University of Pennsylvania
ScholarlyCommons

Technical Reports (CIS)

Department of Computer & Information Science

December 1980

Computer Architecture for Object Recognition and Sensing

David James Brown
University of Pennsylvania

Follow this and additional works at: https://repository.upenn.edu/cis_reports

Recommended Citation

David James Brown, "Computer Architecture for Object Recognition and Sensing", . December 1980.

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-80-22.

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/cis_reports/724
For more information, please contact repository@pobox.upenn.edu.

Computer Architecture for Object Recognition and Sensing

Abstract

The notion of using many, most likely different, sensory subsystems in a computer object recognition system immediately provokes several questions:

- How will multiple sensors be used in conjunction?

- What object qualities are best described by which sensor, and how is sensor utilization optimized?

- To what extent does the information provided by each sensor overlap with that provided by others, and how then is this used?

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-80-22.

This technical report is available at ScholarlyCommons: https://repository.upenn.edu/cis_reports/724

MS-CIS-
80-22

University of Pennsylvania

The Moore School of Electrical Engineering

Computer Architecture for Object Recognition and Sensing

David James Brown

Presented to the Faculty of the Department of Computer and Information Science in the College of Engineering and Applied Science, in partial fulfillment of the requirements for the degree of Master of Science in Engineering.

Philadelphia, Pennsylvania

December, 1980

Thesis Supervisor

Graduate Group Chairman

Computer Architecture for Object Recognition and Sensing

David James Brown

December 1980

Table of Contents

Introduction

The notion of using many, most likely different, sensory subsystems in a computer object recognition system immediately provokes several questions:

- How will multiple sensors be used in conjunction?
- What object qualities are best described by which sensor, and how is sensor utilization optimized?
- To what extent does the information provided by each sensor overlap with that provided by others, and how then is this used?

1.1 Unifying Issues

Several of these topics have commonality with analogous problems in the areas of manipulator control, control theory, distributed computer architecture, and generally, robotics. We have attempted to draw together, compactly and concisely, a summary of the major unifying issues involved in this type of coordinated subsystem control:

Planning

- How do we evolve actions which approach the solution of system goals.

Control

- How is sensor control carried out — how are sensory subsystems orchestrated
- Will susbsystems be mostly autonomous, or receive a great deal of direction from a higher level
- How is control partitioned throughout the various levels of the system

Representation and Knowledge

- How is knowledge incorporated, and at what levels of the system
- How does knowledge relate to planning and control issues
- How is information (object characteristics, perceived sensory features) described. Does this vary as it moves through the system, and if so how?

Communication

- What are the paths of knowledge transfer.
- Does information transfer imply a representational transition?
- How is compatibility of sensor-derived information achieved as it moves toward, and is ultimately incorporated into, high-level descriptions?

Many, if not all, of the above issues are difficult to resolve. It is certain that they will remain within research interest for some time to come. Secondly, it is highly important to realize the interrelation of each of the above issues and categories. While we may approach solutions to the problem of designing such a system by giving consideration to these issues separately, it is quite clear that decisions made in any category have immediate consequence in others. For example, if we decide to enact a particular control structure, this will mandate certain paths of communication. Similarly, planning and control are closely interrelated. Planning strategies imply that certain control be available to carry out these plans. Most generally, we can say that it is important to consider the solution to such a design problem as being a cooperative one -- one which effectively incorporates solutions to all problems: planning, control, representation, and communication, into an integrated unified system.

Hierarchical Systems — A Central Theme

As may be intuited from what has already transpired in this discussion, a notion of "levels" is important to a system of the type we are designing. In fact, hierarchy is a central theme in almost all computer systems that presently exist. It is common practice to place such levels of software as are necessary to bring that pile of solid state semiconductor devices, amorously termed "computer", "up to the level" where it can be used effectively. Namely, said machine can execute instructions that require a minimal (or "reasonable") amount of effort on the part of some operator (usually human) to specify some requisite task. Plainly, we are discussing "power of instruction set". The aim of artificial intelligence proponents is to make a task description for computer systems be as close as possible to the manner of specifying it to another human.

2.1 Relevance to Object Recognition Systems

Let us now discuss the importance of hierarchical systems as it relates to performance of object recognition. It is generally agreed that systems of the type proposed — namely recognition systems based on multisensing, will be multilevel ones. That is, we may consider our system to have a fundamentally hierarchical structure about it. Winston states that "Intelligent programs are built upon many layers of information processing". Why is this the case? There are goals which relate to the description and recognition of objects or scenes in our systems, but these ultimately reside at a "level" above many underlying levels of subgoals and actions. It is clear that, at the lowest level, very primitive operators

gather raw data from sensors by means of very low level operations. It is the selective use and/or processing of this raw data that yields useful information at the next higher level. Processing is a term yet to be fully exposed to light. What does processing of sensory information entail, such that it is relevant to the goal of object recognition? Primarily, processing involves refinement (i.e. noise reduction, filtering, or enhancement) at the lowest level. This is incorporated with knowledge that is at first sensor specific, and progressively higher-level representation or description specific. Knowledge, or more importantly, its incorporation into the descriptive process (along with information derived from selected raw data), is contextual in this regard. In the process of recognition, it is through successive refinement, collaboration by other information sources, and abstraction, that we ultimately "describe" the observed object.

2.2 Stratified Architecture - Virtual Machines

In the architectural context we may regard our system as a hierarchy of virtual machines [1], each one more powerful than the one(s) below. What do we mean by "powerful"? Simply, we can define the power of a virtual machine as the extent to which one of its primitives encompasses the operative and/or descriptive capabilities of primitives on lower level virtual machines. Power is a relative term. In a traditional architecture which supports one or several high level programming languages, several levels are described:

== Hardware ==

Level

0: (electronic devices) transistors and solid state devices

1: (logical entities) latches and gates

2: (Register Transfer Level) registers and processing elements

== Software ==

Level

3: (Machine Level) Machine language programs

4: (Assembly Language Level) Assembly language programs

5: (Compilers and Interpreters)

6: (Higher Level Programming Languages)

7 & above: Systems building upon Levels 0-6
(DBMS, A.I. systems,
Mathematical and Statistical Models etc.)

Everything below level 3 is fairly rigidly fixed "hardware" in that it is implemented in actual electronic devices and their composition. It is very interesting to note the overwhelming importance of these low levels in determining to a large extent the limitations, and ease of implementation of hierarchically superior virtual machines. Classically, all that transpires at and above level 3 is carried out in "software" — programs which are more easily subject to change. Ostensibly, levels 3, 4, and 5 are fairly firm in a general purpose computer. It is above the level of higher level programming languages that special purpose systems are developed, and reside.

It is important to realize one point here, which will be central to subsequent discussion: The progression from level 0 to level k involves two main trends that are of great importance. First, and motivational to the

development of higher level machines, is that, as we move higher up the architectural hierarchy, we gain facility in the specification of tasks — power. Why then is not everything done at the highest possible level, and why if this is possible do we not continually evolve more powerful virtual machines? While high level virtual machines are increasingly powerful, the price is paid in two ways: Abstraction from lower levels occurs — we lose the specificity of the lower levels. And specialization is inevitable after level 6 — systems are developed with specific intention; generality must be lost. In fact, the observation can be made that certain high level (problem oriented) languages favor certain applications more than others. If this were not the case, far fewer such languages would exist.

2.3 Beneficial Consequences of Stratified Architecture

Let us now examine the consequences of such an architectural scheme in so far as it is relevant to the design of an object recogntion system.

One of the chief advantages of a hierarchical system is the increasing power of operators as we progress toward higher level virtual machines. These virtual machines gain power by defining new primitives which exploit the capabilities of lower level virtual machines, in conjunction with processing. Other benefits are as follows:

Benefits

1) Higher level virtual machines support operations which encompass greater scope.

2) Higher levels organize the use of lower level systems, and hence reduce their complexity of use.

3) Distributed architecture affords the potential exploitation of concurrent execution of low level operations and tasks.

Interestingly, some of the aforementioned disadvantages that exist in light of hierarchical structure may also be of advantage in a system performing recognition.

Further Benefits

1) High level control loses touch with the specific nature and detail of low level tasks. This is beneficial in that it ABSTRACTS from the low level details.

2) Overhead for organization and coordination of the components of a distributed system is well invested. That is, it ultimately implements the basis of our control and communication structure.

Beside the clear advantages of hierarchical system structure in certain instances, we can point to several reasons why a such a scheme is necessary in this instance. First, the complexity of a system employing many diverse sensing devices, many representational schemes, and various types of knowledge (knowledge relevant to sensing, relation between sensory information, and ultimately high level descriptive knowledge), will be great. In order to handle this complexity, a well defined architectural and organizational structure will be required. The apparent multi-level nature of the problem at hand, is indicative that a likewise hierarchical architecture and organization is called for. Second, there are tasks within the scope of our system that are intrinsically resident at a given conceptual level. For example, the control of sensors is one such case. We do not want to burden the processor associated with high level goals, with such low-level sensor control tasks. We would rather provide a

separate processor to perform localized sensor control at a low level, thus removing this from the concern of conceptually higher level processors and processes. Once again this calls for hierarchy. Finally, low level implementation details will hamper the performance of conceptually higher level intentions. It is a strata of virtual machines that will provide the power that we need to carry out high level goals. Intrinsic to virtual machines will be languages suited to the nature and level of the tasks they carry out within their planning and objective schemes. At higher levels, languages which are desensitized from the restrictions imposed by low level objectives will exist. This allows operators of the particular virtual machine to be well suited to its own needs rather than encumbered always by the nature in which low level tasks are specified.

To summarize, three points have been given to justify stratified architecture:

1) Need to handle system complexity —> Reduction of complexity through multi-level organization

2) Special needs at conceptual levels —> Local control of local goals, plans, and methods of solution

3) Requirement of greater scope and generality at higher levels —> Abstraction from low level implementation details by a layering of increasing power virtual machines

III. The Design of Object Recognition Systems

Preliminary sections of this treatment have attempted to sensitize some important issues (in particular, planning, the use and representation of knowledge, control and communication) and intimated the importance of multi-level systems. It is now time to draw issues and inclinations together, and elucidate that which we anticipate and aspire to, in the design of a computer recognition system. In the section following this one, strategies for carrying out the ideas contained in this one are developed.

3.1 Fundamental Objectives

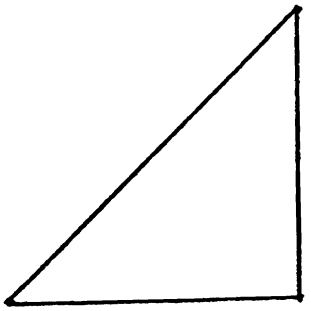
A computer system for object recognition (herein after referred to as an ORS), has as its functional objective the description and recognition of objects, and to extend this, the description and recognition of the perceived environment. While the preceding statement sounds almost tautological, the terms 'description' and 'recognition' are somewhat unclear. We will need to elucidate the ideas of 'means' and 'level' of description — the how and what of it, before we extend this by considering the notion of 'recognition'.

3.2 Description

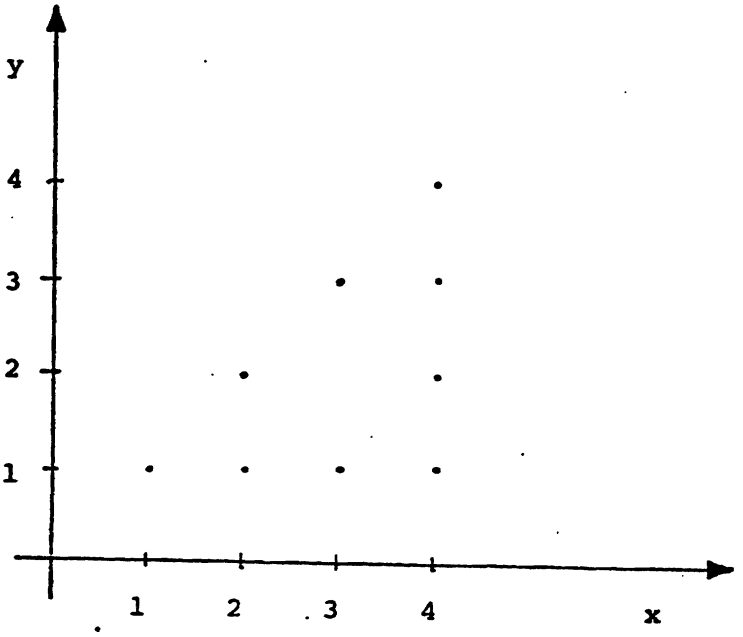
In order to perform 'description' of an object, we would like to provide a means of access to high-level properties or descriptors which serve this function. This is to say, that while we may 'describe' an observed object by specifying its locus of points in the region of

observation (or observation world), as a list of coordinate n-tuples (triples in three-space), this, along with such other low-level information as may be obtained, is not what we ultimately desire. For example, in the diagram below (see figure #1), which is an object in a two-dimensional observation world, we may 'describe' it as follows:

(1,1) (2,2) (3,3) (4,4) (4,3) (4,2) (4,1) (3,1) (2,1).



This is achieved as a result of applying orthogonal coordinate axes to the observation region, and then providing a list (in this case ordered) which provides the locus of points at which the object resides in this world, with some finite resolution (see figure #2).



As the reader no doubt observes, the description given in this low-level form is not the one which immediately comes to mind when one first observes figure 1. We would much rather receive the description 'triangle,' or some other appropriate higher-level one. What we are observing by this example is the difference between a description which is likely to be provided by a low level processor of the ORS, and the more desirable high-level result which is convenient to the human recognition system. It is the application of generalization, categorization, and knowledge association that achieves this recognition (as yet undefined), that in turn yields the high level description which we desire. What is more important here, is that raw data received from a potential low-level ORS processor does not constitute all that we desire in that object's description.

As a second example, consider a list of coordinate triples 'describing' an object in a three dimensional object world, with a component-wise resolution of 1mm (that is x,y,and z coordinates which are components of triples are specified to one millimeter precision). The object is a roughly spherical blob with granular surface (granules much larger than 1mm deep), and having approximate volume of 1000 cubic centimeters. You can imagine that these triples, whose resultant conglomeration enumerates a list of over one hundred thousand, is of questionable worth in so far as giving an immediate and clear understanding of the object. There can be little debate, however, that such a list does convey a rather explicit detailed specification of the object — in fact, a description.

The problem with this list is twofold: First, the description is at too low a level. Second, the resolution may be excessive, and hence the

work done by the sensory acquisition processor, which is roughly quadratically related in giving the surface description, is far more excessive. Consider a cube in the same region which may be processed for description in the same fashion (1mm pointwise resolution with external surface description of the cube). The resultant list is clearly a gross waste, as the cube may be equally well described by specifying only two points (any pair of diagonally opposed corners).

What we must do to acceptably describe an object involves several things alluded to by the words 'approximate volume', 'rather granular surface', and 'roughly spherical blob', given in the earlier decription of that object. We will without doubt, however, invoke plans at a low level to perform raw data acquisition. These plans and their inherent actions underlie higher level plans which intend to perform several operations on this retrieved low-level information, in order to raise the level of the description.

* Associate pre-existing system knowledge with what is observed
* Omit unnecessary low level data in order to generalize
* Approximate where necessary by applying pre-existing 'means of description' (categorization or parameterized descriptions such as volume, texture, or other morphological or conceptual notions).

Some of what is being suggested here incorporates the use of general means by which data is composed into less specific descriptions at a higher system level. These descriptions may be somewhat vague (that is, less precise and detailed), but this is one means by which we gain aforementioned power, and progress toward the realization of higher level descriptions in higher level virtual machines. We are performing

abstraction, which consists of a reduction, or stripping away of some of the detailed low level information. This may be achieved by association of knowledge to observation and inference of set membership. Or, observed detail may be compacted by excepting description-irrelevant low level information.

