1. Report No.	2. Government Accession No.	3. Re
TX-97/2926-1F		

4. Title and Subtitle	5. Report Date
IMPLEMENTATION OF AN AUTOMATED CRACK SEALER	November 1996
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Carl Haas, Al Traver, Gerald Easter, Richard Greer, Young-Suk Kim, and Angela Reagan	Research Report 2926-1F

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Center for Transportation Research The University of Texas at Austin 3208 Red River, Suite 200 Austin, Texas 78705-2650	11. Contract or Grant No. Research Project 7-2926
	13. Type of Report and Period Covered Final
12. Sponsoring Agency Name and Address Texas Department of Transportation Research and Technology Transfer Office P. O. Box 5080 Austin, Texas 78763-5080	14. Sponsoring Agency Code

15. Supplementary Notes

Project conducted in cooperation with the Texas Department of Transportation.
Research project title: "Implementation of an Automated Crack Sealing System"

16. Abstract

Automated pavement crack sealing technology has been developed in the TxDOT- and FHWA-funded project described in this report. Performance of the prototype has been demonstrated, and the economics are attractive. A follow-up implementation and testing project involving field trials across the state has been approved.

Crack sealing is a widespread, dangerous, costly, and labor-intensive operation. Labor turnover and training are increasing problems related to crack sealing crews, and as traffic volumes increase, crack sealing operations become increasingly disruptive to the general public. Automating crack sealing can reduce labor and road user costs, improve work quality, and decrease worker exposure to roadway hazards. Prior to the project described in this report, three interim studies at The University of Texas were completed. A study of methods, practices, and productivity for crack sealing in Texas compiled detailed critical data. A study of sensor fusion methods led to the current tele-operated control strategy. And a study of maintenance automation needs based on surveys of TxDOT employees ranked automating crack sealing as one of the highest needs in the state.

Approximately $200,000,000 is spent annually on crack sealing in North America. About 25% is privately contracted. In Texas, this rises to about 50% of the approximately $7,000,000 spent annually. Labor costs average between 50% and 60% of total crack sealing costs.

Parts for the system developed at UT are mostly off-the-shelf and total approximately $70,000. Additional costs for assembly, marketing and profit will require a sale price up to $125,000. Since approximately 3 laborers will be eliminated, the payback should be 1 to 2 years.

According to the most recent economic analysis, if the automated crack sealing systems were implemented statewide, direct savings could amount to $2.43 million for TxDOT (at 4% MARR) and $2.64 million for private contractors (at 20% MARR) over a 6-year planing horizon. Using the widely accepted QUEWZ-E model, we estimated the user-cost savings to be $11.0 million for the 5,196 km of interstate highways in Texas. Total user-cost savings would be much higher, since the savings on urban freeways and streets, farm-to-market roads, and secondary roads are not included in this $11 million estimate. Over a 30-year planning horizon and from a national perspective, the net present worth of automated crack sealing could be in the hundreds of millions of dollars.

17. Key Words	18. Distribution Statement
Automated crack sealing systems, road maintenance, sealants, Automated Road Maintenance Machine (ARMM)	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	379	

IMPLEMENTATION OF AN AUTOMATED CRACK SEALER

Carl Haas
Al Traver
Gerald Easter
Richard Greer
Young-Suk Kim
Angela Reagan

Research Report 2926-1F

Research Project 7-2926
Implementation of an Automated Crack Sealing System

conducted for the

Texas Department of Transportation

by the

CENTER FOR TRANSPORTATION RESEARCH
Bureau of Engineering Research
THE UNIVERSITY OF TEXAS AT AUSTIN

November 1996

IMPLEMENTATION RECOMMENDATIONS

A project is proposed for implementation, one that would involve field trials at five or more locations around the state, evaluations by maintenance personnel, assembling of focus groups for market analysis, pursuit of news media coverage, negotiation with key vendors concerning establishment of a commercialization entity, analysis of productivity, publications, and production of user documentation.

Evaluation would involve several steps and procedures, including field trials, evaluations by maintenance personnel, market focus group analysis, key vendor input, and detailed productivity analysis.

Field trials would be conducted in five or more districts. Objectives of the field trials would include collection of productivity data, demonstration of the technology, and acquisition of feedback from maintenance personnel. Secondary objectives would include proof testing the equipment under real working conditions, acquiring video footage, and gaining field experience. Feedback would be obtained from informal discussions, formal interviews, and focus group discussions. The schedule of each field trial would be:

- 1 day for transport and setup
- 2 days for field demonstrations and productivity analysis
- 1 day for focus group discussions and review
- 1 day return transport and demobilization

Time between field trials would be dedicated to productivity data analysis, survey data analysis, documentation, publication, publicizing, local demonstrations, and equipment repair and modification. Productivity analysis would follow the form of previous studies by the UT team that have been widely reviewed and accepted. Economic analyses would be conducted from several perspectives, including the district, the state, the country, the system manufacturer, and the contractor. Market analysis would focus on Texas but would project results nationwide based on national statistical databases.

Vendors would be convened at one or more demonstrations to discuss and negotiate commercialization. Technology will be transferred via demonstrations, distribution of documentation, and personal explanations.

Automated pavement crack sealing is a leading edge technology. Broad applications of the technology are anticipated, including automated routing, joint sealing, message painting, pothole filling, and marker placement.

Prepared in cooperation with the Texas Department of Transportation.

DISCLAIMERS

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the Texas Department of Transportation. This report does not constitute a standard, specification, or regulation.

There was no invention or discovery conceived or first actually reduced to practice in the course of or under this contract, including any art, method, process, machine, manufacture, design or composition of matter, or any new and useful improvement thereof, or any variety of plant, which is or may be patentable under the patent laws of the United States of America or any foreign country.

NOT INTENDED FOR CONSTRUCTION,
BIDDING, OR PERMIT PURPOSES

Carl Haas, P.E.
Research Supervisor

TABLE OF CONTENTS

SUMMARY

Automated pavement crack sealing technology has been developed in the TxDOT- and FHWA-funded project described in this report. Performance of the prototype has been demonstrated, and the economics are attractive. A follow-up implementation and testing project involving field trials across the state has been approved.

Crack sealing is a widespread, dangerous, costly, and labor-intensive operation. Labor turnover and training are increasing problems related to crack sealing crews, and as traffic volumes increase, crack sealing operations become increasingly disruptive to the general public. Automating crack sealing can reduce labor and road user costs, improve work quality, and decrease worker exposure to roadway hazards. Prior to the project described in this report, three interim studies at The University of Texas were completed. A study of methods, practices, and productivity for crack sealing in Texas compiled detailed critical data. A study of sensor fusion methods led to the current tele-operated control strategy. And a study of maintenance automation needs based on surveys of TxDOT employees ranked automating crack sealing as one of the highest needs in the state.

Approximately $200,000,000 is spent annually on crack sealing in North America. About 25% is privately contracted. In Texas, this rises to about 50% of the approximately $7,000,000 spent annually. Labor costs average between 50% and 60% of total crack sealing costs.

Parts for the system developed at UT are mostly off-the-shelf and total approximately $70,000. Additional costs for assembly, marketing and profit will require a sale price up to $125,000. Since approximately 3 laborers will be eliminated, the payback should be 1 to 2 years.

According to the most recent economic analysis, if the automated crack sealing systems were implemented statewide, direct savings could amount to $2.43 million for TxDOT (at 4% MARR) and $2.64 million for private contractors (at 20% MARR) over a 6-year planing horizon. Using the widely accepted QUEWZ-E model, we estimated the user-cost savings to be $11.0 million for the 5,196 km of interstate highways in Texas. Total user-cost savings would be much higher, since the savings on urban freeways and streets, farm-to-market roads, and secondary roads are not included in this $11 million estimate. Over a 30-year planning horizon and from a national perspective, the net present worth of automated crack sealing could be in the hundreds of millions of dollars.

1.0 Introduction to the Crack Maintenance Process

Pavements represent the largest portion of the hundreds of billions of dollars invested in our transport infrastructure. But pavements deteriorate with time, traffic and climate Consequently, preservation of the investment through timely maintenance and rehabilitation is essential (Haas 1992).

Cracking in asphalt concrete (AC) paved surfaces has been a problem that state and local agencies have had to deal with for many years. It is considered one of the two primary reasons for deterioration in AC pavements, the other being rutting. Neglect of cracking often leads to more severe cracks and/or pot holing which in turn leads to reduced pavement life. Maintenance of cracks is done in order to extend the pavements service life a few more years and is considered an economic way to do so without going to the expense of such treatments as resurfacing.

Crack sealing is usually conducted by a five or six person road crew (AASHTO 87). The equipment used includes pylons, a heavy truck, a material tank, a heated air torch, a sealing wand, and a routing machine if the cracks are being routed prior to being filled. One or two crew members may be necessary to direct traffic and place pylons. One operator walking behind the truck blows out the cracks with the torch and others in turn fill them in with sealing material. A sand covering may be applied to permit immediate use by traffic. The procedure varies significantly from region to region.

The sealing and filling of cracks are tedious, labor-intensive operations. A typical maintenance crew can seal between one and two miles per day. The associated costs are approximately $1800 per mile with 66% attributed to labor, 22% to equipment, and 12% to materials. Furthermore, the procedure is not standardized in practice and there is a large distribution in the quality of the resultant seal. In addition, the crack sealing team is exposed to a great deal of danger from moving traffic in adjacent lanes (SHRP-H-659).

Crack and joint sealing is done for several reasons: it prevents water infiltration, it keeps out foreign matter, and it protects joint filling material. The material of choice to fill cracks appears to be rubberized asphalt compounds because of their favorable characteristics. They have less tendency to become brittle in cold weather and to soften and track in hot weather. Requirements for crack and joint sealants are covered in the current specifications of the American Society for Testing and Materials (ASTM), the American Association of State Highways and Transportation Officials (AASHTO), and federal agencies. ASTM specifications D3405-78 for hot applied products as well as AASHTO M173 are the respective sections (Asphalt Institute, 1983).

1.1 General Cracking Types

There are five main types of cracking that occur in pavement surfaces.

Longitudinal - These cracks follow a course approximately parallel to the centerline of the pavement

Transverse - These are cracks that follow a course approximately perpendicular to the centerline of the pavement

Diagonal - These are cracks that are diagonal to the centerline of the pavement

Alligator - These are interconnected cracks forming a series of small blocks resembling an alligator's skin

Restraint - These are cracks which develop near the outside edges of the pavement and progress in an irregular path towards the longitudinal joint

(Asphalt Institute, 1983)

1.2 Important Considerations When Planning the Crack Maintenance Project

There are several things that should be considered when planning crack filling or sealing maintenance operations. They include:

1. Climate conditions at the time of maintenance operations and in general
2. Highway Classification
3. Level of traffic and percent of trucks
4. Amount of cracking
5. Crack characteristics (type and width)
6. Type of filling/sealing material
7. Material placement configurations
8. Equipment and procedures
9. Safety

When planning a crack maintenance project, the selection of an appropriate sealing/filling material, and determination of the equipment and procedures based on existing and future road conditions are key decisions. Climate conditions are also important because moisture or cold temperatures will effect material adhesion properties. Additionally, adverse weather conditions present scheduling problems, and may indicate the use of specialized equipment such as a heat lance to eliminate any moisture in the crack. General climate conditions should also be taken into consideration when making a decision on which type of material to use. Hot climates require the use of materials that will not soften a great deal under high temperatures. Conversely, cold climates will require materials that retain good flexibility during freezing temperatures.
(Smith, K.L. ; Romine, R.A. ; Peshkin, D.G.; 1992)

There are also considerations from a highway classification and travel level standpoint. Highway geometrics and traffic levels may be such that levels of safety are not sufficient. Safety may be greatly increased by applying longer lasting treatments which in turn means fewer applications and less time maintenance crews are required to spend on the roadway.

Crack characteristics such as crack sizes, moving or non-moving, etc. will also have an impact on which materials, equipment and procedures to employ. Amount of cracking

and width of typical cracking will be required information in order to estimate the amount of material that will be required for a given project.

1.3 Selecting a Sealant or Filler Material

In order to select the proper sealing or filling material, one must determine the key properties that the material must possess in order to perform effectively in the given conditions. The following is a list of some of the more desirable properties.

- Reasonable material preparation time
- Good material workability (ease of placement)
- Short curing time
- Adhesives (materials ability to remain bonded to the sidewalls of crack)
- Cohesiveness (materials ability to resist rupture)
- Flexibility (materials ability to stretch as pavement movement occurs)
- Elasticity (materials ability to recover from deformation and resist foreign material intrusion)
- Resistance to softening to the point that flow occurs
- Resistance to weathering and aging
- Resistance to abrasion

Table 1.0 shows the types of materials that possess most of the properties listed above. It can be seen that materials containing rubber make them applicable to sealing cracks, whereas emulsified products tend to contain good preparation and placement properties that make them applicable to filling cracks. Table 1.0 can be used as a guideline to determine which type or types of material should be considered for any type crack condition.

Table 1.0 - Sealant material types and individual characteristics

Property		Material	Type					
	Asphalt Emulsion	Polymer-Modified Emulsion	Asphalt Cement	Fiberized Cement	Asphalt Rubber	Rubberized Asphalt	Low-Modulus Rubberized Asphalt	Self- leveling Silicone
Short Preparation	X	X						XX
Easily placed	X	X	XX	XX	XX	XX	XX	
Short Curing Time			XX	XX	XX	XX	XX	X
Adhesiveness	XX	X	XX	X	X	X	X	X
Cohesiveness		X			X	X	XX	X
Flexibility					X	X	XX	XX
Elasticity		X			X	X	X	XX
Resistance to Softening and Flow		X			X	X	XX	XX
Resistance to Aging and Weathering							X	XX
Resistance to Abrasion					X	XX	X	
Recommended Application	filling	Filling (maybe sealing)	Filling	Filling	Sealing (maybe filling)	Sealing	Sealing	Sealing

X Good

XX Very Good (Smith, K.L.; Romine, R.A. ; Peshkin, D.G. ; 1992)

1.4 Crack Preparation (Drying and Cleaning)

The preparation of the crack may very well be the most important part of a sealing/filling maintenance project. This is because a large percentage of crack maintenance failures are due to adhesion failures resulting from dirt, debris, or moisture that was not eliminated from the reservoir at the time of treatment.

High pressure compressed air is one of four methods used to prepare cracks for sealing/filling . The other three are hot compressed air, sandblasting, and wire brushing. In this report we will only discuss compressed air but other systems are discussed in (Malek, 1993). High pressure compressed air is fairly effective at removing dirt, loose debris and some asphalt concrete (AC) fragments. It is not considered nearly as effective at removing laitance (fine particles) or moisture from the crack reservoir. The minimum recommended pressure that a compressed air unit should provide is 100 $lb./in^2$ and a flow rate of 150 ft^3 /min. In addition, it is recommended that compressed air units have an oil- and moisture-filtering system. The introduction of oil and moisture can severely inhibit bonding of the sealant material to the sidewalls.

The crack should be dry and thoroughly cleaned prior to the sealing/filling of any cracks. Because high pressure air blasting provides no heat and little drying capabilities, it should only be conducted when the pavement and cracks are completely dry and there is no threat of rain. Furthermore, ambient temperatures should be above 40 degrees Fahrenheit and rising. Any moisture will prevent the sealant filler material from bonding properly to the sidewalls of the crack. At least two passes should be made along each crack segment in order to dislodge dirt and other debris from the crack. The final pass is done to remove all remaining particles from the crack reservoir and surrounding pavement surface.

1.5 Preparation and Application of Material

Hot applied materials are heated and mixed within a mobile container that utilizes indirect-heat, agitator-type kettle or melter. The machine runs off of either diesel or propane fuel. The material is heated using a double-jacketed boiler system which consists of a space between the inner and outer shells of the melting vat. The space between the inner and outer shell is filled with oil that acts as a transfer medium. This indirect method of heating is considered to be safer and provides a more controlled and uniform manner for heating the material. Agitation devices are also standard so as to help provide uniform heating of the material.

ASTM D3405-78 lists the following precautions concerning the heating and dispensing of hot poured products. Care should be exercised to use equipment that is suitable for the purpose and approved by the manufacturer of the material. The material should be heated in a double boiler, have positive temperature control, mechanical agitation, and recirculating pumps. Direct heating must not be used so as to guard against fire. (American Society for Testing and Materials; 1980)

Material preparation of hot-pour material consists of loading/filling the kettle, heating it to the appropriate temperature, and continuous mixing to ensure proper consistency and

uniform heating. Manufacturers specifications and recommendations as to how material should be prepared and placed should be followed.

1.6 Material Finishing/Shaping and Blotting

Material finishing is usually done by using a "V" or "U" shaped squeegee that is attached to the end of a pole or broom handle. The squeegee is intended to help force the material down into the crack and eliminate bumps on the pavement surface after the material cures. The squeegee operation also aids in creating a bond between the material and pavement surface. The following finishing recommendations should be considered:

- Operate the squeegee immediately behind the crack filling procedure
- All attempts to center the band-aid configuration over the crack reservoir should be made
- Keep the squeegee from developing a buildup of material by regularly scraping it off. It may be necessary to periodically remove build-up using a propane torch.

(Smith, K.L.; Romine, R.A.; Peshkin, D.G.; 1992)

1.7 Equipment Cleanup

Most asphalt kettles must have their application system lines cleansed of hot-pour materials at the end of each work day. Reheatable materials can have their applicator lines thoroughly cleaned using reverse flow methods followed by solvent flushing procedures. When using non-reheatable materials, every effort should be made to use as much of the material as possible so as to minimize waste. Leftover material will need to be put into sealed containers for disposal.

When solvents are used to flush out the application systems, care must be taken to ensure that the flushing agents do not contaminate the sealer/filler materials. Manufacturers

recommended instructions of both heating kettles and sealant/ filler materials should always be followed. Chapter 13.0 includes a basic operating instruction guide for the Crafco Super Shot 60 melter used in this project. The Crafco SuperShot 60 sealant wand does not require cleaning.

2.0 Evolution of an Automated Road Maintenance Machine

This Chapter discusses the background of the development of the ARMM . Development of automated construction technology such as the crack sealer involves several issues including the design cycle, economic feasibility analysis, financing, and implementation. Each of these issues is dealt with in turn in the following sections, and in more detail in subsequent chapters. Demographic, regulatory and competitive forces impose increasing pressures on the construction industry. Automated pavement crack sealing is a typical example of the technology developments that are emerging in response to these industry pressures. Automated crack sealing is of considerable interest for several reasons. First, crack sealing is a widespread and common operation in the United States. If even modest cost savings could be achieved in crack sealing operations, the total savings would be substantial. Second, automated crack sealing may achieve improved quality over existing field operations, so that the need for maintenance operations may be reduced over time. Finally, automated crack sealing would reduce the exposure of maintenance workers to injury and accident.

Crack sealing is a wide spread, dangerous, costly, and labor intensive operation. Labor turnover and training are increasing problems related to crack sealing crews, and as traffic volumes increase, crack sealing operations become increasingly disruptive to the general public. Automating crack sealing can reduce labor and road user costs, improve work quality, and decrease worker exposure to roadway hazards. In a series of research projects at Carnegie Mellon University (CMU) and then the University of Texas, an automated method for sealing pavement cracks has been developed. This report describes the most recent prototype.

Automated crack sealing is a technically challenging operation for several reasons. Since pavement cracks are irregular in nature and extent, simple numerical controlled devices cannot be used directly. Some means of perceiving crack location and controlling maintenance equipment is required. Moreover, crack sealing is undertaken under field

conditions which may involve extremes of temperature, precipitation and debris. Maintenance of the equipment used in roadway work varies considerably in quality, therefore robust and reliable equipment is required. Finally, introduction of automated equipment in this domain must be justified by cost savings and quality improvements, so inexpensive and effective equipment is imperative (Haas, March 1991).

2.1 The Technology Development Design Cycle

Technology development can be modeled as a design cycle. Prior to any action taking place, a need must emerge, or needs must be identified in a systematic analysis and prioritized. Next, the design development problem posed by the need must be rigorously defined, and the criteria by which solutions to the problem will be evaluated must be identified and ranked. It is generally advantageous to develop alternative solutions which can be compared objectively in subsequent feasibility analyses. Optimization of selected solutions requires some combination of modeling, experimentation, simulation, analysis, and further acquisition of information concerning the operating environment. Each of these steps involves feedback between each other, and the design cycle is iterative in nature. Solutions normally begin as "back of the envelope" conceptual prototypes and then progress through laboratory, field, and commercial prototype stages. Development of the crack sealer has followed this design cycle.

2.2 Design Issues

The primary objective of the design is to reduce the overall cost of crack sealing. Improving the quality of the work and worker safety are also objectives. The final design must be evaluated based on these objectives.

Functional Approach

There are a variety of functional approaches that can be considered for crack sealing. Alternatives may have varying degrees of manual supervision. For example, multiple

nozzles can be used for blowing and sealing rather than individual effectors. Arrays of nozzle effectors would be costly, and the necessary switching patterns would be extremely difficult to implement. In particular, the viscosity of sealing material makes short frequent bursts difficult. In contrast, use of individual effectors requires that each effector will somehow be drawn precisely along the length of the cracks to be filled. A multiple degree of freedom manipulator is necessary to control the path of the individual effectors. Options include: (a) having the truck driver tele-operate the manipulator while the truck is stopped, (b) having the truck driver check or add crack location information to a partially automated system in which the manipulator would be controlled automatically, and (c) having the operator simply monitor with interrupt control a largely autonomous crack sealing operation. The design described in this report follows option (b).

A simple solution for manipulation is an xy-table. Such a device works much like a large scale plotter with a gantry and mounted cart implementing x and y motions respectively. Control is much simpler than a robot arm. With a framework constructed of tubular steel, a table is more impact resistant and stable than an arm in that reactions are always within the framework and distances to points of support are minimized. The effectors are easily kept perpendicular to the pavement surface. All the design constraints can conceivably be met. As for the design criteria, the device may be simply constructed and therefore maintainable, but its transportability is an issue of concern. The design measures well with respect to the remaining design criteria.

Surface Perception and Modeling

Identifying cracks in the road automatically is not an easy problem (Haas 84, Fukuhara 90, Ballard 82). Mapping the layout of the cracks in detail and selecting those to be filled increases the difficulty. In the case of routed cracks, the problem is simplified by distinct visual patterns of debris and by consistent groove dimensions.

Surface data can be acquired in raster scan or arbitrary patterns. It can also be acquired with non-contact or with contact sensors. Contact sensors such as a pin cushion type

roller or a linear array of brush sensors are not feasible because of their costs and their probable insensitivity to narrow cracks. Non-contact sensors include vision, range, and forward looking infrared devices. Video cameras can acquire a raster pattern of digitized surface gray level values very quickly. Range sensors such as ultra-sonic and infrared laser devices can be drawn over the surface in any acquisition pattern by the effector equipment. In practice, all these sensors experience noise because of the varied topological and color conditions of the pavement surface, and because of environmental factors such as wind and sunlight. Moreover, a single sensor perception can be fooled. Analysis of a video image alone shows that it is almost impossible to automatically detect the difference between a routed crack, a filled crack, and a strip of dark oil. With the corroboration of range information from multiple sources in a common surface representation, the accuracy of crack perception can be increased. The facilities required to do this exist in a general pavement surfaces model which includes facilities for data fusing and structuring (Haas, 1990). Human direction has proven to be more efficient however.

From the current prototype's machine vision assisted crack identification procedure, ordered lists of cartesian coordinates describing points along the cracks are derived and then interpreted to yield a graph representation of the crack network in the area to be worked. The most efficient traversal through this network can then be derived by a variety of means.

Crack Sealing Control

Control of the crack sealing process is exercised at several levels. Control of a system that moves continuously down the road at a constant speed introduces complexities in terms of perception, planning, and especially manipulator control that are simplified by operating the system in a stop/start manner. Assuming a start/stop strategy, the highest level of system control implements the following steps repetitively as the system moves down the road:

1. Acquire sensor data and develop a representation of the pavement surface
2. Map the cracks to filled
3. Develop a work plan
4. Execute the blowing and sealing operations
5. Repeat the above four steps

In the process of developing the surface representation for the current area to be worked, the system may compile a list of commands to the equipment to enact scanning patterns and acquire data. Once the surface representation is complete, the system must also choose the order and direction of the cracks to be traversed. This plan must then be compiled into a list of commands to the manipulator and to the actuators such as the open and close valve on the sealing wand.

2.3 Needs Analysis

Pavement crack sealing is a costly operation, and it imposes significant costs on road users due to traffic interference. Initially, the motivation to automate crack sealing arose from this observation. Removing laborers from the process would reduce their exposure and reduce operating costs, as well as open up the possibility of increasing speed thus decreasing road user costs. This observation was further substantiated with statistical data (FHWA 1990), expert opinions, and literature from the Ontario Ministry of Transportation. For the first phase of development, this analysis of need was considered sufficient.

For the second phase of development, a survey of all fifty states and ten provinces provided additional information about sealing practices and the potential market for automated crack sealing (McNeil 1992). Wide variations in methods, equipment, materials, and crew composition were observed which prompted reconsideration of the design objectives and performance criteria.

For the third phase of development, discussed in this report, needs analysis focused on Texas. A productivity study of conventional operations detailed operating conditions and performance requirements for an automated system and revealed that district engineers valued the safety benefits of an automated system most highly (Malek 1992). Using the system to measure and record work units completed has contractual advantages as well. Borrowing from construction needs analysis methods (Tucker 1990), a subsequent systematic evaluation of all road maintenance activities in Texas resulted in a prioritization matrix that uses cost impact and concern axes to characterize the need to automate (Osmani 1994). The concern rating formula incorporates productivity, quality, safety, socio-political, technology feasibility, ergonomic, and user cost factors. Crack sealing was identified as a automation need in Texas.

2.4 Conceptual Prototype

Originally, the automated crack sealing system was envisioned as an equipment train including an equipment trailer, a manipulator, and a large van containing computer and power equipment. Manipulator options were considered, and an xy-table configuration was selected because of its ease of control and robust physical characteristics. Machine vision was proposed for automated crack mapping with system autonomy as the end goal.

2.5 Laboratory Prototype

Design objectives for the laboratory prototype focused on "proof of concept". Demonstrating a system that could ultimately improve safety and productivity by working autonomously was the primary objective. Low cost was a secondary objective at this stage. An xy-manipulator was assembled in the lab, and pavement test sections were fabricated. A video camera mounted above the workspace was used to acquire images which were digitized and then combined with laser range data of surface contours using a specially developed multi-layer quadtree model and image analysis algorithms implemented on a UNIX type workstation. Combination of sensor data, often termed

"fusion" in robotics literature, is required, because neither source of data is sufficient or fast enough in itself for completely accurate mapping. A working system was demonstrated one year after initiation of the development project (Haas 1990).

Problems encountered in the first phase of development included unacceptably slow operation. While accurate and essentially autonomous, the system required 20 to 30 minutes to complete the scanning, mapping, and work process cycle. Calibration and alignment between the sensing and manipulator subsystems proved difficult because of the hasty assembly of the prototype. Despite these problems, a consensus existed that the approach was feasible and that the design cycle should begin anew. Solutions were proposed, and a second phase of development was funded.

2.6 Field Prototypes

First Field Prototype

Design objectives for the first field prototype were to consolidate control and data processing on a single Intel 386 type PC, and to demonstrate operation on unrouted cracks, in a parking lot. The system was still connected by an umbilical cord to the lab. A more robust xy-manipulator was fabricated and a revised control loop was implemented. Though demonstrated successfully a year after commencing the second phase, the system was retarded by slow range scanning speed, and development along this track by the associated personnel ceased temporarily (Haas 1992).

CalDavis Field Prototype

In a subsequent and related development effort, the University of California at Davis developed a field prototype of an automated crack sealing machine. Level of effort in terms of funding was about 8 times the sum of previous efforts. The final prototype, which has multiple manipulator arms, was demonstrated preparing and sealing longitudinal joints at 2 miles/hour. If produced in volume, it is estimated that the machine could be sold for US $550,000 (Velinsky 1993). Market analysis indicates that few private

contractors or government departments will be willing or capable of paying this amount (Velinsky 1993).

Development Studies

Concurrently, three related development efforts were conducted. They included a study of methods, practices, and productivity for crack sealing in Texas (Malek 1992), a study of sensor fusion methods (Gharpuray 1993), and a study of maintenance automation needs (Osmani 1994).

Second Field Prototype

Based on knowledge and experience gained from preceding development efforts, design objectives were modified. A commercial prototype must operate at manual crew speed or faster (approximately 10-30 seconds per work cycle),and the unit cost must be less than $100,000. A tele-operation approach was proposed and accepted. This next iteration of the design cycle was funded by a consortium including the Federal Highway Administration's Office of Technology Applications, the TxDOT Maintenance and Construction Division, and Crafco Ltd. (a crack sealing equipment manufacturer). It is the prototype described in this report.

A remote, graphically controlled system, using an xy-table manipulator, was designed. Manual graphical input is used in order to negate the need for range sensing, and all the software is integrated in one application program. These two design changes allow the system to meet work cycle time constraints. In addition, machine vision is used to correct for lack of operator hand/eye coordination, and automated path planning is used to minimize crack network transversal time resulting in substantial cycle time savings. Field trials conducted in June 1995 indicated that 10 to 30 second work cycles are achievable. It is estimated that the system could be manufactured from OEM equipment for as little as $70,000. Field trials in July 1996 demonstrated a 30-45 second work cycle on a full scale prototype. With increased motor speeds and more ergonomic mounting of the central mechanisms, 10 to 30 second work cycles will be achieved by December 1996.

2.7 Economic Analysis

With a conceptual design in mind, it was originally estimated that the system would have a purchase cost of $100,000, a useful life of 5 years, $10,000 annual maintenance costs, and $100,000 in annual cost savings by eliminating two laborers. A very high rate of return (ROR) results. From a basic technology development perspective, this expected ROR, and the annual national expenditures were sufficient justification to fund development of the laboratory prototype. Later economic analyses took public agency, public economy (McNeil 1992), and private organization(Velinsky 1993) perspectives.

The most recent economic analysis uses more detailed information to augment the analysis from each of the preceding perspectives (Osmani 1994). The results were used to justify development funding from national, state, and private organizations. These organizations essentially form a consortium with the university based developers, and economic justification was necessary from each of their perspectives for an agreement to proceed.

According to the most recent economic analysis, if the automated crack sealing systems were implemented statewide, the direct savings are estimated to be $2.43 million for the Texas Department of Transportation (TxDOT) at a 4% MARR and $2.64 million for the private contractors at a 20% MARR over a six year planning horizon (Osmani 1994). The user-cost savings are estimated using the widely accepted QUEWZ-E model (Memmot 1982, Seshadri 1993) to be $11.0 million for the 3229 miles of the interstate highways in Texas. Total user-costs savings would be much higher, since the savings on urban freeways and streets, farm-to-market-roads, and secondary roads are not included in this $11 million estimate. Over a 30 year planning horizon and from a national perspective, the net present worth of automated crack sealing could be in the hundreds of millions of dollars. This analysis is elaborated upon in chapter 6.0.

2.8 Market Analysis

Only 16% of private contractors who perform crack sealing earn annual revenues of over $1,000,000, and their revenues cannot be solely attributed to crack sealing. A small percent will be able to invest in automated systems initially, however the associated benefits should increase their competitive advantage. As in other industries, automation may force consolidations. Government can accelerate this process by letting longer term, larger contracts.

Government agencies may also purchase automated crack sealers. Texas has 25 highway districts which are authorized to purchase equipment such as an automated crack sealer. Local municipalities and contractors augment the potential market. Impediments exist however. They include the occasional practice of performing crack sealing with crews otherwise left idle when larger construction projects are threatened by weather conditions, or simple reluctance expressed by focus groups to spend large amounts of money because of perceived risk (Velinsky 1993). Increasingly though, agencies are becoming more sensitive to safety given the related litigious environment over the last decade, and they are becoming sensitive to the road user costs imposed by lane closures, as drivers become more vocal. Automated crack sealing will address these concerns as well as reduce operating costs.

2.9 Implementation of the Automated Road Maintenance Machine

Implementation of automated maintenance equipment into the Texas Department of Transportation (TxDOT) takes years of preparation and development from the conceptual idea to the eventual final design product. A strategic plan is needed and should be thoroughly developed from past experiences to avoid repeating mistakes and to streamline the timeline for the equipment to be procured within TxDOT.

The automated pavement crack sealer research and development project described here was funded by TxDOT and the Federal Highway Administration (FHWA) in July 1994 and contracted to The University of Texas under the guidance of Dr. Carl Haas. Additional support was provided by Crafco, a private company in Chandler, AZ, who donated use of a propane melter to the research. The requirements for the funding was to demonstrate within a year the feasibility of automated pavement crack sealing and to perform an overall detailed economic analysis. During the process, the research team performed software development, hardware interfacing, design work, and procurement of equipment. During the first phase of the project, which lasted until the end of June 1995, several demonstrations and presentations were provided in educating TxDOT personnel, university students, and interested parties about the automated pavement crack sealer. The prototype in Figure 2.1 was presented on June 16, approximately one year after commencing the project, to FHWA, TxDOT, Crafco, and other guests. The supporting organizations were in agreement that the technology was promising and continued financial support for the second phase of the project.

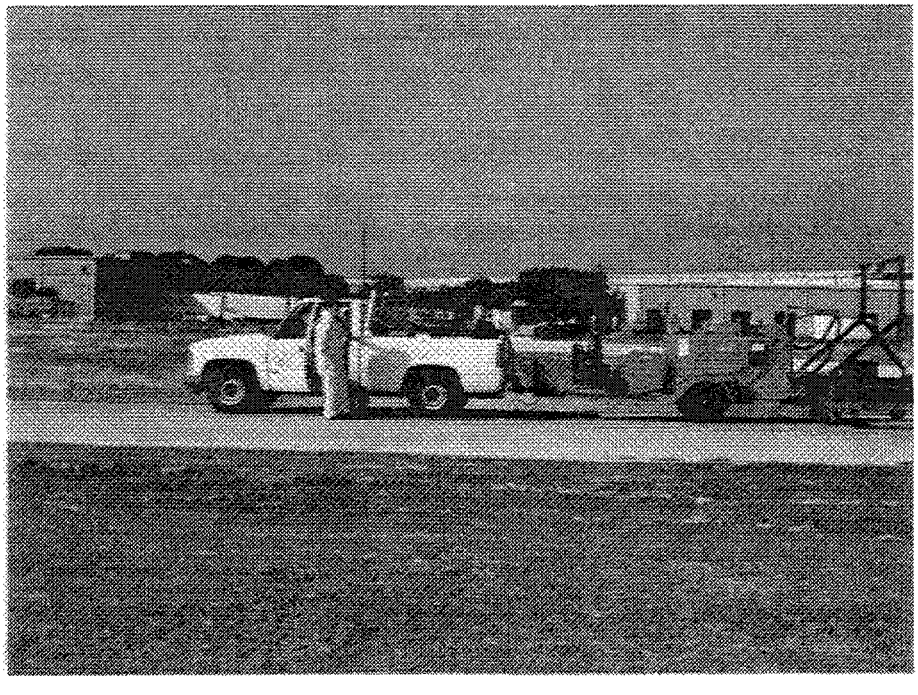


Figure 2.1 The Automated Crack Sealer Prototype presented June 16, 1995

The second phase was funded until August 1996. During the second phase, designs and equipment were contracted and ordered to improve the existing prototype. A new xy-table was procured and a new control system amplifier with controllers was ordered for an additional degree of freedom on the xy-table.

On July 2, 1996 a demonstration of a fully functional commercial prototype of an automated crack sealer took place for all interested parties. This prototype is shown in Figure 2.2 below.

Figure 2.2 ARMM Prototype presented July 2, 1996

3.0 Crack Maintenance Practices In Texas by State District

There was a need for information on Texas state district for crack sealing and procedures for implementation of the crack sealer before the current prototype could be developed. This chapter reports the findings of a survey by the University of Texas team. These findings were used to set limits and compare the efficiency (chapter 4.0) of the ARMM to a traditional crack sealing crew.

3.1 State District Surveys

In the spring of 1993 a survey form was developed to find out how each of the state's districts (Figure 3.1) were attempting to maintain cracking in asphalt cement pavements. The survey contained questions concerning methods and materials employed, contracting procedures, unit prices, and typical crew organization. The survey form was sent out by the local Texas Department of Transportation (TxDOT) and a response rate of 96% (23 out of 24 districts) was achieved within approximately six weeks (Malek 1994).

Appendix A contains an example survey and a summary of district survey responses. A blank answer area indicates that no response was given and/or the response should have been zero for that particular question. "N/A" indicates that the respondent felt that the question did not apply to their district. District 21 did not participate in the survey and district 22 is non-existent.

The last portion of Appendix A indicates the expenditures by district on crack and joint sealing for the first five months of the 1993 fiscal year for both in house and contracted work (Texas Department of Transportation, 1993). The corresponding unit prices are also included. The state's "Maintenance Efficiency and Analysis Report" contains code 221 for asphalt rubber/rubberized materials and code 222 for all other materials that are used to seal/fill cracks and joints. This is the first fiscal year that expenditures for crack and joint sealing/filling were tracked as a separate entity which is why the data covers less

than one year. The right hand column contains statewide averages for applicable categories. The only exception to this is for the expenditures for codes 221 and 222. The "state averages" column contains total dollar expenditures for in-house work for the time period of September 1, 1992 through February 1, 1993. The corresponding unit rates reflect state averages.

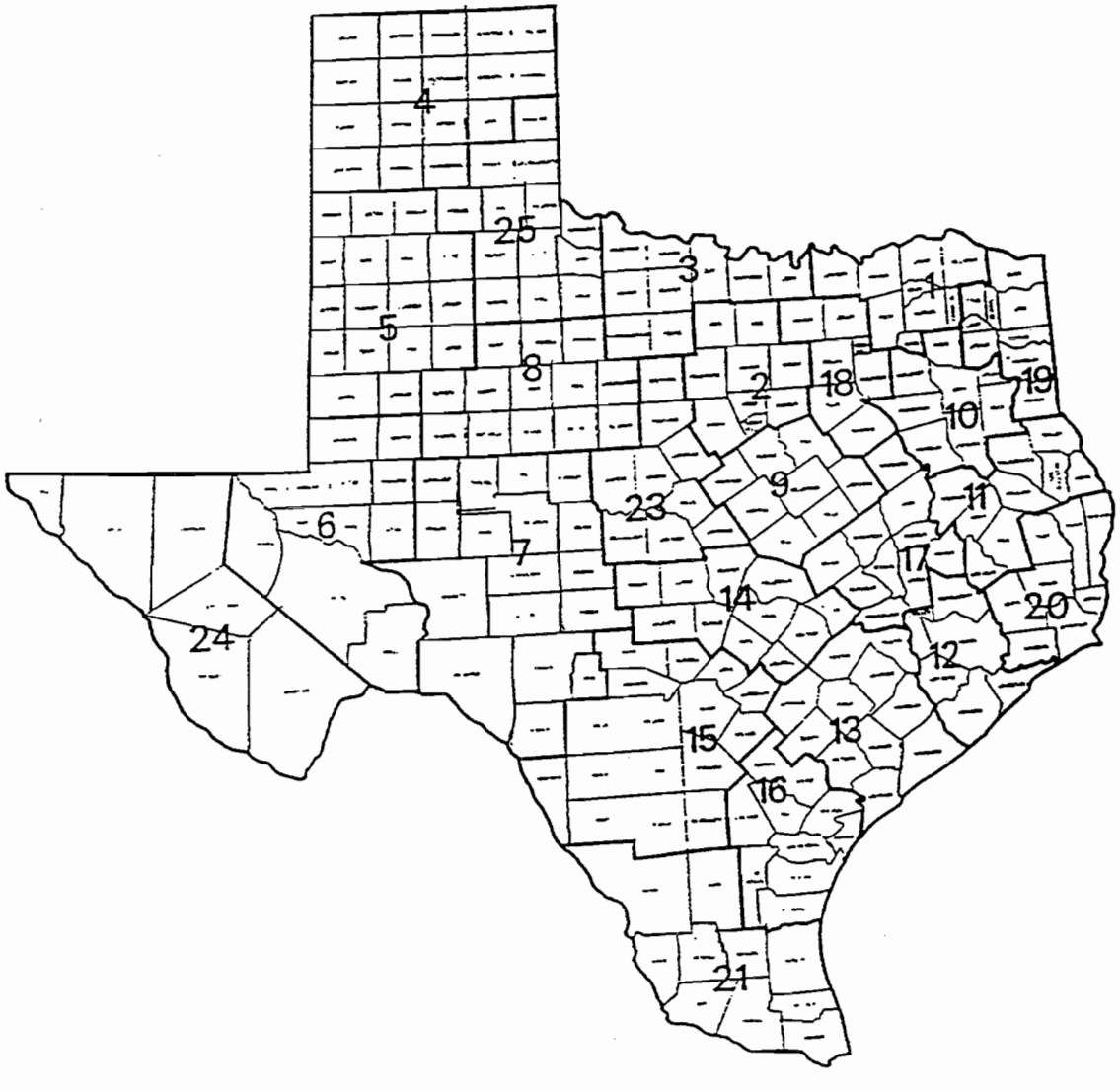


Figure 3.1 TxDOT Districts

Some districts supplied additional information that is noteworthy;

- District 6 indicated that their crew size varied according to crack density, effectiveness of employing crack cleaning methods, and traffic control requirement.
- District 11 responded that one pound of material covers approximately 10 linear feet of cracking.
- District 13 assigns an additional man to spread blotting material (sand or cover rock) when material tracking is prevalent.
- District 25 indicated that the rate of application is approximately 55 gallons per lane mile at a cost of $12 per gallon resulting in an approximate total cost of $660 per lane mile.

3.2 Methods and Materials Used

Twenty out of twenty-three districts (87%) responding, indicated that they perform hot-pour applications 90% or more of the time. The percentage of cold pour applications has been low but there have been sample projects conducted (Malek, 1994) that suggest that the percentage of cold applied materials used will be increasing. The operation of cutting/routing the cracks was almost non-existent among the 23 districts.

Crafco materials have been the sealant of choice amongst most of the districts. Other materials by manufacturer are Kengo, Koch, Allied, and Elf which have been applied in some districts. When contracting work out, 15 out of 23 districts (65%) reported that they furnish the material 100% of the time. Only two districts indicated that the contractor furnished the material all of the time. It appears that the Texas Department of Transportation believes there is an advantage to controlling the amount of material dispensed in order to minimize waste and control the cost of contracted work.

3.3 Contracting Practices

The percentage of work that is contracted out varies by district. Survey respondents indicated that an average of approximately 57% of all the crack sealing/filling was contracted out statewide. There were 14 districts that said they contracted 70% or more of their crack sealing maintenance projects. The "Maintenance Efficiency and Analysis Report" indicates that 61.6% of the statewide crack maintenance expenditures from September 1992 through January 1993 were contracted (Texas Department of Transportation, 1993). It should be noted that some districts spent much more on crack maintenance than others during the fiscal year of this study. Figure 3.2 shows the relationship between district expenditures on crack sealing/filling from September 1992 through January 1993. It can be seen that some districts such as numbers 1, 11, 13, and 20 spent little or no money on crack maintenance while districts 18 and 24 spent considerable amounts during this time period.

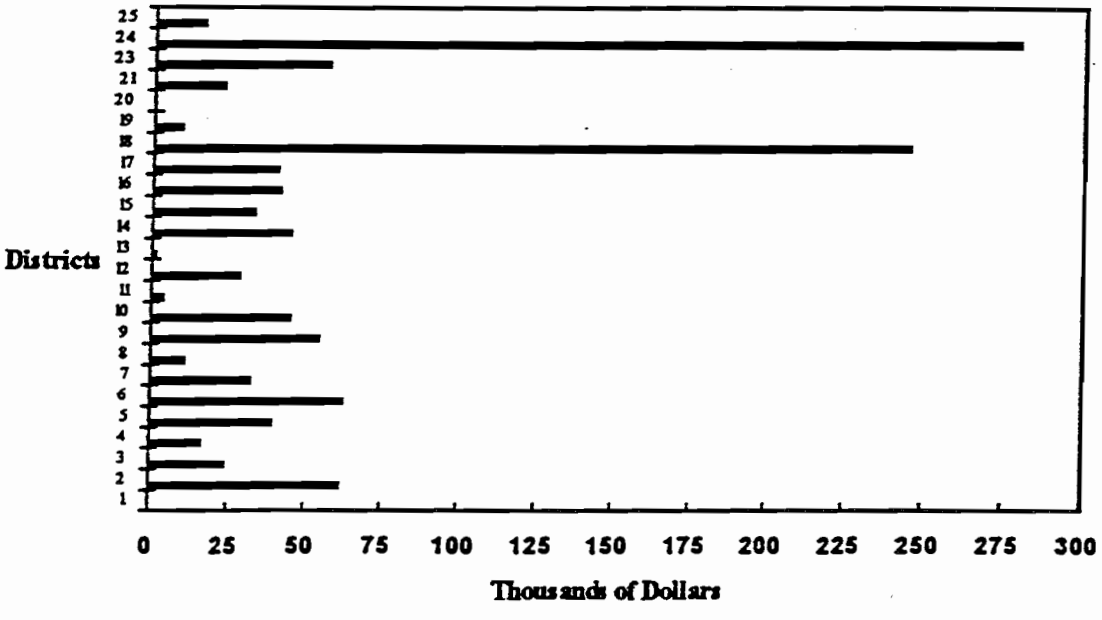


Figure 3.2 Total Crack Sealing/Filling Expenditure by State District

Approximate unit prices for contract crack sealing/filling maintenance were given using one or more of the following measurement classifications: dollars per linear foot of cracking, dollars per lane mile of application, dollars per pound of material, and dollars per gallon of material. Typical crew sizes varied from district o district but generally ranged from 7 to 9 people with the survey average being 7.4 men. A typical crack sealing crew would take on the following structure:

Work Task	Quantity of workers assigned
Foreman (may or may not be a laborer)	1
Drivers (pulling compressor/heating kettle)	2
Crack cleaner	1
Crack filler	1
Squeegee operator	2
Flagmen	0 to 2

Note: The foreman may work (usually as a crack filler or driver) allowing another man to squeegee or possibly be eliminated. Flagmen are often not used if other safety precautions such as lane closure methods are employed. In the field the configuration of this crew can vary dramatically depending on the needs and usual practices of the work tasks.

4.0 Analysis of Productivity of Crack Sealing Maintenance Crews in Texas

In a previous study by Gary Malek (Malek 1994), five projects were observed between the dates of February 10, 1993 and March 7, 1993. Each project was performed by either state, city, or contract forces. Because of differences in crack sizes, degrees of cracking (density), and crew size/organization, the productivity was found to vary considerably from project to project.

The Construction Automation Laboratory used the results of this study to approximate the needed efficiency of the ARMM system and to set some reasonable goals for the work cycle of the ARMM. We use this data to compare with our results which are located in chapter 5.0.

Each project in the study contained the following information:

- Observation date
- Work force
- Temperature
- Site description
- Crew organization
- Types of cracking present
- Location of cracking
- Typical crack sizes present
- Productivity results

- Observations
  - -Equipment
  - -Material
  - -Workmanship
  - -Worker protection
  - -Traffic control
- Project Photographs

Productivity results were obtained by observing 1/2 hour increments of work and counting how many linear feet of sealant were placed and the distance traveled during that time increment. This data was then multiplied by two in order to convert to a full hour. Knowing linear feet of sealant placed and man hours enabled the calculation of linear feet of sealant placed per man hour. Average composite wage rates were obtained from the state (district 14) and from the city of Austin. Multiplying this average by the number of people in the maintenance crew allowed two additional measurements of productivity to be obtained; average labor rate per linear feet of sealant placed and average labor rate per lane mile.

Linear feet of sealant placed was used as a measurement of productivity instead of linear feet of cracking because it was impossible to know the actual length of cracking once the sealant was placed. The squeegee would spread the material further than the actual crack during the finishing and shaping operation. It might be a reasonable assumption to say that the actual crack length would be about 75% of the length of the finished band of sealant but that is only a guess. Therefore it was decided to establish consistency by using measurements in terms of linear feet of sealant placed for all projects.

Every attempt was made to only evaluate the crew during the time that they were actually conducting maintenance operations. Times when the crew was idle for a considerable amount of time were not counted in the one half hour observation times. Typical idle times consisted of one fifteen minute break in the morning and one in the afternoon.

Additional idle times of approximately five to ten minutes resulted when more than one solid block of material was added to the heating of the kettle therefore lowering the overall kettle temperature until the sealant material became liquid. The study attempted to calculate productivity during one half hour increments when the crew was actually filling cracks. This allowed consistent productivity measurements and comparisons to be made from project to project.

4.1 Data Analysis

Pavement cracking occurs throughout most of the roadway's width, but in the situations observed there was a tendency for meandering longitudinal cracking to be prevalent along the pavement edge and in areas where vehicle wheels predominately traveled. Typical crack size in these situation was from 1/8" to 1/2" wide. Figure 4.1 shows an example of common workmanship and crack size of Texas crack sealing maintenance crews.

Figure 4.1 Asphalt Concrete Cracks Sealed by Maintenance Crews

Typical equipment consists of a trailer mounted air compressor that delivers from 90 to 100 psi of pressure. Heating kettles are double boiler constructed equipped with the

necessary temperature controls and agitation devices, and generally have a holding capacity of 250 gallons. Material shaping/finishing using an industrial rubber squeegee is standard procedure.

Crafco is the number one supplier of hot-applied sealant materials in Texas. The sealants being used can generally be classified as rubberized asphalt or practices asphalt rubber materials although there are also materials that have been employed containing emulsions and polymers. The five projects observed all used materials classified as asphalt rubber sealants.

Workmanship for all five projects was relatively equal. Some crews appeared to work somewhat more effectively together as a unit than others which had a slight impact on productivity results. Figure 4.2 is an example crack maintenance crew set-up. It is advantageous to have two crew members to conduct the squeegee operation when pavements have a significant amount of cracking. This allows the crack filler to work continuously and not be required to slow down to a single squeegee operator's pace. The material finishers/shapers walk much more than the crack filler does because they are continually traveling to the end of a crack segment to begin their task. For this reason a crack filler can perform his job much quicker than a squeegee operator.

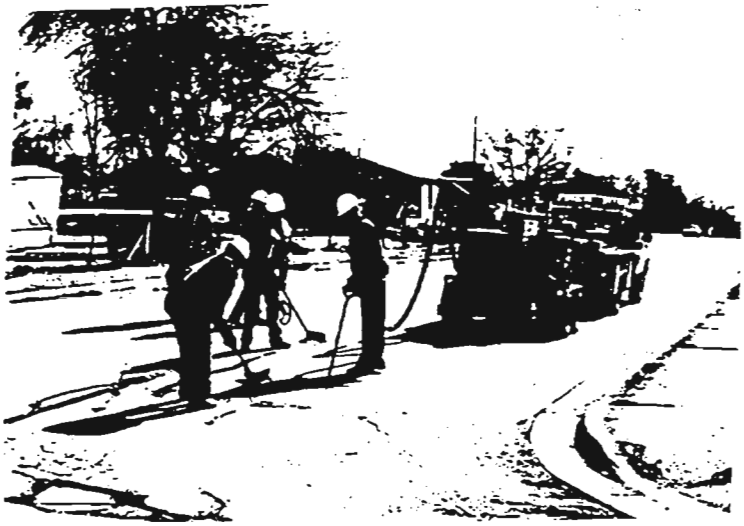


Figure 4.2 Crack Sealing Maintenance Crew

Traffic Control was handled in various ways but for the state highways there are generally two methods that are used. One way is to have complete lane closure by setting up signs, and an arrow board truck to warn traffic that road work is ahead. The second way to handle traffic is to simply conduct the maintenance project as a moving operation closing off only the segment of the lane length between that leading crack cleaning crew and the tailing arrow board truck as the operation moves along. Moving operations may or may not include a flag person depending on individual project characteristics such as traffic levels, roadway classification and geometry.

Survey responses indicated that application of hot applied rubberized materials is the most common type of crack maintenance treatment at the present time. It is highly likely that the cold applied materials will be increasingly used if sample project results continue to prove that there is an advantage to them. There appears to be no reason for the state and local agencies to continue to use hot applied materials if cold products will offer equal material performance, a costs savings, productivity increase, and increased worker safety.

The rate of application of sealant materials will be dependent upon crack size (width and depth), density of cracking, and to a lesser extent whether one or two squeegee operators are utilized. Very wide and deep cracking such as occurred in project 2 will decrease productivity because more time is spent attempting to fill the entire crack reservoir. Conversely, more typical crack sizes (1/8" to 1/2") such as was seen in project 3 yielded much higher productivity rates. Crack density or amount of cracking in a given length of roadway can also affect the productivity rate. Project 1 had an inconsistent amount of cracking in some places which required the crew to temporarily halt operations and proceed further down the road until more cracking was found. this contributed to a decrease in the amount of sealant placed in a given time period. Project 3 had a much more consistent amount of cracking so that the crew never had to search for additional cracking. This resulted in less starts and stops therefore increasing the amount of sealant placed per hour. In addition the third project employed one more squeegee operator than the first project.

Projects 1,4, and 5 seem to have the most common levels of productivity that can be expected in crack maintenance operations that contain typical crack sizes. Project 1 may be interpreted as the low end mainly due to small delays caused by a lower crack density and having only one squeegee operator. Projects 4 and 5 conducted by contract forces had very consistent amounts of cracking so that the crew was able to work at a relatively constant rate. All five projects experienced delays when adding solid blocks of material into the heating kettle because the kettle temperature would drop until those blocks melted. There was no attempt to include those delays in the observation times.

Figure 4.3 indicates the productivity rates for material placed for each of the five projects. It can be seen that the first, fourth, and fifth projects placed similar amounts of sealant per man hour. Figures 4.4 and 4.5 show how each of the first three projects compare to each other for composite labor rate per linear feet of sealant and for composite labor rate per lane mile. No crew rates were available for projects 4 and 5 which were state contracted work. It is believed that the average crew rate for these projects would be slightly less than the city of Austin's rate of $78.72 /hr. Based on this, the contracted projects would fall in between values shown for projects 1 and 3 on both the second and the third figures.

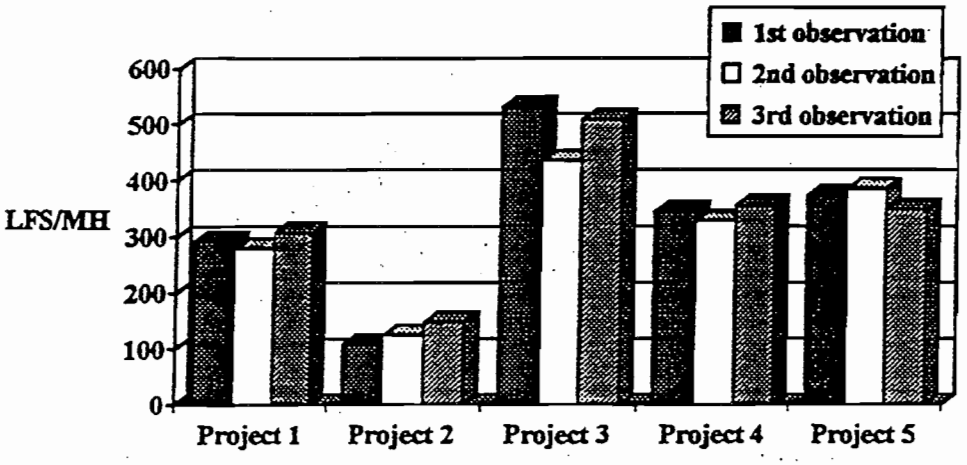


Figure 4.3 Linear Feet of Sealant Placed per Man Hour

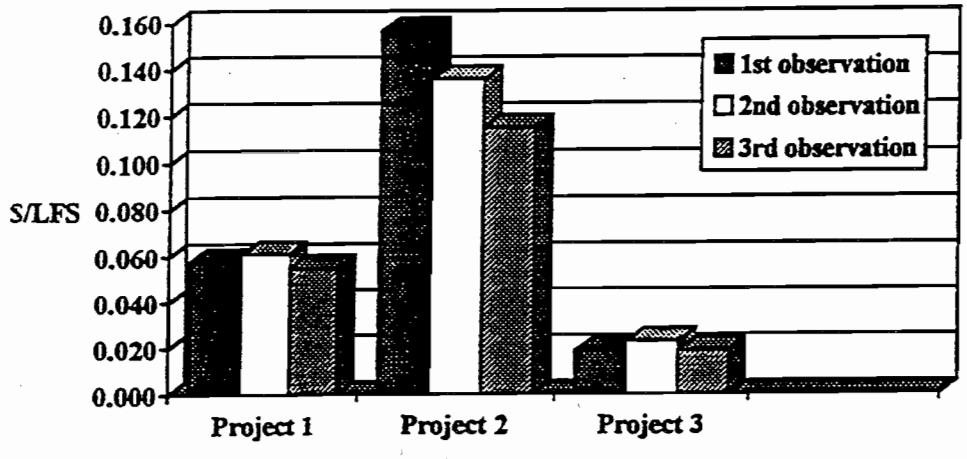


Figure 4.4 Composite Labor Rate per Linear Feet of Sealant

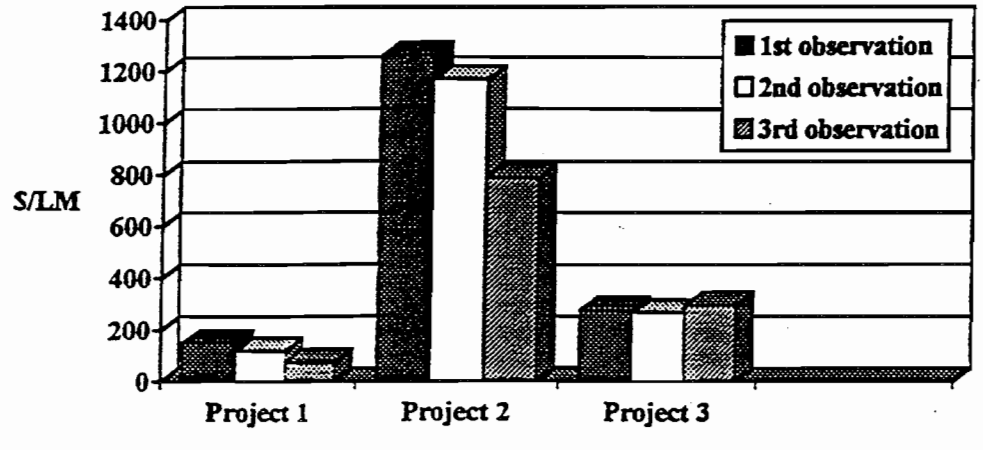


Figure 4.5 Composite Labor Rate per Lane Mile

It appears clear that figures 4.4 and 4.5 contain measurements that may be useful for similar projects but are not representative for projects that have vastly different crack sizes and/or densities. Extremely large cracks as in project 2 slowed the crew down, placing less linear feet of sealant and traveling less distance per hour. The results are an inflated composite labor rate per linear feet of material placed and composite labor rate per mile.

Figure 4.5 indicates that crack size and amount of cracking or density of cracking greatly influence labor expenditures per lane mile. Project 1 had a low composite labor rate because the maintenance crew traveled a considerable amount of distance in one hours time, therefore costing less per lane mile than project 2 which traveled a very short distance in the same period of time due to extremely large crack sizes present.

The amount of cracking that can be treated in a given time interval may depend upon several factors. Three of the largest factors are the following:

1) Width and depth of cracking Large cracks can drastically reduce the amount of linear feet of cracking that can be treated in a given time interval. More time is spent attempting to fill and squeegee larger cracks than is spent with typical cracking.

2) Density or amount of cracking Sporadic cracking has a tendency to reduce productivity because much of a given time interval may be spent traveling to areas of additional cracking. This travel may be as insignificant as walking ten or twenty feet or may be as involved as temporarily stopping operations and moving equipment to where additional cracking is present.

3) Crew Composition Depending on the amounts of cracking present, having only one squeegee operator can have an impact on productivity. To maximize the amount of cracks treated in a given time interval, The crack filler should determine the pace of the operation. This will usually mean that two squeegee operators will be needed so as not to delay the crack filling.

These are only a few of the more important conditions that can affect productivity. The following things can also play a role:

- Original material preparation time (block melting)
- Idle time while waiting for kettle to attain the proper temperature when additional material is added
- Worker fatigue
- Temperature (too hot or too cold)

Productivity comparisons should only be made between projects that can be considered similar. Data such as linear feet of cracking maintained or sealant placed is sufficient for a calculation of productivity between projects with similar cracking characteristics. Pounds or gallons of material placed can be used to compare similar projects assuming that both crack fillers apply roughly equal amounts of material or any given cracking condition.

The combination of projects 1,4, and 5 had an average linear feet of sealant placed per man hour of approximately 335 linear feet. It is likely that this average could have been increased if more consistent cracking densities were found on project 1 and a slightly more experienced crew was used in projects 4 and 5. An educated guess is that the average of the three projects could have been increased by ten percent or so which would result in an average of approximately 375 linear feet of sealant placed per man hour. These numbers are fairly accurate for projects that contained typical crack sizes (1/8" to 1/2"), and included relatively consistent cracking densities.

4.2 Conclusions

There is no doubt that crack sealing/filling will continue to be an inexpensive alternative to extend asphalt concrete pavement life. The typical equipment utilized consists of a trailer mounted air compressor and a 250 gallon heating kettle which are pulled by separate vehicles, at least one other vehicle is used to act as a warning to traffic and usually

contains an attached arrow board. The use of rubber squeegees are standard to finish and shape sealant materials. Although crew size can vary, it is typical to see 7 to 9 persons conducting maintenance operations. Although recommended practices and procedures sometimes include the use of such equipment as routing machines and lances, they are rarely used if at all in the state of Texas for crack sealing/filling purposes.

Productivity can vary a considerable amount from project to project but generally is dependent on crack size, crack density, and crew composition. In addition, some heating kettles may be more effective at melting additional blocks of material that are added during the maintenance project which may lower idle time and increase productivity.

5.0 Analysis of Productivity of Automated Road Maintenance Machine

This chapter defines a framework for the productivity evaluation of a crack sealer and makes preliminary estimates of anticipated productivity from observation of field trials. Coordinating this information with the data in chapter 4.0 gives an approximate idea of how the ARMM system compares to traditional maintenance crews.

5.1 Introduction

To measure the productivity of the ARMM all of the tasks necessary for the process were first classified into three major categories.

- Mobilization/Demobilization
- Crack tracing and path planning
- Blow, seal, finish

Each of these categories were detailed further to isolate specific tasks which could then be timed. This method of specifying the tasks for a given category allows for more accurate results when calculating productivity. Problem areas that affect productivity are easily identified and corrected. The linear feet of sealant poured is measured and the productivity was calculated.

The mobilization/demobilization category consists of tasks including the attachment of the melter to the truck and the attachment of the X-Y table to the truck/melter, the raising and lowering of the canopy, and the connection of the cables. The canopy must be lowered whenever in the ARMM is in transit. Only two people are required to raise or lower the canopy safely. The crack tracing and path planning category consists of tracing the crack, line snapping, point adjusting, and path planning. The operator traces the image of the

crack on the monitor. The tracing of the crack is adjusted to ensure the most accurate representation by the computer and then the computer calculates the best path for the ARMM to follow. The blow, seal, finish category consists of tasks necessary to blow, seal, and squeegee the cracks. This category is the major component affecting, and in some cases limiting the productivity.

5.2 Methodology

The productivity was calculated by observing the operation of the ARMM during a series of demonstrations at our research facility. The ARMM was operated through a workzone created at the lab. The cracks within the workzone were designed to represent various types and patterns of cracks present in pavements. The time required for the ARMM to seal the cracks in the series of workspaces in the workzone was recorded. Next, the amount of sealant placed in that time was determined. The linear feet of sealant placed was then divided by the time to produce a productivity in Linear Feet Sealant/Hour (LFS/H). This figure is divided by 3 to determine Linear Feet Sealant/Man Hour (LFS/MH). The number 3 represents the assumed number of crew members needed to operate the ARMM. The productivity was calculated in LFS/MH in order to compare the productivity of the ARMM to the results calculated from previous productivity studies on conventional crack sealing methods. Choosing a rate that is measured per man hour allows for comparisons to be made when crew sizes vary between various projects. Idle time to add more material to the melter and time to move the ARMM to the next workspace within the same workzone were not included in these preliminary calculations.

It is very difficult to create a model that can be applied to all situations. The data model created for these calculations assume that workspaces follow each other immediately in the workzone. Distances between workzones are impossible to include. The model will contain information on the travel speeds of the ARMM. This data can then be used to determine the time elapsed due to travel between workzones.

5.3 Preliminary Calculations

The productivity figures calculated for this report are very rough estimates. More accurate calculations will be performed later this fall when more data can be collected from actual crack sealing situations. As of now, these calculations show the present productivity of the ARMM, but these figures will increase substantially in the near future.

The following is a summary of data collected from preliminary trials of the ARMM.

Average time to complete workspace - 90 seconds (s)
Anticipated time to complete workspace - 30 seconds (This figure will be used in the calculations).
Average linear feet of sealant poured per workspace - 17 LFS

Average rate = LFS/Time
Average rate = 17 LFS/90 s = 0.567 LFS/s

Conversion to LFS/MH

0.567 LFS/s x 3600 s/hr = 2041 LFS/hr

Data used for comparison was taken from Project Evaluation #1 from the study on conventional crack sealing methods (Malek 1993).

Table 5.1 - Productivity comparison chart: ARMM vs. Conventional method

Item	Conventional	ARMM
Labor	7@Avg. Crew rate $ 17.04/hr	3@Avg. Crew rate $ 17.04/hr
Equipment	$40/day	$100/day
Productivity	Average 2 Lane Mi./day 2045LFS/H	Average 2 Lane Mi./day 2045LFS/H
Material	800 lbs/LM @ $0.15/lb	800 lbs/LM @ $0.15/lb
Average cost per pound of 1 days work	**$0.78/lb**	**$0.38/lb**

5.4 Factors affecting productivity

The ARMM utilizes the hot-pour application method of crack sealing, which is most commonly used in industry today. The productivity of this automated process is affected by several factors. Many of these factors are also common to the conventional hot-pour application method of crack sealing. They include size of cracking (length, width, height), degrees of cracking (density), and idle time while waiting for the melter to come up to application temperature when additional material is added. Factors that are unique to the productivity of the ARMM include the speed at which the ARMM recognizes the cracks in a given workspace, the dexterity of the operator in tracing the cracks, and the speed at which the ARMM actually blows, seals, and squeegees the cracks. The speed of this series of processes is limited by the size of the motor

The current prototype of the ARMM is fitted with a 2 ft-lb. motor which allows the turret travel at approximately 2.5 inches/second. The speed can also be attributed to a 3-to-1

gear reducer which was place on the motor to provide adequate torque to move the turret in the X direction. With the installation of a 5 ft-lb. motor, this speed can be improved to approximately 7.5 inches/second, by approximately 3 times. This faster speed can be achieved safely, but the turret does have a speed limit. The speed of the turret must be coordinated with the rate of flow of the materiel in order to assure proper amounts of sealant fill the cracks. The productivity of the ARMM will also be improved as the operator becomes more proficient with the crack tracing procedure. The introduction of faster computers in the future will increase productivity as well. Faster computers would allow for faster line snapping and path planning.

5.5 Conclusions

The current prototype of the ARMM is anticipated to seal cracks as fast as the conventional method of crack sealing. Preliminary calculations show the average productivity for a typical conventional crack sealing project is 2045 LFS/H. However, the ARMM needs at most 3 crew members to operate. This dramatically decreases the costs and also the safety factors involved with this type of operation. The savings in an operation similar to that of Project #1 (Malek 1993) is about 40 cents per pound of sealant. These savings are due, in most part, to the fact that the crew size is dramatically decreased. Previous studies show that labor costs can be as much as 60% of total costs (Osmani 1995). Other benefits include the fact that the ARMM can be operated at night. The fact that the ARMM can work longer days (work shifts at night) means that shorter schedules will be needed to complete jobs thus reducing inconvenience to motorists caused by closed lanes. The accuracy by which the sealant is poured is greatly enhanced. The human error aspect of missing a crack is now reduced to the ability of the operator when tracing the image of that crack on the monitor. If a mistake is made, it can be easily erased. Since there is no intense manual labor involved, the ARMM is not affected by worker fatigue. Nonetheless, the productivity of the ARMM can and will be improved in the near future.

6.0 Evaluation of an Automated Road Maintenance Machine

This chapter is drawn from a preceeding study (Osmani 1994). It analyzes the economic impact and the need for an ARMM in some detail. The study uses the ARME (Automated Road Maintenance Evaluation) model which is described in detail in (Osmani 1994).

6.1 Introduction

The need for sealing cracked pavements will remain for as long as there are paved roads. The technologies used in crack sealing have advanced slowly. New, more effective hot poured asphaltic binders are being continuously improved and significant advances are being made in cold poured materials. The application techniques, however, have remained much the same over the last 15 years. Crack sealing crews are typically composed of 7 personnel, 3-4 of which are involved in identifying, cleaning, or sealing cracks [McNeil 1992]. A prototype automated system that will drastically reduce these labor requirement is evaluated from TxDOT, private contractors', and road-users' perspectives in this study.

Crack sealing operations are a significant component of a typical road maintenance budget. It is estimated that approximately $188 million per year are spent on crack sealing by all the public agencies in the United States excluding expenditures by private organizations, airports and the military [Hsieh 1992]. The TxDOT spends approximately 7 million dollars every year on crack sealing which is approximately 1.5% of their total routine maintenance budget [TxDOT 1993C]. The most significant fact concerning these expenditures is the percentage that is spent on labor which ranges from 40 to 60% of the total as discussed in detail later in this chapter. Through automation, these labor expenditures can be reduced drastically.

There are many motivators for developing an Automated Road Maintenance Machine (ARMM). In addition to the quantifiable categories of labor-costs and user-costs savings, the system can have positive impacts in the areas of worker safety, job quality, and

working environment. The most important of these concerns is safety, and the best way to improve worker safety in a highway environment is to remove the worker from danger. A survey of public agencies throughout the United States and Canada show that there were 3681 injury accidents related to road maintenance activities in 1991 [Deng 1992]. An ARMM system that minimizes the requirement for on-road labor will certainly reduce these statistics significantly if implemented nationwide.

The ARMM system is described in more detail in subsequent chapters. In chapter 12.0 the total cost for the ARMM is estimated. Its selling price would have to be higher to include marketing and profit. Table 12.1 shows a cost breakdown for the ARMM.

6.2 Economic Analysis

Although the study area for this evaluation is the state of Texas, the implications of automated crack sealing have nation-wide applications. Most crack sealing in the United States is performed by in-house maintenance crews, although some states utilize contract work to supplement their capabilities.

The automation needs assessment survey for the TxDOT maintenance activities has shown that crack sealing is a very desirable candidate for automation (Osmani 1994). Automated crack sealing operations throughout the state would require approximately 26 units based upon current expenditures. Of these 26 units, 10 would be required by the TxDOT and the other units would be required by the private contractors working for TxDOT. This estimate is based upon current contracting trends as discussed in detail later in this section.

The costs of the ARMM system can be divided into four major categories: initial (discussed above), maintenance, operational, and overheads. The savings of the ARMM system can be divided into two major groups: direct maintenance cost savings and user-cost savings. The direct maintenance cost savings can result from labor reduction, improved productivity, and reduced material wastage. The user-cost savings mostly result

from reduction in lane closure times due to expedited operation of the ARMM system. Besides the costs and savings of the ARMM system, other economic factors to be considered in the life cycle analysis are the cost impact (expenditures per year), the unit costs for current practices and the division of current maintenance expenditures into labor, material, equipment (rentals), overheads, etc.

For the life cycle economic analysis, the computer spreadsheet from the Automated Road Maintenance Evaluation (ARME) model was used (Osmani 1994). The economic analysis performed by the spreadsheet includes the estimation of both the direct (TxDOT or private contractors) and indirect (users) cost savings. The input data included the costs of the ARMM system, number of typical crew members to be reduced by the ARMM system, and the TxDOT or private contractors' expenditure data. The spreadsheet then automatically converted all these costs and savings into net present worth (NPW) values to compute the life cycle savings (or extra expenditures) of the ARMM system, from the TxDOT or private contractors' perspective. Three different minimum attractive rate of return (MARR) scenarios for the both the TxDOT and the private contractors were tested.

After the computation of the direct maintenance cost savings for the life cycle of the ARMM system, a case study on a 20 mile long Interstate 35 section was also performed to determine the user-cost savings. The user-costs are estimated at the project level as they can vary significantly from the maintenance operations performed on a busy urban expressway to a low volume rural farm-to-market road. Hence, an economically unfeasible ARMM system from TxDOT or private contractors' perspective can be very feasible from users' perspective if used on a urban expressway because of the high user-cost savings.

All the economic analysis parameters of the ARMM system are discussed in detail as following:

Costs & working life of the ARMM system: The initial cost of the system is estimated to be $70,000 as shown in table 7.1. Other annual costs of the system are operating costs

($4,500), maintenance costs ($3,000), and overheads ($3,750). The working life of the system is expected to be six years [McNeil 1992B].

Units of Measurement: The unit of measurement used, by the different districts of TxDOT, for crack sealing vary from pounds (lb.) and gallons to square yards, linear foot, or lane-miles. However, all the districts are required to convert this data in either pounds (for asphaltic rubber materials) or gallons (for all other materials) when reporting this data to the maintenance division in Austin. This data is recorded and maintained under code 221 (asphaltic rubber materials, in pounds) or code 222 (all other materials, in gallons) by the maintenance division. All crack sealing done using asphaltic rubber materials (code 221) is hot poured while most crack sealing done using other materials (code 222) is cold poured. As the productivity rates for both these methods are different, besides the change in equipment requirements, the economic analysis performed is divided into either hot pour or cold pour crack sealing throughout this report. Hence, hot pouring refers to code 221 and is reported in pounds (lb.) whereas cold pouring refers to code 222 and is reported in gallons.

The crack sealing operations can be further divided into two major categories: in-house work by TxDOT and contracted work. The TxDOT maintenance division also keep track of both the in-house and contracted expenditures and the unit-costs. Hence, for this analysis, the crack sealing operations in Texas were divided into the following four categories:

- In-house crack sealing using hot pouring,
- Contracted crack sealing using hot pouring,
- In-house crack sealing using cold pouring, and
- Contracted crack sealing using cold pouring.

TxDOT annual expenditures on crack sealing: The annual expenditure data, from the TxDOT maintenance division in Austin, was only available for the first nine months of 1993 when this evaluation study was performed. Also, this is the first year that TxDOT is

tracking detailed cost data on crack sealing. The total expenditures on crack sealing are estimated to be $7 million in 1993, and the expenditure trend from the first nine-months of 1993 was used to estimate the annual expenditures for the whole year as shown in table 6.2 [TxDOT 1993B and 1993C].

Table 6.1 TxDot Annual Expenditures on Crack Sealing

Crack Sealing Category	Expenditure trend from the first nine months of 1993	Estimated Annual expenditures in 1993
In-house expenditures (hot pour)	24.08%	$1,685,600
Contracted expenditures (hot pour)	61.72%	$4,320,400
In-house expenditures (cold pour)	9.84%	$688,800
Contracted expenditures (cold pour)	4.36%	$305,200
Total	**100.00%**	**$7,000,000**

Unit-Costs and Unit-work performed in Texas: The unit-costs for both hot pour and cold pour crack sealing (for both in-house and contracted work) were available from TxDOT. These unit-costs were used with the expenditure data to determine the annual work performed in pounds or gallons. However, to see the relativity between these two units of work, both of these were also converted into lane-miles by utilizing the results of an earlier research study conducted by TxDOT [Malek 1992]. The basic findings of this study were that approximately 800 lb. of hot pour materials (Crafco Road Saver #523) or 55 gallons of cold pour materials (Kengo) are required to seal a typical lane-mile of a road section. Also, the productivity rates were 2 lane-miles/day (or 1600 lb./day) for hot pour crack sealing and 3 lane-miles/day (or 1600 lb./day) for cold pour crack sealing.
These unit-costs and annual units of work for Texas are given on the next page:

In-house hot pour crack sealing:

Unit-costs = $0.73/lbs.

$$\text{Annual unit-work performed} = \frac{(\$1{,}685{,}600)}{(\$0.73/\text{lbs.})} = 2{,}309{,}041 \text{ lb.}$$

$$\text{Annual unit-work performed} = \frac{(2{,}309{,}041 \text{ lbs.})}{(800 \text{ lbs./lane-mile})} = 2886 \text{ lane-miles}$$

Contracted hot pour crack sealing:

Unit-costs = $0.70/lbs.

$$\text{Annual unit-work performed} = \frac{(\$4{,}320{,}400)}{(\$0.70/\text{lbs.})} = 6{,}172{,}000 \text{ lb.}$$

$$\text{Annual unit-work performed} = \frac{(6{,}172{,}000 \text{ lbs.})}{(800 \text{ lbs./lane-mile})} = 7715 \text{ lane-miles}$$

In-house cold pour crack sealing:

Unit-costs = $4.76/gallons

$$\text{Annual unit-work performed} = \frac{(\$688{,}800)}{(\$4.76/\text{gallons})} = 144{,}706 \text{ gallons}$$

$$\text{Annual unit-work performed} = \frac{(144{,}706 \text{ gal.})}{(55 \text{ gal./lane-mile})} = 2631 \text{ lane-miles}$$

Contracted cold pour crack sealing:

Unit-costs = $8.40/gallons

$$\text{Annual unit-work performed, gallons} = \frac{(\$305{,}200)}{(\$8.40/\text{gallons})} = 36{,}333 \text{ gallons, or}$$

$$\text{Annual unit-work performed} = \frac{(36{,}333 \text{ gal.})}{(55 \text{ gal./lane-mile})} = 661 \text{ lane-miles}$$

Percentage of Labor: The percentage of expenditures spent on labor were also computed using the statewide summary report of TxDOT expenditures [TxDOT 1993C]. These percentages are only for the in-house work but due to the absence of any data from private contractors, these were also used for the private contractor scenarios in the economic analysis. The data from TxDOT divides the total expenditures into labor, equipment, material, overheads, and contract; and the percentage of labor expenditures were computed as following:

$$\% \text{ expenditures on labor} = \frac{\text{labor expenditures} + 50\% \text{ overheads}}{\text{total expenditures} - \text{contract expenditures}}$$

Based on the above formula and the data available from the TxDOT for the first nine months of 1993, the following labor expenditure percentages were obtained:

$$\% \text{ expenditures on labor, in-house hot pouring} = \frac{(\$830{,}528 + 50\% \times \$1{,}703{,}777)}{(\$5{,}657{,}247 - \$2{,}010{,}818)}$$

$$= 46.14\%$$

$$\% \text{ expenditures on labor, in-house cold pouring} = \frac{(\$333{,}052 + 50\% \times \$242{,}580)}{(\$970{,}504 - \$221{,}970)}$$

$$= 60.70\%$$

Working capacity of ARMM Systems: To estimate the working capacity of the ARMM system, first the current productivity rate for the hot pouring and cold pouring cases was determined for a typical crew. It was assumed for this study that the productivity rates for both the in-house and contracted work are not significantly different. The current annual productivity rates in pounds or gallons were determined using the results of the TxDOT research study, discussed before, as following:

Productivity rate, hot pour crack sealing (both in-house & contracted)

$$\frac{2 \text{ lane-miles}}{\text{day}} \times \frac{800 \text{ lbs.}}{\text{lane-mile}} \times \frac{250 \text{ workdays}}{\text{year}} = \frac{(400{,}000 \text{ lbs.})}{\text{year}}$$

Productivity rate, cold pour crack sealing (both in-house & contracted)

$$\frac{3 \text{ lane-miles}}{\text{day}} \times \frac{55 \text{ gallons}}{\text{lane-mile}} \times \frac{250 \text{ workdays}}{\text{year}} = \frac{(41{,}250 \text{ gallons})}{\text{year}}$$

It is estimated that an ARMM system can enhance the current productivity by 20% due to faster effective operating speed and reduced setup/removal times between widely spaced cracks on long road sections. Also, a 90% efficiency rate was used to account for the unexpected breakdowns for 10% of the time. Hence, the rated capacity of the automated crack sealer was estimated as following:

ARMM productivity rate, hot pour crack sealing (both in-house & contracted)

$$\frac{(400{,}000 \text{ lbs.})}{\text{year}} \times 1.20 \times 0.90 = \frac{(432{,}000 \text{ lbs.})}{\text{year}}$$

ARMM productivity rate, cold pour crack sealing (both in-house & contracted)

$$\frac{(41{,}250 \text{ gallons})}{\text{year}} \times 1.20 \times 0.90 = \frac{(44{,}450 \text{ gallons})}{\text{year}}$$

Number of Automated Crack Sealers required for Texas: The total number of ARMM systems required for Texas were determined for the four basic cases as following:

ARMM systems required for in-house work with hot pour materials

$$\frac{(2{,}309{,}041 \text{ lbs.})}{(432{,}000 \text{ lbs.})} = 5.3 \text{ or } 6 \text{ systems}$$

ARMM systems required for contracted work with hot pour materials

$$\frac{(6{,}172{,}000 \text{ lbs.})}{(432{,}000 \text{ lbs.})} = 14.3 \text{ or } 15 \text{ systems}$$

ARMM systems required for in-house work with cold pour materials

$$\frac{(144{,}706 \text{ gallons})}{(44{,}550 \text{ gallons})} = 3.2 \text{ or } 4 \text{ systems}$$

ARMM systems required for contracted work with cold pour materials

$$\frac{(36{,}333 \text{ gallons})}{(44{,}550 \text{ gallons})} = 0.8 \text{ or } 1 \text{ systems}$$

Total number of ARMM systems required for Texas = 24 to 26 systems

Sensitivity Analysis : The following Minimum Attractive Rate of Return (MARR) scenarios were used for the sensitivity analysis:

In-house scenarios, MARR range = 4%, 6% and 8%

Contracted scenarios, MARR range = 20%, 25% and 30%

User-Costs: Lane closures due to crack sealing work on highways reduce the capacity of the highway, and hence result in additional fuel consumption, delays, harmful emissions, and higher operating/other costs of the vehicles. For determining the user-cost savings for typical crack sealing operations, a case study was performed for the 20 mile segment of

Interstate 35 between the cities of San Marcos and New Braunfels. This is a four lane segment with a consistent ADT of approximately 55,000 throughout the section length. The percentage of trucks on this segment is estimated to be 20% due to the inter-city nature of the highway.

A typical one lane closure was considered for both the conventional and automated operation. The user-costs for a conventional crack sealing operation from 8 AM to 4 PM, including an hour both in the morning and evening as setup and removal time, were found out to be $10,625 from the QUEWZ-E model. With a 20% estimated reduction in the closure time or 1 hour due to the expedited ARMM operation (8 AM to 3 PM), the user-costs came out to be $8,368. To determine the number of crack sealing operations on this section, an "operation" was defined as one-lane mile of crack sealing. Using this definition, the annual number of these operations on this segment were estimated as following:

$$\frac{(20 \text{ x } 4 \text{ lane-miles})}{(183{,}550 \text{ Texas lane-miles})} \text{ x } \frac{(13{,}893 \text{ operations})}{\text{year}} = \frac{6 \text{ operations}}{\text{year}}$$

This data was then entered into the spreadsheet to determine the life-cycle savings from the users and TxDOT or private contractors perspectives, for this project.

Results of Economic Analysis: The detailed output of the economic analysis, from the computer spreadsheet, are given in appendix B. Figures B.1a & B.1b provides results for in-house hot pour crack sealing, figures B.2a & B.2b for contracted hot pouring, figures B.3a & B.3b for in-house cold pour, and figures B.4a & B.4b for contracted cold pouring.

The ARMM can offer significant savings for both TxDOT and private contractors (with much higher MARR than TxDOT) for the cold pour than the hot pour crack sealing. The basic reason behind this is that the most significant savings of the ARMM system are from the reduction in labor, and the cold pouring has a much higher share of labor in the total expenditures (60.70% for cold pour verses 46.14% for hot pour).

Another major finding of this research is the amount of user-cost savings which can result from the ARMM system. Crack sealing by the ARMM on a small segment of Interstate-35 alone (0.04% of all the crack sealing operations in Texas, 6/13,893) can result in so significant user-costs savings that even an ARMM system with no direct maintenance cost savings is also feasible(Osmani, 1994).

6.3 Qualitative Analysis

There are many intangible or qualitative benefits of the ARMM system. These include improvements in concerns including safety, quality, working environment, etc. The working environment will see improvements by reducing noisy, dirty conditions and limiting meticulous, exhaustive activities. Improvements in safety are numerous especially the reduction in health hazards and the potential for physical injuries for the work crew. The improvement in project quality is also expected.

A qualitative analysis of the benefits of ARMM system was also performed using the tools developed from the ARME model (Osmani 1994). The ARMM system scored a high Overall Concern Rating number of 8.4 on a 0 to 10 scale. Hence, it can be safely concluded that this system is very desirable from the qualitative perspective in addition to the economic perspective.

6.4 Environmental/Energy Analysis

The estimated reduction in the emissions and fuel/oil consumption on the case study were also reported by QUEWZ-E model. These results are shown in table 6.4. As can be seen from this table, the emissions and fuel/oil consumption decreased between 15% to 23%; and hence it is estimated that very significant reductions in these variables will result if the ARMM system is implemented on the state level.

Several scenarios for the implementation of the ARMM system, from both the TxDOT and private contractors' perspectives were analyzed in this evaluation study. In every scenario of the economic analysis, the system proves to be very feasible. Also, the user-cost savings of this system are tremendous as can be seen from the case study for typical crack sealing operations on an interstate highway. Moreover, in both the qualitative and environmental/ energy analysis, the ARMM system came out to be very feasible.

If the ARMM systems are implemented at the statewide level, the direct savings are estimated to be $2.43 million for the TxDOT (4% MARR) and $2.64 million for the private contractors (20% MARR). The total user-cost savings are estimated to be $11.0 million for the 3229 miles of the interstate highways in Texas. Also, the actual user-cost savings would be much higher as the savings on urban freeways, farm-to-market roads, and secondary roads, etc. are not included in this $11.0 million estimate. The analysis is, however, limited in scope due to the development stage of this ARMM system. But, automation of crack sealing is inevitable: the economic benefits are numerous besides the very significant improvements in qualitative, environmental and energy concerns.

TABLE 6.2 Reduction in Environmental and Energy Factors for the Case Study

	Conventional Operation	Automated Operation	Reduction (%)
Carbon-monoxide, CO (Kgs.)	57.4	48.2	19.1
Hydrocarbon, HC (Kgs.)	5.8	4.8	20.8
Oxides of Nitrogen, NOx (Kgs.)	1.6	1.3	23.1
Fuel Consumption (Gallons)	1404.1	1179.9	19.0
Oil Consumption (Gallons)	38.7	33.6	15.1

6.5 Conclusions and Recommendations

The feasibility of implementing an Automated Road Maintenance Machine (ARMM) system in Texas was evaluated using the ARME model's technology evaluation phase.

In evaluating the feasibility of an ARMM system; estimated maintenance cost savings, user-cost savings, reduction in emissions/energy consumption, and the improvements in the qualitative factors (safety, quality, working environment, etc.) were determined. Several scenarios, from both the TxDOT and private contractors' perspectives, were analyzed; and in every scenario the system proved to be feasible for all the analysis parameters (economic, qualitative, environmental and energy).

If the ARMM systems are implemented at the statewide level, the total life-cycle direct savings are estimated to be $5 million with $2.43 million for the TxDOT and $2.64 million for the private contractors. The life-cycle user-cost savings are estimated to be $11.0 million for the 3229 miles of the interstate highways in Texas, and the total user-cost savings would be much higher as the savings on urban freeways, farm-to-market roads, and secondary roads, etc. are not included in this estimate. Savings could exceed hundreds of millions of dollars nationwide.

As the productivity and cost of the ARMM are more accurately determined in the upcoming year, economic analysis results can be adjusted.

7.0 Control Loop of The Automated Crack Sealing Process

7.1 Introduction

The field prototype system developed by the University of Texas at Austin is a unique system which has the potential to greatly reduce the hazards currently associated with the crack sealing process. The operation of the automated crack sealer includes several different steps. First, a computer imaging system is used to detect cracks to be sealed on the roadway. The system operator identifies the crack location by drawing a line over the crack image on a screen using a mouse, and the xy coordinates data of the drawn lines are stored in a variable within the crack detection software. This information is processed and fed to the motor controller which commands the xy table to follow the cracks and operate the cleaning air and sealant valves. A drag behind squeegee is included to flatten the sealant and rotate the sealing turret. The entire equipment system is then moved to the next crack location and the process is then repeated. Figure 7.1 shows the physical configuration of the ARMM system.

7.2 Tele-Operation Architecture

Man-Machine Interaction

The automated pavement crack sealer was originally designed to be fully automated (Haas 1994). To detect and map cracks in the machine's work space, the crack networks have to be represented in a form that can be processed by an algorithm. Several algorithms to automatically analyze the pavement image features and accurately select the crack locations have been proposed and experimented with during the last few years. However, it was found that automatically identifying cracks in the pavement could not be done in real-time. This was due to the fact that the vision system can be mislead by oil marks, skid marks, previously sealed cracks, and other noise that is inherently found in video data using computer vision. The range sensor could distinguish real cracks from rutting and sealed cracks by range information. However, the range sensor took much time to scan a

work space, and data fusion of multi-sensors made the data analysis more complicated slowing down the data processing. Thus, it was identified that the autonomous crack detection methods proposed were technically feasible but time consuming. Therefore, an operator had to be brought into the control loop (Haas 1994) in an effort to solve the problem. As a result, the current automated crack sealer combines computer vision and operator identification of the cracks to be sealed in order to map their exact location in the machine's work space coordinates. Figure 7.2 briefly describes the process flow for computer assisted tele-operation of the automated crack sealer. The resulting tele-operation architecture involves several steps including: (1) image acquisition, (2) crack detection and mapping, (3) path planning, and (4) manipulator and end effector control.

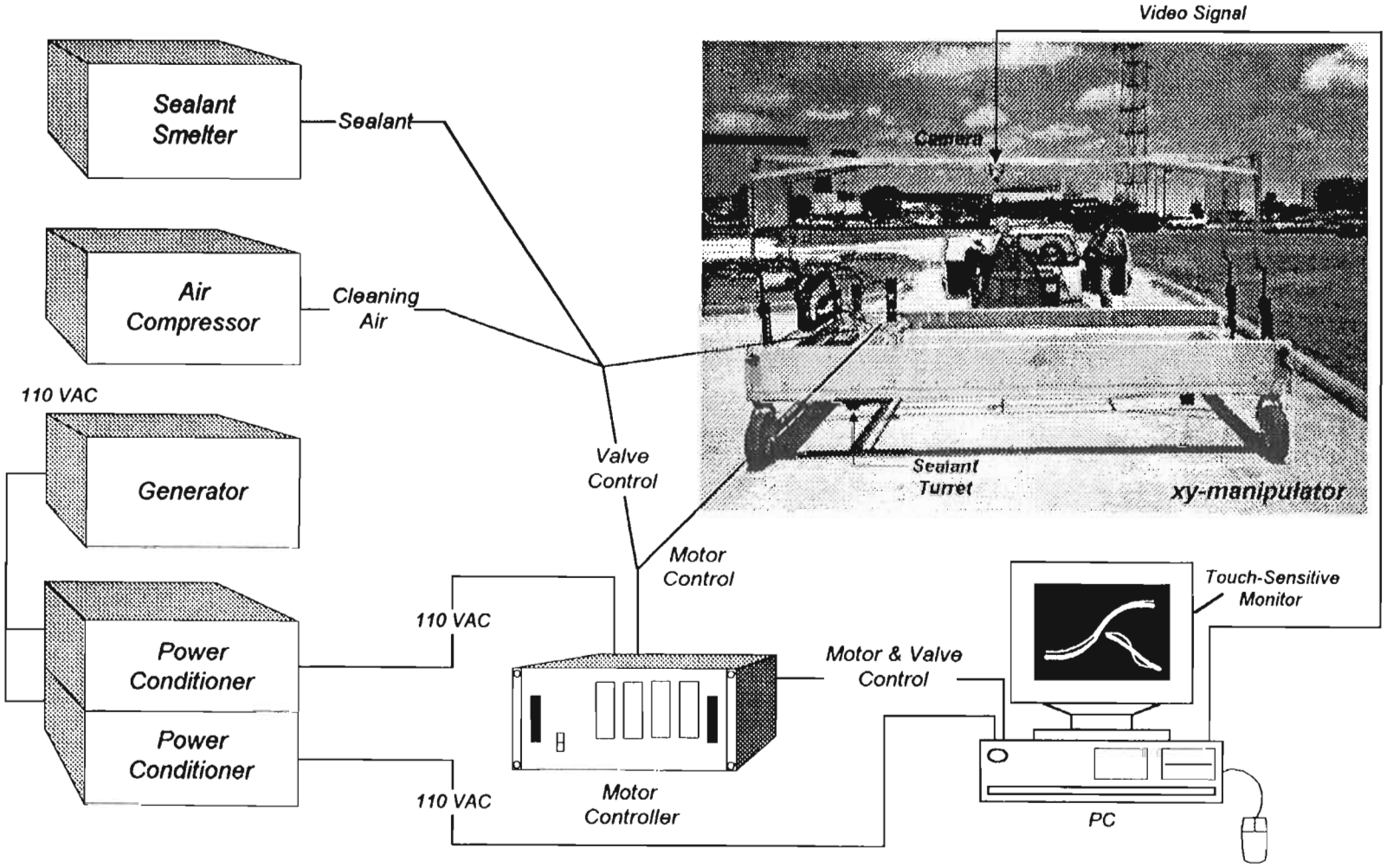


Figure 7.1 Physical Configuration of the ARMM System

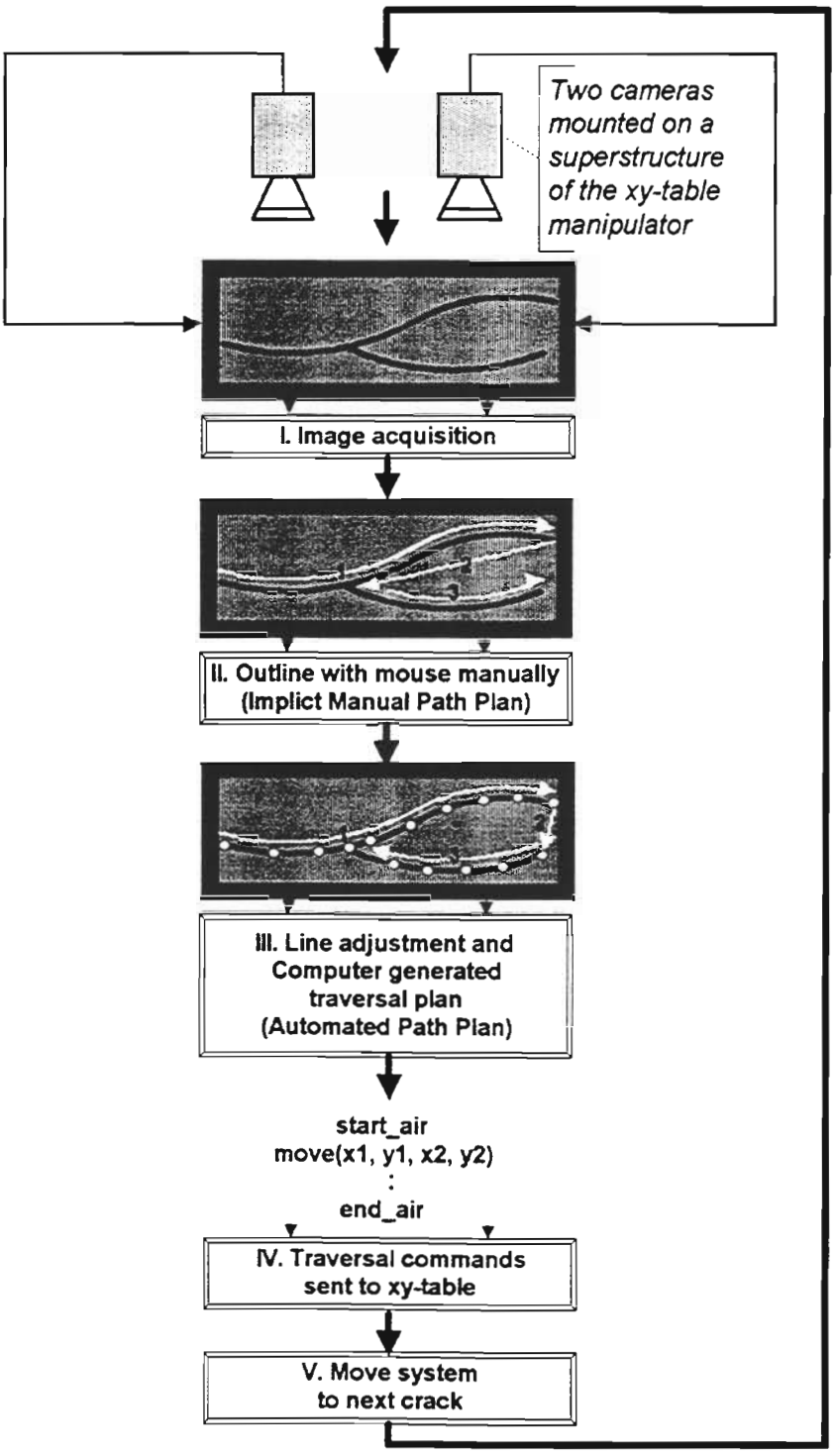


Figure 7.2 Machine Vision Assisted and Tele-operated Automatic Pavement Crack Sealing Architecture

Crack Detection and Mapping Process

To capture and process video image of the cracks in the machine's work space, a commercial image capture board is added to the PC. The DT3852 board manufactured by Data Translation allows the user to access the image data in the buffer to process it. Two commercial security cameras acquire live pavement surface images which are shown on a touch-sensitive video display. The system operator uses these images to manually locate the cracks by tracing over them on the video screen. An example of what the monitor displays is shown in Figure 7.3.

