DXTP = +3/+1
NROW = 4
NCOL = 4
the following scan pattern will be followed



and the rectangle stored will be:

D w q j

E x r k

M F z s

N G A t

METHOD : OBL uses two DLN's. One finds a line from (XC, YC) perpendicular to the slope of the rectangle, and thus finds the rectangle's "left" sideline. Each point in this line is used as an initial point for the scan lines which are produced by the second DLN.

CALLS :

DLS, DLN: Draw-line
D2S, D2N: A second copy of Draw-line

SIZE : 202_8 locations

4.7 CONTOUR TRACE AND PROPERTY DETERMINATION

CRL, CRD, and CRG are a package of programs for tracing the contour of an object in a picture. SZC can find properties of an object whose contour is known. SQR and DSQ find square-roots for SZC and other routines.

4.7.1 CONTOUR TRACE (RIGHT)

CODE : CRL, CRD, CRG

PURPOSE : CRG performs a contour trace on a stored picture along a light-dark contour, keeping the light area (set up by CRL) or the dark area (set up by CRD) on the right.

DISCUSSION: The contour trace program will find a series of border points between a light region and a dark region ("light" and "dark" being determined by a selected threshold value). This will be used to find the boundary of a vehicle and thus pick out the vehicle from its background. In addition, from the set of contour points of a vehicle several important vehicle properties can be found. These include vehicle position, size, shape, and orientation. Another important use of the contour trace is in finding road-edges.

CALLED BY :

```
CALL  CRL,XO,YO,THSH,PICT
or
CALL  CRD,XO,YO,THSH,PICT
WHERE
  (XO, YO) = the initial point
  THSH = the threshold
  PICT is the first location of the stored picture
```

<u>and</u>

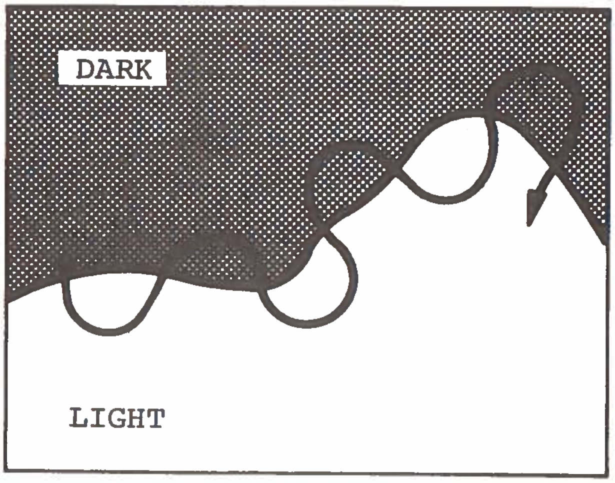
```
CALL CRG
returning
  A-Register = x value of next contour point
  B-Register = y value of next contour point
```

METHOD : To have a contour trace keeping the "light" area on the right, CRL is called once to set up the contour trace (i.e. to define the first point and the threshold) and then CRG is called repeatedly, returning the successive light contour points. To have a contour trace keeping the dark area on the right, CRD is called to set up the contour trace and then CRG is called repeatedly, returning the successive dark contour points.

One general algorithm for contour tracing of a light region in a dark background is:

1. Go clockwise in a dark region - go counterclockwise in a light region.
2. Output the light point at every light-dark transition.

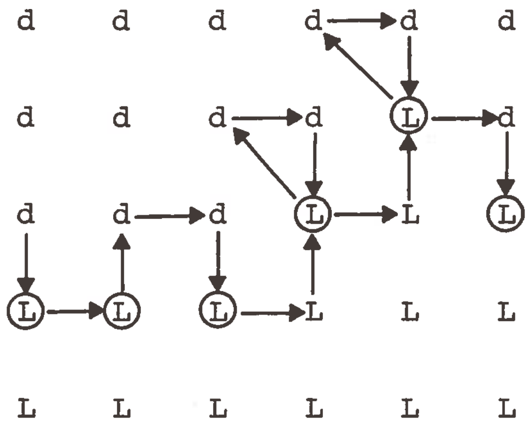
This will produce a series of light points on the contour, as shown in the following figure for the continuous picture case:



In the case of a digitized picture, the algorithm becomes:

1. Go right (from the last direction of motion) when a dark point is hit - go left (from the last direction of motion) when a light point is hit.
2. If hit three light (dark) points in a row, assume next point hit must be dark (light), and thus take a diagonal step.
3. Whenever there is a light-dark transition, output the light point (except if it has just been output). For diagonal steps output as if it two steps: one vertical and one horizontal. An example of this is shown in the following figure:

b



The complete algorithm* is shown in Table 1. It was written in the form of a finite-state machine, with the state representing the immediate past history of the trace. Since this information does not have to be found each time (as it is inherent in the state), the routine is fast.

SIZE : 1074_8 locations

---

*Adapted from: Prerau, David S.: Computer Pattern Recognition of Standard Engraved Music Notation, PhD Thesis, M.I.T., September 1970, pp. 77-83

TABLE 1. THE CONTOUR TRACING ALGORITHM

STATE NUMBER	STATE NAME*	LIGHT move	LIGHT state	LIGHT output	DARK move	DARK state	DARK output
1	L-U	←	7		→	12	
2	L-D	→	8		←	11	
3	L-L	↓	6		↑	9	
4	L-R	↑	5		↓	10	
5	LL-U	←↑	9	↑⋯•	→	12	• ⋯→
6	LL-D	→↓	10	•⋯↓	←	11	• ←⋯
7	LL-L	↓←	11	•⋯←	↑	9	↑ •
8	LL-R	↑→	12	•⋯→	↓	10	• ↓
9	D-U	←	3	←⋯•	→	16	
10	D-D	→	4	•⋯→	←	15	
11	D-L	↓	2	•⋯↓	↑	13	
12	D-R	↑	1	•⋯↑	↓	14	
13	DD-U	←	3	←⋯•	→↑	1	
14	DD-D	→	4	•⋯→	←↓	2	
15	DD-L	↓	2	•⋯↓	↑←	3	
16	DD-R	↑	1	•⋯↑	↓→	4	
0	INITIAL	D Below - State 9 D Above - State 10 D Right - State 11 D Left - State 12			ERROR		

*For example, LL-U means have hit two light points in a row, and have just moved Up.

4.7.2 SIZE AND CENTER

CODE : SZC

PURPOSE : Finds the length, width, center of mass, and center of extent of a vehicle from a stored picture.

DISCUSSION: These properties determine vehicle position and extent. In addition, they will be used as features for matching vehicles in successive frames.

CALLED BY :

```
CALL  SZC,DELX,DELY,INX,INY,THSH,LOC
MDAC  LNTH,WDTH,XCTR,XGRV,YGRV
where
  DELY/DELX = the directed slope of the center
              line of the vehicle (the signs of
              DELY and DELX specifying the
              quadrant).
  (INX, INY) = a point on the contour of the
               vehicle (in stored-picture
               coordinates).
  THSH = The threshold used to determine the
         contour.  LOC is the address of the
         stored picture.

The following is returned:
  LNTH = the extent of the vehicle in the direc-
         tion along its center line.
  WDTH = the extent of the vehicle in the direc-
         tion perpendicular to its center line.
  (XCTR, YCTR) = the center point of the vehicle,
                 based on extent (i.e. the center
                 of a rectangle along the
                 vehicle center-line which cir-
                 cumscribes the vehicle.
  (XGRV, YGRV) = the center of gravity of the
                 vehicle based on its contour.
```

METHOD : SZC calls CRL and CRG which will find the contour points of the vehicle. Then SZC computers the projection, V, of each contour point on the center line of the vehicle, and the projection, U, of each contour point on an axis perpendicular to the center line. The following equations are used (Figure 26):

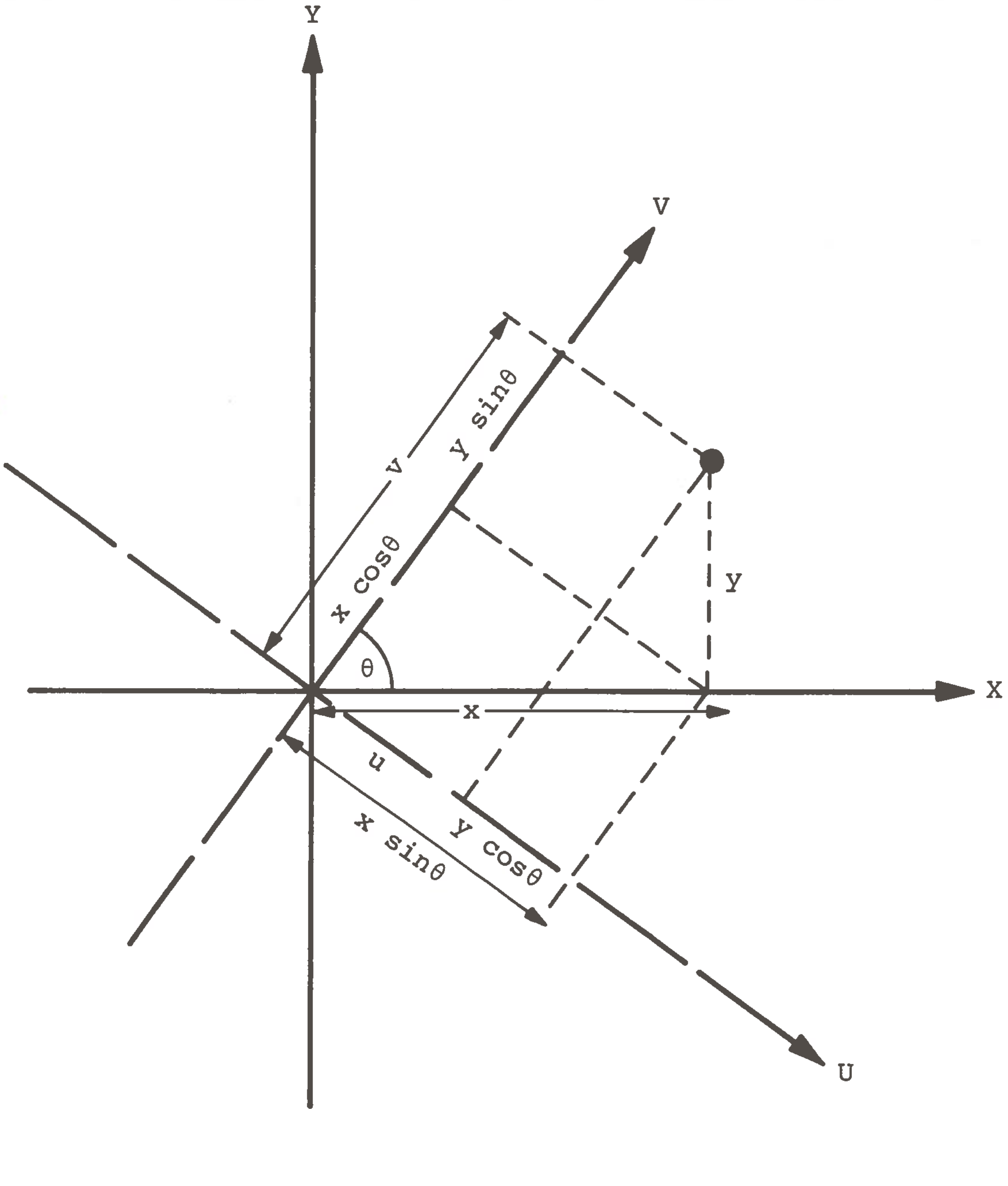


Figure 26. Projections of a Point onto the Vehicle Axes, U and V

$u = x \sin\theta - y \cos\theta = (x\,\Delta Y - y\,\Delta X)/Q$

$v = x \cos\theta + y \sin\theta = (x\,\Delta X + y\,\Delta Y)/Q$

where θ = the angle between the Y axis and the center-line of the vehicle
and $\Delta Y/\Delta X$ = the slope of the vehicle center-line
and $Q = (\Delta X^2 + \Delta Y^2)^{1/2}$

Then, the length and width of the vehicle are found by:

LNTH= Vmax - Vmin
WDTH= Umax - Umin

The center of extent (i.e. the point midway between Vmax and Vmin and midway between Umax and Umin) is found by:

$$V_c = \frac{1}{2}(Vmax + Vmin)$$

$$U_c = \frac{1}{2}(Umax + Umin)$$

$$X_c = V_c \cos\theta + U_c \sin\theta = (V_c\,\Delta X + U_c\Delta Y)/Q$$

$$Y_c = V_c \sin\theta - U_c \cos\theta = (V_c\,\Delta Y - U_c\Delta X)/Q$$

The center of gravity is found by

$$X_G = \frac{1}{2} \sum_{1}^{N} X$$

$$Y_G = \frac{1}{2} \sum_{1}^{N} Y$$

where
N = the number of contour points

CALLS : CRL, CRG: Contour Trace
SQR: Square Root

SIZE : 327_8 locations

4.7.3 SQUARE ROOT, DOUBLE PRECISION SQUARE ROOT

CODE : SQR, DSQ

NAME : SQUARE ROOT, DOUBLE PRECISION SQUARE ROOT

PURPOSE : Finds the square root of a given number.

DISCUSSION: The square root function, either is single or double precision, is used by several routines (e.g. SZC).

CALLED BY :

```
CALL  SQR
where
  A-Register = N, the number whose square root will
               be found returning.
  A-Register = Square root (N), for N>0.
  A-Register = 0, otherwise.
```

<u>and</u>

```
CALL  DSQ
where
  A, B Registers = N, in double precision.
returinng
  A-Register = Square root (N), for N>0.
  A-Register = 0, otherwise.
```

METHOD : The following algorithm is used:

$$A_1 = 2^{P/2}$$

where 2^P = highest power of 2 less than N.

$$A_{k+1} = \frac{1}{2} (N/A_k + A_k)$$

Four iterations will suffice.
Rounding takes place such that:

$$A(A+1) \leq N$$

SIZE : SQR 60_8 locations
DSQ 104_8 location

5.0 MACROS AND THE MACRO PROCESSOR

The programming of the routines described above is significantly aided by the use of macro-instructions (macros). In this section, we shall discuss the macro-processor and then give a brief description of each of the macros used.

5.1 THE MACRO PROCESSOR

The purpose of the macro-processing program is to convert source language statements that are written in a form convenient for the programmer into statements or sequences of statements to achieve the desired end that are in the form of DAP assembly language and can thence be converted into machine code by the standard assembler. In effect, it allows the standard assembly language to be expanded by the inclusion of special statements suitable for the class of programs to be written.

For example, we have found it convenient to implement a special mnemonic instruction to read the TRIM analog to digital converter. The mnemonic is

RADC (for Read Analog to Digital Converter).

The macro-processor converts a source statement of the form

```
NAME RADC COMMENT
```

to the two lines of assembly language code

```
NAME INA   '1070 COMMENT
     JMP   *-1
```

The advantage to the programmer of having the menemonic is clear: it is more concise than the equivalent assembly language statements, and more easily remembered. It frees the programmer of having to remember the numerical (octal) device and function code 1070.

The macro-definition that causes the macro-processing program to carry out the translation illustrated is the following:

```
[@@@@ RADC &
```

```
%00%01%02%03 INA   '1070 &
     JMP   *-1
]
```

The significance of the various symbols of this macro-definition to the program is the following:

The symbol [marks the beginning of the macro-definition, and the symbol] the end.

The first line of the macro-definition is in essence a template against which each candidate line of source text is to be matched to determine whether it should be converted to the text of the subsequent lines.

& is a "rest" symbol, and will match any continuation of the source line.

Each @ will match any character in the corresponding location of the given source line.

If the remainder of the given line of text matches the template exactly (in the present instance, if the 5th through the 10th characters of the given line are " RADC "), then the macro-processing program will substitute for the source line the second line (if there is one) and all subsequent lines of the macro-definition (except the final]).

In the process, whatever string of characters was taken to match the & of the template will be inserted for any & that may appear in the subsequent line of the macro-definition.

Whatever character matched the 1st @ of the template will be inserted in place of %01, etc.

The program operates on a source file of text, containing both the macro-definitions and the text which may be converted. It produces another file of text from which the macro-definitions have been deleted and in which the macro-instructions have been appropriately converted.

The program operates as follows:

The source text is read one line at a time. If the line read starts with [, then it and subsequent lines are placed in a table of macro-definitions until the end macro symbol] is read.

Any other input line is compared against all the templates of the macro-definitions in turn. If it matches none, the line is added to the output file. If the line is found to match a template, then it is converted as described above and placed into a stack - a pushdown storage table whose unit entry is one line of text.

The program then reads and processes the stack instead of the source file until the stack is empty. If the current top line of the stack does not match any macro-definition template, it is removed from the stack and placed on the bottom of the output file. If it does match, it is converted and the converted form is placed on top of the stack in its place. The program returns to reading from the source file only after the stack has been emptied.

The following two properties of this manner of operation should be noted: (1) A source statement translated by one macro may result in one or more statements that themselves are further translated by other macros or even by the same one. (2) Each statement that is a candidate for translation is compared with all the macro-definitions preceding it in the source text in the order in which they occur. This allows macros to be used in combinations.

The implications of these properties are illustrated by the pair of macros that transform one line of text of arbitrary length and of the form

```
NAME >MOCT<12345,2345,...,670
```

into the set of data-defining assembly-language statements

```
NAME OCT   12345
     OCT   2345
      .
      .
      .
     OCT   670
```

The macro definition that effects this translation are these:

```
[@@@@ >MOCT<@@@@@,&
```

```
%00%01%02%03  OCT   %04%05%06%07%08
     >MOCT<&
]
[@@@@>MOCT<&
%00%01%02%03  OCT   &
]
```

Assume that the source text contains a line

```
NAME >MOCT< 123  ,456  ,70
```

Then this will match the template of the first macro-definition and will be converted to

```
NAME OCT  123

     >MOCT<456,70
```

which will be placed in the stack. Note that it would also have matched the template of the second macro-definition; but this comparison would not be performed. The first line now in the stack matches neither macro, and is therefore placed at the bottom of the output file. The second line again matches the first macro, and is converted to

```
OCT  456
>MOCT<70
```

in the stack.

The first line is again transferred to the output file. The second now does not match the template of the first macro-definition (there is no comma), but does match the second and is converted to

```
OCT  70
```

which is ultimately placed at the bottom of the output file.

5.2 MACROS DEVELOPED

The following are the macros that have been developed thus far.

5.2.1 5ARGA

The set of macros invoked by

5ARGA

serve to transform source statement sequences of fields of varying lengths into sequences of fields of a fixed format. Generally, other macros are then invoked to produce the final assembly language text. These can be much simpler if they can operate on statements of a fixed format rather than operating on statements in any of a number of formats.

The 5ARGA formatting macros transform statements of the form

NAME_XXXXXX5ARGA(sequence of varying fields)#

into statements of the form

NAME_XXXXXX(sequence of fixed fields)

The source sequence of varying fields may contain an arbitrary number of fields of the form

A,A*,BB,BB*,CCC,CCC*,DDDD,DDDD*,E

which will be converted into the fixed format sequence

```
A   ,A   *,BB  ,BB  *,CCC ,CCC *,DDDD ,DDDD*,E
```

The delimiters between the fields may be either commas as shown or left parenthesis.

The first four macro-definitions of the set insert the special symbol Λ before each field delimiter (i.e. command and left parenthesis), and insert Λ followed by a blank before the end symbol #.

The operation of the macros may be understood by considering the line resulting from successive applications of the macros to the input line

NAME UVWXYZ5ARGAAB,C*#

By the fourth macro this becomes first

NAME UVWXYZ5ARGAB,C*#A

and then

NAME UVWXYZ5ARGA,C*#AB

then by the second macro

NAME UVWXYZ5ARGAC*#ABΛ,

then again by the fourth macro

NAME UVWXYZ5ARGA*#ABΛ,C

and

NAME UVWXYZ5ARGA#ABΛ,C*

Finally, by the first macro

NAME UVWXYZ>5ARGA<ABΛ,C*Λ#

Had the delimiter (been present, the third macro would have performed the function analogous to that of the second.

It may be noted that further field delimiters may be simply established by adding macros with the appropriate characters replacing the commas in the second macro above. The symbol Λ preceding the delimiters identifies them as such for the subsequent macros.

The set of nine macros

>5ARG<

expand the fields to the proper length. A separate macro is provided for expanding each field of 1,2,3, or 4 characters with and without a star.

Continuing the example

NAME UVWXYZ>5ARGA<ABΛ,C*Λ#

becomes

NAME UVWXYZ>5ARGA<C*Λ#AB   ,

by virtue of macro 8, then

NAME UVWXYZ>5ARGA<#AB   ,C *

by macro 7, and finally,

NAME UVWXYZAB ,C *

by macro 5.

The relevant macros definitions are

```
1)  [@@@@ @@@@@@5ARGA#&
    %00%01%02%03 %04%05%06%07%08%09>5ARG<&⌢ #
    ]
2)  [@@@@ @@@@@@5ARGA,&
    %00%01%02%03 %04%05%06%07%08%095ARGA&⌢,
    ]
3)  [@@@@ @@@@@@5ARGA(&
    %00%01%02%03 %04%05%06%07%08%095ARGA&⌢(
    ]
4)  [@@@@ @@@@@@5ARGA@&
    %00%01%02%03 %04%05%06%07%08%095ARGA&%10
    ]
5)  [@@@@ @@@@@@>5ARG<#&
    %00%01%02%03 %04%05%06%07%08%09&
    ]
6)  [@@@@ @@@@@@>5ARG<@⌢@&
    %00%01%02%03 %04%05%06%07%08%09>5ARG<&%10    %11
    ]
7)  [@@@@ @@@@@@>5ARG<@*⌢@&
    %00%01%02%03 %04%05%06%07%08%09>5ARG<&%10   *%11
    ]
8)  [@@@@ @@@@@@>5ARG<@@⌢@&
    %00%01%02%03 %04%05%06%07%08%09>5ARG<&%10%11   %12
    ]
9)  [@@@@ @@@@@@>5ARG<@@*⌢@&
    %00%01%02%03 %04%05%06%07%08%09>5ARG<&%10%11  *%12
    ]
10) [@@@@ @@@@@@>5ARG<@@@⌢@&
    %00%01%02%03 %04%05%06%07%08%09>5ARG<&%10%11%12  %13
    ]
11) [@@@@ @@@@@@>5ARG<@@@*⌢@&
    %00%01%02%03 %04%05%06%07%08%09>5ARG<&%10%11%12 *%13
    ]
12) [@@@@ @@@@@@>5ARG<@@@@⌢@&
    %00%01%02%03 %04%05%06%07%08%09>5ARG<&%10%11%12%13 %14
    ]
13) [@@@@ @@@@@@>5ARG<@@@@@⌢@&
    %00%01%02%03 %04%05%06%07%08%09>5ARG<&%10%11%12%13%14%15
    ]
```

5.2.2 INTERACTIVE MACROS

The following macros provide for interactive operation of the system. The first reads operator input from the graphics tablet. The others allow interactive operations via the teletype.

5.2.2.1 Tablet

This macro generates a routine to read the coordinates of the stylus on the graphics tablet and skip the instruction immediately following if the stylus is depressed. A source statement of the form

```
NAME TABLET X,Y
```

is transformed to

```
NAME INA   '1151     READ TABLET X
     JMP   *-1
     LLR   3
     ARS   2
     STA   X         TABLET X
     INA   '1251     READ TABLET Y
     JMP   *-1
     LLR   3
     ARS   2
     STA   Y         TABLET Y
     IAB
     STA   *+2
     SKP
     BSS   1
     ANA   ='33
     SZE
     JMP   *-16      TABLET ERROR
     LDA   *-4
     ANA   ='44
     SZE             SKIP ON STYLUS DOWN
```

and a source statement of the form

```
NAME TABLET XAT*,YAT*
```

is transformed to

```
NAME INA   '1151     READ TABLET X
     JMP   *-1
     LLR   3
     ARS   2
     STA*  XAT       TABLET X
     INA   '1251     READ TABLET Y
     JMP   *-1
```

```
LLR   3
ARS   2
STA*  YAT   TABLET Y
IAB
STA   *+2
SKP
BSS   1
ANA   ='33
SZE
JMP   *-16  TABLET ERROR
LDA   *-4
ANA   ='44
SZE         SKIP ON STYLUS DOWN
```

The relevant macro definitions are

```
[@@@@ TABLET &
%00%01%02%03 >TAB< 5ARGA&#
]
[@@@@ >TAB< @@@@@,@@@@@&
%00%01%02%03 INA   '1151   READ TABLET X
     JMP   *-1
     LLR   3
     ARS   2
     STA%08  %04%05%06%07    TABLET X
     INA   '1251   READ TABLET Y
     JMP   *-1
     LLR   3
     ARS   2
     STA%13  %09%10%11%12    TABLET Y
     IAB
     STA   *+2
     SKP
     BSS   1
     ANA   ='33
     SZE
     JMP   *-16    TABLET ERROR
     LDA   *-4
     ANA   ='44
     SZE           SKIP ON STYLUS DOWN
]
```

5.2.2.2 Request And Requester

This set of macros provides for teletype output of messages requesting values for variables and for accepting values typed by the operator.

The source statement

REQUESTER

is transformed into the subroutine >RR<. Statements of the form

NAME REQUEST AB,CDE*

result in calls to subroutine >RR< of the form

```
*REQUEST AND READ AB
NAME JST*  *+4
     BCI   3,AB
     DAC   >RR<
     STA   AB
*REQUEST AND READ CDE*
     JST*  *+4
     BCI   3,CDE*
     DAC   >RR<
     STA*  CDE
```

Subroutine >RR< types the name of the variable requested followed by an equals sign. It then awaits input from the operator. It will accept, properly interpret, and store variable values typed in either as decimal numbers or as octal numbers. In keeping with the conventions of the DAP assembly language, octal numbers must be identified as such by a preceding apostrophe. The significance of a star following a variable name in a source statement (and subsequently the typed request) is that the named location contains the machine address in which the new value is to be stored.

