A Tactile Sensing System and Optical Tactile-Sensor Array for Robotic Applications

Stefan Begej

COINS Technical Report 85-06

Laboratory for Perceptual Robotics
Computer and Information Science Department
University of Massachusetts, Amherst, MA 01003

Communicated by Professor Michael A. Arbib

This research was supported by the Martin Marietta Aerospace Corporation (Contract GH3-116796) and the National Science Foundation (Contract ECS-8108818).

Abstract

The development of a modular tactile sensing system (TSS) intended for robotic applications is described. The TSS consists of a preprocessor module for tactile data acquisition and preprocessing, and a planar, optical tactile sensor array (TSA) module. The motivation behind development of the TSS was: to give researchers unobstructed access to tactile data in a laboratory or industrial environment; to provide a versatile processor module capable of interfacing to many TSA technologies, to remove a significant computational burden from the main robot controller (host computer); and to develop a promising new optical TSA technology relying upon the frustration of total internal reflection (TIR).

The preprocessor module is based upon the DEC T-11 microprocessor, and is housed in a small, cabinet-mountable enclosure. It performs the functions of tactile data acquisition, calibration, correction, preprocessing, and threshold monitoring under direction from the host. Tactile data preprocessing functions included calculation of the taxel sum for the imprint array, the first moment, and changes in the sum or first moment in comparison to a stored reference imprint. Communication with the host is over a serial line and includes only high-level information such as command codes, computational results (which might also include taxel-level data), and threshold violation data. The preprocessor module is quite general and is designed to interface with any TSA module that conforms to its simple communication protocol.

The TSA module consists of a compact, high-density TIR-TSA with 100 taxels/sq-cm and an active area measuring 31 x 31 mm. The visual imprint representing the contact pressure distribution is conveyed to an optical interface via a coherent optical fiber cable, and transduced into digital form by a 32 x 32 pixel CCD camera. The effective gray scale force resolution is 5 bits, and the force response range of each taxel is approximately 0 - 0.4 N (0 - 42 gm). The development history of the TIR-TSA is reviewed, and includes a description of a large-area tactile sensor with a very high taxel density (1500 taxels/sq-cm), an early prototype of the TSA developed in this project, and a non-planar, fingertip-shaped sensor intended as a prototype for sensors to be eventually used on a complex 9 DOF mechanical hand.

The TSS performance speeds range from 18 Hz for simple data acquisition functions down to approximately 5 Hz for more time-consuming functions such as calculation of first moments. Directions for future research are mentioned, and include: increasing the calculational speed of the preprocessor through the addition of hardware multiplication, division, and array (convolution) operations; increasing the memory size, addition of functions to extract tactile features such as surfaces, edges, holes, and texture; and creation of compact fingertip-shaped TIR sensors for use on complex mechanical hands.

Acknowledgements

I would like to thank those who have contributed significantly to this project. These persons include Ken Overton for his technical assistance and encouragement, Michael Arbib for his guidance, Nico Spinelli for his initial involvement and contributions to sensor development, Randy Ellis, Gerald Pocock, Damian Lyons, Stan Remizsewki, Thea Iberall, and the other members of the LPR staff for their aid and helpful comments. Finally, I thank my wife, Cathy, for her patience and understanding during this endeavor.

I would like to acknowledge the financial assistance of the Martin Marietta Aerospace Corporation (Contract GH3-116796) without which much of this work would not have been performed, and aid from the National Science Foundation (Grant ECS-8108818) for partial funding of this work.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction to Tactile Sensing for Robotics

It is increasingly recognized (see Harmon [9, 10, 11]) that haptic perception can play an important role in the automated manufacturing and inspection process. The two components of haptic perception are tactile sensation (i.e., cutaneous sensation) and kinesthetic sensation (e.g., joint angles, joint torques, and limb positions). In human terms, haptic perception tells us how an object "feels" when it is touched or when a manipulative action is executed. Robots with the capacity for haptic perception would be able to characterize surface texture and shape, recognize the identity, position, and orientation of objects during visual occlusion, detect slip caused by gravity or collision during grasping, transportation, or positioning of an object, and provide force feedback to the control system during object manipulation and assembly tasks.

This project addressed the tactile sensing aspect of the haptic perception problem. The objective of this project was to develop a modular "turnkey" tactile sensing system (TSS) consisting of a preprocessor module and a tactile sensor array (TSA) module. The development of the TSS was motivated by the need to:

1. Provide easy access to tactile data by researchers in a laboratory or industrial environment;
2. Remove a significant computational burden from the robot controller (host computer) by assigning data collection and processing duties to the preprocessor;
3. Provide a general preprocessor capable of interfacing to many TSA technologies;
4. Serve as a vehicle for development of a promising new TSA technology based upon the frustration of total internal reflection (TIR) that promised large tactile element (taxel) densities and simple fabrication of non-planar sensing surfaces, e.g. fingertip-shaped.

Figures 1.1 and 1.2 show photographs of the TSS and the TSA head, respectively, and Figure 1.3 illustrates how a TSS might be used in actual practice to perform an assembly or inspection task.

The preprocessor module was based upon the DEC T-11 microprocessor and was housed in a small, cabinet-mountable enclosure. It performed the functions of tactile data acquisition, calibration, correction, preprocessing, and threshold-monitoring under direction from the host. Preprocessing functions included calculation of the taxel sum for the imprint array, the first moment, and changes in the sum or first moment in comparison to a stored reference imprint. Communication with the host was over a serial line and was intended to include only "high-level" information such as command codes, computational results (which could also include taxel-level data), and threshold-violation data. The preprocessor module was quite general and could be interfaced with any TSA that conformed to its simple communication protocol.

The companion TSA module was based upon a new approach using optical technology. The local pressure distribution associated with a contacting object was used to frustrate total internal reflection (TIR) at an optical surface. The visual image (imprint) so formed was conveyed by a coherent optical fiber cable to a solid state camera chip, converted to digital form, and transmitted to the preprocessor module. The size of the active area and taxel density of the TIR-TSA was 31 x 31 mm and 100 taxels/sq-cm, respectively. The TSS data acquisition and preprocessing rates ranged from 18 Hz for simple functions such as data acquisition, down to 5 Hz for more time-consuming operations such as the calculation of the first moments.

This report is divided into 5 chapters. Chapter 2 reviews the literature concerning previous research efforts on TSAs. Chapter 3 discusses work on the TIR-TSA that was performed by the author prior to initiation of this project. The development history of the TIR-TSA includes a large-area tactile sensor with a very high taxel density (1500 taxels/sq-cm), and a non-planar, fingertip-shaped sensor. The latter was a prototype version of a new sensor intended for eventual attachment to a complex 9 DOF mechanical hand. Chapter 4 reviews design

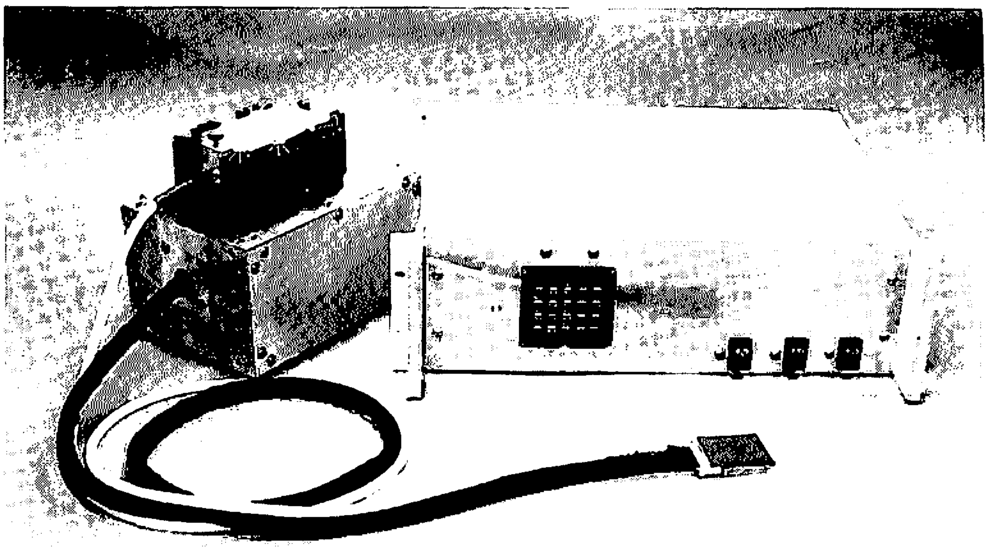


Figure 1.1: Tactile sensor system consisting of TIR-TSA module (left), illuminator (upper-left), preprocessor module (right). The TSA head is in the foreground.

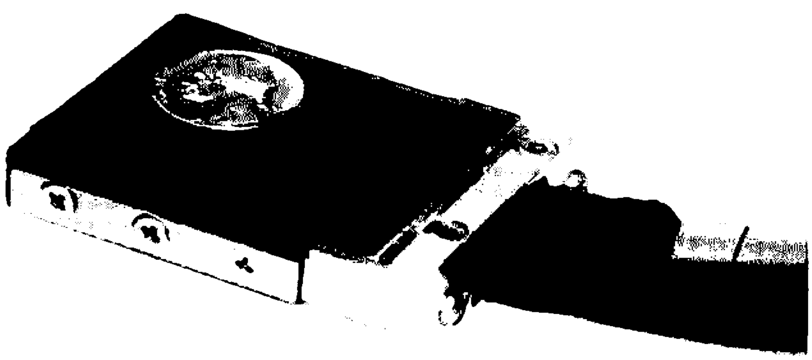


Figure 1.2: Close-up view of the TIR-TSA head.

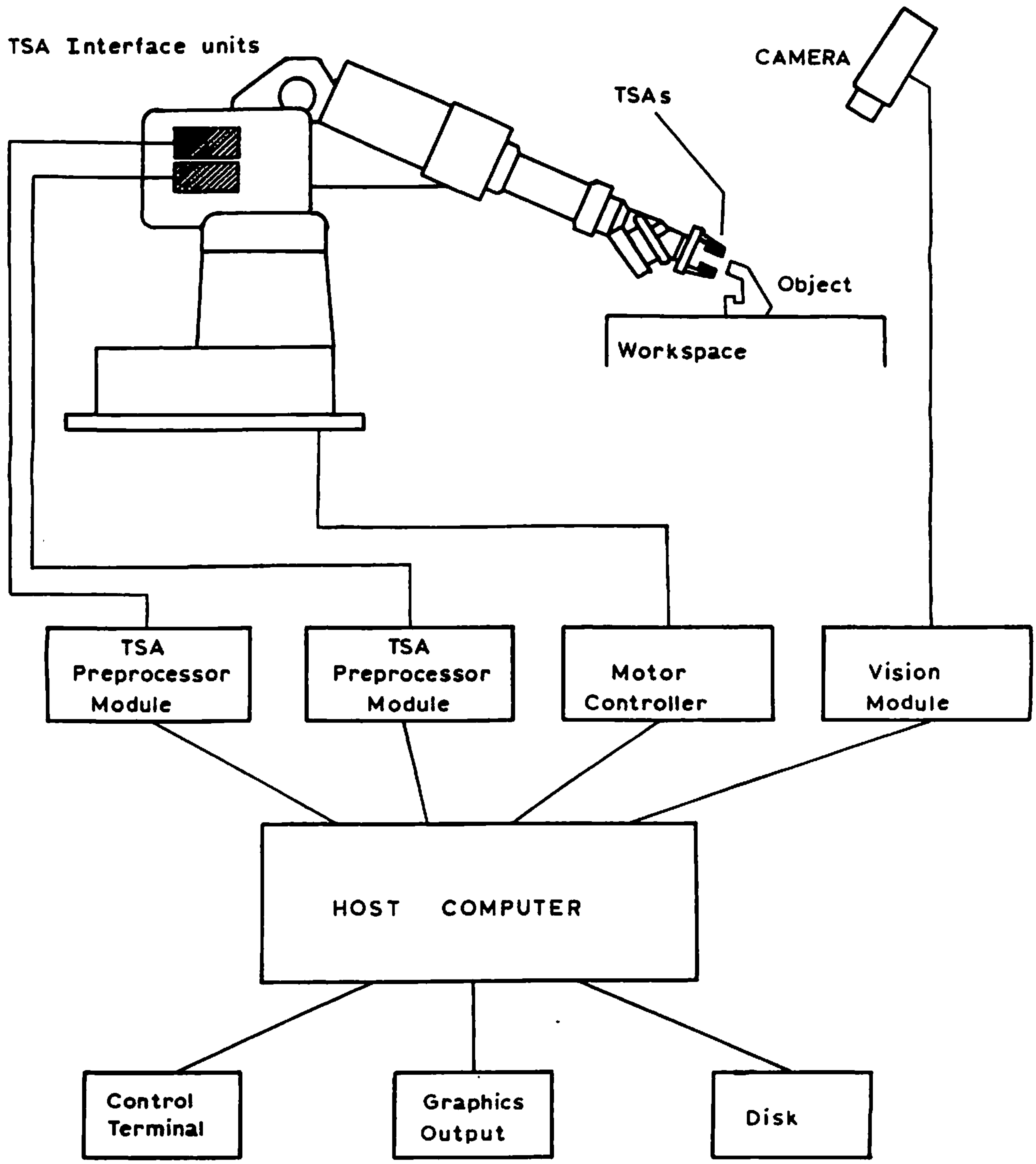


Figure 1.3: Example of how several TSSs might be incorporated into a robotic assembly or inspection process.

specifications for the TSS using a TIR-TSA module, and provides construction and assembly details of the major components within the system. Chapter 5 provides instructions regarding the use of the TSS, and illustrates its performance with a variety of examples. Suggestions for future research are listed in the Conclusion section, e.g., increasing the calculational speed and memory size of the preprocessor, addition of functions to extract tactile features such as surfaces, edges, holes, and texture, and creation of compact fingertip-shaped TIR sensors for use on complex mechanical hands. Finally, Appendix A provides a listing of the PDP-11 assembly language code for the data acquisition and preprocessing operations implemented by the preprocessor module.

Chapter 2

Tactile-Sensor Array Technology

2.1 Introduction

TSAs are the robot's equivalent to human skin. They provide information regarding the shape of an object, its position, and force feedback that is useful in numerous physical tasks, e.g., object identification, determination of object orientation and position, detection of slip, and control of manipulative actions during transport or assembly operations. A summary of the characteristics that a "good" TSA would possess is shown in Table 2.1. No single TSA technology currently meets all these requirements, though some meet the more important ones better than others, e.g., ruggedness, conformability, repeatability, and sensory preprocessing. The following sections review in some detail the progress made by researchers in the development of TSAs based upon electrical conductance, optical, capacitive, and piezoelectric technology. A more general review concerning the design and use of tactile sensors for robotics may be found in Harmon [9, 10, 11].

2.2 Electrically Conductive Materials

Technology employing electrically conductive materials encompasses all transduction methods which involve conversion of contact-induced displacements or pressures into changes in electrical conductivity. This could occur by changing the contact resistance between two conductors, or by changing the electrical conductance of the material, or both.

TABLE 2.1

Desirable TSA characteristics for robotic applications.

- Low cost
- Ruggedness
- Low sensing head volume (compactness)
- Easily tailored sensor shape (planar, hemispherical)
- Compact communication/interface cable
- On-board preprocessing capability
- Monotonic response (preferably linear)
- Low hysteresis
- Frequency response from DC to several hundred Hertz
- Freedom from cross-talk (ghosting, cross-coupling)
- Good lateral spatial resolution
- Large depth resolution and range
- Large force resolution and range
- Good reproducibility
- High accuracy
- Low drift

Briot [4] described a planar sensor in which a 24 x 24 array of electrodes on a printed circuit board were each surrounded by electrode ring to which a voltage was applied. An elastic cover coated on one side with conductive paint was placed over the electrode array and the contact force was measured as a variation of electrical resistance between each electrode pair. The exact size of the sensor was not reported, but appeared to be approximately 10 x 10 cm. The sensor was used only in the binary sensing mode due to imperfections in the conductive paint coating. Briot also briefly describes a 10 x 10 array with 1 cm square pads that employ a conductive foam cover, though no performance characteristics were reported for this particular sensor.

Larcombe [14] and Taylor et al. [26] discussed the use of a conductive carbon-fiber felt sandwiched between metal foil electrodes. Though inexpensive, difficulties were encountered in establishing good electrical and mechanical connections between the felt and the foil electrodes. In general, research has tended to move away from such materials to conductive elastomers which are considerably easier to work with.

Purbrick [19] reported on the development of a 16 x 16 sensor array in which the top and bottom electrodes were "D" shaped conductive silicone rubber chords and metal wires, respectively. When pressure was applied to the rubber electrode, it was deformed as it pressed against the metal wire and caused the electrical resistance across the pair to vary in correspondence with changes in the area of contact.

Overton and Williams [16, 17] developed a tactile sensor array which used discrete hairpin-shaped loops of conductive silicone rubber embedded in a nonconductive silicone matrix. Both ends of the loops made contact with a circuit board that was interior to the sensor body. The great advantage of this arrangement was that the front face of the sensor consisted of a rugged silicone rubber membrane that was free from breakage-prone metallic electrodes. An array of 8 x 16 taxels in an area measuring 2.5 x 2.5 cm (i.e., 20 taxels/sq-cm) was developed. Each taxel had a force sensing range of 0 - 10 N, and a force resolution of approximately 1 N. Electrical cross-talk was minimized by the inclusion of diodes near each sensing electrode. The maximum array scan rate was reported to be 44 Hz. Further details regarding sensor construction and performance may be found in Overton [18].

Hillis [12] developed a 16 x 16 tactile array sensor with a 1 x 1 cm active area in which the force range for each taxel was 0 - 1 N. The active area measured 1 x 1 cm. A linear conductive elastomer was used that conducted only in one direction. It was placed upon a printed circuit board covered with fine etched lines. A separator intervened between the circuit board and the conductive material, and acted to push them apart when the pressure was released. The conductive rubber pressed through the separator particles so that the area of contact (and hence the contact conductance) varied with applied pressure. The sensitivity and range was found to depend on the properties of the separator material. High pressure ranges were achieved with nylon-stocking material, whereas spray deposition of small nonconductive paint particles produced a separator that promoted high sensitivity. The array was scanned by applying voltage to one column at a time and measuring the voltages in each row. Ghost images caused by simultaneous activation of several points on a single line were avoided by using a mirror-voltage technique similar to Purbrick [19]. This method, however, was valid only when the linear resistances were low compared to the contact resistances. All devices were rugged, and showed good electrical stability after an initial settling period, though the response was reported to be non-linear.

Salerno [23] briefly described a commercial sensor based upon proprietary technology manufactured by the Barry Wright Corporation (model TS 402 Tactile Sensor). The sensor was mounted on a printed circuit board base which carried an intermediate arrangement of conducting elastomer members, and was covered by a tough protective rubber material. A 16 x 16 array measured 4 x 4 cm and resulted in a taxel density of 16 taxels/sq-cm. Hysteresis was claimed to be exceptionally low, and the response characteristics were logarithmically related to the applied load. The force range of each taxel was reported to be 0 - 1.8 N.

Raibert and Tanner [20] took a large step forward in the development of TSAs by incorporating a VLSI processor at the site of each taxel. The processors were interconnected to form a special-purpose parallel computer designed to handle transduction, computation, and communication. This approach promised to greatly decrease data acquisition and processing time, as well as significantly reduce the number of cables required for communication with an external robot control system. This approach was also consistent with the desire to have the tactile sensor communicate high level information over a minimum number of wires. A prototypic 3 x 3 VSLI-TSA was described in which each taxel had a force range of 0 - 0.5 N. The major problems encountered with the sensor were a restriction of the sensor shape to planar geometries by the VLSI technology, and the inherent lack of ruggedness of a VLSI chip when placed in a mechanically and electrically noisy robot environment.

Raibert [21] extended his earlier work and described a 6 x 8 VLSI-TSA that was entirely digital in its method of operation. Wedge-shaped masks were used to constrain the deflection of the conductive elastomer membrane, thereby controlling the degree of contact with an underlying pattern of electrode strips oriented orthogonally to the wedge. The shape of the wedge could be tailored to provide linear, log, or exponential response to applied forces or displacements. Elimination of analog signal acquisition, conversion, and processing circuitry promised to greatly simplify the processor elements, and thereby increase reliability. However, the sensor was somewhat limited in that only 12 discrete force values were detectable, and the sensor was still inherently restricted by VLSI technology to planar geometries.

2.3 Optical Technology

Bejczy [2] described a 4 x 4 sensor array based upon optical reflection. Each sensing element consisted of an emitter and receiver optical fiber pair, each pair being set at a fixed distance from a reflective elastomer membrane. The latter was deflected inwardly by applied forces and thus changed the quantity of reflected light detected by the receiving fiber. The number of emitters and detectors at the end of each fiber was minimized by multiplexing. No performance characteristics of the sensor were provided.

Schneiter and Sheridan [25] expanded upon Bejczy's approach and developed a high-resolution TSA with a taxel density of 326 taxels/sq-cm. A transparent rubber was used to maintain the separation between the optical fibers and the elastic reflector membrane. Additionally, an optical arrangement was used that required only one fiber per taxel, as the fiber acted as both an emitter and receiver. The TSA was reported to suffer from several problems, including low fatigue life of the transparent rubber, a restricted dynamic range, and slow (33 ms) reaction time.

Rebman and Trull [22] described a sensor (model LTS 200) manufactured by the Lord Corporation of Cary, North Carolina. It is one of two tactile sensor arrays commercially available today. It transduced mechanical strains into an electrical signal by eclipsing a light beam with a mechanical edge that was connected to an elastically-deflectable membrane. Thus, force and displacement response characteristics were alterable independently of the strain transduction mechanism, which led to freedom from crosstalk, greater flexibility in design, and inherently high ruggedness. This sensor was available with 16 taxels/sq-cm over an active area measuring 2 x 3 cm. Each sensing element had a force range of 0 - 1.2 N. However, the resolution was only 0.2 N.

Jacobsen et al. [13] described the use of optical fibers to sense the change in the optical properties of birefringent acrylic materials when subjected to contact stresses. The output was found to be a linear function of the applied pressure. However, only discrete sensor elements were described (i.e., not TSAs), and were designed for mounting at various locations on a tendon-driven anthropomorphic hand. A major concern not addressed was the the question of optical fiber longevity when subjected to frequent finger bending motions.

2.4 Piezoelectric Materials

A piezoelectric material produces a transient voltage output in response to mechanical deformation. This effect has been investigated by numerous researchers in a search for robot sensors and "artificial skin" for prosthetic purposes. A fundamental difficulty will all piezoelectric devices is that they are responsive only to changes in pressure or strain, and are insensitive to static strains. The advantages, however, are that they are small, lightweight, and conformable to the shape of the robot gripper or fingertips, and also exhibit very fast response times.

Dario [5], for example, described work with the material PVF (polyvinylidene fluoride), which can be sensitized and made piezoelectric by appropriate thermo-mechanical treatments. It is thin and flexible, but somewhat delicate. Dario describes a hybrid sensor design using PVF and a conductive elastomer in an effort to combine the dynamic and static response characteristics of both technologies. Additionally, a heating film was incorporated into the back of the sensor in an attempt to use the pyroelectric properties of the PVF film to sense the thermal conductivity of the contacting object. However, this approach was not entirely successful as there was some difficulty in distinguishing the superimposed piezoelectric and thermoelectric signals.

2.5 Capacitive Technology

Geschke [7] reported a simple, single-element tactile sensor consisting of two printed circuit board electrodes separated by 2.4 mm of plastic foam. The sensor had a force range of 0 - 22.5 N with a maximum force resolution of 0.06 N at the low-force end of the range. Drift, hysteresis, and fatigue were reported to be small.

Abramowitz et al. [1] described the design of a 2 x 4 element capacitive TSA. The primary advantages of this technology were high sensitivity, compactness, and light weight. The disadvantages mentioned were piezoelectric and pyroelectric noise in the dielectric film, and mechanical crosstalk. No performance data was cited.

Boie [3] described a capacitive TSA composed of a 3-layer sandwich. The top and bottom layers were flexible circuit boards with electrode strips running transverse to one another. A dielectric layer was placed between the circuit boards, thus forming a capacitor element at those locations where the strips overlapped. Forces and displacements were measured as changes in the capacitance of each element. If the compressibility of the dielectric layer followed a linear spring law, then the dynamic and static response of the sensor was inherently linear and would show little hysteresis. Boie demonstrated an 8 x 8 sensor array with an active area measuring 2.5 x 2.5 cm in which frame rates of 390 Hz were achieved. The force or depth range of the sensor was not stated. The chief disadvantages of capacitive sensors were: 1) the top electrode strips were susceptible to puncture damage from sharp objects or from fatigue damage from extended use; 2) the low signal levels inherent in the measurement of small capacitances made the sensor susceptible to radio frequency noise interference from the laboratory or industrial environment; and 3) mechanical and electrical crosstalk was not completely eliminated.

Chapter 3

TIR Tactile Sensor Array

3.1 Introduction

This chapter discusses a new optical TSA which is based upon the generation of tactile images by the frustration of total internal reflection (TIR). The basic principles are elaborated in Section 3.2. This approach to tactile sensing offered several advantages of other TSA technologies: high spatial resolution, fine force resolution, and an ability to fabricate the sensor into non-planar shapes (e.g., cylindrical and hemispherical). The performance characteristics of two early planar TSAs and one non-planar TIR sensor are presented in Section 3.3. Finally, the conclusions from these early studies are summarized in Section 3.4.

3.2 TIR Technique for Tactile Sensing

The basic physical principle upon which the optical TSA is based involves the frustration of TIR (see Haliday and Resnick [8]) at an optical surface caused by contact between an object and an opaque elastomeric film resting upon the optical surface. An example of a similar effect may be observed by looking down into a glass of water and noting the disappearance of the silver lining wherever the outer surface of the glass is touched. The general arrangement of components in the TIR tactile sensor is shown in Figure 3.1. A transparent plate is illuminated from one or more edges. Most of this light is confined to the interior of the plate by total internal reflection at the two non-edge surfaces. Additionally, all edges through which light does not enter are made reflective so as to further minimize light losses.

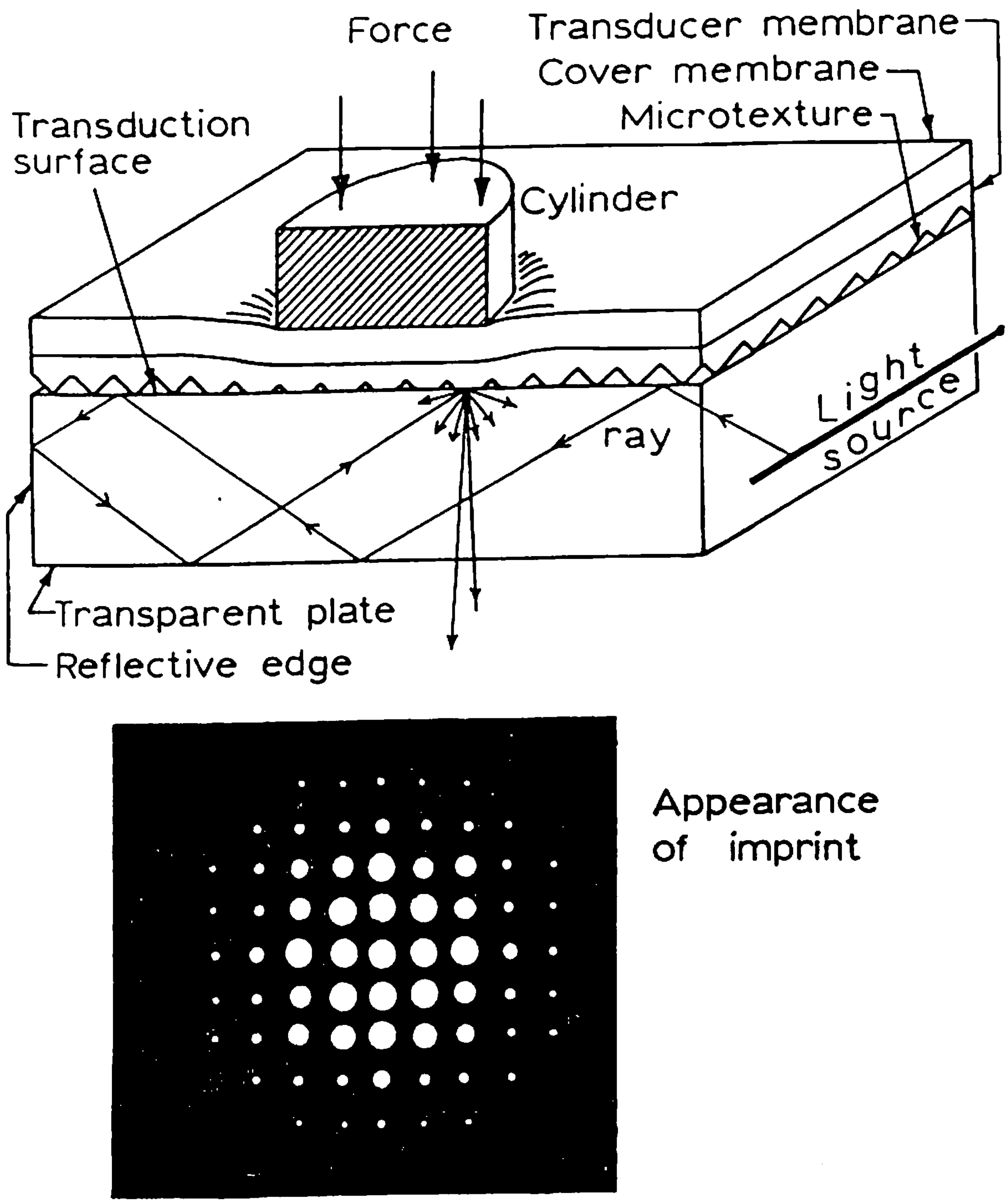


Figure 3.1: General arrangement of components in the TIR tactile sensor. Top diagram is a cross-sectional view. Bottom diagram illustrates how areas of greater contact appear as regions of higher intensity.

A thin elastic transducer membrane is placed in contact with the transducing (top) surface of the transparent plate. This membrane is opaque and suitably textured on one side, e.g., possessing a distribution of pyramidal, conical, or hemispherical asperities. Finally, a cover membrane is placed over the transducer membrane. The latter is black to prevent leakage of ambient room-light into the sensor. It also provides mechanical protection for the transducer membrane and serves as a mechanism for altering the response characteristics of the sensor.

An object pressed against the front of the TIR tactile sensor causes the asperities on the transducer membrane to come into contact with the transducing surface of the transparent plate, thereby frustrating total internal reflection at those sites where contact has occurred. Light that is normally confined within the transparent plate is absorbed by the opaque transducer membrane at the contact sites, and then diffusely re-emitted or reflected back into the transparent plate. However, this light is emitted at all angles, and those light rays that no longer satisfy the conditions for total internal reflection within the transparent plate may exit from the opposite side, and thereby be detected.

It is possible to infer the strains imparted by an object to the transducer and cover membranes from the average intensity of light in a local area of the imprint. In the idealized situation shown in Figure 3.1, each asperity is considered to be the center of a square cell. The average intensity of light emanating from the cell is directly related to the fraction of the cell filled by the deformed asperity (assuming total internal reflection is substantially destroyed wherever contact occurs between the asperity and the transducer surface). By this mechanism, an imprint may be acquired in which local average light intensities represent the magnitude of the normal (but not shear) strains or pressures associated with an object in contact with the sensor.

3.3 Review of Past Developments

This section will review the state of development of the TIR tactile sensor technology that existed prior to initiation of this project. Three TIR tactile sensors prototypes were developed. The first sensor was a large-area tactile sensor (LA-TS), the second was a small, compact tactile sensor array (C-TSA) designed primarily for attachment to robot gripper fingertips, and the third was a fingertip-shaped tactile sensor (FS-TS) designed for eventual placement on the Salisbury Hand (Salisbury [24]). The primary differences between these sensors were their physical size, their geometry, and the means by which the force imprint was transported from the transduction area to the camera.

3.3.1 Large-Area Tactile Array

A photograph and schematic of the LA-TS are show in Figure 3.2 and Figure 3.3, respectively. A 128 x 128 pixel CID camera (General Electric TN2200) was used to image the imprint. The taxel density depended on the camera-to-sensor separation distance and the magnification provided by the lens system. The transparent plate was ordinary window glass 2 mm thick, and provided an active tactile sensing area measuring 7 x 12 cm. Light was injected only through one edge of the plate, and the other edges were covered with aluminum foil to minimize light losses. The light source was a 40 W linear-filament light bulb measuring 10 cm in length. Aluminum foil was used as a reflective shroud to direct as much radiation as possible into the edge of the transparent plate. The behavior of the LA-TS with various transducer and cover membranes was investigated. The membranes and their characteristics are listed in Table 3.1.

Figure 3.4 presents six tactile imprints acquired with a 128 x 128 pixel CID camera that imaged a 3.3 x 3.3 cm section of the LA-TS, and shows imprints of typical objects that might be encountered in a robot environment. This particular camera and lens configuration produced a TSA with an effective taxel

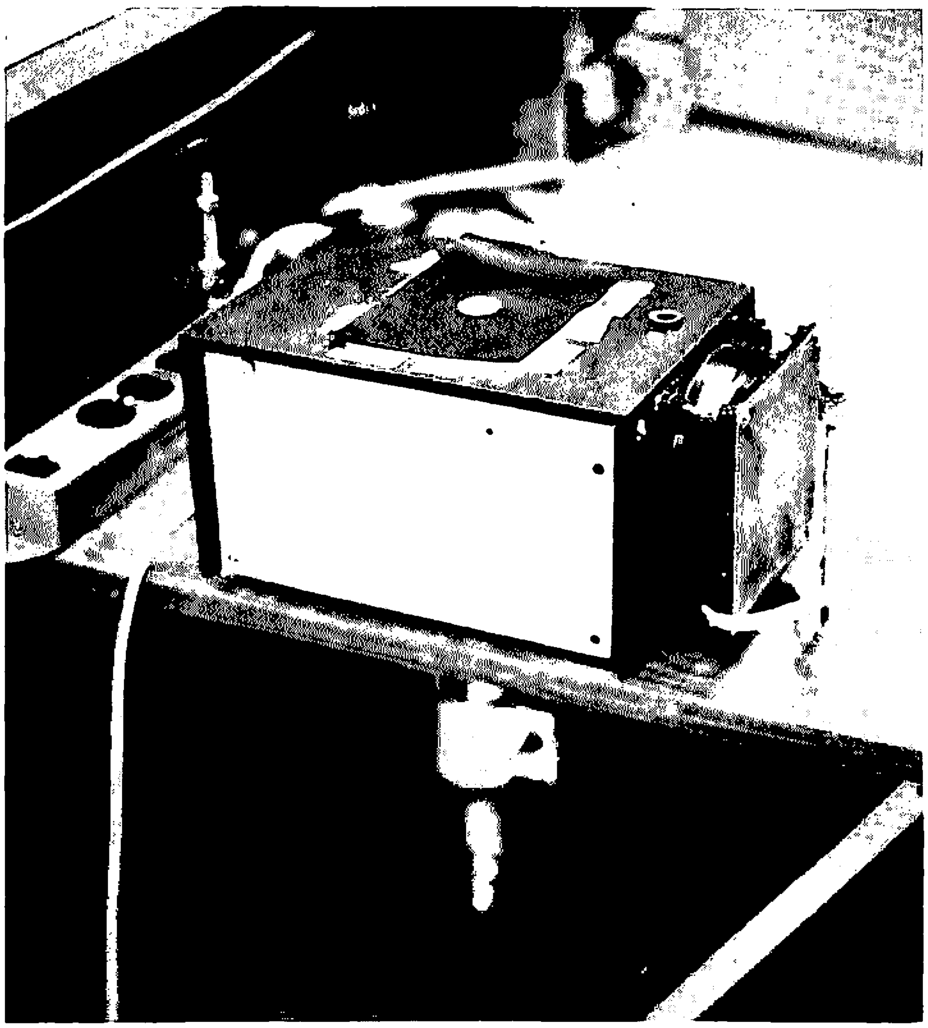


Figure 3.2: LA-TS with a 128 x 128 pixel CID camera mounted beneath.

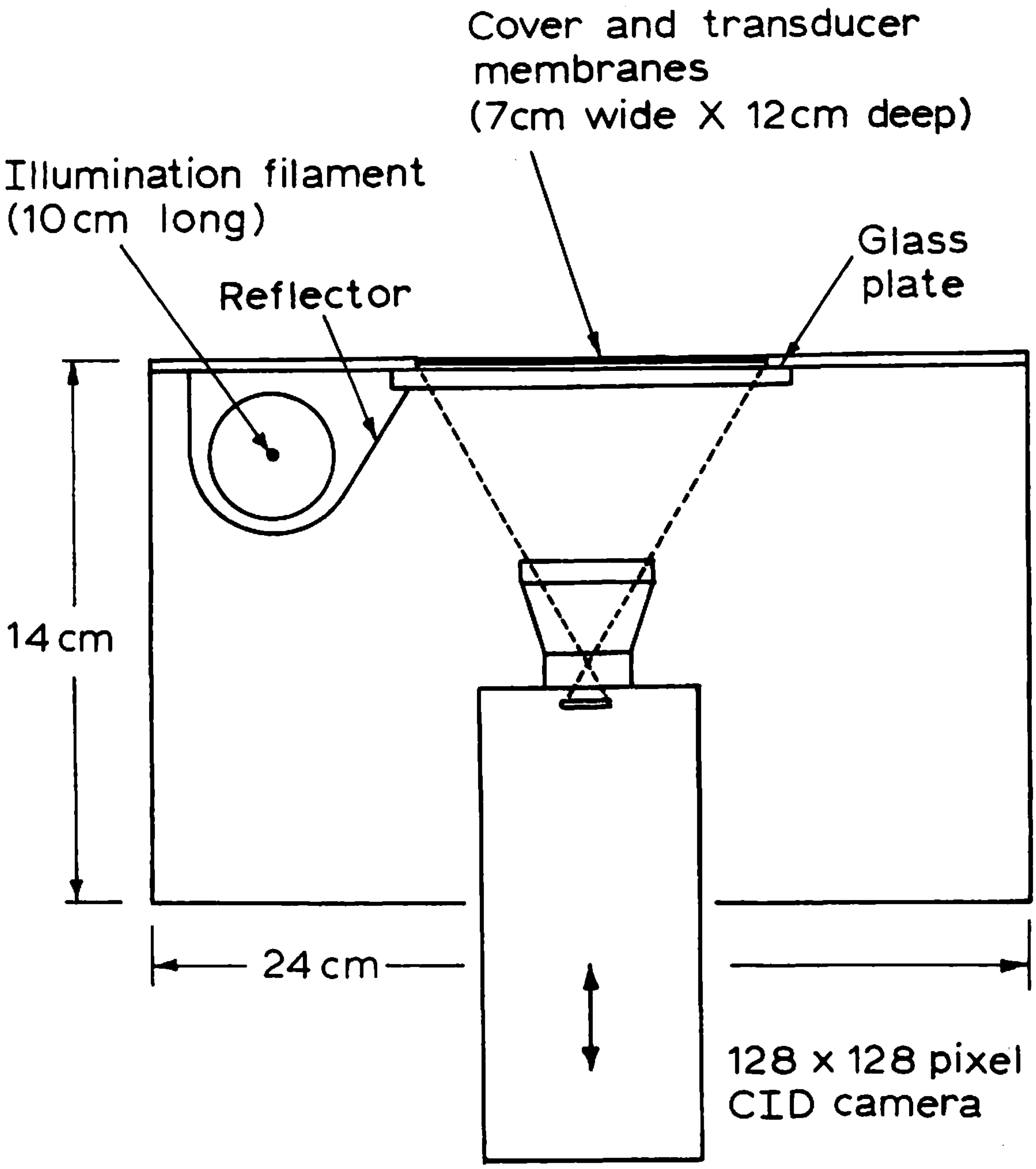


Figure 3.3: Cross-sectional view of the LA-TS. The resolution of the sensor could be varied by moving the camera along the vertical axis.

TABLE 3.1

Transducer and cover membranes codes and descriptions.

Code	Description
Transducer membranes:	
WP25	- White plastic (bread-wrapper), 25 micrometers thick.
WP125	- White plastic (frozen-food bag), 125 micrometers thick. Abraded on one side with coarse sandpaper to a matte finish.
L725	- Parallel rows of white Lycra fibers (sock) 600 micrometers diameter bonded to a tape backing 125 micrometers thick. 725 micrometers total thickness.
Cover membranes:	
BP125	- Black plastic sheet, 125 micrometers thick.
RB200	- Rubber sheet (balloon wall), 200 micrometers thick.
RB380	- Rubber sheet (balloon wall), 380 micrometers thick.
BN890	- Black neoprene sheet (rubber glove), 890 micrometers thick.

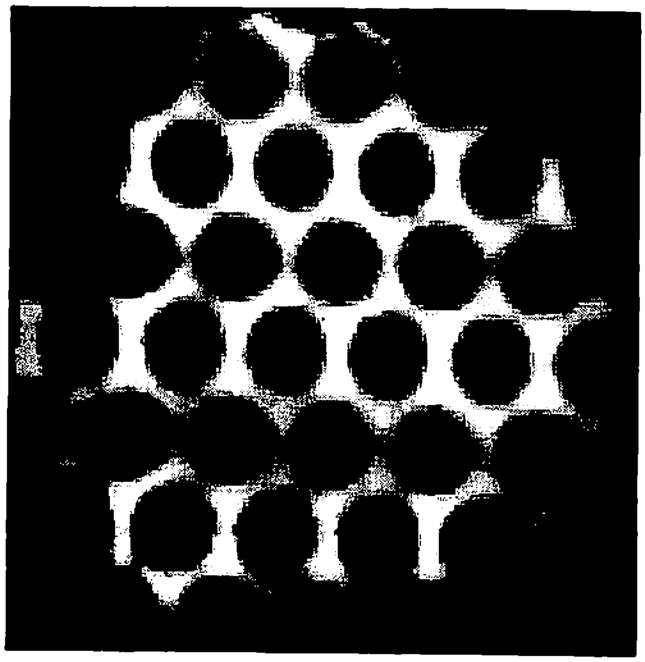


Figure 3.4a: LA-TS imprint of a perforated metal panel (WP125 transducer membrane with RB380 cover membrane).

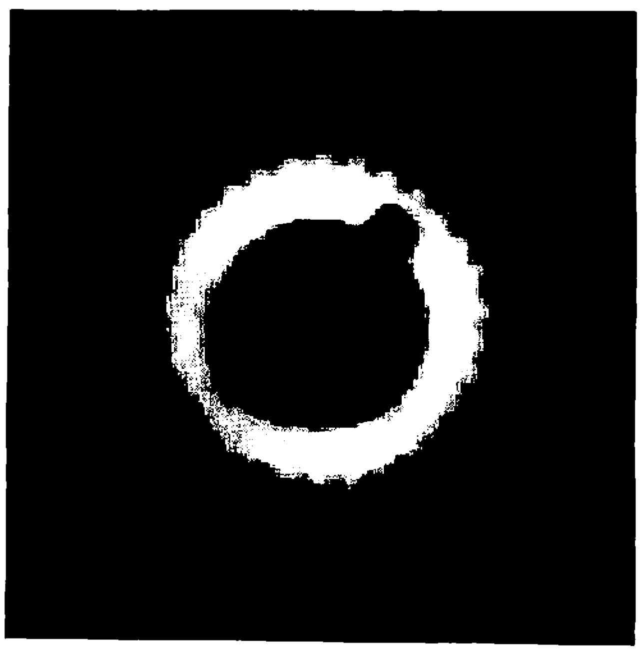


Figure 3.4b: Gear (WP125 + RB380).

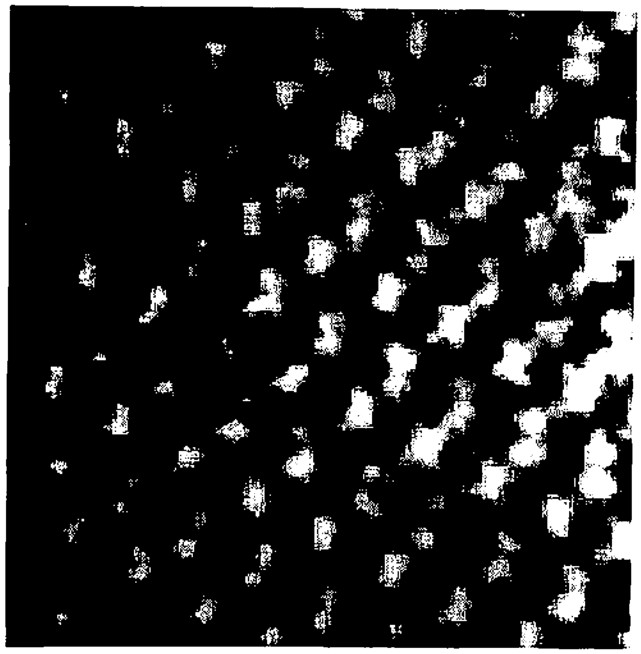


Figure 3.4c: Sweater (WP125 only).

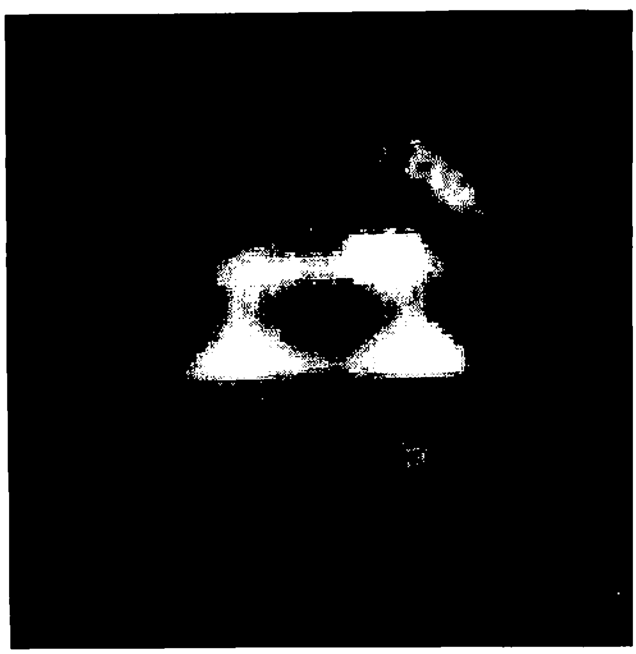


Figure 3.4d: Penny (WP125 + RB380).

Figure 3.4e: Palm print (WP25 only).

Figure 3.4f: Printed circuit board showing the surface relief after etching (WP125 + RB380).

density of approximately 1500 taxels/sq-cm. Figure 3.3e was taken using the transducer membrane WP25 only, and illustrates the fine force and spatial resolution that the LA-TS is capable of achieving. These imprints may be interpreted as detailed normal-force maps associated with object contact, or as narrow-range depth maps that reveal the topology of the object surface.

The response characteristics were found to depend on a number of interrelated parameters concerning the physical properties of the transducer and cover membranes, e.g., the total thickness of the membranes, the size and shape of the microtexture on the transducer membrane, and the mechanical properties of the membranes. These parameters directly influenced the depth and lateral spatial resolution, force or strain response, degree of hysteresis, and time response characteristics of the sensor.

It would be expected that if the membranes were thick or if the microtexture on the transducer membrane was coarse, then the mechanical spread function would be large and consequently fine spatial detail would be lost. However, the ruggedness would be high and the sensor would have a large depth sensing range. Furthermore, the mechanical properties of the transducer membrane and the shape of the microtexture would be expected to have a strong influence on the pressure range and the shape of the response curve, e.g., conical or pyramidal asperities might produce significantly different response curves than hemispherical asperities. This would make it possible to tailor the shape of the response to be linear, logarithmic, or whatever form the particular application demanded.

The primary parameters that were considered in this study were the total membrane thickness and the microtexture of the transducer membrane. The influence of the mechanical properties of the transducer or cover membrane upon the response characteristics was not examined at this time.

The response of the tactile sensor to uniform pressure was measured with the aid of the pneumatic pressure applicator shown in Figure 3.5. This device was capable of exerting uniform pressures of up to 0.414 MPa (60 psi) over an area

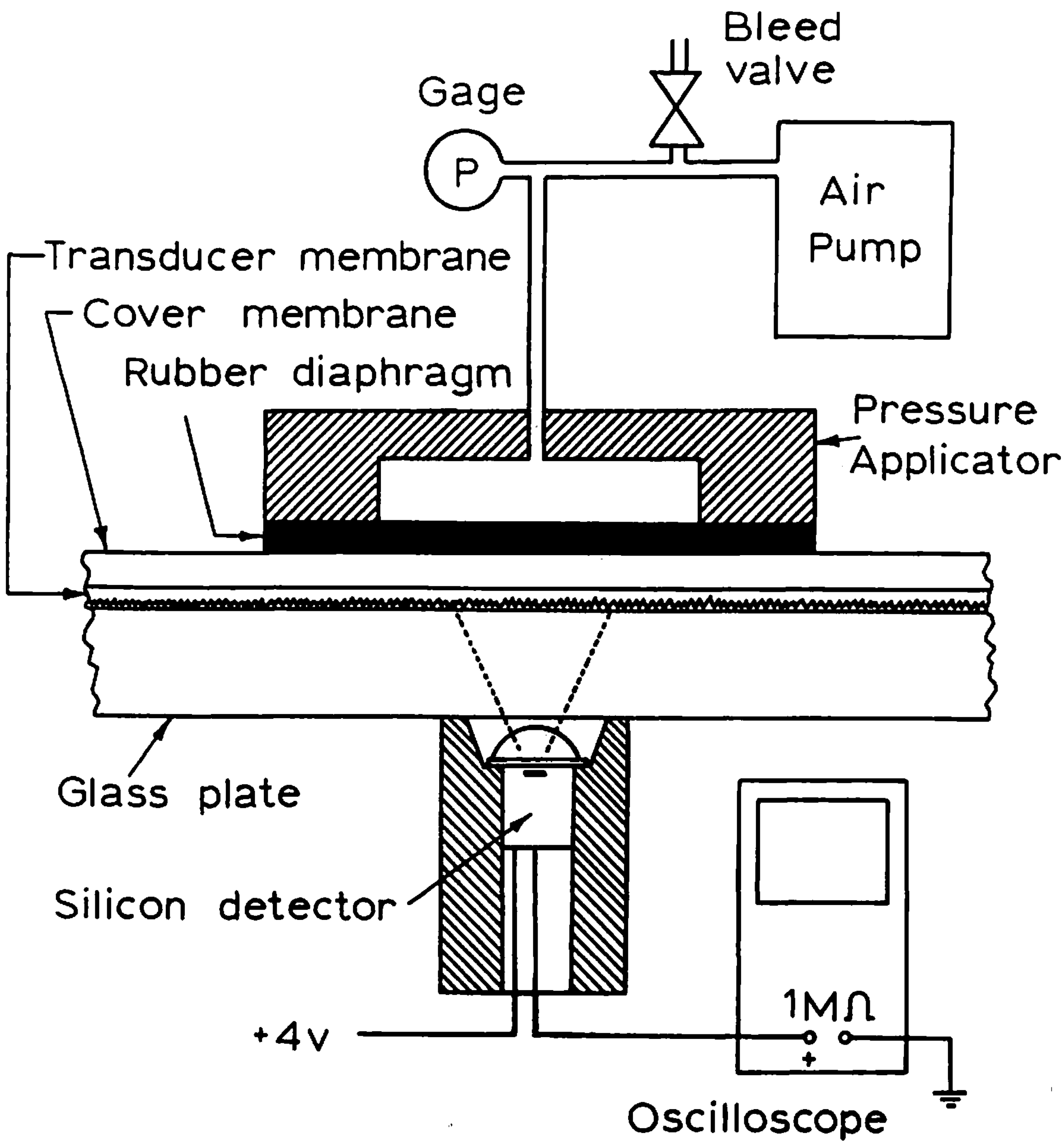


Figure 3.5: Pressure applicator and light sensor used to evaluate the response characteristics of the LA-TS.

measuring 1.3 x 1.3 cm. A single silicon photodetector with a response time of approximately 2 ms and a viewing window of 7 mm diameter was used to characterize the pressure response of the LA-TS with several transducer and cover membranes. The results are shown in Figure 3.6, where the light intensity output has been normalized in each situation to the maximum output occurring at the maximum applied pressure of 0.414 MPa (60 psi). The following points may be inferred from Figure 3.6:

1. All membranes resulted in a non-linear, monotonically increasing response with increasing pressure.
2. A noticeable amount of hysteresis was present. Furthermore, the degree of hysteresis was significantly larger for the L725 transducer membrane than for the WP125 membrane. (The latter effect may be attributable to the differing microtexture or glass surface adhesion properties of these two materials.)
3. The pressure response characteristics were not dramatically influenced by the thickness of the cover material.
4. Despite the large differences in microtexture shape and size, the response of the L725 transducer membrane was very similar to that of the WP125 + RB200 + RB380 membrane. (Note that the total membrane thicknesses were nearly the same.)

