SHRP-IDEA/UFR-90-003

A Transverse Profiler Using Laser-Beam-Carried Microwave-Interferometry

Dwight D. Egbert
M. Sami Fadali

Electrical Engineering/Computer Science Department
University of Nevada, Reno
Reno, NV 89557

Strategic Highway Research Program
National Research Council
Washington, D.C.

SHRP-IDEA/UFR-90-003
SHRP-87-ID006

December 1990

keyword(s)
cross-profile
interferometer
laser
microwave
transverse profiler

Strategic Highway Research Program
2101 Constitution Avenue, N.W.
Washington, D.C. 20418

(202) 334-3774

Acknowledgments

The research herein described was supported by the Strategic Highway Research Program (SHRP). SHRP is a unit of the National Research Council that was authorized by section 128 of the Surface Transportation and Uniform Relocation Assistance Act of 1987.

CONTENTS

Abstract

This report documents the design, construction and successful testing of a prototype transverse profiler based on the theory of laser-beam-carried microwave-interferometry (LBCMI). Implementation of such a profiler would provide a rapid, economical, reliable, and repeatable means of measuring pavement rutting. The measurement system may be carried in a vehicle operating at normal highway speeds. The basic parameter measured is the highway transverse profile.

Executive Summary

A roadway's topography influences motorist ride and safety. Features considered important for characterizing topography include cross slope variations, longitudinal profile, surface undulations or waves, depressions, holes, ruts, bumps, shoulder/pavement discontinuities, and dropoffs. A mobile and automated system capable of rapidly surveying a roadway's surface to quantify its topography accurately will be a useful measurement system for evaluating roadway conditions affecting motorist ride and safety. Previous transverse profile measurement procedures involved manual measurements, noncontact sensors, and 35mm photographs with a hairline projector light beam. The problem attendant with all these systems is that the transverse profile is sampled only at discrete intervals.

The basic concept of the laser-beam-carried microwave-interferometer (LBCMI) system consists of a vehicle-mounted scanning mirror which sweeps a laser beam transversely across the pavement lane as the vehicle proceeds. The road-surface height measurement is obtained by comparing the phase delay of the modulated wave reflected from the roadway with respect to a reference phase obtained at the laser source. This phase difference is proportional to the distance between the scanning mirror and the road surface at each point along the transverse scan.

The LBCMI system provides advantages over all of the current methods.

1. It can be operated in daylight at normal highway speeds.
2. It produces a smooth transverse profile of the roadway.
3. The instantaneous field-of-view of the laser determines the minimum horizontal resolution, and is adjustable.
4. The LBCMI output can be directly digitized and stored by an on-board computer in the survey vehicle.
5. The amount of data stored for analysis is under software control.
6. Partial data analysis and reduction can be accomplished in real time.

As a result of this project, a prototype LBCMI system has been designed and constructed, and laboratory scanning tests have been successfully performed using the

system. Several hundred field scans were produced with a variety of pavement surfaces. Many of these scans were duplicates from the same surface and averages were calculated to simulate the effect of increased signal to noise ratio to be expected with improved sensor technology. These scans demonstrate the ability of the LBCMI system to produce profile data from road surfaces. The target design specifications have not yet been fully met with the LBCMI laboratory prototype in field scanning tests.

The profile height resolution of the laboratory prototype system is between 5 and 10 mm, and the useable scan width is approximately 2 meters, though data are available for the full 4 meters. The fundamental limitation was the lack of a commercially available light detector with a combination of high speed and high sensitivity. This had forced the use of a maximum microwave modulation frequency of 300 MHz. This limitation has reduced the accuracy of the measurements. A recently announced detector would allow a modulation frequency of up to 500 MHz, which implies a resolution on the order of 1 mm.

The laboratory scanning experiments were performed at a scanning distance of approximately 1.5 meters with white poster board as the target surface. With this configuration, the LBCMI system produced sufficient reflected light to measure a full 90 degree scan. Also, the signal-to-noise ratio was sufficient to produce a properly decoded surface profile. However, when the system was tested in the field at its design height of 2.0 meters while scanning pavement surfaces, the reflected signal was only strong enough to produce reliable results for the center 2 meters of the 4-meter scan path. Likewise, because of the reduced signal-to-noise ratio, the decoding algorithm did not perform properly, and the apparent surface profile tends to be curved rather than flat. However, even with the overall surface distortion, specific surface perturbations are still visible and measurable on the decoded scans. The results of this research have proven that the laser-beam-carried microwave-interferometer theory will work and indicates that a highway profiler using it could be built with further development.

1.0 Introduction

This project has been supported by the SHRP-IDEA Program. It directly addresses several stated objectives of the Program. Specifically, an innovative method has been developed and proven feasible for measuring roadway topography. The objective has been to provide a rapid, economical, reliable, and repeatable means of measuring pavement rutting. The measurement system is capable of being carried in a vehicle operating at normal highway speeds. The basic parameter measured is the highway transverse profile.

The technique used is Laser-Beam-Carried Microwave Interferometry (LBCMI). The depth measurement is determined by the phase shift of the microwave-modulated laser beam reflected from the road surface. The location and size of each measurement is defined by the pavement area instantaneously illuminated by the transversely scanned laser beam. The transverse profile scanned is approximately 4 meters wide.

The transverse profile prototype which has been developed shows promise that such an instrument could be used for data acquisition in all four major areas of interest in the SHRP program: Asphalt, Pavement Performance, Highway Operations, and Concrete and Structures. It is based on technology which is new within highway practice, and offers the potential for significant improvement in both the efficiency and quality of data acquisition and analysis.

The present state-of-the-art in microwave-modulated lasers and microwave phase resolvers has provided the technological basis for the development of the LBCMI system. This technology allows operation in sunlight by using a low power laser with appropriate narrow band interference filters. Such lasers are safe, reliable and inexpensive to operate.

A preliminary analysis performed before beginning the project resulted in the following target design specifications for the LBCMI approach.

a) road surface height resolution	< 1 millimeter
b) road surface horizontal resolution	< 3 millimeters
c) transverse road width covered	> 4 meters
d) number of samples across road width	> 1,300 samples
e) longitudinal skew at 96 kph	< 100 millimeters

As a result of this project the LBCMI prototype system has been designed and constructed, and laboratory scanning tests have been successfully performed using the system. Consequently, it is believed that the above specifications could be met using the LBCMI technology together with newly developed sensor technology.

Several hundred field scans were produced with a variety of pavement surfaces. Many of these scans were duplicates from the same surface and averages were calculated to simulate the effect of increased signal to noise ratio. Examples of these scans are included in later sections of this report. Also, a demonstration diskette with a graphical display and decoding program and actual LBCMI scan data is included as part of this report. The example scans demonstrate the ability of the LBCMI system to produce profile data from road surfaces and show the various stages of signal decoding.

With the current LBCMI laboratory prototype the target design specifications have not yet been fully met in field scanning tests. However, with improvements in microwave modulation frequency and sensor sensitivity now available the design specifications could be met. As will be discussed in more detail in the Conclusions Section of this report a new high-speed detector with a refrigeration chamber to increase sensitivity has recently been announced by Hamamatsu Corporation which promises to provide the necessary speed and sensitivity.

The profile height resolution of the laboratory prototype system is between 5 and 10 millimeters, and the usable scan width is approximately 2 meters even though data are available for the full 4 meters. The fundamental limitation was the lack of a commercially available light detector with a combination of high speed and high sensitivity. The lack of a high speed detector had forced the use of a maximum microwave modulation frequency of 300 MHz. This limitation has reduced the accuracy of the measurements. The recently announced detector will allow a modulation frequency of up to 500 MHz, which implies a resolution of 1 mm.

The result of marginal detector sensitivity on the operation of the laboratory prototype LBCMI is a limit on the signal to noise ratio of the returned signal, which in turn reduces the height resolution. Also, as the scan angle increases to cover a larger width the reflected signal decreases in intensity which further reduces the signal to noise ratio. This has resulted in a reduced valid scan width in the original field tests. The new detector now available should allow a full 4 meter scan width.

The laboratory scanning experiments were performed at a scanning distance of approximately 1.5 meters with white posterboard as the target surface. With this configuration the LBCMI system produced sufficient reflected light to measure a full 90° scan. Also, the signal to noise ratio was sufficient to produce a properly decoded surface profile. However, when the system was tested in the field at its design height of 2.0 meters while scanning pavement surfaces the reflected signal was only strong enough to produce reliable results for the center 2 meters of the 4 meter scan path. Likewise, because of the reduced signal to noise ratio the decoding algorithm did not perform properly and the apparent surface profile tends to be curved rather than flat. However, even with the overall surface distortion specific surface perturbations are still visible and measurable on the decoded scans.

This research has been a cooperative effort between the University of Nevada, Reno (UNR) and Roadman PCES, Inc. of Sparks, Nevada. The primary research has been performed by UNR faculty and students with Roadman PCES acting as a subcontractor. Roadman PCES has provided facilities, hardware, and a wealth of experience in highway pavement evaluation, rehabilitation and maintenance.

This research has proven that the laser-beam-carried microwave-interferometer theory will work and a highway profiler using it is practical. The resolution of the laboratory prototype system constructed during this project can be increased as a result of recently announced advances in commercially available high-speed sensitive photo-detectors.

1.1 Background

A roadway's topography influences motorist ride and safety. Features considered important for characterizing topography include cross slope variations, longitudinal profile, surface undulations or waves, depressions, holes, ruts, bumps, shoulder/pavement discontinuities, and drop offs. A mobile and automated system capable of rapidly surveying a roadway's surface to quantify it's topography accurately will be a useful measurement system for evaluating roadway conditions affecting motorist ride and safety. [1][2]

Previous transverse profile measurement procedures involved; manual measurements, non-contact sensors, and 35mm photos with a "hairline projector" light beam. Manual methods require the occupancy of the pavement surface, and the process is dangerous and slow. [3]

Of the non-contact systems three are currently in use; laser, ultrasonic, and infrared. These systems use individual sensors arranged transversely over 8 to 12 feet. The smallest number of sensors used is three (wheel path-center lane-wheel path) and the highest number used is 17. The problem attendant with all these systems is that the transverse profile measured is sampled only at discrete intervals. [4]

The 35mm film, with hair-line projector, method has the advantage that a transverse profile is not discretely sampled but smoothly covers the full transverse width. However, with this method the data are currently manually reduced and the measuring process is limited to night operation.

The Laser-Beam-Carried Microwave Interferometer (LBCMI) system provides advantages over all of these current methods.

a) It can be operated in daylight at normal highway speeds.
b) It produces a smooth transverse profile of the roadway.
c) The instantaneous-field-of-view of the laser determines the minimum horizontal resolution, and is adjustable.
d) The LBCMI output can be directly digitized and stored by an on-board computer in the survey vehicle.
e) The amount of data stored for analysis is under software control.
f) Partial data analysis and reduction can be accomplished in real-time.

Because the depth measurement is obtained from the microwave modulation of the laser beam, low power lasers can be employed and a smaller signal to noise ratio can be tolerated than with existing laser techniques.

1.2 Theory of Measurement

The basic concept of the LBCMI system consists of a scanning mirror mounted on a vehicle which sweeps a laser beam transversely across the pavement lane as the vehicle proceeds. The laser used is a Gallium-Aluminum-Arsenide solid state laser which can easily be amplitude modulated at frequencies up to 500 MHz. Modulation frequencies between 400 and 500 megaHertz result in a wavelength on the order of 100 centimeters. Thus, with a phase difference resolution on the order of 1 degree a road surface topography can be obtained with a precision on the order of 1 millimeters. These concepts are shown schematically in Figure 1.

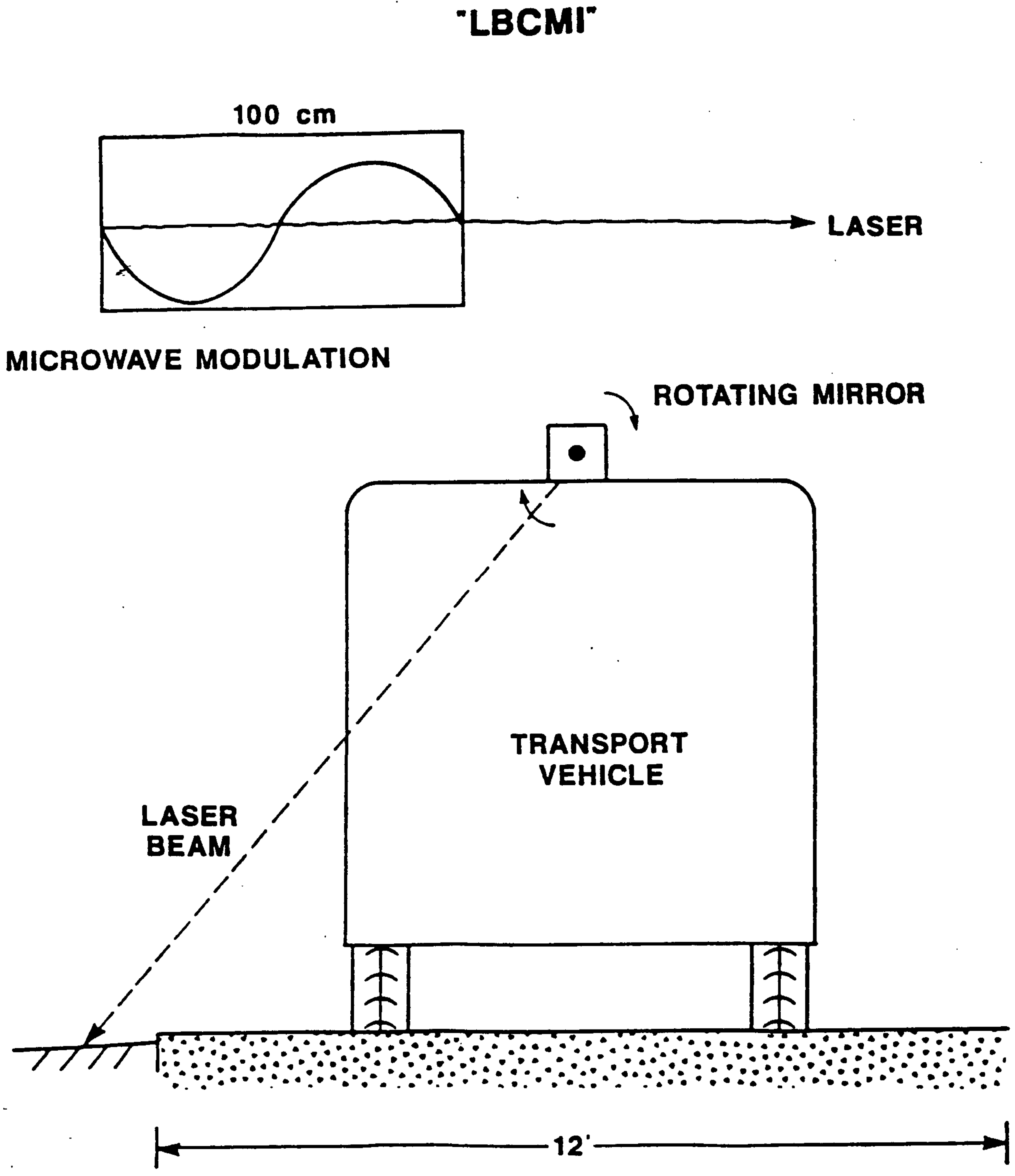


Figure 1. Basic concepts of the laser-beam-carried microwave-interferometer.

The laboratory prototype LBCMI system constructed during this project is shown in Figure 2. It is mounted on a transport device which can be moved along a longitudinal track to obtain multiple transverse profiles of sample pavement surfaces.

This laboratory prototype configuration can be compared to the schematic of the LBCMI shown in Figure 3. The road-surface height measurement is obtained by comparing the phase delay of the modulated wave reflected from the roadway with respect to a reference phase obtained at the laser source. The basic interferometer operation is shown in Figure 3. The beam splitter mirror at "A" extracts a small portion of the beam and sends it to the reference receiver where it is converted to an electrical reference signal in the 400 to 500 megaHertz frequency range.

The mirror at "B" routes the laser beam to the scanner mirror by which it is swept across the region of interest. Mirror "B" is a small mirror directly in front of the main receiver lens. This configuration simultaneously creates a scanning transmitter and a scanning receiver. Thus, the instantaneous-field-of-view (IFOV) of the the main receiver lens is always centered on the same pavement position that the main laser beam is illuminating.

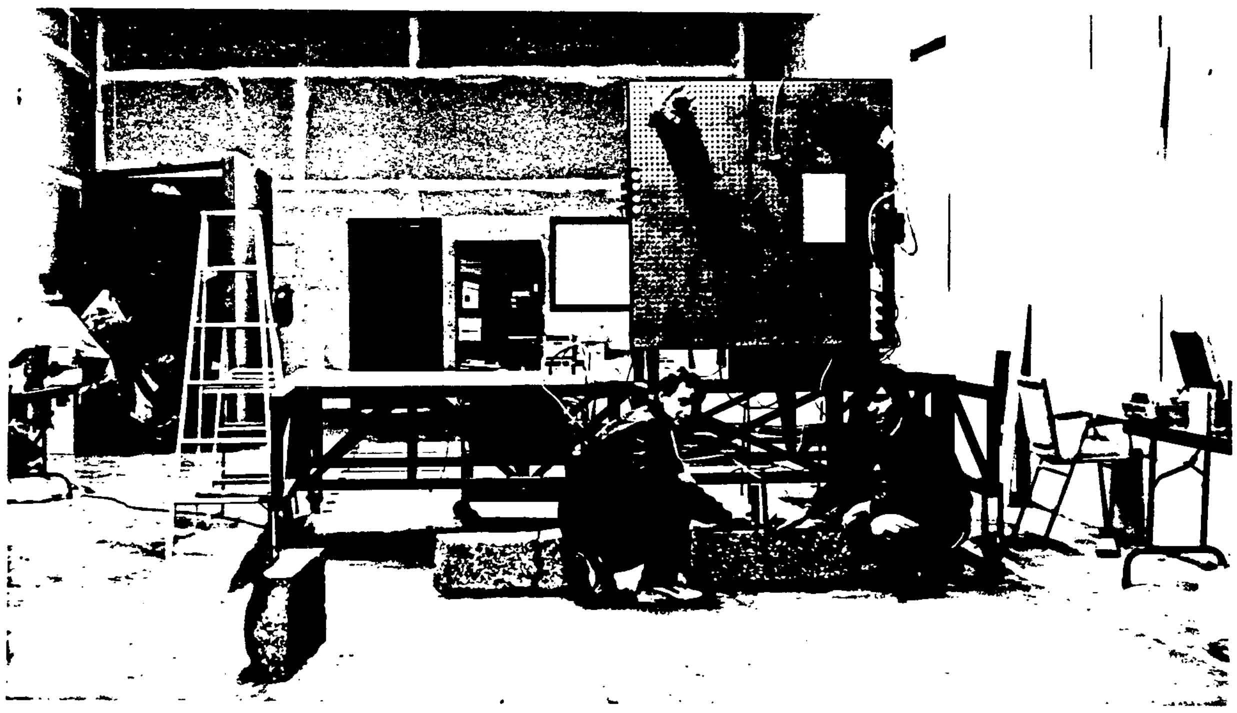


Figure 2. Laser-beam-carried microwave-interferometer on field test support.

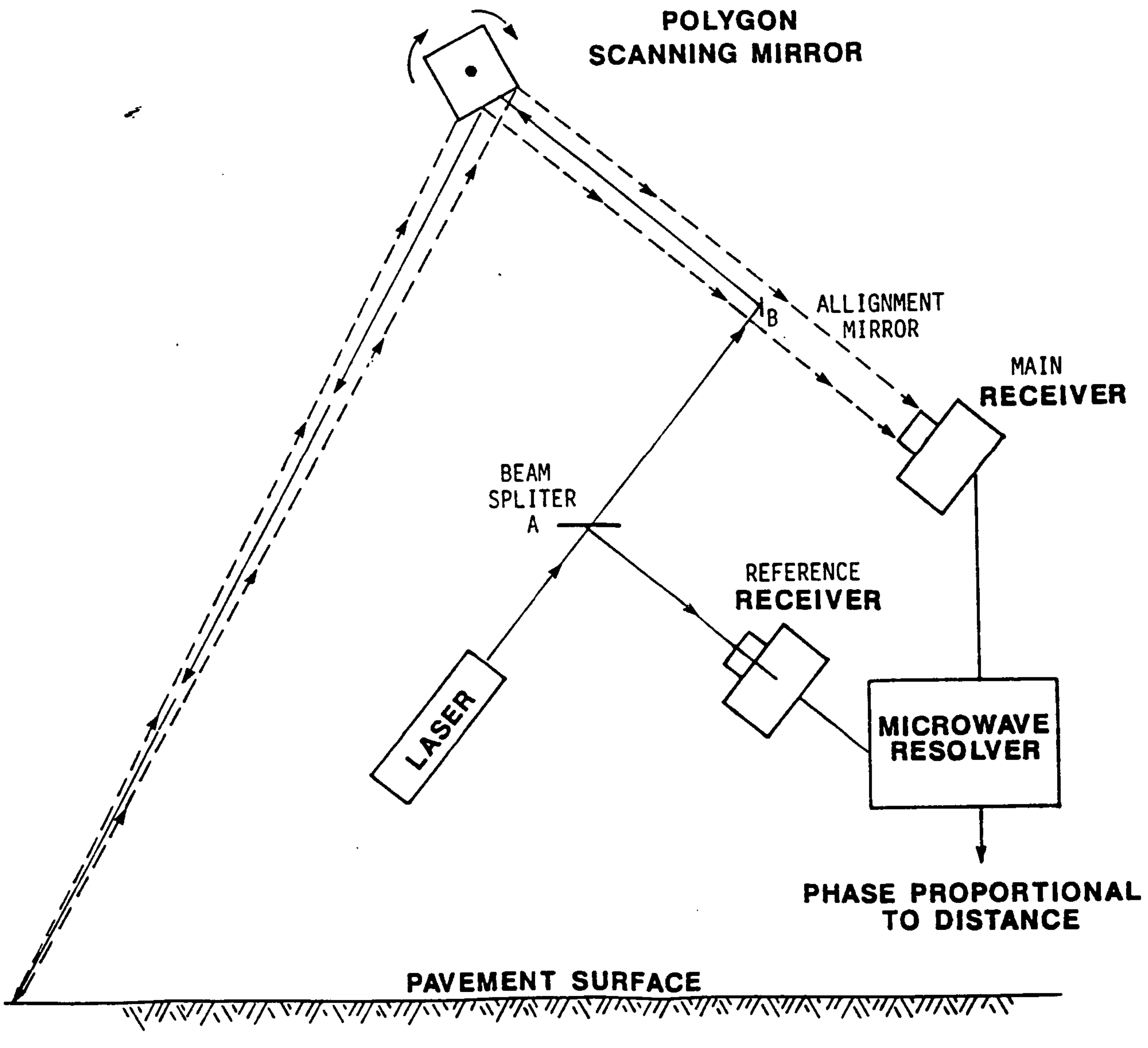


Figure 3. Schematic diagram of the laser-beam-carried microwave-interferometer.

This optical path from mirror "B" to the scanning mirror and then down to the pavement surface has two-way laser information. First, the main incident laser beam propagates toward the road surface, as shown by the solid ray in Figure 3. Second, the diffuse-reflected laser beam also returns from the road surface toward the main receiver over this path as shown by the dashed rays in Figure 3.

The main receiver detects the reflected laser beam and converts it to an electrical signal in the 400 to 500 megaHertz range. The outputs of the two receivers are then input to a microwave phase resolver which measures the phase difference between them. This phase difference is proportional to the distance between the scanning mirror and the road surface at each point along the transverse scan. This distance is shown as "d" in Figure 4a. This distance is a function of the angle Θ which is measured between the vertical and the instantaneous direction of the laser beam. The raw signal shown in Figure 4b is a combination of the transverse profile and this trigonometric scanning function.

The desired distance is the vertical distance between the scanning mirror and the road surface which is shown by "L" in Figure 4a. The relationship between "d" and "L" is simply: $L = d$ cosine(Θ). The profile shown in Figure 4c results from this simple trigonometric transformation, and represents the information to be recorded.

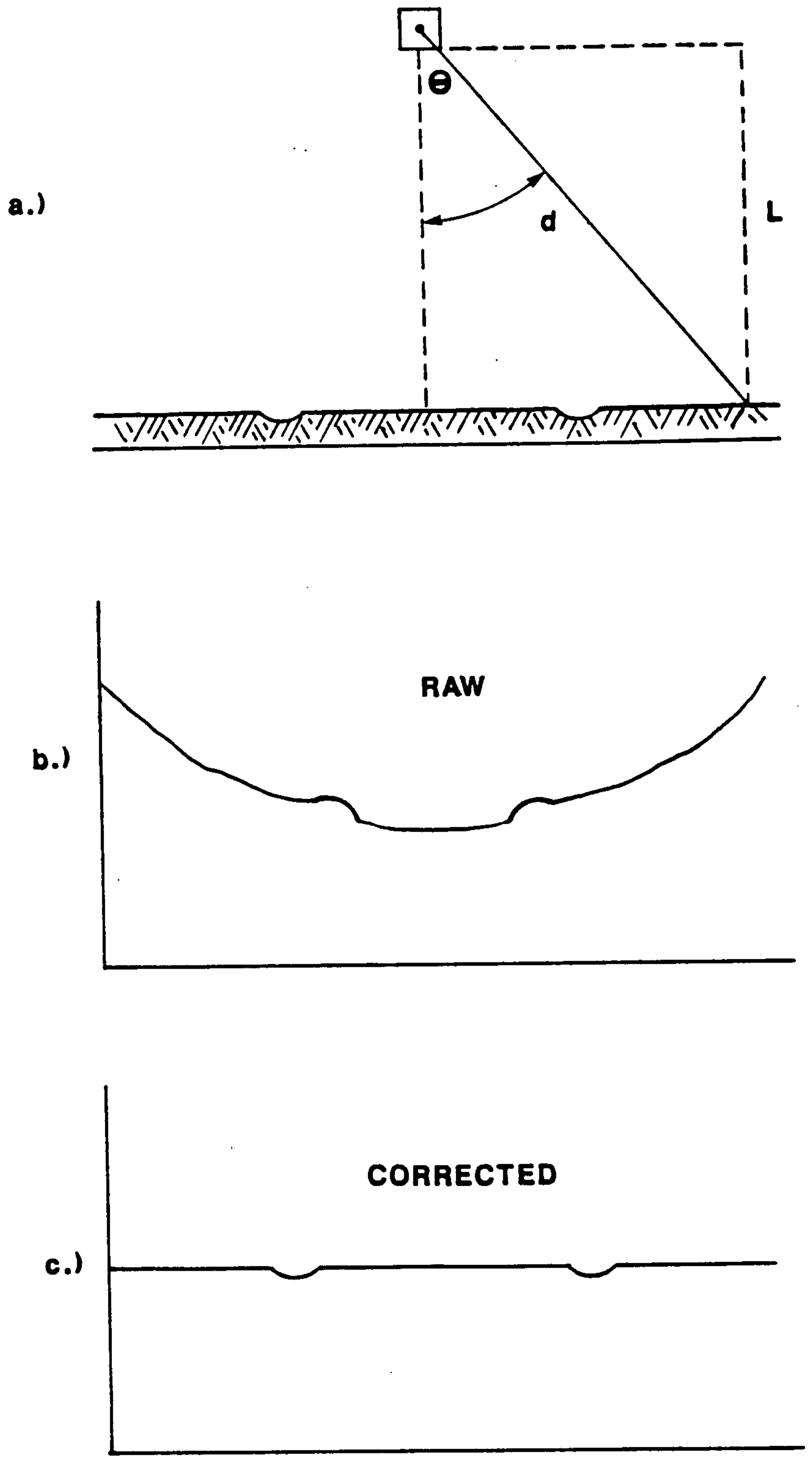


Figure 4. Data correction procedures for the laser-beam-carried microwave-interferometer. a) System Geometry, b) Raw Data Format, c) Corrected Data Format

This Page Left Blank Intentionally

2.0 LBCMI System Construction

The fundamental tenet that the LBCMI project is based upon is the concept that the electric field orientation of a propagating electromagnetic wave is a function of distance. It is a function of time as well, however, that aspect of wave nature will not be a factor in this application and therefore will not be considered. Different orientations of electromagnetic waves are called phases of waves. Any given frequency of an electromagnetic wave propagating in free space will change it's phase ϕ with distance in accordance with the following formula.

$$\phi = 2\pi / l \tag{2.1}$$

where l = c (speed of light) / f (frequency)

The phase detection circuit shown in Figure 5 is concerned essentially with decoding a differential phase from an electromagnetic wave which has been split into two signals and forced to travel different distances. The differential distance that the waves have to travel is the quantity to be measured.