The relevant macro definitions are

```
[@@@@ REQUEST &
%00%01%02%03 >RQST<5ARGA&#
]
[@@@@ >RQST<
]
[@@@@ >RQST<@@@@@@&
* REQUEST AND READ %04%05%06%07%08
%00%01%02%03 JST*   <RR>
      BCI    3,%04%05%06%07%08
      STA%08  %04%05%06%07
      >RQST<&
```

```
]
[REQUESTER
* PARAMETER REQUEST AND READ ROUTINE
        BCI    1,=
        OCT    '177531  ='
>RR<    DAC    **
        LDA    ='212
        JST*   '145
        LDA    >RR<
        STA    *+5
        ADD    =4
        STA    >RR<
        JST*   '152
        DEC    -3
        DAC    **
        JST*   '152
        DEC    -1
        DAC    >RR<-2
        JST*   '147
        SNZ
        SKP
        JMP*   >RR<
        IAB
        CAS    >RR<-1
        SKP
        JMP    *+3
        CRA
        SKP
        JST*   '146
        JMP*   >RR<
        FIN
]
```

5.2.2.3 Macros to Invoke Teletype Routines

The following set of macros generate assembly language statement that include jumps to system teletype routines:

```
NAME SPACE
```

is converted to

```
NAME LDA = '240
     JST*  '145
```

This results in a space being typed by the teletype,

```
NAME RETC
```

is converted to

```
NAME LDA = '212
     JST*  '145
```

This results in a teletype carriage return.

```
NAME QMARK
```

is converted to

```
NAME LDA = '277 QUEST MK
     JST*  '145
```

which results in a question mark being typed by the teletype.

```
NAME WOCT A
```

is converted to

```
NAME LDA = '247
     JST* '145 TYPE '
     LDA   A
     JST   '153 TYPE OCTAL
```

and

```
NAME WOCT* A
```

to

```
NAME LDA = '247
     JST* '145 TYPE '
     LDA* A
     JST* '153 TYPE OCTAL
```

This code results in the contents of the symbolic location A (WOCT) or the contents of the location whose address is in location A (WOCT*) being typed as an octal number preceded by an apostrophe.

Similarly, the source statements

```
ONE WDEC A

TWO WDEC* B
```

are converted to

```
ONE  LDA   A
     JST*  '154  TYPE DECIMAL
TWO  LDA*  B
     JST*  '154  TYPE DECIMAL
```

which types the contents of the respective locations as decimal numbers.

Finally, a source statement of the form

```
NAME TYPE TELETYPE MESSAGE
```

is transformed to

```
*TYPE 'TELETYPE MESSAGE'
NAME JST*   '152
     DEC    -09
     DAC    *+2
     JMP    *+1+09
     BCI    09,TELETYPE MESSAGE
```

which results in the text

```
TELETYPE MESSAGE
```

being typed by the teletype.

```
[@@@@ SPACE
%00%01%02%03 LDA    ='240
     JST*   '145
]
[@@@@ RETC
%00%01%02%03 LDA    ='212
     JST*   '145
]

[@@@@ QMARK
%00%01%02%03 LDA    ='277  QUEST MK
     JST*   '145

[@@@@ TYPE &
* TYPE '&'
%00%01%02%0301234567890123>TYP<& ‸
]
[@@@@@@@@@@@@@@@@@@>TYP<‸&
%00%01%02%03%04123456789%14123>TYP< ‸&
]
[@@@@@@@@@@@@@@@@@@>TYP<@‸&
%00%01%02%03 JST*   '152
     DEC    -%14%04
     DAC    *+2
     JMP    *+1+%14%04
     BCI    %14%04,&
```

```
]
[@@@@9012345678@@@@>TYP<@@&
%00%01%02%030123456789%05%06%07%04>TYP<&%08%09
]
[@@@@@@@@@@@@@@@@@@@>TYP<@@&
%00%01%02%03%05%06%07%08%09%10%11%12%13%04%14%15%16%17>TYP<&%18%19
]
[@@@@ TOP
%00%01%02%03 JMP*  '156  RETURN TO TOP
]
[@@@@ WOCT@ &
%00%01%02%03 LDA    ='247
     JST*  '145  TYPE '
     LDA%04  &
     JST*  '153  TYPE OCTAL
]
[@@@@ WDEC@ &
%00%01%02%03 LDA%04  &
     JST*  '154  TYPE DECIMAL
]
```

5.2.3 MACROS FOR SUBROUTINE CALLS

The following macros facilitate the calling of subroutines. LINKS TO allows calling a subroutine from any sector, CALL calls a subroutine, FUNCTION transfers values to a subroutine, and EXIT exits a subroutine.

5.2.3.1 LINKS TO

This macro results in assembly language statements defining subroutine links. It assumes, consistent with the conventions we have adopted in our programming, that a subroutine name will consist of exactly three characters, and that the symbolic addresses of the links are those three letters followed by the equals sign.

The source statements of the form

```
LINKS TO ABC,DEF, ...XYZ
```

are transferred to

```
* SUBROUTINE LINKS
     ORG   '400
ABC= DAC   ABC
DEF= DAC   DEF
```

```
XYZ= DAC    XYZ
TEMP BSS    1
```

TEMP defines a temporary storage location common for a number of subroutines. The relevant macro-definitions are:

```
[LINKS TO &
* SUBROUTINE LINKS
     ORG    '400
>LNKS<&
]
[>LNKS<@@@
%00%01%02= DAC    %00%01%02
TEMP BSS    1
]
[>LNKS<@@@,&
%00%01%02= DAC    %00%01%02
>LNKS<&
]
```

5.2.3.2 CALL

This set of macros generates a subroutine calling sequence. The overall effect is to transform a source statement of the form

```
NAME CALL   SUB,ABC,D*,17
```

into the assembly language sequence

```
* CALL   SUBROUTINE SUB
NAME JST    SUB
     DAC    ABC
     DAC*   D
     DAC    17
```

or a source statement of the form

```
     CALL   SUB=*,A,B
```

into

```
* CALL SUBROUTINE SUB
     JST*   SUB=
     DAC    A
     DAC    B
```

The relevent macro definitions are

```
[@@@@ CALL &
%00%01%02%03 >CAL< 5ARGA&#
]
```

```
[@@@@ >CAL< @@@@ @&
* CALL SUBROUTINE %04%05%06%07
%00%01%02%03 JST   %04%05%06%07
     >DAC< &

]
[@@@@ >DAC<
]
[@@@@ >DAC< @@@@@@&
%00%01%02%03 DAC%08   %04%05%06%07
     >DAC< &
]
```

5.2.3.3 FUNCTION

This set of macros defines storage locations within a subroutine and transfers the calling parameters to the storage locations so defined.

A source statement of the form

```
SUBR FUNCTION A,BC,DEF
```

results in

```
A    BSS   1
BC   BSS   1
DEF  BSS   1
SUBR DAC*  **
     LDA*  SUBR
     STA   A
     IRS   SUBR
     LDA*  SUBR
     STA   BC
     IRS   SUBR
     LDA*  SUBR
     STA   DEF
     IRS   SUBR
```

When executed, the resulting program segment transfers the values of the calling parameters in the subroutine calling sequence to their assigned storage locations within the subroutine.

The relevant macro-definitions are

```
[@@@@ FUNCTION &
%00%01%02%03 >XFA< 5ARGA&#
]
```

```
[@@@@ >XFA< &
     >DEF< &
%00%01%02%03 DAC*  **
%00%01%02%03 >XFB< &#
]
[@@@@ >XFB< #
]
[@@@@ >XFB< @@@@ @&
     LDA*   %00%01%02%03
     STA    %04%05%06%07
     IRS    %00%01%02%03
%00%01%02%03 >XFB< &
]
```

5.2.3.4 EXIT

This macro will provide an exit for a subroutine which gets its calling parameters using the macro FUNCTION. EXIT will store the return address (without an indirect flag) in a temporary location, and then jump to the return address.

A source statement of the form

```
NAME EXIT SUBR
```

will become

```
NAME LDA   SUBR
     ANA   ='37777
     STA   TEMP
     JMP*  TEMP
     FIN
```

The relevant macro definition is

```
[@@@@ EXIT  @@@@&
%00%01%02%03 LDA  %04%05%06%07  &
     ANA   = '37777
     STA   TEMP
     JMP*  TEMP
     FIN
]
```

5.2.4 DISK FILE HANDLING MACROS

The following macros are provided for calling the system subroutines to open close, read and write disk files:

A source statement of the form

```
NAME OPEN  N,NAME
```

where N = 1-4 and NAME is the storage location of the 6-character file name is transformed to

```
* OPEN FILE FOR READING AND WRITING
NAME JST*  '162
     DEC   3
     DAC   NAME LOC OF FILE NAME
     DEC   N   UNIT NUMBER
```

A source statement of the form

```
NAME CLOSE N
```

is transformed to

```
* CLOSE UNIT N
NAME JST*  '162
     DEC   4
     BSS   1
     DEC   N
```

and the statement

```
NAME CLOSE ALL
```

is transformed to

```
* CLOSE ALL FILES
     LDA   =-4
     STA   *+5
     LDA   =1
     STA   *+4
* CLOSE UNIT *
     JST*  '162
     DEC   4
     BSS   1
     DEC   *
     IRS   *-1
     IRS   *-3
     JMP   *-6
```

The following two statements result in reading a fixed number of words from the file associated with a given unit into the successive core location beginning at LOC, or of writing a fixed number of consecutive words commencing at LOC into the file associated with the given unit.

The source statement

```
NAME FETCH 1,LOC,17
```

is transformed to

```
* READ FROM DISK
NAME JST*  '160
     DEC   1      UNIT
     DAC   LOC    CORE LOC
     DEC   17     NR OF WORDS
```

and a statement of the form

```
NAME STORE 1,LOC,17
```

is transformed to

```
* WRITE ON DISK
NAME JST*  '161
     DEC   1      UNIT
     DAC   LOC    CORE LOC
     DEC   17     NR OF WORDS
```

The following two macros serve to read into core or store on disk pictures stored in standard format

```
NAME FILE PICTURE 1,PICT
```

is transformed to

```
* FILE ON DISK PICTURE PICT
NAME LDA*  *+9
     MPY*  *+5
     IAB          PRODUCT TO A
     ADD   =2
     STA   *+6
     SKP
     DAC   1+PICT

* WRITE ON DISK
     JST*  '161
     DEC   1      UNIT
     DAC   PICT   CORE LOC
     DEC   0      NR OF WORDS
```

and a statement of the form

NAME RECALL 1,PICT

is transformed to

```
* RECALL FROM DISK PICTURE PICT
* READ FROM DISK
NAME JST*  '160
     DEC   1      UNIT
     DAC   PICT   CORE LOC
     DEC   2      NR OF WORDS
*COMPUTE PICTURE SIZE
     LDX   *+9
     LDA   0,1
     IRS   0
     MPY   0,1
     IAB          PRODUCT TO A
* READ FROM DISK
     JST*  '160
     DEC   1      UNIT
     DAC   *      CORE LOC
     DEC   0      NR OF WORDS
```

The last two macros of this set serve to set up code to skip over a portion of an open file.

A source statement of the form

NAME SKIP 1,17

is transformed to

```
* SKIP (17) WORDS ON DISK UNIT 1
NAME LDA   17
     TCA
     STA   *+8
* READ FROM DISK
     JST*  '160
     DEC   1      UNIT
     DAC   *+6    CORE LOC
     DEC   1      NR OF WORDS
     IRS   *+3
     JMP   *-5
     JMP   *+3
     BSS   2
```

and a source statement of the form

```
NAME SKIP PICTURE 2
```

is transformed to

```
* SKIP OVER PICTURE ON DISK UNIT 2
* READ FROM DISK
NAME JST*   '160
     DEC    2      UNIT
     DAC    *+17   CORE LOC
     DEC    2      NR OF WORDS
     LDA    *+15
     TCA
     STA    *+13
     LDA    *+13
     TCA
     STA    *+13
* READ FROM DISK
     JST    '160
     DEC    2      UNIT
     DAC    *+10   CORE LOC
     DEC    1      NR OF WORDS
     IRS    *+7
     JMP    *-5
     IRS    *+3
     JMP    *-10
     JMP    *+5
     BSS    4
```

The relevent macro definitions are:

```
[@@@@ OPEN  @,&
* OPEN FILE FOR READING AND WRITING
%00%01%02%03 JST*  '162
     DEC   3
     DAC   &  LOC OF FILE NAME
     DEC   %04   UNIT NUMBER
]
[@@@@ CLOSE @

* CLOSE UNIT %04
%00%01%02%03 JST*  '162
     DEC   4
     BSS   1
     DEC   %04
]
```

```
[@@@@ FETCH @,&
* READ FROM DISK
%00%01%02%03 JST*  '160
      DEC    %04      UNIT
      >RWD< 5ARGA&#
]
[@@@@ STORE @,&
* WRITE ON DISK
%00%01%02%03 JST*  '161
      DEC    %04      UNIT
      >RWD< 5ARGA&#
]
[       >RWD< @@@@ ,&
      DAC    %00%01%02%03  CORE LOC
      DEC    &NR OF WORDS
]
[@@@@ FILE PICTURE @,&
* FILE ON DISK PICTURE &
%00%01%02%03 LDA*  *+9
      MULT*  *+5
      ADD    =2
      STA    *+6
      SKP
      DAC    1+&
      STORE %04,&,0
]
[@@@@ RECALL @,&
* RECALL FROM DISK PICTURE &
%00%01%02%03 FETCH %04,&,2
* COMPUTE PICTURE SIZE
      LDX    *+9
      LDA    0,1
      IRS    0
      MULT   0,1
      IRS    0
      STA    *+7
      STX    *+5
      SKP
      DAC    &
      FETCH %04,*,0
]
[@@@@ CLOSE ALL
* CLOSE ALL FILES
      LDA    =-4
      STA    *+5
      LDA    =1
      STA    *+4
      CLOSE *
      IRS    *-1
```

```
        IRS    *-3
        JMP    *-6
]
[@@@@ SKIP PICTURE @
* SKIP OVER PICTURE ON DISK UNIT %04
%00%01%02%03 FETCH %04,*+17,2
        LDA    *+15
        TCA
        STA    *+13
        LDA    *+13
        TCA
        STA    *+12
        FETCH  %04,*+10,1
        IRS    *+7
        JMP    *-5
        IRS    *+3
        JMP    *-10
        JMP    *+5
        BSS    4
]
[@@@@ SKIP  @,&
* SKIP (&) WORDS ON DISK UNIT %04
%00%01%02%03 LDA    &
        TCA
        STA    *+8
        FETCH  %04,*+6,1
        IRS    *+3
        JMP    *-5
        JMP    *+3
        BSS    2
]
```

5.2.5 TRIM DEFLECTION SIGNAL MACROS

These translate mnemonics for the various output instructions to transmit signals to the TRIM scanner. In each case, the assembly language instructions generated will send the contents of the A-register and an appropriate function code to the scanner. The following are provided

```
[@@@@ XS
%00%01%02%03 OTA    '52    X, SHORT
      JMP    *-1
]
[@@@@ XM
%00%01%02%03 OTA    '152   X, MEDIUM
      JMP    *-1
]
```

```
[@@@@ XL
%00%01%02%03 OTA    '252  X, LONG
      JMP    *-1
]
[@@@@ XSG
%00%01%02%03 OTA    '352  X, SHORT, GO
      JMP    *-1
]
[@@@@ XMG
%00%01%02%03 OTA    '452  X, MEDIUM, GO
      JMP    *-1
]
[@@@@ XLG
%00%01%02%03 OTA    '552  X, LONG, GO
      JMP    *-1
]
[@@@@ YS
%00%01%02%03 OTA    '652  Y, SHORT
      JMP    *-1
]
[@@@@ YM
%00%01%02%03 OTA    '752  Y, MEDIUM
      JMP    *-1
]
[@@@@ YL
%00%01%02%03 OTA    '1052 Y, LONG
      JMP    *-1
]
[@@@@ YSG
%00%01%02%03 OTA    '1152 Y, SHORT, GO
      JMP    *-1
]
[@@@@ YMG
%00%01%02%03 OTA    '1252 Y,MEDIUM, GO
      JMP    *-1
]
[@@@@ YLG
%00%01%02%03 OTA    '1352 Y, LONG, GO
      JMP    *-1
]
[@@@@ Z
%00%01%02%03 OTA    '1452 Z
      JMP    *-1
]
[@@@@ ZG
%00%01%02%03 OTA    '1552 Z, GO
      JMP    *-1
]
```

5.2.6 LOCATION DEFINING MACROS

The following macros define storage locations. DEFINE defines BSS locations. MOCT,MDEC, and MDAC define multiple OCT,DEC, and DAC locations.

5.2.6.1 DEFINE

The set of macros invoked by DEFINE serve to create a set of storage-defining assembly language statements.

A source statement of the form

```
DEFINE A,BCDE,FGH(2),I(89),J
```

ultimately results in

```
A    BSS   1
BCDE BSS   1
FGH  BSS   2
I    BSS   89
J    BSS   1
```

The macros SQUASH) which appear at an intermediate stage serve to delete the right parenthesis and any superfluous spaces that would otherwise interfere with the proper operation of the set of macros 5ARGA.

The relevant macro definitions are

```
[DEFINE &
SQUASH) &#
]
[SQUASH) #&
     >DEF< 5ARGA&#
]
[SQUASH) )&
SQUASH) &
]
[SQUASH)  &
SQUASH) &
]
[SQUASH) @&
SQUASH) &%00
]
[      >DEF< @@@@ ,&
%00%01%02%03 BSS    1
      >DEF< &
]
```

```
[       >DEF< @@@@ (@@@@@,&
%00%01%02%03 BSS    %04%05%06%07%08
     >DEF< &
]
[       >DEF< @@@@
%00%01%02%03 BSS    1
]
[       >DEF< @@@@ (@@@@@
%00%01%02%03 BSS    %04%05%06%07%08
]
```

5.2.6.2 MOCT,MDEC

These macros result in assembly language statements that define constants.

Source statements of the form

```
NAME MOCT 1,22,777
```

become

```
NAME OCT   1
     OCT   22
     OCT   777
```

and source statements of the form

```
NAME MDEC 1,22,999
```

become

```
NAME DEC   1
     DEC   22
     DEC   999
```

The relevant macro - definitions are:

```
[@@@@ MOCT  &
%00%01%02%03 >MOCT<5ARGA&#
]
[@@@@ MDEC  &
%00%01%02%03 >MDEC<5ARGA&#
]
[@@@@ >MOCT<@@@@@
%00%01%02%03 OCT    %04%05%06%07%08
]
```

```
[@@@@ >MOCT<@@@@@,&
%00%01%02%03 OCT   %04%05%06%07%08
     >MOCT<&
]
[@@@@ >MDEC<@@@@@
%00%01%02%03 DEC   %04%05%06%07%08
]
[@@@@ >MDEC<@@@@@,&
%00%01%02%03 DEC   %04%05%06%07%08
     >MDEC<&
]
```

5.2.6.3. MDAC

MDAC will form a series of DAC's. A source statement of the form:

```
NAME MDAC  X,YY*,ZZZZ
```

becomes

```
NAME DAC  X
     DAC* YY
     DAC  ZZZZ
```

The relevent macro definitions are:

```
[@@@@ MDAC  &
%00%01%02%03 >MDAC<5ARGA&#
]
[@@@@ >MDAC<@@@@@
%00%01%02%03 DAC%08  %04%05%06%(
]
[@@@@ >MDAC<@@@@@,&
%00%01%02%03 DAC%08  %04%05%06%0
     >MDAC<&
]
```

5.2.7 ARITHMETIC AND LOGICAL OPERATIONS

The following macros allow often-used arithmetic and logical operations to be called conveniently

5.2.7.1 DVDE,MULT

The machine instruction

```
        DIV   QUOT
```

has the effect of dividing the contents of the A and B registers considered as one double-precision number by QUOT

The macro

```
NAME DVDE QUOT
```

results in code

```
NAME CSA              SET UP DIVIDE
     IAB                    .
     CRA                    .
     SRC                    .
     CMA                    .
     DIV   QUOT
```

which first takes the single-precision number in the A register and expands it into the form of a double precision number in the combined A and B registers and then performs the division.

The machine instruction to multiply

```
        MPY   FCTR
```

multiplies the contents in the A register by the number at location FCTR and leaves the product as a double precision number in the combined A and B registers.

A source statement of the form

```
NAME  MULT  FCTR
```

results in

```
NAME MPY   FCTR
           IAB      PRODUCT TO A
```

This has the effect of reducing the product to a single-recision number in the A register.

The relevant macro definitions are

```
[@@@@ DVDE@ &
%00%01%02%03 CSA            SET UP DIVIDE
     IAB               :
     CRA               :
     SRC               :
     CMA               :
     DIV%04   &
]
[@@@@ MULT@ &
%00%01%02%03 MPY%04   &
     IAB             PRODUCT TO A
]
```

5.2.7.2 ABS

This short macro finds the absolute value of the number in the A-Register.

The source statement

```
NAME ABS
```

becomes

```
NAME SPL
     TCA
```

The relevant macro definition is

```
[@@@@ ABS
%00%01%02%03 SPL
     TCA
]
```

5.2.7.3 ORA

There is no machine instruction that performs the logical "inclusive or" operation. The macro ORA constructs code to leave in the accumulator the result of a logical inclusive OR of the two specified variables.

Source statement of the form

```
NAME ORA   X,Y
NEXT ORA   V*,W*
```

are transformed to

```
NAME LDA    X      INCLUSIVE OR
     ANA    W          :
     ERA    X          :
     ERA    W          :
NEXT LDA*   Y      INCLUSIVE OR
     ANA*   Z          :
     ERA*   Y          :
     ERA*   Z          :
```

The relevant macro definitions are

```
[@@@@ ORA   &
%00%01%02%03 >ORA< 5ARGA&#
]
[@@@@ >ORA< @@@@@,@@@@@
%00%01%02%03 LDA%08  %04&05%06%07    INCLUSIVE OR
         ANA%13  %09%10%11%12         :
         ERA%08  %04%05%06%07         :
         ERA%13  %09%10%11%12         :
]
```

5.2.8 MISCELLANEOUS MACROS

The DELAY macro inserts a delay in the program. The MATCH macro tests two strings of text for a match. The SLASH macro allows easy input of programs by providing a facility for inserting tabs.

5.2.8.1 DELAY

This macro will put in a delay loop of any multiple of 20 macroseconds, if sense-switch 4 is up.

The source statement

```
NAME DELAY 50
```

becomes

```
* DELAY 20*50 MICROSECONDS IF SS4 IS UP
NAME SS4
     JMP   *+11
     STA   *+8
     LDA   *+8
     AOA
     LLR   32
     SPL
```

```
      JMP    *-3
      LDA    *+2
      JMP    *+3
      BSS    1
      DEC    -50
    * * * * * * * *
```

The relevant macro definition is

```
[@@@@ DELAY &
* DELAY 20*& MICROSECONDS IF SS4 IS UP
%00%01%02%03 SS4
      JMP    *+11
      STA    *+8
      LDA    *+8
      AOA
      LLR    32
      SPL
      JMP    *-3
      LDA    *+2
      JMP    *+3
      BSS    1
      DEC    -&
* * * * * * * *
]
```

5.2.8.2 MATCH

This macro generates a call to the subroutine MCH, which compares two strings of text. If they are equal, the subroutine return is to the location immediately following the call, if not, to the next subsequent location.

A source statement of the form

```
NAME MATCH 5,TEXT,SOMETHING
```

is converted to:

```
NAME JST*  MCH=  TEXT COMP. - SKP IF N.E.
     DEC   -5
     DAC   TEXT
     BCI   5,SOMETHING
```

This results in the comparison of the string of characters "SOMETHING" with the characters stored in 5 consecutive words starting with location TEXT.