Sensor drift was assessed by applying a constant pressure of 0.345 MPa (50 psi) and monitoring the output over time. The results for the WP125 and L725 transducer membranes are shown in Figure 3.7. The drift was similar for both membranes and was seen to continually increase over a period of 300 seconds. This behavior is satisfactory for most robotic applications in which the task execution time ranges from 0.1 to 10 seconds.

An assessment of the sensor response time was made by observing the output signal in response to an input pulse supplied by a firm rubber ball bouncing on the LA-TS surface. Typical output pulse widths were approximately 10 ms, indicating that the response time of the sensor was no greater than 5 ms.

The relationship between the membrane thickness, microtexture, lateral spatial resolution, and depth resolution was qualitatively explored through an analysis of tactile imprints. One would intuitively expect that fine lateral or depth features would require the use of membranes that are thin or have a fine microtexture

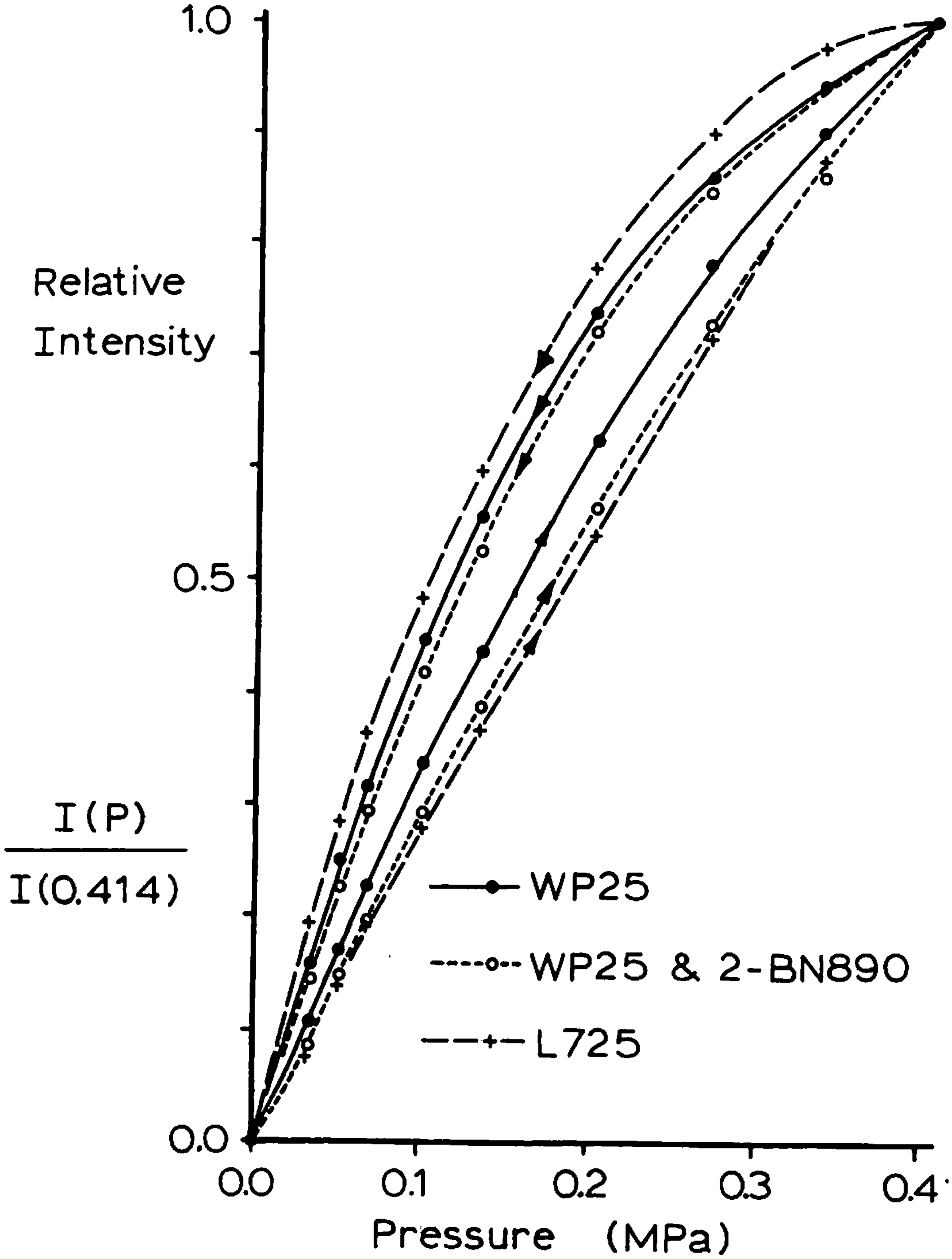


Figure 3.6: Pressure response characteristics of the LA-TS for various transducer and cover membranes.

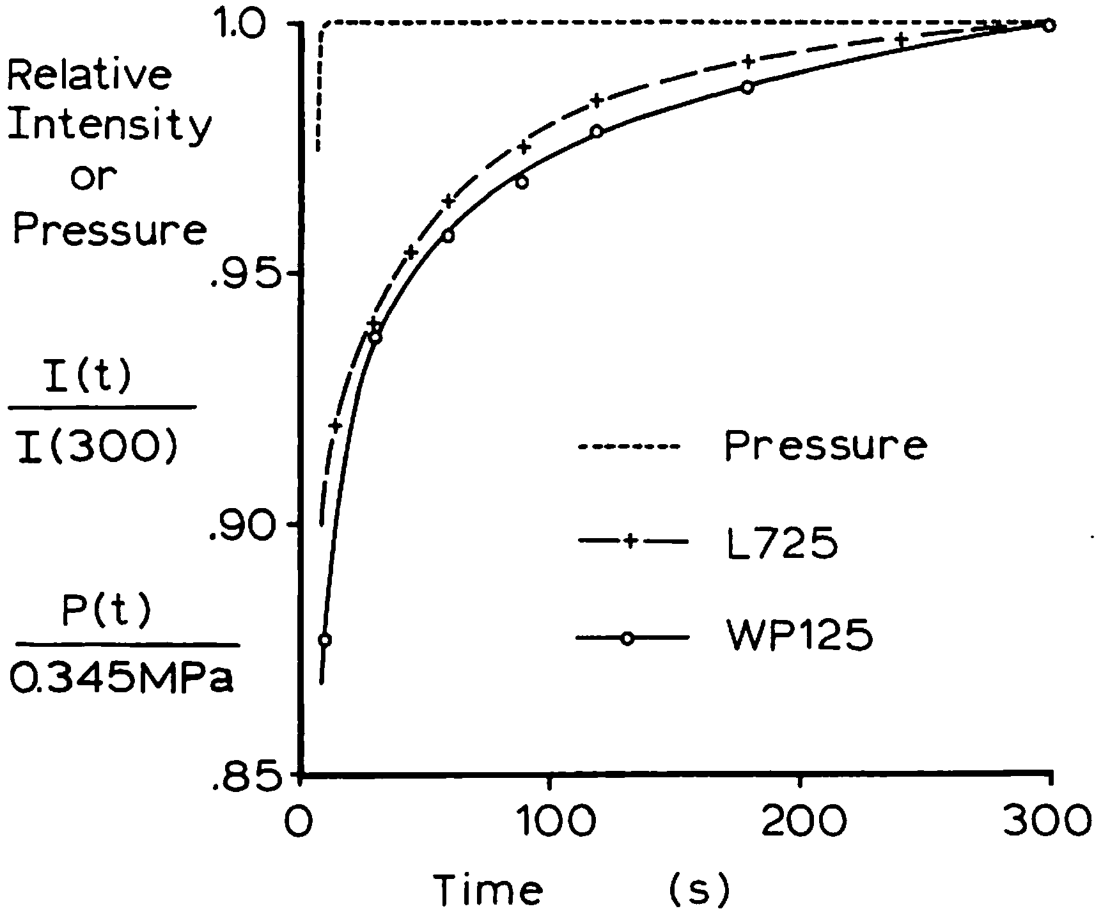


Figure 3.7: Drift in the pressure response of the LA-TS for the L725 and WP125 transducer membranes under an applied pressure of 0.345 MPa (50 psi).

relative to the dimensions of the features of interest, i.e., it is necessary to minimize the lateral spread inherent in the transmission of stresses and strains through a homogeneous elastic medium. (However, more sophisticated transducer materials could employ mechanically decoupled elements to avoid this problem. Such materials were not considered in this study.)

This interrelationship was demonstrated with the LA-TS using the CID camera. Figure 3.8a shows that WP125 + RB380 was a good combination for detecting details such as the columns in the building or the text on a penny. However, as illustrated in Figure 3.8b, this combination did not have sufficient depth range to render an identifiable imprint of the head on a nickel. Conversely, Figure 3.8d shows that increasing the total membrane thickness to 705 micrometers did provide the needed depth resolution to acquire a good imprint of the nickel, but, as Figure 3.8c shows, the sensor then lacked sufficient lateral or depth resolution to register the finer details on a penny.

A similar effect can be demonstrated regarding the size of the microtexture on the transducer membrane. The total membrane thickness was kept constant in Figures 3.9a and 3.9b, but WP125 and L725 were used as the transducer membranes, respectively. As expected, the larger microtexture of the L725 membrane resulted in a larger depth range, and thus registered deeper features compared to the performance of the other membrane.

3.3.2 Compact Tactile Sensor Array

The LA-TS unit was physically large and would only be useful in robotic applications in which the part to be examined or operation to be performed could be brought to the sensor. In contrast, the C-TSA unit was designed to enable convenient installation on the fingertips of robot grippers. Figure 3.10 shows various photographs of the C-TSA. A coherent array of optical fibers were used to convey the tactile imprint to a remote location where it was imaged onto a camera. The sensor head measured 3.5 x 3.5 x 1.0 cm. 295 fibers were arranged in a triangular array measuring 2.2 x 2.4 cm, each fiber being separated by a

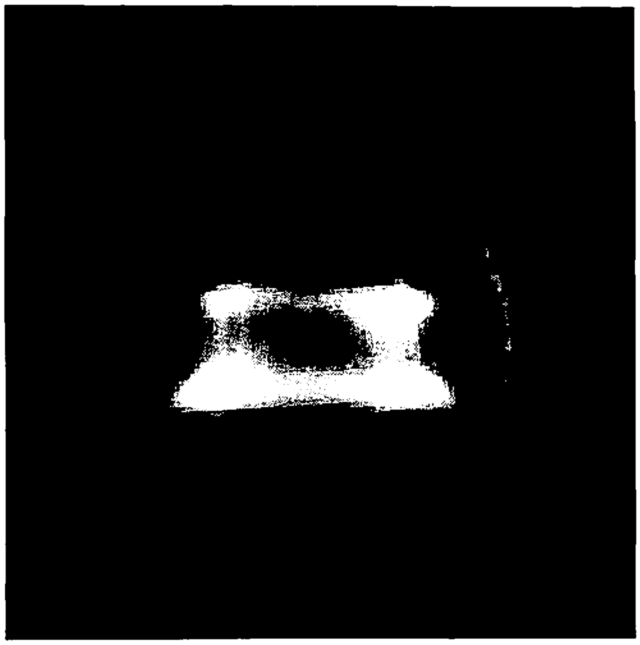


Figure 3.8a: LA-TS imprint of a penny under 28.9 N load. The WP125 transducer membrane and RB380 cover membrane was used.

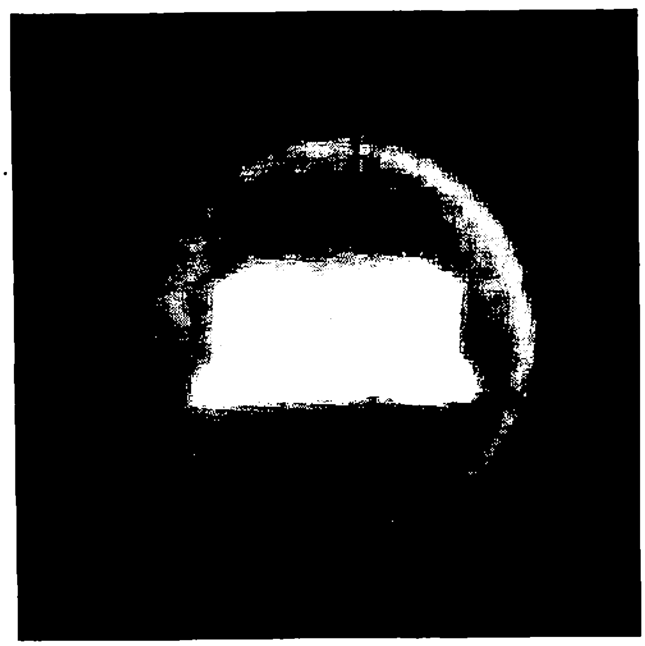


Figure 3.8c: Same as 3.8a except using the BN890 cover membrane.

Figure 3.8b: LA-TS imprint of the front of a nickel under 28.9 N load. The WP125 transducer membrane and RB380 cover membrane was used.

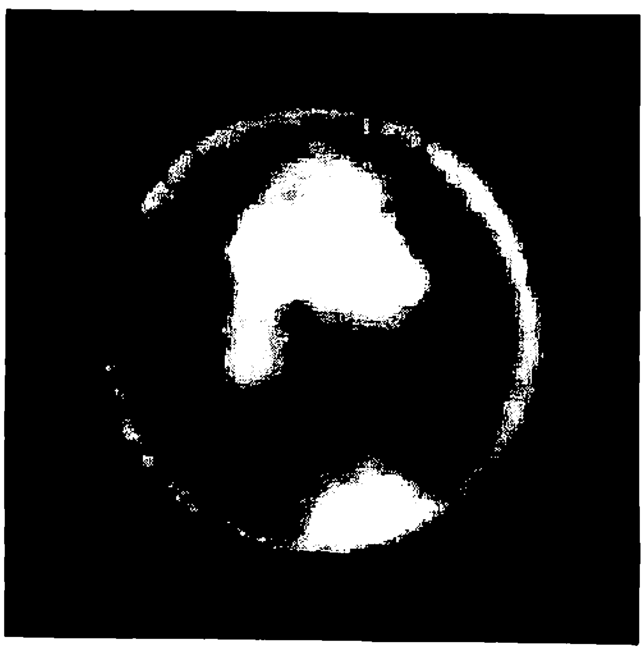


Figure 3.8d: Same as 3.8b except using the BN890 cover membrane.

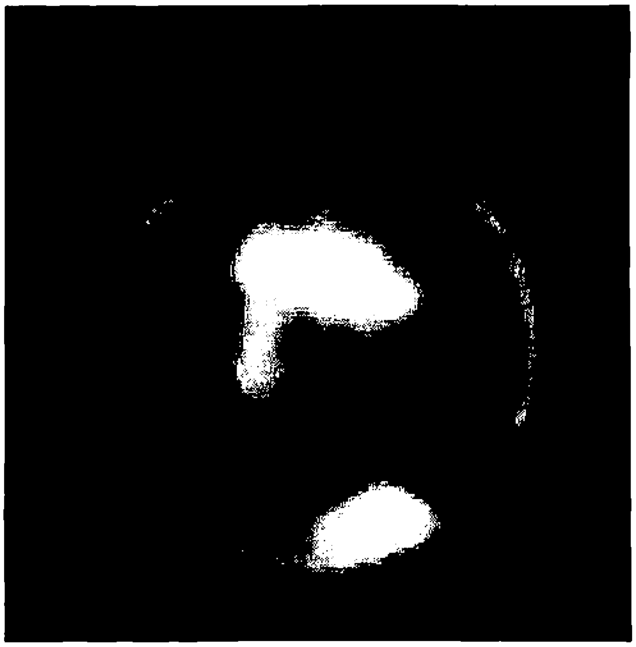


Figure 3.9a: LA-TS imprint of the front of a nickel under 28.9 N load. The WP125 transducer membrane and RB200 + RB380 cover membranes were used.

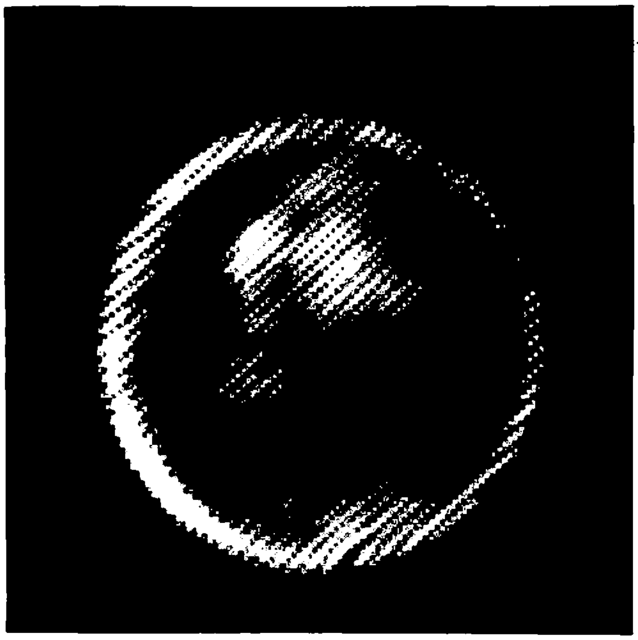


Figure 3.9b: Same as 3.9a except using the L725 transducer membrane and no cover membranes.

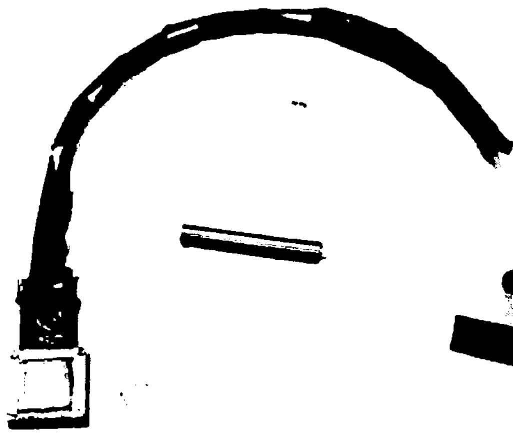


Figure 3.10a: C-TSA sensor head (left), illuminator coupler (center), and imprint display array (right).

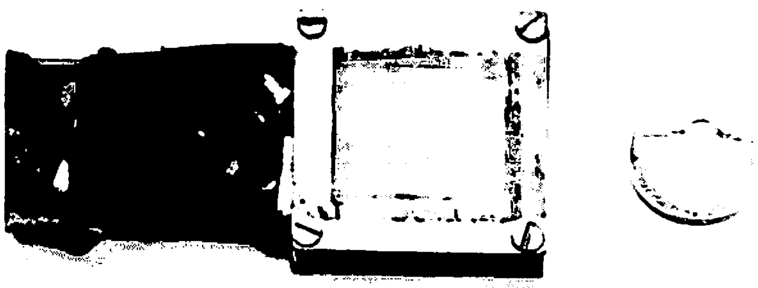


Figure 3.10b: Detailed view of C-TSA head.

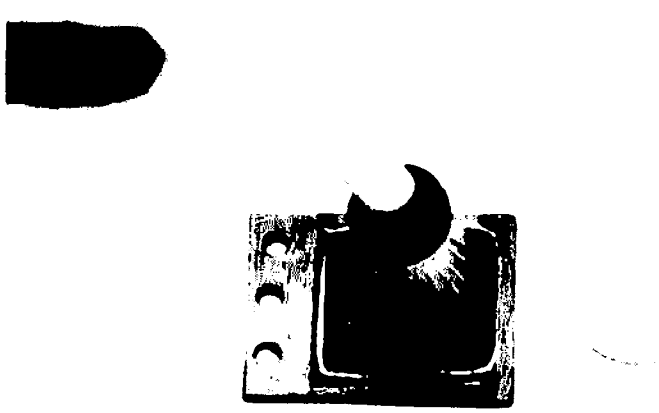


Figure 3.10c: Rear view of imprint display array.

distance of 1.6 mm. The effective taxel density was 54 taxels/sq-cm. Several tactile imprints are shown in Figure 3.11. The limiting factor in the resolution of a sensor based upon this approach was primarily determined by the optical fiber spacing.

3.3.3 Fingertip-Shaped Tactile Sensor

The third prototype was a non-planar, fingertip-shaped TIR tactile sensor (FS-TS). A photograph of this unit is shown in Figure 3.12. The sensor head consisted of a 1.6 cm diameter by 10 cm long test tube, where the hemispherical end and approximately 1 cm of the adjoining cylindrical wall were sensitive to touch. This area was internally monitored with a fish-eye lens, and the imprint transmitted to the camera through a 625-strand (25 x 25) coherent optical-fiber cable. This arrangement yielded an average taxel density of approximately 70 taxels/sq-cm. Figure 3.13 shows tactile imprints (taken from a TV monitor) of a "corner" tactile feature formed by three perpendicular planes, and of a "hole" tactile feature extracted from a hole in a plate. These imprints illustrate the economy of movement inherent in acquiring large quantities of tactile data with a fingertip-shaped sensor, since accurate pre-alignment of the sensor with the object surface is not necessary.

3.4 Conclusions from Past Work

The TIR tactile sensor was found to offer several important advantages over other technological approaches to the problem of acquiring distributed tactile-sense data:

1. Higher spatial resolutions (easily exceeding that of human fingertips);
2. Ability to easily fabricate the sensor in a variety of shapes (e.g., flat, curved, or fingertip shaped) useful to the solution of robotic problems;

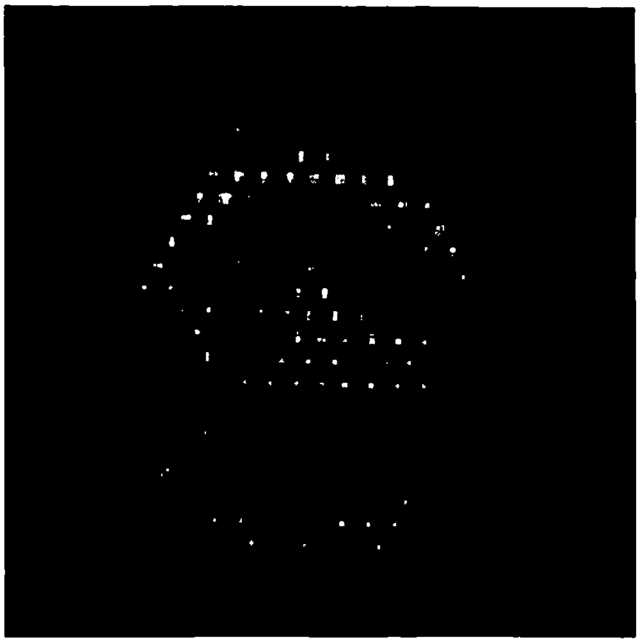


Figure 3.11a: C-TSA sensor imprint of the back of a nickel. The L725 transducer membrane with BP125 cover membrane was used.

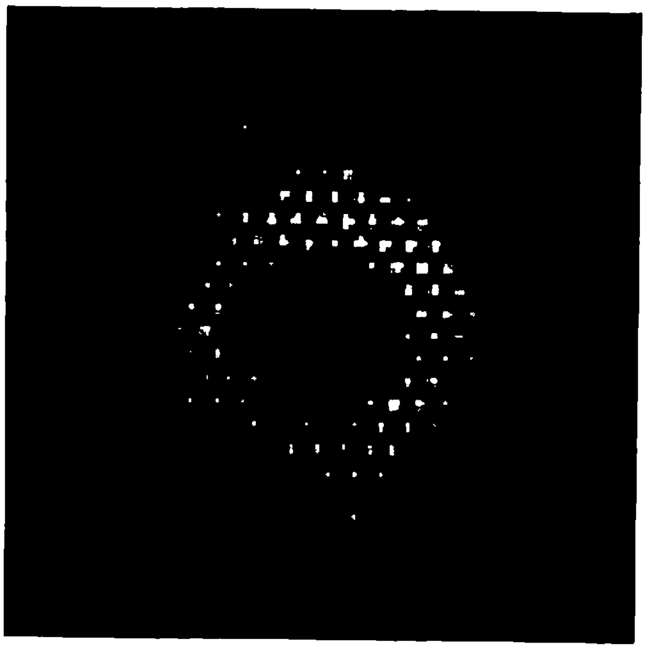


Figure 3.11b: Gear imprint (L725 + BP125).

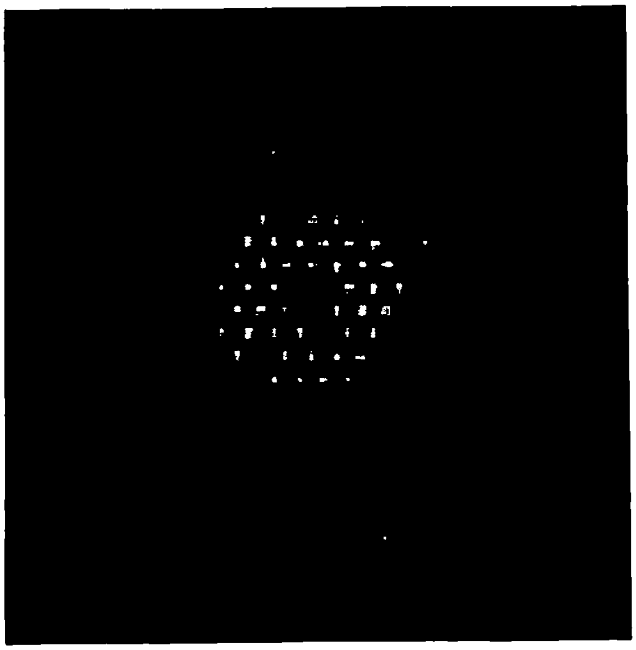


Figure 3.11c: Phillips-head screw imprint (L725 + BP125).

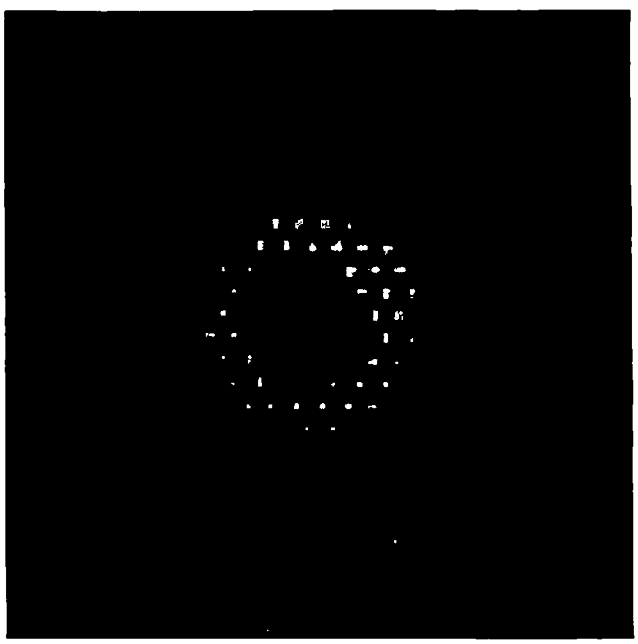


Figure 3.11d: Allen-head screw imprint (L725 + BP125).

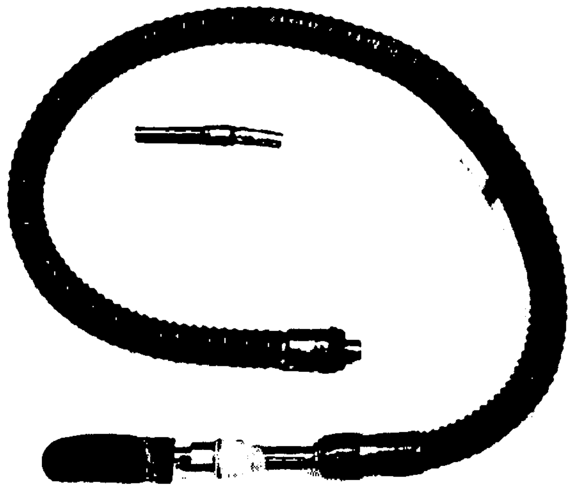


Figure 3.12: Photograph of the FS-TS unit. TIR sensor with fingertip-shaped sensing head and cylindrical boundary (bottom), display array and camera coupler (center), and illuminator coupler (top).

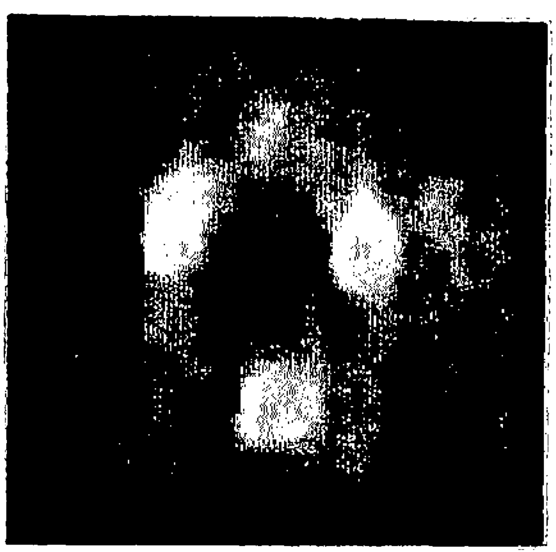


Figure 3.13a: FS-TS imprint of three perpendicular walls (WP25 and BN890)

Figure 3.13b: FS-TS imprint of a 12 mm diameter hole in a plate. (WP25 and BN890)

3. Potential ability to tailor the pressure and displacement response characteristics over a wide range.

The previous sections have documented the advantages of the TIR approach to tactile sensing. However, there are some significant disadvantages that must also be noted:

1. Real time application of tactile data may be more difficult due to lower data acquisition and processing rates associated with higher taxel densities;
2. Compact, multi-input optical-to-digital converters are not readily available, thereby requiring that the present generation of such converters be physically large compared to the sensor head;
3. The total number of fibers running to the sensor head is large, and may range from 1.1 to 1.5 times the number of taxels in the sensor. This situation results from the inability to optically multiplex the taxel signals (thereby requiring the use of one fiber for each taxel), and the need to provide a non-heating illumination source at the sensor head, e.g., an auxiliary optical fiber cable;
4. The ruggedness of the optical cabling between the sensor head and the optocoupler unit is not as great as that possessed by electrical cables used in other designs. Additionally, problems regarding the flexural fatigue life of the optical fibers may be encountered, especially in the joint regions where frequent and/or severe fiber flexing is most likely to occur.

High spatial resolution was listed as both an advantage and a disadvantage. The merits of high spatial resolution for tactile sensation are presently unresolved, and depend in part upon the intended application. Fine manipulative or object recognition tasks argue for tactile sensors with high spatial resolution, whereas rapid arm or hand control and fast sensory processing favor the converse situation. The attitude taken in this project was that high spatial resolution sensors should be explored as they are potentially very useful in robotic applications, but the taxel density in such sensors should be reasonably limited to a value comparable to the effective taxel density present in the human fingertip, e.g., approximately 100 taxels/sq-cm (Loomis [15]).

Chapter 4

TIR Tactile Sensing System

4.1 Introduction

The following sections discuss the design, fabrication, and operation of the major components within the TSS. A block diagram of the preprocessor module and the TSA module is shown in Figure 4.1. The functions of the preprocessor module were to acquire tactile data, perform automatic calibration and data correction, extract simple tactile features such as the total force and first moment, and monitor various thresholds. The TIR tactile sensor technology was chosen for the TSA module for the reason that prior work had indicated the approach held significant promise for robotic application, and the TSS project provided an excellent opportunity to further the evaluation and development of the TIR sensor. It should be noted that the design of the preprocessor module was quite general and could accommodate any of the TSA technologies mentioned in Chapter 2.

4.2 TSS Preprocessor Module

The function of the preprocessor module (Figure 1.1) was to acquire data from the TSA module and to preprocess the data for more convenient presentation to the host. The microcomputer chosen for the TSS preprocessor was the Digital Equipment Corporation DCT-11EM evaluation board. A photograph of the DCT-11EM is shown in Figure 4.2. The major features of this board were: T-11 16-bit microprocessor (7.5 MHz); 16 KByte keypad and console monitor; 7 KByte RAM (expandable to 40 KByte); two serial RS232C ports with speeds from 110 to 19200 baud; an 8-bit parallel I/O port; and access to the internal data lines via a 60-pin connector. The T-11 was chosen primarily for its compatibility with existing DEC equipment in the LPR, and its programmability in PDP-11 assembly

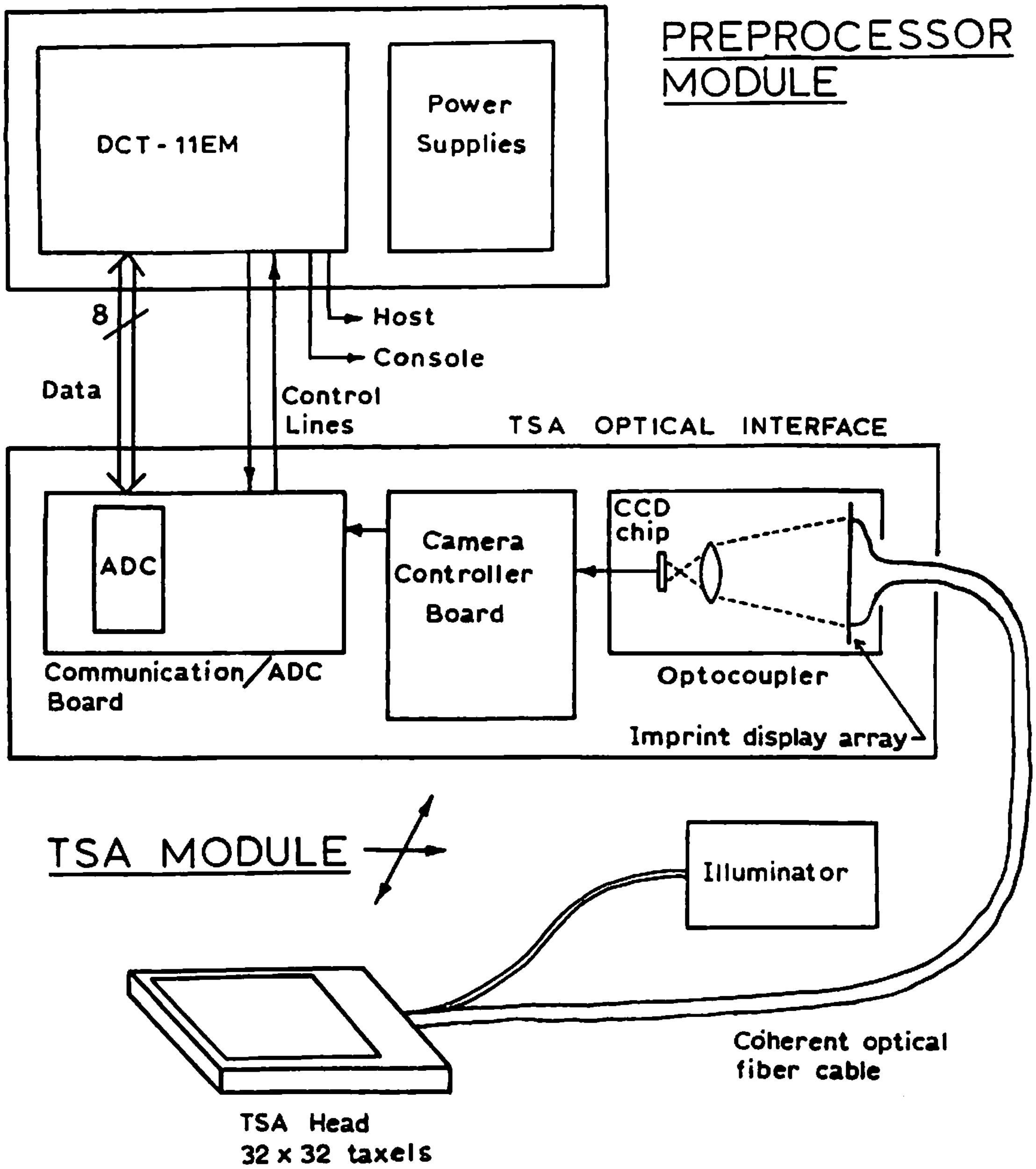


Figure 4.1: Block diagram of the TSS showing the preprocessor module and the TIR-TSA module.

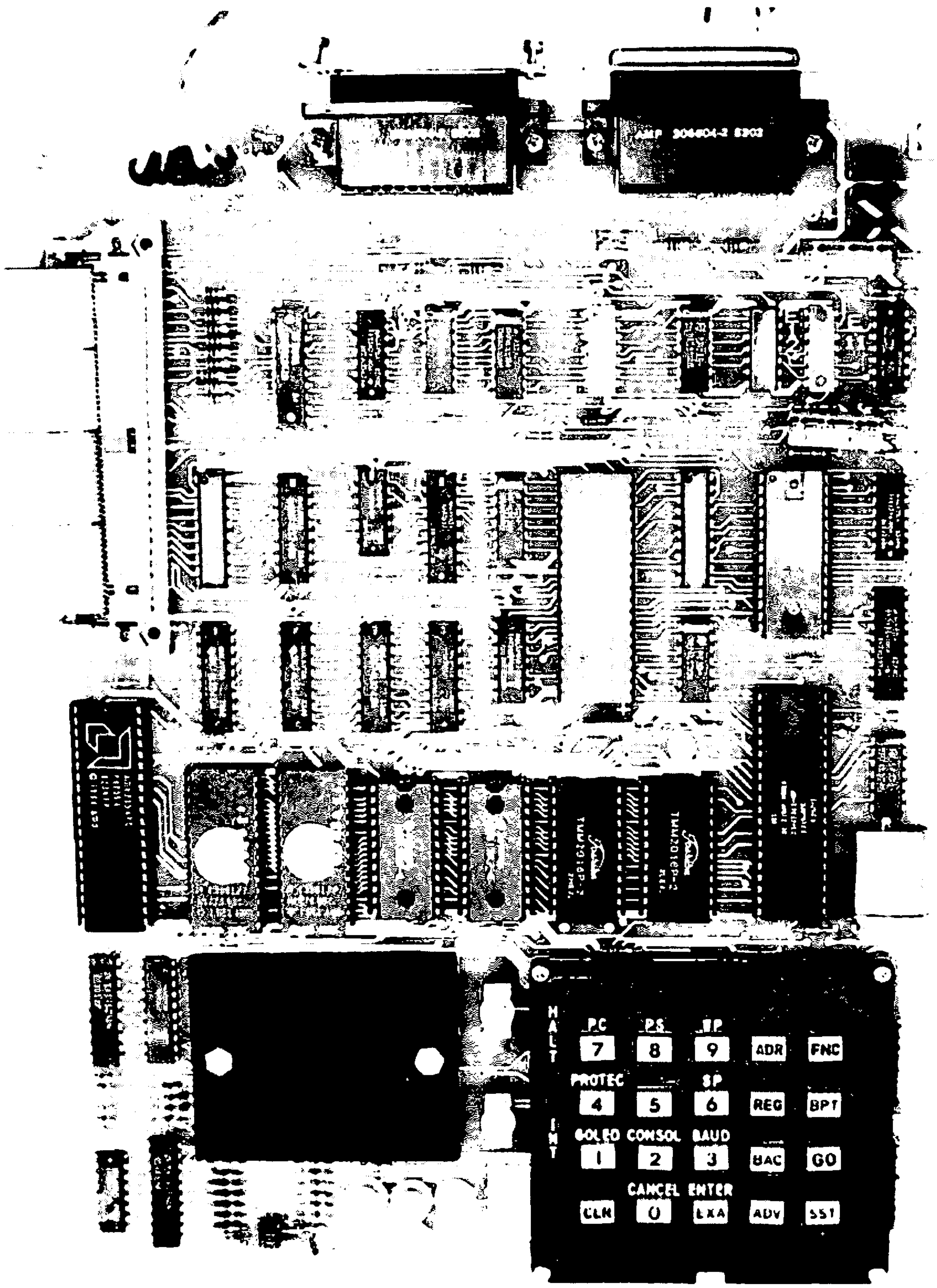


Figure 4.2: The TSS preprocessor was based upon the DCT-11EM evaluation board (Digital Equipment Corporation) which used the T-11 microprocessor chip.

language. This enabled past laboratory experience to be used to rapidly develop a working system.

The DCT-11EM was housed inside a fan-cooled, cabinet-mountable enclosure along with the power supplies required by the microcomputer, optical interface, and the sensor illuminator. The power supplies were mounted beneath the microcomputer and were separated by a 3.2 mm thick aluminum shielding plate. This is illustrated in Figures 4.3 and 4.4. The electrical schematics for power and communication are shown in Figures 4.5 and 4.6, respectively, and the connector pin definitions are listed in Table 4.1.

Communication connections with the optical interface unit included an 8-bit parallel data line from the interface to the DCT-11EM, an ANF (acquire new frame) command line from the DCT-11EM to the interface, and an INTB (interrupt) line from the interface to the DCT-11EM. When the latter is enabled, an interrupt routine is called which reads data on the parallel input port and stores each taxel if it is valid. The interrupt sequence is automatically terminated after one frame (1024 taxels) of tactile data has been read. Additional details concerning tactile data acquisition may be found in Section 4.3.3 and Chapter 5.

Communication between the host and the TSS preprocessor occurred in one of two modes. In the Console Control (CC) mode, the user interacted with the system from a console (video terminal) keyboard, and thereby directed acquisition of data or viewed the processed results. Alternatively, in the Host Control (HC) mode, the robot controller (host) treated the TSS simply as a another I/O device with the specialized function of providing tactile data in response to simple numerical command codes. The user interacted with the TSS only to the degree

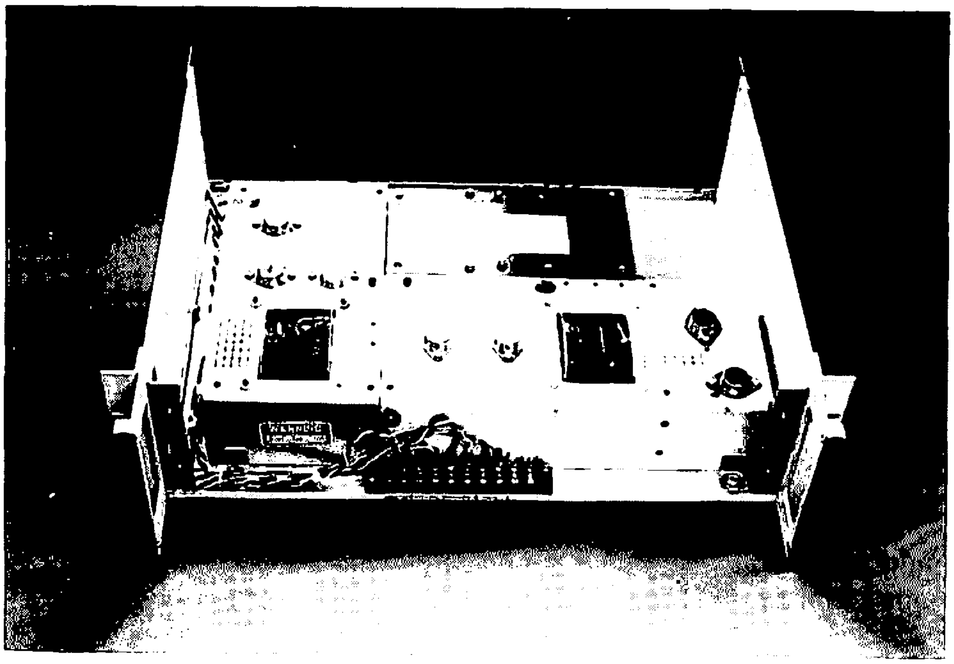


Figure 4.3: Power supplies located within the preprocessor module for the microcomputer, optical interface, and illuminator.

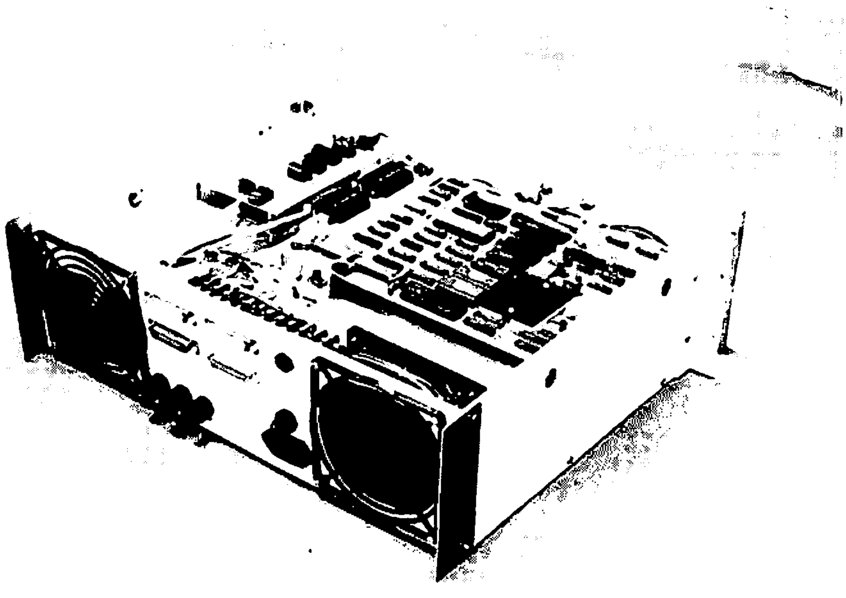


Figure 4.4: Rear view of the uncovered preprocessor module. The DCT-11EM is mounted above the power supplies.

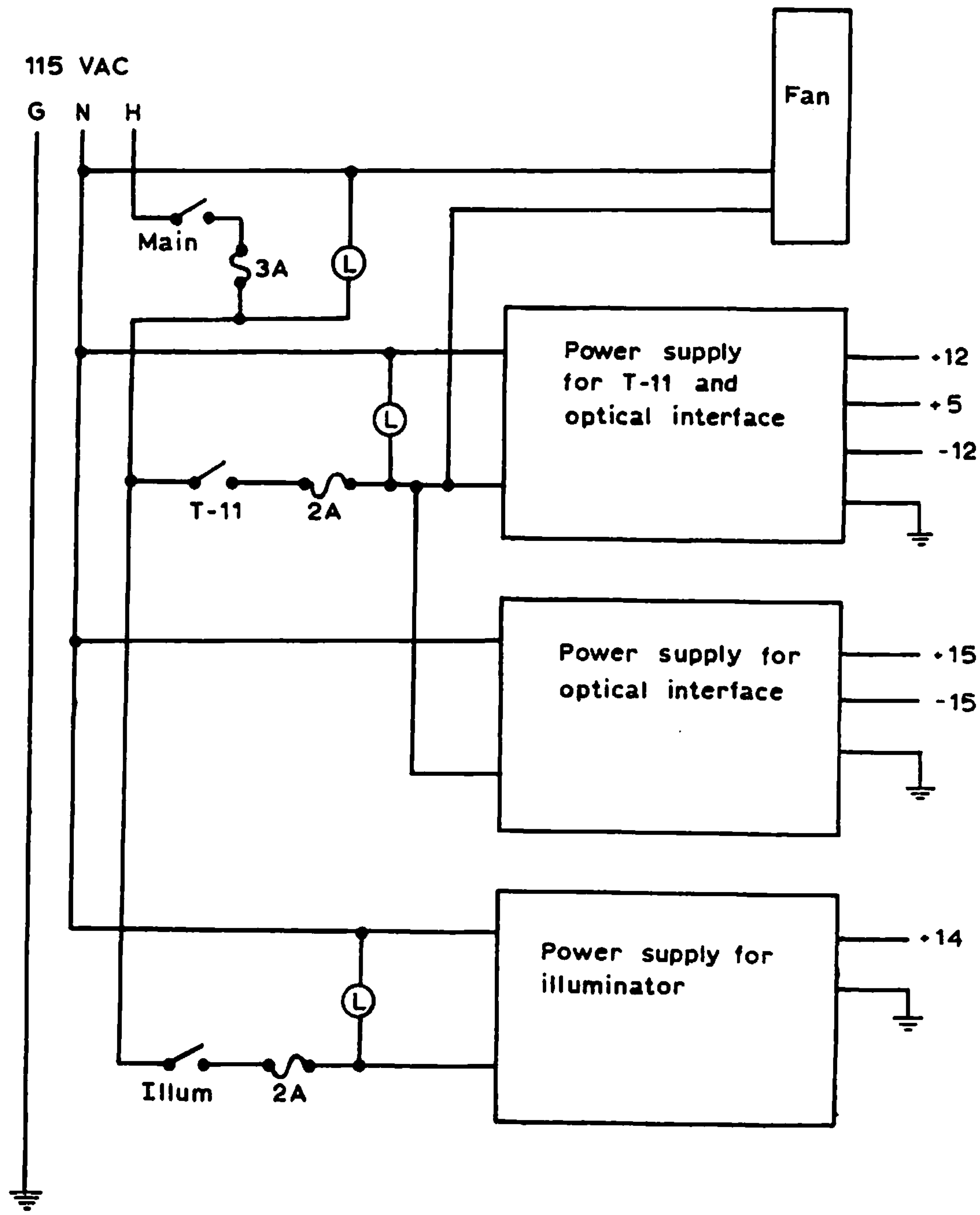


Figure 4.5: Electrical schematic of the TSS power supply circuit.