Goals, and plans to implement them, must enact these ideas. Further, multilevel planning seems essential in the accomplishment of these goals. Low level system components will have different goals than those of high level components. The types of descriptions they deal with, and the manner of handling these -- their strategies for accomplishment of objectives for them, will also be different. Figure #3 illustrates a preliminary breakdown of some of this intended planning hierarchy.

3.3 Recognition

As yet, we have not explained our understanding of 'recognition.' What is it? Recognition, we think, involves the notion of set membership, or classification. In other words, the ability to perceive, followed by generalization by relating pre-existing system knowledge to what has been observed. Knowledge may be used in one of two important ways: Classically, knowledge has been used to classify objects based on properties derived from the various sensory data acquisition devices of the system [1]. Knowledge may also be applied to relationships between objects in the observed world.

We have seen so far, that while we may provide a low level description of an observed object simply by enumerating its properties as observed by low level sensory processors, this does not provide all that we desire. Higher level descriptions are what we really desire, and it is recognition

that applies to the derivation of these. Recognition in turn is accomplished via the incorporation of knowledge with that which has been perceived. We must combine basic features into conclusions in order to achieve higher level descriptions.

3.3.1 Production Systems

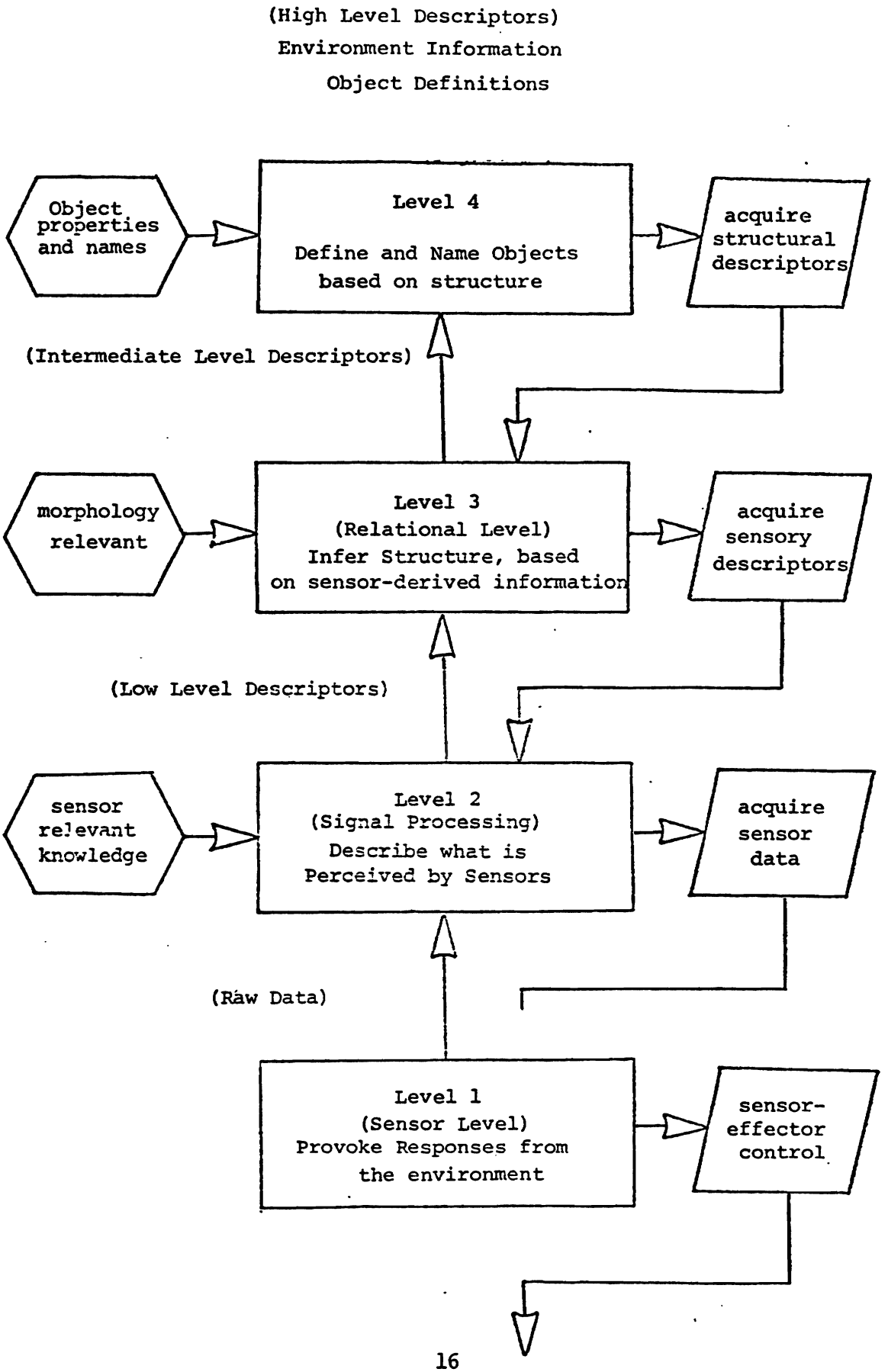
Systems which prescribe to these goals have been referred to as production systems. Production systems consist of three computational components: a global database, a set of rules called productions, and a control system. The database contains the system's knowledge, and the production rules describe the manner in which this knowledge is applied and changed, based on the situation at hand. The control system in a deductive system such as ours — one which intends to draw conclusions about observed objects, applies productions to derive facts. Productions are situation-valid, and may be considered to be premise-conclusion pairs. For example, "If I observe texture x, shape y," and so on, "I may conclude that I have a 'z'." Low level information is thus composed under the governance of the production system to draw conclusions which yield higher level descriptions.

King and Davis [4] among many others describe production systems at greater length. These ideas will certainly be consequential in the realization of an effective ORS.

3.4 Control, Communication, and Representation

Now that some insight has been provided about description and recognition, a few words must be said about control, communication, and representation. Representation is an issue central to our system. It refers to the manner in which information will be conveyed or described.

Figure #3 - Preliminary Planning Hierarchy



Winston defines representation as "a set of conventions about how to describe things." (p.15 in [3]). A control system acts upon perceived situations within the framework of the representation we have chosen. It is important for this reason that we choose an "appropriate" one; one which makes the goal of the system as easy to achieve as possible. Nilsson states: "Efficient problem solution requires more than an efficient control strategy. It requires selecting good representations for problem states, moves, and goal conditions." (p.27 [2]). Since our goal in proposing an ORS design is to decompose our problem into a hierarchy of sub-problems, each having relevance to a different sensory environment, we must consider how we will represent states, knowledge and goals at each of these levels.

Control in our system will implement the application of rules to perceived information. And, importantly, it will effect the conveyance of low level descriptions to higher level ORS processors. To accomplish this, control involves communication, and one or both of control and communication are responsible for the possible translation that occurs between system levels. Such a translation applies to the manner of representation at the interface between system levels, and only between such levels at which the respective manner of describing is different. Such transformations are necessary, and can be justified in a general way as follows: At a low level, acquired information is sensor specific, and most likely, the representations at such low system levels will cater to the specific problems and goals associated with the sensors. This is in order that these processors be most effective in solving these sensor-specific problems. In order to compose the information retrieved from different sensors however, a compatible representation must be realized at some level. It must be one which allows the interaction of, and comparison among, these different lower level frames of reference.

3.4.1 Representational Structures

Consideration has been given to the topic of representation by several researchers. (Minsky 1975, [6]). Such structures as frames, semantic networks, and property lists have been proposed as representational schemes. Property lists, while most concrete, are most specific to the nature of the pre-existing knowledge of the system. Recognition may be performed based on the joint satisfaction of property lists known to be associated with a particular object. Property lists have problems however, in situations where loose association, or partial satisfaction of properties exist. Specifically, this is likely to be the case in an instance where the ORS encounters a new object (one for which the ORS has as yet little or no knowledge). While Minsky's frames ideas seem much more flexible in this regard, it is much less clear how to implement a system embodying them. They are presently more of an abstract notion on how to handle the process of combining information derived from many different contexts of reference.

3.4.2 Fuzzy Automata

It is possible that we may integrate these ideas in an implementation which allows the aforementioned loose association with, or partial satisfaction of properties, using the ideas proposed by Zadeh. ([7]-Zadeh). Recognition may be represented as fuzzy set membership, which is based on a closeness of association with set properties. Stronger set membership is established for an observed object, based on the extent to which low level observed properties concur with set properties as knowledge is applied at successively higher levels in the system. When knowledge applied to

observations provides a discrepancy, or less extensive associations exist with set properties, weaker set membership exists. The result may then be expressed as weak membership in several sets, or rather, that the object lies in a region of fuzzy association with one or several sets.

There is tremendous power in the application of fuzzy sets, fuzzy algebra, and more generally, fuzzy automata, to the recognition process. The power of Zadeh's notion is that the progression from set membership to non membership is not abrupt, as in classical set theory, but rather a gradual progression. The intention of ours in this regard is that close association may be achieved with system-known entities without strict success or failure in identifying what has been observed with an explicit element of the knowledge base. The concept of fuzzy sets is particularly valuable in that it allows for nearness measures for association of observed instance with existing knowledge. For this reason, it seems particularly applicable in situations where there is a limited knowledge base — a situation which is necessitated due to the bounded capacity of computer systems.

An example of the use of "fuzzyism" can be obtained by observing a human performing the description/recognition process for some object with which he is unfamiliar, that is, an object for which he lacks a specific high level term — the object is not in his knowledge base. What is most likely observed in this situation, is a description based on constituent components of the object — lower level descriptions, along with statements about the closeness of association of the new object with those that are known. Descriptions like "predominantly spherical", or "large", or "rough", are examples of this. These terms are applied to attempt to put the object into relative perspective with the rest of the knowledge base. This is one important application of fuzzy automata.

Thus far, this section has attempted to clarify the intention of the terms 'description' and 'recognition', and provide insights into possible representational approaches which may be incorporated into an effective ORS. In light of these ideas, let us now consider what we desire in an architecture, such that it will be supportive both of the notions discussed in preceeding sections, and supportive of various possible vehicles of solving the recognition problem.

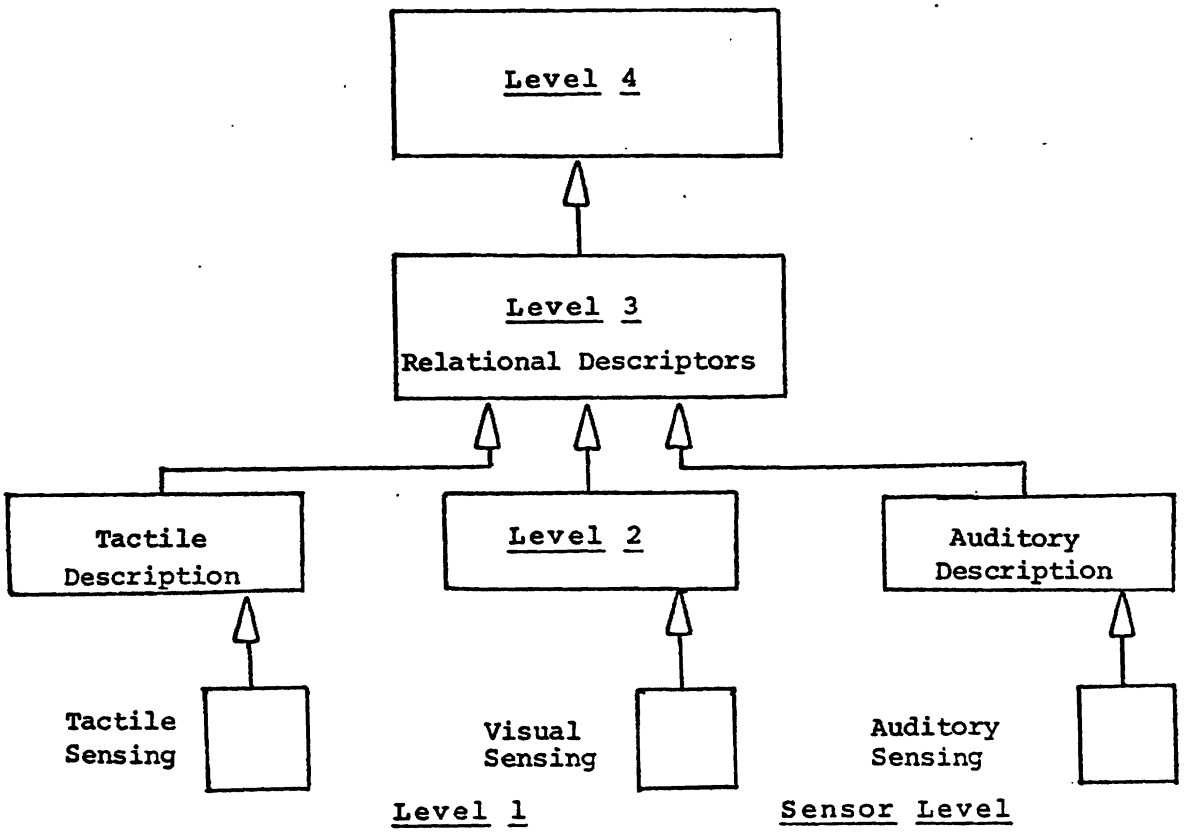
3.5 Computer Architecture for an Object Recognition System

We have stated several major conceptual desires for our ORS:

- Stratification
- Multilevel Planning
- Goals provoking Subgoals
- Local Control of local plans, goals, and descriptions
- Accomodation of various representational schemes
- Inter-level and same-level compatibility between processes

The hierarchical nature of the system provides a natural partition — a stratification which encompasses several levels of processing (refer back to figure #3). This stratification may be carried out in software, all residing within one powerful general purpose computer, as has been done for example in implementing virtual machines for problem oriented languages. The alternative to this solution is that we can allocate limited capacity dedicated hardware processors at some or all of these system levels. This is a wise decision, as it allows us to make the nature of the hardware in each instance, as close a possible to the nature of the virtual machine that the level intends. This implementation can be carried out in microcode and/or in firmware which caters to the respective levels' goals, plans and representations.

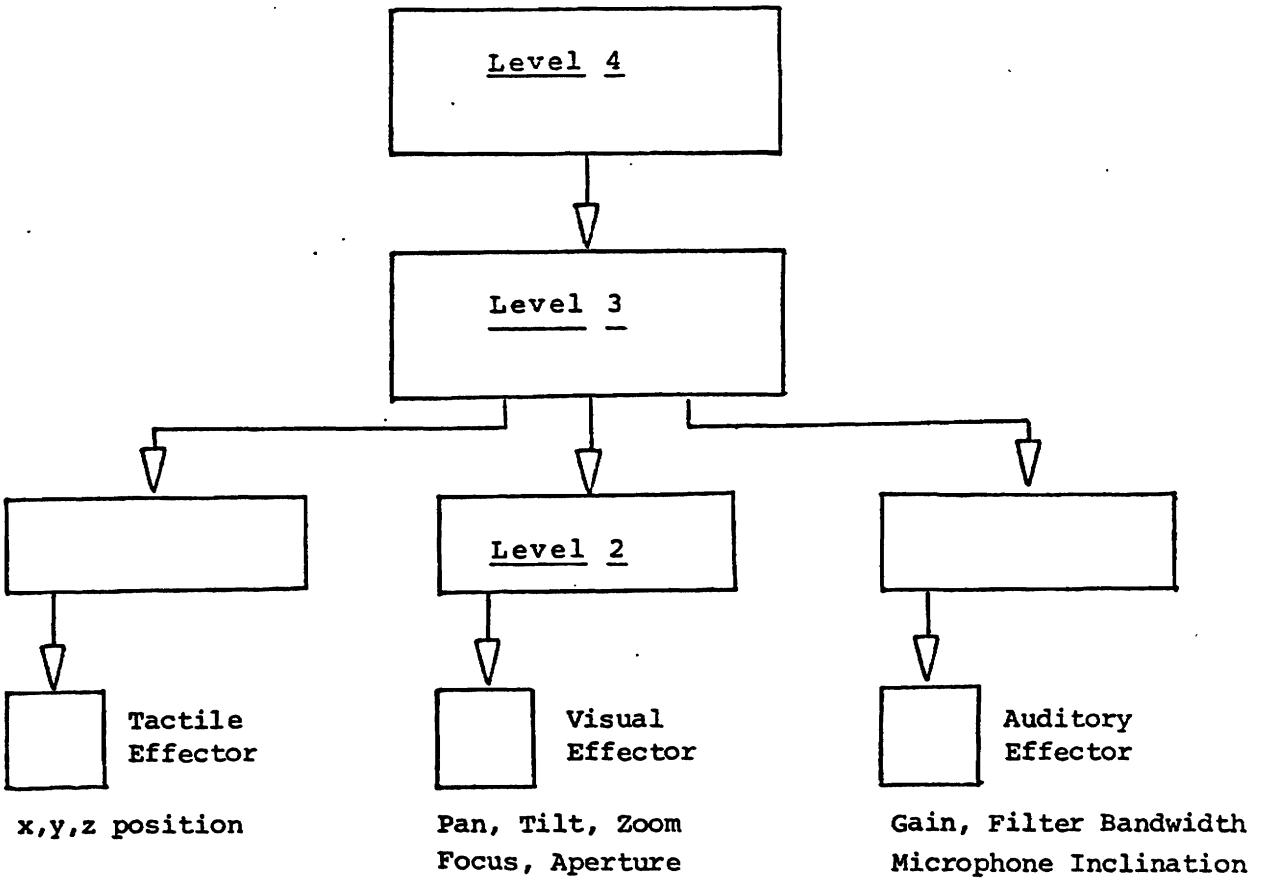
Before advancing to a discussion of architectural support for multilevel planning, goals and the remainder of the conceptual desires above, let us first diagram the skeleton of an architectural scheme following what has just been discussed. Figure #4 portrays an ORS with three primary means of sensing: tactile, visual, and audible.