The relevant macro definitions are

```
[@@@@ MATCH @,@@@@@&
%00%01%02%03 JST*  MCH=  TEXT COMP. - SKP IF N.E.
     DEC   -%04
     >TRNK<5ARGA%05%06%07%08%09XXXXX#
     >TXT<%04%05%06%07%08%09&
]
[@@@@ >TRNK<@@@@@&
%00%01%02%03 DAC%08  %04%05%06%07
]
[@@@@ >TXT<@,&
%00%01%02%03 BCI   %04,&
]
[@@@@ >TXT<@@&
%00%01%02%03 >TXT<%04&
]
```

5.2.8.3 SLASH

This macro facilitates the keying-in of a program on the teletype by providing tabs. The tabs are typed as slashes, and tab to columns 6,12, and 20. To allow the first entry to begin in column 1, an initial two slashes are used. Thus, source statements of the form

```
/OPC/ADDR/COMMENT
//NAME/OPC*/ADD+1/COMMENT2
//NAME/OPC
/OPC//COMMENT:Z=X/Y
```

are transformed to

```
     OPC   ADDR  COMMENT
NAME OPC*  ADD+1 COMMENT2
NAME OPC
     OPC         COMMENT:Z=x/Y
```

The pertinent macro definitions are

```
[//@/&
%00   ⌒&
]
[//@@/&
%00%01  ⌒&
]
[//@@@/&
%00%01%02 ⌒&
]
[//@@@@/&
%00%01%02%03⌒&
]
[/&
    ⌒&
]
[@@@@⌒@/&
%00%01%02%03 %04    ⌒&
]
[@@@@⌒@@/&
%00%01%02%03 %04%05   ⌒&
]
[@@@@⌒@@@/&
%00%01%02%03 %04%05%06  ⌒&
]
[@@@@⌒@@@@/&
%00%01%02%03 %04%05%06%07 ⌒&
]
[@@@@⌒&
%00%01%02%03 &
]
[@@@@@@@@@@⌒/&
%00%01%02%03%04%05%06%07%08%09          &
]
[@@@@@@@@@@⌒@/&
%00%01%02%03%04%05%06%07%08%09 %10        &
]
[@@@@@@@@@@⌒@@/&
%00%01%02%03%04%05%06%07%08%09 %10%11       &
]
[@@@@@@@@@@⌒@@@/&
%00%01%02%03%04%05%06%07%08%09 %10%11%12      &
]
[@@@@@@@@@@⌒@@@@/&
%00%01%02%03%04%05%06%07%08%09 %10%11%12%13     &
]
[@@@@@@@@@@⌒@@@@@/&
%00%01%02%03%04%05%06%07%08%09 %10%11%12%13%14    &
]
```

```
[@@@@@@@@@@_@@@@@@/&
%00%01%02%03%04%05%06%07%08%09 %10%11%12%13%14%15  &
]
[@@@@@@@@@@_@@@@@@@/&
%00%01%02%03%04%05%06%07%08%09 %10%11%12%13%14%15%16 &
]
[@@@@@@@@@@_&
%00%01%02%03%04%05%06%07%08%09 &
]
```