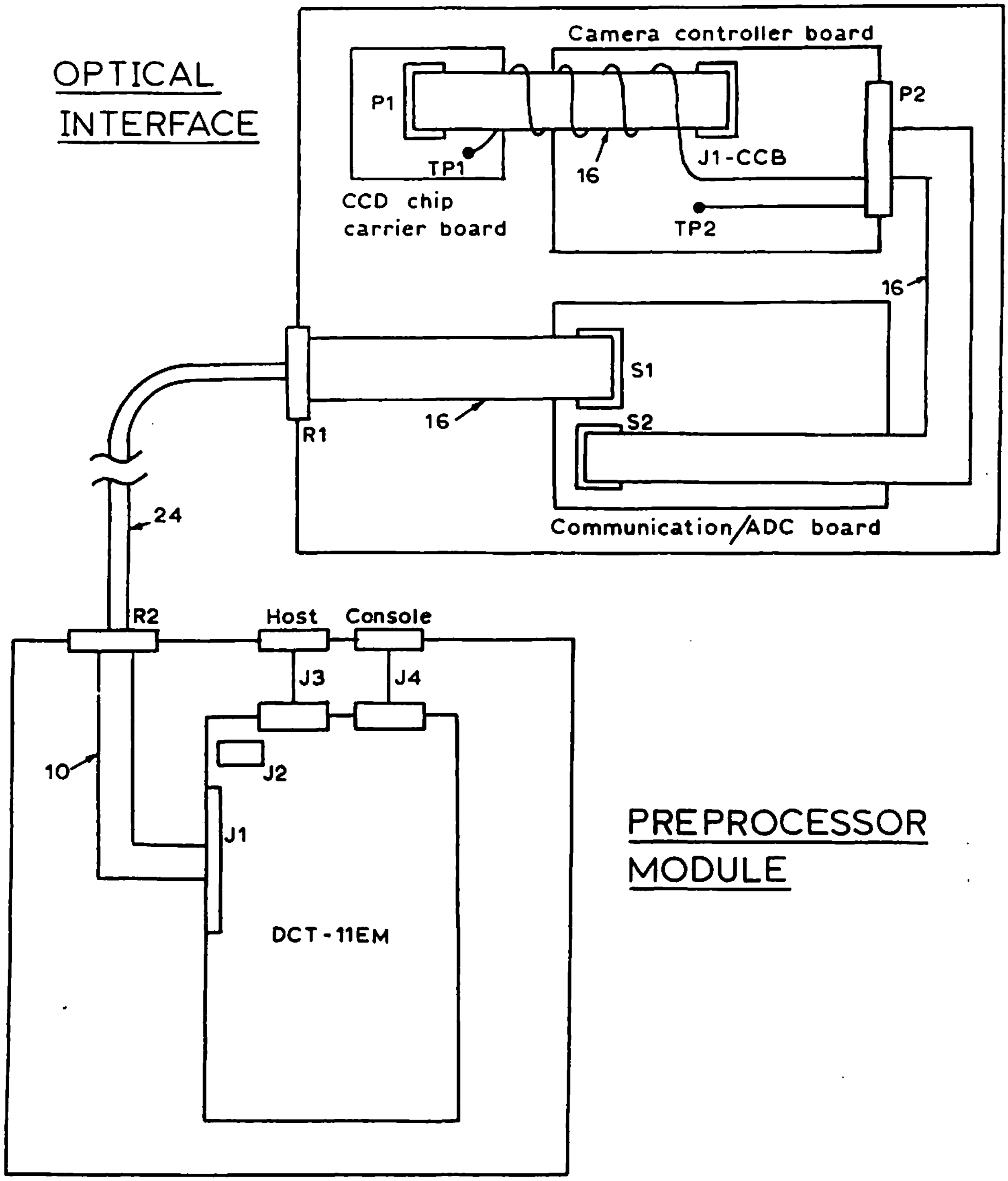


Figure 4.6: Schematic of the communication and control connections for the preprocessor and optical interface.

Table 4.1

Connector pin definitions within the TIR-TSS

FUNCTION	CONNECTOR P2	S2	S1	R1,R2	J1(DCT)
Ground	1-22	1,12	1,12	19-21,1	25,41,53-59
MSB(bit 7)	—	—	2	2	34
(bit 6)	—	—	3	3	33
(bit 5)	—	—	4	4	32
(bit 4)	—	—	5	5	31
(bit 3)	—	—	6	6	30
(bit 2)	—	—	7	7	29
(bit 1)	—	—	8	8	28
LSB(bit 0)	—	—	9	9	27
INTB	—	—	10	10	38
LED	—	—	11	11	51
+15v	—	—	16	15	—
+12v	—	—	—	16	—
+ 5v	E	13,15	13,15	17,18	—
- 5v	—	—	—	22,23	—
-12v	—	—	—	24	—
-15v	Y	14	14	25	—
Video	N	4	—	—	—
Sample ready	R	2	—	—	—
EOF	T	3	—	—	—
EOL	B	5	—	—	—

provided by the robot control program running on the host. The communication line connections on the DCT-11EM for each of these modes is shown below:

CC mode:
- "Host" connector (J3) is connected to the host,
- "Console" connector (J4) is connected to the console.

HC mode:
- "Host" connector (J3) is not used.
- "Console" connector (J4) is connected to the host.

The present version of the TSS does not store the tactile data acquisition and processing routines in ROM. Instead, these must be down-loaded from the host. The TSS must therefore be initially configured in the CC mode, the program downloaded, and afterwards the connectors switched to place the TSS into the HC mode.

4.3 Interface Module

The TIR-TSA module (Figure 4.1) consisted of the TIR-TSA sensor head, the sensor illuminator, and the optical interface. A detailed discussion concerning these components is presented in the following three sections.

4.3.1 TIR Tactile Sensor Head

A photograph of the TIR-TSA head is shown in Figure 1.2, whereas the general characteristics are listed in Table 4.2. The following paragraphs describe in detail the procedures used to fabricate the TIR-TSA head.

The internal constitution of the TSA was very similar to the C-TSA described earlier (Figure 3.10). The primary components of the sensor head were the sensor body, illuminator cable assembly, glass plate, tactile transducer assembly, optical cable clamp, and imprint display array. An exploded view of these components and their relationships is shown in Figure 4.7. The sensor head had

Table 4.2

Characteristics of the TIR-TSA

Sensor size	Length: 64 mm Width: 44 mm Thickness: 10 mm
Active area	31 x 31 mm
Number of Taxels	1024
Taxel Density	100 taxels/sq-cm
Force range	0 - 0.414 N/taxel (0 - 42 gm)
Pressure range	0 - 0.414 MPa (0 - 60 psi)
Force resolution	One part in 32 (5-bits)
Spatial resolution	2 mm
Optical Cable Length	1.5 m (60 inches)

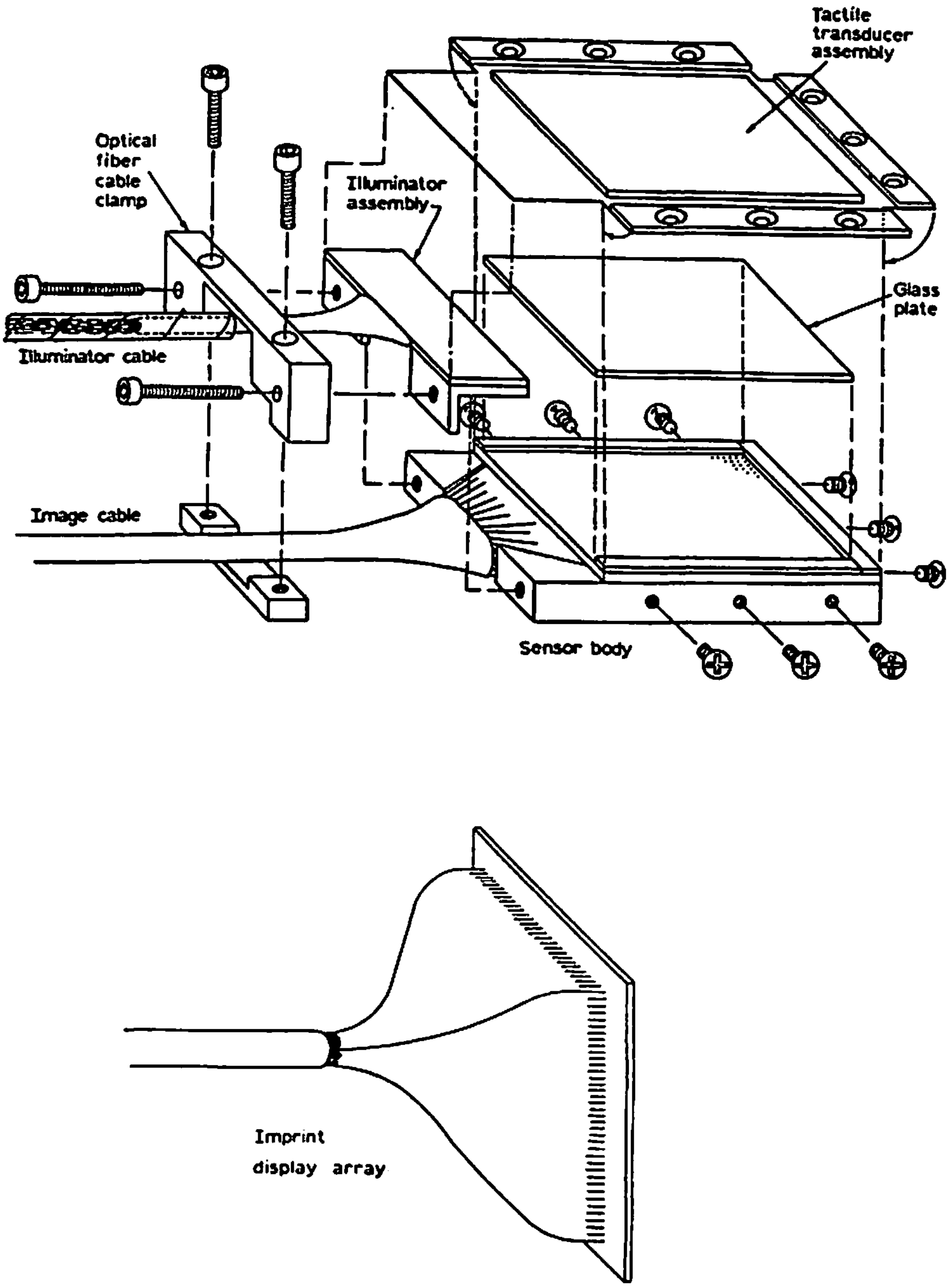


Figure 4.7: Exploded assembly drawing of the TSA head showing the primary components.

an integral coherent optical-fiber cable terminated on the viewing end with an imprint display array. An epoxy body encapsulated the fibers and supported the glass plate and fiber alignment board. The sensing surface was designed to be removable and replaceable with only moderate difficulty, and to permit selection of various pressure ranges and sensitivities by choosing an appropriate combination of transduction and cover membranes.

Fabrication of the sensor body began with cutting 1024 2-meter lengths of 0.25 mm diameter "Crofton" optoelectronic-grade plastic optical fiber (DuPont Corporation, Wilmington, DE) for the image cable. These fibers were threaded through the sensor-head and display fiber-alignment-masks (Figure 4.8) with the aid of the fiber assembly jig (Figure 4.9). The fibers were threaded one at a time through both array boards, and the long end of each fiber draped over the back of the jig to keep it out of the way. Once threading was completed, the fibers were epoxied into position in the lower fiber alignment board with low-viscosity epoxy (Ealing Company, South Natick, MA), and fiber pigtails protruding below the board were trimmed away with a sharp razor once the epoxy cured.

Next, the fibers were bent 90 degrees with the aid of the fiber bending jig (Figure 4.10). The sensor alignment mask was inserted into the jig and the emerging fiber bundle was tightly gathered and centered. Low-viscosity epoxy was then poured into the jig to encapsulate the fibers. After curing, the resultant assembly was fixed into the sensor head frame (Figure 4.11) with five-minute epoxy, and the fiber alignment board trimmed to the same external dimensions as the frame. Five-minute epoxy was then used to attach the glass plate containment strips (Figure 4.12) to the top of the sensor body.

Once the fibers were fixed in the sensor head, the imprint display alignment board was moved to the other end of the fiber bundle. The fiber bundle was loosely covered with three to four layers of black polyester fabric stitched together to form a tube (Figure 4.13). This preserved the flexibility of the bundle and provided a shield from ambient light, in addition to affording a moderate degree of mechanical protection. The fibers were then fixed into position in the display

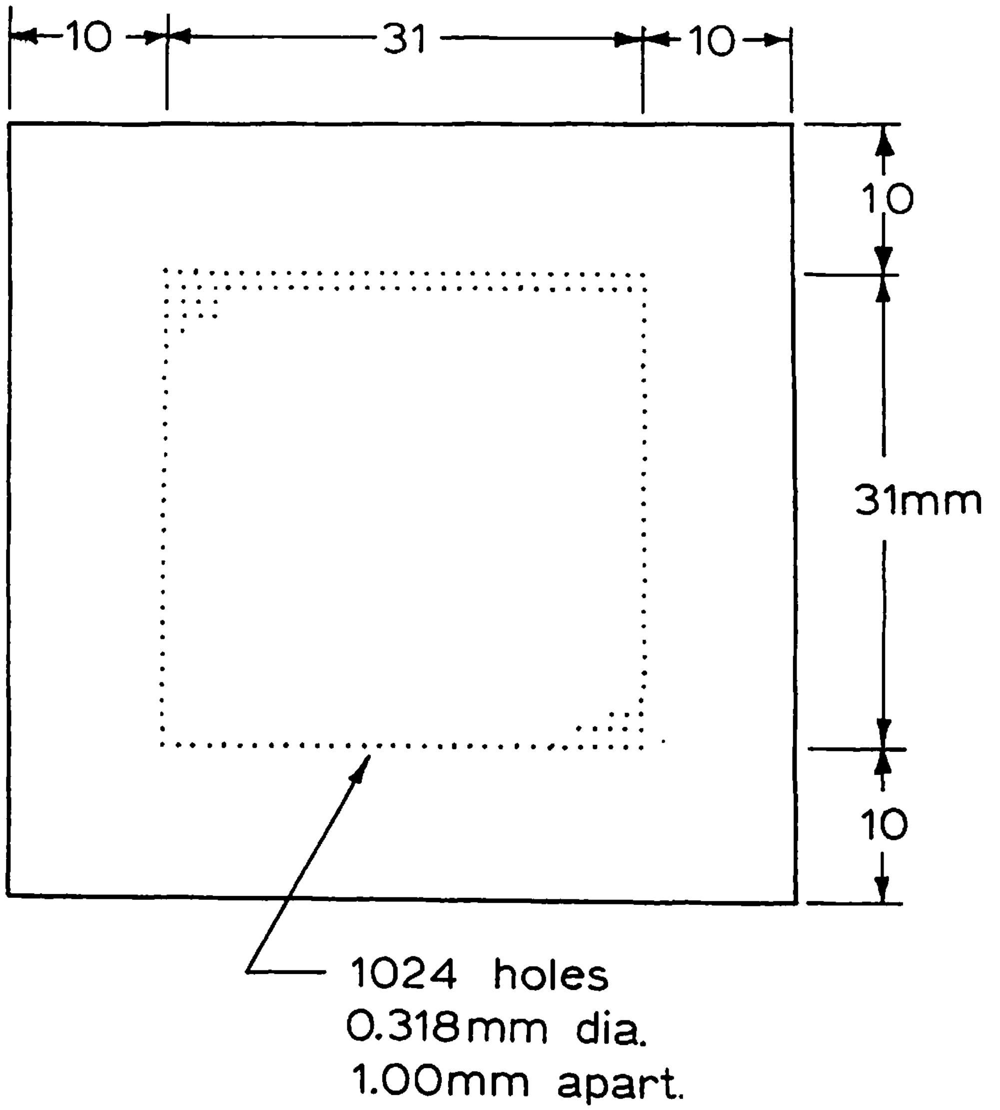


Figure 4.8: Optical fiber alignment boards. Two are required. The material is printed-circuit-board epoxy-substrate 1.55 mm thick.

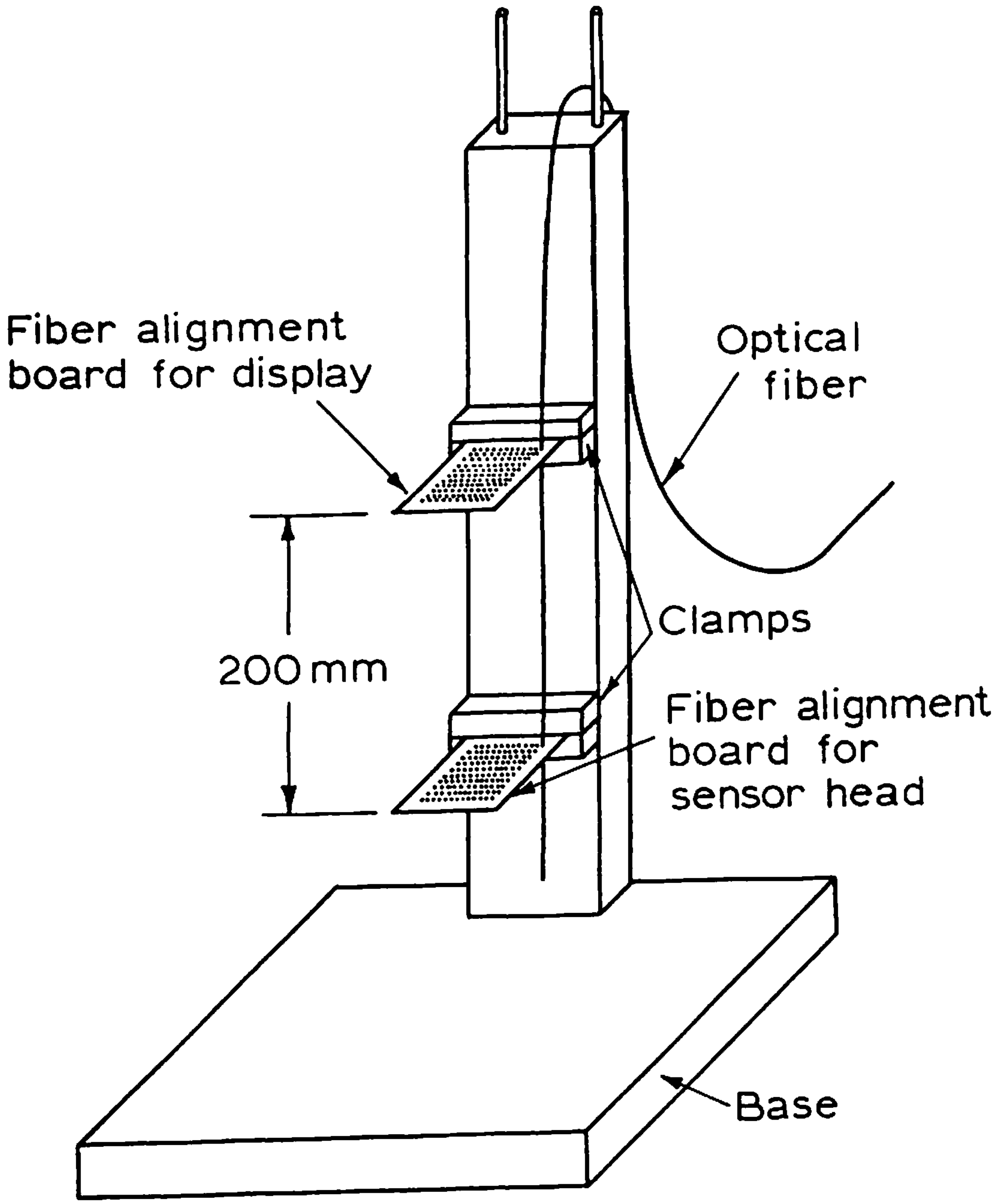


Figure 4.9: Image cable assembly jig.

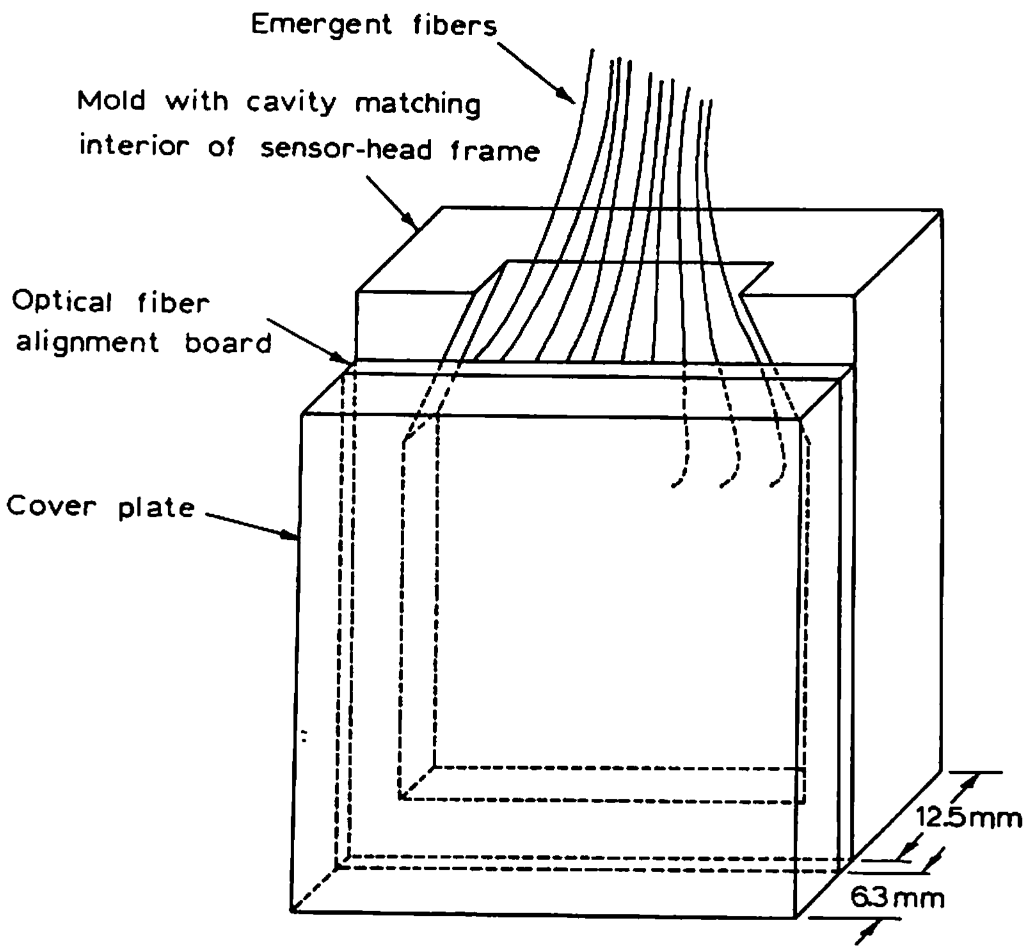


Figure 4.10: Polyethylene jig for bending and encapsulating the plastic optical fibers during fabrication of the TSA head.

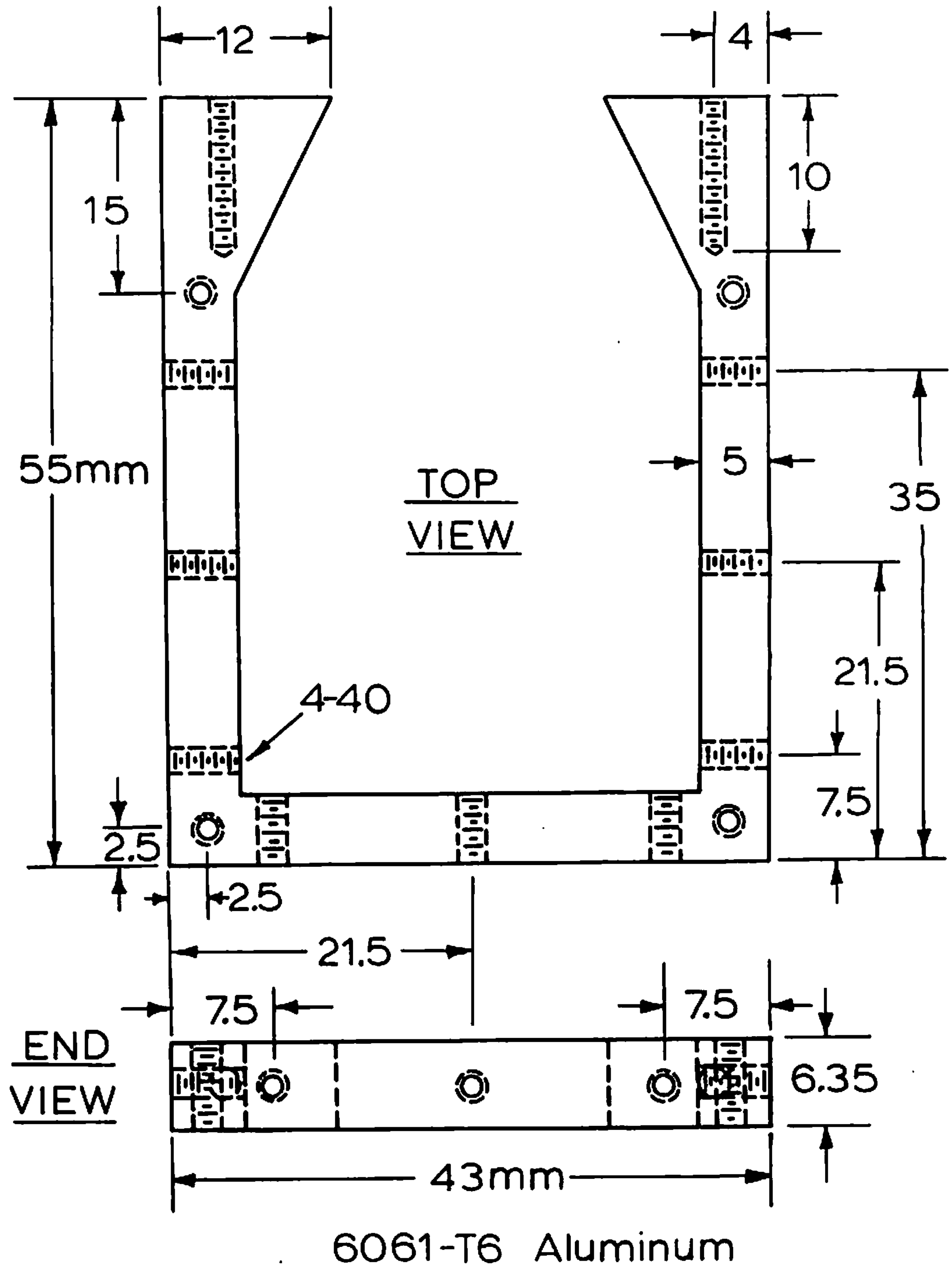


Figure 4.11: Sensor head frame.

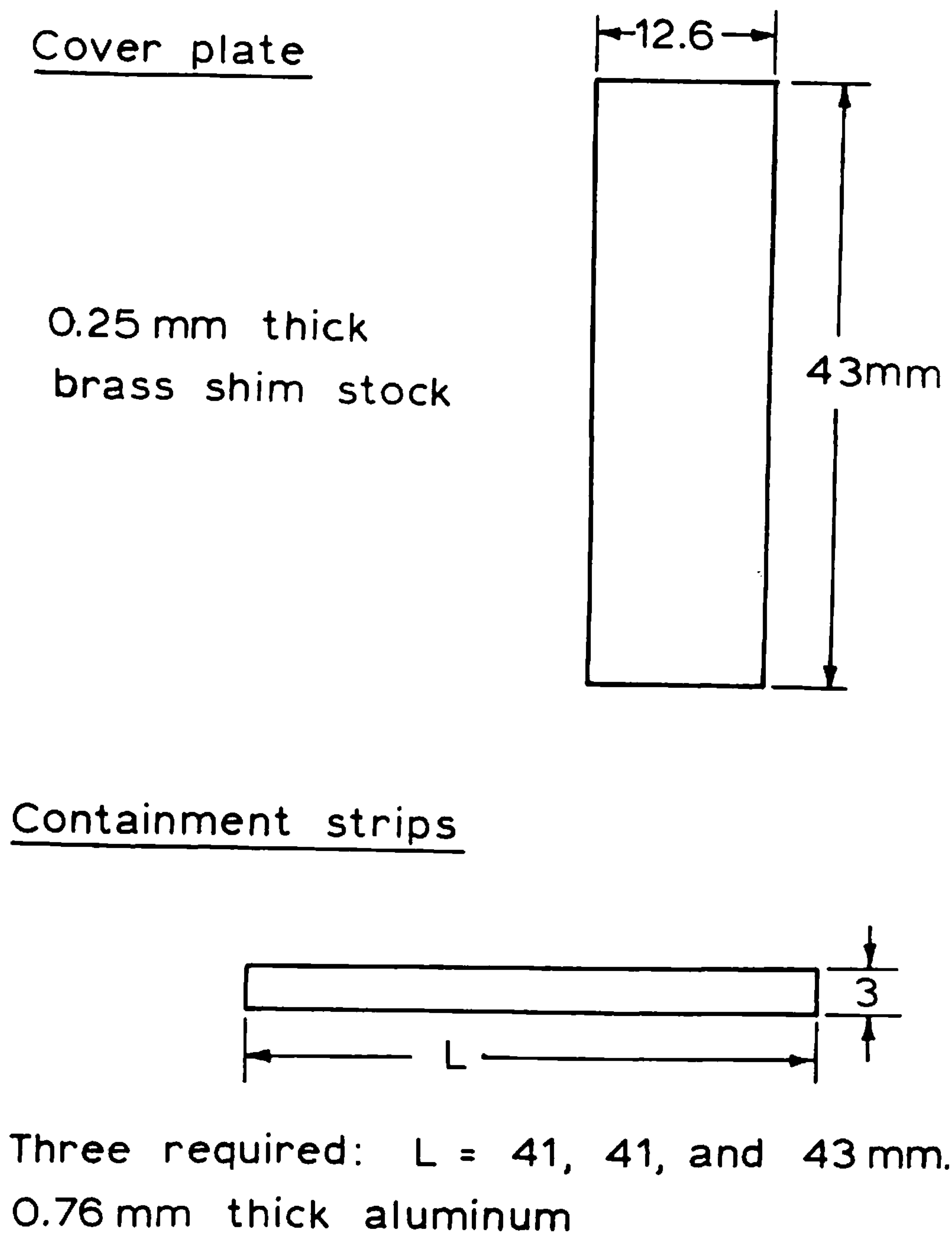


Figure 4.12: Cover plate for illuminator assembly, and glass plate containment strips.

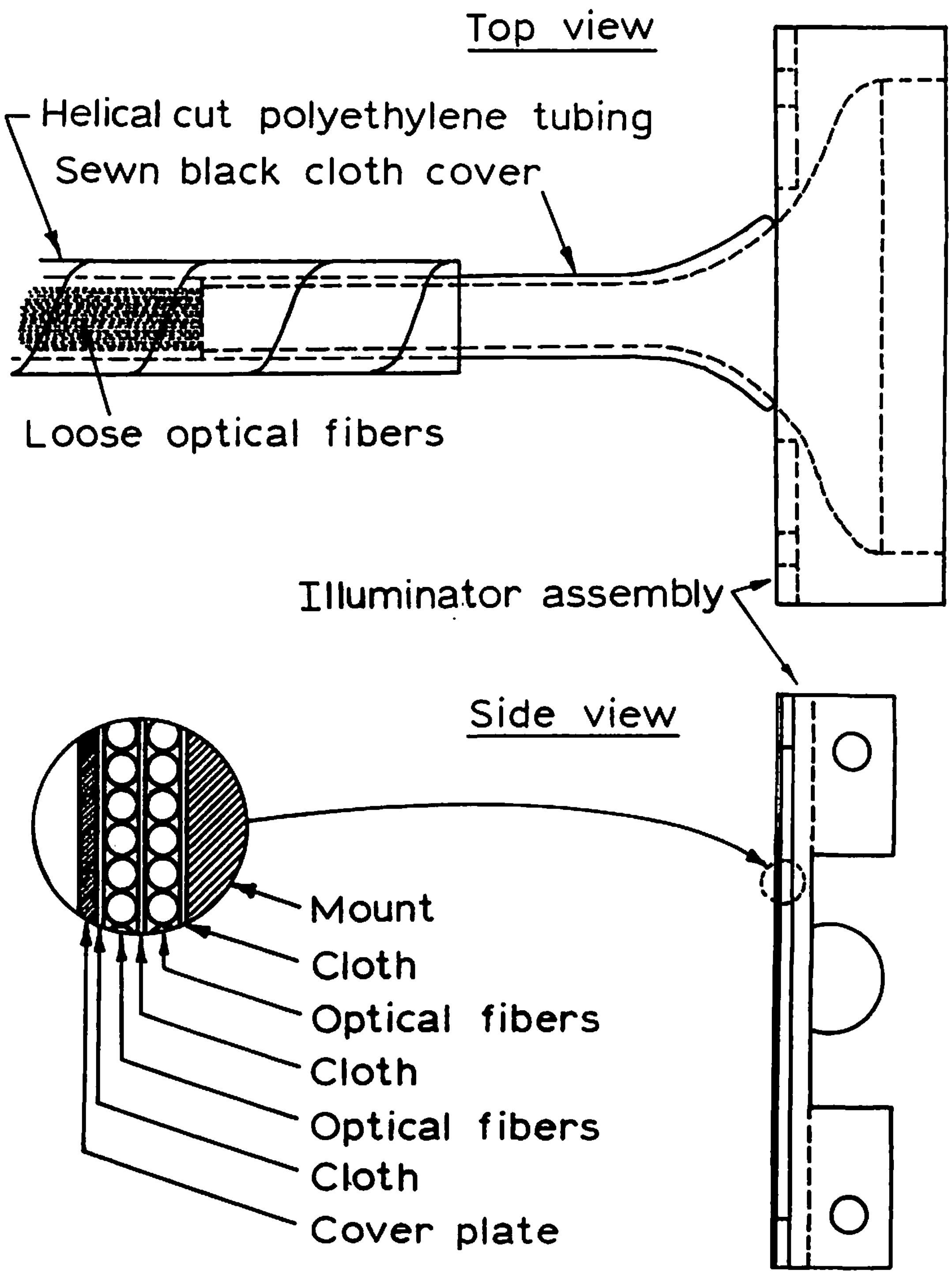


Figure 4.13: Sensor-end of illuminator cable.

array with low-viscosity epoxy.

Fabrication of the illuminator cable assembly (Figures 4.13 and 4.14) proceeded with the attachment of a two-level optical-fiber linear array to the illuminator cable base (Figure 4.15). Each layer of the illuminator cable contained 140 optical fibers. These fibers were gathered and wrapped in a short sewn tube of black polyester fabric. The remainder of the cable was covered with helically-cut polyethlyene tubing. A cover plate (Figure 4.12) was placed on the top of the illuminator array and the intervening void filled with five-minute epoxy. The illuminator was embedded into the fiber spacing sleeve and illuminator coupling tube (Figure 4.16) with 5-minute epoxy. PVC (polyvinylchloride) shrink tubing was then used to attach the polyethylene sheath to the coupler tube (Figure 4.14).

The side edges of the glass transducer plate (Figure 4.17) were covered with aluminum foil to reduce light leakage losses, and the end opposite the input end was covered with black, absorbent cloth to prevent spurious illumination of detection fibers located at the edge. Any play between the glass substrate and the containment strips was accommodated with cardboard shims. (Note that any rigid transparent material may be used in place of the glass, e.g., acrylic plastic.)

The tactile transducer assembly (Figure 4.18) consisted of WP125 transducer membrane material (Table 3.1), a black polyester cloth to eliminate noise from ambient light, a woven nylon-fabric structural membrane, and an outermost cover membrane consisting of BN890 material (Table 3.1) to provide good frictional properties and ruggedness. The nylon membrane was the primary means of attachment to the sensor body and also provided resistance to lateral deformation or sliding due to shear forces applied to the sensor surface. The top three layers were bonded together with rubber cement, whereas the transducer membrane was simply held in place by the top layers. Three cover clamps (Figure 4.19) were attached to the edges of the nylon structural membrane with a particularly adherent and flexible brand of rubber adhesive called "Shoe Goo" (Eclectric Products Corporation). The fourth side of the nylon membrane was attached with rubber cement directly to the cover plate on the illuminator assembly (Figure 4.13).

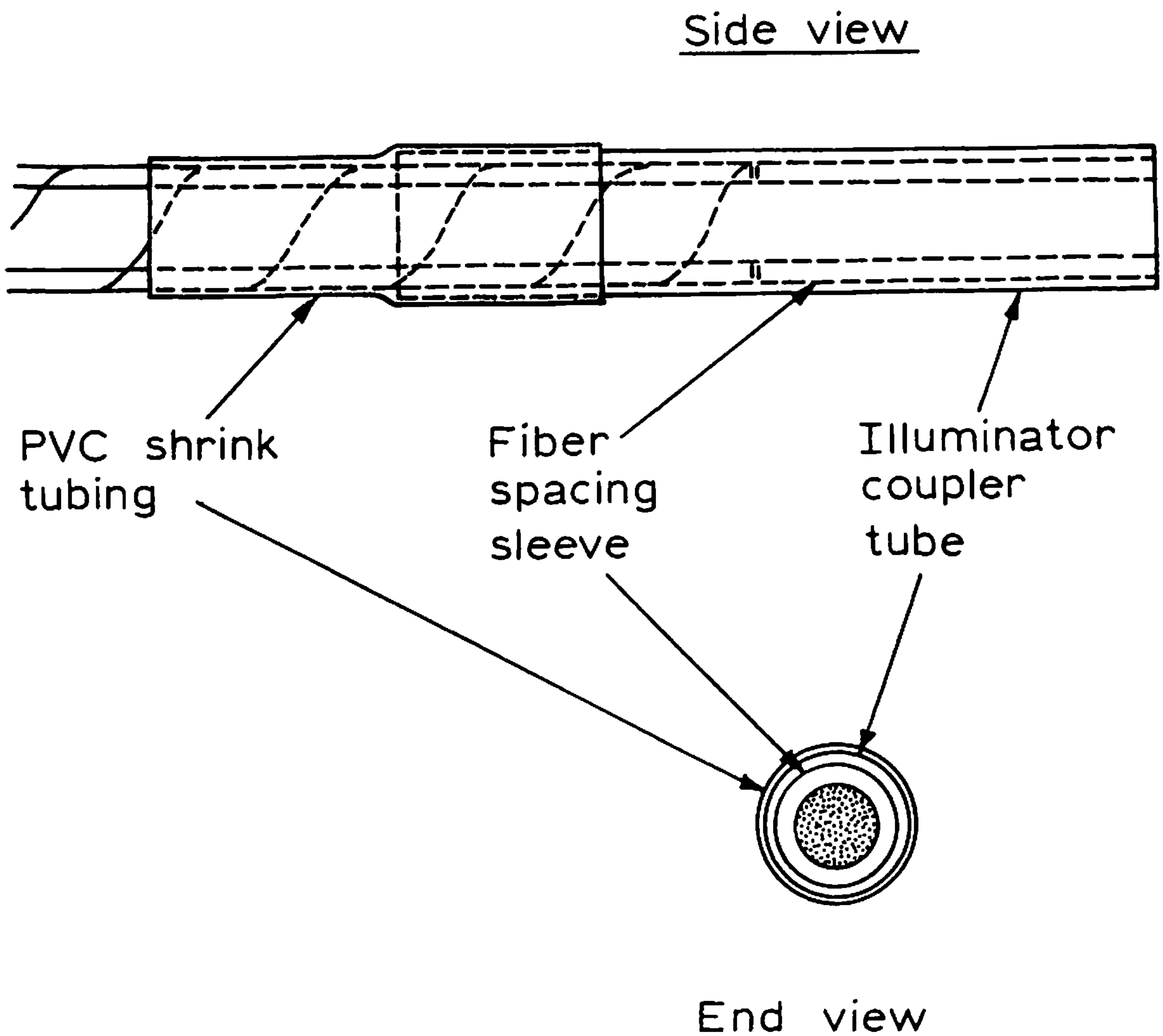


Figure 4.14: Illuminator coupler tube.

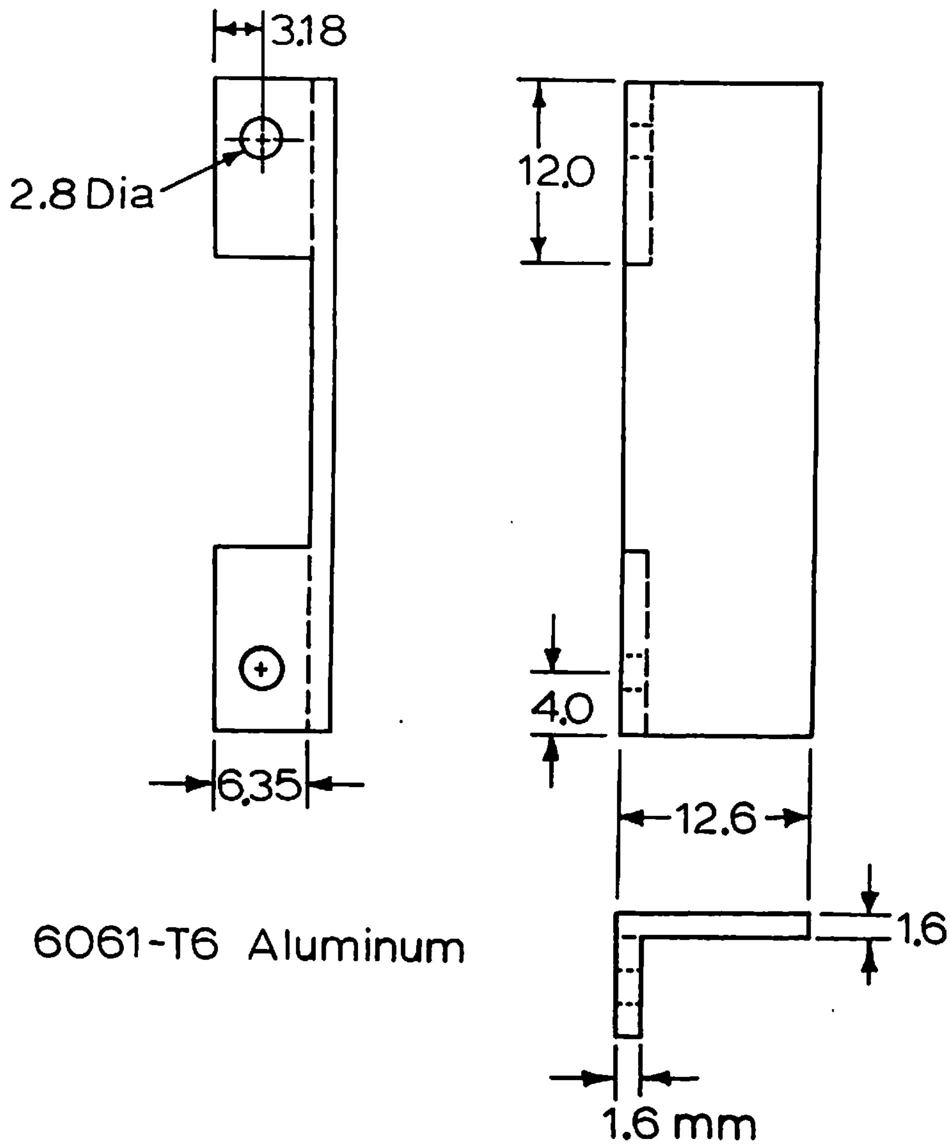


Figure 4.15 Illuminator cable base.

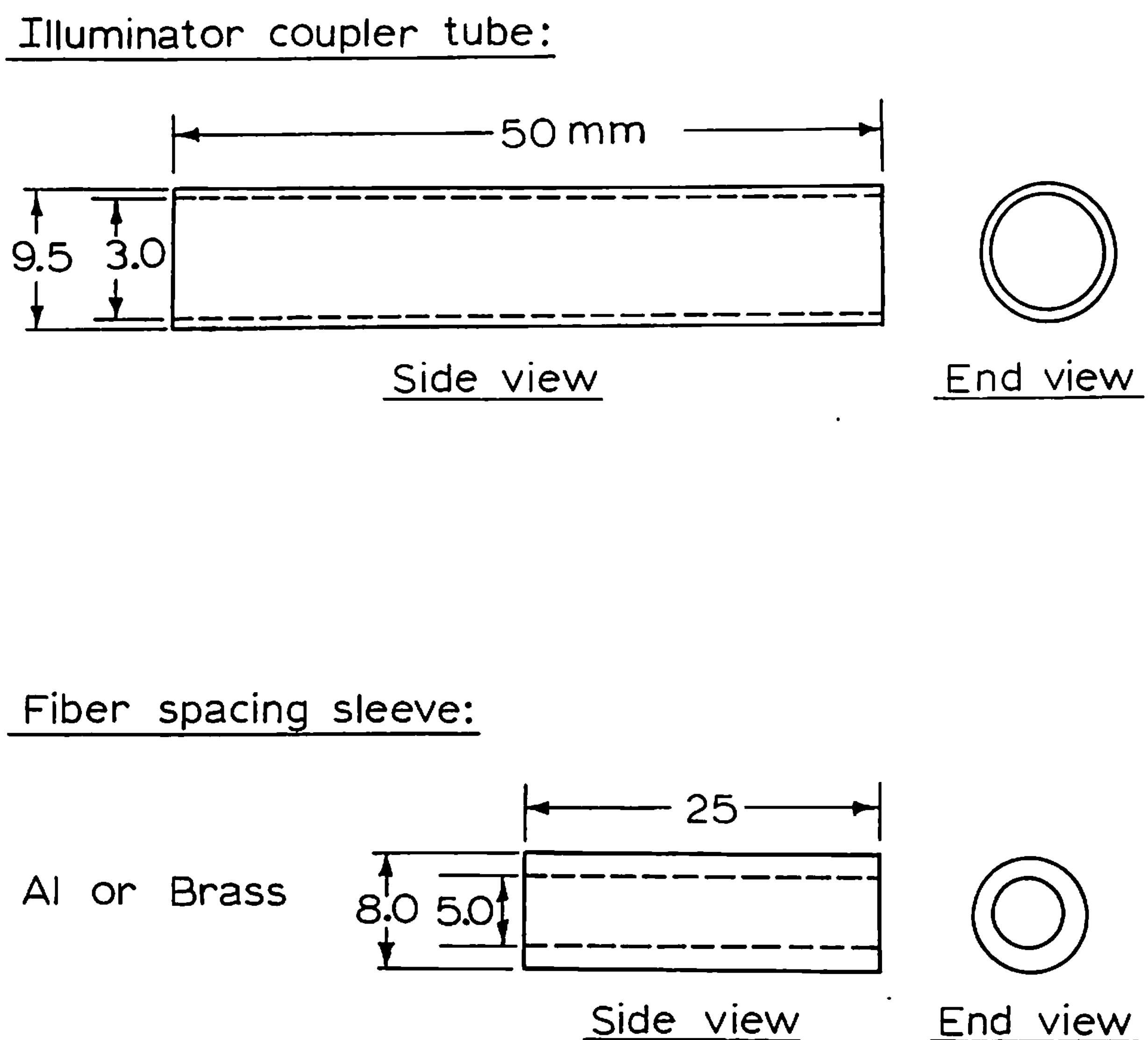


Figure 4.16: Illuminator coupler tube and spacing sleeve.

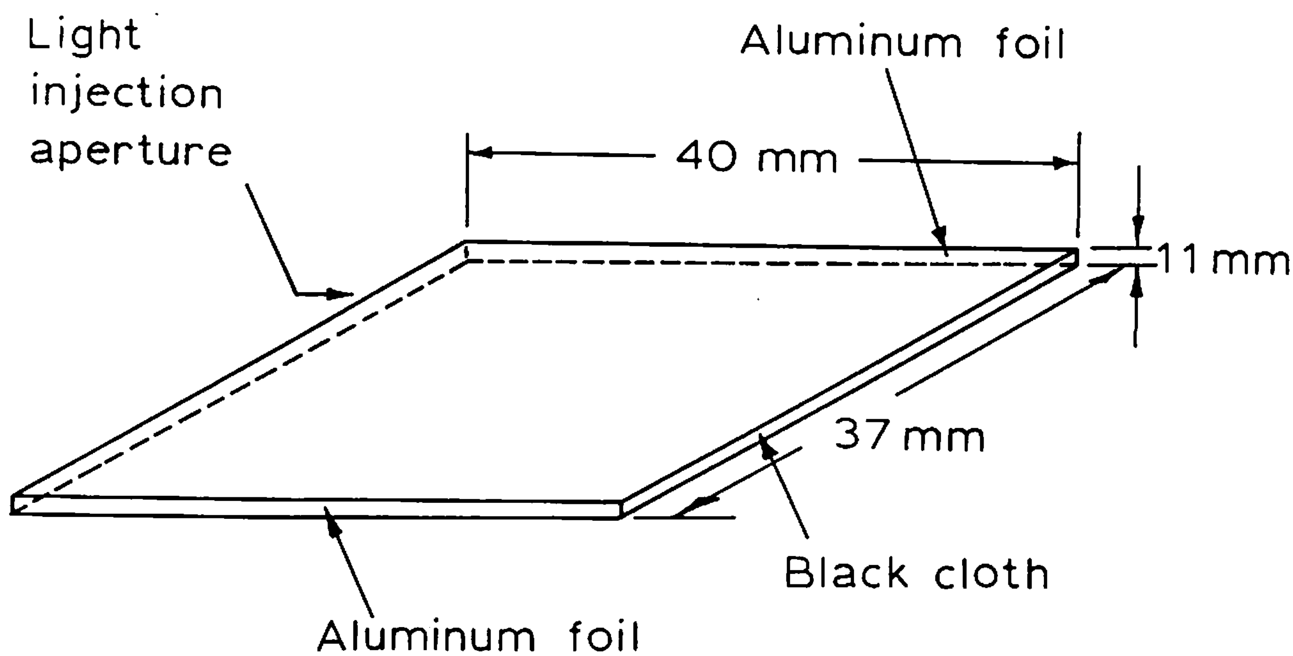


Figure 4.17: Glass transducer plate for TIR-TSA.

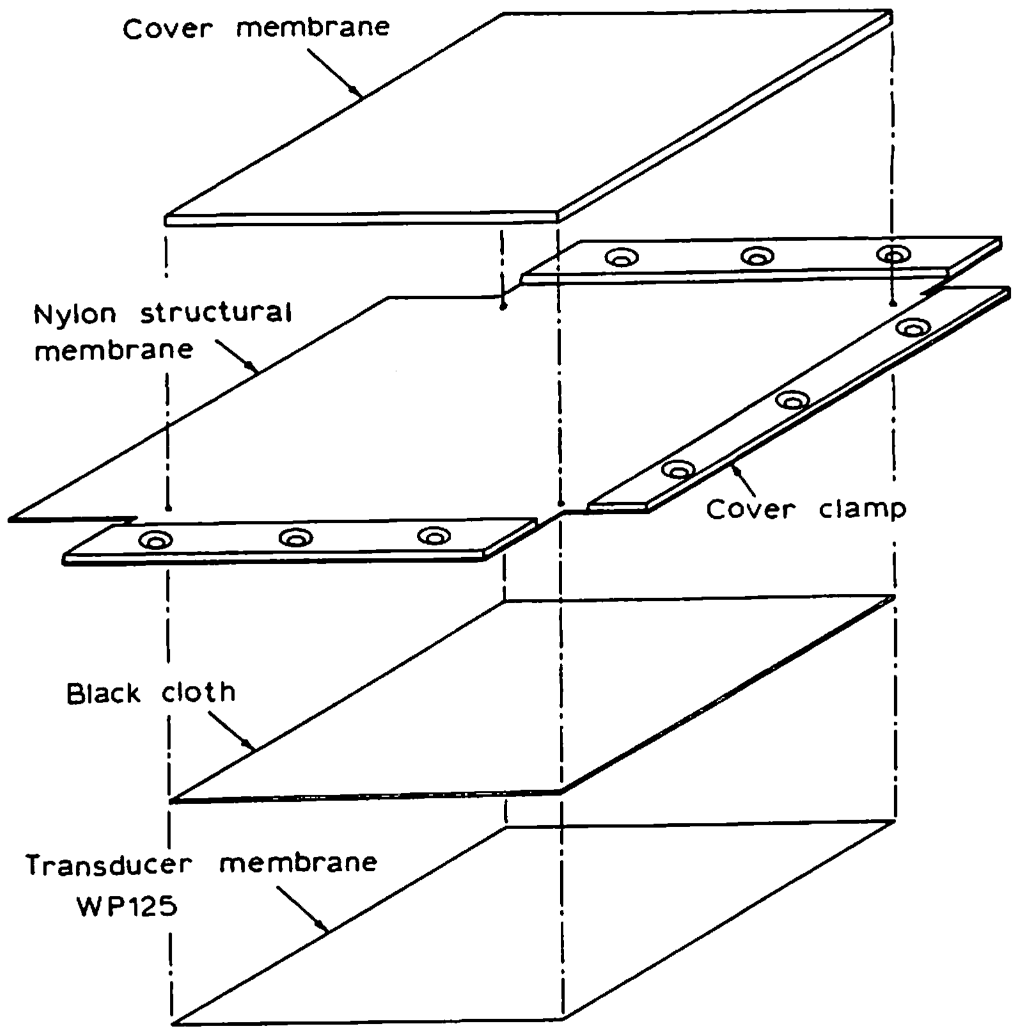


Figure 4.18: Tactile transducer and cover assembly.