These three sensory mechanisms form the lowest level in the ORS. Architecturally, this introduces a need for distributed control within the hierarchical scheme. This is because these devices are retrieving raw data that is sensor-specific at this level, and hence each has no relevance to its lateral architectural counterparts. At the next level we can delegate

one hardware processor to each of these sensing units, each catering in function to the nature of its respective sensor. At the Relational (morphological) level, we have only one hardware processor. This choice seems reasonable since this is the first stage at which sensor descriptions are related to derive structure. It seems unwise to retain separate control at or above this point in terms of hardware processors, since an integrated, sensor independent descriptive level has been achieved at this stage. The succeeding level(s) consist of one hardware processor chiefly dedicated to the performance of a knowledge association task to derive representation-relevant descriptions of that level.

Now let us continue our discussion of the other conceptual desires, seeing how they are supported by, and further qualify, the architecture thus far described. Multilevel Planning can be supported as strata of plans. The goal of recognition at the highest level and the plan(s) to achieve it provoke actions at the next lower level. This may be considered as a sub-plan of the high level system, to carry out a sub-goal, or as a concurrent lower level plan and goal. In this fashion plans at high levels propagate to plan enactment at lower levels. This should be structured so that plans invoked at a given level are those which are suited to the temporal needs of the immediately superior processor in the hierarchy. This situation, it seems, should be a dynamic one, in which plans are enacted, cancelled, replaced, and possibly even developed autonomously based on the demands posed by higher levels. This demands architecturally that there be communication of planning needs and goals downward to lower levels from higher ones (see figure #4a). There is much room for research into the development of ideas for multilevel planning and implementation of such ideas as propagation of planning such that plan generation within



lower levels is near-autonomous and responsive to a dynamic system (one in which they must accommodate changes in high level plans and/or demands).

Local control of local plans, goals and descriptions falls neatly into what has so far been described, as does accommodation of various representational schemes. This is true, because we have stratified the architecture and the planning scheme. Thus, plans which are local, that is relate to a specific sensor, goal, representation or other issue, have a position in the planning hierarchy. One thing, which has perhaps been skimmed over, is that there will be provision for the enactment of more than one process, goal, or plan at one level, and more specifically, this should also be true for one hardware processor. That is, plans and goals

should be able to be added or removed within one architectural mode. This means that one processor shall be able to dynamically start or terminate tasks, in addition to those already running in order to support additional plans or goals, as a need for these becomes apparent. This structure has been incorporated within the tactile sensing system's implementation, as will be touched upon in Part II, and is given full detail in Wolfeld (1981). This idea does not reflect itself diagramatically, but may be imagined by the reader, as the robustness of the processing mode in question - the extent to which it is being used, and the number of tasks it is running to support plans or goals. This strategy is very important in so far as it supports the idea of allowing higher level processes to rely more heavily upon some subsystems than on others. In practical application, this may be an instance in which high level recognition goals are supported better by visual information, during certain parts of plan enactment, than by other sensory information. In this regard, as has previously been mentioned, sensory subsystems act as low level specialists. It is natural that in certain instances - certain objects or situations, there is more call for one specialist than another.

Inter-level information has been explained to a major extent. This consists of upward communication of descriptive information on the one hand, and downward communication of plans and goals on the other hand. What of the exchange of information between processors at the same stratum then? Such paths are extremely important in that they provide routes by which control and recognition process loops may be closed. Use of control loops is highly important to the ORS. At a low level, this is observed in servoing of sensors - interaction between effector paths and sensor paths. Examples of this are pan, tilt, zoom, aperture and focus of camera in

conjunction with what the camera observes, or gain and/or bandwidth adjustment of filters in conjunction with auditory response. These control loops provide response paths for sensor control at a low level, to perform such functions as automatic focus and aperture adjustment of the visual unit, automatic gain and filter range adjustment of acoustic sensor, and motor control to maintain proper pressure between tactile sensor and the object contacted. A complete system diagram is shown in figure #4b.

Beside the obvious need for control loops where low level servoing is performed, there is also merit for lateral communication between higher level processors. Such communication will be necessary to facilitate loops within the description/recognition process.

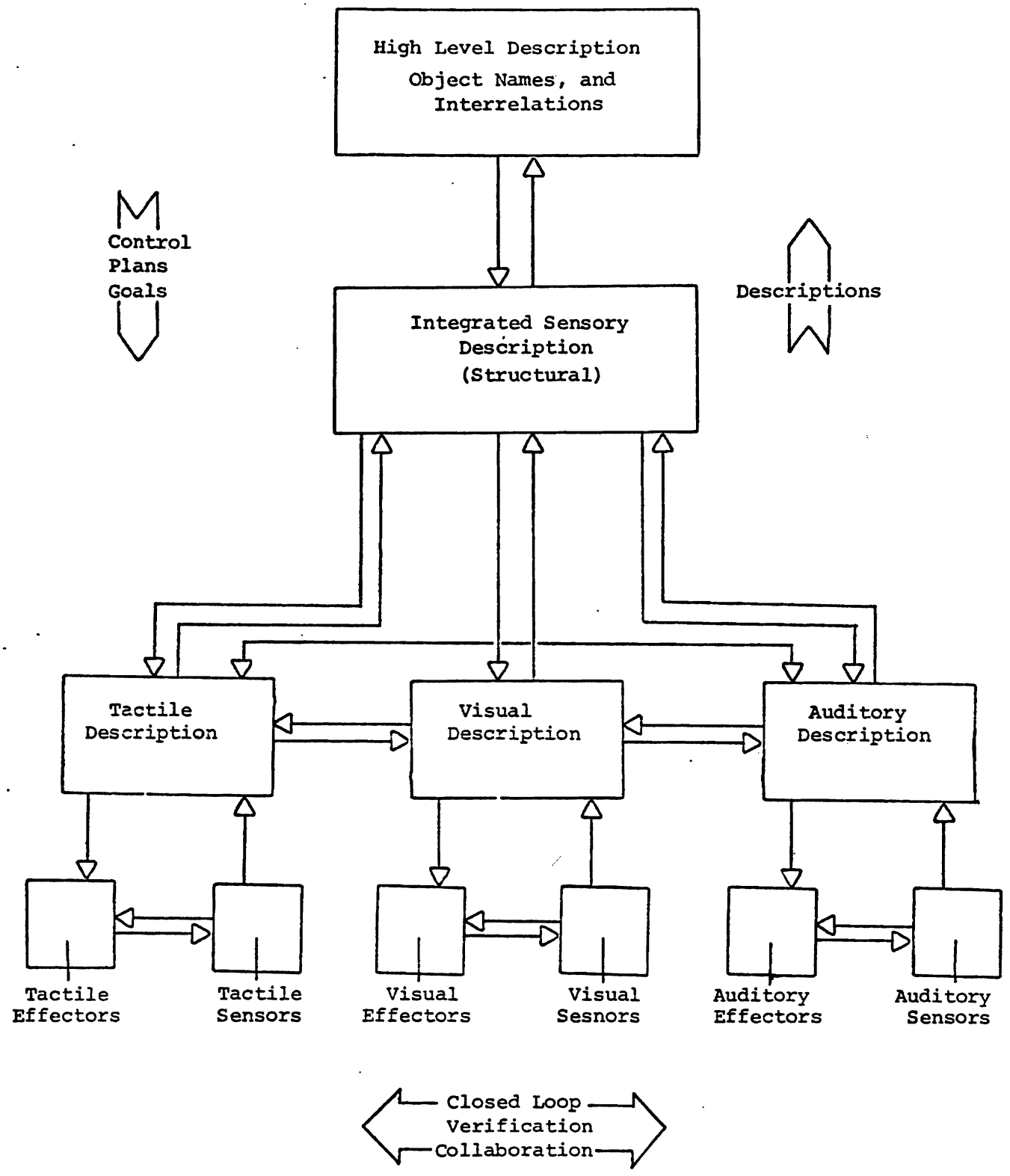

High Level Description
Object Names, and
Interrelations
Control
Plans
Goals
Descriptions
Integrated Sensory
Description
(Structural)
Tactile
Description
Visual
Description
Auditory
Description
Tactile
Effectors
Tactile
Sensors
Visual
Effectors
Visual
Sesnors
Auditory
Effectors
Auditory
Sensors
Closed Loop
Verification
Collaboration

Figure #4b

3.6 Pertinent Principles

In light of the architecture and ideas described, and in summary, we list the following pertinent ideas :

Separation of Control, Goals, and Plans and its Affects

+ Division of the Responsibility of Control - Distributed Control
+ Levels of Knowledge, Control and Representation
+ Modularity
+ Parallelism
+ Cooperation and Collaboration
+ Intrinsic Verification as a result of multi-sensing

Manifestations of the Overall System

+ Procedures are Directed by Plans and Goals
+ Near-Autonomous Task Specification within Virtual Machines
+ Sensor Adaptive Control - Dynamic Response to What is Observed
+ Sensors Acting as Low Level Specialists
+ Upward Abstraction in the Representation of Information
+ Filtering-out of Unnecessary Low Level Information as it moves Upward

Requirements of Partitioned Control

+ Unified Operating System
+ Distributed Executive

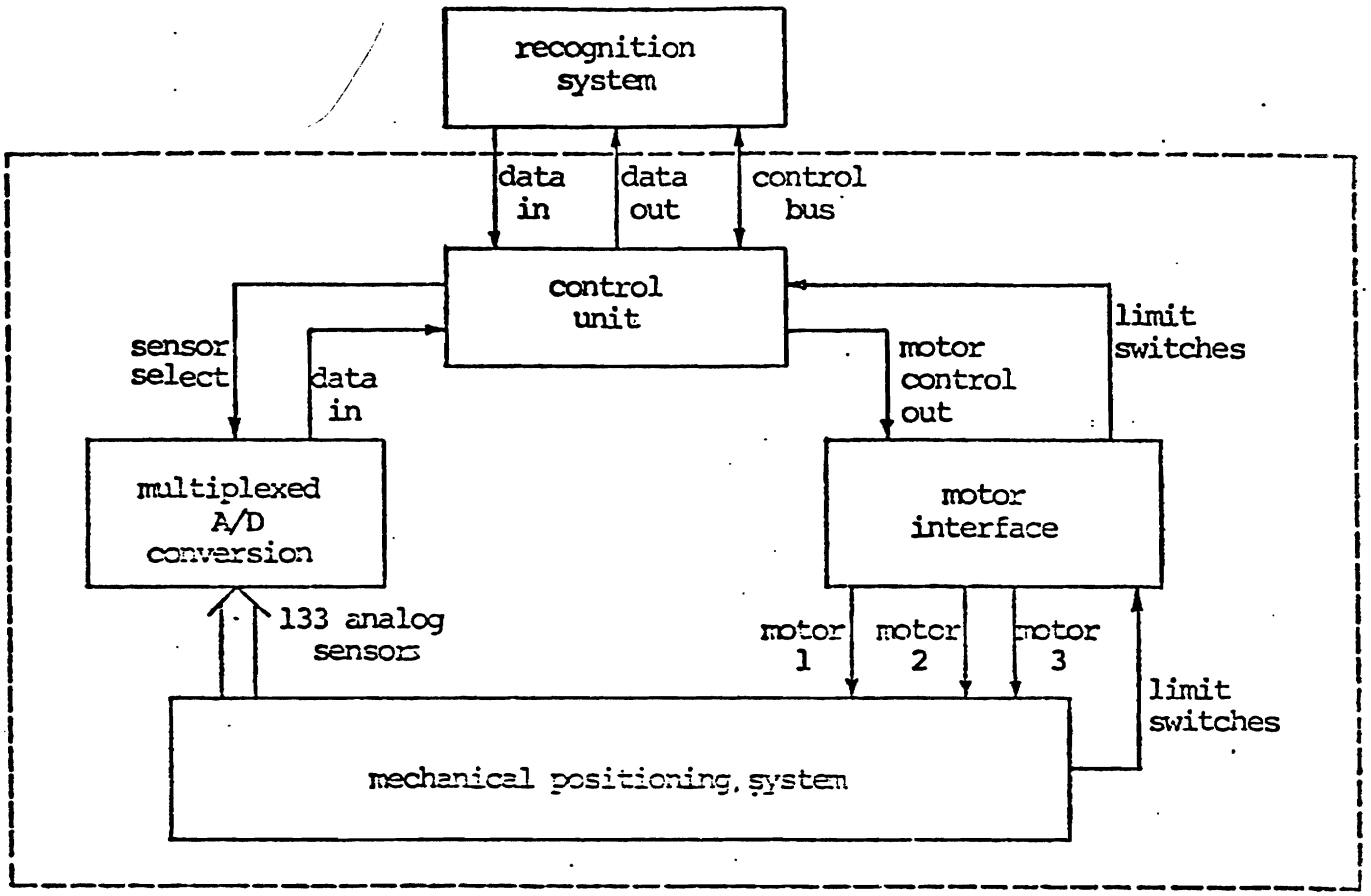
The importance of these notions should be evident from foregoing discussion, and are the motivational bases for our system. Furthermore, many of these notions are discussed from the standpoint of their implementation in the Tactile Sensing System (see Part II).

Part II
Implementation of the Tactile Sensing System

Now that an exposition of general motivations and design strategies has been given, we turn our attention to the implementation of the tactile sensing system itself. As has been previously discussed, one thematic strategy in our design is to implement a hierarchy of virtual machines which in conjunction form a system that performs object recognition. The tactile sensing system implements one such virtual machine.

I. Overview

The block diagram for the Tactile Sensing System is given within the dotted box of figure #1. It shows the major logical units of the system.