APPENDIX A

THE PROGRAMS

```
*
* DEMONSTRATION PROGRAM
*
*
*
*
*
* SUBROUTINE LINKS
      ORG    '400
ILC= DAC    ILC
CSS= DAC    CSS
CSD= DAC    CSD
DIS= DAC    DIS
PMD= DAC    PMD
PML= DAC    PML
STO= DAC    STO
OBL= DAC    OBL
DLS= DAC    DLS
DLN= DAC    DLN
D2S= DAC    D2S
D2N= DAC    D2N
SQR= DAC    SQR
SZC= DAC    SZC
CRL= DAC    CRL
CRD= DAC    CRD
CRG= DAC    CRG
AVG= DAC    AVO
*
*STORAGE LOCATION LINKS
REC= DAC    RECT
LOC= DAC    LOC
*
*
*
*
*
TEMP BSS    50
*
*
*
***SFW
*
*
* STYLUS FOLLOWER AND INTERLACED SCAN
*
* ALTERNATES INTERLACED SCAN WITH CROSS AT
*   LOCATION OF STYLUS.
* IF STYLUS IS UP, CROSS IS DISPLAYED BLINKING.
*
* CALLS CSS (CSD), AND ILC.
*
      ORG    '1000
      STRT
*SET CROSS
      JST    CSS
      DAC    SFN
      DAC    SFDL
```

```
         DAC   SFZ
         JMP   BGN
RQST NOP          ENTRY FOR PARAMETER REQUESTS
* REQUEST DISPLAY PARAMETERS
* REQUEST AND READ YDIS
         JST*  *+4
         BCI   3,YDIS
         DAC   >RR<
         STA   YDIS
* REQUEST AND READ XDIS
         JST*  *+4
         BCI   3,XDIS
         DAC   >RR<
         STA   XDIS
* REQUEST AND READ DDIS
         JST*  *+4
         BCI   3,DDIS
         DAC   >RR<
         STA   DDIS
* REQUEST SCAN PARAMETERS
* REQUEST AND READ YTOP
         JST*  *+4
         BCI   3,YTOP
         DAC   >RR<
         STA   YTOP
         STA   YTP2
* REQUEST AND READ YBOT
         JST*  *+4
         BCI   3,YBOT
         DAC   >RR<
         STA   YBOT
         STA   YBT2
* REQUEST AND READ XLFT
         JST*  *+4
         BCI   3,XLFT
         DAC   >RR<
         STA   XLFT
         STA   XLT2
* REQUEST AND READ XRGT
         JST*  *+4
         BCI   3,XRGT
         DAC   >RR<
         STA   XRGT
         STA   XRT2
* REQUEST AND READ DELT
         JST*  *+4
         BCI   3,DELT
         DAC   >RR<
         STA   DELT
* GET SCAN WIDTHS
         LDA   YTOP
         SUB   YBOT
         STA   YWDH
         ARS   1
         STA   YWD2
         LDA   XRGT
         SUB   XLFT
```

```
      STA    XWDH
      ARS    1
      STA    XWD2
      JMP    CILC
* RE-REQUEST ON SS1
BGN   SS1
      SKP
      JMP    RQST
* DIGITIZE AND DISPLAY ON SS2
      SR2
      JMP    DIGT
*CALL ILC
      SS3
      SKP
      JMP    AILC
      SS4
      SKP
      JMP    BILC
CILC  JST    ILC
      DAC    YTOP
      DAC    YBOT
      DAC    XLFT
      DAC    XRGT
      DAC    DELT
      JMP    RDTB
* BIG SCAN ON SS4
BILC  JST    ILC
      DAC    YTP2
      DAC    YBT2
      DAC    XLT2
      DAC    XRT2
      DAC    DEL2
      JMP    RDTB
* ALTERNATE LARGE AND SMALL SCAN ON SS3
AILC  LDA    ALT
      SZE
      JMP    DSSM
      AOA
      STA    ALT
      JST    ILC
      DAC    YTP2
      DAC    YBT2
      DAC    XLT2
      DAC    XRT2
      DAC    DEL2
      JMP    RDTB
DSSM  CRA
      STA    ALT
      JMP    CILC      DISPLAY SMALL
*READ TABLET
RDTB  CRA
      STA    UPDN      UP-DOWN SWT TO 0 (=DWN)
      INA    '1151     TABLET X
      JMP    *-1
      STA    TABX
      INA    '1251     TABLET Y
      JMP    *-1
```

```
      STA   TABY
      ANA   TABX
      SPL               TEST STYLUS
      JMP   STUP        STYLUS UP
TTAB  ALS   1           TEST TABLET
      SPL
      JMP   BGN         DATA NOT READY
      ALS   1
      SPL
      JMP   BGN         ERROR IN TABLET
      JMP   CCSS        CALL CROSS
STUP  STA   AND
      LDA   =1
      STA   UPDN        UP-DOWN SWT TO 1 (=UP)
      LDA   AND
      JMP   TTAB
*CALL CROSS
CCSS  LDA   TABX
      ALS   1
      ANA   ='3777
      STA   TABX
      LDA   TABY
      ALS   1
      ANA   ='3777
      STA   TABY
      JST   CSD
      DAC   TABX
      DAC   TABY
      LDA   UPDN        TEST SWITCH
      SZE
      JMP   BGN
* MOVE SCAN ON STYLUS DOWN
      LDA   TABX
      SUB   XWD2
      STA   XLFT
      SUB   XWDH
      SUB   XWDH
      STA   XLT2
      LDA   TABX
      ADD   XWD2
      STA   XRGT
      ADD   XWDH
      ADD   XWDH
      STA   XRT2
      LDA   TABY
      SUB   YWD2
      STA   YBOT
      SUB   YWDH
      SUB   YWDH
      STA   YBT2
      LDA   TABY
      ADD   YWD2
      STA   YTOP
      ADD   YWDH
      ADD   YWDH
      STA   YTP2
      LDA   DELT
```

```
      MPY   =6
      IAB
      STA   DEL2
      JMP   BGN
*
* CALL SUBROUTINE TO DIGITIZE, THEN TO DISPLAY
DIGT JST*   STO=
      DAC   YTOP
      DAC   YBOT
      DAC   XLFT
      DAC   XRGT
      DAC   DELT
      DAC   LOC
      LDA   =-7
      STA   DGCT
DIG2 JST*   AVG=
      DAC   YTOP
      DAC   YBOT
      DAC   XLFT
      DAC   XRGT
      DAC   DELT
      DAC   LOC
      IRS   DGCT
      JMP   DIG2
      JST*  DIS=
      DAC   YDIS
      DAC   XDIS
      DAC   DDIS
      DAC   LOC
      JMP   BGN
* PARAMETER REQUEST AND READ ROUTINE
      BCI   1,=
      OCT   '177531  ='
>RR< DAC   **
      LDA   ='212
      JST*  '145
      LDA   >RR<
      STA   *+5
      ADD   =4
      STA   >RR<
      JST*  '152
      DEC   -3
      DAC   **
      JST*  '152
      DEC   -1
      DAC   >RR<-2
      JST*  '147
      SNZ
      SKP
      JMP*  >RR<
      IAB
      CAS   >RR<-1
      SKP
      JMP   *+3
      CRA
      SKP
      JST*  '146
```

```
      JMP*   >RR<
      FIN
SFN   OCT    67
SFDL  OCT    1
SFZ   OCT    3777
UPDN  OCT    0
TABX  OCT    0
TABY  OCT    0
AND   OCT    0
YTOP  OCT    2050
YBOT  OCT    1730
XLFT  OCT    1730
XRGT  OCT    2050
DELT  OCT    1
CNTR  OCT    0
XWD2  OCT    50
YWD2  OCT    50
XWDH  OCT    120
YWDH  OCT    120
ALT   OCT    0
YTP2  OCT    2310
YBT2  OCT    1470
XLT2  OCT    1470
XRT2  OCT    2310
DEL2  OCT    6
YDIS  DEC    1424
XDIS  DEC    624
DDIS  DEC    10
DGCT  OCT    0
*
*
*
***ILC
*
* INTERLACED RASTER SCAN
*
* PERFORMS AN INTERLACED ZIG-ZAG RASTER SCAN FOR DISPLAY
*
* INPUT PARAMETERS:
*   YTOP,YBOT,XLFT,XRGT  :  SCAN BOUNDARIES
*   DELT : SPACING BETWEEN SCAN POINTS
*
* CALLED BY:
*   JST  ILC
*   DAC  YTOP
*   DAC  YBOT
*   DAC  XLFT
*   DAC  XRGT
*   DAC  DELT
*
*
ILC   DAC*   **
      LDA*   ILC
      STA    ILCT   YTOP
      IRS    ILC
      LDA*   ILC
      STA    ILCB   YBOT
```

```
        IRS    ILC
        LDA*   ILC
        STA    ILCL   XLFT
        IRS    ILC
        LDA*   ILC
        STA    ILCR   XRGT
        IRS    ILC
        LDA*   ILC
        STA    ILCD   DELT
        IRS    ILC
        LDA    ILCD
        ADD    ILCD
        STA    ILCV   VERTICAL DELTA = 2*DELT
        CRA
        STA    ILCI   INTERLACE SWITCH
        STA    ILCS   LEFT-RIGHT SWITCH
*
        JMP    ILCU   UPPER INTERLACE
ILCA    LDA    ILCI   INTERLACE SWITCH TEST
        SNZ
        JMP    ILCE   EXIT
        CRA           LOWER INTERLACE
        STA    ILCI
        LDA    ILCT
        SUB    ILCD
        JMP    *+4
ILCU    AOA           UPPER INTERLACE
        STA    ILCI
        LDA    ILCT   INITIALIZE Y
        STA    ILCY
        OTA    '752   LOAD Y MED
        JMP    *-1
        LDA    ILCL   INITIALIZE X
        STA    ILCX
        OTA    '452   LOAD X MED,GO
        JMP    *-1
        JMP    ILCM   BEGIN MOVING RIGHT
*
ILCN    SUB    ILCD   GET NEXT ROW
        STA    ILCX
        LDA    ILCY
        SUB    ILCV
        CAS    ILCB
        JMP    *+3
        SKP
        JMP    ILCA   END, BEGIN ANOTHER INTERLACE
        OTA    '1152 LOAD Y SHORT, GO
        JMP    *-1
        STA    ILCY
        LDA    ILCS   L-R SWITCH TEST
        SNZ
        JMP    ILCM   JUMP TO MOVE RIGHT
        CRA           MOVE LEFT
        STA    ILCS   SET L-R SWITCH OFF
        LDA    ILCX
*
ILC1    SUB    ILCD   LOOP 1 (GO LEFT)
```

```
      CAS   ILCL
      JMP   *+4
      JMP   *+3
      LDA   ILCL
      JMP   ILCN+1  END OF ROW
      OTA   '352  LOAD X SHORT, GO
      JMP   *-1
      JMP   ILC1
ILCM  AOA           MOVE RIGHT
      STA   ILCS  SET L-R SWITCH ON
      LDA   ILCX
ILC2  ADD   ILCD  LOOP 2 (GO RIGHT)
      CAS   ILCR
      JMP   ILCN  END OF ROW
      NOP
      OTA   '352  LOAD X SHORT, GO
      JMP   *-1
      JMP   ILC2
*
ILCE  LDA   ILC
      ANA   ='077777
      STA   ILCO
      JMP*  ILCO  EXIT OUT
ILCT  OCT   0     YTOP
ILCB  OCT   0     YBOT
ILCL  OCT   0     XLFT
ILCR  OCT   0     XRGT
ILCD  OCT   0     DELT
ILCI  OCT   0     INTERLACE SWITCH
ILCS  OCT   0     LEFT-RIGHT SWITCH
ILCV  OCT   0     VERTICAL DELTA
ILCX  OCT   0     SCAN X
ILCY  OCT   0     SCAN Y
ILCO  OCT   0     EXIT OUT
      FIN
*
*

*
*
*
***CSS
*
* SET-CROSS, DISPLAY-CROSS
*
*
*
*
* CSS SETS THE PARAMETERS FOR THE DISPLAY OF A CROSS,
*  I.E. A "+".
*    CSS IS CALLED BY:
*      CALL CSS,N,DELT,Z
*     WHERE
*        N = THE NUMBER OF DISPLAYED POINTS
*           IN THE HORIZONTAL (OR VERTICAL) LINE OF THE CROSS
*        DELT = THE SEPARATION BETWEEN DISPLAY POINTS
*        Z = THE INTENSITY OF THE CROSS DISPLAY
```

```
*     (NOTE:  DELT SHOULD NOT BE ODD WHEN N IS EVEN)
*
*
*
* CSD DISPLAYS ONCE A CROSS WHICH HAS BEEN SET UP BY CSS.
*      CSD IS CALLED BY:
*        CALL CSD,XMID,YMID
*       WHERE
*          (XMID,YMID) = THE MIDPOINT OF THE CROSS
*
*
*
*----------
*  CSS
*----------
*
CSN  BSS   1
CSDL BSS   1
CSZ  BSS   1
CSS  DAC*  **
     LDA*  CSS
     STA   CSN
     IRS   CSS
     LDA*  CSS
     STA   CSDL
     IRS   CSS
     LDA*  CSS
     STA   CSZ
     IRS   CSS
     LDA   CSN
     TCA
     STA   CSMN     -N
     LDA   CSN
     SUB   =1
     MPY   CSDL
     IAB
     ARS   1
     STA   CSLG     LEG LENGTH=(N-1)*DELT/2
     LDA   CSS
     ANA   ='37777
     STA   *+2
     JMP*  *+1
     BSS   1
*
*----------
*   CSD
*----------
*
CSXM BSS   1
CSYM BSS   1
CSD  DAC*  **
     LDA*  CSD
     STA   CSXM
     IRS   CSD
     LDA*  CSD
     STA   CSYM
     IRS   CSD
```

```
      LDA    CSYM
      OTA    '652     Y, SHORT
      JMP    *-1
      LDA    CSZ
      OTA    '1452      Z
      JMP    *-1
      LDX    CSMN     INDEX=-N
      LDA    CSXM
      SUB    CSLG     XLEFT=XMID-LEG
* HORIZONTAL LINE LOOP
CSHL  OTA    '352     X, SHORT, GO
      JMP    *-1
      ADD    CSDL
      IRS    0
      JMP    CSHL
      LDA    CSXM
      OTA    '52      X, SHORT
      JMP    *-1
      LDX    CSMN
      LDA    CSYM
      ADD    CSLG     YTOP=YMID+YLEG
*VERTICAL LINE LOOP
CSVL  OTA    '1152    Y, SHORT, GO
      JMP    *-1
      SUB    CSDL
      IRS    0
      JMP    CSVL
      LDA    CSD
      ANA    ='37777
      STA    *+2
      JMP*   *+1
      BSS    1
*
CSMN  OCT    0        -N
CSLG  OCT    0        LEG LENGTH
      FIN
*
*
*
*
*
*
***DIS
*
      ORG    '2000
*
*
* RASTER DISPLAY ROUTINE
* DISPLAYS PICTURE STORED IN STD FORM AT LOC
* CALLED BY
*        CALL  DIS,YTOP,XLEFT,DELT,LOC
* EXITED IF SS2 IS DOWN
*
DIY   BSS    1
DIX   BSS    1
DID   BSS    1
DIS   DAC*   **
```

```
      LDA*  DIS
      STA   DIY
      STA   DIYD
      IRS   DIS
      LDA*  DIS
      STA   DIX
      STA   DIXD
      IRS   DIS
      LDA*  DIS
      STA   DID
      IRS   DIS
      LDA   DIS
      ANA   ='37777
      STA   DIEX
      LDA*  DIEX
      STA   DISS
      LDA   =-1
      STA   DISW
      CRA
      STA   DIOX
      STA   DIOY
DIAG  LDX   DISS
      LDA   0,1
      TCA
      STA   DIR
      LDA   1,1
      TCA
      STA   DIC
* DISPLAY RASTER
DIA   LDA   DISW
      CAS   =-1
      JMP   DIA2
      SKP
      JMP   DIA2
      LDA   DIOX      TEST OLD X BOUNDS
      CAS   =600
      CAS   =1500
      NOP
      JMP   *+4       IF OUT, NO CSD
      JST*  CSD=
      DAC   DIOX
      DAC   DIOY
      CRA
      STA   DIST      STYLUS SWITCH
      INA   '1151     READ TABLET X
      JMP   *-1
      LLR   3
      ARS   2
      STA   DITX      TABLET X
      INA   '1251     READ TABLET Y
      JMP   *-1
      LLR   3
      ARS   2
      STA   DITY      TABLET Y
      IAB
      STA   *+2
      SKP
```

```
      BSS   1
      ANA   ='33
      SZE
      JMP   *-16     TABLET ERROR
      LDA   *-4
      ANA   ='44
      SZE            SKIP ON STYLUS DOWN
      IRS   DIST
      LDA   DITX     TEST TABX BOUNDS
      CAS   =600
      CAS   =1500
      NOP
      JMP   DIA2
      LDA   DITY     TEST TABY BOUNDS
      CAS   =600
      CAS   =1500
      NOP
      JMP   DIA2
      JST*  CSD=     CROSS DISPLAY
      DAC   DITX
      DAC   DITY
      LDA   DIST
      SZE
      JMP   DIA2
      LDA   DITX
      STA   DIOX
      SUB   DIX
      IAB
      CRA
      DIV   DID
      AOA
      STA   XCNR
      LDA   DITY
      STA   DIOY
      LDA   DIY
      SUB   DITY
      IAB
      CRA
      DIV   DID
      ADD   DIR    -NO.ROWS
      TCA
      STA   YCNR
      JST*  CSD=     CROSS DISPLAY AGAIN
      DAC   DITX
      DAC   DITY
* GO THROUGH RASTER
DIA2  LDX   DISS
      LDA   DIR
      STA   DIRN
      LDA   DIY
      STA   DIYN
DINR  LDA   DIYN
      OTA   '652  Y, SHORT
      JMP   *-1
      SUB   DID
      STA   DIYN
      LDA   DIX
```

```
      IAB
      LDA   DIC
      STA   DICN
DINP  IAB
      NOP
      OTA   '52    X, SHORT
      JMP   *-1
      ADD   DID
      IAB
      LDA   2,1
      OTA   '1552 Z, GO
      JMP   *-1
      IRS   0
      IRS   DICN
      JMP   DINP
      IRS   DIRN
      JMP   DINR
      SR1
      JMP   DICP
      SS2
      JMP   *+6
      LDA   DISW
      CAS   =-2      ROAD EDGE TEST
      JMP   DIA
      JMP   DIRQ
      JMP   DIA
      IRS   DIEX
      JMP*  DIEX
* CALL CRG (CRL) ON SS1, SWT=2
* CALL PMD AND PML ON SS1, SWT=0
* CALL OBL ON SS1, SWT=1
* CALL DIS FOR LOC ON SS1, SWT=-1
* CALL CRG FOR ROAD EDGE ON SS1, SWT =-2
*
* REQUEST AND READ SWT
DICP  JST*  *+4
      BCI   3,SWT
      DAC   >RR<
      STA   DISW
      CAS   =1
      JMP   DICR      CRGT ON SWT=2 OR MORE
      JMP   DIOB      SWT ON =1
      CAS   =-1
      JMP   DIPT     PRINT ON =0
      SKP            DIS WITH LOC ON =-1
      JMP   DICR      ROAD EDGE CRG ON =-2
      LDA   LOC=
      STA   DISS
      LDA   DIXD
      STA   DIX
      LDA   DIYD
      STA   DIY
      CRA
      STA   DIOX
      STA   DIOY
      JMP   DIAG
DIPT  JST*  PMD=     PRINT ON =0
```

```
      DAC    LOC
      JST*   PML=
      DAC    LOC
      JMP    DIA
* REQUEST AND READ DXTP
DIOB JST*    *+4
      BCI    3,DXTP
      DAC    >RR<
      STA    DXTP
* REQUEST AND READ DYTP
      JST*   *+4
      BCI    3,DYTP
      DAC    >RR<
      STA    DYTP
* REQUEST AND READ NROW
      JST*   *+4
      BCI    3,NROW
      DAC    >RR<
      STA    NROW
* REQUEST AND READ NCOL
      JST*   *+4
      BCI    3,NCOL
      DAC    >RR<
      STA    NCOL
      JST*   OBL=
      DAC    XCNR
      DAC    YCNR
      DAC    DXTP
      DAC    DYTP
      DAC    NROW
      DAC    NCOL
      DAC    RECT
      DAC    LOC
      LDA    REC=
      STA    DISS
      LDA    =875
      STA    DIX
      LDA    =1225
      STA    DIY
      JMP    DIAG
* CRG ON SWT=2 OR MORE
* CRG FOR ROAD EDGE ON SWT=-2
DICR SR1
      JMP    DIA
      LDA    DISW     TEST SWT
      SPL
      JMP    DIRE     ROAD EDGE
* REQUEST AND READ X0
      JST*   *+4
      BCI    3,X0
      DAC    >RR<
      STA    DIX0
* REQUEST AND READ Y0
      JST*   *+4
      BCI    3,Y0
      DAC    >RR<
      STA    DIY0
```

```
* REQUEST AND READ THSH
        JST*  *+4
        BCI   3,THSH
        DAC   >RR<
        STA   DITH
        LDA   =10
        STA   DICN
DICQ    JST*  CRL=
        DAC   DIXO
        DAC   DIYO
        DAC   DITH
        DAC   RECT
        CAS   =-1
        SKP
        JMP   DIC2
        LDA   ='3777
        OTA   '1452 Z
        JMP   *-1
DIC3    JST*  CRG=
        ALS   4
        ADD   =860     CENTER X DISPLAY
DIXS    OTA   '52      X, SHORT
        SKP
        JMP   DIC7
        STA   TEMP
        ALS   16       WAIT
        LDA   ='20000
        SMK   '20
        CRA
        SMK   '20      RESET READY FF
        LDA   TEMP
        JMP   DIXS
DIC7    IAB
        ALS   4
        ADD   =800     CENTER Y DISPLAY
DIYS    OTA   '1152    Y, SHORT, GO
        SKP
        JMP   DIC8
        STA   TEMP
        ALS   16       WAIT
        LDA   ='20000
        SMK   '20
        CRA
        SMK   '20      RESET READY FF
        LDA   TEMP
        JMP   DIYS
DIC8    SR1
        JMP   DIA
        JMP   DIC3
DIC2    IAB
        SZE
        JMP   DIC4     NOT-ON-CONTOUR OR ISOLATED
        LDA   DICN
        SMI
        JMP   DIC6     TRY AGAIN
        JMP   DIC9     DARK POINT
* ROAD EDGE
```

```
DIRE LDA    XCNR
     STA    DIXO
     LDA    YCNR
     STA    DIYO
* REQUEST AND READ THSH
     JST*   *+4
     BCI    3,THSH
     DAC    >RR<
     STA    DITH
* REQUEST AND READ LNTH
     JST*   *+4
     BCI    3,LNTH
     DAC    >RR<
     TCA
     STA    DIML     -LNTH
     LDA    =20
     STA    DICN
DIRQ JST*   CRL=
     DAC    DIXO
     DAC    DIYO
     DAC    DITH
     DAC    LOC
     CAS    =-1
     SKP
     JMP    DIC2
     LDA    ='3777
     OTA    '1452    Z
     JMP    *-1
     LDA    DIML
     STA    DILG
DIR3 JST*   CRG=
     IAB
     STA    TEMP
     IAB
     MPY    DID      SCALE X
     IAB
     ADD    DIXD     CENTER X
DIRX OTA    '52      X,SHORT
     SKP
     JMP    DIR7
     STA    TEMP+1
     ALS    16       WAIT
     LDA    ='20000
     SMK    '20
     CRA
     SMK    '20      RESET READY FF
     LDA    TEMP+1
     JMP    DIRX
DIR7 LDA    TEMP
     ADD    DIR     (-NO.ROWS)
     AOA            LOWEST ROW IS 0
     MPY    DID      SCALE Y
     IAB
     ADD    DIYD     CENTER Y
DIRY OTA    '1152    Y,SHORT,GO
     SKP
     JMP    DIR8
```

```
      STA   TEMP+1
      ALS   16       WAIT
      LDA   ='20000
      SMK   '20
      CRA
      SMK   '20      RESET READY FF
      LDA   TEMP+1
      JMP   DIRY
DIR8  SR1
      JMP   DICP
      IRS   DILG
      JMP   DIR3
      JMP*  DI=Z     SZC ON STYLUS DOWN; THEN DISPLAY LOC
* TYPE 'DARK POINT'
DIC9  JST*  '151
      DEC   -05
      DAC   *+2
      JMP   *+1+05
      BCI   05,DARK POINT
      JMP   DICR
DIC4  AOA
      SZE
      JMP   DIC5
      LDA   DICN
      SMI
      JMP   DIC6
* TYPE 'NOT ON CONTOUR'
      JST*  '151
      DEC   -07
      DAC   *+2
      JMP   *+1+07
      BCI   07,NOT ON CONTOUR
      JMP   DICR
* TYPE 'ISOLATED LIGHT POINT'
DIC5  JST*  '151
      DEC   -10
      DAC   *+2
      JMP   *+1+10
      BCI   10,ISOLATED LIGHT POINT
      JMP   DICR
DIC6  LDA   DICN
      SUB   =1
      STA   DICN
      LDA   DIYO
      AOA
      STA   DIYO
      LDA   DISW
      CAS   =-2
      JMP   DICQ
      JMP   DIRQ
      JMP   DIRQ
DIXO  BSS   1
DIYO  BSS   1
DILG  BSS   1
DIML  BSS   1
DITH  BSS   1
DIXD  BSS   1
```

```
DIYD BSS   1
DIOX BSS   1
DIOY BSS   1
DIR  BSS   1
DIRN BSS   1
DIYN BSS   1
DIC  BSS   1
DICN BSS   1
DISS BSS   1
DIEX BSS   1
DITX BSS   1
DITY BSS   1
DIST BSS   1
DISW BSS   1
XCNR BSS   1
YCNR BSS   1
DXTP BSS   1
DYTP BSS   1
NROW BSS   1
NCOL BSS   1
DI=Z DAC   DISZ
     FIN
*
*
*
*
*
*
*
** DIS CONTINUED
*
*
     ORG   '3000
*
*
* CALL SZC ON STYLUS DOWN
*
DISZ INA   '1151   GET STYLUS BIT
     JMP   *-1
     SPL           TEST STYLUS
     JMP*  DI=2    STYLUS UP: DISPLAY LOC
*REQUEST AND READ DELX
     JST*  *+4
     BCI   3,DELX
     DAC   >RR<
     STA   DIZX
* REQUEST AND READ DELY
     JST*  *+4
     BCI   3,DELY
     DAC   >RR<
     STA   DIZY
* CALL SUBROUTINE SZC
     JST*  SZC=
     DAC   DIZX
     DAC   DIZY
     DAC   DIXO
     DAC   DIYO
```

```
       DAC   DITH
       DAC   LOC
       DAC   DIL
       DAC   DIW
       DAC   DIXC
       DAC   DIYC
       DAC   DIXG
       DAC   DIYG
* CARR RET
       LDA   ='212
       JST*  '145
* TYPE 'LNTH='
       JST*  '152
       DEC   -03
       DAC   *+2
       JMP   *+1+03
       BCI   03,LNTH=
       LDA   DIL
       JST*  '154  TYPE DECIMAL
* TYPE ',WDTH='
       JST*  '152
       DEC   -03
       DAC   *+2
       JMP   *+1+03
       BCI   03,,WDTH=
       LDA   DIW
       JST*  '154  TYPE DECIMAL
* TYPE ',CNTR OF EXTENT=('
       JST*  '152
       DEC   -09
       DAC   *+2
       JMP   *+1+9
       BCI   09,,CNTR OF EXTENT=(
       LDA   DIXC
       JST*  '154  TYPE DECIMAL
* TYPE ','
       JST*  '152
       DEC   -01
       DAC   *+2
       JMP   *+1+01
       BCI   01,,
       LDA   DIYC
       JST*  '154  TYPE DECIMAL
* TYPE '),CNTR OF GRAV=('
       JST*  '152
       DEC   -08
       DAC   *+2
       JMP   *+1+08
       BCI   08,),CNTR OF GRAV=(
       LDA   DIXG
       JST*  '154  TYPE DECIMAL
* TYPE ','
       JST*  '152
       DEC   -01
       DAC   *+2
       JMP   *+1+01
       BCI   01,,
```

```
      LDA    DIYG
      JST*   '154   TYPE DECIMAL
* TYPE ')'
      JST*   '152
      DEC    -01
      DAC    *+2
      JMP    *+1+01
      BCI    01,)
* CARR RET
      LDA    ='212
      JST*   '145
      JMP*   DI=2
DI=2  DAC    DIA2
DIZX  BSS    1
DIZY  BSS    1
DIL   BSS    1
DIW   BSS    1
DIXC  BSS    1
DIYC  BSS    1
DIXG  BSS    1
DIYG  BSS    1
*
*
*
*
*
*
*
*
*
*
*
*
***PMD,IML
*
*
*
*
* PRINT MATRIX
*
* THE LINE PRINTER PRINTS A MATRIX FROM A
* STORED PICTURE.
*
* PMD PRINTS AN ALPHABETIC 'DIGIT' PROPORTIONAL TO
*     THE OCTAL VALUE OF THE POINT. (FROM DARK TO
*     BRIGHT, 64 LEVELS: @,A-Z,1-5 WITH OVERPRINTED
*     '.', AND @,A-Z,1-5 WITHOUT OVERPRINTING)
* PML PRINTS AN OVERPRINTED GREY-LEVEL CHARACTER
*     CORRESPONDING TO THE BRIGHTNESS OF THE POINT.
* PMD AND PML USE THE CLEAR FILTER PICTURE
*
*
* CALLED BY:
*     CALL   PM--,ADD1
*            WHERE ADD1 IS THE NAME OF THE FIRST ADDRESS
*              OF THE PICTURE
*               (I.E. ADD1 CONTAINS THE NO. OF ROWS,
```

```
*                             ADD1+1 CONTAINS THE NO. OF PTS/ROW,
*                             ADD1+2 ONWARDS CONTAIN THE PICTURE.)
*
* THE PICTURE IS ASSUMED TO BE IN STANDARD FORM.
* IF NO. OF COLUMNS > 120, PRINTS FIRST 120 COLUMNS ONLY.
* NO SPACES ARE LEFT AT THE BOTTOMS AND TOPS OF PAGES.
* THE INDEX REGISTER IS USED BY THESE ROUTINES
PMD  DAC   **
     LDA*  PMD
     STA   PMA1     FIRST ADDR OF PICTURE
     LDA   PMD
     AOA
     STA   PMEX     EXIT ADDRESS
     CRA
     STA   PMCS     COLOR SWITCH = 0
     STA   PMDG     DIGIT-GRAYLEVEL SWITCH = 0
     AOA
     STA   PMSW     INITIALIZE OVERPRINT SWITCH TO 1
     JMP   PMIN
PML  DAC   **
     LDA*  PML
     STA   PMA1     FIRST ADDR OF PICTURE
     LDA   PML
     AOA
     STA   PMEX     EXIT ADDRESS
     CRA
     STA   PMCS     COLOR SWITCH = 0
     AOA
     STA   PMSW     INITIALIZE OVERPRINT SWITCH TO 1
     STA   PMDG     DIGIT-GRAYLEVEL SWITCH = 1
     JMP   PMIN
PMIN LDA   PMSW
     SNZ
     JMP   *+4
     LDA*  PMA1
     ALS   1        2 * NO. ROWS
     SKP
     LDA*  PMA1     NO. ROWS
     TCA
     STA   PMYC     Y COUNTER
     IRS   PMA1
     LDA*  PMA1
     TCA
     STA   PMMP     -PTS/ROW
     IRS   PMA1
     LDA   PMA1     FIRST Z-POINT LOCATION
     STA   PMZ      Z LOCATION
     STA   PMOZ     OLD Z
     SKS   '0403    TEST IF CAN FORM-FEED
     JMP   *-1
     OCP   '1303    FORM-FEED
*-------------------------
* PRINT A LINE LOOP
*-------------------------
* CLEAR BUFFER
PMPL LDX   =-60
PM2B LDA   ='120240 TWO BLANKS
```

```
      STA   PMBU+60,1
      IRS   0
      JMP   PM2B
* GET NEXT LINE INTO BUFFER
      LDA   =1
      CAS   PMCS      TEST COLOR SWITCH
      JMP   PMCF      CS = 0, CLEAR FILTER
      JMP   PMRF      CS = 1, RED FILTER