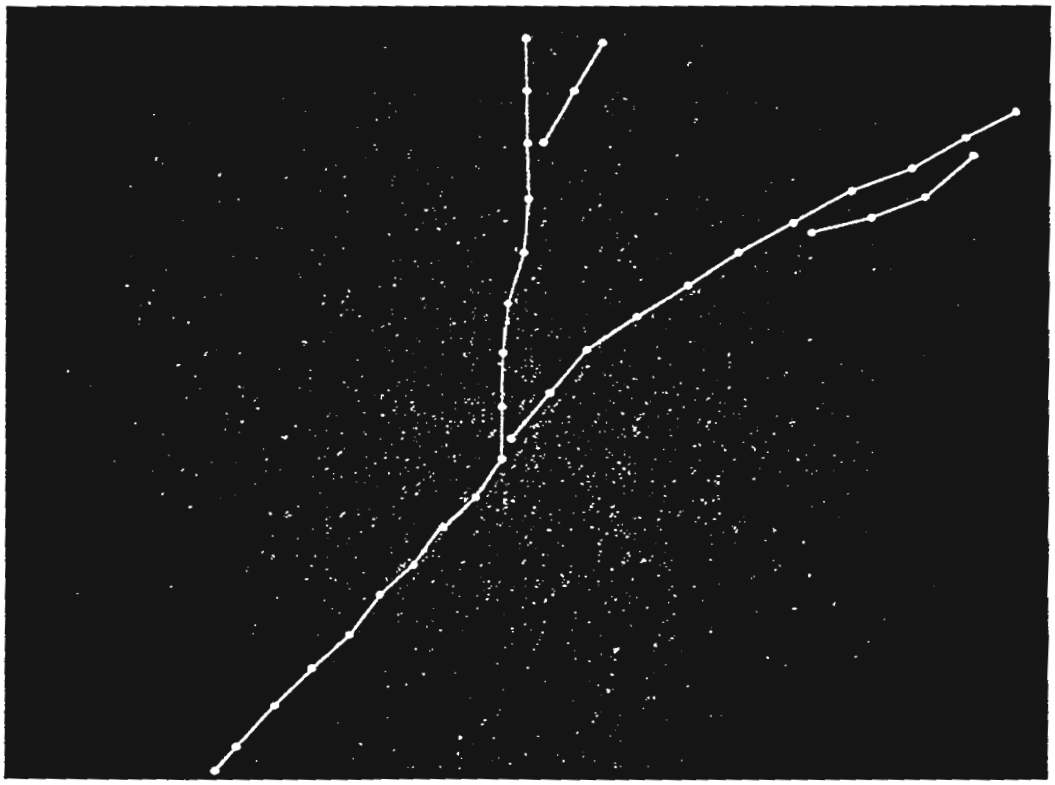


Figure 7.3 Crack Mapping

Connected line segments are drawn over the pavement image for each crack as it is traced to provide feedback to the operator as to the accuracy of his tracing (Greer 1996). Once this is done, an automated fine tuning routine, the Rotating & Bounding Box Algorithm, is started. It uses a small moving rectangular box to search an area perpendicular to each line segment for the middle of the crack. The Rotating & Bounding Box Algorithm moves the line segment to the middle of the box that was found to be the darkest (lowest average pixel value), since the cracks are dark (0's: Darkest and 255's: Brightest). Noise is compensated for since the average pixel value for the entire box is used. Once the fine tuning algorithm is complete, an update drawing of the line segments is performed to allow the operator to verify they represent the cracks that need to be sealed (Kim 1996). The obtained graph structure are then directly used for efficient traversal plans of the automated crack sealer. As the cracks are being sealed, live video updates provide visual feedback that the cracks are being sealed properly.

Path Planning and Machine Control

The path planning procedure can be divided into two major steps: (1) Path Planning using a Greedy Algorithm, (2) Generation of XY-Manipulator Control Command in order of the generated path. Chapter 8.0 and 9.0 of this report thoroughly explain the path planning methods.

The optimal motion planning of the automated pavement crack sealer is a very important task. Compared to conventional crack sealing operations, enhancing the operating speed in an automated crack sealer is a significant and critical factor for overall project success. Effective crack detection and mapping, generation of more efficient paths for crack filling, and optimal XY-manipulator control is one of the major tasks required to accomplish the desired operating speed. Generally, the following three major tasks are performed sequentially by crews in conventional crack sealing operations: (1) Identify the cracks to be filled on the road, (2) Blow clean and fill the cracks with compressed air and sealant, respectively and (3) Squeegee clean filled surfaces and cover with sand or other materials.

Generation of an efficient path for crack filling is related to Tasks 2 and 3 which take the most time in the crack sealing operation. Thus, minimizing operating time, by generating more efficient paths for an automated crack sealer, is a very important factor. The path planning software developed uses a greedy path planning algorithm in an effort to provide a feasible solution for paths in automated crack sealing. Then, the path generation results are used as an input data for the machine control.

7.3 Overall Strategy for the Tele-Operated ARMM

A revolutionary approach was used in an effort to provide more user friendly crack detection and sealing environment. Manually drawn lines on the video monitor were used to guide the crack sealer over the crack, however a lack of hand eye coordination introduces errors in this approach. To solve this problem, the developed system employed a remote, graphically controlled system with machine vision software to assist in centering the manually drawn lines along the crack spines. The machine vision system is able to quickly move the manually drawn lines closer to the actual crack locations. Two optional control strategies also have been experimented with during this project period. Those are: (1) Line adjustment and (2) Automated path planning. The automated road maintenance machine system also uses an xy table manipulator to move blowing, sealing and squeegee tools over the crack. It now has four major components: (1) a vision system for detecting and mapping cracks to be filled, (2) line adjustment software to adjust the lines closer to the actual crack locations using machine vision, (3) path planning software for generating a path that efficiently traverses the identified cracks, and (4) a motor and effector control system for the xy manipulator used for blowing, filling and squeegee the cracks.

8.0 Logic and Description of Path Planning Software

8.1. Background

In general, cracks must be abstracted into a representation that can be acted on by automated planning and machine vision algorithms. Cracks in pavement exhibit macroscopic morphological properties of connected and disconnected graphs, however in their digitized form at the pixel level they are highly noisy and unstructured. Options for converting from a digital image to a graph representation include manual input and bitmap-to-graph conversion after preprocessing. The first version of the path planning software developed in summer of 1995 included the bitmap-to-graph conversion process. Its data structure has been described in detail elsewhere (Kim 1995). This chapter illustrates the logic and description of the 2nd version of the path planning software that is recently developed. Although the 1st and 2nd version of the path planning software use basically same greedy algorithm, there have been significant changes in the data structures. In fact, the bitmap-to-graph conversion process is not necessary in the path planning process of the ARMM, since a graph structure for path planning can be constructed directly from the results of the operator input and the automated line snapping. Such crack networks are now represented as a disconnected graph . Each crack segment is thus represented as an edge with exactly two vertices without any branch. This makes the traversal plan of the ARMM much simpler. As a result, the 2nd version of the path planning software could substantially reduce its computational time and load by eliminating bitmap-to-graph conversion process from the path planning loop. Finally, next section describes in detail the logic and description of the developed code.

8.2. Logic and Description of the Code

The path planning software [2nd version] uses a greedy algorithm in which the end node point of a current component (Here, component means a crack line in a given image, and again, one component has only two node points, the start node point and the end node

point) seeks its closest node point that exists in other components for traversing next to be filled. First, to generate the shortest path in a given crack image, the program reads its crack data from a global array, snap_out[crack] which is storing the x-y coordinate pairs adjusted by the line snapping process using a localized crack detector. At the same time, the indices of all of the node points searched from the snap_out[ ] are stored in turn within an integer array, Node_Points[ ]. Then, a correct path order by the proposed greedy algorithm is determined through the distance comparison among the node points stored in Node_Points[ ] (Such searching process for the distance comparison is always started from home point(0,0)), and indices of Node_Points[ ] in the generated path order are then stored within an integer array, Visited_Order[ ]. Finally, all the x-y coordinate pairs in the generated path order are stored in a global array, path_out[ ]. The x-y manipulator control command is later created from this path_out[ ]. Figure 8.1 through 8.6 illustrates in detail the data structures of the developed path planning software. This program also includes the function which is for printing out 1)the path generation output, 2)the total traversed distance, and 3)the program execution time. This function is optional, and are used for some experimental purposes. More detailed description of the path planning algorithm is commented within the code.

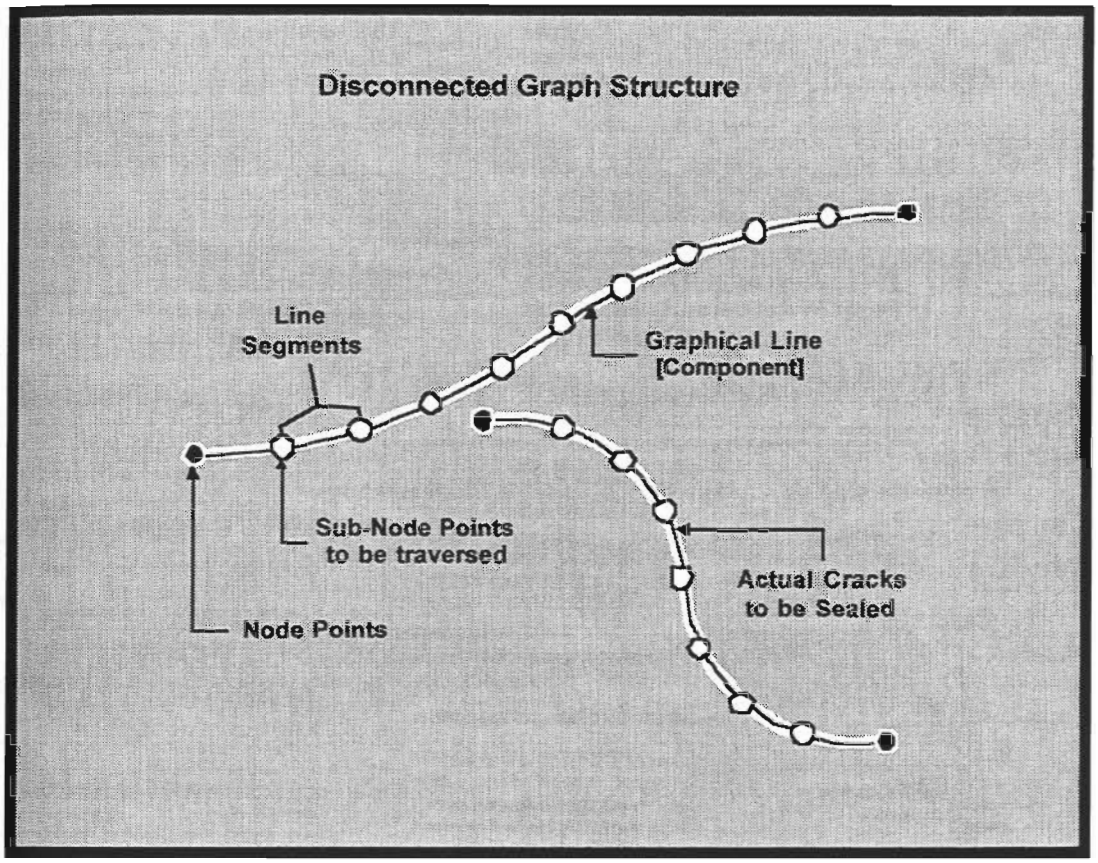


Figure 8.1. Disconnected Graph Representation for Path Planning

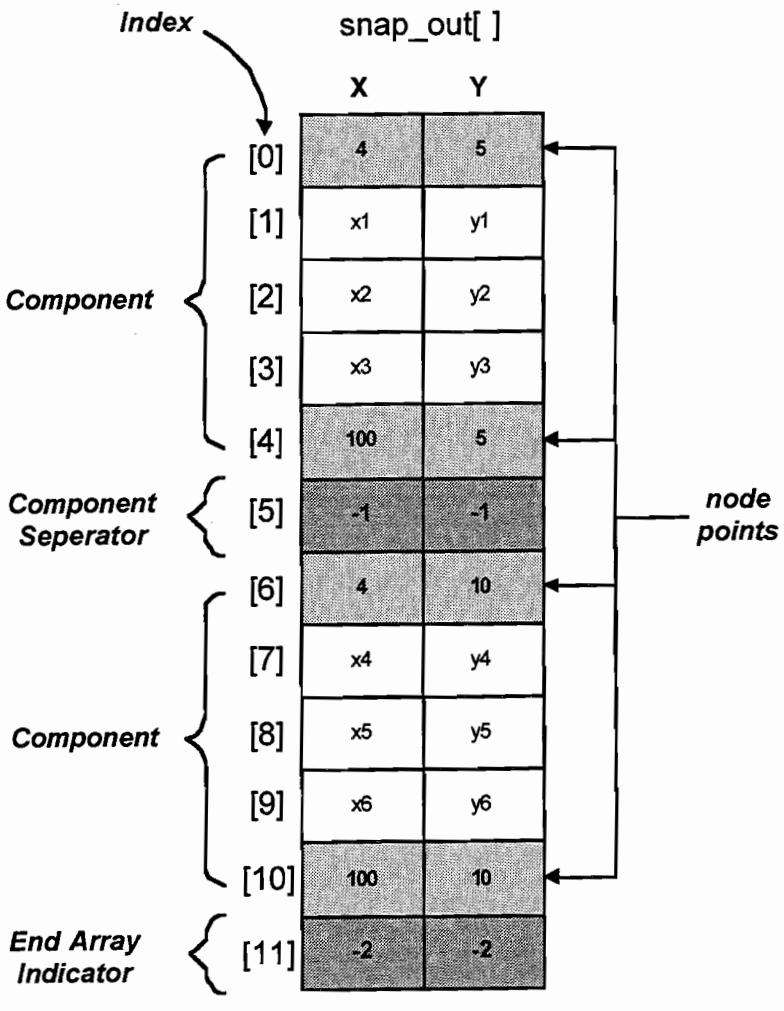


Figure 8.2. Summary of Terms Used

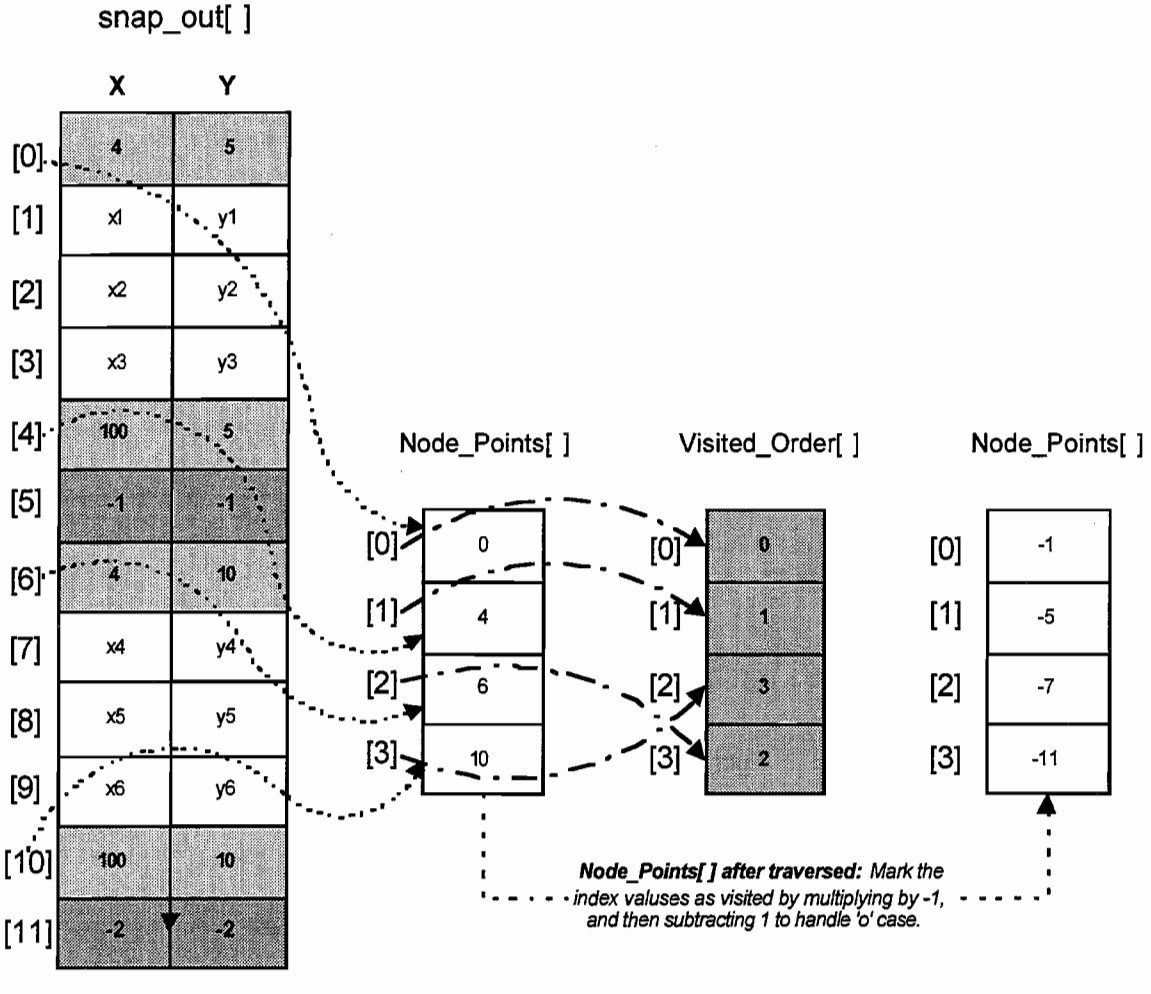
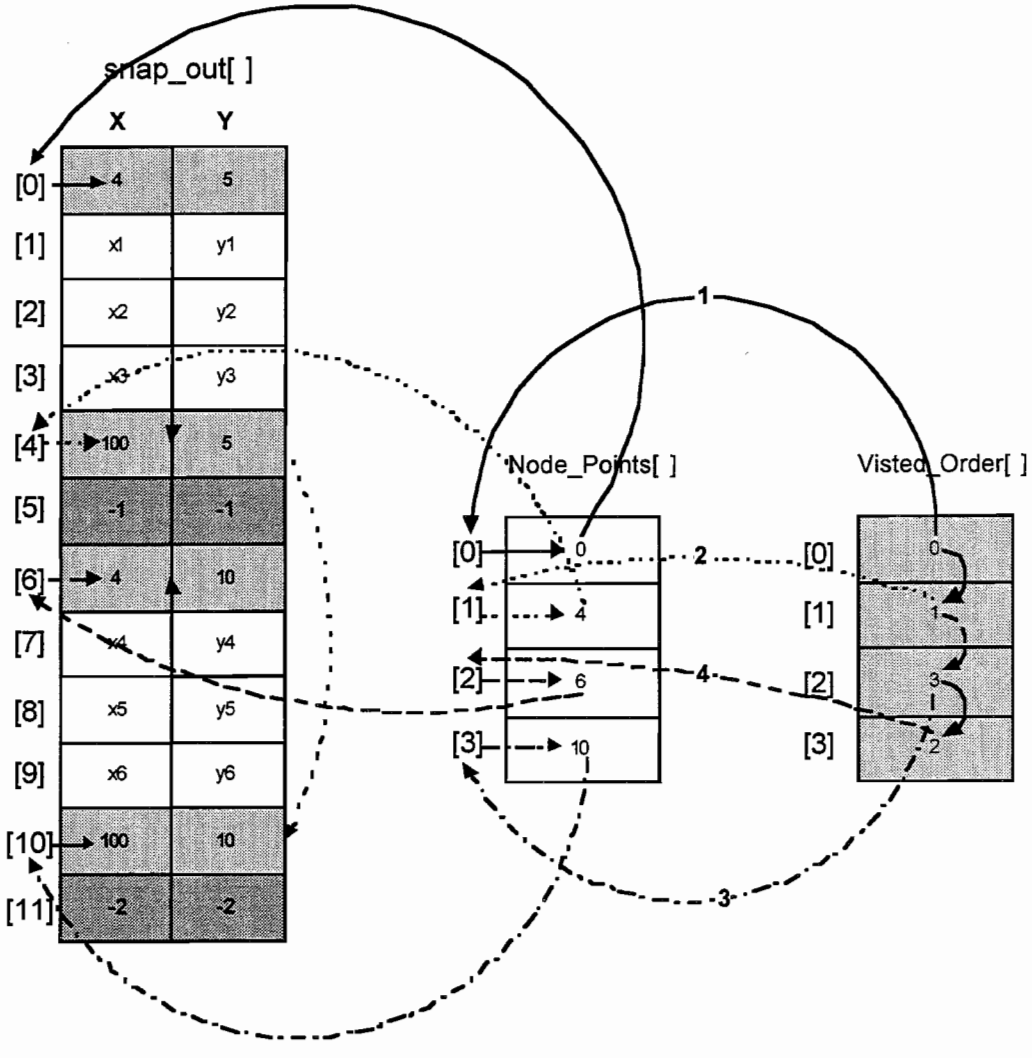


Figure 8.3. Data Flow Diagram for Path Generation

EASY TO ACCESS USING INDICES
IN CREATING path_out[ ]



Path Order [(4,5) ~ (100,5)] ---> [(100,10) ~ (4,10)]

Figure 8.4. path_out[ ] Creation Process using Indices

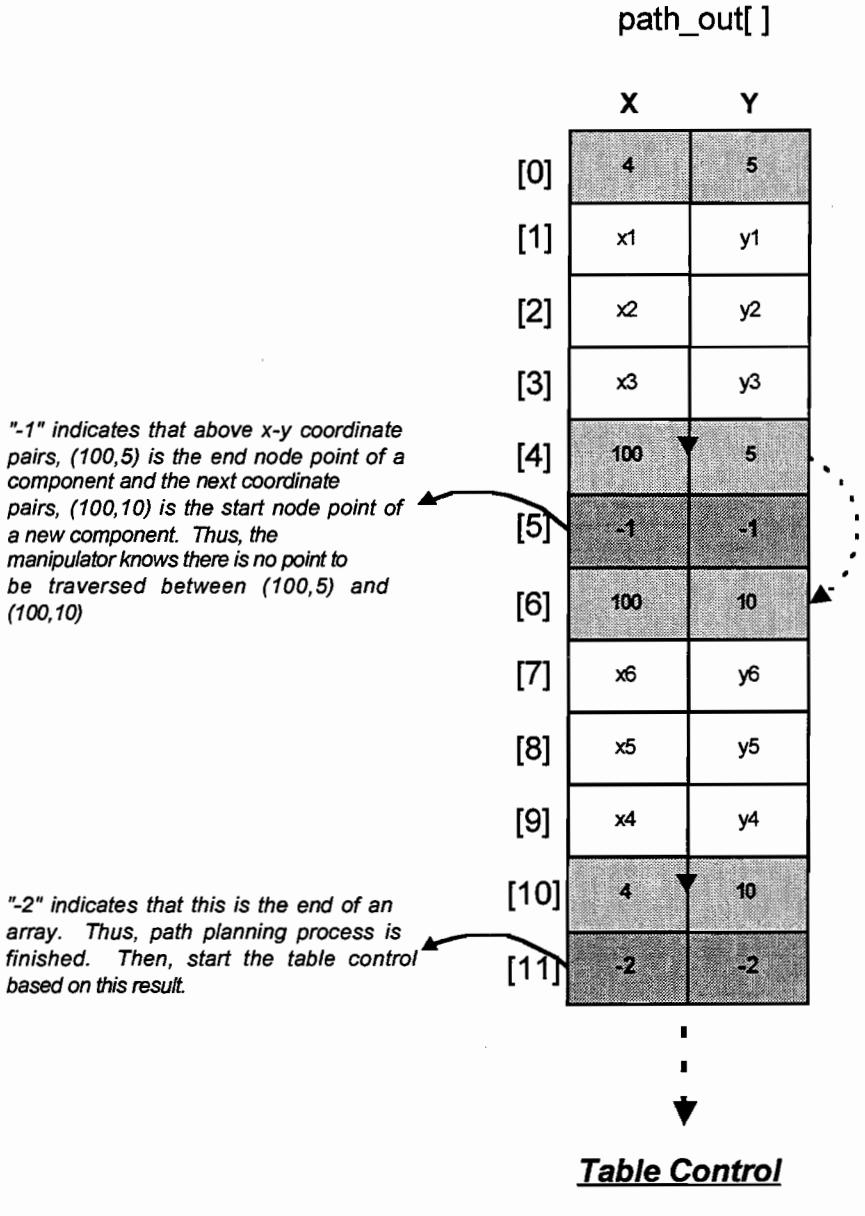
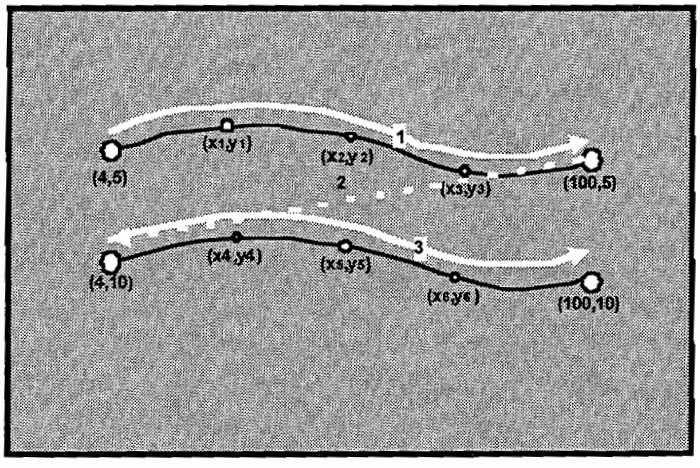
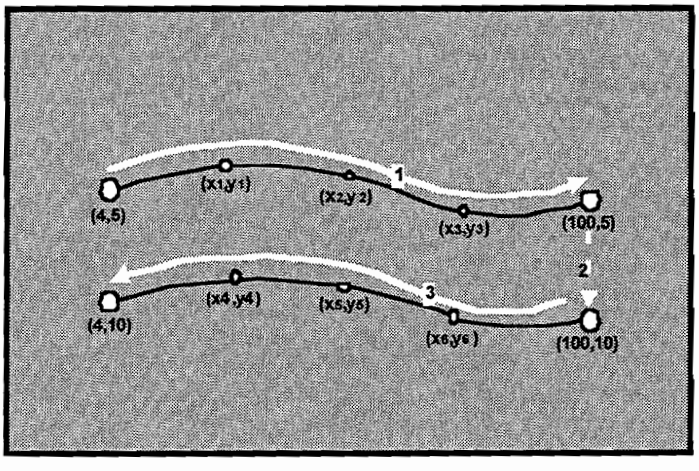


Figure 8.5. Path Generation Result



<Storage process of x-y coordinate pairs by the drawing order>
:stored in ***snap_out[ ]*** as [(4,5) ~ (100,5) ——> (4,10) ~ (100,10)]



<Storage process of x-y coordinate pairs by the path generation order >
:stored in ***path_out[ ]*** as [(4,5) ~ (100,5) ——> (100,10) ~ (4,10)]

Figure 8.6. Storage Process of x-y Coordinate Pairs in snap_out[] and path_out[]

More detailed description of the developed greedy path algorithm are shown in Chapter 9.0.

9.0 Path Planning for a Machine Vision Assisted, Tele-Operated Pavement Crack Sealer

9.1 Path Planning Algorithm For the Automated Road Maintenance Machine

There are many different types of crack shapes on the road, with occasionally complex morphologies. Possible paths can easily exceed 1 million (Haas 1994). Objectives include minimizing distance and value switching. Thus, it is very difficult to generate an optimal solution method that can be applied efficiently. Generation of optimal paths is outside the scope of this project, but should be performed in the future for benchmark comparisons. Presented is a greedy algorithm which was implemented and tested. The primary objective of the algorithm was to find a feasible solution that generates an efficient path for the automated crack sealer, guaranteeing traversal of all cracks in a work area.

9.2 Need for Greedy Path Generation

Generally, a greedy algorithm takes an action that seems the best at the given time without any consideration of future actions. The advantage of a greedy algorithm is that it can save the time which may be wasted by looking for future actions. The greedy algorithm has been applied to a wide variety of problems where achieving an optimal solution is computationally excessive. However, the essential problem of the greedy algorithm is that the proposed greedy method does not yield optimal solutions under all situations. Therefore, the greedy method is preferred in a case where an optimal solution is hard to be found, if all possible situations and problems are not considered and analyzed for a given problem, or where there is an obvious way to determine a feasible solution, but not necessarily an optimal solution. When this occurs, it can be more reasonable to establish an appropriate greedy algorithm for a given problem than to find its optimal solution, because it will obviously require too much time and effort. The best solution in this case is to seek a greedy algorithm which is capable of providing a near optimal solution, and

sometimes an optimal solution. Surprisingly, it has been proven that greedy algorithms have yielded optimal results in many circumstances.

9.3 Automated Path Planning Process

The path planning software employed a greedy algorithm in an effort to provide a feasible solution for paths in automated crack sealing. The primary objective of a greedy algorithm is to find a feasible solution that generates an efficient path, ultimately the shortest path for automated crack sealing, while guaranteeing traversal of any kind of crack morphologies in a given crack scene. It is anticipated that the algorithm described in this report may eventually have broader applications in infrastructure maintenance and in welding operations. Its simple algorithm is described as follows:

- Seek the closest vertex on an active list and stop if the active list is empty, otherwise
- Traverse the link between the vertices, and if it is a crack edge, then remove it from the active list.

The graph which has already been constructed through the graph conversion process is used as an input data for the path planning process. In other words, the graph representation is considered the active list. In the path planning software, the program sets the closest vertex from the home point (0,0) as a start vertex for actual path generation. So, the current pointer moves from the home point to the start vertex. Then the adjacent vertex is sought. From the adjacent vertex the next closest vertex is sought, and the process is repeated (Figure 9.1). Figures 9.2 and 9.3 graphically contrast implicit and automated path planning. Figure 9.3 illustrates the potential advantage of automated path planning.

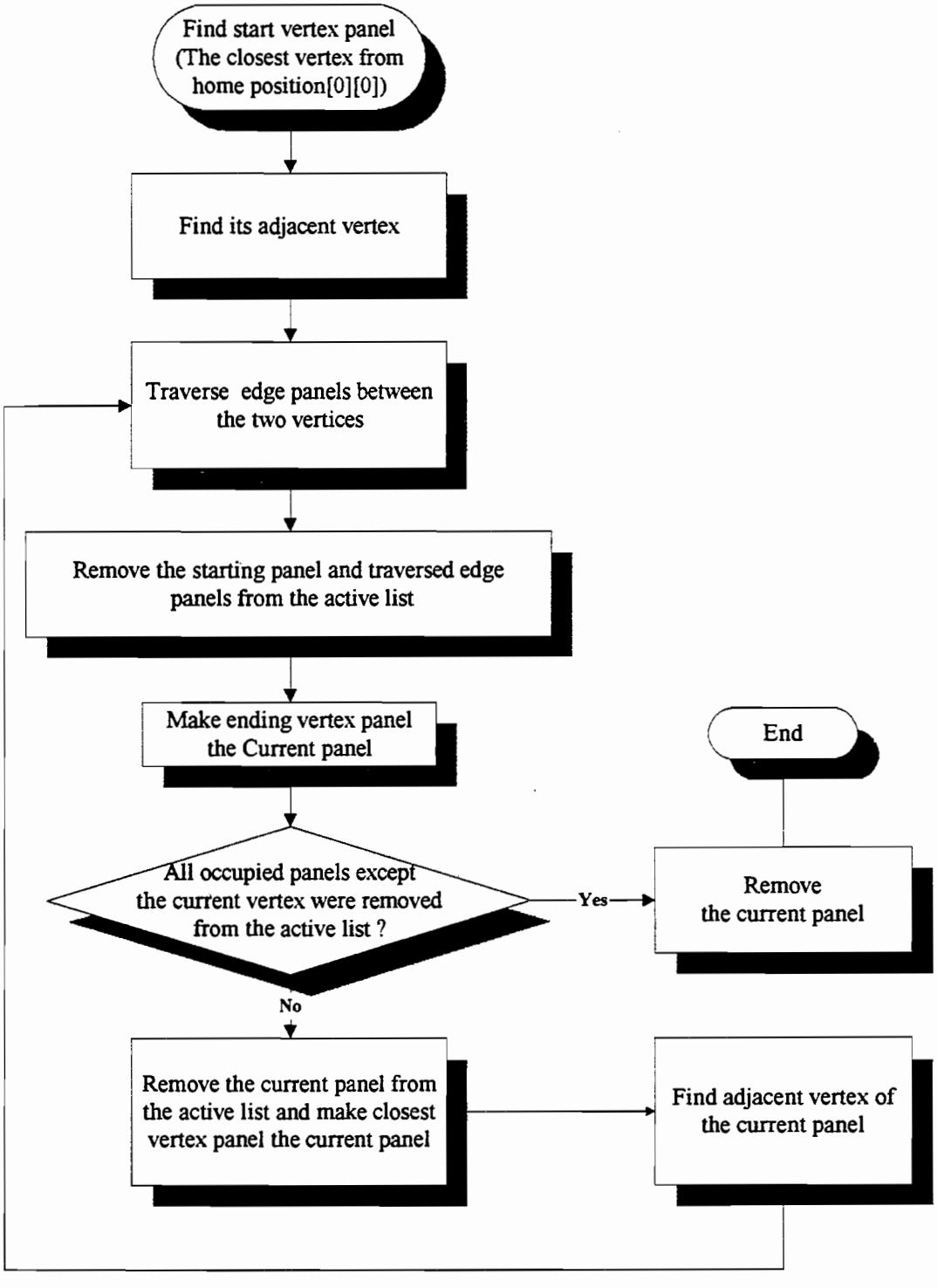
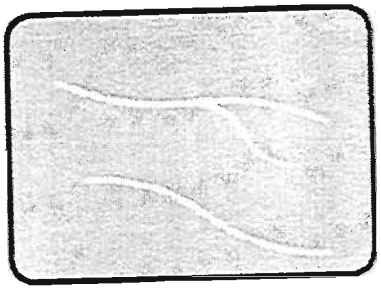
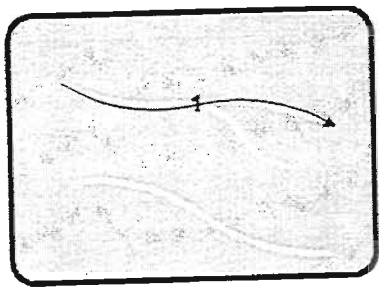


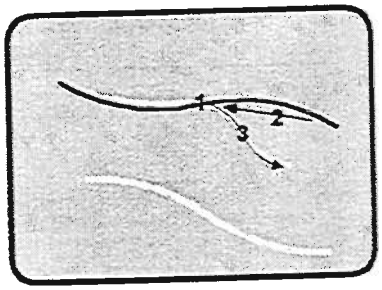
Figure 9.1 Flow Chart of Path Planning Process for Disconnected Graph



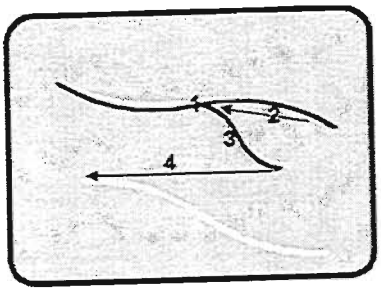
A Crack Image on
Computer Screen



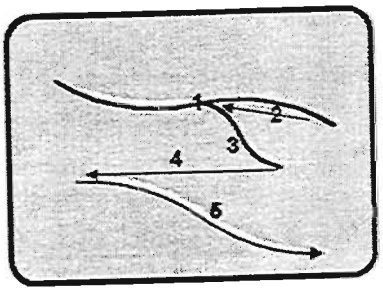
(I) Implicit Path
Generation by Manual Input



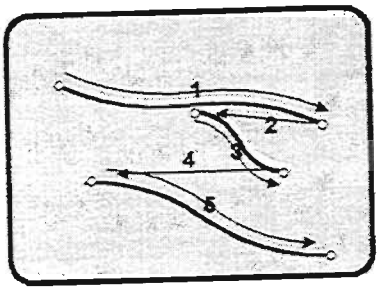
(II)



(III)

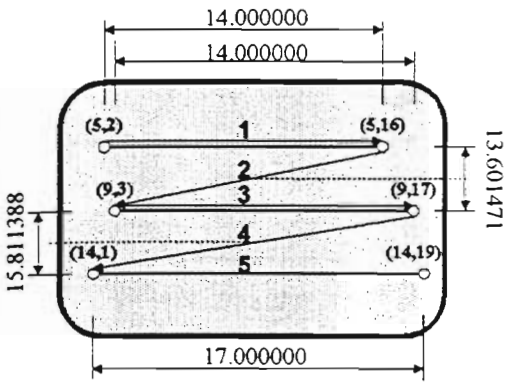


(IV)

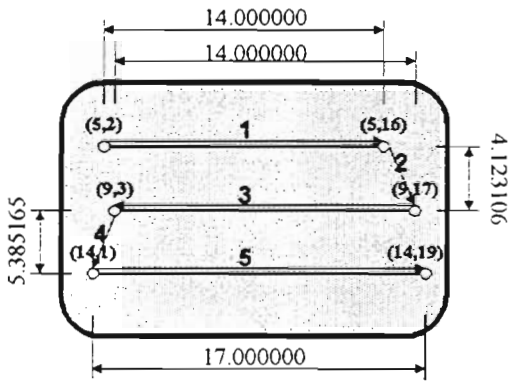


(V) Implicit Path
Planning Result

Figure 9.2 An Example of Implicit Path Planning using Mouse



(I) Implicit Path Planning Method



(I) Greedy Path Planning Method

	Total Traversed Distance (1+2+3+4+5)	Edge Length of Crack (1+3+5)	Length for Moving Table without Tracking Edge Panels (2+4)
(I)	74.412859	45.000000	29.412859
(II)	54.508271	45.000000	9.508271

Unit: Pixels

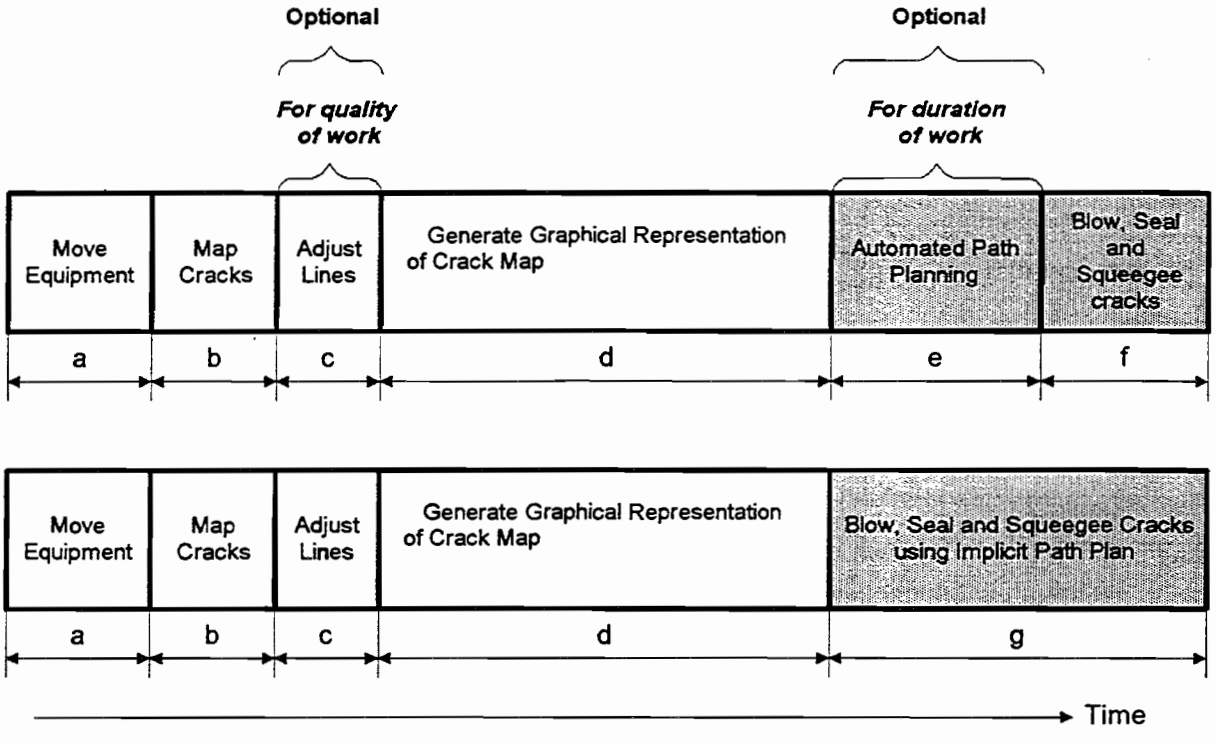
Figure 9.3 An Example of Efficiency Comparison using Total Traversed Distance

9.4 Efficiency Experiments

Automated Path Plan vs. Implicit Manual Path Plan

A study was conducted to compare the efficiency of the automated greedy path plans with the implicit path plans. Here, implicit path plan is defined as the drawing sequence of the operator when he is not explicitly trying to generate an optimal path. The current automated crack sealer can be manipulated by both automated path plans using the proposed greedy algorithm and implicit manual path plan. To describe the trade-off between the automated path plans and implicit manual path plans, figure 9.4 is included in

this chapter. The upper time line in the figure 9.4 shows that the automated path plan is an optional function in the automated crack sealing process.



If **e + f < g** then automated path planning is a net benefit to the process

Figure 9.4 Time Sequence and Constraints in the Tele-operation Process Flow

To verify that the automated path plan is feasible for the automated crack sealing, both approaches were computationally compared in terms of time and distance. That is;

If the processing time and distance of 'e' and 'f' is less than those of 'g' (Te+Tf < Tg) Then the automated path plan will be an obvious net benefit to the automated crack sealing process.

Computational Efficiency Comparisons

A survey was conducted in order to compare the computational efficiency in both approaches. For this survey, twenty real crack images from the vision software were prepared and distributed to each of five participants in this survey. An example test image is presented in Figure 9.5. The five respondents consisted of project members involved in the automated crack sealer project who understand the objectives and procedures of this survey.

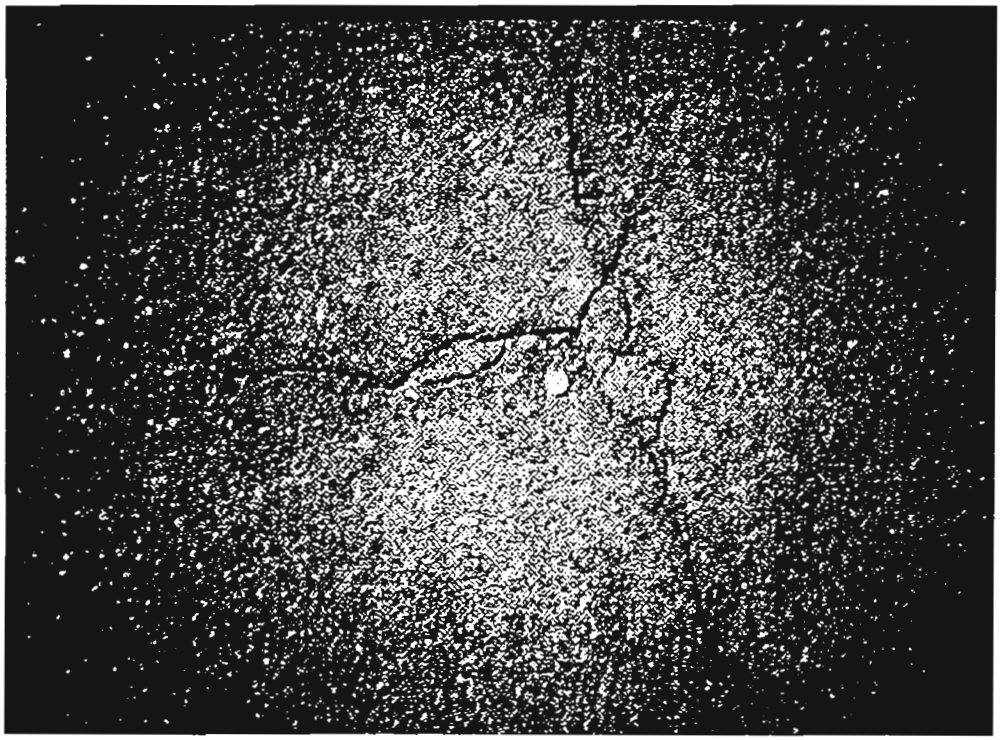


Figure 9.5 Example Test Image

From this survey, it was revealed that a system operator who draws the lines over a crack image using a mouse usually generates the path of a crack from the left-hand side toward the right-hand side. Generally, the system operators drag the mouse over a crack image

based on their intuition. Thus, there can be several path generation solutions by each different user in this approach. That is, the difference between both approaches is in the path information generated for the x-y manipulator. The developed path planning software automatically provides users with information of the total traversed distance and the software running time required to generate the path for the automated crack sealer as well as the each edge length of the tested crack morphology. The length (idle length) required for moving only the machine without edge tracking in the implicit path planning method can be simply obtained through a Cartesian coordinate calculation. As a result, the total traversed distance of the x-y manipulator for each crack image in both approaches can now be compared computationally. From this information, the total traversal time required for the x-y manipulator in each approach can also be estimated and compared as Figure 9.3.

The result of the efficiency comparison revealed through the software testing and the implicit path planning survey for twenty crack morphologies from vision software are summarized in Table 9.1.

Table 9.1 The Result of Efficiency Comparison between the Automated Greedy Path Planning Method and the Implicit Path Planning Method

	Proposed Method	**Total Traversed Distance (Idle Length + Edge Length of Crack)**
TOTAL	**Automated**	12,225 Pixels / 20 Crack Images
	Implicit	14,300 Pixels / 20 Crack Images
AVERAGE.	**Automated**	611.26 Pixels / Crack Image
	Implicit	715.01 Pixels / Crack Image

The results report the total traversed distance of the x-y manipulator required for filling all of the twenty tested crack images by the greedy method to be 12,225 pixels, while 14,300

pixels were required for the x-y manipulator in the implicit path planning method. For a better comparison of the total traversed distances mentioned above, this research attempted to translate the total number of pixels traversed by the x-y table in each approach to centimeters. Relatively small 200 x 200 array size was utilized for the experimental purpose. Figure 9.6 briefly describes the conversion process and the approximate work area of the automatic crack sealer that was assumed for this calibration process.

Based on the results of the calibration process, the total traversed distance required to fill all of the tested crack images in each approach is converted to centimeter as follows:

- Total Traversed Distance / 20 Crack Images

Auto. :	12,225 Pixels x 1.25 cm	= 15,281 cm = 152.81 m
Implicit:	14,300 Pixels x 1.25 cm	= 17,875 cm = 178.75 m
Difference:	(17,875 - 15,281) cm	= 2,593.7 cm = 25.94 m

- Traversed Distance / Crack Image

Auto.:	611.26 Pixels x 1.25 cm	= 764.08 cm
Implicit:	715.01 Pixels x 1.25 cm	= 893.76 cm
Difference:	(893.76 - 764.08) cm	= 129.7 cm = 1.30 m

The difference in the total traversal time it would take the manipulator to fill all twenty of the crack images is estimated for both approaches (See Table 9.2) based on the above results. This is easily calculated using the velocity (17.28 cm / second) of the automatic crack sealer and the difference in the total traversed distances for each method:

Velocity of the automatic crack sealer = 17.28 cm / second

Difference in the total traversed distance of each approach = 2,593.7 cm

Therefore,

$$\Delta T = \frac{2{,}593.7 \text{ cm} / 20 \text{ images}}{17.28 \text{ cm} / \text{second}} = 150 \text{ seconds} / 20 \text{ images} = 7.5 \text{ seconds} / \text{image}$$

The average time to complete a bitmap-to-graph conversion was 0.67 seconds per image using a 486 PC, whereas the time required to plan paths was about 0.12 per image. This time has since been reduced significantly. In reference to Figure 9.4, it is clear that the time required to automatically compute the path plan is much less than the time saved and thus well worth the effort.

Table 9.2 Result of Efficiency Comparison for 20 Crack Scenes

	Total Traversed Distance	Total Traversal Time
Implicit	17,875 cm	18 Min.
Automated	15,281 cm	15 Min.
Difference	**2,593.7 cm = 25.94 m / 20 Images**	**3 Min. = 180 Seconds / 20 Images = 9 seconds / Image**

9.5 Conclusions

In conclusion, the advantage of computing a path plan rather than relying on implicit path plans is clear. Recent field trials of the full scale crack sealer appear to support this conclusion as well. While completely autonomous and reasonably accurate crack recognition and mapping have been demonstrated in previous studies to be technically possible (Haas 1990, Haas 1996, Gharpuray 1993), until new solutions or technologies are

available, complete autonomy is undesirable for automated crack sealing in the field. Humans are superb at picking signals and patterns out of a noisy background, so allowing the operator to point out the existence and location of a pavement crack uses human abilities to advantage. Humans are not as good at making numerous calculations, so using the computer to compute a short path uses its capabilities to advantage. Using machine vision for line snapping can also compensate for imperfect human hand-eye coordination. Experiments with the crack sealer have determined these functional balances for economic field operation.

10.0 Line Snapping Software

10.1 Crack Detection and Mapping

The crack detection and mapping process of the automated crack sealer can be divided into several steps. To capture and process video image of the cracks to be sealed in the machine's work space, a commercial image capture board is added to the PC. The DT3852 board manufactured by Data Translation allows the user to access the image data in the buffer to process it. Two commercial security cameras acquire live pavement surface images which are shown on a video screen. The system operator uses these images to manually locate the cracks by tracing over them on the video screen. Connected line segments are drawn over the pavement image for each crack as it is traced to provide feedback to the operator as to the accuracy of his/her tracing. Then, the connected line segments are stored in an array of vision software which is called out[]. Once this is done, an automated line snapping algorithm is started.

10.2 Automated Line Snapping Algorithm

The line snapping algorithm which is called "Rotating & Bounding Box Algorithm (RBBA)" uses the connected line segments that are traced by the system operator and then are stored in out[] of vision software. The main purpose of this program is to improve the approximation of the user-input points to be closer to the actual crack lines to be sealed by bounding a small rectangular box along the normal between two points, a connected line segment. That is, the RBBA uses a small moving rectangular box to search an area perpendicular to each line segment for the middle of the crack. The search range of the bounding box is $\pm$ 10 pixels from the position of each line segment, so that twenty one boxes are created to identify the accurate crack locations to be sealed. Then, the bounding boxes get the gray level values of all the pixels within each box from the buffer containing the picture of the cracks and find the best box by comparing the total gray level values of each box. Finally, it moves the line segment to the middle of the box that was found to be

the darkest (lowest average pixel value), since the cracks are dark (0's: Darkest and 255's: Brightest).

The width and height of the buffer used is 640 x 480, respectively. It is anticipated that noise can be compensated for since the average pixel value for the entire box is used. Once the RBBA is completed, an update drawing of the line segment is performed to allow the operator to verify they represent the cracks that need to be sealed. This software documentation will present more detailed description of the RBBA proposed. Below Figure 10.1 briefly describes the RBBA.

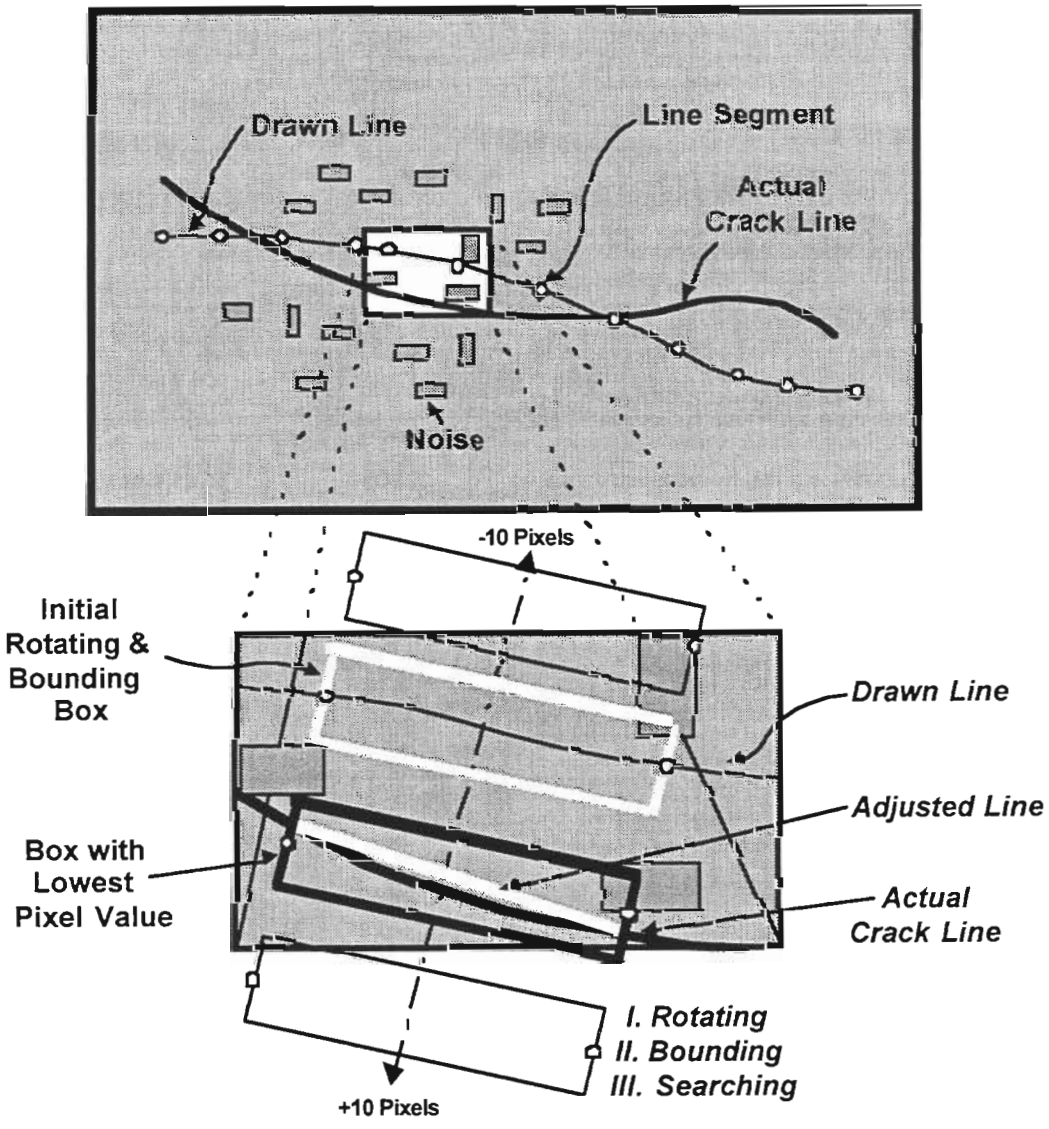


Figure 10.1 Illustration of Rotating & Bounding Box Algorithm

10.3 Logic and Description of Code

The automated line snapping process is divided into five major steps as follows:

I. Creating 21 bounding boxes along the normal of each line segment.

II. Getting the gray level values for all pixels that the created each bounding box contains.

III. Finding the best box by comparing the total gray level values of each box.

IV. Creating an array which is called snap_out[].

V. Moving the line segment to the middle of the best box.

The snap_out[] created by the line snapping software is then used as an input data for the path planning software developed for the efficient path generation of automated crack sealer. This software documentation will graphically describe the details of the RBBA using a simple crack example presented in Figure 10.2.

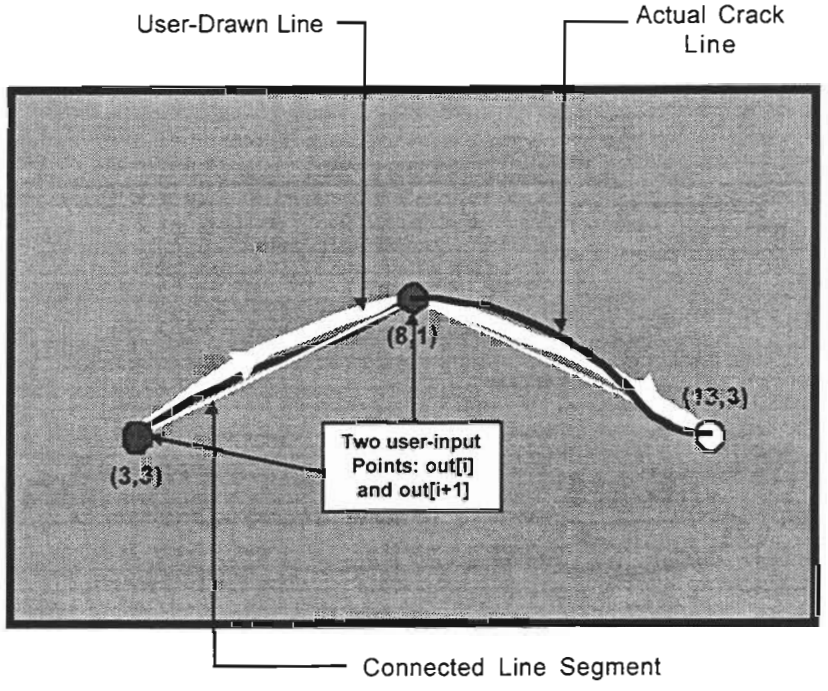


Figure 10.2 Example of a Simple Crack

Functions for Automated Line Snapping

Table 10.1 Automated Line Snapping Functions and Their Descriptions

Function	Description
line_snap	Take out[i] and out[i+1] to create a box: main function
Find_Best_Bounding_Box	Find the best box by comparing the total gray level values
Determine_Angles	Calculate the parallel and normal along the x and y axis
Create_Bounding_Box	Calculate four corner points of each box to be created
Value_of_Box	Return total gray value using "Value_of_Detect_Pixs" after ordering the 4 corner points
Order_The_Points	Order four corner points of each box in a desired sequence
swap_pts	Automatically swap 4 corner points after ordering
Box_Bounds	Detect the pixels for 4 boundaries of each box
Value_of_Detect_Pixs	Detect all pixels within boundary detectors and get total pixel value of each box
get_value	Get gray level value of each pixel from buffer
Put_Bounding_In_Snapout	Create snap_out[] for efficient path generation of ARMM

line_snap:

This is the main function of the line snapping software developed. First, "line_snap" takes two user-input points which are shown on a video screen to create a small moving rectangular box between the two points and tries to correct any larger error by the system operator. Thus, this function searches two user-input points, out[i] and out[i+1], from out[] of vision software. Then, it improves the approximation of the user input to be closer to the actual crack locations to be sealed. This function also calculates time (starting and ending time) required for the line snapping process in an effort to measure its computational efficiency.

Find_Best_Bounding_Box:

The objective of this function is to calculate and add up the gray level values for all pixels of each bounding box from 10 pixels one way to 10 pixels the other way along the normal of the given line segment. Then, it compares the total gray level values of each bounding box and selects the best box that is found to be the darkest. Here, the dark_values in this function is the sum for the gray level values of all pixels that each bounding box holds.

Determine_Angles :

When two points that define the beginning and ending points of a line segment are given from out[] of vision software, this function calculates the parallel and normal along the x and y axis. These indicate the relative x and y values of parallel and normal to the crack (based on a hypotenuse of length = 1.0). Figure 10.3 uses the example in Figure 10.2 to demonstrate this function.

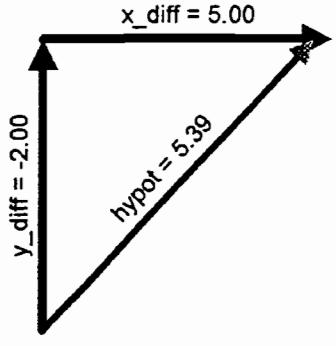
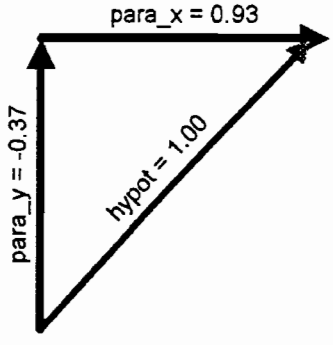
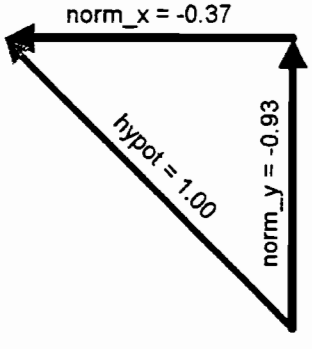
x_diff = 8-3 = 5.00 y_diff = 1-3 = -2.00 hypot = sqrt(25+4) = 5.39	para_x = 8-3 = 5.00 para_y = 1-3 = -2.00 hypot = 1.00
	 * Calculate the parallel to the box. Then, the hypotenuse is normalized to 1.00
norm_x = -0.37 norm_y = -0.93	
 * This figure indicates when a small rectangular box bounds along the normal of the line segment, ([3,3], [8,1]), x and y axis is decreased about 0.37 and 0.93 pixel, respectively.	

Figure 10.3 Determine_Angles Function

Create_Bounding_Box:

When two user-input points that indicate a section of a crack and the perpendicular in the x and y directions normalized to 1.0 by "Determine_Angles" are given, "Create_Bounding_Box" calculates four corner points of each bounding box. The height of the bounding box to be created is fixed to 5 pixels (± 2.5 pixel from the middle of the box) but its width depends on the length of the given line segment. Thus:

```
bounding_box[0].x = (int) (x_norm * (-2.5 + offset_x) + one.x);     pt1.x = [7]
bounding_box[0].y = (int) (y_norm * (-2.5 + offset_y) + one.y);     pt1.y = [14]
bounding_box[1].x = (int) (x_norm * (-2.5 + offset_x) + two.x);     pt2.x = [12]
bounding_box[1].y = (int) (y_norm * (-2.5 + offset_y) + two.y);     pt2.y = [12]
bounding_box[2].x = (int) (x_norm * (2.5 + offset_x) + one.x);      pt3.x = [5]
bounding_box[2].y = (int) (y_norm * (2.5 + offset_y) + one.y);      pt3.y = [9]
bounding_box[3].x = (int) (x_norm * (2.5 + offset_x) + two.x);      pt4.x = [10]
bounding_box[3].y = (int) (y_norm * (2.5 + offset_y) + two.y);      pt4.y = [7]
```

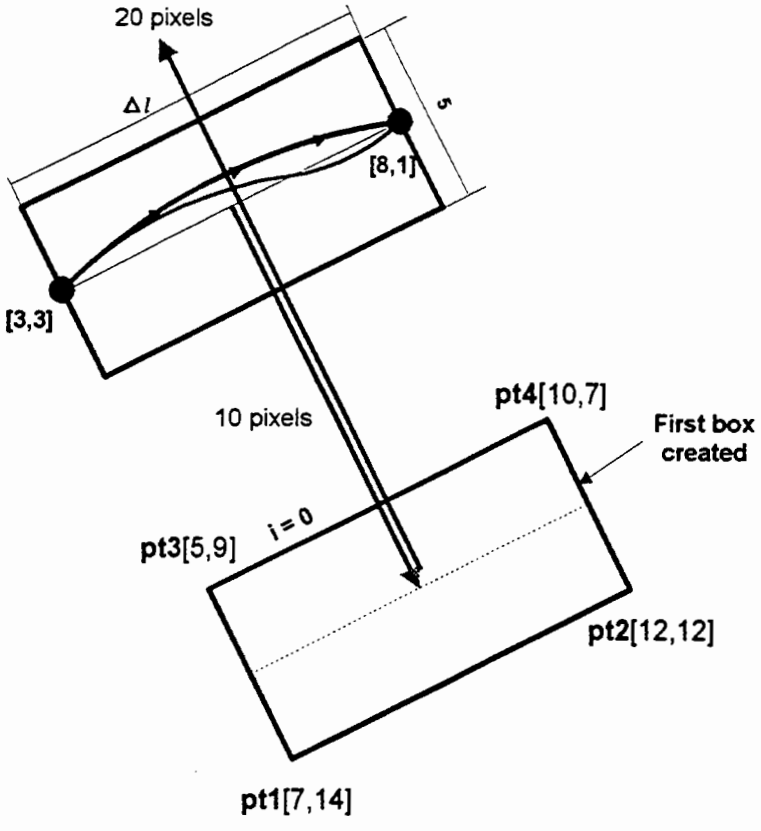


Figure 10.4 Create_Bounding_Box Function

Value_of_Box:

This function uses "Value_of_Detect_Pixs" after ordering the four corner points (pt1, pt2, pt3, pt4) in a standard way to determine the total gray level value for all the detected pixels in a given bounding box.

Order_The_Points / swap_pts:

In every bounding box, pt1 should be above pt3 and pt2. Also, pt2 and pt3 should be above pt4. The detection for four boundaries (sides) of each bounding box is always accomplished in order of pt1 -> pt2 -> py4 and pt1 -> pt3 -> pt4. This is described in

more detail in "Box_Bounds." Thus, the four corner points of each bounding box is automatically swapped in the desired order by the "Order_The_Points" and swap_pts."

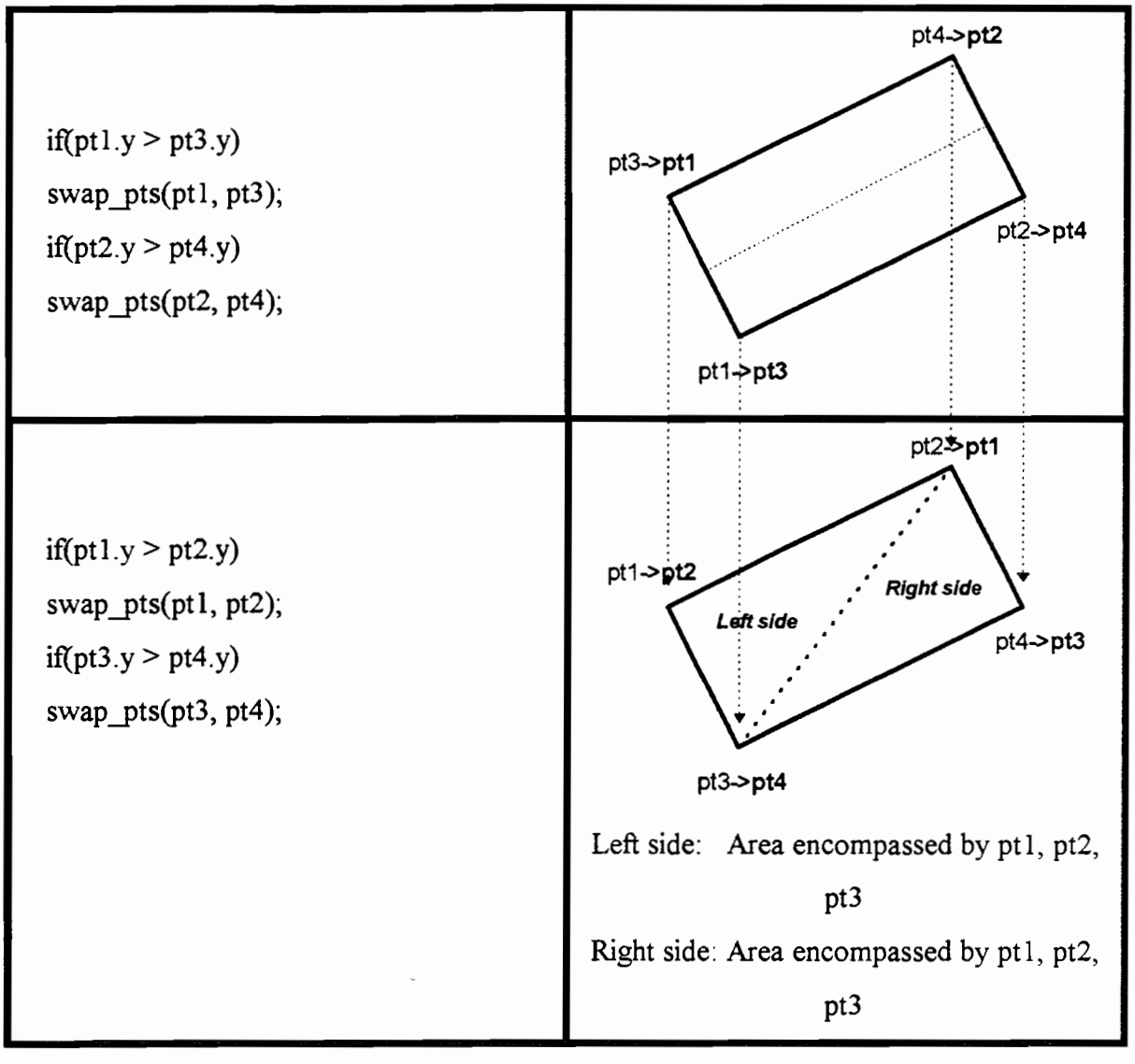


Figure 10.5 Order_the_Points and Swap_Pts Functions

When a system operator traces a crack image shown on a video screen using a mouse, he/she usually drags the mouse from left-hand side toward right-hand side. However, it is also possible for the user to trace the crack image to be sealed from right-hand side toward left-hand side. The RBBA can consistently handle both cases in a little different

manner but uses exactly same algorithm in both cases. Only difference in two cases is in the creation order of the bounding box. The difference in box creation order between two cases can be made by comparing Figure 10.4 and Figure 10.5, and Figure 10.6.

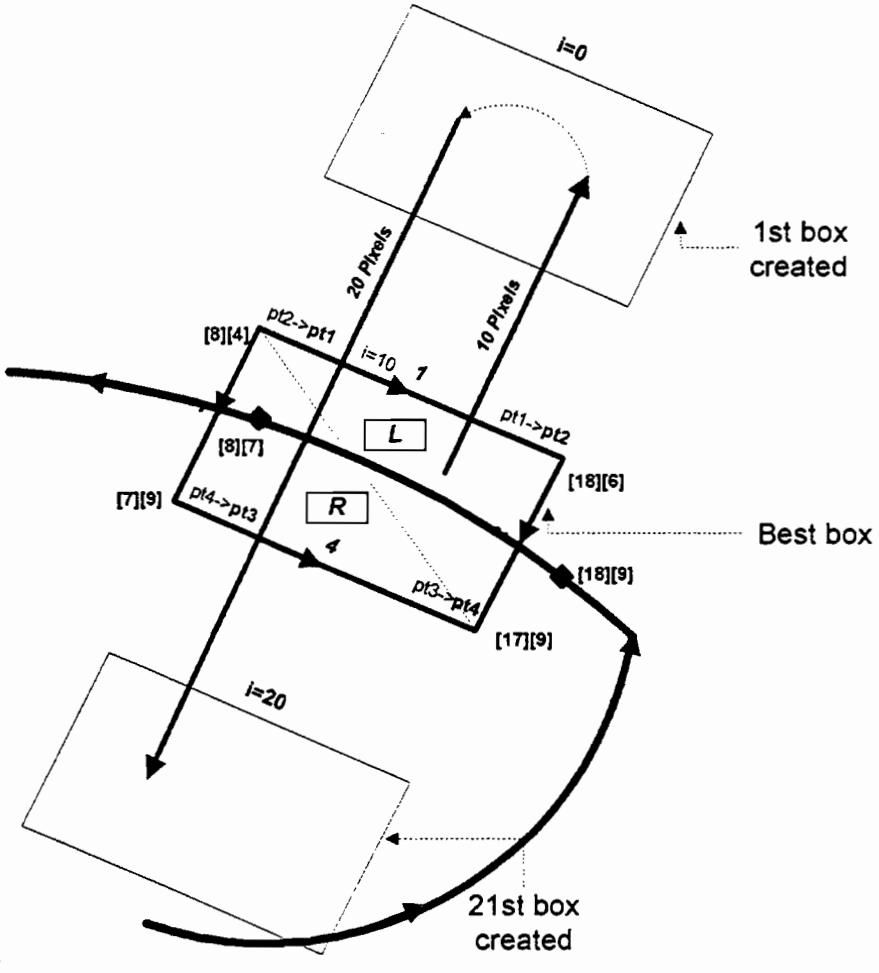


Figure 10.6 Difference in Box Creation Order

Box_Bounds:

"Box_Bounds" calculates the x-y coordinate pairs for the left and right side of each bounding box which are determined by "Order_The_Points" and "swap_pts". That is, this function detects the x-y coordinate pairs that represent the boundaries (four sides) of each bounding box to be examined. As a result, it allows "Value_of_Detect_Pixs" to be able to determine the total pixel values within the bounding box through the scan from one side to the other side. The boundary detection process is always performed from pt1 -> pt2 -> pt4 (Left side) to pt1 -> pt3 -> pt4 (Right side). Here, the 'length' indicates the number of pixels detected in each side. Then, the detected pixels and 'length' are used as an important information for "Value_of_Detect_Pixs." Below Figure 10.7 illustrates the boundary detection process using the example presented in this software documentation.

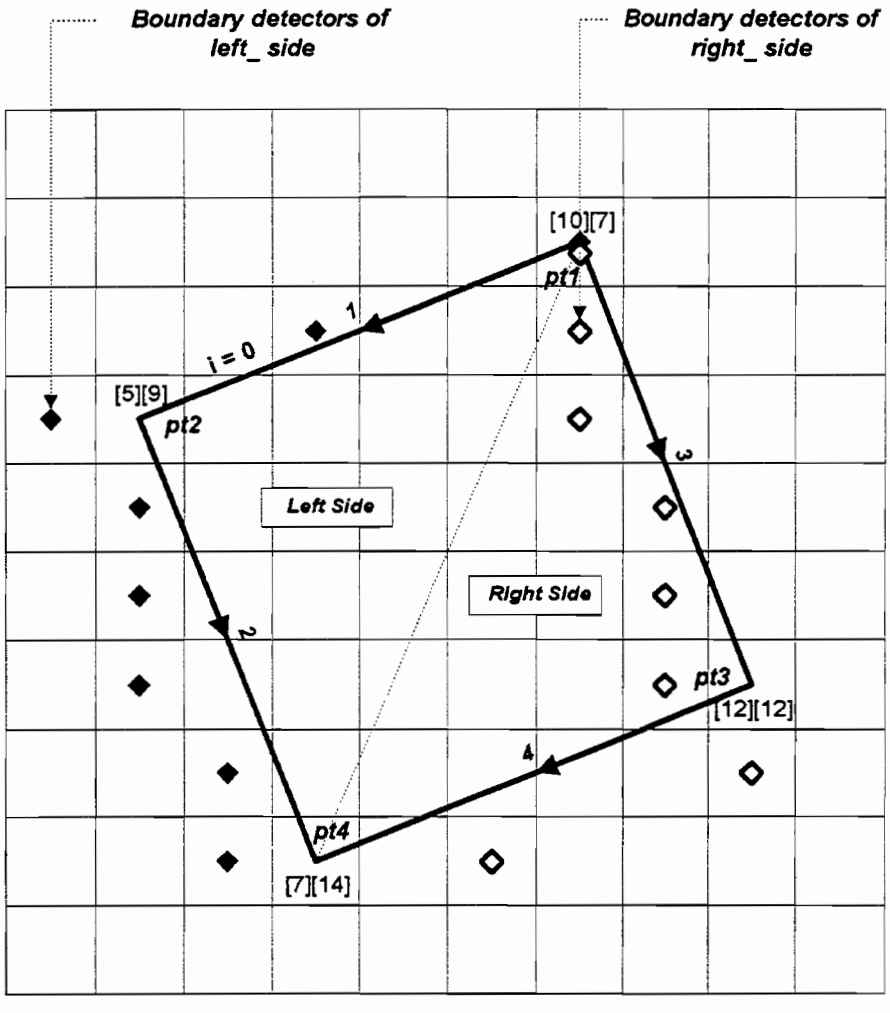


Figure 10.7 Bounding Detection Process

Value_of_Detect_Pixs / get_value:

This function detects all the x-y coordinate pairs within the boundary detectors searched by "Box_Bounds" along the x-axis. "Value_of_Detect_Pixs" also gets the gray level value of each detected pixel from "get_value" provided by vision software and calculates the total pixel value of the given bounding box. Below Figure 10.8 graphically describes the process for getting the total pixel value of the given bounding box along the x-axis. More detailed description of this process is presented in the comments of the source code of the line snapping software.

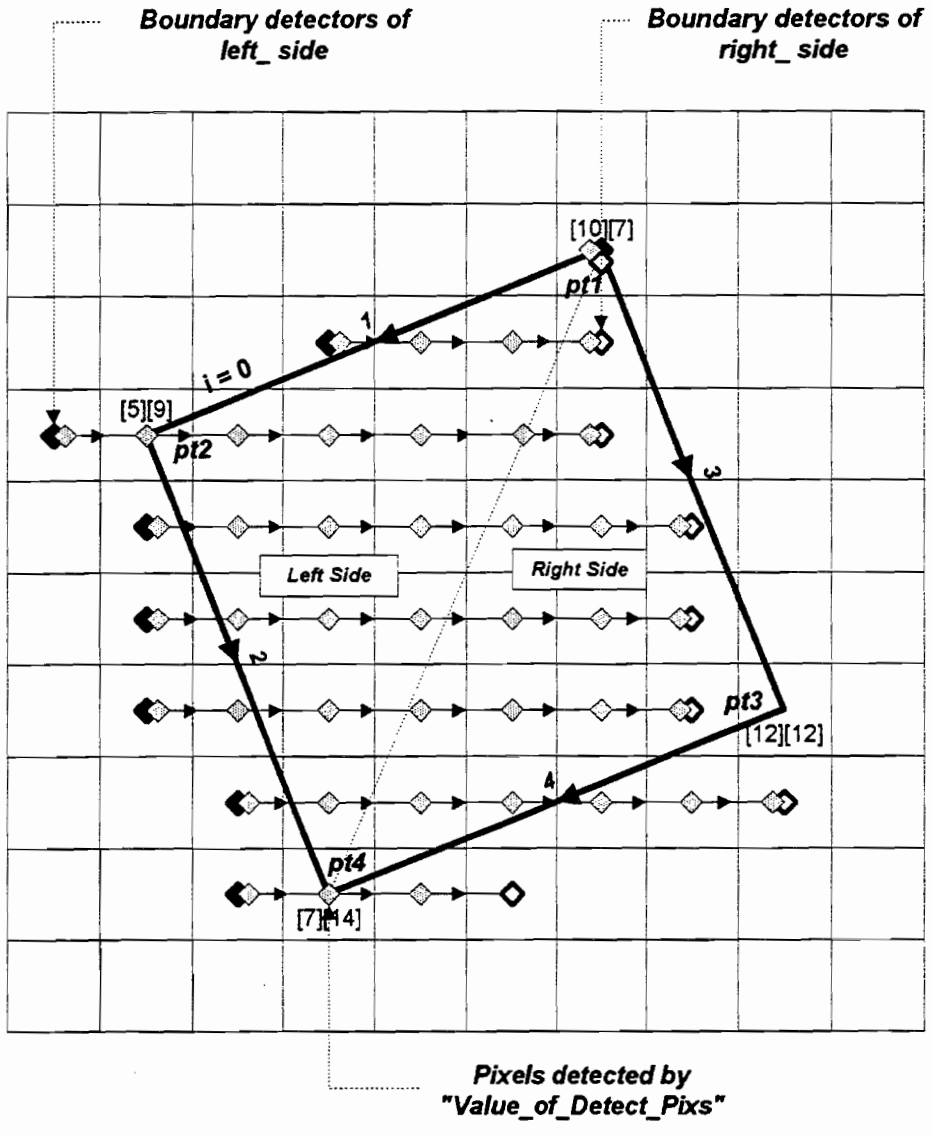


Figure 10.8 Process for Obtaining Total Pixel Value

In calculating the total pixel value of the given box, this function assigns 255's (white) for pixels that are not included in the image array size, [640 x 480]. Finally, the total pixel value of the given box is assigned to a variable, 'Total' and "Value_of_Detect_Pixs" returns the 'Total' to a variable, 'Box_Value' in "Value_of_Box." Then, "Value_of_Box" returns the 'Box_Value' to "Find_Best_ Bounding_Box." Once this is done, the returned 'Box_Value' is assigned to an array, 'dark_values[]' for the comparison with the total pixel values of other bounding boxes to be created. These processes which have been

described are repeated until the best bounding box with the lowest average pixel value that implies the exact location of the crack to be sealed is determined. Thus, the "Find_Best_ Bounding_ Box" (1) creates 21 bounding boxes along the normal, (2) compares total pixel values of bounding boxes created and (3) finally finds the best box.

Then, the last step of the line snapping software is to create an array which is called 'snap_out[]' to be used as an input data for the efficient traversal plan of the automated crack sealer and then move the line segment to the middle of the best box.

Put_Bounding_In_Snapout:

For the efficient path generation of the automated crack sealer, this function puts three points (left, middle and right) of the line segment adjusted to the middle of the best box into the snap_out[]. As shown in Figure 10.9, the line snapping software replaces the right side point of current best box with the midpoint between the current point and the left side point of the best box to be selected in next line segment. Casts are done to avoid compile warnings.

Figure10.9 and Figure 10.10 illustrates an example of the results which can be accomplished through the software running. Here, it was assumed that the 10th box in the given example was selected as the best box.

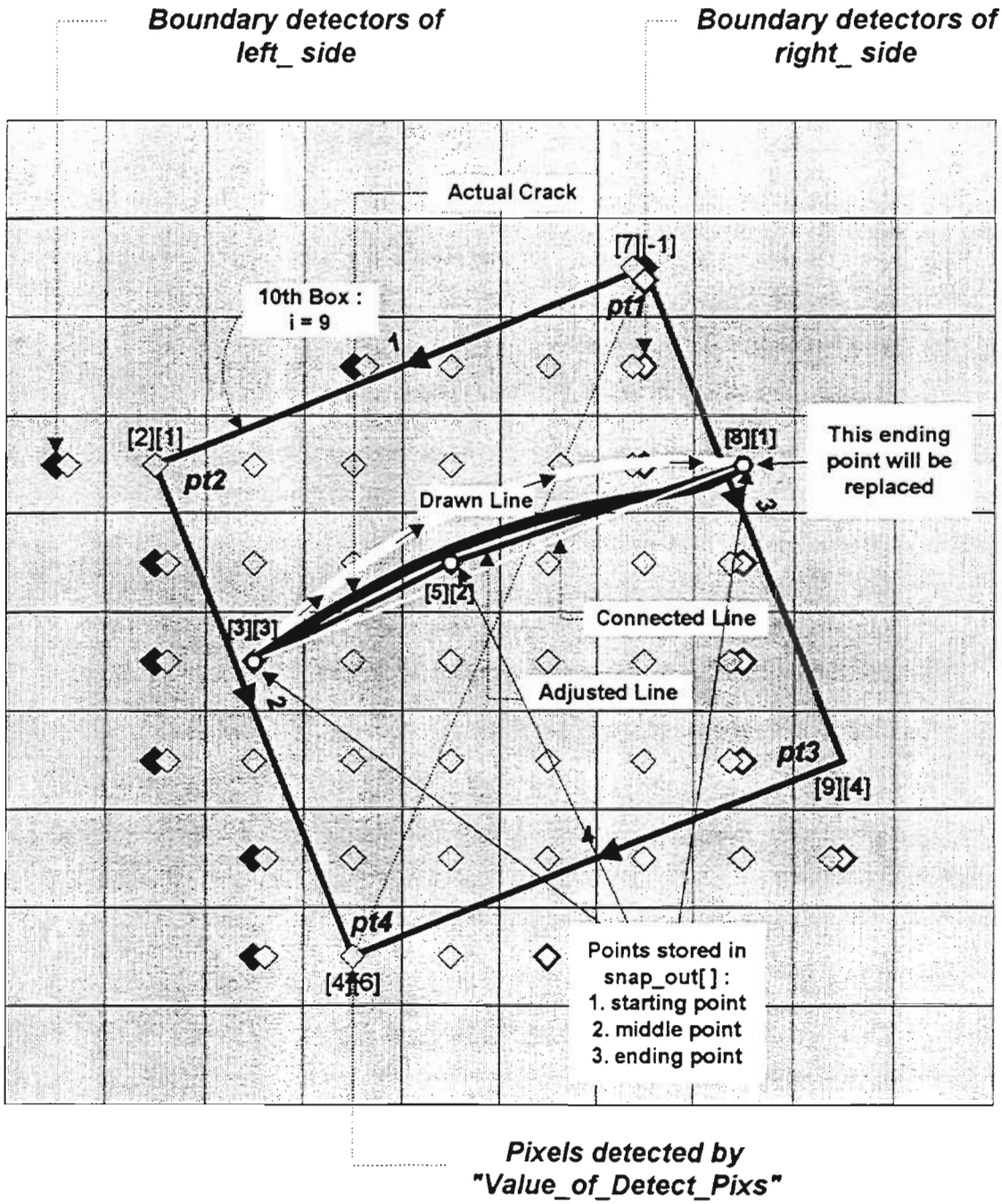


Figure 10.9 Example Results

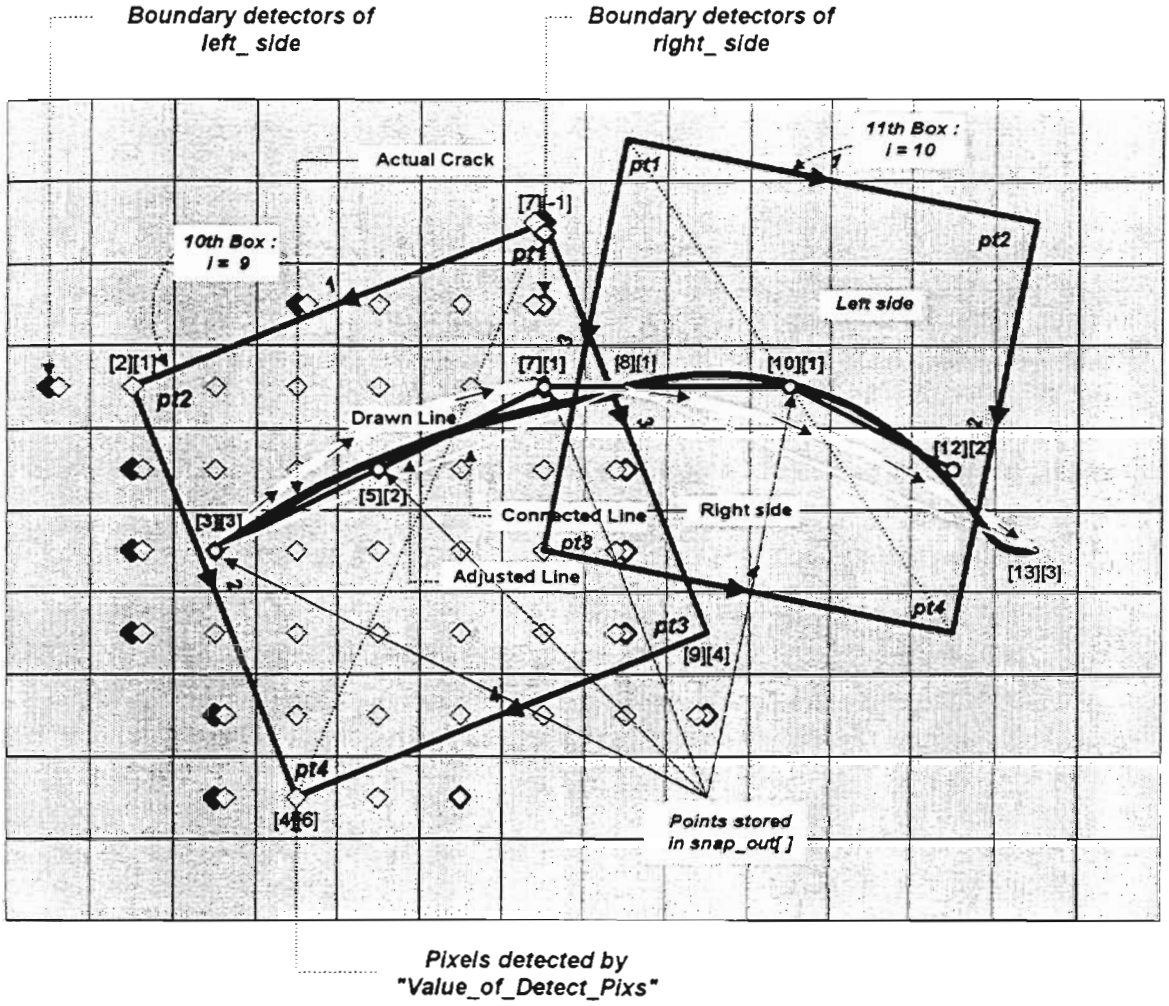


Figure 10.10 Example Results

10.4 Added Features

I. Once the RBBA is completed, an updated drawing of the line segments is performed to allow the system operator to verify they represent the cracks that need to be sealed. The update drawing lines (adjusted line) of the line segment are shown in Figure 10.9 and Figure 10.10. It consists of three points including a middle point of each line segment while the connected line segments which are drawn over the pavement image for each crack as it is traced have two points, beginning and ending point.

II. An algorithm which is called "Rubber Band Algorithm" can be also used for the line segments that are not properly adjusted by the RBBA. It is anticipated that the "Rubber Band Algorithm" can maximize the effectiveness of the line snapping software.

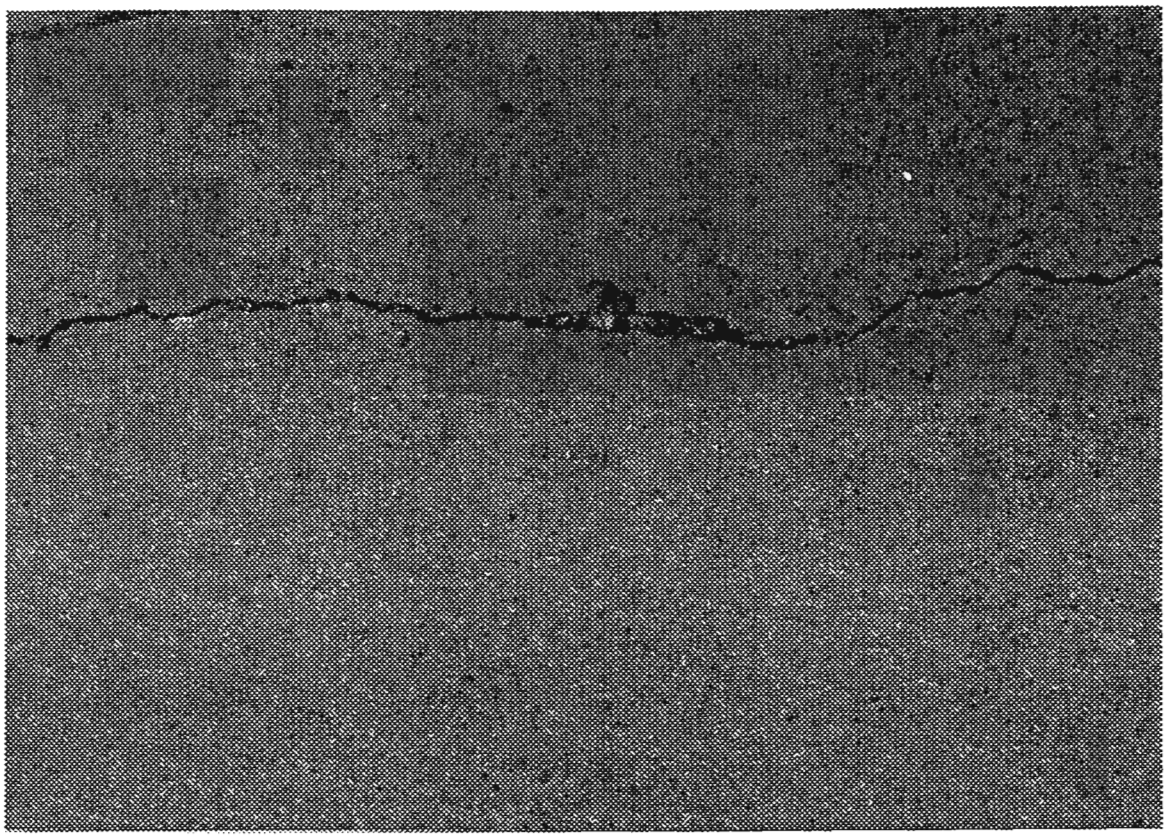
III. The line snapping software automatically calculates time (starting and ending time) required for the line snapping process in an effort to measure its computational efficiency. A time unit of a microsecond was utilized to measure the computation time of the line snapping software as accurately as possible. It can also print out all the x-y coordinate pairs of both connected lines and update drawing lines.