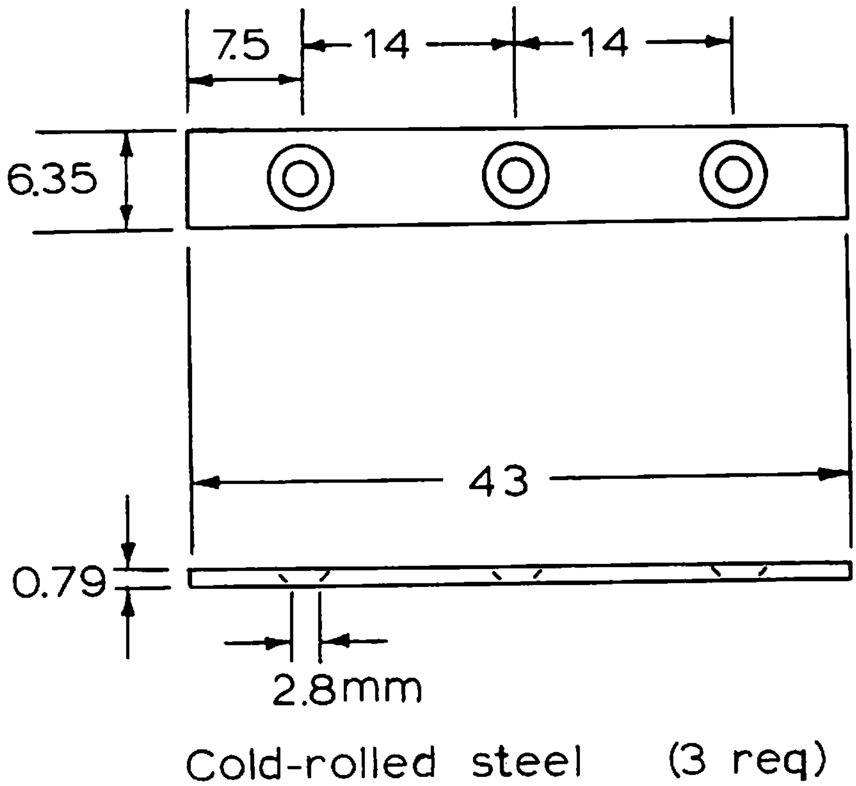


Figure 4.19: Cover clamps.

The final component of the sensor head assembly was the optical-fiber cable-cover clamp (Figure 4.20). The exact dimensions were tailored to accommodate the outcome of previous TSA assembly steps. It was used to clamp the coherent fiber bundle and the illuminator bundle together, and to firmly hold the polyester cloth sheaths in place. The cable clamp and illuminator terminator assembly were both affixed to the sensor head body with the same two screws, thus forming a rigid, interconnected unit.

4.3.2 Sensor Illuminator

The illuminator unit was the light source that "powered" the optical tactile sensor. No suitable commercial illuminators were found that were inexpensive and compact, thus requiring the design of the special unit shown in Figure 1.1 and 4.21. The principle focusing element was a reflector from a projection lamp. The old bulb was discarded and replaced with a 50 W unit (GTE Sylvania 1994 Halogen) which drew 3.5 A at 14 VDC. Cooling fins were used to dissipate the heat generated by the lamp. The thermal load striking the optical fibers was minimized by passing the beam through two 6.3 mm thick heat absorbing glass plates, and then passing the beam through a heat reflecting mirror. A 64 mm focal length lens focused the light to a circle approximately 5 mm in diameter which coincided with the optical-fiber illuminator cable leading to the tactile sensor head. Detailed drawings of various components for the illuminator unit are shown in Figures 4.22 through 4.29.

4.3.3 Optical Interface

The function of the optical interface was to convert the optical signal from each fiber in the display array into a digital value, and to provide a communication interface compatible with the protocol required by the preprocessor module.

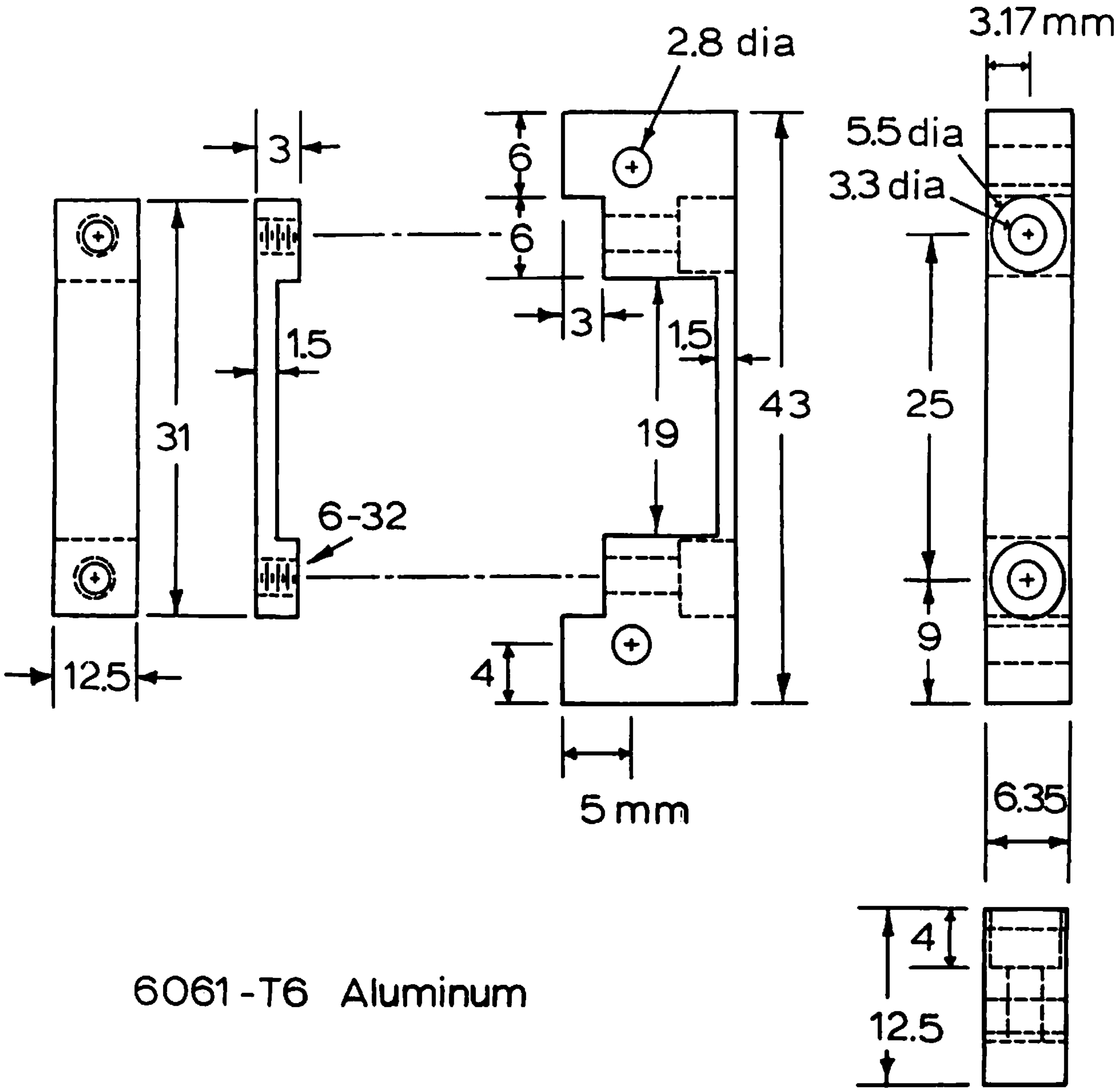


Figure 4.20: Optical-fiber cable-cover clamp.

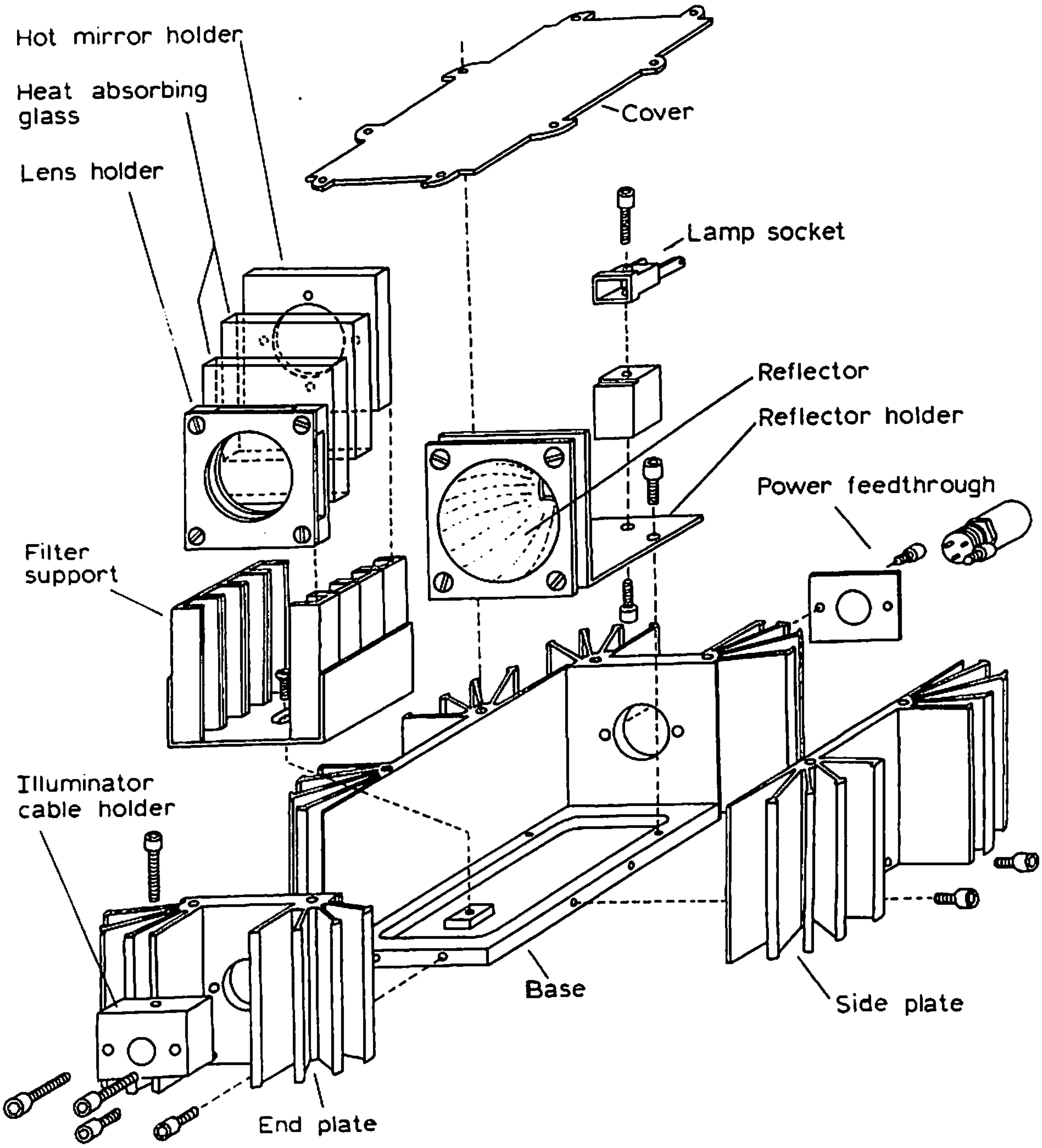


Figure 4.21: Exploded drawing of TSA illuminator.

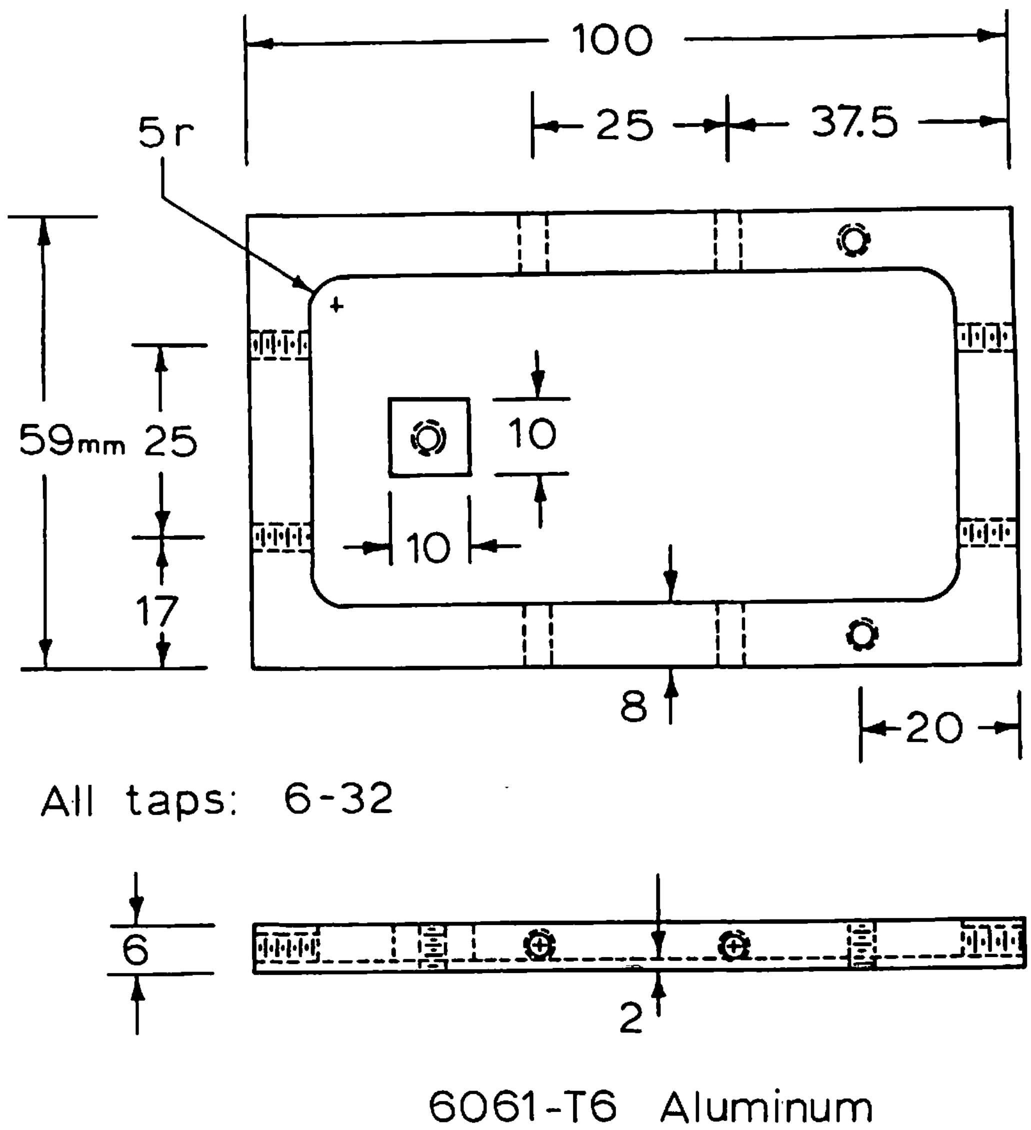


Figure 4.22: Illuminator base plate.

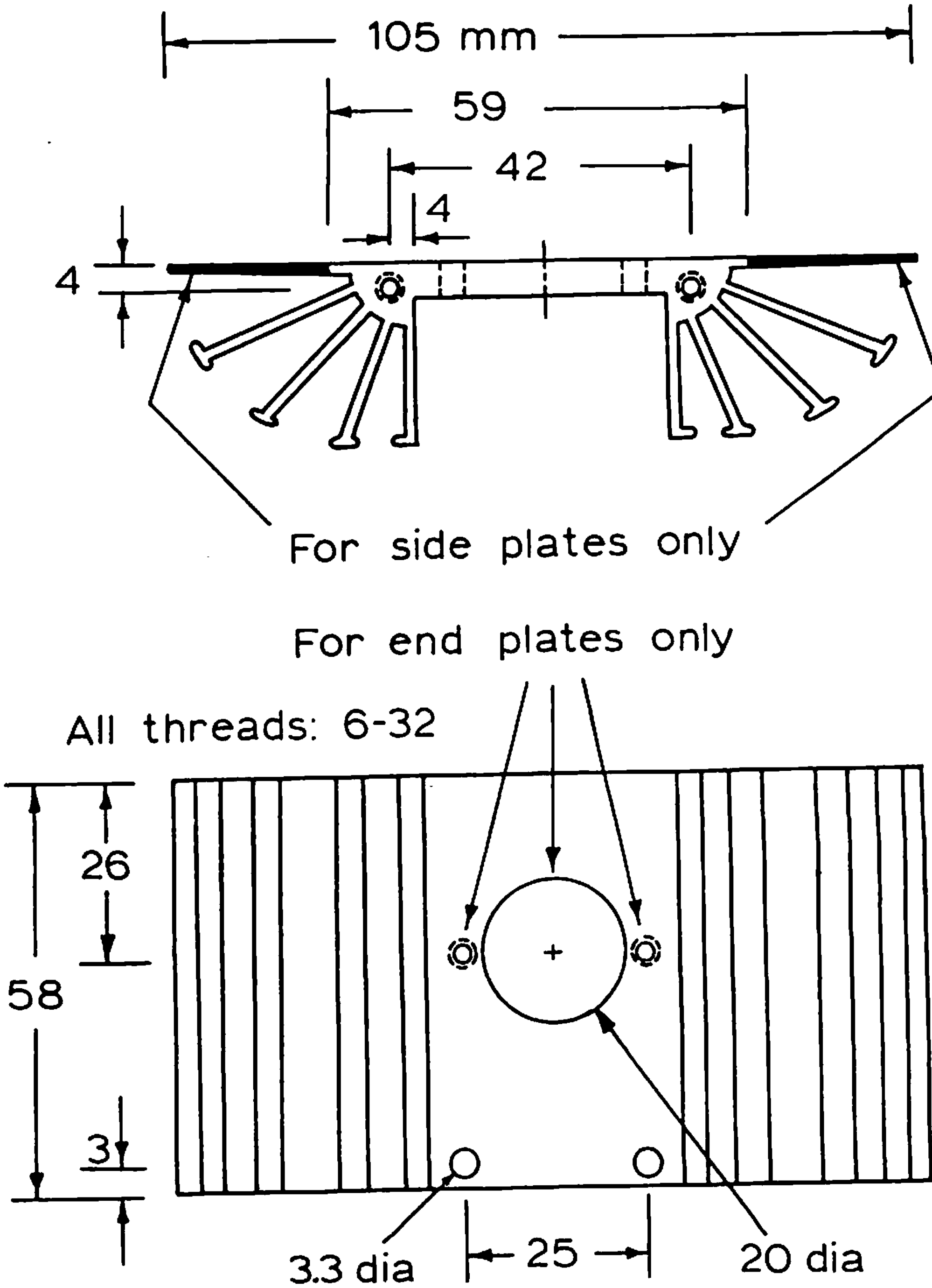


Figure 4.23: Illuminator side plates were fabricated from aluminum heat sinks (Wakefield 641A). Four were required.

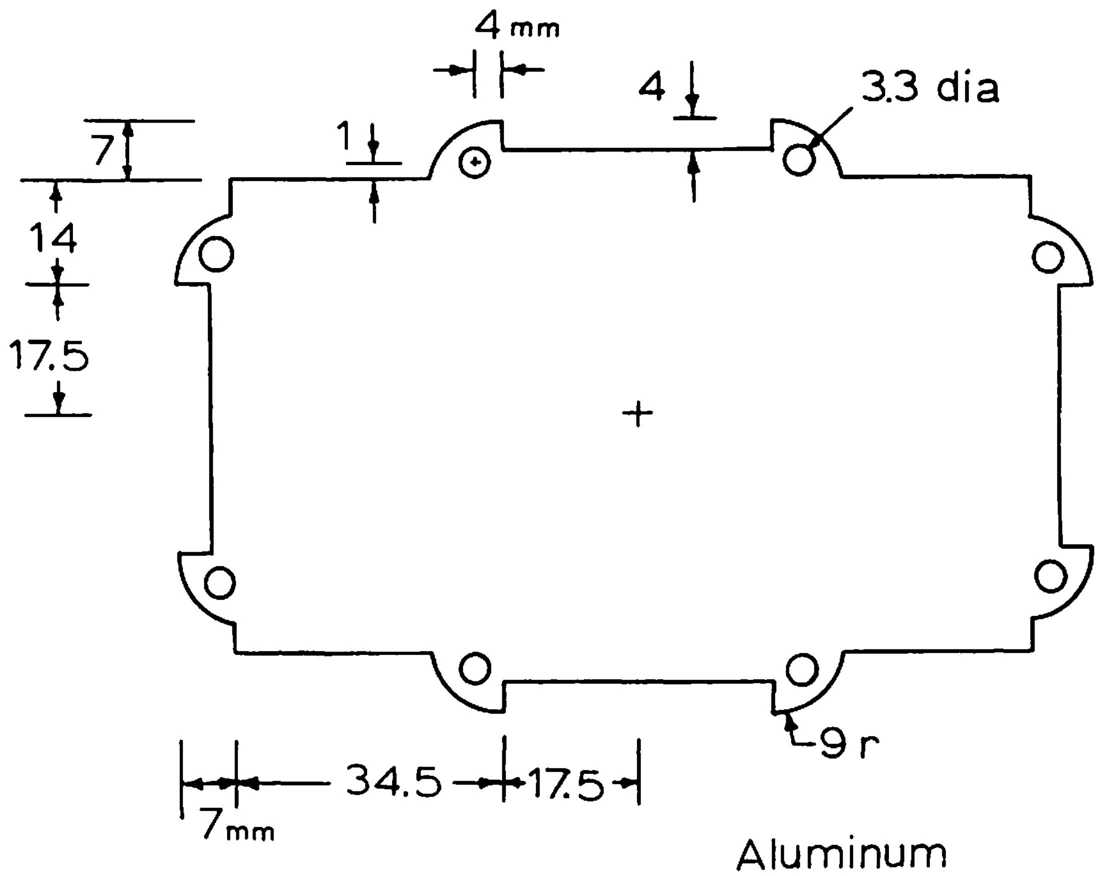
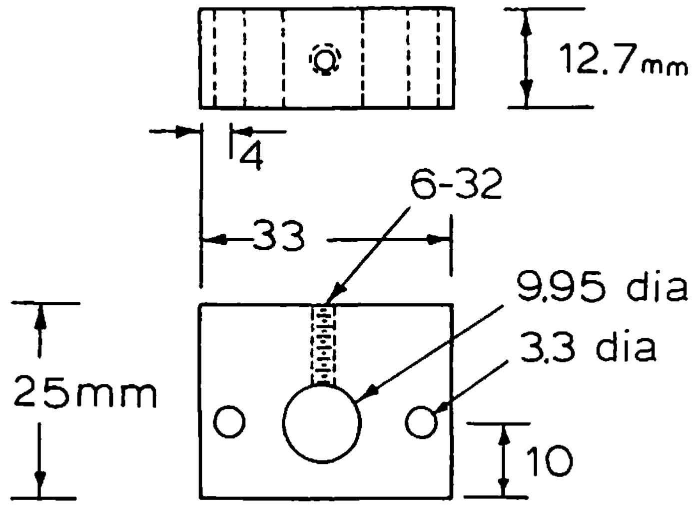
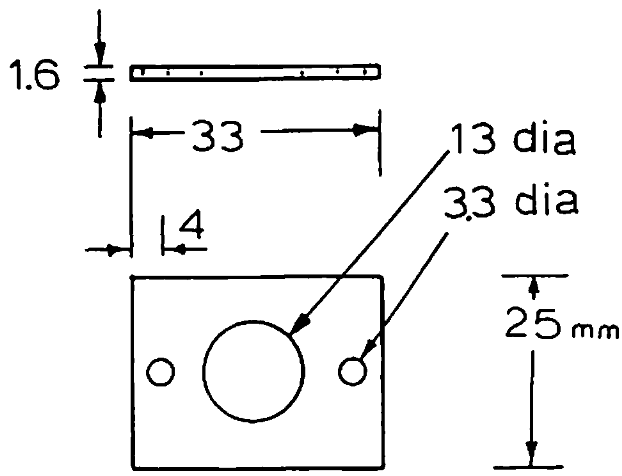


Figure 4.24: Illuminator cover plate.



ILLUMINATOR CABLE HOLDER



POWER FEEDTHROUGH PLATE

Figure 4.25: Illuminator cable holder (top) and power feedthrough plate (bottom).

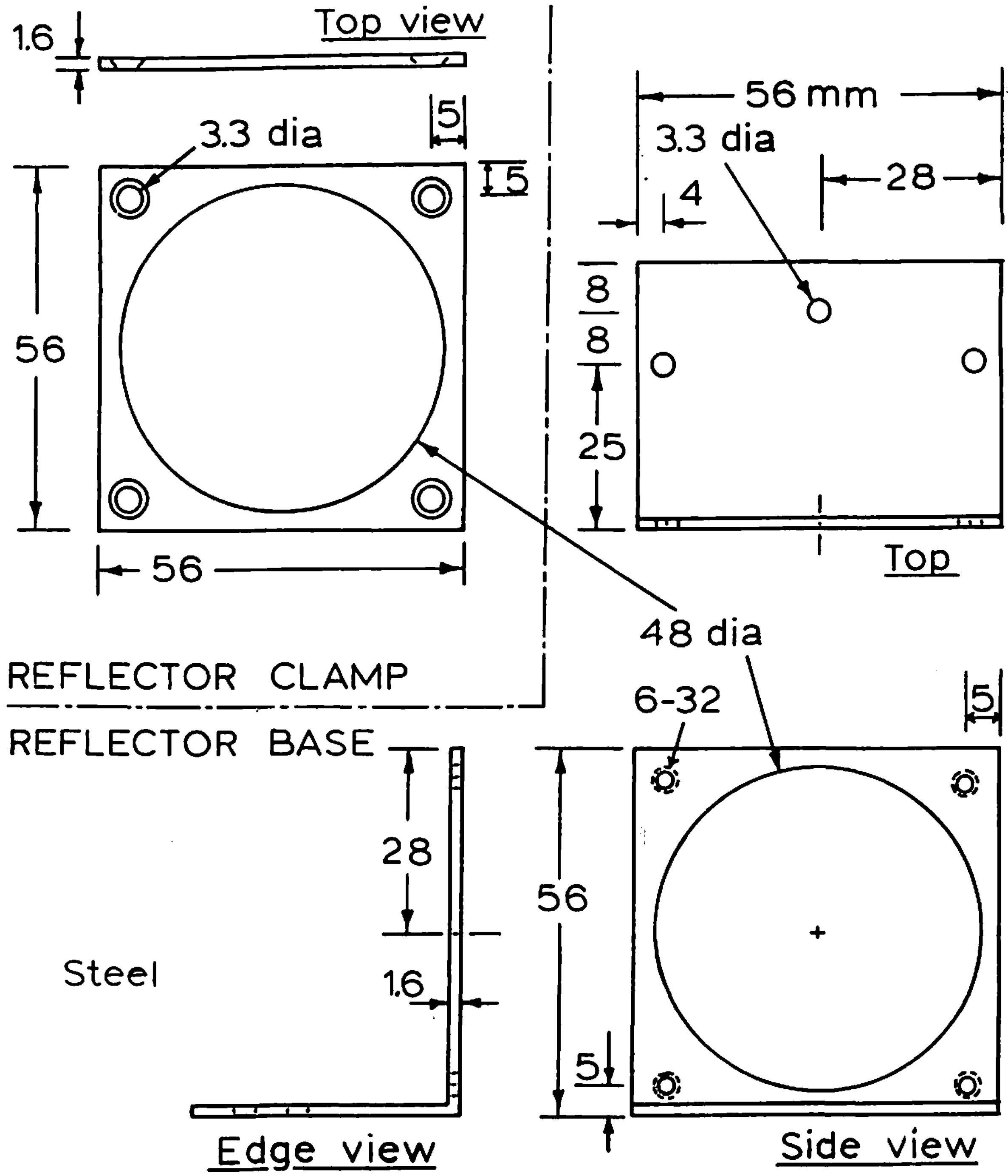


Figure 4.26: Reflector holder.

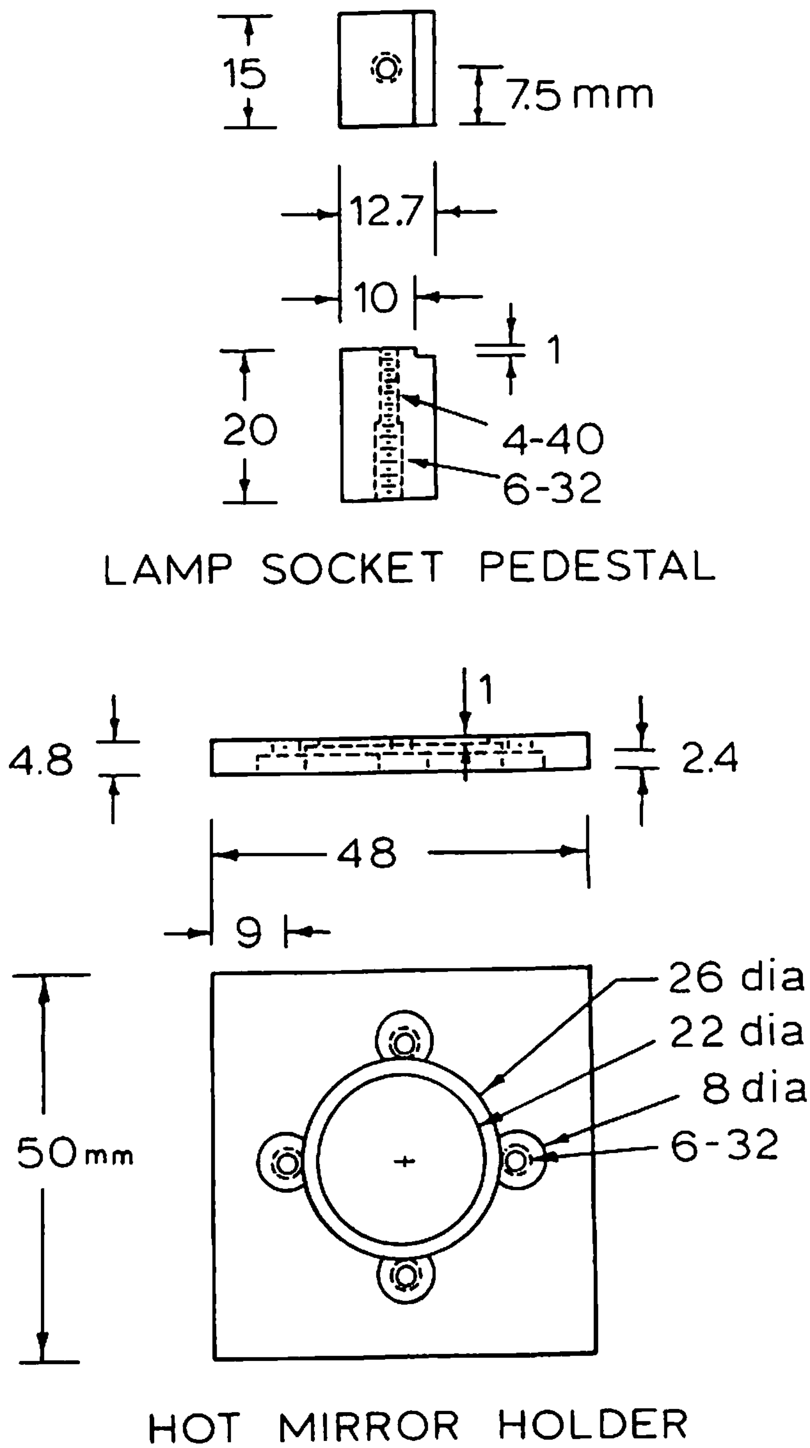


Figure 4.27: Lamp socket pedestal (top) and hot mirror holder (bottom).

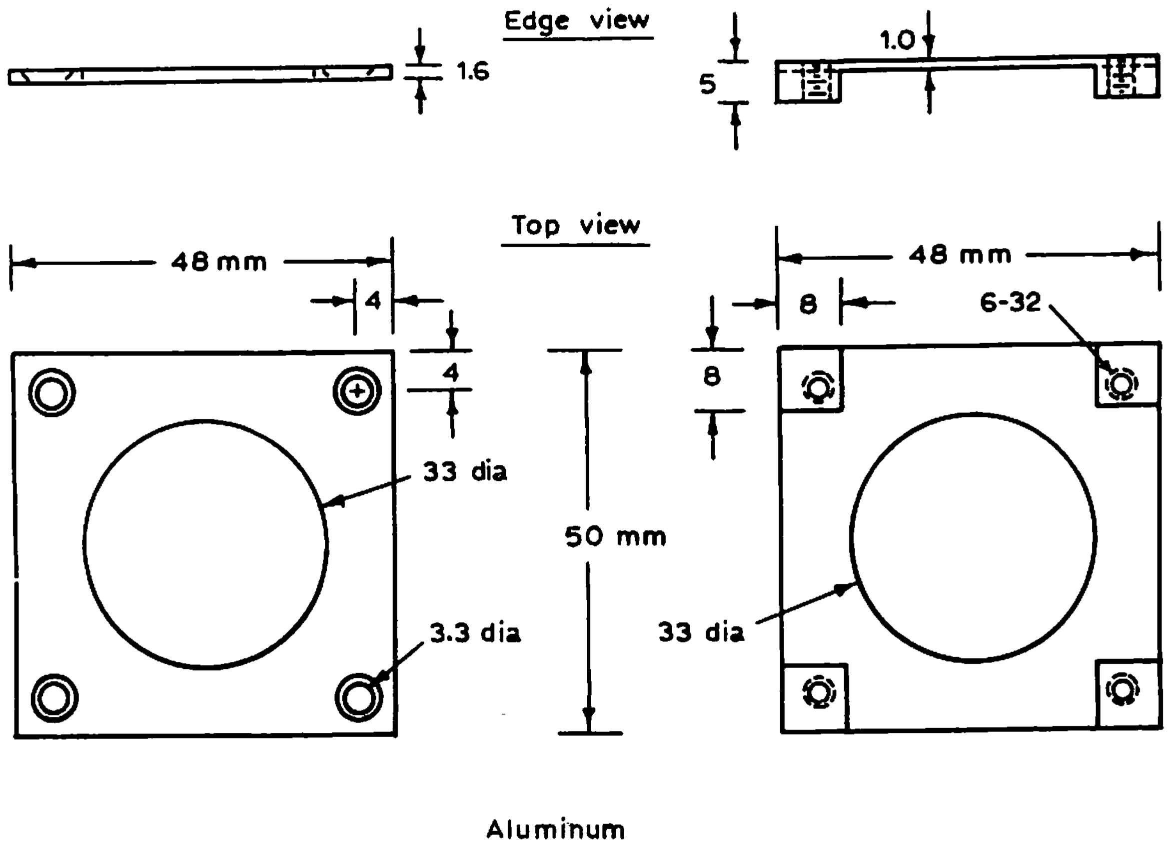


Figure 4.28: Lens holder.

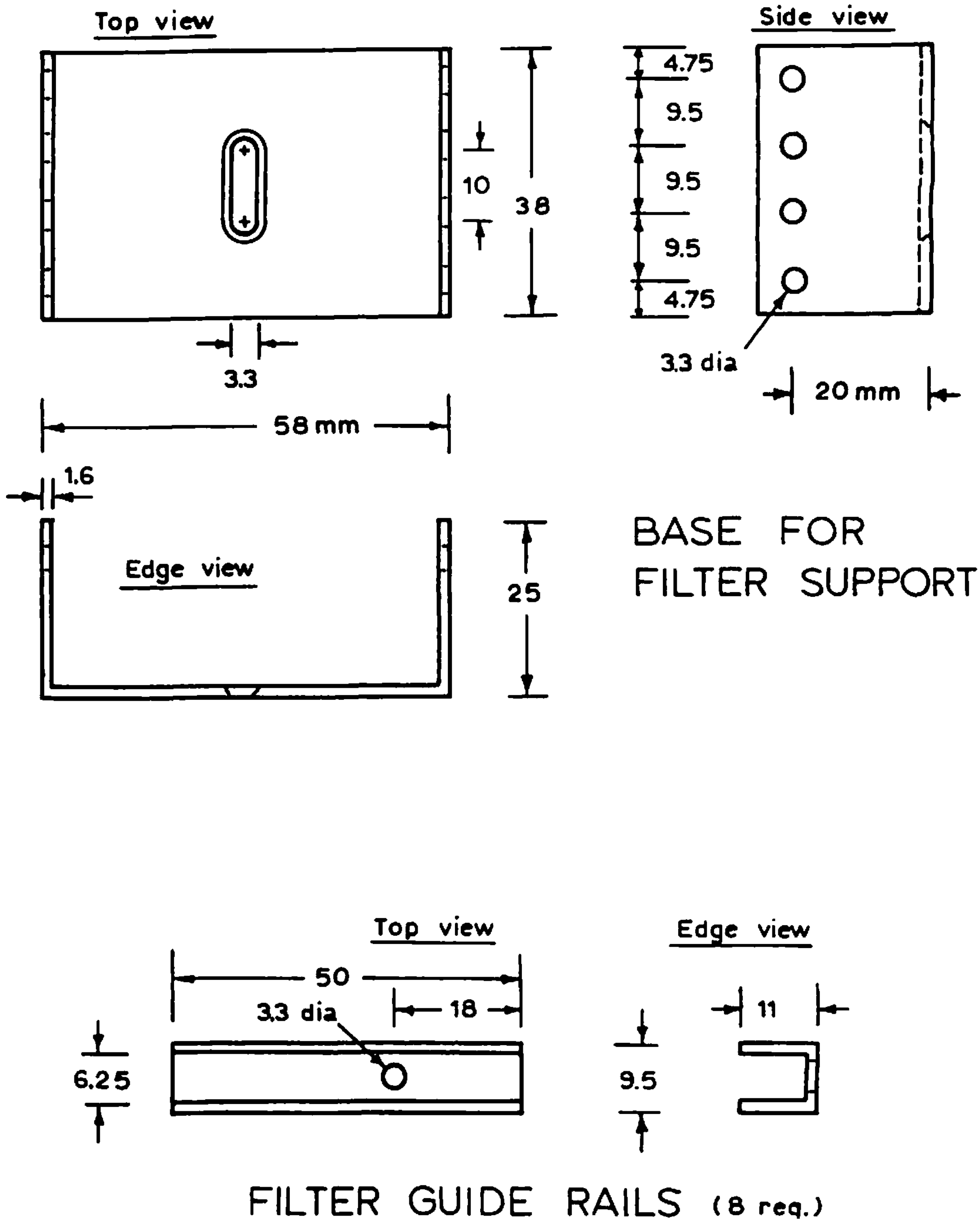


Figure 4.29: Filter support assembly.

The principal components of the optical interface were the optocoupler unit, camera controller board, and communication/ADC board. The optical interface is shown in block-diagram form in Figure 4.1, and in various stages of assembly in Figures 1.1, 4.30, and 4.31.

An exploded view of the optocoupler is shown in Figure 4.32, and the detailed mechanical drawing for the various components are shown in Figures 4.33 through 4.39. The optocoupler converted the light signal on each optical fiber in the display array into an analog electrical signal. This was done by imaging the imprint from a 32 x 32 optical fiber display array (Figure 4.40) onto a 32 x 32 pixel CCD solid-state camera chip (EG&G Reticon RA32x32A: see Figure 4.41) through a 12.5 mm focal length lens (Figure 4.30).

The positions of the display array, lens, and CCD array were adjusted such that a one-to-one mapping existed between each optical fiber in the display array and each pixel in the CCD camera. The Moire Effect was extensively relied upon to obtain the correct angular and positional alignment between a fully illuminated imprint display array and the checker-board pattern of detectors on the CCD camera chip.

The sampled-and-held output from the CCD camera controller board (EG&G Reticon RC100B/110-2: see Figure 4.30) was fed to an 8-bit high-speed analog-to-digital converter (Analog Devices, model MAH-1001-3) which had a conversion time of 0.8 microseconds. The ADC was located on the communication/ADC board, photographs of which are shown in Figures 4.30 and 4.31. The physical layout, electrical schematic, signal timing diagram, connector schematic, and connector pin list are shown in Figures 4.42, 4.43, 4.44, 4.6, and Table 4.1, respectively.

Data acquisition by the DCT-11EM was performed asynchronously by an interrupt service routine. Referring to Figure 4.44, acquisition of an imprint frame (1024 taxels) was initiated by driving ANF (acquire new frame, which is the same as the LED on the DCT-11EM) low by writing 11 (octal) to memory location 177444. ANF and FS (frame start) are then combined and used to gate the

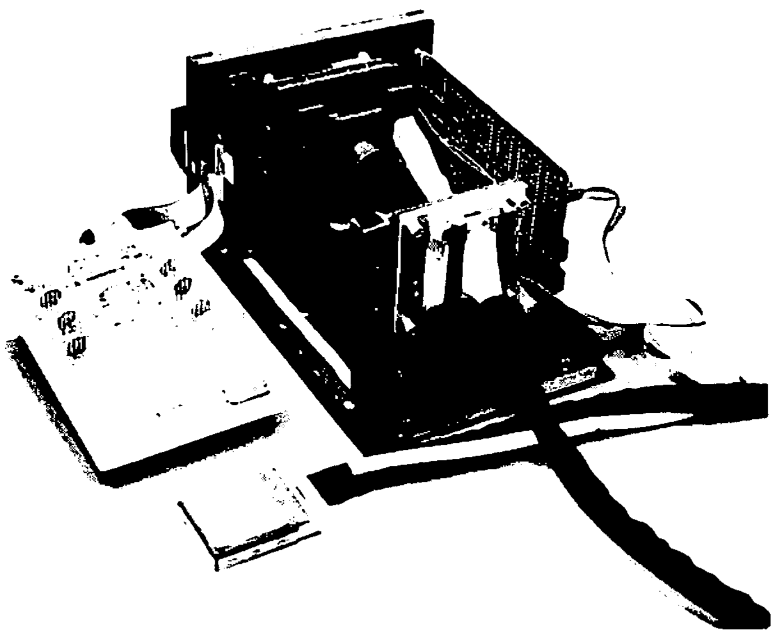


Figure 4.30: Components of the optical interface: TSA head (foreground), communication/ADC board (left), optocoupler (center), and camera controller board (right).

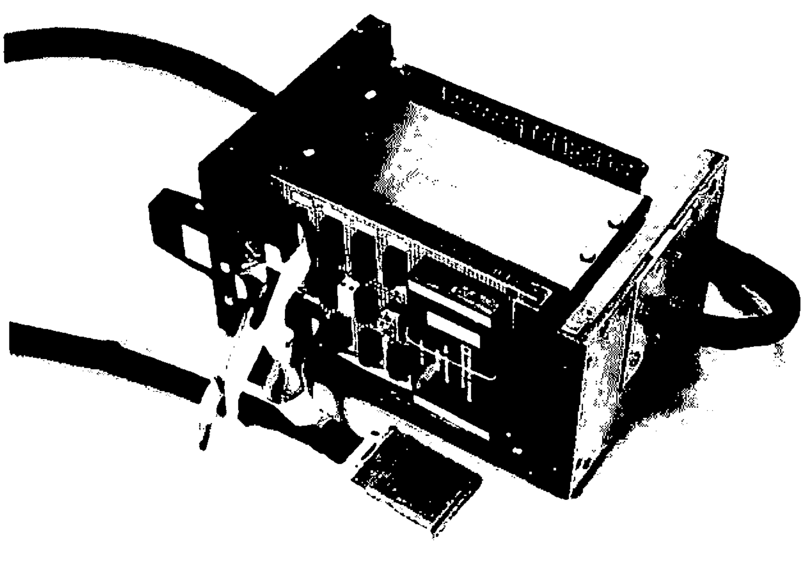


Figure 4.31: Assembled optical interface shown with the case covers removed.

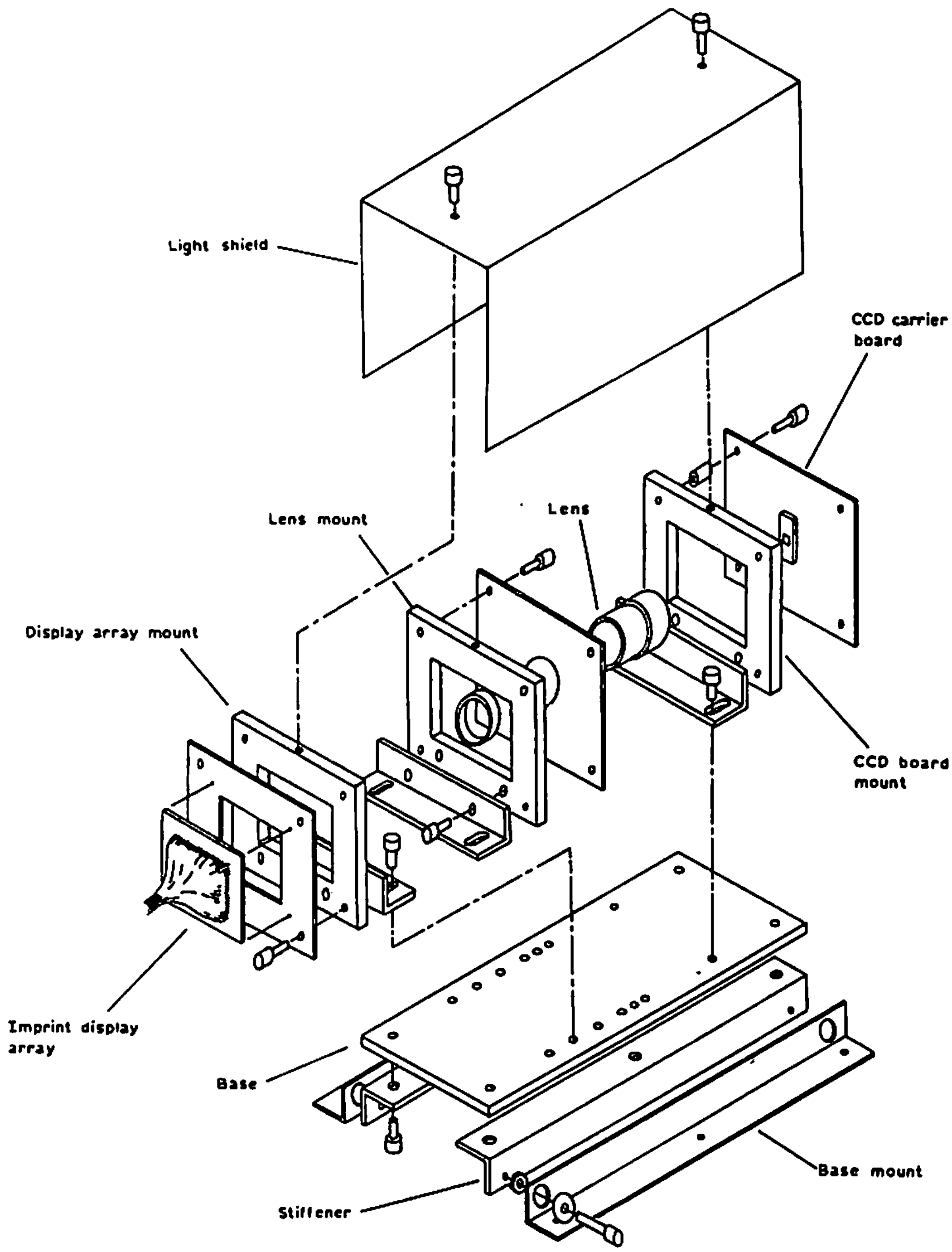


Figure 4.32: Exploded drawing of optocoupler unit.