      LDA   =3
      CAS   PMCS      TEST COLOR SWITCH
      JMP   PMGF      CS = 2, GREEN FILTER
      JMP   PMBF      CS = 3, BLUE FILTER
* PREPARE TO PRINT
PMPP  LDX   =-60      TO PRINT 120 CHARS
      SKS   '0203     TEST IF PRINTER READY
      JMP   *-1
      SKS   '0403     TEST IF CAN MOVE PAPER
      JMP   *-1
      OCP   '1703     LINE FEED ONE LINE
      OCP   '0103     PREPARE TO PRINT VIA I/O BUS
      JMP   PMPB
* PREPARE TO PRINT FOR OVERPRINTING
PMP2  LDX   =-60      TO PRINT 120 CHARS
      SKS   '0203     TEST IF PRINTER READY
      JMP   *-1
      LDA   PMSW      TEST OVERPRINT SWITCH
      SNZ
      JMP   PMSS      OVERPRINT: NO LINE FEED
      SKS   '0403     TEST IF CAN MOVE PAPER
      JMP   *-1
      OCP   '1703     LINE-FEED ONE LINE
      LDA   PMOZ      GET OLD Z
      STA   PMZ       STORE FOR NEXT OVERPRINT
      CRA
      STA   PMSW
      OCP   '0103     PREPARE TO PRINT VIA I/O BUS
      JMP   PMPB
PMSS  LDA   =1        OVERPRINT
      STA   PMSW
      LDA   PMZ
      STA   PMOZ      SET OLD Z
      LDA   =-500     DELAY 500*18 USEC
      AOA                   "
      LLR   32              "
      SPL                   "
      JMP   *-3             "
      OCP   '0103     PREPARE TO PRINT VIA I/O BUS
* PRINT BUFFER-WORD LOOP
PMPB  LDA   PMBU+60,1 ONE WORD (2 CHARS)
      OTA   '0003     2 CHARS TO PRINTER
      JMP   *-1
      IRS   0         TEST IF WORD FINISHED
      JMP   PMPB      GET NEXT 2 CHARS.
      OCP   '0203     PRINT TWO CHARS.
      IRS   PMYC      TEST Y COUNTER
      JMP   PMPL      NEXT LINE
      JMP*  PMEX      EXIT
```

```
*------------------------
* LOAD BUFFER (FOR CLEAR FILTER)
*------------------------
PMCF LDA    PMMP      -PTS/ROW
     STA    PMPC      POINT COUNTER
     LDX    =-60      FOR 120 POINTS
     LDA    PMDG .    DIGIT-GRAYLEVEL SWITCH
     SZE              TEST SWITCH
     JMP    PMCC
* LOAD BUFFER WITH ALPHABETIC DIGITS (@,A-Z,1-5)
PMCO LDA    PMSW
     SZE
     JMP    PMCA
     LDA*   PMZ       Z
     ANA    ='2000    GET TOP BIT
     SZE
     JMP    *+3
     LDA    ='27000 "."
     JMP    PMCK
     LDA    ='120000 BLANK
     JMP    PMCK
PMCA LDA*   PMZ       Z
     ANA    ='1740    GET BITS 2-6
     ALS    1
     CAS    ='3200
     ADD    ='2600    MAKE 1-5
     NOP
     ALS    2
PMCK STA    PMBU+60,1 FIRST CHAR TO BUFFER
     IRS    PMZ
     IRS    PMPC      TEST POINT-COUNTER
     JMP    *+4
     ERA    ='000240 PUT IN BLANK
     STA    PMBU+60,1
     JMP    PMP2      RETURN
     LDA    PMSW
     SZE
     JMP    PMCQ
     LDA*   PMZ       Z
     ANA    ='2000    GET TOP BIT
     SZE
     JMP    *+3
     LDA    ='56
     JMP    PMCJ
     LDA    ='40
     JMP    PMCJ
PMCQ LDA*   PMZ       Z
     ANA    ='1740
     LGR    5         GET BITS 2-6
     CAS    ='32
     ADD    ='26      MAKE 1-5
     NOP
PMCJ ERA    PMBU+60,1
     STA    PMBU+60,1 SECOND CHAR TO BUFFER
     IRS    PMZ
     IRS    PMPC      TEST POINT-COUNTER
     SKP
```

```
      JMP    PMP2     RETURN
      IRS    0
      JMP    PMC0
      LDA    PMZ
      SUB    PMPC     -(-REMAINING POINTS)
      STA    PMZ
      JMP    PMP2     RETURN
* LOAD BUFFER WITH GRAYLEVEL CHARACTERS
PMCC  LDA    =1
      STA    PM12     CHAR1-CHAR2 SWITCH
PMCN  LDA    PMSW
      SNZ
      JMP    *+3
      LDA    PML1     LEVEL LIST 1 LOC
      SKP
      LDA    PML2     LEVEL LIST 2 LOC
      STA    PMLC     LEVEL LIST LOC
      LDA*   PMZ      Z
      ANA    ='3600   TOP 4 BITS
      ARS    7        RIGHT JUSTIFY
PMCG  SR3
      SUB    PMDP     DARKEN PRINTOUT ON SS3 UP
      ADD    PMLC
      STA    PMLC     GRAY LEVEL LOCATION
      LDA*   PMLC
      STA    PMLV     GRAY LEVEL
      LDA    PM12     CHAR1-CHAR2 SWITCH
      CAS    =1
      JMP    PMCT
      LDA    PMLV     CHARACTER ONE
      ALS    8
      STA    PMBU+60,1 TO BUFFER
      LDA    =2       NEXT CHAR IS 2ND
      STA    PM12
      IRS    PMZ
      IRS    PMPC     TEST POINT-COUNTER
      JMP    PMCN     NEXT CHAR
      LDA    PMBU+60,1
      ERA    ='240    PUT IN BLANK
      STA    PMBU+60,1
      JMP    PMP2     RETURN
PMCT  LDA    PMLV     CHARACTER TWO
      ERA    PMBU+60,1
      STA    PMBU+60,1 TO BUFFER
      LDA    =1       NEXT CHAR IS 1ST
      STA    PM12
      IRS    PMZ
      IRS    PMPC     TEST POINT-COUNTER
      SKP
      JMP    PMP2     RETURN
      IRS    0
      JMP    PMCN     NEXT CHAR
      LDA    PMZ
      SUB    PMPC     -(- REMAINING POINTS)
      STA    PMZ
      JMP    PMP2
** TEMP
```

```
PMRF JMP    PMPP
PMGF JMP    PMPP
PMBF JMP    PMPP
** END TEMP
PMA1 OCT    0          FIRST ADDR OF PICTURE
PMCS OCT    0          COLOR SWITCH
PMDG OCT    0          DIGIT-GRAYLEVEL SWITCH
PMEX OCT    0          EXIT ADDRESS
PMBU BSS    60         BUFFER CONTAINING PRINTLINE
PMMP OCT    0          -PTS/ROW
PMPC OCT    0          POINT COUNTER
PMYC OCT    0          Y COUNTER
PMZ  OCT    0          Z LOCATION
PMOZ OCT    0          OLD Z
PMLZ OCT    0          LINE-PRINTER Z
PMLV OCT    0          GRAYLEVEL
PM12 OCT    0          CHAR1-CHAR2 SWITCH
PMSW OCT    0          OVERPRINT SWITCH
PMDP OCT    2          DARKEN PRINTOUT ON SS3 UP
PMLC OCT    0          LOCATION
PML1 DAC    PM1        FIRST PRINT LOC
PML2 DAC    PM2        SECOND PRINT LOC
PM1  OCT    0          @. TO C. = @
     OCT    0          D. TO G. = @
     OCT    0          H. TO K. = @
     OCT    0          L. TO O. = @
     OCT    0          P. TO S. = @
     OCT    0          T. TO W. = @
     OCT    0          X. TO 1. = @
     OCT    0          2. TO 5. = @
     OCT    0          @  TO C  = @
     OCT    0          D  TO G  = @
     OCT    0          H  TO K  = @
     OCT    3          L  TO O  = C
     OCT    3          P  TO S  = C
     OCT    50         T  TO W  = (
     OCT    36         X  TO 1  = ^
     OCT    40         2  TO 5  =
PM2  OCT    2          @. TO C. = B
     OCT    2          D. TO G. = B
     OCT    2          H. TO K. = B
     OCT    2          L. TO O. = B
     OCT    2          P. TO S. = B
     OCT    2          T. TO W. = B
     OCT    2          X. TO 1. = B
     OCT    2          2. TO 5. = B
     OCT    2          @  TO C  = B
     OCT    0          D  TO G  = @
     OCT    3          H  TO K  = C
     OCT    11         L  TO O  = I
     OCT    3          P  TO S  = C
     OCT    50         T  TO W  = (
     OCT    40         X  TO 1  =
     OCT    40         2  TO 5  =
     FIN
*
*
```

```
*
*
***STO
     ORG    '4000
*
* ROUTINE TO RASTER SCAN AREA
* BETWEEN YTOP,YBOT,XL, AND XR
* AT SPACING DELT AND STORE IN
* LOCATIONS FROM LOC IN STD FORM.
* CALLING SEQUENCE:
*       JST    STO
*       DAC    YTOP
*       DAC    YBOT
*       DAC    XL
*       DAC    XR
*       DAC    DELT
*       DAC    LOC
*
STEX BSS    1
STL  BSS    1
STCC BSS    1
STOT BSS    1
STOB BSS    1
STOL BSS    1
STOR BSS    1
STOD BSS    1
STO  DAC*   **
     LDA*   STO
     STA    STOT
     IRS    STO
     LDA*   STO
     STA    STOB
     IRS    STO
     LDA*   STO
     STA    STOL
     IRS    STO
     LDA*   STO
     STA    STOR
     IRS    STO
     LDA*   STO
     STA    STOD
     IRS    STO
     LDA    STOT
     OTA    '1052 Y, LONG
     JMP    *-1
     LDA    STO
     SSP
     STA    STEX
     LDA*   STEX
     STA    STL
     IRS    STEX
     AOA
     STA    STCC
     AOA
     STA    0
     SKP
     IRS*   STCC
```

```
      LDA    *-1
      STA    STO2
      CRA
      STA*   STL
      STA*   STCC
      LDA    STOL
STO3  OTA    '352   X, SHORT, GO
      JMP    *-1
      CAS    STOR
      JMP    STO4
STO5  NOP
      ADD    STOD
      IAB
      INA    '1070
      JMP    *-1
      ARS    7
      STA    0,1
      IRS    0
STO2  IRS*   STCC   COLUMN COUNT
      IAB
      JMP    STO3
* END OF LINE
STO4  LDA    STO5    (NOP)
      STA    STO2    NOP FOR IRS*  STCC
      IRS*   STL     ROW COUNT
      LDA    STOT
      SUB    STOD
      CAS    STOB
      NOP
      JMP    STO6
      JMP*   STEX
STO6  OTA    '1052 Y, LONG
      JMP    *-1
      STA    STOT
      JMP    STO3-1
*
*
*
***OBL
*
* OBLIQUE SCAN
*
* PERFORMS, ON A STORED PICTURE, A RASTER SCAN OF AN OBLIQUE
*    RETANGLE, AND STORES THE SCANNED POINTS IN STANDARD
*    FORM.  THE SCAN LINES ARE AT THE OBLIQUE ANGLE OF THE
*    RECTANGLE.
*
*    CALLED BY
*          CALL OBL,XC,YC,DXTP,DYTP,NROW,NCOL,RECT,PICT
*    WHERE
*          (XC,YC) = THE CORNER POINT OF THE OBLIQUE
*                    RECTANGLE WHICH WILL BE STORED AS
*                    THA FIRST POINT OF THE SCAN, I.E.
*                    THE  UPPER LEFT  POINT IF THE
*                    RECTANGLE WAS ALONG THE COORDINATE
*                    AXES .
*
```

```
*           DYTP/DXTP = THE DIRECTIONED SLOPE OF THE 'TOP' OF THE
*                       RECTANGLE STARTING FROM (XC,YC).
*                        THE SIGNS OF DXTP,DYTP GIVE THE
*                        QUADRANT.
*           NROW = THE NUMBER OF ROWS TO BE SCANNED.
*           NCOL = THE NUMBER OF COLUMNS TO BE SCANNED.
*           RECT IS THE FIRST LOCATION OF THE MATRIX
*                  WHERE THE RECTANGLE WILL BE STORED.
*           PICT IS THE FIRST LOCATION OF THE STORED PICTURE
*
*    NOTE: XC,YC,DXTP,DYTP ARE IN THE STORED PICTURE'S
*     COORDINATES.
*
* INITIALIZE
OBXC BSS   1
OBYC BSS   1
OBDX BSS   1
OBDY BSS   1
OBNR BSS   1
OBNC BSS   1
OBST BSS   1
OBPC BSS   1
OBL  DAC*  **
     LDA*  OBL
     STA   OBXC
     IRS   OBL
     LDA*  OBL
     STA   OBYC
     IRS   OBL
     LDA*  OBL
     STA   OBDX
     IRS   OBL
     LDA*  OBL
     STA   OBDY
     IRS   OBL
     LDA*  OBL
     STA   OBNR
     IRS   OBL
     LDA*  OBL
     STA   OBNC
     IRS   OBL
     LDA   OBL
     ANA   ='37777
     STA   OBL2
     LDA*  OBL2
     STA   OBST
     IRS   OBL2
     LDA*  OBL2
     STA   OBPC
     IRS   OBL2
     LDA   OBNR
     STA*  OBST     NO. ROWS TO BE SCANNED
     TCA
     STA   OBRC     ROW COUNTER
     IRS   OBST
     LDA   OBNC
     STA*  OBST     NO. COLS TO BE SCANNED
```

```
      TCA
      STA   OBMC     - NO. COLS
      IRS   OBST     TOP OF RECT STORAGE
      LDA   OBXC
      STA   OBX
      LDA   OBYC
      STA   OBY
      LDA*  OBPC
      STA   OBRP     ROWS OF PICTURE
      SUB   =1
      STA   OBR1     ROWS-1
      IRS   OBPC
      LDA*  OBPC
      STA   OBCL     COLS OF PICTURE
*  SET-UP SECOND DRAW-LINE FOR 'LEFT' SIDELINE
      LDA   OBDX
      TCA
      STA   OBMX     SLOPE OF SIDELINE = (-DX)/DY
      JST   D2S
      DAC   OBXC
      DAC   OBYC
      DAC   OBDY
      DAC   OBMX
*  SET-UP PRIMARY DRAW-LINE FOR SCAN LINE
OBSL  JST   DLS
      DAC   OBX
      DAC   OBY
      DAC   OBDX
      DAC   OBDY
* SCAN A LINE
      LDA   OBMC     -NO. COLUMNS
      STA   OBCC     COLUMN COUNTER
      LDA   OBR1     NO. ROWS-1
      SUB   OBY
      MPY   OBCL
      IAB            PRODUCT TO A
      ADD   OBX
      ADD   OBPC     PTNO = PICT(1)-1 +
      STA   OBPT        (PCRWS-(Y+1))*PCCLS + X+1
OBLP  LDA*  OBPT     SCAN-A-POINT LOOP
      STA*  OBST     STORE RECT POINT
      IRS   OBST
      IRS   OBCC
      SKP
      JMP   OBRW     END OF ROW
      JST   DLN
      CAS   OBX      X VS. OLD X
      JMP   OBGX     X > OLD X
      JMP   OBTY     X = OLD X
      STA   OBX      X < OLD X
      LDA   OBPT
      SUB   =1
      STA   OBPT
OBTY  IAB            TEST Y
      CAS   OBY      Y VS. OLD Y
      JMP   OBGY     Y > OLD Y
      JMP   OBLP     Y = OLD Y
```

```
      STA   OBY       Y < OLD Y
      LDA   OBPT
      ADD   OBCL
      STA   OBPT
      JMP   OBLP
OBGX STA   OBX
      IRS   OBPT
      JMP   OBTY
OBGY STA   OBY
      LDA   OBPT
      SUB   OBCL
      STA   OBPT
      JMP   OBLP
*  GET NEXT ROW INITIAL POINT
OBRW IRS   OBRC      ROW COUNTER
      SKP
      JMP*  OBL2
      JST   D2N
      STA   OBX
      IAB
      STA   OBY
      JMP   OBSL      SCAN NEXT LINE
OBRC BSS   1
OBMC BSS   1
OBX  BSS   1
OBY  BSS   1
OBRP BSS   1
OBR1 BSS   1
OBCL BSS   1
OBMX BSS   1
OBCC BSS   1
OBPT BSS   1
OBL2 BSS   1
      FIN
*
*
*
*
*
*
*
***DLN
*
* DRAW LINE
*
*  DRAWS THE BEST DIGITAL LINE AT A GIVEN
*   SLOPE FROM A GIVEN POINT.  EACH POINT ON
*   THE DIGITAL LINE IS ONE OF THE EIGHT POINTS
*   ADJACENT TO THE PREVIOUS POINT.
*
*  DLS SETS UP THE PARAMETERS FOR A NEW LINE
*   IT IS CALLED BY
*          CALL  DLS ,XO,YO,DELX,DELY
*            WHERE XO,YO = THE INITIAL POINT ON THE LINE.
*            DELY = THE SIGNED CHANGE IN Y
*              FOR EACH SIGNED CHANGE IN X
*              OF DELX.
```

```
*
*       (NOTE:  DELY/DELX = THE SLOPE OF THE LINE.
*            THE SIGNS OF DELX AND DELY DETERMINE
*            THE QUADRANTS.)
*
* DLN IS CALLED REPEATEDLY, AND EACH TIME
*       IT RETURNS THE X AND Y VALUES OF
*       THE NEXT POINT ON THE DIGITIZED LINE:
*          THE X VALUE IS RETURNED IN THE A-REGISTER.
*          THE Y VALUE IS RETURNED IN THE B-REGISTER.
*       IT IS CALLED BY:
*            CALL   DLN
*
*
*
*----------
* SET-UP DRAW LINE
*----------
*
DLX  BSS   1
DLY  BSS   1
DLDX BSS   1
DLDY BSS   1
DLS  DAC*  **
     LDA*  DLS
     STA   DLX
     IRS   DLS
     LDA*  DLS
     STA   DLY
     IRS   DLS
     LDA*  DLS
     STA   DLDX
     IRS   DLS
     LDA*  DLS
     STA   DLDY
     IRS   DLS
     LDA   DLDX
     SMI
     JMP   DLXP
     LDA   =-1
     SKP
DLXP LDA   =1
     STA   DLSX     SX = SGN(DX)
     LDA   DLDY
     SMI
     JMP   DLYP
     LDA   =-1
     SKP
DLYP LDA   =1
     STA   DLSY     SY = SGN(DY)
     LDA   DLDX
     SPL
     TCA
     STA   DLAX     AX = ABS(DX)
     LDA   DLDY
     SPL
     TCA
```

```
      STA   DLAY      AY = ABS(DY)
      CAS   DLAX
      JMP   DLYG
      NOP
*  ABSDY < ABSDX        (OCTANTS 1,4,5,8)
      ADD   DLAY
      STA   DLA       A = 2*ABSDY
      SUB   DLAX
      STA   DLT       INITIAL T = 2*ABSDY - ABSDX
      SUB   DLAX
      STA   DLB       B = 2*ABSDY - 2*ABSDX
      CRA
      JMP   DLEX
*  ABSDY > ABSDX        (OCTANTS 2,3,7,8)
DLYG  LDA   DLAX
      ADD   DLAX
      STA   DLA       A = 2*ABSDX
      SUB   DLAY
      STA   DLT       INITIAL T = 2*ABSDX - ABSDY
      SUB   DLAY
      STA   DLB       B = 2*ABSDX - 2*ABSDY
      CRA
      AOA
DLEX  STA   DLSW      OCTANT SWITCH
      LDA   DLS
      ANA   ='37777
      STA   *+2
      JMP*  *+1
      BSS   1
      FIN
*
*----------
* DRAW LINE
*----------
*
DLN   DAC   **
      LDA   DLSW      TEST OCTANT SWITCH
      SZE
      JMP   DLO2
*  FOR OCTANTS 1,4,5,8:
      LDA   DLT
      SPL
      JMP   DL1L
      ADD   DLB       T >0 OR T = 0
      STA   DLT       T = T+B
      LDA   DLY
      ADD   DLSY      Y = Y+SY
      STA   DLY
      IAB
      LDA   DLX
      ADD   DLSX      X = X+SX
      STA   DLX
      JMP*  DLN
DL1L  ADD   DLA       T<0
      STA   DLT       T = T+A
      LDA   DLY       Y = Y+0
      IAB
```

```
      LDA   DLX
      ADD   DLSX     X = X+SX
      STA   DLX
      JMP*  DLN
*  FOR OCTANTS 2,3,6,7:
DLO2 LDA   DLT
      SPL
      JMP   DL2L
      ADD   DLB      T > 0 OR T = 0
      STA   DLT      T = T + B
      LDA   DLY
      ADD   DLSY     Y = Y + SY
      STA   DLY
      IAB
      LDA   DLX
      ADD   DLSX     X = X + SX
      STA   DLX
      JMP*  DLN
DL2L ADD   DLA      T < 0
      STA   DLT      T = T + A
      LDA   DLY
      ADD   DLSY     Y = Y + A
      STA   DLY
      IAB
      LDA   DLX      X = X + 0
      JMP*  DLN
*
DLSX BSS   1
DLSY BSS   1
DLAX BSS   1
DLAY BSS   1
DLA  BSS   1
DLT  BSS   1
DLB  BSS   1
DLSW BSS   1
      FIN
*
*
*
*
***D2N
*
*
* A SECOND VERSION OF DLN, FOR DRAWING TWO LINES
*   SIMULTANEOUSLY: ONE WITH DLN AND ONE WITH D2N.
*
*
*    USE D2N AND D2S
*
*
*----------
* SET-UP DRAW LINE
*----------
*
D2X  BSS   1
D2Y  BSS   1
D2DX BSS   1
```

```
D2DY BSS    1
D2S  DAC*   **
     LDA*   D2S
     STA    D2X
     IRS    D2S
     LDA*   D2S
     STA    D2Y
     IRS    D2S
     LDA*   D2S
     STA    D2DX
     IRS    D2S
     LDA*   D2S
     STA    D2DY
     IRS    D2S
     LDA    D2DX
     SMI
     JMP    D2XP
     LDA    =-1
     SKP
D2XP LDA    =1
     STA    D2SX     SX = SGN(DX)
     LDA    D2DY
     SMI
     JMP    D2YP
     LDA    =-1
     SKP
D2YP LDA    =1
     STA    D2SY     SY = SGN(DY)
     LDA    D2DX
     SPL
     TCA
     STA    D2AX     AX = ABS(DX)
     LDA    D2DY
     SPL
     TCA
     STA    D2AY     AY = ABS(DY)
     CAS    D2AX
     JMP    D2YG
     NOP
*   ABSDY < ABSDX      (OCTANTS 1,4,5,8)
     ADD    D2AY
     STA    D2A      A = 2*ABSDY
     SUB    D2AX
     STA    D2T      INITIAL T = 2*ABSDY - ABSDX
     SUB    D2AX
     STA    D2B      B = 2*ABSDY - 2*ABSDX
     CRA
     JMP    D2EX
*   ABSDY > ABSDX      (OCTANTS 2,3,7,8)
D2YG LDA    D2AX
     ADD    D2AX
     STA    D2A      A = 2*ABSDX
     SUB    D2AY
     STA    D2T      INITIAL T = 2*ABSDX - ABSDY
     SUB    D2AY
     STA    D2B      B = 2*ABSDX - 2*ABSDY
     CRA
```

```
      AOA
D2EX  STA    D2SW     OCTANT SWITCH
      LDA    D2S
      ANA    ='37777
      STA    *+2
      JMP*   *+1
      BSS    1
      FIN
*
*----------
* DRAW LINE
*----------
*
D2N   DAC    **
      LDA    D2SW     TEST OCTANT SWITCH
      SZE
      JMP    D2O2
*  FOR OCTANTS 1,4,5,8:
      LDA    D2T
      SPL
      JMP    D21L
      ADD    D2B      T >0 OR T = 0
      STA    D2T      T = T+B
      LDA    D2Y
      ADD    D2SY     Y = Y+SY
      STA    D2Y
      IAB
      LDA    D2X
      ADD    D2SX     X = X+SX
      STA    D2X
      JMP*   D2N
D21L  ADD    D2A      T<0
      STA    D2T      T = T+A
      LDA    D2Y      Y = Y+0
      IAB
      LDA    D2X
      ADD    D2SX     X = X+SX
      STA    D2X
      JMP*   D2N
*  FOR OCTANTS 2,3,6,7:
D2O2  LDA    D2T
      SPL
      JMP    D22L
      ADD    D2B      T > 0 OR T = 0
      STA    D2T      T = T + B
      LDA    D2Y
      ADD    D2SY     Y = Y + SY
      STA    D2Y
      IAB
      LDA    D2X
      ADD    D2SX     X = X + SX
      STA    D2X
      JMP*   D2N
D22L  ADD    D2A      T < 0
      STA    D2T      T = T + A
      LDA    D2Y
      ADD    D2SY     Y = Y + A
```

```
        STA   D2Y
        IAB
        LDA   D2X       X = X + 0
        JMP*  D2N
*
D2SX BSS   1
D2SY BSS   1
D2AX BSS   1
D2AY BSS   1
D2A  BSS   1
D2T  BSS   1
D2B  BSS   1
D2SW BSS   1
*
*
*
*
*
*
*
*
*
       FIN
*
*
*
*
***SQR
*
       ORG   '5000
*
* SQUARE ROOT
*
* FOR A-REGISTER = N > 0,
*          RETURNS A-REGISTER = SQUAREROOT(N)
* FOR A-REGISTER = 0, OR A-REGISTER < 0,
*          RETURNS A-REGISTER = 0.
*
*
*   ROUNDING TAKES PLACE SO AS TO RETURN THE LARGEST
*   NUMBER, A, SUCH THAT :
*      A*(A+1) <OR= N.
*
*
* THE X REGISTER IS PRESERVED.
*
* CALLED BY:
*          CALL  SQR
*
*
SQR  DAC   **
     SNZ
     JMP*  SQR
     SPL
     JMP   SQR0
     STX   SQRX     SAVE X
     STA   SQRN     SAVE NUMBER
```

```
      NRM            COMPUTE FIRST APPROX
      SCA
      ARS    1
      STA    0
      LDA    SQRI,1  2**(P/2)
      LDX    =-3     THREE ITERATIONS
SQR1  STA    SQRA    STORE APPROX
      LDA    SQRN    GET ORIG NUMBER
      IAB
      CRA
      DIV    SQRA    NUMBER/LASTAPPROX
      ADD    SQRA
      LGR    1       NEWAPPROX = (N/A + A)/2
      IRS    0
      JMP    SQR1
      STA    SQRA
      AOA
      MPY    SQRA    A*(A+1)
      IAB
      RCB
      CAS    SQRN
      NOP            A*(A+1) > N : USE A
      SCB                    = N : USE A
      LDA    SQRA            < N : USE A+1
      SSC
      AOA
      LDX    SQRX    RESTORE X
      JMP*   SQR     RETURN WITH SQRT IN A-REG
SQRO  CRA
      JMP*   SQR     RETURN ZERO FOR N < 0
SQRN  OCT    0
SQRX  OCT    0
SQRA  OCT    0
SQRI  OCT    000200  INITIAL APPROXIMATIONS
      OCT    000100
      OCT    000040
      OCT    000020
      OCT    000010
      OCT    000004
      OCT    000002
      OCT    000001
      FIN
*
*
*
*
*
*
***SZC
*
* SIZE AND CENTER
*
* FINDS THE SIZE OF A VEHICLE AND ITS CENTER POINT
*     FROM A STORED PICTURE
*
* CALLED BY:
*    CALL   SZC,DELX,DELY,INX,INY,THSH,LOC
```

```
*    MDAC  LNTH,WDTH,XCTR,YCTR,XGRV,YGRV
*
*   WHERE
*
*     DELY/DELX = THE DIRECTED SLOPE OF THE CENTER LINE
*            OF THE VEHICLE (THE SIGNS OF DELY AND DELX
*            GIVING THE QUADRANT).
*     (INX,INY) = THE INITIAL POINT OF THE CONTOUR-TRACE
*            IN STORED-PICTURE COORDINATES.
*     THSH = THE CONTOUR-TRACE THRESHOLD.
*     LOC IS THE ADDRESS OF THE STORED PICTURE.
*
*    THE FOLLOWING IS FOUND:
*
*     LNTH = THE EXTENT OF THE VEHICLE IN THE DIRECTION
*        ALONG ITS CENTER LINE.
*     WDTH = THE EXTENT OF THE VEHICLE IN THE DIRECTION
*        PERPENDICULAR TO ITS CENTER LINE.
*     (XCTR,YCTR) = THE CENTER POINT OF THE VEHICLE, BASED
*        ON EXTENT.
*     (XGRV,YGRV) = THE CENTER OF GRAVITY OF THE VEHICLE
*        BASED ON ITS CONTOUR.
*
*
*     OVERFLOW MAY OCCUR FOR X OR Y > 100.
*     THIS WILL NOT ARISE IF PICTURE SUBSETS LESS
*      THAN 100 X 100 ARE USED.
*
*
*
*
*
SZDX BSS    1
SZDY BSS    1
SZIX BSS    1
SZIY BSS    1
SZTH BSS    1
SZC  DAC*   **
     LDA*   SZC
     STA    SZDX
     IRS    SZC
     LDA*   SZC
     STA    SZDY
     IRS    SZC
     LDA*   SZC
     STA    SZIX
     IRS    SZC
     LDA*   SZC
     STA    SZIY
     IRS    SZC
     LDA*   SZC
     STA    SZTH
     IRS    SZC
     LDA    SZC
     SSP
     STA    TEMP
     LDA*   TEMP
```

```
      STA    SZLC
      IRS    SZC
* INITIALIZE
      CRA
      STA    SZXT
      STA    SZYT
      STA    SZNP
      LDA    ='100000 MOST NEG NO.
      STA    SZHS
      STA    SZHR
      LDA    ='77777 MOST POS NO.
      STA    SZLS
      STA    SZLR
      LDA    =-2
      STA    SZOA
* SCALE UP DX AND DY SO HIGHEST ABS VALUE = 100
      LDA    SZDY
      SPL
      TCA
      STA    SZAY     ABS DY
      LDA    SZDX
      SPL
      TCA             ABS DX
      CAS    SZAY
      JMP    *+3      ABSDX > ABSDY
      NOP             ABSDX = ABSDY
      LDA    SZAY     ABSDY > ABSDX
      STA    SZHA     HIGHEST ABS D
      SUB    =100
      SMI
      JMP    SZSU     HIGHEST ABS D > 100
      LDA    =100
      IAB
      CRA
      DIV    SZHA
      STA    SZSF     SCALE FACTOR
      MPY    SZDX
      LLR    1
      IAB
      ARR    1
      STA    SZDX     SCALE UP DX
      LDA    SZSF
      MPY    SZDY
      LLR    1
      IAB
      ARR    1
      STA    SZDY     SCALE UP DY
* SET-UP CONTOUR TRACE
SZSU  JST*   CRL=
      DAC    SZIX
      DAC    SZIY
      DAC    SZTH
SZLC  DAC    **
      CAS    =-1
      SKP
      JMP    SZER     ERROR RETURN
* GET CONTOUR POINT
```

```
SZPT JST*  CRG=
     STA   SZX
     IAB              Y TO A-REG
     STA   SZY
* TEST IF COMPLETED ONCE AROUND CONTOUR
     CAS   SZIY
     JMP   SZPJ
     SKP              Y = INIT Y
     JMP   SZPJ
     LDA   SZX
     CAS   SZIX
     JMP   SZPJ
     SKP              X = INIT X
     JMP   SZPJ
     IRS   SZOA
     JMP   SZPJ
     JMP   SZFQ       ONCE AROUND
* FIND PROJECTIONS OF CONTOUR POINT
SZPJ LDA   SZY
     MPY   SZDX
     LLR   1
     IAB
     ARR   1
     STA   TEMP
     LDA   SZX
     MPY   SZDY
     LLR   1
     IAB
     ARR   1
     SUB   TEMP
     STA   SZR        R = U*Q = X*DY - Y*DX