10.5 Testing

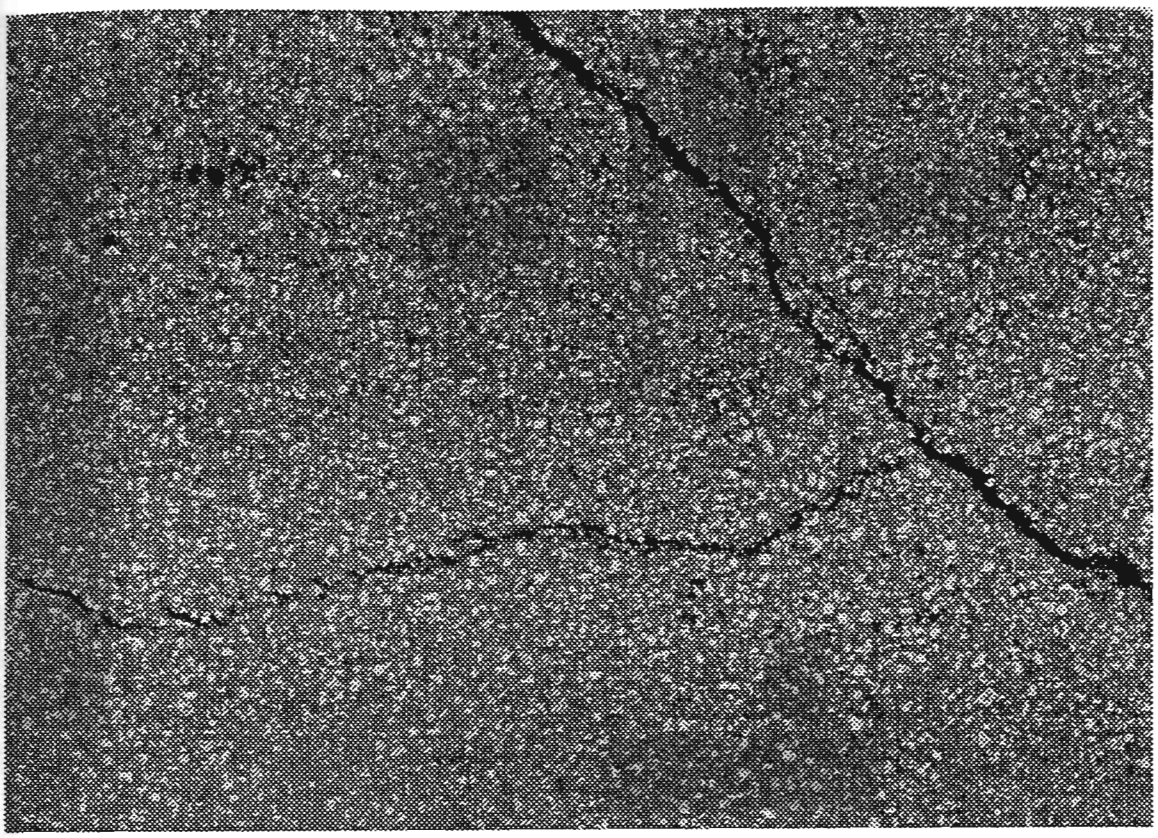
This software has been tested on twenty real pavement images which are obtained through the vision sensor. Followings include the tested images and the results. Finally, it is anticipated that the algorithms described in this documentation may eventually have broader applications in infrastructure maintenance, surface finishing, material handling and welding.

Table 10.2 The Results of Time and Accuracy Measurement Based on the Collected 20 Pavement Crack Images

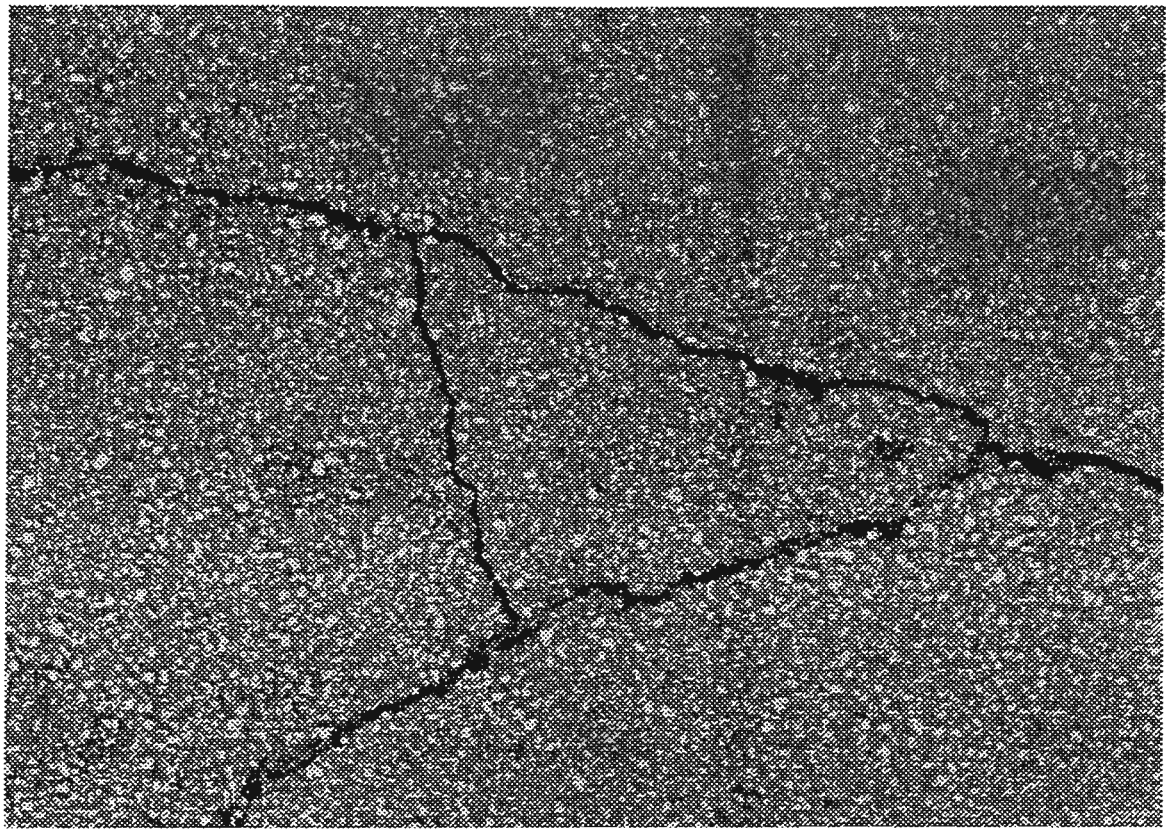
	Time Required for Line Snapping [Second]	Total Seg. Points Generated from Line Snapping	Seg. Points to be Manually Edited (Over ±3 pixels)	Accomplished Accuracy [%]
Image 1	2.09	43	4	90.70
Image 2	3.57	62	2	96.77
Image 3	4.72	85	4	95.29
Image 4	3.08	55	5	90.90
Image 5	4.45	83	7	91.57
Image 6	4.84	89	3	96.63
Image 7	3.41	70	2	97.14
Image 8	2.64	52	2	96.15
Image 9	4.72	89	1	98.88
Image 10	2.31	50	0	100
Image 11	3.41	61	3	95.08
Image 12	3.79	70	4	94.20
Image 13	3.24	65	3	95.38
Image 14	2.08	40	4	90
Image 15	4.94	94	6	93.62
Image 16	4.06	78	6	92.30
Image 17	4.94	102	4	96.08
Image 18	2.74	51	6	88.24
Image 19	3.68	76	5	93.42
Image 20	3.13	55	6	89.09
TOTAL	**71.84**			
AVERAGE	**3.59 Seconds**			**94.07 %**



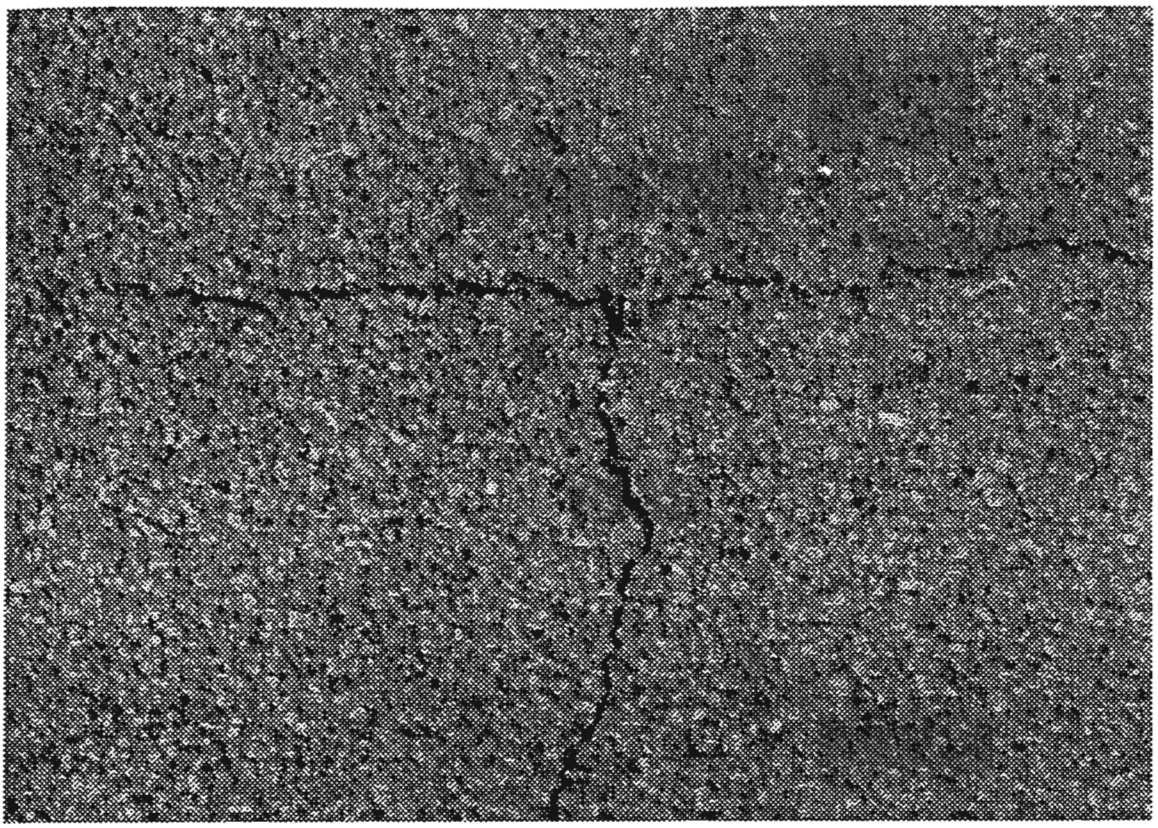
Test Image 1



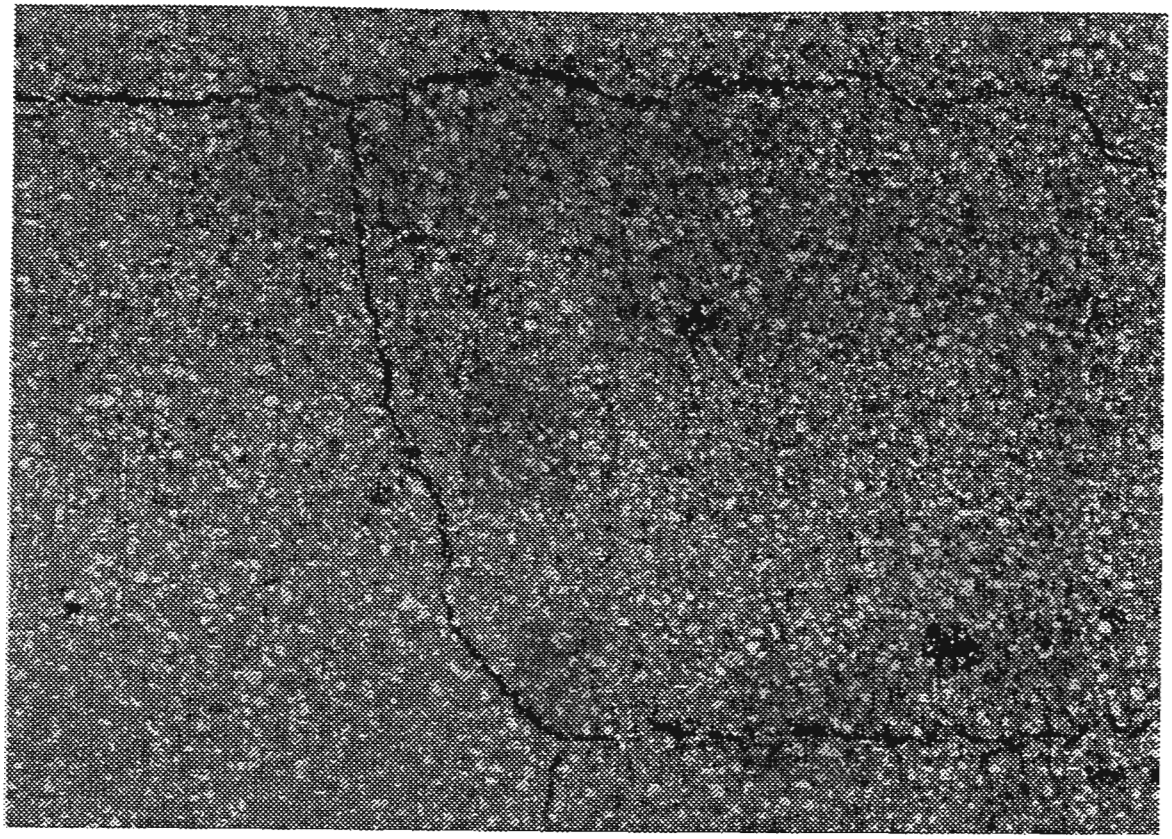
Test Image 2



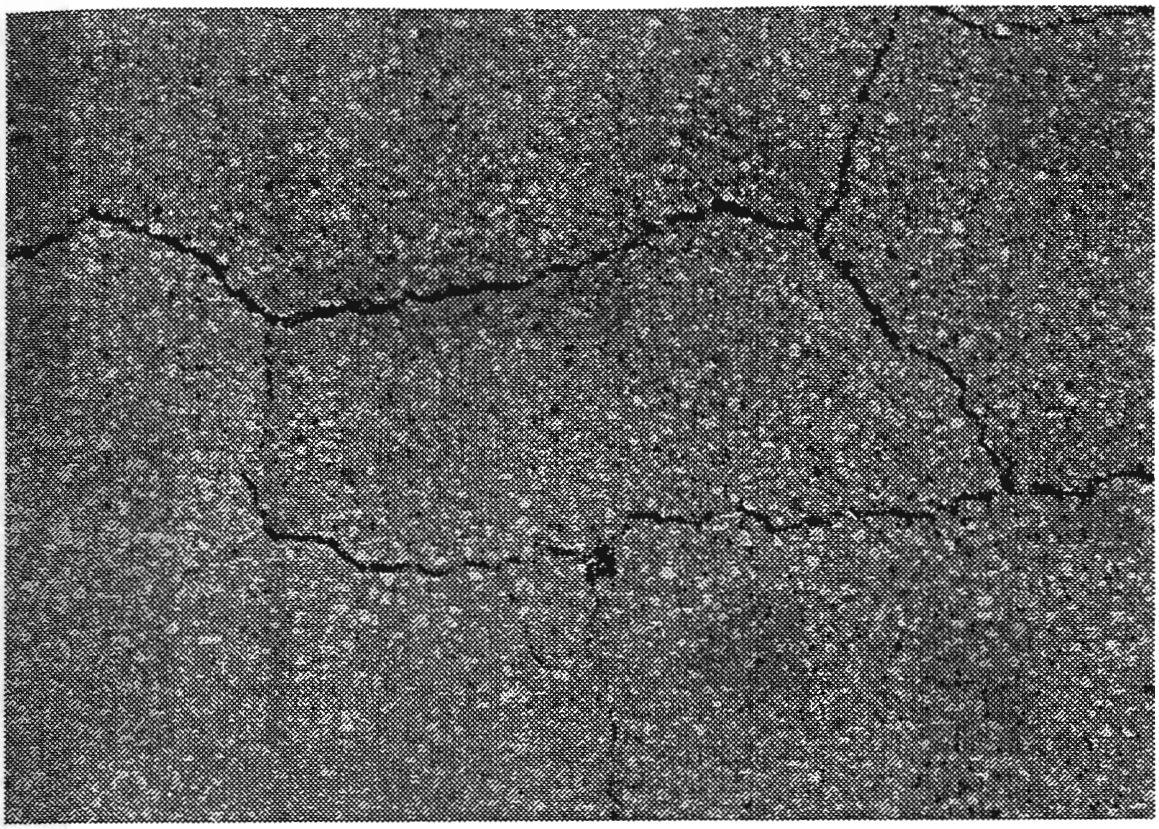
Test Image 3



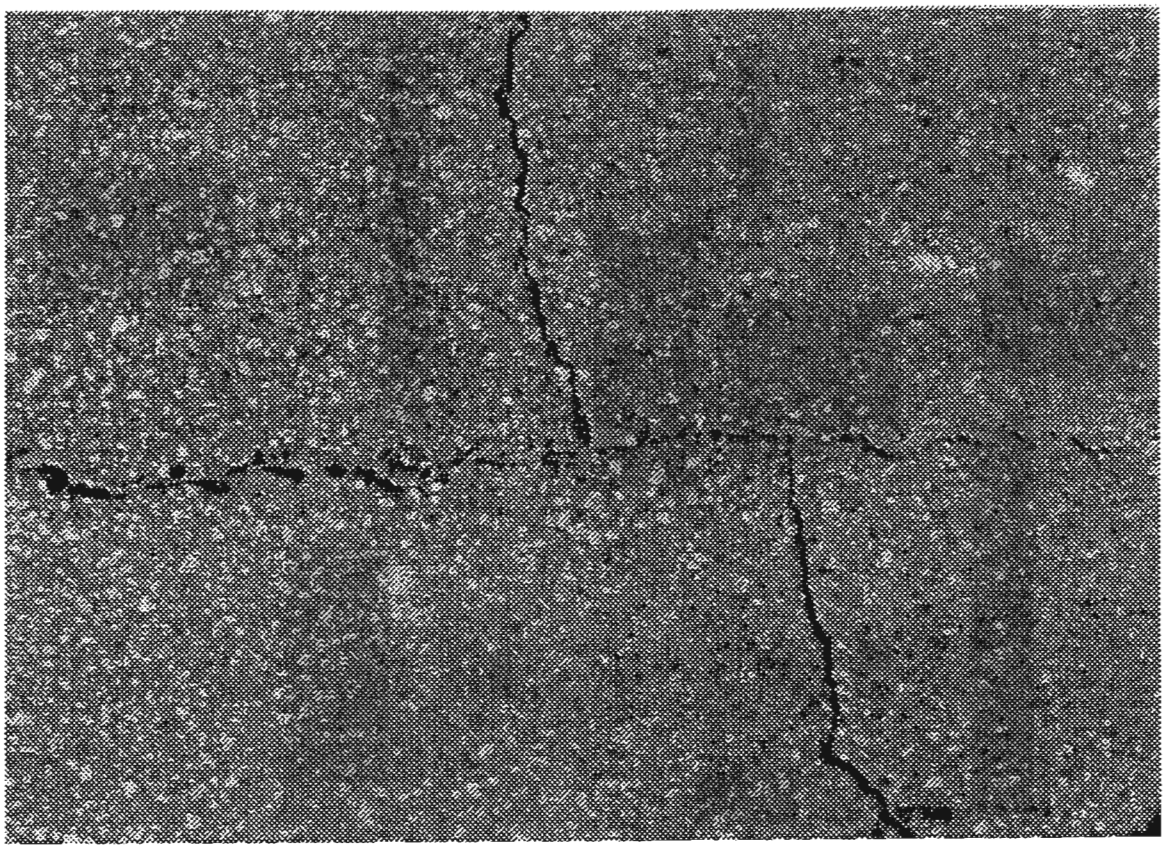
Test Image 4



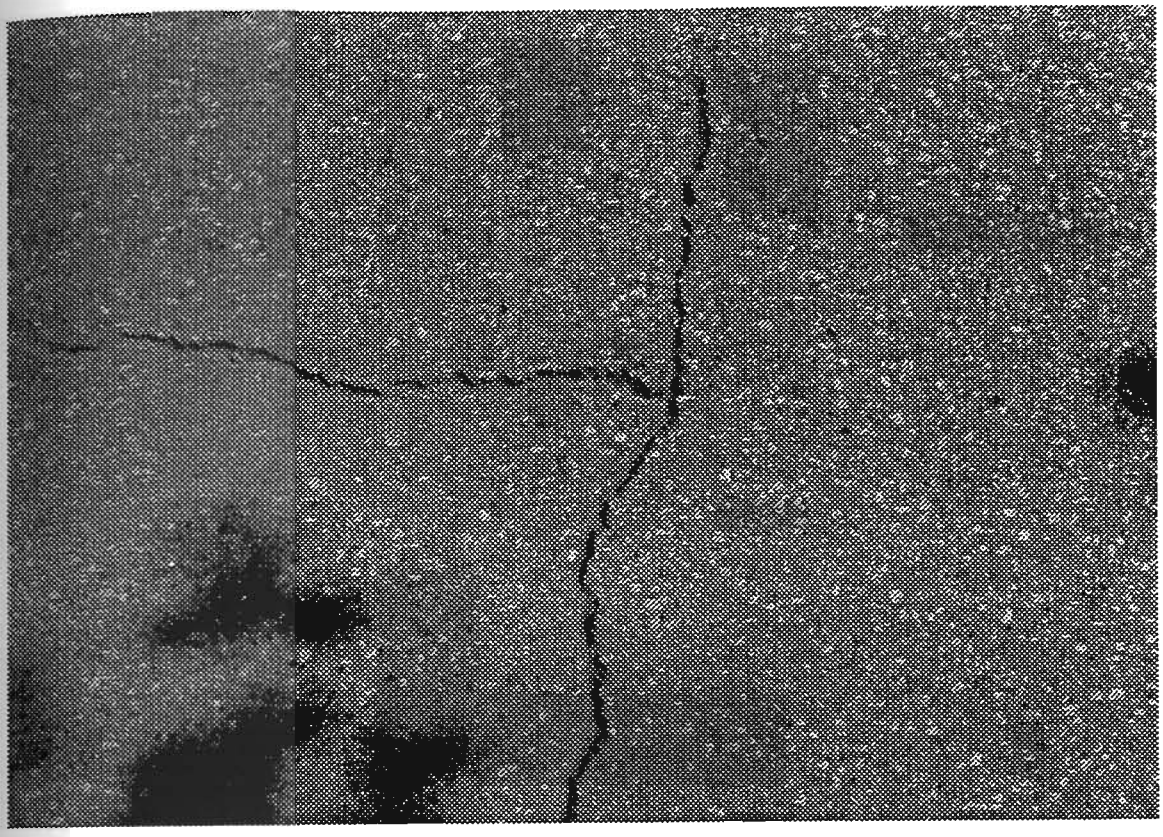
Test Image 5



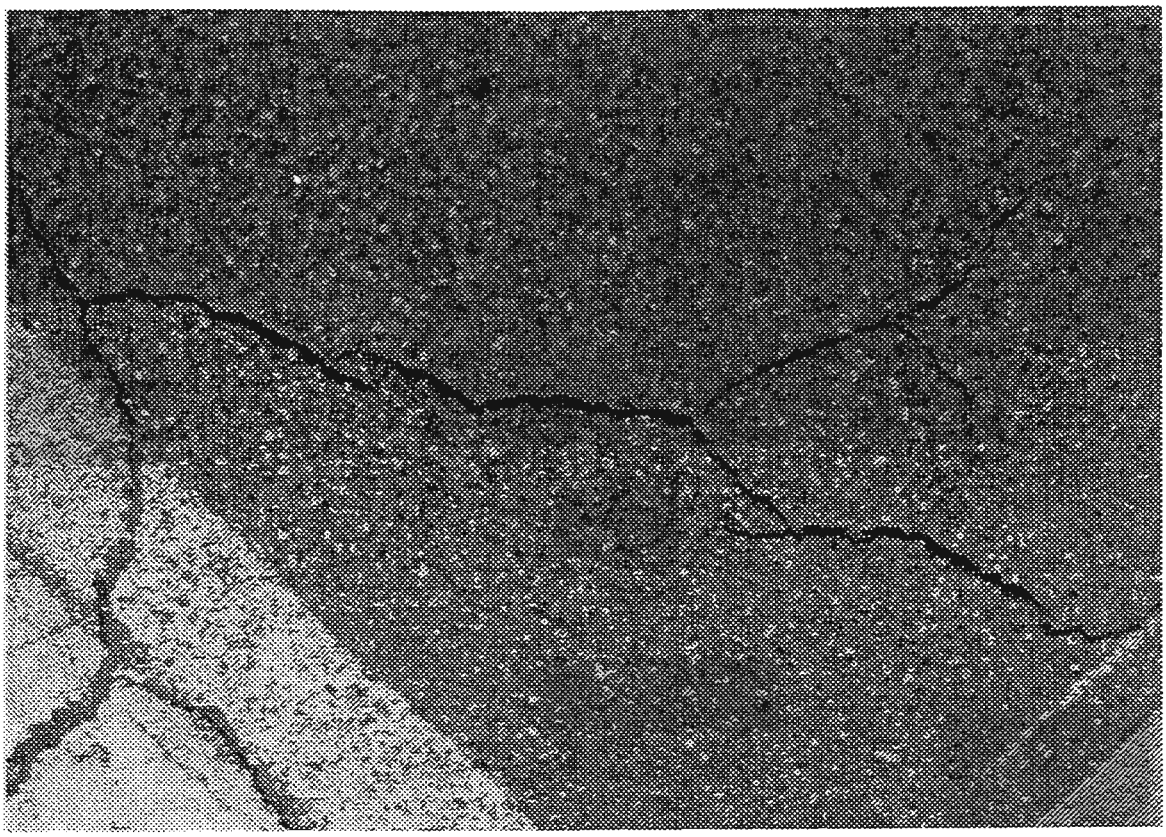
Test Image 6



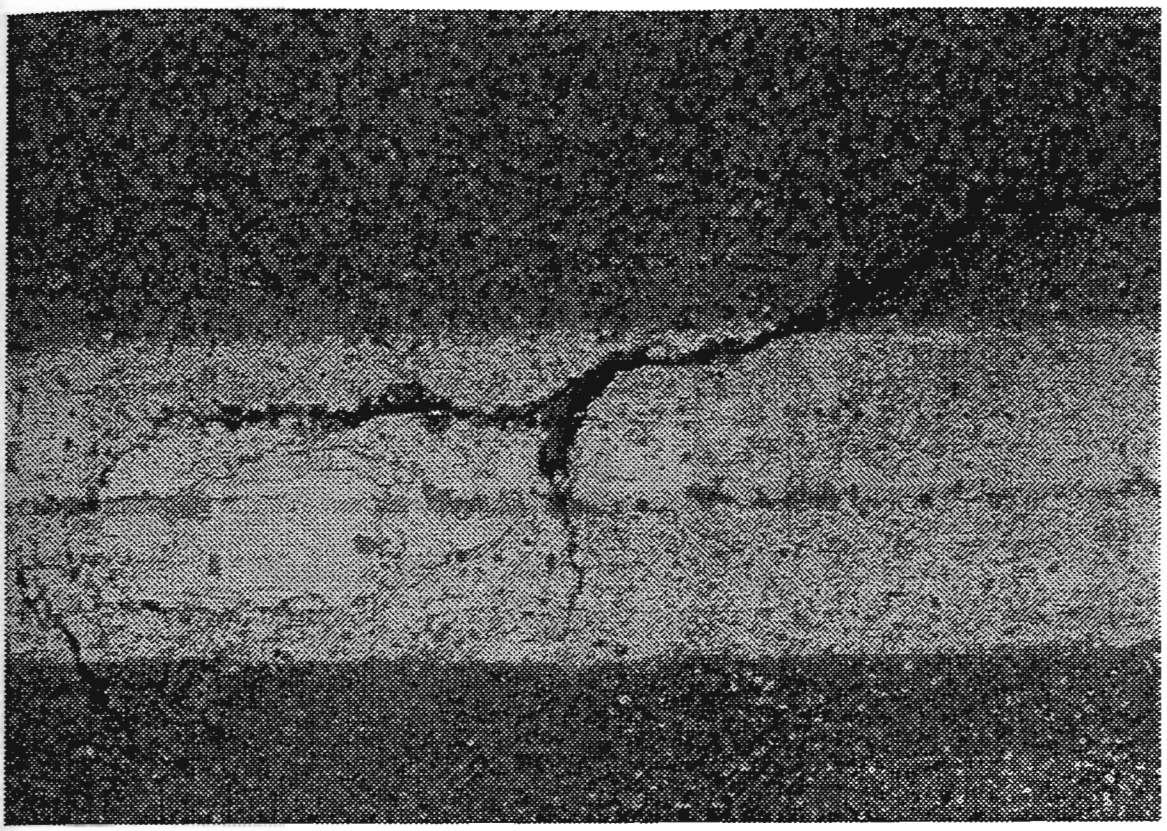
Test Image 7



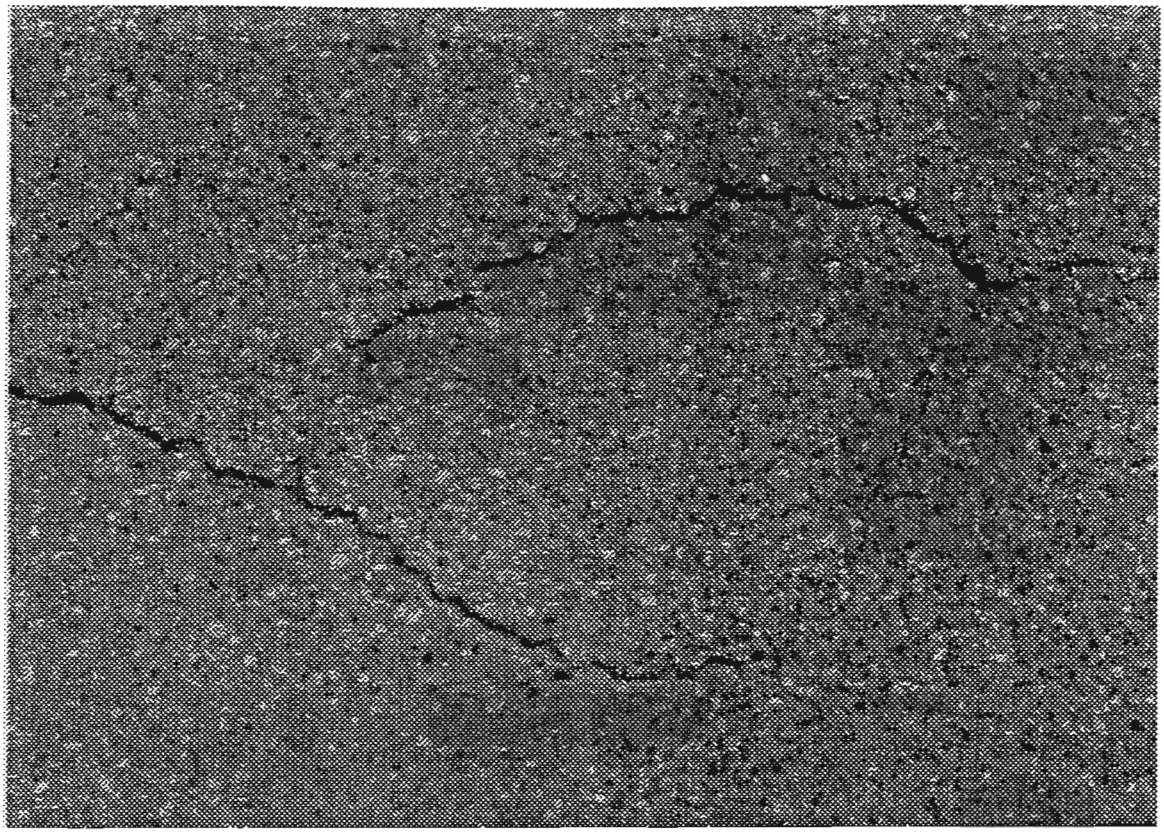
Test Image 8



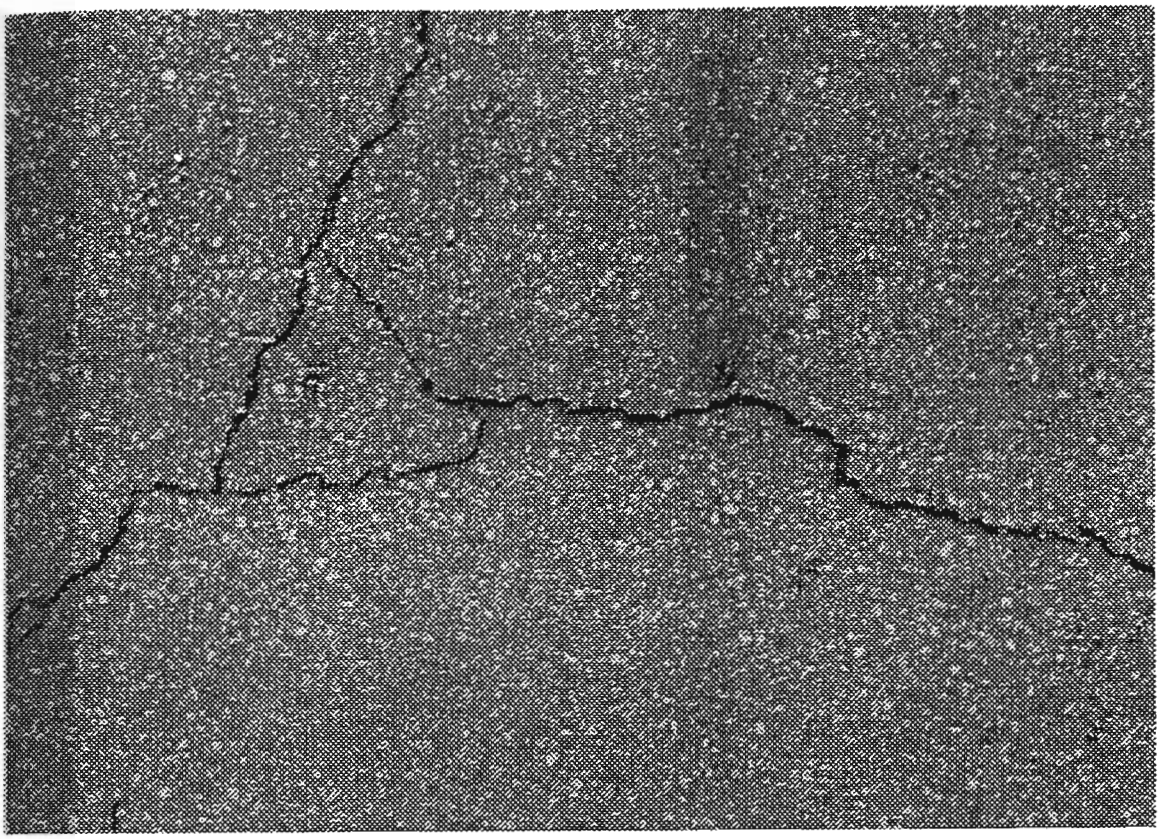
Test
Image 9



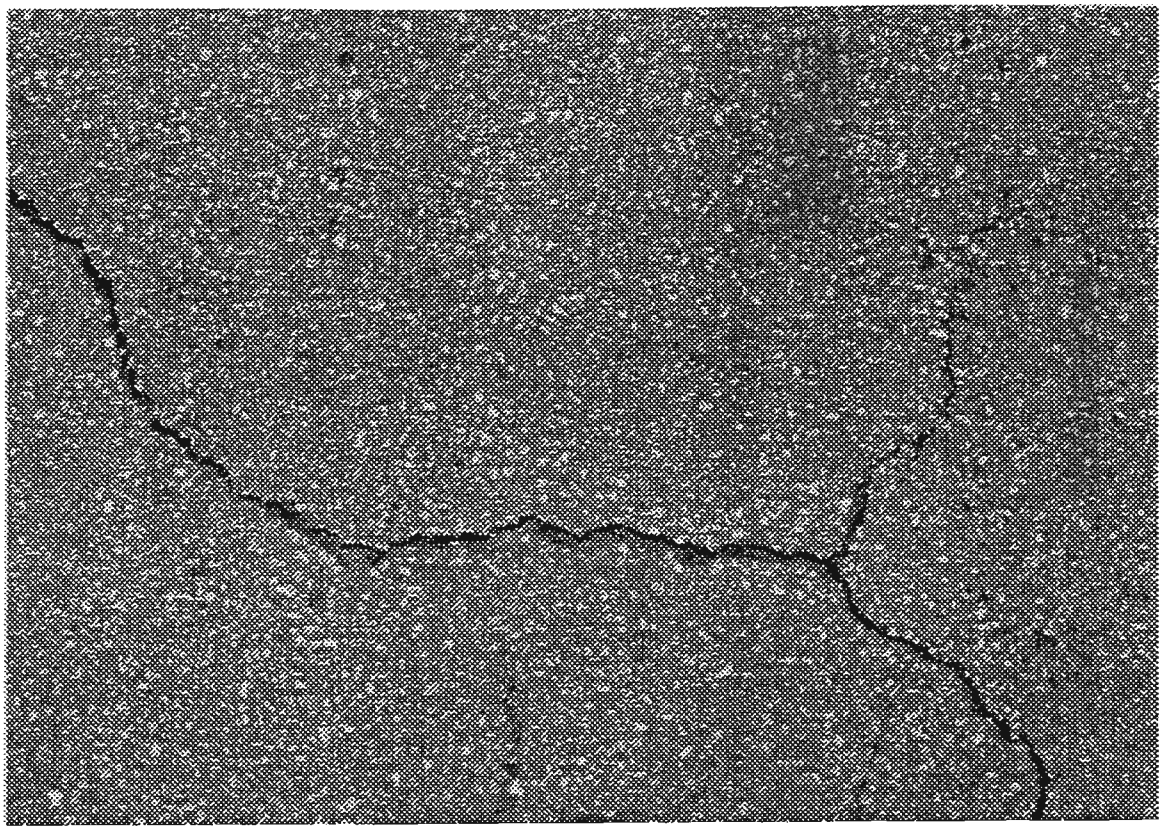
Test Image 10



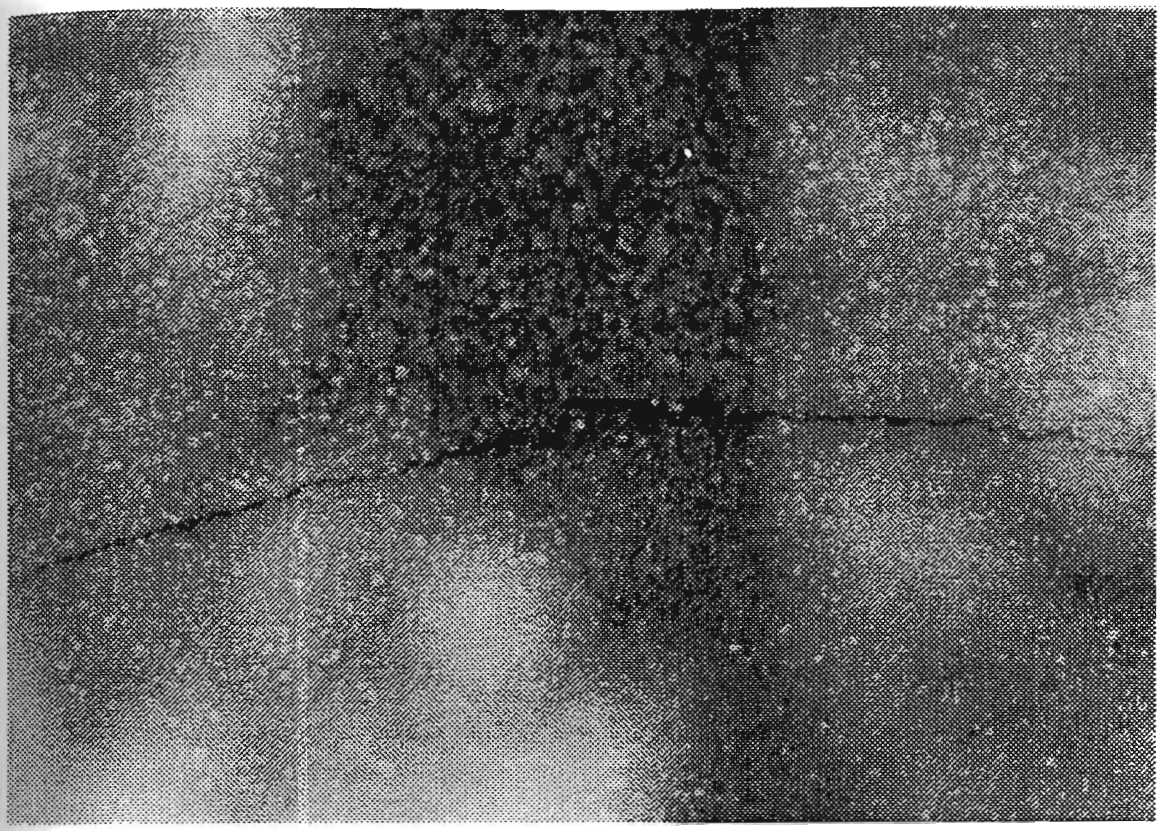
Test Image 11



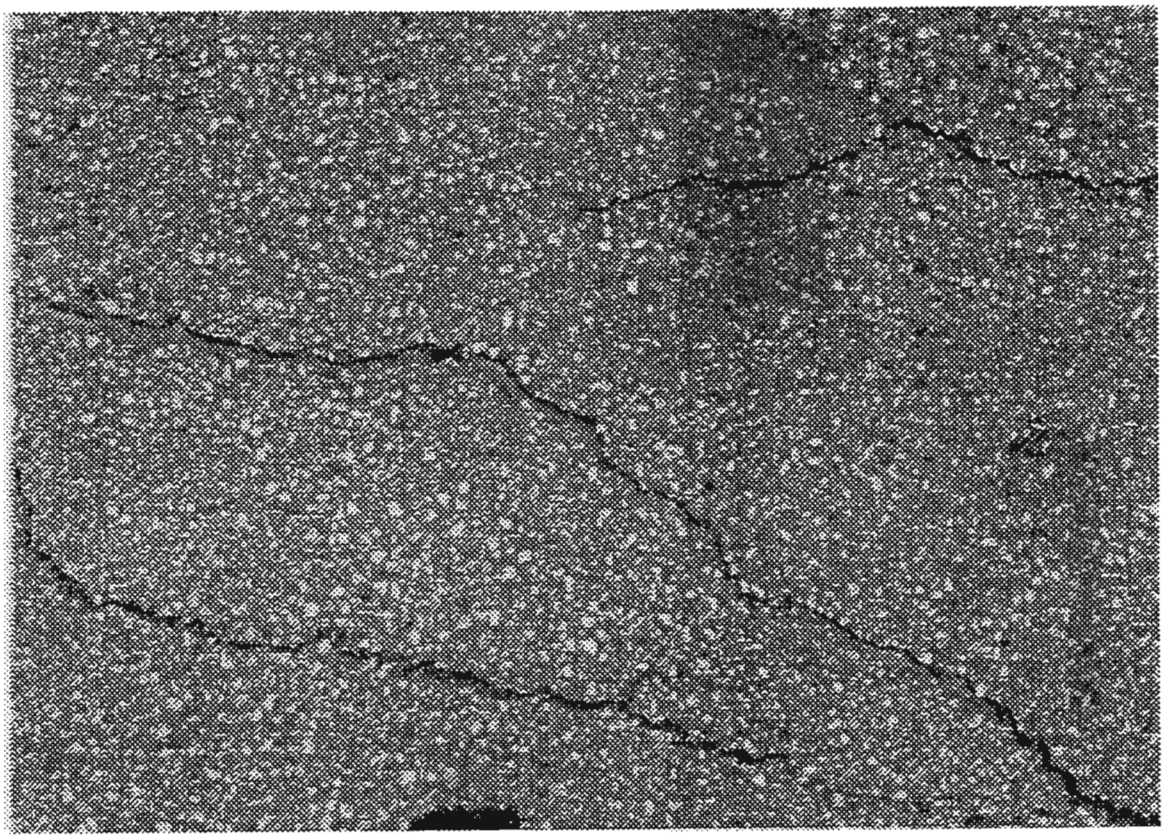
Test Image 12



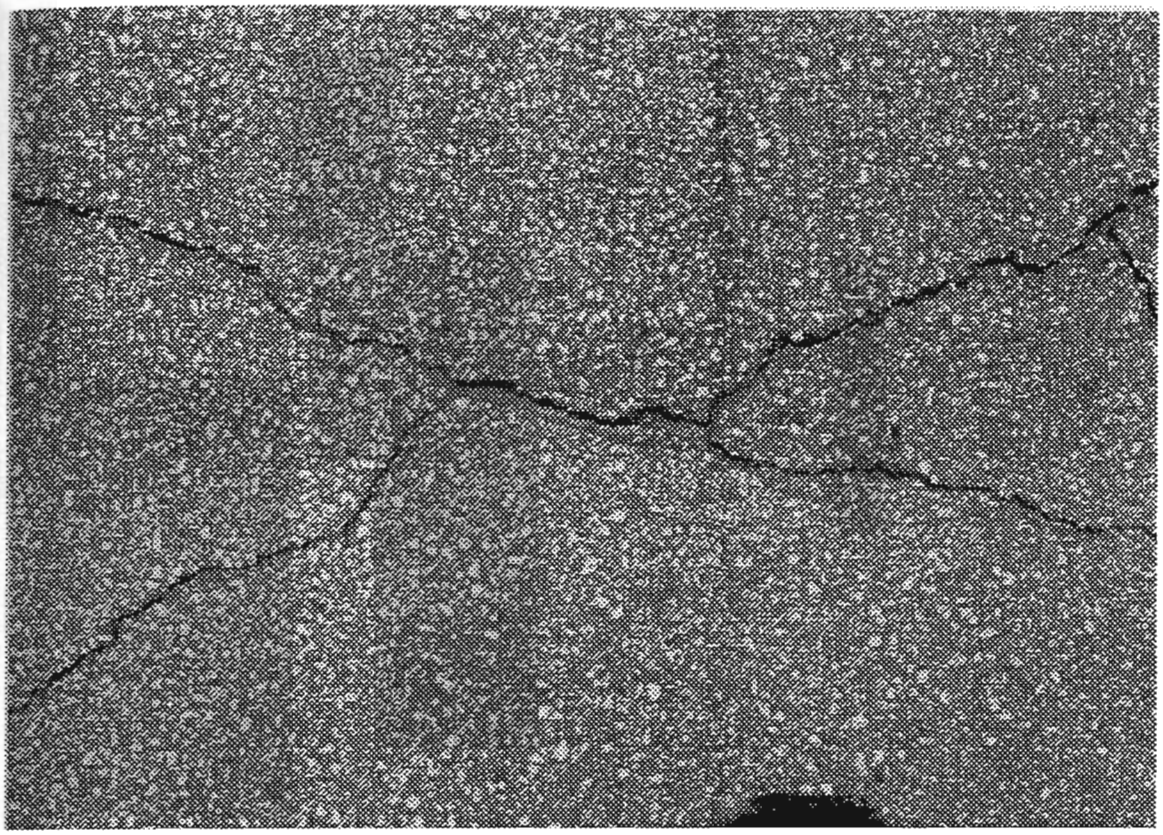
Test Image 13



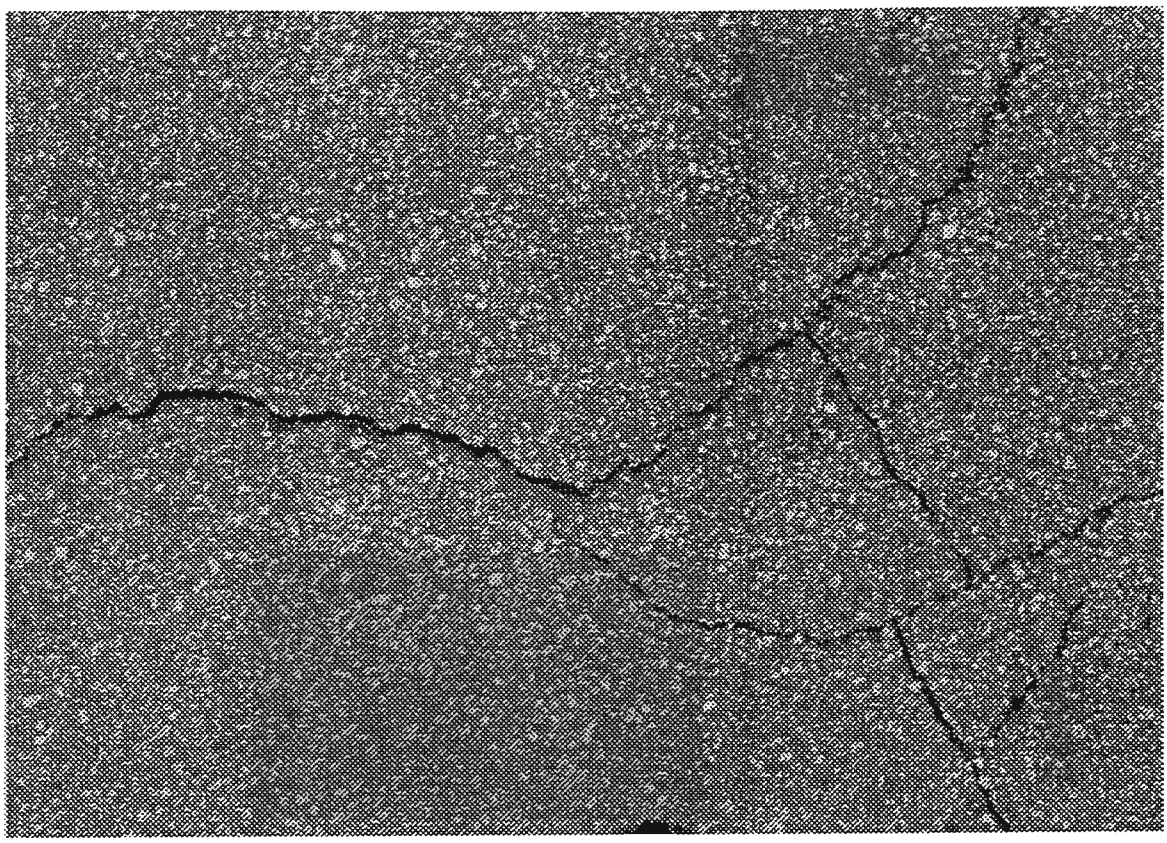
Test Image 14



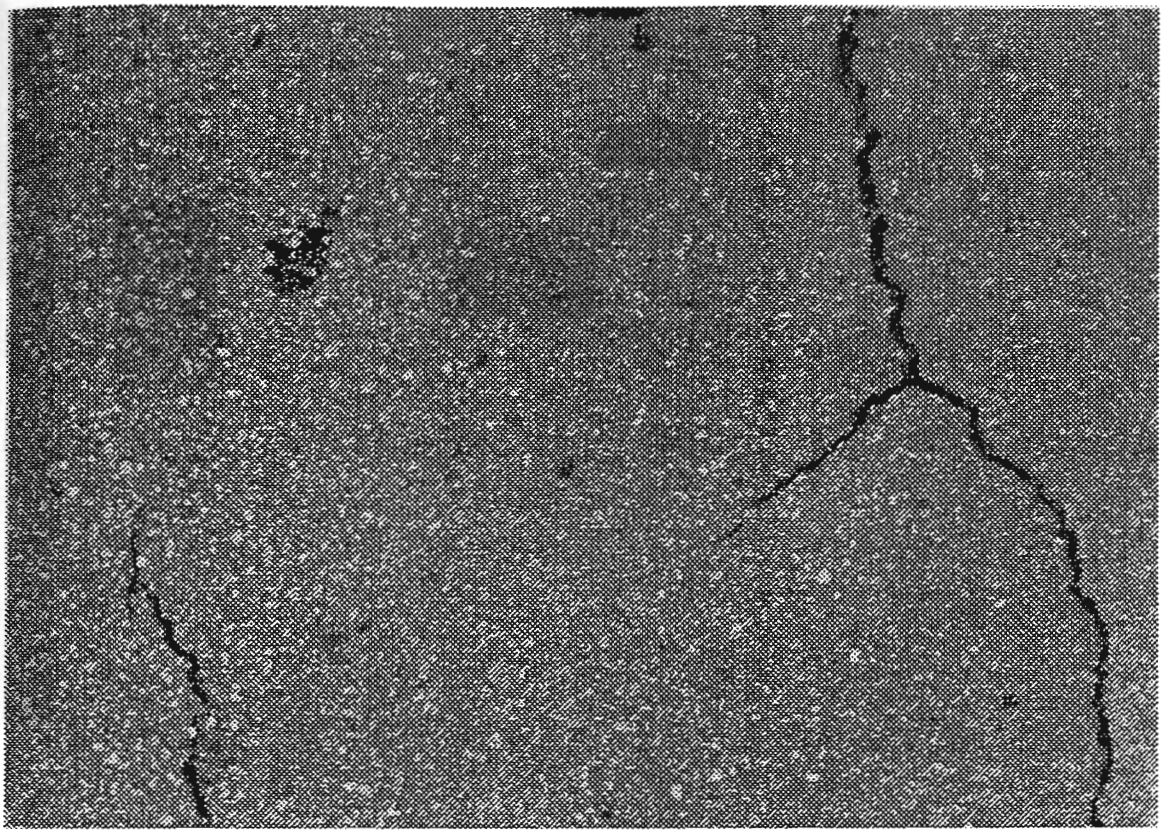
Test Iamge 15



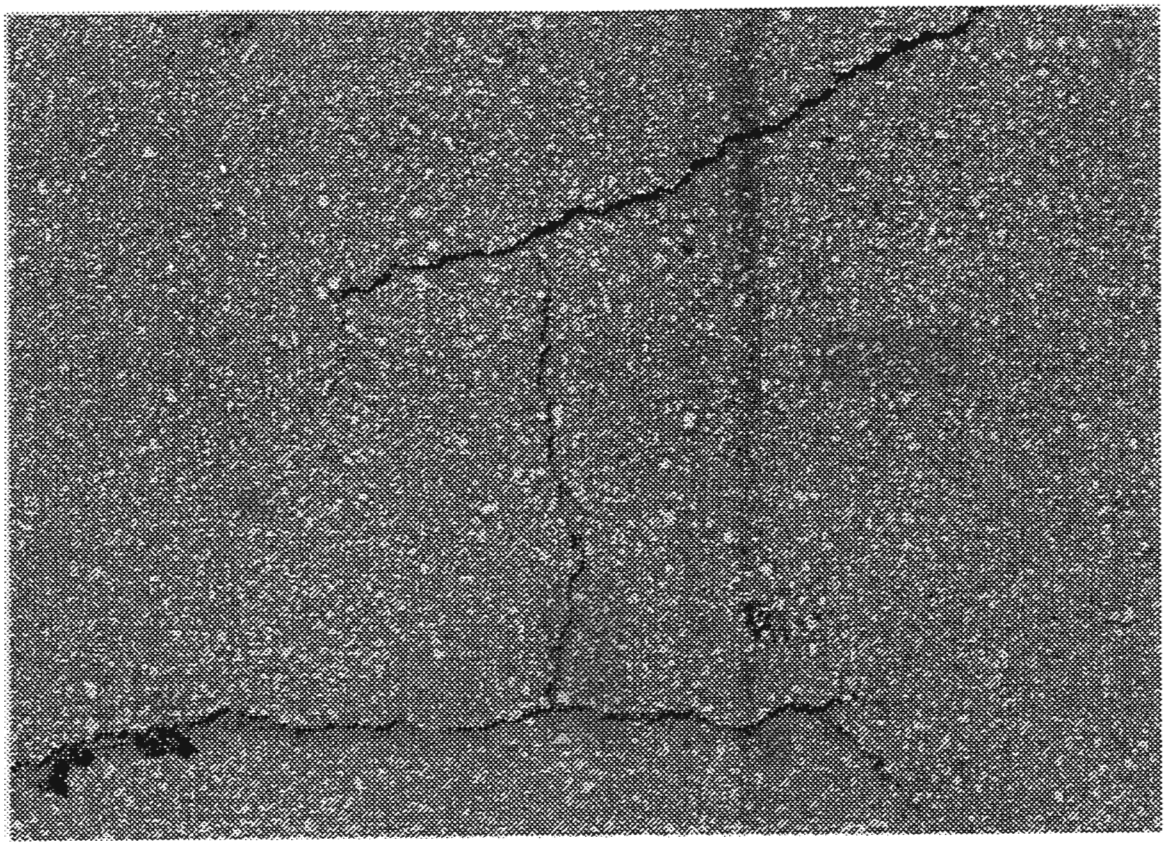
Test Image 16



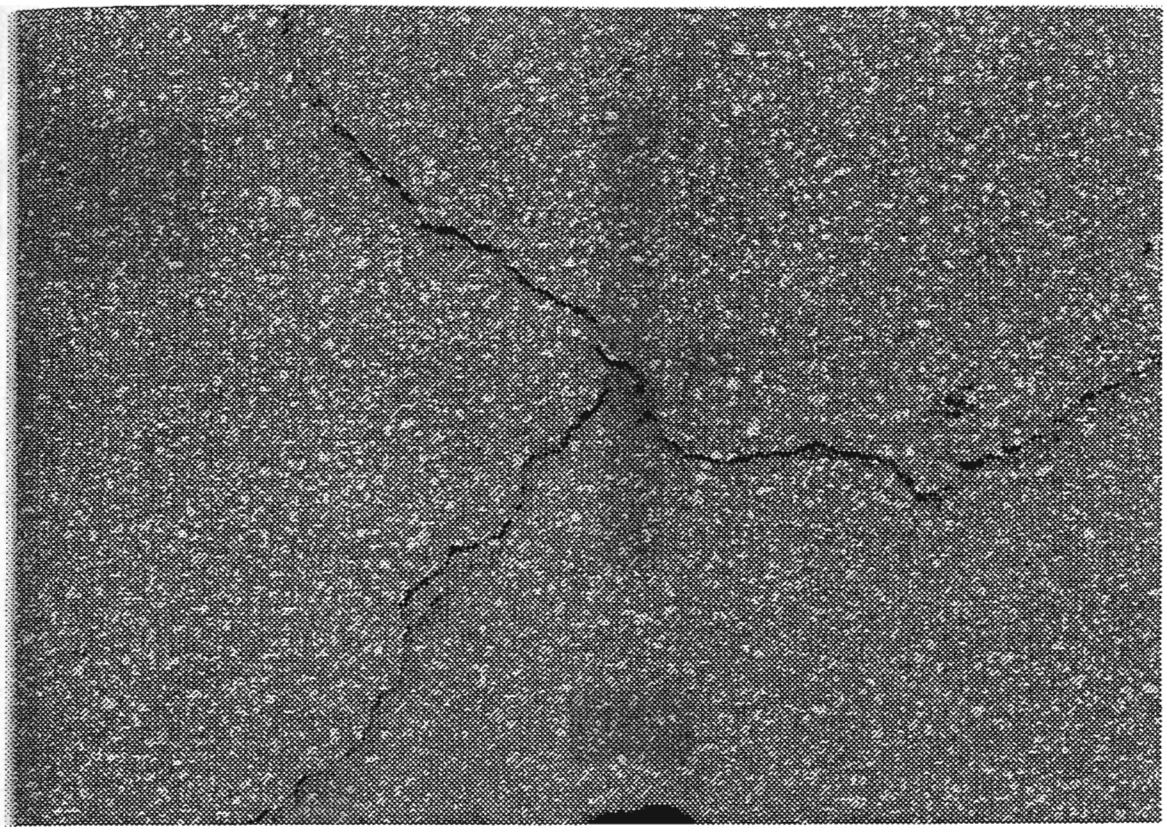
Test Image 17



Test Image 18



Test Image 19



Test Image 20

11.0 Operation Instructions

11.1 System Operation

ALL OPERATING MANUALS FOR ALL COMPONENTS OF THE ARMM MUST BE READ THOROUGHLY BEFORE OPERATING THE ARMM. ALL SAFETY PRECAUTIONS MUST BE FOLLOWED.

I. Required Equipment

1. Truck with hitch attachment
2. X-Y manipulator table with attached video camera system
3. Generator, compressor, amplifier, motor controller, personal computer
4. Crafco Super Shot 60™ Propane melter
5. Cables for connecting all electronic equipment

II. Required Software

1. Automated Road Maintenance Machine operating software
2. Microsoft Windows 3.1

III. Safety Precautions

The instructions provided in this manual were produced under the assumption that the user has a basic understanding of the safe operating procedures for electrical and high pressure equipment (100 psi). If this is not the case, the user should refer to the safety instructions in this report and to the safety instructions included in the manual of the individual component. Users should familiarize themselves with all safety procedures before attempting to operate and/or maintain any of the equipment. If this equipment is ever left in an unsafe manner please tag accordingly to reduce the risk of harm to the user or further damage to the equipment.

IV. Assembly of the ARMM System

1. Attach truck to the melter trailer.

2. Attach truck/melter trailer to the X-Y table.

3. Place the sealant wand from the melter into the opening at the top of the turret on the X-Y table. See figures 13.1 and 13.2 on the next page.

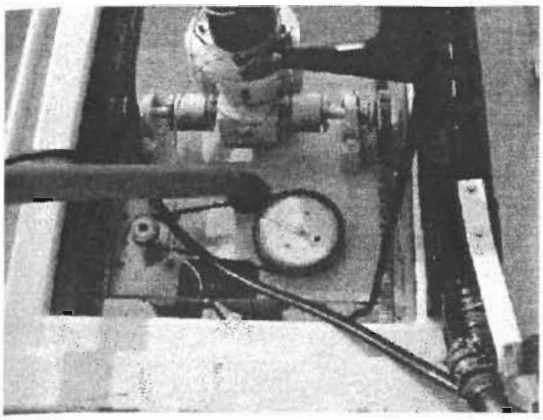


Figure 11.1 Placing the Sealant Wand

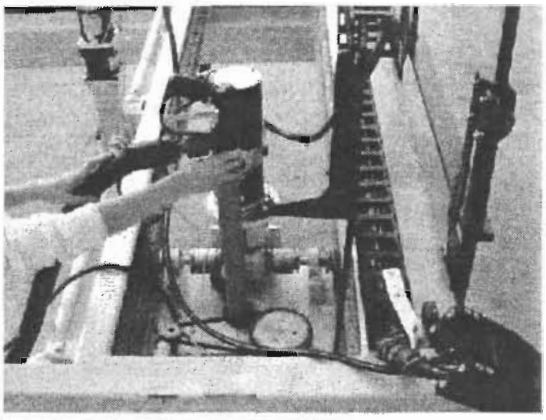


Figure 11.2 Placing the Wand

4. Remove the canopy securing bolts and raise the canopy. Once canopy is raised, secure canopy by inserting bolts at the base. There are four bolts to be inserted. See figures 13.3 and 13.4 below.

Figure 11.3 Lifting the Canopy

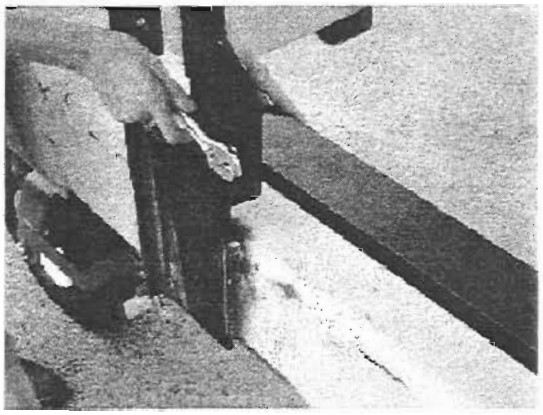


Figure 11.4 Tightening the Bolts

5. Connect the cables from the controlling equipment on the truck to the proper location on the X-Y table. Connections include three cables to control X, Y and rotation directions, an air hose, and two cables for each of the video cameras. There is also a power cable for the cameras Do not let cables drag on the pavement. Cables should be secured firmly to the truck. See figures 11.5 and 11.6 below.

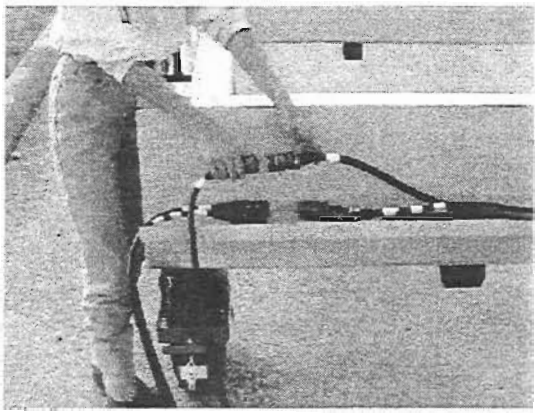


Figure 11.5 Connecting Cables

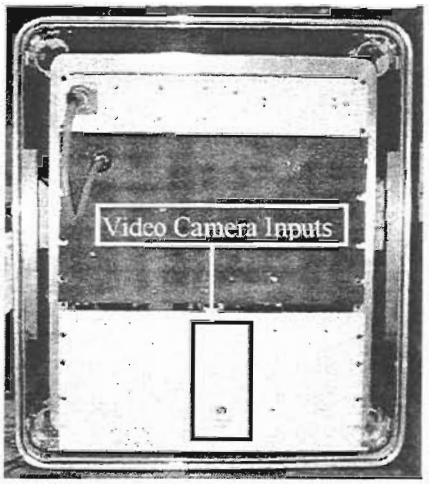


Figure 11.6 Video Inputs

6. Start the generator and the air compressor following the proper start-up procedures outlined in their respective manuals.

7. Flip the electronics switch located on the Aerotech DR 500 amplifier chassis to the "ON" position. This will supply power to the motors. See figures below.

8. Flip the electronics switch located on the Tripp Lite AC voltage regulator to the "ON" position. This will supply power to all of the electronic equipment. See figure 11.7 below.

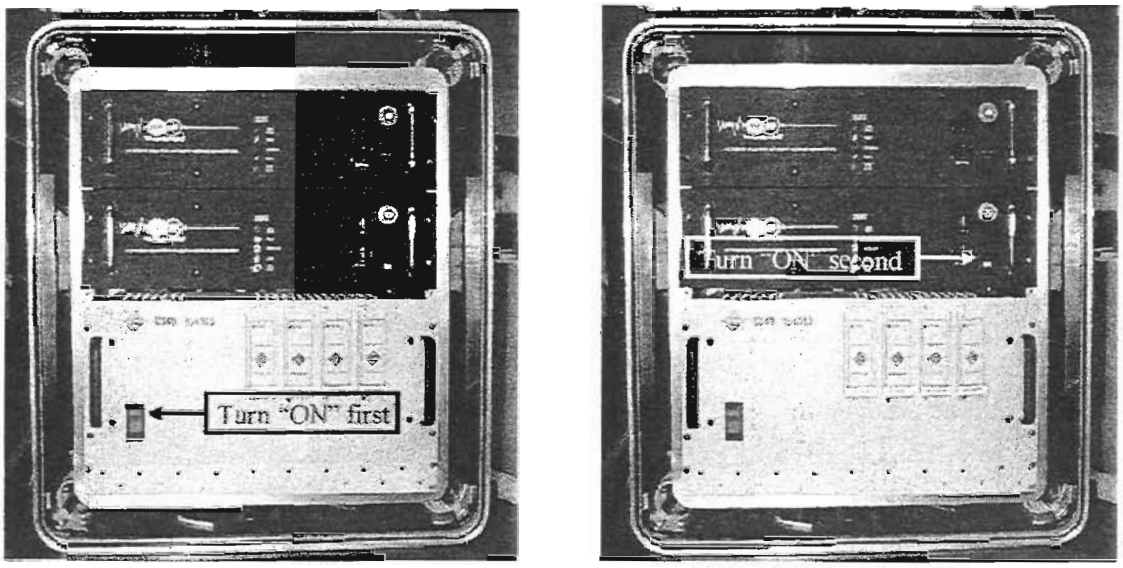


Figure 11.7 Display of Electronic Power Switches

9. Start the melter following proper start-up procedures outlined in the operating manual. Please note that sealant will take approximately 45 minutes to heat and melt.

10. Turn on PC. Run the ARMM software. An image of the road surface currently in the workspace should appear on the monitor. If not, please refer to the troubleshooting section.

V. Operation of the ARMM

1. Move the table to the desirable workspace.

2. Select the "Acquisitions" sub-menu from the "Acquire" menu. Select "Acquire full frame...." from the "Acquisitions" sub-menu. This function allows the operator to view the video image of the workspace on the monitor.

3. Trace the crack to be sealed with the mouse or digitizing pen. This trace need not be exact. The next steps will correct any errors, thus resulting in an accurate tracing of the crack.

4. Select "Auto Line Snap" from the "SW_Extra" menu on the screen. This will fit the estimated line drawn by operator to the precise location of the crack. If the line needs further editing, this can be accomplished with the mouse or digitizing pen. After editing repeat the "Auto Line Snap" process. Repeat until a satisfactory line is produced. If an unsatisfactory line is produced, select "Erase and Start Over" from the "SW_Extra" menu and return to step 3.

Select "Adjust the Points" from the "SW_Extra" menu.

6. Select "Initialize Controller" from the "SW_Extra" menu. This step is only required for the first pass of the machine. The machine does not have to be initialized before every pass.

7. Select "Trace Crack" from the "SW_Extra" menu. The machine will proceed to blow, seal and squeegee the cracks selected in the workspace.

8. Move to next workspace and repeat steps 2-7.

V. Disassembly of the ARMM System

1. Exit the ARMM software.

2. Turn off the PC.

3. Flip the electronics switch located on the Aerotech DR 500 amplifier chassis to the "OFF" position.

4. Flip the electronics switch located on the Tripp Lite AC voltage regulator to the "OFF" position.

5. Turn off the generator and the compressor.

6. Shutdown the melter following proper shutdown procedures outlined in the operations manual.

7. Unhook all cables from the X-Y table and place them in the truck.

8. Remove the sealant wand from the opening in the turret and place in its proper location on the melter trailer.

9. Unhitch the X-Y table from the truck/melter and lower the canopy.

10. Unhitch the melter trailer from the truck.

Summary for operation of SuperShot 60

BEFORE STARTING

1. Read safety precautions in the manual thoroughly.
2. Fill propane tank.
3. Check engine oil level.
4. Check hydraulic fluid level.
5. Check heat transfer oil level.
6. All toggle switches should be in off position.
7. Both temperature control dials should be at minimum setting.

TO START MELTER

1. Open valve on propane tank. (Unscrew valve counterclockwise to open)
2. Open line valve. (Turn yellow handle parallel to line)
3. Depress prime button on the regulator next to the propane tank for 2 seconds.
4. Start engine by turning the ignition key to the start position.
5. Turn on the toggle switch at burner box. Burner should ignite in approximately 4 seconds. Ignition verified with green light.
6. Set hot oil temperature at 500°F.
7. Safe heating temperature for this sealant is 400°F with a recommended pour temperature of 380°F. These temperatures will change depending on the manufacturer of the sealant
8. Turn on the hose controller when hot oil reaches 300°F.
9. Set hose temperature at 400°F. The hose will take approximately 30 minutes to come up to temperature. When hose is hot the red light will go off and the temperature can be reduced to 360°F. The hose should be run at the lowest temperature setting possible.
10. When sealant reaches liquid state turn the toggle switch for the mixer. If mixer does not move, allow material to heat longer. Mixer will not move if melter lid is open.
11. When material and hose have reached proper operating temperature you are ready to dispense the material. Turn the pump flow control to the lowest setting (counterclockwise).
12. Remove sealant wand and depress trigger. Turn the pump flow control clockwise until pump motor starts to turn. Material should start to flow. Adjust pump speed for the desired rate of flow.

TO SHUTDOWN MELTER

1. Turn toggle switch on burner box to the off position.
2. Turn hose controller to the off position.
3. Place hose in hose hanger. DO NOT KINK OR TWIST HOSE.
4. Turn mixer toggle switch to the off position.
5. Turn engine off at engine key switch.
6. Close the line valve and the propane tank valve.

****** High operation temperatures of this machine requires protective clothing, hard soled shoes and heat resistant gloves be worn by the operator! ******

12.0 Hardware Design

This chapter is an accumulation of information about the physical components of the ARMM. Table 12.1 is a listing of the individual cost of the components as of August 1996. The total approximate cost for an ARMM system is located at the bottom of the table but does not include costs for profit and marketing. The ARMM is constructed with items both custom made for the Automated Construction Laboratory and items that were bought off the store shelf. The Crafco Super Shot 60 melter was provided on loan by Crafco for this project. A list of the equipment manufacturers and an index of the manufacturers addresses and phone numbers are listed in sections 12.1 and 12.2. This information is useful for any questions or concerns.

Several different pictures and drawings of the Automated Road Maintenance Machine system components are shown in sections 12.3 and 12.4 to help further explain the action mechanisms on the machine. Section 12.5 gives information on the workspace for the ARMM and a weight estimate of the whole system which can be used for trailer design options.

Table 12.1 Component Costs for the ARMM System

Item	Qty.	Model #	Cost Each	Total Cost
AEROTECH				
410 oz-in motor	2	1410LT-MS01-E500LD	$ 1,000.00	$ 2,000.00
960 oz-in motor	1	1960LT-MS01/E500	$ 1,435.00	$ 1,435.00
Motion Controller	1	U500ULTRA	$ 2,390.00	$ 2,390.00
Three Axis of Amplifiers	1	BB501/X4-BA20-80/X4-BAC/TVO.4	$ 5,290.00	$ 5,290.00
Interconn. Cables	1	OP500-12	$ 195.00	$ 195.00
Feedback Cable	3	DC-MSO	$ 250.00	$ 750.00
Double Shielded Cable	1	ECX413	$ 800.00	$ 800.00
BEARINGS INC.				
Swiveling End Bearings	8	TRE12	$ 19.66	$ 157.28
Pillow Block Bearings	8	P2BSC012	$ 22.06	$ 176.48
CYBO ROBOTS				
XY-Table	1	Custom	$ 26,950.00	$ 26,950.00
DATA TRANSLATION				
Frame Grabber Board	1	DT3852B-20	$ 3,805.20	$ 3,805.20
DAYTON				
Generator (5000W)	1	3W795B-20	$ 1,078.69	$ 1,078.69
HAMILTON CASTER				
Wheel Assembly	2	S-70188-4SL-FCB	$ 361.68	$ 723.36
K&K WELDING				
Canopy	1	Custom	$ 950.00	$ 950.00
LOVEJOY				
Shaft Couplings	1	U62X5/8	$ 80.00	$ 80.00
PELCO				
1/2" Camera	2	PCHM575	$ 468.00	$ 936.00
Camera Lens	2	TV6EX-1	$ 236.00	$ 472.00
SKB				
Equipment Racks	2	544635	$ 319.88	$ 639.76
SPEEDAIRE				
Air Compressor	1	4BB20	$ 549.99	$ 549.99
SPILLAR HITCHES				
Drop Hitch (3-inch)	1	Custom	$ 400.00	$ 400.00
TOUCH TECHNOLOGY				
Touch Screen Monitor	1	GT	$ 844.00	$ 844.00
TRIPPLITE				
Line Conditioner	2	LCR-2400	$ 574.00	$ 1,148.00
UNIVERSITY COMPUTER				
Computer, Monitor, Keyboard, DOS, Windows Software, Etc.	1	Real PC 486-66CV	$ 1,970.00	$ 1,970.00
Miscellaneous Items			$ 500.00	$ 500.00
Approximate Total System Cost				$ 54,240.76

Table 12.1 Component Costs for the ARMM System

Item	Qty.	Model #	Cost Each	Total Cost
AEROTECH				
410 oz-in motor	2	1410LT-MS01-E500LD	$ 1,000.00	$ 2,000.00
960 oz-in motor	1	1960LT-MS01/E500	$ 1,435.00	$ 1,435.00
Motion Controller	1	U500ULTRA	$ 2,390.00	$ 2,390.00
Three Axis of Amplifiers	1	BB501/X4-BA20-80/X4-BAC/TVO.4	$ 5,290.00	$ 5,290.00
Intercomn. Cables	1	OP500-12	$ 195.00	$ 195.00
Feedback Cable	3	DC-MSO	$ 250.00	$ 750.00
Double Shielded Cable	1	ECX413	$ 800.00	$ 800.00
BEARINGS INC.				
Swiveling End Bearings	8	TRE12	$ 19.66	$ 157.28
Pillow Block Bearings	8	P2BSC012	$ 22.06	$ 176.48
CYBO ROBOTS				
XY-Table	1	Custom	$ 26,950.00	$ 26,950.00
DATA TRANSLATION				
Frame Grabber Board	1	DT3852B-20	$ 3,805.20	$ 3,805.20
DAYTON				
Generator (5000W)	1	3W795B-20	$ 1,078.69	$ 1,078.69
HAMILTON CASTER				
Wheel Assembly	2	S-70188-4SL-FCB	$ 361.68	$ 723.36
K&K WELDING				
Canopy	1	Custom	$ 950.00	$ 950.00
LOVEJOY				
Shaft Couplings	1	U62X5/8	$ 80.00	$ 80.00
PELCO				
1/2" Camera	2	PCHM575	$ 468.00	$ 936.00
Camera Lens	2	TV6EX-1	$ 236.00	$ 472.00
SKB				
Equipment Racks	2	544635	$ 319.88	$ 639.76
SPEEDAIRE				
Air Compressor	1	4BB20	$ 549.99	$ 549.99
SPILLAR HITCHES				
Drop Hitch (3-inch)	1	Custom	$ 400.00	$ 400.00
TOUCH TECHNOLOGY				
Touch Screen Monitor	1	GT	$ 844.00	$ 844.00
TRIPPLITE				
Line Conditioner	2	LCR-2400	$ 574.00	$ 1,148.00
UNIVERSITY COMPUTER				
Computer, Monitor, Keyboard, DOS, Windows Software, Etc.	1	Real PC 486-66CV	$ 1,970.00	$ 1,970.00
Miscellaneous Items			$ 500.00	$ 500.00
Approximate Total System Cost				$ 54,240.76

12.1 Replacement Equipment List

Aerotech

Model 1410-MS01/E500 LD (410 oz-inch motors -2 qty.)

Model 1910-MS01/E500 LD (960 oz.-inch motor)

Model U500 (Ultra Motion Controller)

Model DR500R-4-A-80/X3-DS-1600C (Motor Drivers)

Model BB501/X4-BA20-80/X4-BAC/TV0.4 (Amplifiers)

Model OP500-12 (Interconnection Cables)

Model DC-MSO (Motor Feedback Cable)

Bayside

Model NE42-003 (3:1 gear ratio gearhead)

Model MM42-045 (Adapter)

Bearings Inc.

End Bearings

Block Bearings

Bogen

Model # 3063 (camera mounts - 2 qty.)

Cone Drive

Model HU 15 (Gear Reducer)

Crafco Inc.

Model #SS60 (Propane melter)

Cybo Robots

2 axis trailer mounted rack & pinion slide table

Data Translation Inc.

Model DT3852B-20 (Frame Grabber Computer Board)

Dayton

Model #3W795B (Portable 5000W generator)

Everglide

bearing rails (Track Assembly)

bearings (Track Assembly)

Hamilton Castor & Mfg

Part #S-70188-4SL-FCB (Wheel assembly)

Igus

E-chain cable carriers

K&K Welding

Canopy from Automated Construction Laboratory blueprints

Lovejoy

Model # L070X5/8HUB3/16 (Shaft Coupling end)

Model # L070X3/4HUB3/16 (Shaft Coupling end)

Model # L070SOX (Spider)

Model # U62X5/8 (Shaft Coupling)

Microsoft

Windows 3.1 software

Norgren

Model # B07 (filter/regulator)

Pelco

Model PCHM575 (1/2" monochrome camera - 2 qty.)

Model # TV6EX-1 (camera lens - 2 qty.)

SKB

12-space shock mounted enclosures (2 qty.)

Speedaire

Model 4B220 (Air Compressor)

Spillar Custom Hitches

Drop hitch

Texaco Lubricants Co.

Code # 00702 Regal Oil R&O 68 (Turbine Oil)

Texas Department of Transportation

Specification #070-66-04 trailer

Touch Technology

GT-Touch Screen Monitor

Tripplite

Model LCR-2400 (Rack mount line conditioners-2 qty.)

University Computer Store

486DX-2 (Computer, video screen and keyboard)

Winsmith (Model 917 (Speed Reducer))

12.2 Equipment Manufacturers Index

Aerotech, Inc.

101 Zeta Drive

Pittsburgh, PA 15238-2897

Telephone (214) 713 - 8858

Fax (214) 713 - 8852

Beckett Corporation

P.O. Box 1289

Elyria, OH 44036

Briggs & Stratton Corporation

Toll Free 1 - 800 - 233 - 3723

Crafco, Inc.

6975 West Crafco ay

Chandler, AZ 85226

Telephone (602) 276 - 0406

Telephone (800) 528 - 8242

Cybo Robots

2701 Fortune Circle East

Indianapolis, Indiana 46241

Telephone (317) 484 - 2926

Data Translation

100 Locke Drive

Marlboro, MA 01752 - 1192

Telephone (800) 525- 8528

Dayton Electric Mfg. Co.

5959 West Howard Street

Niles, IL 60714

Hamilton Caster & Mfg.

1637 Dixie Hwy

Hamilton, OH 45011

Telephone (513) 863 - 3300

Fax (513) 863 - 5508

Kohler Corporation

Engine Division

Kohler, Wisconsin 53044

Telephone (800) 544 - 2444

Norgren

Littleton, CO

Telephone (303) 794 - 2611

Fax (303) 795 - 9487

Pelco

300 West Pontiac Way

Clovis, CA 93612 - 5699

Telephone (800) 289 - 9100

Fax (800) 289 - 9150

DataFax (800) 289 - 9108

Roper Pump Company

P.O. Box 269

3475 Old Maysville Rd

Commerce, GA 30529

Telephone (706) 335 - 5551

Fax (706) 335 - 5505

SKB

931 Chevy Way

Medford, OR 97524

Telephone (800) 776 - 5173

Fax (503) 772 - 9723

SpeedAire

Manufactured by Dayton Electric Mfg Co.

Spillar Custom Hitches, Inc.

9204 United

Austin, TX 78759

Telephone (512) 837 - 7142

Texaco Lubricants Co.

Division of TRMI

P.O. Box 52332

Houston, TX 77052

Emergency (914) 831 - 3400

General (914) 838 - 7204

Technical (914) 838 - 7509

dba University Computer Store

305 W. Martin L. King Jr. Blvd.

Austin, TX 78701

Telephone (512) 476 - 6788

Fax (512) 474 - 2532

Winco Inc.

225 South Cordova Avenue

Le Center, Minnesota 56057

Telephone (800) 433 - 8123

Fax (612) 357 - 4857

12.3 Descriptive Photographs and Drawings

The following photographs and drawings describe some important aspects of the Automated Road Maintenance Machine. Figures 12.1 through 12.4 show the equipment train from different views in drawings and photographs, with Figure 12.4 showing a closer view of the Crafco sealant melter connection to the xy table. Figures 12.5 and 12.6 show the sealant melter that was provided to the Automated Construction Laboratory on loan from Crafco, Inc.

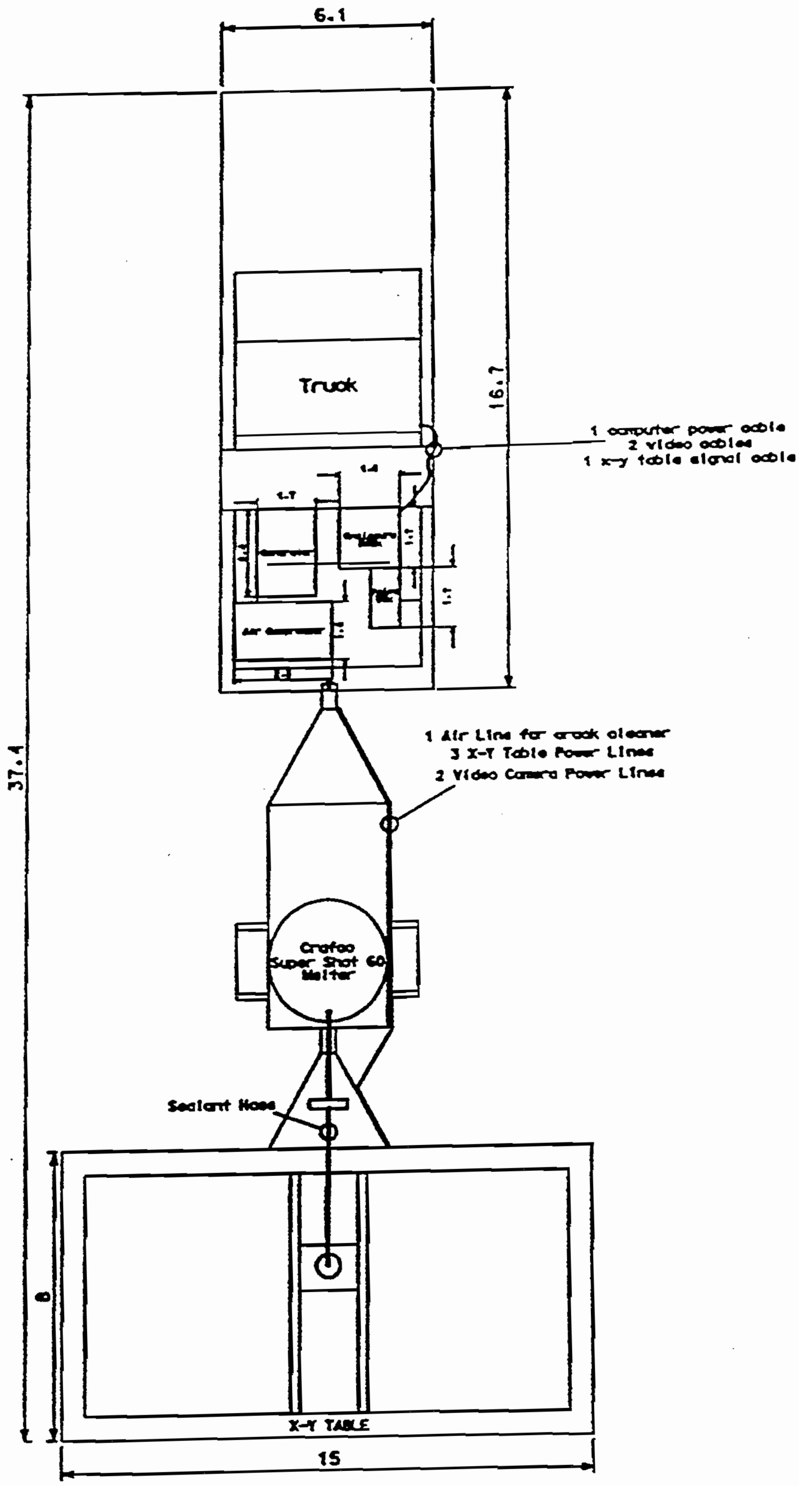


Figure 12.1 ARMM Equipment Train (Top View)

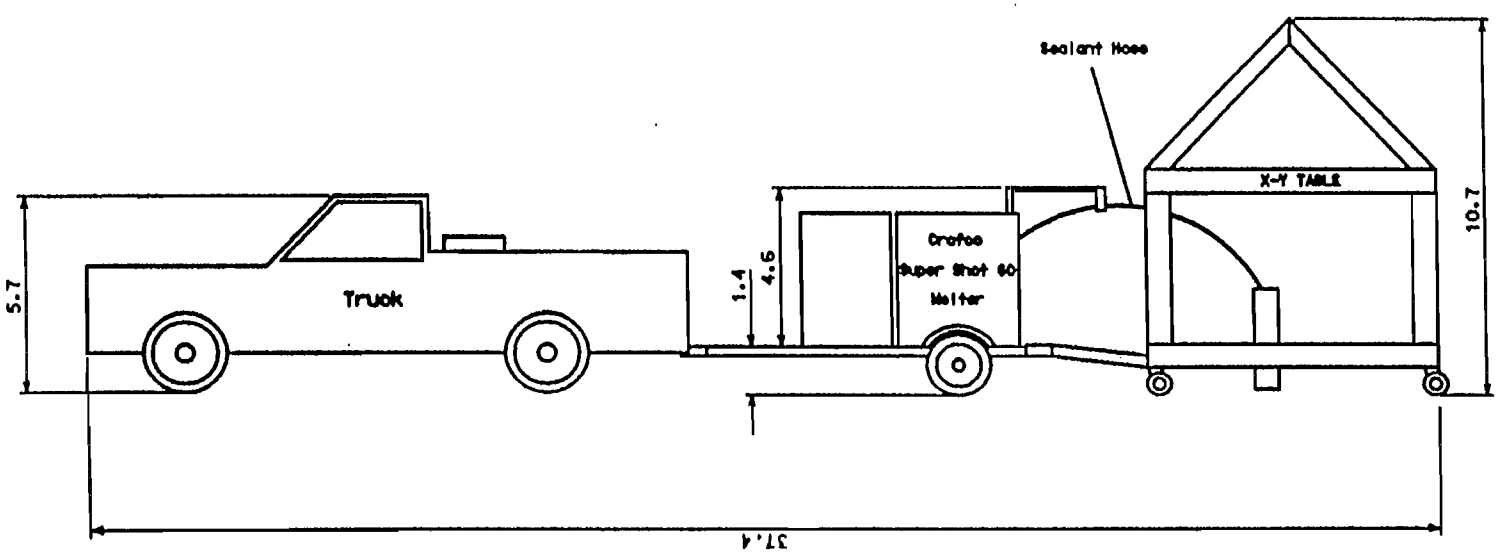


Figure 12.2 ARMM Equipment Train (Side View)

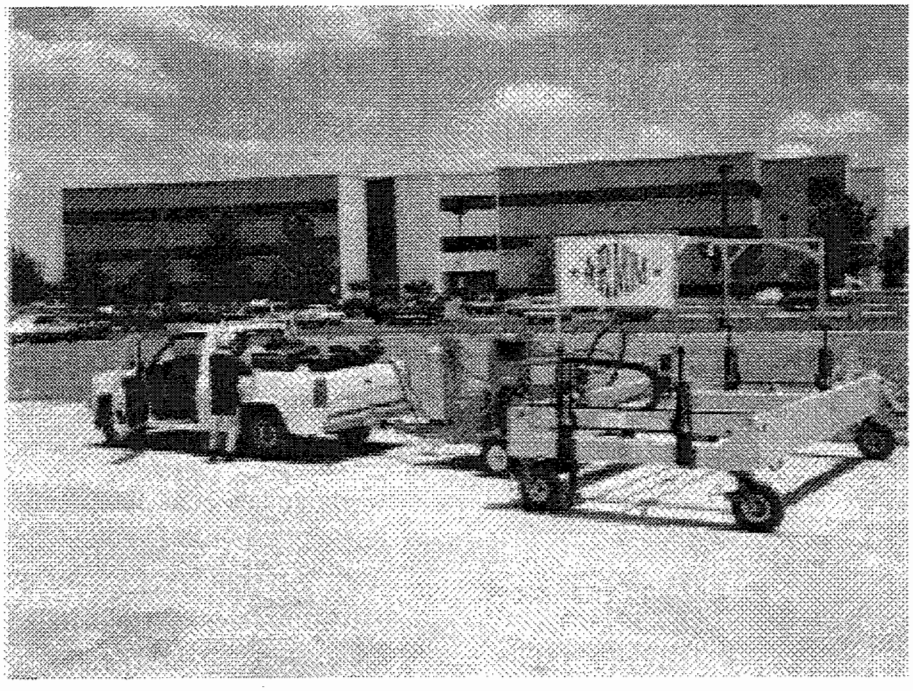


Figure 12.3 Photograph of the ARMM Equipment Train

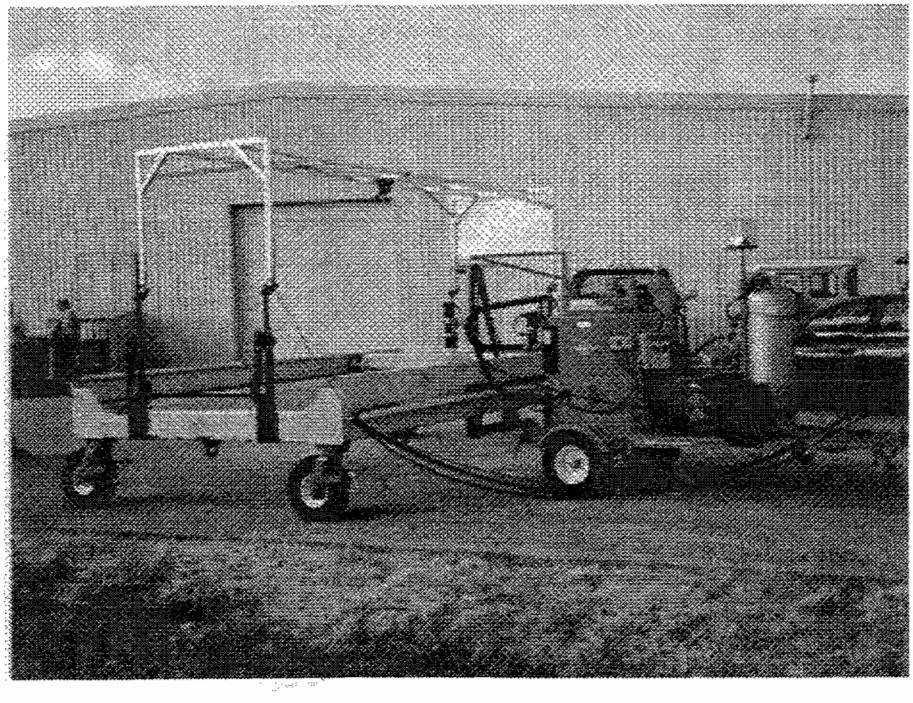


Figure 12.4 Photograph of the ARMM Equipment Train Showing Melter Connection

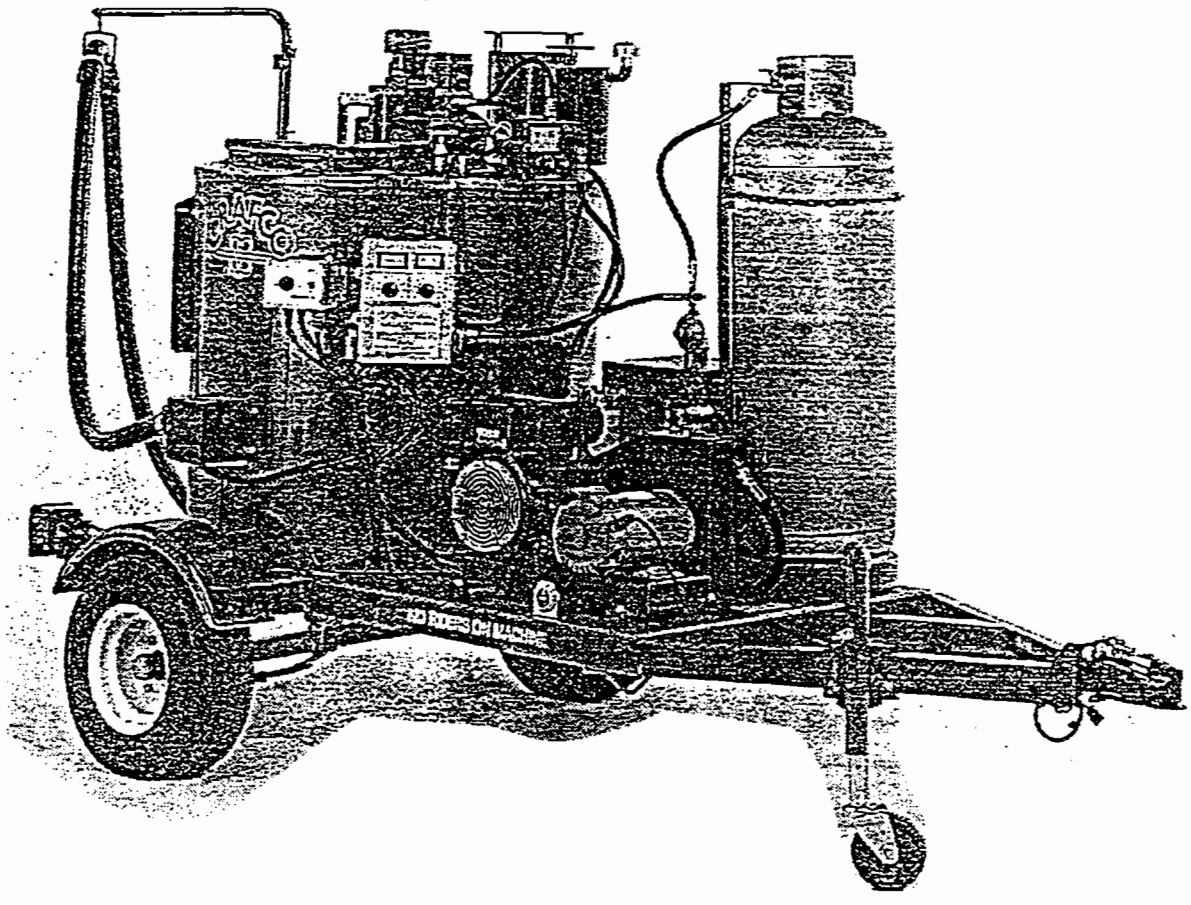


Figure 12.5 Crafco Super Shot 60 Propane Melter

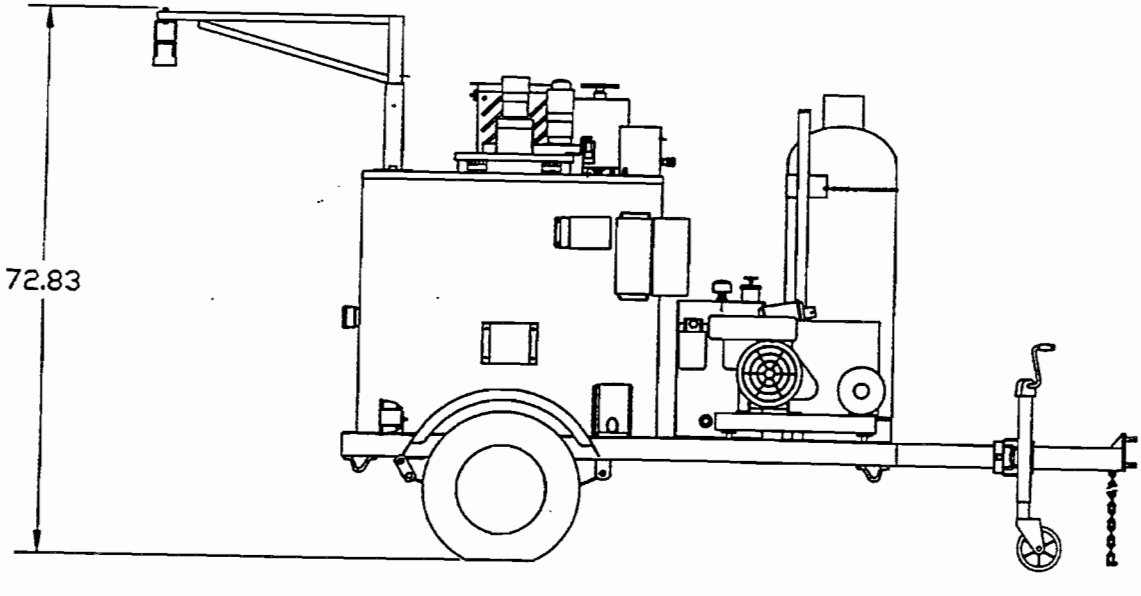


Figure 12.6 Crafco Super Shot 60 Propane Melter Drawing (Side View)

12.4 Manufactured Components of The ARMM

The Automated Construction Laboratory had several components custom made for the ARMM project. Shown in Figure 12.7 is a close up of the turret on the machine. The turret houses the air and sealant hose as it moves in the xy table frame. Figures 12.8 and 12.9 are the drawings of the turret. Cable hard connects were also manufactured to enable the operators to disconnect the xy-table from the truck when transporting the ARMM. Figures 12.10 and 12.11 show these drawings.