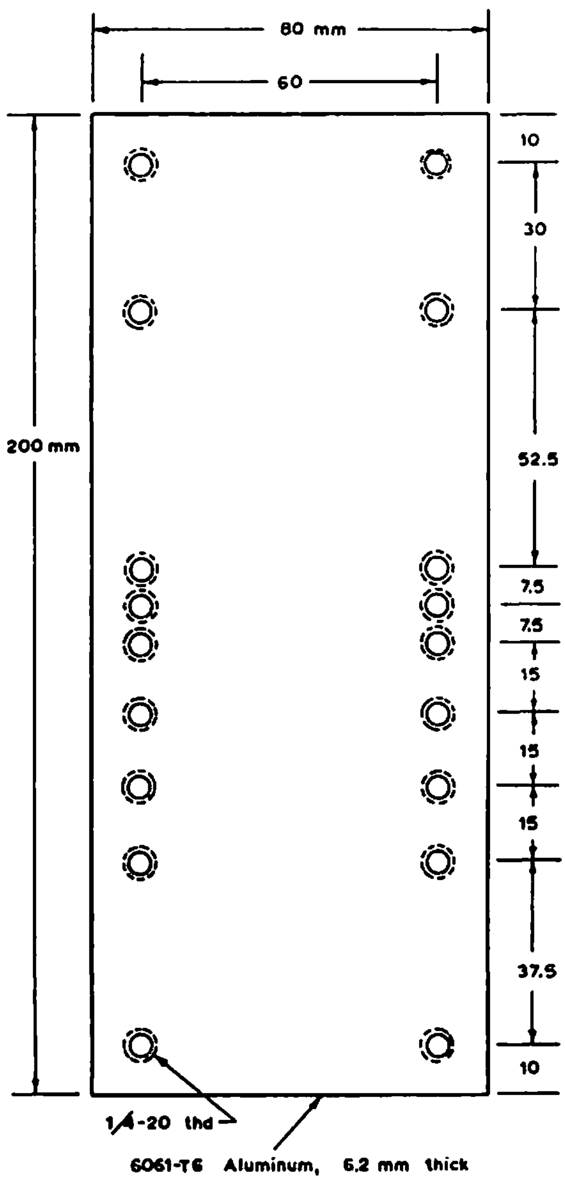
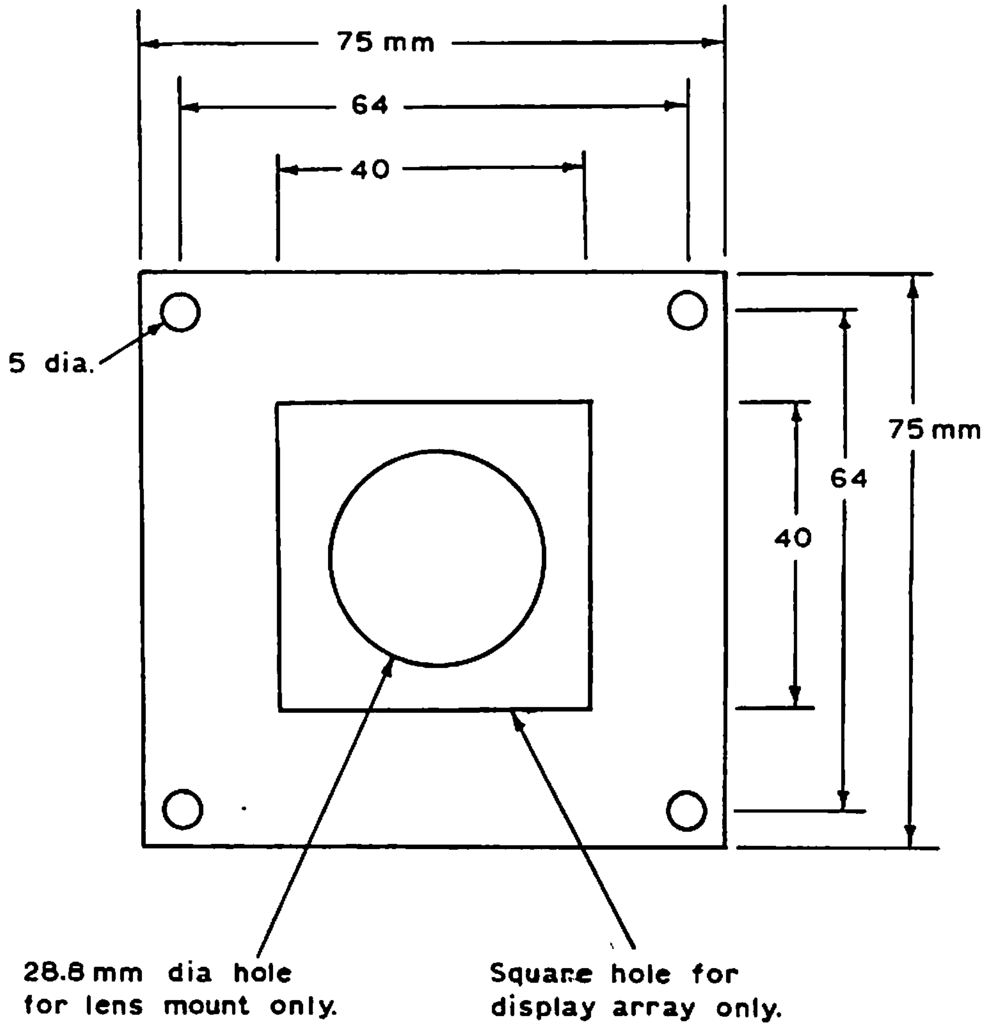
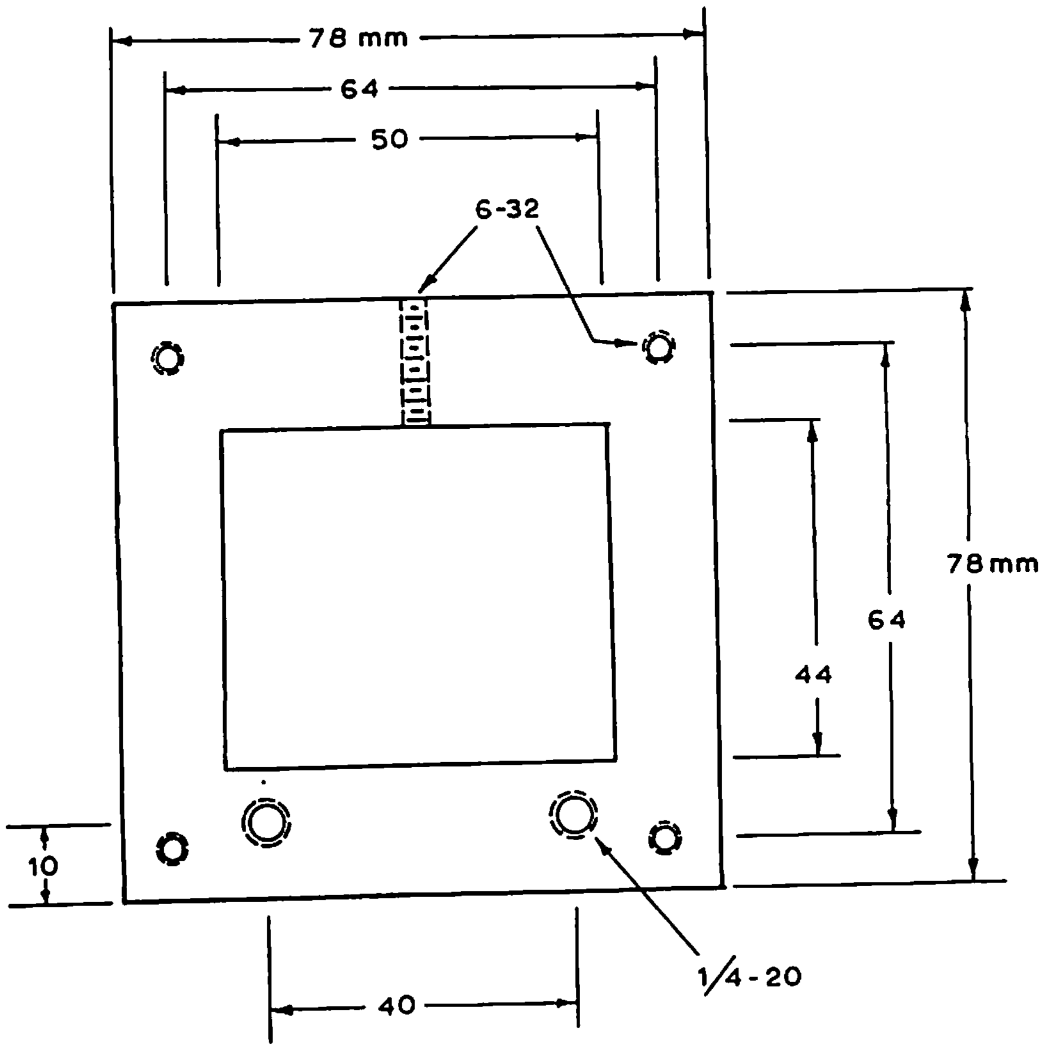


Figure 4.33: Base plate for optocoupler.



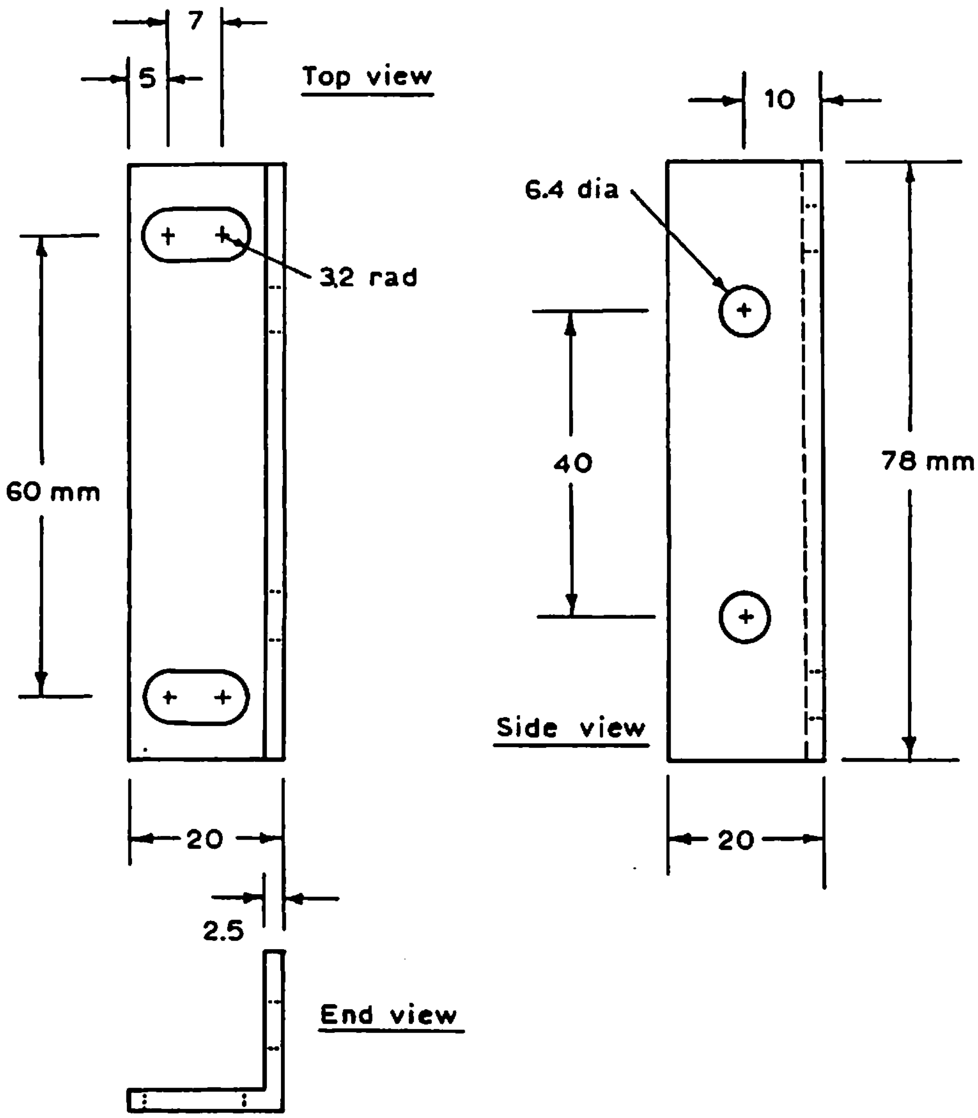
Material: 6061-T6 Aluminum,
3.2 mm thick for array carrier,
1.1 mm thick for lens carrier.

Figure 4.34: Carriers for display array and lens within the optocoupler.



Material: 6061-T6 Aluminum, 6.3 mm thick.
3 req.

Figure 4.35: Vertical mounts for CCD-chip carrier board, lens, and display array.



Material: Aluminum
Quantity: 3

Figure 4.36: Mount brackets.

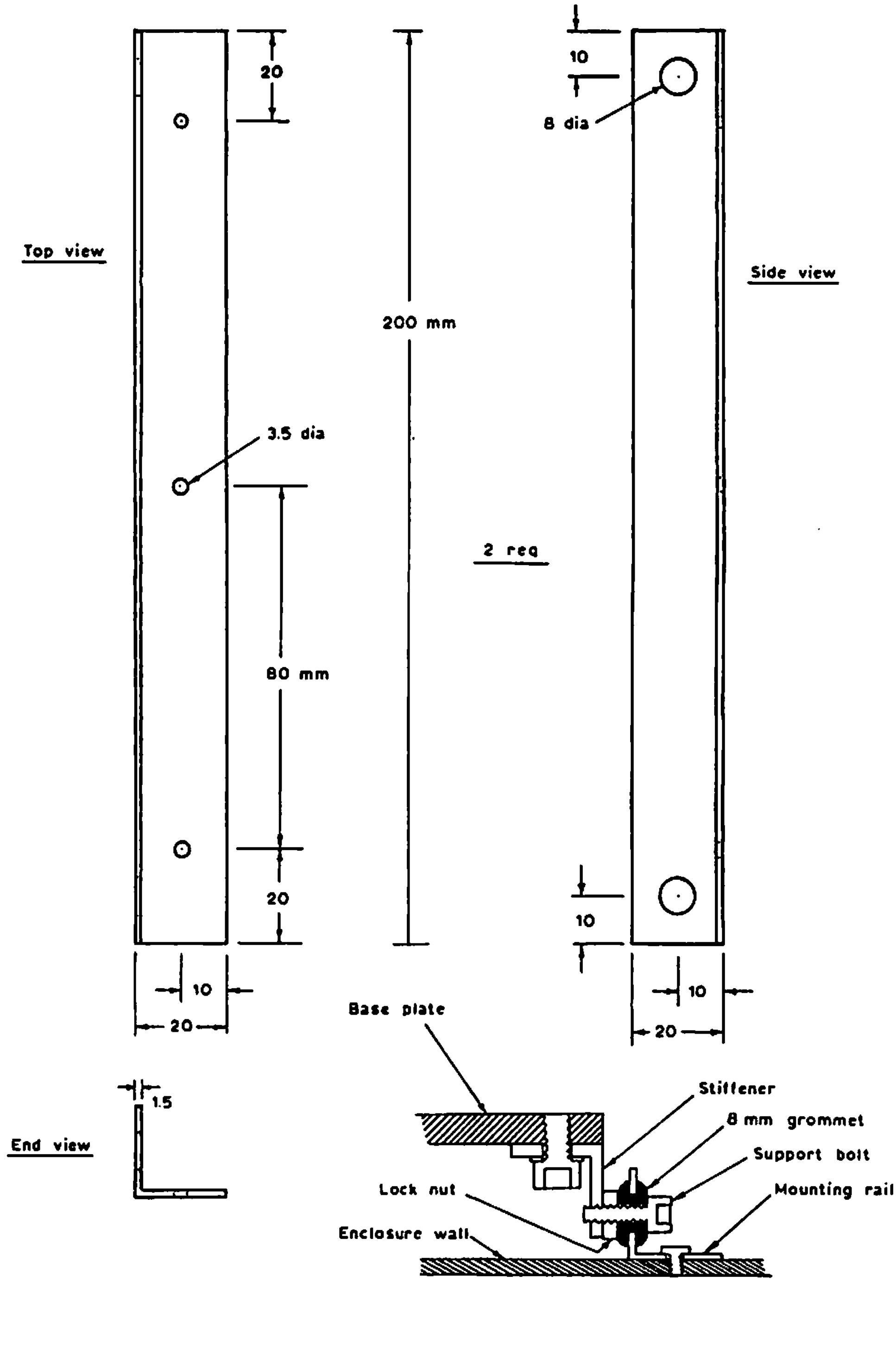


Figure 4.37: Mounting rails for optocoupler base, and details of the shock-absorbing mount system.

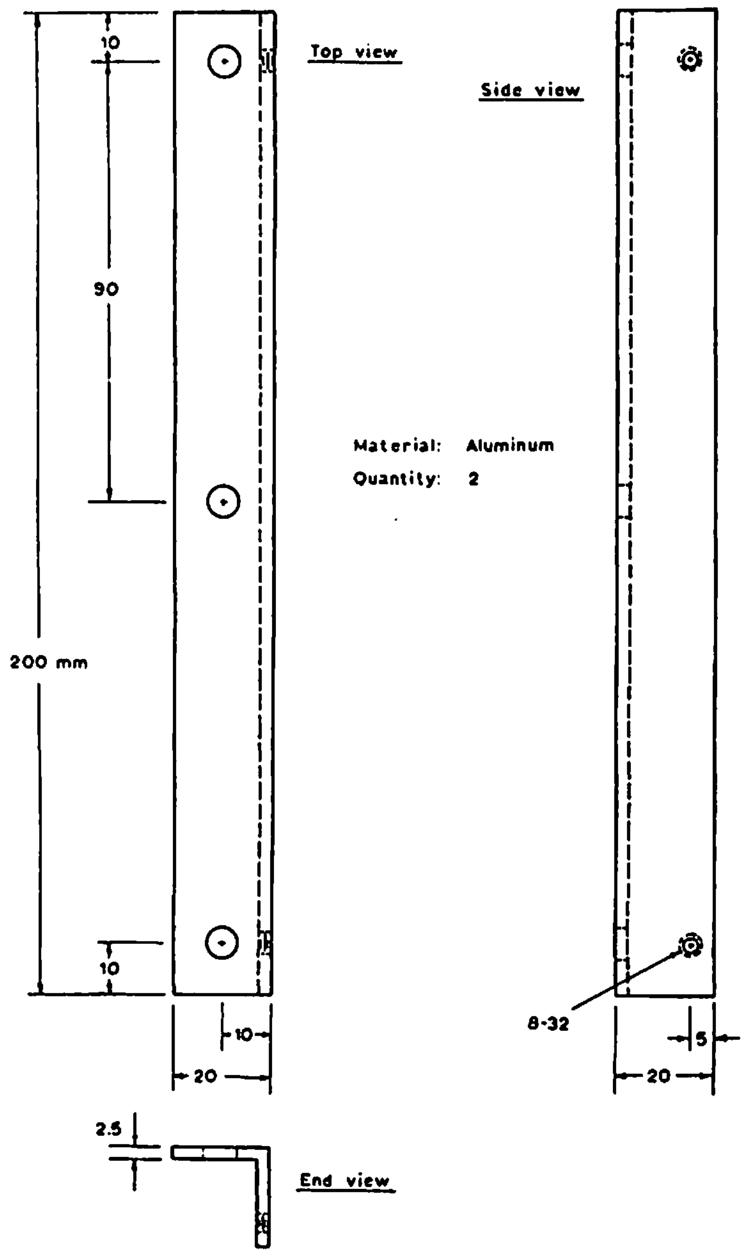


Figure 4.38: Base stiffener rails.

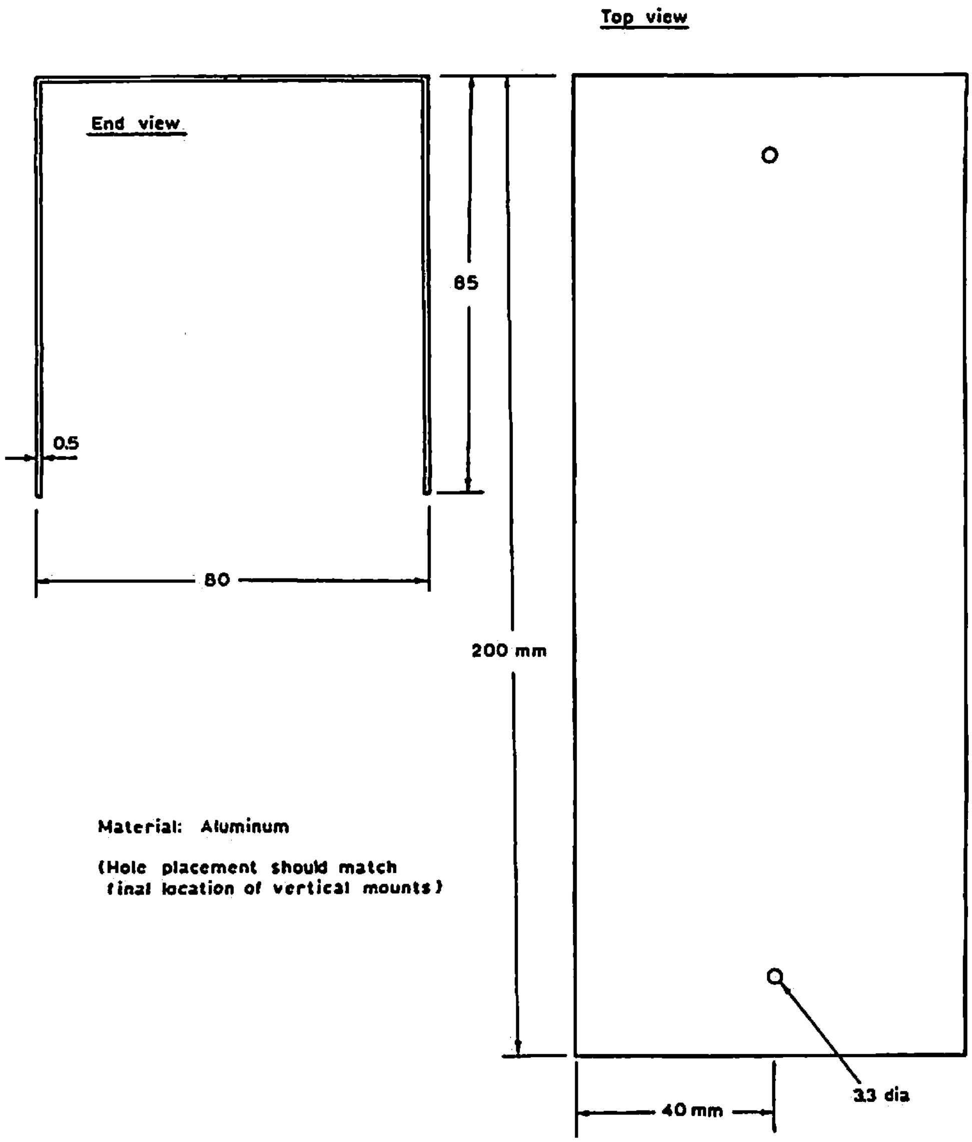


Figure 4.39: Light shield for optocoupler.

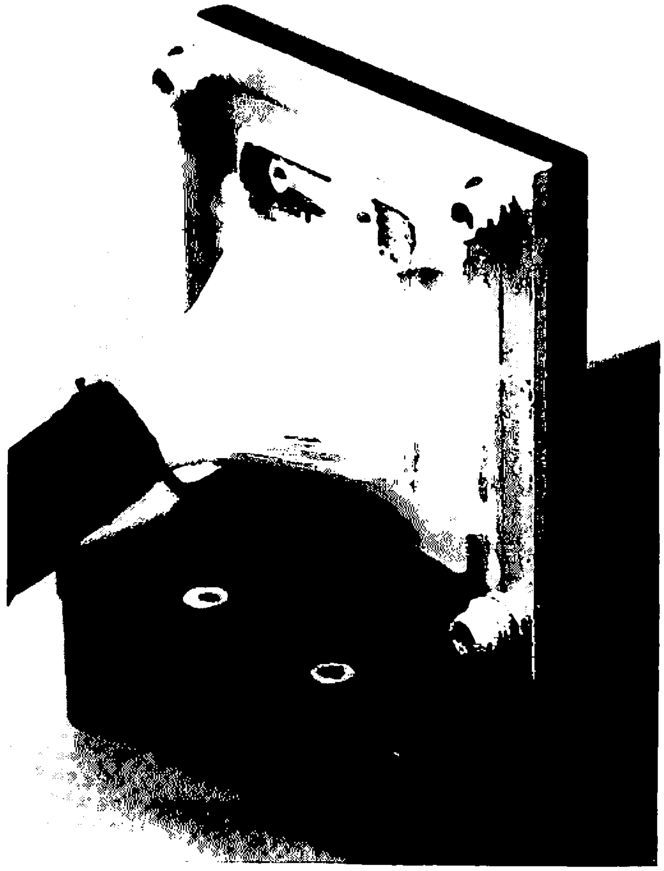


Figure 4.40: Rear view of the imprint display array when mounted on the optocoupler unit. The array contained 32 x 32 optical fibers and measured 31 x 31 mm.

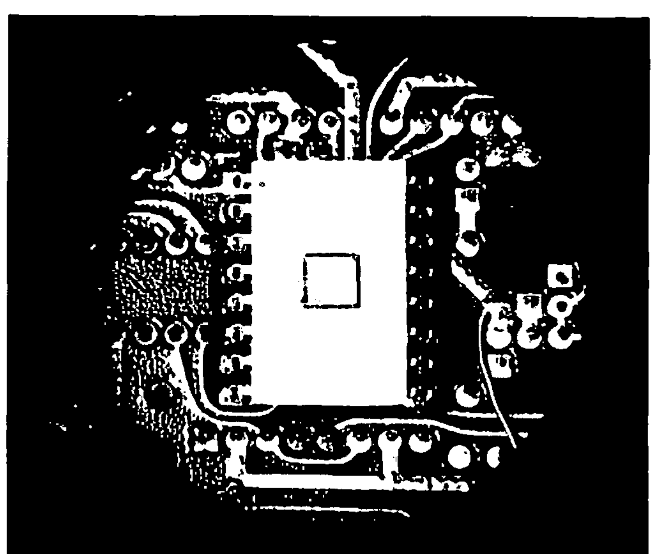


Figure 4.41: 32 x 32 pixel CCD camera chip mounted on the optocoupler. The active area measured 3.15 x 3.15 mm.

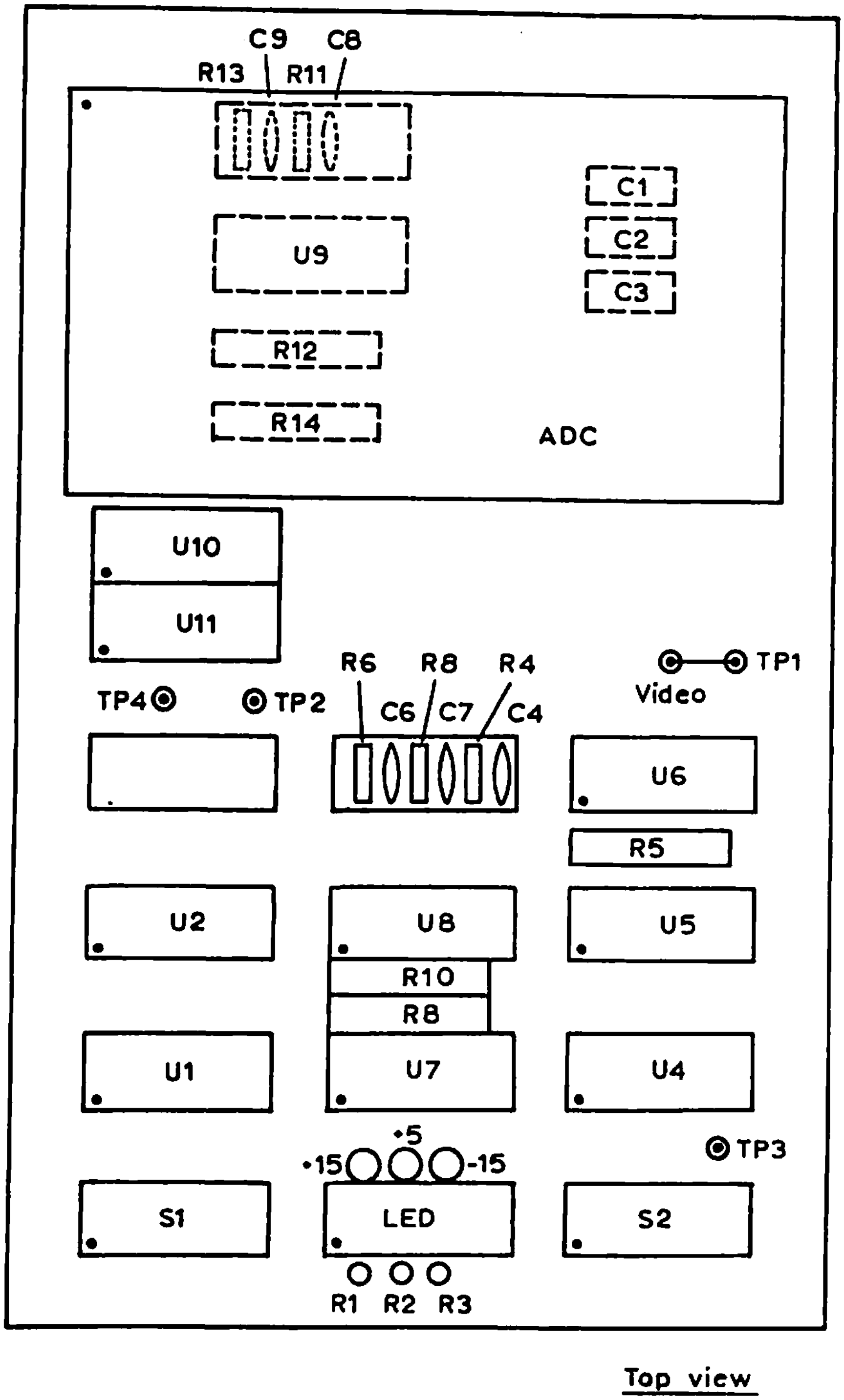


Figure 4.42: Layout of the communication/ADC board.

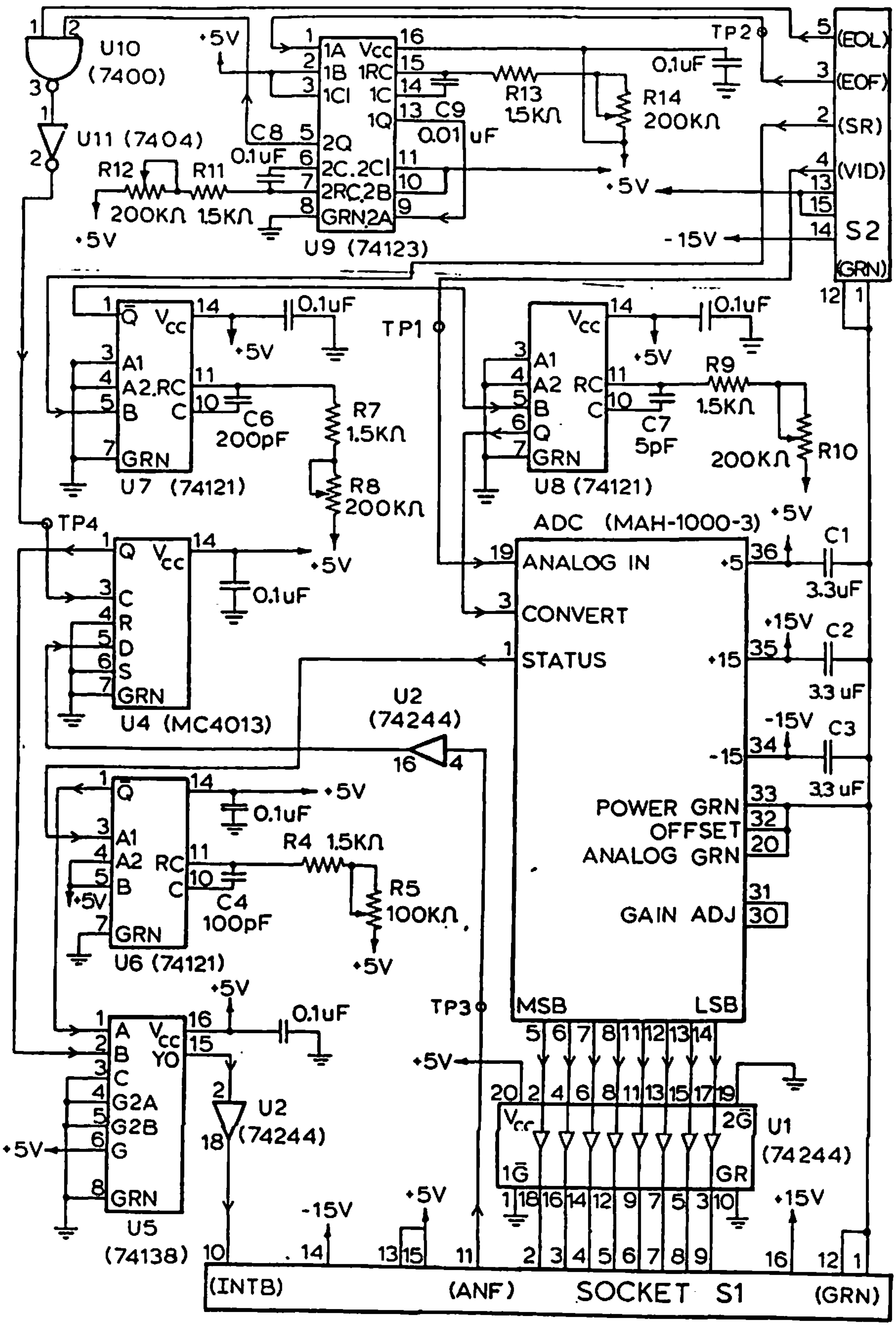


Figure 4.43: Schematic of communication/ADC board.

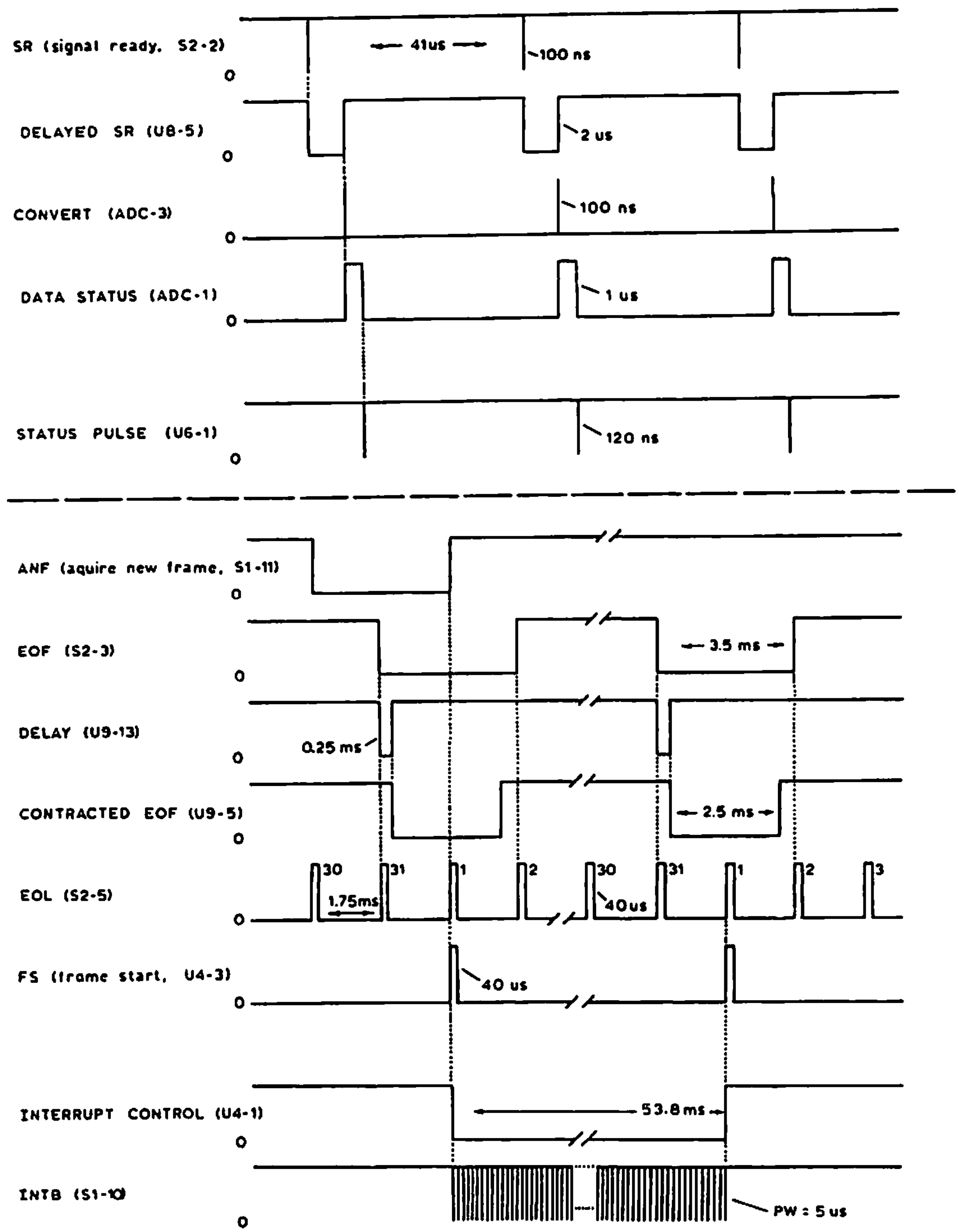


Figure 4.44: Signal timing diagram concerning the camera controller, communication/ADC, and DCT-11EM units within the TSS.

DATA STATUS PULSE from the ADC to the INTB (interrupt) line of the DCT-11EM.

Each interrupt signal activated an interrupt service routine which checked to ensure that the data was valid. This was necessary because the camera controller output consisted of 32 lines of 42 taxels. However, the first 2 and the last 8 taxels were part of the blanking period. The interrupt routine maintained a taxel counter and only stored valid taxels (e.g., numbers 3 through 34) from each line. The length of the frame period was set to 53.4 milliseconds. This value was established as a compromise between the need for short sampling periods necessary to achieve real time performance, and the need for long sampling periods to compensate for the low level of the optical signal delivered to the imprint display from the TIR-TSA head.

Chapter 5

Tactile Data Acquisition and Processing

5.1 Introduction

The preprocessor module within the TSS had an inherent ability to do more than control the sensor, condition the signal, maintain calibration, and output the raw data. Unused processing power was put to effective use by preprocessing tactile data so as to reduce the computational burden upon the host computer. Listed below are tactile data processing algorithms specific to inspection and automatic assembly tasks that were implemented on the preprocessor in addition to the data acquisition functions mentioned above:

1. Calculation of the total applied force;
2. Calculation of the first moments about the x and y axes (these values correspond to the torque exerted on the sensor);
3. Detection of change in the taxel sum or first moments with respect to a reference imprint;
4. Thresholding operations on taxels, taxel sum, change in the taxel sum, first moments, and change in the first moments.

These algorithms are sufficient to allow decisions to be made in simpler robotic tasks such as grasping, slip detection, force feedback during assembly operations, and collision detection. Left for future implementation were algorithms which extract tactile features such as surfaces, edges, protrusions, texture, and holes to aid in object identification and pose estimation during visual occlusion.

The TSS was designed to be configured as a general I/O device with respect to the host computer. Commands to the TSS were sent via an RS232C communication line, and data and responses were returned to the host over the same line. The control commands, their function, and the speed of execution are presented in Section 5.3.

The software for tactile data acquisition and processing resided in approximately 3.2 KByte of RAM on the preprocessor. The programs were written in PDP-11 assembly language and were downloaded into RAM from the host via a serial line (9600 baud default). Another 4 KByte of memory was used for storage of the current tactile imprint, two calibration imprints, and a reference imprint. A complete listing of the program is provided in Appendix A.

The following sections discuss start-up, calibration, and use of the TSS while in the CC mode (see Section 4.2). It is the responsibility of the user to develop a program for the host computer that is able to communicate with the TSS in the HC mode. Note that all numbers are octal unless followed by a decimal point, user responses are **bold-faced**, and comments will appear within parentheses.

5.2 Start-Up procedure

The following sequence of operations are to be followed to bring the TSS into the running state, i.e., ready to accept commands from the console:

1. Turn MAIN "ON".

2. Turn T-11 "ON".

3. Turn ILLUM "ON".

4. Set baud rate on the DCT-11EM (default = 9600):
   Use Keypad (see Figure 4.2):

   FNC
   3
   EXA
   ADV/BAC (Set desired baud rate)

5. Start the console monitor:
   Use Keypad:

   FNC
   2
   EXA

The following message should appear on the console screen, indicating that the system is now under console keyboard control.

TEM CONSOLE MONITOR V1.0
TEM>

6. Down-load the program from the host:

TEM> **VTON(CR)**
TEM> **HOST(CR)**
TEM> **LOAD(CR)(CR)**

Log onto host and access program directory:

Username: **BEGEJ(CR)**
Password: **XXXX(CR)**
$ set default [.taction](CR)

Direct the host to type the program on the console screen (the DCT-11EM listens and automatically loads all instructions between the ".START" and ".END" directives):

$ type tacsen8.pdp(CR)
TEM> .START
TEM> .
TEM> .
TEM> . (BODY OF PROGRAM)
TEM> .
TEM> .
TEM> .END
TEM>

Approximately 2 minutes are required to down-load the program at 9600 baud.

The default mode of the DCT-11EM is INSTRU, or WORD. There are situations (see Section 5.3) in which data transfer to the host is much faster in the BYTE mode, e.g., transmission of taxel values. The mode of the DCT-11EM may be changed using the following command sequences:

To enter BYTE mode:

```
> control-Y(CR)
TEM> BYTE(CR)
TEM> GO(CR)
>
```

To enter WORD mode:

```
> control-Y(CR)
TEM> WORD(CR)
TEM> GO(CR)
>
```

To enter INSTRU mode:

```
> control-Y(CR)
TEM> INSTRU(CR)
TEM> GO(CR)
>
```

7. Start the program:

```
TEM> GO(CR)
>
```

At this point the program is ready to start accepting 6-digit octal command codes.

5.3 TSS Command Codes

The TSS is under the control of the a console or host computer, and will perform functions in response to 6-digit command codes sent over the serial console line. Listed below are the command codes, the functions they implement, and the speed of implementation under a variety of conditions. As with all examples in this report, the TSS is assumed to be configured in the CC mode (see Section 4.2). Note that the display of computational results is an option, so the speed of display is reported separately from the speed of the computational portion.

Tactile imprint acquisition (32 x 32 taxels) and storage:

001370 (GETI): Get new imprint. Required 53.5 ms for single command. Cycle period was 107 ms (one frame period must be allowed to pass before another imprint may be acquired).

001410 (GZPI): Get zero-pressure imprint. 73 ms for single command, 107 ms for cyclic execution.

001440 (GMPI): Get maximum-pressure imprint. 73 ms for single command, 107 ms for cyclic execution.

001470 (GCIR): Move current imprint to reference imprint. 8 ms.

Tactile data correction functions:

001520 (COR0): Does nothing. 30 us.
001530 (COR1): Subtract a user-specified background value from each taxel in the imprint. 20 ms.
001600 (COR2): Subtract the average taxel value for the imprint from each taxel in the imprint. 20 ms.
001650 (COR3): Each taxel was linearized between the bounds set by the average value in the zero-pressure imprint and the average taxel value in the maximum-pressure imprint, and re-scale into the range [0..37]. Cycle speed ranged from 160 to 210 ms, depending upon the magnitude of the taxel values.
002040 (COR4): Each taxel was linearized between its zero-pressure and maximum-pressure response value and re-scaled into the range [0..37]. Cycle speed ranged between 160 to 210 ms.

Computation of the average taxel value:

002340 (AVGI): Imprint average. 16 ms for the basic computation. Add 3.3 ms if the output is included.
002410 (AVGZ): Zero-pressure imprint average. Speed same as AVGI.
002460 (AVGM): Maximum-pressure imprint average. Speed same as AVGI.
002530 (AVGR): Reference imprint average. Speed same as AVGI.

Send all taxel values to the host via the serial line (BYTE mode recommended):

003060 (TAXI): Imprint taxels. 3.2 s at 9600 baud.
003100 (TAXZ): Zero-pressure imprint taxels. 3.2 s.
003120 (TAXM): Maximum-pressure imprint taxels. 3.2 s.
003140 (TAXR): Reference imprint taxels. 3.2 s.

Print the tactile data in a 32 x 32 array format on the console (BYTE mode recommended):

003260 (PRII): Print imprint. 4.5 s at 9600 baud.
003300 (PRIZ): Print zero-pressure imprint. 4.5 s.
003320 (PRIM): Print maximum-pressure imprint. 4.5 s.
003340 (PRIR): Print reference imprint. 4.5 s.
002600 (DEMO): Demonstration program continually acquired a new imprint, applied a user-selected correction function, and displayed the imprint on the console in an array format.

Calculate the taxel sum (i.e., total force):

003500 (TSMI): Imprint. 12 ms to perform the calculation, and 12.5 ms to display the results.

003520 (TSMR): Reference imprint. Same speed as TSMI.

003540 (TSCG): Change in the taxel sum relative to the reference imprint. 40 us to perform the computation, and 12.5 ms to display the results.

003660 (TSAB): Sum of the absolute taxel differences relative to the reference imprint. 24 ms for the computation, and 12.5 ms to display the result.

Calculation of the x and y first moments:

004470 (FMOI): Imprint. 18.5 ms to 350 ms for the case where all taxel values are 000 and 037, respectively. Add 34 ms for output display.

004520 (FMOR): Reference imprint. Same speed as FMOI.

004550 (FMCG): Change in the first moment relative to the reference imprint. 90 us for computation, and 34 ms for display of the results.

Threshold monitoring routines:

001070 (IWRD): Set a threshold or parameter value (see program listing in Appendix A for memory location assignments). 17 ms.

001130 (OWRD): Examine a specified memory location. 17 ms.

005020 (THRE): Continuous threshold monitoring using selected correction and threshold functions.

005310 (TTCH): Continually check taxel values against limits and report all violations. 107 ms cycle time for COR1 or 160 ms for COR4 if there were no violations. Otherwise, allow an additional 23 ms per violation (it is not possible to suppress the display). Thus, 23.3 s would be required to complete one threshold monitor cycle if all taxels were in violation.

005450 (TSCK): Check taxel sum against thresholds. 107 ms cycle time for COR1 and 160 ms for COR4 if there were no violations to report. Otherwise the cycle time is 160 ms and 210 ms, respectively.

005510 (TSCC): Check change in taxel sum. Same cycle time as TSCK.

006030 (TSAT): Check absolute taxel sum. Same cycle time as TSCK.

005620 (FMCK): Check x and y first moments. 160 ms for either COR1 or COR4 if there are no violations. Otherwise, the cycle time is 350 ms and 600 ms for COR1 and COR4, respectively.

005720 (FMCC): Check change in first moments. Same cycle speeds as FMCK.

5.4 Calibration procedure

The calibration procedure consisted of the acquisition of zero-pressure and maximum-pressure imprints. The latter is defined by the user to be a convenient maximum pressure (e.g., just below the saturation level of the most responsive taxel), and in this study this value was chosen to be 0.414 MPa (60 psi). The zero-pressure imprint is acquired with the command

> 001410(CR)_XXXXXX

where "XXXXXX" (or "XXX" in the BYTE mode) is the average taxel value that is automatically computed using procedure AVGZ(002410).

The data within any tactile data block may be displayed in one of two formats. The first is the "PRI" format in which the data are displayed as a 32 x 32 array of taxel values. (The orientation of the array relative to the sensor body is shown in Figure 5.1) This display mode is intended primarily for use while the TSS is configured in the Console Mode. The second is the "TAX" format in which the data are presented as a continuous stream of numeric characters (i.e., no taxel delimiters, carriage returns, or line feeds). This format is intended for use in the Host Mode where the time of data transmission must be kept to a minimum. The BYTE and WORD mode of the DCT-11EM cause the taxel value to be output as a 3 or 6 digit octal number, respectively. The zero-pressure tactile imprint in an array format is generated by the command

> 003300(CR)

The result is shown in Figure 5.2.

The maximum pressure imprint may be generated by applying a uniform hydrostatic pressure (using a pneumatically-inflated diaphragm) of approximately 0.414 MPa (60 psi) to the sensor surface, and then issuing the command

> 001440(CR)_XXXXXX

where "XXXXXX" is the average taxel value computed by the procedure

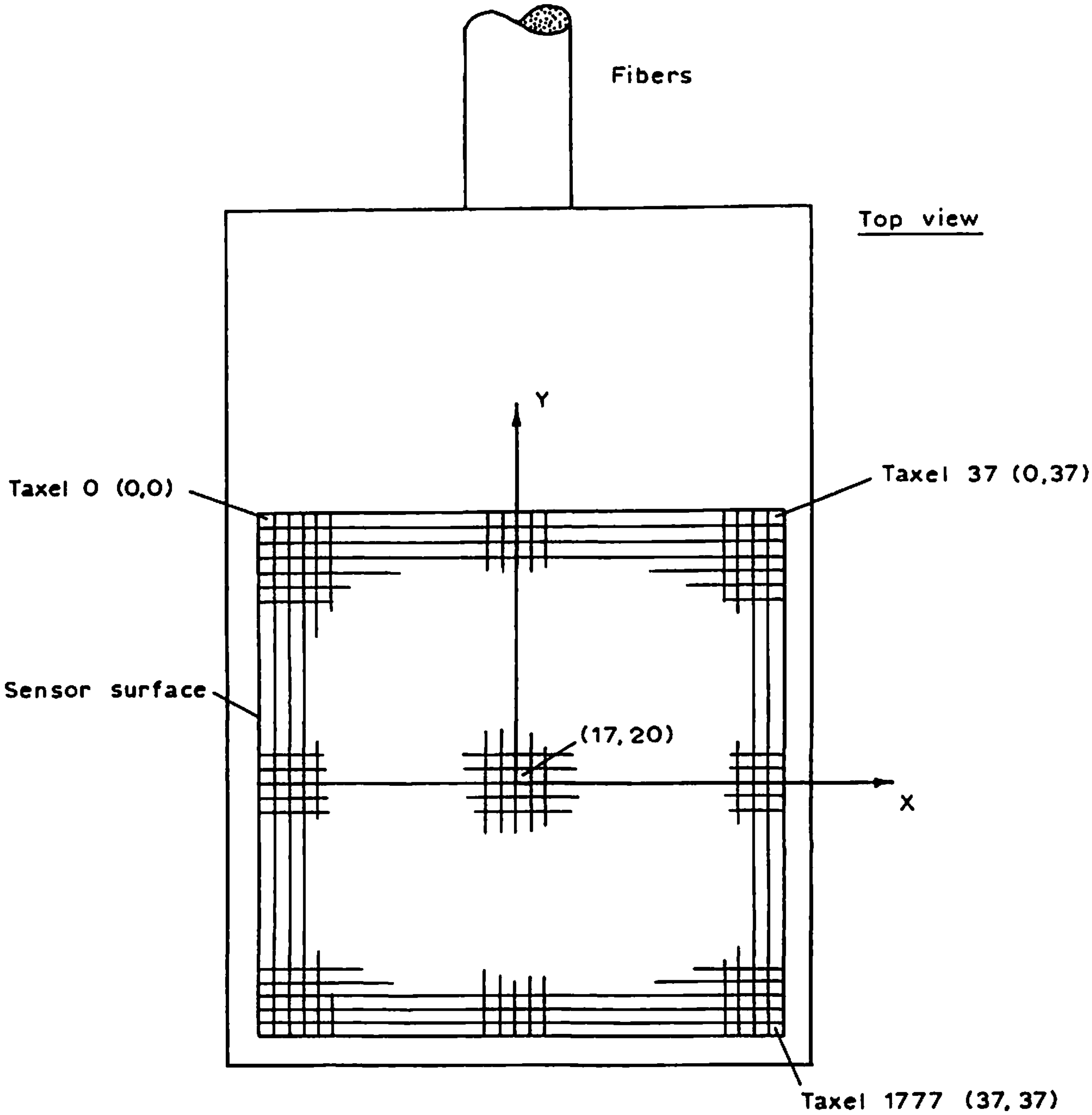


Figure 5.1: Labeling convention for taxel indices, and location of the TSA coordinate system (x and y axes). All numbers are octal.

```
001410_223
> 003300
160-217-222-223-221-223-222-221-220-222-222-226-224-226-230-227-226-226-227-225-226-226-226-226-235-233-230-232-226-226-224-215
161-222-223-223-222-223-222-220-220-220-220-223-223-223-223-224-224-226-225-224-223-225-226-226-230-230-225-227-230-226-227-221
160-222-223-222-223-222-217-220-217-217-220-221-221-222-222-223-222-226-224-222-222-224-221-225-224-224-226-227-225-224-225-221
161-221-222-221-222-220-216-220-215-216-215-222-221-221-220-224-222-224-223-222-222-223-223-223-223-223-223-227-227-227-225-222
157-217-217-220-216-224-216-216-215-217-216-222-221-223-222-226-225-226-226-226-223-222-222-222-223-222-220-227-226-226-225-223
157-215-216-216-216-217-217-215-214-217-217-223-222-221-224-225-224-226-227-225-223-223-221-221-222-220-220-225-223-225-225-223
156-216-217-216-216-216-215-217-217-217-220-223-223-225-226-225-224-232-226-223-225-224-221-222-221-221-221-222-222-224-226-223
156-215-215-215-215-217-215-217-216-221-223-223-224-225-227-230-225-227-226-225-226-223-222-222-221-224-221-223-221-224-223-222
156-215-215-215-215-216-215-217-216-222-230-227-224-230-230-227-226-225-225-225-225-225-223-223-223-222-220-223-221-222-223-222
155-215-215-216-214-215-220-217-217-221-223-230-223-227-227-226-225-226-225-226-231-225-224-223-223-221-217-221-220-220-222-221
155-216-216-213-216-216-216-221-220-222-222-222-223-230-227-227-225-226-226-225-226-226-224-224-226-222-217-222-220-217-221-220
156-215-216-215-216-217-216-222-220-223-222-225-224-227-227-225-224-232-226-225-226-226-225-225-224-223-217-222-220-220-222-217
156-217-215-215-215-217-216-220-222-225-223-225-224-225-226-225-224-225-225-226-225-227-226-224-223-222-217-221-217-216-217-215
156-215-215-214-215-215-216-221-223-224-223-223-227-230-226-224-224-225-224-226-223-225-225-233-223-223-216-220-216-215-216-216
156-217-216-215-216-217-217-223-223-223-223-225-226-230-226-225-226-226-225-231-226-227-224-232-223-222-216-220-216-217-216-217
157-217-217-215-215-217-220-223-223-225-223-223-224-227-227-227-226-224-224-226-226-225-223-224-221-220-216-221-216-217-215-217
155-216-216-216-215-220-220-222-223-223-224-224-225-226-227-226-225-226-224-225-226-226-225-223-221-220-215-216-217-216-216-216
155-217-217-215-217-223-223-225-223-222-225-225-225-227-230-226-225-226-225-227-227-227-223-222-222-221-216-217-217-217-217-217
157-222-217-216-220-221-224-226-224-224-226-226-226-230-226-226-226-231-230-227-227-230-225-224-223-224-217-223-217-221-220-217
161-222-220-217-221-222-222-224-224-224-227-232-225-227-225-227-226-231-231-226-227-226-226-223-223-226-217-222-217-220-220-217
155-221-220-217-220-220-222-223-223-222-224-224-225-231-226-226-226-227-227-225-226-230-224-223-223-222-220-222-220-220-220-215
157-222-223-220-220-220-220-223-222-222-223-223-223-226-225-226-225-233-230-226-226-226-230-225-222-221-220-222-220-224-221-217
156-223-223-220-222-222-222-223-222-224-225-225-224-227-225-225-226-226-226-226-226-224-224-223-221-221-220-220-220-222-217-216
162-224-223-223-222-222-221-223-222-223-224-226-225-226-223-223-225-230-225-226-225-225-223-223-223-222-220-221-221-222-216-215
161-225-226-227-225-224-222-225-223-224-226-231-226-226-224-224-223-231-224-225-225-226-224-225-227-224-221-222-224-220-220-216
164-227-227-226-233-225-225-231-224-230-230-230-230-226-225-224-224-227-231-225-227-225-226-227-230-230-221-224-223-222-221-217
163-227-231-231-231-227-230-233-227-232-231-227-230-231-225-227-223-226-225-226-230-227-227-227-224-225-222-224-223-222-217-213
163-230-232-230-235-233-233-235-235-233-233-232-232-234-232-227-224-227-227-230-231-231-230-232-226-224-223-227-226-230-225-220
166-231-233-235-235-234-237-244-237-241-234-233-237-237-240-235-226-231-232-231-231-230-233-236-231-230-226-224-232-233-232-226
166-233-235-234-235-231-235-242-236-235-236-233-233-236-243-233-227-234-232-235-232-233-226-230-231-232-226-230-237-241-235-230
162-233-234-236-233-234-234-237-236-235-235-232-232-231-233-232-231-235-233-235-232-231-226-231-227-227-225-235-242-241-235-227
162-226-222-232-230-230-233-234-232-236-232-230-231-234-230-232-231-233-235-233-233-230-231-232-233-233-227-241-241-241-235-233

>
```

Figure 5.2: Zero-pressure tactile imprint displayed in a 32 x 32 taxel format. Average value is 223 (octal).