The GaAlAs diode laser is the source of the electromagnetic waves. It emits coherent light at a wavelength of 750 nm with a total intensity of 4 mW (.004 Watts). A Varil VCO-105 voltage controlled oscillator supplies a +10 dBm (10 mW) signal at 300 MHz (out of a possible 200-400 MHz range) into the power supply of the laser forcing the laser to turn on and off at that frequency. This amplitude modulation (AM) is the critical frequency with which the phase detection will be accomplished. As mentioned previously, the use of a frequency as low as 300 MHz was dictated by the response time of the photomultiplier tube detector.

A glass beam splitter is placed in the beam of the laser output and deflects a small portion of the light into the Reference Detector. This detector is an MRD-721 photodiode detector. It is biased with a -15 V reversed biased voltage (cathode to +15V) and 50 ohm resistor to ground. The output of this detector is taken at the anode. It is sensitive to the 750 nm light and can respond quickly enough to respond to the modulation of the laser light. The output of the photodiode is a 300 MHz signal which is fed into the main circuit at the point labeled reference input in Figure 5.

The remaining laser output is reflected by mirrors shown in Figure 6 onto the road surface and then reflected ultimately back to the Response Detector. The response detector is a Hamamatsu R1547 photomultiplier tube (PMT). It has a spectral response of 16% of its peak response at 750 nm which is less than ideal. However, it has a current amplification of 1.3 million and has an anode sensitivity of 59 amps per milliwatt. Like the photodiode, it strips off the 300 MHz signal and feeds it into the circuit board at the input marked response input in Figure 5. The PMT requires a 1500 V power supply and the input light needs to be tightly filtered to eliminate extraneous light from overloading the PMT. The 1500 V-DC supply to the PMT is also filtered to reduce high frequency signals.

Both input channels to the circuit board are processed similarly as shown on the left side of Figure 5. The first stage of processing is at 300 MHz, referred to as the first intermediate frequency (IF) stage because it is less than the carrier frequency (Laser light) and more than the output frequency (Audio signal). Both channels have two MAR-6 amplifiers which each have approximately 16 dB of gain at 300 MHz. The 560 nH inductors are used to keep the 300 MHz signal isolated from the power supply. The Voltage Controlled Oscillator (VCO) is a VARIL-105 which is voltage controlled (0-20V yields 200-400 MHz) and supplies the local oscillator (LO) signal to the mixers (U3,U16). The local oscillator input of the mixers requires a +7 dBm signal and is set to a frequency which is the 1st IF + or - the 2nd IF. In this case the LO frequency is set to 310.7 MHz since the 2nd IF is 10.7 MHz. A Conversion loss of about 6 dB is suffered across the mixers.

The second IF strip is 10.7 MHz. The reference channel has about about 65 dB of gain on the second IF strip with about 50 dB of gain control (set by the operator to obtain optimal performance. The gain control is obtained with the Plessy 610c gain control amplifier. The gain of a Plessy 610c amplifier is 20 dB with the AGC pin of the 610c amplifier below 2.0 volts. When the AGC input of the 610c is between 2.0 and 5.0 volts the gain varies, approximately, linearly between +20 dB gain and -30 dB gain.

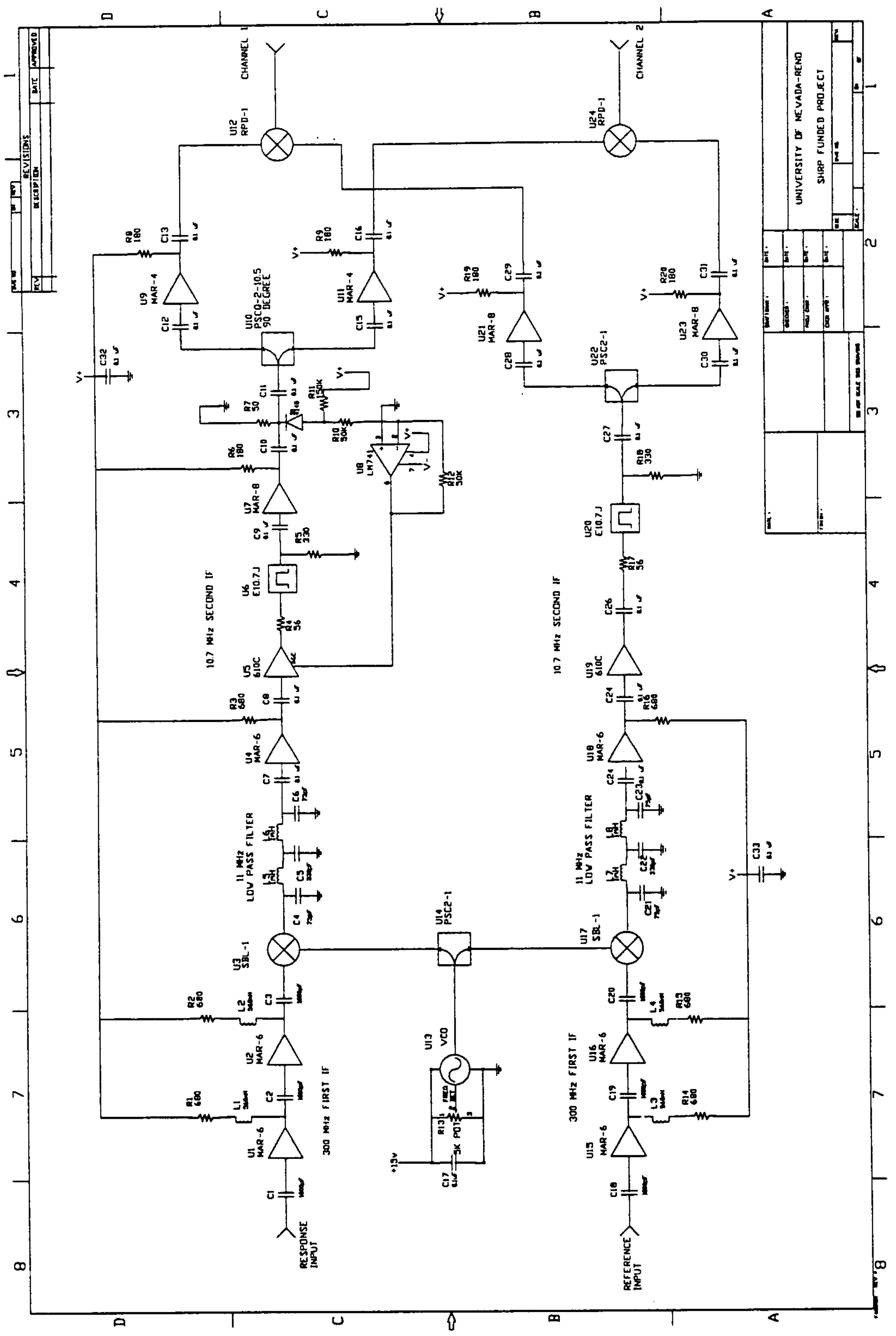


Figure 5. Microwave phase resolver circuit used in the LBCMI system.

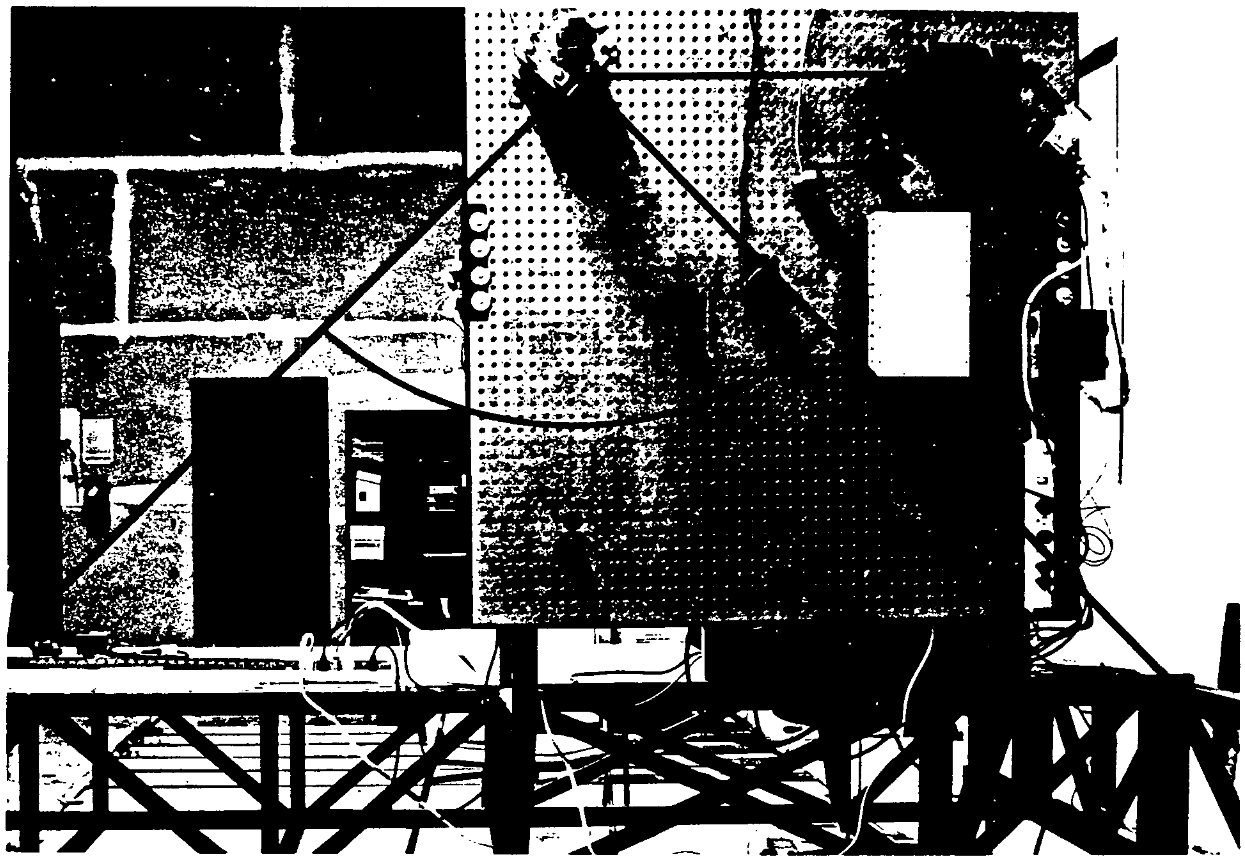


Figure 6. Laboratory Prototype LBCMI scanner showing laser beam path.

The gain of the all of the amplifiers on the circuit board is as follows:

amplifier	gain
mar-6	20 dB
mar-8	20 dB
mar-4	8 dB
610c	20 dB, 50 dB gain control

The response channel has approximately 80 dB of gain overall with 50 dB of gain control. The gain control is automatic and works as follows. The 1n4148 schottky diode detects the power level and outputs a larger negative voltage corresponding to increasingly higher input power levels. The diode output is fed into the lm741 op-amp which is configured in a inverting mode with a gain of approximately 100. This output is then fed into the AGC pin of the 610c. The resulting gain control curve is shown in Figure 7.

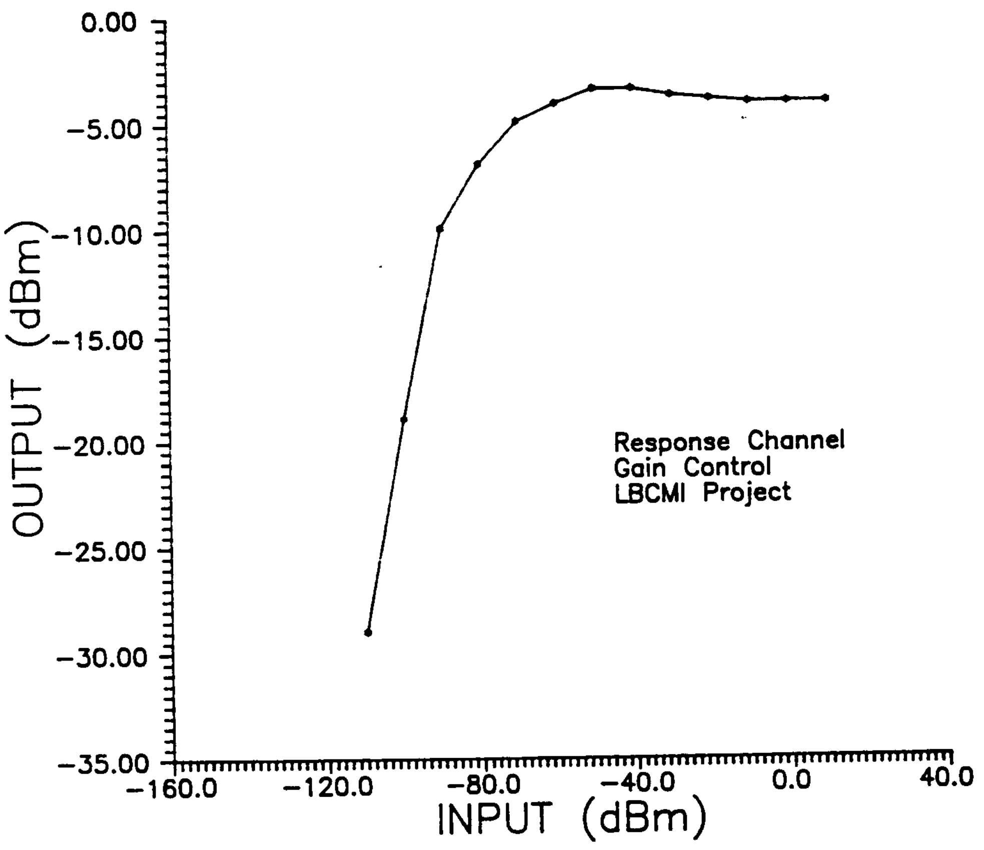


Figure 7. Automatic gain control response for LBCMI phase resolver.

A critical feature of any RF circuit is it's operative bandwidth, the smaller the bandwidth the smaller the noise power seen at the output. The bandwidth in this system is set by the two 11 MHz low pass filters between U3-U4 and U16-U17 and narrowband 10.7 MHz bandpass filters following U6 and U19. The resulting circuit 3 dB bandwidth is 150kHz.

In order to implement quadrature phase detection, both the reference and response channels are split. One splitter (PSC2-1) has zero phase difference between it's output signals and the other splitter (PSCQ-2-10.5) has 90° phase difference between it's output signals. Both splitters have about 4 dB of conversion loss.

Both splits of the response channel require 8 dB of amplification to obtain the proper power level into the phase detector chips which is accomplished with two MAR-4 amplifiers. The reference path of the circuit board requires 20 dB of gain after the splitter and therefore, MAR-8 amplifiers are used in these paths.

The LBCMI phase resolver circuit uses RPD-1 phase detector chips. These chips require input power levels of +7 dBm and output 12 mV per degree of phase difference in the linear range. The output of the phase detector chips is fed directly into the Metrabyte Dash-16 A/D convertor circuit board located in the IBM/AT compatible computer where it is converted into distance information.

The entire LBCMI amplifier and phase resolver circuit is contained on a single circuit board. Figure 8 shows this circuit board layout, and Figure 9 shows a photograph of the circuit board in operation.

The most persistent problem encountered with making this RF circuit work properly was signal isolation. Originally, whenever the laser diode was running and modulated at it's max level, the modulation frequency could be found on any and every circuit board in the same physical vicinity. Therefore, a great deal of time was spent isolating different portions of the circuit from one another. Finally, the laser diode, it's power supply, and the modulation circuitry was entirely enclosed in an aluminum box with only a small circular hole for the laser light to escape. Likewise, both detector circuits are built in enclosures constructed from double sided copper clad boards and have three pole, low-pass filters on their power supply lines. All interconnections of signals from the detectors to the phase detector circuit board are carried in shielded, SMA cables.

The main amplifier and phase resolver circuit board is also isolated in its own RF shielded case, and the power supplies for this board are filtered. Once on the circuit board, the bias voltage lines are radiated from a central point to avoid ground loop problems. Also, different parts of the circuit board are protected from each other with low pass filters.

Gain control is critical in a phase detection system. The output of a phase detector chip is AB $\cos(\Theta)$; where A and B are the input amplitudes into the phase detector chip and Θ is the phase difference. From this expression it is clear why quadrature phase detection is necessary. When Θ is near 0 or π $\cos(\Theta)$ is insensitive to small changes in Θ. It is also clear that the output of the phase detectors is equally sensitive to input amplitudes. Practically speaking, the gain control circuitry is good only to about 1 dB, mainly because the gain control circuitry is sensitive to changes in signal to noise ratios (as signal strength decreases, the S/N also decreases). Improvement is expected here with the addition of a more sensitive and noise free photomultiplier tube detector.

2.1 Mechanical Construction

The laboratory prototype LBCMI is constructed on a Melles Griot optical bench as shown in Figure 6. All components are attached to the optical bench using standard component mounts. The laser with power supply and 300 MHz modulation circuit are contained in the large aluminum box mounted on the right center part of the bench as shown in Figure 6. Just to the right of this box at the extreme right edge of the bench is the DC power supply for the modulation circuit.

The mount immediately above the laser is the beam splitter (labeled A in Figure 3) which diverts a small portion of the outgoing laser beam to the reference receiver which is just to the left of the beam splitter. The small mount just above the beam splitter is the alignment mirror (labeled B in Figure 3). To the far left of the alignment mirror is the polygon scanning mirror which provides the 90° sweep of the laser beam. This mirror is driven by a small DC motor which controls the variable scan speed.

As the reflected laser beam is intercepted by the polygon scanning mirror it is reflected to the right past the alignment mirror into the collecting lens of the main receiver. The collecting lens is in the upper right part of the bench as shown in Figure 6. The objective eyepiece of this lens is on the right side and pointing down and to the right at approximately 45°. This eyepiece focuses the returned light beam onto the photomultiplier tube which constitutes the main receiver.

The mount in the center of the bench just to the left of the laser is a trigger circuit which is not shown in Figure 3. This trigger contains a simple photodiode detector which produces a TTL logic pulse each time the scanning mirror sweeps the laser beam across it. The diode is positioned at 45° from the vertical with respect to the scanning mirror, and the output trigger pulse causes the computer to start collecting data for one scan.

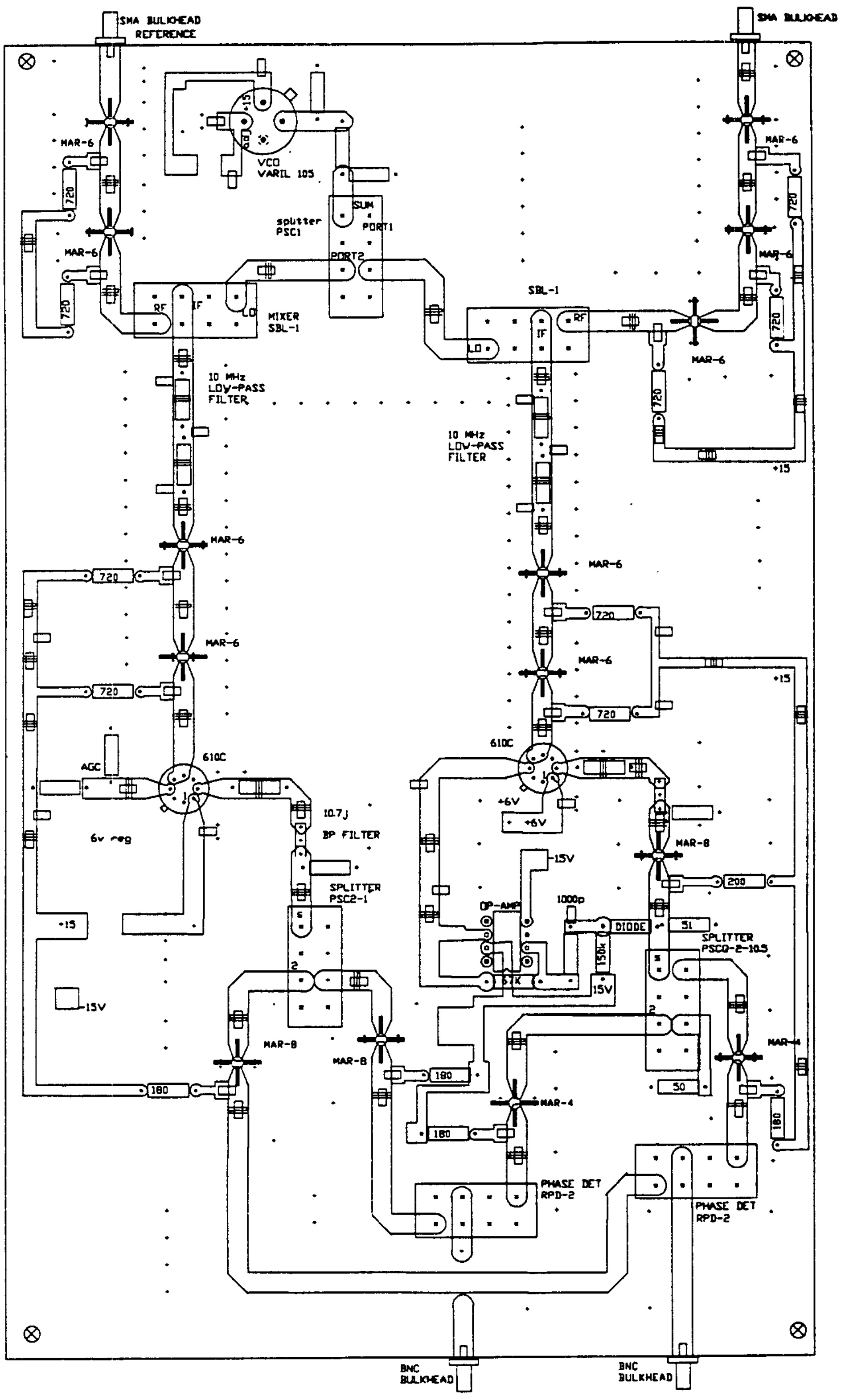


Figure 8. Diagram of LBCMI main amplifier and phase resolver circuit board.

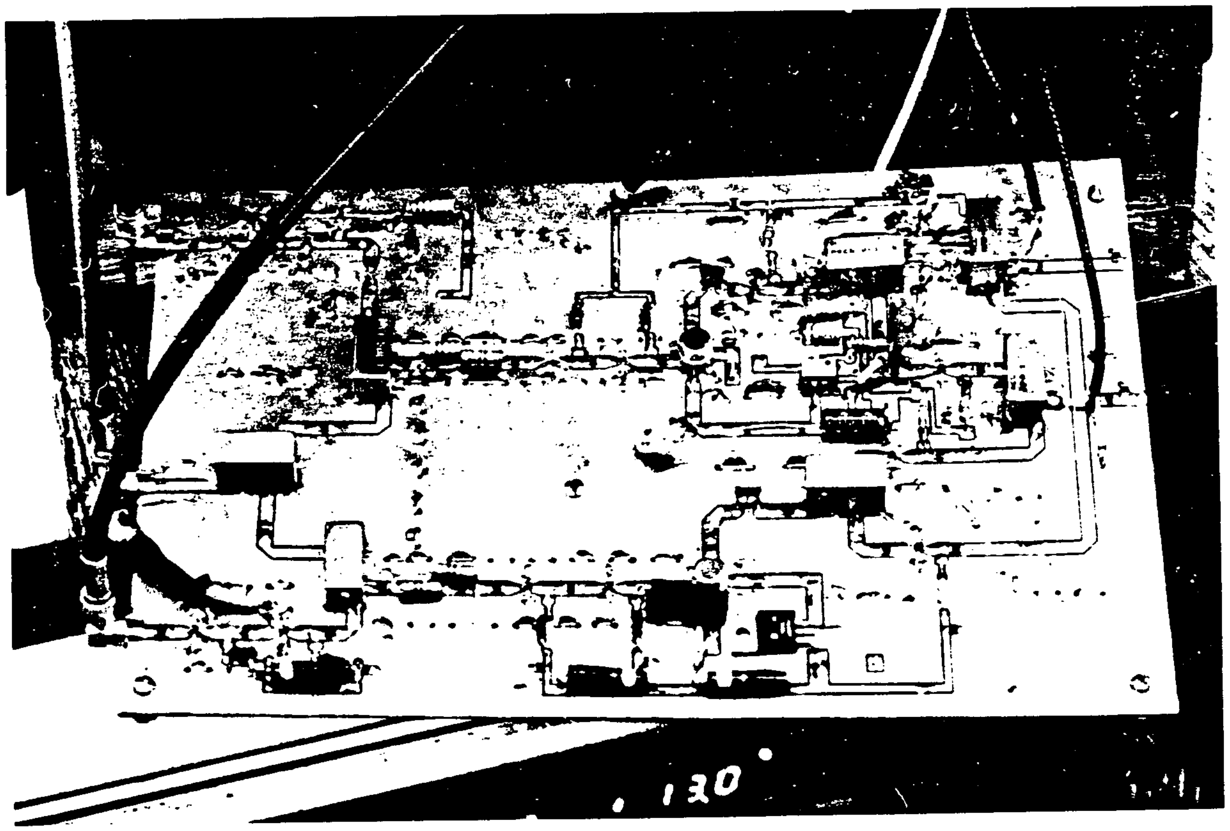


Figure 9. Photograph of LBCMI main amplifier and phase resolver circuit board positioned just outside the door of its shielded case.