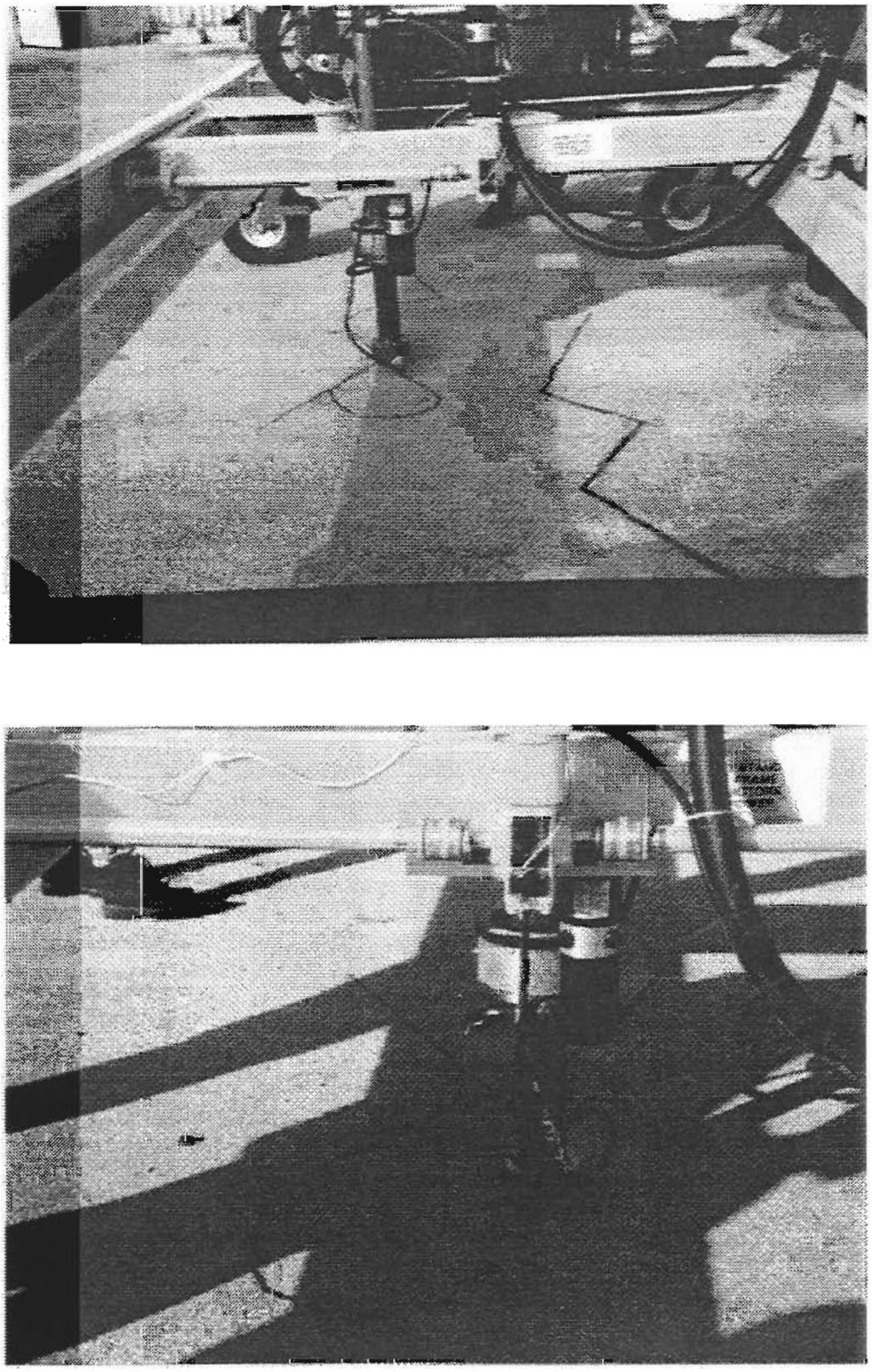


Figure 12.7 The turret mechanism on the ARMM as it proceeds down a crack

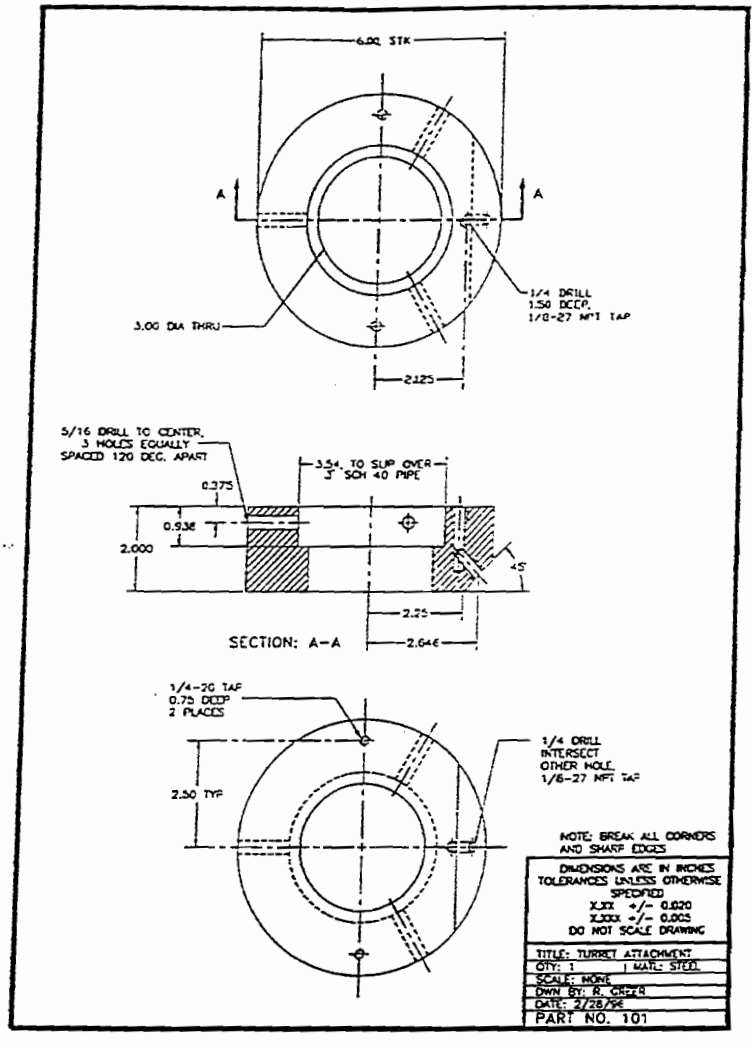


Figure 12.8 Turret Attachment Drawings -Manufactured by the Automated Construction Laboratory

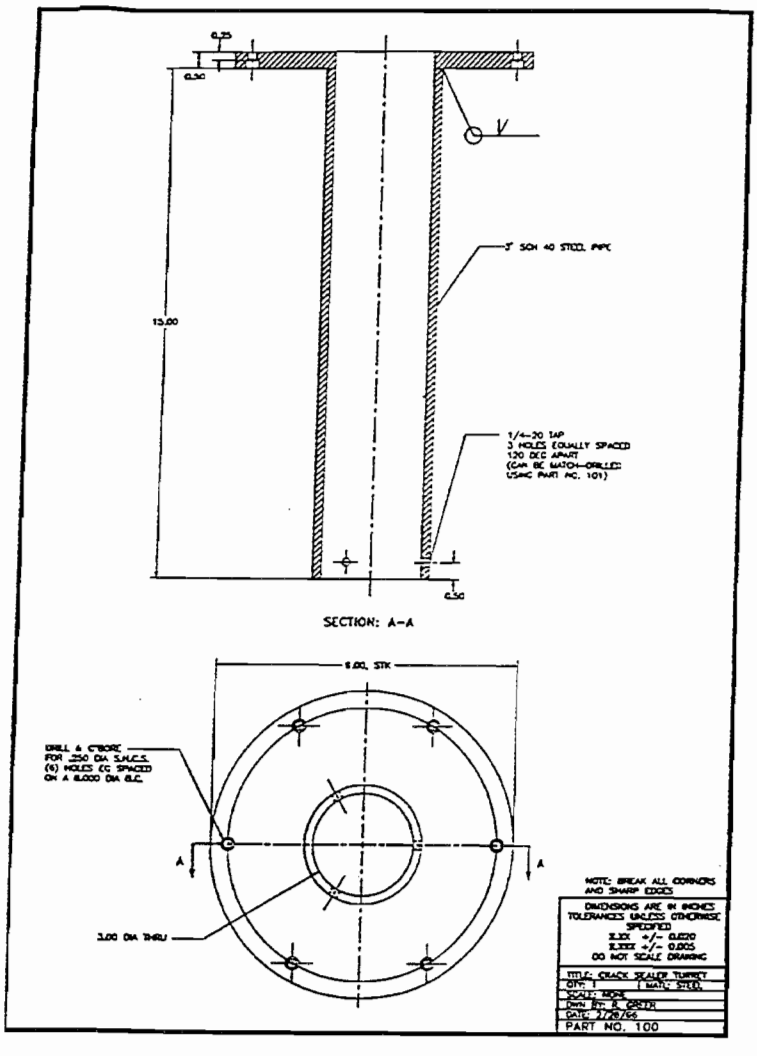
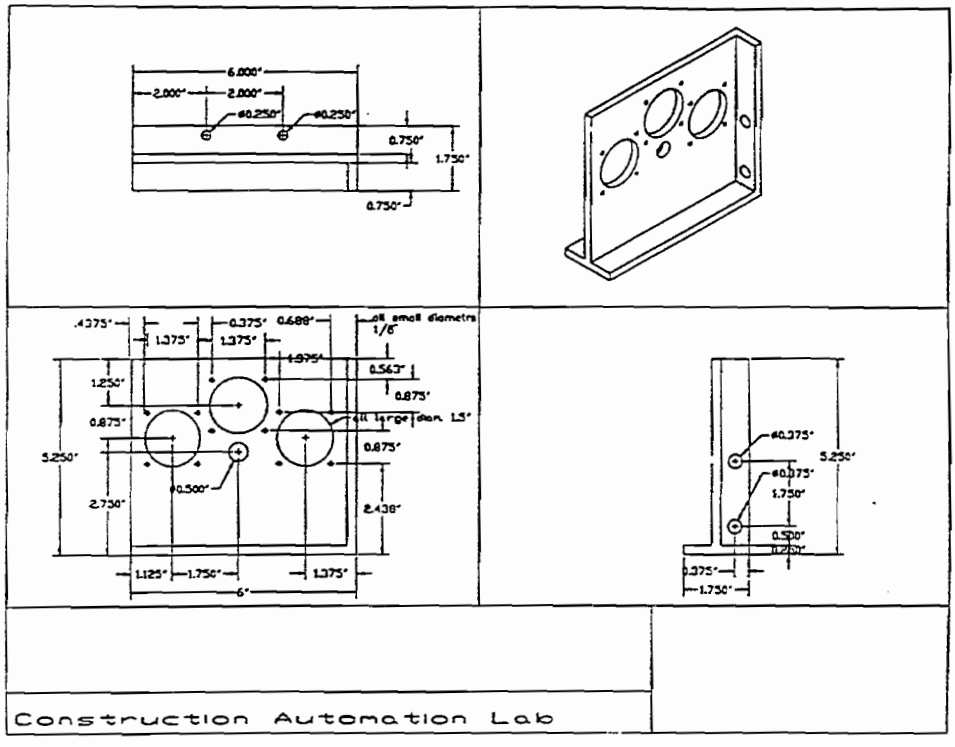


Figure 12.9 Turret Drawings -Manufactured by the Automated Construction Laboratory



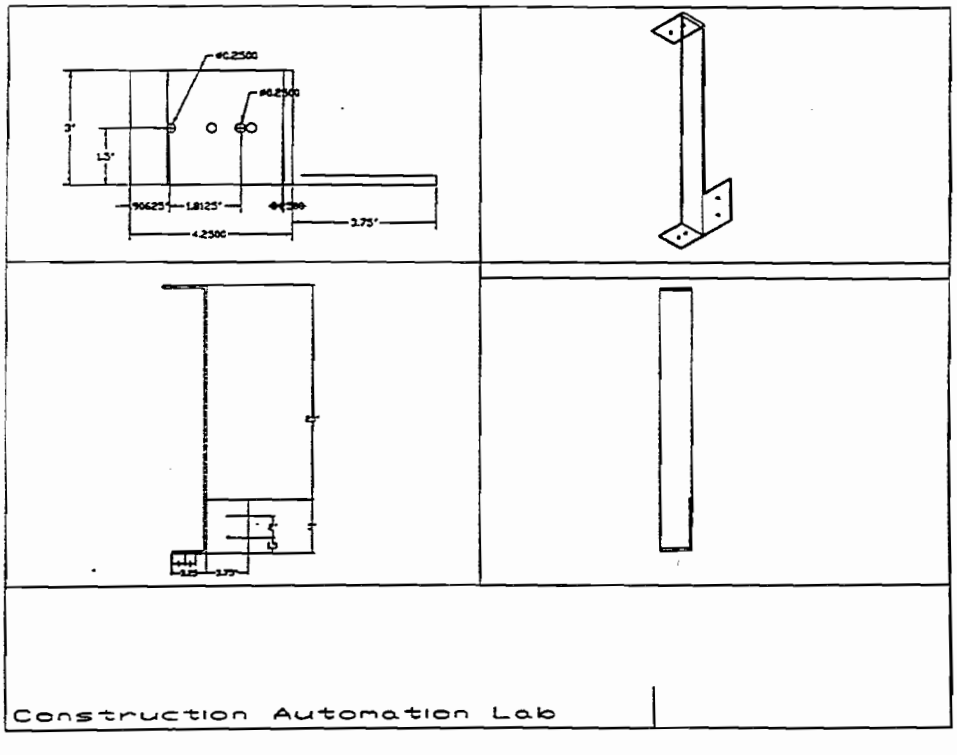


Figure 12.10 Cable Hard Connect Drawings -Manufactured by the Automated Construction Laboratory

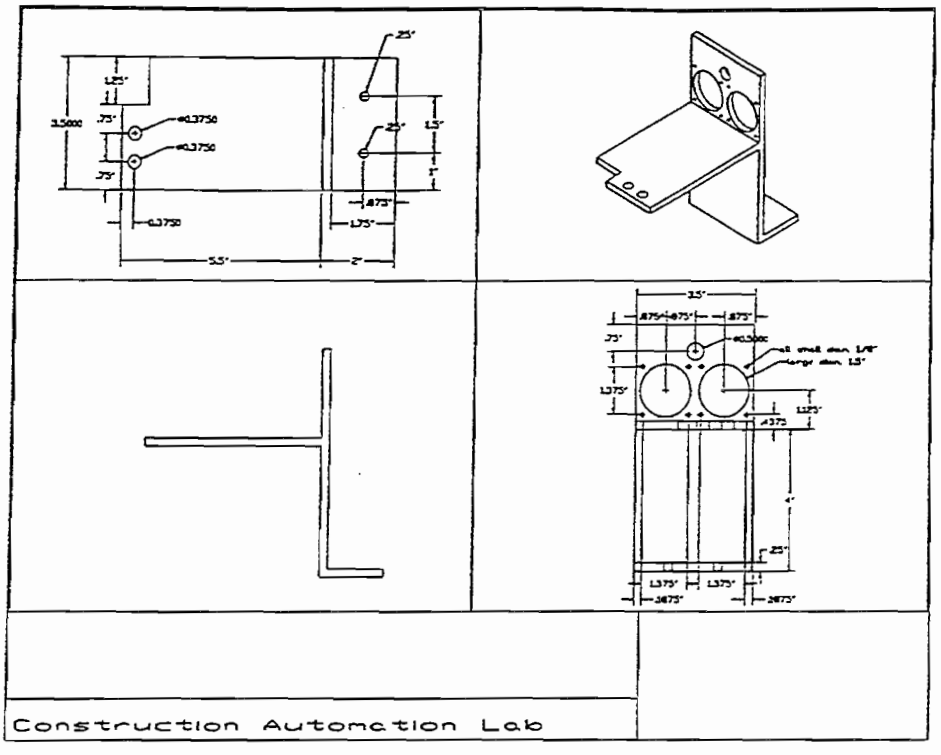


Figure 12.11 Cable Hard Connect Drawing - Manufactured by the Automated Construction Laboratory

12.5 Characteristics of the ARMM

Workspace

The workspace for filling cracks is limited by three things. First and foremost the main frame of the table which has dimensions of eight feet long by 15 feet wide. This would give a theoretical workspace area of 120 feet squared. The width and length are relative to the direction the table travels down the road, thus making 15 feet the width of the machine. Next, the workspace is limited by the traverse distance of the through trolley. The trolley is limited by its axle, wheels on the right side, and the extruding cable tray on the left side. This reduces the possible width of the workspace to 12.1125 feet. The length of the workspace is limited by the traverse distance of the inner trolley where the wand will actually be located. This distance is restricted by the size of the main trolley thus limiting the length of the workspace to 5.95 feet. Therefore, the final actual area of the workspace is 5.95 ft x 12.1125 ft which equals 72.06 feet squared.

Weight Estimate

The following is an estimate of the weight of the ARMM by adding the weights of its components. The weight of each component was found by calculating the volume of the component, adding them, and multiplying the total volume by the specific weight. Some components were weighed on a scale.

Object	**Weight (lb.)**
XY-Table	3042.6
Canopy Frame	285
Cable Holders	51
Pneumatic Wheels	215
Wheel Mounts	40
Motors (20 lb. each)	60
Approx. Total Weight	3693.6

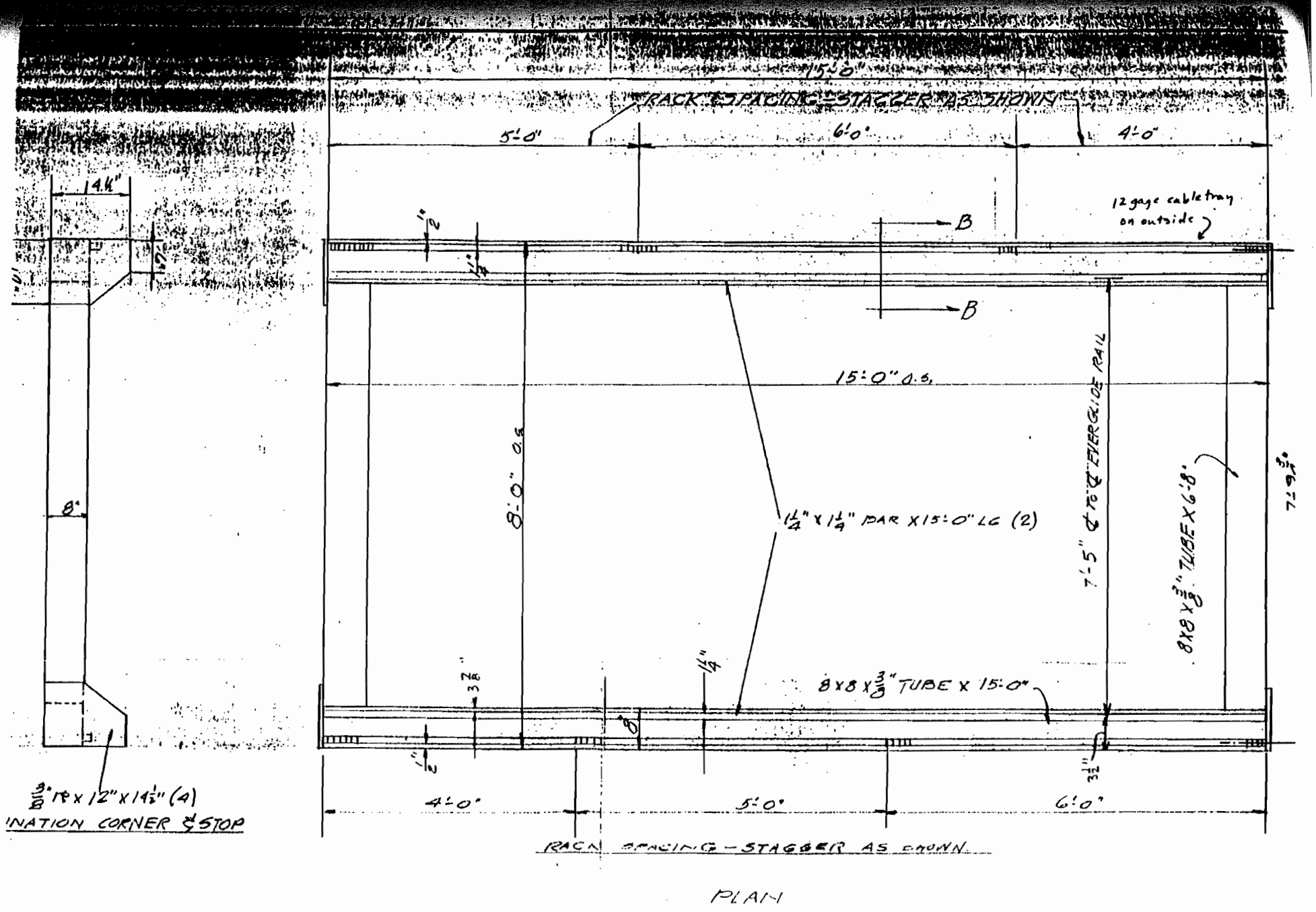
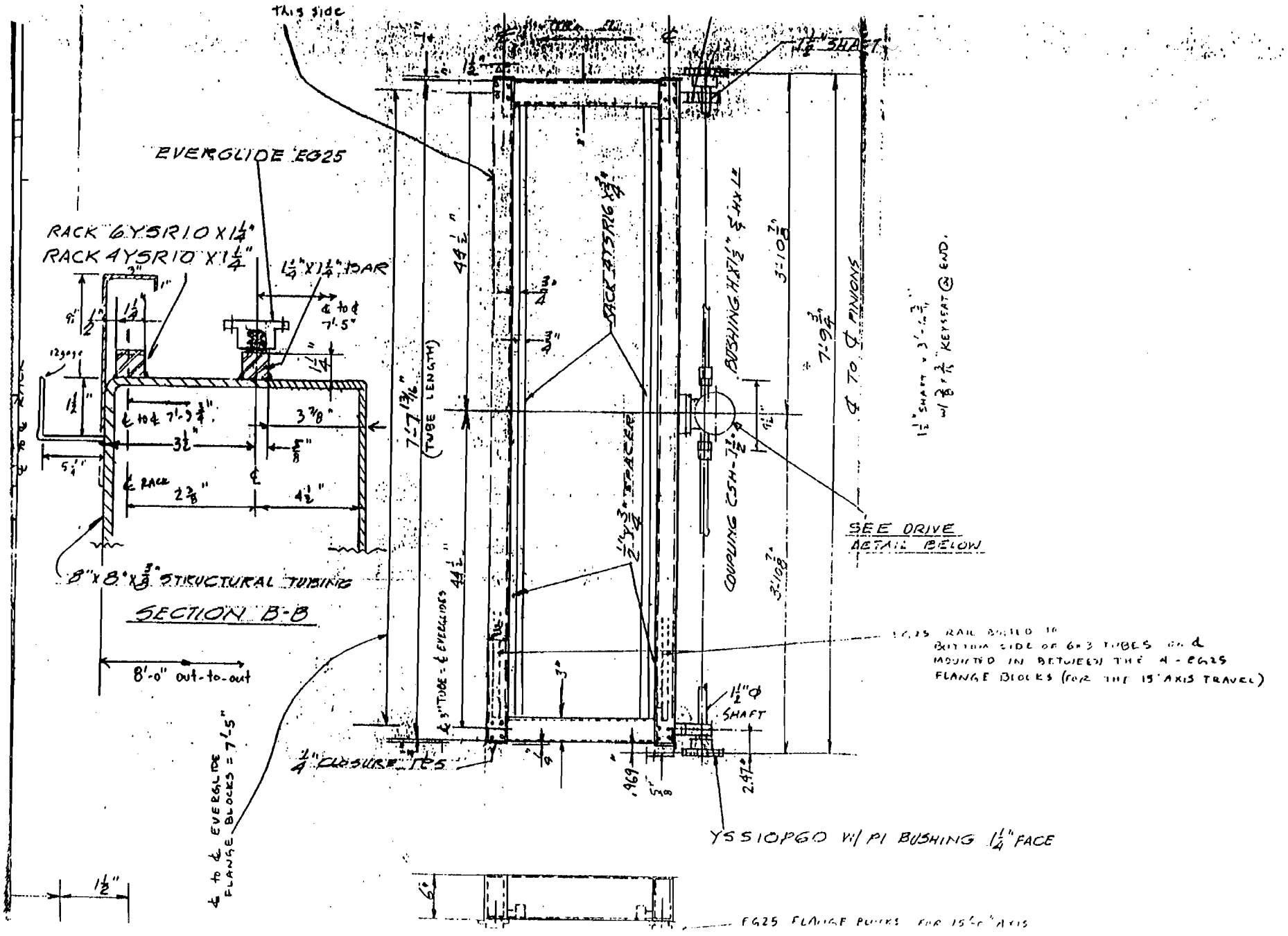
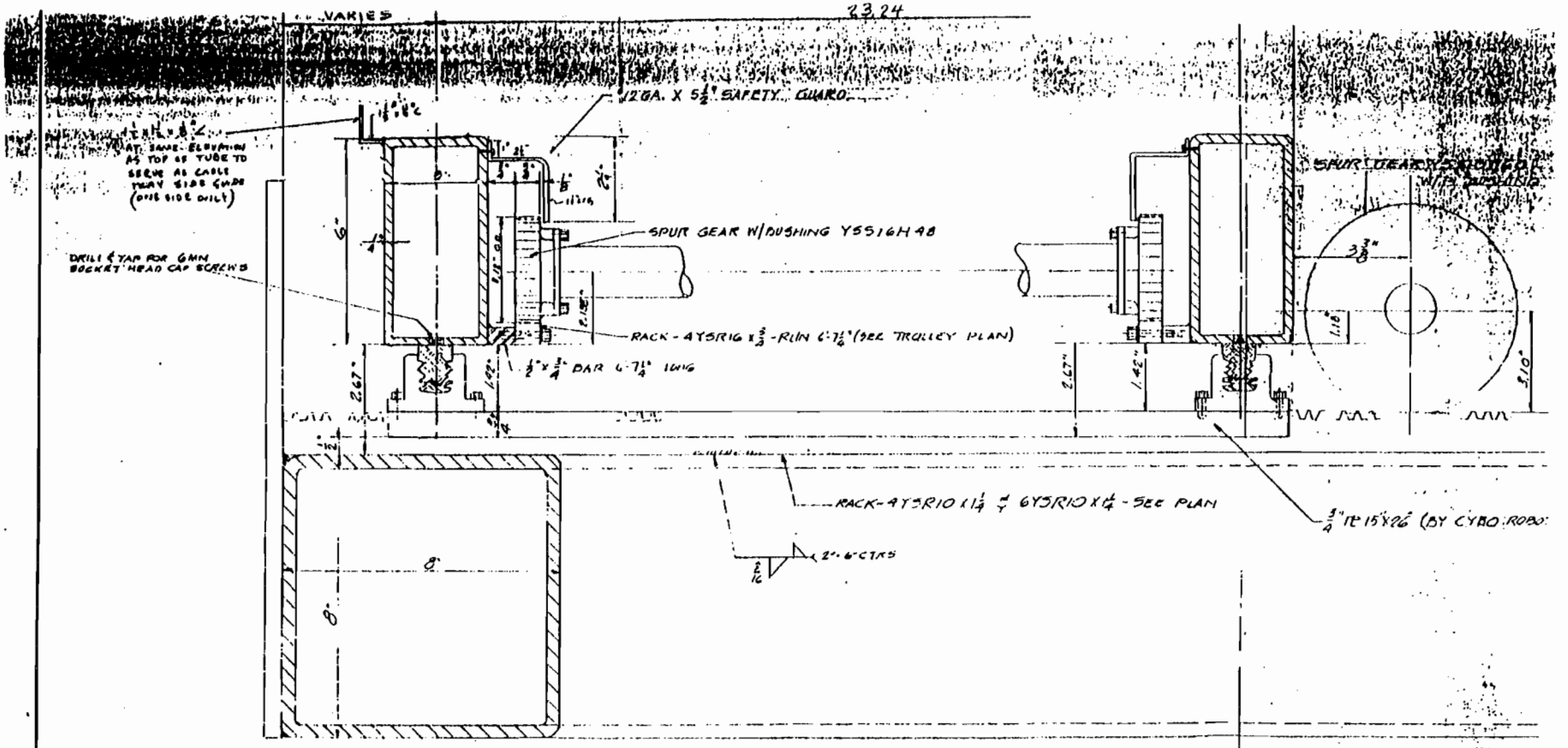
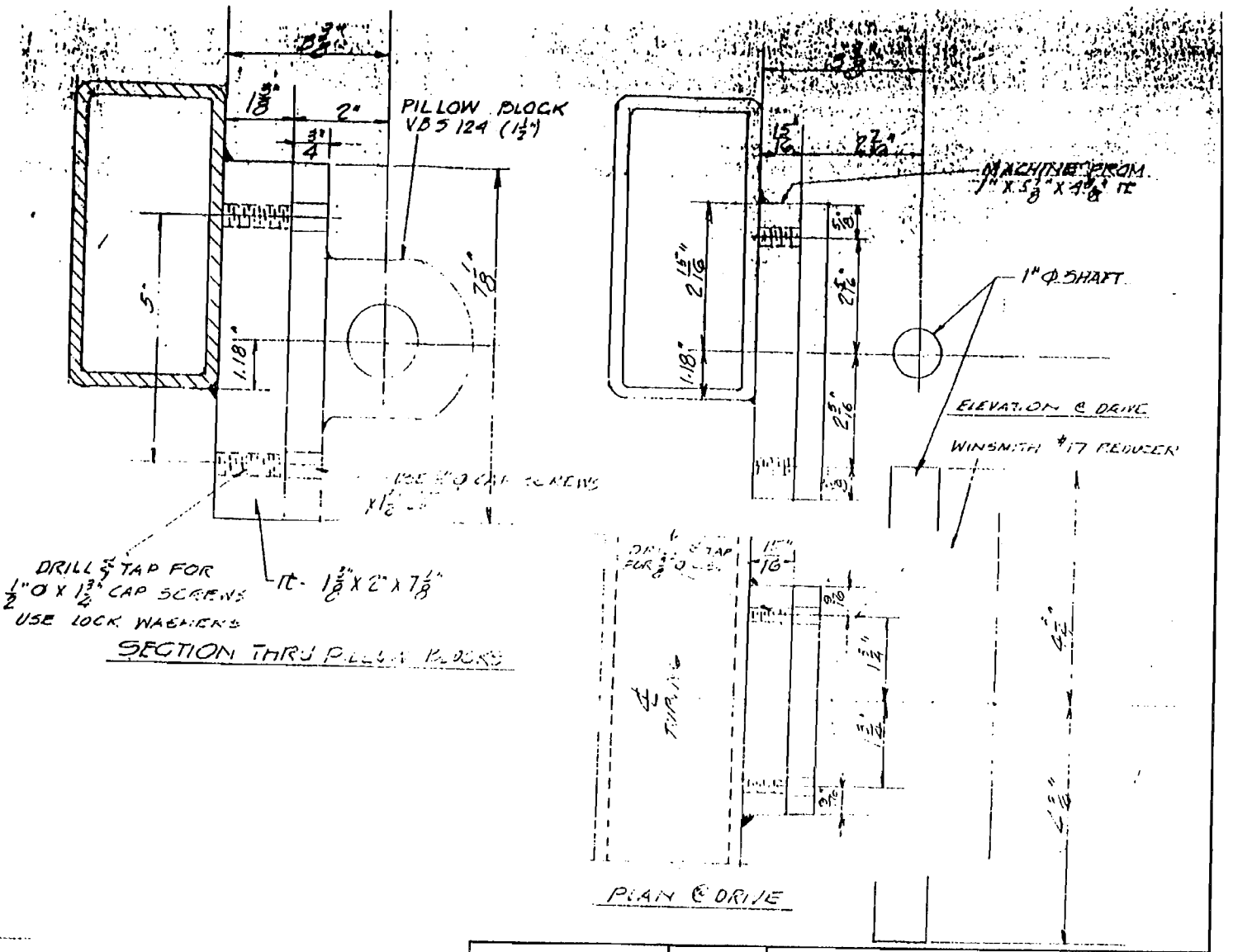


Figure 12.12 Table Drawings


This side
EVERGLIDE EG25
RACK 6YSR10 X 1¼"
RACK 4YSR10 X 1¼"
1½"X1¼" BAR
8"x8"x⅜" STRUCTURAL TUBING
SECTION B-B
8'-0" out-to-out
¢ to ¢ EVERGLIDE FLANGE BLOCKS = 7'-5"
¼" CLOSURE PLATE
1½" SHAFT
44½"
7'-7 13/16" (TUBE LENGTH)
RACK 2YS16 X ¾"
2½"x3½" TRACER
COUPLING CSH-7½"
BUSHING HXT 1½" x ¾" x 7"
3'-10 7/8"
7-9½"
¢ TO ¢ PINIONS
1½" SHAFT x 3'-1 3/4" w/ ⅜" x ¾" KEYSEAT @ END.
EG25 RAIL BOLTED TO BOTTOM SIDE OF 6x3 TUBES ON ¢ MOUNTED IN BETWEEN THE 4 - EG25 FLANGE BLOCKS (FOR THE 15' AXIS TRAVEL)
1½"Φ SHAFT
YSS10PG0 W/ PI BUSHING 1¼" FACE
EG25 FLANGE BLOCKS FOR 15'-0" AXIS



SECTION A-A THRU TROLLEY



SECTION THRU PILLOW BLOCKS

TOLERANCES (EXCEPT AS NOTED)	TERRE HAUTE EVANSVILLE	HYBRID ROBOTICS	INDIANAPOLIS FT. WAYNE
DECIMAL .		SCALE VARIES	DRAWN BY C
FRACTIONAL .	TITLE CART - AXIS 2 & REF. UTCE2-1 FOR CYBO ROBOTS		APPROVED BY
ANGULAR .	DATE 11-21-95	DRAWING NUMBER 167[illegible]-0004-1	

REV. 1-2-96 JW
REV. 11-25-95 JW

13. 0 Implementation Recommendations

A project is proposed for implementation. The proposed project would involve field trials at five or more locations around the state, evaluations by maintenance personnel, assembling of focus groups for market analysis, pursuit of news media coverage, negotiation with key vendors concerning establishment of a commercialization entity, analysis of productivity, publications, and production of user documentation. Details are provided below.

Evaluation would involve several steps and procedures including field trials, evaluations by maintenance personnel, market focus group analysis, key vendor input, and detailed productivity analysis.

Field trials would be conducted in five or more maintenance districts. Objectives of the field trials would include collection of productivity data, demonstration of the technology, and acquisition of feedback from maintenance personnel. Secondary objectives would include proof testing the equipment under real working conditions, acquiring video footage, and gaining field experience. Feedback would result from informal discussions, formal interviews, and focus group discussions. The schedule of each field trial would be:

- 1 day for transport and setup
- 2 days for field demonstrations and productivity analysis
- 1 day for focus group discussions and review
- 1 day return transport and demobilization

Time between field trials would be dedicated to productivity data analysis, survey data analysis, documentation, publication, publicizing, local demonstrations, and equipment repair and modification.

Productivity analysis would follow the form of previous studies by the UT team that have been widely reviewed and accepted. Economic analysis would be conducted from several perspectives including the district, the state, the country, the system manufacturer, and the contractor. Market analysis would focus on Texas but would project results nation wide based on national statistical databases.

Vendors would be convened at one or more demonstrations to discuss and negotiate commercialization. Technology will be transferred via demonstrations, distribution of documentation, and personal explanations.

Automated pavement crack sealing is a leading edge technology. Broad applications of the technology are anticipated including automated routing, joint sealing, message painting, pothole filling, and marker placement. Crafco has been involved for two years as a private partner and will likely continue its participation in implementation activities.

References

American Association of State Highway and Transportation Officials, *AASHTO Maintenance Manual*, American Association of State Highway and Transportation Officials, 444 N. Capital St. NW,Washington, D.C.,1987.

American Society of Testing and Materials, "ASTM D3405-78, Standard Specification for Joint Sealants, Hot Poured, For Concrete and Asphalt Pavements", 1980 Annual Book of Standards, Vol 15, Pg. 911-13.

Asphalt Institute, Asphalt in Maintenance , Manual Series No. 16 (MS-16), Pg. 15-24, 67-87, 1983.

Drozdek, A. and Simon, D. L., "Data Structures in C", PWS publishing company, Boston, MA., 1995, pp. 18, 58-65, 267-306.

Federal Highway Administration, *Highway Statistics*, U.S. Department of Transportation, Washington, D.C., 1990.

Gharpuray, D., and Haas, C., "Comparison of Multi-Sensing Methods for the Detection of Cracks in Pavement Surfaces," Proceedings of the ASCE Pacific Rim TransTech Conference, Seattle, Washington, July, pp. 425-429.

Greer, R., Haas, C., Gibson, G., Traver, A., and Tucker, R., "Advances in Control Systems for Construction Manipulators", Proceedings of the 13th International Symposium on Automation and Robotics in Construction, pp.615-624, Japan Robot Association, Toyko, 1996.

Haas, C., "Evolution of an Automated Crack Sealer: A Study in Construction Technology Development", Automation in Construction 4, pp. 293-305,1996.

Haas, C., Hendrickson, C., McNeil, S., and Bullock, D., " A Field Prototype of a Robotic Pavement Crack Sealing System," Proceedings of the 9th International Symposium on Automation and Robotics in Construction, pp. 313-322, Tokyo, Japan, June 1992.

Haas, C., Hendrickson , C., McNeil, S., Bullock, D., Peters, D., Grove, D., Kenneally, K., and Wichman, S., "Perception and Control Automated Pavement Crack Filling," Proceedings of the Second International Conference on Applications of Advanced Technologies Transportation Engineering, pp. 66-70, Minneapolis, MN, August 1991.

Haas, C., Hendrickson, C., McNeil, S., "A Design for Automated Pavement Crack Sealing," Preparing for Construction in the 21st. Century, ASCE, pp. 222-227, 1991.

Haas, C., Hendrickson, C., "Computer-Based Model of Pavement Surfaces," Transportation Research Record, pp. 91-98, Washington, D.C., No. 1260, 1990.

Haas, C., Shen, H., Phang, W.A., and Haas, R., "An Expert System for Automation of Pavement Condition Inventory Data", Proceedings, North American Pavement Management Conference, Toronto, September 1984.

Hsieh, T., "Costs and Benefits of Automated Road Maintenance", Department of Civil Engineering, University of Texas at Austin, 1993.

Kim, Y.S., "Path Planning for an Automatic Pavement Crack Sealer," Thesis, The University of Texas at Ausitn, Austin, TX. 1995.

Kim, Y.S., "Path Planning," Research Summary in Construction Automation Group of the University of Texas at Ausitn, Austin, TX. 1996.

Kim, Y.S. and Greer, R., "Graphical Controls," Research Summary in Construction Automation Group of the University of Texas at Ausitn, Austin, TX. 1996.

Maeda, J., "Development and Application of the SMART System", Automation and Robotics in Construction XI, D.A. Chamberlain ed., Elsevier, Amsterdam, 1994.

Malek, G.J., "Methods, Practices, and Productivity Study of Crack Sealing/Filling in Texas", Thesis, Department of Civil Engineering, University of Texas at Austin, May 1994.

McNeil, S., "An Analysis of the Costs and Impacts of the Automation of Pavement Crack Sealing", Presented at the 6th World conference on Transportation Research, Lyon, France, June 1992.

Memmot, J., and Dudek, C., "A Model to Calculate the Road User Costs at Work Zones", Research Report 292-1, Texas Transportation Institute, September 1982.

Osmani, A., "A Model for Evaluating Automation in Road Maintenance", Thesis, Department of Civil Engineering, The University of Texas at Austin, 1994.

Peshkin, D.G., Romine, R.A., Smith, K.L."Maintenance of Cracks in Asphalt Concrete Surfaced Pavements", Strategic Highway Research Program, National Research Council, Washington, D.C., 1992.

Seshadri, P., Solminihac, H.E., and Harrison, R., "Modification of the QUEWZ Model to Estimate Fuel Costs and Tailpipe Emissions", Proceedings of the 72nd Annual Meeting of the Transportation Research Board, Washington, D.C., January 1993.

Skibniewski, Miroslaw and Chris Hendrickson, "Automation and Robotics for Road Construction and Maintenance," *ASCE Journal of Transportation Engineering*, May 1990.

Tucker, R.L., Peterson, C., Meyer, J., and Simonson, T., "A Methodology for Identifying Automation Opportunities in Industrial Construction", Source Document 56, Construction Industry Institute, Austin, Texas, September 1990.

Velinsky, S.A., "Fabrication and Testing of an Automated Crack Sealing Machine", National Research Council, SHRP-H-659, Washington, D.C. 1993.

Appendix A.1 Sample District Survey

THE UNIVERSITY OF TEXAS AT AUSTIN
CRACK SEALING AND FILLING SURVEY

DISTRICT NO.: 7 NAME: Carrol Templeton

DATE: 2-18-93 TITLE: District Roadway Maint. Supt. II

PHONE NO.: 915-944-1501 ext. 214
STS 248-5214

Methods and Materials

1) Methods of crack sealing and filling performed within your district.

Hot filling 100 %

Cold filling _____ %

Rout and fill _____ %

2) Type of crack sealing and filling material used by manufacturer. (e.g., CRAFCO)

In House CRAFCO

By contractor (if known) CRAFCO

3) Crack sealing and filling material for contracts furnished by:

Department 100 %

Contractor _____ %

Appendix A.1 Sample District Survey

Outside Contracting

4) Amount of crack sealing and filling work contracted out. 72 %

5) Unit basis for estimates and approximate unit price.

Per linear foot of cracking	____%	$/lf	____
Per lane mile	72 %	$/lm	614.00
Per pound of material	28 %	$/lb	.89
Per gallon of material	____%	$/gal	____

6) No. of different contractors used during the previous year. 1

7) Please indicate the number of workers that occupy the following positions on a typical crack sealing and filling work crew.

Foreman (crew leader)	1	
Drivers	2	
Crack cleaner (air jet)		1
Applicator (wands man)	1	
Squeegee operator	2	
Flaggers	0	

Other (please indicate activity and quantity)

_____	____
_____	____
_____	____

Appendix A.2 District Survey Results

Survey Questions / District	1	2	3	4	5	6	7	8	
1.) Sealing /Filling Method as a %									
Hot Applied Materials	100%	90%	99%	20%	100%	100%	100%	100%	
Cold Applied Materials		10%		80%					
Rout and Fill (Hot or Cold applied)			1%						
2.) Type of Material									
Department	Crafco	Crafco	Crafco	Crafco	Crafco	Crafco &	Crafco	Crafco	
		Emulsion				Koch			
Contracted	Crafco	Crafco		Kengo	N/A	N/A	Crafco	Crafco	
				Cold Pour					
3.) Material Furnished By:									
Department as a Percent	100%	100%	N/A	20%	N/A	N/A	100%	100%	
Contractor as a Percent	0%	0%	N/A	80%	N/A	N/A	0%	0%	

Survey Questions / District	9	10	11	12	13	14	15	16	
1.) Sealing /Filling Method as a %									
Hot Applied Materials	100%	100%	98%	100%	80%	100%	100%	50%	
Cold Applied Materials			1%					50%	
Rout and Fill (Hot or Cold applied)			1%		20%	0%		0%	
2.) Type of Material									
Department	Crafco	None	Crafco	N/A	Crafco	Crafco	Crafco &	Crafco &	
							Koch	Koch	
Contracted	Crafco	Crafco	Crafco	Rubber	Crafco	Crafco		Crafco &	
				Asphalt				Koch	
3.) Material Furnished By:									
Department as a Percent	100%	100%	100%	0%	100%	100%	100%	100%	
Contractor as a Percent	0%	0%	0%	100%	0%	0%			

Survey Questions / District	17	18	19	20	21	23	24	25	State Average
1.) Sealing /Filling Method as a %									
Hot Applied Materials	100%	100%	99%	100%		100%	100%	95%	88%
Cold Applied Materials			1%					5%	6%
Rout and Fill (Hot or Cold applied)								0%	1%
2.) Type of Material									
Department	Crafco	Crafco	Crafco	Crafco		Crafco		Crafco	
								Koch	
Contracted	Crafco &	Crafco	None			Crafco &	Crafco &	Kengo	
	Koch					Koch	Koch	Elf & Koch	
3.) Material Furnished By:									
Department as a Percent	100%	100%	100%				100%	N/A	57%
Contractor as a Percent			0%			100%		N/A	12%

Appendix A.2 District Survey Results

Survey Questions / District	1	2	3	4	5	6	7	8
4.) Amount of Contracting (%)	90%	20%	0%	80%	0%	0%	72%	75%
5.) Unit basis for estimate used (%) and Approximate unit price for:			N/A			N/A		
Per Linear Foot of Cracking (LFC)	100% $0.08/LF							
Per lane mile (LM)							72% $614 / LM	
Per Pound of Material (LB)				40% $0.66 /LB	100% $0.75 /LB		28% $0.89 /LB	100% $0.63/LB
Per Gallon of Material(GAL)		100% $10 /GAL		60% $11 /GAL				
6.) Number of contractors used in Previous Year	1		0	0	0	0	1	2

Survey Questions / District	9	10	11	12	13	14	15	16
4.) Amount of Contracting (%)	85%	100%	100%	100%	70%	30%	80%	90%
5.) Unit basis for estimate used (%) and Approximate unit price for:								
Per Linear Foot of Cracking (LFC)			100% $0.07 /LF	100% $0.53 /LF			100% $0.08 /LF	100% $0.06 /LF
Per lane mile (LM)	100% $280 /LM							
Per Pound of Material (LB)		100% $0.51 /LB				100% $0.60 /Lb		
Per Gallon of Material(GAL)					100% $7 /GAL			
6.) Number of contractors used in Previous Year	3	2	2	3	2	1	3	3

Survey Questions / District	17	18	19	20	21	23	24	25	State Average
4.) Amount of Contracting (%)	95%	80%	0%	0%		62.40%	90%	0%	57%
5.) Unit basis for estimate used (%) and Approximate unit price for:									
Per Linear Foot of Cracking (LFC)									22% $0.16/LF
Per lane mile (LM)		$388 /LM						$660 /LM	7.50% $485/LM
Per Pound of Material (LB)	100% $0.45 /LB	$0.33 /LB				100% $0.82 /LB	100% $0.64 /LB		33% $0.63/LB
Per Gallon of Material(GAL)								$12 /GAL	11% $10/GAL
6.) Number of contractors used in Previous Year	3	4	0			1	4	0	1.50

Appendix A.2 District Survey Results

Survey Questions / District	1	2	3	4	5	6	7	8	
7.) Typical Crew Organization									
Foreman (Crew Leader)	1	1	1/worker	1			1	1	
Drivers	1		2	1	2	2	2	2	
Crack Cleaners	1	2	1	1	1	1	1	2	
Crack Fillers	1	2	1	2	1/leader	2	1	1	
Squeegee Operators	1	1	1	2		2	2	3	
Flagmen	2	1	0	1	As Needed		0	1	
Total Crew Members	7	6	6	8	3	7	7	10	

Survey Questions / District	9	10	11	12	13	14	15	16	
7.) Typical Crew Organization									
Foreman (Crew Leader)	1	1	1	1	1	1	1	1	
Drivers	2	2	3	2	1	2	2	1	
Crack Cleaners	1	1	1	1	1	1	1	1	
Crack Fillers	1	1	2	1	1	1	2	1	
Squeegee Operators	2	2	2	1	1	2	1	1	
Flagmen	2	2	2		2	2		1	
Total Crew Members	9	9	11	6	7	9	7	6	

Survey Questions / District	17	18	19	20	21	23	24	25	State Average
7.) Typical Crew Organization									
Foreman (Crew Leader)	1	1	1	1		1	1	1	1
Drivers	2	1	2	2		2	1	1	2
Crack Cleaners	1	2	1	1		1	1	0	1
Crack Fillers	2	1	1	1		1	1	1	1
Squeegee Operators	2	2	1	1		1	3	1	2
Flagmen	2	0	1	2		0	2	1	1
Total Crew Members	10	7	7	8	0	6	9	5	7.4

Appendix A.3 Crack and Joint Maintenance Expenditures and Unit Pricing

09/92 Through 1/93 Crack and Joint Maintenance Expenditures and Unit Pricing

Total Percentage of Contracted Work Statewide 61.6%
Total Percentage of In-House Work Statewide 38.4%

District	1	2	3	4	5	6	7
Code 221 Asphalt Rubber Applications							
Inhouse Total Expenditure	$359	$15,093	$24,188	$14,574	$32,471	$15,996	$32,054
Inhouse Unit price	$1.31/LB	$0.46/LB	$0.39/LB	$0.95/LB	$0.77/LB	$0.41/LB	$0.89/LB
Contracted Total Expenditure							
Contracted Unit Price							
Code 222 Other Sealant Applications							
Inhouse Total Expenditure		$46,581		$1,916	$6,856	$46,638	$93.19
Inhouse Unit price		$9.88/Gal		$1.69/Gal	$25.87/Gal	$2.37/Gal	$1.86/Gal
Contracted Total Expenditure							
Contracted Unit Price							

District	8	9	10	11	12	13	14
Code 221 Asphalt Rubber Applications							
Inhouse Total Expenditure	$11,085		$781				$16,117
Inhouse Unit price	$0.64/LB		$0.69/LB				$1.32/LB
Contracted Total Expenditure		$53,033	$44,241	$3,395	$23,992		
Contracted Unit Price		$0.38/LB	$3.30/LB		$4.76/LB		
Code 222 Other Sealant Applications							
Inhouse Total Expenditure		$1,468		$469	$3,785	$1,447	$28,911
Inhouse Unit price		$3.45/Gal		$3.03/Gal	$18.93/Gal	$3.05/Gal	$6.45/Gal
Contracted Total Expenditure					$507		
Contracted Unit Price							

District	15	16	17	18	19	20	21
Code 221 Asphalt Rubber Applications							
Inhouse Total Expenditure	$7,984	$13,165		$2,262	$8,614		$551
Inhouse Unit price	$0.70/LB	$0.22/LB		$0.81/LB	$0.40/LB		$32.41/LB
Contracted Total Expenditure			$38,865	$243,611			$21,838
Contracted Unit Price			$1.03/LB	$0.90/LB			
Code 222 Other Sealant Applications							
Inhouse Total Expenditure	$24,940	$28,020	$1,413		$289		
Inhouse Unit price	$1.07/Gal	$9.57/Gal	$3.93/Gal		$14.43/Gal		
Contracted Total Expenditure							
Contracted Unit Price							

District	23	24	25	Statewide
Code 221 Asphalt Rubber Applications				
Inhouse Total Expenditure	$19,169	$7,631	$13,187	$235,281
Inhouse Unit price	$0.44/LB	$0.72/LB	$0.48/LB	$0.68/LB
Contracted Total Expenditure	$34,502	$255,772		$719,249
Contracted Unit Price	$0.79/LB	$1.08/LB		$1.75/LB
Code 222 Other Sealant Applications				
Inhouse Total Expenditure	$2,862	$16,187	$2,449	$214,324
Inhouse Unit price	$6.36/Gal	$9.07/Gal	$5.20/Gal	$7.42/Gal
Contracted Total Expenditure	$24			$531
Contracted Unit Price				

Appendix B

ECONOMIC ANALYSIS OF AUTOMATED CRACK SEALER (In-house, hot pour crack sealing)			
GENERAL DATA	**SCENARIO 1**	**SCENARIO 2**	**SCENARIO 3**
WORK UNITS OF THE ARM SYSTEM	Lbs.	Lbs.	Lbs.
ESTIMATED WORK CAPACITY OF THE ARM SYSTEM (UNITS/YEAR)	432,000	432,000	432,000
ESTIMATED YEARLY EFFICIENCY OF THE ARM SYSTEM	90%	90%	90%
ANALYSIS PERIOD OR LIFE OF THE ARM SYSTEM (YEARS)	6	6	6
MINIMUM ATTRACTIVE RATE OF RETURN (MARR)	4.00%	6.00%	8.00%
INITIAL COST OF ARM SYSTEM (PRESENT $)	$70,000	$70,000	$70,000
OPERATING COSTS OF ARM SYSTEM (ANNUAL $)	$4,000	$4,000	$4,000
MAINTENANCE COSTS OF ARM SYSTEM (ANNUAL $)	$3,000	$3,000	$3,000
OVERHEADS OF ARM SYSTEM (ANNUAL $)	$3,500	$3,500	$3,500
SALVAGE VALUE OF ARM SYSTEM (FUTURE $)	$15,000	$15,000	$15,000
LIFE CYCLE COST/UNIT-WORK OF 1 ARM SYSTEM ($/UNIT)	$0.05	$0.05	$0.05
CURRENT TXDOT COSTS/UNIT WORK ($ /UNIT)	$0.73	$0.73	$0.73
• • YEARLY EXPENDITURES FOR THIS ACTIVITY ($/YEAR)	$1,685,600	$1,685,600	$1,685,600
• • EXPENDITURES ON LABOR FOR THIS ACTIVITY (%)	46.14%	46.14%	46.14%
• • TYPICAL CREW SIZE FOR THIS MAINTENANCE ACTIVITY	7	7	7
NUMBER OF CREW MEMBERS TO BE REDUCED BY THE ARM SYSTEM	4	4	4
YEARLY UNIT WORK DONE BY TXDOT (UNITS)	2,309,041	2,309,041	2,309,041
% WORK PERFORMED BY 1 ARM SYSTEM	18.71%	18.71%	18.71%
LIFE CYCLE LABOR SAVING/UNIT-WORK OF1 ARM SYSTEM	$0.19	$0.18	$0.16
TXDOT LIFE-CYCLE SAVINGS (1 ARM SYSTEM)	$322,680	$297,803	$275,290
TXDOT LIFE-CYCLE SAVINGS (ALL WORK BY ARM SYSTEMS)	$1,613,399	$1,489,014	$1,376,452

Note: Output in bold format

Figure B.1 Economic Analysis of ARMM for hot pour in-house crack sealing by TxDOT (Direct Savings)

ECONOMIC ANALYSIS OF AUTOMATED CRACK SEALER (In-house, hot pour crack sealing)			
CASE STUDY	**SCENARIO 1**	**SCENARIO 2**	**SCENARIO 3**
PROJECT	I-35, San Marcos to New Braunfils (20 mile segment)		
ESTIMATED TOTAL UNIT WORK TO BE PERFORMED	4,800	4,800	4,800
NO. OF LANES EACH DIRECTION	2	2	2
ADT EACH DIRECTION	27,500	27500	27500
ESTIMATED LENGTH OF ROAD CLOSURE, MILES (CONV., ARM)	(1,1)	(1,1)	(1,1)
NO. OF LANES CLOSED (CONV., ARM)	(1,1)	(1,1)	(1,1)
NO. OF DAYS CLOSED (CONV., ARM)	(1,1)	(1,1)	(1,1)
ESTIMATED REDUCTION IN CLOSURE TIME (FROM CONV. TO ARM)*	20%	20%	20%
USER COSTS PER OPERATION, CONV. METHODS *	10,625	10,625	10,625
USER COSTS PER OPERATION, ARM SYSTEM *	8,368	8,368	8,368
NO. OF THESE TYPICAL OPERATIONS PER YEAR	6	6	6
TOTAL USER COSTS, CONVENTIONAL METHODS *	**$334,186**	**$313,479**	**$294,709**
TOTAL USER COSTS, ARM SYSTEM *	**$263,197**	**$246,889**	**$232,106**
TOTAL UNIT COST USING CONV. METHODS (TXDOT+USER COSTS)	**$70.35**	**$66.04**	**$62.13**
TOTAL UNIT COST USING ARM SYSTEM (TXDOT+USER COSTS)	**$55.42**	**$52.04**	**$48.97**
TXDOT SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$664**	**$613**	**$566**
USER-COST SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$70,989**	**$66,590**	**$62,603**
TOTAL SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$71,653**	**$67,203**	**$63,169**

Note: Output in bold format

Figure B.216 Economic Analysis of ARMM for hot pour in-house crack sealing by TxDOT (User-Cost Savings)

ECONOMIC ANALYSIS OF AUTOMATED CRACK SEALER (Contractors, hot pour crack sealing)			
GENERAL DATA	**SCENARIO 1**	**SCENARIO 2**	**SCENARIO 3**
WORK UNITS OF THE ARM SYSTEM	Lbs.	Lbs.	Lbs.
ESTIMATED WORK CAPACITY OF THE ARM SYSTEM (UNITS/YEAR)	432,000	432,000	432,000
ESTIMATED YEARLY EFFICIENCY OF THE ARM SYSTEM	90%	90%	90%
ANALYSIS PERIOD OR LIFE OF THE ARM SYSTEM (YEARS)	6	6	6
MINIMUM ATTRACTIVE RATE OF RETURN (MARR)	20.00%	25.00%	30.00%
INITIAL COST OF ARM SYSTEM (PRESENT $)	$70,000	$70,000	$70,000
OPERATING COSTS OF ARM SYSTEM (ANNUAL $)	$4,000	$4,000	$4,000
MAINTENANCE COSTS OF ARM SYSTEM (ANNUAL $)	$3,000	$3,000	$3,000
OVERHEADS OF ARM SYSTEM (ANNUAL $)	$3,500	$3,500	$3,500
SALVAGE VALUE OF ARM SYSTEM (FUTURE $)	$15,000	$15,000	$15,000
LIFE CYCLE COST/UNIT-WORK OF 1 ARM SYSTEM ($/UNIT)	$0.04	$0.04	$0.04
CURRENT TXDOT COSTS/UNIT WORK ($ /UNIT)	$0.70	$0.70	$0.70
• • YEARLY EXPENDITURES FOR THIS ACTIVITY ($/YEAR)	$4,320,400	$4,320,400	$4,320,400
• • EXPENDITURES ON LABOR FOR THIS ACTIVITY (%)	46.14%	46.14%	46.14%
• • TYPICAL CREW SIZE FOR THIS MAINTENANCE ACTIVITY	7	7	7
NUMBER OF CREW MEMBERS TO BE REDUCED BY THE ARM SYSTEM	4	4	4
YEARLY UNIT WORK DONE BY TXDOT CONTRACTORS (UNITS)	6,172,000	6,172,000	6,172,000
% WORK PERFORMED BY 1 ARM SYSTEM	7.00%	7.00%	7.00%
LIFE CYCLE LABOR SAVING/UNIT-WORK OF1 ARM SYSTEM	$0.11	$0.10	$0.09
CONTRACTORS' LIFE-CYCLE SAVINGS (1 ARM SYSTEM)	$165,248	$138,259	$116,065
CONTRACTORS' LIFE-CYCLE SAVINGS (ALL WORK BY ARM SYSTEMS)	$2,313,476	$1,935,626	$1,624,906

Note: Output in bold format

FigureB.2a Economic Analysis of ARMM for hot pour crack sealing by Contractors (Direct Savings)

ECONOMIC ANALYSIS OF AUTOMATED CRACK SEALER (Contractors, hot pour crack sealing)			
CASE STUDY 1	**SCENARIO 1**	**SCENARIO 2**	**SCENARIO 3**
PROJECT	I-35, San Marcos to New Braunfils (20 mile segment)		
ESTIMATED TOTAL UNIT WORK TO BE PERFORMED	4,800	4,800	4,800
NO. OF LANES EACH DIRECTION	2	2	2
ADT EACH DIRECTION	27,500	27500	27500
ESTIMATED LENGTH OF ROAD CLOSURE, MILES (CONV., ARM)	(1,1)	(1,1)	(1,1)
NO. OF LANES CLOSED (CONV., ARM)	(1,1)	(1,1)	(1,1)
NO. OF DAYS CLOSED (CONV., ARM)	(1,1)	(1,1)	(1,1)
ESTIMATED REDUCTION IN CLOSURE TIME (FROM CONV. TO ARM)*	20%	20%	20%
USER COSTS PER OPERATION, CONV. METHODS *	10,625	10,625	10,625
USER COSTS PER OPERATION, ARM SYSTEM *	8,368	8,368	8,368
NO. OF THESE TYPICAL OPERATIONS PER YEAR	6	6	6
TOTAL USER COSTS, CONVENTIONAL METHODS *	**$212,001**	**$188,153**	**$168,475**
TOTAL USER COSTS, ARM SYSTEM *	**$166,967**	**$148,185**	**$132,687**
TOTAL UNIT COST USING CONV. METHODS (CONTRACTOR+USER COSTS)	**$44.87**	**$39.90**	**$35.80**
TOTAL UNIT COST USING ARM SYSTEM (CONTRACTOR+USER COSTS)	**$35.41**	**$31.51**	**$28.29**
CONTRACTORS' SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$340**	**$284**	**$239**
USER-COST SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$45,034**	**$39,968**	**$35,788**
TOTAL SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$45,374**	**$40,253**	**$36,027**

Note: Output in bold format

Figure B.2b Economic Analysis of ARMM for hot pour crack sealing by Contractors (User-Costs Savings)

ECONOMIC ANALYSIS OF AUTOMATED CRACK SEALER (In-house, cold pour crack sealing)			
GENERAL DATA	**SCENARIO 1**	**SCENARIO 2**	**SCENARIO 3**
WORK UNITS OF THE ARM SYSTEM	Gallons	Gallons	Gallons
ESTIMATED WORK CAPACITY OF THE ARM SYSTEM (UNITS/YEAR)	44,550	44,550	44,550
ESTIMATED YEARLY EFFICIENCY OF THE ARM SYSTEM	90%	90%	90%
ANALYSIS PERIOD OR LIFE OF THE ARM SYSTEM (YEARS)	6	6	6
MINIMUM ATTRACTIVE RATE OF RETURN (MARR)	4.00%	6.00%	8.00%
INITIAL COST OF ARM SYSTEM (PRESENT $)	$70,000	$70,000	$70,000
OPERATING COSTS OF ARM SYSTEM (ANNUAL $)	$4,000	$4,000	$4,000
MAINTENANCE COSTS OF ARM SYSTEM (ANNUAL $)	$3,000	$3,000	$3,000
OVERHEADS OF ARM SYSTEM (ANNUAL $)	$3,500	$3,500	$3,500
SALVAGE VALUE OF ARM SYSTEM (FUTURE $)	$15,000	$15,000	$15,000
LIFE CYCLE COST/UNIT-WORK OF 1 ARM SYSTEM ($/UNIT)	$0.47	$0.46	$0.45
CURRENT TXDOT COSTS/UNIT WORK ($ /UNIT)	$4.76	$4.76	$4.76
• • YEARLY EXPENDITURES FOR THIS ACTIVITY ($/YEAR)	$688,800	$688,800	$688,800
• • EXPENDITURES ON LABOR FOR THIS ACTIVITY (%)	60.70%	60.70%	60.70%
• • TYPICAL CREW SIZE FOR THIS MAINTENANCE ACTIVITY	7	7	7
NUMBER OF CREW MEMBERS TO BE REDUCED BY THE ARM SYSTEM	4	4	4
YEARLY UNIT WORK DONE BY TXDOT (UNITS)	144,706	144,706	144,706
% WORK PERFORMED BY 1 ARM SYSTEM	30.79%	30.79%	30.79%
LIFE CYCLE LABOR SAVING/UNIT-WORK OF1 ARM SYSTEM	$1.60	$1.50	$1.41
TXDOT LIFE-CYCLE SAVINGS (1 ARM SYSTEM)	$272,392	$250,631	$230,943
TXDOT LIFE-CYCLE SAVINGS (ALL WORK BY ARM SYSTEMS)	$817,175	$751,892	$692,828

Note: Output in bold format

Figure B.3a Economic Analysis of ARMM for cold pour in-house crack sealing by TxDOT (Direct Savings)

ECONOMIC ANALYSIS OF AUTOMATED CRACK SEALER (In-house, cold pour crack sealing)			
CASE STUDY	SCENARIO 1	SCENARIO 2	SCENARIO 3
PROJECT	I-35, San Marcos to New Braunfils (20 mile segment)		
ESTIMATED TOTAL UNIT WORK TO BE PERFORMED	330	330	330
NO. OF LANES EACH DIRECTION	2	2	2
ADT EACH DIRECTION	27,500	27500	27500
ESTIMATED LENGTH OF ROAD CLOSURE, MILES (CONV., ARM)	(1,1)	(1,1)	(1,1)
NO. OF LANES CLOSED (CONV., ARM)	(1,1)	(1,1)	(1,1)
NO. OF DAYS CLOSED (CONV., ARM)	(1,1)	(1,1)	(1,1)
ESTIMATED REDUCTION IN CLOSURE TIME (FROM CONV. TO ARM)*	20%	20%	20%
USER COSTS PER OPERATION, CONV. METHODS *	10,625	10,625	10,625
USER COSTS PER OPERATION, ARM SYSTEM *	8,368	8,368	8,368
NO. OF THESE TYPICAL OPERATIONS PER YEAR	6	6	6
TOTAL USER COSTS, CONVENTIONAL METHODS *	**$334,186**	**$313,479**	**$294,709**
TOTAL USER COSTS, ARM SYSTEM *	**$263,197**	**$246,889**	**$232,106**
UNIT COST (USER-COSTS) FROM CONV. SYSTEM ($/UNIT)	**$1,012.69**	**$949.94**	**$893.06**
UNIT COST (USER-COSTS) FROM ARM SYSTEM ($/UNIT)	**$797.57**	**$748.15**	**$703.35**
TOTAL UNIT COST USING CONV. METHODS (TXDOT+USER COSTS)	**$1,017.45**	**$954.70**	**$897.82**
TOTAL UNIT COST USING ARM SYSTEM (TXDOT+USER COSTS)	**$801.20**	**$751.67**	**$707.15**
TXDOT SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$374**	**$344**	**$317**
USER-COST SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$70,989**	**$66,590**	**$62,603**
TOTAL SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$71,363**	**$66,934**	**$62,920**

Note: Output in bold format

Figure B.3b Economic Analysis of ARMM for cold pour in-house crack sealing by TxDOT (User-Costs Savings)

ECONOMIC ANALYSIS OF AUTOMATED CRACK SEALER (Contractors, cold pour crack sealing)			
GENERAL DATA	**SCENARIO 1**	**SCENARIO 2**	**SCENARIO 3**
WORK UNITS OF THE ARM SYSTEM	Gallons	Gallons	Gallons
ESTIMATED WORK CAPACITY OF THE ARM SYSTEM (UNITS/YEAR)	44,550	44,550	44,550
ESTIMATED YEARLY EFFICIENCY OF THE ARM SYSTEM	90%	90%	90%
ANALYSIS PERIOD OR LIFE OF THE ARM SYSTEM (YEARS)	6	6	6
MINIMUM ATTRACTIVE RATE OF RETURN (MARR)	20.00%	25.00%	30.00%
INITIAL COST OF ARM SYSTEM (PRESENT $)	$70,000	$70,000	$70,000
OPEHATING COSTS OF ARM SYSTEM (ANNUAL $)	$4,000	$4,000	$4,000
MAINTENANCE COSTS OF ARM SYSTEM (ANNUAL $)	$3,000	$3,000	$3,000
OVERHEADS OF ARM SYSTEM (ANNUAL $)	$3,500	$3,500	$3,500
SALVAGE VALUE OF ARM SYSTEM (FUTURE $)	$15,000	$15,000	$15,000
LIFE CYCLE COST/UNIT-WORK OF 1 ARM SYSTEM ($/UNIT)	**$0.42**	**$0.40**	**$0.39**
CURRENT TXDOT COSTS/UNIT WORK ($ /UNIT)	$8.40	$8.40	$8.40
• • YEARLY EXPENDITURES FOR THIS ACTIVITY ($/YEAR)	$305,200	$305,200	$305,200
• • EXPENDITURES ON LABOR FOR THIS ACTIVITY (%)	60.70%	60.70%	60.70%
• • TYPICAL CREW SIZE FOR THIS MAINTENANCE ACTIVITY	7	7	7
NUMBER OF CREW MEMBERS TO BE REDUCED BY THE ARM SYSTEM	4	4	4
YEARLY UNIT WORK DONE BY TXDOT CONTRACTORS (UNITS)	**36,333**	**36,333**	**36,333**
% WORK PERFORMED BY 1 ARM SYSTEM	**122.61%**	**122.61%**	**122.61%**
LIFE CYCLE LABOR SAVING/UNIT-WORK OF1 ARM SYSTEM	**$1.79**	**$1.59**	**$1.43**
CONTRACTORS' LIFE-CYCLE SAVINGS (1 ARM SYSTEM)	**$331,760**	**$286,040**	**$248,390**
CONTRACTORS' LIFE-CYCLE SAVINGS (ALL WORK BY ARM SYSTEMS)	**$331,760**	**$286,040**	**$248,390**

Note: Output in bold format

Figure B.4a Economic Analysis of ARMM for cold pour crack sealing by Contractors (Direct Savings)

ECONOMIC ANALYSIS OF AUTOMATED CRACK SEALER (Contractors, cold pour crack sealing)			
CASE STUDY	**SCENARIO 1**	**SCENARIO 2**	**SCENARIO 3**
PROJECT	I-35, San Marcos to New Braunfils (20 mile segment)		
ESTIMATED TOTAL UNIT WORK TO BE PERFORMED	330	330	330
NO. OF LANES EACH DIRECTION	2	2	2
ADT EACH DIRECTION	27,500	27500	27500
ESTIMATED LENGTH OF ROAD CLOSURE, MILES (CONV., ARM)	(1,1)	(1,1)	(1,1)
NO. OF LANES CLOSED (CONV., ARM)	(1,1)	(1,1)	(1,1)
NO. OF DAYS CLOSED (CONV., ARM)	(1,1)	(1,1)	(1,1)
ESTIMATED REDUCTION IN CLOSURE TIME (FROM CONV. TO ARM)*	20%	20%	20%
USER COSTS PER OPERATION, CONV. METHODS *	10,625	10,625	10,625
USER COSTS PER OPERATION, ARM SYSTEM *	8,368	8,368	8,368
NO. OF THESE TYPICAL OPERATIONS PER YEAR	6	6	6
TOTAL USER COSTS, CONVENTIONAL METHODS *	**$212,001**	**$188,153**	**$168,475**
TOTAL USER COSTS, ARM SYSTEM *	**$166,967**	**$148,185**	**$132,687**
TOTAL UNIT COST USING CONV. METHODS (CONTRACTOR+USER COSTS)	**$650.83**	**$578.56**	**$518.93**
TOTAL UNIT COST USING ARM SYSTEM (CONTRACTOR+USER COSTS)	**$512.98**	**$456.26**	**$409.45**
CONTRACTORS' SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$455**	**$392**	**$341**
USER-COST SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$45,034**	**$39,968**	**$35,788**
TOTAL SAVINGS ON CASE STUDY 1 (LIFE CYCLE OF ARM SYSTEM)	**$45,489**	**$40,361**	**$36,129**

Note: Output in bold format

Figure B.4b Economic Analysis of ARMM for cold pour crack sealing by Contractors (User-Costs Savings)

Appendix C: Compilation of Research, Studies, and Articles Related to the Automated Road Maintenance Machine

Appendix C.1 Presentations

Haas, C., "Progress in the Implementation of an Automated Pavement Crack Sealer," presentation to group invited to demonstration of the crack sealer, J.J. Pickle Research Center, July 2, 1996.

Haas, C., Greer, R., Gibson, G., Traver, A., and Tucker, R., "Advances in Control Systems for Construction Manipulators," The 13th International Symposium on Automation and Robotics in Construction, Toyko, June 13, 1996.

Haas, C., "Construction Rationalization and Activities of the Construction Industry Institute," invited speaker to the Waseda University Construction Robot Research International Workshop, Toyko, June 10, 1996.

Haas, C., Greer, R., Boehme, K., and Graff, J., "Technology for Automating Highway Maintenance Joint Sealing Operations in Texas," presentation in session 227 at the 75th Annual National Research Council's Transportation Research Board Meeting, Washington DC, January 10, 1996.

Haas, C., "Advanced Control Systems for Construction Equipment," presented at the ASCE Construction Congress IV, San Diego, CA, October 25, 1995.

Haas, C., "University of Texas Construction Automation Laboratory," poster presentation at the ASCE Construction Congress IV, San Diego, CA, October 24, 1995.

Haas, C., "Robotics and Automation," Center for Transportation Research 15th Annual Symposium, The Commons Building, J.J. Pickle Research Center, July 11, 1995.

Haas, C., "Progress in the Implementation of an Automated Pavement Crack Sealer," presentation to group invited to demonstration of the crack sealer and the large scale hydraulic manipulator, J.J. Pickle Research Center, June 16, 1995.

Haas, C., "Overview of the 10th International Symposium on Automation

and Robotics in Construction: Part B," presentation at 73rd Annual Transportation Research Board Meeting, Washington, DC, January 1994.

Haas, C., "Comparison of Multi-sensing Methods for the Detection of Cracks in Pavement Surfaces," presentation at 73rd Annual Transportation Research Board Meeting, Washington, DC, January 1994.

Haas, C., and Dailey, C., "UT Construction Automation Group," poster presentation for a Transportation Symposium, hosted by UT at the LBJ Library, for Transportation Secretary Federico Pena, November 1993.

Haas, C., and Dailey, C., Automated Crack Sealer Demonstration and Presentation, Balcones Research Center, The University. of Texas at Austin, September 3, 1993.

Appendix C.2 Related Reports and Theses

Couch, S., Hinojosa, J., Iglesias, R., "The Design of a Transport System for an Automated Crack Sealer," Design Project, Department of Mechanical Engineering, The University of Texas at Austin, Fall 1995.

Crowell, G., Razmi, A., "Design of Automated Construction Systems," Design Project Report, Department of Civil Engineering, The University of Texas at Austin, April 1995.

Crowell, G., "Implementation of Promising Automated Maintenance Equipment Within the Texas Department of Transportation," Thesis, Department of Civil Engineering, The University of Texas at Austin, May 1996.

Gharpuray, D.M., "An Evaluation of the Use of Video and Range Sensing for Pavement Crack Detection", Thesis, Department of Civil Engineering, The University of Texas at Austin, August 1993.

Haas, C., Shen, H., and Haas, R., "ADDA SYSTEM I (Automated Pavement Distress Data Acquisition and Evaluation System) Report with User Manual," Prepared for Ontario Ministry of Transportation and Communications, Research Program Project 21156, Phase III, August 1985.

Haas, C., et al, "Investigation of a Pavement Crack-Filling Robot", Report to the Strategic Highway Research Program, Carnegie Mellon University, Pittsburgh, PA, April 1992.

Haas, C., "A Model of Pavement Surfaces", Dissertation, Carnegie Mellon University, Pittsburgh, PA, September 1990.

Hsieh, T., Haas, C., and Hudson, W., "Automated Maintenance Technology to Reduce Fuel Consumption by Minimizing Lane Closure Time," report no. SWUTC/93/60035-1, Center for Transportation Research, The University of Texas at Austin, July 1993.

Kim, Y., "Path Planning For An Automatic Pavement Crack Sealer", Thesis, Departement of Civil Engineering, The University of Texas at Austin, August 1995.

Malek, G.J., "Methods, Practices, and Productivity Study of Crack Sealing/Filling in Texas", Masters Report, Department of Civil Engineering, The University of Texas at Austin, Spring 1993.

Osmani, A.I., "A Model for Evaluating Automation in Road Maintenance", Thesis, Department of Civil Engineering, The University of Texas at Austin, May 1994.

Appendix C.3 Related Publications and Proceedings

Submitted

Greer, R., Kim, Y., Haas, C., "Tele-Operated Control for Automated Construction and Maintenance", submitted to the Transportation Research Board in August 1996.

Kim, Y., Haas, C., "Path Planning for Machine Vision Assisted, Tele-Operated Pavement Crack Sealer," submitted to the ASCE Journal of Computing and Civil Engineering in July 1996.

Published

Boehme, K., "Automated Crack Sealer Has Potential To Reduce Danger And Cut Costs," Technical Quarterly, ed. K. Jones, TQ10-1, Texas Department of Transportation, Austin, TX, October 1995.

Greer, R., Haas, C., Gibson, G., Traver, A., and Tucker, R., "Advances in Control Systems for Construction Manipulators," Proceedings of the 13th International Symposium on Automation and Robotics in Construction, pp. 615-624, Japan Robot Association, Toyko, 1996.

Gharpuray, D.M., and Haas, C.T., "Comparison of Multi-sensing Methods for the Detection of Cracks in Pavement Surfaces", Proceedings of the ASCE Pacific Rim TransTech Conference, Seattle, WA, July 1993.

Haas, C., "Evolution of an Automated Crack Sealer: A Study in Construction Technology Development," Automation in Construction 4, pp. 293-305, 1996.

Haas, C., and Ahman, P., "Automatisering och robotanvandning revolutionerar USA's byggindustri (Automation and Robotics Revolution in the US Construction Industry), published in Byggindustrin (The Swedish Construction Industry Magazine), no. 19, pp. 24-27, May 31, 1996.

Haas, C., and Dailey, C., "Automation Opportunities in Highway Construction and Maintenance - Applications in Texas," National Research Council's TR News, No. 176, 1995.

Haas, C., "Building A Better Mousetrap," Civil Engineering Newsletter, Vol. 7, The University of Texas at Austin, Austin, TX, Fall 1995.

Haas, C., Skibniewski, M., and Budny, E., "Robotics in Civil Engineering", Microcomputers in Civil Engineering, No. 10, pp. 371-381, 1995.

Haas, C., "Implementation of an Automated Crack Sealing System Study Problem Statement", pp. 1-11,1994.

Haas, R., Abdelhalim, A., and Haas, C., "Highway Pavement Research: Issues, Opportunities, and Innovations in Construction and Maintenance," Proceedings, Colloquium on Transportation, Technical Center of the Hilti Corporation, Schaan, Principality of Liechtenstein, pp. 1-17, September 16, 1994

Haas, C., "A Look at the Construction Automation Laboratory, UT Austin," Technical Quarterly, ed. K. Jones, TQ8-3, Texas Department of Transportation, Austin, TX, October 1993.

.

Haas, C., Hendrickson, C., McNeil, S., and Bullock, D., "A Field Prototype of a Robotic Pavement Crack Sealing System", Proceedings of the 9th International Symposium on Automation and Robotics in Construction, Tokyo, Japan, June 1992.

Haas, C., Hajek, J., and Haas, R., "Opportunities for Automation in Pavement Maintenance," Proceedings, Transportation Association of Canada Annual Conference, Vol. I, pp. 873-891, Winnipeg, Canada, September 1991.

Haas, C., McNeil, S., Hendrickson, C., and Haas, R., "A Pavement Surface Model for Integrating Automated Management Data", Pavement Management Implementation, ASTM. STP 1121, Frank B. Holt, Wade L. Gramling, Editors, American Society for Testing and Materials, Philadelphia, PA, pp. 394-410, 1991.

Haas, C., and Hendrickson, C., "Integration of Diverse Technologies for

Pavement Sensing", Transportation Research Record, No. 1331, pp. 92-102, 1991.

Haas, C., Hendrickson, C., and McNeil, S., "A Design for Automated Pavement Crack Filling", Proceedings of the Construction Congress, Boston, MA, April 1991.

Haas, C., and Hendrickson, C., "A Computer-Based Model of Pavement Surfaces", Transportation Research Record, No. 1260, pp. 91-98, 1990.

Haas, C., Shen, H., Phang, W.A., and Haas, R., "An Expert System for Automation of Pavement Condition Inventory Data," Proceedings, North American Pavement Management Conference, Toronto, pp. 4.46-4.57, March 1985.

Haas, R., and Haas, C., "The Paving Industry: Issues, New Technologies, and Opportunities," Proceedings, Canadian Society of Civil Engineers' Conference, Vol. III pp. 265-274, Ottawa, Canada, June 1995.

Hajek, J., and Haas, C., "Applications of Artificial Intelligence in Highway Pavement Maintenance," Proceedings, Third International Conference on Applications of Artificial Intelligence in Engineering, Stanford, CA, August 1988.

Hendrickson, C., McNeil, S., Bullock, D., Haas, C., Peters, D., Grove, D., Kenneally, K., and Wichman, S., "Perception and Control for Automated Pavement Crack Filling", Proceedings of the Second International Conference on Applications of Advanced Technologies in Transportation Engineering, Minneapolis, MN, August 1991.

Hsieh, T., and Haas, C., "Costs and Benefits of Automated Road Maintenance", Transportation Research Record, No. 1406, pp. 10-19, 1994.

Osmani, A., Hudson, W., and Haas, C., "A Model for Evaluating Automation in Road Maintenance," report No. SWUTC/94/60035-2, Center for Transportation Research, The Univ. of Texas at Austin, August 1994.

Osmani, A., Haas, C., and Hudson, W., "Evaluation of Road Maintenance Automation," ASCE Journal of Transportation Engineering, vol. 122, no. 1, Jan/Feb. 1996.

Appendix D: Glossary of Terms

Alligator Cracks -Interconnected cracks that form a series of small blocks resembling alligator skin.

ARMM -Automated Road Maintenance Machine

Canopy -The upper portion of the X-Y table that supports the video cameras.

Diagonal Cracks - Cracks that are diagonal to the centerline of the pavement.

Joint Cracks - Cracks that develop near the joints of the pavement.

Transverse Cracks - Cracks that are perpindicular to centerline of the pavement.

Longitudinal Cracks -Cracks that follow a course approximately parallel to the centerline of the pavement.

Map Cracks - interconnected cracks forming blocks with an area of one meter squared or more.

Progressive Edge Cracking - Cracks that develop near the outside edges of the pavemement and progress toward a longitudinal joint.

Restraint Cracks - Cracks which develop near the outside edges of the pavement and progress in an irregular path towards the longitudinal joint.

RBBA - Rotating Bounding Box Algorithm

Section - Portion of roadway ranging widely in length and depending on maintenance program..

Tele-Operation - Controlling equipment from a distance. It involves several components including: remote control of the device (hardwired or radio), some form of visual feedback (human, video, or graphical), and usually computer assisted control (path planning and motion control).

Transverse Cracks - Cracks that follow a course approximately perpindicular to the centerline of the pavement.

Turret - The cylindrical rotating mechanism mounted in the z direction that frames and holds the sealant hose, air hose and squeegee onto the xy table.

Workspace - The area of the work zone that is framed within the X-Y table and is bounded either by the reach of the main manipulator's end effector or the view of the video cameras.

Work Zone - The area within a section of roadway where maintenance is taking place.

XY Table - The main frame of the ARMM that carries all of the equipment including motors, sealant hose connections, air hose connections, etc. The frame moves in both x and y directions.