AVGM(002460). This data may be displayed in the array format:

> **003320(CR)**

The result is shown in Figure 5.3.

The pneumatic calibration diaphragm may also be used to quantify the relationship between the applied pressure and the sensor response. Figure 5.4 shows the pressure response of the TSA using WP125 and BN890 transducer and cover membranes, respectively. The results were qualitatively similar to the observed behavior of the LA-TS (Figure 3.6). The force response range of each taxel was approximately 0 - 0.4 N.

At this point, all necessary calibration data is acquired and any data processing procedure listed in Section 5.3 may now be executed.

5.5 Performance Examples

The most common operation performed by the TSS will probably be the acquisition of the tactile imprint. An imprint is first acquired with the command:

> **001370(CR)** (GETI: imprint of a ring)

These unprocessed data may be displayed in an array format:

> **003260(CR)** (PRII)

The result is shown in Figure 5.5.

These raw or unprocessed data may be corrected by one of five correction functions. COR0(001520) leaves the data unmodified, and is merely a "nil" operation placeholder for the threshold monitor subroutine THRE(005020) (to be discussed later). COR1(001530) subtracts a constant, user-specified background (or zero-pressure) value from each taxel in the imprint. It is a simple function to implement and has the advantage of being rapid (see Section 5.3). The background value is stored as a word starting at location 000262 and has the

```
001440.250
> 003320
164-226-233-235-232-236-235-234-235-242-241-246-246-246-251-251-251-250-251-245-244-244-241-241-253-251-246-245-240-235-232-220
165-232-237-240-236-243-243-235-241-240-240-250-246-244-244-251-251-262-252-252-246-251-247-247-252-252-244-246-244-237-236-223
164-233-237-237-241-242-236-237-240-243-241-250-250-244-246-250-247-263-262-251-251-252-244-252-252-247-253-254-245-240-236-230
166-235-240-242-240-241-243-241-237-244-242-256-247-244-243-250-246-254-254-251-252-255-254-256-261-263-253-257-253-247-244-231
165-234-236-240-235-245-241-235-236-244-242-254-250-251-245-253-253-255-257-257-255-254-256-256-260-261-253-257-251-247-243-234
165-231-233-234-235-237-242-234-236-245-246-260-245-243-245-247-245-255-262-264-257-253-255-256-257-252-251-251-247-246-246-234
165-234-237-237-236-240-240-236-244-244-246-256-252-254-255-247-251-264-257-255-264-257-253-255-254-254-252-255-246-246-247-236
165-234-236-235-235-245-241-237-242-250-253-254-257-255-260-255-253-262-257-256-257-254-253-253-254-260-253-255-247-245-242-234
165-234-233-233-235-241-237-237-241-245-252-256-252-255-261-253-254-251-254-255-261-255-253-254-255-251-246-254-244-244-243-234
165-233-235-235-236-242-243-241-243-247-247-263-245-255-263-255-255-257-253-257-263-257-253-251-253-252-246-252-246-242-244-237
165-240-236-232-241-247-246-251-252-252-246-245-246-262-264-261-256-262-260-256-262-256-252-253-255-252-246-251-246-243-246-241
167-236-242-240-244-247-245-247-252-254-247-256-255-262-266-256-256-270-263-254-256-257-254-260-256-254-246-253-247-246-246-237
170-243-242-247-242-250-242-243-252-257-251-261-255-262-261-261-260-262-266-260-255-261-255-255-257-252-252-256-245-244-246-237
171-244-241-240-237-233-243-245-255-256-247-257-263-266-275-261-263-265-264-263-256-255-254-264-255-255-251-253-247-246-253-241
171-245-245-240-242-244-244-251-256-255-254-264-266-275-272-263-267-262-262-265-261-262-254-261-255-252-247-254-247-251-246-244
170-244-244-240-246-244-243-254-256-265-263-264-262-271-275-265-260-262-261-260-255-254-253-260-254-254-253-254-255-252-246-253
164-241-241-240-246-247-246-254-264-255-262-262-264-266-272-263-260-261-256-257-256-264-255-260-256-253-251-247-252-252-247-251
166-240-244-240-246-255-254-262-260-255-267-265-265-266-272-263-262-263-260-261-263-263-253-253-262-253-246-247-250-250-246-252
171-242-243-242-247-252-260-261-262-260-266-266-266-271-271-264-267-272-264-256-261-262-257-262-262-260-246-253-250-247-246-246
173-245-243-245-247-251-254-261-262-260-267-271-266-267-266-266-264-266-266-257-262-260-262-260-260-262-252-256-256-250-250-244
164-246-245-242-247-250-254-257-260-260-266-264-264-272-266-262-266-266-264-262-260-263-257-260-257-261-256-256-252-251-251-240
166-242-246-242-246-247-245-260-257-254-262-261-260-264-265-262-264-272-266-262-262-262-262-260-253-255-254-252-253-255-251-237
163-242-244-242-250-251-252-256-253-262-265-264-261-264-262-257-265-263-270-263-260-254-261-256-252-253-254-251-252-254-243-243
170-243-246-247-251-250-251-256-255-257-264-265-260-264-261-256-256-265-264-261-261-260-254-256-255-252-250-252-254-254-245-240
167-243-247-254-255-254-250-257-253-260-266-266-261-260-255-255-251-265-260-257-261-257-255-263-262-254-253-255-256-246-247-244
172-244-247-251-262-252-253-263-251-262-262-264-261-257-253-251-254-262-263-257-260-254-256-264-264-264-251-255-252-251-252-241
170-243-247-254-256-254-260-264-256-261-257-255-264-262-260-256-251-261-254-256-262-255-256-263-253-251-250-252-250-253-244-223
167-240-246-246-256-257-262-263-257-257-260-256-261-264-262-252-251-253-256-256-255-253-251-261-254-251-252-253-254-254-247-235
171-240-244-251-253-256-264-272-262-264-257-256-263-262-262-255-250-254-256-254-252-251-254-255-253-251-250-243-256-257-253-244
172-241-245-246-251-246-255-264-255-255-256-253-253-257-261-252-246-256-254-260-251-251-243-246-250-250-244-246-262-262-254-243
165-241-243-247-246-251-250-255-254-254-254-247-247-244-247-247-247-254-253-254-250-246-242-244-243-241-240-252-254-257-247-240
164-233-225-242-241-240-246-246-244-253-251-243-244-246-240-244-243-244-252-245-244-237-242-241-242-242-235-246-247-247-244-240
```

Figure 5.3: Maximum-pressure tactile imprint. A pressure of 0.414 MPa (60 psi) was applied with a pneumatic diaphragm. The average taxel value is 250 (octal).

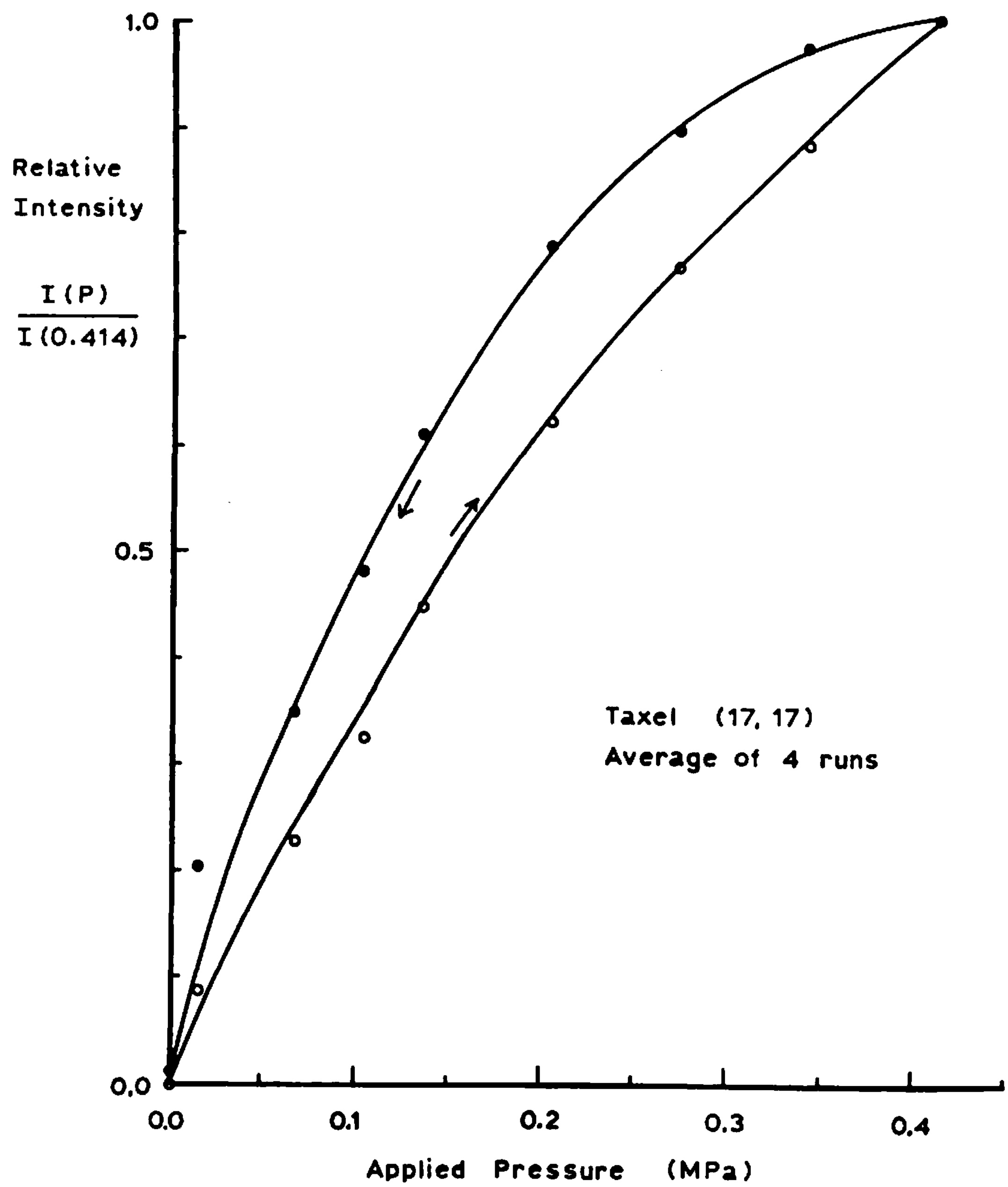


Figure 5.4: Typical pressure response characteristics of the TIR-TSA using WP125 transducer membrane and BN890 cover membrane.

001370

> 003260

156-215-216-220-217-222-220-220-217-221-222-224-223-224-226-226-225-224-225-223-223-224-224-223-230-227-223-224-221-220-216-213

156-215-221-221-221-223-222-220-217-220-221-223-223-223-223-224-224-226-224-224-223-224-224-225-226-225-221-223-223-222-220-213

155-217-221-222-223-222-217-220-216-220-220-221-222-222-223-224-223-227-225-223-222-224-221-225-223-223-223-225-222-220-217-215

156-217-221-221-221-221-217-220-216-217-217-223-223-222-222-226-224-226-225-224-223-224-223-223-223-223-221-224-224-222-220-216

155-216-217-220-217-224-217-217-216-220-220-224-224-226-226-231-231-232-232-232-226-224-223-223-223-221-217-223-221-222-217-215

156-215-215-216-216-217-217-217-217-222-225-232-230-230-233-235-235-241-241-242-236-230-224-224-224-221-217-222-220-221-221-215

155-215-216-216-216-217-217-221-223-225-230-240-242-246-252-246-251-262-252-245-244-235-226-225-223-221-220-221-217-220-220-215

155-215-215-215-216-220-217-223-224-233-242-250-257-253-260-254-251-270-267-264-255-243-233-227-225-224-220-222-217-220-217-216

155-215-215-215-215-221-221-224-227-237-247-246-236-241-240-235-235-237-246-253-263-261-243-234-230-224-217-222-217-217-217-216

155-215-215-216-216-222-227-231-235-241-233-240-226-231-231-230-227-231-232-237-251-260-257-243-234-226-221-221-217-216-217-215

155-215-216-215-221-224-230-242-241-233-226-224-225-231-227-227-225-227-227-230-235-241-251-256-245-232-222-222-217-215-216-215

155-216-217-217-222-230-234-237-233-230-224-226-225-227-230-225-223-232-226-225-227-232-240-264-262-240-224-223-217-217-220-215

155-217-216-220-222-235-233-230-227-230-223-226-224-225-226-225-223-224-225-225-225-227-233-246-266-243-227-224-217-215-215-213

155-216-216-217-222-225-232-226-225-226-224-223-227-230-226-224-224-225-224-226-223-225-227-245-257-257-231-225-217-215-214-215

156-217-217-221-225-235-232-227-225-225-224-225-226-231-232-225-226-226-224-231-226-226-226-237-246-265-234-226-221-216-215-215

156-217-221-221-227-236-230-227-225-226-224-224-225-227-230-226-226-224-223-225-225-225-224-230-241-265-241-231-222-217-215-215

155-217-220-222-231-241-227-226-225-224-224-224-225-226-226-225-224-226-223-226-225-226-225-226-242-264-236-225-222-216-215-215

156-217-222-223-233-247-235-230-225-224-226-226-225-226-227-226-225-227-226-226-226-227-226-226-243-260-235-227-222-217-217-216

157-221-222-223-232-244-237-232-227-225-226-227-226-227-225-225-225-230-230-226-226-227-226-231-244-267-234-231-223-221-216-216

160-221-222-223-232-242-236-231-226-224-230-232-226-227-224-224-225-227-231-226-227-226-227-233-250-267-235-231-223-221-216-216

155-221-221-222-227-236-240-231-225-223-225-226-226-227-225-224-225-226-227-226-226-230-227-237-257-257-233-226-222-220-216-215

155-221-222-222-226-234-235-236-227-224-224-224-224-227-226-225-225-233-231-227-227-231-235-253-261-243-227-226-222-222-217-215

155-221-222-222-225-232-242-244-230-227-227-226-225-227-224-225-226-226-227-227-227-230-243-261-251-233-224-222-220-220-215-215

157-223-222-223-223-226-233-246-237-231-231-232-226-226-223-223-225-227-225-230-232-240-254-257-243-230-222-220-215-217-215-213

157-223-224-225-226-227-231-243-244-246-242-237-232-227-225-226-224-233-230-234-242-255-260-251-237-230-220-220-222-217-215-214

160-223-224-225-232-226-230-236-240-254-254-253-243-235-232-231-231-236-243-250-264-256-245-240-233-227-217-221-217-216-214-213

160-222-224-226-226-227-230-234-233-244-247-253-262-255-250-250-244-257-254-264-261-244-233-232-224-222-216-217-217-216-213-211

157-222-223-222-231-226-227-232-231-235-237-243-252-261-263-254-251-251-252-244-237-232-227-227-223-220-215-220-217-217-215-213

160-222-223-224-226-226-226-234-232-235-233-235-240-243-244-242-234-237-236-233-227-226-226-227-222-220-215-215-217-217-216-216

160-221-222-222-224-223-225-231-227-227-231-227-230-233-234-230-226-227-226-227-223-223-220-220-220-220-214-216-220-221-216-220

156-221-221-222-223-224-225-227-226-226-226-225-224-224-225-224-223-224-222-223-221-220-216-217-216-217-214-220-217-222-217-222

157-221-214-223-222-222-225-226-224-227-224-222-224-226-223-225-222-223-225-223-223-221-221-221-221-221-216-224-223-226-223-232

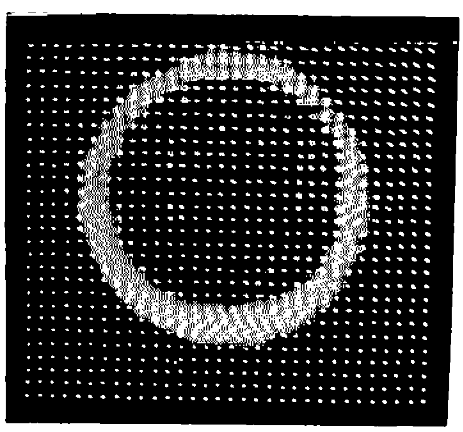


Figure 5.5: Top: uncorrected imprint of a ring. Bottom: appearance of the imprint on the display array.

default value of 224. For example:

```
> 001530(CR)     (COR1: applies correction )
                  (function 1 to the current)
                  (imprint data, e.g., a ring.)

> 003260(CR)     (PRII: prints data stored )
                  (in imprint data area in )
                  (an array format.         )
```

The results are shown in Figure 5.6.

Any parameter or threshold value that is stored in memory may be changed by the complementary procedures IWRD(001070) and OWRD(001130) which write a word to memory and read a word from memory, respectively. For example, the background value used by COR1 may be examined with the following command sequence:

```
> 001130(CR)A      (OWRD: system responds by )
                    (requesting the 6-digit    )
                    (octal address of the word )

> 000262(CR)_000224    ("000262" is the entered)
                        (address, and the value )
                        (stored at that address  )
                        (is 000224.              )
```

Similarly, the background value may be changed as follows:

```
> 001070(CR)A       (IWRD: system responds by )
                     (requesting the address of  )
                     (the word.                  )

> 000262(CR)V      (System requests 6-digit     )
                    (octal value of the word    )
                    (to be stored.              )

> 000240(CR)        (The value to be stored is)
                     (entered.                  )
```

Another imprint may now be acquired and corrected using the new background value for COR1:

```
> 001370(CR)    (GETI: get new ring imprint.)
```

```
 001130A
> 000262:224
> 001370
> 001530
> 003260
000-000-000-000-000-000-000-000-000-000-000-001-000-000-002-002-002-001-002-000-000-000-000-000-005-004-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-002-000-000-000-000-000-001-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-003-002-000-000-001-000-001-000-000-000-001-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-002-001-003-002-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-001-001-004-003-006-006-007-007-007-003-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-001-003-012-010-010-014-015-015-022-022-024-015-006-002-001-001-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-002-004-010-023-026-033-042-033-035-052-037-030-025-014-005-002-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-001-003-013-025-036-051-040-046-037-034-055-055-053-043-025-012-005-001-000-000-000-000-000-000-000
000-000-000-000-000-000-000-002-010-021-031-027-016-020-017-013-013-017-027-036-052-051-024-013-006-001-000-000-000-000-000-000
000-000-000-000-000-000-005-012-017-024-013-017-003-006-006-005-003-005-007-014-030-044-043-025-013-004-000-000-000-000-000-000
000-000-000-000-000-002-011-025-023-012-003-001-001-005-003-004-002-004-004-005-013-017-031-042-025-007-000-000-000-000-000-000
000-000-000-000-000-007-016-020-011-005-001-002-001-003-004-000-000-006-002-001-004-006-016-046-046-020-002-000-000-000-000-000
000-000-000-000-001-016-015-007-004-004-001-003-000-002-003-002-000-002-002-002-002-005-011-025-055-025-006-001-000-000-000-000
000-000-000-000-001-004-012-005-003-002-000-001-004-00'-003-001-001-002-002-003-001-002-004-023-042-043-010-002-000-000-000-000
000-000-000-000-004-020-011-004-001-001-000-001-001-005-007-001-002-002-000-005-001-002-002-013-026-052-012-004-000-000-000-000
000-000-000-000-007-020-006-003-002-002-000-001-000-003-004-003-002-001-000-001-001-001-002-006-021-052-022-006-000-000-000-000
000-000-000-000-011-023-006-003-002-000-001-001-001-002-002-001-000-002-000-003-001-003-002-003-021-052-017-004-000-000-000-000
000-000-000-000-013-031-012-005-001-000-002-002-001-002-002-001-000-002-001-000-002-002-000-002-021-042-014-004-000-000-000-000
000-000-000-000-010-025-015-006-003-002-002-003-003-004-002-001-001-005-004-002-002-003-002-006-024-055-013-007-000-000-000-000
000-000-000-001-011-023-015-007-003-002-005-010-003-003-001-001-001-004-005-002-003-002-003-010-031-053-013-007-001-000-000-000
000-000-000-000-006-017-020-007-003-000-002-002-002-003-001-000-002-003-004-002-002-004-005-015-041-040-011-003-000-000-000-000
000-000-000-000-003-013-015-014-002-000-001-001-000-003-001-001-001-006-005-003-003-005-012-033-043-021-005-001-000-000-000-000
000-000-000-000-003-011-023-025-006-004-004-003-001-003-001-001-003-003-004-003-005-006-022-044-033-012-002-000-000-000-000-000
000-000-000-000-001-005-013-030-017-007-007-006-003-003-000-000-001-004-003-005-007-017-036-043-023-006-000-000-000-000-000-000
000-000-000-002-002-004-005-021-025-026-020-014-006-003-001-002-001-007-005-011-020-036-043-033-016-005-000-000-000-000-000-000
000-000-000-001-006-002-005-015-017-036-035-033-021-012-011-007-007-014-022-031-050-041-025-017-011-005-000-000-000-000-000-000
000-000-000-002-003-002-005-011-012-022-027-033-045-036-031-031-025-041-040-051-042-023-012-007-001-000-000-000-000-000-000-000
000-000-000-000-004-002-003-006-007-012-014-020-032-042-050-035-034-032-033-025-016-007-003-003-000-000-000-000-000-000-000-000
000-000-000-000-002-002-002-007-006-011-007-011-015-020-022-021-013-015-014-010-004-003-003-003-000-000-000-000-000-000-000-000
000-000-000-000-001-000-002-005-004-005-006-005-005-011-012-007-004-005-003-005-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-001-004-001-002-002-001-001-000-003-001-000-002-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-001-002-000-003-000-000-001-002-000-001-000-000-001-000-000-000-000-000-000-000-000-000-000-001-000-005
```

Figure 5.6: Tactile imprint of a ring after correction by COR1. The default background value (= 224) was used.

> **001530(CR)** (COR1: correction function 1)

The results are shown in Figure 5.7.

Overton [18] has examined a number of tactile data correction algorithms and concluded that sensor linearization gave satisfactory performance at reasonable computational cost in terms of memory and time requirements. A similar algorithm was implemented on the TSS preprocessor. Correction function 3, COR3(001650), linearizes the response of each taxel between the average zero-pressure and average maximum-pressure values, and then normalizes the response so that it lies in the range [0, 37]. This range was chosen as being approximately the average difference in values that taxels exhibited when the pressure was varied from zero to the maximum value. Figure 5.8 shows the results of COR3 applied to an imprint of a ring.

Correction function 4, COR4(002040), is identical to COR3 except that each taxel is linearized between its own zero-pressure and maximum-pressure response value. Figure 5.9 shows the result of COR4 applied to a ring imprint. Note that there is significantly less variation in the COR4 than in the COR3 imprint. This is to be expected as COR4 takes into account the specific response of each taxel, whereas the COR3 does not and therefore cannot remove any variations in taxel response behaviors that are initially present. Figures 5.10 to 5.13 show good TSA response to uniform pressures after applying COR4. Figures 5.14 to 5.18 show imprints of other objects after application of COR4. Figure 5.19 shows the imprint of a knife-edge wedge from which the spatial resolution of the sensor was determined to be approximately 2 mm.

The function TSMI(003500) computes the sum of all taxel values in the imprint (after the taxels have been corrected) and effectively represents the total force applied to the sensor (see Figure 5.4). For example, the taxel sum of a zero-pressure imprint after correction by function 4 is determined by the command sequence

> **001370(CR)** (GETI: no contacting objects.)

> **002040(CR)** (COR4: correction function 4.)

```
001070A
> 000262V
> 000240
> 001370
> 001530
> 003260
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-002-004-004-013-016-015-006-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-012-016-025-032-022-022-037-036-024-025-013-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-002-015-026-031-014-014-007-003-014-015-021-037-031-013-002-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-011-020-012-000-000-000-000-000-000-000-001-013-025-034-016-004-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-006-020-000-002-000-000-000-000-000-000-000-000-000-002-013-027-017-000-000-000-000-000-000-000
000-000-000-000-000-000-000-014-024-002-000-000-000-000-000-000-000-000-000-000-000-000-014-034-012-000-000-000-000-000-000-000
000-000-000-000-000-000-004-022-005-000-000-000-000-000-000-000-000-000-000-000-000-000-000-002-026-035-000-000-000-000-000-00C
000-000-000-000-000-001-010-006-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-011-041-011-000-000-000-000-000
000-000-000-000-000-000-015-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-041-021-000-000-000-000-000
000-000-000-000-000-007-024-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-024-023-004-000-000-000-000
000-000-000-000-000-013-020-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-013-036-006-000-000-000-000
000-000-000-000-000-023-021-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-013-036-002-000-000-000-000
000-000-000-000-000-024-035-003-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-014-026-001-000-000-000-000
000-000-000-000-000-022-042-011-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-027-023-002-000-000-000-000
000-000-000-000-000-015-042-022-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-004-042-020-001-000-000-000-000
000-000-000-000-000-004-035-031-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-014-042-012-000-000-000-000-000
000-000-000-000-000-000-012-042-014-000-000-000-000-000-000-000-000-000-000-000-000-000-000-007-034-030-000-000-000-000-000-000
000-000-000-000-000-000-007-041-026-007-000-000-000-000-000-000-000-000-000-000-000-000-003-025-035-010-000-000-000-000-000-000
000-000-000-000-000-000-000-015-036-030-015-004-000-000-000-000-000-000-000-000-000-000-001-021-041-024-000-000-000-000-000-000
000-000-000-000-000-000-000-003-013-041-042-034-007-000-000-000-000-000-000-000-000-011-026-041-037-013-000-000-000-000-000-000
000-000-000-000-000-000-000-000-001-022-042-042-042-024-016-013-007-017-023-031-042-041-023-012-002-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-006-015-026-042-042-041-042-041-042-042-042-037-016-002-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-002-005-016-025-035-025-022-023-025-015-006-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-002-004-007-005-000-002-002-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000

>
```

Figure 5.7: Tactile imprint of a ring after correcting the data using COR1 with a background value of 240; compare to Figure 5.6.

```
002410_223
002460_250
001370
001650
003260
000-000-000-000-000-000-000-000-000-000-000-001-000-001-003-004-003-001-004-000-000-000-000-000-007-004-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-001-003-001-001-000-000-001-001-003-001-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-003-000-007-006-001-000-003-000-003-000-000-001-003-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-001-000-001-001-007-006-012-011-007-006-006-003-001-001-003-000-003-001-000-000-000
000-000-000-000-000-001-000-000-000-000-000-003-003-006-007-015-020-023-026-025-017-012-007-004-003-000-000-003-000-000-000-000
000-000-000-000-000-000-000-000-000-000-004-017-014-014-023-031-033-037-037-037-037-031-022-017-012-003-000-003-000-000-000-000
000-000-000-000-000-000-000-000-000-003-011-026-034-037-037-034-034-037-037-037-037-037-037-030-014-006-001-000-000-000-000-000
000-000-000-000-000-000-000-000-001-012-026-037-037-037-034-025-015-020-017-017-025-031-037-037-031-020-006-003-000-000-000-000
000-000-000-000-000-000-000-000-003-017-037-037-022-022-017-012-007-004-006-006-011-012-020-037-037-031-006-004-000-000-000-000
000-000-000-000-000-000-001-004-012-031-026-033-006-011-011-006-004-004-004-007-012-007-007-012-033-037-020-007-000-000-000-000
000-000-000-000-000-000-000-014-030-036-014-004-003-007-006-004-001-003-004-003-004-003-001-004-014-026-026-014-000-000-000-000
000-000-000-000-000-000-003-020-037-026-006-007-003-006-006-001-000-011-003-001-003-004-003-004-006-017-031-026-006-000-000-000
000-000-000-000-000-001-006-020-037-023-006-007-001-003-004-003-000-003-003-004-003-006-004-003-004-007-031-037-011-000-000-000
000-000-000-000-000-000-007-026-034-015-004-003-007-011-006-003-001-003-003-004-001-003-003-015-001-004-017-037-014-000-000-000
000-000-000-000-000-001-012-036-025-011-003-003-004-011-012-003-004-003-001-007-004-003-001-011-000-001-007-036-015-001-000-000
000-000-000-000-000-003-014-037-022-014-003-001-003-006-007-006-004-003-001-004-003-004-001-004-000-000-012-036-023-004-000-000
000-000-000-000-000-004-014-034-026-007-004-003-003-004-004-003-001-003-000-004-003-004-003-000-000-000-004-022-015-001-000-000
000-000-000-000-000-007-020-037-025-006-006-004-003-004-004-003-001-004-003-003-004-004-000-000-000-001-004-023-011-000-000-000
000-000-000-000-000-004-015-037-033-012-011-007-004-006-003-003-003-007-004-004-003-004-001-000-000-004-011-031-007-001-000-000
000-000-000-000-000-004-012-033-037-017-017-014-006-006-003-003-003-006-011-004-006-004-004-003-001-015-022-025-007-000-000-000
000-000-000-000-000-000-006-020-033-022-014-006-004-006-001-000-003-004-004-001-003-004-001-000-004-017-034-015-001-000-000-000
000-000-000-000-000-000-000-014-025-023-014-007-003-007-001-001-001-011-006-004-004-004-007-011-012-030-022-011-000-000-000-000
000-000-000-000-000-000-001-007-012-033-036-020-011-011-003-003-004-004-006-006-006-004-007-014-022-022-011-003-000-000-000-000
000-000-000-000-000-000-000-006-011-017-036-037-022-014-004-004-004-007-006-006-007-011-014-025-030-012-001-000-000-000-000-000
000-000-000-003-001-000-000-004-003-007-023-036-034-026-015-014-006-017-011-012-014-025-025-031-022-007-000-000-000-000-000-000
000-000-001-003-012-001-001-007-003-011-015-022-030-036-030-026-023-034-037-026-031-023-022-020-014-011-000-000-000-000-000-000
000-000-001-004-004-003-004-011-003-007-007-011-017-025-026-026-020-034-023-025-023-015-011-006-001-000-000-000-000-000-000-000
000-000-000-000-011-004-004-006-004-006-006-007-011-012-012-007-004-007-011-011-011-004-001-003-000-000-000-000-000-000-000-000
000-000-000-000-003-003-003-011-006-006-004-004-007-007-012-006-000-004-003-001-000-000-001-004-000-000-000-000-000-000-000-000
000-000-000-000-001-000-003-007-006-006-006-001-001-004-006-003-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-001-001-006-003-003-003-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-003-004-001-004-000-000-000-003-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-003-000-011
```

Figure 5.8: Imprint of a ring after application of correction function COR3.

```
 001370
> 002040
> 003260
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-001-001-000-001-001-001-000-001-001-003-001-001-002-002-001-001-001-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-001-001-002-001-003-001-002-002-002-002-001-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-004-001-002-005-005-005-007-007-007-011-010-007-012-011-002-002-002-002-000-000-000-000-000-000
000-000-000-000-000-001-003-002-004-006-007-010-011-013-015-020-020-016-013-013-011-005-002-002-001-000-000-000-000-000-000
000-000-001-002-004-002-003-006-006-011-012-017-025-030-034-033-035-032-030-023-016-012-010-005-002-000-000-000-000-000-000
000-000-000-000-000-001-003-006-010-013-025-036-033-033-032-030-030-031-037-037-026-022-012-006-003-000-000-000-000-000-000
000-000-000-000-001-003-006-010-013-024-031-030-016-013-010-006-010-013-020-025-036-037-026-015-010-003-000-000-000-000-000
000-000-000-002-001-005-010-015-024-030-020-012-006-003-002-002-002-002-004-010-015-030-037-026-014-006-002-000-000-000-000
000-001-000-000-004-006-013-023-027-016-007-002-001-000-000-000-000-000-000-001-002-010-027-037-026-012-002-000-000-000-000
000-000-000-002-004-010-020-022-016-006-002-001-000-000-000-000-000-000-000-000-001-002-013-035-037-016-007-001-000-000-000
000-001-000-003-007-022-023-020-007-003-001-002-001-001-000-001-001-000-000-000-002-002-005-021-037-027-012-005-001-001-000
000-000-002-006-010-022-026-013-004-003-001-001-000-000-004-001-000-000-000-000-000-000-002-010-027-037-014-005-002-000-000
000-000-003-005-012-025-022-007-002-002-001-000-000-000-000-000-000-000-000-000-000-000-001-005-021-037-021-007-003-000-000
000-000-001-007-014-031-020-007-002-001-001-001-001-000-000-000-000-000-000-001-000-000-001-003-017-037-023-012-003-001-000
003-001-004-010-017-031-017-005-002-002-000-002-000-000-000-000-000-000-001-000-000-000-001-002-013-035-024-012-003-000-000
000-000-003-006-014-031-016-004-003-000-000-001-000-000-000-000-000-000-000-000-000-000-000-002-011-034-022-006-003-000-000
000-000-000-005-015-027-015-005-002-001-002-000-000-000-000-000-000-000-000-000-000-000-000-004-014-036-020-006-003-000-000
000-000-001-005-013-024-021-006-003-002-001-001-001-000-000-000-000-000-000-001-001-000-002-005-020-035-016-004-002-001-000
000-000-000-003-012-022-022-007-003-001-000-001-000-000-000-000-000-000-000-000-000-001-000-010-030-030-010-002-001-000-000
000-000-000-003-004-013-025-015-003-001-001-000-000-000-000-000-000-000-000-000-000-001-005-016-032-020-006-003-000-000-000
000-000-000-001-004-010-022-025-011-004-003-002-001-000-000-001-000-000-000-000-001-003-012-026-031-012-004-001-000-000-000
000-000-000-000-001-005-013-023-021-010-005-002-001-000-000-001-000-000-001-001-003-012-024-034-021-010-001-000-000-000-000
000-000-000-000-000-003-005-013-023-022-010-006-001-002-001-000-000-000-001-004-007-017-030-021-005-001-000-000-000-000-000
000-000-000-000-000-000-001-003-013-022-021-016-010-006-005-002-005-005-010-020-027-030-014-010-002-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-006-014-021-026-025-020-017-020-023-024-025-021-007-001-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-003-006-013-020-025-026-027-025-020-010-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-004-002-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000

>
```

Figure 5.9: Imprint of a ring after application of correction function COR4.

```
002040
002340_000
003260
010-000-000-000-000-000-000-000-000-000-002-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-002-002-003-004
000-002-002-001-000-001-001-000-001-001-001-000-001-001-001-001-001-001-001-000-000-000-000-000-001-000-000-000-002-002-004
000-002-002-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-001-002-000-000-000-000-000-001-000-000-001-001-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-001-000-000-000-000-000-000-001-000-001-001-000-000-000
004-001-002-002-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-001-000-000-000-000-001-001-001-001
000-001-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-002-000-001-000-000-000-001-000-001-000-001-000-000-000-001-002-001-000-001-001-001-000-000-001-000-001-001-000-002-001-001
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000
000-000-001-000-001-000-000-001-000-001-000-000-000-000-000-000-001-000-001-000-000-001-000-000-001-001-001-001-000-001-000-001
001-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-001-000-000-000-001-000-001-000-000-001-001-000-003
000-001-001-000-000-001-000-001-001-001-000-001-001-000-000-000-000-001-000-000-001-000-002-001-000-001-001-000-000-001-000-001
000-000-000-000-001-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-001-001-001-000-001-001-000-000-003
000-004-001-000-000-000-001-001-001-000-001-001-001-001-001-001-000-001-001-001-001-000-001-001-000-000-000-001-000-000-000-000
000-002-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-003-000
000-000-000-001-000-000-001-000-001-001-000-000-000-001-001-000-001-001-000-000-001-001-000-001-001-000-000-000-000-001-000-000
000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-002-000-000-000-000-000-000-002-001-000-000-002-000-000-000-002-000-000-000-000-000-000-000-000-001-000-000
000-005-000-000-000-002-002-000-000-002-000-000-000-000-002-000-000-002-002-000-002-000-000-000-000-003-002-002-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
```

Figure 5.10: Taxel values after applying correction function COR4. The applied pressure was 0.0 MPa.

002040
> 002340.016
> 003260
020-011-012-011-012-013-010-013-014-016-016-014-014-014-013-014-015-014-014-014-013-015-016-013-011-013-013-013-014-011-012-000
020-010-015-014-015-016-015-016-017-020-020-017-017-015-015-015-015-015-015-020-017-016-017-017-016-014-016-014-015-012-011-000
020-021-015-016-015-016-016-021-017-023-022-020-022-020-016-017-017-020-021-020-020-020-020-020-016-020-020-017-016-020-016-022
014-015-020-017-013-017-015-017-020-020-017-020-016-015-017-016-016-016-017-020-020-015-016-016-015-020-020-016-016-014-016-015
020-016-016-016-021-017-020-021-020-017-021-016-020-020-017-017-020-015-021-017-021-020-021-021-020-017-021-021-020-022-022-025
025-022-023-022-021-022-020-023-025-021-023-022-020-021-020-020-022-023-021-024-024-021-021-021-021-020-020-020-023-020-022-025
015-020-016-020-020-020-020-021-022-020-020-017-017-017-015-020-020-020-017-020-021-017-020-020-017-017-020-020-020-021-017-021
022-021-020-020-020-020-020-020-021-020-020-020-020-016-020-017-016-017-017-017-017-017-020-020-020-020-021-020-020-020-021-020
022-021-020-020-020-017-020-020-020-017-020-017-020-017-020-016-020-020-017-016-016-020-020-020-021-020-020-021-020-021-024-023
020-022-020-021-020-020-020-021-020-020-020-020-016-015-015-014-020-017-016-016-015-015-015-020-020-020-020-020-020-020-016-015
020-020-020-016-017-020-016-020-016-016-016-017-015-016-017-020-016-014-016-017-016-015-016-015-017-016-017-015-016-020-017-017
016-020-020-020-020-020-020-017-021-017-015-017-017-016-017-017-020-016-017-015-016-016-015-020-020-020-017-020-020-021-021-022
020-020-023-023-022-017-020-020-020-021-020-020-016-020-016-016-020-020-020-020-016-015-015-016-020-020-020-017-021-020-020-021
021-022-020-021-020-015-017-016-016-016-020-020-016-020-020-020-020-020-020-017-017-016-014-016-020-016-021-017-020-021-021-020
021-020-020-020-021-017-017-016-017-020-017-017-020-020-023-021-020-016-017-016-020-017-020-015-016-020-021-021-021-021-021-022
021-020-020-020-021-017-017-017-020-021-022-020-021-020-017-020-016-020-017-015-017-017-016-016-017-021-022-020-021-020-017-022
022-022-020-020-020-017-020-016-017-020-021-016-020-020-016-017-017-017-020-020-020-015-015-015-020-021-020-017-017-020-020-021
021-017-020-022-020-020-016-015-017-016-020-020-017-017-016-020-017-017-017-015-016-016-016-016-017-021-020-020-016-017-020-020
020-016-020-016-017-020-020-016-016-020-020-021-020-020-021-021-020-020-020-020-016-020-020-021-020-016-020-016-017-021-021-017
020-015-020-021-021-020-020-020-016-016-015-017-021-020-020-020-016-017-021-014-015-014-016-020-017-016-020-016-020-021-020-017
011-015-017-017-017-020-016-016-015-016-020-016-021-020-021-016-020-017-020-020-016-016-020-016-016-016-020-016-021-021-021-020
015-016-017-020-020-020-017-020-016-016-017-021-017-021-021-020-017-021-020-020-020-020-015-021-020-020-016-016-021-020-021-020
014-016-017-020-020-020-021-017-017-016-021-017-020-016-017-016-017-021-021-021-016-020-020-020-021-016-020-020-021-021-021-020
020-021-017-020-020-016-020-017-017-016-017-020-017-016-020-017-017-020-020-020-016-017-020-020-020-016-020-017-016-020-020-017
012-015-017-017-020-020-020-020-020-020-021-020-020-020-020-020-020-021-021-020-021-021-021-020-021-021-020-020-021-021-022-022
020-021-016-020-017-017-020-020-020-021-021-021-020-017-020-017-016-020-020-020-020-017-016-017-021-020-020-016-017-020-017-020
014-015-013-015-017-017-016-016-017-017-020-021-016-020-020-017-020-020-017-021-021-020-017-016-017-020-016-016-017-017-020-020
010-004-012-013-015-013-014-015-016-016-015-014-017-015-015-015-015-014-015-016-016-014-013-015-013-015-013-014-015-016-016-014
012-011-016-015-020-020-020-020-017-015-015-020-016-017-020-020-016-017-016-017-017-017-015-014-014-015-016-016-020-020-017-020
010-012-010-014-015-016-016-016-014-016-014-014-016-017-011-014-014-014-012-015-012-013-011-013-014-011-013-013-015-017-014-010
012-012-011-007-010-014-015-013-013-014-012-014-011-013-015-011-013-012-014-012-013-011-012-013-012-014-016-011-014-013-011-012
000-014-012-020-021-020-016-014-014-016-014-016-016-014-014-020-014-012-014-011-012-015-007-015-015-015-012-014-012-012-011-006

Figure 5.11: Same as Figure 5.10 except pressure = 0.137 MPa.

002040

002340_031

003260

030-026-030-023-025-027-027-027-030-030-031-030-030-030-030-030-027-030-030-030-031-031-032-032-033-033-031-032-031-033-032-025

037-034-030-030-030-032-032-030-032-034-034-031-031-032-032-031-031-032-031-033-031-031-032-032-032-030-033-031-030-034-033-037

037-034-032-033-031-032-031-031-032-033-032-032-033-032-031-031-031-032-032-033-032-032-032-033-032-032-031-031-032-032-034-033

031-032-033-030-031-032-031-030-032-032-031-032-032-027-031-031-031-031-031-032-032-030-030-032-030-033-031-032-031-030-027-033

032-030-033-032-031-030-032-033-032-033-031-031-032-030-031-030-032-032-034-032-034-031-033-033-033-033-033-034-034-034-033-034

032-032-035-031-031-034-031-033-032-032-032-032-031-030-032-030-032-033-032-033-033-032-032-032-033-033-031-033-033-032-032-030

026-031-030-030-032-032-031-031-033-031-032-032-031-031-027-032-031-033-031-033-032-032-031-032-032-032-032-032-031-030-032-032

033-031-032-032-034-032-033-034-031-032-031-032-033-032-032-031-032-032-031-031-031-032-032-031-032-033-031-031-030-032-033-034

033-033-033-033-032-031-030-032-032-032-030-032-030-031-032-031-032-033-032-032-031-032-032-032-033-032-032-032-031-032-034-031

030-033-030-031-030-031-031-032-031-030-030-030-030-027-030-026-030-031-030-030-030-030-031-032-030-031-032-031-032-030-030-026

034-030-030-033-032-031-030-032-030-030-031-031-031-031-032-033-031-032-033-032-032-031-030-032-032-032-032-032-032-033-031-032

030-034-031-034-030-032-032-031-033-031-031-031-031-030-031-032-031-031-032-032-031-030-031-033-033-031-032-031-033-033-034-032

031-031-033-034-031-031-031-031-031-031-030-032-030-031-027-031-031-031-031-030-030-030-027-027-030-030-030-030-030-030-032-030

027-031-031-033-030-031-030-031-030-030-030-030-030-031-032-031-030-032-032-030-032-030-027-031-031-030-033-030-032-034-032-032

027-032-032-032-031-030-030-030-030-030-031-030-033-031-032-032-032-031-031-031-032-030-030-027-030-032-031-032-034-033-034-033

030-031-031-032-032-031-027-027-032-032-033-031-032-032-031-031-031-031-031-027-031-031-030-030-030-031-031-032-032-030-031-032

026-031-032-030-030-031-030-030-031-031-031-031-032-031-030-031-030-032-030-031-031-030-030-030-032-032-032-031-032-033-031-032

034-032-031-032-033-033-031-031-032-032-032-033-032-032-032-032-031-032-032-030-031-032-030-031-032-031-032-032-031-032-033-034

026-030-026-024-027-030-030-026-030-030-031-032-031-031-031-030-031-031-030-027-030-030-030-031-031-031-031-030-030-030-032-031

031-027-032-030-032-032-033-032-031-031-031-030-033-032-032-032-031-032-032-030-030-030-031-031-032-031-032-032-032-034-032-031

033-026-031-032-032-032-030-032-030-031-032-031-033-033-033-032-033-032-033-032-033-032-033-032-031-032-033-033-033-035-034-034

033-032-032-032-032-032-031-033-032-031-031-033-033-033-033-032-031-033-033-033-032-031-030-033-031-031-032-031-033-032-034-034

031-031-030-030-032-031-032-032-031-031-033-032-033-031-032-030-032-033-033-033-031-031-032-033-032-031-032-032-033-033-033-031

025-031-031-031-032-027-031-032-030-031-031-033-032-030-031-032-031-032-032-032-032-032-032-032-033-031-032-032-030-031-033-032

025-033-032-033-032-032-033-031-032-033-033-033-032-033-032-032-033-034-034-033-032-035-032-033-034-034-033-032-034-033-035-035

032-030-030-031-027-031-030-033-031-031-031-033-031-030-032-033-031-032-031-031-032-031-031-031-033-031-031-030-031-032-030-030

023-025-031-027-031-031-031-031-031-032-033-032-031-031-032-031-032-032-031-031-031-032-031-031-031-031-030-030-031-031-031-034

030-030-032-031-032-031-032-032-032-033-031-031-032-031-031-031-031-031-031-032-031-030-032-031-027-030-027-030-030-031-030-030

025-026-030-025-026-030-031-032-032-031-032-032-033-031-032-034-030-031-030-031-032-032-030-031-030-030-030-031-031-031-032-031

030-032-024-031-030-030-030-030-027-032-032-026-030-030-026-027-027-027-025-027-027-026-026-031-027-026-026-026-027-032-027-024

025-025-026-025-024-026-032-026-026-031-027-030-030-027-032-030-031-027-030-027-033-033-030-032-032-034-035-030-034-031-034-034

037-031-025-034-030-034-032-031-031-033-031-027-032-031-030-031-031-030-030-031-025-033-030-026-033-026-025-037-032-032-033-023

Figure 5.12: Same as Figure 5.10 except pressure = 0.276 MPa.

```
 002040
> 002340_036
> 003260
037-037-037-037-037-037-035-037-037-036-035-037-037-037-037-037-037-036-037-037-037-037-037-035-037-037-037-037-037-037-037-012
037-037-037-035-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-036-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-036-037-037-037-036-037-037-037-037-036-037-037-037-037-037-037-035
037-037-037-037-037-037-036-037-036-036-037-036-037-037-037-037-037-037-037-037-037-036-036-037-037-036-037-036-037-036-036-036
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
035-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-036
037-036-037-037-037-037-037-036-036-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-036
032-037-036-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-036-037-037-037-037-037-037-037-037-037-037-037-037-035-037-037-037-037-037-037-035
037-032-037-037-035-037-037-037-037-037-037-037-035-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037
037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037-037

>
```

Figure 5.13: Same as Figure 5.10 except pressure = 0.414 MPa.

```
001370
002040
> 003[illegible]
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-002-000-000-002-001-000-000-000-000-001-001-000-000-001-001-000-000-000-001-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-001-001-001-001-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-001-000-000-001-001-000-000-001-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-001-000-001-000-000-000-001-000-001-000-001-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-003-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-001-001-001-001-000-001-000-000-001-000-002-003-003-002-000-000-000-001-001-000-000-001-001-001-000-000-000-000
000-000-000-000-000-000-000-001-000-000-000-000-000-001-003-016-004-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-002-005-026-014-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-002-007-002-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000

>
```

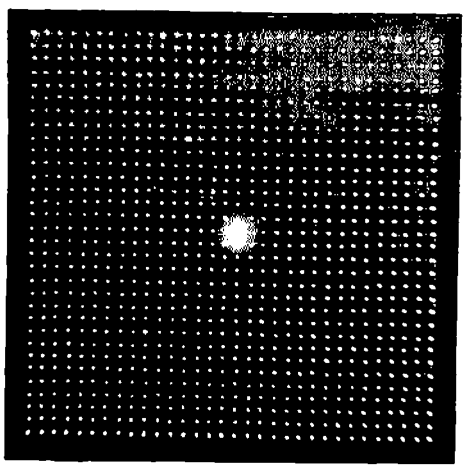


Figure 5.14: Top: imprint of a pinpoint after correction with COR4. Bottom: appearance of imprint on display array.

```
001370003260
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-001-003-000-003-001-001-000-001-001-003-001-003-002-002-001-001-001-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-001-000-000-000-000-000-001-001-001-000-001-001-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-002-001-001-001-002-001-001-001-001-001-004-001-003-003-000-001-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-001-000-001-000-001-001-001-003-002-002-003-003-001-001-000-000-000-000-000-000-000-000
000-000-000-000-002-000-000-002-001-002-001-002-003-002-002-003-005-003-003-003-003-002-001-001-001-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-003-005-006-006-006-003-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-001-000-000-001-000-002-000-001-002-005-012-017-022-016-007-004-001-001-000-001-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-001-003-003-006-011-021-024-023-023-015-006-002-002-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-002-001-001-002-005-007-011-020-023-015-014-017-021-014-005-003-001-000-000-000-000-000-000
000-000-000-000-000-000-001-001-001-001-002-004-006-013-020-023-015-006-005-011-016-022-014-006-002-000-000-000-000-000-000
000-001-000-001-000-000-001-001-002-003-010-010-012-022-026-020-010-006-004-005-013-022-024-013-004-002-002-001-000-000-000-000
000-000-000-001-000-001-003-002-003-006-010-014-023-026-020-011-005-003-003-005-012-022-030-015-004-001-001-000-000-000-000-000
000-000-000-000-001-001-001-002-004-007-014-021-024-020-007-004-003-003-003-006-015-024-022-010-003-002-000-000-000-000-000-000
000-000-000-000-001-001-001-004-006-014-022-022-015-010-005-004-003-004-007-015-022-024-012-004-003-002-002-001-000-000-000-000
003-000-000-000-000-002-001-005-013-021-025-015-007-004-002-004-004-007-016-024-022-012-006-002-001-001-000-001-000-000-000-000
000-000-000-000-000-000-001-005-020-021-016-010-005-002-001-002-005-014-021-017-007-003-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-003-011-023-020-010-005-004-003-003-006-013-022-022-010-004-002-000-000-000-000-000-000-001-000-000-000
000-000-000-000-000-002-001-010-017-022-015-007-005-002-010-013-023-020-011-005-002-001-001-000-000-000-000-000-000-000-000-000
000-000-000-000-001-002-003-005-011-020-022-013-010-010-021-023-021-011-004-002-001-001-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-001-003-006-012-025-023-017-017-024-020-011-005-003-001-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-001-001-003-004-010-015-024-024-026-023-013-006-002-002-001-000-001-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-001-000-000-001-003-006-010-020-021-010-006-002-002-001-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-002-003-006-002-003-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
```

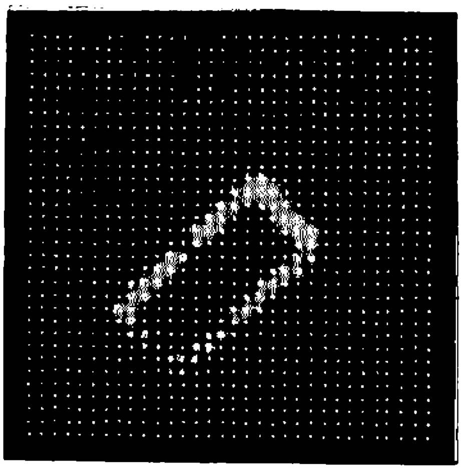


Figure 5.15: Top: imprint of a hollow, rectangular plastic block after application of correction function COR4. Bottom: appearance of imprint on display array.

```
003260
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-001-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-001-003-003-004-002-002-001-001-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-001-005-010-011-014-012-013-007-006-001-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-002-005-013-016-020-017-020-020-015-014-011-005-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-002-006-013-021-021-024-024-023-021-021-020-015-011-002-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-003-011-016-022-025-024-027-024-025-022-021-020-015-007-001-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-002-005-014-023-023-026-025-025-026-023-024-023-020-017-010-002-001-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-002-006-014-023-024-026-026-025-026-024-024-023-022-016-011-002-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-001-004-015-022-024-025-025-025-024-023-023-021-016-015-010-002-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-001-005-012-021-025-023-025-025-025-024-024-022-015-013-010-002-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-003-006-013-021-024-023-024-024-022-021-021-014-011-004-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-001-005-013-015-020-022-023-020-022-020-017-015-007-002-001-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-001-007-013-015-020-021-016-017-020-015-012-005-002-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-002-006-011-013-014-014-015-015-012-006-002-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-001-003-004-006-011-011-007-007-005-002-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-001-002-003-002-002-001-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
```

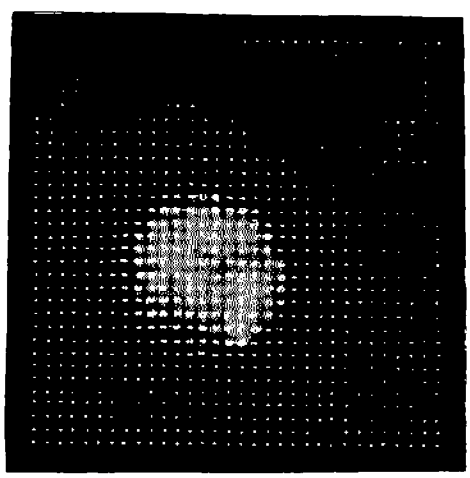


Figure 5.16: Top: imprint of the end of a 12 mm diameter cylinder. Correction function COR4 was applied to the data. Bottom: appearance of imprint on display array.

v03260

```
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
004-005-002-002-000-000-000-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
015-012-006-004-002-002-002-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
026-022-016-010-004-002-001-000-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
024-026-025-020-012-005-005-002-001-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
015-020-026-025-025-020-013-012-007-005-005-005-000-000-000-000-000-000-002-002-000-000-000-000-000-000-000-000-000-000-000
004-010-013-020-022-023-024-016-015-011-004-003-002-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-003-006-012-015-021-021-021-016-015-010-005-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-002-001-005-007-012-020-020-021-023-012-006-007-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-002-000-003-001-007-011-013-020-027-020-020-011-005-004-002-001-001-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-001-004-006-010-014-024-025-021-014-007-005-003-002-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-003-004-006-014-025-027-027-024-021-014-006-004-002-002-000-001-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-001-004-010-016-024-030-033-027-022-011-011-003-002-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-001-004-005-012-021-033-034-032-026-015-013-006-003-002-001-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-004-002-003-005-013-021-030-033-030-024-020-010-007-003-002-001-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-002-004-007-013-020-026-034-034-025-021-012-010-004-002-003
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-002-004-004-011-021-030-035-031-022-016-010-005-001
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-002-006-014-022-030-031-030-023-017-010
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-002-004-007-016-020-026-031-026-017
000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000-003-000-000-000-000-001-001-002-003-007-015-021-022-021
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-001-004-007-011-012
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-003-001-001-000-000-001-004-005-004-005
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-001-003-001-001
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-002-001-001-001-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
```

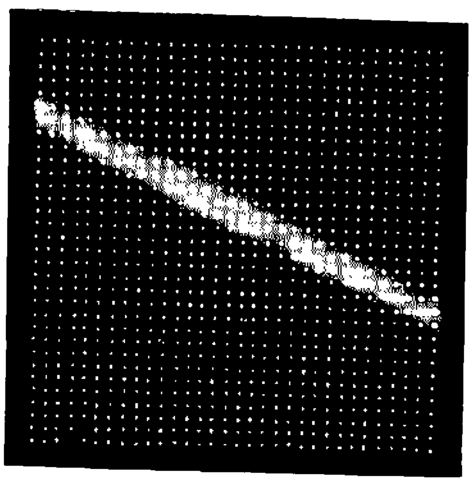


Figure 5.17: Top: imprint of a long cylindrical rod 7 mm in diameter. Correction function COR4 was applied to the data. Bottom: appearance of imprint on display array.