     LDA   SZX
     MPY   SZDX
     LLR   1
     IAB
     ARR   1
     STA   TEMP
     LDA   SZY
     MPY   SZDY
     IAB              PRODUCT TO A
     ADD   TEMP
     STA   SZS        S = V*Q = X*DX + Y*DY
* TEST IF EXTREME POINT
     CAS   SZHS       S VS. HI S
     STA   SZHS       NEW HI S
     NOP
     CAS   SZLS       S VS. LO S
     NOP
     SKP
     STA   SZLS       NEW LO S
     LDA   SZR
     CAS   SZHR       R VS. HI R
     STA   SZHR       NEW HI R
     NOP
     CAS   SZLR       R VS. LO R
     NOP
     SKP
```

```
      STA   SZLR     NEW LO R
* ADD POINT TO CENTER OF GRAVITY
      LDA   SZXT
      ADD   SZX
      STA   SZXT     CUMULATIVE X TOTAL
      LDA   SZYT
      ADD   SZY
      STA   SZYT     CUMULATIVE Y TOTAL
      IRS   SZNP     NO. OF POINTS
      JMP   SZPT     GET NEXT POINT
* FIND Q
SZFQ  LDA   SZDX
      MPY   SZDX
      IAB            PRODUCT TO A
      STA   TEMP
      LDA   SZDY
      MPY   SZDY
      IAB            PRODUCT TO A
      ADD   TEMP
      STA   SZQ2     Q2 = DX**2 + DY**2
      JST*  SQR=
      STA   SZQ      Q = SQRT(DX**2 + DY**2)
* FIND HEIGHT AND WIDTH
      LDA   SZHS
      SUB   SZLS
      IAB
      CRA
      DIV   SZQ
      STA   TEMP     ROUND DIVISION
      LLS   1          "
      CRA              "
      DIV   SZQ        "
      ADD   TEMP       "
      STA   SZL      LNTH = HIV-LOV = (HIS-LOS)/Q
      LDA   SZHR
      SUB   SZLR
      IAB
      CRA
      DIV   SZQ
      STA   TEMP     ROUND DIVISION
      LLS   1          "
      CRA              "
      DIV   SZQ        "
      ADD   TEMP       "
      STA   SZW      WDTH = HIU-LOU = (HIR-LOR)/Q
* FIND CENTER BASED ON EXTENT (XCTR,YCTR)
      LDA   SZHS
      ADD   SZLS
      MPY   SZDX
      DBL
      STA   TEMP
      SGL
      LDA   SZHR
      ADD   SZLR
      MPY   SZDY
      DBL
      ADD   TEMP
```

```
      DIV   SZQ2
      SGL
      STA   TEMP     ROUND DIVISION
      LLS   1         "
      CRA             "
      DIV   SZQ2      "
      ADD   TEMP      "
      ARS   1
      STA   SZXC     XC=( (HIS+LOS)*DX + (HIR+LOR)*DY )/2*Q**2
      LDA   SZHR
      ADD   SZLR
      MPY   SZDX
      DBL
      STA   TEMP
      SGL
      LDA   SZHS
      ADD   SZLS
      MPY   SZDY
      DBL
      SUB   TEMP
      DIV   SZQ2
      SGL
      STA   TEMP     ROUND DIVISION
      LLS   1         "
      CRA             "
      DIV   SZQ2      "
      ADD   TEMP      "
      ARS   1
      STA   SZYC     YC=( (HIS+LOS)*DY - (HIR+LOR)*DX )/2*Q**2
* FIND CENTER OF GRAVITY (XGRV,YGRV)
      LDA   SZXT
      IAB
      CRA
      DIV   SZNP
      STA   SZXG     XGRV = XTOT/NO.PTS
      LDA   SZYT
      IAB
      CRA
      DIV   SZNP
      STA   SZYG     YGRV = YTOT/NO.PTS
* RETURN VALUES
SZRV  LDA   SZL
      STA*  SZC
      IRS   SZC
      LDA   SZW
      STA*  SZC
      IRS   SZC
      LDA   SZXC
      STA*  SZC
      IRS   SZC
      LDA   SZYC
      STA*  SZC
      IRS   SZC
      LDA   SZXG
      STA*  SZC
      IRS   SZC
      LDA   SZYG
```

```
      STA*  SZC
      IRS   SZC
      LDA   SZC
      SSP
      STA   TEMP
      JMP*  TEMP
* ERROR RETURN
SZER CRA
      STA   SZL
      STA   SZW
      STA   SZXC
      STA   SZYC
      STA   SZXG
      STA   SZYG
      JMP   SZRV
SZX  BSS   1
SZY  BSS   1
SZAX BSS   1
SZAY BSS   1
SZHA BSS   1
SZSF BSS   1
SZOA BSS   1
SZR  BSS   1
SZS  BSS   1
SZHR BSS   1
SZLR BSS   1
SZHS BSS   1
SZLS BSS   1
SZXT BSS   1
SZYT BSS   1
SZNP BSS   1
SZQ  BSS   1
SZQ2 BSS   1
SZL  BSS   1
SZW  BSS   1
SZXC BSS   1
SZYC BSS   1
SZXG BSS   1
SZYG BSS   1
*
*
*
*
*
*
*
***CRG,CRL,CRD
*
*
* CONTOUR TRACE ALONG A LIGHT-DARK CONTOUR
* KEEPING AN AREA ON THE RIGHT.
*
* FUNCTION CRL SETS UP A CONTOUR TRACE
* KEEPING THE LIGHT AREA ON THE RIGHT.
*
* FUNCTION CRGT PERFORMS THE TRACE.  IT STARTS FROM
* AN INITIAL LIGHT POINT (XO,YO) WHICH IS ON
```

```
* THE CONTOUR, AND FINDS THE NEXT CONTOUR POINT.
* A THRESHOLD DETERMINES WHETHER A POINT IS
*  CONSIDERED DARK (DK) OR LIGHT (LT).
*
* CRG LIGHT-ON-RIGHT ALGORITHM:
*     -GO COUNTERCLOCKWISE IN LIGHT REGION.
*     -GO CLOCKWISE IN DARK REGION.
*     -IF HIT THREE LIGHT POINTS IN SUCCESSION,
*        ASSUME NEXT POINT IS DARK, AND THUS
*        TAKE A DIAGONAL STEP.
*      -A LIGHT POINT IS OUTPUT WHENEVER THERE IS
*       A LIGHT-DARK TRANSITION (EXCEPT IF THE
*       POINT HAS ALREADY BEEN OUTPUT).
* SEE SEC. 5B OF PRERAU THESIS FOR MORE DETAILS.
* THE ALGORITHM IS SHOWN THERE IN FIG. 5B-4.
*
*
* CRL IS CALLED BY
*  CALL CRL,XO,YO,THSH,PICT
*     WHERE (XO,YO) IS THE INITIAL CONTOUR POINT
*           THSH IS THE THRESHOLD
*           PICT IS THE FIRST LOCATON OF THE CTORED
*              PICTURE.
* IF THE INITIAL POINT IS IN ERROR, CRL RETURNS:
*     A-REGISTER = -1
*     B-REGISTER =  0 IF (XO,YO) IS NOT LIGHT
*                  -1 IF (XO,YO) IS NOT ON THE CONTOUR
*                  -2 IF (XO,YO) IS AN ISOLATED LIGHT POINT.
*
* CRG IS THEN CALLED BY
*  CALL   CRG
* IT RETURNS THE NEXT CONTOUR POINT WITH
*     A-REGISTER = X VALUE
*     B-REGISTER = Y VALUE
* THIS POINT IS NOT ALWAYS THE SAME AS THE POINT THE TRACE IS
* UP TO, WHICH IS STORED IN (CRXT,CRYT).
*
*
* THE INDEX REGISTER IS CHANGED BY CRL AND
* CRG, AND THIS VALUE IS USED BY THE NEXT
* CALL TO CRG
*
*
* CRVL IS AN INTERNAL FUNCTION WHICH FINDS
* WHETHER (CRXT,CRYT) IS DARK OR LIGHT:
*     IF LIGHT (I.E. Z>THSH), RETURN A-REGISTER = 1.
*     IF DARK (I.E. Z<THSH), RETURN A-REGISTER = 0.
*
*           ------------------------------------
*
* FUNCTION CRD SETS UP A CONTOUR TRACE
* KEEPING THE DARK AREA ON THE RIGHT.
*
* FUNCTION CRG PERFORMS THE TRACE IN THIS
* CASE ALSO. IT STARTS FROM AN INITIAL
* DARK POINT (XO,YO) WHICH IS ON THE
* CONTOUR, AND FINDS THE NEXT CONTOUR POINT.
```

```
* CRL SETS UP CRVL TO RETURN
*    A-REGISTER = 1 IF POINT IS DARK.
*    A-REGISTER = 0 IF POINT IS LIGHT.
*
* OTHERWISE , CRD IS THE SAME AS THE
* CRL CASE ABOVE, (EXCEPT THE 'DK' AND 'DARK'
* SHOULD BE INTERCHANGED WITH 'LT' AND 'LIGHT' IN THE
* CRL COMMENTS).
*
*           -------------------------------------
*
*
*
*-------------------
* SET-UP 'CONTOUR TRACE KEEPING
* LIGHT AREA ON RIGHT'
*-------------------
*
*
CRL  DAC*  **
     LDA*  CRL
     STA*  CRX=
     IRS   CRL
     LDA*  CRL
     STA*  CRY=
     IRS   CRL
     LDA*  CRL
     STA*  CRT=
     IRS   CRL
     LDA   CRL
     ANA   ='37777
     STA   CRRL
     LDA*  CRRL
     STA   CRPC     FIRST LOC OF PICT
     IRS   CRRL     RETURN LOCATION
* SET UP CRVL OUTPUTS
     CRA
     STA*  CR=D     CRVL OUTPUT = 0, FOR DK POINT
     AOA
     STA*  CR=L     CRVL OUTPUT = 1, FOR LT POINT
* SET-UP PICTURE REFERENCES
CRPR LDA*  CRPC     ROWS OF PICT
     STA*  CRW=
     SUB   =1
     STA*  CRR=     ROWS - 1
     IRS   CRPC
     LDA*  CRPC
     STA*  CRC=     COLS OF PICT
     LDA   CRPC     PICT(1)-1
     AOA
     STA*  CRP=     PICT(1)-1 + 1
* FIND STARTING STATE
     JST*  CRV=     TEST X0,Y0
     SNZ
     JMP   CRE0     DK:  ERROR --- NOT LIGHT
     LDA*  CRX=
     STA   CRX0     X0
```

```
      LDA*  CRY=
      STA   CRYO      YO
      SUB   =1
      STA*  CRY=
      JST*  CRV=      TEST XO,YO-1
      SNZ
      JMP   CRTI      DK:  TEST FOR ISOLATED POINT
      LDA   CRYO
      AOA
      STA*  CRY=
      JST*  CRV=      TEST XO,YO+1
      SNZ
      JMP   CRSO      DK:  START AT STATE 10
      LDA   CRYO
      STA*  CRY=
      LDA   CRXO
      AOA
      STA*  CRX=
      JST*  CRV=      TEST XO+1,YO
      SNZ
      JMP   CRS1      DK:  START AT STATE 11
      LDA   CRXO
      SUB   =1
      STA*  CRX=
      JST*  CRV=      TEST XO-1,YO
      SNZ
      JMP   CRS2      DK:  STAT AT STATE 12
      LDA   =-1       LT:  ERROR -NOT ON CONTOUR
      IAB
      LDA   =-1
      JMP   CREX
CREO  CRA             ERROR O EXIT
      IAB
      LDA   =-1
      JMP   CREX
* TEST IF ISOLATED DARK POINT
CRTI  LDA   CRYO
      AOA
      STA*  CRY=
      JST*  CRV=      TEST XO,YO+1
      SZE
      JMP   CRS9      LT:  START AT STATE 9
      LDA   CRXO
      SUB   =1
      STA*  CRX=
      JST*  CRV=      TEST XO-1,YO+1
      SZE
      JMP   CRS9      LT:  START AT STATE 9
      LDA   CRYO
      STA*  CRY=
      JST*  CRV=      TEST XO-1,YO
      SZE
      JMP   CRS9      LT:  START AT STATE 9
      LDA   CRXO
      AOA
      STA*  CRX=
      JST*  CRV=      TEST XO+1,YO
```

```
      SZE
      JMP   CRS9    LT:  START AT STATE 9
      LDA   CRY0
      SUB   =1
      STA*  CRY=
      JST*  CRV=    TEST X0+1,Y0-1
      SZE
      JMP   CRS9    LT:  START AT STATE 9
      LDA   CRY0
      AOA
      STA*  CRY=
      JST*  CRV=    TEST X0+1,Y0+1
      SZE
      JMP   CRS1    LT:  START AT STATE 11
      LDA   CRX0
      SUB   =1
      STA*  CRX=
      LDA   CRY0
      SUB   =1
      STA*  CRY=
      JST*  CRV=    TEST X0-1, Y0-1
      SZE
      JMP   CRS1    LT:  START AT STATE 11
      LDA   =-2     DK:  ERROR -- ISOLATED DARK POINT
      IAB
      LDA   =-1
      JMP   CREX
* EXITS
CRS9  LDX   =9
      JMP   CRCE
CRS0  LDX   =10
      JMP   CRCE
CRS1  LDX   =11
      JMP   CRCE
CRS2  LDX   =12
CRCE  CRA           CONTINUE
      STA*  CR=X    OLD X OUTPUT
      STA*  CR=Y    OLD Y OUTPUT
      LDA   CRY0
      STA*  CRY=
      IAB
      LDA   CRX0
      STA*  CRX=
CREX  JMP*  CRRL    EXIT
*
CRX=  DAC   CRXT
CRY=  DAC   CRYT
CRT=  DAC   CRTH
CRX0  BSS   1
CRY0  BSS   1
CRW=  DAC   CRRW
CRC=  DAC   CRCL
CRRL  BSS   1
CR=D  DAC   CR0D
CR=L  DAC   CR0L
CR=X  DAC   CR0X
CR=Y  DAC   CR0Y
```

```
CRR= DAC   CRR1
CRPC BSS   1
CRP= DAC   CRP1
CRV= DAC   CRVL
*
*
*
*-------------------
* SET UP  CONTOUR TRACE KEEPING
*  DARK AREA ON RIGHT
*-------------------
*
CRD  DAC*  **
     LDA*  CRD
     STA*  CRX=
     IRS   CRD
     LDA*  CRD
     STA*  CRY=
     IRS   CRD
     LDA*  CRD
     STA*  CRT=
     IRS   CRD
     LDA   CRD
     ANA   ='37777
     STA   CRRL
     LDA*  CRRL
     STA   CRPC     FIRST LOC OF PICT
     IRS   CRRL     RETURN LOCATION
* SET UP CRVL OUTPUTS
     CRA
     STA*  CR=L     CRVL OUTPUT = 0, FOR LT POINT
     AOA
     STA*  CR=D     CRVL OUTPUT = 1, FOR DK POINT
* USE REST OF CRL ALGORITHMS (INTERCHANGING
* 'LT' AND 'DK' IN CRL AND CRG COMMENTS).
     JMP   CRPR
*
*
     FIN
*
*---------------------------------------------------------
*  GET VALUE OF (CRXT,CRYT)
*---------------------------------------------------------
*
     ORG   '6000
*
CRVL DAC   **
     LDA   CRXT     TEST X BOUNDS
     CAS   CRCL
     NOP
     JMP   CRRO
     SMI
     SPL
     JMP   CRRO
     LDA   CRYT     TEST Y BOUNDS
     CAS   CRRW
     NOP
```

```
       JMP   CRR0
       SPL
       JMP   CRR0
       LDA   CRR1
       SUB   CRYT
       MPY   CRCL
       IAB
       ADD   CRXT
       ADD   CRP1      PTNO= PICT(1)-1 +
       STA   CRPT        (PCRWS-(Y+1))*PCCLS + X+1
       LDA*  CRPT      GET STORED POINT
       CAS   CRTH
       JMP   CRV2      Z>THSH (I.E. LT)
       NOP             Z=THSH (CALLED DK)
       LDA   CROD      Z<THSH (I.E. DK)
       JMP*  CRVL      RETURN A-REG = CROD
CRV2   LDA   CROL
       JMP*  CRVL      RETURN A-REG = CROL
CRR0   CRA             RETURN A-REG = 0
       JMP*  CRVL
*
*
*
*-----------------------------------------------------
*
*
*
*
*
*
*
*----------------------------
* CONTOUR TRACE KEEPING AN AREA ON THE RIGHT
*----------------------------
*
CRG    DAC   **
CRAG   JMP*  *,1       JUMP TO NEXT STATE
       DAC   CR1
       DAC   CR2
       DAC   CR3
       DAC   CR4
       DAC   CR5
       DAC   CR6
       DAC   CR7
       DAC   CR8
       DAC   CR9
       DAC   CR10
       DAC   CR11
       DAC   CR12
       DAC   CR13
       DAC   CR14
       DAC   CR15
       DAC   CR16
* STATE 1:  HIT LT-WENT UP
CR1    JST   CRVL
       SNZ
       JMP   CR1J
```

```
      LDX   =7       LT:  S7,
      LDA   CRXT
      SUB   =1
      STA   CRXT          LEFT,
      JMP   CRAG          AGAIN
CR1J  LDX   =12      DK:  S12,
      LDA   CRXT
      AOA
      STA   CRXT          RIGHT,
      JMP   CRAG          AGAIN.
* STATE 2:  HIT LT-WENT DOWN
CR2   JST   CRVL
      SNZ
      JMP   CR2J
      LDX   =8       LT:  S8,
      LDA   CRXT
      AOA
      STA   CRXT          RIGHT,
      JMP   CRAG          AGAIN.
CR2J  LDX   =11      DK: S11,
      LDA   CRXT
      SUB   =1
      STA   CRXT          LEFT,
      JMP   CRAG          AGAIN.
* STATE 3:  HIT LT-WENT LEFT
CR3   JST   CRVL
      SNZ
      JMP   CR3J
      LDX   =6       LT: S6,
      LDA   CRYT
      SUB   =1
      STA   CRYT          DOWN,
      JMP   CRAG          AGAIN.
CR3J  LDX   =9       DK: S9,
      LDA   CRYT
      AOA
      STA   CRYT          UP,
      JMP   CRAG          AGAIN.
* STATE 4:  HIT LT-WENT RIGHT
CR4   JST   CRVL
      SNZ
      JMP   CR4J
      LDX   =5       LT: S5,
      LDA   CRYT
      AOA
      STA   CRYT          UP,
      JMP   CRAG          AGAIN.
CR4J  LDX   =10      DK: S10,
      LDA   CRYT
      SUB   =1
      STA   CRYT          DOWN,
      JMP   CRAG          AGAIN.
* STATE 5:  HIT LT,LT-WENT UP
CR5   JST   CRVL
      SNZ
      JMP   CR5J
      LDX   =9       LT: S9,
```

```
        LDA   CRXT
        STA   CRXX
        SUB   =1
        STA   CRXT          LEFT,
        LDA   CRYT
        STA   CRYY
        AOA
        STA   CRYT          UP,
        LDA   CRXX
        IAB
        LDA   CRYY
        JMP   CROT          OUTPUT.
CR5J    LDX   =12      DK: S12,
        LDA   CRXT
        STA   CRXX
        AOA
        STA   CRXT          RIGHT,
        LDA   CRXX
        IAB
        LDA   CRYT
        SUB   =1
        JMP   CROT          OUTPUT ADJACENT POINT.
* STATE 6:  HIT LT,LT-WENT DOWN
CR6     JST   CRVL
        SNZ
        JMP   CR6J
        LDX   =10      LT: S10,
        LDA   CRXT
        STA   CRXX
        AOA
        STA   CRXT          RIGHT,
        LDA   CRYT
        STA   CRYY
        SUB   =1
        STA   CRYT          DOWN,
        LDA   CRXX
        IAB
        LDA   CRYY
        JMP   CROT          OUTPUT.
CR6J    LDX   =11      DK: S11,
        LDA   CRXT
        STA   CRXX
        SUB   =1
        STA   CRXT          LEFT,
        LDA   CRXX
        IAB
        LDA   CRYT
        AOA
        JMP   CROT          OUTPUT ADJACENT POINT.
* STATE 7:  HIT LT,LT-WENT LEFT
CR7     JST   CRVL
        SNZ
        JMP   CR7J
        LDX   =11      LT: S11,
        LDA   CRYT
        STA   CRYY
        SUB   =1
```

```
      STA   CRYT        DOWN,
      LDA   CRXT
      STA   CRXX
      SUB   =1
      STA   CRXT        LEFT,
      LDA   CRXX
      IAB
      LDA   CRYY
      JMP   CROT        OUTPUT.
CR7J  LDX   =9      DK: S9,
      LDA   CRYT
      STA   CRYY
      AOA
      STA   CRYT        UP,
      LDA   CRXT
      AOA
      IAB
      LDA   CRYY
      JMP   CROT        OUTPUT ADJACENT POINT.
* STATE 8:  HIT LT,LT-WENT RIGHT
CR8   JST   CRVL
      SNZ
      JMP   CR8J
      LDX   =12     LT: S12,
      LDA   CRYT
      STA   CRYY
      AOA
      STA   CRYT        UP,
      LDA   CRXT
      STA   CRXX
      AOA
      STA   CRXT        RIGHT,
      LDA   CRXX
      IAB
      LDA   CRYY
      JMP   CROT        OUTPUT.
CR8J  LDX   =10     DK: S10,
      LDA   CRYT
      STA   CRYY
      SUB   =1
      STA   CRYT        DOWN,
      LDA   CRXT
      SUB   =1
      IAB
      LDA   CRYY
      JMP   CROT        OUTPUT ADJACENT POINT.
* STATE 9:  HIT DK-WENT UP
CR9   JST   CRVL
      SNZ
      JMP   CR9J
      LDX   =3      LT: S3,
      LDA   CRXT
      STA   CRXX
      SUB   =1
      STA   CRXT        LEFT,
      LDA   CRXX
      IAB
```

```
      LDA   CRYT
      JMP   CROT          OUTPUT.
CR9J  LDX   =16     DK: S16,
      LDA   CRXT
      AOA
      STA   CRXT          RIGHT,
      JMP   CRAG          AGAIN.
* STATE 10:  HIT DK-WENT DOWN
CR10  JST   CRVL
      SNZ
      JMP   CRAJ
      LDX   =4      LT: S4,
      LDA   CRXT
      STA   CRXX
      AOA
      STA   CRXT          RIGHT,
      LDA   CRXX
      IAB
      LDA   CRYT
      JMP   CROT          OUTPUT.
CRAJ  LDX   =15     DK: S15,
      LDA   CRXT
      SUB   =1
      STA   CRXT          LEFT,
      JMP   CRAG          AGAIN.
* STATE 11:  HIT DK-WENT LEFT
CR11  JST   CRVL
      SNZ
      JMP   CRBJ
      LDX   =2      LT: S2,
      LDA   CRYT
      STA   CRYY
      SUB   =1
      STA   CRYT          DOWN,
      LDA   CRXT
      IAB
      LDA   CRYY
      JMP   CROT          OUTPUT.
CRBJ  LDX   =13     DK: S13,
      LDA   CRYT
      AOA
      STA   CRYT          UP,
      JMP   CRAG          AGAIN.
* STATE 12:  HIT DK-WENT RIGHT
CR12  JST   CRVL
      SNZ
      JMP   CRCJ
      LDX   =1      LT: S1,
      LDA   CRYT
      STA   CRYY
      AOA
      STA   CRYT          UP,
      LDA   CRXT
      IAB
      LDA   CRYY
      JMP   CROT          OUTPUT.
CRCJ  LDX   =14     DK: S14,
```

```
      LDA    CRYT
      SUB    =1
      STA    CRYT         DOWN,
      JMP    CRAG         AGAIN.
* STATE 13:  HIT DK,DK-WENT UP
CR13 JST    CRVL
      SNZ
      JMP    CRDJ
      LDX    =3       LT: S3,
      LDA    CRXT
      STA    CRXX
      SUB    =1
      STA    CRXT         LEFT,
      LDA    CRXX
      IAB
      LDA    CRYT
      JMP    CROT         OUTPUT.
CRDJ LDX    =1       DK: S1,
      LDA    CRXT
      AOA
      STA    CRXT         RIGHT,
      LDA    CRYT
      AOA
      STA    CRYT         UP,
      JMP    CRAG         AGAIN.
* STATE 14:  HIT DK,DK-WENT DOWN
CR14 JST    CRVL
      SNZ
      JMP    CREJ
      LDX    =4       LT: S4,
      LDA    CRXT
      STA    CRXX
      AOA
      STA    CRXT         RIGHT,
      LDA    CRXX
      IAB
      LDA    CRYT
      JMP    CROT         OUTPUT.
CREJ LDX    =2       DK: S2,
      LDA    CRXT
      SUB    =1
      STA    CRXT         LEFT,
      LDA    CRYT
      SUB    =1
      STA    CRYT         DOWN,
      JMP    CRAG         AGAIN.
* STATE 15:  HIT DK,DK-WENT LEFT
CR15 JST    CRVL
      SNZ
      JMP    CRFJ
      LDX    =2       LT: S2,
      LDA    CRYT
      STA    CRYY
      SUB    =1
      STA    CRYT         DOWN,
      LDA    CRXT
      IAB
```

```
      LDA   CRYY
      JMP   CROT          OUTPUT.
CRFJ  LDX   =3       DK:  S3,
      LDA   CRYT
      AOA
      STA   CRYT          UP,
      LDA   CRXT
      SUB   =1
      STA   CRXT          LEFT,
      JMP   CRAG          AGAIN.
* STATE 16:  HIT DK,DK-WENT RIGHT
CR16  JST   CRVL
      SNZ
      JMP   CRGJ
      LDX   =1       LT:  S1,
      LDA   CRYT
      STA   CRYY
      AOA
      STA   CRYT          UP,
      LDA   CRXT
      IAB
      LDA   CRYY
      JMP   CROT          OUTPUT.
CRGJ  LDX   =4       DK:  S4,
      LDA   CRYT
      SUB   =1
      STA   CRYT          DOWN,
      LDA   CRXT
      AOA
      STA   CRXT          RIGHT,
      JMP   CRAG          AGAIN.
* PRODUCE OUTPUT
CROT  CAS   CROY     TEST: Y VS. OLD Y
      SKP            OKAY
      JMP   CRO2     Y = OLD Y
      STA   CROY     OKAY
      IAB
      STA   CROX
      JMP*  CRG      RETURN WITH X,Y IN A,B.
CRO2  IAB
      CAS   CROX     TEST: X VS. OLD X
      SKP            OKAY
      JMP   CRAG     X = OLD X. CONTINUE TRACING.
      STA   CROX     OKAY
      JMP*  CRG      RETURN WITH X,Y IN A,B.
*
*
*
CRXX  BSS   1
CRYY  BSS   1
CRPT  BSS   1
CRXT  BSS   1
CRYT  BSS   1
CRTH  BSS   1
CRP1  BSS   1
CROD  BSS   1
CROL  BSS   1
```

```
CRRW BSS    1
CRR1 BSS    1
CRCL BSS    1
CROX BSS    1
CROY BSS    1
*
     FIN
*
*
*
*
***AVG
*
*
*
*
*
* ROUTINE TO RASTER SCAN AREA
* BRTWEEN YTOP,YBOT,XL, AND XR
* AT SPACING DELT AND ADD RESULT
* RIGHT SHIFTED BY 5 TO CONTENTS
* OF LOC IN STD FORM. CALLING
* SEQUENCE AS FOR STO
*       JST    AVG
*       DAC    YTOP
*       DAC    YBOT
*       DAC    XL
*       DAC    XR
*       DAC    DELT
*       DAC    LOC
*
AVEX BSS    1
AVL  BSS    1
AVCC BSS    1
AVOT BSS    1
AVOB BSS    1
AVOL BSS    1
AVOR BSS    1
AVOD BSS    1
AVO  DAC*   **
     LDA*   AVO
     STA    AVOT
     IRS    AVO
     LDA*   AVO
     STA    AVOB
     IRS    AVO
     LDA*   AVO
     STA    AVOL
     IRS    AVO
     LDA*   AVO
     STA    AVOR
     IRS    AVO
     LDA*   AVO
     STA    AVOD
     IRS    AVO
     LDA    AVOT
     OTA    '1052 Y, LONG
```

```
      JMP   *-1
      LDA   AVO
      SSP
      STA   AVEX
      LDA*  AVEX
      STA   AVL
      IRS   AVEX
      AOA
      STA   AVCC
      AOA
      STA   0
      SKP
      IRS*  AVCC
      LDA   *-1
      STA   AVO2
      CRA
      STA*  AVL
      STA*  AVCC
      LDA   AVOL
AVO3  OTA   '352  X, SHORT, GO
      JMP   *-1
      CAS   AVOR
      JMP   AVO4
AVO5  NOP
      ADD   AVOD
      IAB
      INA   '1070
      JMP   *-1
      ARS   7
      ADD   0,1
      STA   0,1
      IRS   0
AVO2  IRS*  AVCC  COLUMN COUNT
      IAB
      JMP   AVO3
* END OF LINE
AVO4  LDA   AVO5   (NOP)
      STA   AVO2   NOP FOR IRS*  STCC
      IRS*  AVL    ROW COUNT
      LDA   AVOT
      SUB   AVOD
      CAS   AVOB
      NOP
      JMP   AVO6
      JMP*  AVEX
AVO6  OTA   '1052 Y, LONG
      JMP   *-1
      STA   AVOT
      JMP   AVO3-1
*
*
*
*
*
*
      FIN
*
```

```
*
*
*
RECT EQU    '7000
LOC  EQU    '20000
*
*
     END
```

```
***LAP
*
* SUBROUTINE TO COMPUTE LAPLACIAN
* OF PICTURE STORED IN STD FORM
* AND STORE LAPLACIAN IN STD FORM
* CALLING SEQUENCE:
*       JST   LAP
*       DAC   LOCP   1ST STORAGE LOCATION OF PICTURE
*       DAC   LOCL   1ST STORAGE LOCATION OF LAPLACIAN
* LOCL MAY BE THE SAME AS LOCP
*
LAP  DAC   **
     LDX*  LAP
     IRS   LAP
     LDA*  LAP
     STA   LAPS
     IRS   LAP
     LDA   0,1       NUMBER OF ROWS
     SUB   =2
     STA*  LAPS
     IRS   LAPS
     TCA
     STA   LAPO      OUTER LOOP COUNT
     LDA   1,1       NUMBER OF POINTS IN ROW
     STA   LAPC
     SUB   =2
     STA*  LAPS
     IRS   LAPS
     TCA
     STA   LAPI      INNER LOOP COUNT
* SET UP INDIRECT ADDRESSES
     LDA   LAPC
     ERA   ='40000
     STA   LAPR
     AOA
     STA   LAPM
     AOA
     STA   LAPL
     ADD   LAPC
     SUB   =1
     STA   LAPB
* OUTER LOOP - BY ROWS
LAPW LDA   LAPI
     STA   LAPN
     IRS   0
     IRS   0
* INNER LOOP - BY POINTS
LAPX LDA*  LAPM
     ALS   2
     SUB*  LAPT
     SUB*  LAPL
     SUB*  LAPR
     SUB*  LAPB
     IRS   0
     STA*  LAPS
     IRS   LAPS
```

```
      IRS   LAPN
      JMP   LAPX
*END OF ROW
      IRS   LAPO
      JMP   LAPW
      JMP*  LAP
LAPO BSS    1
LAPC BSS    1
LAPI BSS    1
LAPS BSS    1
LAPM BSS    1
LAPR BSS    1
LAPL BSS    1
LAPB BSS    1
LAPN BSS    1
LAPT OCT    '40001
      FIN
      END
```

```
***CDC
*
* ROUTINE TO SUM PICTURES ON DISK AND IN CORE
*
* ROUTINE PRODUCES  NX*NY PICTURE I3 IN
* CORE LOCATIONS AT LOC FF. IF I1 IS ON DISK
* IN FILE ASSOCIATED WITH UNIT U AND I2 IS
* IN CORE AT LOC, THEN IT WILL PRODUCE
* I3(X,Y)=(C1*I1(X+DX!,Y+DY!)
*           +C2*I2(X+DX2,Y+DY2))/C3
* CALLING SEQUENCE
* CALL  CDC,U(LITERAL),C1,DY1,DX1,LOC,C2,DY2,DX2
* MDAC  C3,NY,NX
* ROUTINE ASSUMES THAT A BUFFER BUFA OF
* SUFFICIENT LENGTH TO CONTAIN A FULL LINE
* OF THE PICTURE ON DISK HAS BEEN DEFINED.
* INCOMPATIBLE PICTURE DIMENSIONS WILL
* CAUSE ERROR EXIT TO TOP W/ ERROR 700
*
CDC  DAC   **
     LDA*  CDC
     STA   CDRA+1
     STA   CDRB+1