APPENDIX E: SOURCE CODES

```
/*****************************************************************************/
/*************                                                  *************/
/*************    ***       ******      **   **     **   **   *************/
/*************   ** **      **   **     *** ***     *** ***   *************/
/*************  **   **     ******      ** * **     ** * **   *************/
/*************  *******     ** **       **   **     **   **   *************/
/*************  **   **     **  **      **   **     **   **   *************/
/*************  **   **     **   **     **   **     **   **   *************/
/*************                                                  *************/
/*****************************************************************************/

/*****************************************************************************/
/*                                                                           */
/*  This is the main program for the ARMM.  It contains many functions      */
/*  calls provided by Data Translation for the image processing board.       */
/*  The source code for all of those functions is not provided in the        */
/*  hard copy listings that follow.  For that code, contact the              */
/*  the Construction Automation Laboratory at The University of Texas at     */
/*  Austin by calling 512-471-8189.                                          */
/*  All of the source code created or modified by the researchers has been   */
/*  included.                                                                */
/*                                                                           */
/*  Date of last modification: Sept. 30, 1996 by Richard Greer               */
/*****************************************************************************/


/*****************************************************************************/
/*****************************************************************************/
/***                                          ****************************/
/***  ARMM Windows Program                    ****************************/
/***                                          ****************************/
/***  MODULE : CF3.C                          ****************************/
/***                                          ****************************/
/***  Last Update: 9/30/96                    ****************************/
/***    by Richard Greer                      ****************************/
/***  Main Module                             ****************************/
/***  Functions: 1) WinMain                   ****************************/
/***             2) WndProc                   ****************************/
/***             3) MenuCommand               ****************************/
/***                                          ****************************/
/***                                                                   ***/
/***                                                                   ***/
/***       Modified by U.T. ARMM researchers to include line drawing   ***/
/***         and editing with a mouse or touch screen, the ability to  ***/
/***         erase and start over, to redraw the drawn lines on the    ***/
/***         screen, to open an auxiliary buffer on startup.           ***/
/***       Menu Items were added to map out camera lens distrotion,    ***/
/***         path-planning, line snapping, and all necessary motor     ***/
/***         functions.  The source for the funciton calls is in       ***/
/***         separate files.                                           ***/
/***                                                                   ***/
/*****************************************************************************/
/*****************************************************************************/

/*****************************************************************************/
/***  Include Files & Function Definitions  ****************************/
```

```
/*****************************************************************************/

#include "dt51.h"
#include <direct.h>
#include <time.h>

#define  CAMERA_TIMER 27
/*****************************************************************************/
/***  Global Variables   ***************************************************/
/*****************************************************************************/

/* 32-bit identifiers used to specify an auxiliary, display, overlay, or  */
/* an acquire memory buffer                              */

BUF_HNDL acq_hndls[FW_NUM_ACQ_BUFS];
BUF_HNDL disp_hndls[FW_NUM_DISP_BUFS];
BUF_HNDL ovl_hndls[FW_NUM_OVL_BUFS];
BUF_HNDL sys_hndls[FW_NUM_SYS_BUFS];
BUF_HNDL tmp_hndls[FW_NUM_SYS_BUFS];
BUF_HNDL dtc_buf_hndl, tmp_handle, src_hndl, dest_hndl;
BUF_HNDL *tmp_hdl;

/* structures composed of 8 fields in which the routines will return       */
/* information on the size and location of the memory buffer          */

HNDL_STRUCT acq_hndl_struct;
HNDL_STRUCT disp_hndl_struct;
HNDL_STRUCT ovl_hndl_struct;
HNDL_STRUCT sys_hndl_struct;
HNDL_STRUCT tmp_hndl_struct;
HNDL_STRUCT *tmp_struct;

/* structures composed of 2 fields (the number of BUF_HNDLS and a pointer */
/* to the array of buffer handles) which are passed into the dt51_acquire */
/* and dt51_passthru routines                              */

BUF_HNDL_LIST acq_hndl_list;
BUF_HNDL_LIST disp_hndl_list;
BUF_HNDL_LIST ovl_hndl_list;
BUF_HNDL_LIST sys_hndl_list;
BUF_HNDL_LIST tmp_list;

/* structures composed of 4 fields in to which the routines return or set */
/* information on the size and starting location of the region of      */
/* interest in the specified memory buffer                  */

XY_rgn src_roi;
XY_rgn dest_roi;

/* structure composed of 19 fields to which the routine returns    */
/* information on the number of buffers, the size of each memory type on  */
/* the board, and other operational features                    */

FW_CONFIG cfg;

/* structure composed of 12 fields to which the routine returns or sets   */
/* information on the image acquisition format                  */
```

```
FMT fmt;

/* structure composed of 4 fields to which the routine returns or sets     */
/* information on the sync reset pulse and sync insert pulse positions     */

SYNC_FMT syncfmt;

/* structure composed of 3 fields to which the routine returns or sets     */
/* information on the sync output width, phase, and vertical sync length  */

SYNC_OUT_FMT syncoutfmt;

/* structure composed of 5 fields to which the routine returns or sets     */
/* the analog to digital parameter values                                  */

A2D a2d;

/* structure composed of 3 fields to which the routine returns or sets     */
/* the zoom, pan, and scroll parameters                                    */

ZPS zps;

/* structure composed of 4 fields to which the routine returns or sets     */
/* the image acquisition setup parameters (replaced by img_acq_setup with */
/* the release of the '-A' boards)                                         */

ACQ_SETUP acq_setup;

/* structure composed of six fields to which the routine returns or sets  */
/* the image acquisition setup parameters                                  */

IMG_ACQ_SETUP img_acq_setup;

/* specifies the device enabling key for the routine in question */

u_long device;

/* generic DWORD variables used to store temporary results */

DWORD wIndex1;
DWORD wIndex2;
DWORD wIndex3;
DWORD wIndex4;
DWORD wIndex5;
DWORD wIndex6;

/* generic string variables used to store temporary results */

char szString[80];
char szItem[20];
char szItem1[20];
char szItem2[20];
char szItem3[20];
char szItem4[20];
char szItem5[20];
char szItem6[20];
```

```
char szItem7[20];
char szItem8[20];
char szItem9[20];
char szItem10[20];
char szItem11[20];
char szItem12[20];

/* global variable to hold board revision type (init by RHard_Disp) */

int glbl_board_type = 0;

/*  global working directory string */
char working_dir[128];

/* generic global variables used to implement file IO dialog     */

int   file_type_flag;
UINT      image_buffer_number;
int   destination_buffer_type;
u_short  file_format;
char      szfile_type_str[256];
char      szNEWItem[256];

/* Global variables added by Richard Greer and others for the
    crack sealer program*/
/* global variables for line-drawing point array   */
//POINT ptarray[5] = { 6,185, 100,250, 150,240, 130,210, 6,185};
        //Polyline array for drawing a dummy polyline on the screen
        //used for testing purposes only, will soon be gotten rid of.
        //This was for an alternative to the rubber band lines if I could
        //not get them working, since I did this will no longer be needed
POINT out[1000] = {0};  //data structure that the program will
POINT path_out[1000] = {0};      //soon be converted to
POINT snap_out[1000] = {0};

long duration;
long dur; //ysk
int timed = 0;
int channel_switch = 0;
//int ptx[1000] = {0};  //x coords of points of drawn line
//int pty[1000] = {0};  //ycoords of points of drawn lines
int oldx = 0;   //x coord of drawn point used for erasing when adjusting
                    // lines
int oldy = 0;   //y coord for erasing
int index = 0;  //keeps track of the number of points in the ptx,pty array
int snapindex = 0; //keeps track of size of snapout array
int AdjustLine = 0; //flag to turn on/off adjusting line algorithm
int found = 0; //flag to indicate if the point clicked by the mouse is a
                //point in the array
int findpoint;  //index of the point being adjusted
int counter = -1; //count variable used for drawing rectangles to reprsent
                  //points in the drawn points array
int refresh = 0; //flag for refreshing the screen
int fresher = 0; //flag for drawing line-snapping lines
int count2;      //count variable
static char szNameText[15] = "saved.dat"; //output file for array of points
static char szNameText2[40] = "c:\\dti\\dt3851\\examples\\win\\saved2.dat";//2nd out
```

```
put file

u_short current_chan =0; //flag for image scaling/switching
clock_t start_time, run_time;
u_long avg_time = 0;
//int outx[1000] = {0};
//int outy[1000] = {0};
/* generic program ID string variables */

char szAppName[20];
char achPr[10];
char achFile[12];

/* generic Message Box string variables */

char lpCaption[51];
char lpMessage[250];
int MaxC = 50;
int MaxText = 249;

/* globally allocated Lookup Tables for input, output, and overlay */

LPINT ILut;
LPINT r;
LPINT g;
LPINT b;
LPINT Or;
LPINT Og;
LPINT Ob;

/* file I/O temporary storage for LUT name */
char LutName[128];

/* globally defined source and destination buffer parameters used for      */
/* image transforms                                    */

u_Cptr SRC_BASE,  DES_BASE;
int    SRC_BUF,   DES_BUF;
int    SRC_SROW,  SRC_SCOL,   DES_SROW,  DES_SCOL;
int    SRC_WIDTH, SRC_HEIGHT, DES_WIDTH, DES_HEIGHT;
u_long SRC_PITCH, DES_PITCH;

/* globally defined source buffer parameters used for the histograming     */
/* of an image.                                    */

BUF_HNDL   hist_src_hndl;
u_Cptr      HIST_SRC_BASE;
int     hist_buffer_type;
int     HIST_SRC_BUF;
int     HIST_SRC_SROW,  HIST_SRC_SCOL;
int     HIST_SRC_WIDTH, HIST_SRC_HEIGHT;
u_long      HIST_SRC_PITCH;
int     HIST_SRC_O_ROW,  HIST_SRC_O_COL;
int     HIST_SRC_O_WIDTH, HIST_SRC_O_HEIGHT;

/*  temporary variable to hold cursor */
```

```
HCURSOR hOldCursor;

/* temporary variable to define srource and definition buffers temporarily */

u_short src_op;
u_short dest_op;

/* used to determine if dialog box parameters are okay to use */

u_short flag;

/* current video & sync channels */

u_short vchan, vsync;


/* DT3851/52 series reponse log variables */

HWND hWndLog;         // board response log window
HANDLE LogMem[LOGENT];    // log memory
LPSTR LogDesc[LOGENT];    // holds text
int scrlp;        // scroll position
u_short Inc;          // line count
u_short LogON;        // is LOG on


/* DT3851/52 menu status bar */

HWND hWndMenu;

/* VGA passthru flag to determine mode */

u_short VGAON;

/* generic variables used to temporarily convey a buffer or operation to  */
/* be performed                                    */

u_short op, op1, op2;

/* flag used by generic dialog box to set it up for proper routine */

u_short     NUM_FREQ;
int      direction;
u_short     update_flag;
u_short     buf_num;
int      cyChildHeight;
char         szItemLut[2];
u_long       MaxRepeat;
u_long       nframes;
int      popData[256];
int      ILData[256];
int      RLData[256];
int      GLData[256];
int      BLData[256];
int      oRLData[16];
int      oGLData[16];
int      oBLData[16];
```

```
int       TmpOvl[16];
int       MousePresent;
int       AvgMode;
u_short      ilut_flg;
u_short      olut_flg;
u_short      ovlut_flg;
u_short      FF_FOCUS;
u_short      PIX_PROF;
u_short      Lflag;
u_short      lutop;
u_short      table;
u_short      value;
u_short      src_buf_num;
u_short      dest_buf_num;
u_short      Tentry;
u_short      Aentry;
u_short      Lut_Tentry;
HANDLE        LutMem[535];
HANDLE        hAccel;
HANDLE        hInst;
HANDLE        TmpInst;
FARPROC      lpfnOldScr[6];
HWND          hwndScrol[4];
HWND          hwndLabel[4];
HWND          hwndValue[4];
HWND          hwndRect;
HWND          hwndButton[2];
HWND          hWndMain;
HWND          hWndILut  = (HWND)0;
HWND          hWndOLut  = (HWND)0;
HWND          hWndOvLut = (HWND)0;
HWND          hWndLut;
HWND          hWndOL;

/*************************************************************************/
/*                                       */
/*                                       */
/***********************  Windows Program  ********************************/
/*                                       */
/*                                       */
/*************************************************************************/
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpszCmdLine, int nC
mdShow)
{
 MSG  msg;
 HWND hWndPrev;


/*               Start of windows code              */


/* initialize global event flags */

 ovlut_flg = FALSE;
 olut_flg = FALSE;
 VGAON = FALSE;
 ilut_flg = FALSE;
```

```
 PIX_PROF = FALSE;

 Inc = 0;
 Tentry = 0;
 Aentry = 0;
 Lut_Tentry = 0;
 scrlp = 0;
 hist_buffer_type = FW_DISPBUF;

 TmpInst = hInst = hInstance;

 strcpy(achPr, "PROSIZE");
 strcpy(achFile, "PROSIZE.INI");
 strcpy(szAppName, "DT51");

/* ensure only one instance of program is running */
 hWndPrev = FindWindow("DT51", NULL);
 if(hPrevInstance != NULL)
   {
    if(IsIconic(hWndPrev))
      ShowWindow(hWndPrev, SW_RESTORE);
    BringWindowToTop (hWndPrev);
    return 0;
   }

/* establish communication with tigacd */
 if(!contact_tiga ())
    {
     ErrMsg ("Unable to contact TIGA, software will be Non-operational!");
     FreeMemory ();
     return (msg.wParam);
    }

/* create classes */
 if(!InitApplication (hInst))
   {
    term_tiga();
    return (FALSE);
   }

/* create the windows */
 if(!InitInstance (hInst, nCmdShow))
   {
    term_tiga();
    return (FALSE);
   }

/* allocate initial memory for logs */
 if(!GetMemory ())
     ErrMsg ("Unable to allocate memory for LOGs!");



/* hide lookup table */
 ShowWindow(hWndLut, SW_HIDE);
 ShowWindow(hWndOL, SW_HIDE);
```

```
/* disable log window */
 LogON = TRUE;
 ShowWindow(hWndLog, SW_HIDE);

/* install error handler to capture tiga timeout messages */
 install_usererror((void (*)(void)) dt_error_handler);

/* initalize the current working directory string */
 {
  int i;
  _getcwd (working_dir, 128);
  i = lstrlen(working_dir);
  if (working_dir[i-1] != '\\')
     lstrcat (working_dir, "\\");
 }

/* initialize LUTs */
 if(!lut_init ())
     ErrMsg ("Unable to initialize LUT routines!");



 hAccel = LoadAccelerators (hInst, szAppName);

/* check to insure mouse present */
 MousePresent = GetSystemMetrics (SM_MOUSEPRESENT);
 if(!MousePresent)
   {
    LoadString(hInst, IDS_WARNING, lpCaption, MaxC);
    LoadString(hInst, IDS_NO_MOUSE, lpMessage, MaxText);
    MessageBox(GetFocus(), lpMessage, lpCaption, MB_OK);
   }


/*********************************************/
/*                           */
/*                           */
/*   WINDOWS MAIN EVENT LOOP          */
/*                           */
/* loop on retrieving messages for the window */
/*                           */
/*                           */
/*********************************************/

   while(GetMessage(&msg, NULL, 0, 0))
       {
    if(!TranslateAccelerator (hWndMain, hAccel, &msg))
    {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
    }
    }

 FreeMemory ();

 return (msg.wParam);
```

```
}


/*****************************************************************************/
/***  Main Windows Loop  *****************************************************/
/*****************************************************************************/
long FAR PASCAL WndProc(HWND hWnd, UINT Message, WPARAM wParam, LPARAM lParam)
{
 HDC          hDC;
 PAINTSTRUCT ps;
 TEXTMETRIC  tm;
 static int  xL, yT, cyH, cxW, Xc, Yc;
 int          i, cyBorder, cxWidth, cyHeight, xChild, yChild;
 int          iReopen, Tr, Tg, Tb, cyh;
 WORD         Warr[2];

 /*****    these four variables were added by Charlene Dailey  2/14/95  */
 /*         also added by Richard Greer 11/21/95 */
 HPEN hPenNew, hPenOld, hPenOld2; /* Handles to change the pen used to draw the line
*/
                          /* so that rubberbanding can be used */
 static POINT   pt;       // The current point in the line drawing scheme
// static   int     pts = 0; //Does nothing right now
 // unsigned char   v,d;
//  static int      ptx[6000] = {0};
//  static int      pty[6000] = {0};
//  static char     chText[100];
    int indexold;   //Allows the screen to be updated only when a new point
                    //is added to the ptx,pty arrays

int hfile;      //hfile is output file handle
char filestuff[27]; //pointer to string to output to file
static OFSTRUCT OfStruct;  //File structure for output of drawn lines array
                           //for calibration


    static int first = 0; // Used to erase the rectangle of selected point
                          // when adjusting the line
    static int taco= 0;  //Used to draw a line to the point where the mouse
                         //button is released no matter if it is less than
                         //the prescribed distance between points.
                         //Needs to be updated to not allow a second point
                         //to be created right on top of onther point if
                         //the mouse button is released right on the last
                         //point drawn and to not allow a component of
                         //one point.

// FILE *out_fp;

 switch (Message)
   {

/* process menu item commands */
    case WM_COMMAND:
       return MenuCommand(hWnd, Message, wParam, lParam);

/* if pixel profile option on calculate linear address within the display */
```

```
/* buufer based off of mouse current X, Y position */
    case WM_NCHITTEST:
       if(PIX_PROF)
        {
         Xc = LOWORD (lParam);
         Yc = HIWORD (lParam);
         pix_config (Xc, Yc);
        }
       return DefWindowProc(hWnd, Message, wParam, lParam);

/* update color pattern within OLUTs when shifting OLUT option enabled */
    case WM_TIMER:
        //Added by R. Greer for image switching
        start_time = clock();
//      if(channel_switch <= 0)
//          {
            vchan = FW_CHAN_0;
            vsync = FW_SYNC_0;
            current_chan = 0;
        dt51_select_input_channel (device, &vchan);
        dt51_get_set_sync_source (device, FW_WRITE, &vsync);
        ACQ_Image(hWnd, hInst);   //added by R.greer 9/9/96
        IScale_Image(hWnd, hInst);//added by r.greer 9/9/96

//          channel_switch = 1;
//      }
//      else
//      {
            vchan = FW_CHAN_1;
            vsync = FW_SYNC_1;
        current_chan = 1;
//          channel_switch = -1;
//      }
        dt51_select_input_channel (device, &vchan);
        dt51_get_set_sync_source (device, FW_WRITE, &vsync);
        //zero_tmphdl ();
        //tmp_struct = &tmp_hndl_struct;
        //buf_num = 0;
        //op = FW_ACQBUF;
        //flag = TRUE;
        //Hand_IT();
        //BInfo_IT();
        //dt51_get_set_format_memory (device, FW_READ, &fmt);
        //format_center();
        //tmp_list.n = 1;
        //tmp_list.hndl_list = tmp_hdl;
        ACQ_Image(hWnd, hInst);   //added by R.greer 9/9/96
        IScale_Image(hWnd, hInst);//added by r.greer 9/9/96

        //dt51_acquire(device, &src_roi, &tmp_list, &dest_roi, 1);
        //dt51_acquire (device, &src_roi, (BUF_HNDL_LIST _far *)-1L, &dest_roi, 1);

        //Test to see if call timer
        timed ++;
        run_time += clock() - start_time;
        avg_time = run_time/timed;
        hfile = OpenFile(szNameText,&OfStruct, OF_WRITE | OF_CREATE);
```

```
    if(hfile >= 0)
    {   wsprintf(filestuff,"Avg Time %ld over %d\n",
                    avg_time, timed);
        _lwrite(hfile, filestuff, 25);
        _lclose(hfile);
    }

    //Test to see if call timer
    //added by R. Greer 9/4/96 for image switching


  /*if (lut_lock ())
     {
     Tb = b[0];
     for(i=0; i<255; i++)
      b[i] = b[i+1];

     b[255] = Tb;
     Tg = g[255];
     for(i=255; i>0; i--)
      g[i] = g[i-1];

     g[0] = Tg;
     Tr = r[0];
     for(i=0; i<128; i++)
      r[i] = r[i+1];

     r[128] = Tr;
     Tr = r[255];
     for(i=255; i>128; i--)
      r[i] = r[i-1];

     r[128] = Tr;
     Warr[0] = dt51_load_read_display_lut (device,
                        FW_WRITE,
                        0,
                        256,
                        (u_Sptr)r,
                        (u_Sptr)g,
                        (u_Sptr)b);
     if(Warr[0] != 0)
               disp_err(Warr[0]);
           else
     update_luts ();
      lut_unlock ();
    }*/
  break;

case WM_CREATE:
   hDC = GetDC (hWnd);
   GetTextMetrics (hDC, &tm);
   ReleaseDC (hWnd, hDC);
   cyBorder = GetSystemMetrics (SM_CYBORDER);
   cyChildHeight = tm.tmHeight + cyBorder * 2;

    //Test to see if call create
    /*timed ++;
```

```
        hfile = OpenFile(szNameText,&OfStruct, OF_WRITE | OF_CREATE);
        if(hfile >= 0)
        {   wsprintf(filestuff,"Went to create %d times\n", timed);
            _lwrite(hfile, filestuff, 27);
            _lclose(hfile);
        }*/
        //Test to see if call create

       //Added by R. Greer 9/4/96 to automatically allocate a 640X480 buffer
       AuxBufs_allox(hWnd, hInst);
       //Added by R. Greer 9/4/96
       break;

    case WM_CLOSE:
    case WM_DESTROY:
       if(hWnd == hWndMain)
         {
/* save current windows dimensions for next execution of program */
           wsprintf(szString,"%d",xL);
           WritePrivateProfileString(achPr, "x", szString, achFile);
           wsprintf(szString,"%d",yT);
           WritePrivateProfileString(achPr, "y", szString, achFile);
           if(cxW==0) cxW = CW_USEDEFAULT;
           wsprintf(szString,"%u",cxW);
           WritePrivateProfileString(achPr, "cx", szString, achFile);
           wsprintf(szString,"%d",cyH);
           WritePrivateProfileString(achPr, "cy", szString, achFile);
           iReopen = REOPEN_NORMAL;
           if(IsZoomed (hWnd)) iReopen = REOPEN_ZOOM;
           if(IsIconic (hWnd)) iReopen = REOPEN_DEFAULT;
           wsprintf(szString,"%d",iReopen);
           WritePrivateProfileString(achPr, "Reopen", szString, achFile);

/* restore LUTs to identity */
           lstrcpy (LutName, working_dir);
           lstrcat (LutName, "SAVE.LUT");
           read_luts ();

/* terminate communication with tigacd */
           term_tiga();

//kill timer
           KillTimer(hWndMain, CAMERA_TIMER);

/* terminate program */
           PostQuitMessage (0);
         }
       else
         DestroyWindow(hWnd);
      break;

    case WM_MENUSELECT:
       SendMessage (hWndMenu, Message, wParam, lParam);
      break;

    case WM_ACTIVATEAPP:
        {
```

```
      static u_short save_tp = 12;

      lstrcpy (LutName, working_dir);
      lstrcpy (LutName, "TEMP.LUT");

/* on gaining FOCUS, restore current lookup tables */
      if (wParam)
        {
         read_luts ();
         //put transparent pixel to what it was
         Warr [0] = dt51_get_set_pass_index ( device, FW_WRITE, &save_tp);
        }
      else
/* on loosing FOCUS, save current lookup tables */
        {
         // put transparent pixel back to default state
         Warr [0] = dt51_get_set_pass_index ( device, FW_READ, &save_tp);
         value = 0;
         Warr [0] = dt51_get_set_pass_index ( device, FW_WRITE, &value);
         write_luts ();
         DVga ();
        }
      }
      break;

 /****    this switch option was added by Charlene Dailey to allow
               a line to be drawn on mouse-move & left-button-down
               (this option actually replaces the case MOVE option
               in the original program)    ***/
/*  It has been modified by Richard Greer on 11/21/95 to stop drawing
    on left button up as well as some other modifications dealing with
    line adjustment etc. */
    case WM_LBUTTONUP:
        if(AdjustLine ==1)  /* If in adjust line mode*/
        {   found = 0;      /* Releases the point being adjusted and
                               prepares the program to adjust another
                               point*/
/*          first = 1;
            InvalidateRect(hWnd,NULL,FALSE);*/
        }
        else if(index > 0)       //if drawing a line
        {   pt = MAKEPOINT (lParam);
            if((((pt.x - out[index-1].x)*(pt.x - out[index-1].x) +
                (pt.y - out[index-1].y)*(pt.y - out[index-1].y)) >= 25) &&
                (out[index-1].x > 0))
            {   //add the last point to the array of points
                out[index].x = pt.x;
                out[index].y = pt.y;
                index ++;   //Signify the end of the line and separate from
                        //other lines
                out[index].x = -1;
                out[index].y = -1;
                index ++;
                taco = 2;   //Set to draw the final line segment and rectangles
                counter ++; //Increment the counter
                InvalidateRect(hWnd,NULL,FALSE);
            }
```

```
        else if((out[index-2].x <= 0) || (index == 1))
        //Don't want just a single point line
        {   index--;
            out[index].x = out[index].y = 0;
        }
        else if(out[index-1].x >= 0)
        {   out[index].x = -1;
            out[index].y = -1;
            index++;
            taco = 1;
            counter ++;
            InvalidateRect(hWnd, NULL, FALSE);
        }
    }
    break;
case WM_RBUTTONUP:  //Toggles the Adjust Line flag
                    //i.e. moves between draw and adjust modes
    if(AdjustLine == 1)
        AdjustLine = 0;
    else
        AdjustLine = 1;
    break;

case WM_MOUSEMOVE:
 //   out_fp = fopen("","w+t");

    switch (wParam)
    {
        case MK_LBUTTON:
            pt = MAKEPOINT (lParam);
            if(AdjustLine == 1) //if adjusting the line
            {   if(found == 0)  //find the point that is to be adjusted
                    {   for(findpoint=0;(findpoint<index) && (found == 0);
                            findpoint++)
                            if((abs(pt.x - out[findpoint].x) < 3) &&
                                (abs(pt.y - out[findpoint].y) < 3) &&
                                (pt.x >= 0) && (pt.y >= 0))
                                if(out[findpoint].x >=  0)
                                {   oldx = out[findpoint].x;
                                    oldy = out[findpoint].y;
                                    out[findpoint].x = pt.x;
                                    out[findpoint].y = pt.y;
                                    found = first = 1;
                                    InvalidateRect(hWnd,NULL,FALSE);
                                }
                        findpoint --;
                    }
                else    //already found the point just adjust the line
                        //segments
                {   oldx = out[findpoint].x;
                    oldy = out[findpoint].y;
                    out[findpoint].x = pt.x;
                    out[findpoint].y = pt.y;
                    InvalidateRect(hWnd,NULL,FALSE);
                }
            }
            else    //Drawing the line
```

```
                {   indexold = index;
                    if(((index == 0) || (out[index-1].x == -1)) && (pt.x >= 0)
                    && (pt.y >= 0))
                    //if the first point in a line
                    {   out[index].x = pt.x;
                        out[index].y = pt.y;
                        index++;
                    }
                  else if((((pt.x - out[index-1].x)*(pt.x - out[index-1].x) +
                    (pt.y - out[index-1].y)*(pt.y - out[index-1].y)) >= 900) &&
                        (pt.x >= 0) && (pt.y >= 0))
                    //else if the point is at least 30 pixels from the last
                    //line
                    {   out[index].x = pt.x;
                        out[index].y = pt.y;
                        index ++;
                    }
            /*  v = get_value(pt.x, pt.y,v);*/

            /* thresholding value testing output
                d = Calculation(&pt.x, &pt.y);
                wsprintf(chText,"pt[%d][%d]=%u,diff=%u",pt.x,pt.y,v,d); */

            // output adjusted results

            /*SearchPoints(&ptx[index],&pty[index],&outx[index],&outy[index]);
            wsprintf(chText,"%d\tpixel[%d][%d]=%u\tmove to pt[%d][%d]\n",
                index,pt.x,pt.y,v,outx[index],outy[index]);

            print_log(chText,(LPSTR)NULL);    */
                // this must be TRUE to display current point
                    if(indexold != index)//if a new point was added draw the
                                         //line segment
                        InvalidateRect(hWnd, NULL, FALSE);
                }
                break;

            /*default:
                ptx[index] = 0;
                pty[index] = 0;
                break;*/
          }
        //fclose(out_fp);
        break;

 /******************** end of added code  ************************/

    case WM_SIZE:
/* obtain resized program window dimensions */
       if(!IsZoomed (hWnd))
        {
         cxW = LOWORD (lParam);
         cxW += 2 * GetSystemMetrics (SM_CXFRAME);
         cyH = HIWORD (lParam);
         cyH += (2 * GetSystemMetrics (SM_CYFRAME)) +
             GetSystemMetrics (SM_CYCAPTION) - 1;
        }
```

```
        cxWidth = LOWORD (lParam);
        cyHeight = HIWORD (lParam);
        xChild = 0;
        yChild = cyHeight - cyChildHeight + 1;
/* resize program window */
        MoveWindow (hWndMenu, xChild, yChild, cxWidth, cyChildHeight, TRUE);
        InvalidateRect(hWndMenu, NULL, TRUE);
        cyh = GetSystemMetrics (SM_CYHSCROLL);
/* resize LUT Log based on size of program window */
        MoveWindow (hWndLut, (2*cxWidth)/3, 0, cxWidth/3, (cyHeight-cyh), TRUE);
        InvalidateRect(hWndLut, NULL, TRUE);
/* resize Board Response Log based on size of program window */
        MoveWindow (hWndLog, 0, 0, (2*cxWidth)/3, (cyHeight-cyh), TRUE);
        InvalidateRect(hWndLog, NULL, TRUE);
       break;

    case WM_PAINT:
       memset(&ps, 0x00, sizeof(PAINTSTRUCT));
       hDC = BeginPaint(hWnd, &ps);
       SetBkMode(hDC, TRANSPARENT);

/*  Start of code added by Richard Greer on 11/21/95 */
       hPenNew = GetStockObject(WHITE_PEN); //Get the white pen
       hPenOld = SelectObject(hDC, hPenNew);//Set the pen to white
       SetROP2(hDC,R2_XORPEN);  //Change the drawing mode to use
                                //an exclusive or pen so the line
                                //segments can be erased and updated
                                //without erasing the whole screen
                                //this allows the rubber band lines
        /*** this code was added by Charlene Dailey
                    to accommodate the drawn line   ***/

       //   TextOut(hDC, 50, 50, chText, strlen(chText));
                // used to display current coord
        //  TextOut(hDC, 30, 30, string, strlen(string));
        if((AdjustLine == 1) && (refresh == 0))
        //if in adjust mode and have not been given a refresh the entire
        //screen command
        {   if(first == 1) //if the point was just selected erase the
                           //rectangle of the old point
            {   Rectangle(hDC, oldx-2, oldy-2, oldx+2, oldy+2);
                first = 0;
            }
            if(((oldx != out[findpoint].x) || (oldy != out[findpoint].y)) &&
            ((oldx != 0) || (oldy != 0)))
            //If the point has been moved
            //Erase old line
            {   if((out[findpoint - 1].x >= 0) && (findpoint > 1))
                    //if not adjusting first point of line
                {   MoveTo(hDC, oldx, oldy);
                    LineTo(hDC, out[findpoint - 1].x, out[findpoint - 1].y);
                }
                if(out[findpoint + 1].x >= 0)
                    //if not adjusting the last point of line
                {   MoveTo(hDC, oldx, oldy);
                    LineTo(hDC, out[findpoint + 1].x, out[findpoint + 1].y);
                }
```

```
    }
    /*NOW DRAW THE NEW LINE */
    if(/*((ptx[findpoint] != ptx[findpoint - 1]) ||
        (pty[findpoint] != pty[findpoint - 1])) &&
        ((ptx[findpoint] >= 0) && (pty[findpoint] >= 0)) && */
        (out[findpoint - 1].x > 0))
    //draw the line if not on the previous point
    {   MoveTo(hDC, out[findpoint].x, out[findpoint].y);
        LineTo(hDC, out[findpoint - 1].x, out[findpoint - 1].y);
    }
    if(/*((ptx[findpoint] != ptx[findpoint + 1]) ||
        (pty[findpoint] != pty[findpoint + 1])) &&
        ((ptx[findpoint] >= 0) && (pty[findpoint] >= 0)) &&*/
        (out[findpoint + 1].x > 0))
    //draw the line if not on the next point
    {   MoveTo(hDC, out[findpoint].x, out[findpoint].y);
        LineTo(hDC, out[findpoint + 1].x, out[findpoint + 1].y);
    }
 }
else if(refresh == 0)
//In draw mode with no refresh command
{   MoveTo(hDC, pt.x, pt.y);
    if((out[index-2].x >= 0) && (out[index-2].y >= 0) &&
        (index > 1))
    /*   ((abs(pt.x - ptx[index-1]) > 5) ||
       (abs(pt.y - pty[index-1]) > 5)))*/
    //if not the first point in a line or the last part of the line
    {
        LineTo(hDC, out[index-2].x, out[index-2].y);
    }
    if(taco >= 1)
    //If the last point in the line
    {   if(taco == 2)
        {   MoveTo(hDC, out[index-2].x, out[index-2].y);
            if ((out[index-3].x > 0) && (out[index-3].y > 0))
            {
                LineTo(hDC, out[index-3].x, out[index-3].y);
            }
        }
        for(counter;counter<index;counter++)
        //draw rectangles for all of the points in that line
            if((out[counter].x >= 0) && (counter >= 0))
    Rectangle(hDC, out[counter].x-2, out[counter].y-2,
                out[counter].x+2, out[counter].y+2);
        taco = 0;
    }
}
/*************************************************/

if(refresh == 1)    //if refresh command given redraw all lines/rects
{   if(fresher == 1)
    {   count2 = 0;
        hPenNew = CreatePen(PS_SOLID, 1, RGB(255,0,0)); //Create a pen
        hPenOld2 = SelectObject(hDC, hPenNew);  //return to old pen color
        SetROP2(hDC,R2_COPYPEN);
        while((snap_out[count2].x > -2) && (count2 < 1000))
        {   if((snap_out[count2].x > 0) && (snap_out[count2].y > 0))
```

```
                {   MoveTo(hDC,snap_out[count2].x,snap_out[count2].y);
                    Rectangle(hDC, snap_out[count2].x-2, snap_out[count2].y-2,
                            snap_out[count2].x+2, snap_out[count2].y+2);
                    count2 ++;
                    while((snap_out[count2].x > 0) && (snap_out[count2].y > 0))
                    {   LineTo(hDC, snap_out[count2].x, snap_out[count2].y);
                        Rectangle(hDC, snap_out[count2].x-2, snap_out[count2].y-
2,
                            snap_out[count2].x+2, snap_out[count2].y+2);
                        count2 ++;
                    }
                }
                if(snap_out[count2].x == -1) count2 ++;
            }
            SetROP2(hDC,R2_XORPEN);
            SelectObject(hDC, hPenOld2);    //return to old pen color
        }
        for(count2 = 0; (out[count2].x > -2) && (count2 < 1000); count2 ++)
            if((out[count2].x > 0) && (out[count2].y > 0))
            {   MoveTo(hDC,out[count2].x,out[count2].y);
                Rectangle(hDC, out[count2].x-2, out[count2].y-2,
                            out[count2].x+2, out[count2].y+2);
                count2 ++;
                while((out[count2].x > 0) && (out[count2].y > 0))
                {   LineTo(hDC, out[count2].x, out[count2].y);
                    Rectangle(hDC, out[count2].x-2, out[count2].y-2,
                            out[count2].x+2, out[count2].y+2);
                    count2 ++;
                }
            }
        refresh = 0;
    }
    SetROP2(hDC,R2_COPYPEN);    //return to original drawing mode
    //Polyline(hDC, (LPPOINT)&ptarray, (int)5); //draw test polyline
    SelectObject(hDC, hPenOld); //return to old pen color
    /*** end of added code    ***/




  //       TextOut(hDC, 50, 50, chText, strlen(chText));

     EndPaint(hWnd, &ps);
     /*****************************ADD BY MA, LINE**********
     /*
     SearchPoints(ptx, pty, ptout_x, ptout_y);

         hDC = GetDC(hWnd);
         for(pts=0; pts<1000; pts++)
         {
             while(ptout_x[pts]!=0 && ptout_y[pts]!=0)
             SetPixel(hDC, ptout_x[pts], ptout_y[pts], 0L);
         }
         ReleaseDC(hWnd, hDC);

*************************************************************/
```

```
      break;

    default:
      return DefWindowProc(hWnd, Message, wParam, lParam);
   }

 return 0L;
}

/*************************************************************************/
/***  User Defined Menu Selections  ****************************************/
/*************************************************************************/
BOOL _cdecl MenuCommand(HWND hWnd,  WORD Message, WORD wParam, LONG lParam)
{
 HMENU    hMenu;
 FARPROC lpfnDlgProc;
 int i, hfile;      //hfile is output file handle
 char filestuff[27]; //pointer to string to output to file
 static OFSTRUCT OfStruct;  //File structure for output of drawn lines array
                            //for calibration

 hMenu = GetMenu(hWnd);  //get handle to programs main menu

 switch (LOWORD(wParam))
  {

/*************************************************************************/
/***  Main Menu FILE Options  **********************************************/
/*************************************************************************/
/* Allows for limited file I/O for Images, Log, LUTs, and complete camera */
/* setup(I.E. format memory, clocks, acq setup, a2d parameters         */
/*************************************************************************/

   case IDM_OPEN:  // Restore Image File to Selected Buffer
     Open_Image(hWnd, hInst);
     break;

   case IDM_SAVE:  // Save Image to File from selected buffer
     Save_Image (hWnd, hInst);
     break;

   case IDM_I_LOGOFF:  // Turn DT3851/52 Response Log OFF
      LogON = FALSE;
      ShowWindow(hWndLog, SW_HIDE);
      ModifyMenu(hMenu, IDM_I_LOGOFF, MF_BYCOMMAND, IDM_I_LOGON, "&Enable DT3851/52
Log \tShift+F1");

      EnableMenuItem( hMenu, IDM_I_REPORTBUFFERINFO,    (MF_BYCOMMAND | MF_DISABLED
| MF_GRAYED));
      EnableMenuItem( hMenu, IDM_I_CONFIGURATION,       (MF_BYCOMMAND | MF_DISABLED
| MF_GRAYED));
      EnableMenuItem( hMenu, IDM_I_DLL_DR,          (MF_BYCOMMAND | MF_DISABLED | MF
_GRAYED));
      EnableMenuItem( hMenu, IDM_I_TIGA_GC,         (MF_BYCOMMAND | MF_DISABLED | MF
_GRAYED));
      EnableMenuItem( hMenu, IDM_I_TIGA_GM,         (MF_BYCOMMAND | MF_DISABLED | MF
```

```
_GRAYED));
      EnableMenuItem( hMenu, IDM_I_TIGA_GSP,        (MF_BYCOMMAND | MF_DISABLED | MF
_GRAYED));
      EnableMenuItem( hMenu, IDM_I_GETHANDLES,      (MF_BYCOMMAND | MF_DISABLED | MF
_GRAYED));
      EnableMenuItem( hMenu, IDM_I_REPORTDISPLAYHANDLE, (MF_BYCOMMAND | MF_DISABLED
| MF_GRAYED));
     break;

   case IDM_I_LOGON:  // Turn DT3851/52 Response Log ON
     LogON = TRUE;
     ShowWindow(hWndLog, SW_SHOWNORMAL);
     ModifyMenu(hMenu, IDM_I_LOGON, MF_BYCOMMAND, IDM_I_LOGOFF, "&Disable DT3851/52
Log \tCtrl+F1");

     EnableMenuItem( hMenu, IDM_I_REPORTBUFFERINFO,    (MF_BYCOMMAND | MF_ENABLED));
     EnableMenuItem( hMenu, IDM_I_CONFIGURATION,       (MF_BYCOMMAND | MF_ENABLED));
     EnableMenuItem( hMenu, IDM_I_DLL_DR,          (MF_BYCOMMAND | MF_ENABLED));
     EnableMenuItem( hMenu, IDM_I_TIGA_GC,         (MF_BYCOMMAND | MF_ENABLED));
     EnableMenuItem( hMenu, IDM_I_TIGA_GM,         (MF_BYCOMMAND | MF_ENABLED));
     EnableMenuItem( hMenu, IDM_I_TIGA_GSP,        (MF_BYCOMMAND | MF_ENABLED));
     EnableMenuItem( hMenu, IDM_I_GETHANDLES,      (MF_BYCOMMAND | MF_ENABLED));
     EnableMenuItem( hMenu, IDM_I_REPORTDISPLAYHANDLE, (MF_BYCOMMAND | MF_ENABLED));
     break;

   case IDM_IO_LOG_READ:     // Restore current DT3851/52 Log
     Log_IO(hWnd, hInst, FILE_READ);
     break;

   case IDM_IO_LOG_WRITE:    // Save current DT3851/52 Log
     Log_IO(hWnd, hInst, FILE_WRITE);
     break;

   case IDM_IO_LUTS_READ:     // Restore Lookup Table data
     Lut_IO(hWnd, hInst, FILE_READ);
     break;

   case IDM_IO_LUTS_WRITE:  // Save Lookup Table data
     Lut_IO(hWnd, hInst, FILE_WRITE);
     break;

   case IDM_I_FILEFORMATMEM_READ:  // Save & Restore current camera setup
     Camera_Setup_IO(hWnd, hInst, FILE_READ);
     break;

   case IDM_I_FILEFORMATMEM_WRITE:  // Save & Restore current camera setup
     Camera_Setup_IO(hWnd, hInst, FILE_WRITE);
     break;

//Code added by Richard Greer 11/21/95
// Adding code here for writing data to a file
   case IDM_ENT: //Write to a file
     hfile = OpenFile(szNameText,&OfStruct, OF_WRITE | OF_CREATE);
     if(hfile >= 0)
     {  for(i = 0; i < index; i++)
     //Loop thru points outputting them to a file
        {   wsprintf(filestuff,"Index %3d pixel[%3d][%3d]\n",
```

```
                i,out[i].x,out[i].y);
            _lwrite(hfile, filestuff, 27);
        }
        _lclose(hfile);
      }
      break;

    case IDM_SPONGE: //write second array to a file
      hfile = OpenFile(szNameText2, &OfStruct, OF_WRITE | OF_CREATE);
      if(hfile >= 0)
      { i=0;
        while(snap_out[i].x != 0)
      //Loop thru points outputting them to a file
        {   wsprintf(filestuff,"Snaps %3d pixel[%3d][%3d]\n",
                i,snap_out[i].x,snap_out[i].y);
            _lwrite(hfile, filestuff, sizeof(filestuff));
            i++;
        }
        wsprintf(filestuff,"Time to snap: %d msec\n", duration);
        _lwrite(hfile, filestuff,sizeof(filestuff));
        _lclose(hfile);
      }
      break;

    case IDM_EXIT:  // Exit Program
      DestroyWindow(hWnd);
      break;

/*************************************************************************/
/***  Main Menu INIT Options  ******************************************/
/*************************************************************************/
/* Allows for reseting the board to a default state, retrieves current     */
/* board configuration, etc.                                 */
/*************************************************************************/

   case IDM_R_RESET_H:   // Reset DT3851/52 Hardware Only
      RHard ();
     break;

   case IDM_R_HD_RESET:  // Reset DT3851/52 Hardware, establish default settings
      RHard_Disp ();
     break;

   case IDM_I_CONFIGURATION:  // Get configuration of DT3851/52
              Get_DTconfig ();
     break;

   case IDM_I_TIGA_GC:   // Get current TIGA configuration
      config_tiga ();
     break;

   case IDM_I_TIGA_GSP:  // Show GSP Heap
      show_gsp_heap ();
     break;

   case IDM_I_GETHANDLES:  // Get buffer handles for: Acquire / Display / Overlay /
Auxiliary
```

```
      Get_Handles (hWnd, hInst);
     break;

   case IDM_I_REPORTDISPLAYHANDLE:  // Get current display handle
      HandlDisp ();
     break;

   case IDM_I_REPORTBUFFERINFO:  // Get buffer info for specified handle
      Get_BInfo (hWnd, hInst);
     break;

/*************************************************************************/
/***  Main Menu LUTs Options  ******************************************/
/*************************************************************************/
/* Allows contents of on board Lookup Tables to be modified        */
/*************************************************************************/

   case IDM_READ_ILUT:   // Read Input Lookup Table and display graphically
      Read_Graph_InLUT(hInst);
     break;

   case IDM_READ_OLUTS:  // Read Output Lookup Table and display graphically
      Read_Graph_OutLUT(hInst);
     break;

   case IDM_READ_OVLUTS:  // Read Overlay Lookup Table and display graphically
      Read_Graph_OvlLUT(hInst);
     break;

   case IDM_I_LUTON:  // Turn Lookup Table Log ON
      ShowWindow(hWndLut, SW_SHOWNORMAL);
      ModifyMenu(hMenu, IDM_I_LUTON, MF_BYCOMMAND, IDM_I_LUTOFF, "&Hide Lookup Table
 Log \tCtrl+F2");
     break;

   case IDM_I_LUTOFF:  // Turn Lookup Table Log OFF
      ShowWindow(hWndLut, SW_HIDE);
      ModifyMenu(hMenu, IDM_I_LUTOFF, MF_BYCOMMAND, IDM_I_LUTON, "Show &Lookup Table
 Log \tShift+F2");
     break;

   case IDM_WRITE_LUTS:  // Manually Adjust Input, Output, and Overlay Lookup Tables
      Up_LUTS (hInst);
     break;

/*************************************************************************/
/***  Main Menu VIDEO Options  *****************************************/
/*************************************************************************/
/* Allows for direct access to software configurable video input       */
/* circuitry, establishing acquisition rate, setting video input and       */
/* sync channels, etc.                                  */
/*************************************************************************/

   case IDM_I_FORMATMEMORY:  // Read/Edit entire format memory
      Set_FormComplt(hWnd, hInst);
     break;
```

```
    case IDM_I_EDITFORMATMEM:  // Edit specific format memory setups
       Set_FormEdit(hWnd, hInst);
      break;

    case IDM_I_SYNCFORMATMEM:  // Read/Edit synchronization format memory
       SyncFormatMem(hWnd, hInst);
      break;

    case IDM_I_EDITSYNCFORMATMEM:  // Edit signal in synchronization format memory
       EditSyncFormatMem(hWnd, hInst);
      break;

    case IDM_I_ALLCLKFREQ:  // Read/Edit all camera clock frequencies
       Set_ALL_ClockFreqs(hWnd, hInst);
      break;

    case IDM_I_EXTCLOCKFREQ:  // Read/Edit external clock frequency
       Set_Extrn_ClockFreq(hWnd, hInst);
      break;

    case IDM_I_HORIZSCANFREQ:  // Read/Edit horizontal scan frequency
       Set_Horiz_ClockFreq(hWnd, hInst);
          break;

    case IDM_I_VERTSCANFREQ:  // Read/Edit vertical scan frequency
       Set_Vert_ClockFreq(hWnd, hInst);
      break;

    case IDM_I_INPUTFREQ:  // Read/Edit input scan frequency
       Set_Input_ClockFreq(hWnd, hInst);
      break;

    case IDM_I_GAINOFFSET:  // Read/Edit Input A/D parameters
       Setup_A2D(hWnd, hInst);
      break;

    case IDM_I_VIDEOINPUTSYNC:  // Read/Edit video input and sync channels
       Setup_Video_Inputs(hWnd, hInst);
      break;

/*************************************************************************/
/***  Main Menu DISPLAY Options  *****************************************/
/*************************************************************************/
/* Allows for direct access of software configurable display circuitry,  */
/* (I.E. display clock rate, display format, zoom, pan, and scroll, etc.  */
/*************************************************************************/

    case IDM_D_DISABLE:   // Disable display
       DDisp ();
      break;

     case IDM_D_ENABLE:  // Enable display
       EDisp ();
      break;

    case IDM_D_CLOCK:  // Read/Edit display clock frequency
       Set_Disp_ClockFreq(hWnd, hInst);
```

```
      break;

    case IDM_D_ZPSM:  // Read/Edit displays Zoom, Pan, and Scroll states
       Set_ZPS(hWnd, hInst);
      break;

    case IDM_D_VGAPASS_EN:  // Enable VGA passthru
       EVga ();
      break;

    case IDM_D_VGAPASS_DIS:  // Disable VGA passthru
       DVga ();
      break;

    case IDM_O_ENAB_OVERLAY:  // Enable Overlay
       EOvl ();
      break;

    case IDM_O_DISAB_OVERLAY:  // Disable Overlay
       DOvl ();
      break;

    case IDM_O_PASSINDEX:  // Set transparent pixel value
       Set_TPvalue(hWnd, hInst);
      break;

/*************************************************************************/
/***  Main Menu ACQUIRE Options  *****************************************/
/*************************************************************************/
/* Allows one to specify image acquisition parameters, specify external  */
/* trigger mode, check for acquisition complete, select passthru, freeze */
/* frame operations, etc.                                                */
/*************************************************************************/

    case IDM_A_ACQUIRESETUP:  // Read/Edit acquire setup configuration
       Set_ACQsetup(hWnd, hInst);
      break;

    case IDM_A_IMGACQUIRESETUP:   // Read/Edit image acquisition parameters
       Set_ImgAcqSetup(hWnd, hInst);
      break;

    case IDM_C_FREEZEFRAME:  // Stop passthru or terminate FOCUS mode
       Stop_Video();
      break;

    case IDM_A_CHECKACQUIRECOMPLETE:  // Check for a completed acquisition
       Check_Video();
      break;

    case IDM_A_ENABLE_TRIGGER:  // Acquire using externally triggered input
       Trig_On(hWnd);
      break;

    case IDM_A_DISABLE_TRIGGER:   // Acquire using standard input
       Trig_Off(hWnd);
      break;
```

```
    case IDM_A_ACQUIREFULLFRAME:  // Acquire full frame
       ACQ_Image(hWnd, hInst);
      break;

    case IDM_A_FOCUSFULLFRAME:  // Continuosly Acuire full frame image
       FOCUS_Image(hWnd, hInst);
      break;

    case IDM_A_PASSTHRUFULLFRAME:  // Acuire live full frame images
       LIVE_Image(hWnd, hInst);
      break;

    case IDM_A_ACQUIREROI:  // Acquire region of interest
       ROI_Acq_Image(hWnd, hInst);
      break;

    case IDM_A_FOCUSROI:  // Continuosly Acquire region of image
       ROI_FOCUS_Image(hWnd, hInst);
      break;

    case IDM_A_PASSTHRUROI:  // Acquire live region of image
       ROI_LIVE_Image(hWnd, hInst);
      break;

   case IDM_A_ACQUIRETAVG:  // Acquire frame, add to total, when all frames acquired
, divide by num frames
       Average_IT(hWnd, hInst, FW_ACQ_TAVG);
      break;

   case IDM_A_ACQUIREPTAVG:  // Acquire all frames, add all frames, divide by num fr
ames
       Average_IT(hWnd, hInst, FW_ACQ_POST_TAVG);
      break;

   case IDM_A_ACQUIRERAVG:  // Acquire & average frame into running average, repeat
till all frames acquired
       Average_IT(hWnd, hInst, FW_ACQ_RAVG);
      break;

   case IDM_A_ACQUIREPRAVG:  // Acquire all frames, average each frame into running
average
       Average_IT(hWnd, hInst, FW_ACQ_POST_RAVG);
      break;

   case IDM_A_ACQUIREMANY:  // Sequentially acquire a series of images
      ACQ_Sequential(hWnd, hInst);
     break;

   case IDM_A_REPLAYMANY:  // Replay sequentially acquired series
      RUN_Sequential(hWnd, hInst);
     break;

   case IDM_A_LOOPREPLAY:  // Continually Loop-Thru acquired series
      LOOP_Sequential(hWnd, hInst);
     break;
```

```
   case IDM_A_FOURFULLFRAME:  // Acquire & Display from all 4 Video Inputs - in seri
es
      Four_ACQ_Images(hWnd, hInst);
     break;

/*************************************************************************/
/***  Main Menu AUXMEM Options   ***************************************/
/*************************************************************************/
/* Allows for limited onboard buffer manipulations              */
/*************************************************************************/

   case IDM_I_ALLOCATEAUXBUF:  // Allocates onboard auxiliary buffer
      AuxBufs_allox(hWnd, hInst);
     break;

   case IDM_I_FREEAUXBUF:  // Free specified auxiliary buffer
      Free_allox(hWnd, hInst);
     break;

   case IDM_A_DELETEMANY:  // Free a series of auxiliary buffers
      FREE_Sequential(hWnd, hInst);
     break;

   case IDM_I_DELETEALLBUFS:  // Free all auxiliary buffers
      AllFree_allox(hWnd);
     break;

/*************************************************************************/
/***  Main Menu SW_EXTRAS Options  *************************************/
/*************************************************************************/
/*  An assortment of generic Image Processing and Image Manipulation       */
/* routines, (ALL performed on HOST system)                 */
/*************************************************************************/

   case IDM_A_MEMORYCOPY:  // Copy a buffer's contents to another buffer
      Move_Image(hWnd, hInst);
     break;

   case IDM_A_MEMORYCOPYPIXELOP:  // Preform pixel operations between 2 buffers
      Move_Image_PixelOp(hWnd, hInst);
     break;

   case IDM_HISTO:  // Generate & Plot Histogram of Destination Buf
      hist_img (hWnd, hInst);
     break;

   case IDM_CONVOLUTION:  // Performs an Convolution on SRC Buffer
      Convolution(hWnd, hInst);
     break;

   case IDM_A_ISCALE:  // Perform Integer Scaling on image data
      IScale_Image(hWnd, hInst);
     break;

   case IDM_A_ROTATE_90:  // 90 Degree Rotation on image
      Img_Rotate_90(hWnd, hInst);
     break;
```

```
case IDM_INITCONTROLLER:      // initialize the Aerotech controller
    InitControl();
    refresh = 1;              //refresh the screen after controller is
                              //initialized
    InvalidateRect(hWnd, NULL, TRUE);
  break;

case IDM_LINESNAP:        //automatic line snapping test
    out[index-1].x = out[index-1].y = -2;
    line_snap(out, snap_out, &duration);
    for(i=0;i<1000;i++)
    {    out[i].x = snap_out[i].x;
         out[i].y = snap_out[i].y;
         if(out[i].x == -2) index = i;

    }
    //fresher = 1;
    AdjustLine = 1;
    refresh = 1;
    InvalidateRect(hWnd, NULL, TRUE);
  break;

case IDM_ADJUSTPOINT:     //runs through points adjusting distortion out
    //out[index-1].x = out[index-1].y = -2;
    AdjustPoints(out);
    path_plan(out,path_out, &dur); //ysk:dur
    refresh = 1;
    InvalidateRect(hWnd, NULL, TRUE);
  break;

case IDM_TRACECRACK:      // command the Aerotech to follow the crack
  //  AdjustPoints(out);
    TraceCrack(path_out);
    index = fresher = 0;   //reset index to zero points
    AdjustLine = 0;  //put in draw mode
    found = 0;   //have not found point in the array
    oldx = oldy = 0; //reset these for next adjust
    findpoint = 0;   //reset index of the point to be adjusted
    counter = -1;         //reset the counter for drawing rectangles
    for(i=0; i<1000; i++)
    {    out[i].x = 0;   //reinitialize the point arrays.
         out[i].y = 0;
         path_out[i].x = 0;
         path_out[i].y = 0;
         snap_out[i].x = 0;
         snap_out[i].y = 0;
    }
    InvalidateRect(hWnd, NULL, TRUE);  //clear all lines from the screen
  break;

case IDM_REFRESH:
//start timer
//SetTimer(hWndMain, CAMERA_TIMER, 3000, NULL);
     refresh = 1;
     InvalidateRect(hWnd, NULL, TRUE);
 break;
```

```
   case IDM_ERASE:

   //kill timer
    //       KillTimer(hWndMain, CAMERA_TIMER);

        index = 0;
        AdjustLine = 0;
        found = fresher = 0;
        oldx = oldy = 0;
        findpoint = 0;
        counter = -1;
        for(i=0;i<1000; i++)
        {   out[i].x = 0;
            out[i].y = 0;
            path_out[i].x = 0;
            path_out[i].y = 0;
            snap_out[i].x = 0;
            snap_out[i].y = 0;
        }
        InvalidateRect(hWnd, NULL, TRUE);
    break;

/*************************************************************************/
/***  Main Menu HELP Options  ******************************************/
/*************************************************************************/

/* About Box routine derived from Martin Heller's Advanced Windows Programming */

   case IDM_ABOUT:
      lpfnDlgProc = MakeProcInstance((FARPROC)AboutMP, hInst);
      DialogBox(hInst, (LPSTR)"SCM_ABOUT", hWnd, lpfnDlgProc);
      FreeProcInstance(lpfnDlgProc);
     break;

/*************************************************************************/
/***  End Main Menu  ***************************************************/
/*************************************************************************/
   default:
     break;

  }
 return TRUE;
}
```

```
/*****************************************************************************/
/*****************************************************************************/
/***                                                 ***************************/
/***  ARMM Windows Program                           ***************************/
/***                                                 ***************************/
/***  MODULE : AEROTCH2.C                            ***************************/
/***                                                 ***************************/
/***  Last Update: 9/30/96                           ***************************/
/***    by Richard Greer                             ***************************/
/***                                                 ***************************/
/***  This program contains two Aerotech functions, one to initialize   ***/
/***    the controller, the other to follow a set of points in an array ***/
/***    It also contain a function that uses a lookup table to map out  ***/
/***    camera distortion before any motion commands are sent.          ***/
/***                                                                    ***/
/*****************************************************************************/

#include <c:\msvc\include\windows.h>
#include <c:\msvc\include\string.h>
#include <c:\u500\lib\c\u500.h>
#include <c:\u500\win_dll\winaer.h>

void AdjustPoints(POINT outs[])
{   int indexer = 0;
    int adjust[15][23][2] = {0};
    int adjx, adjy;

    adjust[0][0][0]= adjust[0][22][1]= adjust[14][0][0]= -13;
    adjust[1][0][0]= adjust[2][0][0]= adjust[0][20][1]= adjust[0][21][1]= -12;
    adjust[12][0][0]= adjust[13][0][0]= adjust[13][1][0]=adjust[14][1][0]=-12;
    adjust[3][0][0]= adjust[4][0][0]= adjust[0][19][1]= adjust[10][0][0]= -11;
    adjust[11][0][0]= adjust[12][1][0]= -11;
    adjust[5][0][0]= adjust[6][0][0]= adjust[7][0][0]= adjust[8][0][0]= -10;
    adjust[9][0][0]= adjust[0][18][1]= adjust[1][21][1]= adjust[1][22][1]=-10;
    adjust[11][1][0]= -10;
    adjust[7][1][0]= adjust[8][1][0]= adjust[9][1][0]= adjust[0][1][0]= -9;
    adjust[0][16][1]= adjust[0][17][1]= adjust[1][1][0]= adjust[1][20][1]=-9;
    adjust[2][21][1]= adjust[2][22][1]= adjust[10][1][0]= adjust[14][2][0]=-9;
    adjust[3][1][0]= adjust[4][1][0]= adjust[5][1][0]= adjust[6][1][0]= -8;
    adjust[0][0][1]= adjust[0][1][1]= adjust[0][15][1]= adjust[1][0][1]= -8;
    adjust[1][18][1]= adjust[1][19][1]= adjust[2][1][0]= adjust[2][19][1]= -8;
    adjust[2][20][1]= adjust[12][2][0]= adjust[13][2][0]= adjust[14][3][0]=-8;
    adjust[14][4][0]= -8;
    adjust[7][2][0]= adjust[8][2][0]= adjust[9][2][0]= adjust[3][21][1]= -7;
    adjust[3][22][1]= adjust[0][2][0]= adjust[0][2][1]= adjust[0][14][1]= -7;
    adjust[1][2][0]= adjust[1][1][1]= adjust[1][2][1]= adjust[1][16][1]= -7;
    adjust[1][17][1]= adjust[2][2][0]= adjust[2][17][1]= adjust[2][18][1]= -7;
    adjust[10][2][0]= adjust[11][2][0]= adjust[13][3][0]= adjust[13][4][0]=-7;
    adjust[3][2][0]= adjust[4][2][0]= adjust[5][2][0]= adjust[6][2][0]= -6;
    adjust[3][18][1]= adjust[3][19][1]= adjust[3][20][1]= adjust[0][3][1]= -6;
    adjust[0][4][1]= adjust[0][5][1]= adjust[0][6][1]= adjust[0][13][1]= -6;
    adjust[1][3][1]= adjust[1][4][1]= adjust[1][15][1]= adjust[2][0][1]= -6;
    adjust[2][1][1]= adjust[2][16][1]= adjust[12][3][0]= adjust[12][4][0]= -6;
    adjust[9][3][0]= adjust[8][3][0]= adjust[7][3][0]= adjust[6][3][0]= -5;
    adjust[3][15][1]= adjust[3][16][1]= adjust[3][17][1]= adjust[4][20][1]=-5;
    adjust[4][21][1]= adjust[4][22][1]= adjust[0][3][0]= -5;
```

```
adjust[0][7][1]= adjust[0][8][1]= adjust[0][9][1]= adjust[0][10][1]= -5;
adjust[0][11][1]= adjust[0][12][1]= adjust[1][3][0]= adjust[1][5][1]= -5;
adjust[1][6][1]= adjust[1][7][1]= adjust[1][13][1]= adjust[1][14][1]= -5;
adjust[2][3][0]= adjust[2][2][1]= adjust[2][15][1]= adjust[10][3][0]= -5;
adjust[10][4][0]= adjust[11][3][0]= adjust[11][4][0]= adjust[14][5][0]=-5;
adjust[3][3][0]= adjust[4][3][0]= adjust[5][3][0]= adjust[3][13][1]= -4;
adjust[3][0][1]= adjust[3][1][1]= adjust[3][2][1]= adjust[4][18][1]= -4;
adjust[3][14][1]= adjust[4][19][1]= adjust[9][4][0]= -4;
adjust[0][4][0]= adjust[1][4][0]= adjust[1][8][1]= adjust[1][9][1]= -4;
adjust[1][10][1]= adjust[1][11][1]= adjust[1][12][1]= adjust[2][3][1]= -4;
adjust[2][4][1]= adjust[2][5][1]= adjust[2][6][1]= adjust[2][13][1]= -4;
adjust[2][14][1]= adjust[13][5][0]= adjust[13][6][0]= adjust[14][6][0]=-4;
adjust[3][3][1]= adjust[3][4][1]= adjust[3][5][1]= adjust[3][6][1]= -3;
adjust[3][7][1]= adjust[3][8][1]= adjust[3][9][1]= adjust[3][10][1]= -3;
adjust[3][11][1]= adjust[3][12][1]= adjust[3][4][0]= adjust[4][4][0]= -3;
adjust[4][0][1]= adjust[4][1][1]= adjust[4][2][1]= adjust[4][16][1]= -3;
adjust[4][17][1]= adjust[5][4][0]= adjust[5][16][1]= adjust[5][17][1]= -3;
adjust[5][18][1]= adjust[5][19][1]= adjust[5][20][1]= adjust[5][21][1]=-3;
adjust[5][22][1]= adjust[6][4][0]= adjust[7][4][0]= adjust[8][4][0]= -3;
adjust[2][4][0]= adjust[2][7][1]= adjust[2][8][1]= adjust[2][9][1]= -3;
adjust[2][10][1]= adjust[2][11][1]= adjust[2][12][1]= adjust[11][5][0]=-3;
adjust[12][5][0]= adjust[12][6][0]= adjust[13][7][0]= adjust[14][7][0]=-3;
adjust[4][3][1]= adjust[4][4][1]= adjust[4][5][1]= adjust[4][6][1]= -2;
adjust[3][5][0]= adjust[3][6][0]= adjust[4][5][0]= adjust[4][6][0]= -2;
adjust[4][7][1]= adjust[4][8][1]= adjust[4][9][1]= adjust[4][10][1]= -2;
adjust[4][11][1]= adjust[4][12][1]= adjust[4][13][1]= adjust[4][14][1]=-2;
adjust[4][15][1]= adjust[5][5][0]= adjust[5][6][0]= adjust[5][0][1]= -2;
adjust[5][1][1]= adjust[5][2][1]= adjust[5][3][1]= adjust[5][4][1]= -2;
adjust[5][13][1]= adjust[5][14][1]= adjust[5][15][1]= adjust[6][5][0]=-2;
adjust[6][6][0]= adjust[6][16][1]= adjust[6][17][1]= adjust[6][18][1]=-2;
adjust[6][19][1]= adjust[6][20][1]= adjust[6][21][1]= adjust[6][22][1]=-2;
adjust[7][5][0]= adjust[7][6][0]= adjust[8][5][0]= adjust[8][6][0]= -2;
adjust[9][5][0]= adjust[9][6][0]= -2;
adjust[0][5][0]= adjust[0][6][0]= adjust[1][5][0]= adjust[1][6][0]= -2;
adjust[2][5][0]= adjust[2][6][0]= adjust[10][5][0]= adjust[10][6][0]= -2;
adjust[11][6][0]= adjust[11][7][0]= adjust[12][7][0]= -2;
adjust[3][7][0]= adjust[4][7][0]= adjust[5][7][0]= adjust[6][7][0]= -1;
adjust[7][7][0]= adjust[8][7][0]= adjust[9][7][0]= adjust[5][5][1]= -1;
adjust[5][6][1]= adjust[5][7][1]= adjust[5][8][1]= adjust[5][9][1]= -1;
adjust[5][10][1]= adjust[5][11][1]= adjust[5][12][1]= adjust[6][10][1]=-1;
adjust[6][11][1]= adjust[6][12][1]= adjust[6][13][1]= adjust[6][14][1]=-1;
adjust[6][15][1]= adjust[7][15][1]= adjust[7][16][1]= adjust[7][17][1]=-1;
adjust[7][18][1]= adjust[7][19][1]= adjust[7][20][1]= adjust[7][21][1]=-1;
adjust[7][22][1]= -1;
adjust[0][7][0]= adjust[1][7][0]= adjust[2][7][0]= adjust[10][7][0]= -1;
adjust[12][9][0]= adjust[13][8][0]= adjust[13][9][0]= adjust[14][8][0]=-1;
adjust[14][9][0]= -1;
adjust[3][14][0]= adjust[3][15][0]= adjust[4][14][0]= adjust[4][15][0]= 1;
adjust[5][14][0]= adjust[5][15][0]= adjust[5][14][0]= adjust[6][15][0]= 1;
adjust[7][14][0]= adjust[7][15][0]= adjust[7][0][1]= adjust[7][1][1]= 1;
adjust[7][2][1]= adjust[7][3][1]= adjust[7][4][1]= adjust[7][5][1]= 1;
adjust[7][6][1]= adjust[8][14][0]= adjust[8][15][0]= adjust[8][6][1]= 1;
adjust[8][7][1]= adjust[8][8][1]= adjust[8][18][1]= adjust[8][19][1]= 1;
adjust[8][20][1]= adjust[8][21][1]= adjust[8][22][1]= adjust[9][14][1]= 1;
adjust[9][15][1]= 1;
adjust[0][14][0]= adjust[0][15][0]= adjust[0][16][0]= adjust[1][14][0]= 1;
adjust[1][15][0]= adjust[2][14][0]= adjust[2][15][0]= adjust[10][14][0]=1;
```

```
adjust[10][15][0]=adjust[10][8][1]= adjust[10][9][1]= adjust[10][10][1]=1;
adjust[11][14][0]= adjust[11][15][0]= adjust[12][13][0]= 1;
adjust[3][16][0]= adjust[4][16][0]= adjust[5][16][0]= adjust[6][16][0]= 2;
adjust[7][16][0]= adjust[8][16][0]= adjust[8][0][1]= adjust[8][1][1]= 2;
adjust[8][2][1]= adjust[8][3][1]= adjust[8][4][1]= adjust[8][5][1]= 2;
adjust[9][16][0]= adjust[9][5][1]= adjust[9][6][1]= adjust[9][7][1]= 2;
adjust[9][8][1]= adjust[9][9][1]= adjust[9][10][1]= adjust[9][11][1]= 2;
adjust[9][12][1]= adjust[9][13][1]= adjust[9][14][1]= adjust[9][15][1]= 2;
adjust[9][16][1]= adjust[9][17][1]= adjust[9][18][1]= adjust[9][19][1]= 2;
adjust[1][16][0]= adjust[2][16][0]= adjust[10][16][0]= adjust[10][7][1]=2;
adjust[10][11][1]=adjust[12][14][0]=adjust[12][15][0]=adjust[13][13][0]=2;
adjust[14][13][0]= adjust[13][15][0]= adjust[14][15][0]= 2;
adjust[3][17][0]= adjust[4][17][0]= adjust[5][17][0]= adjust[6][17][0]= 3;
adjust[7][17][0]= adjust[8][17][0]= adjust[9][17][0]= adjust[9][2][1]= 3;
adjust[9][3][1]= adjust[9][4][1]= adjust[9][20][1]= adjust[9][21][1]= 3;
adjust[9][22][1]= 3;
adjust[2][17][0]= adjust[10][17][0]= adjust[10][5][1]= adjust[10][6][1]=3;
adjust[10][12][1]=adjust[10][13][1]=adjust[10][14][1]=adjust[10][15][1]=3;
adjust[10][16][1]=adjust[10][17][1]=adjust[10][18][1]=adjust[10][19][1]=3;
adjust[10][20][1]=adjust[11][16][0]=adjust[11][8][1]= adjust[11][9][1]= 3;
adjust[11][10][1]=adjust[11][11][1]=adjust[11][12][1]=adjust[11][13][1]=3;
adjust[11][14][1]=adjust[11][15][1]=adjust[11][16][1]=adjust[13][14][0]=3;
adjust[14][14][0]= 3;
adjust[9][0][1]= adjust[9][1][1]= 4;
adjust[0][17][0]= adjust[1][17][0]= adjust[10][2][1]= adjust[10][3][1]= 4;
adjust[10][4][1]=adjust[10][21][1]= adjust[10][22][1]=adjust[11][17][0]=4;
adjust[11][7][1]=adjust[11][17][1]=adjust[12][16][0]= adjust[12][10][1]=4;
adjust[12][11][1]= 4;
adjust[3][18][0]= adjust[4][18][0]= adjust[5][18][0]= adjust[6][18][0]= 5;
adjust[7][18][0]= adjust[8][18][0]= adjust[9][18][0]= 5;
adjust[2][18][0]=adjust[10][18][0]=adjust[10][0][1]= adjust[10][1][1]= 5;
adjust[11][5][1]=adjust[11][6][1]=adjust[11][18][1]= adjust[11][19][1]= 5;
adjust[11][20][1]=adjust[12][17][0]=adjust[12][6][1]= adjust[12][7][1]= 5;
adjust[12][8][1]=adjust[12][9][1]=adjust[12][12][1]= adjust[12][13][1]= 5;
adjust[13][16][0]= adjust[14][16][0]= 5;
adjust[0][18][0]= adjust[1][18][0]= adjust[11][18][0]= adjust[11][3][1]=6;
adjust[11][4][1]=adjust[11][21][1]= adjust[12][5][1]=adjust[12][14][1]= 6;
adjust[12][15][1]=adjust[13][17][0]=adjust[13][9][1]=adjust[13][10][1]= 6;
adjust[13][11][1]= 6;
adjust[11][1][1]=adjust[11][2][1]=adjust[11][22][1]=adjust[12][18][0]= 7;
adjust[12][4][1]=adjust[12][16][1]=adjust[12][17][1]=adjust[12][18][1]= 7;
adjust[12][19][1]=adjust[13][7][1]= adjust[13][8][1]=adjust[13][12][1]= 7;
adjust[14][17][0]= 7;
adjust[3][19][0]= adjust[4][19][0]= adjust[3][20][0]= adjust[4][20][0]= 8;
adjust[5][19][0]= adjust[5][20][0]= adjust[6][19][0]= adjust[6][20][0]= 8;
adjust[2][19][0]=adjust[11][0][1]=adjust[12][2][1]=adjust[12][3][1]= 8;
adjust[12][20][1]=adjust[12][21][1]=adjust[13][18][0]=adjust[13][6][1]= 8;
adjust[13][13][1]=adjust[13][14][1]=adjust[13][15][1]=adjust[14][18][0]=8;
adjust[7][19][0]= adjust[7][20][0]= adjust[8][19][0]= adjust[8][20][0]= 9;
adjust[9][19][0]= adjust[9][20][0]= 9;
adjust[0][19][0]=adjust[1][19][0]=adjust[2][20][0]=adjust[10][19][0]= 9;
adjust[10][20][0]=adjust[11][19][0]=adjust[11][20][0]=adjust[12][0][1]= 9;
adjust[12][1][1]=adjust[12][22][1]=adjust[13][5][1]=adjust[13][16][1]= 9;
adjust[13][17][1]=adjust[13][18][1]=adjust[13][19][1]=adjust[14][8][1]= 9;
adjust[14][9][1]=adjust[14][10][1]=adjust[14][11][1]= 9;
adjust[1][20][0]=adjust[12][19][0]=adjust[12][20][0]=adjust[13][2][1]= 10;
adjust[13][3][1]=adjust[13][4][1]=adjust[13][20][1]=adjust[14][7][1]= 10;
```

```
    adjust[14][12][1]=adjust[14][13][1]= 10;
    adjust[3][21][0]= adjust[4][21][0]= adjust[5][21][0]= adjust[6][21][0]=11;
    adjust[0][20][0]=adjust[13][19][0]=adjust[13][20][0]=adjust[13][1][1]=11;
    adjust[13][21][1]=adjust[14][19][0]=adjust[14][5][1]=adjust[14][6][1]=11;
    adjust[14][14][1]=adjust[14][15][1]=adjust[14][16][1]=adjust[14][17][1]=11;
    adjust[14][18][1]=11;
    adjust[7][21][0]= adjust[8][21][0]= adjust[9][21][0]= 12;
    adjust[2][21][0]=adjust[10][21][0]=adjust[11][21][0]=adjust[13][0][1]=12;

    adjust[13][22][1]=adjust[14][20][0]=adjust[14][3][1]=adjust[14][4][1]=12;
    adjust[14][19][1]=12;
    adjust[1][21][0]=adjust[12][21][0]=adjust[14][2][1]=adjust[14][20][1]=13;
    adjust[14][21][1]=13;
    adjust[3][22][0]= adjust[4][22][0]= adjust[5][22][0]= 14;
    adjust[0][21][0]=adjust[13][21][0]=adjust[14][21][0]=adjust[14][0][1]=14;
    adjust[14][1][1]=adjust[14][22][1]=14;
    adjust[6][22][0]= adjust[7][22][0]= adjust[8][22][0]= adjust[9][22][0]= 15;
    adjust[2][22][0]=adjust[10][22][0]=15;
    adjust[1][22][0]=adjust[11][22][0]=16;
    adjust[0][22][0]=adjust[12][22][0]=17;
    adjust[13][22][0]=19;
    adjust[14][22][0]=20;

    while((indexer < 1000) && (outs[indexer].x != -2) &&
                (outs[indexer].y != -2))
    {   if((outs[indexer].x < 31) || (outs[indexer].x > 628))
            indexer++;
        else if((outs[indexer].y < 31) || (outs[indexer].y > 420))
            indexer++;
        else //in the workspace
        {   adjx = (int)(outs[indexer].x - 31)/26;
            adjy = (int)(outs[indexer].y - 31)/26;
            outs[indexer].x = outs[indexer].x + adjust[adjy][adjx][0];
            outs[indexer].y = outs[indexer].y + adjust[adjy][adjx][1];
            indexer++;
        }
    }

}

void InitControl(void)
    {
    char config[] = {"c:\\u500\\u500.cfg"};  /*    */
    char firmware[] = {"c:\\u500\\u500.jwp"};  /*  */
    char parameter[] = {"c:\\u500\\test.prm"};  /*  */

    WAPIAerInitialize(config, firmware, parameter);
    WAPIAerSend("enable x y z");
    WAPIAerSend("home x y");
    WAPIAerSend("dwell");
     }

void TraceCrack(POINT out2[])
    {
    int index = 0;
    char string[30];
    int xpoint;
```

```
int ypoint;
int xval, yval, rotation, rotold;
int resolution = 1;
int k = 0;
double radtodeg;

radtodeg = 180.0/PI;
rotation = rotold = 0;

WAPIAerSend("program me ab un un/se");
WAPIAerRetry();
WAPIAerSend("F500");
WAPIAerRetry();
WAPIAerSend("TR SINE");
WAPIAerRetry();
WAPIAerSend("RAMP 50");
WAPIAerRetry();
WAPIAerSend("ROUNDING 20");
WAPIAerRetry();
WAPIAerSend("velocity on");

while ((index < 1000) && (out2[index].x != 0) ||
                         (out2[index].y != 0))
       {
            // plot the current point

            if((out2[index+1].x > 0) && (out2[index].x > 0))
            {   xval = out2[index+1].x - out2[index].x;
                yval = out2[index+1].y - out2[index].y;
                rotation = (int)(atan2(yval,xval)*radtodeg + 180);
                while((rotation - rotold) > 180)
                    rotation -= 360;
                while((rotold - rotation) > 180)
                    rotation += 360;
            }

            xpoint = (627 - out2[index].x)*31.9328 + 3000;
            ypoint = (414 - out2[index].y)*16.0950 + 350;

            if (xpoint >= 24000) xpoint = 24000;   // these 4
            if (ypoint >= 6500) ypoint = 6500;   // statements keep
            if (xpoint <= 2500)   xpoint = 2500;     // the controller
            if (ypoint <= 300)   ypoint = 300;     // w/in bounds

            wsprintf(string, "LINEAR X%d Y%d Z%d F700\n",
                        xpoint, ypoint, rotation);

            WAPIAerSend(string);
            WAPIAerRetry();

            rotold = rotation;
        //  if you are at the beginning of a line:
        if ((index == 0) || (out2[index - 1].x == -1))
            {
        /*    wait until motor gets to first point before turning
                    air and sealant on */
            WAPIAerSend("WAIT ON");
```

```
                WAPIAerSend("WAIT OFF");

                // turn air and sealant on
                // use corner rounding to smooth motions
                WAPIAerSend("output 0,1,1,1");
                //WAPIAerSend("output 1,1");
                WAPIAerSend("rounding on");
                }

            // skip "resolution" number of pixels

            /*for (k = 1; k <= resolution; k++)
                {
                index ++;
                // if you are at the last point on line, stop
                if (local_x[index + 1] == -1) break;

                // if not, keep incrementing
                }*/

            // if you are at the end of the line, turn off air and sealant
            if (out2[index + 1].x <= -1)
                {
                // plot the last point in the line

/*              xpoint = (612 - local_x[index])*3.597;
                ypoint = (319 - local_y[index])*3.597;

                if (xpoint >= 2100) xpoint = 990;   // these 4
                if (ypoint >= 700) ypoint = 650;   // statements keep
                if (xpoint <= 0)   xpoint = 0;     // the controller
                if (ypoint <= 0)   ypoint = 0;     // w/in bounds

                wsprintf(string, "LINEAR X%d Y%d F100\n", xpoint, ypoint);*/

                WAPIAerSend(string);
                WAPIAerRetry();

                WAPIAerSend("WAIT ON");
                WAPIAerSend("WAIT OFF");

                WAPIAerSend("output 0,0,1,0");
                //WAPIAerSend("output 1,0");
                WAPIAerSend("rounding off");

                // move index to first point of next line
                index ++;
                index ++;
                }
            else
                index ++;
            }

    WAPIAerSend("velocity off");
    WAPIAerSend("home x y");
    WAPIAerSend("f100");
    WAPIAerSend("g1 Z0");
```

```
}
```

```
/*************************************************************************/
/*************************************************************************/
/***                                                 ********************/
/***  ARMM Windows Program                           ********************/
/***                                                 ********************/
/***  MODULE : LINESNAP.C                            ********************/
/***                                                 ********************/
/*************************************************************************/
/*************************************************************************/

#include "dt51.h"
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "c:\dti\dt3851\tiga.h"

#define WIDTH  640
#define HEIGHT 480

/*************************************************************************************/
/*************************************************************************************/
/************************                                 ********************/
/************************     LINE SNAPPING SOFTWARE      ********************/
/************************                                 ********************/
/*************************************************************************************/
/*************************************************************************************/
/*                                                                                   */
/*  This program uses the user-input points (out[]) for crack lines drawn by         */
/*  a mouse over the crack image obtained through the vision sensor.  The main       */
/*  purpose of this program is to improve the approximation of the user-input        */
/*  to be closer to the actual crack image by bounding a rectangular box along       */
/*  the normal between two points.  The bounding box gets the grey level values      */
/*  of each pixel within the box through the buffer containing the picture of        */
/*  the cracks.  The WIDTH and HEIGHT of the buffer is 640 x 480, respectively.      */
/*  More detailed description of this bounding box algorithm is presented in         */
/*  each function.                                                                   */
/*                                                                                   */
/*                          University of Texas at Austin                            */
/*                            Created by Young-Suk Kim                               */
/*                          Supervised by Dr. C.T. Haas                              */
/*                               Date: Feb. 9, 1996                                  */
/*                                                                                   */
/*************************************************************************************/
/*************************************************************************************/

/*************************************************************************************/
/*                                                                                   */
/*    get_value from Ma Ling gets the pixel gray-level value of the ImageRow,        */
/*    ImageCol image coordinates.                                                    */
/*                                                                                   */
/*************************************************************************************/
extern HNDL_STRUCT   disp_hndl_struct;
extern int   destination_buffer_type;

unsigned char get_value(int ImageCol, int ImageRow)
```

```
{
     unsigned long  dest;
     unsigned char  pixel_ptr;

      destination_buffer_type = FW_DISPBUF;
      dest = disp_hndl_struct.base_addr;
      dest += (unsigned long)(8 * (ImageRow  * disp_hndl_struct.pitch + ImageCol));
      gsp2host(dest, &pixel_ptr, sizeof(unsigned char), 0);
      return pixel_ptr;
}

/*****************************************************************************/
/*                                                                           */
/*    Determine_Angles : When two points that indicate the beginning and ending  */
/*    point of a section of a crack are given from out[], this function      */
/*    calculates the parallel and normals along the x and y axis.  These     */
/*    indicate the relative x and y values of parallel and normal to the crack   */
/*    (based on a hypotenuse of length = 1.0).                               */
/*                                                                           */
/*****************************************************************************/

void Determine_Angles(POINT one, POINT two, float *para_x, float *para_y,
                                            float *norm_x, float *norm_y)
{
        int x_diff, y_diff;
        float hypot;       // This is a hypotenuse of a triangle
                           // See the drawing for more detailed description
        x_diff = two.x - one.x;
        y_diff = two.y - one.y;
// Calculate the parallel to the box.  Then normalized to 1.0
// The drawing is graphically describing what the below variables are
        hypot = (float) sqrt(x_diff*x_diff + y_diff*y_diff);
        *para_x = x_diff/hypot;
        *para_y = y_diff/hypot;

        if(fabs(*para_x) < 0.001)  {*para_x = (float)0.001; // not exactly zero
                                    *para_y = (float)(1-0.001*0.001);}
        if(fabs(*para_y) < 0.001)  {*para_y = (float)0.001; // causes div 0 problems
                                    *para_x = (float)(1-0.001*0.001);}
        *norm_x = *para_y;
        *norm_y = (float) -1.0 * *para_x;
        return;
}



/*****************************************************************************/
/*                                                                           */
/*      Swap_pts : Swaps two points by Order_The_Points                      */
/*                                                                           */
/*****************************************************************************/

void swap_pts(POINT *one, POINT *two)
{
   POINT temp;

   temp.x = one->x;
```

```
   temp.y = one->y;
   one->x = two->x;
   one->y = two->y;
   two->x = temp.x;
   two->y = temp.y;
//   return;
}


/*****************************************************************************/
/*                                                                           */
/*      Order_The_Points : In every bounding box, pt1 should be above pt3 and */
/*      pt2.  pt2 and pt3 should be above pt4.  The detection order for four  */
/*      boundaries of each bounding box is accomplished by pt1->pt2, pt2->py4 */
/*      and pt1->pt3 pt3->pt4.                                               */
/*                                                                           */
/*****************************************************************************/
/*****************************************************************************/
/*void Order_The_Points(POINT box[])
{  POINT temp1, temp2;

   if(box[0].y > box[2].y)
   {  temp1.x = box[0].x;
    temp1.x = box[0].x;
    temp1.y = box[0].y;
    temp2.x = box[2].x;
    temp2.y = box[2].y;
    swap_pts(&temp1, &temp2);
    box[0].x = temp1.x;
    box[0].y = temp1.y;
    box[2].x = temp2.x;
    box[2].y = temp2.y;
   }
   if(box[1].y > box[3].y)
   {  temp1.x = box[1].x;
      temp1.y = box[1].y;
      temp2.x = box[3].x;
      temp2.y = box[3].y;
      swap_pts(&temp1, &temp2);
      box[1].x = temp1.x;
      box[1].y = temp1.y;
      box[3].x = temp2.x;
      box[3].y = temp2.y;
   }
   if(box[0].y > box[1].y)
   {  temp1.x = box[0].x;
      temp1.y = box[0].y;
      temp2.x = box[1].x;
      temp2.y = box[1].y;
      swap_pts(&temp1, &temp2);
      box[0].x = temp1.x;
      box[0].y = temp1.y;
      box[1].x = temp2.x;
      box[1].y = temp2.y;
   }
   if(box[2].y > box[3].y)
```

```
   {  temp1.x = box[2].x;
      temp1.y = box[2].y;
      temp2.x = box[3].x;
      temp2.y = box[3].y;
      swap_pts(&temp1, &temp2);
      box[2].x = temp1.x;
      box[2].y = temp1.y;
      box[3].x = temp2.x;
      box[3].y = temp2.y;
   }
   return;
}*/
/*****************************************************************************/
void Order_The_Points(POINT box[])
{

   if(box[0].y > box[2].y)
     {swap_pts(&box[0], &box[2]);}

   if(box[1].y > box[3].y)
     {swap_pts(&box[1], &box[3]);}

   if(box[0].y > box[1].y)
     {swap_pts(&box[0], &box[1]);}

   if(box[2].y > box[3].y)
     {swap_pts(&box[2], &box[3]);}

 return;
}
/*****************************************************************************/
/*                                                                           */
/*    Box_Bounds : Given the 4 corner points of a bounding box --            */
/*                                                                           */
/*        1-------2                                                          */
/*        |       |                                                          */
/*        |       |                                                          */
/*        3-------4                                                          */
/*                                                                           */
/*    This function calculates the x and y coordiates of both sides, left    */
/*    and right side so that a scan from one side to the other can determine */
/*    the total pixel value within the given bounding box.                   */
/*                                                                           */
/*****************************************************************************/

void Box_Bounds(POINT pt1, POINT pt2, POINT pt3, POINT pt4,
                POINT left_side[], POINT right_side[], int *length)
   // length is the length of the left and right hand sides: the no. of
{  // boundary detector of each side
    float para_x, para_y, norm_x, norm_y;
    int count, count2, y_start, y_stop, y_coord;
    float x_track;

   // Determine the angle from pt1 to pt2 and calculate the boundary on that
   // side of the box

    Determine_Angles(pt1, pt2, &para_x, &para_y, &norm_x, &norm_y);
```

```
    y_start = pt1.y;
    y_stop  = pt2.y;
    count = 0;
    for(y_coord = y_start; y_coord <= y_stop; y_coord++)
    {
       x_track = (count * para_x/para_y) + pt1.x;
       left_side[count].x = (int) x_track;
       left_side[count].y = y_coord;
       count++;
    }

   // Determine the angle from pt2 to pt4 and calculate the boundary on that
   // side of the box
    y_start = pt2.y + 1;
    y_stop =  pt4.y;
    count2 = 0;
    Determine_Angles(pt2, pt4, &para_x, &para_y, &norm_x, &norm_y);
    for(y_coord = y_start; y_coord <= y_stop; y_coord++)
    {
       x_track = (count2 * para_x/para_y) + pt2.x;
       left_side[count].x = (int) x_track;
       left_side[count].y = y_coord;
       count++;
       count2++;
    }

   // Determine the angle from pt1 to pt3 and calculate the boundary on that
   // side of the box
    Determine_Angles(pt1, pt3, &para_x, &para_y, &norm_x, &norm_y);
    y_start = pt1.y;
    y_stop  = pt3.y;
    count = 0;
    for(y_coord = y_start; y_coord <= y_stop; y_coord++)
    {
       x_track = (count * para_x/para_y) + pt1.x;
       right_side[count].x = (int) x_track;
       right_side[count].y = y_coord;
       count++;
    }

   // Determine the angle from pt3 to pt4 and calculate the boundary on that
   // side of the box
    y_start = pt3.y + 1;
    y_stop =  pt4.y;
    count2 = 0;
    Determine_Angles(pt3, pt4, &para_x, &para_y, &norm_x, &norm_y);
    for(y_coord = y_start; y_coord <= y_stop; y_coord++)
    {
       x_track = (count2 * para_x/para_y) + pt3.x;
       right_side[count].x = (int) x_track;
       right_side[count].y = y_coord;
       count++;
       count2++;
    }
    *length = count;     // The number of boundary detector of each side
    return;
}
```

```
/*****************************************************************************/
/*                                                                           */
/*  Value_of_Detect_Pixs : Actually calculates the total pixel value of the  */
/*  given box (between the points -- along the x-axis) from left[] to right[]*/
/*  or right[] to left[].-- For more details, see drawing.                   */
/*                                                                           */
/*****************************************************************************/

long Value_of_Detect_Pixs(POINT left[], POINT right[], int length)
{
   long Total = (long)0; // Initialization
   int i, x, y;

   for(i = 0; i < length ; i++)
   {
      y = right[i].y;    // same as left
      if(right[i].x > left[i].x)
      {
         for(x = left[i].x; x <= right[i].x; x++)
            if( (x >= 0) && (x < WIDTH) && (y >= 0) && (y < HEIGHT) )
                     Total = Total + get_value(x-1,y+36);
                else Total = Total + 255;  // Assign 255[white] for pixels
      }                                     // which is not in the image array
      else                                  // size, [640x480]
      {
         for(x = right[i].x; x <= left[i].x; x++)
            if( (x >= 0) && (x < WIDTH) && (y >= 0) && (y < HEIGHT) )
                     Total = Total + get_value(x-1,y+36);
                else Total = Total + 255;
      }
   }
   return Total;
}


/*****************************************************************************/
/*                                                                           */
/*     Value_of_Box : This function uses Value_of_Detect_Pixs after ordering */
/*     the points in a standard way to determine the total grey level value  */
/*     of the detected pixels in a box.                                      */
/*                                                                           */
/*****************************************************************************/

long Value_of_Box(POINT The_Box[])
{  long Box_Value;
   POINT Left[100], Right[100];
   int length;

   Order_The_Points(The_Box);  // Four points
   Box_Bounds(The_Box[0],The_Box[1],The_Box[2],The_Box[3],Left,Right,&length);
   Box_Value = Value_of_Detect_Pixs(Left, Right, length);
   return Box_Value;
}
```

```
/*****************************************************************************/
/*                                                                           */
/*    Create_Bounding_Box : When two points that indicate the part of a crack  */
/*    and the perpendicular in the x and y directions normalized to 1.0 are    */
/*    given,  it calculats four corner points of the bounding box[From i=0     */
/*    to i=20].                                                              */
/*    bounding_box[0]: pt1, bounding_box[1]: pt2                             */
/*    bounding_box[2]: pt3, bounding_box[3]: pt4                             */
/*                                                                           */
/*****************************************************************************/

//  Create Bounding Box: pt1, pt2, pt3, pt4
void Create_Bounding_Box(POINT one, POINT two, int offset_x, int offset_y,
                         POINT bounding_box[], float x_norm, float y_norm)
{
        bounding_box[0].x = (int) (x_norm * (-2.5 + offset_x) + one.x);
        bounding_box[0].y = (int) (y_norm * (-2.5 + offset_y) + one.y);
        bounding_box[1].x = (int) (x_norm * (-2.5 + offset_x) + two.x);
        bounding_box[1].y = (int) (y_norm * (-2.5 + offset_y) + two.y);
        bounding_box[2].x = (int) (x_norm * (2.5 + offset_x) + one.x);
        bounding_box[2].y = (int) (y_norm * (2.5 + offset_y) + one.y);
        bounding_box[3].x = (int) (x_norm * (2.5 + offset_x) + two.x);
        bounding_box[3].y = (int) (y_norm * (2.5 + offset_y) + two.y);
        return;
}



/*****************************************************************************/
/*                                                                           */
/*    Find_Best_Bounding_Box : The objective of this function is to calculate  */
/*    the grey level values of 21 bounding boxes from 10 pixels one way      */
/*    to 10 pixels the other way along the normal of the given box.          */
/*    Here, the dark_values of a box is the sum for the grey level values of   */
/*    all pixels that it hold.                                               */
/*                                                                           */
/*****************************************************************************/


void Find_Best_Bounding_Box(POINT one, POINT two, POINT bounding_box[])
{                               // one:First point and two:Second point
        int i, offset_x, offset_y, Target_Index;

        long dark_values[21];  // Sum of the grey level values of a box

        float para_x, para_y, norm_x, norm_y;   // See drawings for more detailed
        POINT dummy_box[4];                     // description
        long Target_Val;       // To get a box with the minimum greyl level value--
                               // There should be a crack
        Determine_Angles(one, two, &para_x, &para_y, &norm_x, &norm_y);
        for(i = 0; i <= 20; i++)                // To make 21 boxes along the normal

        {                                               // from 10 pixels one way to
10 pixels
                offset_x = i - 10; offset_y = i - 10;  // the other way along the no
```

```
rmal
                Create_Bounding_Box(one,two,offset_x,offset_y,dummy_box,norm_x,norm_
y);
                dark_values[i] = Value_of_Box(dummy_box);
        }

// Find a box with the minimum grey level value
        Target_Val = dark_values[0];
        Target_Index = 0;

        for(i = 1; i <= 20; i++)
                if((dark_values[i] < Target_Val) ||
                   (dark_values[i] == Target_Val && (abs(i-10) < abs(Target_Index -
10))))
                {
                        Target_Val = dark_values[i];
                        Target_Index = i;
                }
        offset_x = Target_Index - 10; offset_y = Target_Index - 10;

// Here it is called with bounding_box[], NOT dummy_box[]
        Create_Bounding_Box(one,two,offset_x,offset_y,bounding_box,norm_x,norm_y);
        return;
}



/*****************************************************************************/
/*                                                                           */
/*  Put_Bounding_In_Snapout: For an efficient path generation for the crack  */
/*  sealer, this function puts x, y coordinate pairs of left, middle and right */
/*  side of each bounding box into the snap_out array in the desired manner. */
/*  Right side point of each bounding box is replaced with the midpoint      */
/*  between the right side point of the previous box and the left side point */
/*  of the current box.                                                      */
/*                                                                           */
/*****************************************************************************/

void Put_Bounding_In_Snapout(POINT bounding_box[], POINT snap_out[], int index, int
First)
{
    float left_x, left_y, mid_x, mid_y, right_x, right_y;

    left_x  = ((float)(bounding_box[0].x + bounding_box[2].x))/(float)2.0;
    left_y  = ((float)(bounding_box[0].y + bounding_box[2].y))/(float)2.0;
    right_x = ((float)(bounding_box[1].x + bounding_box[3].x))/(float)2.0;
    right_y = ((float)(bounding_box[1].y + bounding_box[3].y))/(float)2.0;
    mid_x = (left_x + right_x) / (float)2.0;
    mid_y = (left_y + right_y) / (float)2.0;
    if(First == 1)
    {
       snap_out[index].x = (int) left_x;
       snap_out[index].y = (int) left_y;
    }
    else
    {
       snap_out[index].x = (int) (left_x+snap_out[index].x)/2;
```

```
        snap_out[index].y = (int) (left_y+snap_out[index].y)/2;
    }
    snap_out[index+1].x = (int) mid_x;
    snap_out[index+1].y = (int) mid_y;
    snap_out[index+2].x = (int) right_x;
    snap_out[index+2].y = (int) right_y;
    return;
}


/*****************************************************************************/
/*****************************************************************************/
/*****************************************************************************/
/*****************************                   *****************************/
/*****************************    line_snap      *****************************/
/*****************************                   *****************************/
/*****************************************************************************/
/*****************************************************************************/
/*****************************************************************************/

void line_snap(POINT out[], POINT snap_out[1000], long *duration)
{

      /*int    status, handle, color;*/
      int    i, j, /*counter,*/ snap_count, index;      // Simple loop counters
      int    DONE, /*New_Crack,*/ New_Box;             // Simple Boolean tests
      int    x, y;                    // From out[]
      /*int    x_diff, y_diff;*/        // To determine angles between two points
      unsigned char pixel_ptr;       // A pixel pointer to obtain the grey level
      clock_t start_time, end_time;
      POINT bounding_box[4];     // Four corner points of a bounding box

      int First;                     // Simple Boolean test in creating snap_out[]

    start_time = clock();

//  Create Bounding Boxes
    DONE = 0;
    index = 0;
    snap_count = 0;
    New_Box = 1;
    First = 1;

    while(DONE == 0)
    {
        if(out[index].x == -2)       // Line snapping process for whole crack lines
        {                            // over a crack image is completed
                DONE = 1;
                snap_count++;
                snap_out[snap_count].x = -2;
                snap_out[snap_count].y = -2;
                snap_count++;
        }

        else if(out[index].x != -1 && New_Box == 0)  // Line snapping process for a
crak
        {                                            // line is started and continue
```

```
d
                Find_Best_Bounding_Box(out[index-1], out[index], bounding_box);
                Put_Bounding_In_Snapout(bounding_box, snap_out, snap_count, First);

                snap_count = snap_count + 2; // 2 points: First point and Midpoint
                First = 0;                   // To take a mid point between the
        }                                    // right side point of the previous box
 and
                                             // left side point of the current box
        else if(out[index].x == -1) // Line snapping process for one crack line
        {                           // over a crack image is completed
                New_Box = 1;        // Indicate the start of a new crack line
                First = 1;
                snap_count++;
                snap_out[snap_count].x = -1;
                snap_out[snap_count].y = -1;
                snap_count++;
        }
        else
                New_Box = 0;

        index++;                    // To continue the line snapping process
    }                               // From out[0] to out[n]


    end_time = clock();
    *duration = end_time - start_time;

/*******************************************************/
/*******************************************************/
/********** Graphics Stuff *****************************/
/*******************************************************/
/*
// Now for the user-given points(user-drawn line): out[]
         DONE = 0;
         counter = 0;
         New_Crack = 1;
         status = _setcolor(DRAWN_LINE_COLOR);
         while (DONE == 0)
         {
                x = out[counter].x; y = out[counter].y;
                if(x == -1)
                {
                        New_Crack = 1;
                }
                else if(x == -2)
                {
                        DONE = 1;
                }
                else
                {
                        if(New_Crack == 1)
                        {
                            _moveto(out[counter].x, out[counter].y);
                                New_Crack = 0;
                        }
                        else if(New_Crack == 0)
```

```
                        {
                            _lineto(out[counter].x, out[counter].y);
                        }
                }
            counter = counter + 1;
        }


// Now for the adjusted crack lines
        DONE = 0;
        counter = 0;
        New_Crack = 1;
        status = _setcolor(ADJUSTED_LINE_COLOR);
        while (DONE == 0)
        {
                x = snap_out[counter].x; y = snap_out[counter].y;
                if(x == -1)