003260

```
037-026-025-026-025-027-021-021-023-024-021-014-010-004-001-000-001-001-003-006-006-013-016-016-020-020-020-016-020-011-005-000
030-024-025-023-022-026-024-021-020-020-014-006-003-000-000-000-000-003-001-002-005-011-015-017-023-025-025-025-020-012-004-000
020-025-022-023-020-022-021-016-015-013-003-001-001-000-000-000-000-001-003-001-001-004-010-015-022-027-031-031-026-022-012-004
031-025-031-022-020-022-017-015-012-005-001-001-000-000-000-000-000-000-000-000-001-001-003-011-022-031-035-035-031-026-016-011
025-030-025-024-021-017-015-012-005-001-000-000-000-000-000-000-000-000-000-000-000-000-002-010-017-025-035-036-037-032-024-016
032-030-030-024-023-022-015-012-005-001-001-001-000-000-000-000-000-001-000-002-003-001-002-005-011-023-031-037-036-032-022-016
022-024-022-022-020-016-013-004-003-000-000-000-000-000-000-000-000-000-000-000-001-000-001-002-007-016-025-027-031-027-020-013
026-025-024-024-022-016-011-006-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-003-012-023-024-022-022-016-006
022-025-022-022-016-013-007-000-000-000-000-000-000-000-001-000-000-000-000-000-000-000-000-000-003-011-020-023-020-021-020-003
020-022-020-016-016-014-006-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-002-007-017-021-020-016-014-004
020-020-020-016-013-013-006-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-005-013-014-020-020-012-007
016-020-020-020-013-012-006-001-001-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-002-007-014-017-020-020-016-012
020-020-022-021-015-013-006-003-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-003-012-016-017-020-015-015-012
021-020-020-020-016-011-007-004-001-001-000-000-000-000-000-000-000-000-000-000-000-000-000-006-011-017-020-017-017-014-012
016-020-017-017-014-012-011-005-003-002-000-000-000-000-000-000-000-000-000-000-001-002-010-014-017-020-016-015-012-007
021-017-017-020-017-014-012-007-007-005-003-000-001-000-000-000-000-000-000-000-001-004-010-016-020-022-020-017-013-012-012
011-017-017-016-016-013-012-011-010-006-005-002-002-001-000-000-000-000-000-001-001-003-006-014-022-022-021-017-015-013-012-011
016-015-014-015-015-015-013-010-013-011-012-007-004-003-000-001-000-001-002-002-004-010-016-021-026-024-024-022-016-014-013-010
014-012-013-011-013-014-014-011-013-013-014-014-011-010-006-005-005-005-006-011-014-021-024-031-031-030-026-021-017-016-013-011
011-010-013-013-013-014-014-014-013-013-012-014-020-015-016-014-014-015-021-017-021-022-026-031-030-031-026-022-020-016-014-011
004-007-011-012-013-014-010-013-010-013-016-015-020-021-023-020-022-023-024-026-024-026-030-032-031-030-027-022-020-017-013-012
000-010-010-012-015-014-012-014-013-011-015-021-021-024-026-025-025-031-032-032-032-032-031-035-034-031-024-022-021-016-015-010
000-006-011-012-013-014-014-015-013-011-016-017-021-021-024-024-030-033-032-035-030-034-034-034-032-027-025-023-021-016-014-011
000-010-010-014-015-013-015-015-015-013-015-020-020-022-025-026-030-032-031-033-033-033-035-033-033-026-024-020-016-015-013-010
000-004-007-012-015-015-015-015-015-014-017-016-017-022-024-027-030-033-033-035-034-037-035-033-032-030-022-020-016-013-010-007
000-000-006-012-014-015-015-016-017-021-021-021-020-021-025-030-030-033-034-036-037-037-036-032-032-025-024-016-013-010-003-000
000-000-000-003-006-011-011-012-013-015-016-020-020-021-025-026-027-033-032-036-035-035-032-031-026-021-016-010-004-001-000-000
000-000-000-000-000-003-005-010-010-016-015-020-023-024-026-031-031-033-035-036-036-034-034-032-022-017-011-004-000-000-000-000
000-000-000-000-000-000-000-001-006-010-013-017-020-025-030-032-030-034-033-032-036-034-032-021-020-007-001-000-000-000-000-000
000-000-000-000-000-000-000-000-000-002-004-010-014-020-015-023-021-023-021-024-025-020-014-011-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-002-005-010-007-011-006-010-010-006-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000-000
```

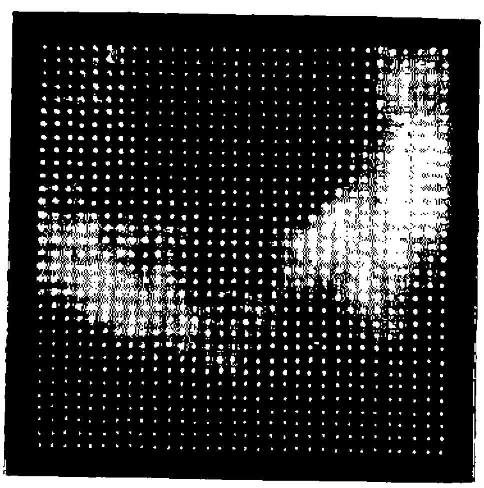


Figure 5.18: Top: imprint of a large washer (45 mm OD, 20 mm ID). Correction function COR4 was applied to the data. Bottom: appearance of imprint on display array.

003260

000-000-000-000-000-000-000-000-000-000-000-000-000-000-001-001-000-001-001-001-000-000-000-000-000-000-000-000-000-000-000
000-000-000-002-000-000-000-000-001-000-000-000-001-001-003-004-007-007-005-004-003-002-000-001-000-001-000-000-000-000-000-000
000-000-000-002-002-000-002-002-001-001-001-002-002-006-006-011-020-017-010-005-002-002-003-001-001-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-001-002-001-002-001-003-007-013-025-023-013-006-002-001-000-002-000-000-000-000-000-000-000-000
000-000-000-000-000-000-001-000-001-001-003-002-002-002-007-017-030-024-014-007-004-001-001-001-002-000-001-001-001-001-000-000
000-000-002-002-002-000-000-004-001-004-005-006-005-005-012-021-026-027-020-014-010-004-001-002-001-001-001-002-000-001-001-002
000-000-000-000-003-000-001-002-001-001-000-001-002-003-007-020-022-020-016-010-006-002-001-000-000-000-000-000-000-000-000-000
000-000-000-000-002-001-000-000-001-000-000-002-002-002-010-023-014-016-015-006-004-002-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-001-003-006-015-024-014-013-020-006-003-002-001-000-000-001-000-000-001-001-001-002
000-000-000-002-001-001-000-001-000-001-000-002-003-005-011-020-011-010-014-011-004-001-001-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-003-005-014-016-006-004-012-012-004-000-000-000-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-001-004-006-016-020-010-006-012-014-005-002-002-001-000-001-001-000-001-000-000-000
000-000-000-000-000-000-000-000-000-000-000-001-002-005-016-016-006-006-010-014-006-003-002-001-000-000-000-001-001-000-000-000
000-000-000-000-000-002-001-000-000-000-000-001-003-007-022-015-005-004-006-012-007-002-001-002-000-000-000-000-000-000-000-000
000-000-000-000-000-001-000-000-001-000-000-003-005-007-022-012-005-004-004-011-011-006-001-001-000-000-001-001-000-002-000-000
000-001-001-000-000-000-000-001-000-001-001-003-005-011-023-011-006-004-004-007-014-007-002-002-002-002-001-001-001-001-000-001
000-001-000-001-000-000-000-000-001-000-000-002-005-011-017-007-003-002-002-005-014-006-002-000-000-000-000-000-000-000-000-000
000-001-000-000-000-000-001-000-001-001-001-004-006-014-017-010-004-002-002-005-012-013-005-002-001-001-000-001-000-000-000-000
000-000-000-000-000-000-001-000-001-001-001-004-006-020-016-006-003-002-003-004-010-020-007-003-002-000-001-000-001-001-001-000
000-000-000-000-000-000-000-000-000-001-003-007-010-020-014-005-003-002-005-002-005-016-011-004-000-000-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-001-004-011-023-013-004-003-002-002-003-004-013-017-006-002-000-000-000-000-000-000-000
000-000-000-000-[illegible]-000-000-000-001-001-003-005-013-024-011-005-004-004-004-003-005-014-024-013-005-003-001-001-001-000-000-001
000-000-001-000-000-000-000-000-001-000-003-007-014-024-011-003-003-002-002-004-004-012-026-016-010-003-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-001-003-007-016-027-007-003-002-001-001-003-003-010-026-021-010-004-002-000-000-000-000-000
000-000-000-000-000-000-000-000-000-001-003-006-022-021-006-002-001-[illegible]-001-002-004-010-022-032-015-005-003-001-000-001-001-001
000-001-003-000-001-000-001-001-002-003-005-012-023-017-010-005-003-002-004-003-004-010-021-[illegible]-014-006-004-003-001-000-000-001
000-[illegible]-000-000-000-000-000-000-001-005-010-024-011-004-002-000-001-001-001-003-006-013-027-012-003-003-000-000-000-000-000
000-[illegible]-000-000-000-000-000-001-003-006-013-005-002-001-001-001-001-001-001-003-011-014-006-003-001-000-000-000-001-000
000-000-000-000-000-000-000-[illegible]-001-002-003-005-002-003-002-001-002-002-002-002-002-003-004-004-004-002-001-000-000-001-001
000-002-000-000-000-001-001-000-[illegible]-002-002-003-003-001-001-002-002-001-002-000-003-002-002-003-001-001-000-000-001-001
004-000-000-000-000-000-000-000-000-000-000-000-000-000-[illegible]-000-000-000-000-000-000-000-001-001-001-000-000-000-000-000-000
000-000-000-000-000-000-000-000-000-000-000-000-001-000-002-001-000-000-001-[illegible]-004-000-003-003-003-002-001-000-000-001-000

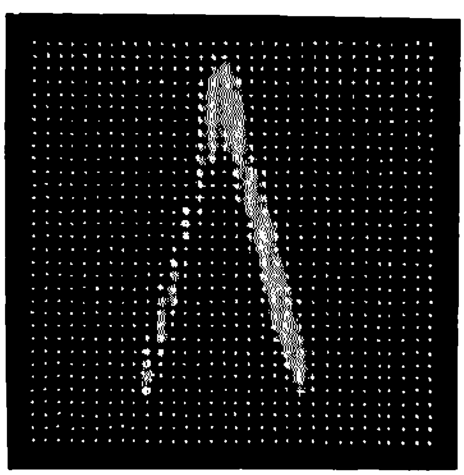


Figure 5.19: Top: the spatial resolution of the TIR-TSA was assessed with a knife-edge wedge 25 mm long and 10 mm wide at the base. The minimum lateral spatial resolution was approximately 2 mm. Correction function COR4 was applied to the data. Bottom: appearance of imprint on display array.

> **003500(CR)_H000000L000406** (TSMI: taxel sum.)

The taxel sum is returned as a double-precision value (i.e., 32 bits). In general, this value is not zero for a zero-pressure imprint due to noise and variation in individual taxel response. The above result may be compared to that obtained when a ring is pressed against the sensor:

> **001370(CR)** (GETI: get imprint of a ring.)

> **002040(CR)** (COR4: correction function 4.)

> **003500(CR)_H000000L007763** (TSMI: taxel sum.)

The actual force that this value corresponds to would need to be computed by the host, given the particular pressure response characteristics of the tactile sensor.

Slip may be detected only under certain conditions with the current generation of TSAs. They are typically sensitive only to normal forces and are unresponsive to shear forces, and thus cannot detect incipient slip. Only slip that involves a temporal change in some tactile feature is considered to be detectable. Examples of situations that cannot be sensed are the sliding a long, thin cylindrical rod in the direction of its axis, or the movement of a large, smooth, planar surface over the sensor.

To detect slip with the TIR-TSS, a "reference" tactile data area has been provided. One may at any time transfer the data from the imprint area to the reference area for the purpose of performing comparisons at later times to detect slippage or possible re-orientation of the object within the gripper. The command GCIR(001470) performs the copying function. For example:

> **003520(CR)_H000000L000000** (TSMR: taxel sum in reference.)

computes the taxel sum for the reference imprint. The value is zero since nothing has been written to the reference area since the system was turned on:

> **001470(CR)** (GCIR: copy last imprint into the)
(reference area.)

> **003520(CR)_H000000L007763** (TSMR: sum of taxels in reference.)

Unsurprisingly, the result is the same as the taxel sum for the current imprint.

Two methods are used to compute the change in force detected by the gripper. The first simply computes the difference between the taxel sums for the imprint and the reference imprint, and the second computes the sum of the absolute values of the taxel differences between the imprint and reference imprint. These procedures are executed by the commands TSCG(003540) and TSAB(003660), respectively. The primary advantage of TSAB is that it is orientation and position sensitive, whereas TSCG is not. The following example illustrates this distinction:

```
> 001370(CR)   (GETI: get an imprint of a )
                (rectangular block oriented )
                (with its center at the     )
                (center of the array, and   )
                (its major axis aligned     )
                (with the Y axis of the     )
                (array.                     )

> 002040(CR)   (COR4: correction function 4)

> 003500(CR)_H000000L006556   (TSMI: taxel sum.)

> 001470(CR)   (GCIR: copy imprint to      )
                (reference area.            )

> 003540(CR)_H000000L000000   (TSCG)

> 003660(CR)_H000000L000000   (TSAB)

> 001370(CR)   (GETI: take another imprint )
                (of the rectangular block   )
                (with the same total force  )
                (applied, but oriented at 45 )
                (degrees with respect to the )
                (y axis of the array.       )

> 002040(CR)   (COR4: correction function 4)

> 003500(CR)_H000000L007103   (TSMI: taxel sum.)

> 003540(CR)_H000000L000315   (TSCG)

> 003660(CR)_H000000L007363   (TSAB)
```

Notice that TSAB is sensitive to orientation, whereas TSCG is not.

```
> 001370(CR)   (GETI: take another imprint of the  )
               (rectangular block with the same total)
               (applied force, but orient the major  )
               (axis to be parallel to the y axis and)
               (the center to be located 8 taxels to )
               (the left or right of the center of   )
               (the array.                           )

> 002040(CR)   (COR4: apply correction function 4.    )

> 003500(CR)_H000000L006342   (TSMI: taxel sum.)

> 003540(CR)_H177777L177536   (TSCG)

> 003660(CR)_H000000L012016   (TSAB)
```

Notice that in this case that TSAB is also sensitive to location, but TSCG is not. In summary, either TSCG and TSAB may be used as indicators or slip or force changes within a robot gripper if only changes in the total contact force are needed. TSCG is computationally faster in this regard (see Section 5.3), but TSAB is more general in its detection capabilities and can signal changes in position and orientation in the absence of a change in the total force.

The first moments about the x and y axes (centered in the middle of the array: see Figure 5.4) may be calculated by the function FMOI(004470). If the taxel values are interpreted as forces normal to the surface of the sensor then the first moment physically corresponds to the torques acting upon the tactile sensor. (This may be contrasted to the analogous situation with visual images in which there is no correspondence between the visual first moment of an object and any physical property of that object.) Again, as with the taxel sum information, the double-precision results returned by FMOI are only linearly related to the true first moment, and it is the host computers responsibility to use the sensor characteristics (e.g., the taxel spacing and pressure response characteristics shown in Figure 5.4) to compute the true value. For many applications, however, a knowledge of the relative rather than the absolute value of the force or first moment values would be sufficient for effective robot control.

The first moments about the X and Y axes resulting from contact with a small object are calculated below:

> **001370(CR)** (GETI: get imprint of object at position [17,37])

> **002040(CR)** (COR4)

> **004470(CR)**_XH177777L177746_YH000000L167370 (FMOI)

> **001370(CR)** (GETI: get imprint of object at position [0,17])

> **002040(CR)** (COR4)

> **004470(CR)**_XH177776L045632_YH000000L001154 (FMOI)

> **001370(CR)** (GETI: get imprint of object at position [0,37])

> **002040(CR)** (COR4)

> **004470(CR)**_XH177776L170600_YH000001L002764 (FMOI)

> **001370(CR)** (GETI: get imprint of object at position [20,20])

> **002040(CR)** (COR4)

> **004470(CR)**_XH177777L167446_YH000000L013344 (FMOI)

Changes in the first moments may also be determined by computing the first moment in the reference imprint using the function FMOR(004520), and then computing the change between the imprint and reference imprint first moment using the function FMCG(004550):

> **001470(CR)** (GCIR: copy last imprint into the reference area)

> **004520(CR)**_XH177777L167466_YH000000L013344 (FMOR)

> **001370(CR)** (GETI: acquire an imprint from location [0,37])

> **002040(CR)** (COR4)

> **004470(CR)**_XH177777L067122_YH000000L072356 (FMOI)

> **004550(CR)**_XH177777L077424_YH000000L057102 (FMCG)

The last data processing function that is performed by the TSS is continuous threshold monitoring. The core routine is THRE(005020) and requires two arguments. The first is the code number for the correction function to be applied to each new imprint. The second is the code for the particular threshold monitor function that is to be implemented. The latter includes subroutines that monitor each taxel value, the taxel sum, the change in the taxel sum, the sum of absolute taxel differences, the first moments, and the change in the first moments. Important applications of these routines include: monitoring the taxel values during object grasping and to reporting any sharp protrusions which might cause damage to the sensor before a wrist force sensor could be alerted; slip of the object within the gripper; and collisions between the object and other elements of the environment. The user may set the threshold values by using IWRD and OWRD. The memory locations and the default values of the thresholds are established in the procedure SDEF, which is listed in Appendix A. Examples of the threshold monitor subroutines are shown below:

```
> 005020(CR)C     (THRE: core routine for threshold)
                  (monitoring.  Requests input of  )
                  (the desired correction sub-     )
                  (routine code.                   )

> 001530(CR)T     (COR1: requests input of the     )
                  (desired threshold monitor subr. )

> 005310(CR)      (TTCH: monitor taxel values.)
```

At this point, the subroutine THRE enters a continuous cycle of imprint acquisition, correction, and monitoring of each taxel value. There is no output until a taxel is detected that is below high range limit (memory location 000402) or greater than or equal the low range limit (memory location 000400). Those taxels within the detection band range are then identified along with their value. In the example that follows, the H and L range limits are the default values 34 and 30, respectively:

```
N000732_V000031   (The only taxel in the range  )
                  ([30..34] is number 732 with a )
                  (value of 31 (octal).          )
```

```
N00660V_V000031    (Three taxes were found in the)
N000717_V000033    (next imprint in the monitored )
N001020_V000030    (range.                        )
```

The threshold monitoring cycle is stopped with the following termination sequence:

```
control-S               (Disable I/O before next imprint.)
control-Y               (Terminate the process.          )
TEM> control-Q          (Enable terminal I/O.            )
TEM> GO(CR)             (Restart the program.            )
>                       (Enter the next command code.    )
```

TSCK(005450) is used to continuously monitor the taxel sum. The threshold values are stored at memory location 000410-000417, and the H and L default values are H000000L015000 and H000000L005000, respectively. For example:

```
> 005020(CR)C     (THRE)

> 002040(CR)T     (Select correction function.   )

> 005450(CR)      (Monitor taxel sum using default)
                  (threshold values.              )

L_H000000L001004  (The "Low" threshold indicator)
L_H000000L001147  (and the taxel sum will be    )
L_H000000L001235  (outputted until the taxel sum )
L_H000000L002452  (becomes in range.             )
L_H000000L004006

H_H000000L015632  (While in range, there is no  )
H_H000000L017243  (output. The "High" threshold)
                  (flag is displayed along with )
                  (taxel sum value until the    )
                  (taxel sum comes into the     )
                  (allowed range.               )

control-S         (Terminate threshold monitoring.)
control-Y
TEM> control-Q
TEM> GO(CR)
>
```

A similar procedure is implemented for monitoring the taxel-sum change and the absolute-taxel-sum using TSCC(005510) and TSAT(005560), respectively.

FMCK(005620) is used to monitor the first moments about the X and Y axes of the tactile sensor. For example, consider the following default threshold values for the first moments:

Axis	H/L	Value	Mem. loc.
X	H	H000000L020000	000446,000444
X	L	H177777L160000	000442,000440
Y	H	H000000L020000	000456,000454
Y	L	H177777L160000	000452,000450

An example of the first moment monitor function using these threshold values is shown below:

```
> 005020(CR)C          (THRE)
> 002040(CR)T          (Select COR4)
> 005620(CR)           (Select FMCK)
                       (There is no output    )
                       (until a threshold is  )
                       (violated.             )
H_H000000L020323_X     (In this imprint, the H  )
                       (threshold for the first )
                       (moment about the X axis)
                       (was exceeded.           )
H_H000000L025102_X     (In this imprint, the H )
H_H000000L002456_Y     (thresholds for both the)
                       (X and Y axes were     )
                       (exceeded.             )
H_H000000L025667_X
L_H177777L146526_Y

control-S              (Termination sequence.  )
control-Y
TEM> control-Q
TEM> GO(CR)
>
```

A similar procedure is used to monitor the change in first moments using the function FMCC(005720).

Conclusions

A tactile sensing system has been described which addresses several problems regarding the creation and control of sensor-based robots. First, the TSS was shown to provide a simple means of access to tactile data by researchers in the field of robotics. The TSS is a complete system in which communication with the robot-control computer relied only upon simple commands sent over a serial line.

Secondly, the TSS consisted of modular, interchangeable components. Any TSA module could be used provided it met the simple communication protocol requirements. Therefore, TSAs could be selected that best met the need of a particular robotic task, e.g., handling forgings versus assembling watches.

Thirdly, the TSS acquired and preprocessed tactile data, thereby removing a significant computational burden from the host computer. The preprocessor module took responsibility for acquiring tactile data from the TSA module and preprocessed it according to commands issued from the host, e.g., computation of the total force, first moments, detection of changes in parameters, and monitoring various thresholds. These functions are thought to be sufficient for many current robot tasks, e.g., contact and collision detection, force-feedback-directed gripping and assembly, slip detection, and texture characterization.

Lastly, the TSS project facilitated the refinement of a new, optically-based tactile sensor array technology. An advanced, 32 x 32 taxel TIR-TSA was developed with an active area measuring 31 x 31 mm, a taxel density of 100 taxels/sq-cm, a 5-bit force gray-scale, and a force range of 0 - 0.41 N (0 - 42 gm) per taxel. This sensor was demonstrated to be compact, rugged, sensitive to small forces, possess a high spatial resolution, and manufacturable in fingertip-shaped form, and thereby was considered to have satisfied an important need in robotics for such devices.

Experience and perspectives gained during this project suggested the following directions for future research in the area of tactile sensors for robotic applications:

1. Increase the data acquisition rate to "real time" (e.g., greater than or equal to 60 Hz) by switching from an interrupt-driven to a memory-mapped approach. This is currently in the process of being implemented, and a prototype circuit has reached speeds of 55 frames/s. This result is an increase of approximately 300% over the present version of the TSS.

2. Increase the speed of computation within the preprocessor by performing multiplication, division, and array (convolution) operations in hardware.

3. Implement tactile-feature extraction algorithms, e.g., parametric expressions of planar and curved surfaces, edges, vertices, holes and hole segments, and texture measures. Tactile features are expected to play an important role in complex tasks such as tactile identification of an object under visual obscuration, determination of object position and orientation using passive and active touch, acquisition and verification of stable grip configurations, and assembly operations requiring force feedback. Some work in this regard is described by Ellis [6] in which tactile features within LA-TS imprints are extracted with modified visual-image processing algorithms.

4. Develop a compact, fingertip-shaped TIR sensor for inclusion on a complex, 9-DOF articulated hand.

5. Develop a custom light-detector/digital-converter chip where on the detectors are appropriately spaced for tactile sensing (e.g., approximately 1 mm apart), and incorporate the circuit directly into the TIR sensor head. This would eliminate the large numbers of optical fibers currently required to transmit the imprint from the sensor head to the camera, and would also remove the associated optical cable. The logical conclusion of such an approach would be placement of the preprocessor within the sensing head, thereby reducing the number of cables to a bare minimum.

Bibliography

[1] Abramowitz, J.D., J.W. Goodnow, and B. Paul, "Articulated Mechanical Hand: An End Effector to Determine Shape by Touch," in the Proceedings of the 1983 Computers in Engineering Conference, Chicago, IL, vol. 2, pp. 103-108, August 7-11, 1983.

[2] Bejczy, A.K., "Touch Sensor Responds to Contact Pressure," NASA Technical Briefs, pp. 207-208, Summer, 1981.

[3] Boie, R.A., "Capacitive Impedance Readout Tactile Image Sensor," in the International Conference on Robotics, Atlanta, GA, pp. 370-378, March 13-15, 1984.

[4] Briot, M., "The Utilization of an Artificial Skin Sensor for the Identification of Solid Objects," in the 9th International Symposium on Industrial Robotics, pp. 529-548, Washington D.C., March 13-15, 1979.

[5] Dario, P., D. DeRossi, C. Domenici, and R. Francesconi, "Ferroelectric Polymer Tactile Sensors with Anthropomorphic Features," in the International Conference on Robotics, Atlanta, GA, pp. 332-340, March 13-15, 1984.

[6] Ellis, R.E., "Extraction of Tactile Features by Passive and Active Sensing," Proceedings of the SPIE Intelligent Robots and Computer Vision Conference, Cambridge, MA, November 4-8, 1984.

[7] Geschke, C.C., "A Variable Capacitance Touch Sensor," in the Proceedings of the 5th International Conference on Artificial Intelligence, MIT, Cambridge, MA, p. 274, August, 1977.

[8] Haliday, D. and R. Resnick, in **Physics, Part I and II,** John Wiley and Sons, pp. 1024-1028, 1967.

[9] Harmon, L.D. "Automated Tactile Sensing," in The International Journal of Robotics Research, vol. 1, no. 2, pp. 3-32, Summer, 1982.

[10] Harmon, L.D., "Tactile Sensing for Robots," paper presented at the Semimar and Workshop for Robotics and Flexible Automation, University of Rhode Island, Kingston, RI, June 8-9, 1983. (The paper can also be found in **Recent Advances in Robotics,** vol. 1, John Wiley, 1983.)

[11] Harmon, L.D., "Automated Touch Sensing: A Brief Perspective and Several New Approaches," International Conference on Robotics, Atlanta, GA, pp. 326-331, March 13-15, 1984.

[12] Hillis, D.W., "A High-Resolution Imaging Touch Sensor," in The International Journal of Robotics Research, Vol. 1, No. 2, pp. 33-44, Summer, 1982.

[13] Jacobsen, S.C., J.E. Wood, D.F. Knutti, and K.B. Biggers, "The Utah/MIT Dextrous Hand: Work in Progress," in **Robotics Research**, M. Brady and R. Paul (eds), MIT Press, pp. 601-653, 1984.

[14] Larcombe, M.H.E., "Carbon Fibre Tactile Sensors," in the Proceedings of the 1st RoViSeC, pp. 273-276, Stratford-upon-Avon, UK, April 1-3, 1981.

[15] Loomis, J.M., "Tactile Pattern Perception," in Perception, vol. 10, pp. 5-27, 1981.

[16] Overton, K.J. and T.D. Williams, "Tactile Sensing for Robots," in the Proceedings of the 7th International Joint Conference on Artificial Intelligence, Vancouver BC, Canada, pp. 791-795, August 24-28, 1981.

[17] Overton, K.J. and T.D. Williams, "Tactile Sensing for Robots: Overview and Experience," in the Proceedings of the 1983 International Computers in Engineering Conference, Chicago, IL, vol. 2, pp. 115-123, August 7-11, 1983.

[18] Overton, K.J. "The Acquisition, Processing, and Use of Tactile Sensor Data for Robot Control," Computer and Information Science Technical Report 84-08, University of Massachusetts, Amherst, MA, 01003, May, 1984.

[19] Purbrick, J.A., "A Force Transducer Employing Conductive Silicone Rubber," in the Proc. 1st Robot Vision and Sensory Control Conference, Stratford-upon-Avon, UK, April 1-3, 1981.

[20] Raibert, M.H., and J.E. Tanner, "A VLSI Tactile Array Sensor," in the Proceedings of the 12th International Symposium on Industrial Robotics, Paris, France, pp. 417-425, June 1982.

[21] Raibert, M.H., "An All Digital VLSI Tactile Array Sensor," in the International Conference on Robotics, Atlanta, GA, pp. 314-319, March 13-15, 1984.

[22] Rebman, J., and M.W. Trull, "A Robust Tactile Sensor for Robotic Applications," in the Proceedings of the 1983 International Computers in Engineering Conference, Chicago, IL, vol. 2, pp. 109-114, August 7-11, 1983.

[23] Salerno, C.M., "Compliant Tactile Sensor Pad," paper presented at the Seminar and Workshop on Sensors for Robotics and Flexible Automation, University of Rhode Island, Kingston, RI, 8 pages, June 8-9, 1983.

[24] Salisbury, K.J. Jr., "Kinematic and Force Analysis of Articulated Hands," Department of Computer Science, Stanford University, Report No. STAN-CS-82-921, July 1982.

[25] Schneiter, J.L. and T.B. Sheridan, "An Optical Tactile Sensor for Manipulators," in Robotics and Computer-Integrated Manufacturing, vol. 1, no. 1, pp. 65-71, 1984.

[26] Taylor, P.M., G.E. Taylor, D.R. Kemp, J. Stein, and A. Pugh, "Sensory Gripping System: the Software and Hardware Aspects," Sensor Review, pp. 184-187, October, 1981.