     STA   CDRC+1
     STA   CDRD+1
     IRS   CDC
     LDX*  CDC
     LDA   0,1
     STA   CDCA    =C1
     IRS   CDC
     LDX*  CDC
     LDA   0,1
     STA   CDTA    =DY1
     IRS   CDC
     LDX*  CDC
     LDA   0,1
     STA   CDTB    =DX1
     IRS   CDC
     LDA*  CDC
     STA   CDBA    STORED PICTURE LOC
     IRS   CDC
     LDX*  CDC
     LDA   0,1
     STA   CDCB    =C2
     IRS   CDC
     LDX*  CDC
     LDA   0,1
     STA   CDTC    =DY2
     IRS   CDC
     LDX*  CDC
     LDA   0,1
     STA   CDTD    =DX2
     IRS   CDC
     LDX*  CDC
     LDA   0,1
     STA   CDCC    =C3
```

```
     IRS   CDC
     LDX*  CDC
     LDA   0,1
     STA   CDNY    =NY
     SNZ
     JMP   CDER
     SPL
     JMP   CDER
     IRS   CDC
     LDX*  CDC
     LDA   0,1
     STA   CDNX    =NX
     SNZ
     JMP   CDER
     SPL
     JMP   CDER
     IRS   CDC
* READ DIMENSIONS OF PICTURE ON DISK
* READ FROM DISK
CDRA JST*  '160
     DEC   *     UNIT
     DAC   CDDD  CORE LOC
     DEC   2     NR OF WORDS
* CHECK IF DIMENSIONS ARE COMPATIBLE
     LDA   CDDD
     SUB   CDTA
     SUB   CDNY
     SPL
     JMP   CDER
     AOA
     TCA
     STA   CDWU
     LDA   CDDD+1
     SUB   CDTB
     SUB   CDNX
     SPL
     JMP   CDER
     LDA*  CDBA
     SUB   CDTC
     SUB   CDNY
     SPL
     JMP   CDER
     LDX   CDBA
     LDA   1,1
     STA   CDAD    LINE LENGTH OF PIC IN CORE
     SUB   CDTD
     SUB   CDNX
     SPL
     JMP   CDER
* SKIP OVER UNUSED TOP OF PICTURE ON DISK
     LDA   CDDD+1
     STA   CDRB+3
     STA   CDRC+3
     STA   CDRD+3
     LDA   CDTA
     AOA
```

```
      TCA
      STA    TEMP
CDPB IRS    TEMP
      SKP
      JMP    CDPA
* READ FROM DISK
CDRB JST*   '160
      DEC    *       UNIT
      DAC    BUFA   CORE LOC
      DEC    *       NR OF WORDS
      JMP    CDPB
* SET UP INDIRECT ADDRESSING
* FOR PICTURE IN CORE
CDPA LDA    CDAD
      MPY    CDTC
      IAB             PRODUCT TO A
      ADD    =2
      ADD    CDTD
      ADD    CDNX
      ADD    CDBA
      ERA    ='40000
      STA    CDSP   IND ADDR TO ORIG PIC IN CORE
* FOR PICTURE TO BE CREATED
      LDA    CDBA
      ADD    =2
      ERA    ='40000   FOR POST-INDEXING
      ADD    CDNX
      STA    CDCP   IND ADDR OF NEW PIC
* FOR PICTURE READ OFF DISK
      LDA    CDBF
      ADD    CDTB
      ADD    CDNX
      ERA    ='40000
      STA    CDSF    IND ADDR TO FILE PIC BUFFER
* DIMENSION NEW PICTURE
      LDA    CDNY
      STA*   CDBA
      LDA    CDNX
      LDX    CDBA
      STA    1,1
* LOOP ROW BY ROW
      LDA    CDNY
      TCA
      STA    TEMP
      LDA    CDNX
      TCA
      STA    TEMP+1
* READ FROM DISK
CDRC JST*   '160
      DEC    *       UNIT
      DAC    BUFA   CORE LOC
      DEC    *       NR OF WORDS
* LOOP WITHIN ROW
      LDX    TEMP+1
CDPC LDA*   CDSF
      MPY    CDCA
```

```
      STA    '500
      IAB
      STA    '501
      LDA*   CDSP
      MPY    CDCB
      DBL
      DAD    '500
      SGL
      DIV    CDCC
      STA*   CDCP
      IRS    0
      JMP    CDPC
* END OF ROW REACHED
      LDA    CDSP
      ADD    CDAD
      STA    CDSP
      LDA    CDCP
      ADD    CDNX
      STA    CDCP
      IRS    TEMP
      JMP    CDRC
* SKIP UNUSED ROWS OF PICTURES ON DISK
CDPD IRS    CDWU
      SKP
      JMP*   CDC
* READ FROM DISK
CDRD JST*   '160
      DEC    *      UNIT
      DAC    BUFA   CORE LOC
      DEC    *      NR OF WORDS
      JMP    CDPD
*
* ERROR EXIT
CDER LDA    =700
      JMP*   '166   TYPE ERROR NR, GO TO TOP
CDCA BSS    1
CDTA BSS    1
CDTB BSS    1
CDBA BSS    1
CDCB BSS    1
CDTC BSS    1
CDTD BSS    1
CDCC BSS    1
CDNY BSS    1
CDNX BSS    1
CDDD BSS    2
CDAD BSS    1
CDCP BSS    1
CDSF BSS    1
CDWU BSS    1
CDSP BSS    1
CDBF DAC    BUFA
      FIN
      END
```

```
* ROUTINE TO TEST PRR
BEG  JST   PRR
     DAC   BLOK
     DEC   3
     BCI   3,ABCDEF
     JMP   BEG
BLOK DEC   3
     DEC   5
     BCI   2,ABCD
ABCD BSS   5
     BCI   2,EFGH
EFGH BSS   5
     BCI   2,IJKL
IJKL BSS   5
MCH= DAC   MCH
     FIN
*
*
*
***PRR
*
*  PARAMETER READING ROUTINE
*
* THIS ROUTINE READS PARAMETERS
* INTO A TABLE.  IT TYPES MESSAGE
* DESCRIBING PARAMETER SET, REQUESTS
* AND ACCEPTS NUMBER OF SET, THEN
* TYPES 'WHICH' AND ACCEPTS
* 'ALL', TYPES PARAMETER NAMES IN
*   SEQUENCE AND ACCEPTS VALUE FOR EACH
* 'NOW', TYPES PARAMETER NAMES AND
*   THEIR CURRENT (DECIMAL) VALUES
* 'COPY', TYPES 'SET NR. ', ACCEPTS
*   NUMBER AND COPIES THAT SET INTO
*   CURRENT ONE, TYPES OUT VALUES
* 'NEXT', ASKS FOR NEW SET NR., ETC.
* 'NONE', EXITS PRR ROUTINE
* BLANKS AND/OR CARRIAGE RETURN, EXITS
* NAME OF ANY PARAMETER IN SET, TYPES ' = ',
*   ACCEPTS VALUE
* ANYTHING ELSE - TYPES QUESTION MARK, WHICH
*   WAITS FOR NEW INPUT
* ALL NUMBERS ASSUMED DECIMAL UNLESS FLAGGED
* AS OCTAL BY PRECEDING '.
*  CALLING SEQUENCE
*    JST   PRR
*    DAC   BLOK           PARAMETER BLOK
*    DEC   N
*    BCI   N,MESSAGE TO BE TYPED
*  BLOK IS THE PARAMETER BLOK
*  SET UP BY MACRO
*  BLOK PARAMETERS N,M,PAR1,PAR2,...PARN
*  THIS RESULTS IN
*  BLOK DEC   N
*       DEC   M
*       BCI   2,PAR1
```

```
*   PAR1 BSS    M
*        BCI    2,PAR2
*   PAR2 BSS    M
*   ...
*   PARN BSS    M
*   IF NUMBER OF SET REQUESTED >/= M,
*   QUESTION MARK WILL BE TYPED
*   AND NEW NUMBER AWAITED.
PRR  DAC    **
     LDA*   PRR
     STA    PRBL
     IRS    PRR
     LDA*   PRR
     TCA
     STA    PRTM     WORDS OF MESSAGE
     IRS    PRR
     LDA    PRR
     STA    PRTM+1   MESSAGE ADDRESS
     SUB    PRTM
     STA    PRR
     JST*   '152    TYPE MESSAGE
PRTM BSS    2
     LDA*   PRBL
     TCA
     STA    PRPN     NUMBER OF PARAMETERS
     IRS    PRBL
     LDA*   PRBL
     STA    PRNS     NUMBER OF SETS
     IRS    PRBL
PRPK LDA    ='212
     JST*   '145
     JST*   '152     TYPE 'SET NR. '
     DEC    -4
     DAC    PRMA
PRPB JST    RNR
     STA    PRSN     SET NUMBER
     LDA    PRNS
     CAS    PRSN
     JMP    PRPA
     NOP
     LDA    ='277
     JST*   '145     TYPE QUESTION MARK
     LDA    ='240
     JST*   '145
     JMP    PRPB     LOOK FOR VALID SET NR
PRPA LDA    ='212
     JST*   '145
     JST*   '151     TYPE WHICH  , C.R.
     DEC    -3
     DAC    PRMB
PRPD JST    RMR      INPUT READER
     DEC    -2
     DAC    PRMC
     JST*   MCH=   TEXT COMP. - SKP IF N.E.
     DEC    -2
     DAC    PRMC
```

```
      BCI   2,ALL
      JMP   PRPC
      JST*  MCH=  TEXT COMP. - SKP IF N.E.
      DEC   -2
      DAC   PRMC
      BCI   2,NOW
      JMP   PRPH
      JST*  MCH=  TEXT COMP. - SKP IF N.E.
      DEC   -2
      DAC   PRMC
      BCI   2,COPY
      JMP   PRPE
      JST*  MCH=
      DEC   -2
      DAC   PRMC
      BCI   2,NEXT
      JMP   PRPK
      JST*  MCH=  TEXT COMP. - SKP IF N.E.
      DEC   -2
      DAC   PRMC
      BCI   2,NONE
      JMP*  PRR
      JST*  MCH=  TEXT COMP. - SKP IF N.E.
      DEC   -2
      DAC   PRMC
      BCI   2,
      JMP*  PRR
*  COMPARE AGAINST NAMES IN BLOCK
      LDX   PRPN
      LDA   PRBL
PRPI  STA   PRBA
      JST*  MCH=
      DEC   -2
PRBA  BSS   1
PRMC  BSS   2
      JMP   PRFD    FOUND NAME
      LDA   PRBA
      ADD   =2
      ADD   PRNS
      IRS   0
      JMP   PRPI
      LDA   ='277
      JST*  '145    TYPE QUESTION MARK
      JMP   PRPA
*  ACCEPT VALUE IF NAME FOUND
PRFD  LDA   ='275
      JST*  '145
      LDA   ='240
      JST*  '145
      LDA   PRBA
      ADD   =2
      ADD   PRSN
      STA   PRSI
      JST   RNR
      STA*  PRSI
      LDA   ='212
```

```
      JST*  '145
      JMP   PRPD     FOR NEXT INPUT
*  IF  ALL  TYPED, TYPE NAMES AND ACCEPT VALUES
PRPC LDA   ='212
      JST*  '145
      LDA   PRPN
      STA   PRPU
      LDA   PRBL
      STA   PRBB
PRPF JST*  '152
      DEC   -2
PRBB DAC   **
      JST*  '152
      DEC   -1
      DAC   PRMD
      LDA   ='240
      JST*  '145
      LDA   PRBB
      ADD   =2
      ADD   PRSN
      STA   PRBN
      JST   RNR
      STA*  PRBN
      LDA   PRBB
      ADD   =2
      ADD   PRNS
      STA   PRBB
      IRS   PRPU
      JMP   PRPF
      JMP*  PRR
*  IF  NOW  TYPED, TYPE NAMES AND VALUES
PRPH LDA   ='212
      JST*  '145
      LDA   PRPN
      STA   PRPU
      LDA   PRBL
PRPG STA   PRBC
      JST*  '152
      DEC   -2
PRBC DAC   **
      JST*  '152
      DEC   -1
      DAC   PRMD
      LDA   ='240
      JST*  '145
      LDA   PRBC
      ADD   =2
      ADD   PRSN
      STA   PRBN
      LDA*  PRBN
      JST*  '154
      LDA   ='212
      JST*  '145
      LDA   PRBC
      ADD   =2
      ADD   PRNS
```

```
      IRS   PRPU
      JMP   PRPG
      JMP   PRPA
*  IF  COPY  TYPED
*  GET NUMBER OF SET TO COPY, COPY IT
PRPE LDA   ='240
     JST*  '145
     JST*  '152
     DEC   -4
     DAC   PRMA
     JST   RNR
     ADD   PRBL
     ADD   =2
     STA   PRCF
     LDA   PRPN
     STA   PRPU
     LDA   PRBL
     ADD   PRSN
PRPJ ADD   =2
     STA   PRBN
     LDA*  PRCF
     STA*  PRBN
     LDA   PRCF
     ADD   =2
     ADD   PRNS
     STA   PRCF
     LDA   PRBN
     ADD   PRNS
     IRS   PRPU
     JMP   PRPJ
     JMP   PRPH
PRBL BSS   1
PRPN BSS   1
PRSN BSS   1
PRNS BSS   1
PRSI BSS   1
PRBN BSS   1
PRPU BSS   1
PRCF BSS   1
     FIN
PRMA BCI   4,SET NR.
PRMB BCI   3,WHICH
PRMD BCI   1, =
*
*
*
***RNR
*
* ROUTINE TO READ AND LEAVE IN A
* EITHER DECIMAL NUMBER OR OCTAL
* NUMBER MARKED BY PRECEDING '.
*
RNR  DAC   **
     JST*  '147
     SZE
     JMP*  RNR
```

```
      IAB
      CAS    ='177531
      JMP    RNP
      SKP
      JMP    RNP
      JST*   '146
      JMP*   RNR
RNP   CRA
      JMP*   RNR
      FIN
*
*
*
***RMR
*
* READ MESSAGE ROUTINE
* CALLED BY
*        JST    RMR
*        DEC    -N
*        DAC    LOC
* ROUTINE READS UP TO 2N CHARACTERS
* INTO LOCATIONS BEGINNING AT LOC,
* 2 CHARS/WORD. RETURNS AFTER 2N CHARS
* OR IF CAR.RET. TYPED. IF CAR.RET.
* BEFORE 2N-TH CHARACTER, LOCATIONS
* FILLED OUT WITH BLANKS.
*
RMR   DAC    **
      LDA*   RMR
      STA    0
      STA    RMN
      IRS    RMR
      LDA*   RMR
      SUB    RMN
      ERA    ='40000
      STA    RML
      LDA    ='120240  BLANKS
      STA*   RML
      IRS    0
      JMP    *-2
      IRS    RMR
      LDX    RMN
* READ ODD CHARACTERS
RMPB  JST*   '144
      CAS    ='212
      SKP
      JMP*   RMR
      ICR
      STA    RMS
* READ EVEN CHARACTERS
      JST*   '144
      CAS    ='212
      SKP
      JMP    RMPA
      ERA    RMS
      STA*   RML
```

```
      IRS    0
      JMP    RMPB
      JMP*   RMR
* FILL OUT WORD WITH BLANK
RMPA  LDA    ='240
      ERA    RMS
      STA*   RML
      JMP*   RMR
RMN   BSS    1
RML   BSS    1
RMS   BSS    1
      FIN
*
*
*
***MCH
*
* ROUTINE TO COMPARE TWO TEXTS
* CALLED BY
* NAME MATCH N,LOC,TEXT
* WHICH EXPANDS TO
* NAME JST*  MCH=
*       DEC   -N
*       DAC   LOC  [OR DAC*  LOC IF LOC* IN MACRO]
*       BCI   N,TEXT
* MCH WILL COMPARE N WORDS STORED AT LOC FF.
* WITH FIRST 2N CHARACTERS OF TEXT, RETURN
* DIRECTLY IF THEY MATCH, WITH SKIP IF NOT.
*
MCH   DAC    **
      LDA*   MCH
      STA    MCA
      IRS    MCH
      LDA*   MCH
      STA    MCB
      IRS    MCH
MCP   LDA*   MCH
      CAS*   MCB
      SKP
      SKP
      JMP    MCR
      IRS    MCH
      IRS    MCB
      IRS    MCA
      JMP    MCP
      JMP*   MCH
MCR   IRS    MCH
      IRS    MCA
      JMP    MCR
      IRS    MCH
      JMP*   MCH
MCA   BSS    1
MCB   BSS    1
      END
```

```
*
* ROUTINE TO TEST LANDMARK INSERTION PROGRAM
* CALLS AND USES LMI
      ORG    '1000
      STRT
BEG   LDA    =-1
      LDX    =-21
PTA   STA    LMIX+21,1
      IRS    0
      JMP    PTA
      JST    LMI
      SS2
      JMP    BEG
* REQUEST AND READ LMTA
      JST*   *+4
      BCI    3,LMTA
      DAC    >RR<
      STA    LMTA
* REQUEST AND READ LMBA
      JST*   *+4
      BCI    3,LMBA
      DAC    >RR<
      STA    LMBA
* REQUEST AND READ LMLA
      JST*   *+4
      BCI    3,LMLA
      DAC    >RR<
      STA    LMLA
* REQUEST AND READ LMRA
      JST*   *+4
      BCI    3,LMRA
      DAC    >RR<
      STA    LMRA
* REQUEST AND READ LMDA
      JST*   *+4
      BCI    3,LMDA
      DAC    >RR<
      STA    LMDA
* REQUEST AND READ LMDX
      JST*   *+4
      BCI    3,LMDX
      DAC    >RR<
      STA    LMDX
* REQUEST AND READ LMDY
      JST*   *+4
      BCI    3,LMDY
      DAC    >RR<
      STA    LMDY
* REQUEST AND READ LMTC
      JST*   *+4
      BCI    3,LMTC
      DAC    >RR<
      STA    LMTC
* REQUEST AND READ LMLC
      JST*   *+4
      BCI    3,LMLC
      DAC    >RR<
```

```
      STA   LMLC
* REQUEST AND READ LMDC
      JST*  *+4
      BCI   3,LMDC
      DAC   >RR<
      STA   LMDC
      JMP   BEG
RNR= DAC   RNR
CSS= DAC   CSS
CSD= DAC   CSD
ILC= DAC   ILC
DIS= DAC   DIS
STO= DAC   STO
AVG= DAC   AVO
*
*
* PARAMETER REQUEST AND READ ROUTINE
      BCI   1,=
      OCT   '177531  ='
>RR< DAC   **
      LDA   ='212
      JST*  '145
      LDA   >RR<
      STA   *+5
      ADD   =4
      STA   >RR<
      JST*  '152
      DEC   -3
      DAC   **
      JST*  '152
      DEC   -1
      DAC   >RR<-2
      JST*  '147
      SNZ
      SKP
      JMP*  >RR<
      IAB
      CAS   >RR<-1
      SKP
      JMP   *+3
      CRA
      SKP
      JST*  '146
      JMP*  >RR<
      FIN
***LMI
*
* ROUTINE TO LET OPERATOR ENTER LANDMARKS
*
* OPERATION
* 1) FULL FRAME IS DISPLAYED WITH TRIM IN
*    READ. LANDMARKS ENTERED SO FAR SHOW
*    AS CROSSES. FINE RESOLUTION SCAN MARKS
*    POSITION OF STYLUS.
* 2) WHEN STYLUS IS DEPRESSED, SURROUNDING
*    AREA IS DIGITIZED, DISPLAYED ENLARGED
*    WITH TRIM IN DISPLAY MODE. CROSS MARKS
```

```
*     LOCATION OF STYLUS.
* 3) WHEN STYLUS IS DEPRESSED, ITS LOCATION
*     IN TERMS OF THE FRAME COORDINATES IS
*     COMPUTED AND ASSUMED TO BE THE LOCATION
*     OF A LANDMARK. TELETYPE REQUESTS 'NR'.
*     IF 0 TYPED, LANDMARK LOCATION IS ASSUMED
*     INVALID, OTHERWISE COORDINATES ARE STORED
*     IN ARRAYS LMIX AND LMIY, STEP 1 IS REPEATED.
*     NO MORE THAN 20 LANDMARKS MAY BE ENTERED.
* RAISING SS1 DURING DISPLAY OF FULL FRAME
* (STEP 1) CAUSES EXIT FROM ROUTINE.
* ROUTINE IS CALLED BY
*        JST    LMI
*
LMI  DAC    **
* DISPLAY FULL FRAME
* CALL SUBROUTINE CSS
     JST*   CSS=
     DAC    ='67
     DAC    =1
     DAC    ='3777
* CALL SUBROUTINE ILC
LMPA JST*   ILC=
     DAC    LMTA
     DAC    LMBA
     DAC    LMLA
     DAC    LMRA
     DAC    LMDA
* LEAVE LMI IF SS1 IS UP
     SR1
     JMP*   LMI
* DISPLAY LANDMARKS IN NOW
     LDA    =-20
     LDX    =1
     STA    LMTM
LMPB LDA    LMIX,1
     SPL
     JMP    LMPH
     STX    LMTX
     JST*   CSD=
     DAC    LMIX,1
     DAC    LMIY,1
     LDX    LMTX
LMPH IRS    0
     IRS    LMTM
     JMP    LMPB
* CHECK TABLET
LMPC LDX    =0  'STYLUS DOWN' MARKER
     INA    '1151   READ TABLET X
     JMP    *-1
     LLR    3
     ARS    2
     STA    LMTX    TABLET X
     INA    '1251   READ TABLET Y
     JMP    *-1
     LLR    3
     ARS    2
```

```
       STA    LMTY       TABLET Y
       IAB
       STA    *+2
       SKP
       BSS    1
       ANA    ='33
       SZE
       JMP    *-16       TABLET ERROR
       LDA    *-4
       ANA    ='44
       SZE               SKIP ON STYLUS DOWN
       LDX    =-1     'STYLUS UP' MARKER
* GET LIMITS FOR SMALL AREA
       LDA    LMDX
       ARS    1
       ADD    LMTX
       STA    LMRB
       SUB    LMDX
       STA    LMLB
       LDA    LMDY
       ARS    1
       ADD    LMTY
       STA    LMTB
       SUB    LMDY
       STA    LMBB
       IRS    0    GENERATE SKIP IF STYLUS IS UP
       JMP    LMPD
* INTENSIFY AREA NEAR STYLUS (IF UP)
* CALL SUBROUTINE ILC
       JST*   ILC=
       DAC    LMTB
       DAC    LMBB
       DAC    LMLB
       DAC    LMRB
       DAC    =6
       JMP    LMPA
* DIGITIZE SMALL AREA (IF STYLUS DOWN)
* CALL SUBROUTINE STO
LMPD JST*   STO=
       DAC    LMTB
       DAC    LMBB
       DAC    LMLB
       DAC    LMRB
       DAC    =1
       DAC    PICT
       LDA    =-7
       STA    LMTM
* CALL SUBROUTINE AVG
LMPE JST*   AVG=
       DAC    LMTB
       DAC    LMBB
       DAC    LMLB
       DAC    LMRB
       DAC    =1
       DAC    PICT
       IRS    LMTM
       JMP    LMPE
```

```
 * DISPLAY FROM CORE, TRACK STYLUS
 * CALL SUBROUTINE DIS
 LMPF JST*  DIS=
      DAC   LMTC
      DAC   LMLC
      DAC   LMDC
      DAC   PICT
 * CALL SUBROUTINE CSD
      JST*  CSD=
      DAC   LMTX
      DAC   LMTY
      INA   '1151     READ TABLET X
      JMP   *-1
      LLR   3
      ARS   2
      STA   LMTX      TABLET X
      INA   '1251     READ TABLET Y
      JMP   *-1
      LLR   3
      ARS   2
      STA   LMTY      TABLET Y
      IAB
      STA   *+2
      SKP
      BSS   1
      ANA   ='33
      SZE
      JMP   *-16      TABLET ERROR
      LDA   *-4
      ANA   ='44
      SZE             SKIP ON STYLUS DOWN
      JMP   LMPF
 * COMPUTE AND STORE LOCATION OF
 * DEPRESSED STYLUS IN FRAME COORDINATES
 * READ COORDINATE NUMBER
 * TYPE 'LNDMK NR = '
 LMPG JST*  '152
      DEC   -06
      DAC   *+2
      JMP   *+1+06
      BCI   06,LNDMK NR =
      JST*  RNR=
      SNZ
      JMP   LMPA
      SPL
      JMP   LMER
      CAS   =20
      JMP   LMER
      NOP
      STA   0
      LDA   LMTX
      SUB   LMLC
      IAB
      CRA
      DIV   LMDC
      ADD   LMLB
      STA   LMIX,1
```

```
       LDA   LMTC
       SUB   LMTY
       IAB
       CRA
       DIV   LMDC
       TCA
       ADD   LMTB
       STA   LMIY,1
       JMP   LMPA
* IF COORDINATE NUMBER <0 OR >20,TRY AGAIN
LMER LDA   ='277  QUEST MK
     JST*  '145
     LDA   ='212
     JST*  '145
     JMP   LMPG
LMIX BSS   21
LMIY BSS   21
LMTX BSS   1
LMTY BSS   1
LMRB BSS   1
LMLB BSS   1
LMTB BSS   1
LMBB BSS   1
LMTM BSS   1
LMTA OCT   3200
LMBA OCT   600
LMLA OCT   600
LMRA OCT   3200
LMDA OCT   10
LMDX OCT   100
LMDY OCT   100
LMTC OCT   3000
LMLC OCT   1000
LMDC OCT   20
PICT EQU   '20000
     FIN
```

```
***PDC
*
* ROUTINE TO COMPARE PICTURES
* PICTURE 1 ON DISK, PICTURE 2 IN CORE
* NX*NY SECTIONS ARE COMPARED BY PRODUCT RULE
* DOUBLE PRECISION RESULT LEFT IN REGISTERS A,B
* IS SUM OVER X AND Y OF
* I1(X+DX!,Y+DY1)*I2(X+DX2,Y+DY2)
* INCOMPATIBLE SIZE SPECIFICATION RESULTS
* IN MESSAGE 'ERROR 701' AND RETURN TO TOP.
* OVERFLOW DURING SUMMING RESULTS IN C BIT
* BEING SET WHEN ROUTINE RETURNS.
* CALLING SEQUENCE:
*   CALL  PDC,U,DY1,DX1,LOC,DY2,DX2,NY,NX
* U IS LITERAL UNIT NUMBER FOR DISK FILE.
* FILE MUST BE OPEN. LOC IS LOCATION OF
* PICTURE IN CORE IN STD FORM. ROUTINE
* ASSUMES A BUFFER CALLED BUFA OF SUFFICIENT
* LENGTH TO HOLD ONE FULL LINE OF PICTURE
* ON DISK HAS BEEN DEFINED FOR ITS USE.
PDC  DAC   **
     LDA*  PDC
     STA   PDRA+1
     STA   PDRB+1
     STA   PDRC+1
     STA   PDRD+1
     IRS   PDC
     LDX*  PDC
     LDA   0,1
     STA   PDTA   =DY1
     IRS   PDC
     LDX*  PDC
     LDA   0,I
     STA   PDTB   =DX1
     IRS   PDC
     LDA*  PDC
     STA   PDBA   STORED PICTURE LOC
     IRS   PDC
     LDX*  PDC
     LDA   0,1
     STA   PDTC   =DY2
     IRS   PDC
     LDX*  PDC
     LDA   0,1
     STA   PDTD   =DX2
     IRS   PDC
     LDX*  PDC
     LDA   0,1
     STA   PDNY   =NY
     SNZ
     JMP   PDER
     SPL
     JMP   PDER
     IRS   PDC
     LDX*  PDC
     LDA   0,1
```

```
      STA   PDNX    =NX
      SNZ
      JMP   PDER
      SPL
      JMP   PDER
      IRS   PDC
* READ DIMENSIONS OF PICTURE ON DISK
* READ FROM DISK
PDRA JST*  '160
      DEC   *      UNIT
      DAC   PDDD   CORE LOC
      DEC   2      NR OF WORDS
* CHECK IF DIMENSIONS ARE COMPATIBLE
      LDA   PDDD
      SUB   PDTA
      SUB   PDNY
      SPL
      JMP   PDER
      AOA
      TCA
      STA   PDWU
      LDA   PDDD+1
      SUB   PDTB
      SUB   PDNX
      SPL
      JMP   PDER
      LDA*  PDBA
      SUB   PDTC
      SUB   PDNY
      SPL
      JMP   PDER
      LDX   PDBA
      LDA   1,1
      STA   PDAD    LINE LENGTH OF PIC IN CORE
      SUB   PDTD
      SUB   PDNX
      SPL
      JMP   PDER
* SKIP OVER UNUSED TOP OF PICTURE ON DISK
      LDA   PDDD+1
      STA   PDRB+3
      STA   PDRC+3
      STA   PDRD+3
      LDA   PDTA
      AOA
      TCA
      STA   TEMP
PDPB IRS   TEMP
      SKP
      JMP   PDPA
* READ FROM DISK
PDRB JST*  '160
      DEC   *      UNIT
      DAC   BUFA   CORE LOC
      DEC   *      NR OF WORDS
      JMP   PDPB
```

```
* RESET SUM
PDPA CRA
     STA    '500
     STA    '501
     STA    PDEX  OVERFLOW MARKER
* SET UP INDIRECT ADDRESSING
* FOR PICTURE IN CORE
     LDA    PDAD
     MPY    PDTC
     IAB
     ADD    =2
     ADD    PDTD
     ADD    PDNX
     ADD    PDBA
     ERA    ='40000
     STA    PDSP  IND ADDR TO ORIG PIC IN CORE
* FOR PICTURE READ OFF DISK
     LDA    PDBF
     ADD    PDTB
     ADD    PDNX
     ERA    ='40000
     STA    PDSF   IND ADDR TO FILE PIC BUFFER
* LOOP ROW BY ROW
     LDA    PDNY
     TCA
     STA    TEMP
     LDA    PDNX
     TCA
     STA    TEMP+1
* READ FROM DISK
PDRC JST*   '160
     DEC    *     UNIT
     DAC    BUFA  CORE LOC
     DEC    *     NR OF WORDS
* LOOP WITHIN ROW
     LDX    TEMP+1
PDPC DBL
     IMA*   PDSF
     MPY*   PDSP
     DAD    '500
     SRC
     IRS    PDEX  OVERFLOW
     DST    '500
     IRS    0
     JMP    PDPC
     SGL
* END OF ROW REACHED
     LDA    PDSP
     ADD    PDAD
     STA    PDSP
     IRS    TEMP
     JMP    PDRC
* SKIP UNUSED ROWS OF PICTURES ON DISK
PDPD IRS    PDWU
     SKP
     JMP    PDPE
```

```
* READ FROM DISK
PDRD JST*   '160
     DEC    *      UNIT
     DAC    BUFA   CORE LOC
     DEC    *      NR OF WORDS
     JMP    PDPD
* EXIT
PDPE SCB
     LDA    PDEX
     SNZ
     RCB
     LDA    '501
     IAB
     LDA    '500
     JMP*   PDC
*
* ERROR EXIT
PDER LDA    =700
     JMP*   '166   TYPE ERROR NR, GO TO TOP
PDTA BSS    1
PDTB BSS    1
PDBA BSS    1
PDTC BSS    1
PDTD BSS    1
PDNY BSS    1
PDNX BSS    1
PDDD BSS    2
PDAD BSS    1
PDSF BSS    1
PDWU BSS    1
PDSP BSS    1
PDEX BSS    1
PDBF DAC    BUFA
```

```
***PAR
*
* ROUTINE TO COMPUTE SUM OF SQUARES OF
* INTENSITIES OF NY*NX SECTION OF A PICTURE
* AT PIC, DY AND DX FROM UPPER LEFT CORNER,
* LEAVE ANSWER AS DOUBLE PRECISION NUMBER IN
* ANS AND ANS+1.
* CALLING SEQUENCE:
*  CALL  SSP,PIC,NY,NX,DY,DX,ANS
*
PAR  DAC   **
     LDX*  PAR
     LDA   0,1
     STA   PATY  NR OF ROWS IN PICTURE
     LDA   1,1
     STA   PATX  NR OF COLUMNS IN PICTURE
     STX   TEMP
     IRS   PAR
     LDX*  PAR
     LDA   0,1
     TCA
     STA   PANY
     IRS   PAR
     LDX*  PAR
     LDA   0,1
     TCA
     STA   PANX
     IRS   PAR
     LDX*  PAR
     LDA   0,1
     MPY   PATX
     IAB
     ADD   TEMP
     ADD   ='40002
     IRS   PAR
     LDX*  PAR
     ADD   0,1
     SUB   PANX
     STA   PAAD  IND ADDR OF PICTURE, POST INDEXED
     IRS   PAR
     LDA*  PAR
     STA   PATO
     IRS   PAR
* COMPUTE SUM OF SQUARES
     CRA
     STA   '500
     STA   '501
PAPF LDX   PANX
PAPC LDA*  PAAD
     MPY*  PAAD
     DBL
     ADD   '500
     STA   '500
     SGL
     IRS   0
     JMP   PAPC
* END OF ROW
```

```
      IRS   PANY
      JMP   PAPD
* STORE RESULTS AND EXIT
      LDX   PATO
      LDA   '500
      STA   0,1
      LDA   '501
      STA   1,1
      JMP*  PAR
* GO TO NEXT ROW
PAPD  LDA   PAAD
      ADD   PATX
      STA   PAAD
      JMP   PAPF
PATY  BSS   1
PATX  BSS   1
PANY  BSS   1
PANX  BSS   1
PAAD  BSS   1
PATO  BSS   1
      FIN
```