The Implementation of the Tactile Sensing System architecture includes hardware at three levels. From lowest to highest, these are as follows:

Level	
Mechanical	: Mechanical Device performing two primitive functions: 1) positioning/movement of the sensor 2) tactile (pressure, contact, texture) sensing
Interface Electronics	: Interface of Mechanical device to Digital control unit 1) provide digital interface to the motors which perform device positioning/movement. 2) provide digital interface to the analog tactile sensor
Digital Control	: Digital control system for tactile sensing device

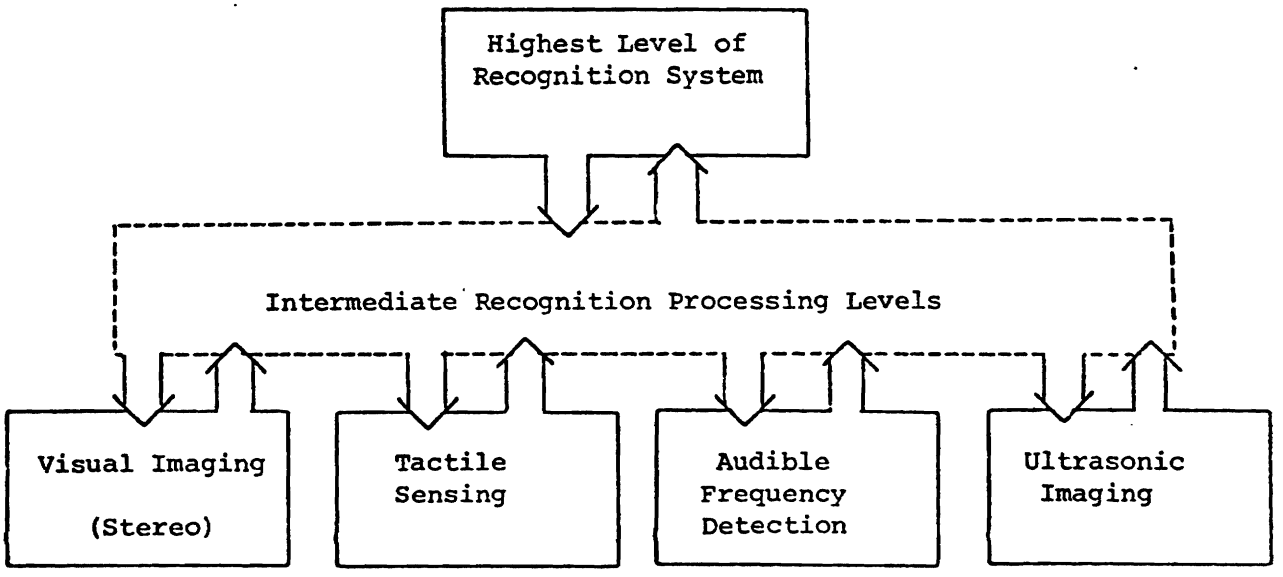
At the lowest level, we provide a mechanical system, which allows the tactile sensor to be positioned anywhere in a constrained three dimensional region (about 0.5 meter per axis) with a reasonable precision and accuracy (within 1.0 mm). The mechanical system's three degrees-of-freedom consist of orthogonal rectangular axes along which the arm containing the tactile sensor is moved by stepping motors.

At a conceptual level directly above the mechanical system and sensors, are two units that function as the interface between the mechanical system and the control unit. The first device, the Motor Interface Unit (MIU), provides all the electronics necessary for control of the motors which position the tactile "finger". It also provides interrupt driven end-of-travel detection on each of the three axes of motion. The second device, the Tactile Interface Unit (TIU), supplies the analog to digital conversion for the interface to the 133 piezo electric pressure sensors on the "finger".

Finally, at the architectural apex of the Tactile Sensing System, is the Contol Unit. It provides the digital control of all that is below it in the system hierarchy. It is comprised of two processors, one dedicated to each of the subsystems just described, with internal interprocesor communication for closed loop operations between motor control and tactile sensing. The Control Unit also provides for the Tactile Sensing System's interface to a higher level system. Plates 1 through 3 are various photographs of the TSS elements.

1.1 The Tactile Sensing System Placed in Perspective

The Tactile Sensing System as a whole (subsequently referred to as TSS), is ultimately intended to perform as a limited capacity subsystem in a more powerful architecture employing several such sensory subsystems. Such a system is illustrated in figure #2 below.



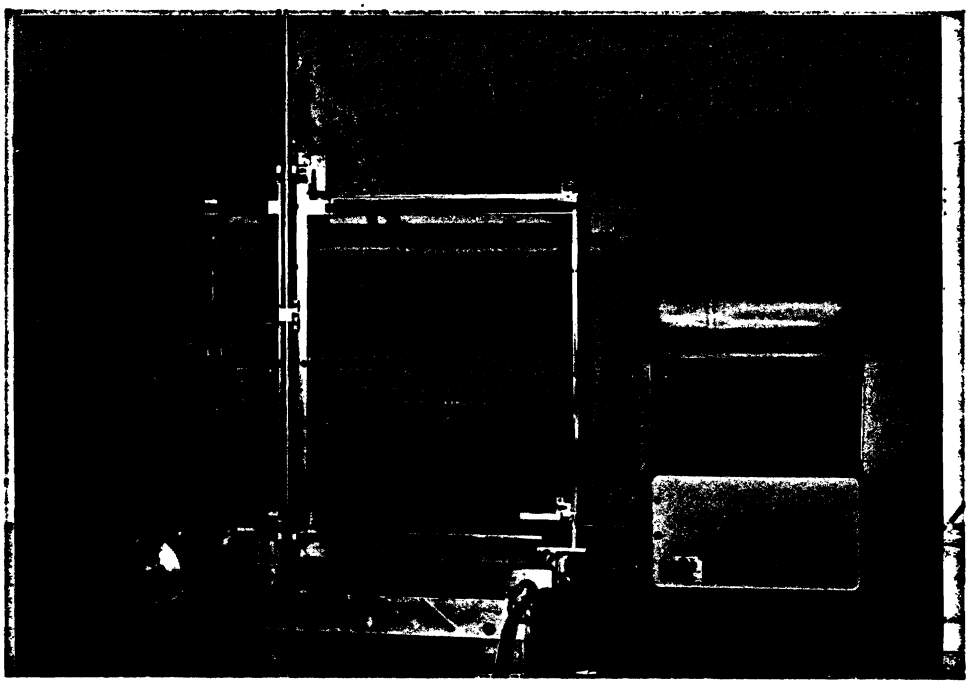


PLATE I - The Tactile Sensing System

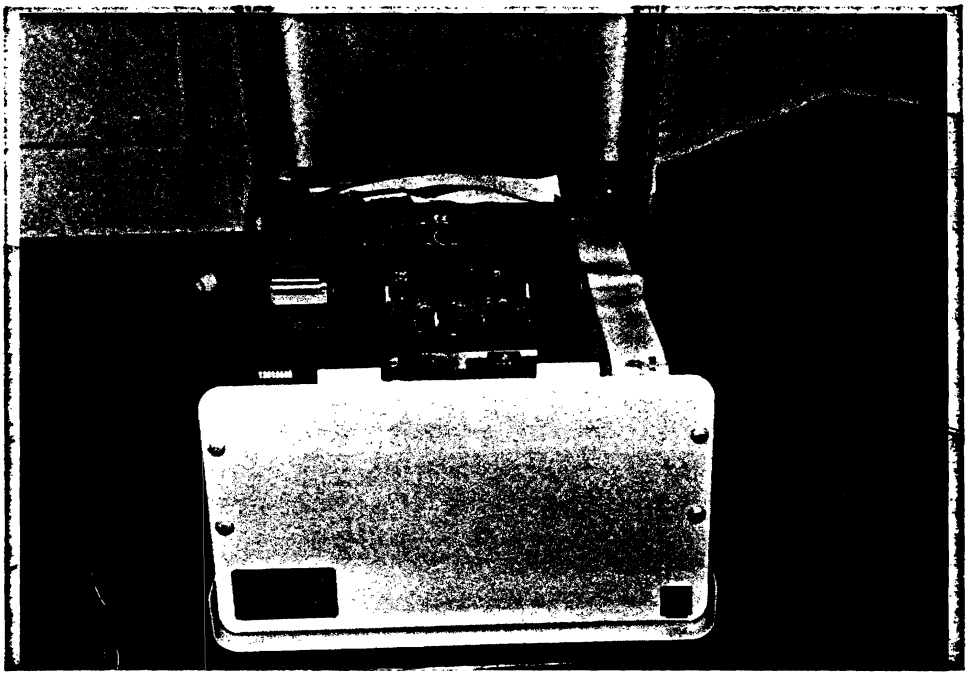
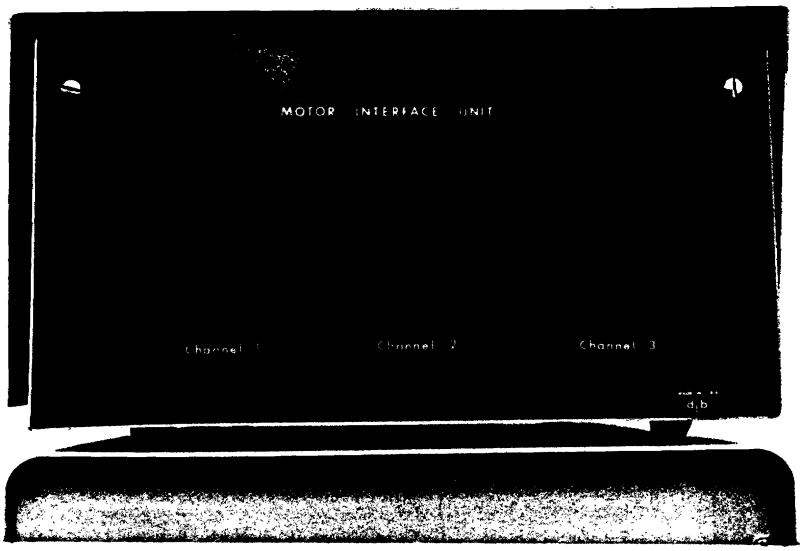


PLATE II - The Control Unit



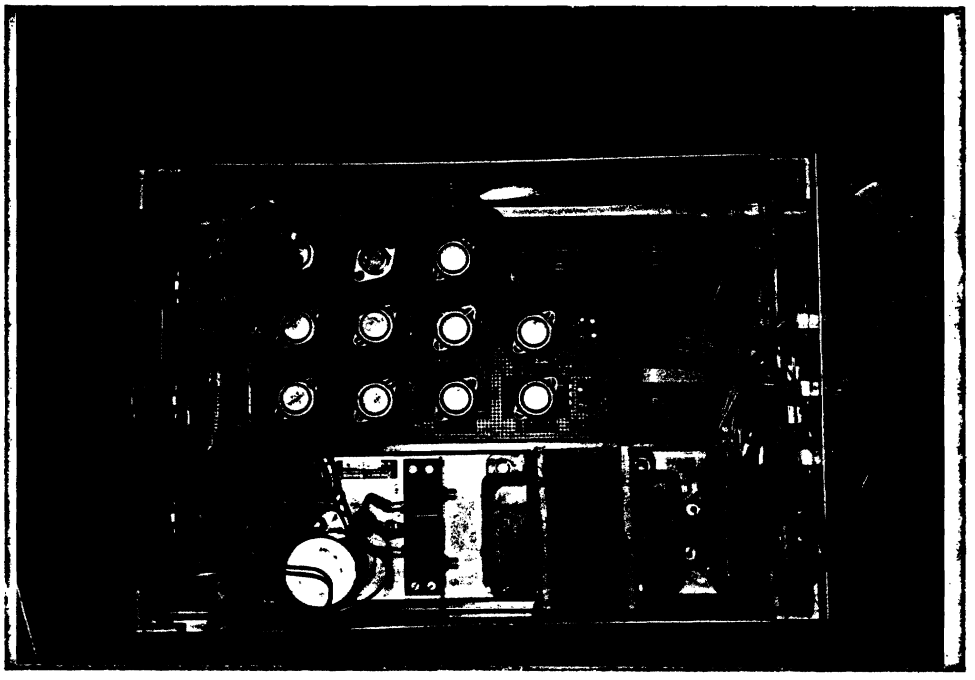


PLATE III - The Motor Interface Unit

With respect to its position as a specialized subsystem of a larger ORS, the design of the TSS intends to afford limited device-relevant intelligence. The TSS will perform the entire control task associated with the sensor, and will thus eliminate this responsibility in higher level virtual machines, but it will not be used to perform any of the recognition task of the higher level system. It should be clarified therefore, that in its intended functions of device associated control and sensory information acquisition, these two main objectives are served as follows:

1) Low level servoing of the sensor is performed. Namely, the tactile sensor is moved and positioned in accordance with the sensor's pressure response. This maintains an appropriate force of contact between the sensor and the object surface.

2) Primitives for both extraction of raw sensor data and communication of descriptive information are provided.

The first objective allows the TSS to autonomously perform such functions as tracing cross sectional contours, following edges, and the like. This is done by maintaining reasonable pressure of surface contact. The second facility serves the intention of gathering and filtering (refining, enhancing or noise-reducing) the raw data from the sensor, and communicating that information to its architectural superior in the recognition system.

1.2 Organization of Control

An additional point of architectural interest concerning the TSS, is the manner in which control is established. At the lowest level, control is carried out by actual hardware devices: Motors for positioning,

electronics for power amplification to drive these, based on TTL level input signals, piezo electric pressure sensors, hardware to perform analog to digital conversion, and so on. At the next level is firmware in which reside the most fundamental control routines associated with the hardware just mentioned. ROM (Read Only Memory) resident routines include such functions as movement algorithms and rate determinations for motor control, and similar low level routines for the tactile sensor. Also firmware-resident are primitives for communication, both to higher level processors, and for inter-task communication between tactile and motor related processes. The firmware may be considered as the microcode which implements the virtual machine (VM) at the sensor and sensor control level (the lowest levels depicted in figure #3 of Part I). The remaining ROM-resident routines implement remaining operating system components, such as conventions for task start-up, termination, and synchronization within the sensory level VM. At the highest level, the TSS maintains dynamic variables in RAM (Random Access read/write Memory). Dynamic variables are the basis for handling of interrupts, context switching, task synchronization (semaphores), and interprocess communication.

Prescript to the Technical Discussion

In the subsequent sections of Part II, a top down approach is taken in order to describe the major units of the system and their respective technical aspects. In succeeding sections, great effort is made to elucidate the functional behavior of the TSS in light of its architecture and organization, and the design aspirations of Part I. Care is taken to give sufficient attention to important principles without dragging the

reader's nose through extensively detailed technical muck. To this end, diagrams, embodied by textual commentary, are central. Lacking throughout will be jumper names and pin numbers, wire colors and connector names.