The alignment mirror, beam splitter, and detectors are all mounted on platforms with spring loaded thumb-screw adjustment for final alignment. This has worked well for the laboratory prototype LBCMI system. However, is not expected to be vibration resistant enough for a vehicle mounted system. Such a system would require x,y translation stages with positive a lock. With this addition, the entire optical bench (or a smaller one) could be shock mounted on the vehicle for road operation.

The operation of the laboratory prototype system during field tests was very consistent with no major alignment problems. It operated for several days with only minor thumb screw adjustments of the beam splitter. The basic alignment procedure described in the System Operation Section of this report was only performed once.

3.0

LBCMI Software Design and Operation

The computer software which operates with the LBCMI performs three basic operations. First, it must synchronize computer operation with the LBCMI scanning speed and digitize data in real time. Second, it must decode the raw LBCMI data. And third, it must produce graphics displays and storage disk files of the profile data. The software is contained in three basic modules; calldash.c, scope.c, and screen.c which perform these functions. All software produced by this project is written in the TURBO C language and a complete source listing is included in Appendix A. The three modules are compiled into a single executable program which is named scanner.exe. This executable program is contained on a demonstration diskette available from the authors upon request together with 31 sample data files produced with the laboratory prototype LBCMI system.

A second program was developed during the project to produce simulated LBCMI data which could then be compared with the actual data which the system produced. This program will also allow the introduction of various deviations from theoretically perfect data which were used to develop part of the correction algorithms used with real data. This program, sim.c and sim.exe, is also included in Appendix A and on the demonstration diskette.

The laboratory prototype LBCMI system was attached to a Metrabyte Dash-16 analog to digital converter circuit board which performed the actual A/D conversions under the control of the calldash.c program. Calldash.c, in turn, calls assembly language subroutines provided by Metrabyte with the DASH-16 in a file called dash16.obj which is not included with this report since it is provided with the Metrabyte hardware.

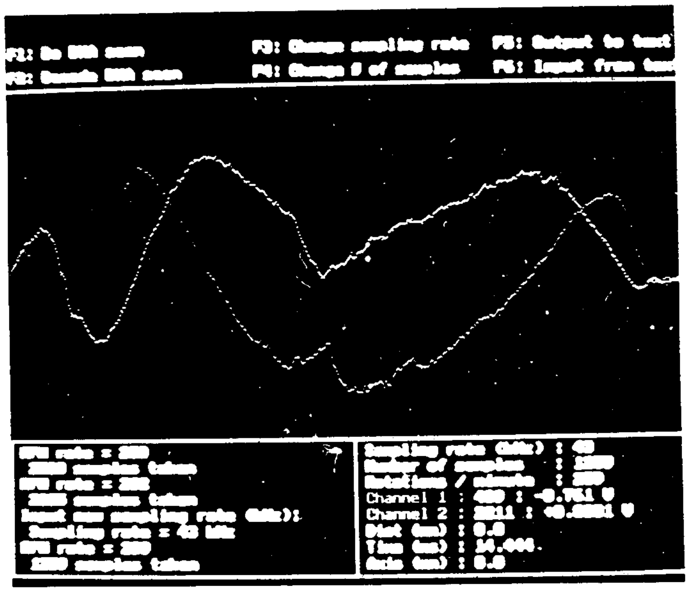
The scope.c module provides executive control of the system software. When the operator initiates an LBCMI scan this module first samples the trigger pulses produced by the trigger circuit described in the System Construction Section of this report to measure the current scanning RPM rate. Then, data for a complete scan are synchronized with a single trigger pulse and digitized and stored in computer memory via a direct memory access (DMA) operation. This is performed by the DASH-16 hardware through a calldash.c system call to the board.

After the data are captured by the computer the raw scan data are plotted on the computer display by screen.c in red and green for the two quadrature output channels of the phase resolver circuit. These two curves will ultimately be combined into a single profile curve. The software determines which of the two channels, which are 90° out of phase with each other, is in the linear portion of its range. Then, the composite profile curve is generated by piecing together alternate portions of each of the two quadrature channels. However, before this composite profile curve can be generated several preprocessing steps are performed.

The preprocessing and decoding steps are summarized in Figure 10. The two channels of raw data are shown in Figure 10a. The first preprocessing step is necessary because of the prototype marginal sensitivity. As the scan moves away from the middle the amplitudes of the two channels falls off slightly. This is corrected through normalizing the peak amplitudes. The result of this normalization are shown in Figure 10b.

The next processing step is to decode the raw data into perceived distance values before the cosine(Θ) correction due to scanning. The results of this step produces two additional curves which represent the total perceived distance between the scanning mirror and the road surface. These curves are shown in Figure 10c.

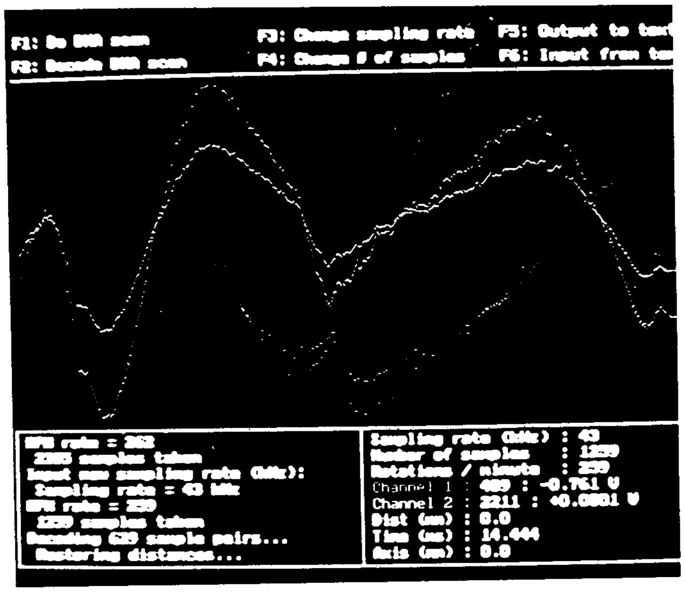
The perceived distance curves are next corrected for the cosine(Θ) factor and added to the screen display. And finally, the composite surface profile is calculated and displayed as a bright white curve as shown in Figure 10d. This final profile curve is currently displayed inverted compared to normal convention. Thus, an object laying on a flat surface will appear as a depression as shown by the dip near the center of the profile in Figure 10d.



```
F1: Do [illegible] scan          F3: Change sampling rate    F5: Output to text
F2: Decode [illegible] scan      F4: Change # of samples     F6: Input from text

[illegible] rate = [illegible]                 Sampling rate (kHz) : 43
  [illegible] samples taken                    Number of samples   : [illegible]
[illegible] rate = [illegible]                 Rotations / minute  : [illegible]
  [illegible] samples taken                    Channel 1 : [illegible] : -0.761 V
Input new sampling rate (kHz):                 Channel 2 : [illegible] : +0.0001 V
  Sampling rate = 43 kHz                       Dist (mm) : 0.0
[illegible] rate = [illegible]                 Time (ms) : 14.444
  [illegible] samples taken                    Axis (mm) : 0.0
```

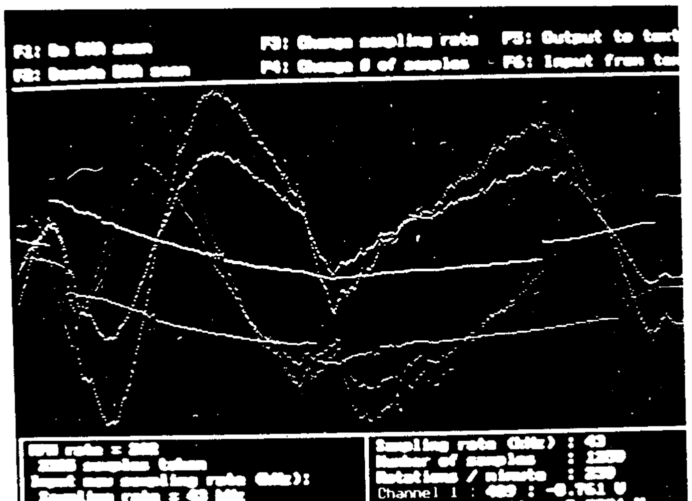
10a. Raw LBCMI scan data.



```
F1: Do [illegible] scan          F3: Change sampling rate    F5: Output to text
F2: Decode [illegible] scan      F4: Change # of samples     F6: Input from text

[illegible] rate = 262                         Sampling rate (kHz) : 43
  [illegible] samples taken                    Number of samples   : 1250
Input new sampling rate (kHz):                 Rotations / minute  : 250
  Sampling rate = 43 kHz                       Channel 1 : 489 : -0.761 V
[illegible] rate = 250                         Channel 2 : 2211 : +0.0001 V
  1250 samples taken                           Dist (mm) : 0.0
Decoding 625 sample pairs...                   Time (ms) : 14.444
  Restoring distances...                       Axis (mm) : 0.0
```

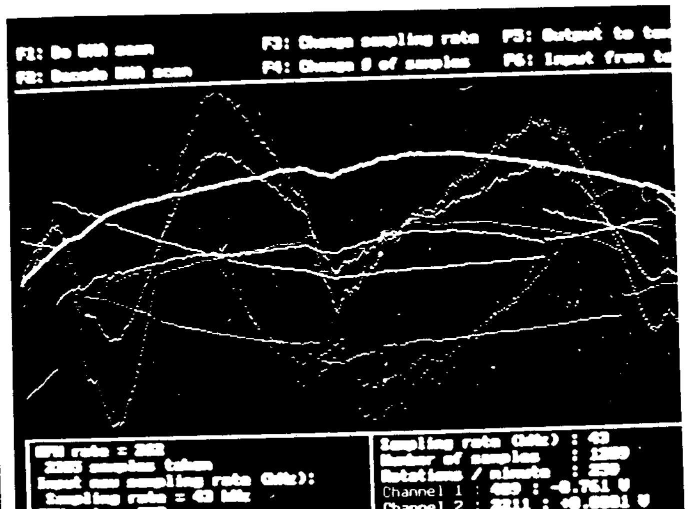
10b. Normalized scan data.



```
F1: Do [illegible] scan          F3: Change sampling rate    F5: Output to text
F2: Decode [illegible] scan      F4: Change # of samples     F6: Input from text

[illegible] rate = 262                         Sampling rate (kHz) : 43
  [illegible] samples taken                    Number of samples   : 1250
Input new sampling rate (kHz):                 Rotations / minute  : 250
  Sampling rate = 43 kHz                       Channel 1 : 489 : -0.761 V
[illegible] rate = 250                         Channel 2 : 2211 : +0.0001 V
  1250 samples taken                           Dist (mm) : 0.0
Decoding 625 sample pairs...                   Time (ms) : 14.444
  [illegible] distance [illegible] = 2.[illegible]   Axis (mm) : 0.0
```

10c. Perceived distance plots.



```
F1: Do [illegible] scan          F3: Change sampling rate    F5: Output to text
F2: Decode [illegible] scan      F4: Change # of samples     F6: Input from text

[illegible] rate = 262                         Sampling rate (kHz) : 43
  [illegible] samples taken                    Number of samples   : 1250
Input new sampling rate (kHz):                 Rotations / minute  : 250
  Sampling rate = 43 kHz                       Channel 1 : 489 : -0.761 V
[illegible] rate = 250                         Channel 2 : 2211 : +0.0001 V
  1250 samples taken                           Dist (mm) : 0.0
Decoding 625 sample pairs...                   Time (ms) : 14.444
  Decoding complete                            Axis (mm) : 0.0
```

10d. Final profile plot.

Figure 10. LBCMI computer display of scanned data through normalization and decoding steps which result in final surface profile.

3.1 Software Laboratory Operation

The output of the LBCMI in the laboratory matches very closely the theoretical predictions. The ideal output of the quadrature phase resolver circuits is given by:

$$V_1 = \sin(2\pi * d/\cos(\Theta)) \text{ and } V_2 = \cos(2\pi * d/\cos(\Theta)) \quad (3.1)$$

where V = output voltage, d = vertical distance from LBCMI to road, and Θ = angle from vertical to current measurement.

Figure 11 shows these two outputs over a 90° scan for an ideal system. The perceived distance from the LBCMI to the road d' is obtained from equation 3.1 according to:

$$d' = d / \cos(\Theta) \quad (3.2)$$

where d' = perceived distance and d = actual distance.

The ideal perceived distance d' is shown in Figure 12 together with the quadrature phase detector outputs.

The two channels of distance measure as output from the quadrature phase resolver are identical except for a 90° phase shift (ie. sine vs. cosine) as shown in Figure 11. However, the actual outputs from the LBCMI for a flat surface are slightly different as shown in Figure 13. Two characteristics of these curves are of interest. First, the high frequency variations along each curve represent noise in the received and amplified signal. Note that the noise is different in each channel. Second, the side peaks are of differing amplitudes while they should be of equal amplitude. This is the result of light reflection variations outside the range of the AGC circuit.

The software which extracts the perceived distance from these signals rates the "goodness" of each of the quadrature output signals based on the position in the sine wave. It assigns a low "goodness" when a sample is near the sine wave peak and a high "goodness" when the sample is in the linear region of the curve. Using these "goodness" ratings as weights an average is found between the two data channels. The average found is the mean when both are equally good and swings closer and closer to the better of the two channels as the goodness levels shift apart.

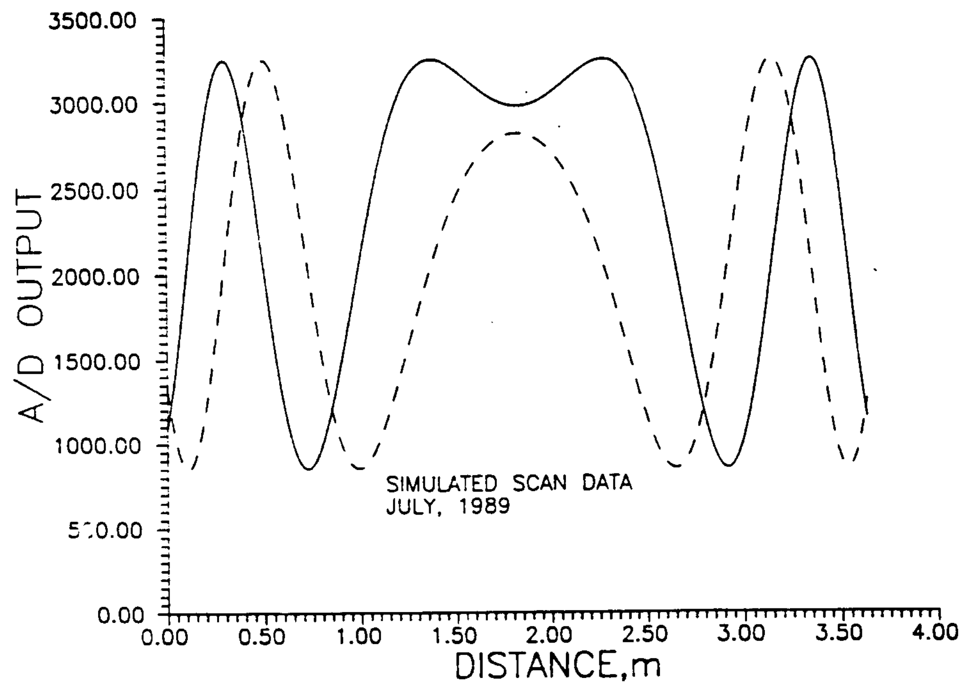


Figure 11. Theoretical output of quadrature phase resolvers.

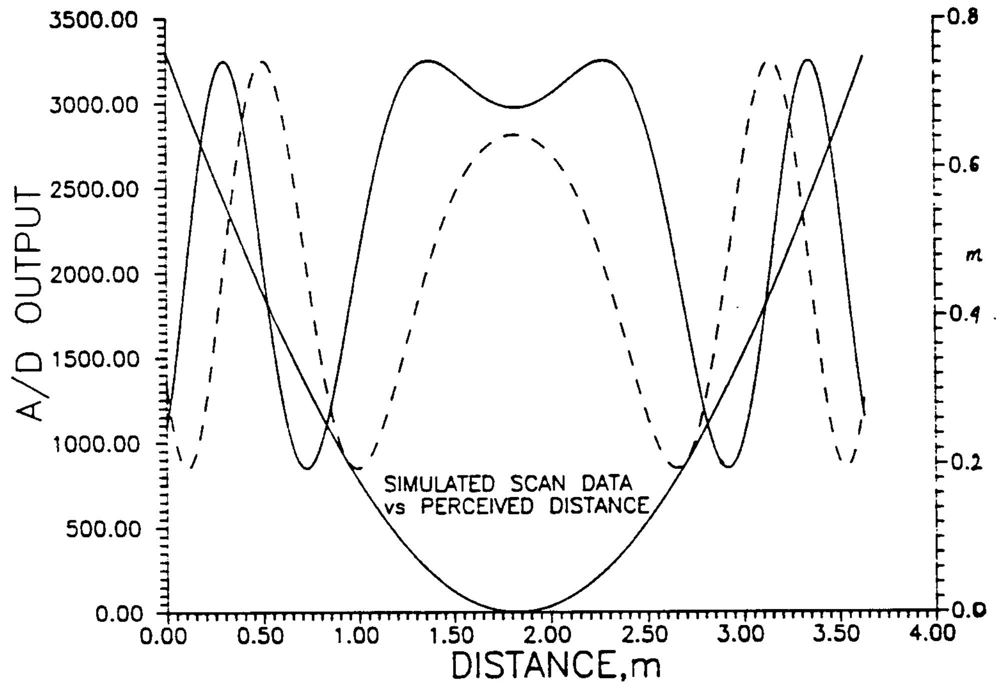


Figure 12. Theoretical perceived distance measured by LBCMI.

The resulting perceived distance for a flat surface is shown in Figure 14 together with the final resolved distance. This resolved distance is derived from the perceived distance according to equation 3.2. The values for this final conversion are derived from a trigger photodetector which is positioned at 45°. Therefore, the final resolved distance is calculated from the first sample taken after the trigger pulse according to:

$$d = d' \cos(45°) \tag{3.3}$$

where d = resolved distance and d' = perceived distance.

These algorithms produce some low frequency variations in resolved distance as shown in Figure 14 which are greater than the high frequency noise variations. We anticipate that calibration of each of the quadrature outputs can be performed and an algorithm can be developed to correct for differences between them. This would reduce the low frequency resolved distance errors. However, before such algorithms can be developed and tested the signal to noise ratio of the system must be improved and a large number of scans obtained.

As mentioned previously the recent announcement of a new high speed cooled detector should provide such improved data. Given such an improvement in raw data the software developed for this project will require some "fine tuning". Otherwise, the software as described here is solid and nearly complete.

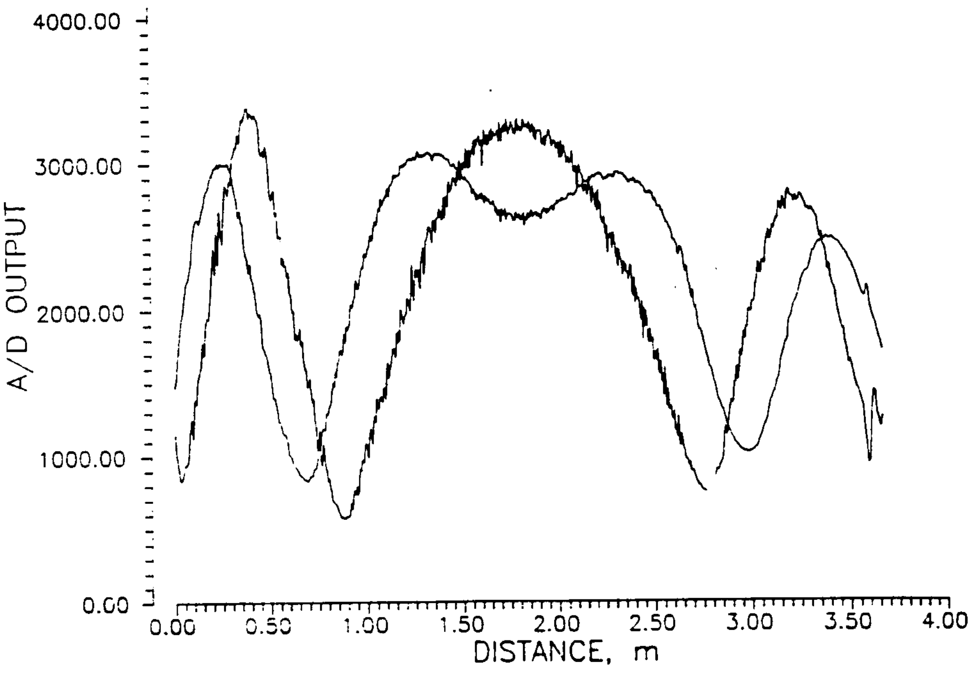


Figure 13. Actual output of quadrature phase resolvers for a flat surface.

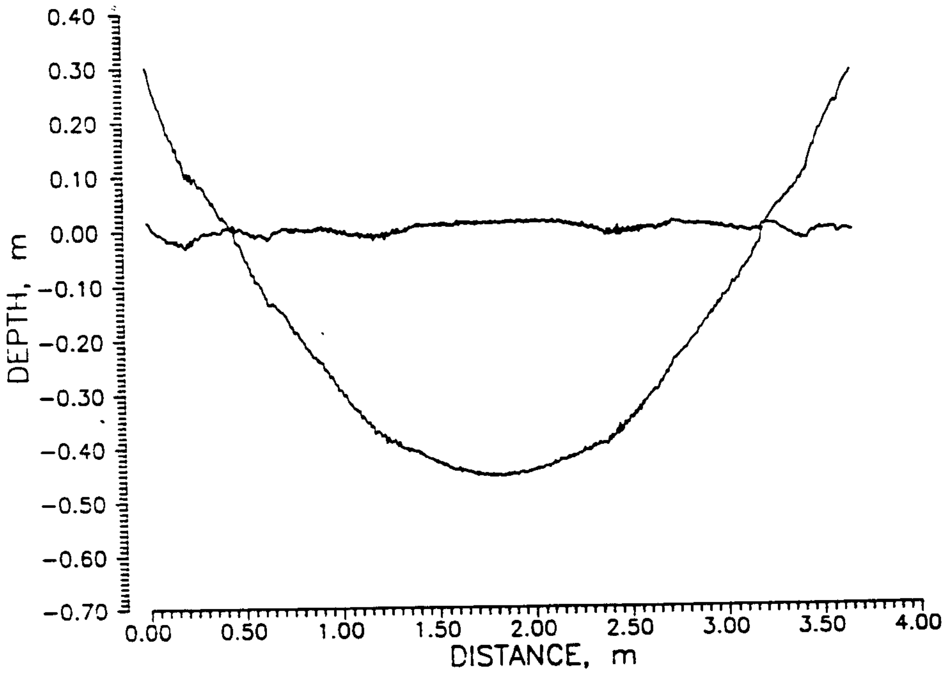


Figure 14. Actual perceived and resolved distance measures for a flat surface.

This Page Left Blank Intentionally

4.0

LBCMI System Alignment and Operation

An orderly procedure for alignment and operation of the LBCMI scanner system was developed during the field testing stage. Following this procedure produces consistent and reliable operation of the system within the limitations of sensitivity previously discussed. For the field test data obtained for this project major alignment was only required once. After this the system operated reliably for several days with only minor adjustments.

The first step of the alignment procedure is to mechanically align all of the system components on the optical bench as shown in Figure 15. First the laser, alignment mirror, and scanning mirror are aligned such that the laser beam is directed to the proper spot on the scanning mirror and is parallel to the surface of the optical bench. The laser beam should hit the center of the scanning mirror cube surface horizontally, and should just hit the edge of the cube vertically when it is rotated at 45° to just begin a scan.

Once the scanning mirror is aligned the beam splitter is adjusted so that the reference beam is centered on the reference receiver detector. This makes almost no difference in the scanning mirror alignment. However, it should be rechecked at this point.

The scanning mirror is now manually adjusted so that the laser beam is directed vertically onto the surface to be scanned. This point is then clearly marked in heavy black lines on a piece of white paper placed on the surface to be scanned. At this point the laser power is turned off and the main receiver lens is adjusted manually. This is accomplished by rotating the lens eye-piece, looking through it, aligning the lens to point at the scanning mirror, and focusing so that the black mark on the white paper is centered and in focus. This procedure takes some time to accomplish and it is imperative that the laser be turned off for safety since the technician is looking directly through the main receiver lens.

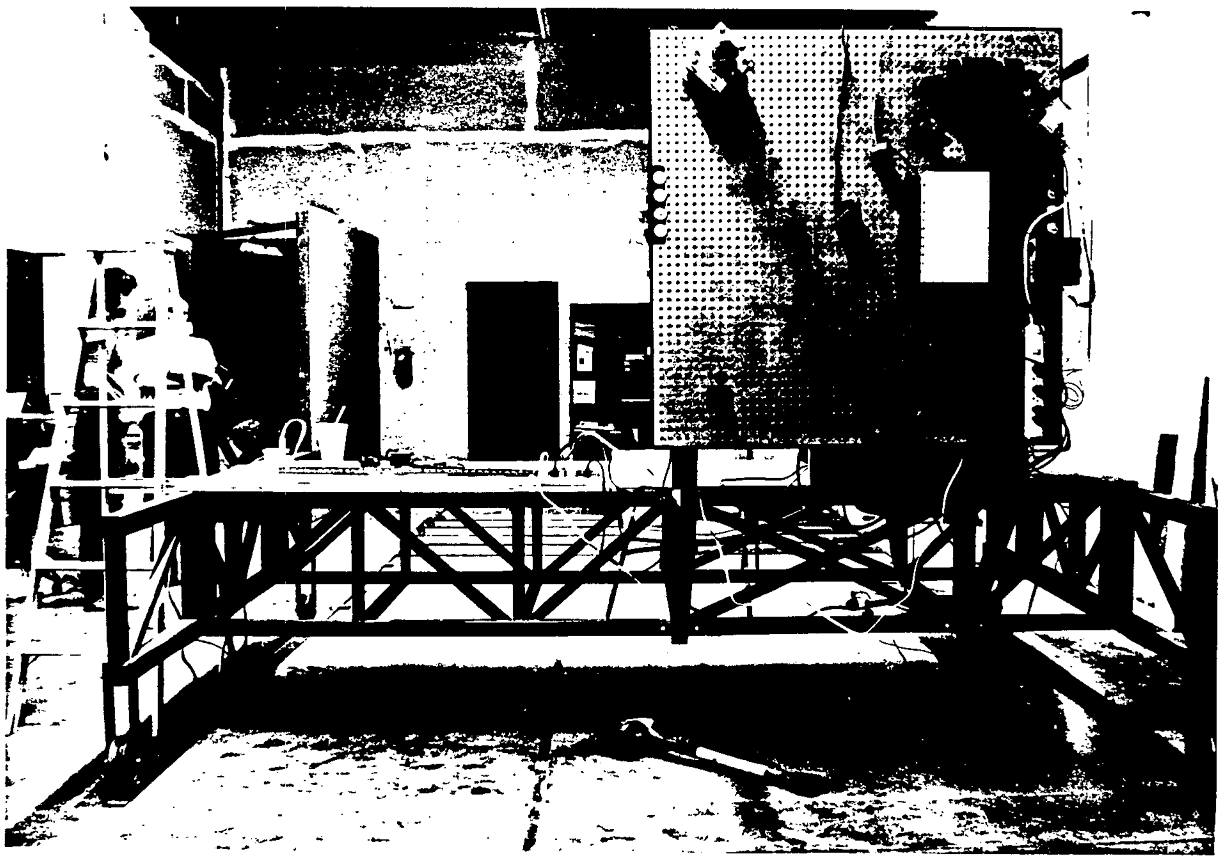


Figure 15. LBCMI system on field test mount after alignment.