APPENDIX B

TRIM OPERATING PROGRAM

APPENDIX B
TRIM OPERATING PROGRAM

The TRIM Operating Program (TOP) is the operating system that is being used on our Honeywell DDP-516. The following routines are provided:

TELETYPE UTILITY ROUTINES

The following teletype routines are provided by TOP.

1. LT11
   call by: JST* '144
   operation: Accept one character
   Character is typed in through keyboard into right half of A. Nulls are ignored. Carriage return ('215) is converted into line feed ('212) and line feed is typed out as well.

   A question mark typed in will not return in A, but will cause an immediate transfer to the TOP executive.

2. LT01
   call by: JST* '145
   operation: Type out a character
   Character in right half of A is typed out. Nulls are ignored; line feed types carriage return as well.

3. IOCT
   call by: JST* '146
   operation: Accept octal number
   A signed 16 bit octal integer terminated by any character not 0-7 is accepted into A; the terminating character returns in B. If the first character typed in is non-numeric (terminating), zero returns in A, - that character in B.

4. IDEC
   call by: JST* '147
   operation: Accept decimal number
   Same as IOCT, except to radix ten.

5. IRAD
   call by: LDA RADX
   JST* '150
   operation: Accept arbitrary radix number.
   Radix in A entering, otherwise similar to IOCT.

6. OOCT
   call by: JST* '153
   Type out octal number.

```
     operation:  Contents of A are typed out as a (16 bit)
                 octal integer.

 7.  ODEC        Type out decimal number
     call by:    JST*   '154
     operation:  Contents of A are typed out as (15 bit + sign,
                 two's complement) decimal number.

 8.  ORAD        Type out arbitrary radix number
     call by:    JST*   '155
     operation:  Type out contents of A to radix in B. if
                 B is positive, the output amounts to 15
                 bits plus sign, if B negative, the output
                 amounts to 16 bits positive.

 9.  TONA        Type out character string
     call by:    JST*   '152
                 DEC    -N
                 DAC    ARRY
     operation:  Characters are typed out starting from location
                 ARRT; -N is the two's complement of the number
                 of words whose contents will be typed.

10.  TON         Type out character string and carriage return.
     call by:    JST*   '151
                 DEC    -N
                 DAC    ARRY
     operation:  Same as TONA, with a carriage return ('212)
                 typed at the end.

11.  GETN        Accept file name from teletype
     call by:    JST*   '167
                 DAC    NAM
     operation:  Up to six characters will be accepted, ter-
                 minated by a carriage return.  The name will
                 be left justified in the three word buffer
                 NAM with spaces ('240) added to make it six
                 characters long.  The carriage return will not
                 be part of the name.
```

DISK INPUT-OUTPUT ROUTINES

There are three subroutines which provide an interface between user programs and the disk: Search, Read, and Write.

1\. SRCX — Search

call by:

```
JST*   '162
DEC    KEY
DAC    NAM
DEC    UNIT
```

operation: This routine performs one of five functions depending on KEY.

KEY = 1 — Open a file for reading. The directory is searched for a file with name as in NAM. If found, the file will be associated with UNIT (1-4) for reading purposes. If no file is found, an error message (8) will result.

KEY = 2 — Open a file for writing. The directory is searched for a file with name as in NAM. If no file is found, a new one will be created. Unit UNIT (1-4) will be associated with the file for writing purposes.

Key = 3 — Open a file for reading and writing; the same as Key = 2 except that reading will be allowed as well.

Key = 4 — Close a unit. If the unit is open for any purpose, its association with whatever file (i.e. NAME is irrelevant) will be terminated. If the unit is not open, SRCX with KEY = 4 produces no result.

Key = 5 — Delete a file. The directory is searched for a file with name as in NAM; if found it is a checked to be not open. If so, the file entry will be deleted from the directory, and the records forming the file will be released to free storage. The unit number is irrelevant for deletion.

2\. REAX — Read from disk

call by:

```
JST*   '160
DEC    UNIT
DAC    ARY
DEC    NWDS
```

operation: REAX reads NWDS words from the file associated with UNIT (must be open for read or read/write) into successive location beginning with ARY. Attempts to read with a unit not open or to read past the end of the file will result in error messages (e or 9).

3. WRIX — Write on disk

call by:
```
JST*   '161
DEC    UNIT
DAC    ARY
DEC    NWDS
```
operation: WRIX writes NWDS words on the file associated with UNIT (must be open for write or read/write) starting at location ARY. Attempts to write with a unit not open will result in error message 3.

RETURNS

1. RETURN

call by:
```
JMP*   '156
```
operation: Returns to TOP from a user program.

2. ERROR RETURN

call by:
```
LDA = NN
JMP   '166
```
operation: This will type "ERROR NN" and return to TOP from a user program.

APPENDIX C

THE MACRO PROCESSING PROGRAM

```
*
*
*
* MACRO COMPILER PROGRAM
* TAILORED AFTER ORGASS SIMCMP
*THIS PROGRAM WAS LAST CHANGED ON FEBRUARY 22,1971
* SEE  CACM SEPT 69 'MOBILE PROGRAMMING SYSTEM'
*
*
     LOAD
*
* TOP LINKS
*
TTI1 EQU   '144   \TYPE IN ONE CHAR
TTO1 EQU   '145   \TYPE OUT ONE CHAR
TONC EQU   '151   \TYPE OUT STRING AND CR
TONN EQU   '152   \TYPE OUT STRING, NO CR
TOP  EQU   '156   \RETURN TO SYSTEM
READ EQU   '160   \READ FROM DISK
WRIT EQU   '161   \WRITE ON DISK
SRCH EQU   '162   \SEARCH ON DISK
ERRT EQU   '166   \ERROR RETURN
NAMG EQU   '167   \GET FILE NAME
*
*SPECIAL CHARACTERS
* [   BEGINNING OF MACRO DEF
* ]   END OF MACRO DEF
* @   PARAMETER FLAG IN MACRO NAME
* %   PARAMETER FLAG IN MACRO BODY
* &   REST OF LINE PARAMETER FLAG IN NAME
*
*
     ORG   '400
* SECTOR ZERO VARIABLES AND LINKS
ANL  OCT   212   NEW LINE
ALBR OCT   333   \LEFT BRACKET
ARBR OCT   335   \RIGHT BRACKET
APCT OCT   245   \% SIGN
AETC OCT   246   ETC SIGN
AATS OCT   300
RDL  DAC   RDX   READ A LINE
WTL  DAC   WTX   WRITE A LINE
GETX OCT   0     GET CHAR
PUTX OCT   0     PUT CHAR
NMAC OCT   0   NO OF MACROS
MMAX OCT   600   MAX NO OF MACROS
CMAX OCT   40000 MAX CHAR LENGTH OF MACRO BUFFER
LLIN OCT   0   LENGTH ILIN
MPTR BSZ   128   MACRO POINTERS
TBUF OCT   1000   CHAR LENTH BUFF LIN
NPAR OCT   0      NO PARAMETERS
PAR  BSZ   64   PARAMETER LIST
TSTK OCT   -1000   LENTGH OF STAK
MAT3 OCT   0   CHAR BUFFER
PUT1 OCT   0
GET1 OCT   0
ILNN DAC   ILIN
```

```
      ORG    '1000
*
*     INITIALIZE ;   OPEN FILES
*
INIT JST*   TONN    TYPE MESSAGE
     OCT    -7
     DAC    MES1
     JST*   NAMG    GET INPUT FILE NAME
     DAC    NAMI
     JST*   TONN
     OCT    -7
     DAC    MES2
     JST*   NAMG    GET OUTPUT FILE NAME
     DAC    NAMO
     JST*   SRCH    OPEN FILES
     OCT    1
     DAC    NAMI
     OCT    1
     JST*   SRCH
     OCT    2
     DAC    NAMO
     OCT    2
     CRA
     STA    NMAC    NO OF MACROS
     STA    MCCT    MACRO CHAR COUNT
     STA    SCCT    STACK CHAR CT
*    CONTROL ROUTINE
CTL  JST*   RDL   READ A LINE
     LDA*   ILNN
     ICL
     CAS    ALBR    LEFT BRACKET
     JMP    EXPN
     JMP    MDEF    MACRO DEF
     JMP    EXPN    EXPAND
* MACRO DEFINITION
MDEF JST    MDIN    INCREMENT POINTERS
     LDA    =1
     STA    NUM1
MDLP LDA    NUM1      GET CHAR
     ARS    1
     SSC
     JMP    RITE
     ADD    ILNN
     STA    GETX
     LDA*   GETX
     CAL
     JMP    NUM2
NUM1 OCT    0
RITE ADD    ILNN
     STA    GETX
     LDA*   GETX
     ICL
NUM2 STA    MDE1
     JST    MA1C    ADD CHAR
     LDA    MDE1
     CAS    ARBR    MACRO TERM CHAR
     JMP    NO24
```

```
      JMP    CTL    DON WITH MACRO
      CAS    ANL    LINE FEED
      SKP
      JMP    MDEL
NO24  IRS    NUM1
      JMP    MDLP
MDE1  OCT    0       SAVED CHAR.
* SETUP POINTERS FOR MACRO STORAGE
MDIN  DAC    **
      LDA    MCCT    MACRO CHAR CT.
      LDX    NMAC    NO OF MACROS
      STA    MPTR,1    MACRO POINTER
      IRS    NMAC
      LDA    NMAC
      CAS    MMAX    MAX NO. OF MACROS
      JMP    ER1    \TOO MANY MACROS
      NOP
      JMP*   MDIN
* END OF MACRO INPUT LINE
MDEL  JST*   RDL    READ NEW LINE
      CRA
      STA    NUM1      ZERO CHAR COUNT
      JMP    MDLP
* ADD A CHARACTER TO MACRO LIST
MA1C  DAC    **
      STA    PUT1
      LDA    MCCT
      ARS    1
      SRC
      JMP    ICH2
      ADD    NUM3
      STA    PUTX
      LDA*   PUTX
      ICR
      ADD    PUT1
      ICA
      STA*   PUTX
      JMP    NUM4
NUM3  DAC    MLST
MCCT  OCT    0
ICH2  ADD    NUM3
      STA    PUTX
      LDA*   PUTX
      CAR
      ADD    PUT1
      STA*   PUTX
NUM4  IRS    MCCT
      LDA    MCCT
      CAS    CMAX
      JMP    ER2    TOO MANY CHARS
      NOP           IN MLIST
      JMP*   MA1C
*     EXPAND A LINE OF CODE
EXPN  CRA
      STA    EXMC    MACRO COUNT
      LDA    NMAC
      SNZ
```

```
      JMP    EXP2    \NO MACROS
EXPL  JST    MATC    TRY TO MATCH LINE TO MACRO
EXMC  OCT    0
      SKP            NO MATCH
      JMP    EXP2+1   MATCH
      IRS    EXMC
      LDA    EXMC
      CAS    NMAC
      NOP
      JMP    EXP2    NO MATCH, OUTPUT LINE
      JMP    EXPL
EXP2  JST*   WTL    OUTPUT LINE
      JST    POP    POP STACK IF POSSIBLE
      JMP    CTL    NOT POSSIBLE, INPUT
      JMP    EXPN    POPPED,EXPAND NEW LINE
*POP STACK ONE LINE
POP   DAC    **
      LDA    SCCT
      SMI
      JMP*   POP    STAK EMPTY
      CRA
      STA    POP1    ZERO LINE LENGTH
      STA    LLIN
POPL  LDA    SCCT      GET ONE CHAR
      ARS    1
      SSC
      JMP    RIT2
      ADD    NUM5
      STA    GETX
      LDA*   GETX
      CAL
      JMP    NUM6
NUM5  DAC    STAK
SCCT  OCT    0         STAK CHAR CT
RIT2  ADD    NUM5
      STA    GETX
      LDA*   GETX
      ICL
NUM6  STA    POP2      SAVE IT
      LDA    POP1      PUT IT IN ILIN
      ARS    1
      SRC
      JMP    ICH3
      ADD    ILNN
      STA    PUTX
      LDA*   PUTX
      ICR
      ADD    POP2
      ICA
      STA*   PUTX
      JMP    NUM7
POP1  OCT    1
ICH3  ADD    ILNN
      STA    PUTX
      LDA*   PUTX
      CAR
      ADD    POP2
```

```
      STA*   PUTX
NUM7  IRS    POP1       INDEX
      IRS    SCCT
      NOP
      IRS    LLIN
      LDA    LLIN
      CAS    =80    CHECK IF ILIN TOO LONG
      JMP    ER4
      NOP
      LDA    POP2
      CAS    ANL    CHECK FOR END OF LINE
      SKP
      SKP
      JMP    POPL
      IRS    POP    SET UP SKIP
      JMP*   POP
POP2  OCT    0
* MATCH AND PROCESS MACRO LINE
MATC  DAC    **
      LDX*   MATC
      IRS    MATC
      LDA    MPTR,1
      STA    GTM1    MACRO ADDR
      CRA
      STA    MAT1
      STA    NPAR
      STA    MAT6
*TRY TO MATCH LINE
MATL  LDA    MAT1       GET LINE CHAR
      ARS    1
      SSC
      JMP    RIT3
      ADD    ILNN
      STA    GETX
      LDA*   GETX
      CAL
      JMP    NUM8
MAT1  OCT    0
RIT3  ADD    ILNN
      STA    GETX
      LDA*   GETX
      ICL
NUM8  IRS    MAT1
      STA    MAT3
      LDA    GTM1       GET MACRO CHAR
      ARS    1
      SSC
      JMP    NO21
      ADD    NUM3
      STA    GETX
      LDA*   GETX
      CAL
      JMP    NO22
NO21  ADD    NUM3
      STA    GETX
      LDA*   GETX
      ICL
```

```
NO22 IRS   GTM1
     CAS   AATS      @ SIGN
     JMP   NO20
     JMP   MAT4      PARAMETER DEF.
     CAS   AETC      & SIGN
     SKP
     JMP   MATR
NO20 CAS   MAT3      CHECK MATCH
     JMP*  MATC      NO MATCH
     SKP
     JMP*  MATC
     CAS   ANL        END OF LINE
     JMP   MATL       LOOP ON DEF LINE
     IRS   MATC
MAT9 JST   GTMC
     CAS   ARBR    TERM  CHAR
     JMP   NO23
     JMP   PUSH    DONE, PUSH BUFF IN STAK
     CAS   APCT
     SKP
     JMP   MAT8    PARAMETER USE
NO23 CAS   AETC
     SKP
     JMP   CONR    CONVERT REST BUFFER
*
MAT7 STA   PUT1       CHAR INTO BUFF
     LDA   MAT6
     ARS   1
     SRC
     JMP   ICH4
     ADD   NUM9
     STA   PUTX
     LDA*  PUTX
     ICR
     ADD   PUT1
     ICA
     STA*  PUTX
     JMP   NUM0
MAT6 OCT   0
NUM9 DAC   BUFF
ICH4 ADD   NUM9
     STA   PUTX
     LDA*  PUTX
     CAR
     ADD   PUT1
     STA*  PUTX
NUM0 IRS   MAT6
     LDA   MAT6
     CAS   TBUF
     JMP   ER6
     NOP
     JMP   MAT9
*
*    CONVERT PARAMETER
MAT8 JST   GTMC    TWO CHARS INTO NO.
     SUB   ='260
     MPY   =10
```

```
      IAB
      STA   MAT2
      JST   GTMC
      SUB   ='260
      ADD   MAT2
      SPL
      JMP   ER5   BAD NO., NOT 0-PAR
      CAS   NPAR
      JMP   ER5
      NOP
      STA   '0
      LDA   PAR,1
      JMP   MAT7
*
* PARAMETER DEFINITION
MAT4  LDX   NPAR
      LDA   MAT3
      CAS   ANL
      SKP
      JMP*  MATC   NO MATCH, NO PARAMETER
      STA   PAR,1   PARAM INTO PAR
      IRS   NPAR   NO. PARAM
      JMP   MATL
*
MAT2  OCT   0   SAVE MACRO CHARACTER
* & SIGN IN MACRO NAME LINE
MATR  CRA
      STA   MAR1   REST BUFFER ZEROED
      JST   GTMC   MOVE PAST CR IN MACRO
      LDA   MAT1
      STA   MAR2   INPUT LINE CONT
      LDA   MAT3   CURRENT INPUT CHAR
MARL  CAS   ANL
      SKP
      JMP   MAT9-1   DONE
      STA   PUT1
      LDA   MAR1
      ARS   1
      SRC
      JMP   ICH5
      ADD   NO10
      STA   PUTX
      LDA*  PUTX
      ICR
      ADD   PUT1
      ICA
      STA*  PUTX
      JMP   NO11
MAR1  OCT   0
NO10  DAC   RBUF
ICH5  ADD   NO10
      STA   PUTX
      LDA*  PUTX
      CAR
      ADD   PUT1
      STA*  PUTX
NO11  IRS   MAR1
```

```
      LDA    MAR2
      ARS    1
      SSC
      JMP    RIT4
      ADD    ILNN
      STA    GETX
      LDA*   GETX
      CAL
      JMP    NO12
MAR2  OCT    0
RIT4  ADD    ILNN
      STA    GETX
      LDA*   GETX
      ICL
NO12  IRS    MAR2
      JMP    MARL
*&- CONVERT REST BUFFER
CONR  LDA    MAT6
      STA    CNR1    BUFF COUNT
      CRA
      STA    CNR2    RBUF COUNT
      LDA    MAR1
      SNZ
      JMP    MAT9    RETURN ON EMPTY RBUF
      TCA
      STA    CNR3    INDEX
CNRL  LDA    CNR2       GET REST OF BUFFER CHAR
      ARS    1
      SSC
      JMP    RIT5
      ADD    NO10
      STA    GETX
      LDA*   GETX
      CAL
      JMP    NO13
CNR2  OCT    0
RIT5  ADD    NO10
      STA    GETX
      LDA*   GETX
      ICL
NO13  STA    PUT1       PUT INTO BUFF
      LDA    CNR1
      ARS    1
      SRC
      JMP    ICH6
      ADD    NUM9
      STA    PUTX
      LDA*   PUTX
      ICR
      ADD    PUT1
      ICA
      STA*   PUTX
      JMP    NO14
CNR1  OCT    0
ICH6  ADD    NUM9
      STA    PUTX
      LDA*   PUTX
```

```
      CAR
      ADD   PUT1
      STA*  PUTX
NO14  IRS   CNR1
      IRS   CNR2
      IRS   CNR3
      JMP   CNRL   LOOP
      LDA   CNR1
      STA   MAT6
      JMP   MAT9
CNR3  OCT   0
*
*PUSH BUFFER INTO STAK
PUSH  LDA   SCCT   STACK TOP COUNT
      SUB   MAT6   BUFFER COUNT
      STA   SCCT   NEW STAK TOP
      STA   PUS1
      CAS   TSTK
      NOP
      SKP
      JMP   ER4   STACK TOO BIG
      CRA
      STA   PUS2
PUSL  LDA   PUS2     GET BUFFER CHAR
      ARS   1
      SSC
      JMP   RIT6
      ADD   NUM9
      STA   GETX
      LDA*  GETX
      CAL
      JMP   NO15
RIT6  ADD   NUM9
      STA   GETX
      LDA*  GETX
      ICL
NO15  STA   PUT1     PUT INTO STAK
      LDA   PUS1
      ARS   1
      SRC
      JMP   ICH7
      ADD   NUM5
      STA   PUTX
      LDA*  PUTX
      ICR
      ADD   PUT1
      ICA
      STA*  PUTX
      JMP   NO16
PUS1  OCT   0
PUS2  OCT   0
ICH7  ADD   NUM5
      STA   PUTX
      LDA*  PUTX
      CAR
      ADD   PUT1
      STA*  PUTX
```

```
NO16 IRS   PUS1      INDEX
     IRS   PUS2
     LDA   PUS2
     CAS   MAT6
     NOP
     JMP*  MATC   DONE,RETURN
     JMP   PUSL   LOOP
*
* GET A CHAR FROM MACRO BUFFER
GTMC DAC   **
     LDA   GTM1
     ARS   1
     SSC
     JMP   RIT7
     ADD   NUM3
     STA   GETX
     LDA*  GETX
     CAL
     JMP   NO17
GTM1 OCT   0
RIT7 ADD   NUM3
     STA   GETX
     LDA*  GETX
     ICL
NO17 IRS   GTM1
     JMP*  GTMC
*
*
ER1  LDA   =401
     JMP*  ERRT
ER2  LDA   =402
     JMP*  ERRT
ER3  LDA   =403
     JMP*  ERRT
ER4  LDA   =404
     JMP*  ERRT
ER5  LDA   =405
     JMP*  ERRT
ER6  LDA   =406
     JMP*  ERRT
ER7  LDA   =407
     JMP*  ERRT
NAMI BSS   3         INPUT  FILE NAME
NAMO BSS   3         OUTPUT FILE NAME
MES1 BCI   7,SOURCE FILE
MES2 BCI   7,OUTPUT FILE
     FIN
* READ AN INPUT LINE
     ORG   '2000
RDX  DAC   **
     CRA
     STA   LLIN
RDX1 LDA   CWRD
     CAS   LBWD
     NOP
     JST   RBLK
     LDA   LLIN
```

```
        ARS    1
        STA    '0
        LDA*   CWRD
        STA    ILIN,1
        IRS    LLIN
        IRS    LLIN
        IRS    CWRD
        CAS    DZO
        SKP
        JMP    RUPP
        ICL
        CAS    ='212
        SKP
        JMP    RDX2
        LDA    ILIN,1
        CAL
        CAS    ='212
        JMP    RDX1
        JMP*   RDX
        JMP    RDX1
RDX2    LDA    LLIN
        SUB    =1
        STA    LLIN
        JMP*   RDX
RBLK    DAC    0
        LDA    BFA
        STA    CWRD
        STA    LBWD
        LDA    =-512
        STA    BCON
RONE    JST*   READ
        OCT    1
LBWD    DAC    **
        OCT    1
        LDA*   LBWD
        IRS    LBWD
        CAS    DZO
        SKP
        JMP*   RBLK
        IRS    BCON
        JMP    RONE
        JMP*   RBLK
DZO     OCT    122260
BCON    OCT    0
BFA     DAC    BADD
CWRD    OCT    100
*COME HERE IF DOLLAR SIGN -ZERO
RUPP    JST*   SRCH
        OCT    4
        OCT    0
        OCT    1
        JST*   WRIT
        OCT    2
        DAC    DZO
        OCT    1
        JST*   WRIT
        OCT    2
```

```
      DAC   ANL
      OCT   1
      JST*  SRCH
      OCT   4
      OCT   0
      OCT   2
      JMP*  TOP
*
* WRITE AN OUTPUT LINE
WTX   DAC   **
      LDA   LLIN    PUT 0 IN LAST CHAR
      STA   NO18
      ARS   1
      SRC
      JMP   ICH8
      ADD   ILNN
      STA   PUTX
      LDA*  PUTX
      CAL
      STA*  PUTX
      JMP   NO19
NO18  OCT   0
ICH8  ADD   ILNN
      STA   PUTX
      LDA*  PUTX
      CAR
      STA*  PUTX
NO19  LDA   NO18      COMPUTE WORD COUNT
      AOA
      ARS   1
      STA   WT1
      JST*  WRIT    WRITE LINE
      OCT   2
      DAC   ILIN
WT1   OCT   0
      JMP*  WTX
*
ILIN  BSS   50    INPUT LINE
RBUF  BSZ   50    REST BUFFER
      FIN
      ORG   '3000
MLST  EQU   '20000    MACRO BUFFER TAKES ALL OF UPPER CORE-
      BSZ   '377
STAK  OCT   0
      NOP
BUFF  BSS   '400
BADD  BSS   512
      END
```