Section 2 describes the Control Unit; section 3 and 4 describe the Motor Interface Unit (MIU) and analog sensor interface (TIU) respectively; and section 5 details the mechanical system, motors, and sensors. In summary:

Section 2	Control Unit
Section 3	Motor Interface Unit
Section 4	Tactile Interface Unit
Section 5	Sensing and Mechanical Positioning System

2. The Control Unit

The Control Unit, architectural pinnacle of the TSS, manages all lower level TSS units and their activities. Additionally, it conducts sensor-derived information to higher level processors in the Object recognition system of which the TSS is a part. In order to effectively describe the Control Unit, we must describe its interaction with the external architectural environment as well as its internal nature. This will be done by describing the following Control Unit features:

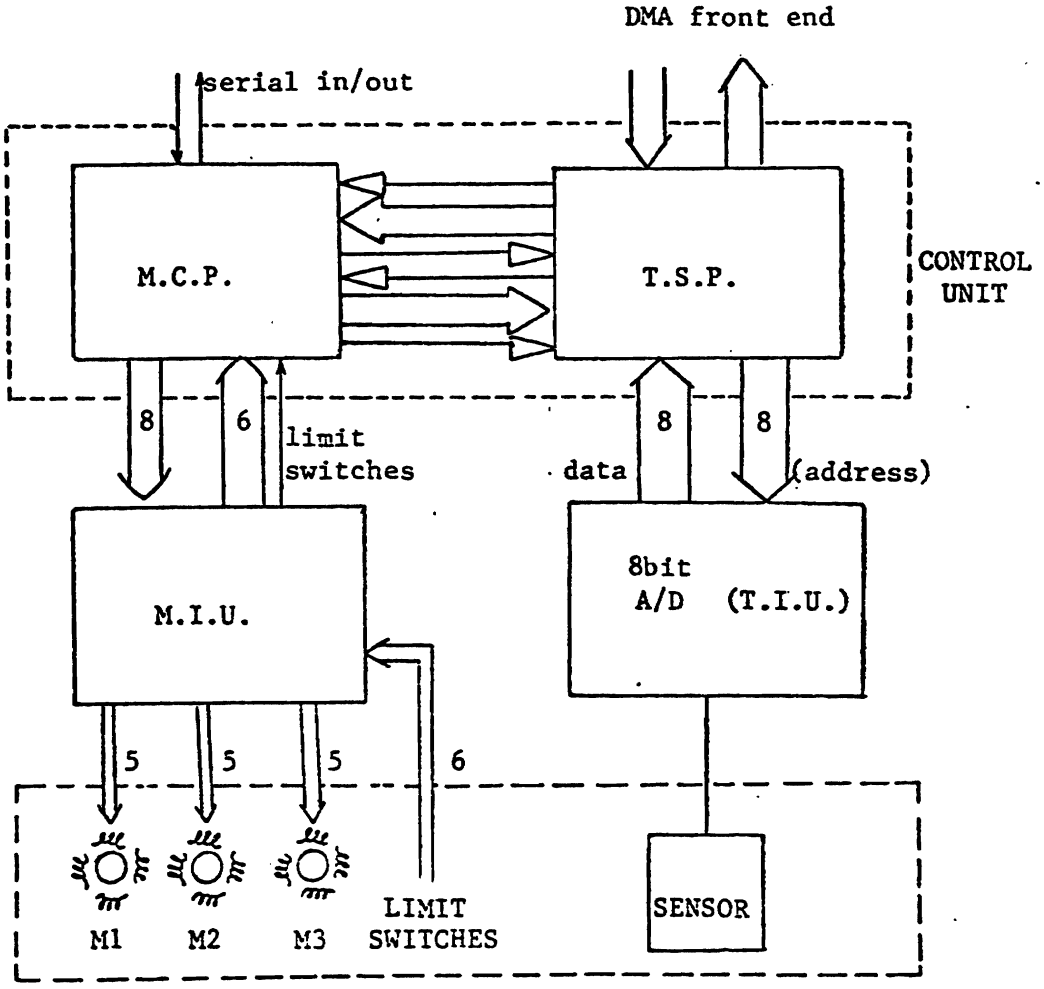
- Internal Nature of the Control Unit
- Interface to Higher Level "Host" Processor
- Interface to Lower Level TSS Units

2.1 Inside the Control Unit

Internally, the CU is comprised of two logically distinct processors. The specific intention of this dual processor realization was to allocate independent control to the two TSS subfunctions of motor control (movement and positioning of the sensor), and sensor control (retrieval and low level processing of the sensor data). It was decided that there was both sufficient work associated with each of these subfunctions to allow each its own processor, and that these subprocesses were sufficiently independent to make decentralized control a practical choice.

Architecturally, the CU may be described as a loosely linked system of two Z-80 based microcomputers. Each computer has an individual local bus along which its local memory and input/output devices reside.

Interprocessor communication is achieved by means of interrupt driven parallel i/o. The control unit is seen as the dotted box at the top of the TSS illustration below.



2.1.1 The Z-80 Microprocessor

The Z-80 microprocessor is a late second generation MOS LSI (Metal Oxide Semiconductor, Large Scale Integration) 8-bit processor. It supports three modes of maskable interrupts (including vectored interrupts — Mode 2), non-maskable interrupt (NMI), a 256 byte i/o space, 64K byte address space, and a reasonably powerful instruction set (158 instruction types) including block move, block comparison, and block i/o, among others. Secondarily, in house systems employing the Z-80 were available, in

addition to a vast quantity of proven-quality software usable for TSS system development work. These were the bases of the choice of the Z-80.

It is possible that a third generation microprocessor such as the Intel 8086, Zilog Z-8000, or Motorola 68000 would have been architecturally salient choices also. We selected not to use these for several reasons however. First, some of these are presently leading edge components, or close to it (earliest of these was the 8086 released in 1978). This means that the use of one of these elements will likely bring the usual demise associated with state-of-the-art design. These problems typically include potential short supply of support family components (e.g. the Z-8000 family MMU, and the relative non-availability of any of the 68000 series elements), and component design flaws which have not yet been totally resolved. More importantly, development systems, pre-built boards employing these elements, and especially good quality software suited to development needs, often lag parts availability by quite some time. As a tertiary concern there is the increased direct-cost (that not associated with the above problems) of these components.

The Z-80 has adequate capability to handle the requisite tasks of this system, and does not require extensive support hardware for a small or near-minimum size system (limited memory and i/o) for the type of dedicated control application intended in our Control Unit.

Two Single Card Computer (SCC) boards were procured from Cromemco Inc. to serve as the two Control Unit processing elements. The Cromemco SCC has 1K bytes of RAM, space for up to 8K bytes of ROM, PROM or EPROM (Intel 2716 compatible), input/output consisting of one serial and three 8-bit parallel ports, surrounding a Z-80 processor. Also available are 5 settable interrupting timers within an on board LSI component (TMS-5501). Figures

#4 is the functional block diagram of the SCC

2.1.2 Control Unit Interprocessor Interface

Although the Cromemco SCC provides an S-100 bus interface, and it would have been possible to establish interprocessor communication via a shared memory along this bus, this was not done. Rather, interrupt driven parallel i/o was chosen. This choice was made so that the S-100 bus would be not become a bottleneck. This would be the case when the two processors also use the S-100 to access other i/o and memory devices residing there, in the performance of their local tasks. Presently, no such facilities are used on the S-100 by the processors in the performance of their local control tasks, but in case of a future memory or i/o expansion need, a private S-100 is now available to each and there is no concern of contention over a shared S-100 bus. Secondly, the nature of the interprocessor communication was such that a shared memory was not necessary. Large blocks of information are not passed in a single communication, and further, the messages passed between Tactile Sensing Processor (TSP), and Motor Control Processor (MCP) require high priority service. Interrupt driven parallel i/o is well suited to the type of messages in this communication path — namely, short ones requiring near-immediate (high priority) service.

The essentiality of communication between CU processors is motivated by a need to convey sensor pressure information from the Tactile Sensor Processor (TSP) to the Motor Control Processor (MCP). The TSP must convey pressure sensor values so that closed loop control may be carried out for positioning of the sensor. With this communication, the MCP may adapt to

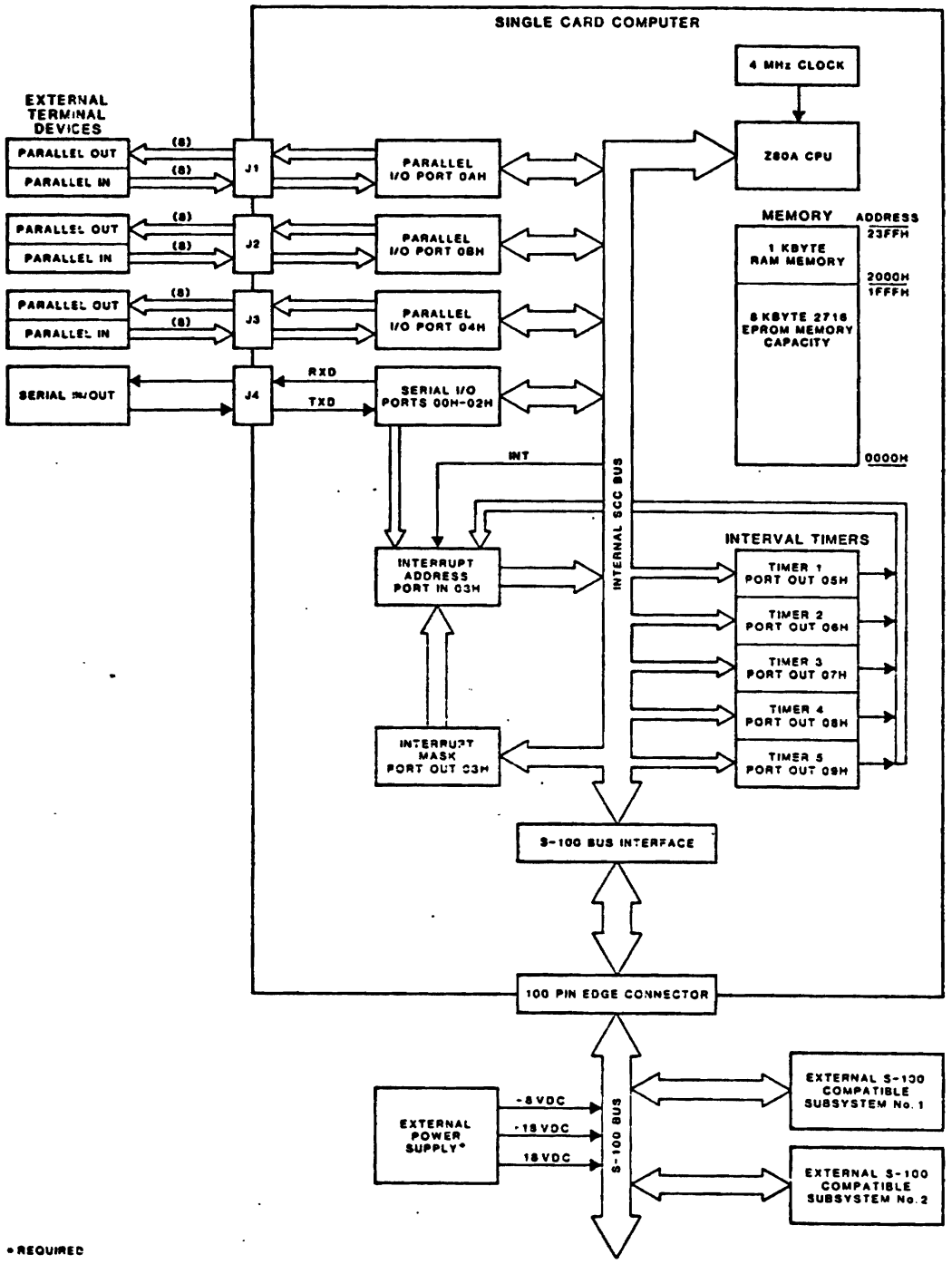
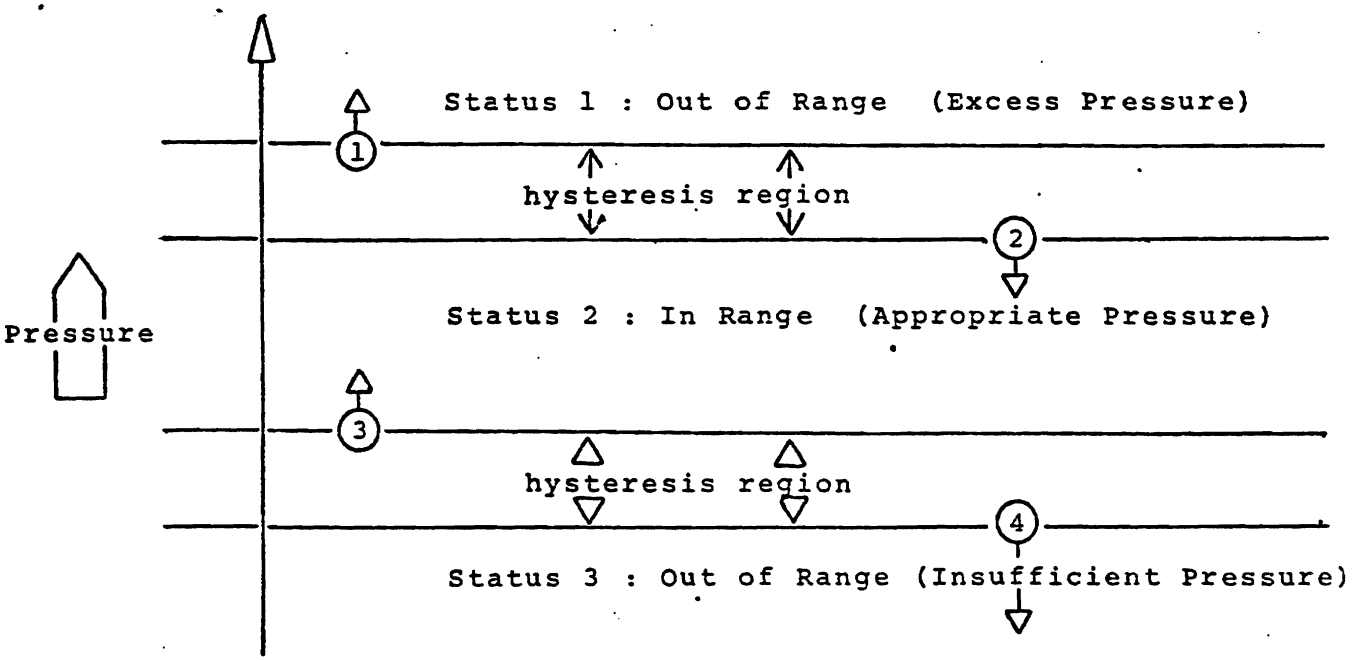


Figure 4 SCC FUNCTIONAL BLOCK DIAGRAM

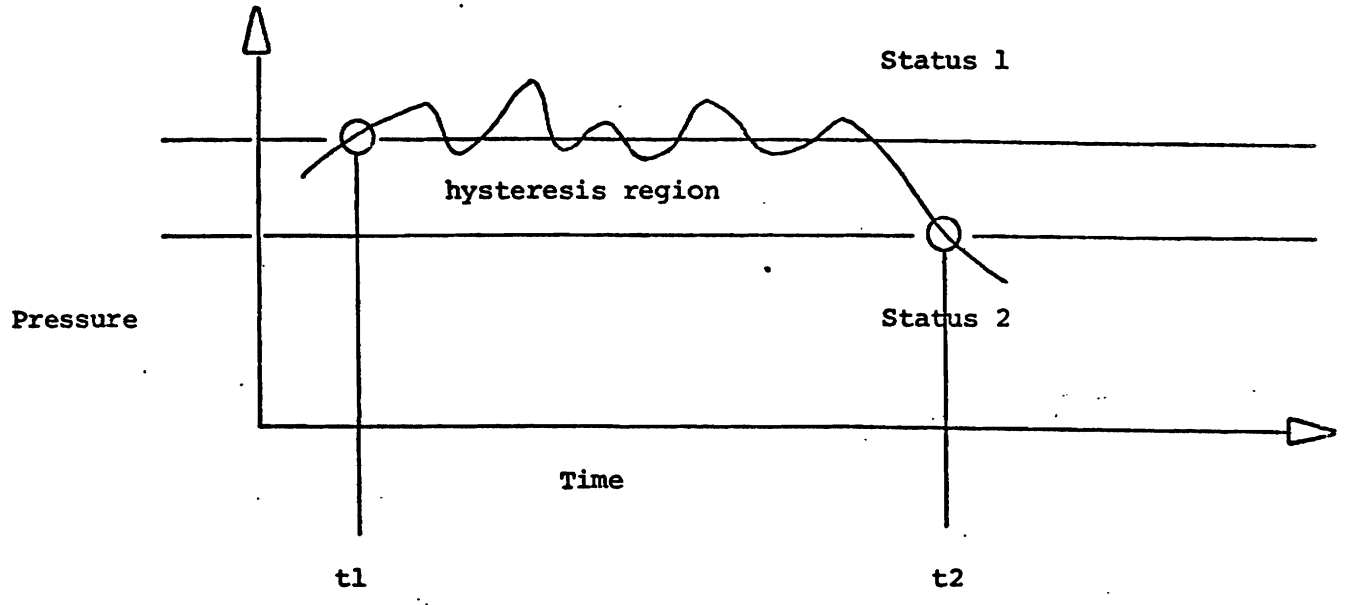
responses from the TSP, moving the sensor into the object surface when pressure falls too low, and away from the surface when pressure rises too high. In this fashion, proper intimate contact is kept with the object surface as it is being examined by the sensor. A scheme has been devised involving pressure threshold reports by the TSP which allow the MCP to make appropriate sensor position adjustments. The scheme is diagrammed in figure #5.