Once the main receiver lens is aligned and clamped solidly the laser power is turned back on and the black mark is replaced with an adjustable mirror. The laser beam is then directed back into the main receiver lens and the point of focus at the eye-piece can be observed through laser safety goggles by reflecting the point on a special infrared to visible wavelength converter. When this point is found the eye-piece and main receiver photomultiplier tube (PMT) detector are both aligned so that the point of focus is on the tube's sensitive surface. It is also important that the high voltage supply to the PMT be turned off at this time to avoid damage to the PMT.

The final step of the mechanical alignment is to positions the trigger detector so that it intercepts the laser beam from the scanning mirror at an angle of 45° from vertical. At this point the LBCMI system is in mechanical alignment and ready for electronic alignment.

It should be stressed that during the mechanical alignment procedure all personnel should wear laser safety goggles because the scanning mirror is not rotating and static laser beams are being moved about during the procedure. However, this is a maintenance procedure, should not be required often, and should only be performed by qualified technicians who are aware of the proper safety precautions.

The LBCMI electronic alignment will probably be a routine daily procedure. However, it is quick and can be performed while the system is mechanically in its operational mode as shown in Figures 16 and 17. Thus, for an operational system this procedure would not pose any special safety considerations. The first step in this procedure is to measure the microwave frequencies of both the laser oscillator and the phase resolver circuit local oscillator. These frequencies can be measured with a frequency counter meter attached to test points on the circuits. First, the laser oscillator should be adjusted so that is at the desired frequency, which for our laboratory prototype system was 300 Mhz. Next, the local oscillator should be adjusted so that it is 10.7 MHz above the laser oscillator, in this case at 310.7 MHz.

The final adjustments of the LBCMI system require a low frequency (20 MHz) oscilloscope as shown in Figure 17. With the LBCMI operating, the scan trigger circuit is used to trigger the oscilloscope resulting in the top trace in Figure 17. Then one of the two output channels from the phase resolver circuit is displayed as in the lower trace in Figure 17. With this display three adjustments can easily be made. First, the local oscillator frequency can be fine tuned to maximize the lower trace. The phase resolver gain control can be adjusted to provide the desired ± 1 volt peak-to-peak signal. Finally, the scanner mirror motor can be adjusted to produce the desired RPM.

These steps complete the required alignment of the LBCMI system. During the field tests the mechanical alignment proved to be very stable. The electronic alignment was performed at the beginning of each day and in most cases was stable throughout the day. Some temperature dependency of the electronic alignment was experienced with the laboratory prototype system one day as the weather changed. However, the changes were minor and quickly adjusted.

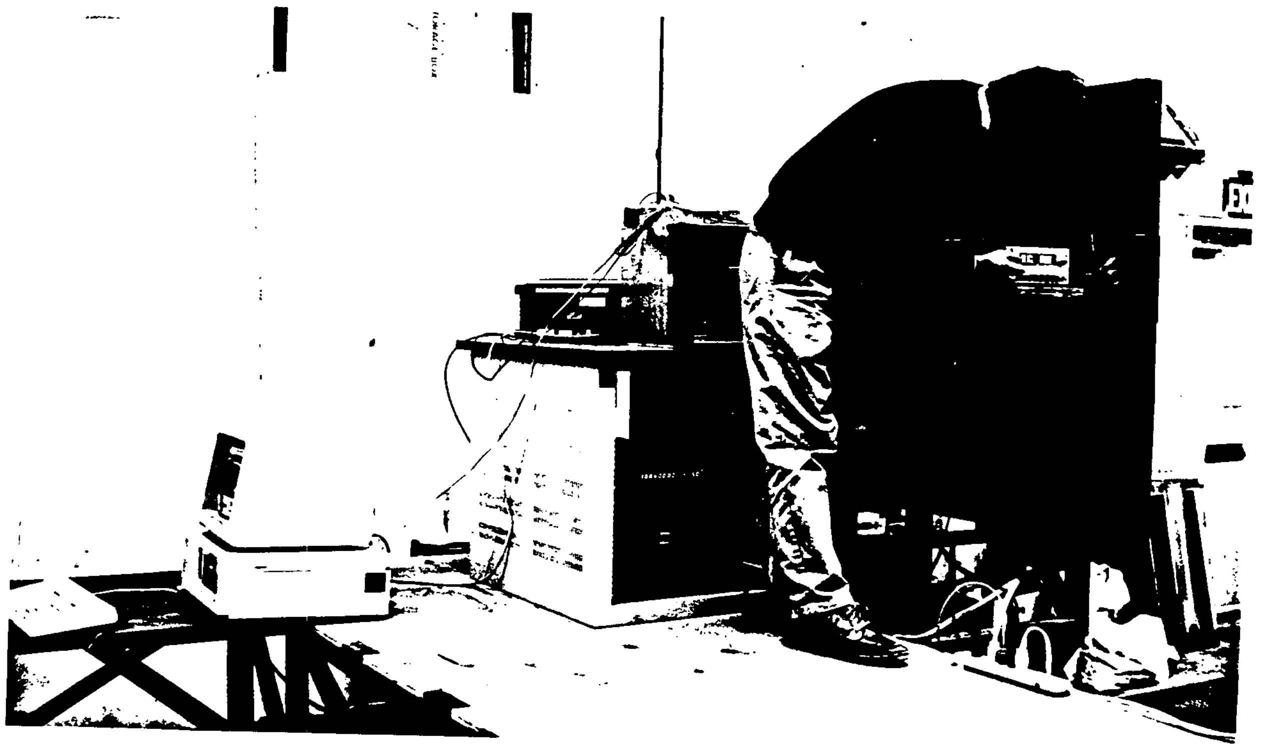


Figure 16. LBCMI alignment steps performed behind the scanning components.

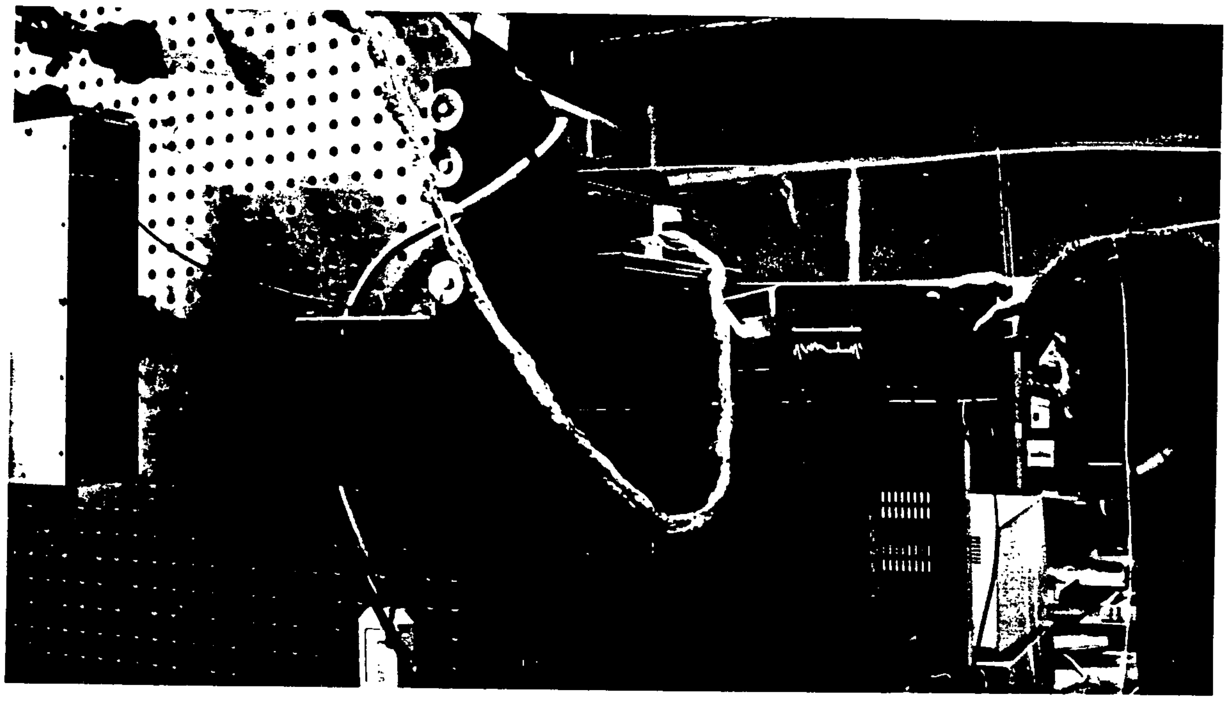


Figure 17. Final LBCMI adjustments using standard laboratory oscilloscope.

4.1 System Operation

The LBCMI system operation proved to be very simple and stable within the limitations previously described for the laboratory prototype system. The scanner computer program provides almost complete control over the system operational parameters as shown in Figure 18. The sampling rate and number of samples per scan are under software control. For the prototype system the scanning mirror RPM was controlled with a variable DC voltage supply located near the computer, even though it is read by the computer. However, for an operational system this could easily be changed to provide a servo feed-back speed system under computer control. In fact, some RPM drift was experienced during the field tests which this change would eliminate.

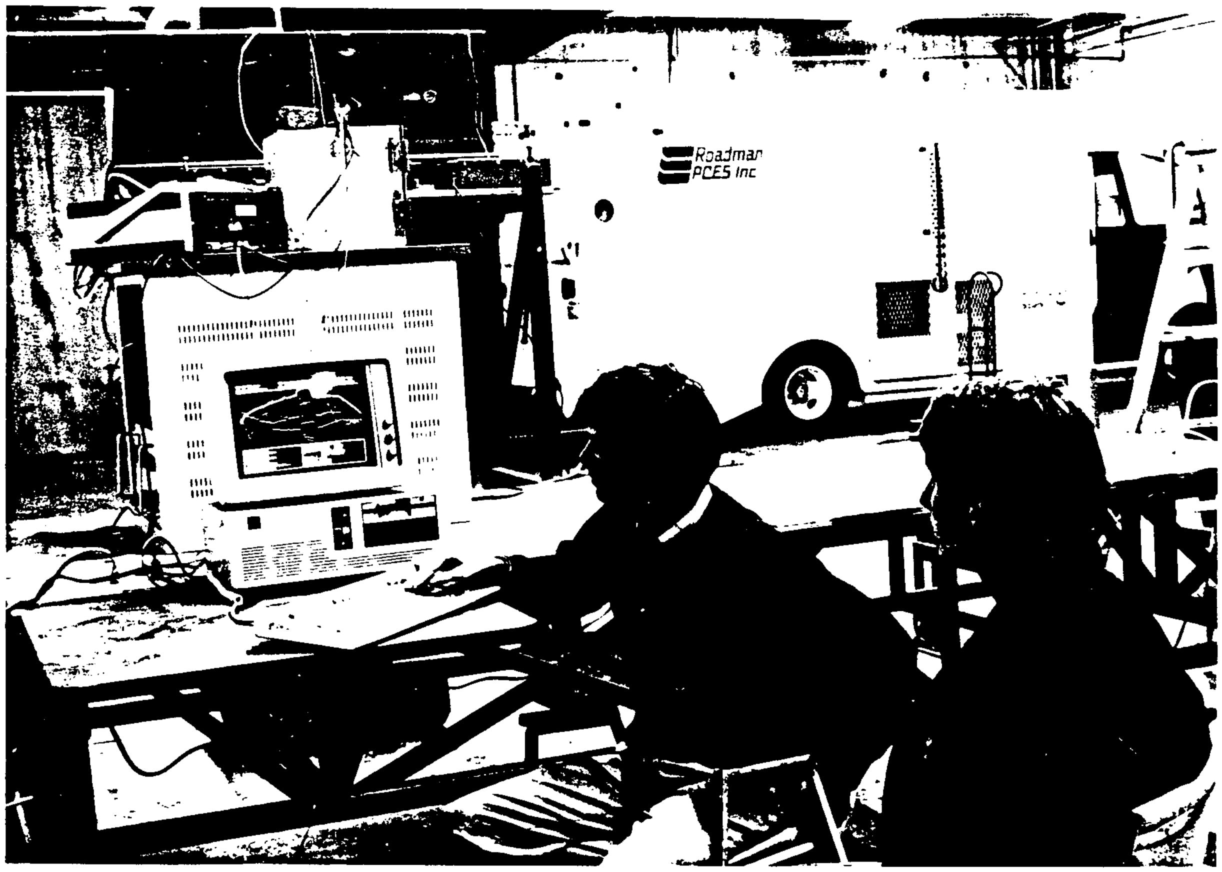


Figure 18. LBCMI operating under computer control during field tests.

The laboratory prototype LBCMI scanner was operated by knowledgeable personnel with laser safety goggles using commercially available laser products which were registered. However, an operational LBCMI system would have to
conform to government regulations for light-emitting products and be
registered as a separate device. Our current calculations show that such a system should be able to conform to a Class I Device, the least restrictive category, if proper shielding and controls are implemented. [5]

A "Class I laser product" means any laser product that does not permit access during operation to levels of laser radiation in excess of the accessible emission limits defined in [5]. Calculations for the level of radiation are made on the basis of a circular aperture stop of 7 mm diameter, with a circular solid angle of acceptance of 1×10^{-3} steradians, with collimating optics of 5 diopters or less. If it is a scanned measurement the detector should rotate to maximize detected radiation with a speed up to 5 radians per second.

Using these criteria together with equations and data from Tables I and V from [5] the acceptable accessible emission limits for a LBCMI system is 4.46×10^{-5} watts. Assuming the system is shrouded down to the point that the direct beam is scanning 6 feet, and assuming the system sweep rate is much greater than 5 radians per second the accessible emission is 1.53×10^{-5} watts using a 4 milliwatt laser as in the laboratory prototype LBCMI system. If an
operational system were to use a 16 milliwatt laser the accessible emission would be 6.12×10^{-5} watts. However, increasing the shroud to 8 feet would result is an accessible emission of 4.49×10^{-5} watts for a 16 milliwatt laser.

Thus, an operational LBCMI system should be able to meet these strict criteria for a Class I laser device. In any event, should an operational device be built it must be registered and tested for safety. In addition to the emission limits several safety devices will also be required. The laser will require a keylock power switch with an interlock to kill the laser if the unit is opened. It will require an audible warning before the laser comes on if the unit is operating with the interlock defeated as during mechanical alignment, and a manual reset to restore laser operation after service. Finally, it would require a scan failure shutoff device and appropriate caution signs at appropriate locations inside and outside of the unit.

All of these criteria and precautions can be satisfied with an operational LBCMI scanning system, at a cost. Therefore, there does not appear to be any problem with safety in implementing an operational system.

5.0

LBCMI System Performance

As mentioned previously in this report, when the laboratory prototype LBCMI was transferred to the moving platform for field tests the signal to noise ratio which had produced smooth results in the laboratory tests at 1.5 meters from a white target became marginal for actual road surfaces at the operational height of 2 meters. In spite of this problem the data presented in this section does demonstrate the potential capabilities of the LBCMI technology.

Due to the fall-off in returned signal at large angles the decoding algorithm could not normalize the raw data sufficiently to produce a reasonably flat profile over the entire 4 meter path. Rather, the scans all show a curved nature. However, the details of the target surfaces in the center portion of the scan clearly show that the system is accurately measuring the surface profile.

Several different types of target surfaces were scanned to test different capabilities of the system. Likewise, in order to simulate the results which could be expected from a system with a higher signal to noise ratio several scans from the same surface were averaged together. If the capabilities of the newly announced Hamamatsu photomultiplier tube are available for a future version of the LBCMI system these averaged scans should be representative of the expected results. However, the averaging process could not correct for the overall curve seen in the data. It can only correct for the high frequency noise which obscures the profile details.

Figure 19 shows the first target surface which consists of a flat surface with two 10 mm high bumps 400 mm long on each side of the center. These bumps represent inverse ruts. The pen in the photograph points to the position on the target surface which is directly below the scanning mirror. This position corresponds to the 2.00 meter mark on the LBCMI plots. This photograph is taken from such an angle that left and right are reversed with respect to the LBCMI plots. However, in this case the surface is left/right symmetric.

The raw LBCMI scan data of this surface for both phase detector output channels are shown in Figures 20 through 24. These plots show a single scan in Figure 20 and the effect of averaging 5, 10, 15, and 20 scans in the succeeding Figures. The reduction in signal noise is clearly seem from one Figure to the next. Likewise, the inside edges of the two "inverse ruts" are clearly visible in both channels. Figures 25 through 29 show the equivalent fully decoded surface profiles. Figure 25 is a single scan and Figures 26 through 29 show the results of averaging 5, 10, 15, and 20 scans.

The next target surface represents 5 mm high inverse ruts which looks the same as the surface shown in Figure 19 except the ruts are only one half as high. Figure 30 shows a single scan for this surface. In this Figure there is no sign of the ruts. They are completely obscured by high frequency noise. However, Figure 31 shows the average of 20 scans of this target surface and the inside edges of the 5 mm inverse ruts can be seen. They would be even more visible if the overall curve were not present.

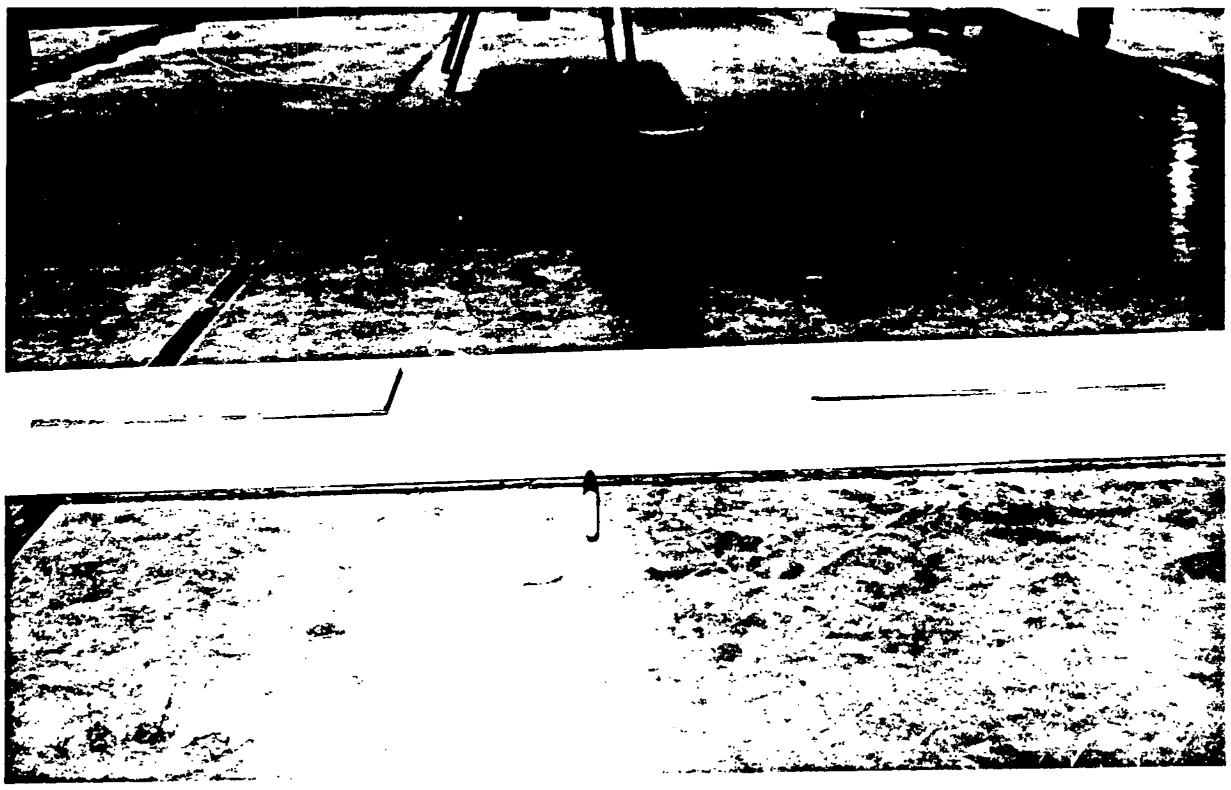


Figure 19. Photograph of sample surface with two 10 mm by 400 mm inverse ruts.

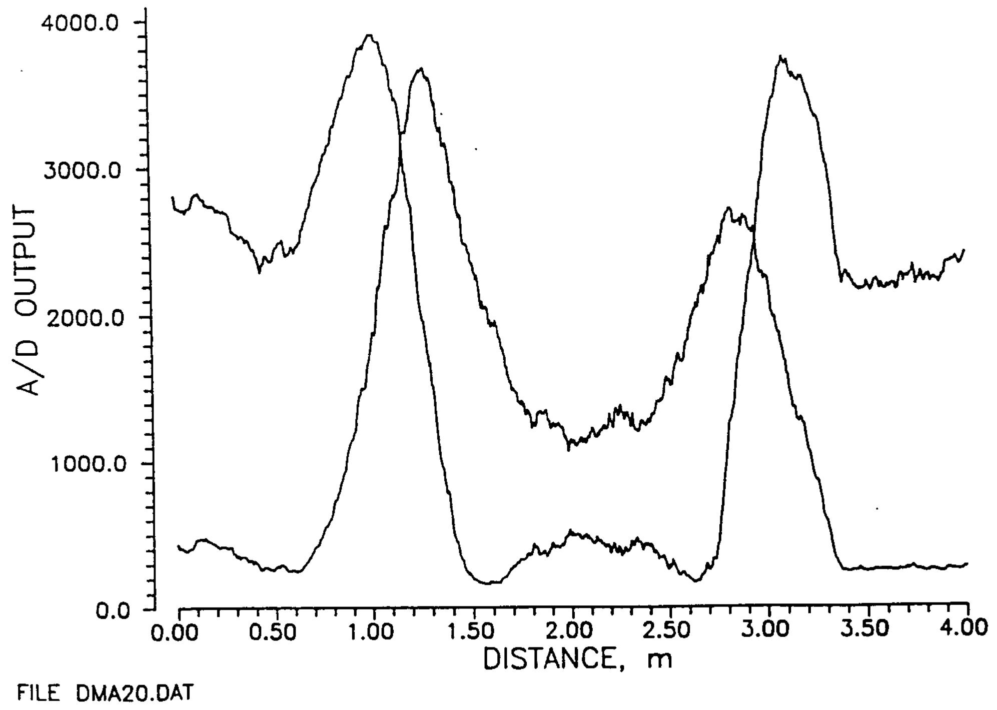


Figure 20. Raw LBCMI data for a single scan of the 10 mm inverse ruts.

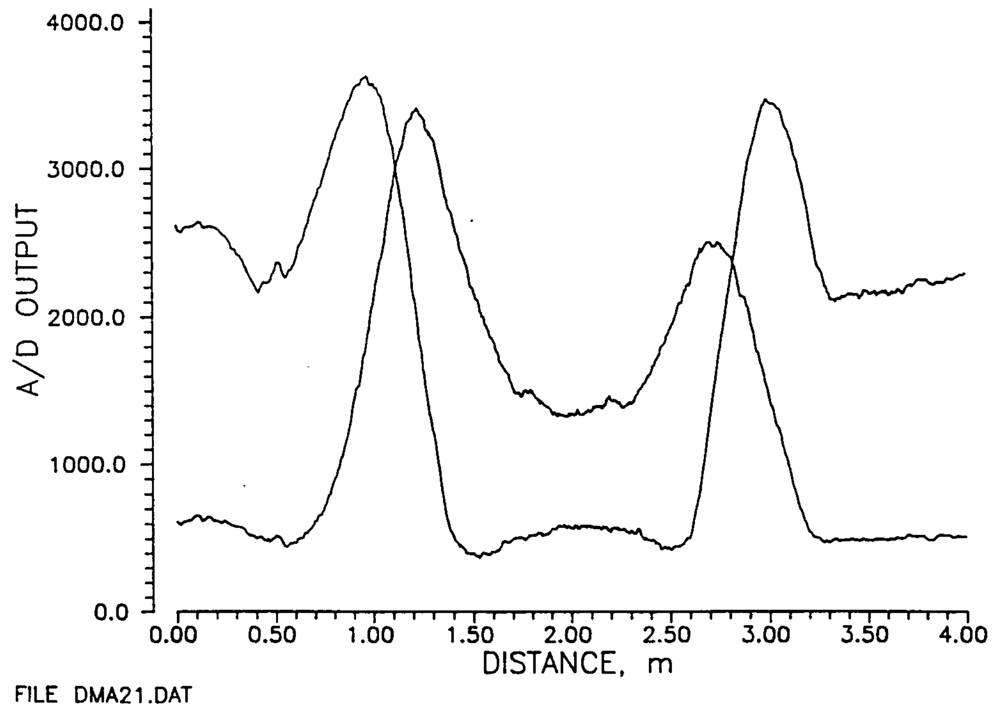


Figure 21. Raw LBCMI data for average of 5 scans of the 10 mm inverse ruts.

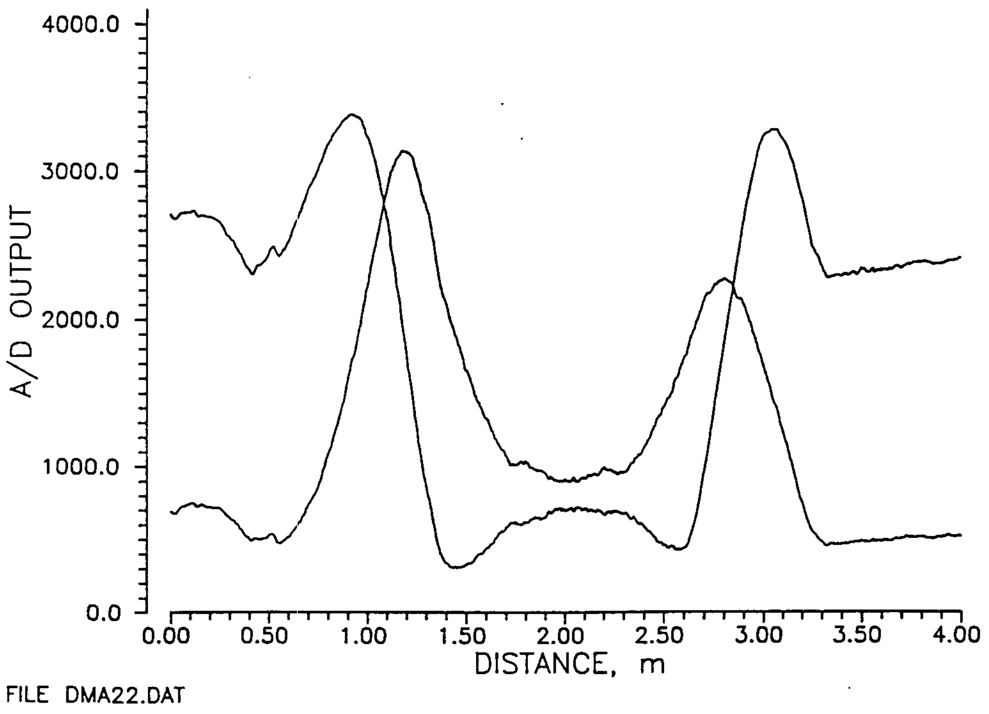


Figure 22. Raw LBCMI data for average of 10 scans of the 10 mm inverse ruts.