Appendix A

Preprocessor Software Listing

The preprocessor software is written in PDP-11 assembly language and is listed below.

```
.START
CLEAR                   ; Clear symbol table.
. = 144                 ; Clear memory [144...6477].
repeat   3156, 0        ;
                        ;
; ******************************************************************************
                        ;
                        ; This program constitutes the software
                        ; portion of the TIR Tactile Sensing
                        ; System.
                        ;
                        ; Author:     Steve Begej,
                        ; Date:       August 17, 1984.
                        ;
                        ; File:       vax2::dra1:[rbtsb.taction]tacsen8.pdp
                        ;
                        ; Notes:
                        ;   - All I/O is done in octal.
                        ;   - All numbers are octal except when
                        ;     followed by a "."
                        ;   - The imprint array is labeled such
                        ;     that the top-left taxel is
                        ;     (row, col) = (0,0), and the
                        ;     bottom-right taxel is (37,37).
                        ;     (The fiber cable end is near (0,20).)
                        ;   - The system should normally be in
                        ;     WORD mode unless heavy use is
                        ;     made of functions that return
                        ;     byte data.
```

```
; ****************************************************************************
;
; DECLARATION OF CONSTANTS:
; ----------------------------
;
; The size of the symbol table (66)
; limits the number of symbols that may
; be conveniently defined.  Therefore,
; a large number of absolute memory
; references must be defined.
;
; SYSTEM ROUTINES:
; ------------------
;
; As a whole, these routines use r0, r1,
; and r3, leaving only r2, r3, and r5
; free for "general" use.
;
; 177542  Port B 8-bit parallel input.
; 177444  LED control line (pin 51 on
;          connector J1).
; 140160  TYPOCT: types six or three
;          digit octal number from R1
;          when in INSTRU/WORD or BYTE
;          modes, respectively. R0 is
;          also used.
; 140140  ASCOUT: ASCII equivalent of
;          low byte in r0.
; 140120  CRLF: performs CR and LF.
;          (Uses r0)
; 140104  GETNXT: get character indexed
;          by r3 from character buffer.
;          Store ASCII value in low byte
;          of r0.
; 140060  GETLIN: Get input line.  Uses
;          r0 and r3.
; 140010  Return to console monitor
;          routine.
;
; IMPRINT DATA STORAGE AREAS (byte):
; --------------------------------------
; 27777  Top of reference imprint.
; 26000  Base address of reference
;         imprint.
; 25777  Top of maximum pressure imprint.
; 24000  Base address of max pressure
;         imprint.
; 23777  Top of zero pressure imprint.
```

```
; 22000  Base of zero pressure imprint.
; 21777  Top of imprint memory block.
; 20000  Base address of imprint block.
;
;
; THRESHOLD VALUES:
; ________________________
; Change in 1st moment:
; 476    Y axis, H threshold, H word.
; 474                         L word.
; 472            L threshold, H word.
; 470                         L word.
; 466    X axis, H threshold, H word.
; 464                         L word.
; 462            L threshold, H word.
; 460                         L word.
;
; First moment:
; 456    Y axis, H threshold, H word.
; 454                         L word.
; 452            L threshold, H word.
; 450                         L word.
; 446    X axis, H threshold, H word.
; 444                         L word.
; 442            L threshold, H word.
; 440                         L word.
;
; Sum of absolute-taxel-differences:
; 436    H threshold, H word.
; 434                 L word.
; 432    L threshold, H word.
; 430                 L word.
;
; Change in taxel sum:
; 426    H threshold, H word.
; 424                 L word.
; 422    L threshold, H word.
; 420                 L word.
;
; Taxel sum:
; 416    H threshold, H word.
; 414                 L word.
; 412    L threshold, H word.
; 410                 L word.
;
; Taxel values:
; 402    H range limit, word.
```

```
; 400    L range limit, word.
;
;
; TAXEL VARIABLES:
; ----------------------
; Change in 1st moment:
; 376    Y axis, H word.
; 374            L word.
; 372    X axis, H word.
; 370            L
;
; 1st moment for imprint:
; 366    Y axis, H word.
; 364            L word.
; 362    X axis, H word.
; 360            L word.
;
; 1st moment for reference imprint:
; 356    Y axis, H word.
; 354            L word.
; 352    X axis, H word.
; 350            L word.
;
; Sum of absolute-taxel-differences:
; 336    H word.
; 334    L word.
;
; Taxel-sum change:
; 332    H word.
; 330    L word.
;
; Taxel-sum for reference imprint:
; 326    H word.
; 324    L word.
;
; Taxel-sum for imprint:
; 322    H word.
; 320    L word.
;
; Average taxel value:
; 306    Reference imprint.
; 304    Maximum pressure imprint.
; 302    Zero pressure imprint.
; 300    Imprint.
;
; 274   Address of various data areas
;        (used in 1st moment routines).
```

```
                            ;
                            ; 262  Imprint background value for
                            ;       correction function #1.
                            ;       (Default value = 000224)
                            ;
                            ; 240 - 257  Temporary storage area.
                            ;
                            ; 232  Address of threshold subroutine.
                            ; 230  Address of correction function
                            ;       associated with thresholding
                            ;       operations.
                            ;
                            ; 226  Output control flag:
                            ;       0/1 ==> no output/output.
                            ;       (Default value = 1)
                            ;
                            ; 224  Output mode:
                            ;       1/2 ==> console/robot controller.
                            ;       (Default value = 1)
                            ;
                            ; 100  Interrupt vector location.
                            ;
                            ;
                            ; SUBROUTINE ADDRESSES:
                            ; ----------------------------
main = 500                  ; Awaits commands from console.
                            ;
sdef = main +  30           ; Set default values to variables.
gwrd = sdef + 240           ; Get 6 digit octal number from console.
iwrd = gwrd + 100           ; Place a word into memory.
owrd = iwrd +  40           ; Read a word from memory.
                            ;
gico = owrd +  40           ; Core routine for getting an imprint.
isub = gico +  40           ; Interrupt subroutine that acquires
                            ; data from Port B.
geti = isub + 140           ; Get imprint.
gzpi = geti +  20           ; Get zero-pressure imprint.
gmpi = gzpi +  30           ; Get maximum-pressure imprint.
gcir = gmpi +  30           ; Copy imprint to reference imprint.
                            ;
                            ; Data correction algorithms:
cor0 = gcir +  30           ; No-op.
cor1 = cor0 +  10           ; Alg 1: sub user-defined baseline.
cor2 = cor1 +  50           ; Alg 2: sub avg zero-pressure baseline.
cor3 = cor2 +  50           ; Alg 3: linearize between average zero
                            ;             and max pressure values.
cor4 = cor3 + 170           ; Alg 4: linearize between actual zero
```

```
                    ;         and max pressure values.
                    ;
                    ; Calculation of average values:
avgc = cor4 + 200   ; Core routine.
avgi = avgc + 100   ; Imprint.
avgz = avgi +  50   ; Zero-pressure imprint.
avgm = avgz +  50   ; Maximum-pressure imprint.
avgr = avgm +  50   ; Reference imprint.
                    ;
demo = avgr +  50   ; Demonstration routine: continually
                    ; acquires and prints imprint after
                    ; correction by selected function.
                    ;
                    ; Return all taxel values:
taxl = demo + 210   ; Core routine.
taxi = taxl +  50   ; Imprint.
taxz = taxi +  20   ; Zero-pressure imprint.
taxm = taxz +  20   ; Maximum-pressure imprint.
taxr = taxm +  20   ; Reference imprint.
                    ;
                    ; Print data in array format:
prin = taxr +  20   ; Core routine.
prii = prin + 100   ; Print imprint taxels.
priz = prii +  20   ; Print zero-pressure imprint taxels.
prim = priz +  20   ; Print maximum-pressure imprint taxels.
prir = prim +  20   ; Print reference imprint taxels.
                    ;
tsum = prir +  20   ; Base routine to sum taxels in imprint.
tsmi = tsum + 120   ; Sum all taxels in imprint.
tsmr = tsmi +  20   ; Sum all taxels in reference imprint.
tscg = tsmr +  20   ; Change in taxel-sum relative to the
                    ; reference imprint.
tsab = tscg + 120   ; Sum of the absolute-value-of-taxel-
                    ; differences relative to the reference.
                    ;
                    ; First moments about X and Y axes:
fmom = tsab + 150   ; Base routine.
fmoi = fmom + 440      ; First moment for imprint.
fmor = fmoi +  30   ; First moment for reference imprint.
fmcg = fmor +  30   ; Determine changed in 1st moments wrt
                    ; reference imprint.
                    ;
thre = fmcg + 250   ; Core routine for threshold monitoring.
tser = thre +  70   ; Service routine.
ttch = tser + 200   ; Check taxels against thresholds.
tsck = ttch + 140   ; Check taxel-sum against thresholds.
tscc = tsck +  40   ; Check taxel-sum-change.
```

```
tsat = tscc +  50    ; Check absolute-taxel-sum.
fmck = tsat +  40    ; Check 1st moments.
fmcc = fmck + 100    ; Check 1st-moment-change.
                     ;
                     ;
; ****************************************************************************
                     ;
. = main             ; MAIN PROGRAM:
                     ;
                     ; Awaits a 6-digit octal command code
                     ; from the console and then branches to
                     ; the appropriate subroutine.
                     ;
cmp  #isub, 100      ; Set default values if not already done.
beq  .+6             ;
jsr  pc, sdef        ;
                     ;
jsr  pc, gwrd        ; Obtain input code from console (r5).
jsr  pc, (r5)        ; Call subr.
jmp  main            ; Await another command.
                     ;
                     ;
; ****************************************************************************
                     ;
. = sdef             ; SET DEFAULT VALUES TO VARIABLES:
                     ;
                     ; Any variables or parameters not
                     ; explicitly mentioned below have the
                     ; default value ZERO.
                     ;
                     ; Change in Y first moment:
mov  #005000, 474    ;   H threshold, H word.
mov  #177777, 472    ;   L threshold, H word.
mov  #173000, 470    ;   L threshold, L word.
                     ; Change in X first moment:
mov  #005000, 464    ;   H threshold, H word.
mov  #177777, 462    ;   L threshold, H word.
mov  #173000, 460    ;   L threshold, L word.
                     ;
                     ; Y first moment:
mov  #020000, 454    ;   H threshold, L word.
mov  #177777, 452    ;   L threshold, H word.
mov  #160000, 450    ;   L threshold, L word.
                     ; X first moment:
mov  #020000, 444    ;   H threshold, L word.
mov  #177777, 442    ;   L threshold, H word.
mov  #160000, 440    ;   L threshold, L word.
```

```
                        ;
                        ; Sum-of-absolute-taxel-differences:
mov   #004000, 434      ;   H threshold, L word.
mov   #001000, 430      ;   L threshold, L word.
                        ;
                        ; Change-in-taxel-sum:
mov   #001000, 424      ;   H threshold, L word.
mov   #177777, 422      ;   L threshold, H word.
mov   #177000, 420      ;   L threshold, L word.
                        ;
                        ; Taxel sum:
mov   #015000, 414      ;   H threshold, L word.
mov   #005000, 410      ;   L threshold, L word.
                        ;
                        ; Taxels:
mov   #000034, 402      ;   H range limit.
mov   #000030, 400      ;   L range limit.
                        ;
mov   #000224, 262      ; Background value for correction
                        ; function #1.
                        ;
mov   #000001, 226      ; Subr output is not suppressed to
                        ; console/controller.
                        ;
mov   #000001, 224      ; Console output mode (i.e. has CRLF after
                        ; each output).
                        ;
mov   #isub, 100        ; Load interrupt vector.
mov   #000340, 102      ; Clear new PSW.
                        ;
rts   pc                ; Return.
                        ;
                        ;
; ******************************************************************************
                        ;
. = gwrd                ; GET OCTAL WORD FROM CONSOLE:
                        ;
                        ; Input six octal digits from the console.
                        ;
clr   r5                ; Result ends up here.
mov   #10000, r2        ; Multiplication factor for various digit
                        ; positions.
                        ;
jsr   pc, 140060        ; Get 6 digit octal input from console.
                        ;
clr   r3                ; Initialize character buffer index.
jsr   pc, 140104        ; Place ASCII code of indexed character
```

```
                        ; into r0 (0 = first).
sub   #60, r0           ; Convert from ASCII to octal number.
cmp   r3, #1            ; Check for sign.
bne   .+16              ; Skip this section if this is not the
                        ; first character.
  tst   r0              ; Examine the value in the first digit.
  beq   .-20            ; Skip next section if first digit = 0.
    mov   #100000, r5   ; Else place a 1 in the sign position.
    jmp   .-26          ; Get next character from buffer.
tst   r0                ; Skip ahead if r0 = 0.
beq   .+6               ;
  add   r2, r5          ; Else sum contributions from digits.
sob   r0, .-6           ;
asr   r2                ; Divide multiplication-factor by 10 = 8..
asr   r2                ;
asr   r2                ;
cmp   r3, #6            ; Loop or exit (r3 is auto-incremented).
blt   .-54              ;
                        ;
rts   pc                ; Return with result in r5.
                        ;
                        ;
; *****************************************************************************
                        ;
. = iwrd                ; WRITE A WORD TO MEMORY:
                        ;
                        ; Write a 6-digit octal word to memory
                        ; from the console.
                        ;
mov   #'A, r0           ; Output "A" to console to request input
                        ; of an 6-digit octal address.
jsr   pc, 140140        ;
jsr   pc, gwrd          ; Get 6 digit octal number.
mov   r5, r4            ;
                        ;
mov   #'W, r0           ; Get the value to be placed in memory.
jsr   pc, 140140        ;
jsr   pc, gwrd          ; Value is placed in r5.
                        ;
mov   r5, (r4)          ; Store value (word) at specified address.
                        ;
rts   pc                ; Return.
                        ;
                        ;
; *****************************************************************************
                        ;
. = owrd                ; READ A WORD FROM MEMORY:
```

```
                    ;
                    ; Output a specified word in memory to the
                    ; console.
                    ;
mov  #'A, r0        ; Request input of memory address (word).
jsr  pc, 140140     ;
jsr  pc, gwrd       ; Get 6-digit word and place in r5.
                    ;
mov  #':, r0        ;
jsr  pc, 140140     ;
mov  (r5), r1       ; Output contents of specified memory
                    ; location to the console.
jsr  pc, 140160     ;
                    ;
rts  pc             ; Return.
                    ;
                    ;
; ****************************************************************************
                    ;
. = gico            ; CORE ROUTINE FOR GETTING AN
                    ; IMPRINT:
                    ;
                    ; A 2000 (octal) byte tactile imprint is
                    ; acquired from the tactile sensor and
                    ; stored in memory starting at location
                    ; r2 and ending at r4.  Thus, this routine
                    ; may be used to get a basic imprint,
                    ; zero-pressure imprint, maximum-pressure
                    ; imprint, or reference imprint.  The
                    ; imprint is organized by rows.  Data
                    ; acquisition is accomplished the
                    ; interrupt routine ISUB, and the primary
                    ; purpose of GICO is to perform some
                    ; initialization steps.
                    ;
mov  #1, r0         ; Initialize line state.
mov  #1, r1         ; Initialize taxel index for each line.
clr  r3             ; Init taxel count (i.e. the number of
                    ; times ISUB has been called).
                    ;
mov  #11, 177444    ; Turn LED ON (zero volts).  This allows
                    ; the interrupt pulses to be gated to the
                    ; DCT-11EB during the next frame cycle.
                    ; The LED is turned OFF by the interrupt
                    ; routine during the first call.
                    ;
cmp  r4, r2         ; Loop until end of data block is reached.
```

```
bge   .-2             ;
                      ;
mov   #3000, r5       ; Time delay = 0.6ms.  This corresponds to
sob   r5, .-0         ; approximately 15 taxel periods to insure
                      ; no interrupts will occur after return to
                      ; the calling routine.
                      ;
rts   pc              ; Return.
                      ;
                      ;
; ***********************************************************************
                      ;
. = isub              ; DATA ACQUISITION INTERRUPT
                      ; SUBROUTINE:
                      ;
                      ; This routine is called by the driving
                      ; the INTB line LOW (0 v).  Each call
                      ; stores another byte of tactile data
                      ; from port B into memory.  A full imprint
                      ; is loaded into a pre-allocated block of
                      ; memory by initializing the following
                      ; registers at the beginning of the
                      ; calling routine:
                      ;
                      ;   r0 = 1 = Line state,
                      ;   r1 = 1 = Taxel index for each line,
                      ;   r2 =     Starting address of desired
                      ;            data block,
                      ;   r3 = 0 = Taxel counter (i.e. the
                      ;            number of times that the
                      ;            interrupt routine was called),
                      ;   r4 =     End address of data block.
                      ;
                      ; Note that only r5 is free for arbitrary
                      ; use by the user.
                      ;
                      ; The line state may take on three values:
                      ;
                      ;   1.  Taxels are extra "starting" ones,
                      ;   2.  Taxels are real data,
                      ;   3.  Taxels are extra "trailing", or
                      ;         "end", ones.
                      ;
                      ; The program cycles through line states
                      ; until the data block is filled with only
                      ; real taxels (all others are ignored).
                      ; Division operations have been avoided in
```

```
                        ; this code to conserve execution time.
                        ;
                        ; It has been determined that the maximum
                        ; rate at which this interrupt routine can
                        ; be executed is 37 microseconds (this
                        ; includes 5 milliseconds for the
                        ; interrupt pulse).  Thus, the frame
                        ; period has been set to 54 milliseconds
                        ; (i.e. interrupt period = 40 us for
                        ; 1344. taxels), which makes the maximum
                        ; data acquisition rate equal to
                        ; approximately 18 Hz.
                        ;
cmpb r3, #0             ; Turn LED OFF during first call.
bne  .+10               ;
 mov  #10, 177444       ;
                        ;
cmp  r3, #2000          ; Check to insure that the allocated space
                        ; for the imprint will not be exceeded.
blt  .+4                ;
  rti                   ; RTI if yes.
                        ;
cmpb r0, #2             ; Determine line state.
bgt  .+56               ; Branch to state 3.
beq  .+24               ; Branch to state 2.  Otherwise fall
                        ; through to state 1.
                        ;
                        ; State 1: taxel is an extra one at the
                        ; start of line.
cmpb r1, #2             ; Compare taxel index with number of extra
beq  .+6                ; taxels (=2).
  incb r1               ; Increment pixel index (skip this taxel).
  rti                   ;
movb #2, r0             ; Change state and skip this taxel.
incb r1                 ;
rti                     ; Return from interrupt.
                        ;
cmpb r1, #34.           ; State 2: Is this a terminal taxel?.
ble  .+12               ;
  movb #3, r0           ; Yes: change state and ignore this taxel.
  incb r1               ;
  rti                   ; Return from interrupt.
movb 177542,(r2)+       ; This is a real taxel so store it and
                        ; increment address.
incb r1                 ; Increment taxel index.
inc  r3                 ; Increment taxel counter.
rti                     ; Return from interrupt.
```

```
                        ;
                        ; State 3: terminal taxels.
cmpb r1, #42.           ; Is this a line-terminating taxel?
beq  .+6                ;
  incb r1               ; No: increment taxel index and ignore
                        ; this taxel.
  rti                   ; Return from interrupt.
movb #1, r0             ; Yes: reset state and ignore this taxel.
movb #1, r1             ;
                        ;
rti                     ; Return from interrupt.
                        ;
                        ;
; ****************************************************************************
                        ;
. = geti                ; GET IMPRINT:
                        ;
mov  #20000, r2         ; Base addr of imprint.
mov  #21777, r4         ; Top of imprint data block.
jsr  pc, gico           ; Call core routine to get data and store
                        ; it in the imprint area.
rts  pc                 ; Return.
                        ;
                        ;
; ****************************************************************************
                        ;
. = gzpi                ; GET ZERO-PRESSURE IMPRINT:
                        ;
mov  #22000, r2         ; Base address of zero-pressure data area.
mov  #23777, r4         ; Top address.
jsr  pc, gico           ; Get data.
jsr  pc, avgz           ; Compute and store average taxel value.
rts  pc                 ; Return.
                        ;
                        ;
; ****************************************************************************
                        ;
. = gmpi                ; GET MAXIMUM-PRESSURE IMPRINT:
                        ;
mov  #24000, r2         ; Base address of maximum-pressure imprint
                        ; data area.
mov  #25777, r4         ; Top address.
jsr  pc, gico           ; Call subroutine to get data.
jsr  pc, avgm           ; Compute and store average taxel value.
rts  pc                 ; Return.
                        ;
                        ;
```

```
; *****************************************************************************
                        ;
. = gcir                ; COPY CURRENT IMPRINT TO
                        ; REFERENCE AREA:
                        ;
mov   #20000, r0        ; Base address of imprint data area.
mov   #26000, r1        ; Base address of reference imprint area.
                        ;
movb (r0)+, (r1)+       ; Copy source (r0) to destination (r1).
cmp   r0, #21777        ; Loop until entire imprint is copied.
ble   .-6               ;
                        ;
rts   pc                ; Return
                        ;
                        ;
; *****************************************************************************
                        ;
. = cor0                ; NO-OP CORRECTION FUNCTION:
                        ;
                        ; Required for thresholding functions.
                        ;
rts   pc                ; Return.
                        ;
                        ;
; *****************************************************************************
                        ;
. = cor1                ; DATA CORRECTION FUNCTION
                        ; NUMBER 1:
                        ;
                        ; Subtract user-defined background.  No
                        ; linearization is performed.  The
                        ; background value must be available in
                        ; memory location 262 (word).
                        ; Default value is 000224.
                        ;
mov   262, r5           ; Get the background value.
                        ;
mov   #20000, r0        ; Address index.
clr   r4                ;
movb  r0, r4            ; Get current taxel.
tst   r4                ; Check to see if a negative value has
bge   .+6               ; been fetched.
  add   #400, r4        ; Correct the value if yes.
sub   r5, r4            ; Subtract background value.
tst   r4                ; Maintain a lower bound of zero.
bge   .+4               ;
  clr   r4              ;
```

```
movb r4, (r0)+          ; Overwrite old taxel value and increment
                        ; taxel addr.
cmp  r0, #21777         ; Loop until all taxels have been
ble  .-32               ; corrected.
                        ;
rts  pc                 ; Return.
                        ;
                        ;
; ****************************************************************************
                        ;
. = cor2                ; DATA CORRECTION FUNCTION
                        ; NUMBER 2:
                        ;
                        ; Similar to function #1, except the
                        ; average value of the zero-pressure
                        ; imprint is used as the baseline.  Note
                        ; that the value of this average is
                        ; computed and stored in 302 when GZPI
                        ; is called.
                        ;
mov  #20000, r0         ; Address index for imprint data block.
clr  r4                 ; Taxels from imprint.
                        ;
mov  302, r5            ; Get avg taxel value of zero pressure imp.
                        ;
movb (r0), r4           ; Get taxel byte.
tst  r4                 ; Convert if needed.
bge  .+6                ;
  add  #400, r4         ;
                        ;
sub  r5, r4             ; Subtract avg value from taxel.
tst  r4                 ; Maintain a lower bound of zero.
bge  .+4                ;
  clr  r4               ;
                        ;
movb r4, (r0)+          ; Overwrite old taxel value and
                        ; increment address.
cmp  r0, #21777         ; Loop until all taxels are corrected.
ble  .-30               ;
                        ;
rts  pc                 ; Return.
                        ;
                        ;
; ****************************************************************************
                        ;
. = cor3                ; DATA CORRECTION FUNCTION
                        ; NUMBER 3:
```

```
                        ;
                        ; Linearizes the response into the range
                        ; [0..37] octal according to the
                        ; expression:
                        ;
                        ;           (imp - avg zp) X 40
                        ;  imp <===  ---------------------
                        ;            (avg mp - avg zp)
                        ;
                        ; where:
                        ;    imp = intensity value for each taxel,
                        ; avg zp = average value of the
                        ;          zero-pressure imprint taxels,
                        ; avg mp = average value of the maximum-
                        ;          pressure imprint taxels.
                        ;
                        ; The average values need to be computed
                        ; prior to calling this routine by AVGZ
                        ; and AVGM, and stored in 302 and 304,
                        ; respectively.
                        ;
                        ; Multiplication and division are
                        ; performed internally to save time.
                        ;
mov  #20000, r0         ; Base addr of imprint.
clr  r5                 ;
                        ;
mov  302, r3            ; Get average taxel value for zero-
                        ; pressure imprint.
mov  304, r5            ; Get average taxel value for maximum-
                        ; pressure imprint.
                        ;
sub  r3, r5             ; Denominator term.
tst  r5                 ; If Den is 0 or negative, then
bgt  .+14               ; set the entire imprint to zero.
  clrb (r0)+            ;
  cmp  r0, #21777       ;
  ble  .-6              ;
  rts  pc               ; Return.
                        ;
mov  r3, r1             ; Save average zero-pressure value.
mov  r5, r2             ; Save Denominator value.
                        ;
movb r0, r4             ; Num terms: get imprint taxel value.
tst  r4                 ; Convert to correct form.
bge  .+6                ;
  add  #400, r4         ;
```

```
sub  r3, r4           ; Subtract. Num must be >= zero.
tst  r4               ;
bge  .+10             ;
  clr  r5             ; If <= 0, then set result to zero and
  jmp  .+62           ; skip the following steps.
asl  r4               ; Multiply result by 40=32. by successive
asl  r4               ; shifting to the left 5 times.
asl  r4               ;
asl  r4               ;
asl  r4               ; Numerator result is in r4.
                      ;
clr  r3               ; Divide by repeated subtraction.
cmp  r5, r4           ;
bgt  .+12             ; Branch if Den > Num.
  sub  r5, r4         ; Else subtract D from N and loop.
  inc  r3             ;
  jmp  .-10           ;
mov  r3, r5           ; Remainder is in r4 and Quotient in r5.
                      ;
tst  r5               ; Check lower bound.
bge  .+10             ;
  clr  r5             ; If negative, then set result to zero.
  jmp  .+16           ;
cmp  r5, #37          ; Check upper bound.
ble  .+6              ;
  movb #37, r5        ; Maximum value is 37=31..
movb r5, (r0)+        ; Store corrected taxel in data block
                      ; (overwrite).
                      ;
mov  r1, r3           ; Restore average value for zero-pressure
                      ; imprint for next loop.
mov  r2, r5           ; Restore value of denominator.
                      ;
cmp  r0, #21777       ; Loop until all taxels are corrected.
ble  .-116            ;
                      ;
rts  pc               ; Return.
                      ;
                      ;
; ****************************************************************************
                      ;
. = cor4              ; DATA CORRECTION FUNCTION
                      ; NUMBER 4:
                      ;
                      ; Linearizes the response into the range
                      ; [0...37] according to the expression:
                      ;
```

```
                    ;    imp <==== (imp - zp) X 40 / (mp - zp)
                    ;
                    ; where
                    ;    imp = intensity value for each taxel,
                    ;    zp  = intensity values for each taxel
                    ;          in the zero-pressure imprint.
                    ;    mp  = intensity values for each taxel
                    ;          in maximum-pressure imprint.
                    ;
                    ; The zero-pressure and maximum-pressure
                    ; imprints thus need to have been
                    ; collected prior to calling this routine.
                    ;
                    ; Multiplication and division is performed
                    ; directly within this code segment to
                    ; avoid time-consuming subroutine calls.
                    ;
mov   #20000, r0    ; Base addr of imprint.
mov   #22000, r1    ; Base addr of zero pressure imprint.
mov   #24000, r2    ; Base addr of maximum pressure imprint.
                    ;
clr  r3             ;
clr  r4             ;
clr  r5             ;
movb  r0, r4        ; Num terms: get imprint taxel value.
tst  r4             ; Check to see if value is negative.
bge  .+6            ;
  add  #400, r4     ; Correct it if yes.
movb (r1)+, r3      ; Get zero-pressure taxel data.
tst  r3             ; Check to see if value is negative.
bge  .+6            ;
  add  #400, r3     ; Correct it if yes.
sub  r3, r4         ; Subtraction.
tst  r4             ; Numerator must be greater than zero.
bgt  .+10           ;
  clr  r4           ; Else, set result to zero
  jmp  .+16         ; and go to next taxel.
asl  r4             ; Multiply result by 40 = 32. by
asl  r4             ; successive shifting to the left.  Do
asl  r4             ; this 5 times to multiply by 32.. Note
asl  r4             ; there is no problem with word overflow.
asl  r4             ; Numerator is in r4.
                    ;
movb (r2)+, r5      ; Den term: get maximum-pressure taxel values.
tst  r5             ; Check to see if value is negative.
bge  .+6            ;
  add #400, r5      ; Correct it if yes.
```

```
sub  r3,  r5          ; Denominator is in r5.
tst  r5               ; Make results become zero.
bgt  .+6              ;
  mov  #40, r5        ;
                      ;
clr  r3               ; Divide by repeated subtraction.
cmp  r5, r4           ;
bgt  .+12             ; Branch if Den > Num.
  sub  r5, r4         ; Else subtract D from N and loop.
  inc  r3             ;
  jmp  .-10           ;
mov  r3, r5           ; Rem is in r4, and Quotient is in r5.
                      ;
tst  r5               ; Check lower bound.
bge  .+10             ; Branch if result is >= zero.
  clr  r5             ; Else set result to zero.
  jmp  .+16           ;
cmp  r5, #37          ; Check upper bound.
ble  .+6              ; Branch if result is <= 31..
  movb #37, r5        ; Else set result to 31..
movb r5, (r0)+        ; Store corrected taxel in imprint data
                      ; block. (overwrite)
                      ;
cmp  r0, #21777       ; Have all taxels been corrected?  If not,
ble  .-156            ; loop.
rts  pc               ;
                      ;
                      ;
; ****************************************************************************
                      ;
. = avgc              ; AVERAGE OF TAXELS (core routine):
                      ;
                      ; Starting address of data block must
                      ; appear in r5.  Result is returned in r0.
                      ;
clr  r1               ; HI word for result.
clr  r0               ; LO word for result.
mov  #2000, r2        ; Taxel counter.
                      ;
                      ; First, compute sum.
clr  r3               ; Temp register.
movb (r5)+, r3        ; Get taxel value.
tst  r3               ; Check to see if taxel value is negative.
bge  .+6              ;
  add  #400, r3       ; Convert to a better form if yes.
add  r3, r0           ; Add taxel value to lower word of sum.
adc  r1               ; Add carry to HI word.
```

```
sob  r2, .-20            ; Loop until all taxels have been added.
                         ;
                         ; Compute average: Div by 2000 = 1024..
asl  r1                  ; Set up HI and LO words appropriately for
tst  r0                  ; division by shifting.
bge  .+10                ;
  add  #100000, r0       ; Change LO word to positive number.
  inc  r1                ; Move 'sign' bit of LO word into HI word.
mov  #12, r2             ; Shift to right 12 = 10. times.
asr  r0                  ; LO word.
asr  r1                  ; HI word.
bcc  .+6                 ; Add carry from HI word to MSB position
  add  #40000, r0        ; of LO word if carry is set.
sob  r2, .-12            ; Shift 12 times.
                         ;
rts  pc                  ; Return with result in r0 (r1 =  zero).
                         ;
                         ;
; ****************************************************************************
                         ;
. = avgi                 ; AVERAGE TAXEL VALUE FOR IMPRINT:
                         ;
mov  #20000, r5          ;
jsr  pc, avgc            ; Call core routine to calculate average.
                         ; Result is returned in r0.
                         ;
mov  r0, r1              ;
mov  r1, 300             ; Store result. (word)
tst  226                 ; Disable output if flag = 0.
beq  .+16                ;
  mov  #'_, r0           ; Delimiter.
  jsr  pc, 140140        ;
  jsr  pc, 140160        ; Print average value for imprint.
                         ;
rts  pc                  ; Return.
                         ;
                         ;
; ****************************************************************************
                         ;
. = avgz                 ; AVERAGE TAXEL VALUE FOR
                         ; ZERO-PRESSURE IMPRINT:
                         ;
mov  #22000, r5          ;
jsr  pc, avgc            ; Call core routine to calculate average.
                         ; Result is returned in r0.
                         ;
mov  r0, r1              ;
```

```
mov   r1, 302          ; Store average value (word).
tst   226              ; Disable output if flag = 0.
beq   .+16             ;
  mov   #'_, r0        ; Delimiter.
  jsr   pc, 140140     ;
  jsr   pc, 140160     ; Print average taxel value for
                       ; zero-pressure imprint.
                       ;
rts   pc               ; Return.
                       ;
                       ;
; ****************************************************************************
                       ;
. = avgm               ; AVERAGE TAXEL VALUE FOR
                       ; MAXIMUM-PRESSURE IMPRINT:
                       ;
mov   #24000, r5       ;
jsr   pc, avgc         ; Call core routine to calc. avg.  Result
                       ; returned in r0.
                       ;
mov   r0, r1           ;
mov   r1, 304          ; Store average value (word).
tst   226              ; Disable output if flag = 0.
beq   .+16             ;
  mov   #'_, r0        ; Delimiter.
  jsr   pc, 140140     ;
  jsr   pc, 140160     ; Print average taxel value for maximum-
                       ; pressure imprint.
                       ;
rts   pc               ; Return.
                       ;
                       ;
; ****************************************************************************
                       ;
. = avgr               ; AVERAGE TAXEL VALUE FOR
                       ; REFERENCE IMPRINT:
                       ;
mov   #26000, r5       ;
jsr   pc, avgc         ; Call core routine to calculate average.
                       ; Result is returned in r0.
                       ;
mov   r0, r1           ;
mov   r1, 306          ; Store average value (word).
tst   226              ; Disable output if flag = 0.
beq   .+16             ;
  mov   #'_, r0        ; Delimiter.
  jsr   pc, 140140     ;
```

```
  jsr   pc, 140160      ; Print average taxel value for reference
                        ; imprint.
                        ;
rts   pc                ; Return.
                        ;
                        ;
; ***************************************************************************
                        ;
. = demo                ; DEMONSTRATION ALGORITHMS:
                        ;
                        ; This is a collection of algorithms that
                        ; cycle through an acquisition and display
                        ; sequence.  The user may select the
                        ; desired demo:
                        ;    000001. Use correction function #1.
                        ;    000002.                          2.
                        ;    000003.                          3.
                        ;    000004.                          4.
                        ;
                        ; All the demo may be run provided the
                        ; zero-pressure and maximum-pressure
                        ; imprints have been taken and the
                        ; corresponding average taxel values
                        ; computed and stored.
                        ;
                        ; The system must be in BYTE mode to
                        ; correctly display the taxels in an
                        ; array format.
                        ;
                        ; The routines will run continuously and
                        ; must be stopped with the following
                        ; control character sequence:
                        ;    ^S (to prevent interrupt generation)
                        ;    ^Y (to abort the program)
                        ;    ^Q (to resume the TEM mode)
                        ;    go (to resume program MAIN)
                        ;
mov   #'N, r0           ; Print symbol requesting menu selection.
jsr   pc, 140140        ;
jsr   pc, gwrd          ; Enter function selection (6 digits,
                        ; e.g., "000003").  Result is in r5.
                        ;
tst   r5                ; Re-enter if number is not valid.
ble   demo              ;
cmp   r5, #1            ; Demo #1.
beq   .+30              ;
cmp   r5, #2            ; Demo #2.
```

```
beq   .+50            ;
cmp   r5, #3          ; Demo #3.
beq   .+70            ;
cmp   r5, #4          ; Demo #4.
beq   .+110           ;
jmp   demo            ; Invalid selection:  try again.
                      ;
                      ; DEMO #1:
jsr  pc, geti         ; Get new imprint.
mov   #10, r0         ; Delay.
sob  r0, .-0          ;
jsr  pc, cor1         ; Apply correction function #1.
jsr  pc, prii         ; Print corrected taxel values.
jmp   .-22            ; Loop until stopped with ^S.....
                      ;
                      ; DEMO #2:
jsr  pc, geti         ; Get new imprint.
mov   #10, r0         ; Delay.
sob  r0, .-0          ;
jsr  pc, cor2         ; Apply correction function #2.
jsr  pc, prii         ; Print corrected taxel values.
jmp   .-22            ; Loop until halted with ^S.....
                      ;
                      ; DEMO #3:
jsr  pc, geti         ; Get new imprint.
mov   #10, r0         ; Delay.
sob  r0, .-0          ;
jsr  pc, cor3         ; Apply correction function #3.
jsr  pc, prii         ; Print corrected data values.
jmp   .-22            ; Loop until stopped with ^S.....
                      ;
                      ; DEMO #4.
jsr  pc, geti         ; Get new imprint.
mov   #10, r0         ; Delay.
sob  r0, .-0          ;
jsr  pc, cor4         ; Apply correction function #4.
jsr  pc, prii         ; Print corrected taxel values.
jmp   .-22            ; Loop until stopped with ^S.....
                      ;
                      ;
; ******************************************************************************
                      ;
. = taxl              ; CORE ROUTINE TO RETURN TAXELS:
                      ;
                      ; R5 must contain the start address of the
                      ; data block of interest.  The system must
                      ; be in BYTE mode to get the TYPOCT
```

```
                        ; routine to return bytes (e.g. "024").
                        ; The data stream sequence is:
                        ; (row0, column0), (r0,c1),...(r0,c31),
                        ; (r1,c0)...(r31,c31).
                        ;
mov   #2000, r4         ; Counter.
mov   #'_, r0           ; Delimiter.
jsr   pc, 140140        ;
                        ;
clrb r1                 ; Register used by TYPOCT.
movb (r5)+, r1          ; Load register with data value.
tst   r1                ; Check to see if value is negative.
bge   .+6               ;
  add   #400, r1        ; Correct value if yes.
jsr   pc, 140160        ; Print byte. (BYTE mode.)
sob   r4, .-20          ; Loop until all taxels are printed.
                        ;
rts   pc                ; Return.
                        ;
                        ;
; ************************************************************************
                        ;
. = taxi                ; RETURN TAXEL VALUES IN IMPRINT:
                        ;
mov   #20000, r5        ; Load start address of imprint.
jsr   pc, taxl          ; Call core routine.
rts   pc                ; Return.
                        ;
                        ;
; ************************************************************************
                        ;
. = taxz                ; RETURN TAXEL VALUES IN
                        ; ZERO-PRESSURE IMPRINT:
                        ;
mov   #22000, r5        ; Load start address of zero-pressure
                        ; imprint.
jsr   pc, taxl          ; Call core routine.
rts   pc                ; Return.
                        ;
                        ;
; ************************************************************************
                        ;
. = taxm                ; RETURN TAXEL VALUES IN
                        ; MAXIMUM-PRESSURE IMPRINT:
                        ;
mov   #24000, r5        ; Load start address of maximum-pressure
                        ; imprint.
```

```
jsr  pc, taxl              ; Call core routine.
rts  pc                    ; Return.
                           ;
                           ;
; ***************************************************************************
                           ;
. = taxr                   ; RETURN TAXELS IN REFERENCE
                           ; IMPRINT:
                           ;
mov  #26000, r5            ; Load start address of reference imprint.
jsr  pc, taxl              ; Call core routine.
rts  pc                    ; Return.
                           ;
                           ;
; ***************************************************************************
                           ;
. = prin                   ; CORE ROUTINE TO PRINT IMPRINTS:
                           ;
                           ; R5 must be loaded with the start address
                           ; of the imprint data block.  System must
                           ; be in byte mode.  Output is printed as
                           ; a 32 x 32 matrix of bytes ranging in
                           ; value from 0 to 37 (octal).
                           ;
mov  #40, r2               ; Row index. (r0 and r1 are used by the
                           ; TYPOCT routine.)
mov  #40, r3               ; Column index.
mov  #' , r4               ; Spacer symbol.
jsr  pc, 140120            ; CRLF.
                           ;
movb (r5)+, r1             ; Move taxel (byte) indexed by r5 into r1.
tst  r1                    ; Test to see if value is negative.
bge  .+6                   ;
  add  #400, r1            ; Correct the value if yes.
jsr  pc, 140160            ; Print byte. (System in BYTE mode.)
cmp  r3, #1                ; CRLF if last taxel on line is reached.
bgt  .+12                  ;
  jsr  pc, 140120          ; CRLF.
  jmp  .+12                ; Skip spacer symbol.
mov  r4, r0                ; Print spacer symbol.
jsr  pc, 140124            ;
sob  r3, .-42              ; Loop until end of line.
mov  #40, r3               ; Reset column index.
sob  r2, .-50              ; Loop until end of taxel array.
                           ;
rts  pc                    ; Return.
                           ;
```

```
                        ;
; ****************************************************************************
                        ;
. = prii                ; PRINT IMPRINT TAXEL VALUES:
                        ;
                        ; System must be in BYTE mode.
                        ;
mov   #20000, r5        ; Load start address of imprint into r5.
jsr   pc, prin          ; Call core routine.
rts   pc                ;
                        ;
                        ;
; ****************************************************************************
                        ;
. = priz                ; PRINT ZERO-PRESSURE IMPRINT
                        ; TAXELS:
                        ;
                        ; System must be in BYTE mode.
                        ;
mov   #22000, r5        ; Load start address of zero-pressure
                        ; imprint into r5.
jsr   pc, prin          ; Call core routine.
rts   pc                ;
                        ;
                        ;
; ****************************************************************************
                        ;
. = prim                ; PRINT MAXIMUM-PRESSURE IMPRINT
                        ; TAXELS:
                        ;
                        ; System must be in BYTE mode.
                        ;
mov   #24000, r5        ; Load start address of maximum-pressure
                        ; imprint into r5.
jsr   pc, prin          ; Call core routine.
rts   pc                ;
                        ;
                        ;
; ****************************************************************************
                        ;
. = prir                ; PRINT REFERENCE IMPRINT TAXEL
                        ; VALUES:
                        ;
                        ; System must be in BYTE mode.
                        ;
mov   #26000, r5        ; Load start address of reference imprint
                        ; into r5.
```

```
jsr   pc, prin          ; Call core routine.
rts   pc                ;
                        ;
                        ;
; ***************************************************************************
                        ;
. = tsum                ; CORE ROUTINE TO CALCULATE SUM
                        ; OF TAXELS:
                        ;
                        ; The base address of the data block of
                        ; interest (e.g. imprint or reference
                        ; imprint) must be initially loaded into
                        ; r5.  Results are double precision
                        ; and are returned in r0 (LO) and r1 (HI).
                        ;
mov   #2000, r3         ; Taxel counter.
clr   r1                ; High order word for sum.
clr   r0                ; Low order word for sum.
                        ;
movb (r5)+, r2          ; Place byte in temporary register.
tst   r2                ; Check to see if value is negative.
bge   .+6               ;
  add   #400, r2        ; Correct the value if yes.
add   r2, r0            ; Add value to sum.
adc   r1                ; Add carry to high order word.
sob   r3, .-16          ; Loop until all taxels have been summed.
                        ;
                        ; Move result into memory:
mov   r1, (r4)          ; H word.
mov   r0, -2(r4)        ; L word.
                        ;
tst   226               ; Disable output if flag = 0.
beq   .+46              ;
  mov   r0, r2          ; Store r0 temporarily (r0 is overwriten).
  mov   #'_, r0         ; Delimiter.
  jsr   pc, 140140      ;
  mov   #'H, r0         ; Identifier.
  jsr   pc, 140140      ;
  jsr   pc, 140160      ; Print H word. (HI word already in r1).
  mov   #'L, r0         ; Identifier.
  jsr   pc, 140140      ;
  mov   r2, r1          ; Print L word.
  jsr   pc, 140160      ;
                        ;
rts   pc                ; Return.
                        ;
                        ;
```

```
; ****************************************************************************
                        ;
. = tsmi                ; CALCULATE SUM OF TAXELS IN
                        ; IMPRINT:
                        ;
                        ; R5 must contain the start address of the
                        ; imprint, and r4 must contain the HI
                        ; address of the permanent storage
                        ; location.  The system must be in WORD
                        ; or INSTRU mode.
                        ;
mov   #20000, r5        ; Load start addr of imprint into r5.
mov   #322, r4          ; Address of H word of permanent storage
                        ; location.
jsr   pc, tsum          ; Call core routine.
                        ;
rts   pc                ; Return.
                        ;
                        ;
; ****************************************************************************
                        ;
. = tsmr                ; CALCULATE SUM-OF-TAXELS IN
                        ; REFERENCE IMPRINT:
                        ;
mov   #26000, r5        ; Load start address of reference imprint
                        ; into r5.
mov   #326, r4          ; Address of H word of permanent storage
                        ; location.
jsr   pc, tsum          ; Call core routine.
                        ;
rts   pc                ; Return.
                        ;
                        ;
; ****************************************************************************
                        ;
. = tscg                ; CALC TAXEL-SUM-CHANGE (relative to
                        ; reference imprint):
                        ;
                        ; The taxel sums for the imprint and
                        ; reference imprint must be calculated
                        ; before use.  Output is double precision.
                        ;
                        ; Calculate difference in taxel sums.
mov   320, r0           ; Don't change stored values.
mov   322, r1           ;
sub   324, r0           ; Double precision subtraction (imp-ref):
                        ;   L order words.
```

```
sbc  r1                ; Subtract carry.
sub  326, r1           ; Sub H order words.
mov  r1,  332          ; Store result: H word.
mov  r0,  330          ;               L word.
                       ;
tst  226               ; Disable output if flag = 0.
beq  .+52              ;
  mov  #'_, r0         ; Delimiter.
  jsr  pc, 140140      ;
  mov  #'H, r0         ; Identifier.
  jsr  pc, 140140      ;
  mov  332, r1         ;
  jsr  pc, 140160      ; Print H word.
                       ;
  mov  #'L, r0         ; Identifier.
  jsr  pc, 140140      ;
  mov  330, r1         ;
  jsr  pc, 140160      ; Print L word.
                       ;
rts  pc                ; Return
                       ;
                       ;
; ****************************************************************************
                       ;
. = tsab               ; SUM OF THE
                       ; ABSOLUTE-VALUE-OF-TAXEL-
                       ; DIFFERENCES (relative to the reference
                       ; imprint):
                       ;
                       ; Output is in double precision form.
                       ;
clr  r5                ; H word for sum.
clr  r4                ; L word for sum.
mov  #20000, r0        ; Base address of imprint.
mov  #26000, r1        ; Base address of reference imprint.
                       ;
movb (r0)+, r2         ; Get imprint taxel value.
tst  r2                ; Check to see if value is negative.
bge  .+6               ;
  add  #400, r2        ; Correct the value if yes.
movb (r1)+, r3         ; Get reference imprint taxel value.
tst  r3                ; Check to see if value if negative.
bge  .+6               ;
  add  #400, r3        ; Correct the value if yes.
sub  r3, r2            ; Subtract reference value from imprint.
tst  r2                ; Negate if result is negative.
bge  .+4               ;
```

```
  neg  r2             ;
add  r2, r4           ; Add result (now always positive) to sum.
adc  r5               ; Add carry to high order word.
cmp  r0, #21777       ; Loop until all taxels have been
                      ; considered.
ble  .-44             ;
                      ;
mov  r5, 336          ; Store H word.
mov  r4, 334          ; Store L word.
                      ;
tst  226              ; Disable output if flag = 0.
beq  .+46             ;
  mov  #'_, r0        ; Delimiter.
  jsr pc, 140140      ;
  mov  #'H, r0        ; Identifier.
  jsr  pc, 140140     ;
  mov  r5, r1         ; Print H word.
  jsr  pc, 140160     ;
  mov  #'L, r0        ; Identifier.
  jsr  pc, 140140     ;
  mov  r4, r1         ; Print L word.
  jsr  pc, 140160     ;
                      ;
rts  pc               ; Return.
                      ;
                      ;
; *****************************************************************************
                      ;
. = fmom              ; CORE ROUTINE TO CALCULATE THE
                      ; FIRST MOMENTS:
                      ;
                      ; This routine calculates values of the
                      ; first moments that are twice those
                      ; obtained by simply summing the lever
                      ; arms.  The results are stored in
                      ; locations 240-247 until moved to the
                      ; permanent location by the calling
                      ; routine.  Location 274 must be initially
                      ; loaded with the starting address of the
                      ; desired data block.  The output is in
                      ; double precision form.
                      ;
                      ; Placement of X and Y axes:
                      ;
                      ;             :  : Fibers
                      ;             :  :
                      ;       ____________________
```

```
                            ;        :                  :
                            ;        :      Y           :
                            ;        :       ^          :
                            ;        :       ^          :
                            ;        :       ^ >>>>     :
                            ;        :             X    :
                            ;        :                  :
                            ;        :                  :
                            ;        --------------------
                            ;
clr   r0                    ; Row index.
clr   r1                    ; Column index.
clr   240                   ; Clear temporary storage locations.
clr   242                   ;
clr   244                   ;
clr   246                   ;
                            ;
movb   274, r2              ; Get taxel value. (Address of data = 274)
inc   274                   ; Increment address (byte) for next loop.
tst   r2                    ;
beq   .+146                 ; Skip to next taxel if value is zero
                            ; (i.e. moment = 0).
tst   r2                    ;
bgt   .+6                   ; Continue if value is positive.
  add #400, r2              ; Else, convert negative word into + word
                            ; and continue. (R2 contains taxel value.)
                            ;
                            ; ****************************************
                            ;
mov   r0, r5                ; Calculate moment about X axis.  Begin X
                            ; with lever arm.
asl   r5                    ; Multiply by 2.
sub   #37, r5               ; r5 = -(twice the x lever arm):
                            ; This makes the sign of the moment OK.
                            ;
cmp   r2, #1                ; Insure that r2 <> 1.
ble   .+14                  ; Skip multiplication code if yes.
mov   r2, r4                ; Move taxel value into r4.
mov   r5, r3                ; Multiplication code segment.
dec   r4                    ; R4 (taxel value) must not be zero.
add   r3, r5                ;
sob   r4, .-2               ; Result is in r5 (word).
                            ;
                            ; Double precision addition of local
                            ; moment to sum.
clr   r3                    ; Local moment, H word.
tst   r5                    ; If local moment is negative, then
```

```
                          ; massage it into the required form
bge   .+6                 ; subsequent double precision addition.
  mov   #177777, r3       ; Move sign into the H word.
add   r5, 240             ; Add L words.
adc   242                 ; Add carry to H result.
add   r3, 242             ; Add H words.
                          ;
                          ; ****************************************
                          ;
                          ;
mov   r1, r5              ; Calc mom about Y axis. Compute Y lever
                          ; arm.
asl   r5                  ;
sub   #37, r5             ; r5 = -(twice y lever arm):
                          ; Sign for moment is OK.
                          ;
cmp   r2, #1              ; Insure that r2 <> 1.
ble   .+14                ; Skip multiplication code if yes.
mov   r2, r4              ; Use taxel value again.
mov   r5, r3              ; Multiply to calculate moment.
dec   r4                  ; R4 (taxel value) must not be zero.
add   r3, r5              ;
sob   r4, .-2             ; Result is in r5 (word).
                          ;
                          ; Double precision addition.
clr   r3                  ; (Same as above)
tst   r5                  ;
bge   .+6                 ;
  mov   #177777, r3       ;
add   r5, 244             ;
adc   246                 ;
add   r3, 246             ;
                          ;
inc   r1                  ; Increment column index.
cmp   r1, #37             ;
ble   .-166               ; Loop if column not yet completed.
  inc   r0                ; Else: increment row index.
  clr   r1                ; Reset column index to zero.
  cmp   r0, #37           ; Has the last row been completed?
  ble   .-200             ; If not, loop.  Else finish.
                          ;
                          ; ****************************************
                          ;
                          ; Copy results from temporary storage
                          ; locations.
mov   #240, r4            ; Base address of temp storage locations.
mov   250, r5             ; Base address of results.
mov   6(r4), 6(r5)        ; Moment about Y axis, H word.
```

```
mov  4(r4), 4(r5)     ;                           L word.
mov  2(r4), 2(r5)     ; Moment about X axis, H word.
mov   (r4),  (r5)     ;                           L word.
                      ;
                      ; Print moments.
tst  226              ; Disable output if flag = 0.
beq  .+140            ;
  mov  #'_, r0        ; Delimiter.
  jsr  pc, 140140     ;
  mov  #'X, r0        ; X first moment.
  jsr  pc, 140140     ;
  mov  #'H, r0        ; H word thereof.
  jsr  pc, 140140     ;
  mov  2(r5), r1      ; H word value.
  jsr  pc, 140160     ;
  mov  #'L, r0        ; L word thereof.
  jsr  pc, 140140     ;
  mov  (r5), r1       ; L word value.
  jsr  pc, 140160     ;
  mov  #'_, r0        ; Delimiter.
  jsr  pc, 140140     ;
  mov  #'Y, r0        ; Y first moment.
  jsr  pc, 140140     ;
  mov  #'H, r0        ; H word thereof.
  jsr  pc, 140140     ;
  mov  6(r5), r1      ; H word value.
  jsr  pc, 140160     ;
  mov  #'L, r0        ; L word thereof.
  jsr  pc, 140140     ;
  mov  4(r5), r1      ; L word value.
  jsr  pc, 140160     ;
                      ;
rts  pc               ; Return.
                      ;
                      ;
; ***************************************************************************
                      ;
. = fmoi              ; FIRST MOMENT FOR IMPRINT:
                      ;
                      ; Compute first moment about the X and
                      ; Y axes for the imprint.  Results are
                      ; double precision. Load starting address
                      ; into 274.
                      ;
mov  #20000, 274      ; Load start address of imprint data block
                      ; into memory location 274.
mov  #360, 250        ; Load L word of X moment ADDRESS into
```

```
                        ; location 250.
jsr  pc, fmom           ; Call core routine.
                        ;
rts  pc                 ; Return.
                        ;
                        ;
; ***********************************************************************
                        ;
. = fmor                ; FIRST MOMENTS FOR REFERENCE
                        ; IMPRINT:
                        ;
                        ; Calculate first moments about X and Y
                        ; axes.  Results are double precision.
                        ;
mov  #26000, 274        ; Load start address of reference imprint
                        ; data block into location 274.
mov  #350, 250          ; Load L word of x moment ADDRESS into
                        ; location 250.
jsr  pc, fmom           ; Call core routine.
                        ;
rts  pc                 ; Return.
                        ;
                        ;
; ***********************************************************************
                        ;
. = fmcg                ; CALCULATE CHANGE IN FIRST
                        ; MOMENTS WITH RESPECT TO THE
                        ; REFERENCE IMPRINT:
                        ;
                        ; This routine presumes that the first
                        ; moments for the imprint and reference
                        ; imprint have already been determined.
                        ;
                        ; Determine change in X first moment.
mov  362, 372           ;
mov  360, 370           ;
sub  350, 370           ; L word result.
sbc  372                ; Subtract carry.
sub  352, 372           ; H word result.
                        ;
tst  226                ; Suppress output if flag = 0.
beq  .+62               ;
  mov  #'_, r0          ; Output delimiter.
  jsr  pc, 140140       ;
  mov  #'X, r0          ; Change in X first moment.
  jsr  pc, 140140       ;
  mov  #'H, r0          ;
```

```
        jsr  pc, 140140      ;
        mov  372, r1         ; Output H word result.
        jsr  pc, 140160      ;
        mov  #'L, r0         ;
        jsr  pc, 140140      ;
        mov  370, r1         ;
        jsr  pc, 140160      ; Output L word result.
                             ;
                             ; Determine change in Y first moment.
    mov  366, 376            ;
    mov  364, 374            ;
    sub  354, 374            ; L word result.
    sbc  376                 ; Subtract carry to HI word.
    sub  356, 376            ; H word result.
                             ;
    tst  226                 ; Suppress output if flag = 0.
    beq  .+62                ;
        mov  #'_, r0         ; Output delimiter.
        jsr  pc, 140140      ;
        mov  #'Y, r0         ; Output "Y" character.
        jsr  pc, 140140      ;
        mov  #'H, r0         ;
        jsr  pc, 140140      ;
        mov  376, r1         ; Output H word.
        jsr  pc, 140160      ;
        mov  #'L, r0         ;
        jsr  pc, 140140      ;
        mov  374, r1         ; Output L word.
        jsr  pc, 140160      ;
                             ;
    rts  pc                  ; Return.
                             ;
                             ;
; ***************************************************************************
                             ;
. = thre                     ; CORE ROUTINE FOR MONITORING
                             ; THRESHOLDS:
                             ;
                             ; The thresholds of a number of parameters
                             ; may be monitored using this routine.
                             ; Two inputs are initially required:  the
                             ; correction function (subroutine address)
                             ; followed by a specific threshold
                             ; monitoring function (subroutine
                             ; address).  This routine will run
                             ; continuously with no output, provided
                             ; no thresholds are violated.  Otherwise,
```

```
                    ; all transgressing taxels and/or
                    ; monitored parameters will be outputted
                    ; to the console.  This routine is
                    ; stopped by the following command
                    ; sequence:
                    ;       ^S  (to prevent interrupts)
                    ;       ^Y  (to terminate execution)
                    ;       ^Q  (to enable the console keyboard)
                    ;       go  (to resume the program)
                    ;
mov  #'C, r0        ; Request address of desired correction
                    ; function.
jsr  pc, 140140     ;
jsr  pc, gwrd       ;
mov  r5, 230        ; Store address in memory.
                    ;
mov  #'T, r0        ; Request address of threshold monitor
                    ; function.
jsr  pc, 140140     ;
jsr  pc, gwrd       ;
mov  r5, 232        ; Store address in memory.
                    ;
jsr  pc, 140120     ; CRLF.
                    ;
jsr  pc, geti       ; Get new imprint.
jsr  pc,  230       ; Apply selected correction function.
jsr  pc,  232       ; Perform selected thresholding operation.
jmp  .-14           ; Repeat the cycle until terminated with
                    ; the [^S, ^Y, ^Q, go] command sequence.
                    ;
rts  pc             ; (Return)
                    ;
                    ;
; ******************************************************************
                    ;
. = tser            ; SERVICE ROUTINE FOR THRESHOLD
                    ; FUNCTIONS:
                    ;
                    ; The thresholding functions perform
                    ; common operations that are implemented
                    ; by this routine.  The following
                    ; registers must contain:
                    ;       r2:  Addr of H word of parameter
                    ;       r3:  Addr of H word of H threshold
                    ;       r4:  ASCII code of a special symbol
                    ;            that will identify the
                    ;            outputted data. (If r4 = 0,
```

```
                                ;           then nothing will be
                                ;           outputted.)
                                ;
                                ;
cmp    (r2),  (r3)              ; Compare the H word of the parameter with
                                ; the H word of the H parameter threshold.
ble   .+12                      ; Skip if OK.
  mov   #'H, r0                 ; Else output the violation.
  jmp   .+56                    ;
                                ;
cmp   -2(r2), -2(r3)            ; Compare the L word of the parameter
                                ; with L word of H parameter threshold.
ble   .+12                      ;
  mov   #'H, r0                 ;
  jmp   .+36                    ; Output the violation.
                                ;
cmp   (r2), -4(r3)              ; Compare H word of the parameter with H
                                ; word of L parameter threshold.
bge   .+12                      ;
  mov   #'L, r0                 ;
  jmp   .+20                    ; Output the violation.
                                ;
cmp   -2(r2), -6(r3)            ; Compare L word of the parameter with L
                                ; word of L parameter threshold.
bge   .+112                     ; If no violations, then skip over output
                                ; section and return.
  mov   #'L, r0                 ; Output the violation.
                                ;
                                ; Output section (skip if there are no
                                ; violations).
  jsr   pc, 140140              ; Indicate which threshold was violated.
  mov   #'_, r0                 ; Delimiter.
  jsr   pc, 140140              ;
  mov   #'H, r0                 ;
  jsr   pc, 140140              ;
  mov   (r2), r1                ; Output H word of parameter.
  jsr   pc, 140160              ;
  mov   #'L, r0                 ; Delimiter.
  jsr   pc, 140140              ;
  mov   -2(r2), r1              ; Output L word of parameter.
  jsr   pc, 140160              ;
  tst   r4                      ; Should a special symbol be printed?
  beq   .+24                    ; No, if r4 = 0.
    mov   #'_, r0               ; Yes, if otherwise.
    jsr   pc, 140140                ; Delimiter.
    mov   r4, r0                ;
    jsr   pc, 140140                ; Output the symbol.
```

```
    mov  #-1, r5        ; Flag for 1st moment routines.
  jsr  pc, 140120       ; CRLF.
                        ;
rts  pc                 ; Return.
                        ;
                        ;
; ******************************************************************************
                        ;
. = ttch                ; MONITOR TAXELS IN SPECIFIED
                        ; RANGE:
                        ;
                        ; Compare each taxel value in an imprint
                        ; against the H and L range limits (words)
                        ; and output the taxel number (0..1777)
                        ; and value (word) of all taxels that are
                        ; within this range (including boundary).
                        ;
                        ; (Note: there are insufficient registers
                        ; to provide the row and column number:
                        ; TYPOCT uses r0 and r1.)
                        ;
                        ;
mov  #20000, r2         ; Start addr of imprint data block.
clr  r3                 ; Taxel index.
clr  r0                 ; If still 0 at the end, then CRLF.
                        ;
movb (r2)+, r4          ; Compare taxel values to thresholds.
tst  r4                 ; Convert to correct form.
bge  .+6                ;
  add  #400, r4         ;
                        ;
cmp  r4, 400            ; Taxel is interesting if
blt  .+60               ; it's value >= L range limit.
                        ;
                        ;
cmp  r4, 402            ; Compare taxel value to H limit.
bgt  .+52               ; Taxel is interesting if <= H limit.
                        ;
                        ; Output taxel number and taxel value
                        ; (words).
  mov  #'N, r0          ;
  jsr  pc, 140140       ;
  mov  r3, r1           ; Taxel number (range = 0..1777).
  jsr  pc, 140160       ;
  mov  #'_, r0          ; Space.
  jsr  pc, 140140       ;
                        ;
```

```
   mov   #'V, r0      ;
   jsr   pc, 140140   ;
   mov   r4, r1       ; Taxel value.
   jsr   pc, 140160   ;
   jsr   pc, 140120   ; CRLF.
                      ;
inc   r3              ; Increment taxel counter.
cmp   r3, #1777       ;
ble   .-104           ; Loop until all taxels have been tested.
                      ;
tst   r0              ; Perform CRLF if any there were any
                      ; violations.  Else do nothing.
beq   .+6             ;
  jsr   pc, 140120    ; CRLF (to space between frame scans).
rts   pc              ; Return.
                      ;
                      ;
; ****************************************************************************
                      ;
. = tsck              ; MONITOR TAXEL-SUM THRESHOLDS:
                      ;
                      ; Compare taxel sum to threshold and
                      ; output the violating value.
                      ;
clr   226             ; Disable output.
jsr   pc, tsmi        ; Compute the taxel sum.
mov   #1, 226         ; Enable output.
                      ;
mov   #322, r2        ; Call subr to check thresholds.
mov   #416, r3        ;
clr   r4              ; No special symbols.
jsr   pc, tser        ;
                      ;
rts   pc              ; Return.
                      ;
                      ;
; ****************************************************************************
                      ;
. = tscc              ; MONITOR TAXEL SUM CHANGE
                      ; THRESHOLDS:
                      ;
                      ; Compare change-in-taxel-sum to threshold
                      ; values and output overrange values.  The
                      ; taxel-sum for the reference imprint must
                      ; have been computed prior to the use of
                      ; this routine.
                      ;
```

```
clr  226                ; Disable subroutine output.
jsr  pc, tsmi           ; Taxel sum for new imprint.
jsr  pc, tscg           ; Calculate taxel sum change relative to
                        ; the reference imprint.
mov  #1, 226            ; Enable subroutine output.
                        ;
mov  #332, r2           ; Call subroutine to check thresholds.
mov  #426, r3           ;
clr  r4                 ; No special symbols.
jsr  pc, tser           ;
                        ;
rts  pc                 ;
                        ;
                        ;
; ***************************************************************************
                        ;
. = tsat                ; MONITOR THRESHOLDS FOR THE
                        ; SUM-OF-ABSOLUTE-TAXEL-DIFFERENCES:
                        ;
                        ; Compare the sum-of-the-absolute-taxel-
                        ; differences to the thresholds and output
                        ; the value of any violations.
                        ;
clr  226                ; Disable subroutine output.
jsr  pc, tsab           ; Calculate sum-of-absolute-taxel-
                        ; differences.
mov  #1, 226            ; Enable subroutine output.
                        ;
mov  #336, r2           ; Call subroutine to check thresholds.
mov  #436, r3           ;
clr  r4                 ; No special symbols.
jsr  pc, tser           ;
                        ;
rts  pc                 ; Return.
                        ;
                        ;
; ***************************************************************************
                        ;
. = fmck                ; MONITOR FIRST-MOMENT THRESHOLDS:
                        ;
                        ; Compare first-moment value with
                        ; thresholds and output the values when
                        ; the thresholds are violated.
                        ;
clr  r5                 ;
clr  226                ; Disable outputs.
jsr  pc, fmoi           ; Calculate first moment for imprint.
```

```
mov   #1, 226          ; Enable outputs.
                       ;
mov   #362, r2         ; Call routine to check X moment
                       ; thresholds.
mov   #446, r3         ;
mov   #'X, r4          ; Special symbol = "X".
jsr   pc, tser         ;
                       ;
mov   #366, r2         ; Call routine to check Y moment
                       ; thresholds.
mov   #456, r3         ;
mov   #'Y, r4          ; Special symbol = "Y".
jsr   pc, tser         ;
                       ;
tst   r5               ; Flag = -1 if thresholds violated.
bge   .+6              ; No CRLF if >= 0.
  jsr   pc, 140120     ; CRLF (to separate frames).
rts   pc               ; Return.
                       ;
                       ;
; *************************************************************************
                       ;
. = fmcc               ; MONITOR CHANGE-IN-FIRST-MOMENTS:
                       ;
                       ; Compare the change in the X and Y first
                       ; moments with respect to the reference
                       ; imprint and the threshold values, and
                       ; output the moment values if the
                       ; thresholds are violated.
                       ;
clr   r5               ;
clr   226              ; Disable output.
jsr   pc, fmoi         ; Calculate first moments.
jsr   pc, fmcg         ; Determine change in first moments with
                       ; respect to the reference imprint.
mov   #1, 226          ; Enable output.
                       ;
mov   #372, r2         ; Call routine to monitor X moment
                       ; thresholds.
mov   #466, r3         ;
mov   #'X, r4          ; Special output symbol.
jsr   pc, tser         ;
                       ;
mov   #376, r2         ; Call routine to monitor Y moment
                       ; thresholds.
mov   #476, r3         ;
mov   #'Y, r4          ; Special output symbol.
```

```
jsr  pc, tser          ;
                       ;
tst  r5                ; Flag = -1 if thresholds violated.
bge  .+6               ; No CRLF if flag >= 0.
  jsr  pc, 140120      ; CRLF (to separate frames).
                       ;
rts  pc                ; Return.
                       ;
                       ;
; ************************************************************************
                       ;
r0 = 0                 ; Clear registers and load pc.
r1 = 0                 ;
r2 = 0                 ;
r3 = 0                 ;
r4 = 0                 ;
r5 = 0                 ;
sp = 6500              ;
pc = main              ;
ps = 0                 ;

.end
```