The scheme involves a three-valued system which indicates out-of-range (pressure in excess), in range (proper intimate pressure), or out-of-range (pressure is insufficient). Notice that there are four boundary transitions however, rather than the two which might have been expected at first. This is as a result of a hysteresis region which has been built into the scheme. The hysteresis is intended to prevent a large amount of reports from being sent by the TSP while the sensor pressure is near a threshold.

To see briefly how this works, let us examine transitions (1) and (2)

from figure #5. The TSP does not report status (1) — pressure excess, until pressure exceeds threshold (1). Now the MCP compensates with a movement of the sensor away from the object's surface. As the sensor moves away, pressure drops, but status (2) — pressure in range is not reported until pressure falls below threshold (2). This response will prevent a jittering of the pressure value near boundaries (where no hysteresis region exists) from causing a flood of reports: excess pressure; in-range; excess pressure; in range; ... and so on, as the sensor vacillates at the boundary. Rather than delimiting the boundary with a line, we give the boundary a width sufficient to eliminate this "noise" (see figure #6).



If P1 were the boundary between Status 1 and Status 2, a report would occur from the TSP at each crossing of the boundary. This would occur frequently as the pressure value "jittered" during the interval t1 to t2. With the hysteresis region however, a slight sensor noise is prevented from

causing this constant reporting, and it is not until pressure falls below P2 that status 2 is reported.

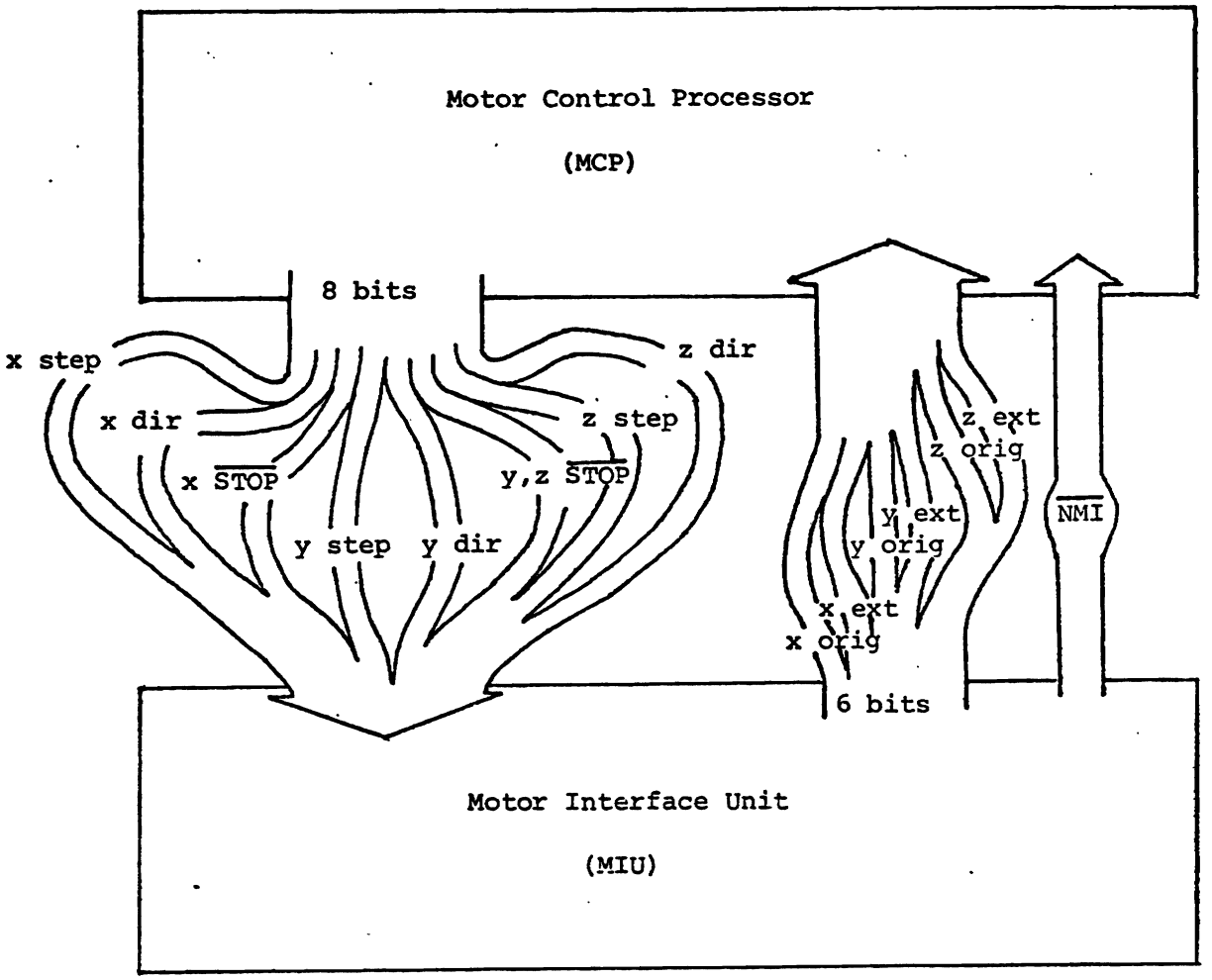
2.2 Interface to MIU and TIU

As previously introduced, the Control Unit directs the function of lower level TSS units in order to carry out the system's functions. The MCP is directly responsible for the control of sensor movement and position, and as such it sends commands to the Motor Interface Unit (MIU). The TSP, which has responsibility for information acquisition from the tactile sensor, performs a similar task through association with the Tactile Interface Unit (TIU).

2.2.1 MCP - MIU Interface

The Motor Control Processor has three stepping motors at its disposal in order to effect sensor movement and positioning. The motors each handle one orthogonal axis of motion (x-lengthwise; y-widthwise; z-depthwise), and are operated in an open loop fashion. That is, step commands are sent to the motors via the MIU, and resulting shaft position of the motors is not checked. The MIU provides all the electronics necessary for running the motors, and passes back the values of six optical limit switches. The limit switches each detect whether the positioner is at an extreme of an orthogonal axis of motion (x-origin, x-extreme, y-origin, y-extreme, z-origin, or z-extreme). The resulting hardware interface from the MCP to MIU is 8 bits which conduct commands to the three motor channels, 6 bits from the MIU returning the values of the limit switches, and non-maskable interrupt (NMI) from the MIU to the MCP. An NMI is generated by the MIU

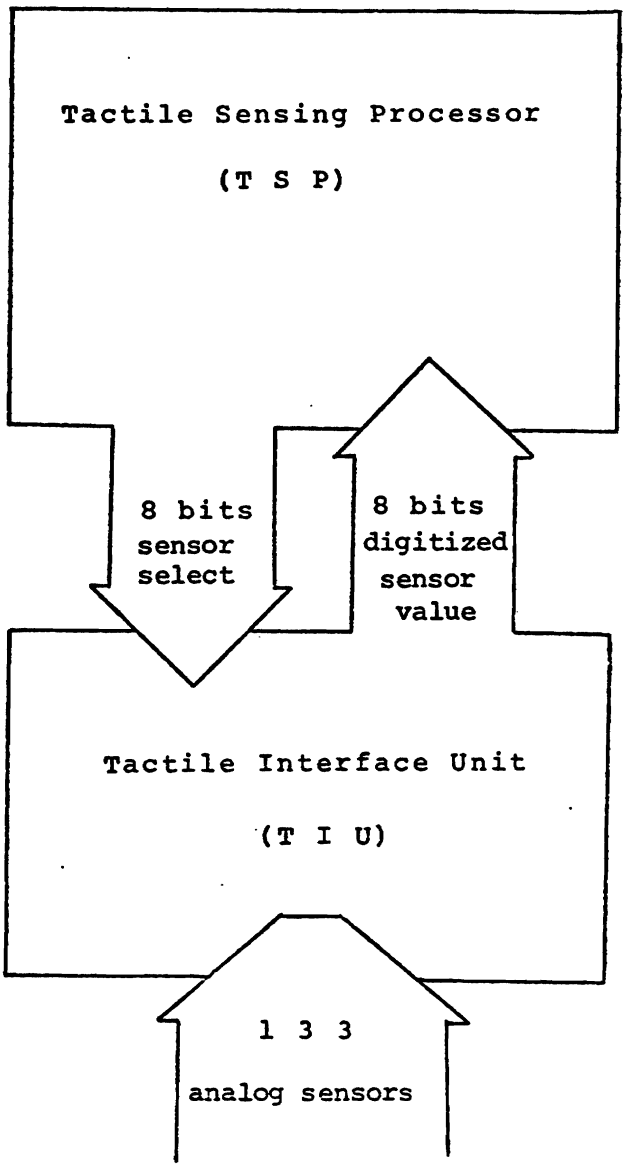
whenever any limit switch changes value. This forces the MCP to examine the switches and take the appropriate action when a mechanical boundary is reached. Figure #7 illustrates the interface.



2.2.2 TSP - TIU Interface

Similar to the foregoing discussion, the TSP controls the sensor's response via the Tactile Interface Unit (TIU). The TIU is simply a multiplexed analog to digital converter, allowing the TSP to select any one of the 133 sensors on the "finger" and digitize the analog pressure

reading. The resulting hardware interface consists of 8 bits to the TIU for sensor selection, and 8 bits from the TIU, which return the digitized pressure value of the sensor selected. The interface is diagrammed in figure #8.



2.3 Interface to the "Host" Recognition System

The TSS, as intended, is an intelligent sensor-dedicated system, to be

used in conjunction with some higher level object recognition system. We will term such an ORS, a "Host" system (as was shown in figure #1). At the time of this writing, we have delegated a PDP 11/60 in the capacity of this "host". We have designed that two interfaces exist between the TSS and the host system, one from each of the MCP and TSP, in order that information pertinent to the recognition task be conveyed.

2.3.1 MCP Interface to Host System

The motor control processor will be the recipient of commands to move, scan, trace object cross sections and the like, from the host system. Communication from the MCP to the host, on the other hand, involves a reporting of points by the TSS, describing the movement performed in response to the command sent by the host.

In order to support this interface, a ROM resident command interpreter will exist. This firmware is the work of Wolfeld (U of P, 1981), and is described therein. It has been decided that a serial interface (RS-232) will exist as the hardware between the MCP and host.

2.3.2 The Tactile Sensing Processor to Host Interface

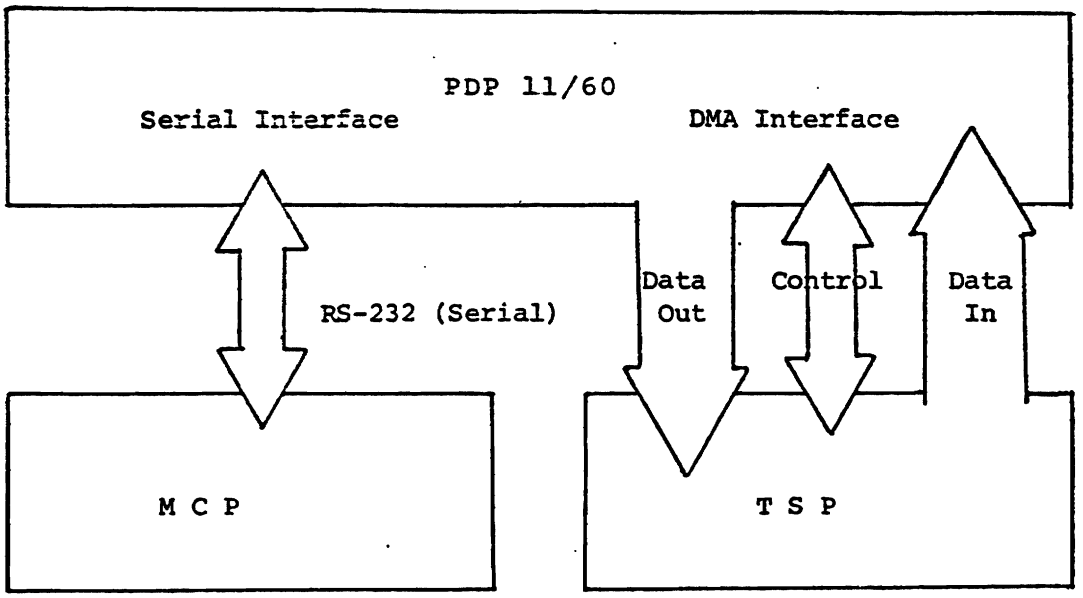
The Tactile Sensing Processor is responsible for reporting surface perturbations (textural information) to the host, as objects are examined by the tactile sensor. Similar to the MCP, the TSP will have a ROM-resident command interpreter, allowing tactile primitives to be instantiated by a higher level processor in the ORS. Primitives resident

in the TSP will be primarily associated with data acquisition for texture determination, and various types of filtering or low level signal processing routines. This development is also described in Wolfeld (1981).

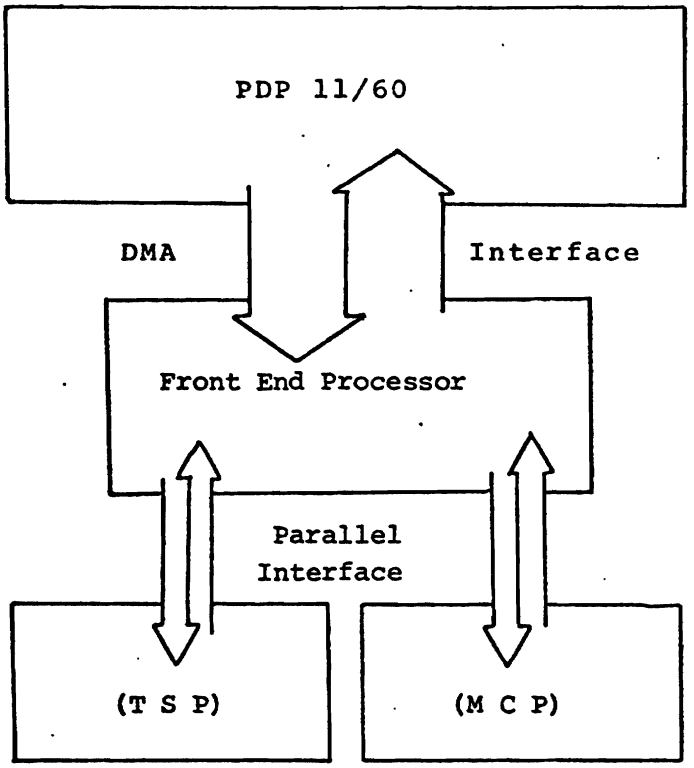
In order to support communication between TSP and host, it has been decided that a DMA unit will be used on the PDP 11/60. This was chosen in order to support the large amount of information that is expected to move across this path. Without DMA, a large demand would be placed on the 11/60 processor in order to handle incoming data, and the machine will be I/O bound.

2.3.3 The PDP-11 DMA and Host Interface Enhancement

The present hardware configuration for TSS/host (11/60) interface is illustrated in figure #9.



Future performance benchmarking may illustrate that a large processor demand exists due to the serial MCP-host interface. For this reason, it may be reasonable to place a "front-end" machine on the TSS. This configuration is illustrated in figure #10.



The front end machine will be a channel, handling I/O between TSS and host. The advantage of such a front end is that the 11/60 DMA may be shared by both MCP and TSP, in the sense that both will interact with the host indirectly (through the channel).

With the proposed front end, the MCP will place no demand on the 11/60, and the TSS I/O is effectively "transparent" to the host machine. The DMA accomplishes communication by direct access to the 11/60 memory, and is co-resident with the processor on the 11/60 bus. The DMA is thus sharing the bus with the processor. Processor and DMA have interleaved bus access, with one locking-out the other when there is conflicting demand for access. The DMA gains access during the processor's instruction-decode and other CPU internal periods, cycle stealing as necessary to complete a memory read or write operation. This is given full detail in the DEC PDP-11 DR11-B/DS11-B reference manual, theory of operation.