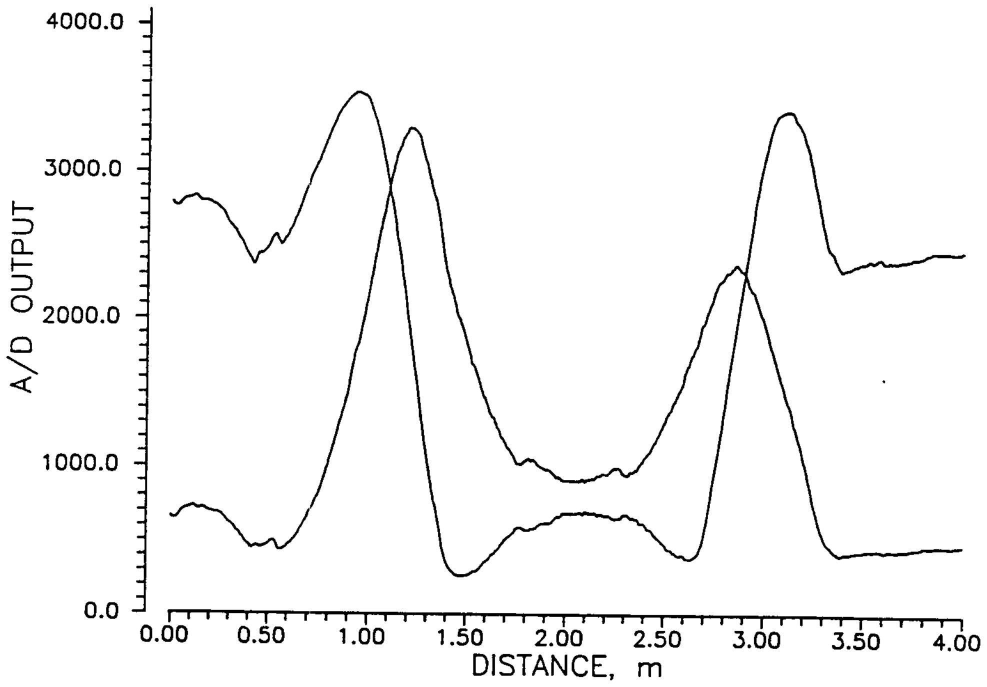


Figure 23. Raw LBCMI data for average of 15 scans of the 10 mm inverse ruts.

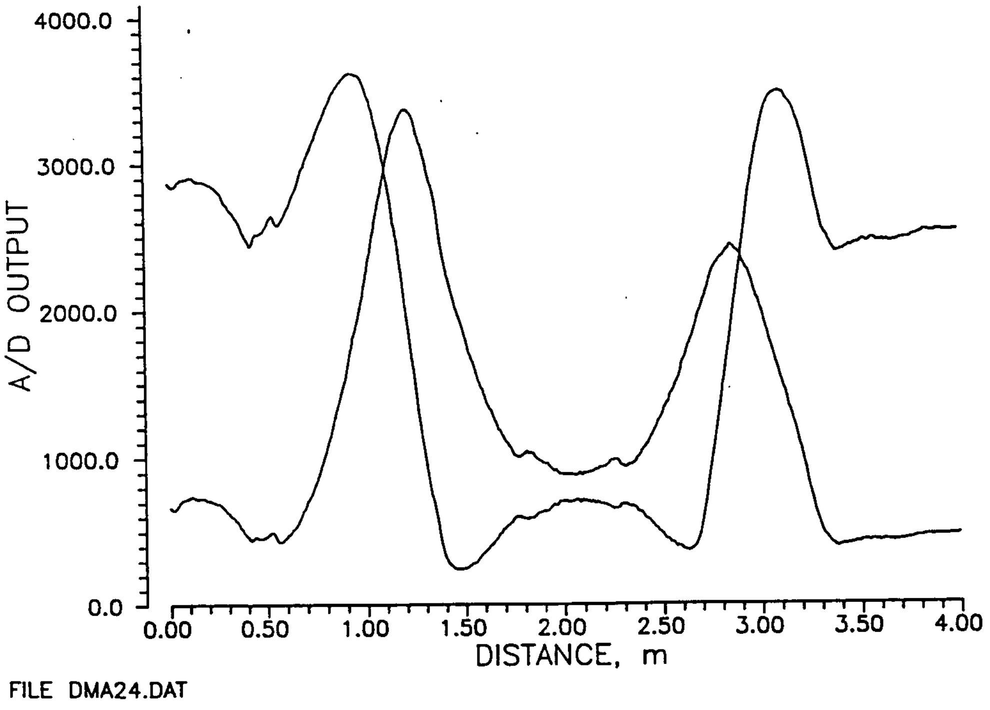


Figure 24. Raw LBCMI data for average of 20 scans of the 10 mm inverse ruts.

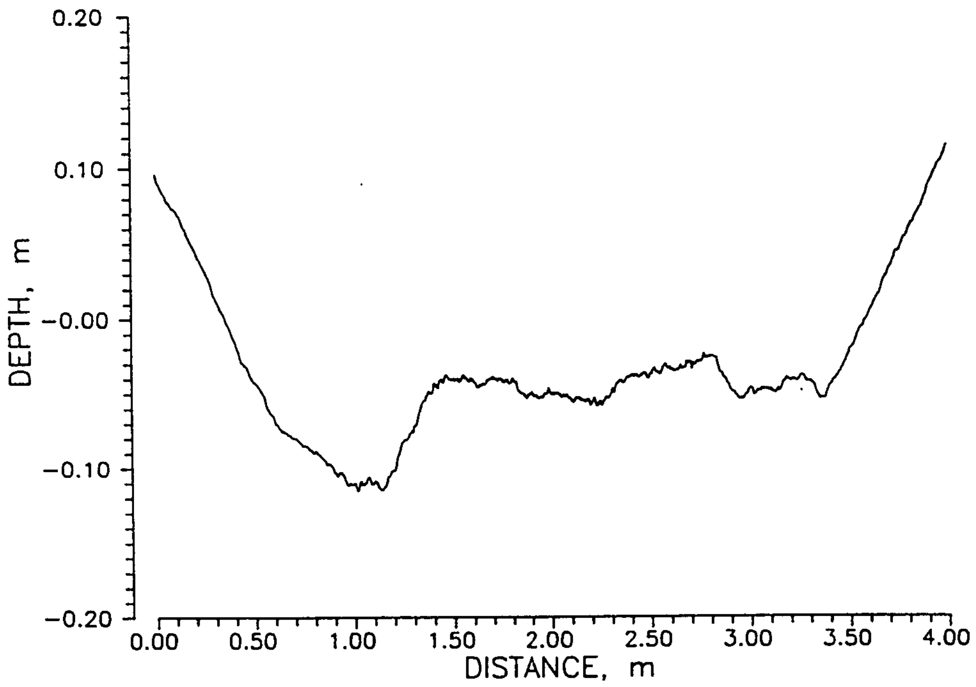


Figure 25. Decoded LBCMI profile for a single scan of the 10 mm inverse ruts.

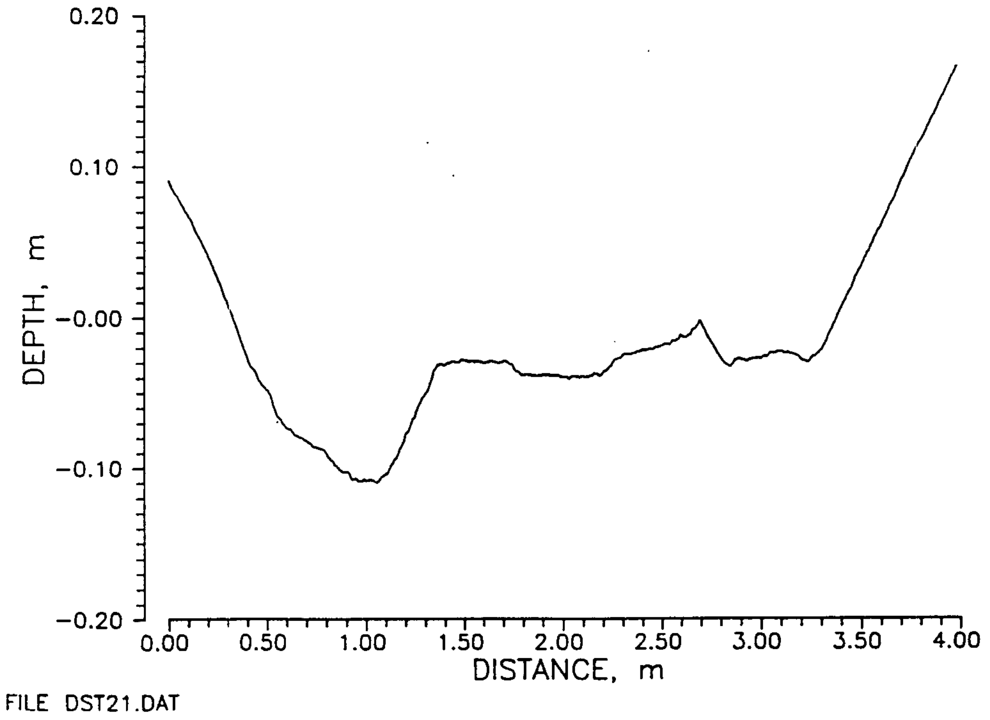


Figure 26. LBCMI profile for average of 5 scans of the 10 mm inverse ruts.

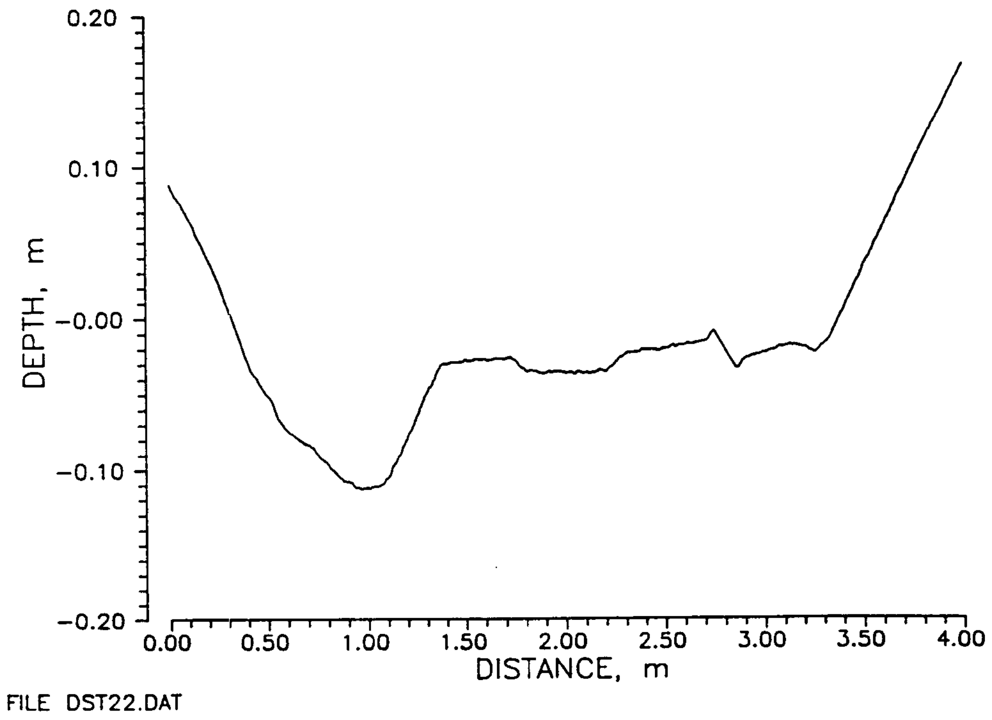


Figure 27. LBCMI profile for average of 10 scans of the 10 mm inverse ruts.

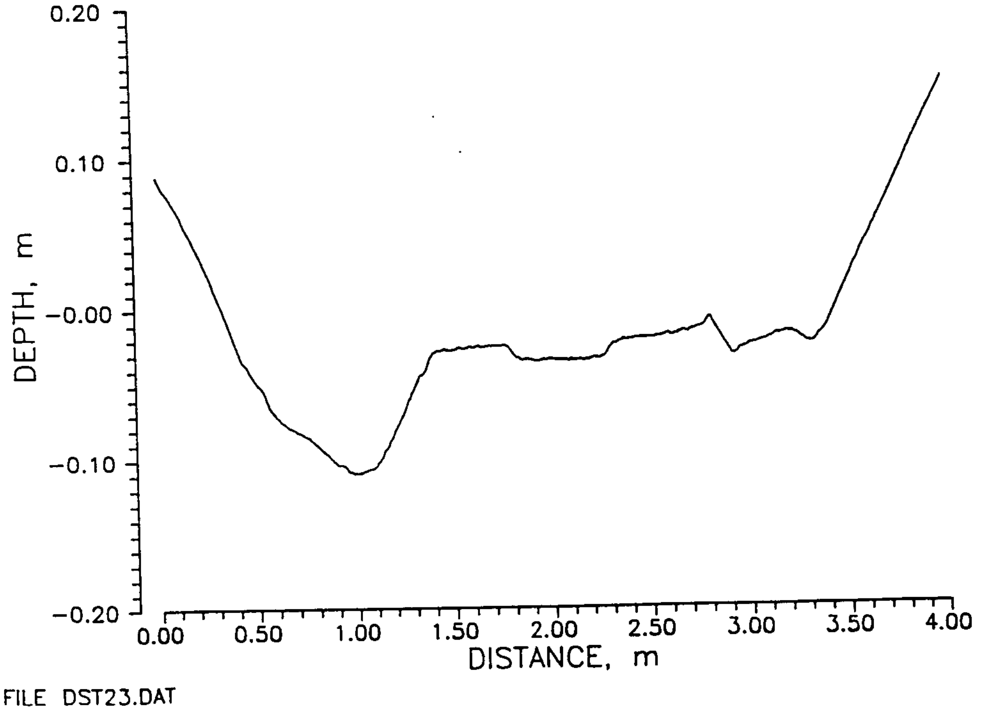


Figure 28. LBCMI profile for average of 15 scans of the 10 mm inverse ruts.

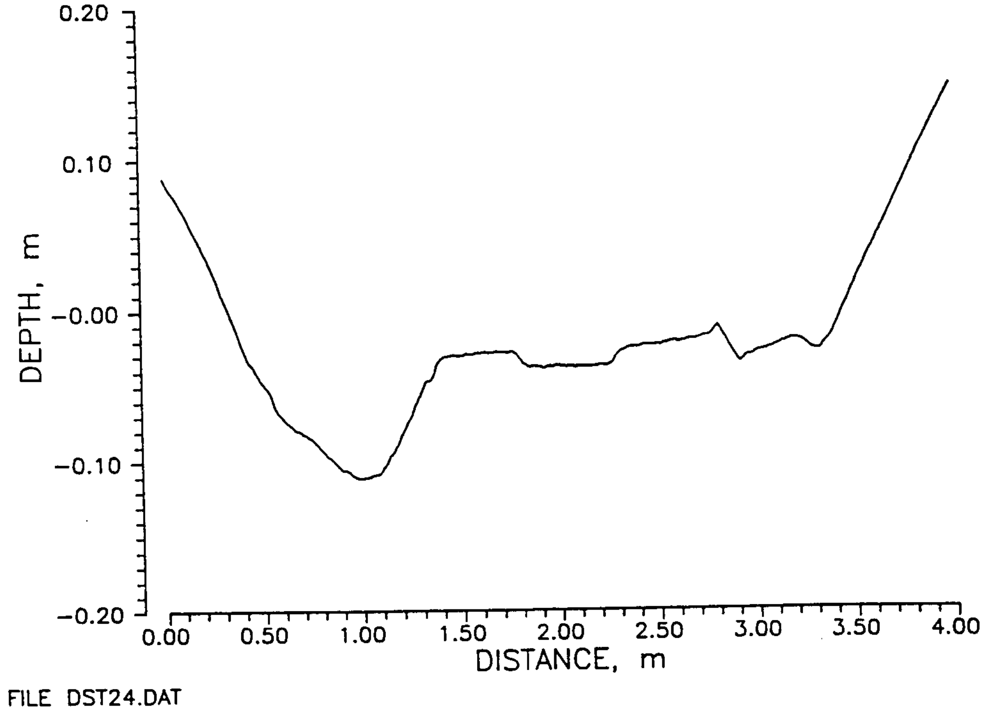


Figure 29. LBCMI profile for average of 20 scans of the 10 mm inverse ruts.

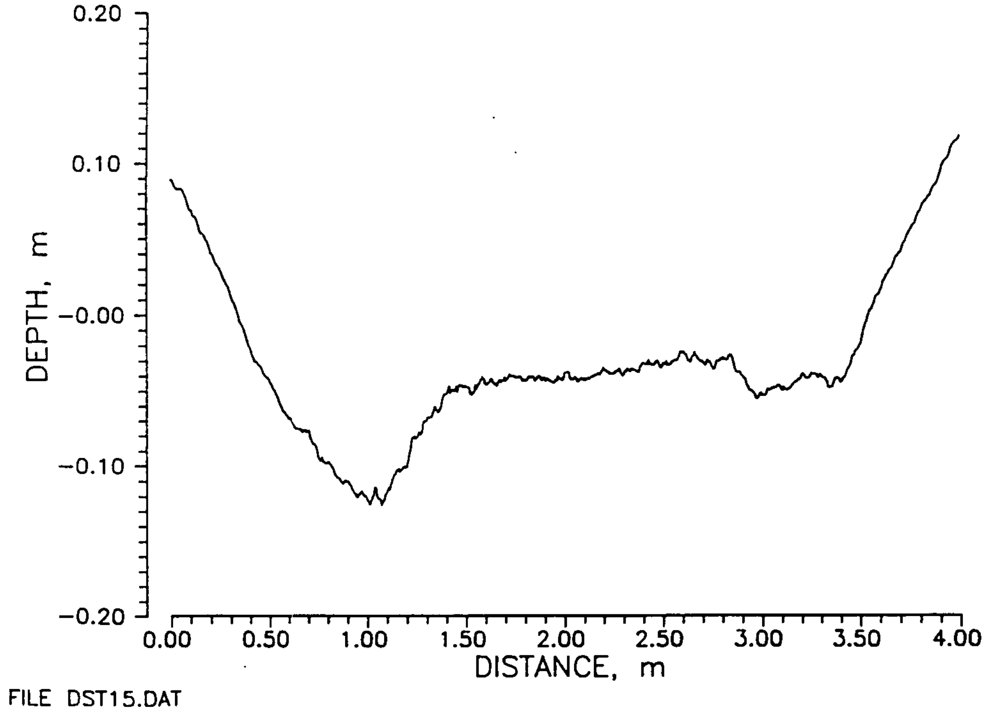


Figure 30. Decoded LBCMI profile for a single scan of the 5 mm inverse ruts.

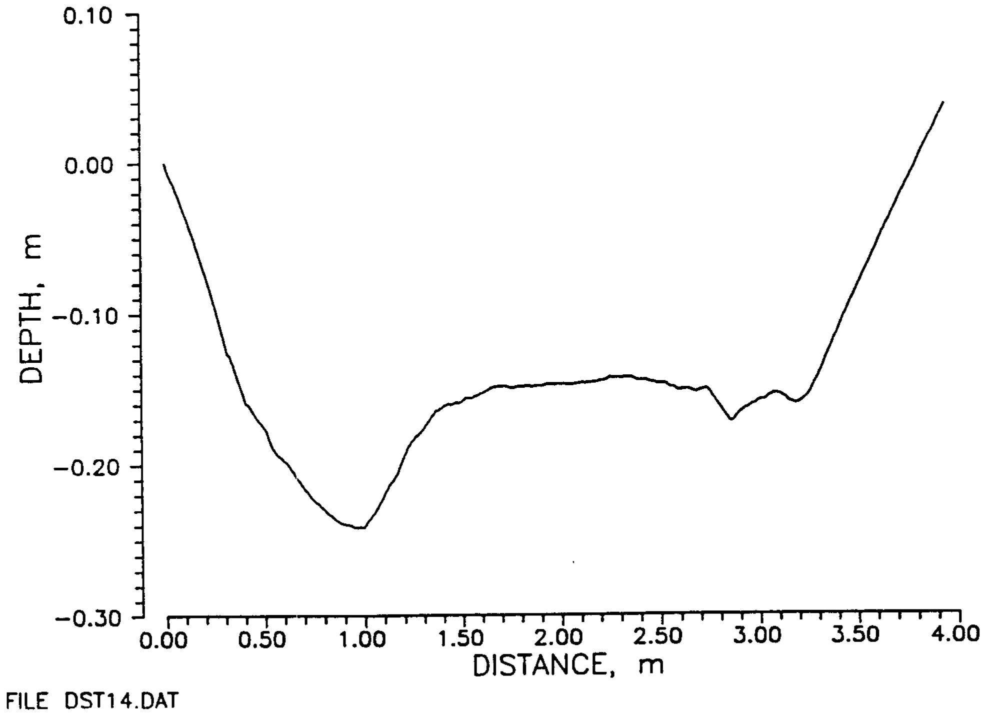


Figure 31. LBCMI profile for average of 20 scans of the 5 mm inverse ruts.

The next sample target surface was constructed to test the ability of the LBCMI to detect pavement cracks as well as ruts. This surface shown in Figure 32 consists if 5 raised surfaces 5 mm high by 150 mm wide which are positioned approximately 5 mm apart. The spaces between the raised surfaces simulate pavement cracks 5 mm deep by 5 mm wide. The pen in the photograph points to the position on the target surface which is directly below the scanning mirror. This position corresponds to the 2.00 meter mark on the LBCMI plots. This photograph is taken from such an angle that left and right are reversed with respect to the LBCMI plots.

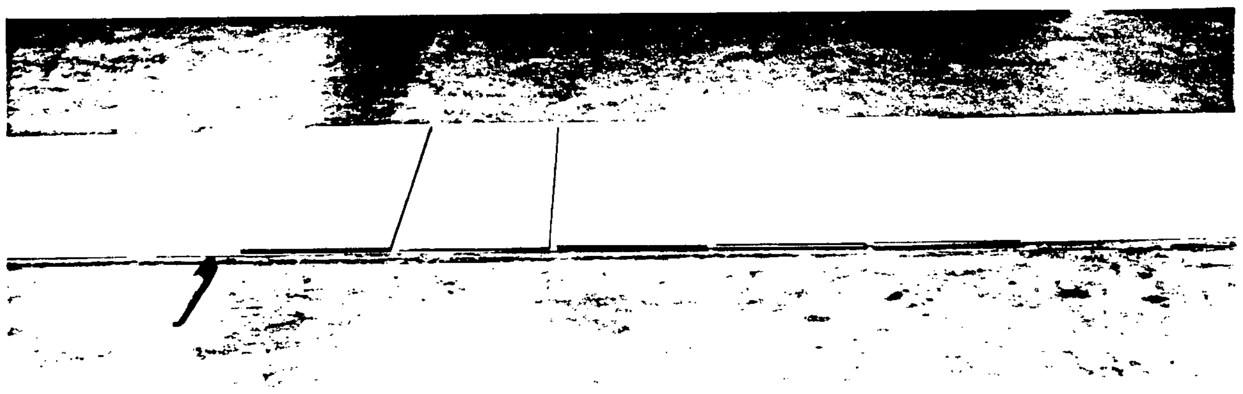


Figure 32. Photograph of sample surface with 5 mm by 5 mm cracks.

Figure 33 shows a single scan profile of this surface. As with the 5 mm inverse ruts, the single scan does not show the surface cracks. Figure 34 shows the average of 10 scans. Even this Figure does not clearly show the cracks. However, it does clearly show the 5 mm rise on the left of the scan.

The remaining sample target surfaces represent real road surface materials. The first two surfaces are made from road gravel, one of which is rough as shown in Figure 35, while the other is smooth as shown in Figure 36. Again, the pen points the center of the LBCMI scan. However, these photographs have the same left/right orientation as the scans.

Figure 37 shows a single scan for the rough gravel surface and Figure 38 shows the average of 20 scans. Likewise, Figure 39 shows a single scan for the smooth gravel surface and Figure 40 shows the average of 20 scans. Comparing these four profile plots it is clear that the LBCMI is accurately reflecting the relative roughness of the two surfaces.

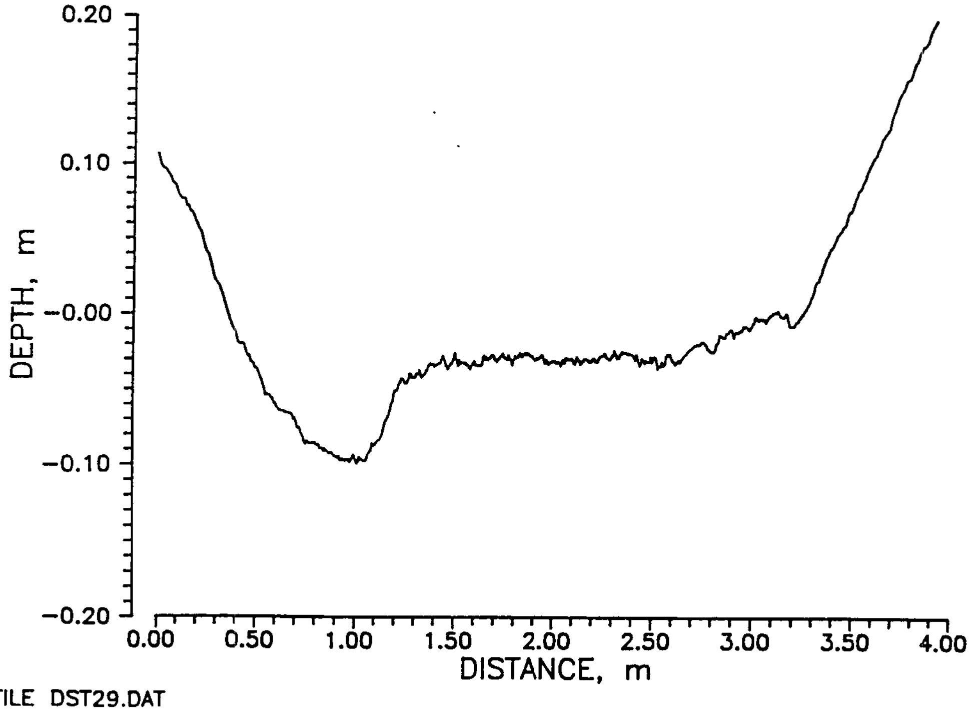


Figure 33. Decoded LBCMI profile for a single scan of 5 mm by 5 mm cracks.