                {
                        New_Crack = 1;
                }
                else if(x == -2)
                {
                        DONE = 1;
                }
                else
                {
                        if(New_Crack == 1)
                        {
                            _moveto(snap_out[counter].x, snap_out[counter].y);
                                New_Crack = 0;
                        }
                        else if(New_Crack == 0)
                        {
                            _lineto(snap_out[counter].x, snap_out[counter].y);
                        }
                }
                    counter = counter + 1;
        }
*/

    return;
}
```

```
/*************************************************************************/
/*************************************************************************/
/***                                          ***************************/
/***  ARMM Windows Program                    ***************************/
/***                                          ***************************/
/***  MODULE : PATHPLN2.C                     ***************************/
/***                                          ***************************/
/*************************************************************************/
/*                                                                       */
/*                 PATH PLANNING SOFTWARE SOURCE CODE[2nd VERSION]       */
/*                                                                       */
/*************************************************************************/
/*                                                                       */
/*                 Function name: pathplan.c                             */
/*                 Created by Young-Suk Kim                              */
/*                 Date: Nov. 14, 1995                                   */
/*                                                                       */
/*************************************************************************/

/*************************************************************************/
/***                                                                   ***/
/***  This file contains all of the components used in the path-planning***/
/***  algorithm.                                                       ***/
/***                                                                   ***/
/*************************************************************************/

#include <math.h>
#include <stdio.h>
#include <time.h>
#include "dt51.h"
#define MAXPOINTS     1000


void path_plan(POINT out[], POINT path_out[], long *dur) //ysk1:dur
{

   double Total_Length;  /* Total_length:Crack_Length+Idle_Length */
   double Crack_Length, Idle_Length;
   double Crack_Length_Add, Idle_Length_Add;
   double Best_Distance, Current_Distance; /* Variables used in   */
   double x_diff, y_diff;                  /* distance comparison */

   int    Total_Comps, Comps_Remaining;  /* Comps means component */
   int    Node_Points[MAXPOINTS]; /* For storing only node points */
   int    Visited_Order[MAXPOINTS];  /* For storing  a path order */
   int    index;
   int    i, j, k;            /* Simple loop counters */
   int    done, done_inner;   /* Variables for while loop control */
   int    Nodes_To_Check;     /* Always assigned Total Comps x 2  */
   int    Best_Index;
   int    Current_X, Current_Y;
   //time_t The_time;           /* To get the program */
   //long    Calc_Time;          /* execution time     */
   clock_t start_t, end_t; //ysk
/*************************************************************************/
   /* Initialization */
```

```
    //Calc_Time = time(&The_time);
    start_t = clock(); //ysk
    done = 0;
    j = 0;
    Total_Comps = 0;
    i = 0;  /* i : an index into the array */


    /* Read all of the data from the global array out[] */
    /* Node_Points[2i] and [2i+1] point out the biginning */
    /* node point and the ending node point of the i+1_th crack */
    /* starting i = 0 */

    while(done == 0)
    {
       if((i == 0) || (out[i+1].x < 0) || ( (i > 0) && (out[i-1].x < 0) ))
       {
          Node_Points[j++] = i;
          if( out[i+1].x < 0) /* This marks the END of a crack */
             Total_Comps = Total_Comps + 1;
       }
       i = i + 1;
       if(out[i].x == -2) done = 1; /* -2 indicates this point is */
    }                               /* the end of the array       */
                                    /* Thus, the search process   */
                                    /* will be ended              */

/*************************************************************************/
/**************** FINISHED to read all of the data from the out[] *********/
/**************** CREATED an integer array, Node_Points[] which is ********/
/**************** now storing the all of the searched node points *********/
/*************************************************************************/


/* Copy the number of components to fill and node points to check */

   Comps_Remaining = Total_Comps;
   Nodes_To_Check = Total_Comps * 2; /* Each component has only 2 node points */
   Current_X = 627;                  /* Always starts at home point (0, 0)    */
   Current_Y = 414;
   k = 0;


/* Determine the order to visit the cracks to be filled in a given image */

   while(Comps_Remaining > 0)
   {
      Comps_Remaining--;   /* We will always find one */
      Best_Distance = 1000000.0;   /* Impossibly large number */

/* First, FIND the closest node point to home point(0,0), then DETERMINE */
/* the closest node point to the end node point of a current component   */
/* for traversing next crack to be filled */

      for(i = 0; i < Nodes_To_Check; i++)
      {
         if(Node_Points[i] >= 0) /* Already traversed node point which is */
```

```
         {                          /* now including a -ve value will be excluded */
            x_diff = Current_X - out[Node_Points[i]].x;
            y_diff = Current_Y - out[Node_Points[i]].y;
            Current_Distance = sqrt(x_diff*x_diff + y_diff*y_diff);
            if(Current_Distance < Best_Distance)
            {
               Best_Distance = Current_Distance;
               Best_Index = i;
            }
         }
      }

/* Visited_Order[] stores the order in which the table will visit */
/* the node points for crack filling */

      Visited_Order[k++] = Best_Index;

/* If [Best_Index % 2 == 0] means FORWARD PASS, else BACKWARD PASS */
      if((Best_Index % 2) == 0)
         Visited_Order[k++] = Best_Index + 1;     /* FORWARD PASS  */
      else Visited_Order[k++] = Best_Index - 1;  /* BACKWARD PASS */

/* Update current position */
      Current_X = out[Node_Points[Visited_Order[k-1]]].x;
      Current_Y = out[Node_Points[Visited_Order[k-1]]].y;

/* Mark it as visited by multiplying by -1, and then subtracting 1 to HANDLE 0 case
*/
      Node_Points[Visited_Order[k-2]] = -1*Node_Points[Visited_Order[k-2]]-1;
      Node_Points[Visited_Order[k-1]] = -1*Node_Points[Visited_Order[k-1]]-1;
   }

/*****************************************************************************
*/
/********** COMPLETED a correct path order through the distance comparison *********
*/
/********** CREATED an interger array, Visited_Order[] which is now including ******
*/
/********** the indices of the Node_Points[] in the generated path order   *********
*/
/*****************************************************************************
*/


/* Now all of the PATHS were stored for us, so JUST need to traverse them */
/* in the generated path order */

   j = 0;
   for(i = 0; i < Total_Comps; i++)
   {
      index = Visited_Order[i*2];

/* REMEMBER WHAT WE DID when we marked them as visited */
      path_out[j].x = out[Node_Points[index]*-1-1].x;
      path_out[j].y = out[Node_Points[index]*-1-1].y;
      j++;
      if(index%2 == 0) /* Starts at first node in the current component */
```

```
      {                 /* FORWARD direction */
        for(k = -1*Node_Points[index]; k <= -1*Node_Points[index+1]-1; k++)
        {
                        path_out[j].x = out[k].x;
                        path_out[j].y = out[k].y;
                        j++;
        }
      }
      else /* Starts at last node in the current component */
      {    /* BACKWARD direction */
        for(k = -1*Node_Points[index]-2;k >= -1*Node_Points[index-1]-1; k--)
        {
                        path_out[j].x = out[k].x;
                        path_out[j].y = out[k].y;
                        j++;
        }
      }
      path_out[j].x = -1;       /* Put '-1' to distinguish component by component
*/
      path_out[j].y = -1;
      j++;
   }
   path_out[j-1].x = -2;        /* Put '-2' to indicate this is the end of the array
*/
   path_out[j-1].y = -2;

   end_t = clock(); //ysk

/*****************************************************************************
*/
/******** COMPLETED the global array, path_out[] which is now storing all of *******
*/
/******** the x-y coordinate pairs in the generated path order.  And the table *****
*/
/******** control command is later created from this path_out[] ******************
*/
/*****************************************************************************
*/


/*****************************************************************************
*/
/***************************** Now generate the output ***************************
*/
/****** This function is optional, and are used for some experimental purposes *****
*/
/**1)Path generation output, 2)Total traversed distance, 3)Program Execution time **
*/
/*****************************************************************************
*/

   Current_X = 0;
   Current_Y = 0;
   done = 0; Crack_Length = 0.0; Idle_Length = 0.0;
   j = 0;
   while (done == 0)
   {
```

```
            //printf("Moving table (idle) from %5d, %5d to %5d , %5d\n", Current_X, C
urrent_Y,
                              //path_out[j].x, path_out[j].y);
            Idle_Length_Add = sqrt( (Current_X-path_out[j].x)*(Current_X-path_out[j].
x) +
                                    (Current_Y-path_out[j].y)*(Current_Y-path_out[j].y) )
;
            Idle_Length = Idle_Length + Idle_Length_Add;
            done_inner = 0;
            Current_X = path_out[j].x;  Current_Y = path_out[j].y;
            while(done_inner == 0)
            {
                  j++;
                  if(path_out[j].x == -1)
                     {done_inner = 1; j++;}
                  else if(path_out[j].x == -2)
                     {done = 1; done_inner = 1;}
                  else
                  {
                    //printf("Moving table (crack_traverse) from %5d, %5d to %5d, %5d
\n", Current_X,
                      //      Current_Y, path_out[j].x, path_out[j].y);
                    Crack_Length_Add = sqrt( (Current_X-path_out[j].x)*(Current_X-pat
h_out[j].x) +
                        (Current_Y-path_out[j].y)*(Current_Y-path_out[j].y) );
                  Crack_Length = Crack_Length + Crack_Length_Add;
                  Current_X = path_out[j].x;  Current_Y = path_out[j].y;
                  }
            }
   }
   //printf("\n\nThe total Idle Distance moved was %10.4lf\n", Idle_Length);
   //printf("\nThe total Crack Distance moved was %10.4lf\n", Crack_Length);
   Total_Length = Crack_Length+Idle_Length;
   //printf("\nThe total length was %10.4lf\n", Total_Length);
   //Calc_Time = time(&The_time) - Calc_Time; //ysk
   *dur = end_t - start_t;  //ysk
   //printf("\nIt took %6d seconds to do the calculation\n",Calc_Time);
   return;
}


/*main()
{

        path_plan();
        return(1);
} */
```

```
/*************************************************************************/
/*************************************************************************/
/***                          ***************************/
/***  ARMM Windows Program    ***************************/
/***                          ***************************/
/***  MODULE : ACQUIRE2.C             ***************************/
/***                          ***************************/
/***  Last Update: 9/30/96            ***************************/
/***    by Richard Greer                     ***************************/
/*************************************************************************/
/***                                                                   ***/
/***  The functions Set_Bufs and ACQ_Image were modified to acquire an  ***/
/***    image automatically into the aux. buffer without going to a    ***/
/***    box.  This was done for image switching purposes.   All other  ***/
/***    functions are as Data Translation wrote them.                  ***/
/***                                                                   ***/
/*************************************************************************/
/***                                                 ***/
/***  Acquire Menu Option Functions                   ***/
/***                                        ***/
/***      A) Acquire Setup:                        ***/
/***           1. Set_ACQsetup                     ***/
/***           2. zero_acq_setup                       ***/
/***           3. zero_wIndex                      ***/
/***           4. ACQSETMP                         ***/
/***           5. acq_config                       ***/
/***                                       ***/
/***      B) Image Acquire Setup:                      ***/
/***           1. Set_ImgAcqSetup                      ***/
/***           2.    zero_img_acq_setup                ***/
/***           3.    ImgAcqSetupMP                     ***/
/***           4.    img_acq_config                    ***/
/***                                       ***/
/***      C) Freeze Frame:                         ***/
/***           1. Stop_Video                       ***/
/***                                       ***/
/***      D) Check Acquire Complete:                        ***/
/***           1. Check_Video                      ***/
/***                                       ***/
/***      E) Enable External Trigger:                       ***/
/***           1. Trig_On                          ***/
/***           2. Trig_Off                         ***/
/***                                       ***/
/***      F) Acquisitions:                         ***/
/***        (Full Frame)                           ***/
/***           1.  ACQ_Image                       ***/
/***           2.  FOCUS_Image                     ***/
/***           3.  Set_Bufs                        ***/
/***           4.  zero_tmphdl                     ***/
/***           5.  LIVE_Image                      ***/
/***           6.  NFramesMP                       ***/
/***        (ROI)                             ***/
/***           7.  ROI_Acq_Image                       ***/
/***           8.  ROI_FOCUS_Image                     ***/
/***           9.  ROI_LIVE_Image                      ***/
/***           10. AcqRoiMP                      ***/
```

```
/***           11. roi_config                        ***/
/***           12. src_roi_config                       ***/
/***           13. dest_roi_config                      ***/
/***         (Average)                            ***/
/***           14. Average_IT                        ***/
/***           15. AverageMP                         ***/
/***         (Mini-Movie)                          ***/
/***           16. ACQ_Sequential                       ***/
/***           17. NSequencesMP                      ***/
/***           18. RUN_Sequential                       ***/
/***           19. LOOP_Sequential                      ***/
/***           20. SeqMotionMP                       ***/
/***         (Four Camera Acquire)                    ***/
/***           21. Four_ACQ_Images                      ***/
/***                                          ***/
/*****************************************************************************/
/*****************************************************************************/

/*****************************************************************************/
/***  Include Files & Function Definitions  ****************************/
/*****************************************************************************/

#include "dt51.h"

/*****************************************************************************/
/***  EXTERNAL DECLARATIONS for ALL Global Variables ***********************/
/*****************************************************************************/

/* 32-bit identifiers used to specify an auxiliary, display, overlay, or  */
/* an acquire memory buffer                                 */
extern BUF_HNDL acq_hndls[FW_NUM_ACQ_BUFS];
extern BUF_HNDL disp_hndls[FW_NUM_DISP_BUFS];
extern BUF_HNDL ovl_hndls[FW_NUM_OVL_BUFS];
extern BUF_HNDL sys_hndls[FW_NUM_SYS_BUFS];
extern BUF_HNDL src_hndl;
extern BUF_HNDL dest_hndl;
extern BUF_HNDL *tmp_hdl;

/* structures composed of 8 fields in which the routines will return       */
/* information on the size and location of the memory buffer           */
extern HNDL_STRUCT acq_hndl_struct;
extern HNDL_STRUCT disp_hndl_struct;
extern HNDL_STRUCT ovl_hndl_struct;
extern HNDL_STRUCT sys_hndl_struct;
extern HNDL_STRUCT tmp_hndl_struct;
extern HNDL_STRUCT *tmp_struct;

/* structures composed of 2 fields (the number of BUF_HNDLS and a pointer */
/* to the array of buffer handles) which are passed into the dt51_acquire */
/* and dt51_passthru routines                               */
extern BUF_HNDL_LIST acq_hndl_list;
extern BUF_HNDL_LIST disp_hndl_list;
extern BUF_HNDL_LIST ovl_hndl_list;
extern BUF_HNDL_LIST sys_hndl_list;
extern BUF_HNDL_LIST tmp_list;

/* structures composed of 4 fields in to which the routines return or set */
```

```
/* information on the size and starting location of the region of      */
/* interest in the specified memory buffer                            */
extern XY_rgn src_roi;
extern XY_rgn dest_roi;

/* structure composed of 19 fields to which the routine returns     */
/* information on the number of buffers, the size of each memory type on  */
/* the board, and other operational features                          */
extern FW_CONFIG cfg;

/* structure composed of 12 fields to which the routine returns or sets   */
/* information on the image acquisition format                        */
extern FMT fmt;

/* structure composed of 4 fields to which the routine returns or sets     */
/* information on the sync reset pulse and sync insert pulse positions     */
extern SYNC_FMT syncfmt;

/* structure composed of 4 fields to which the routine returns or sets     */
/* the image acquisition setup parameters (replaced by img_acq_setup with */
/* the release of the '-A' boards)                                    */
extern ACQ_SETUP        acq_setup;

/* structure composed of six fields to which the routine returns or sets  */
/* the image acquisition setup parameters                             */
extern IMG_ACQ_SETUP     img_acq_setup;

/* specifies the device enabling key for the routine in question */
extern u_long device;

/* generic DWORD variables used to store temporary results */
extern DWORD wIndex1, wIndex2, wIndex3, wIndex4, wIndex5, wIndex6;

/* generic string variables used to store temporary results */
extern char szString[80];
extern char szItem[20];
extern char szItem1[20];
extern char szItem2[20];
extern char szItem3[20];
extern char szItem4[20];
extern char szItem5[20];
extern char szItem6[20];
extern char szItem7[20];
extern char szItem8[20];
extern char szItem9[20];
extern char szItem10[20];
extern char szItem11[20];
extern char szItem12[20];

/* temporary variable to hold cursor */
extern HCURSOR hOldCursor;

/* used to determine if dialog box parameters are okay to use */
extern u_short flag;

/* current video & sync channels */
extern u_short vchan, vsync;
```

```
/* DT3851/52 menu status bar */
extern HWND hWndMenu;

/* generic variables used to temporarily convey a buffer or operation to  */
/* be performed                                    */
extern u_short op;

/* flag used by generic dialog box to set it up for proper routine */
extern u_short NUM_FREQ;

extern u_short FF_FOCUS;
extern u_long nframes;
extern u_short value;
extern u_short buf_num;
extern int AvgMode;
extern u_short update_flag;
extern HWND hWndMain;


/*=#########################################################################=*/
/*                                                                         */
/*                START OF ROUTINE CODE                                    */
/*                                                                         */
/*=#########################################################################=*/


/*=========================================================================*/
/*                                                                         */
/* NAME   :  Set_ACQsetup                                                  */
/* DESCRIPTION   :  This routine is to start the Dialog box that is used   */
/*          to specify the setup for acquisition and to write              */
/*          that setup to the board for the acquisition sequence.          */
/*                                                                         */
/* ARGUMENTS     : HWND   : hWnd                                           */
/*         HANDLE : hInst                                                  */
/*                                                                         */
/* RETURN VALUE : none (void)                                              */
/*                                                                         */
/*=========================================================================*/

void Set_ACQsetup(HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 WORD      Warr[5];
 FARPROC   lpfnDlgProc;

/*----------------------- Start of routine code     ----------------------*/
 zero_acq_setup();
 zero_wIndex();
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)ACQSETMP, hInst);
 DialogBox(hInst, (LPSTR)"SCM_SETUP", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);
 if (flag)
   {
    Warr[0] = op = FW_WRITE;
```

```
    acq_setup.start_field = wIndex1;
    acq_setup.acq_mode = wIndex2;
    acq_setup.invert_clk = wIndex3;
    acq_setup.invert_lf = wIndex4;

    Warr [1] = dt51_acquire_setup (device, op, &acq_setup);
    print_log ("dt51_acquire_setup (device, 0x%x, &acq_setup) = 0x%x", (LPSTR) Warr)
;
    if(Warr[1] == 0)
      acq_config ();
    else
      disp_err (Warr[1]);
   }
}


/*=============================================================================*/
/*                                        */
/* NAME  : zero_acq_setup                        */
/* DESCRIPTION   :  This routine zero's the acquire structure.         */
/*                                        */
/* ARGUMENTS     :  none (void)                        */
/*                                        */
/* RETURN VALUE :  none (void)                        */
/*                                        */
/*=============================================================================*/

void zero_acq_setup(void)
{
/*----------------------- Start of routine code     ----------------------*/
 acq_setup.start_field =
 acq_setup.acq_mode =
 acq_setup.invert_clk =
 acq_setup.invert_lf = 0L;
}


/*=============================================================================*/
/*                                        */
/* NAME  : zero_wIndex                        */
/* DESCRIPTION   : Zero indexing variables                */
/*                                        */
/* ARGUMENTS     :  none (void)                        */
/*                                        */
/* RETURN VALUE :  none (void)                        */
/*                                        */
/*=============================================================================*/

void zero_wIndex(void)
{
/*----------------------- Start of routine code     ----------------------*/
wIndex1 = 0;
wIndex2 = 0;
wIndex3 = 0;
wIndex4 = 0;
wIndex5 = 0;
wIndex6 = 0;
```

```
}


/*============================================================================*/
/*                                                                            */
/* NAME   : ACQSETMP                                                          */
/* DESCRIPTION    : This routine implements the dialog box used to specify    */
/*          the operating characteristics of the acquire sequence.            */
/*                                                                            */
/* ARGUMENTS      : HWND : hWndDlg                                            */
/*          WORD : Message                                                    */
/*          WORD : wParam                                                     */
/*          LONG : lParam                                                     */
/*                                                                            */
/* RETURN VALUE : TRUE or FALSE (BOOL)                                        */
/*                                                                            */
/* NOTE : This is representative of the board/firmware combination prior      */
/*      to the REV A architecture.                                            */
/*                                                                            */
/*============================================================================*/

BOOL FAR PASCAL ACQSETMP(HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------ declaration of local variables ----------------------*/
 static HANDLE hWndCombo1;
 static HANDLE hWndCombo2;
 static HANDLE hWndCombo3;
 static HANDLE hWndCombo4;
 static int wIndex;
 static short i;
 WORD Warr[5];
 static DWORD sf[] = { FW_ACQ_ST_FLD_EVEN,
            FW_ACQ_ST_FLD_ODD,
            FW_ACQ_ST_FLD_NEXT,
            FW_ACQ_ST_FLD_NI,
              };
 static DWORD am[] = { FW_ACQ_MODE_INT,
            FW_ACQ_MODE_EXT,
            FW_ACQ_MODE_SS,
              };
 static DWORD ic[] = { FW_ENABLE,
            FW_DISABLE,
              };
 static DWORD ilf[] = { FW_ENABLE,
             FW_DISABLE,
               };

/*----------------------- Start of routine code     ----------------------*/
 switch(Message)
   {
    case WM_INITDIALOG:
         hWndCombo1    = GetDlgItem(hWndDlg, SCM_ASF);
         hWndCombo2    = GetDlgItem(hWndDlg, SCM_SAM);
         hWndCombo3    = GetDlgItem(hWndDlg, SCM_ICS);
         hWndCombo4    = GetDlgItem(hWndDlg, SCM_ILF);

         SendMessage (hWndCombo1,
```

```
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_ACQ_ST_FLD_EVEN");
     SendMessage (hWndCombo1,
                                CB_ADDSTRING,
                                0,
                                (LONG)(LPSTR) "FW_ACQ_ST_FLD_ODD");
  SendMessage (hWndCombo1,
                                   CB_ADDSTRING,
                                0,
                                (LONG)(LPSTR) "FW_ACQ_ST_FLD_NEXT");
           SendMessage (hWndCombo1,
                                CB_ADDSTRING,
                                0,
                                (LONG)(LPSTR) "FW_ACQ_ST_FLD_NI");

 SendMessage (hWndCombo2,
                                 CB_ADDSTRING,
                                 0,
                                 (LONG)(LPSTR) "FW_ACQ_MODE_INT");
       SendMessage (hWndCombo2,
                                CB_ADDSTRING,
                                0,
                                (LONG)(LPSTR) "FW_ACQ_MODE_EXT");
       SendMessage (hWndCombo2,
                                CB_ADDSTRING,
                                0,
                                (LONG)(LPSTR) "FW_ACQ_MODE_SS");

  SendMessage (hWndCombo3,
                                CB_ADDSTRING,
                                0,
                                (LONG)(LPSTR) "FW_ENABLE");
  SendMessage (hWndCombo3,
                                CB_ADDSTRING,
                                0,
                                (LONG)(LPSTR) "FW_DISABLE");

  SendMessage (hWndCombo4,
                                CB_ADDSTRING,
                                0,
                                (LONG)(LPSTR) "FW_ENABLE");
  SendMessage (hWndCombo4,
                                CB_ADDSTRING,
                                0,
                                (LONG)(LPSTR) "FW_DISABLE");

  Warr [0] = dt51_acquire_setup (device, FW_READ, &acq_setup);
         if (Warr[0] != 0)
           {
disp_err (Warr[0]);
             flag = FALSE;
             EndDialog(hWndDlg, FALSE);
             break;
           }
  else
    {
```

```
                for(i=0; i<4; i++)
            if(acq_setup.start_field == sf[i])
              {
               SendMessage (hWndCombo1,
                                          CB_SETCURSEL,
                                           i,
                                           0L);

               wIndex1 = sf[i];
               SendMessage (hWndCombo1,
                                            CB_GETLBTEXT,
                                            i,
                                            (LONG)(LPSTR) szItem12);

              }
                for(i=0; i<3; i++)
            if(acq_setup.acq_mode == am[i])
              {
               SendMessage (hWndCombo2,
                                          CB_SETCURSEL,
                                          i,
                                          0L);

               wIndex2 = am[i];
               SendMessage (hWndCombo2,
                                          CB_GETLBTEXT,
                                          i,
                                          (LONG)(LPSTR) szItem11);

                        }
                for(i=0; i<2; i++)
            if(acq_setup.invert_clk == ic[i])
              {
               SendMessage (hWndCombo3,
                                          CB_SETCURSEL,
                                          i,
                                          0L);

               wIndex3 = ic[i];
               SendMessage (hWndCombo3,
                                          CB_GETLBTEXT,
                                          i,
                                          (LONG)(LPSTR) szItem10);

              }
                for(i=0; i<2; i++)
            if(acq_setup.invert_lf == ilf[i])
              {
               SendMessage (hWndCombo4,
                                          CB_SETCURSEL,
                                          i,
                                          0L);

               wIndex4 = ilf[i];
               SendMessage (hWndCombo4,
                                          CB_GETLBTEXT,
                                          i,
                                          (LONG)(LPSTR) szItem9);

                        }
             }
          cwCenter(hWndDlg, 0);
          break;

        case WM_COMMAND:
```

```
switch(wParam)
  {
   case IDOK:
    flag = TRUE;
    EndDialog(hWndDlg, TRUE);
    break;

   case IDCANCEL:
    flag = FALSE;
    EndDialog(hWndDlg, FALSE);
    break;

   case SCM_ASF:
     {
   switch(HIWORD(lParam))
       {
        case CBN_SELCHANGE:
               wIndex = (WORD) SendMessage (LOWORD (lParam),
                       CB_GETCURSEL,
                       0,
                       0L);
               wIndex1 = sf[wIndex];
         SendMessage (LOWORD (lParam),
                                          CB_GETLBTEXT,
                                             wIndex,
                                             (LONG)(LPSTR) szItem12);
         break;
       }
     }
     break;

   case SCM_SAM:
     {
   switch(HIWORD(lParam))
       {
        case CBN_SELCHANGE:
               wIndex = (WORD) SendMessage (LOWORD (lParam),
                       CB_GETCURSEL,
                       0,
                       0L);
               wIndex2 = am[wIndex];
         SendMessage (LOWORD (lParam),
                                          CB_GETLBTEXT,
                                             wIndex,
                                             (LONG)(LPSTR) szItem11);
         break;
       }
     }
     break;

   case SCM_ICS:
     {
   switch(HIWORD(lParam))
       {
        case CBN_SELCHANGE:
               wIndex = (WORD) SendMessage (LOWORD (lParam),
                       CB_GETCURSEL,
```

```
                                0,
                                0L);
                        wIndex3 = ic[wIndex];
                    SendMessage (LOWORD (lParam),
                                                    CB_GETLBTEXT,
                                                       wIndex,
                                                       (LONG)(LPSTR) szItem10);
                        break;
                      }
                    }
                    break;

                  case SCM_ILF:
                    {
                  switch(HIWORD(lParam))
                      {
                       case CBN_SELCHANGE:
                              wIndex = (WORD) SendMessage (LOWORD (lParam),
                                      CB_GETCURSEL,
                                      0,
                                      0L);
                              wIndex4 = ilf[wIndex];
                       SendMessage (LOWORD (lParam),
                                                    CB_GETLBTEXT,
                                                       wIndex,
                                                       (LONG)(LPSTR) szItem9);
                        break;
                      }
                    }
                    break;
                }
              break;

            default:
              return FALSE;
           }
         return TRUE;
        }


        /*=========================================================================*/
        /*                                     */
        /* NAME   : acq_config                         */
        /* DESCRIPTION    : This routine prints the acquire setup settings to the   */
        /*          I/O log window.                    */
        /*                                     */
        /* ARGUMENTS      : none (void)                        */
        /*                                     */
        /* RETURN VALUE : none (void)                          */
        /*                                     */
        /*=========================================================================*/

        void acq_config (void)
        {
        /*------------------ declaration of local variables ---------------------*/
        char TmpBuf[100];
```

```
/*----------------------- Start of routine code     ----------------------*/

 sprintf (TmpBuf, " * acq_setup.start_field = 0x%lx,  *** %s ***", acq_setup.start_f
ield, szItem12);
 print_log (TmpBuf, (LPSTR) NULL);
 sprintf (TmpBuf, " * acq_setup.acq_mode = 0x%lx,     *** %s ***", acq_setup.acq_mod
e, szItem11);
 print_log (TmpBuf, (LPSTR) NULL);
 sprintf (TmpBuf, " * acq_setup.invert_clk = 0x%lx,   *** %s ***", acq_setup.invert_
clk, szItem10);
 print_log (TmpBuf, (LPSTR) NULL);
 sprintf (TmpBuf, " * acq_setup.invert_lf = 0x%lx,    *** %s ***", acq_setup.invert_
lf, szItem9);
 print_log (TmpBuf, (LPSTR) NULL);
}


/*=========================================================================*/
/*                                      */
/* NAME  : Set_ImgAcqSetup                          */
/* DESCRIPTION   : This routine is used to initialize the dialog box       */
/*         used to support IMAGE ACQUIRE SETUP type of acquire     */
/*         setup support. This us used on the REV A and later      */
/*         hardware/firmware setup.                 */
/*                                      */
/* ARGUMENTS     : HWND   : hWnd                        */
/*         HANDLE : hInst                       */
/*                                      */
/* RETURN VALUE : none (void)                       */
/*                                      */
/*=========================================================================*/

void Set_ImgAcqSetup (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 WORD      Warr[5];
 FARPROC   lpfnDlgProc;

/*----------------------- Start of routine code     ----------------------*/
 zero_img_acq_setup();
 zero_wIndex();
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)ImgAcqSetupMP, hInst);
 DialogBox(hInst, (LPSTR)"SCM_IMGACQSETUP", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);

 if (flag)
   {
   img_acq_setup.start_field = wIndex1;
   img_acq_setup.img_acq_mode = wIndex2;
   img_acq_setup.invert_ext_clk = wIndex3;
   img_acq_setup.invert_line_sync = wIndex4;
     img_acq_setup.invert_frame_sync = wIndex5;
     img_acq_setup.clock_mode = wIndex6;

   Warr[0] = op = FW_WRITE;
   Warr [1] = dt51_image_acquire_setup (device, op, &img_acq_setup);
```

```
   print_log ("dt51_image_acquire_setup (device, 0x%x, &img_acq_setup) = 0x%x", (LPS
TR) Warr);
   if(Warr[1] != 0)
     disp_err (Warr[1]);
    else
     img_acq_config ();
   }
}


/*=========================================================================*/
/*                                        */
/* NAME  : zero_img_acq_setup                          */
/* DESCRIPTION   : This routine zeros the img_acq_setup data structure.    */
/*                                        */
/* ARGUMENTS     : none (void)                         */
/*                                        */
/* RETURN VALUE : none (void)                          */
/*                                        */
/*=========================================================================*/

void zero_img_acq_setup (void)
{
/*----------------------- Start of routine code     ----------------------*/
 img_acq_setup.start_field = 0L;
 img_acq_setup.img_acq_mode =   0L;
 img_acq_setup.invert_ext_clk = 0L;
 img_acq_setup.invert_line_sync = 0L;
 img_acq_setup.invert_frame_sync = 0L;
 img_acq_setup.clock_mode = 0L;
}


/*=========================================================================*/
/*                                        */
/* NAME  : ImgAcqSetupMP                          */
/* DESCRIPTION   : This routine implements the dialog box used to specify  */
/*         the operating characteristics of the acquire sequence.  */
/*                                        */
/* ARGUMENTS     : HWND : hWndDlg                      */
/*         WORD : Message                         */
/*         WORD : wParam                          */
/*         LONG : lParam                          */
/*                                        */
/* RETURN VALUE : TRUE or FALSE (BOOL)                     */
/*                                        */
/* NOTE : This is representative of the board/firmware combination     */
/*     after and including the REV A architecture.             */
/*                                        */
/*=========================================================================*/

BOOL FAR PASCAL ImgAcqSetupMP (HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------ declaration of local variables ---------------------*/
/*----------------------- Start of routine code     ----------------------*/
 HWND hWndCombo1;
 HWND hWndCombo2;
```

```
HWND hWndCombo3;
HWND hWndCombo4;
HWND hWndCombo5;
HWND hWndCombo6;

int wIndex;

short i;
WORD Warr[5];
DWORD sf[] = { FW_ACQ_ST_FLD_EVEN,
        FW_ACQ_ST_FLD_ODD,
        FW_ACQ_ST_FLD_NEXT,
        FW_ACQ_ST_FLD_NI,
          };
DWORD iam[] = { FW_IMG_ACQ_MODE_RS170,
         FW_IMG_ACQ_MODE_SS,
         FW_IMG_ACQ_MODE_ASYNC_RST,
        };
DWORD iec[] = { FW_ENABLE,
         FW_DISABLE,
        };
DWORD ils[] = { FW_ENABLE,
         FW_DISABLE,
        };
DWORD ifs[] = { FW_ENABLE,
         FW_DISABLE,
        };
DWORD cm[] =  { FW_IMG_CLK_MODE_INT,
         FW_IMG_CLK_MODE_EXT,
        };

/*------------------ declaration of local variables ----------------------*/
/*------------------------ Start of routine code     ----------------------*/
switch(Message)
  {
   case WM_INITDIALOG:
     hWndCombo1      = GetDlgItem(hWndDlg, SCM_IAS_ASF);
     hWndCombo2      = GetDlgItem(hWndDlg, SCM_IAS_IAM);
     hWndCombo3      = GetDlgItem(hWndDlg, SCM_IAS_IECS);
     hWndCombo4      = GetDlgItem(hWndDlg, SCM_IAS_IL);
     hWndCombo5      = GetDlgItem(hWndDlg, SCM_IAS_IF);
     hWndCombo6      = GetDlgItem(hWndDlg, SCM_IAS_CM);

     SendMessage (hWndCombo1,
              CB_ADDSTRING,
              0,
              (LONG)(LPSTR) "FW_ACQ_ST_FLD_EVEN");
       SendMessage (hWndCombo1,
              CB_ADDSTRING,
              0,
              (LONG)(LPSTR) "FW_ACQ_ST_FLD_ODD");
     SendMessage (hWndCombo1,
              CB_ADDSTRING,
              0,
              (LONG)(LPSTR) "FW_ACQ_ST_FLD_NEXT");
     SendMessage (hWndCombo1,
              CB_ADDSTRING,
```

```
            0,
            (LONG)(LPSTR) "FW_ACQ_ST_FLD_NI");

SendMessage (hWndCombo2,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_IMG_ACQ_MODE_RS170");
SendMessage (hWndCombo2,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_IMG_ACQ_MODE_SS");
SendMessage (hWndCombo2,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_IMG_ACQ_MODE_ASYNC_RST");

SendMessage (hWndCombo3,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_ENABLE");
SendMessage (hWndCombo3,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_DISABLE");

SendMessage (hWndCombo4,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_ENABLE");
SendMessage (hWndCombo4,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_DISABLE");

SendMessage (hWndCombo5,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_ENABLE");
SendMessage (hWndCombo5,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_DISABLE");

SendMessage (hWndCombo6,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_IMG_CLK_MODE_INT");
SendMessage (hWndCombo6,
            CB_ADDSTRING,
            0,
            (LONG)(LPSTR) "FW_IMG_CLK_MODE_EXT");

Warr [0] = dt51_image_acquire_setup (device, FW_READ, &img_acq_setup);
    if (Warr[0] != 0)
     {
   disp_err (Warr[0]);
   flag = FALSE;
```

```
      EndDialog(hWndDlg, FALSE);
      break;
        }
    else
      {
        for(i=0; i<4; i++)
       if(img_acq_setup.start_field == sf[i])
      {
       SendMessage (hWndCombo1,CB_SETCURSEL,i,0L);
       wIndex1 = sf[i];
       SendMessage (hWndCombo1,CB_GETLBTEXT,i,(LONG)(LPSTR) szItem6);
      }
        for(i=0; i<3; i++)
       if(img_acq_setup.img_acq_mode == iam[i])
      {
       SendMessage (hWndCombo2,CB_SETCURSEL,i,0L);
       wIndex2 = iam[i];
       SendMessage (hWndCombo2,CB_GETLBTEXT,i,(LONG)(LPSTR) szItem11);
            }
        for(i=0; i<2; i++)
       if(img_acq_setup.invert_ext_clk == iec[i])
      {
       SendMessage (hWndCombo3,CB_SETCURSEL,i,0L);
       wIndex3 = iec[i];
       SendMessage (hWndCombo3,CB_GETLBTEXT,i,(LONG)(LPSTR) szItem10);
      }
        for(i=0; i<2; i++)
       if(img_acq_setup.invert_line_sync == ils[i])
      {
       SendMessage (hWndCombo4,CB_SETCURSEL,i,0L);
       wIndex4 = ils[i];
       SendMessage (hWndCombo4,CB_GETLBTEXT,i,(LONG)(LPSTR) szItem9);
            }
        for(i=0; i<2; i++)
       if(img_acq_setup.invert_frame_sync == ifs[i])
      {
       SendMessage (hWndCombo5,CB_SETCURSEL,i,0L);
       wIndex5 = ifs[i];
       SendMessage (hWndCombo5,CB_GETLBTEXT,i,(LONG)(LPSTR) szItem8);
            }
        for(i=0; i<2; i++)
       if(img_acq_setup.clock_mode == cm[i])
      {
       SendMessage (hWndCombo6,CB_SETCURSEL,i,0L);
       wIndex6 = cm[i];
       SendMessage (hWndCombo6,CB_GETLBTEXT,i,(LONG)(LPSTR) szItem7);
            }
    }

  cwCenter(hWndDlg, 0);
  break;

case WM_COMMAND:
  switch(wParam)
    {
     case IDOK:
      flag = TRUE;
```

```
            EndDialog(hWndDlg, TRUE);
            break;

          case IDCANCEL:
            flag = FALSE;
            EndDialog(hWndDlg, FALSE);
            break;

          case SCM_IAS_ASF:
            switch(HIWORD(lParam))
               {
                case CBN_SELCHANGE:
                     wIndex = (WORD) SendMessage (LOWORD(lParam),CB_GETCURSEL,0,0L);
                     wIndex1 = sf[wIndex];
                 SendMessage (LOWORD(lParam),CB_GETLBTEXT,wIndex,(LONG)(LPSTR) szItem6
);
                  break;
               }
            break;

          case SCM_IAS_IAM:
            switch(HIWORD(lParam))
               {
                case CBN_SELCHANGE:
                 wIndex = (WORD) SendMessage (LOWORD (lParam),CB_GETCURSEL,0,0L);
                     wIndex2 = iam[wIndex];
                 SendMessage (LOWORD (lParam),CB_GETLBTEXT,wIndex,(LONG)(LPSTR) szItem
11);
                  break;
               }
            break;

          case SCM_IAS_IECS:
            switch(HIWORD(lParam))
               {
                case CBN_SELCHANGE:
                        wIndex = (WORD) SendMessage (LOWORD(lParam),CB_GETCURSEL,0,0L)
;
                     wIndex3 = iec[wIndex];
                 SendMessage (LOWORD(lParam),CB_GETLBTEXT,wIndex,(LONG)(LPSTR) szIteml
0);
                  break;
               }
            break;

          case SCM_IAS_IL:
            switch(HIWORD(lParam))
               {
                case CBN_SELCHANGE:
                     wIndex = (WORD) SendMessage (LOWORD (lParam),CB_GETCURSEL,0,0L);
                     wIndex4 = ils[wIndex];
                 SendMessage (LOWORD (lParam),CB_GETLBTEXT,wIndex,(LONG)(LPSTR) szItem
9);
                  break;
               }
            break;
```

```
          case SCM_IAS_IF:
           switch(HIWORD(lParam))
              {
               case CBN_SELCHANGE:
                    wIndex = (WORD) SendMessage (LOWORD (lParam),CB_GETCURSEL,0,0L);
                    wIndex5 = ifs[wIndex];
                SendMessage (LOWORD (lParam),CB_GETLBTEXT,wIndex,(LONG)(LPSTR) szItem
8);
                 break;
              }
           break;

          case SCM_IAS_CM:
           switch(HIWORD(lParam))
              {
               case CBN_SELCHANGE:
                    wIndex = (WORD) SendMessage (LOWORD (lParam),CB_GETCURSEL,0,0L);
                    wIndex6 = cm[wIndex];
                SendMessage (LOWORD (lParam),CB_GETLBTEXT,wIndex,(LONG)(LPSTR) szItem
7);
                 break;
              }
           break;
         }
       break;

     default:
      return FALSE;
    }
 return TRUE;
}


/*=============================================================================*/
/*                                        */
/* NAME  : img_acq_config                        */
/* DESCRIPTION   : This routine writes the img_acq_config values to    */
/*         the I/O log window.                        */
/*                                        */
/* ARGUMENTS     : none (void)                        */
/*                                        */
/* RETURN VALUE : none (void)                        */
/*                                        */
/*=============================================================================*/

void img_acq_config (void)
{
/*------------------ declaration of local variables ----------------------*/
char TmpBuf[100];

/*----------------------- Start of routine code     ----------------------*/
 sprintf (TmpBuf, " * img_acq_setup.start_field = 0x%lx,      *** %s ***", img_acq_s
etup.start_field, szItem6);
 print_log (TmpBuf, (LPSTR) NULL);
 sprintf (TmpBuf, " * img_acq_setup.img_acq_mode = 0x%lx,     *** %s ***", img_acq_s
etup.img_acq_mode, szItem11);
 print_log (TmpBuf, (LPSTR) NULL);
```

```
 sprintf (TmpBuf, " * img_acq_setup.invert_ext_clk = 0x%lx,    *** %s ***", img_acq_s
etup.invert_ext_clk, szItem10);
 print_log (TmpBuf, (LPSTR) NULL);
 sprintf (TmpBuf, " * img_acq_setup.invert_line_sync = 0x%lx, *** %s ***", img_acq_s
etup.invert_line_sync, szItem9);
 print_log (TmpBuf, (LPSTR) NULL);
 sprintf (TmpBuf, " * img_acq_setup.invert_frame_sync = 0x%lx,*** %s ***", img_acq_s
etup.invert_frame_sync, szItem8);
 print_log (TmpBuf, (LPSTR) NULL);
 sprintf (TmpBuf, " * img_acq_setup.clock_mode = 0x%lx,        *** %s ***", img_acq_s
etup.clock_mode, szItem7);
 print_log (TmpBuf, (LPSTR) NULL);
}


/*============================================================================*/
/*                                                                            */
/* NAME  : Stop_Video                                                         */
/* DESCRIPTION   : This routine stops the live video command.                 */
/*                                                                            */
/* ARGUMENTS     : none (void)                                                */
/*                                                                            */
/* RETURN VALUE : none (void)                                                 */
/*                                                                            */
/*============================================================================*/

void Stop_Video (void)
{
/*------------------ declaration of local variables ----------------------*/
 WORD        Warr[3];

/*----------------------- Start of routine code     ----------------------*/
 FF_FOCUS = FALSE;
 print_log ("", (LPSTR) NULL);
 Warr[0]= dt51_freeze_frame (device);
 print_log (" dt51_freeze_frame (device) = 0x%x", (LPSTR) Warr);
 if(Warr[0] != 0)
   disp_err (Warr[0]);
}


/*============================================================================*/
/*                                                                            */
/* NAME  : Check_Video                                                        */
/* DESCRIPTION   : This routine checks the current state of the video         */
/*         acquisition process.                                               */
/*                                                                            */
/* ARGUMENTS     : none (void)                                                */
/*                                                                            */
/* RETURN VALUE : none (void)                                                 */
/*                                                                            */
/*============================================================================*/

void Check_Video (void)
{
/*------------------ declaration of local variables ----------------------*/
 WORD        Warr[3];
```

```
/*----------------------- Start of routine code     ----------------------*/
 print_log ("", (LPSTR) NULL);
 Warr[0] = dt51_test_acq_done (device, &value);
 print_log ("dt51_test_acq_done (device, value) = 0x%x", (LPSTR)Warr);
 if(Warr[0] == 0)
   {
    print_log (" * Current State : 0x%x", (LPSTR) &value);
    if(value == 0)
      print_log (" * Acquisition in Progress", (LPSTR) NULL);
    else
      print_log (" * Acquisition is Complete", (LPSTR) NULL);
   }
 else
   disp_err (Warr[0]);
}


/*=========================================================================*/
/*                                        */
/* NAME  : Trig_On                            */
/* DESCRIPTION   : This routine enables the external trigger mode.     */
/*                                        */
/* ARGUMENTS     : HWND : hWnd                          */
/*                                        */
/* RETURN VALUE : none (void)                           */
/*                                        */
/*=========================================================================*/

void Trig_On (HWND hWnd)
{
/*------------------ declaration of local variables ---------------------*/
 WORD        Warr[3];
 HMENU       hMenu;

/*----------------------- Start of routine code     ----------------------*/
 print_log ("", (LPSTR) NULL);
 Warr[0] = FW_ENABLE;
 Warr[1] = dt51_select_external_trigger (device, FW_ENABLE);
 print_log ("dt51_select_external_trigger (device, 0x%x) = 0x%x", (LPSTR)Warr);
 if(Warr[1] == 0)
  {
   hMenu = GetMenu(hWnd);
   ModifyMenu(hMenu, IDM_A_ENABLE_TRIGGER, MF_BYCOMMAND, IDM_A_DISABLE_TRIGGER, "&Di
sable Ext. Trig \tDel");
   print_log (" * Trigger is Enabled", (LPSTR) NULL);
  }
 else
  disp_err (Warr[1]);
}


/*=========================================================================*/
/*                                       */
/* NAME  : Trig_Off                           */
/* DESCRIPTION   : This routine disables the external trigger mode.    */
/*                                       */
```

```
/* ARGUMENTS      : HWND : hWnd                                    */
/*                                                                 */
/* RETURN VALUE : none (void)                                      */
/*                                                                 */
/*=================================================================*/

void Trig_Off (HWND hWnd)
{
/*------------------ declaration of local variables ----------------------*/
/*------------------------ Start of routine code     ----------------------*/
 WORD         Warr[3];
 HMENU       hMenu;

/*------------------ declaration of local variables ----------------------*/
/*------------------------ Start of routine code     ----------------------*/
 print_log ("", (LPSTR) NULL);
 Warr[0] = FW_DISABLE;
 Warr[1] = dt51_select_external_trigger (device, FW_DISABLE);
 print_log ("dt51_select_external_trigger (device, 0x%x) = 0x%x", (LPSTR)Warr);
 if(Warr[1] == 0)
  {
   hMenu = GetMenu(hWnd);
   ModifyMenu(hMenu, IDM_A_DISABLE_TRIGGER, MF_BYCOMMAND, IDM_A_ENABLE_TRIGGER, "&En
able Ext. Trig \tIns");
   print_log (" * Trigger is Disabled", (LPSTR) NULL);
  }
 else
  disp_err (Warr[1]);
}


/*=================================================================*/
/*                                                                 */
/* NAME   : ACQ_Image                                              */
/* DESCRIPTION    : This routine starts the acquire process.       */
/*                                                                 */
/* ARGUMENTS      : HWND    : hWnd                                 */
/*          HANDLE : hInst                                         */
/*                                                                 */
/* RETURN VALUE : none (void)                                      */
/*                                                                 */
/*=================================================================*/

void ACQ_Image (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 WORD       Warr[5];

/*------------------------ Start of routine code     ----------------------*/
 if(Get_DTconfig())
   {
     //Set_Bufs (hWnd, hInst);
     //changes by R. Greer 9/4/96
     zero_tmphdl ();
     tmp_struct = &tmp_hndl_struct;
     buf_num = 0;
     op = FW_SYSBUF;
```

```
      flag = TRUE;
      Hand_IT();
      BInfo_IT();
      //Changes by R Greer 9/4/96

      if(flag)
        {
       Warr[0] = dt51_get_set_format_memory (device,
                         FW_READ,
                         &fmt);
       print_log ("dt51_get_set_format_memory (device, FW_READ, &fmt) = 0x%x", (LPSTR
)Warr);
       if(Warr[0] == 0)
         {
              hOldCursor = SetCursor (LoadCursor(NULL, IDC_WAIT));
           format_center();
           tmp_list.n = 1;
           tmp_list.hndl_list = tmp_hdl;

           if(op == FW_ACQBUF)
         Warr[0] = dt51_acquire (device, &src_roi, (BUF_HNDL_LIST _far *)-1L, &dest_r
oi, 1);
           else
         Warr[0] = dt51_acquire (device, &src_roi, &tmp_list, &dest_roi, 1);

           print_log (" dt51_acquire (device, src_roi, tmp_list, dest_roi, 1) = 0x%x"
, (LPSTR) Warr);
           if(Warr[0] != 0)
             disp_err (Warr[0]);
          SetCursor (hOldCursor);
         }
       else
         disp_err (Warr[0]);
        }
   }
}


/*=========================================================================*/
/*                                        */
/* NAME  : FOCUS_Image                          */
/* DESCRIPTION   : This routine performs the passthrough function via      */
/*         multiple image acquisitions.                */
/*                                        */
/* ARGUMENTS     : HWND   : hWnd                        */
/*         HANDLE : hInst                       */
/*                                        */
/* RETURN VALUE : none (void)                         */
/*                                        */
/*=========================================================================*/

void FOCUS_Image (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 WORD      Warr[5];

/*---------------------- Start of routine code     ----------------------*/
```

```
 NUM_FREQ = 2;
 print_log ("", (LPSTR) NULL);
 nframes = 1;
 if(Get_DTconfig())
   {
    Set_Bufs (hWnd, hInst);
    if(flag)
      {
     Warr[0] = dt51_get_set_format_memory (device,
                          FW_READ,
                          &fmt);
     print_log ("dt51_get_set_format_memory (device, FW_READ, &fmt) = 0x%x", (LPSTR)
Warr);
     if(Warr[0] == 0)
       {
         format_center();
         tmp_list.n = 1;
         tmp_list.hndl_list = tmp_hdl;

         Warr[0] = (int)nframes;
         FF_FOCUS = TRUE;
         while(FF_FOCUS)
           {
           Warr[1] = dt51_acquire (device, &src_roi, &tmp_list, &dest_roi, nframes);
                    if(Warr[1] != 0)
                      {
                         FF_FOCUS = FALSE;
                        break;
                        }
           SpinTheMSGLoop ();
             }

         if(!flag)
           PostQuitMessage (0);
         print_log ("dt51_acquire (device, src_roi, tmp_list, dest_roi, 0x%x) = 0x%x
", (LPSTR) Warr);
                 if(Warr[1] != 0)
           disp_err (Warr[1]);
       }
     else
       disp_err (Warr[0]);
      }
   }
}

/*=========================================================================*/
/*                                        */
/* NAME  : Set_Bufs                           */
/* DESCRIPTION   : This routine starts the dialog box that is used to      */
/*         specify hwich buffer to acquire to.            */
/*                                        */
/* ARGUMENTS     : HWND   : hWnd                        */
/*         HANDLE : hInst                      */
/*                                        */
/* RETURN VALUE : none (void)                         */
/*                                        */
/*=========================================================================*/
```

```
void Set_Bufs (HWND hWnd, HANDLE hInst)
{
/*------------------- declaration of local variables ----------------------*/
FARPROC lpfnDlgProc;

/*------------------------ Start of routine code     ----------------------*/
zero_tmphdl ();
tmp_struct = &tmp_hndl_struct;

print_log ("", (LPSTR) NULL);
/*Taken out by Richard Greer */
lpfnDlgProc = MakeProcInstance((FARPROC)EditBMP, hInst);
DialogBox(hInst, (LPSTR)"SCM_BSN", hWnd, lpfnDlgProc);
FreeProcInstance(lpfnDlgProc);
if(flag)
   {

/* The following code has been modified to prevent the user from having
to mess with a dialog box.  It insteads automatically assumes you are using
the display buffer */
    buf_num = atoi(szItem);
    //op = FW_DISPBUF;//added by Richard Greer 2/5/96
    //flag = TRUE;  //added by Richard Greer 2/5/96
    Hand_IT();
    BInfo_IT();
   }
}


/*=========================================================================*/
/*                                        */
/* NAME  : zero_tmphdl                          */
/* DESCRIPTION   : This routine zero's the buffer handle descriptor    */
/*         structure.                           */
/*                                        */
/* ARGUMENTS     : none (void)                          */
/*                                        */
/* RETURN VALUE : none (void)                           */
/*                                        */
/*=========================================================================*/

void zero_tmphdl (void)
{
/*------------------- declaration of local variables ----------------------*/
/*------------------------ Start of routine code     ----------------------*/
 tmp_hndl_struct.flags = 0L;
 tmp_hndl_struct.width = 0L;
 tmp_hndl_struct.height = 0L;
 tmp_hndl_struct.pitch = 0L;
 tmp_hndl_struct.psize = 0L;
 tmp_hndl_struct.lock = 0L;
 tmp_hndl_struct.check_word = 0;
}


/*=========================================================================*/
```

```
/*                                                          */
/* NAME  : LIVE_Image                                       */
/* DESCRIPTION   : This routine is used to initialize the dialog box to    */
/*         support the live passthru option of the board.      */
/*                                                          */
/* ARGUMENTS     : HWND   : hWnd                            */
/*         HANDLE : hInst                                   */
/*                                                          */
/* RETURN VALUE : none (void)                               */
/*                                                          */
/*==========================================================================*/

void LIVE_Image (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 WORD      Warr[5];
 FARPROC   lpfnDlgProc;

/*----------------------- Start of routine code     ----------------------*/
 NUM_FREQ = 2;
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)NFramesMP, hInst);
 DialogBox(hInst, (LPSTR)"SCM_FRAMES", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);
 if(flag)
   {
     hOldCursor = SetCursor(LoadCursor(NULL,IDC_WAIT));
     nframes = atoi (szItem);
     if(Get_DTconfig())
       {
      Warr[0] = dt51_get_set_format_memory (device,
                        FW_READ,
                        &fmt);
      print_log ("dt51_get_set_format_memory (device, FW_READ, &fmt) = 0x%x", (LPSTR
)Warr);
      if(Warr[0] == 0)
        {
                format_center();
          Warr[0] = (int)nframes;
          Warr[1] = dt51_passthru (device, &src_roi, &dest_roi, nframes);
          print_log ("dt51_passthru (device, src_roi, dest_roi, 0x%x) = 0x%x", (LPST
R) Warr);
                 if(Warr[1] != 0)
                   disp_err (Warr[1]);
        }
      else
        disp_err (Warr[0]);
       }
     SetCursor (hOldCursor);
   }
}


/*==========================================================================*/
/*                                                          */
/* NAME  : NFramesMP                                        */
/* DESCRIPTION   : This Dialog support routine is used to support the      */
```

```
/*         specification of a single value from a dialog box.      */
/*                                                                 */
/* ARGUMENTS     : HWND : hWndDlg                                  */
/*         WORD : Message                                          */
/*         WORD : wParam                                           */
/*         LONG : lParam                                           */
/*                                                                 */
/* RETURN VALUE : TRUE or FALSE (BOOL)                             */
/*                                                                 */
/*=================================================================*/

BOOL FAR PASCAL NFramesMP (HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------ declaration of local variables ----------------------*/
static HANDLE Snv;
static u_short MaxB, Strt;

/*----------------------- Start of routine code     ----------------------*/
 switch(Message)
   {
    case WM_INITDIALOG:
      Snv = GetDlgItem(hWndDlg, 103);
      switch(NUM_FREQ)
        {
         case 1:
          strcpy(szString, "Free Auxiliary Buffer");
          SetDlgItemInt(hWndDlg, SCM_NUMBER, 1, FALSE);
                  Strt = 1;
                      MaxB = cfg.sys_bufs;
          break;
         case 2:
          strcpy(szString, "Every Nth Frame");
          SetDlgItemInt(hWndDlg, SCM_NUMBER, 1, FALSE);
                  Strt = 0;
                      MaxB = 60;
          break;
         case 3:
          strcpy(szString, "Threshold Pixel Value");
          SetDlgItemInt(hWndDlg, SCM_NUMBER, 0, FALSE);
                  Strt = 0;
                      MaxB = 255;
          break;
         case 4:
          strcpy(szString, "Number to Acquire");
          SetDlgItemInt(hWndDlg, SCM_NUMBER, 15, FALSE);
                  Strt = 0;
                      MaxB = 255;
          break;
        }
      SetWindowText (hWndDlg, (LPSTR)szString);
      cwCenter(hWndDlg, 0);
     break;

    case WM_COMMAND:
      switch(wParam)
        {
         case IDOK:
```

```
            flag = TRUE;
            GetDlgItemText (hWndDlg, SCM_NUMBER, szItem, 20);
            EndDialog(hWndDlg, TRUE);
            break;
           case IDCANCEL:
            flag = FALSE;
            EndDialog(hWndDlg, FALSE);
            break;
          }
        break;

      case WM_VSCROLL:
        switch(wParam)
          {
                  case SB_LINEDOWN:
          if(Snv == HIWORD(lParam))
                        decrease_val(hWndDlg, SCM_NUMBER, Strt, MaxB);
                     break;
                  case SB_LINEUP:
          if(Snv == HIWORD(lParam))
                        increase_val(hWndDlg, SCM_NUMBER, Strt, MaxB);
          break;
          }
        break;
      default:
       return FALSE;
     }
return TRUE;
}


/*=========================================================================*/
/*                                        */
/* NAME  : ROI_Acq_Image                          */
/* DESCRIPTION   : This routine starts the dialog box to support image     */
/*         acquisition into a user defined region_of_intrest.      */
/*                                        */
/* ARGUMENTS     : HWND   : hWnd                          */
/*         HANDLE : hInst                         */
/*                                        */
/* RETURN VALUE : none (void)                          */
/*                                        */
/*=========================================================================*/

void ROI_Acq_Image (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 WORD      Warr[5];
 FARPROC   lpfnDlgProc;

/*----------------------- Start of routine code     ----------------------*/
 NUM_FREQ = 0;
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)AcqRoiMP, hInst);
 DialogBox(hInst, (LPSTR)"SCM_ACQ_ROI", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);
 if (flag)
```

```
   {
    if(Get_DTconfig())
      {
    hOldCursor = SetCursor(LoadCursor(NULL,IDC_WAIT));
       roi_config();
       Get_Handles (hWnd, hInst);
       if(flag)
       {
     tmp_list.n = 1;
     tmp_list.hndl_list = tmp_hdl;

     if(op == FW_ACQBUF)
       Warr[0] = dt51_acquire (device, &src_roi, (BUF_HNDL_LIST _far *)-1L, &dest_ro
i, nframes);
     else
       Warr[0] = dt51_acquire (device, &src_roi, &tmp_list, &dest_roi, nframes);

     print_log ("dt51_acquire (device, src_roi, tmp_list, dest_roi, nframes) = 0x%x"
, (LPSTR) Warr);
     if(Warr[0] == 0)
       {
        src_roi_config ();
        dest_roi_config ();
       }
     else
       disp_err (Warr[0]);
       }
    SetCursor (hOldCursor);
      }
   }
}


/*=========================================================================*/
/*                                    */
/* NAME  : ROI_FOCUS_Image                    */
/* DESCRIPTION   : This routine is used to support "LIVE" image       */
/*         acquisition via multiple acqiusitions into a       */
/*         region_of_intrest.                    */
/*                                    */
/* ARGUMENTS     : HWND   : hWnd                    */
/*         HANDLE : hInst                    */
/*                                    */
/* RETURN VALUE : none (void)                    */
/*                                    */
/*=========================================================================*/

void ROI_FOCUS_Image (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ---------------------*/
 WORD      Warr[5];
 FARPROC   lpfnDlgProc;

/*----------------------- Start of routine code     ---------------------*/
 NUM_FREQ = 99;
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)AcqRoiMP, hInst);
```

```
  DialogBox(hInst, (LPSTR)"SCM_ACQ_ROI", hWnd, lpfnDlgProc);
  FreeProcInstance(lpfnDlgProc);
  if (flag)
    {
     if(Get_DTconfig())
       {
        roi_config();
        Get_Handles (hWnd, hInst);
        if(flag)
      {
       hOldCursor = SetCursor(LoadCursor(NULL,IDC_WAIT));
       tmp_list.n = 1;
       tmp_list.hndl_list = tmp_hdl;

       FF_FOCUS = TRUE;
       while(FF_FOCUS)
         {
          Warr[0] = dt51_acquire (device, &src_roi, &tmp_list, &dest_roi, nframes);
                 if(Warr[0] != 0)
                   {
                    FF_FOCUS = FALSE;
                     break;
                   }
          SpinTheMSGLoop ();
       SetCursor (hOldCursor);
      }

       if(!flag)
         PostQuitMessage (0);
       print_log ("dt51_acquire (device, src_roi, tmp_list, dest_roi, nframes) = 0x%x
", (LPSTR) Warr);
       if(Warr[0] == 0)
         {
          src_roi_config ();
          dest_roi_config ();
         }
       else
         disp_err (Warr[0]);
     }
      }
    }
}


/*=========================================================================*/
/*                                        */
/* NAME   : ROI_LIVE_Image                          */
/* DESCRIPTION    : This routine is used to support "LIVE" image        */
/*          acquisition via passthru mode into a            */
/*          region_of_intrest.                        */
/*                                        */
/* ARGUMENTS      : HWND    : hWnd                        */
/*          HANDLE : hInst                        */
/*                                        */
/* RETURN VALUE : none (void)                          */
/*                                        */
/*=========================================================================*/
```

```
void ROI_LIVE_Image (HWND hWnd, HANDLE hInst)
{
/*------------------- declaration of local variables ----------------------*/
 WORD       Warr[5];
 FARPROC    lpfnDlgProc;

/*------------------------ Start of routine code      ----------------------*/
 NUM_FREQ = 0;
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)AcqRoiMP, hInst);
 DialogBox(hInst, (LPSTR)"SCM_ACQ_ROI", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);
 if (flag)
   {
    if(Get_DTconfig())
      {
    hOldCursor = SetCursor(LoadCursor(NULL,IDC_WAIT));
       roi_config();
       Warr[0] = dt51_passthru (device, &src_roi, &dest_roi, nframes);
       print_log ("dt51_passthru (device, src_roi, dest_roi, nframes) = 0x%x", (LPST
R) Warr);
       if(Warr[0] == 0)
      {
       src_roi_config ();
       dest_roi_config ();
      }
       else
      disp_err (Warr[0]);
    SetCursor (hOldCursor);
      }
   }
}


/*=========================================================================*/
/*                                        */
/* NAME  : roi_config                              */
/* DESCRIPTION   : This routine copies the region_of_intrest information   */
/*         from the dialog into the numeric structure.         */
/*                                        */
/* ARGUMENTS     : none (void)                          */
/*                                        */
/* RETURN VALUE : none (void)                          */
/*                                        */
/*=========================================================================*/

void roi_config (void)
{
/*------------------------ Start of routine code      ----------------------*/
 src_roi.x = atoi(szItem2);
 src_roi.y = atoi(szItem3);
 src_roi.width = atoi(szItem4);
 src_roi.height = atoi(szItem5);
 dest_roi.x = atoi(szItem7);
 dest_roi.y = atoi(szItem8);
 dest_roi.width = atoi(szItem9);
```

```
 dest_roi.height = atoi(szItem10);
 nframes = atoi(szItem11);
}


/*=========================================================================*/
/*                                                                         */
/* NAME   : AcqRoiMP                                                       */
/* DESCRIPTION    : This routine supports the region_of_intrest dialog     */
/*          box selection/definition process.                              */
/*                                                                         */
/* ARGUMENTS      : HWND : hWndDlg                                         */
/*          WORD : Message                                                 */
/*          WORD : wParam                                                  */
/*          LONG : lParam                                                  */
/*                                                                         */
/* RETURN VALUE : TRUE or FALE (BOOL)                                      */
/*                                                                         */
/*=========================================================================*/

BOOL FAR PASCAL AcqRoiMP (HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------ declaration of local variables ----------------------*/
 static HANDLE hCtl;
 static HANDLE Sssc;
 static HANDLE Sssr;
 static HANDLE Ssbw;
 static HANDLE Ssbh;
 static HANDLE Sdsc;
 static HANDLE Sdsr;
 static HANDLE Sdbw;
 static HANDLE Sdbh;
 static HANDLE Sevryf;

/*----------------------- Start of routine code     ----------------------*/
 switch(Message)
   {
    case WM_INITDIALOG:
      Sevryf = GetDlgItem(hWndDlg, SCM_ENF_VS);
      Sssc = GetDlgItem(hWndDlg, SCM_SSC_VS);
      Sssr = GetDlgItem(hWndDlg, SCM_SSR_VS);
      Ssbw = GetDlgItem(hWndDlg, SCM_SBW_VS);
      Ssbh = GetDlgItem(hWndDlg, SCM_SBH_VS);
      Sdsc = GetDlgItem(hWndDlg, SCM_DSC_VS);
      Sdsr = GetDlgItem(hWndDlg, SCM_DSR_VS);
      Sdbw = GetDlgItem(hWndDlg, SCM_DBW_VS);
      Sdbh = GetDlgItem(hWndDlg, SCM_DBH_VS);
      SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
      if (NUM_FREQ == 99)
        {
```

```
      SetDlgItemInt(hWndDlg, SCM_ENF, 1, FALSE);
      hCtl = GetDlgItem (hWndDlg, SCM_ENF);
      EnableWindow (hCtl, FALSE);
      hCtl = GetDlgItem (hWndDlg, SCM_ENF_VS);
      EnableWindow (hCtl, FALSE);
     }
   else
          SetDlgItemInt(hWndDlg, SCM_ENF, 2, FALSE);
   cwCenter(hWndDlg, 0);
   break;

case WM_COMMAND:
   switch(wParam)
     {
      case IDOK:
       flag = TRUE;
       GetDlgItemText (hWndDlg, SCM_SSC, szItem2, 20);
       GetDlgItemText (hWndDlg, SCM_SSR, szItem3, 20);
       GetDlgItemText (hWndDlg, SCM_SBW, szItem4, 20);
       GetDlgItemText (hWndDlg, SCM_SBH, szItem5, 20);
       GetDlgItemText (hWndDlg, SCM_DSC, szItem7, 20);
       GetDlgItemText (hWndDlg, SCM_DSR, szItem8, 20);
       GetDlgItemText (hWndDlg, SCM_DBW, szItem9, 20);
       GetDlgItemText (hWndDlg, SCM_DBH, szItem10, 20);
       GetDlgItemText (hWndDlg, SCM_ENF, szItem11, 20);
       EndDialog(hWndDlg, TRUE);
       break;
      case IDCANCEL:
       flag = FALSE;
       EndDialog(hWndDlg, FALSE);
       break;
     }
   break;

case WM_VSCROLL:
   switch(wParam)
     {
          case SB_LINEDOWN:
       if(Sssc == HIWORD(lParam))
                  decrease_val(hWndDlg, SCM_SSC, 0, 1022);
                else
         if(Sssr == HIWORD(lParam))
                       decrease_val(hWndDlg, SCM_SSR, 0, 1022);
                     else
           if(Ssbw == HIWORD(lParam))
                       decrease_val(hWndDlg, SCM_SBW, 1, 1023);
                       else
         if(Ssbh == HIWORD(lParam))
                           decrease_val(hWndDlg, SCM_SBH, 1, 1023);
                       else
           if(Sdsc == HIWORD(lParam))
                        decrease_val(hWndDlg, SCM_DSC, 0, 1022);
                         else
             if(Sdsr == HIWORD(lParam))
                          decrease_val(hWndDlg, SCM_DSR, 0, 1022);
                            else
               if(Sdbw == HIWORD(lParam))
```

```
                                        decrease_val(hWndDlg, SCM_DBW, 0, 1023);
                                        else
                    if(Sdbh == HIWORD(lParam))
                                           decrease_val(hWndDlg, SCM_DBH, 0, 1023);
                                         else
                      if(Sevryf == HIWORD(lParam))
                                        decrease_val(hWndDlg, SCM_ENF, 0, 60);
                          break;
                    case SB_LINEUP:
               if(Sssc == HIWORD(lParam))
                           increase_val(hWndDlg, SCM_SSC, 0, 1022);
                         else
                 if(Sssr == HIWORD(lParam))
                                increase_val(hWndDlg, SCM_SSR, 0, 1022);
                              else
                   if(Ssbw == HIWORD(lParam))
                                increase_val(hWndDlg, SCM_SBW, 1, 1023);
                                else
                 if(Ssbh == HIWORD(lParam))
                                    increase_val(hWndDlg, SCM_SBH, 1, 1023);
                                else
                   if(Sdsc == HIWORD(lParam))
                                 increase_val(hWndDlg, SCM_DSC, 0, 1022);
                                  else
                     if(Sdsr == HIWORD(lParam))
                                   increase_val(hWndDlg, SCM_DSR, 0, 1022);
                                    else
                      if(Sdbw == HIWORD(lParam))
                                    increase_val(hWndDlg, SCM_DBW, 0, 1023);
                                    else
                     if(Sdbh == HIWORD(lParam))
                                      increase_val(hWndDlg, SCM_DBH, 0, 1023);
                                     else
                      if(Sevryf == HIWORD(lParam))
                                    increase_val(hWndDlg, SCM_ENF, 0, 60);
                          break;
          }
        break;

      default:
       return FALSE;
    }
return TRUE;
}


/*=========================================================================*/
/*                                      */
/* NAME   : src_roi_config                      */
/* DESCRIPTION    : This routine prints the source region_of_intrest data   */
/*          to the I/O log.                     */
/*                                      */
/* ARGUMENTS      : none (void)                         */
/*                                      */
/* RETURN VALUE : none (void)                         */
/*                                      */
/*=========================================================================*/
```

```
void src_roi_config (void)
{
/*---------------------- Start of routine code     ----------------------*/
 print_log (" * Source Starting Column      = 0x%x", (LPSTR) &src_roi.x);
 print_log (" * Source Starting Row         = 0x%x", (LPSTR) &src_roi.y);
 print_log (" * Source Width                = 0x%x", (LPSTR) &src_roi.width);
 print_log (" * Source Height               = 0x%x", (LPSTR) &src_roi.height);
}


/*=========================================================================*/
/*                                        */
/* NAME  : dest_roi_config                      */
/* DESCRIPTION   : This routine prints the destination region_of_intrest   */
/*         data to the I/O log.                    */
/*                                        */
/* ARGUMENTS     : none (void)                        */
/*                                        */
/* RETURN VALUE : none (void)                         */
/*                                        */
/*=========================================================================*/

void dest_roi_config (void)
{
/*---------------------- Start of routine code     ----------------------*/
 print_log (" * Destination Starting Column = 0x%x", (LPSTR) &dest_roi.x);
 print_log (" * Destination Starting Row    = 0x%x", (LPSTR) &dest_roi.y);
 print_log (" * Destination Width           = 0x%x", (LPSTR) &dest_roi.width);
 print_log (" * Destination Height          = 0x%x", (LPSTR) &dest_roi.height);
}


/*=========================================================================*/
/*                                        */
/* NAME  : Average_IT                           */
/* DESCRIPTION   : This routine performs the image acquire and average     */
/*         function. The style of averaging to perform is retuned  */
/*         from the dialog box and acted on from this routine.     */
/*                                        */
/* ARGUMENTS     : HWND    : hWnd                       */
/*         HANDLE  : hInst                      */
/*         int     : Amode                      */
/*                                        */
/* RETURN VALUE : none (void)                         */
/*                                        */
/*=========================================================================*/

void Average_IT (HWND hWnd, HANDLE hInst, int Amode)
{
#define MAX_BUFS 60

/*------------------ declaration of local variables ----------------------*/
 WORD       Warr[5];
 FARPROC    lpfnDlgProc;
 HMENU      hMenu;
 int        fskip;
```

```
 int         avgframes;
 int         ival;
 long        display_flag;
 long        weight;
 XY_rgn     avg_roi;
 XY_rgn     a_src_roi;
 XY_rgn     a_sys_roi;

/*----------------------- Start of routine code     ----------------------*/
 AvgMode = Amode;
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)AverageMP, hInst);
 DialogBox(hInst, (LPSTR)"SCM_AVERAGE", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);

 if (flag)
   {
    if (Get_DTconfig())
      {
       fskip = atoi(szItem1);
       avgframes = atoi(szItem2);
       if (avgframes > MAX_BUFS)
       {
       Warr[0] = MAX_BUFS;
       ErrMsg("number of frames to average too large (example code maximum = %d)", W
arr[0]);
       }
       else
       {
       a_src_roi.x = atoi(szItem3);
       a_src_roi.y = atoi(szItem4);
       a_src_roi.width = atoi(szItem5);
       a_src_roi.height = atoi(szItem6);
       a_sys_roi.x = atoi(szItem7);
       a_sys_roi.y = atoi(szItem8);
       a_sys_roi.width = atoi(szItem9);
       a_sys_roi.height = atoi(szItem10);
       display_flag = update_flag;
       if ( (Amode == FW_ACQ_RAVG) || (Amode == FW_ACQ_POST_RAVG) )
         weight = atoi(szItem11);

       /* set up average region of interest structure */
       avg_roi.x = (a_src_roi.x > a_sys_roi.x) ? a_src_roi.x : a_sys_roi.x;
       avg_roi.y = (a_src_roi.y > a_sys_roi.y) ? a_src_roi.y : a_sys_roi.y;
       avg_roi.height = (a_src_roi.height < a_sys_roi.height) ? a_src_roi.height : a
_sys_roi.height;
       avg_roi.width = (a_src_roi.width < a_sys_roi.width) ? a_src_roi.width : a_sys
_roi.width;

       /* if mode is acquire with post average , allocate temp */
           /* buffers from system memory                */
       ival=0;
       if ( (Amode == FW_ACQ_POST_TAVG) || (Amode == FW_ACQ_POST_RAVG) )
         {
          Warr[0]=0;
          while ((ival < avgframes) && (Warr[0] == 0))
        {
```

```
        Warr[0] = dt51_alloc_aux (device, avg_roi.height, avg_roi.width, 8L, &sys_h
ndls[ival+cfg.sys_bufs] );
        print_log ("dt51_alloc_aux = 0x%x", (LPSTR) Warr);
        if(Warr[0] != 0)
          {
           disp_err (Warr[0]);
           break;
          }
        else
          {
           hMenu = GetMenu (hWndMain);
           EnableMenuItem (hMenu, IDM_I_FREEAUXBUF, MF_ENABLED);
           EnableMenuItem (hMenu, IDM_I_DELETEALLBUFS, MF_ENABLED);
          }
        ival++;
       }
         sys_hndl_list.n = ival;
         sys_hndl_list.hndl_list = (BUF_HNDL *)(&sys_hndls[cfg.sys_bufs]);
       }
      else
        {
         sys_hndl_list.n = 0;
         sys_hndl_list.hndl_list = (BUF_HNDL *)NULL;
        }

      /* allocate 16 bit buffer for calculations */
      Warr[0] = dt51_alloc_aux (device, avg_roi.height, avg_roi.width, 16L, &sys_hn
dls[cfg.sys_bufs+ival]);
      print_log ("dt51_alloc_aux = 0x%x", (LPSTR) Warr);
      if(Warr[0] != 0)
        disp_err(Warr[0]);
      else
        {
         hMenu = GetMenu (hWndMain);
         EnableMenuItem (hMenu, IDM_I_FREEAUXBUF, MF_ENABLED);
         EnableMenuItem (hMenu, IDM_I_DELETEALLBUFS, MF_ENABLED);
             EnableMenuItem (hMenu, IDM_A_DELETEMANY, MF_ENABLED);
         EnableMenuItem (hMenu, IDM_A_REPLAYMANY, MF_ENABLED);
         EnableMenuItem (hMenu, IDM_A_LOOPREPLAY, MF_ENABLED);
         hOldCursor = SetCursor(LoadCursor(NULL,IDC_WAIT));
         /* READY... do the acquire average */
         if ((Amode == FW_ACQ_TAVG) || (Amode == FW_ACQ_POST_TAVG))
       {
        Warr[0] = dt51_acquire_average (device, Amode, &a_src_roi, &sys_hndl_list,
&avg_roi,
                  sys_hndls[cfg.sys_bufs+ival], fskip, avgframes, display_flag);
        print_log ("dt51_acquire_average = 0x%x", (LPSTR) Warr);
       }
         else
       {
        Warr[0] = dt51_recursive_average (device, Amode, &a_src_roi, &sys_hndl_list
, &avg_roi,
                  sys_hndls[cfg.sys_bufs+ival], fskip, avgframes, weight, display_f
lag);
        print_log ("dt51_recursive_average = 0x%x", (LPSTR) Warr);
       }
        if(Warr[0] != 0)
```

```
        disp_err (Warr[0]);
        else
        {
         if(Get_DTconfig())
           {
            Warr [0] = cfg.sys_bufs;
            Warr [1] = dt51_get_buffer_handles (device, FW_SYSBUF, cfg.sys_bufs, sys
_hndls);
            print_log ("dt51_get_buffer_handles (device, FW_SYSBUF, 0x%x, sys_hndls)
 = 0x%x", (LPSTR) Warr);
            if(Warr[1] != 0)
              disp_err (Warr[1]);
           }
            }
         }
       hOldCursor = SetCursor(LoadCursor(NULL,IDC_ARROW));
      }
      }
   }
}


/*=========================================================================*/
/*                                        */
/* NAME   : AverageMP                              */
/* DESCRIPTION    : This routine supports the averaging specification       */
/*          dialog box.                           */
/*                                        */
/* ARGUMENTS      : HWND : hWndDlg                          */
/*          WORD : Message                        */
/*          WORD : wParam                         */
/*          LONG : lParam                         */
/*                                        */
/* RETURN VALUE : TRUE OR FALSE (BOOL)                      */
/*                                        */
/*=========================================================================*/

BOOL FAR PASCAL AverageMP (HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------ declaration of local variables ----------------------*/
static HWND hWndDI;
static HWND hWndChkBox;

static HANDLE Sssc, Sssr, Ssbw, Ssbh;
static HANDLE Sdsc, Sdsr, Sdbw, Sdbh;
static HANDLE Senf, Snof, Sawt;

/*----------------------- Start of routine code     ----------------------*/
 switch(Message)
   {
    case WM_INITDIALOG:
      Sssc = GetDlgItem(hWndDlg, SCM_SSC_VS);
      Sssr = GetDlgItem(hWndDlg, SCM_SSR_VS);
      Ssbw = GetDlgItem(hWndDlg, SCM_SBW_VS);
      Ssbh = GetDlgItem(hWndDlg, SCM_SBH_VS);
      Sdsc = GetDlgItem(hWndDlg, SCM_DSC_VS);
      Sdsr = GetDlgItem(hWndDlg, SCM_DSR_VS);
```

```
    Sdbw = GetDlgItem(hWndDlg, SCM_DBW_VS);
    Sdbh = GetDlgItem(hWndDlg, SCM_DBH_VS);
    Senf = GetDlgItem(hWndDlg, SCM_ENF_VS);
    Snof = GetDlgItem(hWndDlg, SCM_NOF_VS);
    Sawt = GetDlgItem(hWndDlg, SCM_AWT_VS);

    SetDlgItemInt(hWndDlg, SCM_ENF, 2, FALSE);
        if (AvgMode == FW_ACQ_TAVG || AvgMode == FW_ACQ_RAVG)
     {
      hWndDI = GetDlgItem(hWndDlg, SCM_ENF);
      EnableWindow (hWndDI, FALSE);
      hWndDI = GetDlgItem(hWndDlg, SCM_ENF_VS);
      EnableWindow (hWndDI, FALSE);
      hWndDI = GetDlgItem(hWndDlg, -10);
      EnableWindow (hWndDI, FALSE);
      if (AvgMode == FW_ACQ_RAVG)
        strcpy(szString, "Acquire Recursive Average");
      else
      strcpy(szString, "Acquire True Average");
     }
    else
     {
      if (AvgMode == FW_ACQ_POST_TAVG)
        strcpy(szString, "Acquire Post True Average");
      else
        strcpy(szString, "Acquire Post Recursive Average");
     }
    SetWindowText (hWndDlg, (LPSTR)szString);

    SetDlgItemInt(hWndDlg, SCM_AWT, 1, FALSE);
           if (AvgMode == FW_ACQ_TAVG || AvgMode == FW_ACQ_POST_TAVG)
      {
       hWndDI = GetDlgItem(hWndDlg, SCM_AWT);
         EnableWindow (hWndDI, FALSE);
       hWndDI = GetDlgItem(hWndDlg, SCM_AWT_VS);
         EnableWindow (hWndDI, FALSE);
       hWndDI = GetDlgItem(hWndDlg, -12);
         EnableWindow (hWndDI, FALSE);
      }

    SetDlgItemInt(hWndDlg, SCM_NOF, 2, FALSE);
    SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
    SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
    SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
    SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
    SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
    SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
    SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
    SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);

    hWndChkBox = GetDlgItem(hWndDlg, SCM_UDD);
        SendMessage(hWndChkBox, BM_SETCHECK, 1, 0L);
    cwCenter(hWndDlg, 0);
    break;

  case WM_COMMAND:
    switch(wParam)
```

```
    {
     case IDOK:
      flag = TRUE;
      GetDlgItemText (hWndDlg, SCM_ENF, szItem1, 20);
      GetDlgItemText (hWndDlg, SCM_NOF, szItem2, 20);

      GetDlgItemText (hWndDlg, SCM_SSC, szItem3, 20);
      GetDlgItemText (hWndDlg, SCM_SSR, szItem4, 20);
      GetDlgItemText (hWndDlg, SCM_SBW, szItem5, 20);
      GetDlgItemText (hWndDlg, SCM_SBH, szItem6, 20);

      GetDlgItemText (hWndDlg, SCM_DSC, szItem7, 20);
      GetDlgItemText (hWndDlg, SCM_DSR, szItem8, 20);
      GetDlgItemText (hWndDlg, SCM_DBW, szItem9, 20);
      GetDlgItemText (hWndDlg, SCM_DBH, szItem10, 20);

      GetDlgItemText (hWndDlg, SCM_AWT, szItem11, 20);

      if (SendMessage(hWndChkBox, BM_GETCHECK, 0, 0L))
        update_flag = 1;
      else
        update_flag = 0;
      EndDialog(hWndDlg, TRUE);
      break;
     case IDCANCEL:
      flag = FALSE;
      EndDialog(hWndDlg, FALSE);
      break;
    }
  break;

case WM_VSCROLL:
  switch(wParam)
    {
     case SB_LINEDOWN:
      if(Sssc == HIWORD(lParam))
        decrease_val(hWndDlg, SCM_SSC, 0, 1022);
      else
        if(Sssr == HIWORD(lParam))
          decrease_val(hWndDlg, SCM_SSR, 0, 1022);
        else
          if(Ssbw == HIWORD(lParam))
        decrease_val(hWndDlg, SCM_SBW, 1, 1023);
          else
        if(Ssbh == HIWORD(lParam))
          decrease_val(hWndDlg, SCM_SBH, 1, 1023);
        else
          if(Sdsc == HIWORD(lParam))
            decrease_val(hWndDlg, SCM_DSC, 0, 1022);
          else
            if(Sdsr == HIWORD(lParam))
              decrease_val(hWndDlg, SCM_DSR, 0, 1022);
            else
              if(Sdbw == HIWORD(lParam))
                decrease_val(hWndDlg, SCM_DBW, 0, 1023);
              else
            if(Sdbh == HIWORD(lParam))
```

```
                decrease_val(hWndDlg, SCM_DBH, 0, 1023);
              else
                if(Senf == HIWORD(lParam))
                  decrease_val(hWndDlg, SCM_ENF, 0, 10);
                else
                  if(Snof == HIWORD(lParam))
                    decrease_val(hWndDlg, SCM_NOF, 1, 60);
                  else
                    if(Sawt == HIWORD(lParam))
                      decrease_val(hWndDlg, SCM_AWT, 1, 30);
          break;
         case SB_LINEUP:
          if(Sssc == HIWORD(lParam))
            increase_val(hWndDlg, SCM_SSC, 0, 1022);
          else
            if(Sssr == HIWORD(lParam))
              increase_val(hWndDlg, SCM_SSR, 0, 1022);
            else
              if(Ssbw == HIWORD(lParam))
                increase_val(hWndDlg, SCM_SBW, 1, 1023);
              else
            if(Ssbh == HIWORD(lParam))
              increase_val(hWndDlg, SCM_SBH, 1, 1023);
            else
              if(Sdsc == HIWORD(lParam))
                increase_val(hWndDlg, SCM_DSC, 0, 1022);
              else
                if(Sdsr == HIWORD(lParam))
                  increase_val(hWndDlg, SCM_DSR, 0, 1022);
                else
                  if(Sdbw == HIWORD(lParam))
                        increase_val(hWndDlg, SCM_DBW, 0, 1023);
                  else
                if(Sdbh == HIWORD(lParam))
                  increase_val(hWndDlg, SCM_DBH, 0, 1023);
                else
                  if(Senf == HIWORD(lParam))
                    increase_val(hWndDlg, SCM_ENF, 0, 10);
                  else
                    if(Snof == HIWORD(lParam))
                      increase_val(hWndDlg, SCM_NOF, 1, 60);
                    else
                      if(Sawt == HIWORD(lParam))
                      increase_val(hWndDlg, SCM_AWT, 1, 30);
          break;
        }
       break;
     default:
      return FALSE;
    }
return TRUE;
}

/*=====================================================================*/
/*                                                                     */
/* NAME  : ACQ_Sequential                                              */
```

```
/* DESCRIPTION   : This routine is used to implement the sequential    */
/*         acquire feature using multiple buffers to capture       */
/*         a sequence of images.                    */
/*                                    */
/* ARGUMENTS     : HWND   : hWnd                        */
/*         HANDLE : hInst                    */
/*                                    */
/* RETURN VALUE : none (void)                        */
/*                                    */
/*====================================================================*/

void ACQ_Sequential (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 WORD       Warr[3];
 FARPROC    lpfnDlgProc;
 HMENU      hMenu;
 HDC        hDC;
 RECT       zClient;
 u_long     rows, cols, psize;
 u_short    sequens, ISOK;
 int        i, seq_val, seq_frame, bad_seq;
 int        status;

/*----------------------- Start of routine code     ----------------------*/
 NUM_FREQ = 0;
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)NSequencesMP, hInst);
 DialogBox(hInst, (LPSTR)"SCM_SEQUENCE", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);
 if(flag)
   {
    seq_val = atoi (szItem1);
    seq_frame = atoi (szItem2);
    hOldCursor = SetCursor (LoadCursor(NULL, IDC_WAIT));
    if(Get_DTconfig())
        {
       sequens = cfg.sys_bufs;
       Warr[0] = dt51_get_set_format_memory (device, FW_READ, &fmt);
         if(Warr[0] != 0)
            disp_err(Warr[0]);
        else
            {
             format_center();
             rows = ((fmt.vfmt_digitize_e - fmt.vfmt_digitize_s) + 1)/fmt.vfmt_digit
ize_inc;
       cols = ((fmt.hfmt_digitize_e - fmt.hfmt_digitize_s) + 1)/fmt.hfmt_digitize_in
c;
       psize = 8;
       ISOK = TRUE;
       for (i=0; i < seq_val; i++)
        {
         Warr[0] = dt51_alloc_aux ( device, rows, cols, psize, &sys_hndls[sequens +
i]);
         if (Warr[0] != 0)
           {
        ISOK = FALSE;
```

```
        bad_seq = i;
        disp_err (Warr[0]);
        break;
           }
        }

       if (ISOK)
        {
         sprintf(szString, " ***>>> F9 to stop focus-mode and begin sequential acqui
re <*** ");
         GetClientRect (hWndMenu, &zClient);
         hDC = GetDC (hWndMenu);
         ExtTextOut (hDC, 0, 0, ETO_OPAQUE, &zClient, szString, lstrlen(szString), N
ULL);
         ReleaseDC (hWndMenu, hDC);

         Warr[0] = dt51_get_buffer_handles (device, FW_DISPBUF, cfg.disp_bufs, disp_
hndls);
                if(Warr[0] != 0)
                  disp_err(Warr[0]);
                else
                  {
        tmp_hdl = &disp_hndls[0];
        tmp_list.hndl_list = tmp_hdl;
               tmp_list.n = 1;

        FF_FOCUS = TRUE;
        while(FF_FOCUS)
         {
          Warr[0] = dt51_test_acq_done (device, &status);
                     if(Warr[0] != 0)
                       {
                        disp_err(Warr[0]);
                        FF_FOCUS = FALSE;
                        break;
                       }
          if (status == 1) /* if no acquisition is in progress */
             Warr[0] = dt51_acquire (device, &src_roi, &tmp_list, &dest_roi, 1);
                     if(Warr[0] != 0)
                       {
                        disp_err(Warr[0]);
                        FF_FOCUS = FALSE;
                        break;
                       }
          SpinTheMSGLoop ();
            }

          Warr[0] = dt51_freeze_frame ( device );
                     if(Warr[0] != 0)
                       disp_err(Warr[0]);

                   if(Warr[0] == 0)
                    {
          dest_roi.x = 0;
          dest_roi.y = 0;

          Warr[0] = dt51_get_buffer_handles ( device, FW_SYSBUF, cfg.sys_bufs, sys_h
```

```
ndls);
                    if(Warr[0] != 0)
                      disp_err(Warr[0]);
                    else
                      {
              tmp_hdl = &sys_hndls[sequens];
              tmp_list.hndl_list = tmp_hdl;
                 tmp_list.n = seq_val;

              sprintf(szString, " ***>>> Acquiring <*** ");
              GetClientRect (hWndMenu, &zClient);
              hDC = GetDC (hWndMenu);
              ExtTextOut (hDC, 0, 0, ETO_OPAQUE, &zClient, szString, lstrlen(szString
), NULL);
              ReleaseDC (hWndMenu, hDC);

              status = 0;
              Warr[0] = dt51_acquire (device, &src_roi, &tmp_list, &dest_roi, seq_fra
me);
              if (Warr[0] != 0)
                             disp_err(Warr[0]);
                              else
                             {
              while (status == 0)
                {
                 Warr[0] = dt51_test_acq_done (device, &status);
             if(Warr[0] != 0)
               {
                disp_err(Warr[0]);
                FF_FOCUS = FALSE;
                break;
               }
                }

              hMenu = GetMenu (hWndMain);
                  EnableMenuItem (hMenu, IDM_A_DELETEMANY, MF_ENABLED);
                   EnableMenuItem (hMenu, IDM_A_REPLAYMANY, MF_ENABLED);
                     EnableMenuItem (hMenu, IDM_A_LOOPREPLAY, MF_ENABLED);
                   EnableMenuItem (hMenu, IDM_I_FREEAUXBUF, MF_ENABLED);
                   EnableMenuItem (hMenu, IDM_I_DELETEALLBUFS, MF_ENABLED);
                sprintf(szString, " ***>>> Acquire Complete <*** ");
                GetClientRect (hWndMenu, &zClient);
                hDC = GetDC (hWndMenu);
                ExtTextOut (hDC, 0, 0, ETO_OPAQUE, &zClient, szString, lstrlen(szStri
ng), NULL);
                ReleaseDC (hWndMenu, hDC);
                            }
                          }
                        }
                      }
                    }
       else
        {
         for (i=sequens; i < bad_seq; i++)
          {
           Warr[0] = dt51_free_aux (device, sys_hndls[i]);
           if (Warr[0] != 0)
```

```
                    {