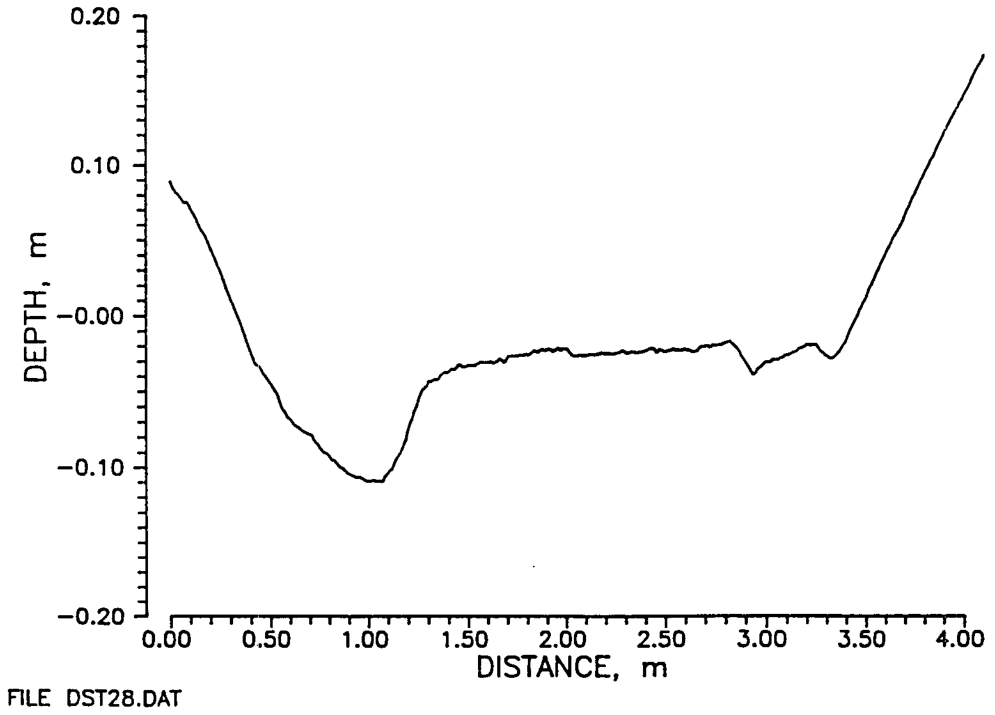


Figure 34. LBCMI profile for average of 10 scans of 5 mm by 5 mm cracks.

The next target consists of two actual road pavement blocks which were removed from an old highway. They were positioned below the scanner as shown in Figure 41. This photograph is taken from such an angle that left and right are reversed with respect to the LBCMI plots. Notice the deep gap between the two blocks. This is not only grossly visible in the scans but also tends to distort the overall shape of the scan even though the high frequency noise seems to be reduced in these scans compared to those shown previously. The reason for the reduced noise is probably the height of the blocks which is approximately 240 mm. This puts their surfaces back to the approximate distance from the scanner mirror as the surfaces used in the laboratory experiments resulting in higher return signal.

The profile for a single scan is shown in Figure 42 and the average of 10 scans is shown in Figure 43. Notice that the pavement crack on the left side of the scans is clearly visible. This crack is approximately 10 mm wide and 10mm deep when viewed from the vertical. The problems which the low signal level and large distance swings cause the decoding software are demonstrated by the profile from a single scan shown in Figure 44. This is the only profile out of 20 made of this target which accurately decoded the rise from the floor to the pavement block on the left edge of the scan.

The final set of LBCMI profiles represents a moving platform test with profiles for a concrete surface obtained at 100 mm intervals. A hand shovel was placed in the scan path to demonstrate the effect of varying perturbations along the longitudinal path of the transverse scans. The area profiled is shown in Figure 45. The scans were taken from the top of the area shown in the photograph and moved toward the area at the bottom at 100 mm intervals. The scans show the profile from left to right across the area shown. Figure 46 shows the eleven scans from top to bottom. The profile perturbations caused by the shovel are clearly visible. However, they are distorted by the non-linearity of the scanned profile of the bare concrete.

Even with the signal and noise problems encountered, this set of experimental data clearly demonstrates the feasibility of the LBCMI technology for producing surface profiles. This is especially significant since this is the first such device which has ever been built using this theoretical approach.

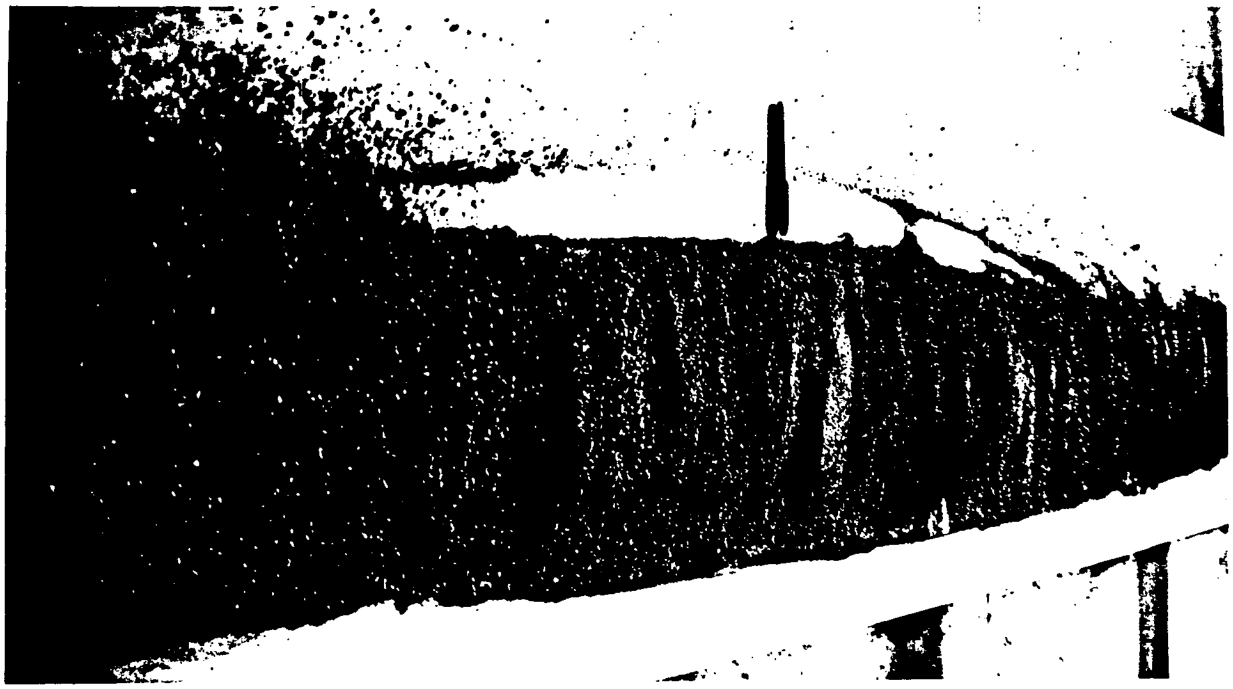


Figure 35. Photograph of rough gravel sample surface.

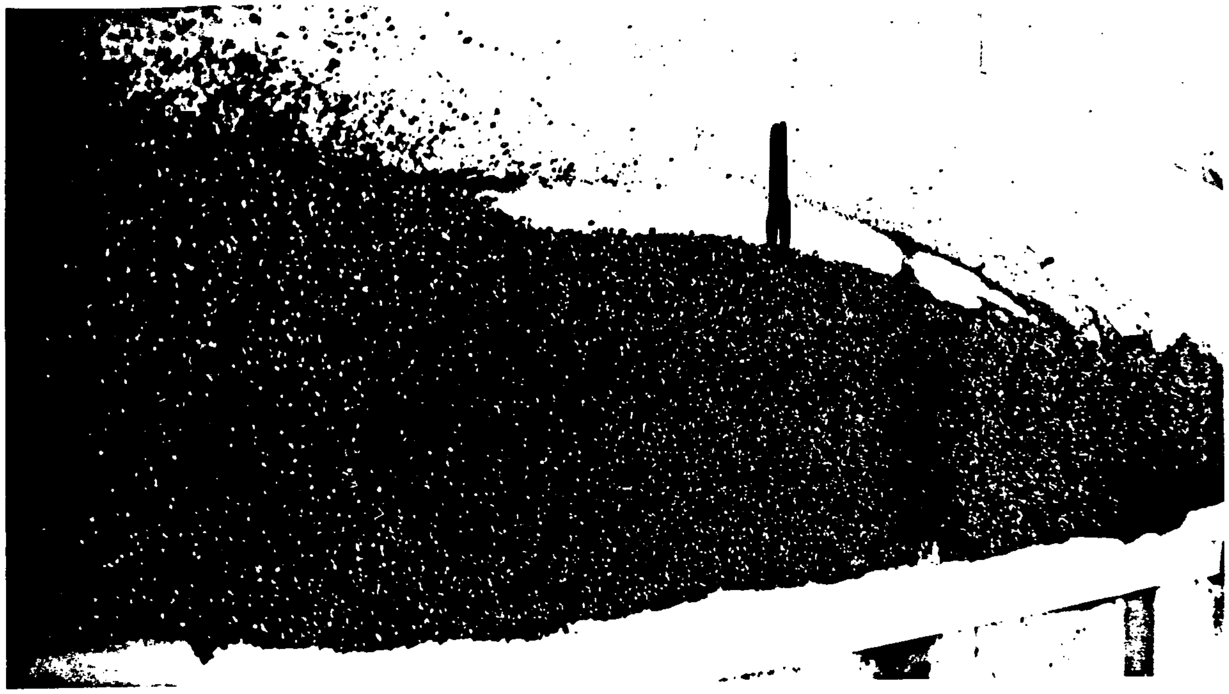


Figure 36. Photograph of smooth gravel sample surface.

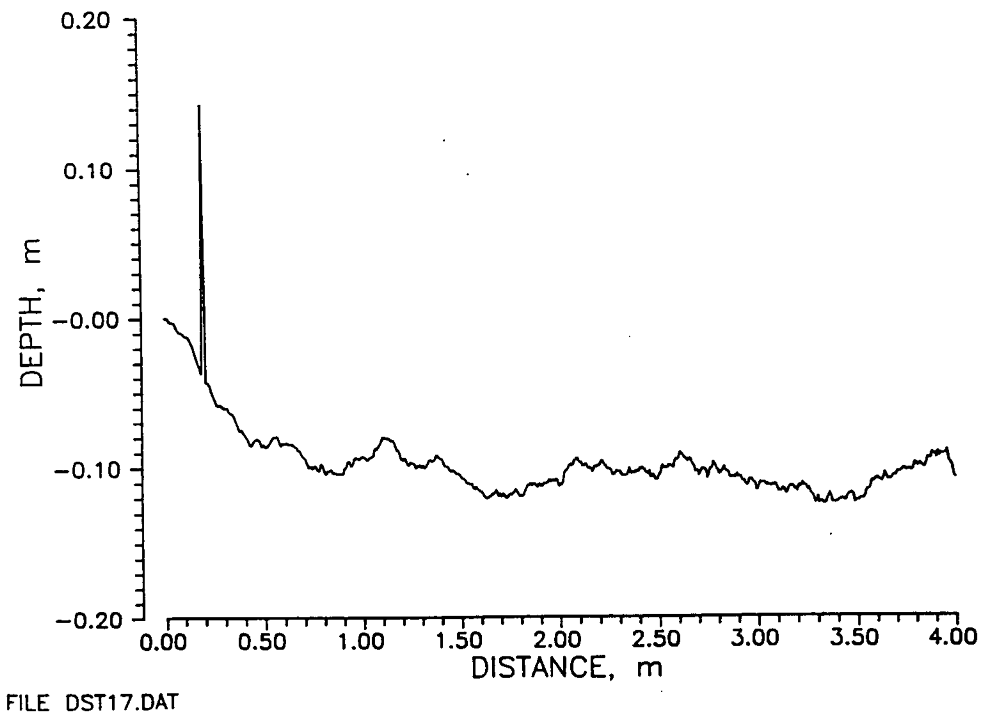


Figure 37. Decoded LBCMI profile from a single scan of rough gravel surface.

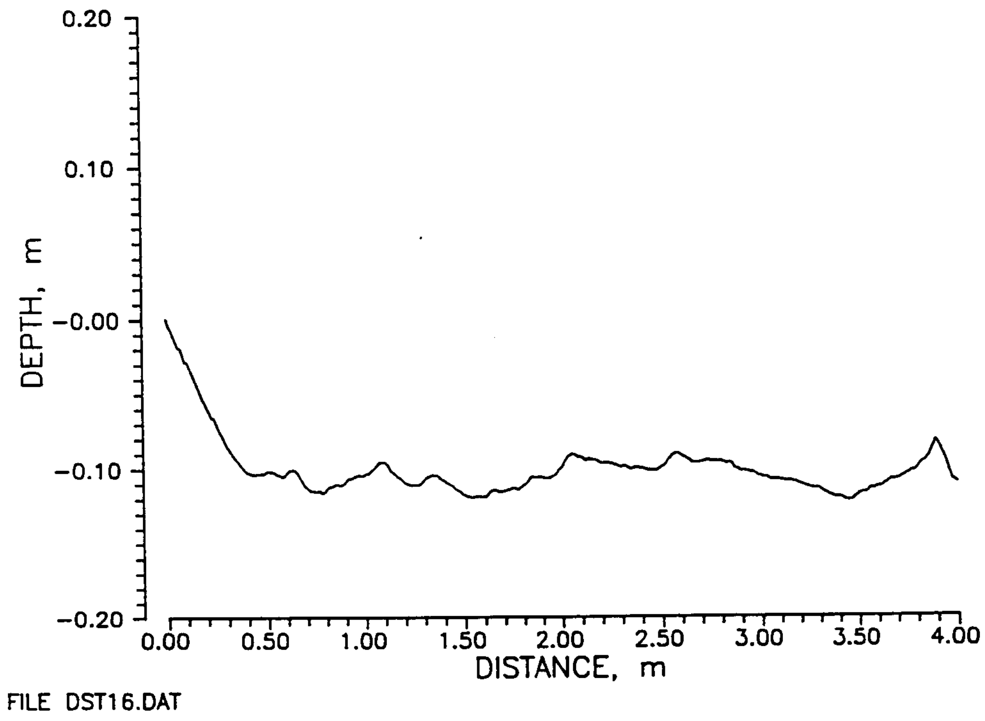


Figure 38. LBCMI profile from average of 20 scans of rough gravel surface.

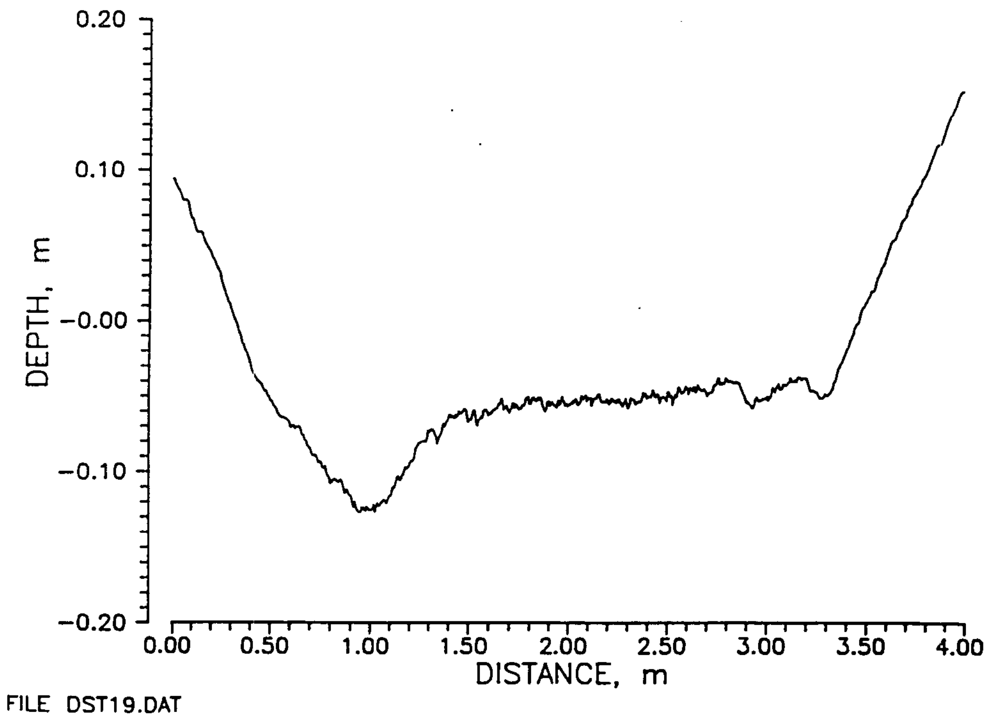


Figure 39. Decoded LBCMI profile from a single scan of smooth gravel surface.

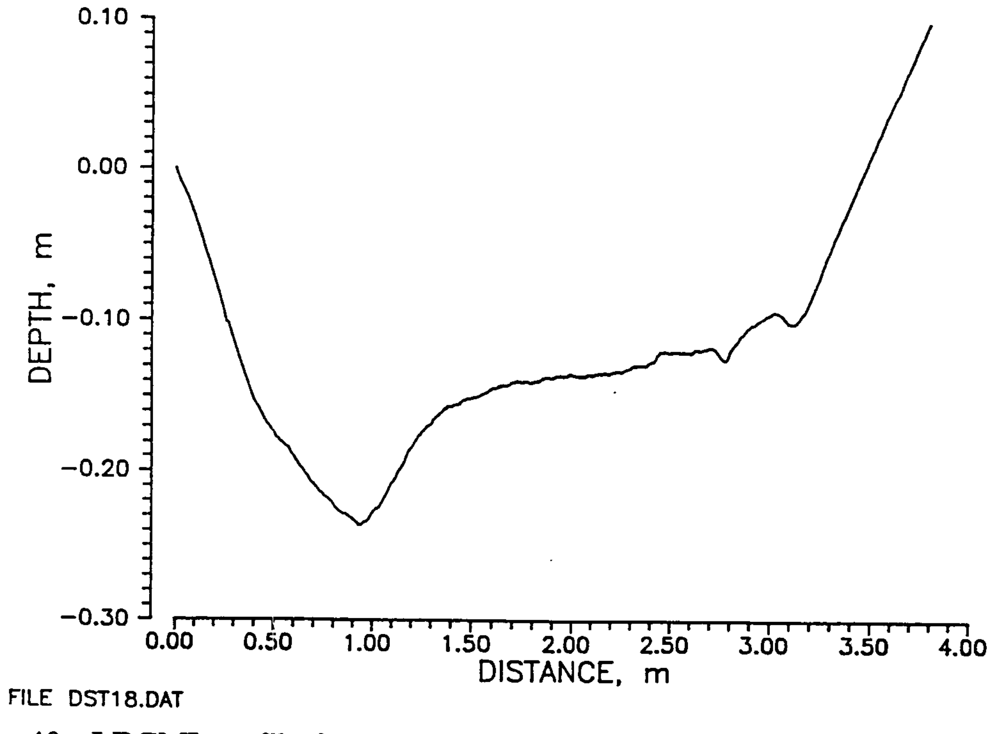


Figure 40. LBCMI profile from average of 20 scans of smooth gravel surface.

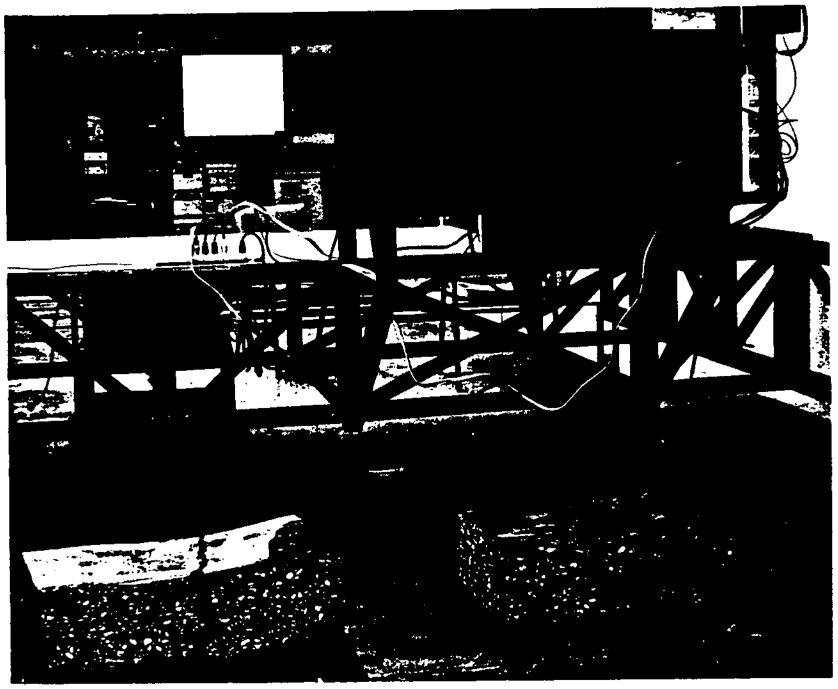


Figure 41. Photograph of pavement blocks removed from old highway.

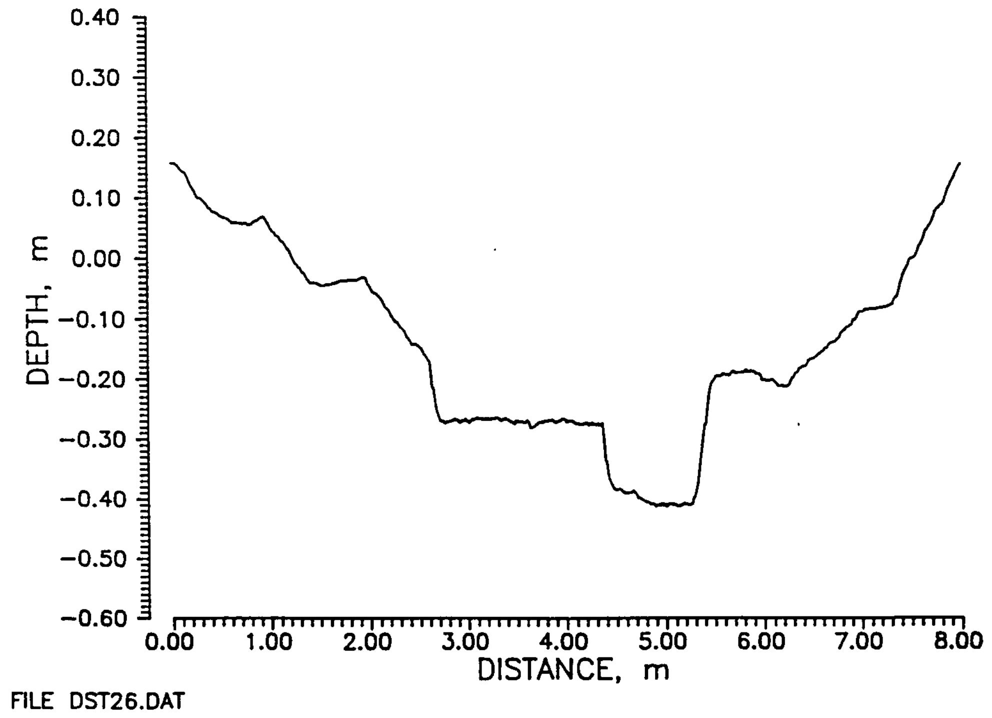


Figure 42. Decoded LBCMI profile from a single scan of pavement blocks.

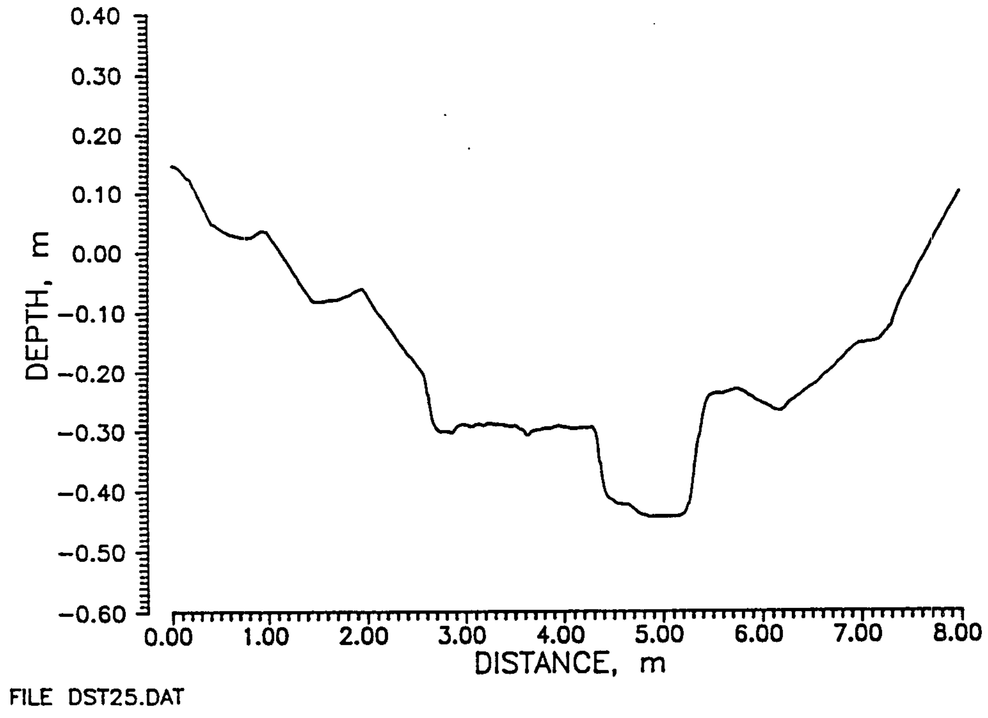


Figure 43. LBCMI profile from average of 10 scans of pavement blocks.

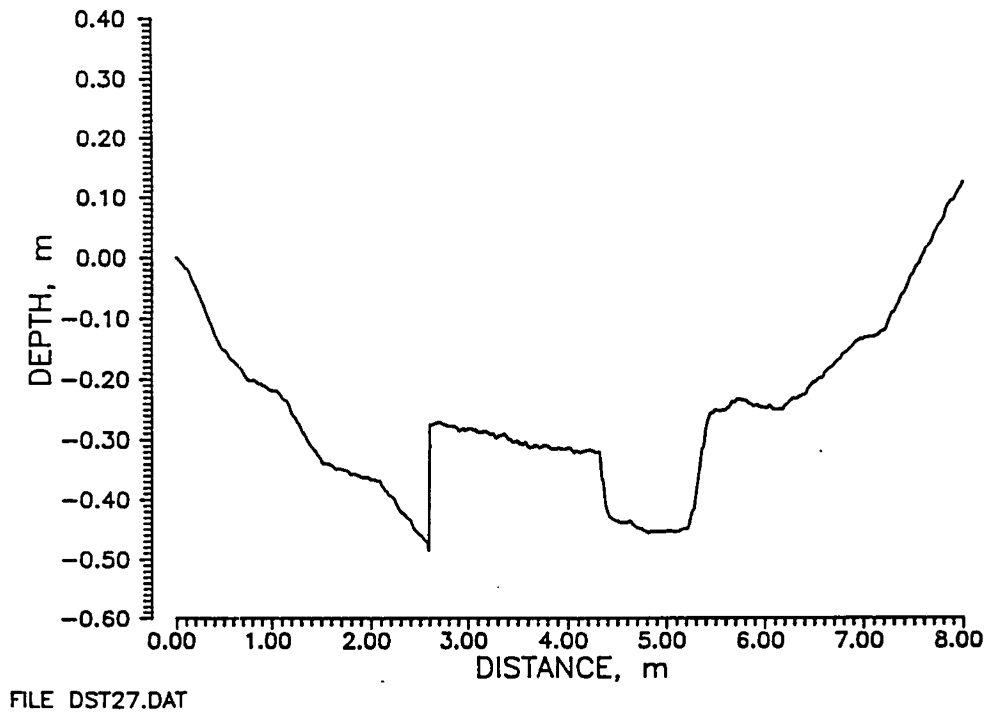


Figure 44. Decoded LBCMI profile from a single scan of pavement blocks with proper decoding of left block edge.