          disp_err (Warr[0]);
                     break;
                    }
           }
         }

        if(Get_DTconfig())
          {
           if (cfg.sys_bufs == 0)
         {
          hMenu = GetMenu (hWndMain);
          EnableMenuItem (hMenu, IDM_I_FREEAUXBUF, MF_GRAYED);
          EnableMenuItem (hMenu, IDM_I_DELETEALLBUFS, MF_GRAYED);

          EnableMenuItem (hMenu, IDM_A_DELETEMANY, MF_GRAYED);
          EnableMenuItem (hMenu, IDM_A_REPLAYMANY, MF_GRAYED);
          EnableMenuItem (hMenu, IDM_A_LOOPREPLAY, MF_GRAYED);
         }
          }
             }
         }
     SetCursor (hOldCursor);
    }
}


/*=========================================================================*/
/*                                        */
/* NAME  : NSequencesMP                          */
/* DESCRIPTION    : This routine implements the dialog box that queries     */
/*          the user for the number of sequential frames and the    */
/*          incremental spacing of the frames that will be      */
/*          during the sequential acquire process.          */
/*                                        */
/* ARGUMENTS      : HWND : hWndDlg                        */
/*          WORD : Message                        */
/*          WORD : wParam                         */
/*          LONG : lParam                         */
/*                                        */
/* RETURN VALUE : TRUE or FALSE (BOOL)                    */
/*                                        */
/*=========================================================================*/

BOOL FAR PASCAL NSequencesMP (HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------- declaration of local variables ----------------------*/
static HANDLE Snv, Snf;

/*------------------------ Start of routine code     ----------------------*/
 switch(Message)
   {
    case WM_INITDIALOG:
      Snv = GetDlgItem(hWndDlg, SCM_NUMBER_VS);
      Snf = GetDlgItem(hWndDlg, SCM_NTHFRAME_VS);
      SetDlgItemInt(hWndDlg, SCM_NUMBER, 15, FALSE);
      SetDlgItemInt(hWndDlg, SCM_NTHFRAME, 2, FALSE);
```

```
        cwCenter(hWndDlg, 0);
       break;

      case WM_COMMAND:
        switch(wParam)
          {
           case IDOK:
            flag = TRUE;
            GetDlgItemText (hWndDlg, SCM_NUMBER, szItem1, 20);
            GetDlgItemText (hWndDlg, SCM_NTHFRAME, szItem2, 20);
            EndDialog(hWndDlg, TRUE);
            break;
           case IDCANCEL:
            flag = FALSE;
            EndDialog(hWndDlg, FALSE);
            break;
          }
        break;

      case WM_VSCROLL:
        switch(wParam)
          {
                  case SB_LINEDOWN:
                        if(Snv == HIWORD(lParam))
                           decrease_val(hWndDlg, SCM_NUMBER, 0, 255);
                        else
                            if(Snf == HIWORD(lParam))
                              decrease_val(hWndDlg, SCM_NTHFRAME, 0, 60);
                        break;
                  case SB_LINEUP:
                        if(Snv == HIWORD(lParam))
                           increase_val(hWndDlg, SCM_NUMBER, 0, 255);
                        else
                            if(Snf == HIWORD(lParam))
                              increase_val(hWndDlg, SCM_NTHFRAME, 0, 60);
                        break;
          }
        break;

      default:
       return FALSE;
     }
return TRUE;
}


/*=========================================================================*/
/*                                                                         */
/* NAME  : RUN_Sequential                                                  */
/* DESCRIPTION   : This routine queries the user for instructions on how   */
/*         to replay the series of acquired images.                        */
/*                                                                         */
/* ARGUMENTS     : HWND   : hWnd                                           */
/*         HANDLE : hInst                                                  */
/*                                                                         */
/* RETURN VALUE : none (void)                                              */
/*                                                                         */
```

```
/*=========================================================================*/

void RUN_Sequential (HWND hWnd, HANDLE hInst)
{
/*------------------- declaration of local variables ----------------------*/
 WORD        Warr[3];
 FARPROC        lpfnDlgProc;
 HDC         hDC;
 RECT        zClient;
 u_short        sequens;
 int         i, seq_val;
 static u_long RepMany;
 u_long j, j2;

/*------------------------ Start of routine code     ----------------------*/
 NUM_FREQ = 2;
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)SeqMotionMP, hInst);
 DialogBox(hInst, (LPSTR)"SCM_MOTION", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);
 if(flag)
   {
    hOldCursor = SetCursor (LoadCursor(NULL, IDC_WAIT));
    seq_val = atoi (szItem1);
    sequens = atoi (szItem2);
    RepMany = atol (szItem3);
    if(Get_DTconfig())
        {
       Warr[0] = dt51_get_set_format_memory (device, FW_READ, &fmt);
         if(Warr[0] != 0)
            disp_err(Warr[0]);
         else
            {
       format_center();
       Warr[0] = dt51_get_buffer_handles ( device, FW_SYSBUF, cfg.sys_bufs, sys_hndl
s);
         if(Warr[0] != 0)
            disp_err(Warr[0]);
            else
               {
          Warr[0] = dt51_get_buffer_handles ( device, FW_DISPBUF, cfg.disp_bufs, dis
p_hndls);
               if(Warr[0] != 0)
                   disp_err(Warr[0]);
              else
                  {
         dest_hndl = disp_hndls[0];
         sprintf(szString, " ***>>> Replaying Series <*** ");
         GetClientRect (hWndMenu, &zClient);
         hDC = GetDC (hWndMenu);
         ExtTextOut (hDC, 0, 0, ETO_OPAQUE, &zClient, szString, lstrlen(szString), N
ULL);
         ReleaseDC (hWndMenu, hDC);

         for (i=sequens; i < seq_val; i++)
          {
           src_hndl = sys_hndls[i];
```

```
            Warr[0] = dt51_copy_buf (device, src_hndl, &src_roi, dest_hndl, &dest_roi
);
                        if(Warr[0] != 0)
                             {
                              disp_err(Warr[0]);
                            break;
                              }
                        else
             for (j=0L; j < 65535L; j++)
              for (j2=0L; j2 < RepMany; j2++);
           }
                     if(Warr[0] == 0)
                        {
             sprintf(szString, " ***>>> Series Complete <*** ");
             GetClientRect (hWndMenu, &zClient);
             hDC = GetDC (hWndMenu);
             ExtTextOut (hDC, 0, 0, ETO_OPAQUE, &zClient, szString, lstrlen(szString)
, NULL);
             ReleaseDC (hWndMenu, hDC);
                        }
                    }
                }
             }
         }
     SetCursor (hOldCursor);
    }
}


/*=========================================================================*/
/*                                        */
/* NAME  : LOOP_Sequential                      */
/* DESCRIPTION   : This routine is used to allow the user to repeat or     */
/*         loop playing a series or sequence of previously     */
/*         acquired string of images.                  */
/*                                        */
/* ARGUMENTS     : HWND   : hWnd                       */
/*         HANDLE : hInst                       */
/*                                        */
/* RETURN VALUE : none (void)                        */
/*                                        */
/*=========================================================================*/

void LOOP_Sequential (HWND hWnd, HANDLE hInst)
{
/*------------------- declaration of local variables ----------------------*/
 WORD        Warr[3];
 FARPROC        lpfnDlgProc;
 HDC         hDC;
 RECT        zClient;
 u_short        sequens;
 int         i, seq_val;
 static u_long RepMany;
 u_long j, j2;

/*----------------------- Start of routine code     ----------------------*/
 NUM_FREQ = 2;
```

```
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)SeqMotionMP, hInst);
 DialogBox(hInst, (LPSTR)"SCM_MOTION", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);
 if(flag)
   {
    hOldCursor = SetCursor (LoadCursor(NULL, IDC_WAIT));
    seq_val = atoi (szItem1);
    sequens = atoi (szItem2);
    RepMany = atol (szItem3);
    if(Get_DTconfig())
        {
       Warr[0] = dt51_get_set_format_memory (device, FW_READ, &fmt);
         if(Warr[0] != 0)
            disp_err(Warr[0]);
         else
            {
       format_center();
       Warr[0] = dt51_get_buffer_handles ( device, FW_SYSBUF, cfg.sys_bufs, sys_hndl
s);
            if(Warr[0] != 0)
                disp_err(Warr[0]);
            else
               {
          Warr[0] = dt51_get_buffer_handles ( device, FW_DISPBUF, cfg.disp_bufs, dis
p_hndls);
               if(Warr[0] != 0)
                disp_err(Warr[0]);
              else
                  {
         dest_hndl = disp_hndls[0];
         SetCursor (hOldCursor);
         sprintf(szString, " Looping on Sequential Acquire Playback ***> F9 to Stop
<*** ");
         GetClientRect (hWndMenu, &zClient);
         hDC = GetDC (hWndMenu);
             ExtTextOut (hDC, 0, 0, ETO_OPAQUE, &zClient, szString, lstrlen(szString
), NULL);
         ReleaseDC (hWndMenu, hDC);

                     FF_FOCUS = TRUE;
         while(FF_FOCUS)
           {
            for (i=sequens; i < seq_val; i++)
               {
            src_hndl = sys_hndls[i];
            Warr[0] = dt51_copy_buf (device, src_hndl, &src_roi, dest_hndl, &dest_ro
i);
                     if(Warr[0] != 0)
                               {
                      disp_err(Warr[0]);
                                FF_FOCUS = FALSE;
                                break;
                               }
                    else
              for (j=0L; j < 65535L; j++)
               for (j2=0L; j2 < RepMany; j2++);
```

```
                }
               SpinTheMSGLoop ();
            }
                     if(Warr[0] == 0)
                        {
             sprintf(szString, " Looping on Sequential Acquire Terminated ");
             GetClientRect (hWndMenu, &zClient);
             hDC = GetDC (hWndMenu);
             ExtTextOut (hDC, 0, 0, ETO_OPAQUE, &zClient, szString, lstrlen(szString)
, NULL);
             ReleaseDC (hWndMenu, hDC);
              }
                    }
                 }
             }
          }
     }
}


/*============================================================================*/
/*                                        */
/* NAME  : SeqMotionMP                          */
/* DESCRIPTION   : This routine supports the dialog box that the user      */
/*         uses to specify the working parameters for sequential   */
/*         frame acquisition.                        */
/*                                        */
/* ARGUMENTS     : HWND : hWndDlg                        */
/*         WORD : Message                       */
/*         WORD : wParam                        */
/*         LONG : lParam                        */
/*                                        */
/* RETURN VALUE : TRUE or FALSE (BOOL)                     */
/*                                        */
/*============================================================================*/

BOOL FAR PASCAL SeqMotionMP (HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------ declaration of local variables ----------------------*/
static HANDLE Secf, Svcf, Shcf;
static HANDLE hCtl;

/*----------------------- Start of routine code     ----------------------*/
 switch(Message)
   {
    case WM_INITDIALOG:
      Secf = GetDlgItem(hWndDlg, SCM_NO_BUF_VS);
      Svcf = GetDlgItem(hWndDlg, SCM_STRT_BUF_VS);
      Shcf = GetDlgItem(hWndDlg, SCM_DEL_FCT_VS);
      SetDlgItemInt(hWndDlg, SCM_NO_BUF, 15, FALSE);
      SetDlgItemInt(hWndDlg, SCM_STRT_BUF, 0, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DEL_FCT, 0, FALSE);
      if (NUM_FREQ == 1)
        {
         hCtl = GetDlgItem (hWndDlg, SCM_DEL_FCT);
         EnableWindow (hCtl, FALSE);
         hCtl = GetDlgItem (hWndDlg, SCM_DEL_FCT_VS);
```

```
          EnableWindow (hCtl, FALSE);
         }
        cwCenter(hWndDlg, 0);
       break;

      case WM_COMMAND:
        switch(wParam)
          {
           case IDOK:
            flag = TRUE;
            GetDlgItemText (hWndDlg, SCM_NO_BUF, szItem1, 20);
            GetDlgItemText (hWndDlg, SCM_STRT_BUF, szItem2, 20);
            GetDlgItemText (hWndDlg, SCM_DEL_FCT, szItem3, 20);
            EndDialog(hWndDlg, TRUE);
            break;
           case IDCANCEL:
            flag = FALSE;
            EndDialog(hWndDlg, FALSE);
            break;
          }
        break;

      case WM_VSCROLL:
        switch(wParam)
          {
                  case SB_LINEDOWN:
                        if( Secf == HIWORD(lParam))
                           decrease_val(hWndDlg, SCM_NO_BUF, 0, 255);
                        else
                            if( Svcf == HIWORD(lParam))
                                decrease_val(hWndDlg, SCM_STRT_BUF, 0, 255);
                            else
                              if( Shcf == HIWORD(lParam))
                                  decrease_long_val(hWndDlg, SCM_DEL_FCT, 0L, 65510L);
                        break;
                  case SB_LINEUP:
                        if( Secf == HIWORD(lParam))
                           increase_val(hWndDlg, SCM_NO_BUF, 0, 255);
                        else
                            if( Svcf == HIWORD(lParam))
                                increase_val(hWndDlg, SCM_STRT_BUF, 0, 255);
                            else
                              if( Shcf == HIWORD(lParam))
                                  increase_long_val(hWndDlg, SCM_DEL_FCT, 0L, 65510L);
                        break;
          }
        break;
       default:
        return FALSE;
     }
  return TRUE;
  }


  /*=========================================================================*/
  /*                                    */
  /* NAME   : Four_ACQ_Images                    */
```

```
/* DESCRIPTION   : This routine supports the ability to acquire and    */
/*         display 4 different images from the 4 input channels     */
/*         in different quadrants on the display monitor.       */
/*         All cameras must be of the same type.            */
/*                                      */
/* ARGUMENTS     : HWND   : hWnd                        */
/*         HANDLE : hInst                       */
/*                                      */
/* RETURN VALUE : none (void)                         */
/*                                      */
/*======================================================================*/

void Four_ACQ_Images (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 WORD           Warr[5];
 int            i;
 FMT            tmp_fmt;
 ACQ_SETUP      tmp_acq_setup;
 IMG_ACQ_SETUP      tmp_img_acq_setup;

/*----------------------- Start of routine code     ----------------------*/
 print_log ("", (LPSTR) NULL);
 if(Get_DTconfig())
   {
      Set_Bufs (hWnd, hInst);
      if(flag)
     {
      Warr[0] = dt51_get_set_format_memory (device, FW_READ, &tmp_fmt);
            if(Warr[0] != 0)
              disp_err(Warr[0]);
            else
             {
        if (cfg.dev_id < ID_DT3851A_1)
                {
        Warr[0] = dt51_acquire_setup (device, FW_READ, &tmp_acq_setup);
                  if(Warr[0] != 0)
                    disp_err(Warr[0]);
                  else
                    {
           acq_setup.start_field = FW_ACQ_ST_FLD_NI;
           acq_setup.acq_mode = FW_ACQ_MODE_INT;
           acq_setup.invert_clk = FW_DISABLE;
           acq_setup.invert_clk = FW_DISABLE;
           Warr[0] = dt51_acquire_setup (device, FW_WRITE, &acq_setup);
                   if(Warr[0] != 0)
                     disp_err(Warr[0]);
                   }
                }
        else
                {
        Warr[0] = dt51_image_acquire_setup (device, FW_READ, &tmp_img_acq_setup);
                  if(Warr[0] != 0)
                    disp_err(Warr[0]);
                  else
                   {
          img_acq_setup.start_field = FW_ACQ_ST_FLD_NI;
```

```
          img_acq_setup.img_acq_mode     = FW_IMG_ACQ_MODE_RS170;
          img_acq_setup.invert_ext_clk = FW_DISABLE;
          img_acq_setup.invert_line_sync = FW_DISABLE;
          img_acq_setup.invert_frame_sync = FW_DISABLE;
          img_acq_setup.clock_mode = FW_IMG_CLK_MODE_INT;
          Warr[0] = dt51_image_acquire_setup (device, FW_WRITE, &img_acq_setup);
                    if(Warr[0] != 0)
                      disp_err(Warr[0]);
                }
                }
              if(Warr[0] == 0)
               {
          fmt.hfmt_digitize_s = 125;
          fmt.hfmt_digitize_e = 764;
          fmt.hfmt_digitize_inc = 2;
          fmt.hfmt_ad_zero_s = 75;
          fmt.hfmt_ad_zero_e = 78;
          fmt.hfmt_ad_zero_inc = 1;
          fmt.hfmt_ad_clamp_s = 75;
          fmt.hfmt_ad_clamp_e = 78;
          fmt.hfmt_ad_clamp_inc = 1;
          fmt.vfmt_digitize_s = 10;
          fmt.vfmt_digitize_e = 249;
          fmt.vfmt_digitize_inc = 1;
          Warr[0] = dt51_get_set_format_memory (device, FW_WRITE, &fmt);
                if(Warr[0] != 0)
                 disp_err(Warr[0]);
                else
                 {
        src_roi.x = 0;
        src_roi.y = 0;
        src_roi.width = ((fmt.hfmt_digitize_e - fmt.hfmt_digitize_s) + 1)/fmt.hfmt_d
igitize_inc;
        src_roi.height = ((fmt.vfmt_digitize_e - fmt.vfmt_digitize_s) + 1)/fmt.vfmt_
digitize_inc;

        tmp_list.n = 1;
        tmp_list.hndl_list = tmp_hdl;

                    dest_roi.width = (unsigned short)tmp_struct->width;
        dest_roi.height = (unsigned short)tmp_struct->height;

        FF_FOCUS = TRUE;
        while(FF_FOCUS)
        {
         vsync = FW_SYNC_0;
         Warr[0] = dt51_get_set_sync_source ( device, FW_WRITE, &vsync);
                   if(Warr[0] != 0)
                      {
                   disp_err(Warr[0]);
                         FF_FOCUS = FALSE;
                         break;
                      }
                   else
                    {
           vchan = FW_CHAN_0;
           Warr[0] = dt51_select_input_channel (device, &vchan);
```

```
          if(Warr[0] != 0)
              {
           disp_err(Warr[0]);
                 FF_FOCUS = FALSE;
                 break;
              }
          else
           {
 dest_roi.x = 0;
 dest_roi.y = 0;
 for (i=0; i<425; i++)
  SpinTheMSGLoop ();
 Warr[0] = dt51_acquire (device, &src_roi, &tmp_list, &dest_roi, 2);
           if(Warr[0] != 0)
                 {
            disp_err(Warr[0]);
                  FF_FOCUS = FALSE;
                  break;
                 }
           else
        {
  vsync = FW_SYNC_1;
  Warr[0] = dt51_get_set_sync_source ( device, FW_WRITE, &vsync);
         if(Warr[0] != 0)
                 {
              disp_err(Warr[0]);
                  FF_FOCUS = FALSE;
                  break;
                 }
        else
         {
vchan = FW_CHAN_1;
Warr[0] = dt51_select_input_channel (device, &vchan);
          if(Warr[0] != 0)
                  {
               disp_err(Warr[0]);
                   FF_FOCUS = FALSE;
                   break;
                  }
         else
          {
 dest_roi.x = 320;
 dest_roi.y = 0;
 for (i=0; i<425; i++)
  SpinTheMSGLoop ();
 Warr[0] = dt51_acquire (device, &src_roi, &tmp_list, &dest_roi, 2);
          if(Warr[0] != 0)
                  {
              disp_err(Warr[0]);
                   FF_FOCUS = FALSE;
                   break;
                  }
          else
           {
  vsync = FW_SYNC_2;
  Warr[0] = dt51_get_set_sync_source ( device, FW_WRITE, &vsync);
           if(Warr[0] != 0)
```

```
                    {
                   disp_err(Warr[0]);
                     FF_FOCUS = FALSE;
                     break;
                    }
               else
             {
      vchan = FW_CHAN_2;
      Warr[0] = dt51_select_input_channel (device, &vchan);
            if(Warr[0] != 0)
                    {
                   disp_err(Warr[0]);
                     FF_FOCUS = FALSE;
                     break;
                    }
            else
             {
    dest_roi.x = 0;
    dest_roi.y = 240;
    for (i=0; i<425; i++)
     SpinTheMSGLoop ();
    Warr[0] = dt51_acquire (device, &src_roi, &tmp_list, &dest_roi, 2);
              if(Warr[0] != 0)
                       {
                      disp_err(Warr[0]);
                        FF_FOCUS = FALSE;
                        break;
                       }
              else
               {
      vsync = FW_SYNC_3;
      Warr[0] = dt51_get_set_sync_source ( device, FW_WRITE, &vsync);
                if(Warr[0] != 0)
                        {
                    disp_err(Warr[0]);
                          FF_FOCUS = FALSE;
                          break;
                         }
                else
                 {
        vchan = FW_CHAN_3;
        Warr[0] = dt51_select_input_channel (device, &vchan);
                  if(Warr[0] != 0)
                       {
                      disp_err(Warr[0]);
                        FF_FOCUS = FALSE;
                        break;
                       }
                  else
                 {
          dest_roi.x = 320;
          dest_roi.y = 240;
          for (i=0; i<425; i++)
       SpinTheMSGLoop ();
          Warr[0] = dt51_acquire (device, &src_roi, &tmp_list, &dest_ro
i, 2);
                  if(Warr[0] != 0)
```

```
                                        {
                                       disp_err(Warr[0]);
                                         FF_FOCUS = FALSE;
                                         break;
                                        }
                                                }
                                              }
                                            }
                                          }
                                        }
                                        }
                                    }
                                  }
                                }
                              }

                  if(!flag)
               PostQuitMessage (0);

                  dt51_get_set_format_memory (device, FW_WRITE, &tmp_fmt);
                  if (cfg.dev_id < ID_DT3851A_1)
               dt51_acquire_setup (device, FW_WRITE, &tmp_acq_setup);
                  else
               dt51_image_acquire_setup (device, FW_WRITE, &tmp_img_acq_setup);
                  vchan = FW_CHAN_0;
                  dt51_select_input_channel (device, &vchan);
                  op = FW_WRITE;
                  vsync = FW_SYNC_0;
                  dt51_get_set_sync_source (device, op, &vsync);
                        }
                     }
                   }
                 }
   }
}

/*--------------------------  END OF ACQUIRE.C --------------------------*/
```

```
/*************************************************************************/
/*************************************************************************/
/***                          ***************************/
/***  ARMM Windows Program    ***************************/
/***                          ***************************/
/***  MODULE : AUXMEM2.C             ***************************/
/***                          ***************************/
/***  Last Update: 9/3096            ***************************/
/***    by Richard Greer                   ***************************/
/*************************************************************************/
/***  The IScale_Image function was modified for image switching purposes */
/***  to allow an image captured to the aux buffer to be moved to the     */
/***  proper part of the display depending on which channel was currently */
/***  in use.  For this, the code for the dialog box was also bypassed.   */
/***                                                                      */
/***  The AuxBufs_allox funciton was modified to bypass the dialog box and*/
/***  automatically allocate one auxiliary 640X480 buffer.  This was done */
/***  to facilitate the buffer being automatically allocated on program   */
/***  startup.                                                            */
/***                                                                      */
/***  All other functions have not been changed from the original code    */
/***  provided by Data Translation.                                       */
/*************************************************************************/
/***                                    ***/
/***  AuxMem Menu Option Functions                ***/
/***                                    ***/
/***                                    ***/
/***      A) Memory Copy:                       ***/
/***           1. Move_Image                    ***/
/***           2. RoiACMP                       ***/
/***           3. get_H_and_BI                  ***/
/***                                    ***/
/***      B) Memory Copy with pixel operations:          ***/
/***           1. Move_Image_PixelOp                ***/
/***           2. RoiACPOPMP                    ***/
/***                                    ***/
/***      C) DLL Scale Image:                   ***/
/***           1. IScale_Image                  ***/
/***           2. RoiIS                         ***/
/***                                    ***/
/***      D) DLL Rotate Image 90:               ***/
/***           1. Img_Rotate_90                 ***/
/***           2. Rotate_90                     ***/
/***                                    ***/
/***      E) Allocate AUX Buf:                  ***/
/***           1. AuxBufs_allox                 ***/
/***           2. AbaMP                         ***/
/***                                    ***/
/***      F) Free AUX Buf:                      ***/
/***           1. Free_allox                    ***/
/***                                    ***/
/***      G) Free Sequence of AUX Bufs:              ***/
/***           1. FREE_Sequential               ***/
/***                                    ***/
/***      H) Free ALL AUX Buffers:              ***/
/***           1. ALLFree_allox                 ***/
```

```
/***                                                    ***/
/************************************************************************/
/************************************************************************/

/************************************************************************/
/***  Include Files & Function Definitions  *****************************/
/************************************************************************/

#include "dt51.h"


/************************************************************************/
/***  EXTERNAL DECLARATIONS for ALL Global Variables *******************/
/************************************************************************/

/* 32-bit identifiers used to specify an auxiliary, display, overlay, or  */
/* an acquire memory buffer                                 */
extern BUF_HNDL acq_hndls[FW_NUM_ACQ_BUFS];
extern BUF_HNDL disp_hndls[FW_NUM_DISP_BUFS];
extern BUF_HNDL ovl_hndls[FW_NUM_OVL_BUFS];
extern BUF_HNDL sys_hndls[FW_NUM_SYS_BUFS];
extern BUF_HNDL    src_hndl;
extern BUF_HNDL    dest_hndl;

/* structures composed of 2 fields (the number of BUF_HNDLS and a pointer */
/* structures composed of 4 fields in to which the routines return or set */
/* information on the size and starting location of the region of      */
/* interest in the specified memory buffer                  */
extern XY_rgn src_roi;
extern XY_rgn dest_roi;

/* structure composed of 19 fields to which the routine returns    */
/* information on the number of buffers, the size of each memory type on  */
/* the board, and other operational features                   */
extern FW_CONFIG cfg;

/* specifies the device enabling key for the routine in question */
extern u_long device;

/* generic string variables used to store temporary results */
extern char szString[80];
extern char szItem[20];
extern char szItem1[20];
extern char szItem2[20];
extern char szItem3[20];
extern char szItem4[20];
extern char szItem5[20];
extern char szItem6[20];
extern char szItem7[20];
extern char szItem8[20];
extern char szItem9[20];
extern char szItem10[20];
extern char szItem11[20];
extern char szItem12[20];

/* temporary variable to hold cursor */
extern HCURSOR hOldCursor;
```

```
/* temporary variable to define srource and definition buffers temporarily */
extern u_short src_op, dest_op;

/* used to determine if dialog box parameters are okay to use */
extern u_short flag;

/* DT3851/52 menu status bar */
extern HWND hWndMenu;

/* generic variables used to temporarily convey a buffer or operation to  */
/* be performed                                    */
extern u_short op, op1, op2;

/* flag used by generic dialog box to set it up for proper routine */
extern int    direction;
extern u_short    NUM_FREQ;
extern u_short    FF_FOCUS;
extern u_short    src_buf_num;
extern u_short    dest_buf_num;
extern u_short    value;
extern HWND   hWndMain;

/* Channel selection flag added by R. Greer 9/9/96*/
extern u_short current_chan;

/* globals used in this file */
DWORD selected_pixelop;

/*=#########################################################################=*/
/*                                        */
/*               START OF ROUTINE CODE            */
/*                                        */
/*=#########################################################################=*/


/*=========================================================================*/
/*                                        */
/* NAME  :  Move_Image                          */
/* DESCRIPTION   :  This routine is used to move an image buffer from      */
/*          one onboard buffer to another onboard buffer.      */
/*                                        */
/* ARGUMENTS     :  HWND   : hWnd                       */
/*          HANDLE : hInst                      */
/*                                        */
/* RETURN VALUE : none (void)                         */
/*                                        */
/*=========================================================================*/

void Move_Image (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
  WORD Warr[5];
  FARPROC lpfnDlgProc;

/*----------------------- Start of routine code     ----------------------*/
 src_buf_num =
```

```
  dest_buf_num = 0;
  print_log ("", (LPSTR) NULL);
  lpfnDlgProc = MakeProcInstance((FARPROC)RoiACMP, hInst);
  DialogBox(hInst, (LPSTR)"SCM_ROI_AC", hWnd, lpfnDlgProc);
  FreeProcInstance(lpfnDlgProc);
  if (flag)
    {
      if (Get_DTconfig())
        {
       src_buf_num = atoi(szItem1);
       dest_buf_num = atoi(szItem6);
       src_roi.x = atoi(szItem2);
       src_roi.y = atoi(szItem3);
       src_roi.width = atoi(szItem4);
       src_roi.height = atoi(szItem5);
       dest_roi.x = atoi(szItem7);
       dest_roi.y = atoi(szItem8);
       dest_roi.width = atoi(szItem9);
       dest_roi.height = atoi(szItem10);

       get_H_and_BI();

       Warr [0] = dt51_copy_buf (device, src_hndl, &src_roi, dest_hndl, &dest_roi);
       print_log ("dt51_copy_buf (device, src_hndl, src_roi, dest_hndl, dest_roi) = 0
x%x", (LPSTR) Warr);
       if(Warr[0] == 0)
         {
           print_log (" * Source Buffer No.          = 0x%x", (LPSTR) &src_buf_num);
           src_roi_config ();
           print_log (" * Destination Buffer No.     = 0x%x", (LPSTR) &dest_buf_num)
;
           dest_roi_config ();
         }
       else
         disp_err (Warr[0]);
       }
     }
}

/*=========================================================================*/
/*                                      */
/* NAME  : RoiACMP                          */
/* DESCRIPTION   : This routine controls the dialog box that is used to    */
/*         select the region of interest in the source and     */
/*         destination buffers.                    */
/*                                      */
/* ARGUMENTS     : HWND : hWndDlg                      */
/*         WORD : Message                   */
/*         WORD : wParam                    */
/*         LONG : lParam                    */
/*                                      */
/* RETURN VALUE : none (void)                        */
/*                                      */
/*=========================================================================*/

BOOL FAR PASCAL RoiACMP (HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
```

```
/*------------------- declaration of local variables ----------------------*/
 static HANDLE Ssbn, Sssc, Sssr, Ssbw, Ssbh;
 static HANDLE Sdbn, Sdsc, Sdsr, Sdbw, Sdbh;
 static HANDLE temp  = NULL;
 static HNDL_STRUCT shndl_strct;
 static HNDL_STRUCT dhndl_strct;
 static FW_CONFIG    strct_cfg;
 static unsigned short smax_buf, dmax_buf;

 static unsigned short NVal;

/*------------------------ Start of routine code     ----------------------*/
 switch(Message)
   {
    case WM_INITDIALOG:
           dt51_get_config(device,&strct_cfg);

        Ssbn = GetDlgItem(hWndDlg, SCM_SBN_VS);
        Sssc = GetDlgItem(hWndDlg, SCM_SSC_VS);
        Sssr = GetDlgItem(hWndDlg, SCM_SSR_VS);
        Ssbw = GetDlgItem(hWndDlg, SCM_SBW_VS);
        Ssbh = GetDlgItem(hWndDlg, SCM_SBH_VS);
        Sdbn = GetDlgItem(hWndDlg, SCM_DBN_VS);
        Sdsc = GetDlgItem(hWndDlg, SCM_DSC_VS);
        Sdsr = GetDlgItem(hWndDlg, SCM_DSR_VS);
        Sdbw = GetDlgItem(hWndDlg, SCM_DBW_VS);
        Sdbh = GetDlgItem(hWndDlg, SCM_DBH_VS);

        CheckRadioButton (hWndDlg, SCM_SAB, SCM_SSB, SCM_SDB);
        src_op = FW_DISPBUF;
        CheckRadioButton (hWndDlg, SCM_DAB, SCM_DSB, SCM_DDB);
        dest_op = FW_DISPBUF;

        dt51_report_buffer_info(device, disp_hndls[0], &shndl_strct);
        src_roi.x = src_roi.y = 0;
        src_roi.width  = (u_short)shndl_strct.width;
        src_roi.height = (u_short)shndl_strct.height;
        smax_buf = (unsigned short)strct_cfg.disp_bufs-1;

        dt51_report_buffer_info(device, disp_hndls[0], &dhndl_strct);
        dest_roi.x = dest_roi.y = 0;
        dest_roi.width  = (u_short)dhndl_strct.width;
        dest_roi.height = (u_short)dhndl_strct.height;
        dmax_buf = (unsigned short)strct_cfg.disp_bufs-1;

        SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
        SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
        SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
        SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
        SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
        SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
        SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
        SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
        SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
        SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
        cwCenter(hWndDlg, 0);
         if(cfg.sys_bufs == 0 )
```

```
            {
            temp = GetDlgItem (hWndDlg, SCM_SSB);
            EnableWindow (temp, FALSE );
            temp = GetDlgItem (hWndDlg, SCM_DSB);
            EnableWindow (temp, FALSE );
            }
        break;

    case WM_COMMAND:
      switch(wParam)
        {
         case IDOK:
          flag = TRUE;
          GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
          GetDlgItemText (hWndDlg, SCM_SSC, szItem2, 20);
          GetDlgItemText (hWndDlg, SCM_SSR, szItem3, 20);
          GetDlgItemText (hWndDlg, SCM_SBW, szItem4, 20);
          GetDlgItemText (hWndDlg, SCM_SBH, szItem5, 20);
          GetDlgItemText (hWndDlg, SCM_DBN, szItem6, 20);
          GetDlgItemText (hWndDlg, SCM_DSC, szItem7, 20);
          GetDlgItemText (hWndDlg, SCM_DSR, szItem8, 20);
          GetDlgItemText (hWndDlg, SCM_DBW, szItem9, 20);
          GetDlgItemText (hWndDlg, SCM_DBH, szItem10, 20);
          EndDialog(hWndDlg, TRUE);
          break;
         case IDCANCEL:
          flag = FALSE;
          EndDialog(hWndDlg, FALSE);
          break;
         case SCM_SAB:
          src_op = FW_ACQBUF;
          GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
              dt51_report_buffer_info(device, acq_hndls[atoi (szItem1)], &shndl_strc
t);
              src_roi.x = src_roi.y = 0;
              src_roi.width  = (u_short)shndl_strct.width;
              src_roi.height = (u_short)shndl_strct.height;
              smax_buf = (unsigned short)strct_cfg.acq_bufs-1;

              SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
              SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
              SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
              SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
              SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
          break;
         case SCM_SDB:
          src_op = FW_DISPBUF;
          GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
              dt51_report_buffer_info(device, disp_hndls[atoi (szItem1)], &shndl_str
ct);
              src_roi.x = src_roi.y = 0;
              src_roi.width  = (u_short)shndl_strct.width;
              src_roi.height = (u_short)shndl_strct.height;
              smax_buf = (unsigned short)strct_cfg.disp_bufs-1;

              SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
              SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
```

```
            SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
         break;
        case SCM_SOB:
         src_op = FW_OVLBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
            dt51_report_buffer_info(device, ovl_hndls[atoi (szItem1)], &shndl_strc
t);
            src_roi.x = src_roi.y = 0;
            src_roi.width  = (u_short)shndl_strct.width;
            src_roi.height = (u_short)shndl_strct.height;
            smax_buf = (unsigned short)strct_cfg.ovl_bufs-1;

            SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
         break;
        case SCM_SSB:
         src_op = FW_SYSBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
            dt51_report_buffer_info(device, sys_hndls[atoi (szItem1)], &shndl_strc
t);
            src_roi.x = src_roi.y = 0;
            src_roi.width  = (u_short)shndl_strct.width;
            src_roi.height = (u_short)shndl_strct.height;
            smax_buf = (unsigned short)strct_cfg.sys_bufs-1;

            SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
         break;
        case SCM_DAB:
         dest_op = FW_ACQBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem6, 20);
            dt51_report_buffer_info(device, acq_hndls[atoi (szItem6)], &dhndl_strc
t);
            dest_roi.x = dest_roi.y = 0;
            dest_roi.width  = (u_short)dhndl_strct.width;
            dest_roi.height = (u_short)dhndl_strct.height;
            dmax_buf = (unsigned short)strct_cfg.acq_bufs-1;

            SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
         break;
        case SCM_DDB:
         dest_op = FW_DISPBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem6, 20);
            dt51_report_buffer_info(device, disp_hndls[atoi (szItem6)], &dhndl_str
ct);
```

```
                dest_roi.x = dest_roi.y = 0;
                dest_roi.width  = (u_short)dhndl_strct.width;
                dest_roi.height = (u_short)dhndl_strct.height;
                dmax_buf = (unsigned short)strct_cfg.disp_bufs-1;

                SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
             break;
            case SCM_DOB:
             dest_op = FW_OVLBUF;
             GetDlgItemText (hWndDlg, SCM_SSR, szItem6, 20);
                dt51_report_buffer_info(device, ovl_hndls[atoi (szItem6)], &dhndl_strc
t);
                dest_roi.x = dest_roi.y = 0;
                dest_roi.width  = (u_short)dhndl_strct.width;
                dest_roi.height = (u_short)dhndl_strct.height;
                dmax_buf = (unsigned short)strct_cfg.ovl_bufs-1;

                SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
             break;
            case SCM_DSB:
             dest_op = FW_SYSBUF;
             GetDlgItemText (hWndDlg, SCM_SSR, szItem6, 20);
                dt51_report_buffer_info(device, sys_hndls[atoi (szItem6)], &dhndl_strc
t);
                dest_roi.x = dest_roi.y = 0;
                dest_roi.width  = (u_short)dhndl_strct.width;
                dest_roi.height = (u_short)dhndl_strct.height;
                dmax_buf = (unsigned short)strct_cfg.sys_bufs-1;

                SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
             break;
           }
         break;

       case WM_VSCROLL:
         switch(wParam)
           {
             case SB_LINEDOWN:
             if (Ssbn == HIWORD(lParam))
              {
                 GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
                 NVal = (unsigned short)atoi (szItem1);

                 if (NVal == 0)
               NVal = (unsigned short)smax_buf;
```

```
    else
   NVal--;

    SetDlgItemInt(hWndDlg, SCM_SBN, NVal, FALSE);
  }
 if (Sssc == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_SSC, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == 0)
   NVal = (short)shndl_strct.width -1;
    else
   NVal--;

    SetDlgItemInt(hWndDlg, SCM_SSC, NVal, FALSE);
  }
 if (Sssr == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == 0)
   NVal = (short)shndl_strct.height -1;
    else
   NVal--;

    SetDlgItemInt(hWndDlg, SCM_SSR, NVal, FALSE);
  }
 if (Ssbw == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_SBW, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == 1)
   NVal = (short)shndl_strct.width;
    else
   NVal--;

    SetDlgItemInt(hWndDlg, SCM_SBW, NVal, FALSE);
  }
 if (Ssbh == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_SBH, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == 1)
   NVal = (short)shndl_strct.height;
    else
   NVal--;

    SetDlgItemInt(hWndDlg, SCM_SBH, NVal, FALSE);
  }
 if (Sdbn == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_DBN, szItem6, 20);
    NVal = (unsigned short)atoi (szItem6);
```

```
    if (NVal == 0)
   NVal = (unsigned short)dmax_buf;
    else
   NVal--;

    SetDlgItemInt(hWndDlg, SCM_DBN, NVal, FALSE);
  }
 if (Sdsc == HIWORD(lParam))
             {
    GetDlgItemText (hWndDlg, SCM_DSC, szItem6, 20);
    NVal = (unsigned short)atoi (szItem6);

    if (NVal == 0)
   NVal = (short)dhndl_strct.width -1;
    else
   NVal--;

    SetDlgItemInt(hWndDlg, SCM_DSC, NVal, FALSE);
  }
 if (Sdsr == HIWORD(lParam))
             {
    GetDlgItemText (hWndDlg, SCM_DSR, szItem6, 20);
    NVal = (unsigned short)atoi (szItem6);

    if (NVal == 0)
   NVal = (short)dhndl_strct.height -1;
    else
   NVal--;

    SetDlgItemInt(hWndDlg, SCM_DSR, NVal, FALSE);
  }
 if (Sdbw == HIWORD(lParam))
             {
    GetDlgItemText (hWndDlg, SCM_DBW, szItem6, 20);
    NVal = (unsigned short)atoi (szItem6);

    if (NVal == 1)
   NVal = (short)dhndl_strct.width;
    else
   NVal--;

    SetDlgItemInt(hWndDlg, SCM_DBW, NVal, FALSE);
  }
 if (Sdbh == HIWORD(lParam))
             {
    GetDlgItemText (hWndDlg, SCM_DBH, szItem6, 20);
    NVal = (unsigned short)atoi (szItem6);

    if (NVal == 1)
   NVal = (short)dhndl_strct.height;
    else
   NVal--;

    SetDlgItemInt(hWndDlg, SCM_DBH, NVal, FALSE);
 }
          break;
```

```
case SB_LINEUP:
if (Ssbn == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == (unsigned short)smax_buf)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SBN, NVal, FALSE);
 }
if (Sssc == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_SSC, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == (unsigned short)shndl_strct.width -1)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SSC, NVal, FALSE);
 }
if (Sssr == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == (unsigned short)shndl_strct.height -1)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SSR, NVal, FALSE);
 }
if (Ssbw == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_SBW, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == (unsigned short)shndl_strct.width)
  NVal = 1;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SBW, NVal, FALSE);
 }
if (Ssbh == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_SBH, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == (unsigned short)shndl_strct.height)
  NVal = 1;
    else
```

```
   NVal++;

      SetDlgItemInt(hWndDlg, SCM_SBH, NVal, FALSE);
  }
 if (Sdbn == HIWORD(lParam))
              {
      GetDlgItemText (hWndDlg, SCM_DBN, szItem6, 20);
      NVal = (unsigned short)atoi (szItem6);

      if (NVal == (unsigned short)dmax_buf)
    NVal = 0;
      else
    NVal++;

      SetDlgItemInt(hWndDlg, SCM_DBN, NVal, FALSE);
  }
 if (Sdsc == HIWORD(lParam))
              {
      GetDlgItemText (hWndDlg, SCM_DSC, szItem6, 20);
      NVal = (unsigned short)atoi (szItem6);

      if (NVal == (unsigned short)dhndl_strct.width -1)
    NVal = 0;
      else
    NVal++;

      SetDlgItemInt(hWndDlg, SCM_DSC, NVal, FALSE);
  }
 if (Sdsr == HIWORD(lParam))
              {
      GetDlgItemText (hWndDlg, SCM_DSR, szItem6, 20);
      NVal = (unsigned short)atoi (szItem6);

      if (NVal == (unsigned short)dhndl_strct.height -1)
    NVal = 0;
      else
    NVal++;

      SetDlgItemInt(hWndDlg, SCM_DSR, NVal, FALSE);
  }
 if (Sdbw == HIWORD(lParam))
              {
      GetDlgItemText (hWndDlg, SCM_DBW, szItem6, 20);
      NVal = (unsigned short)atoi (szItem6);

      if (NVal == (unsigned short)dhndl_strct.width)
    NVal = 1;
      else
    NVal++;

      SetDlgItemInt(hWndDlg, SCM_DBW, NVal, FALSE);
  }
 if (Sdbh == HIWORD(lParam))
              {
      GetDlgItemText (hWndDlg, SCM_DBH, szItem6, 20);
      NVal = (unsigned short)atoi (szItem6);
```

```
            if (NVal == (unsigned short)dhndl_strct.height)
           NVal = 1;
            else
           NVal++;

            SetDlgItemInt(hWndDlg, SCM_DBH, NVal, FALSE);
          }
         break;
        }
      break;
    default:
     return FALSE;
   }
return TRUE;
}


/*============================================================================*/
/*                                        */
/* NAME  : get_H_and_BI                          */
/* DESCRIPTION   : This routine gets the handle to a specific source and   */
/*         destination buffer.                        */
/*                                        */
/* ARGUMENTS     : none (void)                          */
/*                                        */
/* RETURN VALUE : none (void)                          */
/*                                        */
/*============================================================================*/

void get_H_and_BI (void)
{
/*----------------------- Start of routine code     ----------------------*/
 op = src_op;
 Hand_IT();
 switch(src_op)
  {
   case FW_ACQBUF:
     src_hndl = acq_hndls[src_buf_num];
   break;
   case FW_DISPBUF:
     src_hndl = disp_hndls[src_buf_num];
   break;
   case FW_OVLBUF:
     src_hndl = ovl_hndls[src_buf_num];
   break;
   case FW_SYSBUF:
     src_hndl = sys_hndls[src_buf_num];
   break;
  }

 op = dest_op;
 Hand_IT();
 switch(dest_op)
  {
   case FW_ACQBUF:
     dest_hndl = acq_hndls[dest_buf_num];
   break;
```

```
    case FW_DISPBUF:
      dest_hndl = disp_hndls[dest_buf_num];
    break;
    case FW_OVLBUF:
      dest_hndl = ovl_hndls[dest_buf_num];
    break;
    case FW_SYSBUF:
      dest_hndl = sys_hndls[dest_buf_num];
    break;
   }
}


/*=========================================================================*/
/*                                                                         */
/* NAME   :  Move_Image_PixelOp                                            */
/* DESCRIPTION    :  This routine is used to move an image buffer from     */
/*           one onboard buffer to another onboard buffer. The             */
/*           requested TMS34020 Pixel Operations is utilized.              */
/*                                                                         */
/* ARGUMENTS      :  HWND    : hWnd                                        */
/*           HANDLE : hInst                                                */
/*                                                                         */
/* RETURN VALUE : none (void)                                              */
/*                                                                         */
/*=========================================================================*/

void Move_Image_PixelOp (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
  WORD Warr[5];
  FARPROC lpfnDlgProc;
  u_short pixelop;

/*----------------------- Start of routine code     ----------------------*/

  src_buf_num =
  dest_buf_num = 0;
  print_log ("", (LPSTR) NULL);
  lpfnDlgProc = MakeProcInstance((FARPROC)RoiACPOPMP, hInst);
  DialogBox(hInst, (LPSTR)"SCM_ROI_ACPOP", hWnd, lpfnDlgProc);
  FreeProcInstance(lpfnDlgProc);
  if (flag)
    {
     if (Get_DTconfig())
       {
        src_buf_num = atoi(szItem1);
        dest_buf_num = atoi(szItem6);
        src_roi.x = atoi(szItem2);
        src_roi.y = atoi(szItem3);
        src_roi.width = atoi(szItem4);
        src_roi.height = atoi(szItem5);
        dest_roi.x = atoi(szItem7);
        dest_roi.y = atoi(szItem8);
        dest_roi.width = atoi(szItem9);
        dest_roi.height = atoi(szItem10);
```

```
        pixelop = (u_short)selected_pixelop;


     get_H_and_BI();

     Warr [0] = dt51_copy_buf_pixel_op (device, src_hndl, &src_roi, dest_hndl, &dest
_roi, pixelop);
        print_log ("dt51_copy_buf_pixel_op (device, src_hndl, src_roi, dest_hndl, de
st_roi, pixelop) = 0x%x", (LPSTR) Warr);
        disp_err (Warr[0]);
        if((Warr[0] == 0) || (Warr[0] <= -200))
          {
           print_log (" * Source Buffer No.          = 0x%x", (LPSTR) &src_buf_num)
;
           src_roi_config ();
           print_log (" * Destination Buffer No.     = 0x%x", (LPSTR) &dest_buf_num
);
           dest_roi_config ();
           print_log (" * Pixel Operation            = 0x%x", (LPSTR) &pixelop);
          }
     else
        disp_err (Warr[0]);
       }
    }
}


/*=========================================================================*/
/*                                    */
/* NAME  : RoiACPOPMP                          */
/* DESCRIPTION   : This routine controls the dialog box that is used to    */
/*         select the region of interest in the source and     */
/*         destination buffers and the pixel operation.        */
/*                                    */
/* ARGUMENTS     : HWND : hWndDlg                       */
/*         WORD : Message                      */
/*         WORD : wParam                       */
/*         LONG : lParam                       */
/*                                    */
/* RETURN VALUE : none (void)                        */
/*                                    */
/*=========================================================================*/

BOOL FAR PASCAL RoiACPOPMP(HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------ declaration of local variables ----------------------*/
 static HANDLE Ssbn, Sssc, Sssr, Ssbw, Ssbh;
 static HANDLE Sdbn, Sdsc, Sdsr, Sdbw, Sdbh;
 static HANDLE temp  = NULL;
 static HNDL_STRUCT shndl_strct;
 static HNDL_STRUCT dhndl_strct;
 static FW_CONFIG    strct_cfg;
 static unsigned short smax_buf, dmax_buf;

 static unsigned short NVal;

/*---------------------- Start of routine code     ----------------------*/
```

```
HWND hWndCombo1;
int wIndex;
DWORD pop[] = { FW_POP_S,
                FW_POP_S_AND_D,
                FW_POP_S_AND_NOT_D,
                FW_POP_0,
                FW_POP_S_OR_NOT_D,
                FW_POP_S_XNOR_D,
                FW_POP_NOT_D,
                FW_POP_S_NOR_D,
                FW_POP_S_OR_D,
                FW_POP_D,
                FW_POP_S_XOR_D,
                FW_POP_NOT_S_AND_D,
                FW_POP_1,
                FW_POP_NOT_S_OR_D,
                FW_POP_S_NAND_D,
                FW_POP_NOT_S,
                FW_POP_S_ADD_D,
                FW_POP_S_ADD_SAT_D,
                FW_POP_D_SUB_S,
                FW_POP_S_SUB_SAT_D,
                FW_POP_S_MAX_D,
                FW_POP_S_MIN_D,
              };

switch(Message)
  {
   case WM_INITDIALOG:
        dt51_get_config(device,&strct_cfg);
        hWndCombo1    = GetDlgItem(hWndDlg, 170);

     Ssbn = GetDlgItem(hWndDlg, SCM_SBN_VS);
     Sssc = GetDlgItem(hWndDlg, SCM_SSC_VS);
     Sssr = GetDlgItem(hWndDlg, SCM_SSR_VS);
     Ssbw = GetDlgItem(hWndDlg, SCM_SBW_VS);
     Ssbh = GetDlgItem(hWndDlg, SCM_SBH_VS);
     Sdbn = GetDlgItem(hWndDlg, SCM_DBN_VS);
     Sdsc = GetDlgItem(hWndDlg, SCM_DSC_VS);
     Sdsr = GetDlgItem(hWndDlg, SCM_DSR_VS);
     Sdbw = GetDlgItem(hWndDlg, SCM_DBW_VS);
     Sdbh = GetDlgItem(hWndDlg, SCM_DBH_VS);

        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_AND_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_AND_NOT_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_0");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_OR_NOT_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_XNOR_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_NOT_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_NOR_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_OR_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_XOR_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_NOT_S_AND_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_1");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_NOT_S_OR_D");
```

```
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_NAND_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_NOT_S");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_ADD_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_ADD_SAT_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_D_SUB_S");
      SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_SUB_SAT_D");
        SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_MAX_D");
      SendMessage (hWndCombo1,CB_ADDSTRING,0,(LONG)(LPSTR) "FW_POP_S_MIN_D");

      wIndex = 0;
        SendMessage (hWndCombo1,CB_SETCURSEL,0,0L);
      selected_pixelop = pop[wIndex];
        SendMessage (hWndCombo1,CB_GETLBTEXT,wIndex,(LONG)(LPSTR) szItem11);

        CheckRadioButton (hWndDlg, SCM_SAB, SCM_SSB, SCM_SDB);
        src_op = FW_DISPBUF;
        CheckRadioButton (hWndDlg, SCM_DAB, SCM_DSB, SCM_DDB);
        dest_op = FW_DISPBUF;

      dt51_report_buffer_info(device, disp_hndls[0], &shndl_strct);
      src_roi.x = src_roi.y = 0;
      src_roi.width  = (u_short)shndl_strct.width;
      src_roi.height = (u_short)shndl_strct.height;
      smax_buf = (unsigned short)strct_cfg.disp_bufs-1;

      dt51_report_buffer_info(device, disp_hndls[0], &dhndl_strct);
      dest_roi.x = dest_roi.y = 0;
      dest_roi.width  = (u_short)dhndl_strct.width;
      dest_roi.height = (u_short)dhndl_strct.height;
      dmax_buf = (unsigned short)strct_cfg.disp_bufs-1;

        SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
        SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
        SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
        SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
        SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
        SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
        SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
        SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
        SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
        SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
        cwCenter(hWndDlg, 0);
    if(cfg.sys_bufs == 0 )
      {
        temp = GetDlgItem (hWndDlg, SCM_SSB);
        EnableWindow (temp, FALSE );
        temp = GetDlgItem (hWndDlg, SCM_DSB);
        EnableWindow (temp, FALSE );
      }
       break;

  case WM_COMMAND:
       switch(wParam)
         {
          case IDOK:
               flag = TRUE;
               GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
```

```
                GetDlgItemText (hWndDlg, SCM_SSC, szItem2, 20);
                GetDlgItemText (hWndDlg, SCM_SSR, szItem3, 20);
                GetDlgItemText (hWndDlg, SCM_SBW, szItem4, 20);
                GetDlgItemText (hWndDlg, SCM_SBH, szItem5, 20);
                GetDlgItemText (hWndDlg, SCM_DBN, szItem6, 20);
                GetDlgItemText (hWndDlg, SCM_DSC, szItem7, 20);
                GetDlgItemText (hWndDlg, SCM_DSR, szItem8, 20);
                GetDlgItemText (hWndDlg, SCM_DBW, szItem9, 20);
                GetDlgItemText (hWndDlg, SCM_DBH, szItem10, 20);
                EndDialog(hWndDlg, TRUE);
                break;
           case IDCANCEL:
                flag = FALSE;
                EndDialog(hWndDlg, FALSE);
                break;
           case SCM_SAB:
                src_op = FW_ACQBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
            dt51_report_buffer_info(device, acq_hndls[atoi (szItem1)], &shndl_strc
t);
            src_roi.x = src_roi.y = 0;
            src_roi.width  = (u_short)shndl_strct.width;
            src_roi.height = (u_short)shndl_strct.height;
            smax_buf = (unsigned short)strct_cfg.acq_bufs-1;

            SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
               break;
          case SCM_SDB:
                src_op = FW_DISPBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
            dt51_report_buffer_info(device, disp_hndls[atoi (szItem1)], &shndl_str
ct);
            src_roi.x = src_roi.y = 0;
            src_roi.width  = (u_short)shndl_strct.width;
            src_roi.height = (u_short)shndl_strct.height;
            smax_buf = (unsigned short)strct_cfg.disp_bufs-1;

            SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
         break;
        case SCM_SOB:
         src_op = FW_OVLBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
            dt51_report_buffer_info(device, ovl_hndls[atoi (szItem1)], &shndl_strc
t);
            src_roi.x = src_roi.y = 0;
            src_roi.width  = (u_short)shndl_strct.width;
            src_roi.height = (u_short)shndl_strct.height;
            smax_buf = (unsigned short)strct_cfg.ovl_bufs-1;
```

```
            SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
         break;
        case SCM_SSB:
         src_op = FW_SYSBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
            dt51_report_buffer_info(device, sys_hndls[atoi (szItem1)], &shndl_strc
t);
            src_roi.x = src_roi.y = 0;
            src_roi.width  = (u_short)shndl_strct.width;
            src_roi.height = (u_short)shndl_strct.height;
            smax_buf = (unsigned short)strct_cfg.sys_bufs-1;

            SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
            SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
         break;
        case SCM_DAB:
         dest_op = FW_ACQBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem6, 20);
            dt51_report_buffer_info(device, acq_hndls[atoi (szItem6)], &dhndl_strc
t);
            dest_roi.x = dest_roi.y = 0;
            dest_roi.width  = (u_short)dhndl_strct.width;
            dest_roi.height = (u_short)dhndl_strct.height;
            dmax_buf = (unsigned short)strct_cfg.acq_bufs-1;

            SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
         break;
        case SCM_DDB:
         dest_op = FW_DISPBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem6, 20);
            dt51_report_buffer_info(device, disp_hndls[atoi (szItem6)], &dhndl_str
ct);
            dest_roi.x = dest_roi.y = 0;
            dest_roi.width  = (u_short)dhndl_strct.width;
            dest_roi.height = (u_short)dhndl_strct.height;
            dmax_buf = (unsigned short)strct_cfg.disp_bufs-1;

            SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
            SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
         break;
        case SCM_DOB:
         dest_op = FW_OVLBUF;
         GetDlgItemText (hWndDlg, SCM_SSR, szItem6, 20);
```

```
                dt51_report_buffer_info(device, ovl_hndls[atoi (szItem6)], &dhndl_strc
t);
                dest_roi.x = dest_roi.y = 0;
                dest_roi.width  = (u_short)dhndl_strct.width;
                dest_roi.height = (u_short)dhndl_strct.height;
                dmax_buf = (unsigned short)strct_cfg.ovl_bufs-1;

                SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
             break;
            case SCM_DSB:
             dest_op = FW_SYSBUF;
             GetDlgItemText (hWndDlg, SCM_SSR, szItem6, 20);
                dt51_report_buffer_info(device, sys_hndls[atoi (szItem6)], &dhndl_strc
t);
                dest_roi.x = dest_roi.y = 0;
                dest_roi.width  = (u_short)dhndl_strct.width;
                dest_roi.height = (u_short)dhndl_strct.height;
                dmax_buf = (unsigned short)strct_cfg.sys_bufs-1;

                SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
                SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
                   break;
            case 170:
                   switch(HIWORD(lParam))
                      {
                       case CBN_SELCHANGE:
                            wIndex = (WORD) SendMessage (LOWORD(lParam),CB_GETCURSEL,0
,0L);
                        selected_pixelop = pop[wIndex];
                                   SendMessage (LOWORD(lParam),CB_GETLBTEXT,wIndex,(LONG)(
LPSTR) szItem11);
                                   break;
                      }
                 break;
           }
        case WM_VSCROLL:
          switch(wParam)
            {
             case SB_LINEDOWN:
              if (Ssbn == HIWORD(lParam))
                 {
                  GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
                  NVal = (unsigned short)atoi (szItem1);

                  if (NVal == 0)
                NVal = (unsigned short)smax_buf;
                  else
                NVal--;

                  SetDlgItemInt(hWndDlg, SCM_SBN, NVal, FALSE);
```

```
    }
 if (Sssc == HIWORD(lParam))
    {
     GetDlgItemText (hWndDlg, SCM_SSC, szItem1, 20);
     NVal = (unsigned short)atoi (szItem1);

     if (NVal == 0)
   NVal = (unsigned short)shndl_strct.width -1;
     else
   NVal--;

     SetDlgItemInt(hWndDlg, SCM_SSC, NVal, FALSE);
    }
 if (Sssr == HIWORD(lParam))
    {
     GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
     NVal = (unsigned short)atoi (szItem1);

     if (NVal == 0)
   NVal = (unsigned short)shndl_strct.height -1;
     else
   NVal--;

     SetDlgItemInt(hWndDlg, SCM_SSR, NVal, FALSE);
    }
 if (Ssbw == HIWORD(lParam))
    {
     GetDlgItemText (hWndDlg, SCM_SBW, szItem1, 20);
     NVal = (unsigned short)atoi (szItem1);

     if (NVal == 1)
   NVal = (unsigned short)shndl_strct.width;
     else
   NVal--;

     SetDlgItemInt(hWndDlg, SCM_SBW, NVal, FALSE);
    }
 if (Ssbh == HIWORD(lParam))
    {
     GetDlgItemText (hWndDlg, SCM_SBH, szItem1, 20);
     NVal = (unsigned short)atoi (szItem1);

     if (NVal == 1)
   NVal = (unsigned short)shndl_strct.height;
     else
   NVal--;

     SetDlgItemInt(hWndDlg, SCM_SBH, NVal, FALSE);
    }
 if (Sdbn == HIWORD(lParam))
    {
     GetDlgItemText (hWndDlg, SCM_DBN, szItem6, 20);
     NVal = (unsigned short)atoi (szItem6);

     if (NVal == 0)
   NVal = (unsigned short)dmax_buf;
     else
```

```
    NVal--;

      SetDlgItemInt(hWndDlg, SCM_DBN, NVal, FALSE);
     }
 if (Sdsc == HIWORD(lParam))
    {
     GetDlgItemText (hWndDlg, SCM_DSC, szItem6, 20);
     NVal = (unsigned short)atoi (szItem6);

      if (NVal == 0)
   NVal = (unsigned short)dhndl_strct.width -1;
      else
   NVal--;

      SetDlgItemInt(hWndDlg, SCM_DSC, NVal, FALSE);
     }
 if (Sdsr == HIWORD(lParam))
    {
     GetDlgItemText (hWndDlg, SCM_DSR, szItem6, 20);
     NVal = (unsigned short)atoi (szItem6);

      if (NVal == 0)
   NVal = (unsigned short)dhndl_strct.height -1;
      else
   NVal--;

      SetDlgItemInt(hWndDlg, SCM_DSR, NVal, FALSE);
     }
 if (Sdbw == HIWORD(lParam))
    {
     GetDlgItemText (hWndDlg, SCM_DBW, szItem6, 20);
     NVal = (unsigned short)atoi (szItem6);

      if (NVal == 1)
   NVal = (unsigned short)dhndl_strct.width;
      else
   NVal--;

      SetDlgItemInt(hWndDlg, SCM_DBW, NVal, FALSE);
     }
 if (Sdbh == HIWORD(lParam))
    {
     GetDlgItemText (hWndDlg, SCM_DBH, szItem6, 20);
     NVal = (unsigned short)atoi (szItem6);

      if (NVal == 1)
   NVal = (unsigned short)dhndl_strct.height;
      else
   NVal--;

      SetDlgItemInt(hWndDlg, SCM_DBH, NVal, FALSE);
     }
 break;

case SB_LINEUP:
 if (Ssbn == HIWORD(lParam))
              {
```

```
    GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == (unsigned short)smax_buf)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SBN, NVal, FALSE);
 }
if (Sssc == HIWORD(lParam))
             {
    GetDlgItemText (hWndDlg, SCM_SSC, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == (unsigned short)shndl_strct.width -1)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SSC, NVal, FALSE);
 }
if (Sssr == HIWORD(lParam))
             {
    GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == (unsigned short)shndl_strct.height -1)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SSR, NVal, FALSE);
 }
if (Ssbw == HIWORD(lParam))
             {
    GetDlgItemText (hWndDlg, SCM_SBW, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == (unsigned short)shndl_strct.width)
  NVal = 1;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SBW, NVal, FALSE);
 }
if (Ssbh == HIWORD(lParam))
             {
    GetDlgItemText (hWndDlg, SCM_SBH, szItem1, 20);
    NVal = (unsigned short)atoi (szItem1);

    if (NVal == (unsigned short)shndl_strct.height)
  NVal = 1;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SBH, NVal, FALSE);
```

```
 }
if (Sdbn == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_DBN, szItem6, 20);
    NVal = (unsigned short)atoi (szItem6);

    if (NVal == (unsigned short)dmax_buf)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DBN, NVal, FALSE);
 }
if (Sdsc == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_DSC, szItem6, 20);
    NVal = (unsigned short)atoi (szItem6);

    if (NVal == (unsigned short)dhndl_strct.width -1)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DSC, NVal, FALSE);
 }
if (Sdsr == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_DSR, szItem6, 20);
    NVal = (unsigned short)atoi (szItem6);

    if (NVal == (unsigned short)dhndl_strct.height -1)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DSR, NVal, FALSE);
 }
if (Sdbw == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_DBW, szItem6, 20);
    NVal = (unsigned short)atoi (szItem6);

    if (NVal == (unsigned short)dhndl_strct.width)
  NVal = 1;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DBW, NVal, FALSE);
 }
if (Sdbh == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_DBH, szItem6, 20);
    NVal = (unsigned short)atoi (szItem6);

    if (NVal == (unsigned short)dhndl_strct.height)
  NVal = 1;
    else
```

```
            NVal++;

               SetDlgItemInt(hWndDlg, SCM_DBH, NVal, FALSE);
             }
           break;
            }
         break;
    default:
        return FALSE;
   }
return TRUE;
}


/*=========================================================================*/
/*                                        */
/* NAME  : IScale_Image                          */
/* DESCRIPTION   : This routine performs an integer scaling function on    */
/*         an image resident on the DT3851 using the built in      */
/*         firmware scale function.                */
/*                                        */
/* ARGUMENTS     :  HWND   : hWnd                        */
/*          HANDLE : hInst                       */
/*                                        */
/* RETURN VALUE : none (void)                          */
/*                                        */
/*=========================================================================*/

void IScale_Image (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
WORD        Warr[5];
FARPROC    lpfnDlgProc;
u_short    vscale, vmode, hscale, hmode;

/*----------------------- Start of routine code     ----------------------*/
 src_buf_num = dest_buf_num = 0;
 vmode = hmode = FW_COMPACT;
 print_log ("", (LPSTR) NULL);
 /*Edited out by R. Greer 9/9/96*/
 /*lpfnDlgProc = MakeProcInstance((FARPROC)RoiIS, hInst);
 DialogBox(hInst, (LPSTR)"SCM_ROI_IS", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);   */
 /*Edited out by R. Greer 9/9/96*/

 /*Added by R. Greer 9/9/96*/
 flag = TRUE;
 /*Added by R. Greer 9/9/96*/

 if (flag)
   {
     if (Get_DTconfig())
       {
      hOldCursor = SetCursor (LoadCursor(NULL, IDC_WAIT));
//All changes from here were done by R. Greer 9/9/96
      src_buf_num = 0; //atoi(szItem1);
      src_roi.x = 0; //atoi(szItem2);
```

```
      src_roi.y = 0; //atoi(szItem3);
      src_roi.width = 640; //atoi(szItem4);
      src_roi.height = 480; //atoi(szItem5);
      src_op = FW_SYSBUF;  //added by R. Greer 9/9/96
      dest_buf_num = 0; //atoi(szItem6);
      if(current_chan == 0)     //added by R. Greer 9/9/96
        dest_roi.x = 0; //atoi(szItem7);
      else
        dest_roi.x = 320;    //added by R.Greer 9/9/96
      dest_roi.y = 0; //atoi(szItem8);
      dest_roi.width = 320; //atoi(szItem9);
      dest_roi.height = 480; //atoi(szItem10);
      dest_op = FW_DISPBUF; //added by R. Greer 9/9/96
         //vmode = op1;
      vscale = 1; //atoi(szItem11);
         //hmode = op2;
      hscale = 2; //atoi(szItem12);
//All changes above here were done by R. Greer 9/9/96

      get_H_and_BI();

      Warr [0] = dt51_iscale (device, src_hndl, &src_roi, dest_hndl, &dest_roi, vsca
le, vmode, hscale, hmode);
      print_log ("dt51_iscale (device, src_hndl, src_roi, dest_hndl, dest_roi, vscal
e, vmode, hscale, hmode) = 0x%x",
            (LPSTR) Warr);
      if(Warr[0] == 0)
        {
         print_log (" * Source Buffer No.          = 0x%x", (LPSTR) &src_buf_num);
         src_roi_config ();
         print_log (" * Destination Buffer No.     = 0x%x", (LPSTR) &dest_buf_num);
         dest_roi_config ();
         if (vmode == FW_EXPAND)
           print_log (" * Vertically Expand          ", (LPSTR) NULL);
         else
           print_log (" * Vertically Compact         ", (LPSTR) NULL);
         print_log (" * Vertical Scale             = 0x%x", (LPSTR) &vscale);
         if (hmode == FW_EXPAND)
           print_log (" * Horizontally Expand        ", (LPSTR) NULL);
         else
           print_log (" * Horizontally Compact       ", (LPSTR) NULL);
         print_log (" * Horizontal Scale           = 0x%x", (LPSTR) &hscale);
        }
      else
        disp_err (Warr[0]);

      hOldCursor = SetCursor (LoadCursor(NULL, IDC_ARROW));
       }
   }
}


/*=========================================================================*/
/*                                    */
/* NAME  : RoiIS                         */
/* DESCRIPTION   : This routine controls the dsilog box for the selection  */
/*         of the region_of_interest and scaling function of the   */
```

```
/*          iscale function.                                    */
/*                                                              */
/* ARGUMENTS      : HWND : hWndDlg                              */
/*          WORD : Message                                      */
/*          WORD : wParam                                       */
/*          LONG : lParam                                       */
/*                                                              */
/* RETURN VALUE : none (void)                                   */
/*                                                              */
/*==============================================================*/

BOOL FAR PASCAL RoiIS (HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------ declaration of local variables ----------------------*/
 static HANDLE Ssbn, Sssc, Sssr, Ssbw, Ssbh;
 static HANDLE Sdbn, Sdsc, Sdsr, Sdbw, Sdbh;
 static HANDLE Svieb, Shieb;
 static HANDLE temp  = NULL;

 static short NVal;

/*----------------------- Start of routine code     ----------------------*/
 switch(Message)
   {
    case WM_INITDIALOG:
      Ssbn = GetDlgItem(hWndDlg, SCM_SBN_VS);
      Sssc = GetDlgItem(hWndDlg, SCM_SSC_VS);
      Sssr = GetDlgItem(hWndDlg, SCM_SSR_VS);
      Ssbw = GetDlgItem(hWndDlg, SCM_SBW_VS);
      Ssbh = GetDlgItem(hWndDlg, SCM_SBH_VS);
      Sdbn = GetDlgItem(hWndDlg, SCM_DBN_VS);
      Sdsc = GetDlgItem(hWndDlg, SCM_DSC_VS);
      Sdsr = GetDlgItem(hWndDlg, SCM_DSR_VS);
      Sdbw = GetDlgItem(hWndDlg, SCM_DBW_VS);
      Sdbh = GetDlgItem(hWndDlg, SCM_DBH_VS);
      Svieb = GetDlgItem(hWndDlg, SCM_VIEB_VS);
      Shieb = GetDlgItem(hWndDlg, SCM_HIEB_VS);

      CheckRadioButton (hWndDlg, SCM_SAB, SCM_SSB, SCM_SDB);
      src_op = FW_DISPBUF;
      CheckRadioButton (hWndDlg, SCM_DAB, SCM_DSB, SCM_DDB);
      dest_op = FW_DISPBUF;
      CheckRadioButton (hWndDlg, SCM_VIEB, SCM_VICB, SCM_VICB);
      op1 = FW_COMPACT;
      CheckRadioButton (hWndDlg, SCM_HIEB, SCM_HICB, SCM_HICB);
      op2 = FW_COMPACT;
      SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
      SetDlgItemInt(hWndDlg, SCM_ED_VIEB, 2, FALSE);
```

```
    SetDlgItemInt(hWndDlg, SCM_ED_HIEB, 2, FALSE);
    cwCenter(hWndDlg, 0);
    if(cfg.sys_bufs == 0 )
       {
       temp = GetDlgItem (hWndDlg, SCM_SSB);
       EnableWindow (temp, FALSE );
       temp = GetDlgItem (hWndDlg, SCM_DSB);
       EnableWindow (temp, FALSE );
       }
     break;

  case WM_COMMAND:
    switch(wParam)
      {
       case IDOK:
        flag = TRUE;
        GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
        GetDlgItemText (hWndDlg, SCM_SSC, szItem2, 20);
        GetDlgItemText (hWndDlg, SCM_SSR, szItem3, 20);
        GetDlgItemText (hWndDlg, SCM_SBW, szItem4, 20);
        GetDlgItemText (hWndDlg, SCM_SBH, szItem5, 20);
        GetDlgItemText (hWndDlg, SCM_DBN, szItem6, 20);
        GetDlgItemText (hWndDlg, SCM_DSC, szItem7, 20);
        GetDlgItemText (hWndDlg, SCM_DSR, szItem8, 20);
        GetDlgItemText (hWndDlg, SCM_DBW, szItem9, 20);
        GetDlgItemText (hWndDlg, SCM_DBH, szItem10, 20);
        GetDlgItemText (hWndDlg, SCM_ED_VIEB, szItem11, 20);
        GetDlgItemText (hWndDlg, SCM_ED_HIEB, szItem12, 20);
        EndDialog(hWndDlg, TRUE);
        break;
       case IDCANCEL:
        flag = FALSE;
        EndDialog(hWndDlg, FALSE);
        break;
       case SCM_SAB:
        src_op = FW_ACQBUF;
        break;
       case SCM_SDB:
        src_op = FW_DISPBUF;
        break;
       case SCM_SOB:
        src_op = FW_OVLBUF;
        break;
       case SCM_SSB:
        src_op = FW_SYSBUF;
        break;
       case SCM_DAB:
        dest_op = FW_ACQBUF;
        break;
       case SCM_DDB:
        dest_op = FW_DISPBUF;
        break;
       case SCM_DOB:
        dest_op = FW_OVLBUF;
        break;
       case SCM_DSB:
        dest_op = FW_SYSBUF;
```

```
        break;
       case SCM_VIEB:
        op1 = FW_EXPAND;
        break;
       case SCM_VICB:
        op1 = FW_COMPACT;
        break;
       case SCM_HIEB:
        op2 = FW_EXPAND;
        break;
       case SCM_HICB:
        op2 = FW_COMPACT;
        break;
     }
   break;

case WM_VSCROLL:
  switch(wParam)
    {
            case SB_LINEDOWN:
      if (Ssbn == HIWORD(lParam))
                    {
          GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
          NVal = atoi (szItem1);

          if (NVal == 0)
        NVal = 255;
          else
        NVal--;

          SetDlgItemInt(hWndDlg, SCM_SBN, NVal, FALSE);
       }
      if (Sssc == HIWORD(lParam))
                    {
          GetDlgItemText (hWndDlg, SCM_SSC, szItem1, 20);
          NVal = atoi (szItem1);

          if (NVal == 0)
        NVal = 1024;
          else
        NVal--;

          SetDlgItemInt(hWndDlg, SCM_SSC, NVal, FALSE);
       }
      if (Sssr == HIWORD(lParam))
                    {
          GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
          NVal = atoi (szItem1);

          if (NVal == 0)
        NVal = 1024;
          else
        NVal--;

          SetDlgItemInt(hWndDlg, SCM_SSR, NVal, FALSE);
       }
      if (Ssbw == HIWORD(lParam))
```

```
            {
    GetDlgItemText (hWndDlg, SCM_SBW, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_SBW, NVal, FALSE);
 }
if (Ssbh == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_SBH, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_SBH, NVal, FALSE);
 }
if (Sdbn == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_DBN, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 0)
  NVal = 255;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_DBN, NVal, FALSE);
 }
if (Sdsc == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_DSC, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 0)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_DSC, NVal, FALSE);
 }
if (Sdsr == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_DSR, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 0)
  NVal = 1024;
    else
  NVal--;
```

```
    SetDlgItemInt(hWndDlg, SCM_DSR, NVal, FALSE);
 }
if (Sdbw == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_DBW, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_DBW, NVal, FALSE);
 }
if (Sdbh == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_DBH, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_DBH, NVal, FALSE);
 }
if (Svieb == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_ED_VIEB, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_ED_VIEB, NVal, FALSE);
 }
if (Shieb == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_ED_HIEB, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_ED_HIEB, NVal, FALSE);
 }
          break;
      case SB_LINEUP:
if (Ssbn == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
    NVal = atoi (szItem1);
```

```
    if (NVal == 255)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SBN, NVal, FALSE);
 }
if (Sssc == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_SSC, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SSC, NVal, FALSE);
 }
if (Sssr == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SSR, NVal, FALSE);
 }
if (Ssbw == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_SBW, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 1;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SBW, NVal, FALSE);
 }
if (Ssbh == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_SBH, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 1;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SBH, NVal, FALSE);
 }
if (Sdbn == HIWORD(lParam))
              {
```

```
    GetDlgItemText (hWndDlg, SCM_DBN, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 255)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DBN, NVal, FALSE);
 }
if (Sdsc == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_DSC, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DSC, NVal, FALSE);
 }
if (Sdsr == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_DSR, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DSR, NVal, FALSE);
 }
if (Sdbw == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_DBW, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 1;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DBW, NVal, FALSE);
 }
if (Sdbh == HIWORD(lParam))
              {
    GetDlgItemText (hWndDlg, SCM_DBH, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 1;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DBH, NVal, FALSE);
```

```
          }
         if (Svieb == HIWORD(lParam))
                     {
            GetDlgItemText (hWndDlg, SCM_ED_VIEB, szItem1, 20);
            NVal = atoi (szItem1);

            if (NVal == 1024)
           NVal = 1;
            else
           NVal++;

            SetDlgItemInt(hWndDlg, SCM_ED_VIEB, NVal, FALSE);
          }
         if (Shieb == HIWORD(lParam))
                     {
            GetDlgItemText (hWndDlg, SCM_ED_HIEB, szItem1, 20);
            NVal = atoi (szItem1);

            if (NVal == 1024)
           NVal = 1;
            else
           NVal++;

            SetDlgItemInt(hWndDlg, SCM_ED_HIEB, NVal, FALSE);
          }
         break;
       }
      break;

    default:
     return FALSE;
   }
return TRUE;
}


/*=========================================================================*/
/*                                        */
/* NAME  : Img_Rotate_90                          */
/* DESCRIPTION   : This routine allows the user to rotate a buffer on      */
/*         the board 90 degrees either clockwise or        */
/*         counterclockwise using the onboard firmware calls.      */
/*                                        */
/* ARGUMENTS     :  HWND   : hWnd                        */
/*          HANDLE : hInst                        */
/*                                        */
/* RETURN VALUE : none (void)                          */
/*                                        */
/*=========================================================================*/

void Img_Rotate_90 (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 WORD      Warr[5];
 FARPROC   lpfnDlgProc;

/*---------------------- Start of routine code     ----------------------*/
```

```
 src_buf_num = 0;
 dest_buf_num = 0;
 direction = FW_CLOCKWISE;
 print_log ("", (LPSTR) NULL);
 lpfnDlgProc = MakeProcInstance((FARPROC)Rotate_90, hInst);
 DialogBox(hInst, (LPSTR)"SCM_ROTATE_90", hWnd, lpfnDlgProc);
 FreeProcInstance(lpfnDlgProc);
 if (flag)
   {
     if (Get_DTconfig())
       {
      hOldCursor = SetCursor (LoadCursor(NULL, IDC_WAIT));

      src_buf_num = atoi(szItem1);
      dest_buf_num = atoi(szItem6);
      src_roi.x = atoi(szItem2);
      src_roi.y = atoi(szItem3);
      src_roi.width = atoi(szItem4);
      src_roi.height = atoi(szItem5);
      dest_roi.x = atoi(szItem7);
      dest_roi.y = atoi(szItem8);
      dest_roi.width = atoi(szItem9);
      dest_roi.height = atoi(szItem10);

      get_H_and_BI();

      Warr [0] = dt51_rotate_90 (device, src_hndl, &src_roi, dest_hndl, &dest_roi, d
irection);
      print_log ("dt51_rotate_90 (device, src_hndl, src_roi, dest_hndl, dest_roi, di
rection) = 0x%x", (LPSTR) Warr);
      if(Warr[0] == 0)
        {
          print_log (" * Source Buffer No.           = 0x%x", (LPSTR) &src_buf_num);
          src_roi_config ();
          print_log (" * Destination Buffer No.      = 0x%x", (LPSTR) &dest_buf_num)
;
          dest_roi_config ();
        }
      else
        disp_err (Warr[0]);

      hOldCursor = SetCursor (LoadCursor(NULL, IDC_ARROW));
      }
    }
}


/*=========================================================================*/
/*                                    */
/* NAME  : Rotate_90                          */
/* DESCRIPTION   : This routine controls the dialog box that is used       */
/*         to specify the ammount, direction and region that       */
/*         is to be rotated.                      */
/*                                    */
/* ARGUMENTS     : HWND : hWndDlg                     */
/*         WORD : Message                     */
/*         WORD : wParam                      */
```

```
/*          LONG : lParam                          */
/*                                                 */
/* RETURN VALUE : none (void)                      */
/*                                                 */
/*=================================================================*/

BOOL FAR PASCAL Rotate_90 (HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------ declaration of local variables ----------------------*/
 static HANDLE Ssbn, Sssc, Sssr, Ssbw, Ssbh;
 static HANDLE Sdbn, Sdsc, Sdsr, Sdbw, Sdbh;
 static HANDLE temp  = NULL;

 static short NVal;

/*------------------------ Start of routine code     ----------------------*/
 switch(Message)
   {
    case WM_INITDIALOG:
      Ssbn = GetDlgItem(hWndDlg, SCM_SBN_VS);
      Sssc = GetDlgItem(hWndDlg, SCM_SSC_VS);
      Sssr = GetDlgItem(hWndDlg, SCM_SSR_VS);
      Ssbw = GetDlgItem(hWndDlg, SCM_SBW_VS);
      Ssbh = GetDlgItem(hWndDlg, SCM_SBH_VS);
      Sdbn = GetDlgItem(hWndDlg, SCM_DBN_VS);
      Sdsc = GetDlgItem(hWndDlg, SCM_DSC_VS);
      Sdsr = GetDlgItem(hWndDlg, SCM_DSR_VS);
      Sdbw = GetDlgItem(hWndDlg, SCM_DBW_VS);
      Sdbh = GetDlgItem(hWndDlg, SCM_DBH_VS);

      CheckRadioButton (hWndDlg, SCM_SAB, SCM_SSB, SCM_SDB);
      src_op = FW_DISPBUF;
      CheckRadioButton (hWndDlg, SCM_DAB, SCM_DSB, SCM_DDB);
      dest_op = FW_DISPBUF;
      SetDlgItemInt(hWndDlg, SCM_SBN, 0, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SSC, src_roi.x, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SSR, src_roi.y, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SBW, src_roi.width, FALSE);
      SetDlgItemInt(hWndDlg, SCM_SBH, src_roi.height, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DBN, 0, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DSC, dest_roi.x, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DSR, dest_roi.y, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DBW, dest_roi.width, FALSE);
      SetDlgItemInt(hWndDlg, SCM_DBH, dest_roi.height, FALSE);
      CheckRadioButton (hWndDlg, SCM_CW, SCM_CCW, SCM_CW);
      direction = FW_CLOCKWISE;
      cwCenter(hWndDlg, 0);
        if(cfg.sys_bufs == 0 )
           {
           temp = GetDlgItem (hWndDlg, SCM_SSB);
           EnableWindow (temp, FALSE );
           temp = GetDlgItem (hWndDlg, SCM_DSB);
           EnableWindow (temp, FALSE );
           }
       break;

   case WM_COMMAND:
```

```
    switch(wParam)
      {
       case IDOK:
        flag = TRUE;
        GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
        GetDlgItemText (hWndDlg, SCM_SSC, szItem2, 20);
        GetDlgItemText (hWndDlg, SCM_SSR, szItem3, 20);
        GetDlgItemText (hWndDlg, SCM_SBW, szItem4, 20);
        GetDlgItemText (hWndDlg, SCM_SBH, szItem5, 20);
        GetDlgItemText (hWndDlg, SCM_DBN, szItem6, 20);
        GetDlgItemText (hWndDlg, SCM_DSC, szItem7, 20);
        GetDlgItemText (hWndDlg, SCM_DSR, szItem8, 20);
        GetDlgItemText (hWndDlg, SCM_DBW, szItem9, 20);
        GetDlgItemText (hWndDlg, SCM_DBH, szItem10, 20);
        EndDialog(hWndDlg, TRUE);
        break;
       case IDCANCEL:
        flag = FALSE;
        EndDialog(hWndDlg, FALSE);
        break;
       case SCM_SAB:
        src_op = FW_ACQBUF;
        break;
       case SCM_SDB:
        src_op = FW_DISPBUF;
        break;
       case SCM_SOB:
        src_op = FW_OVLBUF;
        break;
       case SCM_SSB:
        src_op = FW_SYSBUF;
        break;
       case SCM_DAB:
        dest_op = FW_ACQBUF;
        break;
       case SCM_DDB:
        dest_op = FW_DISPBUF;
        break;
       case SCM_DOB:
        dest_op = FW_OVLBUF;
        break;
       case SCM_DSB:
        dest_op = FW_SYSBUF;
        break;
       case SCM_CW:
        direction = FW_CLOCKWISE;
        break;
       case SCM_CCW:
        direction = FW_COUNTERCLOCKWISE;
        break;
      }
    break;

  case WM_VSCROLL:
    switch(wParam)
      {
              case SB_LINEDOWN:
```

```
if (Ssbn == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 0)
  NVal = 255;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_SBN, NVal, FALSE);
 }
if (Sssc == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_SSC, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 0)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_SSC, NVal, FALSE);
 }
if (Sssr == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 0)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_SSR, NVal, FALSE);
 }
if (Ssbw == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_SBW, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_SBW, NVal, FALSE);
 }
if (Ssbh == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_SBH, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1)
  NVal = 1024;
    else
  NVal--;
```

```
    SetDlgItemInt(hWndDlg, SCM_SBH, NVal, FALSE);
 }
if (Sdbn == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_DBN, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 0)
  NVal = 255;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_DBN, NVal, FALSE);
 }
if (Sdsc == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_DSC, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 0)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_DSC, NVal, FALSE);
 }
if (Sdsr == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_DSR, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 0)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_DSR, NVal, FALSE);
 }
if (Sdbw == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_DBW, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1)
  NVal = 1024;
    else
  NVal--;

    SetDlgItemInt(hWndDlg, SCM_DBW, NVal, FALSE);
 }
if (Sdbh == HIWORD(lParam))
            {
    GetDlgItemText (hWndDlg, SCM_DBH, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1)
```

```
   NVal = 1024;
     else
   NVal--;

     SetDlgItemInt(hWndDlg, SCM_DBH, NVal, FALSE);
  }
            break;
       case SB_LINEUP:
 if (Ssbn == HIWORD(lParam))
              {
     GetDlgItemText (hWndDlg, SCM_SBN, szItem1, 20);
     NVal = atoi (szItem1);

     if (NVal == 255)
   NVal = 0;
     else
   NVal++;

     SetDlgItemInt(hWndDlg, SCM_SBN, NVal, FALSE);
  }
 if (Sssc == HIWORD(lParam))
              {
     GetDlgItemText (hWndDlg, SCM_SSC, szItem1, 20);
     NVal = atoi (szItem1);

     if (NVal == 1024)
   NVal = 0;
     else
   NVal++;

     SetDlgItemInt(hWndDlg, SCM_SSC, NVal, FALSE);
  }
 if (Sssr == HIWORD(lParam))
              {
     GetDlgItemText (hWndDlg, SCM_SSR, szItem1, 20);
     NVal = atoi (szItem1);

     if (NVal == 1024)
   NVal = 0;
     else
   NVal++;

     SetDlgItemInt(hWndDlg, SCM_SSR, NVal, FALSE);
  }
 if (Ssbw == HIWORD(lParam))
              {
     GetDlgItemText (hWndDlg, SCM_SBW, szItem1, 20);
     NVal = atoi (szItem1);

     if (NVal == 1024)
   NVal = 1;
     else
   NVal++;

     SetDlgItemInt(hWndDlg, SCM_SBW, NVal, FALSE);
  }
 if (Ssbh == HIWORD(lParam))
```

```
                {
    GetDlgItemText (hWndDlg, SCM_SBH, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 1;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_SBH, NVal, FALSE);
 }
if (Sdbn == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_DBN, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 255)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DBN, NVal, FALSE);
 }
if (Sdsc == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_DSC, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DSC, NVal, FALSE);
 }
if (Sdsr == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_DSR, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 0;
    else
  NVal++;

    SetDlgItemInt(hWndDlg, SCM_DSR, NVal, FALSE);
 }
if (Sdbw == HIWORD(lParam))
                {
    GetDlgItemText (hWndDlg, SCM_DBW, szItem1, 20);
    NVal = atoi (szItem1);

    if (NVal == 1024)
  NVal = 1;
    else
  NVal++;
```

```
            SetDlgItemInt(hWndDlg, SCM_DBW, NVal, FALSE);
          }
         if (Sdbh == HIWORD(lParam))
                    {
            GetDlgItemText (hWndDlg, SCM_DBH, szItem1, 20);
            NVal = atoi (szItem1);

            if (NVal == 1024)
           NVal = 1;
            else
           NVal++;

            SetDlgItemInt(hWndDlg, SCM_DBH, NVal, FALSE);
          }
         break;
       }
      break;
    default:
     return FALSE;
   }
return TRUE;
}


/*=========================================================================*/
/*                                                                         */
/* NAME  : AuxBufs_allox                                                   */
/* DESCRIPTION   : This routine is used to allocate auxiliary (system)     */
/*         buffers on the DT3851.                                          */
/*                                                                         */
/* ARGUMENTS     :  HWND   : hWnd                                          */
/*          HANDLE : hInst                                                 */
/*                                                                         */
/* RETURN VALUE : none (void)                                              */
/*                                                                         */
/*=========================================================================*/
/*  Modified by Richard Greer to bypass dialog box and automatically       */
/*  one 640X480 auxiliary buffer                                           */
/***************************************************************************/

void AuxBufs_allox (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 HMENU           hMenu;
 FARPROC         lpfnDlgProc;
 u_short         aux_buf;
 int             i, nbufs;
 u_long       rows, cols, psize;
 WORD            Warr[5];
 DWORD           Darr[5];

/*----------------------- Start of routine code     ----------------------*/
 print_log ("", (LPSTR) NULL);
 rows = cols = psize = 0L;
 nbufs = 0;
 if (Get_DTconfig())
   {
```

```
      Darr[3] = aux_buf = cfg.sys_bufs;

      //Taken out be R. Greer 9/4/96 to bypass dialog box
      /*lpfnDlgProc = MakeProcInstance((FARPROC)AbaMP, hInst);
      DialogBox(hInst, (LPSTR)"SCM_ABA", hWnd, lpfnDlgProc);
      FreeProcInstance(lpfnDlgProc);*/
      //Taken out by R. Greer 9/4/96

      //put in to make sure go in to if.
        flag = TRUE;
      //end of first change

      if (flag)
     {                  //stuff at end commented out by R. Greer 9/4/96
           nbufs = 1; //atoi (szItem1);
      Darr[0] = rows = 480; //atol (szItem2);
      Darr[1] = cols = 640; //atol (szItem3);
           Darr[2] = psize = 8; //atol (szItem4);

           for (i=0; i<nbufs; i++)
            {
           Darr [4] = (u_long)dt51_alloc_aux ( device, rows, cols, psize, &sys_hndls
[aux_buf]);
           print_log ("dt51_alloc_aux ( device, 0x%lx, 0x%lx, 0x%lx, &sys_hndls[0x%x
]) = 0x%lx", (LPSTR) Darr);
           if ((int)Darr[4] == 0)
         {
          print_log (" * Auxiliary Buffer  Rows : 0x%lx", (LPSTR) &rows);
          print_log (" *                   Cols : 0x%lx", (LPSTR) &cols);
          print_log (" *                   Pixel Size : 0x%lx", (LPSTR) &psize);
          print_log ("", (LPSTR) NULL);
                }
           else
         disp_err ((int)Darr[4]);
             }

       if (Get_DTconfig())
         {
           if (cfg.sys_bufs > 0)
         {
          hMenu = GetMenu (hWndMain);
              EnableMenuItem (hMenu, IDM_I_FREEAUXBUF, MF_ENABLED);
             EnableMenuItem (hMenu, IDM_I_DELETEALLBUFS, MF_ENABLED);
            EnableMenuItem (hMenu, IDM_A_DELETEMANY, MF_ENABLED);
             EnableMenuItem (hMenu, IDM_A_REPLAYMANY, MF_ENABLED);
                   EnableMenuItem (hMenu, IDM_A_LOOPREPLAY, MF_ENABLED);
          Warr [0] = cfg.sys_bufs;
          Warr [1] = dt51_get_buffer_handles (device, FW_SYSBUF, cfg.sys_bufs, sys_h
ndls);
          print_log ("dt51_get_buffer_handles (device, FW_SYSBUF, 0x%x, sys_hndls) =
 0x%x", (LPSTR) Warr);
                 if(Warr[1] != 0)
            disp_err (Warr[1]);
         }
              }
          }
   }
```

```
}


/*=========================================================================*/
/*                                                                         */
/* NAME   : AbaMP                                                          */
/* DESCRIPTION    : This routine controls the dialog box that is used to   */
/*          specify the parameters for the system buffer to                */
/*          allocate.                                                      */
/*                                                                         */
/* ARGUMENTS      : HWND : hWndDlg                                         */
/*          WORD : Message                                                 */
/*          WORD : wParam                                                  */
/*          LONG : lParam                                                  */
/*                                                                         */
/* RETURN VALUE : none (void)                                              */
/*                                                                         */
/*=========================================================================*/

BOOL FAR PASCAL AbaMP (HWND hWndDlg, WORD Message, WORD wParam, LONG lParam)
{
/*------------------- declaration of local variables ----------------------*/
static HANDLE      Snb, Snr, Snc, Snp;
static u_short     NVal;
static u_short     MaxB;

/*------------------------ Start of routine code     ----------------------*/
 switch(Message)
   {
    case WM_INITDIALOG:
      Snb = GetDlgItem(hWndDlg, SCM_NABS_VS);
      Snr = GetDlgItem(hWndDlg, SCM_ROW_VS);
      Snc = GetDlgItem(hWndDlg, SCM_COL_VS);
      Snp = GetDlgItem(hWndDlg, SCM_PSIZE_VS);
            MaxB = 2000;
      SetDlgItemInt(hWndDlg, 108, cfg.sys_bufs, FALSE);
      SetDlgItemInt(hWndDlg, SCM_NABS, 1, FALSE);
      SetDlgItemInt(hWndDlg, SCM_ROWS, 480, FALSE);
      SetDlgItemInt(hWndDlg, SCM_COLS, 640, FALSE);
      SetDlgItemInt(hWndDlg, SCM_PSIZE, 8, FALSE);
      cwCenter(hWndDlg, 0);
      break;

    case WM_COMMAND:
      switch(wParam)
        {
         case IDOK:
          flag = TRUE;
          GetDlgItemText (hWndDlg, SCM_NABS, szItem1, 20);
          GetDlgItemText (hWndDlg, SCM_ROWS, szItem2, 20);
          GetDlgItemText (hWndDlg, SCM_COLS, szItem3, 20);
          GetDlgItemText (hWndDlg, SCM_PSIZE, szItem4, 20);
          EndDialog(hWndDlg, TRUE);
          break;
         case IDCANCEL:
          flag = FALSE;
          EndDialog(hWndDlg, FALSE);
```

```
      break;
    }
  break;
case WM_VSCROLL:
  switch(wParam)
    {
          case SB_LINEDOWN:
     if (Snb == HIWORD(lParam))
                 {
        GetDlgItemText (hWndDlg, SCM_NABS, szItem1, 20);
        NVal = atoi (szItem1);

        if (NVal == 1)
       NVal = 250;
        else
       NVal--;

        SetDlgItemInt(hWndDlg, SCM_NABS, NVal, FALSE);
      }
     if (Snr == HIWORD(lParam))
                 {
        GetDlgItemText (hWndDlg, SCM_ROWS, szItem1, 20);
        NVal = atoi (szItem1);

        if (NVal == 0)
       NVal = MaxB;
        else
       NVal--;

        SetDlgItemInt(hWndDlg, SCM_ROWS, NVal, FALSE);
      }
     if (Snc == HIWORD(lParam))
                 {
        GetDlgItemText (hWndDlg, SCM_COLS, szItem1, 20);
        NVal = atoi (szItem1);

        if (NVal == 0)
       NVal = MaxB;
        else
       NVal--;

        SetDlgItemInt(hWndDlg, SCM_COLS, NVal, FALSE);
      }
     if (Snp == HIWORD(lParam))