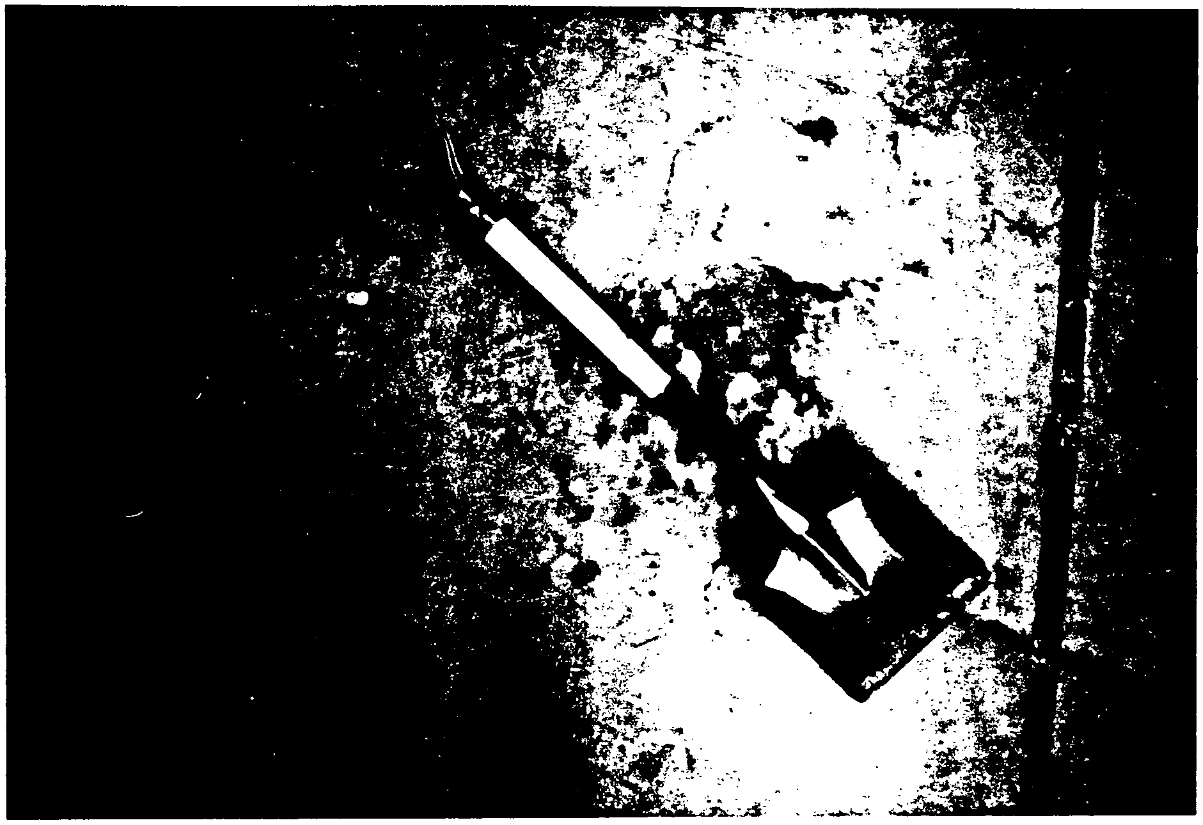


Figure 45. Photograph of area traversed in moving platform scanning test.

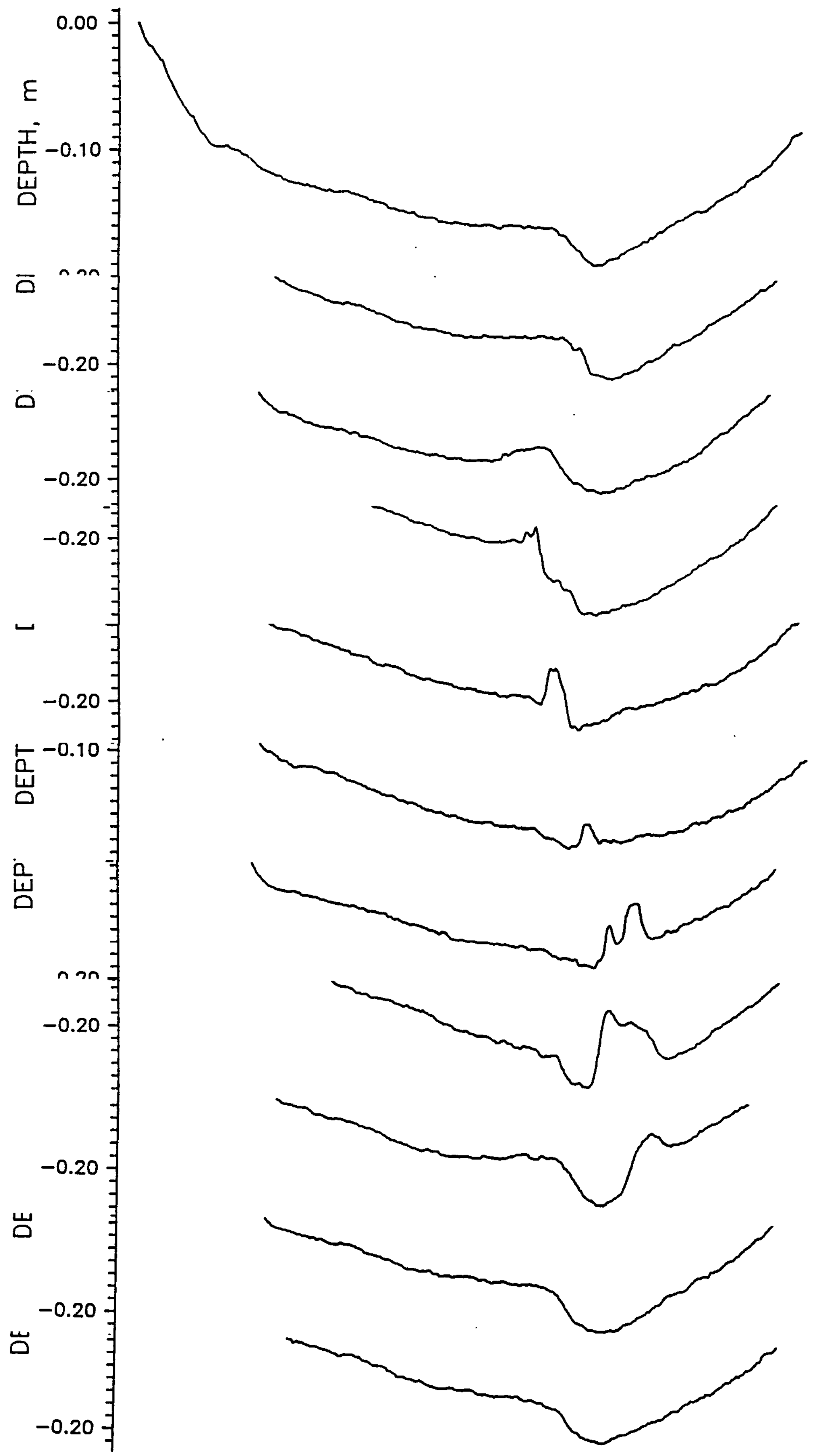


Figure 46. Multiple profiles from moving platform test.

6.0 Conclusions

The results of this research have proven that the laser-beam-carried microwave-interferometer theory will work and indicates that a highway profiler using it could be developed with further work. For the first time a prototype LBCMI system has been designed and constructed, and laboratory scanning tests have been successfully performed using the system. Further, several hundred field scans were produced from a variety of pavement surfaces. Many of these scans were duplicates from the same surface and averages were calculated to simulate the effect of increased signal to noise ratio. These example scans demonstrate the ability of the LBCMI system to produce profile data from road surfaces.

The laboratory scanning experiments were performed at a scanning distance of approximately 1.5 meters with white posterboard as the target surface. With this configuration the LBCMI system produced sufficient reflected light to measure a full 90° scan. Also, the signal to noise ratio was sufficient to produce a properly decoded surface profile. However, when the system was tested in the field at its design height of 2.0 meters while scanning pavement surfaces the reflected signal was only strong enough to produce reliable results for the center 2 meters of the 4 meter scan path. Likewise, because of the reduced signal to noise ratio the decoding algorithm did not perform properly and the apparent surface profile tends to be curved rather than flat.

The conclusion drawn from these tests is that the fundamental limitation of the prototype LBCMI was the lack of a commercially available light detector with a combination of high speed and high sensitivity. For the laboratory prototype system we used the highest performance tube that we had found to be commercially available, a Hamamatsu R1547 photomultiplier tube which has a rise time of 1.4 nanosecond. The lack of a higher speed detector forced the use of a maximum microwave modulation frequency of 300 MHz. This limitation has reduced the accuracy of the measurements.

As a result of this conclusion we contacted the engineers at Hamamatsu whom we had worked with earlier to discuss the possibilities of obtaining a custom version of the R1547 tube with a faster response time. At that point they replied that Hamamatsu had announced a new photomultiplier tube with a rise time of 0.4 nanoseconds. Further, this tube can be enclosed in a cooling chamber to decrease the noise level which will increase the range of operation. This combination will allow LBCMI operation at frequencies up to 500 MHz and will reduce the noise level by approximately three orders of magnitude.

With these detector specifications we believe that the LBCMI system could produce profiles with a 1 mm resolution and a full scan width of 4 meters. Alternatively, two synchronous systems might be used side-by-side to provide two 2 meter scan widths. However, this would also involve additional development to make sure that both systems are truly synchronous and properly calibrated.

The actual implementation of the LBCMI system involved many such engineering problems that were not envisioned in the theoretical concepts used in the development of the proposal. All but this one problem were eventually solved and the current system is functional as initially conceived with the exception of the resolution. The development of an operational instrument would involve additional engineering problems as well. Specifically, the problems associated with road vibrations and vehicle motion will need to be addressed.

With the use of this new sensor technology and further development we believe the transverse profiler prototype which has been tested here could result in an operational system which could be used for data acquisition in all four major areas of interest in the SHRP program: Asphalt, Pavement Performance, Highway Operations, and Concrete and Structures. It is based on technology which is new within highway practice, and offers the potential for significant improvement in both the efficiency and quality of data acquisition and analysis.

Appendix A

Computer Program Listings

```
/* =============================================================== */
/* == program scope.c                                              */
/* == by Nick Ingegneri                                            */
/* ==    University of Nevada, Reno                                */
/* =============================================================== */
/* == Revision 2: 10/17/89                                         */
/* == Combination of channels at end.  Simple norm. of input.      */
/* == Saving of DMA, DST, VLT.                                     */
/* =============================================================== */

#include <stdio.h>
#include <string.h>
#include <dos.h>
#include <graphics.h>
#include <stdlib.h>
#include "dash16.h"
#include <alloc.h>
#include <time.h>
#include <math.h>

int i,maxx,maxy,cx,y1,y2,ys,crs_cnt,crs_flg,crs_time,ss11,ss12,ss13,ss14,
 ss21,ss22,ss23,ss24,sss1,sss2,sss3,sss4,*orgptr,*dmaptr,mode,data[5],
 flag,leftedge,numsamp,rpm;
char g_msg[80],ytext[8][40],ch1lvl[20],ch2lvl[20],chtime[20],chdist[20],
 chaxis[20],oldkhz[10],oldnum[10],oldrpm[10];
long samprate;
float surface[1200],surf[1200][2],xdist[1200];
int best[1200];


init_dma_data()          /* procedure to clear DMA data              */
{
 for (i=0; i<=numsamp; i=i+2) { *(dmaptr+i)=2047; *(dmaptr+i+1)=2047; }
}

init_surf_data()         /* procedure to clear surface data          */
{
 for (i=0; i<=numsamp/2; i=i+1)  surface[i]=0;
}


graph_data()             /* procedure to plot data on screen         */
{
 int x;
 setviewport(0,36,639,263,1);
 clearviewport();
 setviewport(0,0,639,349,1);
 x=0;
 for (i=leftedge*2; (x <= 639) && (i<numsamp); i+=2)
  { putpixel(x,255-(*(dmaptr+i))/19,RED);
    putpixel(x,255-(*(dmaptr+i+1))/19,GREEN);
    /* putpixel(x,148-200*surface[i/2],WHITE); */
    x++; }
 update_channels();
}


update_channels()       /* procedure to display channel levels     */
{
 int sample,ch1,ch2;
 float volt1,volt2,cht;
 setcolor(BLACK);
 outtextxy(406,298,ch1lvl);
 outtextxy(406,308,ch2lvl);
 outtextxy(406,318,chdist);
 outtextxy(406,328,chtime);
 outtextxy(406,338,chaxis);
 sample=(leftedge+cx)*2;
 ch1=*(dmaptr+sample);
 ch2=*(dmaptr+sample+1);
 volt1=(float) (ch1-2047)/2047;
 volt2=(float) (ch2-2047)/2047;
 cht=((float) samprate*sample/1000);
 sprintf(ch1lvl,"%d : %+.3g V",ch1,volt1);
 sprintf(ch2lvl,"%d : %+.3g V",ch2,volt2);
 sprintf(chtime,"%g",cht);
 sprintf(chdist,"%.1f",surface[sample/2]*1000);
 sprintf(chaxis,"%.1f",xdist[sample/2]*1000);
 setcolor(WHITE);
 outtextxy(406,298,ch1lvl);
 outtextxy(406,308,ch2lvl);
 outtextxy(406,318,chdist);
 outtextxy(406,328,chtime);
 outtextxy(406,338,chaxis);
 y1=255-ch1/19;
 y2=255-ch2/19;
 if (sample<numsamp) ys=148-200*surface[sample/2];
  else ys=-1;
}



```

```
find_rpm()                  /* procedure to calculate rpm RATE          */
{
 struct time clock1,clock2;
 int tick1,tick2,ticks;
 move_text();
 outtextxy(10,338,"Measuring RPM...");
 gettime(&clock1);
 for (i=1; i<=100; i++)
  { dodma(2,dmaptr);
    while (backgroundbusy())
    { if (kbhit()!=0) { } } }
 gettime(&clock2);
 tick1=100*clock1.ti_sec+clock1.ti_hund;
 tick2=100*clock2.ti_sec+clock2.ti_hund;
 if (tick2>tick1) ticks=tick2-tick1;
  else ticks=6000+tick2-tick1;
 rpm=75000.0/ticks;
 numsamp=3750000/(samprate*rpm);
 setcolor(BLACK);
 outtextxy(10,338,"Measuring RPM...");
 setcolor(WHITE);
 sprintf(ytext[7],"RPM rate = %d",rpm);
 update_text();
 update_status();
}



do_dma()                    /* procedure to do a DMA scan into memory  */
{
 move_text();
 dodma(numsamp,dmaptr);
 while (backgroundbusy())
  { outtextxy(10,338,"Executing DMA scan...");
    if (kbhit() != 0) { }; }
 setcolor(BLACK);
 outtextxy(10,338,"Executing DMA scan...");
 setcolor(WHITE);
 sprintf(ytext[7]," %d samples taken",numsamp);
 update_text();
 leftedge=0; cx=numsamp/4;
 shiftdata(numsamp,dmaptr);
}

wipe_bottom()                         /* wipes the bottom text line clean*/
{
 setviewport(10,338,299,346,1);
 clearviewport();
 setviewport(0,0,639,349,1);
}

simple_normal()
{
 int wave;
 float hi,lo,gnd;

 wipe_bottom();
 sprintf(ytext[7]," Peak normalizing voltages...");
 update_text();
 for (wave=0; wave<=1; wave++)
  { hi=0; lo=0;
    for (i=0; i<numsamp/2; i++)
     { if (surf[i][wave]<lo)  lo=surf[i][wave];
       if (surf[i][wave]>hi)  hi=surf[i][wave]; }
    gnd=(hi+lo)/2.0;
    for (i=0; i<numsamp/2; i++)
     surf[i][wave]=surf[i][wave]-gnd;
    hi=hi-gnd; lo=lo-gnd; /* should be equal with opposite signs   */
    for (i=0; i<numsamp/2; i++)
     { surf[i][wave]=surf[i][wave]/hi;
       putpixel(i,148-108*surf[i][wave],4-2*wave); }
  } /* end of wave loop */
}



normalize_input()
{
 int wave,mode,peak[10],peaknum;
 float pslope,nslope,pbase,nbase,peaklvl,peakv[4];

 wipe_bottom();
 sprintf(ytext[7]," Normalizing input...");
 update_text();
 for (wave=0; wave<=1; wave++)
  { if ( *(dmaptr+wave)<2047 ) mode=0;
    else mode=1;
    peaklvl=0; peaknum=0;
    for (i=0; i<numsamp/2; i++)
     { if (mode==0 && surf[i][wave]> 0.165) mode=2;
       if (mode==1 && surf[i][wave]< -.165) mode=3;
       if (mode==2)
```

```
        { if (surf[i][wave]>peaklvl)
          { peak[peaknum]=i;
            peaklvl=surf[i][wave]; }
          if (surf[i][wave]< 0.07)
          { if (peak[peaknum]>10)
    { putpixel(peak[peaknum],148-108*surf[peak[peaknum]][wave],WHITE);
              peaknum++; }
            mode=1; } }
       if (mode==3)
        { if (surf[i][wave]<peaklvl)
          { peak[peaknum]=i;
            peaklvl=surf[i][wave]; }
          if (surf[i][wave]> -.07)
          { if (peak[peaknum]>10)
    { putpixel(peak[peaknum],148-108*surf[peak[peaknum]][wave],WHITE);
              peaknum++; }
            mode=0; } }
    } /* end of i loop */
    if (peak[peaknum]<numsamp/2-10)
    { putpixel(peak[peaknum],148-108*surf[peak[peaknum]][wave],WHITE);
      peaknum++; }

    peak[2]=peak[peaknum-2];
    peak[3]=peak[peaknum-1];
    for (i=0; i<4; i++)  { peakv[i]=surf[peak[i]][wave];
                           if (peakv[i]<0) peakv[i]=-peakv[i]; }
    pslope=(peakv[3]-peakv[0])/(peak[3]-peak[0]);
    pbase=peakv[0]-pslope*peak[0];
    nslope=(peakv[2]-peakv[1])/(peak[2]-peak[1]);
    nbase=peakv[1]-nslope*peak[1];
 for(i=0;i<numsamp/2;i++){putpixel(i,148-108*(pbase+pslope*i),4-2*wave);
                          putpixel(i,148+108*(nbase+nslope*i),4-2*wave);
                                if (surf[i][wave]>0)
                           surf[i][wave]=surf[i][wave]/(pbase+pslope*i);
                                else
                           surf[i][wave]=surf[i][wave]/(nbase+nslope*i);
    if (surf[i][wave]>1.0) surf[i][wave]=1.0;       /* truncate the*/
    if (surf[i][wave]<-1.0) surf[i][wave]=-1.0;     /* input voltage*/
    putpixel(i,148-108*surf[i][wave],4-2*wave);
    }
 } /* end of wave loop */
}


restore_input()           /* this procedure converts the normalized */
{                               /* voltages into depth information  */
```

```
 int wave,offset;
 float shift0,shift1,dist0,dist1,good0,good1,old;

 wipe_bottom();
 sprintf(ytext[7]," Restoring distances...");
 update_text();

 for (i=0; i<numsamp/2; i++)
  { shift0=asin(surf[i][0]) / (2*M_PI) ;  /* find the phase shift  */
    shift1=acos(surf[i][1]) / (2*M_PI) ;  /* distance in meters    */

    if (shift0<0) good0=-shift0*4;                /* find goodness   */
     else good0=shift0*4;                         /* of the samples: */
    if (shift1<0.25) good1=(0.25-shift1)*4;       /*  0=linear       */
     else good1=(shift1-0.25)*4;                  /*  1=in peak      */

    dist0=shift0;  dist1=shift1;                  /* recreate the     */
    if (shift0>0 && shift1>0.25)   dist0=0.5-shift0; /* original data */
    if (shift0<0 && shift1>0.25) { dist0=0.5-shift0; /* from the trig */
                                   dist1=1.0-shift1;}/* data given by */
    if (shift0<0 && shift1<0.25) { dist0=1.0+shift0; /* sin, cos of  */
                                   dist1=1.0-shift1;}/* phase shift   */

    if (good0<good1) best[i]=0;                   /* pick the best   */
     else best[i]=1;                              /* data channel    */

    surf[i][0]=dist0;                             /* save the        */
    surf[i][1]=dist1;                             /* restored values */
  }

  for (wave=0; wave<2; wave++)                    /* eliminate the   */
  { offset=0; old=0.5;                            /* 1 meter jumps   */
    for (i=0; i<numsamp/2; i++)                   /* in the restored */
    { if ((old-surf[i][wave]) > 0.75) offset++;   /* distance.       */
      if ((old-surf[i][wave]) < -0.75) offset--;  /*                 */
      old=surf[i][wave];                          /* caused by the   */
      surf[i][wave]=(surf[i][wave]+offset)/2;     /* one m modolo.   */
      putpixel(i,148-100*(surf[i][wave]),4-2*wave); } }
}


decode()
{
 int wave,i,pick;
 float rad,avg,base,peak,splice;

 move_text();
 sprintf(ytext[7],"Decoding %d sample pairs...",numsamp/2);
 update_text();
 move_text();

 for (i=0; i<numsamp/2; i++)                 /* load data into array     */
  { surf[i][0]=(float) (*(dmaptr+2*i)-2047)/2047;
    surf[i][1]=(float) (*(dmaptr+2*i+1)-2047)/2047;}

/* normalize_input(); */                     /* scale to +-1v, no slope  */
 simple_normal();                            /* non-envelope checking    */
 restore_input();                            /* convert to distance info */

 for (wave=0; wave<2; wave++)
 { wipe_bottom();
   sprintf(ytext[7]," Removing sweep curve #%d...",wave);
   update_text();
   base=0;                                   /* lowest point             */
   peak=0;                                   /* highest point            */

  for (i=0; i<numsamp/2; i++) if (surf[i][wave]<base) base=surf[i][wave];
   for (i=0; i<numsamp/2; i++) { surf[i][wave]=surf[i][wave]-base;
                                 if (surf[i][wave]>peak)
                                  peak=surf[i][wave]; }
   base=peak/(M_SQRT2-1);                /* (1/sin(45)) -1 */
   wipe_bottom();
   sprintf(ytext[7]," Base distance #%d = %.2f m",wave,base);
   update_text();

   /* revalue the distances to include the surface distance */
   for (i=0; i<numsamp/2; i++) surf[i][wave]=surf[i][wave]+base;


   /* revalue the distances to cancel out the sine caused by sweeping */
   for (i=0; i<numsamp/2; i++)
    { rad=M_PI_2*i/(numsamp/2)+M_PI_4;

      rad = rad /2;
      surf[i][wave]=surf[i][wave]*sin((double)rad);
      xdist[i]=(base-base/tan((double)rad))/0.75; }

      /* find overall average */
      avg=0;
      for (i=0; i<numsamp/2; i++) avg=avg+surf[i][wave];
      avg=avg/(numsamp/2);

      /* center surface relative to average */
      /* and correct for mirror sweep (if .00 restored to .05) */
      for (i=0; i<numsamp/2; i++)
      { surf[i][wave]=surf[i][wave]-avg-i*(.00/(numsamp/2));
        putpixel(i,148-200*(int)surf[i][wave],20-wave*2); }

 }                              /* end of wave loop                    */

 wipe_bottom();
 strcpy(ytext[7]," Building scan composite...");
 update_text();
 pick=best[i]; splice=0;
 for (i=0; i<numsamp/2; i++)
  { if (pick!=best[i]) { pick=best[i];
                         splice=surf[i][pick]-surface[i-1]; }
    surface[i]=surf[i][pick]-splice;
    putpixel(i,148-200*surface[i],WHITE); }
 wipe_bottom();
 strcpy(ytext[7]," Decoding complete");
 update_text();
}


int dumpdata()          /* procedure to output data file              */
{
  FILE *datfile;
  char filnam[80];

  sprintf(filnam,"DMA%s",g_msg);
  if ((datfile = fopen(filnam,"w")) == NULL)  return(1);
  fprintf(datfile,"%5d  %5d\n",rpm,samprate);
  for (i=0; i < numsamp; i=i+2)
   fprintf(datfile,"%5d  %5d\n",*(dmaptr+i),*(dmaptr+i+1));
  if (fclose(datfile)!=0) return(1);
  sprintf(ytext[7]," DMA%s saved",g_msg);
  update_text();
  move_text();

  sprintf(filnam,"VLT%s",g_msg);
  if ((datfile = fopen(filnam,"w")) == NULL) return(1);
  for (i=0; i<numsamp; i=i+2)
   fprintf(datfile,"%5f %5f %5f\n",xdist[i/2],(*(dmaptr+i)-2047)/2047.0,
    (*(dmaptr+i+1)-2047)/2047.0);
  if (fclose(datfile)!=0) return(1);
  sprintf(ytext[7]," VLT%s saved",g_msg);
  update_text();
  move_text();

   sprintf(filnam,"DST%s",g_msg);
  if ((datfile = fopen(filnam,"w")) == NULL) return(1);
  for (i=0; i<numsamp; i=i+2)
   fprintf(datfile,"%5f  %5f\n",xdist[i/2],surface[i/2]);
  if (fclose(datfile)!=0) return(1);
  sprintf(ytext[7]," DST%s saved",g_msg);
  update_text();
  move_text();
}


int loaddata()          /* procedure to input data file            */
{
 FILE *datfile;
 if ((datfile = fopen(g_msg,"r")) == NULL)  return(1);
 fscanf(datfile,"%5d  %5d",&rpm,&samprate);
 for (i=0; feof(datfile)==0; i=i+2)
  fscanf(datfile,"%5d  %5d",dmaptr+i,dmaptr+i+1);
 numsamp=i-2;
 leftedge=0; cx=numsamp/4;
 if (fclose(datfile)==0) return(0);
  else return(1);
}


atexit_t quit(void)     /* procedure called when ending            */
{
 closegraph();
 stopbackground();
 printf("\n %s \n",g_msg);
}

int controlbreak(void)  /* procedure called to handle ctrl-break   */
{
 sprintf(g_msg,"\nUser Break\nProgram aborted");
 exit(1);
}


/* ============================================================ */
/* ==  main program begins here                                 */
/* ============================================================ */
main()
{
 int g_dr,g_mo;
 g_dr = EGA;                           /* EGA graphic driver        */
 g_mo = EGAHI;                         /* EGAHI graphic mode        */
 registerbgidriver(EGAVGA_driver);     /* include EGAVGA_driver     */
 initgraph(&g_dr,&g_mo,"");            /* initialize graphics       */

 atexit((void(*)(void))quit);          /* define exit function      */
 ctrlbrk(controlbreak);

 maxx=getmaxx();
 maxy=getmaxy();
 cx=maxx/2;                            /* start cursor in center    */

 for (i=0; i<8; i++)                   /* clear output strings      */
  strcpy(ytext[i],"");
 strcpy(ch1lvl,"");
 strcpy(ch2lvl,"");

 setupdash16(&numsamp,&samprate);      /* loads initial values      */
 orgptr=farmalloc(1);                  /* finds place for DMA       */
 dmaptr=setuppointer(orgptr,32766)     /* prepares DMA pointer      */
 rpm=0;                                /* no rpm to start           */

 set_scrn();                           /* draw screen               */
 update_status();                      /* display initial text      */
 leftedge=0;                           /* scrolling is at zero      */
 init_dma_data();                      /* clear out dma data        */
 init_surf_data();                     /* clear out surface data    */
 graph_data();                         /* draw initial waveforms    */
 crs_time=500;                         /* time for cursor blink     */
 crs_on();                             /* initialize cursor         */

/* ==  central program loop                                      == */
 do
 {
  crs_cnt=crs_cnt-1;                   /* decrement cursor count    */
  if (crs_cnt==0) crs_blk();           /* is it time to blink?      */
  if (kbhit()!=0) cmd_proc();          /* check keyboard            */
 }
 while (1);                            /* repeat indefinitely       */

}