                 {
        GetDlgItemText (hWndDlg, SCM_PSIZE, szItem1, 20);
        NVal = atoi (szItem1);

        if (NVal == 0)
       NVal = 16;
        else
       NVal -=4;

        SetDlgItemInt(hWndDlg, SCM_PSIZE, NVal, FALSE);
      }
            break;
       case SB_LINEUP:
```

```
            if (Snb == HIWORD(lParam))
                        {
                GetDlgItemText (hWndDlg, SCM_NABS, szItem1, 20);
                NVal = atoi (szItem1);

                if (NVal == 250)
              NVal = 1;
                else
              NVal++;

                SetDlgItemInt(hWndDlg, SCM_NABS, NVal, FALSE);
             }
            if (Snr == HIWORD(lParam))
                        {
                GetDlgItemText (hWndDlg, SCM_ROWS, szItem1, 20);
                NVal = atoi (szItem1);

                if (NVal == MaxB)
              NVal = 0;
                else
              NVal++;

                SetDlgItemInt(hWndDlg, SCM_ROWS, NVal, FALSE);
             }
            if (Snc == HIWORD(lParam))
                        {
                GetDlgItemText (hWndDlg, SCM_COLS, szItem1, 20);
                NVal = atoi (szItem1);

                if (NVal == MaxB)
              NVal = 0;
                else
              NVal++;

                SetDlgItemInt(hWndDlg, SCM_COLS, NVal, FALSE);
             }
            if (Snp == HIWORD(lParam))
                        {
                GetDlgItemText (hWndDlg, SCM_PSIZE, szItem1, 20);
                NVal = atoi (szItem1);

                if (NVal == 16)
              NVal = 0;
                else
              NVal +=4;

                SetDlgItemInt(hWndDlg, SCM_PSIZE, NVal, FALSE);
             }
            break;
          }
        break;

      default:
       return FALSE;
     }
return TRUE;
}
```

```
/*=========================================================================*/
/*                                                                         */
/* NAME  : Free_allox                                                      */
/* DESCRIPTION   : This routine is used to specify an aux. system buffer   */
/*         to free from allocation.                                        */
/*                                                                         */
/* ARGUMENTS     :  HWND   : hWnd                                          */
/*          HANDLE : hInst                                                 */
/*                                                                         */
/* RETURN VALUE : none (void)                                              */
/*                                                                         */
/*=========================================================================*/

void Free_allox (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 HMENU      hMenu;
 FARPROC       lpfnDlgProc;
 WORD       Warr[5];

/*----------------------- Start of routine code     ----------------------*/
 print_log ("", (LPSTR) NULL);
 if(Get_DTconfig())
   {
    Warr [0] = dt51_get_buffer_handles ( device, FW_SYSBUF, cfg.sys_bufs, sys_hndls)
;
    print_log ("dt51_get_buffer_handles (device, FW_SYSBUF, cfg.sys_bufs, sys_hndls)
 = 0x%x", (LPSTR) Warr);
    if(Warr[0] != 0)
     disp_err (Warr[0]);
    else
      {
       NUM_FREQ = 1;
       lpfnDlgProc = MakeProcInstance((FARPROC)NFramesMP, hInst);
       DialogBox(hInst, (LPSTR)"SCM_FRAMES", hWnd, lpfnDlgProc);
       FreeProcInstance(lpfnDlgProc);
       if(flag)
      {
        Warr[0] = value = (atoi (szItem) - 1);
        Warr [1] = dt51_free_aux ( device, sys_hndls[value]);
        print_log ("dt51_free_aux ( device, sys_hndls[0x%x]) = 0x%x", (LPSTR) Warr);

        if(Warr[1] != 0)
          disp_err (Warr[1]);
        else
          {
           if(Get_DTconfig())
         {
          if (cfg.sys_bufs == 0)
            {
             hMenu = GetMenu (hWndMain);
                  EnableMenuItem (hMenu, IDM_I_FREEAUXBUF, MF_GRAYED);
                      EnableMenuItem (hMenu, IDM_I_DELETEALLBUFS, MF_GRAYED);

                    EnableMenuItem (hMenu, IDM_A_DELETEMANY, MF_GRAYED);
```

```
                    EnableMenuItem (hMenu, IDM_A_REPLAYMANY, MF_GRAYED);
                  EnableMenuItem (hMenu, IDM_A_LOOPREPLAY, MF_GRAYED);
            }

          Warr [0] = dt51_get_buffer_handles ( device, FW_SYSBUF, cfg.sys_bufs, sys_
hndls);
          print_log ("dt51_get_buffer_handles (device, FW_SYSBUF, cfg.sys_bufs, sys_
hndls) = 0x%x", (LPSTR) Warr);
          if(Warr[0] != 0)
            disp_err (Warr[0]);
         }
          }
      }
      }
    }
}


/*=========================================================================*/
/*                                        */
/* NAME  : FREE_Sequential                        */
/* DESCRIPTION   : This routine is used to specify a sequence of aux.      */
/*         buffers to free from allocation.           */
/*                                        */
/* ARGUMENTS     :  HWND   : hWnd                     */
/*          HANDLE : hInst                    */
/*                                        */
/* RETURN VALUE : none (void)                     */
/*                                        */
/*=========================================================================*/

void FREE_Sequential (HWND hWnd, HANDLE hInst)
{
/*------------------ declaration of local variables ----------------------*/
 WORD        Warr[5];
 FARPROC     lpfnDlgProc;
 HMENU       hMenu;
 HDC         hDC;
 RECT        zClient;
 u_short     sequens;
 u_short     ISOK;
 int         i;
 int         seq_val;

/*----------------------- Start of routine code     ---------------------*/
 print_log ("", (LPSTR) NULL);
 if(Get_DTconfig())
   {
    Warr [0] = dt51_get_buffer_handles ( device, FW_SYSBUF, cfg.sys_bufs, sys_hndls)
;
    print_log ("dt51_get_buffer_handles (device, FW_SYSBUF, cfg.sys_bufs, sys_hndls)
 = 0x%x", (LPSTR) Warr);
    if(Warr[0] != 0)
      disp_err (Warr[0]);
    else
      {
       NUM_FREQ = 1;
```

```
        print_log ("", (LPSTR) NULL);
        lpfnDlgProc = MakeProcInstance((FARPROC)SeqMotionMP, hInst);
        DialogBox(hInst, (LPSTR)"SCM_MOTION", hWnd, lpfnDlgProc);
        FreeProcInstance(lpfnDlgProc);
        if(flag)
       {
        seq_val = atoi (szItem1);
        sequens = atoi (szItem2);

        hOldCursor = SetCursor (LoadCursor(NULL, IDC_WAIT));
        ISOK = TRUE;

        sprintf(szString, " ***>>> Deleting Sequence <*** ");
        GetClientRect (hWndMenu, &zClient);
        hDC = GetDC (hWndMenu);
        ExtTextOut (hDC, 0, 0, ETO_OPAQUE, &zClient, szString, lstrlen(szString), NUL
L);
        ReleaseDC (hWndMenu, hDC);

        for (i=(int)sequens; i < (int)(seq_val+sequens); i++)
          {
           Warr[0] = dt51_free_aux (device, sys_hndls[i]);
           if (Warr[0] != 0)
         {
          ISOK = FALSE;
          disp_err (Warr[0]);
                     break;
           }
          }

        if (ISOK)
          {
           if(Get_DTconfig())
            {
          if (cfg.sys_bufs == 0)
            {
             hMenu = GetMenu (hWndMain);
             EnableMenuItem (hMenu, IDM_I_FREEAUXBUF, MF_GRAYED);
                EnableMenuItem (hMenu, IDM_I_DELETEALLBUFS, MF_GRAYED);
                EnableMenuItem (hMenu, IDM_A_DELETEMANY, MF_GRAYED);
             EnableMenuItem (hMenu, IDM_A_REPLAYMANY, MF_GRAYED);
                    EnableMenuItem (hMenu, IDM_A_LOOPREPLAY, MF_GRAYED);
            }

          Warr [0] = dt51_get_buffer_handles ( device, FW_SYSBUF, cfg.sys_bufs, sys_h
ndls);
          print_log ("dt51_get_buffer_handles (device, FW_SYSBUF, cfg.sys_bufs, sys_h
ndls) = 0x%x", (LPSTR) Warr);
          if(Warr[0] != 0)
            disp_err (Warr[0]);

             sprintf(szString, " ***>>> Sequence Deleted <*** ");
                GetClientRect (hWndMenu, &zClient);
          hDC = GetDC (hWndMenu);
          ExtTextOut (hDC, 0, 0, ETO_OPAQUE, &zClient, szString, lstrlen(szString), N
ULL);
          ReleaseDC (hWndMenu, hDC);
```

```
        }
         }
        else
          ErrMsg("Unable to deallocate an auxiliary buffer check system");

        SetCursor (hOldCursor);
         }
       }
   }
}


/*=========================================================================*/
/*                                        */
/* NAME   : AllFree_allox                          */
/* DESCRIPTION    : This routine is used to free all allocated aux. buffers */
/*                                        */
/* ARGUMENTS      :  HWND    : hWnd                    */
/*                                        */
/* RETURN VALUE : none (void)                       */
/*                                        */
/*=========================================================================*/

void AllFree_allox (HWND hWnd)
{
/*------------------ declaration of local variables ---------------------*/
 HMENU        hMenu;
 WORD         Warr[3];
 int          i;

/*----------------------- Start of routine code     ----------------------*/
 hOldCursor = SetCursor (LoadCursor(NULL, IDC_WAIT));
 if(Get_DTconfig())
   {
    Warr [0] = dt51_get_buffer_handles ( device, FW_SYSBUF, cfg.sys_bufs, sys_hndls)
;
    print_log ("dt51_get_buffer_handles (device, FW_SYSBUF, cfg.sys_bufs, sys_hndls)
 = 0x%x", (LPSTR) Warr);
    if(Warr[0] != 0)
      disp_err (Warr[0]);
    else
        {
       for (i=0; i < (int)cfg.sys_bufs; i++)
      {
        Warr[0] = dt51_free_aux (device, sys_hndls[i]);
        if (Warr[0] != 0)
                {
          disp_err (Warr[0]);
                 break;
                }
      }

       if(Get_DTconfig())
      {
       if (cfg.sys_bufs == 0)
         {
          hMenu = GetMenu (hWnd);
```

```
            EnableMenuItem (hMenu, IDM_I_FREEAUXBUF, MF_GRAYED);
            EnableMenuItem (hMenu, IDM_I_DELETEALLBUFS, MF_GRAYED);
            EnableMenuItem (hMenu, IDM_A_DELETEMANY, MF_GRAYED);
            EnableMenuItem (hMenu, IDM_A_REPLAYMANY, MF_GRAYED);
            EnableMenuItem (hMenu, IDM_A_LOOPREPLAY, MF_GRAYED);
           }
       }
       }
    }
 SetCursor (hOldCursor);
}

/*----------------------------   END OF AUXMEM.C ----------------------------*/
```

```
/*****************************************************************************/
/*****************************************************************************/
/***                           ***************************/
/***  ARMM Windows Program           ***************************/
/***                           ***************************/
/***  Last Update: 9/30/96           ***************************/
/***  by  Richard Greer ***************************/
/*****************************************************************************/
/*****************************************************************************/
/***                           ***************************/
/***  Main Header File               ***************************/
/***   & Functions Declarations      ***************************/
/***                           ***************************/
/*****************************************************************************/
/*****************************************************************************/

/*****************************************************************************/
/***                                                                       ***/
/***  This file defines variables and function prototypes needed for       ***/
/***    the DT3852 board.  The UT Research team has added in the           ***/
/***    function prototypes for the functions they added for the crack     ***/
/***    sealer operation.                                                  ***/
/***                                                                       ***/
/*****************************************************************************/

/*****************************************************************************/
/***  Compiler Dependencies  ***************************************************/
/*****************************************************************************/
/* if using borland c establish the following */
#if defined (__BORLANDC__)
#define _cdecl cdecl
#define __MSC
#endif

/* if not using MSC V7.0 redefine functions */
#if (_MSC_VER < 700)
#define _ltoa ltoa
#define _itoa itoa
#endif

/*****************************************************************************/
/***  Standard Include Files  **************************************************/
/*****************************************************************************/
#include <windows.h>
#include <commdlg.h>
#include <dlgs.h>
#include <string.h>
#include <stdlib.h>
#include <stdarg.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>

/*****************************************************************************/
/***  Data Translation Include Files  ******************************************/
/*****************************************************************************/
```

```
#include <c:\dti\include\dtityp.h>
#include <c:\dti\include\id_info.h>
#include <c:\dti\dt3851\fwdefs.h>
#include <c:\dti\dt3851\fwstrcts.h>
#include <c:\dti\dt3851\tiga.h>
#include <c:\dti\dt3851\typedefs.h>
#include <c:\dti\dt3851\dt51dll.h>

/*****************************************************************************/
/***  Include File for Menu ID Numbers   ***********************************/
/*****************************************************************************/
#include "51menus.h"

/*****************************************************************************/
/***  Globally Defined Constants  ******************************************/
/*****************************************************************************/

/* Global definitions for file I/O read/write case decision operations */
#define FILE_READ      0
#define FILE_WRITE     1

/* define board types */
#define FW_ORIGINAL_HARDWARE    0x00    /* hardware revision 0 - 4 */
#define FW_REVA_HARDWARE    0x01    /* hardware revision 5 - present rev A version *
/
#define FW_REVB_HARDWARE    0x02    /* B-board hardware revision 0 - present */

/* Constants used for graphing Lookup Tables */
#define MAXDEFLECTION  255
#define X_START 51
#define X_ORIGIN       0

#define Y_ORIGIN       0
#define FLIP_Y_AXIS    -1

/* Constants used for graphing Histogram Data */
#define NUM_HIS_X_VALUES 5
#define X_HIS_START    0
#define X_HIS_STEP     128

/* Maximum possible row & column dimensions */
#define MAX_COL      1024
#define MAX_ROW      1024

/* Maximum number of convolution coefficient mask arrays */
#define NUM_MASKS 5

/* board response log constants */
#define LOGENT      500
#define LINE_COUNT     10

/* generic windowing constants */
#define REOPEN_NORMAL  0
#define REOPEN_ZOOM    1
#define REOPEN_DEFAULT 2

/* Maximum number of allowed buffers */
```

```
#define FW_NUM_ACQ_BUFS  16
#define FW_NUM_DISP_BUFS    4
#define FW_NUM_OVL_BUFS     4
#define FW_NUM_SYS_BUFS     256

/* defined constants for file types (image,log,lut, etc)  */
/* Image File Format Types */
#define FORMbin      0
#define FORMiris         1
#define FORMtiff         2
#define FORMpcx      3
#define FORMgif      4
#define FORMpict         5
#define FORMtarga        6
#define FORMdib      7
#define FORMexmpl        8
#define FORMgli      9

/* Misc. support file types */
#define LUT_FILE_TYPE        0
#define LOG_FILE_TYPE        1
#define CAMERA_SETUP_FILE_TYPE   2
#define SEQUENCE_FILE_TYPE   3


/*************************************************************************/
/***  Structure Definitions  *******************************************/
/*************************************************************************/

/*-----------------------------------------------------------------------*/
/*  Structure Definition for GLI setup values for camera setup.  */
/*-----------------------------------------------------------------------*/

typedef struct {
    float  vert;      // vertical video frequency
    float  horz;      // horizontal video frequency
    float  pixel;     // pixel clock frequency
    int    signal;    // input signal = 0 for composite video
             //      = 1 for variable scan
    int    async;     // wait for asyncronous video if = 1
    int    inter;     // no interlaced = 0
             // interlaced    = 1 for start on even field
             // interlaced    = 2 for start on odd field
             // interlaced    = 3 for start on next field
    int    trigger;  // invert frame trigger = 1 if on; = 0 if off
    int    line;      // invert line sync = 1 if on; = 0 if off
    int    clock;     // invert pixel clock = 1 if on; = 0 if off
    int    volt;      // DC if = 0
             // AC Coupled if = 1;
    int    filter;    // filter = 0 if off
             //    = 1 if (60 hz)
             //    = 2 if (50 hz)
    int    inputChan;     // input channel #
    int    syncChan;      // sync channel #
    int    syncMode;      // camera drives FG if = 0
             // FG drives camera if = 1
    float  gain;      // gain value
```

```
    float  offset;   // offset value
    float  reference;    // reference value
    int    level;    // sync level
    int    pixPerLine;   // total pixels per line
    int    startPix;     // starting pixel on line
    int    lenPix;   // length pixel line
    int    stepPix;  // increment steps
    int    startClamp;   // clamp start
    int    lenClamp;     // clamp length
    int    startZero;    // zero start
    int    lenZero;  // zero length
    float  lenIgnore;    // sync ignore length
    float  posInsert;    // sync insert position
    int    rows;     // total rows
    int    startRow;     // starting row
    int    lenRow;   // length of row
    int    stepRow;  // row increment steps
    float  rowIgnore;    // sync ignore row
    float  rowInsert;    // sync insert row
} GLI_CAMERA_SETUP;


/*************************************************************************/
/***  Function Declarations  *******************************************/
/*************************************************************************/

/*************************************************************************/
/***  Main Windows Program Declarations  *******************************/
/*************************************************************************/
long FAR PASCAL WndProc (HWND, UINT, WPARAM, LPARAM);  // Main Windows Loop
BOOL _cdecl MenuCommand (HWND, WORD, WORD, LONG);      // process menu item commands

/*************************************************************************/
/***  File Menu/Module Routines  ***************************************/
/*************************************************************************/
void Open_Image (HWND, HANDLE);               // Restore Image File to Selected Buff
er
void restore_img (u_long, u_long, u_short);                              // function th
at performs image restoration
void put_pix (u_long, u_long, int, int, u_char);          // put pixel into on board
 memory buffer
void put_pix_line (u_long, u_long, u_long, int, int, u_char *);  // put line of pixe
ls into on board memory buffer
void put_pix_rgn (int, u_char far *);                 // put region of pixels into o
n board memory buffer

/*************************************************************************/
/*                                                                       */
/*  These are functions that have been created by the ARMM team          */
/*                                                                       */
/*************************************************************************/

void line_snap(POINT out[], POINT snap_out[1000], long *duration);
    //YSK's line snapping function
void path_plan(POINT out[], POINT path_out[], long *dur); //YSK's path planning func
tion
```

```
void InitControl(void); //Initialize Aerotech controller
void TraceCrack(POINT out2[]); //Trace the crack w/ Aerotech controller
void AdjustPoints(POINT outs[]); //Adjust points into the array
unsigned char get_value(int,int);

/*********************************************************************/
/*                                                                   */
/*   End of added functions                                          */
/*                                                                   */
/*********************************************************************/

HANDLE FAR PASCAL Open_Save_Dlg (HWND, unsigned, WORD, LONG);     // Dialog Box for
Save or Restore Image File

void Save_Image (HWND, HANDLE);               // Save Image to File from selected bu
ffer
void write_img (u_long, u_long, u_short);             // function that performs imag
e writing
u_char get_pix (u_long, u_long, int, int);            // get pixel from on board mem
ory buffer
void get_pix_line (u_long, u_long, u_long, int, int, u_char *);  // get line of pixe
ls from on board memory buffer
void get_pix_rgn (int, u_char far *);                 // get region of pixels from o
n board memory buffer

// TIFF I/O Support Functions:
void fputWord (FILE *, int);
void fputLong (FILE *, long);
unsigned int fgetWord (FILE *);
unsigned long fgetLong (FILE *);
char *tag_name (int);
void WriteTifTag (FILE *, int, int, long, long);
void DecodeTag (FILE *);

void Log_IO (HWND, HANDLE, int);      // Save & Restore current Log
void write_F_log (void);           // write current log to specified file
void read_F_log (void);        // read a stored log back from a file

void Lut_IO (HWND, HANDLE, int);      // Save & Restore current Luts
void write_luts (void);        // write ILut/OLut/OVLut to a specified file
void read_luts (void);             // restore previously saved ILut/OLut/OVLut

void Camera_Setup_IO (HWND, HANDLE, int);  // Save & Restore current camera setup
void write_F_form (void);                     // write current format/sync/clks/and
a2d structures to file
void write_F_form_2 (void);                  // write current format/sync/clks/and a
2d structures to file
int read_F_form (void);         // read previously stored structure setups from a fi
le
int read_F_form_2 (void);           // read previously stored structure setups from
a file
void fill_gli_structure_for_write ( float, float, float, float, GLI_CAMERA_SETUP * )
;
void unload_gli_structure_after_read ( float *, float *, float *, float *, GLI_CAMER
A_SETUP * );
float dt51_int2offs (int);          // GLI value conversion routine
float dt51_int2ref (int);           // GLI value conversion routine
```

```
int dt51_offs2int (float);          // GLI value conversion routine
int dt51_ref2int (float);           // GLI value conversion routine

UINT CALLBACK Open_Save_Dlg (HWND, UINT, WORD, LONG);
UINT CALLBACK Camera_Setup_Read_Save_Dlg (HWND, UINT, WORD, LONG);
int        increment_value (int, int, int);
int        decrement_value (int, int, int);
void           read_image_file (void);
void           write_image_file (void);
int        misc_support_file_read_or_write (int);

/*************************************************************************/
/***  Init Menu/Module Routines  ***************************************/
/*************************************************************************/
void RHard (void);    // Reset DT3851/52 Hardware Only
void RHard_Disp (void);  // Reset DT3851/52 Hardware, establish default settings

BOOL _cdecl Get_DTconfig (void);  // Get configuration of DT3851/52
void zero_config (void);       // zero out configuration structure

BOOL _cdecl config_tiga (void);   // Get current TIGA configuration
BOOL _cdecl cmode_tiga (HWND, HANDLE);     // Get possible TIGA mode configuration
void print_mode_info (MODEINFO far *);     // print current tiga mode info to board r
esponse log

void show_gsp_heap (void);  // calls gsph_maxheap & gsph_totalfree  to give indicati
on of available memory

void Get_Handles (HWND, HANDLE);              // Get buffer handles for: Acquire / D
isplay / Overlay / Auxiliary
BOOL FAR PASCAL SelectBMP (HWND, WORD, WORD, LONG);  // Dialog Box for Get buffer ha
ndles for: Acquire/Display/Overlay/Auxiliary
void Hand_IT (void);                                                  // based on dialog
 box selection obtain/display proper buffer handle info

void HandlDisp (void);   // Get current display handle

void Get_BInfo (HWND, HANDLE);              // Get buffer info for specified handle
BOOL FAR PASCAL EditBMP (HWND, WORD, WORD, LONG);  // Dialog Box for Get buffer info
 for specified handle
void BInfo_IT (void);                                          // based on dialog box s
election obtain proper buffer info information
void Report_BInfo (void);               // display proper buffer information

/*************************************************************************/
/***  LUTs Menu/Module Routines  ***************************************/
/*************************************************************************/
void Read_Graph_InLUT (HANDLE);   // Read Input Lookup Table and display graphically
void Read_Graph_OutLUT (HANDLE);  // Read Output Lookup Table and display graphicall
y
void Read_Graph_OvlLUT (HANDLE);  // Read Overlay Lookup Table and display graphical
ly

// Graph Support Functions:
VOID FAR PASCAL PopLutInitMapping (HWND, HDC);
VOID FAR PASCAL LutDrawMarker (HDC, int, int, int, int, int);
long FAR PASCAL ILGraphWndProc (HWND, UINT, WPARAM, LPARAM);  // child window routin
```

```
e to monitor graphs of LUTs and Histograms
long FAR PASCAL OLGraphWndProc (HWND, UINT, WPARAM, LPARAM);  // child window routin
e to monitor graphs of LUTs and Histograms
long FAR PASCAL OvLGraphWndProc (HWND, UINT, WPARAM, LPARAM);  // child window routi
ne to monitor graphs of LUTs and Histograms

long FAR PASCAL LutWndProc (HWND, UINT, WPARAM, LPARAM);  // lut response child wind
ow, displays current LUT values
void update_luts (void);                    // update lookup table log
BOOL _cdecl print_lut (const LPSTR, LPSTR);        // write to lookup table log

void Up_LUTS (HANDLE);                    // Manually Adjust Input, Output, and Overl
ay  Lookup Tables
BOOL FAR PASCAL LutsMP (HWND, WORD, WORD, LONG);     // Dialog Box for Manually Adju
st Input, Output, and Overlay Lookup Tables
long FAR PASCAL OLutProc (HWND, UINT, WPARAM, LPARAM);   // child window routine use
d to manually edit OLUTs to specific values
long FAR PASCAL ScrollProc (HWND, WORD, WORD, LONG);     // child window routine to
control scroll bars within OLutProc

BOOL _cdecl lut_mem (void);   // allocate memory for lookup tables
BOOL _cdecl lut_init (void);  // initialize luts: ILut & OLut - set to Identity, OVL
ut - read in
BOOL _cdecl lut_lock (void);  // lock lut memory before use
void lut_unlock (void);       // unlock lut memory after use
void lut_free (void);          // free all memory allocated for luts

/*************************************************************************/
/***  VIDEO Menu/Module Routines  ****************************************/
/*************************************************************************/
void Set_FormComplt (HWND, HANDLE);          // Read/Edit entire format memory
void zero_formem (void);                  // zero out format memory structure
BOOL FAR PASCAL FormatMP (HWND, WORD, WORD, LONG);  // Dialog Box for Read/Edit enti
re format memory
void cpy_formem (void);               // copy format memory values established in dia
log box into format memory structure
void format_config (void);                // prep data for proper display in response
 log
void format_center (void);                // establish source & destination ROI's bas
ed on format memory

void Set_FormEdit (HWND, HANDLE);              // Edit specific format memory setups
BOOL FAR PASCAL EdFormatMP (HWND, WORD, WORD, LONG);  // Dialog Box for Edit specifi
c format memory setups

void SyncFormatMem (HWND, HANDLE);           // Read/Edit synchronization format mem
ory
void zero_syncformem (void);                 // zero out sync format memory structur
e
BOOL FAR PASCAL SyncFormatMP (HWND, WORD, WORD, LONG);   // Dialog Box for Read/Edit
 synchronization format memory
void cpy_syncformem (void);              // copy syncformat memory values establishe
d in dialog box into syncformat structure
void syncformat_config (void);               // prep data for proper display in resp
onse log

void EditSyncFormatMem (HWND, HANDLE);             // Edit signal in synchronization
```

```
 format memory
BOOL FAR PASCAL EdSyncFormatMP (HWND, WORD, WORD, LONG);  // Edit signal in synchron
ization format memory

void Set_ALL_ClockFreqs (HWND, HANDLE);       // Read/Edit all camera clock frequenc
ies
BOOL FAR PASCAL AllClocks (HWND, WORD, WORD, LONG);  // Dialog Box for Read/Edit all
 camera clock frequencies
void Set_Extrn_ClockFreq (HWND, HANDLE);          // Read/Edit external clock freque
ncy
void Set_Horiz_ClockFreq (HWND, HANDLE);          // Read/Edit horizontal scan frequ
ency
void Set_Vert_ClockFreq (HWND, HANDLE);       // Read/Edit vertical scan frequency
void Set_Input_ClockFreq (HWND, HANDLE);          // Read/Edit input scan frequency
BOOL FAR PASCAL DClockMP (HWND, WORD, WORD, LONG);   // Dialog Box for Read/Edit ext
/hor/vert/input/disp clock frequency

void Setup_A2D (HWND, HANDLE);                 // Read/Edit Input A/D parameters
BOOL FAR PASCAL InputADMP (HWND, WORD, WORD, LONG);  // Dialog Box for Read/Edit Inp
ut A/D parameters
void zero_A2D (void);                      // zero out A2D structure
void gain_config (void);                   // prep data for proper display in respons
e log

void Setup_Video_Inputs (HWND, HANDLE);        // Read/Edit video input and sync cha
nnels
BOOL FAR PASCAL SelChnSync (HWND, WORD, WORD, LONG);  // Dialog Box for Read/Edit vi
deo input and sync channels

/*************************************************************************/
/***  DISPLAY Menu/Module Routines  ************************************/
/*************************************************************************/
void DDisp (void);  // Disable display
void EDisp (void);  // Enable display

void Set_Disp_ClockFreq (HWND, HANDLE);  // Read/Edit display clock frequency

void Set_ZPS (HWND, HANDLE);               // Read/Edit displays Zoom, Pan, and Scro
ll states
void zero_zps (void);                                      // zero out zoop, pan,
and scroll structure
BOOL FAR PASCAL ZoomMP (HWND, WORD, WORD, LONG);  // Dialog Box for Read/Edit displa
ys Zoom, Pan, and Scroll states
void zps_config (void);            // prep data for proper display in response log

void EVga (void);  // Enable VGA passthru
void DVga (void);  // Disable VGA passthru

void EOvl (void);  // Enable Overlay
void DOvl (void);  // Disable Overlay

void Set_TPvalue (HWND, HANDLE);             // Set transparent pixel value
BOOL FAR PASCAL NValueMP (HWND, WORD, WORD, LONG);  // Dialog Box for Set transparen
t pixel value

/*************************************************************************/
/***  ACQUIRE Menu/Module Routines  ************************************/
```

```
/*************************************************************************/
void Set_ACQsetup (HWND, HANDLE);            // Read/Edit acquire setup configuratio
n
void zero_acq_setup (void);                 // zero out acquire setup structure
void zero_wIndex (void);                    // zero out generic variables
BOOL FAR PASCAL ACQSETMP (HWND, WORD, WORD, LONG);  // Dialog Box for Read/Edit acqu
ire setup configuration
void acq_config (void);                  // prep data for proper display in response log

void Set_ImgAcqSetup (HWND, HANDLE);              // Read/Edit image acquisition par
ameters
void zero_img_acq_setup (void);            // zero out image acquire setup structure
BOOL FAR PASCAL ImgAcqSetupMP (HWND, WORD, WORD, LONG);  // Dialog Box for Read/Edit
 image acquisition parameters
void img_acq_config (void);                // prep data for proper display in respons
e log

void Stop_Video (void);  // Stop passthru or terminate FOCUS mode

void Check_Video (void);  // Check for a completed acquisition

void Trig_On (HWND);   // Acquire using externally triggered input
void Trig_Off (HWND);  // Acquire using standard input

void ACQ_Image (HWND, HANDLE);                // Acquire full frame
void FOCUS_Image (HWND, HANDLE);              // Continuosly Acuire full frame image
void Set_Bufs (HWND, HANDLE);                 // establish buffer for acquire
void zero_tmphdl (void);                    // zero out tmpporary handle structure
void LIVE_Image (HWND, HANDLE);             // Acuire live full frame images
BOOL FAR PASCAL NFramesMP (HWND, WORD, WORD, LONG);  // Dialog Box for Number of fra
mes

void ROI_Acq_Image (HWND, HANDLE);           // Acquire region of interest
void ROI_FOCUS_Image (HWND, HANDLE);             // Continuosly Acquire region of im
age
void ROI_LIVE_Image (HWND, HANDLE);          // Acquire live region of image
BOOL FAR PASCAL AcqRoiMP (HWND, WORD, WORD, LONG);  // Dialog Box for Acquire region
 of interest
void roi_config (void);                  // prep data for proper display in response log
void src_roi_config (void);                 // prep data for proper display in response
 log
void dest_roi_config (void);                // prep data for proper display in resp
onse log

void Average_IT (HWND, HANDLE, int);              // True & Recursive frame averagin
g
BOOL FAR PASCAL AverageMP (HWND, WORD, WORD, LONG);  // Dialog Box for Average frame
s true/recursive/post

void ACQ_Sequential (HWND, HANDLE);          // Sequentially acquire a series of ima
ges
BOOL FAR PASCAL NSequencesMP (HWND, WORD, WORD, LONG);   // Dialog Box for Sequentia
lly acquire a series of images
void RUN_Sequential (HWND, HANDLE);          // Replay sequentially acquired series
void LOOP_Sequential (HWND, HANDLE);             // Continually Loop-Thru acquired s
eries
BOOL FAR PASCAL SeqMotionMP (HWND, WORD, WORD, LONG);    // Dialog Box for Replay se
```

```
quentially acquired series

void Four_ACQ_Images (HWND, HANDLE);  // Acquire & Display from all 4 Video Inputs -
 in series

/*************************************************************************/
/***  AUXMEM Menu/Module Routines  *************************************/
/*************************************************************************/
void Move_Image (HWND, HANDLE);          // Copy a buffer's contents to another buffe
r
BOOL FAR PASCAL RoiACMP (HWND, WORD, WORD, LONG);  // Dialog Box for Copy a buffer's
 contents to another buffer
void get_H_and_BI (void);                // establish coorect source and destination
buffer handles/info

void Move_Image_PixelOp (HWND, HANDLE);      // Perform pixel operation on 2 buffers
BOOL FAR PASCAL RoiACPOPMP (HWND, WORD, WORD, LONG);  // Dialog Box for Performing p
ixel operation on 2 bufs

void IScale_Image (HWND, HANDLE);          // Perform Integer Scaling on image data
BOOL FAR PASCAL RoiIS (HWND, WORD, WORD, LONG);  // Dialog Box for Perform Integer S
caling on image data

void Img_Rotate_90 (HWND, HANDLE);             // 90 Degree Rotation on image
BOOL FAR PASCAL Rotate_90 (HWND, WORD, WORD, LONG);  // Dialog Box for 90 Degree Rot
ation on image

void AuxBufs_allox (HWND, HANDLE);         // Allocates onboard auxiliary buffer
BOOL FAR PASCAL AbaMP (HWND, WORD, WORD, LONG);  // Dialog Box for Allocates onboard
 auxiliary buffer

void Free_allox (HWND, HANDLE);  // Free specified auxiliary buffer

void FREE_Sequential (HWND, HANDLE);  // Free a series of auxiliary buffers

void AllFree_allox (HWND);  // Free all auxiliary buffers

/*************************************************************************/
/***  SW_EXTAS Menu/Module Routines  ***********************************/
/*************************************************************************/
void Hist_Src_Set_Glbls (int);               // set base_addr, pitch, hieght, width o
f histogram image buffer parameters

void disp_acq (void);  // Copy Source directly to Destination Buffer
void acq_disp (void);  // Copy Destination directly to Source Buffer

// Wipes Any Buffer ROI to specified Color
void ClearImageArea (u_long, u_long, unsigned, unsigned, unsigned, unsigned, unsigne
d);

/* Image Processing Routines derived from Craig A. Lindley's Practical Image Process
ing in C */

void hist_img (HWND, HANDLE);                // Calculate Histogram of Destination Bu
f
BOOL FAR PASCAL HistoCM (HWND, WORD, WORD, LONG);      // Dialog Box for Obtain Hist
ogram ROI
```

```
BOOL FAR PASCAL ShowHistoCM (HWND, WORD, WORD, LONG);  // Dialog Box for Generate &
Plot Histogram of Destination Buf
void hist_stats (int, int);              // Generate Statistics based on Histogram of
 Destination Buf
void Set_HStats (HWND);              // display stats of histogram within dialog box
void Set_HGraph (HWND);              // display graph of histogram within dialog box
void perform_histogram ( void );            // perform the histogramming of a buffer
void Paint_Histogram_ROI_values ( HWND );       // paint histogram buffer and roi va
lues
BOOL Verify_Histogram_and_ROI_values ( HWND );      // verify histogram buffer and r
oi values

// Graph Support Functions:
VOID FAR PASCAL MrkDrawMarker (HDC, int, int, int, int, int);
VOID FAR PASCAL PopInitMapping (HWND, HDC);
VOID FAR PASCAL PopDrawLabels (HDC);
VOID FAR PASCAL CheckmenuCreate (HWND);

void pix_config (int, int);  // display X, Y, and linear address of pixel locations
within display buffer

void Convolution (HWND, HANDLE);             // Setup to perform Convolution on SRC
Buffer
BOOL FAR PASCAL ConvoCM (HWND, WORD, WORD, LONG);   // Dialog Box for Performs an Co
nvolution on SRC Buffer

// draw horizontal & vertical colored lines (destructive) within buffer
void _cdecl draw_V_line (u_long, u_long, u_long, u_long, int, int, int, int);
void _cdecl draw_H_line (u_long, u_long, u_long, u_long, int, int, int, int);

/*************************************************************************/
/***  Utils.c Function Support Routines  *********************************/
/*************************************************************************/
void ErrMsg (LPSTR sz,...);  // display error messages
void disp_err (int);
void SpinTheMSGLoop (void);

BOOL _cdecl GetMemory (void);
void FreeMemory (void);

BOOL _cdecl InitApplication (HWND);              // create classes
BOOL _cdecl InitInstance (HWND, int);               // create the windows
void cwCenter (HWND, int);

BOOL _cdecl contact_tiga (void);
BOOL _cdecl init_tiga (BOOL);
BOOL _cdecl term_tiga (void);
BOOL _cdecl dt_error_handler (WORD, WORD);

long FAR PASCAL LogWndProc (HWND, UINT, WPARAM, LPARAM);     // board response log c
hild window routine displays status after each command
BOOL _cdecl print_log (const LPSTR, LPSTR);

long FAR PASCAL MenuWndProc (HWND, UINT, WPARAM, LPARAM);    // status line child wi
ndow routune to give description of menu options

void increase_val (HWND, int, u_short, u_short);
```

```
void decrease_val (HWND, int, u_short, u_short);
void increase_long_val (HWND, int, u_long, u_long);
void decrease_long_val (HWND, int, u_long, u_long);

/*************************************************************************/
/***  Help Menu/Module Routines  ***************************************/
/*************************************************************************/
BOOL FAR PASCAL AboutMP (HWND, WORD, WORD, LONG);      // Dialog Box for About Box r
outine
WORD GetSystemResources (void);
```

```
/*****************************************************************************/
/*****************************************************************************/
/***                           ***************************/
/***  ARMM Windows Program     ***************************/
/***                           ***************************/
/***  Menu & Dialog Box Variable Declarations    ***************************/
/***                           ***************************/
/*****************************************************************************/
/*****************************************************************************/
/***  The only menu constants added by the research team are in the file***/
/***   and SW_Extras menus.  IDM_ENT and IDM_SPONGE in the file menu    ***/
/***   are used to save points in the data array to a file for testing  ***/
/***   purposes.  In the SW_Extras menu, there are constants for        ***/
/***   erasing, refreshing the screen, line-snapping, adjusting the     ***/
/***   points to get rid of image distortion, initializing the Aerotech ***/
/***   motor controller,  and sealing the cracks,                       ***/
/*****************************************************************************/

/*****************************************************************************/
/***  Defined Menu Structure  ************************************************/
/*****************************************************************************/

/*****************************************************************************/
/***  HELP Menu Option Routiness  ********************************************/
/*****************************************************************************/
#define IDM_ABOUT          900         // About program & author

/*****************************************************************************/
/***  File Menu Option Routines  *********************************************/
/*****************************************************************************/
#define IDM_FILE           1000
#define IDM_OPEN            1010       // Restore Image File to Selected Buffer
#define IDM_SAVE            1015       // Save Image to File from selected buffer
#define IDM_I_LOGOFF            1020       // Turn DT3851/52 Response Log OFF
#define IDM_I_LOGON         1025       // Turn DT3851/52 Response Log ON
#define IDM_I_LUTON         1030       // Turn Lookup Table Log ON
#define IDM_I_LUTOFF            1035       // Turn Lookup Table Log OFF
#define IDM_IO_LOG_READ     1040       // Restore current DT3851/52 Log
#define IDM_IO_LOG_WRITE        1045       // Save current DT3851/52 Log
#define IDM_IO_LUTS         1050       // Restore Lookup Table data
#define IDM_IO_LUTS_READ        1055       // Restore Lookup Table data
#define IDM_IO_LUTS_WRITE       1057       // Save Lookup Table data
#define IDM_I_FILEFORMATMEM     1060       // Restore current camera setup
#define IDM_I_FILEFORMATMEM_READ       1062    // Restore current camera setup
#define IDM_I_FILEFORMATMEM_WRITE      1065    // Save current camera setup
//added by Richard Greer
#define IDM_ENT             1070        //For printing to a file
#define IDM_SPONGE          1071        //Also for printing to a file
//End of added for this part of menu

#define IDM_EXIT            1080       // Exit Program

/*****************************************************************************/
/***  Init Menu Option Routines  *********************************************/
/*****************************************************************************/
#define IDM_INIT           2000
```

```
#define IDM_R_RESET_H           2005        // Reset DT3851/52 Hardware Only
#define IDM_R_HD_RESET               2006   // Reset DT3851/52 Hardware, establish de
fault settings
#define IDM_I_CONFIGURATION     2010        // Get configuration of DT3851/52
#define IDM_I_DLL_DR                 2011   // Set display resolution of DT3851/52
#define IDM_I_TIGA_GC                2012   // Get current TIGA configuration
#define IDM_I_TIGA_GM                2013   // Get possible TIGA mode configuration
#define IDM_I_TIGA_GSP               2014   // Get available heap space
#define IDM_I_GETHANDLES        2015        // Get buffer handles
#define IDM_I_REPORTDISPLAYHANDLE      2020     // Get current display handle
#define IDM_I_REPORTBUFFERINFO      2025        // Get buffer info for specified hand
le

/*****************************************************************************/
/***  LUTs Menu Option Routines  *******************************************/
/*****************************************************************************/
#define IDM_LUTS            3000
#define IDM_READ_ILUT            3005       // Read Input Lookup Table and display gr
aphically
#define IDM_READ_OLUTS               3006   // Read Output Lookup Table and display g
raphically
#define IDM_READ_OVLUTS           3007  // Read Overlay Lookup Table and display grap
hically
#define IDM_WRITE_LUTS           3010       // Manually Adjust Input, Output, and Ove
rlay  LUTs

/*****************************************************************************/
/***  VIDEO Menu Option Routines  *******************************************/
/*****************************************************************************/
#define IDM_INPUT           4000
#define IDM_I_FORMATMEMORY       4005       // Read/Edit entire format memory
#define IDM_I_EDITFORMATMEM      4010       // Edit specific format memory setups
#define IDM_I_SYNCFORMATMEM      4015       // Read/Edit synchronization format memor
y
#define IDM_I_EDITSYNCFORMATMEM 4020        // Edit signal in synchronization format
memory
#define IDM_I_ALLCLKFREQ         4025       // Read/Edit all camera clock frequencies
#define IDM_I_EXTCLOCKFREQ           4026   // Read/Edit external clock frequency
#define IDM_I_HORIZSCANFREQ          4027   // Read/Edit horizontal scan frequency
#define IDM_I_VERTSCANFREQ           4028   // Read/Edit vertical scan frequency
#define IDM_I_INPUTFREQ           4029  // Read/Edit input scan frequency
#define IDM_I_GAINOFFSET         4030       // Read/Edit Input A/D parameters
#define IDM_I_VIDEOINPUTSYNC        4035        // Read/Edit video input and sync cha
nnels

/*****************************************************************************/
/***  DISPLAY Menu Option Routines  ****************************************/
/*****************************************************************************/
#define IDM_DISPLAY         5000
#define IDM_D_DISABLE            5005       // Disable display
#define IDM_D_ENABLE                 5006   // Enable display
#define IDM_D_CLOCK          5010       // Read/Edit display clock frequency
#define IDM_D_ZPSM           5015       // Read/Edit displays Zoom, Pan, and Scroll s
tates
#define IDM_D_VGAPASS_EN         5020       // Enable VGA passthru
#define IDM_D_VGAPASS_DIS            5021   // Disable VGA passthru
#define IDM_O_ENAB_OVERLAY       5025       // Enable Overlay
```

```
#define IDM_O_DISAB_OVERLAY          5026  // Disable Overlay
#define IDM_O_PASSINDEX     5030        // Set transparent pixel value

/*************************************************************************/
/***  ACQUIRE Menu Option Routines  ****************************************/
/*************************************************************************/
#define IDM_ACQUIRE         6000
#define IDM_A_ACQUIRESETUP       6005        // Read/Edit acquire setup parameters
#define IDM_A_IMGACQUIRESETUP             6006  // Read/Edit image acquisition parame
ters
#define IDM_C_FREEZEFRAME        6010        // Stop passthru or terminate FOCUS mode
#define IDM_A_CHECKACQUIRECOMPLETE     6015    // Check for a completed acquisition
#define IDM_A_ENABLE_TRIGGER         6020       // Acquire using externally triggered
 input
#define IDM_A_DISABLE_TRIGGER             6021  // Acquire without triggered input
#define IDM_A_ACQUIREFULLFRAME       6025       // Acquire full frame
#define IDM_A_FOCUSFULLFRAME              6026  // Acquire full frames sequentially
#define IDM_A_PASSTHRUFULLFRAME 6030        // Acuire live full frame images
#define IDM_A_ACQUIREROI         6035        // Acquire region of interest
#define IDM_A_FOCUSROI                 6036  // Acquire region of interest sequentiall
y
#define IDM_A_PASSTHRUROI        6040        // Acquire live region of image
#define IDM_A_ACQUIRETAVG        6045        // True frame average
#define IDM_A_ACQUIREPTAVG             6046  // Post frame average
#define IDM_A_ACQUIRERAVG        6050        // True recursive frame averaging
#define IDM_A_ACQUIREPRAVG             6051  // Post recursive frame averaging
#define IDM_A_ACQUIREMANY        6055        // Sequentially acquire a series of image
s
#define IDM_A_REPLAYMANY               6056  // Replay sequentially acquired series
#define IDM_A_LOOPREPLAY               6057  // Continually Loop-Thru acquired series
#define IDM_A_FOURFULLFRAME      6060        // Acquire from 4 video inputs in series

/*************************************************************************/
/***  AUXMEM Menu Option Routines  *****************************************/
/*************************************************************************/
#define IDM_AUXMEM          7000
#define IDM_I_ALLOCATEAUXBUF        7020   // Allocates onboard auxiliary buffer
#define IDM_I_FREEAUXBUF         7025    // Free specified auxiliary buffer
#define IDM_A_DELETEMANY         7030    // Free a series of auxiliary buffers
#define IDM_I_DELETEALLBUFS      7035    // Free all auxiliary buffers

/*************************************************************************/
/***  SW_EXTRAS Menu Option Routines  **************************************/
/*************************************************************************/
#define IDM_A_MEMORYCOPY         7005  // Copy a buffer's contents to another buffer
#define IDM_A_MEMORYCOPYPIXELOP    7007  // Perform pixel operation between 2 buffers
#define IDM_HISTO           8030  // Generate & Plot Histogram
#define IDM_CONVOLUTION     8055  // Performs a Convolution on SRC Buffer
#define IDM_A_ISCALE             7010  // Perform Integer Scaling on image data
#define IDM_A_ROTATE_90     7015  // 90 Degree Rotation on image
//Added menu items by Richard Greer
#define IDM_INITCONTROLLER  8060    //Initializes aerotech motor
#define IDM_LINESNAP         8061    //Performs Line Snapping
#define IDM_ADJUSTPOINT      8062    //Adjusts the points for distortion
#define IDM_TRACECRACK       8063    //Traces the crack
#define IDM_REFRESH          8064    //Refreshes the screen
#define IDM_ERASE            8065    //Erase and start over
```

```
/*****************************************************************************/
/***  System Menu Option Routines  *******************************************/
/*****************************************************************************/
#define IDM_SYS         9000        // handle to system menu

/*****************************************************************************/
/***  Dialog Box Variables  **************************************************/
/*****************************************************************************/

// generic File I/O dialog box parameters (see fileopen.dlg, filelog.dlg)
#define     IDC_FILENAME    400   // current filename
#define     IDC_EDIT         401  // edit box
#define     IDC_FILES        402  // list of files
#define     IDC_PATH         403  // current path
#define     IDC_LISTBOX     404   // list box

// About Dialog Box Variables (see about.dlg)
#define IDD_MEM     100  // host system memory
#define IDD_RESOURCE         101  // host system resources
#define IDD_MODE        102  // current mode of host system
#define IDD_FRAME       103  // ems in use
#define IDD_8087        104  // co-processor present

// Generate & Plot Histogram Stats dialog box variables (see histogrm.dlg)
#define SCM_HIST_GRAPH       101   // area dedicated to histogram graph
#define SCM_DISP_RNG         102   // histogram graph display range
#define SCM_HIST_STRT        103   // histogram display range starting value
#define SCM_HIST_STRT_VS    104    // histogram starting value vertical scroll bar
#define SCM_HIST_END         105   // histogram display range ending value
#define SCM_HIST_END_VS     106    // histogram ending value vertical scroll bar
#define SCM_STAT_RNG         107   // histogram statistical calculation range
#define SCM_STAT_STRT        108   // statistical range starting value
#define SCM_STAT_STRT_VS    109    // statistical starting range vertical scroll bar
#define SCM_STAT_END         110   // statistical range ending value
#define SCM_STAT_END_VS     111    // statistical ending range vertical scroll bar
#define SCM_MEAN        112   // histogram's mean value
#define SCM_MEDIAN      113   // histogram's median value
#define SCM_MODE        114   // histogram's mode
#define SCM_MIN_VAL     115   // minimum value within histogram
#define SCM_MAX_VAL     116   // maximum value within histogram
#define SCM_RANGE       117   // histogram's range of values
#define SCM_HIST_COUNT       118   // histogram's count
#define SCM_HIST_STDDEV     119    // histogram's standard deviation
#define SCM_HIST_VAR         120   // histogram's variance
#define SCM_HIST_DATA_VALS 121     // listbox containing histogram's values
#define SCM_RECALCULATE     122    // re-calculate stats based on new range values
#define SCM_REGRAPH_HIST    123    // re-graph histogram based on new range values
#define SCM_HIST_SAB         124   // source acquire buffer
#define SCM_HIST_SDB         125   // source destination buffer
#define SCM_HIST_SOB         126   // source overlay buffer
#define SCM_HIST_SSB         127   // source auxiliary buffer
#define SCM_HIST_SBN         128   // source buffer number
#define SCM_HIST_SBN_VS     129    // source buffer number vertical scroll bar
#define SCM_HIST_SSC         130   // source buffer starting col
#define SCM_HIST_SSC_VS     131    // source buffer starting col vertical scroll bar
#define SCM_HIST_SSR         132   // source buffer starting row
```

```
#define SCM_HIST_SSR_VS    133    // source buffer starting row vertical scroll bar
#define SCM_HIST_SBW        134   // source buffer width
#define SCM_HIST_SBW_VS    135    // source buffer width vertical scroll bar
#define SCM_HIST_SBH        136   // source buffer hieght
#define SCM_HIST_SBH_VS    137    // source buffer hieght vertical scroll bar

// Sequentially acquire dialog box variables (see sequence.dlg)
#define SCM_NUMBER      101   // number of frames
#define SCM_NUMBER_VS       102   // number of frames vertical scroll bar
#define SCM_NTHFRAME        103   // every nth frame
#define SCM_NTHFRAME_VS    104    // every nth frame vertical scroll bar

// edit format memory dialog box variables (see editfm.dlg)
#define SCM_VFM     109   // vertical format memory
#define SCM_HFM     110   // horizontal format memory
#define SCM_SIG_EFM     112   // edit format memory signal
#define SCM_SEF     113   // edit format starting value
#define SCM_SEF_VS      114   // starting value vertical scroll bar
#define SCM_EEF     115   // edit format ending value
#define SCM_EEF_VS      116   // ending value vertical scroll bar
#define SCM_INC_EFM     117   // edit format incrementing value
#define SCM_INC_VS      118   // incrementing value vertical scroll bar

// Zoom, Pan, and Scroll dialog box variables (see zoom.dlg)
#define SCM_ZMF     101   // zoom factor
#define SCM_PNF     102   // pan factor
#define SCM_PNF_HS      103   // pan factor horizontal scroll bar
#define SCM_SLF     104   // scroll factor
#define SCM_SLF_VS      105   // scroll factor vertical scroll bar

// Tranparent Pixel Value
#define SCM_TPV     101   // transparent pixel value
#define SCM_TPV_VS      102   // transparent pixel value vertical scroll bar

// Input A/D dialog box variables (see inputad.dlg)
#define SCM_SDL     101   // sync detect level
#define SCM_INF     102   // chrominance notch filter
#define SCM_IGL     103   // gain factor
#define SCM_IZO     104   // zero offset
#define SCM_IZO_VS      105   // zero offset vertical scroll bar
#define SCM_ADR     106   // A2D reference
#define SCM_ADR_VS      107   // A2D reference vertical scroll bar

// format memory dialog box variables (see format.dlg)
#define SCM_HWS     102   // horizontal window start
#define SCM_HWS_VS      103   // horizontal window start vertical scroll bar
#define SCM_HWE     104   // horizontal window end
#define SCM_HWE_VS      105   // horizontal window end vertical scroll bar
#define SCM_HWI     106   // horizontal window increment
#define SCM_HWI_VS      107   // horizontal window increment vertical scroll bar
#define SCM_VWS     109   // vertical window start
#define SCM_VWS_VS      110   // vertical window start vertical scroll bar
#define SCM_VWE     111   // vertical window end
#define SCM_VWE_VS      112   // vertical window end vertical scroll bar
#define SCM_VWI     113   // vertical window increment
#define SCM_VWI_VS      114   // vertical window increment vertical scroll bar
#define SCM_HZS     116   // horizontal A2D zero start
```

```
#define SCM_HZS_VS      117   // horizontal A2D zero start vertical scroll bar
#define SCM_HZE     118   // horizontal A2D zero end
#define SCM_HZE_VS      119   // horizontal A2D zero end vertical scroll bar
#define SCM_HZI     120   // horizontal A2D zero increment
#define SCM_HZI_VS      121   // horizontal A2D zero increment vertical scroll bar
#define SCM_HCS     123   // horizontal A2D clamp start
#define SCM_HCS_VS      124   // horizontal A2D clamp start vertical scroll bar
#define SCM_HCE     125   // horizontal A2D clamp end
#define SCM_HCE_VS      126   // horizontal A2D clamp end vertical scroll bar
#define SCM_HCI     127   // horizontal A2D clamp increment
#define SCM_HCI_VS      128   // horizontal A2D clamp increment vertical scroll bar

// Allocate Onboard Auxiliary Buffer Dialog Box Variables (see aba.dlg)
#define SCM_NABS        90    // number of auxiliary buffers to create
#define SCM_NABS_VS     91    // number of buffers vertical scroll bar
#define SCM_ROWS        101   // hieght or row dimension
#define SCM_ROW_VS      102   // hieght or row dimension vertical scroll bar
#define SCM_COLS        103   // width or column dimension
#define SCM_COL_VS      104   // width or column dimension vertical scroll bar
#define SCM_PSIZE       105   // pixel size or depth
#define SCM_PSIZE_VS        106   // pixel size or depth vertical scroll bar

// acquire setup configuration variables (see acqset.dlg)
#define SCM_ASF     101   // acquire start field (next/odd/even...)
#define SCM_SAM     102   // acquisition mode
#define SCM_ICS     103   // invert clock signal
#define SCM_ILF     104   // invert line / frame signal

// RIO Source Variables (see acqroi.dlg, avg.dlg, filelog.dlg, fileopen.dlg, iscale.
dlg, roiac.dlg, rot90.dlg,
//                  slct_buf.dlg, edit_buf.dlg)
#define SCM_SAB     102   // source acquire buffer
#define SCM_SDB     103   // source destination buffer
#define SCM_SOB     104   // source overlay buffer
#define SCM_SSB     105   // source auxiliary buffer
#define SCM_SBN     106   // source buffer number
#define SCM_SBN_VS      107   // source buffer number vertical scroll bar
#define SCM_SSC     108   // source buffer starting col
#define SCM_SSC_VS      109   // source buffer starting col vertical scroll bar
#define SCM_SSR     110   // source buffer starting row
#define SCM_SSR_VS      111   // source buffer starting row vertical scroll bar
#define SCM_SBW     112   // source buffer width
#define SCM_SBW_VS      113   // source buffer width vertical scroll bar
#define SCM_SBH     114   // source buffer hieght
#define SCM_SBH_VS      115   // source buffer hieght vertical scroll bar

// RIO Destination Variables (see acqroi.dlg, avg.dlg, iscale.dlg, roiac.dlg, rot90.
dlg)
#define SCM_DAB     117   // destination acquire buffer
#define SCM_DDB     118   // destination destination buffer
#define SCM_DOB     119   // destination overlay buffer
#define SCM_DSB     120   // destination auxiliary buffer
#define SCM_DBN     121   // destination buffer number
#define SCM_DBN_VS      122   // destination buffer number vertical scroll bar
#define SCM_DSC     123   // destination starting col
#define SCM_DSC_VS      124   // destination starting col vertical scroll bar
#define SCM_DSR     125   // destination starting row
```

```
#define SCM_DSR_VS      126   // destination starting row vertical scroll bar
#define SCM_DBW     127   // destination width
#define SCM_DBW_VS      128   // destination width vertical scroll bar
#define SCM_DBH     129   // destination hieght
#define SCM_DBH_VS      130   // destination hieght vertical scroll bar
#define SCM_CW          131   // rotation in clockwise direction
#define SCM_CCW     132   // rotation in counter-clockwise direction

// Average frames true/recursive/post dialog box variables (see avg.dlg)
#define SCM_ENF     131   // every nth frame
#define SCM_ENF_VS      132   // every nth frame vertical scroll bar
#define SCM_NOF     133   // number of frames
#define SCM_NOF_VS      134   // number of frames vertical scroll bar
#define SCM_AWT     135   // frame wieght
#define SCM_AWT_VS      136   // frame wieght vertical scroll bar
#define SCM_UDD     137   // update display

//sequential image replay dialog box variables
#define SCM_NO_BUF      101   // number of buffers
#define SCM_NO_BUF_VS       102   // number of buffers vertical scroll bar
#define SCM_STRT_BUF        103   // starting buffer
#define SCM_STRT_BUF_VS    104    // starting buffer vertical scroll bar
#define SCM_DEL_FCT     105   // delay factor
#define SCM_DEL_FCT_VS      106   // delay factor vertical scroll bar

// Integer Scaling dialog box variables (see iscale.dlg)
#define SCM_VIEB        131   // vertically expand
#define SCM_VICB        132   // vertically compact
#define SCM_ED_VIEB     133   // vertical scale factor
#define SCM_VIEB_VS     134   // vertical scale factor vertical scroll bar
#define SCM_HIEB        135   // horizontally expand
#define SCM_HICB        136   // horizontally compact
#define SCM_ED_HIEB     137   // horizontal scale factor
#define SCM_HIEB_VS     138   // horizontal scale factor vertical scroll bar

// generic file format dialog box variables (see fileopen.dlg)
#define SCM_BIN     133   // straight Binary pixel data, no header, user must rememb
er dimensions of buffer
#define SCM_IRS     134   // IRIS file format, Data Translation Inc.
#define SCM_TIF     135   // tagged image file format (TIFF)
#define SCM_PCX     136   // PCX file format
#define SCM_GIF     137   // GIF file format
#define SCM_PCT     138   // PICT file format
#define SCM_TGA     139   // TARGA file format
#define SCM_DIB     140   // device independent bitmap
#define SCM_BUF_SEL     141   // buffer selected group box
#define SCM_FILE_FMT        142   // file format group box
#define SCM_XMPL_FMT        143   // example program format (camera setup)
#define SCM_GLI_FMT     144   // Global Lab Image file format (camera setup)

// Lookup Table dialog box variables (see luts.dlg)
#define SCM_LUT_SLCT        101   // lookup table selection
#define SCM_ILUT        102   // input lookup table
#define SCM_OLUT        103   // output lookup table
#define SCM_OVL_LUT     104   // overlay lookup table
#define SCM_FLUT        105   // lookup table functions
#define SCM_IDENTITY        106   // identity gray scale pattern
```

```
#define SCM_INVERSE     107   // invert current table
#define SCM_BINARY      108   // binary pattern 0-black, 255-white
#define SCM_ZEROED      109   // zero out
#define SCM_MANUAL      110   // manually adjust create specific color and set to de
sired index
#define SCM_ADC     111   // add a constant to existing values in table
#define SCM_SBC     112   // subtract a constant from existing values in table
#define SCM_MTC     113   // multiply existing values in table by a constant
#define SCM_DVC     114   // devide existing values in table by a constant
#define SCM_CONST_VAL       115   // set table to constant value within specified ra
nge
#define SCM_RGBA        116   // output lookup tables to modify: red/green/blue/or a
ll
#define SCM_RED     117   // output lookup table red values
#define SCM_GREEN       118   // output lookup table green values
#define SCM_BLUE        119   // output lookup table blue values
#define SCM_ALL     120   // all output lookup table color values
#define SCM_LUT_DATA        121   // lookup table ROI & Constant values
#define SCM_SETR        122   // set range flag
#define SCM_CONSTANT        123   // constant value
#define SCM_CONSTANT_VS    124    // constant value vertical scroll bar
#define SCM_STRT_INDX       125   // starting index
#define SCM_STRT_INDX_VS   126    // starting index vertical scroll bar
#define SCM_END_INDX        127   // ending index
#define SCM_END_INDX_VS    128    // ending index vertical scroll bar
#define SCM_RESET_LUTS      129   // reset luts flag to prestored identity values

// Convolution on SRC Buffer dialog box variables (see convo.dlg)
#define SCM_PRE_FIL         101   // predefined filters
#define SCM_LOW_P1          102   // low pass #1
#define SCM_HIGH_P1         103   // high pass #1
#define SCM_VERT_EDGE       104   // vertical edge
#define SCM_HORIZ_EDGE      105   // horizontal edge
#define SCM_LAP1            106   // laplacian #1

// generic ROI parameters used in various dialog boxes (see  convo.dlg)
#define SCM_SOURCE_ROW      152   // starting row in source buffer
#define SCM_SOURCE_COL      153   // starting col in source buffer
#define SCM_BUF_WIDTH       154   // starting width
#define SCM_BUF_HEIGHT      155   // starting height
#define SCM_DEST_ROW        156   // starting row in destination
#define SCM_DEST_COL        157   // starting col in destination

//Normalize options
#define SCM_SIGNED          180
#define SCM_OFFSET_SIGNED   181
#define SCM_ABS_VAL         182  // use absolute values
#define SCM_UNSIGNED        183
#define SCM_NONE         184

// generic parameters used in various IP dialog boxes (see  convo.dlg )
#define SCM_DIVISOR         190  // divisor
#define SCM_FILTERS         191

//Convolution Edge options
#define SCM_0           200
#define SCM_255         201
```

```
#define SCM_MEM         202


//Control bit radio button options
#define SCM_CB1_ON   107
#define SCM_CB1_OFF  108
#define SCM_CB2_ON   112
#define SCM_CB2_OFF  113
#define SCM_CB3_ON   117
#define SCM_CB3_OFF  118

//Edit Control bit radio button options
#define SCM_CB1      101
#define SCM_CB2      104
#define SCM_CB3      105
#define SCM_CB_ON    110
#define SCM_CB_OFF   116


// Wipe ROI dialog box variables (see wipe.dlg)
#define SCM_HCG       130  // color value to wipe buffer to

// Rubber-Sheeting algorithm dialog box variables (see rubber.dlg)
#define SCM_ULR      101   // upper left row
#define SCM_ULC      102   // upper left column
#define SCM_URR      103   // upper right row
#define SCM_URC      104   // upper right column
#define SCM_LLR      105   // lower left row
#define SCM_LLC      106   // lower left column
#define SCM_LRR      107   // lower right row
#define SCM_LRC      108   // lower right column

// all camera clock frequency dialog box variables (see clk_freq.dlg, motion.dlg)
#define SCM_EXT_SCAN         101   // external clock freq
#define SCM_EXT_VS       102   // external clock freq vertical scroll bar
#define SCM_VERT_SCAN        103   // vertical clock freq
#define SCM_VERT_VS      104   // vertical clock freq vertical scroll bar
#define SCM_HORIZ_SCAN       105   // horizontal clock freq
#define SCM_HORIZ_VS         106   // horizontal clock freq vertical scroll bar
#define SCM_INPUT_SCAN       107   // input clock freq
#define SCM_INPUT_VS         108   // input clock freq vertical scroll bar

// video input and sync channel dialog box variables (see chn_sync.dlg)
#define SCM_C0           102   // video channel 0
#define SCM_C1           103   // video channel 1
#define SCM_C2           104   // video channel 2
#define SCM_C3           105   // video channel 3
#define SCM_S0           107   // sync channel 0
#define SCM_S1           108   // sync channel 1
#define SCM_S2           109   // sync channel 2
#define SCM_S3           110   // sync channel 3

// Edit signal in synchronization format memory dialog box variables (see edsyncfm.d
lg)
#define SCM_ESFM_SIGNAL       114  // edit sync format mem - signal
#define SCM_ESFM_PERCENT      112  // edit sync format mem - percent
#define SCM_ESFM_PERCENT_VS 113  // edit sync vertical scroll bar
```

```
// synchronization format memory dialog box variables (see syncfm.dlg)
#define SCM_HSI      122  // sync format memory - horiz sync ins
#define SCM_HSI_VS       123  // sync format memory vertical scroll bar
#define SCM_HSRT         120  // sync format memory - horiz sync rst
#define SCM_HSR_VS       121  // sync format memory vertical scroll bar
#define SCM_VSI      118  // sync format memory - vert sync ins
#define SCM_VSI_VS       119  // sync format memory vertical scroll bar
#define SCM_VSR      116  // sync format memory - vert sync rst
#define SCM_VSR_VS       117  // sync format memory vertical scroll bar

// image acquisition parameters dialog box variables (see iacqset.dlg)
#define SCM_IAS_ASF      101  // image acq setup - start field
#define SCM_IAS_IAM      102  // image acq setup - image acq mode
#define SCM_IAS_IECS         103  // image acq setup - ext clock state
#define SCM_IAS_IL       122  // image acq setup - line sync invert
#define SCM_IAS_IF       121  // image acq setup - frame sync invert
#define SCM_IAS_CM       123  // image acq setup - clock mode

/*************************************************************************/
/***  Warning Flags  ***************************************************/
/*************************************************************************/
#define IDS_ERR_REGISTER_CLASS    1  // error registering class message
#define IDS_ERR_CREATE_WINDOW     2  // error creating window message
#define IDS_WARNING       5  // warning symbol
#define IDS_NO_MOUSE          6  // no mouse present message

/*************************************************************************/
/***  File input/output type desiginators ******************************/
/*************************************************************************/


#define DT_ID_BITMAP          4001
#define DT_ID_IMAGE_IRIS      4002
#define DT_ID_IMAGE_TIFF      4003
#define DT_ID_IMAGE_BINARY    4004
#define DT_ID_ALL_LUTS        4005
#define DT_ID_INPUT_LUT  4006
#define DT_ID_OUTPUT_LUT      4007
#define DT_ID_CAMERA_SETUP    4008
#define DT_ID_INPUT_SETUP     4009
#define DT_ID_DISPLAY_SETUP   4010
#define DT_ID_SEQ_ACQUIRE     4011
#define DT_ID_LOG_FILE        4012
#define DT_ID_ALL_FILES  4013
#define DT_GLI_ID_CAMERA_SETUP    4014
```

```
/*************************************************************************/
/*************************************************************************/
/***                                          ****************************/
/***  ARMM Windows Program                    ****************************/
/***                                          ****************************/
/***  Resource File -                         ****************************/
/***    Menu Format & Accellerator Keys Defined ***************************/
/***                                          ****************************/
/*************************************************************************/
/***  This contains the actual set up of the program menus.  Only the   ***/
/***    FILE and SW_EXTRAS menus have additions.  All of the others are ***/
/***    the same as Data Translation originally created.  The file manu ***/
/***    additions allow the user to save two different point arrays.    ***/
/***    The additions in the SW_EXTRAS menu are for the ARMM program's  ***/
/***    functionality.  They include controller initialization, crack   ***/
/***    sealing, line-snapping, and distortion adjustment, along with   ***/
/***    editing functions such as redrawing the lines on the screen and ***/
/***    an erase option for starting over.                              ***/
/*************************************************************************/

/*************************************************************************/
/***  Standaerd Include Files  *****************************************/
/*************************************************************************/
#include <windows.h>
#include <commdlg.h>
#include <dlgs.h>

/*************************************************************************/
/***  Include File for Menu ID Numbers  ********************************/
/*************************************************************************/
#include "51menus.h"

/*************************************************************************/
/***  Defined Icons  ***************************************************/
/*************************************************************************/
DT51 ICON BUGICON.ICO
BUGICON ICON BUGICON.ICO
RGBICON ICON RGBICON.ICO

/*************************************************************************/
/***  Defined Menu Structure  ******************************************/
/*************************************************************************/
DT51 MENU
{
/*************************************************************************/
/***  File Menu Option Routines  ***************************************/
/*************************************************************************/
    POPUP  "&File"
      {
    MENUITEM   "&Open Image...",                    IDM_OPEN              // Restore
Image File to Selected Buffer
    MENUITEM   "&Save Image as...",                 IDM_SAVE              // Save Ima
ge to File from selected buffer
        MENUITEM   SEPARATOR
        MENUITEM   "&Disable DT3851/52 Log \tCtrl+F1",  IDM_I_LOGOFF          // Turn
 DT3851/52 Response Log OFF
```

```
        MENUITEM    SEPARATOR
        MENUITEM    "Open Log File...",                 IDM_IO_LOG_READ       // Sav
e & Restore current DT3851/52 Log
        MENUITEM    "Save Log File as...",              IDM_IO_LOG_WRITE      // Sav
e & Restore current DT3851/52 Log
         MENUITEM    SEPARATOR
        MENUITEM    "Open Lookup Table File...",        IDM_IO_LUTS_READ     // Save
 & Restore Lookup Table data
        MENUITEM    "Save Lookup Table File as...",     IDM_IO_LUTS_WRITE    // Save
 & Restore Lookup Table data
        MENUITEM    SEPARATOR
        MENUITEM    "Open Camera Setup File...",        IDM_I_FILEFORMATMEM_READ  //
 Save & Restore current camera setup
        MENUITEM    "Save Camera Setup File as...",     IDM_I_FILEFORMATMEM_WRITE  /
/ Save & Restore current camera setup
        MENUITEM    SEPARATOR
        MENUITEM     "Save Array"                   IDM_ENT //Save an array of point
s
        MENUITEM     "Save 2nd Array"               IDM_SPONGE  //Save a 2nd array o
f points
        MENUITEM     SEPARATOR
    MENUITEM    "E&xit",                            IDM_EXIT              // Exit Pro
gram
      }
/*************************************************************************/
/***  Init Menu Option Routines  ***************************************/
/*************************************************************************/
    POPUP  "&Init"
      {
      MENUITEM "Reset &Hardware\tF7",             IDM_R_RESET_H      // Reset DT3851
/52 Hardware Only
      MENUITEM "Reset Hardware and &Disp\tF8",    IDM_R_HD_RESET       // Reset DT38
51/52 Hardware, establish default settings
      MENUITEM SEPARATOR
      MENUITEM "&Get DLL Config",                 IDM_I_CONFIGURATION  // Get config
uration of DT3851/52
      MENUITEM "Get TIGA &Config",                IDM_I_TIGA_GC      // Get current
TIGA configuration
      MENUITEM "Show GSP Heap",                   IDM_I_TIGA_GSP       // Get availa
ble heap space
      MENUITEM SEPARATOR
      MENUITEM "Get &Handles...",                 IDM_I_GETHANDLES           // Get
buffer handles
      MENUITEM "Report &Display Handle",          IDM_I_REPORTDISPLAYHANDLE  // Get
current display handle
      MENUITEM "Report &Buffer Info...",          IDM_I_REPORTBUFFERINFO   // Get bu
ffer info for specified handle
      }
/*************************************************************************/
/***  LUTs Menu Option Routines  ***************************************/
/*************************************************************************/
    POPUP  "&LUTs"
      {
        POPUP  "&Read Lookup Tables"
          {
             MENUITEM "Display &Input Lookup Table",       IDM_READ_ILUT     // Read
 Input Lookup Table and display graphically
```

```
            MENUITEM "Display &Output Lookup Table ",     IDM_READ_OLUTS     // Read
 Output Lookup Table and display graphically
            MENUITEM "Display O&verlay Lookup Table ",    IDM_READ_OVLUTS    // Read
 Overlay Lookup Table and display graphically
          }
        MENUITEM SEPARATOR
        MENUITEM   "Show &Lookup Table Log\tShift+F2",  IDM_I_LUTON          // Turn
 Lookup Table Log ON
        MENUITEM SEPARATOR
        MENUITEM "&Modify Lookup Tables...",          IDM_WRITE_LUTS         // Manu
ally Adjust Input, Output, and Overlay  LUTs
      }
/*************************************************************************/
/***  VIDEO Menu Option Routines  ******************************************/
/*************************************************************************/
    POPUP  "&Video"
      {
        MENUITEM "&Format Memory...",                   IDM_I_FORMATMEMORY       //
Read/Edit entire format memory
        MENUITEM "&Edit Format Mem...",                 IDM_I_EDITFORMATMEM  // Edit
 specific format memory setups
        MENUITEM "&Sync Format Memory...",              IDM_I_SYNCFORMATMEM      //
Read/Edit synchronization format memory
        MENUITEM "E&dit Sync Format Mem...",            IDM_I_EDITSYNCFORMATMEM  //
Edit signal in synchronization format memory
        MENUITEM SEPARATOR
        POPUP  "&Adjust Frequencies"
          {
            MENUITEM "&All Frequencies...",             IDM_I_ALLCLKFREQ         //
Read/Edit all camera clock frequencies
            MENUITEM SEPARATOR
            MENUITEM "Ext &Clock Freq...",              IDM_I_EXTCLOCKFREQ   // Read
/Edit external clock frequency
            MENUITEM "&Horiz Scan Freq...",             IDM_I_HORIZSCANFREQ      //
Read/Edit horizontal scan frequency
            MENUITEM "&Vert Scan Freq...",              IDM_I_VERTSCANFREQ       //
Read/Edit vertical scan frequency
            MENUITEM "&Input Clock Freq...",            IDM_I_INPUTFREQ          //
Read/Edit input scan frequency
          }
        MENUITEM "&Input A/D...",                       IDM_I_GAINOFFSET         //
Read/Edit Input A/D parameters
        MENUITEM SEPARATOR
        MENUITEM "&Video Inputs...",                    IDM_I_VIDEOINPUTSYNC     //
Read/Edit video input and sync channels
      }
/*************************************************************************/
/***  DISPLAY Menu Option Routines  ****************************************/
/*************************************************************************/
    POPUP  "&Display"
      {
        MENUITEM "&Disable Disp\tF2",                   IDM_D_DISABLE       // Disab
le display
        MENUITEM SEPARATOR
        MENUITEM "Display &Clock Freq...",              IDM_D_CLOCK         // Read/
Edit display clock frequency
        MENUITEM SEPARATOR
```

```
        MENUITEM "&Zoom/Pan/Scroll...",                IDM_D_ZPSM          // Read/
Edit displays Zoom, Pan, and Scroll states
        MENUITEM SEPARATOR
        MENUITEM "Enable &VGA Passthru\tF3",           IDM_D_VGAPASS_EN    // Enabl
e VGA passthru
        MENUITEM SEPARATOR
        MENUITEM "Enable &Overlay Plane\tF5",          IDM_O_ENAB_OVERLAY  // Enabl
e Overlay
        MENUITEM "&Transparent Pixel Value...",        IDM_O_PASSINDEX     // Set t
ransparent pixel value
      }
/*************************************************************************/
/***  ACQUIRE Menu Option Routines  ****************************************/
/*************************************************************************/
    POPUP  "&Acquire"
      {
        MENUITEM "&Image Acquire SetUp...",            IDM_A_IMGACQUIRESETUP
// Read/Edit image acquisition parameters
        MENUITEM SEPARATOR
        MENUITEM "&Freeze Frame\tF9",                  IDM_C_FREEZEFRAME
// Stop passthru or terminate FOCUS mode
        MENUITEM "&Check Acquire Complete",            IDM_A_CHECKACQUIRECOMPLETE
// Check for a completed acquisition
        MENUITEM SEPARATOR
        MENUITEM "&Enable Ext. Trig\tIns",             IDM_A_ENABLE_TRIGGER
// Acquire using externally triggered input
        MENUITEM SEPARATOR
        POPUP  "&Acquisitions"
          {
            MENUITEM "&Acquire Full Frame...",         IDM_A_ACQUIREFULLFRAME
// Acquire full frame
            MENUITEM "&Passthru Full Frame...",        IDM_A_PASSTHRUFULLFRAME
// Acuire live full frame images
            MENUITEM SEPARATOR
            MENUITEM "Acquire &ROI...",                IDM_A_ACQUIREROI
// Acquire region of interest
            MENUITEM "Passthru R&OI...",               IDM_A_PASSTHRUROI
// Acquire live region of image
            MENUITEM SEPARATOR
            POPUP "Acquire Avera&ge"
        {
        MENUITEM "Acquire & True &Average...",         IDM_A_ACQUIRETAVG   // True fr
ame average
        MENUITEM "Acquire & &Post True Average...",    IDM_A_ACQUIREPTAVG  // Post fr
ame average
                MENUITEM SEPARATOR
        MENUITEM "Acquire & &Recursive Average...",    IDM_A_ACQUIRERAVG   // True re
cursive frame averaging
        MENUITEM "Acquire & P&ost Recursive Average...", IDM_A_ACQUIREPRAVG  // Pos
t recursive frame averaging
        }
            MENUITEM SEPARATOR
            POPUP  "&Mini-Movie"
              {
               MENUITEM "&Acquire Sequence...",          IDM_A_ACQUIREMANY
  // Sequentially acquire a series of images
               MENUITEM SEPARATOR
```

```
                MENUITEM "&Play Sequence...",              IDM_A_REPLAYMANY, GRAYED
  // Replay sequentially acquired series
                MENUITEM "&Loop Sequence...",              IDM_A_LOOPREPLAY, GRAYED
  // Continually Loop-Thru acquired series
            }
              MENUITEM SEPARATOR
              MENUITEM "Four &Camera Acq/Disp...",       IDM_A_FOURFULLFRAME
  // Acquire from 4 video inputs in series
           }
      }
/*************************************************************************/
/***  AUXMEM Menu Option Routines  *****************************************/
/*************************************************************************/
    POPUP  "Aux&Mem"
      {
        MENUITEM "&Allocate AUX Buf...",                 IDM_I_ALLOCATEAUXBUF
 // Allocates onboard auxiliary buffer
        MENUITEM "&Free AUX Buf...",                     IDM_I_FREEAUXBUF, GRAYED
 // Free specified auxiliary buffer
        MENUITEM "&Free Sequence of AUX Bufs...",        IDM_A_DELETEMANY, GRAYED
 // Free a series of auxiliary buffers
        MENUITEM "&Free ALL AUX Buffers",                IDM_I_DELETEALLBUFS, GRAYED
 // Free all auxiliary buffers
      }
/*************************************************************************/
/***  SW_EXTRAS Menu Option Routines  **************************************/
/*************************************************************************/
    POPUP  "&SW_Extra"
      {
        MENUITEM "&Memory Copy...",                  IDM_A_MEMORYCOPY    // Copy a bu
ffer's contents to another buffer
        MENUITEM "Memory Copy with &Pixel operation...", IDM_A_MEMORYCOPYPIXELOP  //
 perform pixel operation between 2 buffers
        MENUITEM SEPARATOR
        MENUITEM "&Histogram...",              IDM_HISTO             // Generate & Plo
t Histogram
        MENUITEM "&Convolution...",            IDM_CONVOLUTION  // Performs a Convolu
tion on SRC Buffer
        MENUITEM SEPARATOR
        MENUITEM "&Scale Image...",          IDM_A_ISCALE       // Perform Integer Sc
aling on image data
        MENUITEM "&Rotate Image 90...",      IDM_A_ROTATE_90    // 90 Degree Rotation
 on image
        MENUITEM SEPARATOR
        MENUITEM "Refresh Drawn Lines"       IDM_REFRESH  //Refresh the Screen
        MENUITEM SEPARATOR
        MENUITEM "Auto. Line Snapping"       IDM_LINESNAP  //Perform line-Snapping
        MENUITEM "Adjust The Points"         IDM_ADJUSTPOINT  //Adjust points for dis
tortion
        MENUITEM SEPARATOR
        MENUITEM "Initialize Controller"     IDM_INITCONTROLLER  //Initialize aerotec
h controller
        MENUITEM "Trace Crack"               IDM_TRACECRACK  //Aerotech controller tr
acves crack
        MENUITEM SEPARATOR
        MENUITEM "Erase and Start Over"      IDM_ERASE   //start over again
      }
```

```
/*****************************************************************************/
/***  HELP Menu Option Routiness  ********************************************/
/*****************************************************************************/
/* About Box routine derived from Martin Heller's Advanced Windows Programming */
    POPUP  "\a&Help"
      {
    MENUITEM    "&About...",                    IDM_ABOUT  // About program & author
      }
}

/*****************************************************************************/
/***  Accelerator Keys  ******************************************************/
/*****************************************************************************/
DT51 ACCELERATORS
 {
  VK_F1,      IDM_D_ENABLE,              VIRTKEY       // Enable display
  VK_F2,      IDM_D_DISABLE,             VIRTKEY       // Disable display
  VK_F3,      IDM_D_VGAPASS_EN,          VIRTKEY          // Enable VGA passthru
  VK_F4,      IDM_D_VGAPASS_DIS,         VIRTKEY       // Disable VGA passthru
  VK_F5,      IDM_O_ENAB_OVERLAY,        VIRTKEY          // Enable Overlay
  VK_F6,      IDM_O_DISAB_OVERLAY,       VIRTKEY       // Disable Overlay
  VK_F7,      IDM_R_RESET_H,             VIRTKEY       // Reset DT3851/52 Hardware Onl
y
  VK_F8,      IDM_R_HD_RESET,            VIRTKEY       // Reset DT3851/52 Hardware, es
tablish default settings
  VK_F9,      IDM_C_FREEZEFRAME,         VIRTKEY       // Stop passthru or terminate F
OCUS mode
  VK_F1,      IDM_I_LOGON,               VIRTKEY, SHIFT    // Turn DT3851/52 Response
Log ON
  VK_F1,      IDM_I_LOGOFF,              VIRTKEY, CONTROL  // Turn DT3851/52 Response
Log OFF
  VK_F2,      IDM_I_LUTON,               VIRTKEY, SHIFT    // Turn Lookup Table Log ON
  VK_F2,      IDM_I_LUTOFF,              VIRTKEY, CONTROL  // Turn Lookup Table Log OF
F
  VK_INSERT, IDM_A_ENABLE_TRIGGER,       VIRTKEY           // Acquire using externally
 triggered input
  VK_DELETE, IDM_A_DISABLE_TRIGGER,      VIRTKEY           // Disable triggered input
}

/*****************************************************************************/
/***  Dialog Boxes to Include  ***********************************************/
/*****************************************************************************/
#include "FILEOPEN.DLG"  // Restore Image File to Selected Buffer
#include "FILELOG.DLG"   // DT3851/52 Log/LUTs/Camera Setup/and Sequential series I/
O
#include "SLCT_BUF.DLG"  // Get buffer handles for: Acquire/Display/Overlay/Auxiliar
y
#include "EDIT_BUF.DLG"  // Get buffer info for specified handle
#include "LUTS.DLG"      // Manually Adjust Input, Output, and Overlay Lookup Tables
#include "FORMAT.DLG"    // Read/Edit entire format memory
#include "EDITFM.DLG"    // Edit specific format memory setups
#include "SYNCFM.DLG"    // Read/Edit synchronization format memory
#include "EDSYNCFM.DLG"  // Edit signal in synchronization format memory
#include "CLK_FREQ.DLG"  // Read/Edit all camera clock frequencies
#include "DCLOCK.DLG"    // Read/Edit ext/hor/vert/input/disp clock frequency
#include "INPUTAD.DLG"   // Read/Edit Input A/D parameters
#include "CHN_SYNC.DLG"  // Read/Edit video input and sync channels
```

```
#include "ZOOM.DLG"       // Read/Edit displays Zoom, Pan, and Scroll states
#include "NVALUE.DLG"     // Set transparent pixel value
#include "ACQSET.DLG"     // Read/Edit acquire setup configuration
#include "IACQSET.DLG"    // Read/Edit image acquisition parameters
#include "FRAMES.DLG"     // Number of frames
#include "ACQROI.DLG"     // Acquire region of interest
#include "AVG.DLG"        // Average frames true/recursive/post
#include "SEQUENCE.DLG"   // Sequentially acquire a series of images
#include "MOTION.DLG"     // Replay sequentially acquired series
#include "ROIAC.DLG"      // Copy a buffer's contents to another buffer
#include "ROIACPOP.DLG"   // Perform pixel operations between 2 buffers
#include "ISCALE.DLG"     // Perform Integer Scaling on image data
#include "ROT90.DLG"      // 90 Degree Rotation on image
#include "ABA.DLG"    // Allocates onboard auxiliary buffer
#include "WIPE.DLG"   // Wipes Destination Buffer ROI to specified Color
#include "HISTOG.DLG"     // Obtain Histogram ROI
#include "HISTOGRM.DLG"   // Generate & Plot Histogram of Destination Buf
#include "CONVO.DLG"      // Performs an Convolution on SRC Buffer
#include "ABOUT.DLG"      // About Box routine

/*****************************************************************************/
/***  Error & Warning Message Stringtable  *********************************/
/*****************************************************************************/
STRINGTABLE LOADONCALL DISCARDABLE
BEGIN
  IDS_WARNING,              "Warning"
  IDS_NO_MOUSE,             "No Mouse found. This program runs more efficiently with
 mouse."
  IDS_ERR_CREATE_WINDOW,    "Window creation failed!"
  IDS_ERR_REGISTER_CLASS,   "Error registering window class"
END
```

```
# Microsoft Visual C++ generated build script - Do not modify

PROJ = DT51J
DEBUG = 1
PROGTYPE = 0
CALLER =
ARGS =
DLLS =
D_RCDEFINES = /d_DEBUG
R_RCDEFINES = /dNDEBUG
ORIGIN = MSVC
ORIGIN_VER = 1.00
PROJPATH = C:\DTI\DT3851\EXAMPLES\WIN\
USEMFC = 0
CC = cl
CPP = cl
CXX = cl
CCREATEPCHFLAG =
CPPCREATEPCHFLAG =
CUSEPCHFLAG =
CPPUSEPCHFLAG =
FIRSTC = ACQUIRE.C
FIRSTCPP =
RC = rc
CFLAGS_D_WEXE = /nologo /Gs /G2 /Zp1 /W3 /Zi /AL /Gx- /Od /D "_DEBUG" /I "c:\msvc\in
clude" /I "c:\dti\include" /Gw /Fd"DT51J.PDB"
CFLAGS_R_WEXE = /nologo /Zp1 /W3 /AL /Gx- /D "NDEBUG" /FR /Gw /G3
LFLAGS_D_WEXE = /NOD /NOE /PACKC:61440 /ALIGN:16 /ONERROR:NOEXE /CO /M:FULL
LFLAGS_R_WEXE = /NOLOGO /NOD /PACKC:61440 /ALIGN:16 /ONERROR:NOEXE
LIBS_D_WEXE = libw llibcew win_ai dt51dll oldnames commdlg.lib toolhelp.lib
LIBS_R_WEXE = oldnames libw llibcew win_ai dt51dll commdlg.lib shell.lib toolhelp.li
b
RCFLAGS =
RESFLAGS = /30 /k /k
RUNFLAGS =
DEFFILE = DT51.DEF
OBJS_EXT =
LIBS_EXT = ..\..\LIB\DT51DLL.LIB ..\..\LIB\WIN_AI.LIB ..\..\..\..\U500\WIN_DLL\WINAE
R.LIB
!if "$(DEBUG)" == "1"
CFLAGS = $(CFLAGS_D_WEXE)
LFLAGS = $(LFLAGS_D_WEXE)
LIBS = $(LIBS_D_WEXE)
MAPFILE = nul
RCDEFINES = $(D_RCDEFINES)
!else
CFLAGS = $(CFLAGS_R_WEXE)
LFLAGS = $(LFLAGS_R_WEXE)
LIBS = $(LIBS_R_WEXE)
MAPFILE = nul
RCDEFINES = $(R_RCDEFINES)
!endif
!if [if exist MSVC.BND del MSVC.BND]
!endif
SBRS = DISPLAY.SBR \
        FILE.SBR \
```

```
        HELP.SBR \
        INIT.SBR \
        LUTS.SBR \
        SWEXTRA.SBR \
        UTILS.SBR \
        VIDEO.SBR \
        PATHPLN2.SBR \
        ACQUIRE2.SBR \
        CF3.SBR \
        LINESNAP.SBR \
        AEROTCH2.SBR \
        AUXMEM2.SBR


DT51DLL_DEP =

WIN_AI_DEP =

WINAER_DEP =

DISPLAY_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


FILE_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


HELP_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


INIT_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
```

```
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


LUTS_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


SWEXTRA_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


UTILS_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


VIDEO_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


DT51_RCDEP = c:\dti\dt3851\examples\win\51menus.h


PATHPLN2_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
```

```
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


ACQUIRE2_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


CF3_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


LINESNAP_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


AEROTCH2_DEP = c:\u500\lib\c\u500.h \
    c:\u500\win_dll\winaer.h


AUXMEM2_DEP = c:\dti\dt3851\examples\win\dt51.h \
    c:\dti\include\dtityp.h \
    c:\dti\include\id_info.h \
    c:\dti\dt3851\fwdefs.h \
    c:\dti\dt3851\fwstrcts.h \
    c:\dti\dt3851\tiga.h \
    c:\dti\dt3851\typedefs.h \
    c:\dti\dt3851\dt51dll.h \
    c:\dti\dt3851\examples\win\51menus.h


all:    $(PROJ).EXE
```

```
DISPLAY.OBJ:    DISPLAY.C $(DISPLAY_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c DISPLAY.C

FILE.OBJ:   FILE.C $(FILE_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c FILE.C

HELP.OBJ:   HELP.C $(HELP_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c HELP.C

INIT.OBJ:   INIT.C $(INIT_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c INIT.C

LUTS.OBJ:   LUTS.C $(LUTS_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c LUTS.C

SWEXTRA.OBJ:    SWEXTRA.C $(SWEXTRA_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c SWEXTRA.C

UTILS.OBJ:  UTILS.C $(UTILS_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c UTILS.C

VIDEO.OBJ:  VIDEO.C $(VIDEO_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c VIDEO.C

DT51.RES:   DT51.RC $(DT51_RCDEP)
    $(RC) $(RCFLAGS) $(RCDEFINES) -r DT51.RC

PATHPLN2.OBJ:   PATHPLN2.C $(PATHPLN2_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c PATHPLN2.C

ACQUIRE2.OBJ:   ACQUIRE2.C $(ACQUIRE2_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c ACQUIRE2.C

CF3.OBJ:    CF3.C $(CF3_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c CF3.C

LINESNAP.OBJ:   LINESNAP.C $(LINESNAP_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c LINESNAP.C

AEROTCH2.OBJ:   AEROTCH2.C $(AEROTCH2_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c AEROTCH2.C

AUXMEM2.OBJ:    AUXMEM2.C $(AUXMEM2_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c AUXMEM2.C


$(PROJ).EXE::   DT51.RES

$(PROJ).EXE::   DISPLAY.OBJ FILE.OBJ HELP.OBJ INIT.OBJ LUTS.OBJ SWEXTRA.OBJ UTILS.OB
J \
    VIDEO.OBJ PATHPLN2.OBJ ACQUIRE2.OBJ CF3.OBJ LINESNAP.OBJ AEROTCH2.OBJ AUXMEM2.OB
J $(OBJS_EXT) $(DEFFILE)
    echo >NUL @<<$(PROJ).CRF
DISPLAY.OBJ +
FILE.OBJ +
HELP.OBJ +
```

```
DISPLAY.OBJ:    DISPLAY.C $(DISPLAY_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c DISPLAY.C

FILE.OBJ:   FILE.C $(FILE_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c FILE.C

HELP.OBJ:   HELP.C $(HELP_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c HELP.C

INIT.OBJ:   INIT.C $(INIT_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c INIT.C

LUTS.OBJ:   LUTS.C $(LUTS_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c LUTS.C

SWEXTRA.OBJ:    SWEXTRA.C $(SWEXTRA_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c SWEXTRA.C

UTILS.OBJ:  UTILS.C $(UTILS_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c UTILS.C

VIDEO.OBJ:  VIDEO.C $(VIDEO_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c VIDEO.C

DT51.RES:   DT51.RC $(DT51_RCDEP)
    $(RC) $(RCFLAGS) $(RCDEFINES) -r DT51.RC

PATHPLN2.OBJ:   PATHPLN2.C $(PATHPLN2_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c PATHPLN2.C

ACQUIRE2.OBJ:   ACQUIRE2.C $(ACQUIRE2_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c ACQUIRE2.C

CF3.OBJ:    CF3.C $(CF3_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c CF3.C

LINESNAP.OBJ:   LINESNAP.C $(LINESNAP_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c LINESNAP.C

AEROTCH2.OBJ:   AEROTCH2.C $(AEROTCH2_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c AEROTCH2.C

AUXMEM2.OBJ:    AUXMEM2.C $(AUXMEM2_DEP)
    $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c AUXMEM2.C


$(PROJ).EXE::   DT51.RES

$(PROJ).EXE::   DISPLAY.OBJ FILE.OBJ HELP.OBJ INIT.OBJ LUTS.OBJ SWEXTRA.OBJ UTILS.OB
J \
    VIDEO.OBJ PATHPLN2.OBJ ACQUIRE2.OBJ CF3.OBJ LINESNAP.OBJ AEROTCH2.OBJ AUXMEM2.OB
J $(OBJS_EXT) $(DEFFILE)
    echo >NUL @<<$(PROJ).CRF
DISPLAY.OBJ +
FILE.OBJ +
HELP.OBJ +
```

```
INIT.OBJ +
LUTS.OBJ +
SWEXTRA.OBJ +
UTILS.OBJ +
VIDEO.OBJ +
PATHPLN2.OBJ +
ACQUIRE2.OBJ +
CF3.OBJ +
LINESNAP.OBJ +
AEROTCH2.OBJ +
AUXMEM2.OBJ +
$(OBJS_EXT)
$(PROJ).EXE
$(MAPFILE)
c:\msvc\lib\+
c:\msvc\mfc\lib\+
..\..\LIB\DT51DLL.LIB+
..\..\LIB\WIN_AI.LIB+
..\..\..\..\U500\WIN_DLL\WINAER.LIB+
$(LIBS)
$(DEFFILE);
<<
    link $(LFLAGS) @$(PROJ).CRF
    $(RC) $(RESFLAGS) DT51.RES $@
    @copy $(PROJ).CRF MSVC.BND

$(PROJ).EXE::   DT51.RES
    if not exist MSVC.BND   $(RC) $(RESFLAGS) DT51.RES $@

run: $(PROJ).EXE
    $(PROJ) $(RUNFLAGS)


$(PROJ).BSC: $(SBRS)
    bscmake @<<
/o$@ $(SBRS)
<<
```

```
NAME            DT51
EXETYPE     WINDOWS
CODE            PRELOAD MOVEABLE DISCARDABLE
DATA            PRELOAD FIXED MULTIPLE
SEGMENTS    FILE_TEXT   LOADONCALL MOVEABLE DISCARDABLE
                INIT_TEXT       LOADONCALL MOVEABLE DISCARDABLE
                LUTS_TEXT       LOADONCALL MOVEABLE DISCARDABLE
                SWEXTRA_TEXT    LOADONCALL MOVEABLE DISCARDABLE
                HELP_TEXT   LOADONCALL MOVEABLE DISCARDABLE
            UTILS_TEXT  LOADONCALL MOVEABLE DISCARDABLE
                _TEXT           PRELOAD MOVEABLE DISCARDABLE
                FILE_DATA   CLASS 'DATA' LOADONCALL FIXED DISCARDABLE
                INIT_DATA       CLASS 'DATA' LOADONCALL FIXED DISCARDABLE
                LUTS_DATA       CLASS 'DATA' LOADONCALL FIXED DISCARDABLE
                VIDEO_DATA      CLASS 'DATA' LOADONCALL FIXED DISCARDABLE
                DISPLAY_DATA    CLASS 'DATA' LOADONCALL FIXED DISCARDABLE
                ACQUIRE_DATA    CLASS 'DATA' LOADONCALL FIXED DISCARDABLE
                AUXMEM_DATA     CLASS 'DATA' LOADONCALL FIXED DISCARDABLE
                SWEXTRA_DATA    CLASS 'DATA' LOADONCALL FIXED DISCARDABLE
                HELP_DATA   CLASS 'DATA' LOADONCALL FIXED DISCARDABLE
                UTILS_DATA      CLASS 'DATA' LOADONCALL FIXED DISCARDABLE
                _DATA           CLASS 'DATA' PRELOAD FIXED DISCARDABLE
HEAPSIZE        1024
STACKSIZE       5120
EXPORTS         WndProc
                MenuWndProc
                LogWndProc
                ILGraphWndProc
                OLGraphWndProc
                OvLGraphWndProc
                LutWndProc
                OLutProc
                ScrollProc
                AboutMP
                Open_Save_Dlg
                Camera_Setup_Read_Save_Dlg
                LutsMP
                SelectBMP
                SelChnSync
                EditBMP
                DClockMP
                AllClocks
                NValueMP
                AbaMP
                ACQSETMP
                FormatMP
                RoiACMP
        RoiACPOPMP
                RoiIS
        AverageMP
                ConvoCM
                HistoCM
                ShowHistoCM
                AcqRoiMP
                InputADMP
                ZoomMP
```

```
        EdFormatMP
        NFramesMP
        NSequencesMP
ImgAcqSetupMP
SyncFormatMP
EdSyncFormatMP
        SeqMotionMP
        Rotate_90
```