```

```
/*==============================================================*/
/*== program screen.c                                          */
/*== by Calvin Taylor                                          */
/*==    University of Nevada, Reno                             */
/*==============================================================*/
/*== Revision 2:  5/20/89                                      */
/*==============================================================*/

#include <graphics.h>

extern int maxx,maxy,cx,y1,y2,ys,crs_cnt,crs_flg,crs_time,ss11,ss12,ss13,
ss14,ss21,ss22,ss23,ss24,sss1,sss2,sss3,sss4,mode,data[5],flag,leftedge,
 numsamp,rpm;
extern char g_msg[80],ytext[8][40],oldkhz[10],
 oldnum[10],oldrpm[10];
extern long samprate;

crs_on()                  /* procedure to turn on graphics cursor   */
{
 crs_flg = 1;
 crs_cnt = crs_time;
 ss11 = getpixel(cx-1,y1);
 ss12 = getpixel(cx+1,y1);
 ss13 = getpixel(cx,y1-1);
 ss14 = getpixel(cx,y1+1);
 ss21 = getpixel(cx-1,y2);
 ss22 = getpixel(cx+1,y2);
 ss23 = getpixel(cx,y2-1);
 ss24 = getpixel(cx,y2+1);
 sss1 = getpixel(cx-1,ys);
 sss2 = getpixel(cx+1,ys);
 sss3 = getpixel(cx,ys-1);
 sss4 = getpixel(cx,ys+1);
 putpixel(cx-1,y1,LIGHTRED);
 putpixel(cx+1,y1,LIGHTRED);
 putpixel(cx,y1-1,LIGHTRED);
 putpixel(cx,y1+1,LIGHTRED);
 putpixel(cx-1,y2,LIGHTGREEN);
 putpixel(cx+1,y2,LIGHTGREEN);
 putpixel(cx,y2-1,LIGHTGREEN);
 putpixel(cx,y2+1,LIGHTGREEN);
 putpixel(cx-1,ys,WHITE);
 putpixel(cx+1,ys,WHITE);
 putpixel(cx,ys-1,WHITE);
 putpixel(cx,ys+1,WHITE);
```

```
}


crs_off()                  /* procedure to turn off graphics cursor  */
{
 crs_flg = 0;
 crs_cnt = crs_time;
 putpixel(cx-1,y1,ss11);
 putpixel(cx+1,y1,ss12);
 putpixel(cx,y1-1,ss13);
 putpixel(cx,y1+1,ss14);
 putpixel(cx-1,y2,ss21);
 putpixel(cx+1,y2,ss22);
 putpixel(cx,y2-1,ss23);
 putpixel(cx,y2+1,ss24);
 putpixel(cx-1,ys,sss1);
 putpixel(cx+1,ys,sss2);
 putpixel(cx,ys-1,sss3);
 putpixel(cx,ys+1,sss4);
}


crs_blk()                  /* procedure to blink graphics cursor     */
{
 crs_cnt = crs_time;
 if (crs_flg==0) crs_on(); else crs_off();
}


set_scrn()                 /* procedure to draw screen               */
{
 cleardevice();
 setcolor(WHITE);
 outtextxy(3,8,"F1: Do DMA scan");
 outtextxy(3,23,"F2: Decode DMA scan");
 outtextxy(216,8,"F3: Change sampling rate");
 outtextxy(216,23,"F4: Change # of samples");
 outtextxy(429,8,"F5: Output to text file");
 outtextxy(429,23,"F6: Input from text file");
 outtextxy(310,268,"Sampling rate (kHz) :");
 outtextxy(310,278,"Number of samples   :");
 outtextxy(310,288,"Rotations / minute  :");
 setcolor(RED);  outtextxy(310,298,"Channel 1 :");
 setcolor(GREEN);  outtextxy(310,308,"Channel 2 :");
 setcolor(WHITE); outtextxy(310,318,"Dist (mm) :");
 setcolor(WHITE); outtextxy(310,328,"Time (ms) :");
 setcolor(WHITE); outtextxy(310,338,"Axis (mm) :");
 line(0,35,maxx,35);
 rectangle(0,maxy-85,maxx,maxy);
 line(300,maxy-85,300,maxy);
}


update_text()           /* procedure to update text windows       */
{
 int i;
 for (i=0; i<8; i++)
  outtextxy(10,268+i*10,ytext[i]);
}


update_status()         /* procedure to update status window      */
{
 setcolor(BLACK);
 outtextxy(486,268,oldkhz);
 outtextxy(486,278,oldnum);
 outtextxy(486,288,oldrpm);
 setcolor(WHITE);
 sprintf(oldkhz,"%ld",1000/samprate);
 sprintf(oldnum,"%d",numsamp);
 sprintf(oldrpm,"%d",rpm);
 outtextxy(486,268,oldkhz);
 outtextxy(486,278,oldnum);
 outtextxy(486,288,oldrpm);
}


move_text()             /* procedure to scroll text in window     */
{
 int i;
 for(i=0; i<7; i++)
  strcpy(ytext[i],ytext[i+1]);
 strcpy(ytext[7],"");
 setcolor(WHITE);
 setviewport(1,maxy-84,299,maxy-1,1);
 clearviewport();
 setviewport(0,0,639,349,1);
 update_text();
}

read_line()       /* procedure to input a number to text line 5   */
{
 int i,key;
 move_text();
 do { key = getch();
      i=strlen(ytext[7]);
      if ((key>=48) && (key<=57) && (i<=10))  { ytext[7][i] = key;
                                                ytext[7][i+1] = 0;
                                                update_text(); }
      if ((key==8) && (i>=1))  { i--;
                                 setcolor(BLACK);
                                 outtextxy(10,338,ytext[7]);
                                 ytext[7][i] = 0;
                                 setcolor(WHITE);
                                 outtextxy(10,338,ytext[7]); }
    } while (key!=13);
 setcolor(BLACK);
 outtextxy(10,338,ytext[7]);
 setcolor(WHITE);
}


change_rate()      /* procedure to change sampling rate            */
{
 move_text();
 strcpy(ytext[7],"Input new sampling rate (kHz):");
 update_text();
 read_line();
 sscanf(ytext[7],"%ld",&samprate);
 samprate=1000/samprate;          /* convert kHz to rate   */
 if (samprate<12) samprate=12;    /* 12 is the fastest rate*/
 setsamplerate(samprate);
 sprintf(ytext[7]," Sampling rate = %ld kHz",1000/samprate);
 update_text();
 update_status();
}


change_numsamp()        /* procedure to change number of samples  */
{
 move_text();
 strcpy(ytext[7],"This command will soon be removed...");
 update_text();
 move_text();
 strcpy(ytext[7],"Input new number of samples:");
 update_text();
 read_line();
 sscanf(ytext[7],"%d",&numsamp);
 numsamp=(numsamp/2)*2;
 if (numsamp > 5000)  numsamp=5000;
 sprintf(ytext[7]," Number of samples = %d",numsamp);
 update_text();
 update_status();
}


cmd_proc()               /* command processor to handle input      */
{
 int ALC,ALF;
 ALC = getch();                                /* get keypress          */
 if (ALC==0) ALF=getch(); else ALF=0;         /* maybe get function key*/

 if (ALC==27)                                  /* ESC handler           */
  { strcpy(g_msg,"Program ended through ESC");
    exit(1); }

 if ((ALF==75) && (cx>0))                      /* move cursor left      */
  { crs_off();
    cx--;
    update_channels();
    crs_on(); }

 if (ALC==52)                                  /* big move left         */
  { crs_off();
    cx=cx-10;
    if (cx<0) cx=0;
    update_channels();
    crs_on(); }

 if (ALF==115)                                 /* scroll screen left    */
  { crs_off();
    leftedge=leftedge-40;
    if (leftedge<0) leftedge=0;
    graph_data();
    crs_on(); }

 if ((ALF==77) && (cx<maxx))                   /* move cursor right     */
  { crs_off();
    cx++;
    if (cx>numsamp/2-1)  cx=numsamp/2-1;
    update_channels();
    crs_on(); }

 if (ALC==54)                                  /* big move right        */
 { crs_off();
   cx=cx+10;
   if (cx>maxx) cx=maxx;
   if (cx>numsamp/2-1) cx=numsamp/2-1;
   update_channels();
   crs_on(); }

if (ALF==116)                            /* scroll screen right   */
 { crs_off();
   leftedge=leftedge+40;
   if (leftedge>(numsamp/2)-640) leftedge=(numsamp/2)-640;
   if (leftedge<0) leftedge=0;
   graph_data();
   crs_on(); }

if (ALF=='G')                            /* move screen to start  */
 { crs_off();
   cx=0;
   leftedge=0;
   graph_data();
   crs_on(); }

if (ALF=='O')                            /* move screen to end    */
 { crs_off();
   cx=639;
   leftedge = (numsamp/2)-640;
   if (leftedge<0) { leftedge=0; cx=numsamp/2-1; }
   graph_data();
   crs_on(); }

if (ALF==63)                             /* output data to file    */
 { move_text();
   strcpy(ytext[7],"Output file number:");
   update_text();
   read_line();
   if (ytext[7][0] != 0)
    { sprintf(g_msg,"%s.DAT",ytext[7]);
      if (dumpdata() != 0)
       { setcolor(RED);
         strcpy(ytext[7]," Error in file operation");
         outtextxy(10,338,ytext[7]);
         setcolor(WHITE); }
      else
       { strcpy(ytext[7]," Save complete");
         update_text(); }
    } else { strcpy(ytext[7]," Output canceled");
```

```
                update_text(); }
  }

if (ALF==64)                                /* input data from file  */
 { move_text();
   strcpy(ytext[7],"Input file number:");
   update_text();
   read_line();
   crs_off();
   if (ytext[7][0] != 0)
    { sprintf(g_msg,"DMA%s.DAT",ytext[7]);
      if (loaddata() != 0)
       { setcolor(RED);
         strcpy(ytext[7]," Error in file operation");
         outtextxy(10,338,ytext[7]);
         setcolor(WHITE); }
      else
       { sprintf(ytext[7]," File %s loaded",g_msg);
         update_text();
         init_surf_data(); }
    } else { strcpy(ytext[7]," Input canceled");
                update_text(); }
   graph_data();
   update_status();
   crs_on();
 }

if (ALF==59)                                /* do a DMA scan         */
 { crs_off();
   find_rpm();
   do_dma();
   init_surf_data();
   graph_data();
   update_status();
   crs_on(); }

if (ALF==60)                                /* decode DMA scan       */
 { crs_off();
   decode();
   /* graph_data(); */
   crs_on(); }

if (ALF==61)                                /* change sampling rate  */
 change_rate();

if (ALF==62)                                /* change sample number  */
```

```
   change_numsamp();

}                                              /* end of cmd_proc()     */


```

```
/*================================================================*/
/*== program calldash.c                                           */
/*== by Mike Hugo                                                 */
/*==    University of Nevada, Reno                                */
/*================================================================*/
/*== Revision 2:  8/29/89                                         */
/*================================================================*/

/* routines to call the dash16 object module */
#include <stdio.h>
#include <dos.h>
#include "dash16.h"
#include <stdlib.h>
#include <alloc.h>
#include <graphics.h>

#define NumChannels 2   /*numbers of input channels */
                        /* printdata should be fixed if NumChanels
                           is changed all other routines should be ok */
extern int mode,/* global variables originally defined in main module */
           data[5],
           flag;

extern char g_msg[80];  /* global variable for error message output */


nfactor(rate,pf1,pf2)   /* Nick's factor routine                    */
int *pf1,*pf2;
long rate;
{
 long x,y,factor;
 x=1; y=1;
 if (rate==5) rate=4;
 for (factor=2; ((factor<=5) && (x==1)); factor++)
  { if ( rate==(rate/factor)*factor )  { x=factor;
                                         y=rate/factor; } }
 if (x==1) { rate--;
             x=2;
             y=rate/2; }

 *pf1=x;
 *pf2=y;
}

void setsamplerate(long samplerate)
```

```
/*
 calls the dash16 driver to set the sample rate of the timer
 before calling the dash driver samplerate is converted into two factors.
 sample rate in hz = 1,000,000/ samplerate

 from 250khz to 0.83 samples per hour

 errors cause this routine stop execution of the program
 only error is the driver not initilized (error code 1)

*/
{
  nfactor(samplerate,&data[0],&data[1]);    /* convert to factors */
  mode = 17;    /*set sample rate call number*/
                /*sample rate in hz is 1,000,000/ (data[0] * data[1])*/
  dash16(&mode,&data[0],&flag);
  if (flag != 0)
    {sprintf(g_msg,"Error.\nError %d\n%d\n%d",flag,data[0],data[1]);
      exit(1);
    }
}

void setupdash16(int *sampleptr, long *samplerateptr)
 /* sets up the dash16 board */
 /* sampleptr = number of samples to perform. Only a return value.
   setsamplerate is called in this function

   routine calls getdash16info which opens the dash16.adr file and reads
   defalt values.  Mode 0 is then called that sets up the dash16 board
   mode 1 is next called that causes all input to come from only one of
   the dash16 board's input.

   Errors cause the program is abort after printing error messages.
   See the dash16 manual page 24-26 for additional information.
*/
{
  int channel;    /* channel number used on dash board */
  getdash16info(&data[0],&channel,samplerateptr,sampleptr);
                                  /*read initilizing parameter*/
  mode = 0;                       /*call driver to set up board*/
  dash16(&mode,&data[0],&flag);
  if (flag != 0)
    { sprintf(g_msg,"Error in initilizing Dash16 board.\nError %d",flag);
      exit(1);                    /*abort program error level 1*/
    }

  mode = 1;                        /*set channel to use for input*/
  data[0] = channel;
  data[1] = channel+NumChannels-1;
  dash16(&mode,&data[0],&flag);
  if (flag != 0)
    {  sprintf(g_msg,"Error in setting channel number.\nError %d",flag);
     exit(1);
    }

  setsamplerate(*samplerateptr);
}

int *setuppointer(int *originalpointer, int maxsamples)
/*allocatates memory for D to A conversions.
Sets up the returned pointer--upon entry originalpointer should be
set up to point to memory with some call to faralloc.  This routine
normalizes the pointer then calculates how much memory is required to
allocate to start on the next 64k block and use maxsamples worth of data
it allocates that memory and returns a pointer to the start of the the
64k block.  The memory before the 64k block start is wasted memory
*/
{
int *pointer;
long allocated;

  if (originalpointer == NULL)         /* check for error*/
   {
     sprintf(g_msg,"Not enough memory  Program aborted.\n");
     exit(1);
   }
  farfree(originalpointer);            /* dealocated memory */
                                       /* normilize the pointer  */
  pointer = MK_FP(FP_SEG(originalpointer) +
         (FP_OFF(originalpointer) >> 4),(FP_OFF(originalpointer) & 15));
  allocated = maxsamples;              /* calculate memory required */
  allocated = allocated*2 +
        (0x1000 - (FP_SEG(pointer) & 0xfff))*0x10 - FP_OFF(pointer) + 2;

  pointer = (int *) farmalloc(allocated); /* allocate hunk of memory */
  if (pointer == NULL)         /* check for error*/
   {
     sprintf(g_msg,"Not enough memory.  Program aborted.\n");
     exit(1);
   }                       /*normalize the pointer*/

  return(MK_FP((FP_SEG(pointer) & 0xf000)+0x1000,0));
  /*return pointer pointing at start of next 64k block*/
}

void shiftdata(int samples, int *pointer)
/* does a similer function as mode 9 does.
   converts the data that dash16 board put into memory into a standard
   integer.  Program assumes that unipolar conversion is being done.
   Dash16 board returns 12 bits of A to D data in high 12 bits and channel
   number in low 4 bits.  This routine simply does a logical shift right
   4 bits.
   To do a bipolar conversion subtract 2048 from the result (or xor 8000h
   to the value before shifting).  Mode 9 reads the dash16 port
   base address + 8 bit 7 to see what mode the board is in before
   converting the number  (see page 11 of the dash16 manual).
   This routine should be faster than mode 9.
*/
{
int i;
  for (i=0; i<samples; i++)
  {
    *(pointer+i) = *(pointer+i) >> 4;       /* arithimetic shift right*/
    *(pointer+i) = *(pointer+i) & 0xfff;    /* zero out the high bits */
     /* The above two instructions do the same as a logical shift right*/
  }
}


void printdata(int samples, int *pointer)
/* prints out the data to the standard output device */
{
int i;
  for (i=0; i < samples; i++)
    printf("%5d:%5d\n",i,*(pointer+i));
}


void dodma(int samples, int *pointer)
/* calls mode 6
mode,data and flags are assumed to be global variable in the data segment
mode 6 is called in the non-recycle mode.  The dash16 board will wait for
IPO/TRGO to go high and then will return control back to the C program.
The board will then do conversions in the background and stop.  The C
program must wait until data is valid before converting it
Errors will be reported and cause the program to stop.
errors are 1 driver not initialized
           11 number of conversion negitive (or greater than 32,767)
```

```
               20 interrupt of DMA already active
               21 page wraparound (similar to 11)
*/

{
  mode = 6;              /* do DMA operation */
  data[0] = samples;     /*number of samples*/
  data[1] = FP_SEG(pointer);     /*location for data for dma operation*/
  data[2] = 1;                   /*wait for IP0/TRG0 to go high before
                                   starting DMA*/
  data[3] = 0;                   /* do one cycle only*/
  dash16(&mode,&data[0],&flag);
  if (flag != 0)
  {
    sprintf(g_msg,"Error %d in DMA operation. Program aborted.\n",flag);
    exit(1);
  }
}

int backgroundbusy(void)
/* checks to see if dma or interrupt background operations are in progress
   if the dash16 driver has not be initilized flag could return and error
   but this routine will not report it.  To check for this compare the
   global varible flag to a 1.

   upon return from this call data[0] will contain the type of operation
   in progress.  0 is none; 1 is DMA ; 2 is interrupt.
   data[1] will contain 0 is done or 1 if active (also return by func)
   data[2] will conatain the current word count
*/
{
  mode = 8;
  dash16(&mode,&data[0],&flag);
  return data[1];
}



void stopbackground(void)
/* Calles mode 7 which will stop any running intrerrupt or DMA operations
of the dash16 board.
the only error possible with this call is driver not initialized
if this is the case flag will be set to 1 upon return from the call
this routine will not check for this error
*/
{
  mode = 7;
  dash16(&mode,&data[0],&flag);
}

void getdash16info(int *data, int *channelptr, long *samplerateptr,
                    int *sampleptr)
/*This routine opens DASH16.ADR file and reads for the DASH16 I/O address,
interrupt level, DMA level, channel number and sample rate. All numbers
(except for data rates) are stored in the DASH16.ADR file in a hex format
(I/O location is compatible with the DASH16 basic programs). Place a &H
before each number in DASH16.ADR except for the data rates - place a &D
before data rate numbers. Notice that programs that came with the card
written in BASIC read only the I/O address from the file - BASIC should
still be able to read the file though. The DASH16.ADR file can contain up
to 6 lines. The first line must have the I/O address of the DASH16 card.
If the file or the address is not found the program prints an error
message and halts.  If the address is not in the proper range than a
default of 300h is assumed. The next line can contain the interrupt level
used, if it is not found then interrupt level 2 is assumed. The third line
can contain the DMA level to use if the interrupt level is omited the DMA
level can not be specified.  If the DMA level is not found then level 3 is
assumed. The fourth line may contain the channel number to use (assuming
DMA level is specified). The Fifth line may contain the sample rate values
to use. (Once again the fith line can only be specified if the channel
number is specified). The sixth line can contain (if everything else has
been specified) the number of samples to perform. Limited error checking
is done for values that were read and can not be correct, default values
are used. Finally the routine prints out parameters that will be used. */

{
 #define defaultIO 0x300    /* default address of card */
 #define defaultint 2       /* default interrupt level */
 #define defaultDMA 3       /* default DMA level */
 #define defaultchannel 0
 #define defaultsamplerate 1000
 #define defaultsamples  1000

 #define NewY (gety()+10) /*macro used to update the text output loc. */

 FILE *dash, *fopen(); /*declare dash and fopen to be pointers to FILE*/
 unsigned float temp;  /* used to print out samplerate is hz */
 char message[80];

  setcolor(EGA_RED);
  moveto(160,80);
  outtext("Loading startup values from DASH16.ADR");
```

```
moveto(160,NewY);
if ((dash = fopen("dash16.adr","r")) == NULL)    /*open the file*/
{
  sprintf(g_msg,"\nDASH16.ADR not found.\nProgram aborted\n");
  exit(1);              /* exit from the program    errer level 1 */
}

if ((fscanf(dash,"&H%x",&data[0]))== EOF)      /*read I/O address*/
{
  sprintf(g_msg,"\nError in DASH16.ADR\nProgram aborted\n");
  exit(1);
}


  if (!((fscanf(dash,"\n&H%x",&data[1])) == EOF)) /*read int. level*/
  {
    if ((fscanf(dash,"\n&H%x",&data[2])) == EOF)  /*read DMA level*/
      {
        data[2] = defaultDMA;                /*no DMA level found*/
        outtext("Default DMA level used.");
        moveto(160,NewY);
      }
  }
  else
  {
    data[1] = defaultint;              /*no DMA or Interupt level*/
    data[2] = defaultDMA;              /* found */
    outtext("Default interrupt level and DMA level used.");
    moveto(160,NewY);
  }

if (!(ferror(dash)))                /* read channel number */
 if ((fscanf(dash,"\n&H%x",channelptr)) == EOF)
 {
   *channelptr = defaultchannel;
   outtext("Default channel number used.");
   moveto(160,NewY);
 }

if (!(ferror(dash)))          /*read sample rate values*/
  if ((fscanf(dash,"\n&D%ld",samplerateptr)) == EOF)
  {
    *samplerateptr = defaultsamplerate;
    outtext("Default sample rate used.");
    moveto(160,NewY);
  }
```

```
if (!(ferror(dash)))          /*read number of samples*/
  if ((fscanf(dash,"\n&D%d",sampleptr)) == EOF)
  {
    *sampleptr = defaultsamples;
    outtext("Default number of samples used.");
    moveto(160,NewY);
  }

fclose(dash);

moveto(160,NewY);
if ((data[0] < 255)  || (data[0] > 1016))
  {
    data[0] = defaultIO;
    outtext("I/O level out of range.  Default value used.");
    moveto(160,NewY);
  }
if ((data[1] < 2) || (data[1] > 7))
  {
    data[1] = defaultint;
    outtext("Interrupt level out of range.  Default value used.");
    moveto(160,NewY);
  }
if ((data[2] != 1) && (data[2] != 3))
{
  data[2] = defaultDMA;
  outtext("DMA level out of range.  Default value used.");
  moveto(160,NewY);
}

if ((*channelptr < 0) || (*channelptr > 15))
  {
    *channelptr = defaultchannel;
    outtext("Channel number out of range.  Default value used.");
    moveto(160,NewY);
  }

if (*samplerateptr < 4)
  {
    *samplerateptr = defaultsamplerate;
    outtext("Sample rate out of range.  Default value used.");
    moveto(160,NewY);
  }

if (*sampleptr < 1)
```

```
    {
      *sampleptr = defaultsamples;
      outtext("Number of samples negative.  Default value used.");
      moveto(160,NewY);
    }
  outtext("Loading completed.");
  moveto(160,NewY);
  moveto(160,NewY);
  moveto(160,NewY);

  setcolor(EGA_WHITE);
  outtext("Using the following values:");
  moveto(160,NewY);
  moveto(160,NewY);
  sprintf(message,"I/O address: %d (%xhex)",data[0],data[0]);
  outtext(message);
  moveto(160,NewY);
  sprintf(message,"Interrupt level: %d",data[1]);
  outtext(message);
  moveto(160,NewY);
  sprintf(message,"DMA level: %d",data[2]);
  outtext(message);
  moveto(160,NewY);
  sprintf(message,"Channel: %d",*channelptr);
  outtext(message);
  moveto(160,NewY);
  temp = 1000000/ *samplerateptr;
  sprintf(message,"Sample rate: %.8g hz",temp);
  outtext(message);
  moveto(160,NewY);
  sprintf(message,"Samples: %d",*sampleptr);
  outtext(message);
  moveto(160,NewY);
  moveto(160,NewY);
  outtext("Press any key to continue.");
  do {} while (kbhit()==0);
  getch();
}
```

REFERENCES

1. Kurt D. Smith, Michael I. Darter, J. Brent Rauhut, and Kathleen T. Hall, Distress Identification Manual For The Long-Term Pavement Performance (LTPP) Studies, FHWA Contract No. DTFH61-85-C-0095, Draft, March, 1987.

2. J. Brent Rauhut, M. I. Darter, R. L. Lytton, P. R. Jordahl, and M. Gardner, Data Collection Guide For Long-Term Pavement Performance Studies, FHWA Contract No. DTFH61-85-C-0095, Draft Report, June, 1987.

3. James C. Wambold, Gordon L. Hayhoe, William H. Park, and Mac E. Bryan, Survey System For Road Roughness, Rutting, and Topography (SIRST), FHWA Contract No. DTFH61-80-C-00049, Report No. FHWA/RD-86/034, February, 1986.

4. Ronald W. Carmichael, Automated Pavement Data Collection Equipment Roughness and Profile Measurement: Equipment Report, FHWA Demonstration Project No. 72, Report No. FHWA-DP-72-1, September, 1986.

5. "Performance Standards For Light-Emitting Products", Part 1040, Code of Federal Regulations, Title 21, April, 1988.