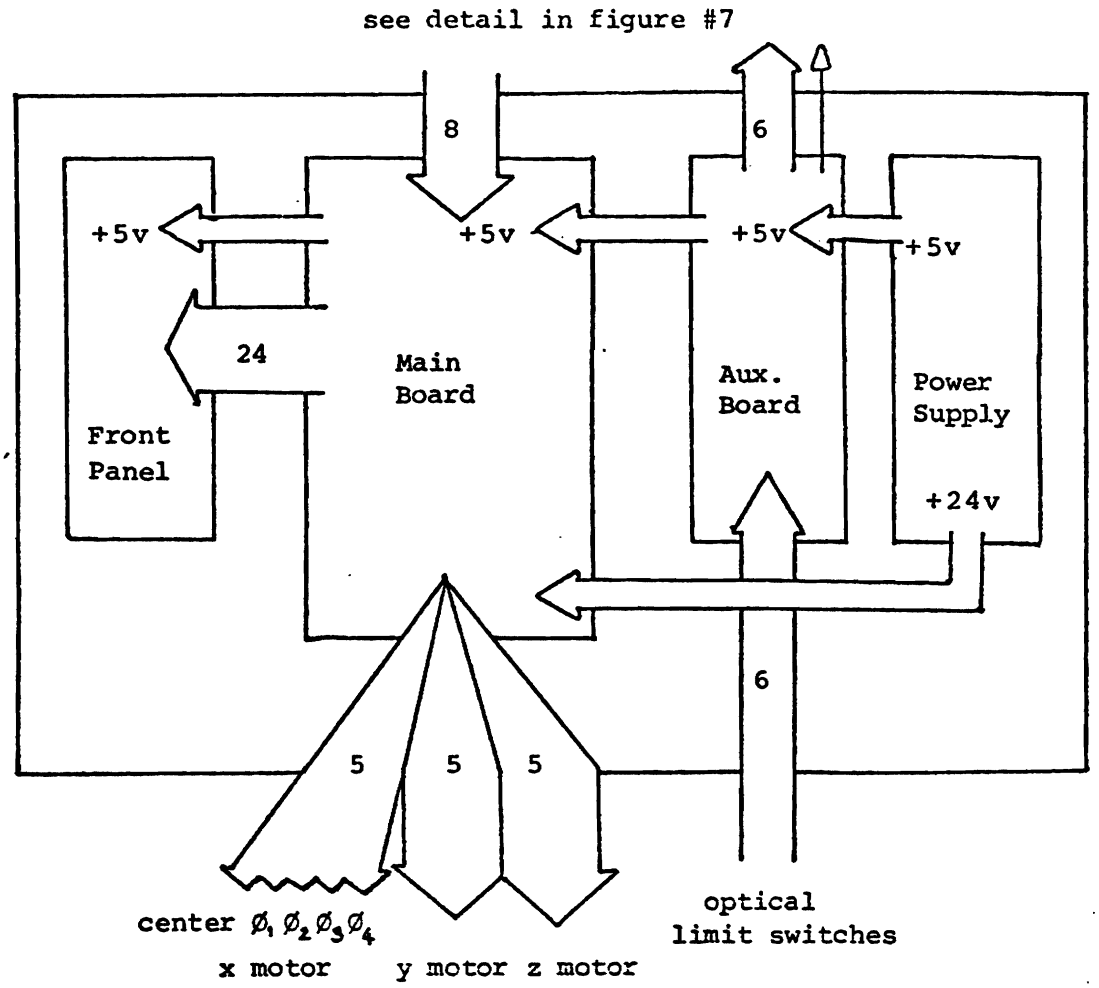
3. The Motor Interface Unit

The motor interface unit is one of the two units providing interface between the control unit above it (digital control level of the TSS) and the electro-mechanical and sensor level below it. The MIU is directly responsible for the control of motor operations (movement and positioning) going on at the mechanical assembly level. It contains a power supply and all the electronics necessary to accept digital (TTL compatible) motor control commands from the CU. The MIU also monitors 6 optical limit switches which detect the end-of-travel of either the x, y or z carriages. A change of status on any of the six limit switches detects a mechanical end of travel, and the MIU generates a non-maskable interrupt to the control unit's MCP, thus signalling the MCP to bring the mechanical assembly back into the constrained sensing region.

The MIU contains four physical units, which accomplish the aforementioned functions. They are:

* main board (three channels to motors)
* auxiliary board (limit switch monitoring and NMI generation)
* front panel (indicators for motor status)
* power supply (power source for motors and MIU digital electronics)

A macroscopic diagram of the MIU, giving attention to top-end (control unit) and bottom-end (mechanical assembly) interfaces is shown in figure #11.



3.1 MIU Main Board

The main board is comprised of three channels, each responsible for the control of one stepping motor on the mechanical assembly. Three digital signals enter each channel, STEP, DIR, and STOP, for this purpose. The step signal is a positive true pulse causing the motor to make one quarter of a revolution in either clockwise or counter-clockwise direction as determined by the DIR signal (1 = clockwise, 0 = counter-clockwise).

Each channel consists fundamentally of two parts, digital logic converting pulses on the STEP line, into step code for the motor windings,

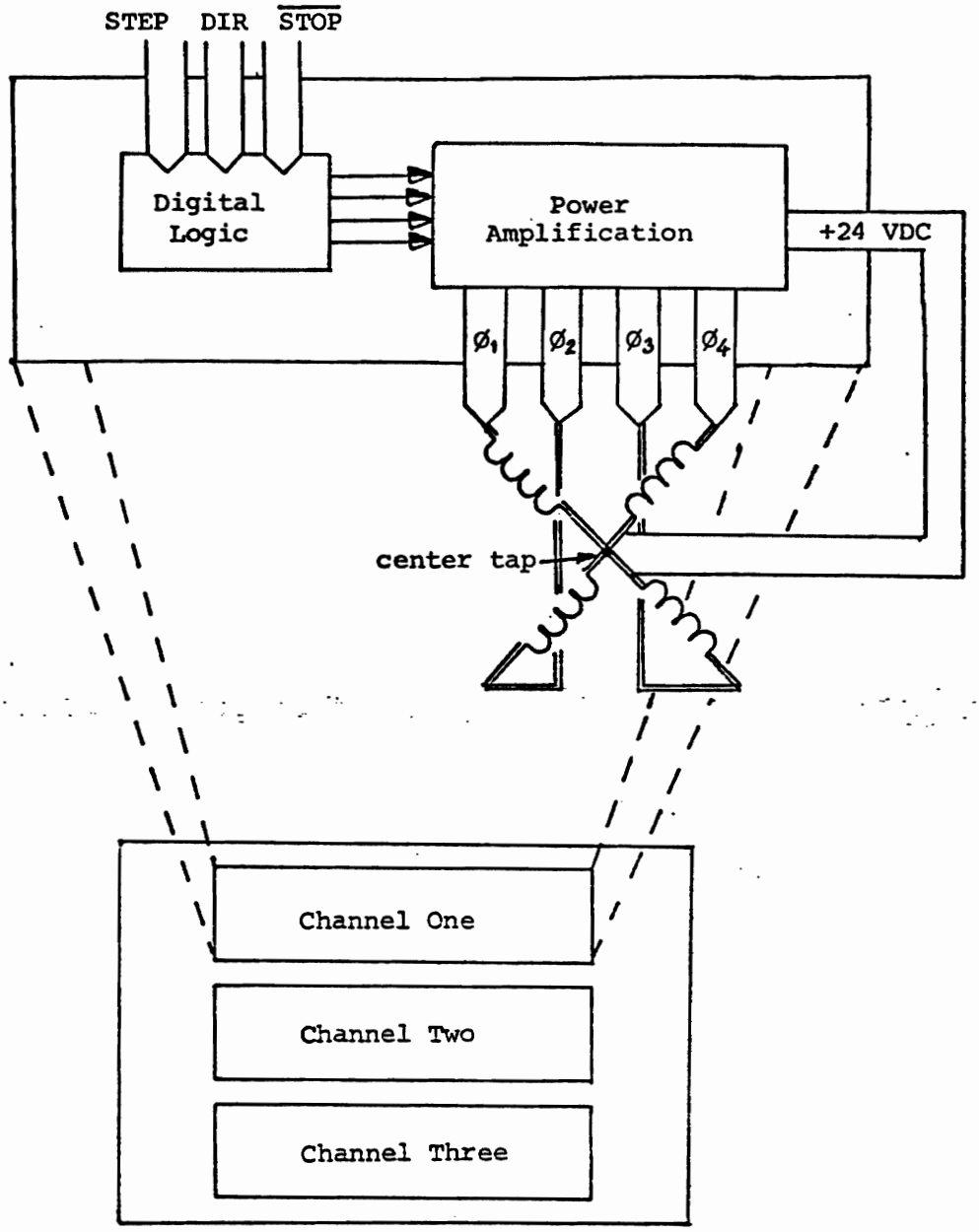
and a power amplification stage bringing the TTL level winding code up to +24v at several amps for each motor. The third input signal, STOP, is used to disable the power amplification stage when the motors are not running. This prevents the dissipation of power in the motor when it is stationary. The block diagram for the main board is shown in figure #12 and the schematic for one channel is given in figure #13. Section 3.1.1 discusses stepping motors, and the generation of appropriate control signals in the MIU main board.

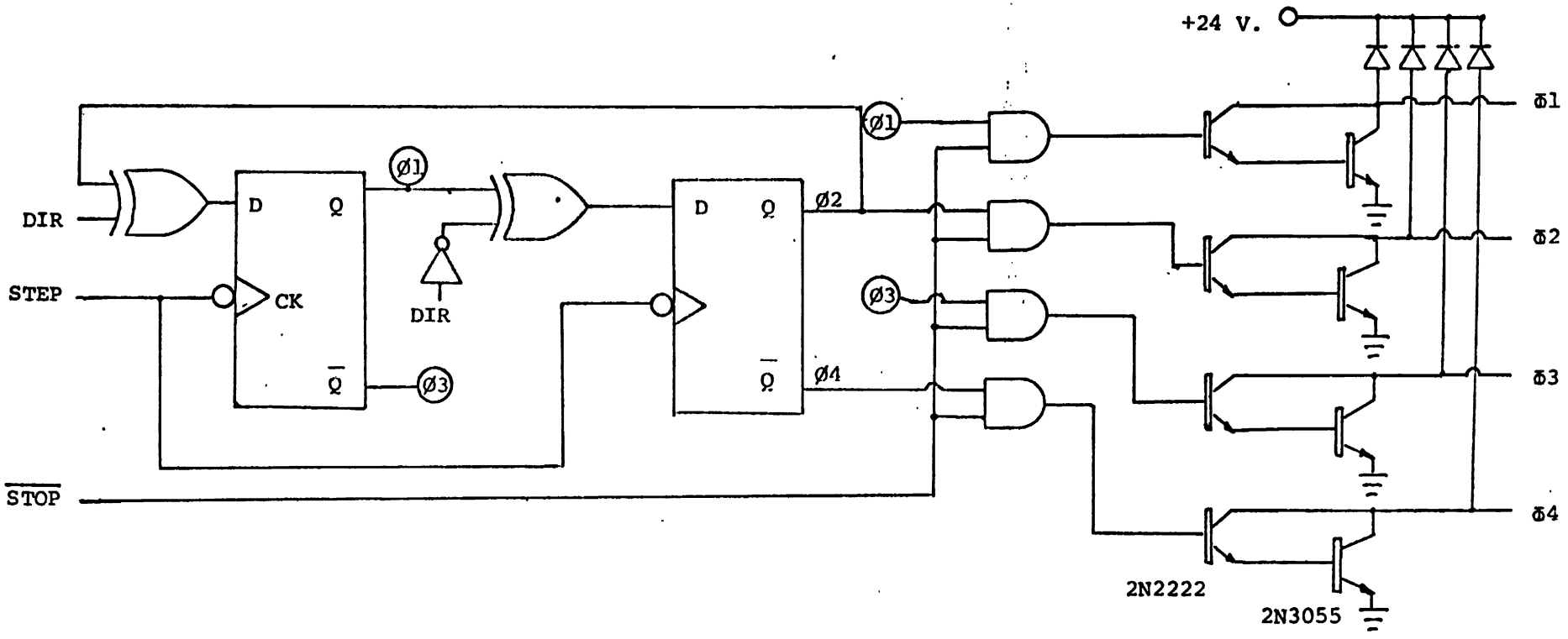
3.1.1 Stepping Motors - A Brief Introduction to Function and Use

In control applications such as the tactile system we wish to have a reasonably precise positioning capability. What is required is some type of motor which we can cause to move our apparatus with a known displacement. We may employ any of a variety of motors, including servoed DC torque motors, AC induction motors, synchronous permanent magnet field motors and others. The advantage of stepping motors over other types is that they have the ability to start and stop at various mechanical rotational positions, or "step". Each time that the motor receives a "step" it moves one such position, or a fixed fraction of one revolution. Stepping motors can be purchased with various numbers of steps per revolution. Some common values are: 200, 180, 144, 72, 48, 36, 24, 12, 8, and 4 steps per revolution (Corresponding to 1.8°, 2°, 2.5°, 5°, 7.5°, 10°, 15°, 30°, 45°, and 90° per step respectively).

Thus the advantage that a stepping motor affords over other types such as the DC torque motor is that it may be run in an open-loop fashion - there need be no feedback information about the motor's rotational position.

(figure #12 - Main Bd block)




+24 V.
DIR
STEP
STOP
D
Q
CK
Q
Ø1
DIR
Ø3
D
Q
Q
Ø2
Ø4
Ø1
Ø3
Φ1
Φ2
Φ3
Φ4
2N2222
2N3055

To give a brief description of the manner in which stepping motors work, we can examine figure #14:

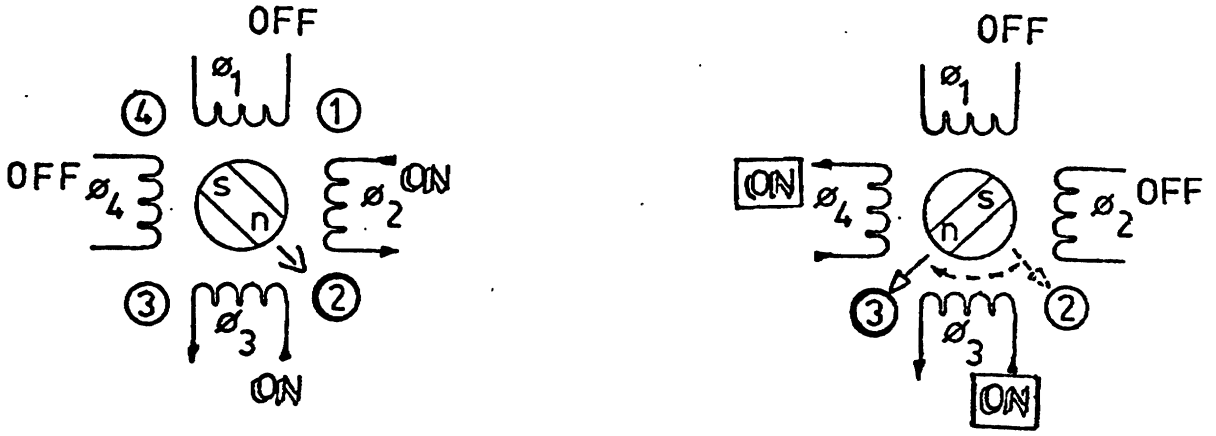
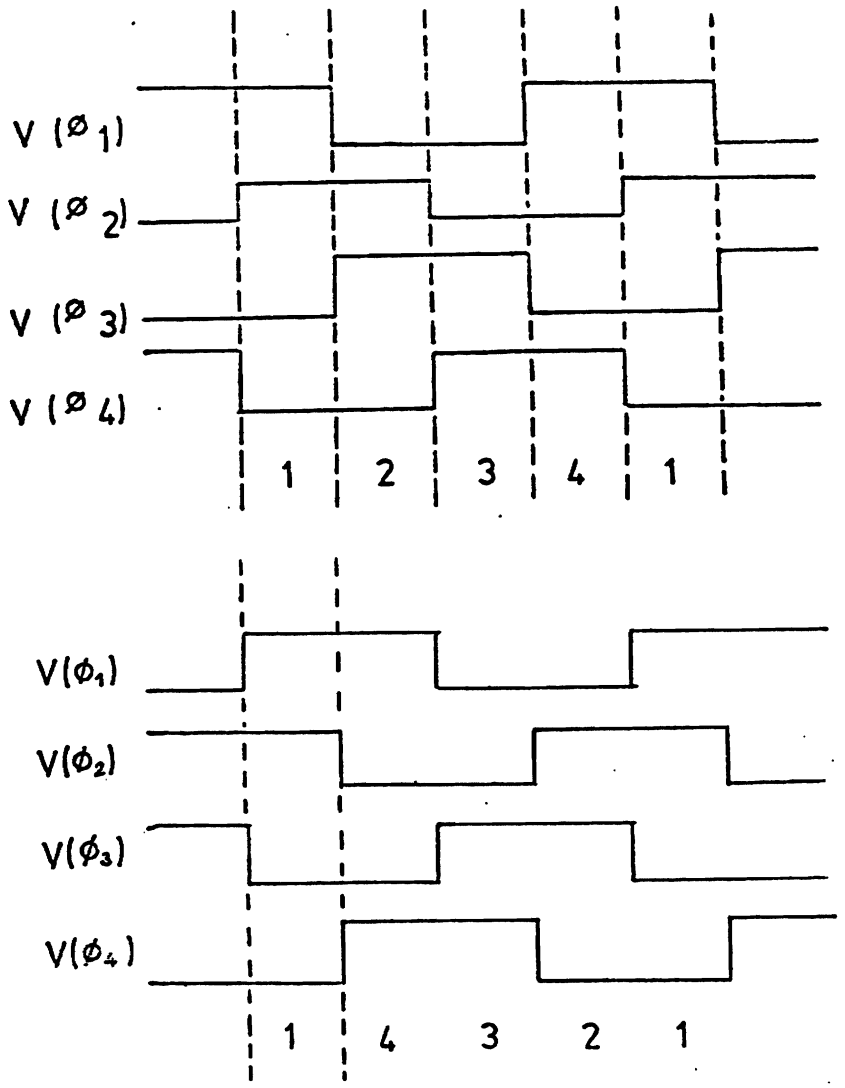


Figure #14 shows a four phase motor with four steps per revolution. Note that when current is run through a winding, the rotor is drawn into a position which brings the rotor's permanent magnet into magnetic equilibrium with an electrically induced field in the energized winding. If we wish to cause the motor to step in the clockwise direction, we simply energize the next winding in clockwise sequence (here ϕ_4 and turn off ϕ_2).

A pulse train which would cause the occurrence of one clockwise revolution then would look like Figure #15's illustration.

Notice similarly, that to run the motor in a counter-clockwise rotation we must simply reverse the sequence in which windings are energized. To move from position 3, back to position 2, for example, we simply de-energize ϕ_4 and energize ϕ_2.

Figure #16 illustrates a pulse train which drives the motor in counter-clockwise rotation.

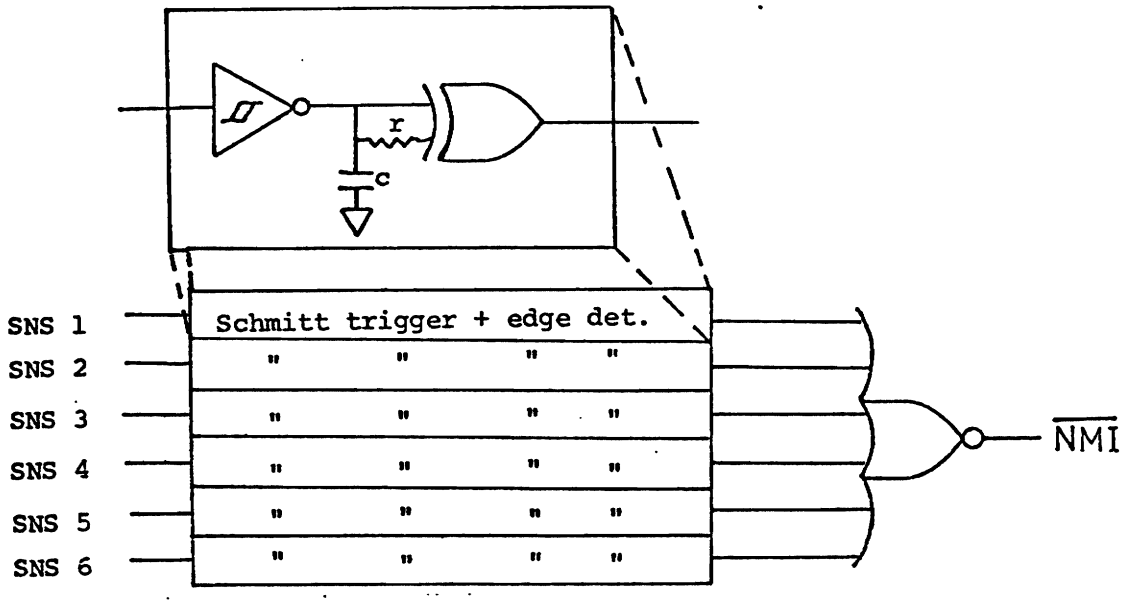


What has been discussed here describes only a restrictive subset of a much wider range of salient features of stepping motors. Specifically, we have discussed full-step mode for a four phase PM stepper with 90 degrees per step. For the reader with more extensive curiosity, a short list of reasonably comprehensive articles describing stepping motors is given in the bibliography.

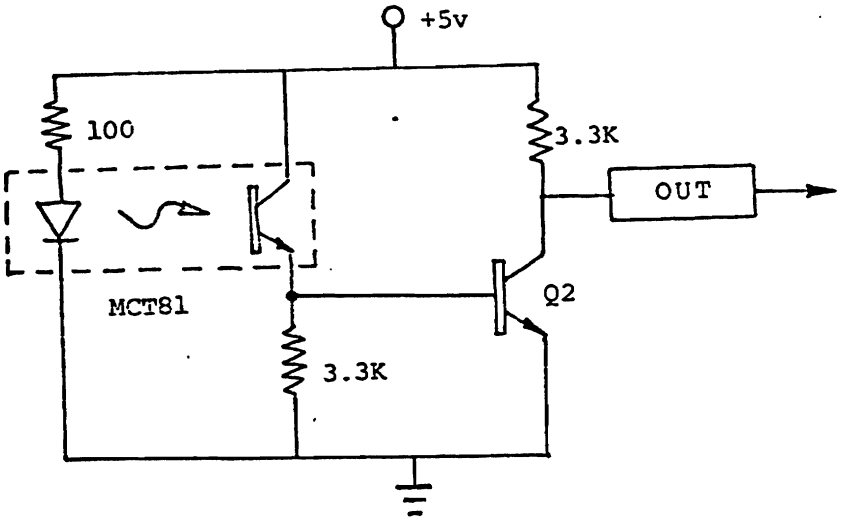
3.2 The Auxiliary Board

Beside the main board, which is the effector path to the motors, is an auxiliary board. It contains digital logic to return the values of the six

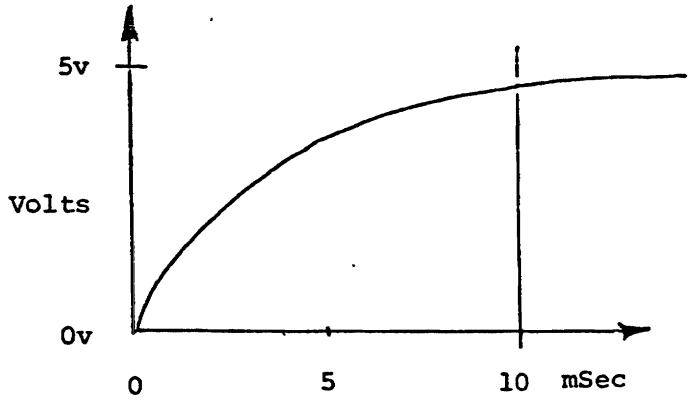
optical limit switches on the mechanical assembly. The switches are activated when a motor causes the assembly to move to a mechanical limit (extreme) of the system, and the change of status of any of these switches results in a non-maskable interrupt to the MCP. The auxiliary board contains the logic to generate this signal. The auxiliary board schematic diagram is given in figure #17 below.



The limit switches on the positioner act like "electric eyes"; each consists of an infrared light emitting diode (LED) and a photosensitive transistor, with a small space between the two. When either x,y, or z carriages reach the end of travel, this photocoupling between LED and phototransistor is broken. The Phototransistor is low power output and operates in the linear amplification region however, rather than saturation region, so the output does not switch, but rather varies gradually as the LED's light is cut off. The output must be amplified by another transistor stage to bring the circuit output up to power for compatibility with a TTL input. Figure #18 shows the amplification circuit used with the optical limit switch.

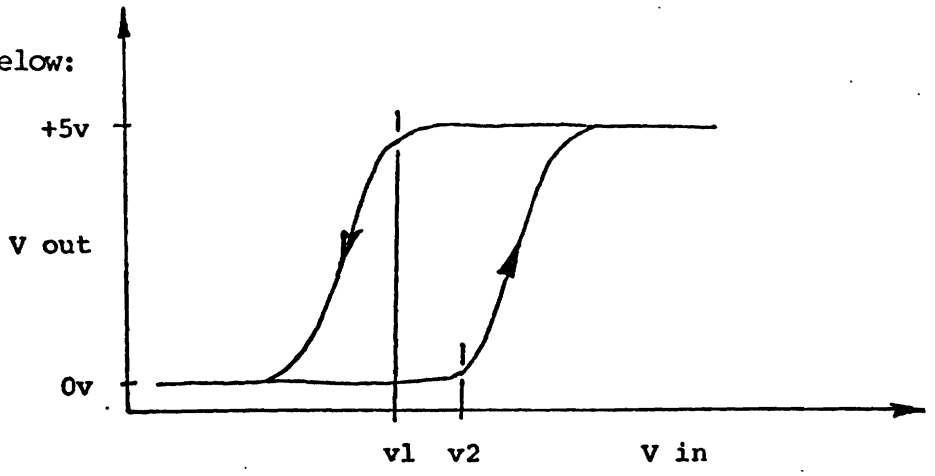


The output response of Q2, the output transistor, is rather a slow transition when the photocoupling is broken. This is illustrated in figure #19.



Since there is quite a slow rise time on the signal from the sense switch circuits, and since we would like a much more rapid switching response as input to the NMI generator (a few microseconds rather than milliseconds), a schmitt trigger has been used between Q2 and subsequent circuitry on the auxiliary board. The schmitt trigger is particularly suitable in situations where there is a slow input signal rise time since it has hysteresis. The input voltage of the schmitt trigger must exceed some level v2 before the output voltage goes high, but it must fall below a voltage v1 << v2 before the output goes low again. This is illustrated in

the figure below:

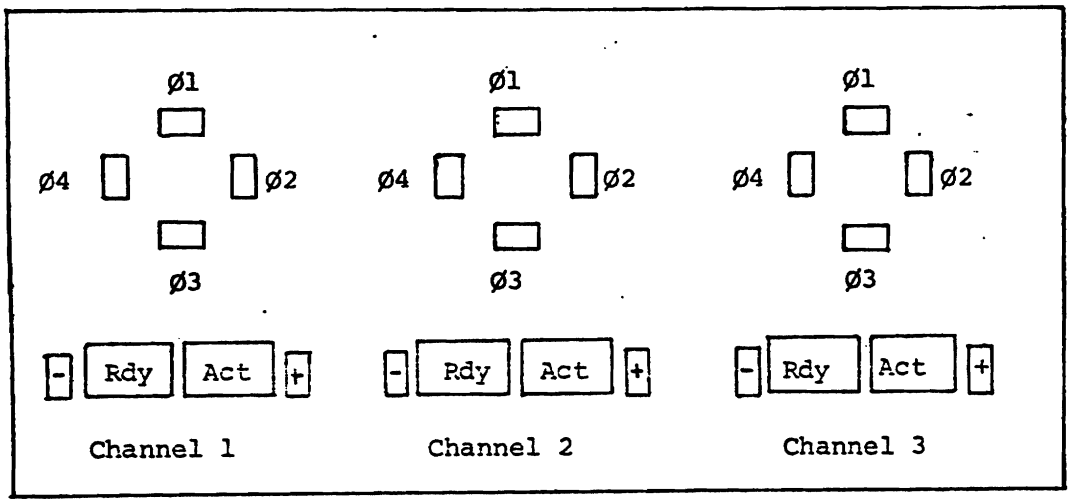


Subsequent circuitry of the auxiliary board consists of edge detectors for each limit switch, implemented by routing the switch's output into both inputs of a two-input exclusive OR gate with one line delayed. An RC element provides the delay line in this instance, and decides the output pulse width of the edge detector. NMI is the negated sum of all the edge detector outputs. This provides that when any sense switch transition is detected, a short negative pulse is generated on the NMI line (going to the Motor Control Processor).

3.3 MIU Front Panel Display

The front panel of the MIU is a digitally generated display which provides visual access to motor status information. Figure #21 shows the panel's layout.

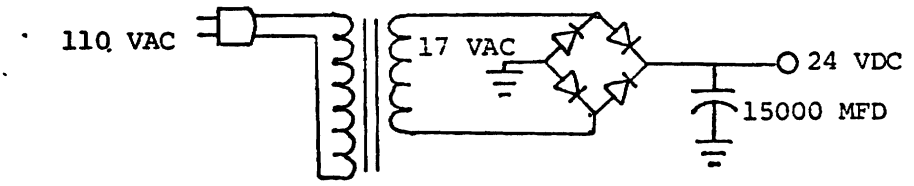
Each channel has a set of LED's indicating the status of the motor controlled by that channel. Four green LED's arranged in a circle indicate the two presently active windings of the motor (present motor shaft position). Two yellow lights indicate the status of the DIR line for the respective channel ('-' is associated with clockwise rotation and '+' is counterclockwise). The red "Active" light indicates the status of the STOP signal for that channel. Recall that STOP = low means that the power



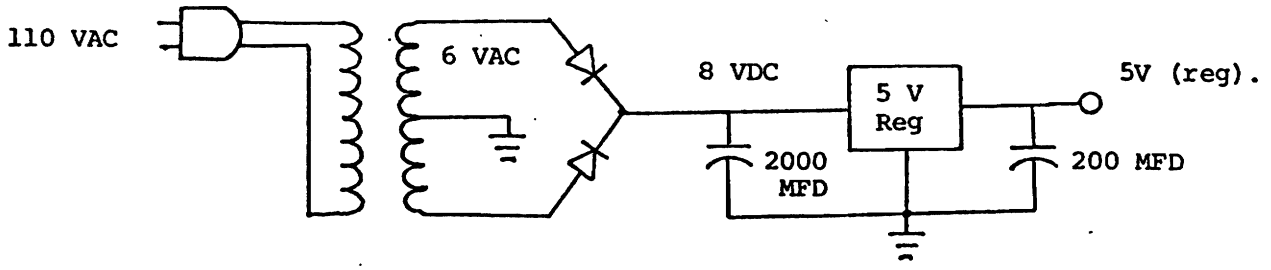
amplification darlingtons to the motor are deactivated; STOP = high implies that current is flowing through the windings indicated by the green lights. The red "Ready" LED's are intended to indicate that power is available to the respective motor channel.

3.4 MIU Power Supply

The MIU contains a 24 volt power supply for the stepping motors, and a 5 volt supply for the internal digital logic and front panel LED's. The 24 volt supply is a simple unregulated DC supply capable of sourcing at least 12 amperes. It is illustrated below:



The five volt supply for on board logic and LED's is regulated DC capabable of sourcing 3 amperes. It is illustrated below:


110 VAC
6 VAC
8 VDC
5 V
Reg
5V (reg).
2000
MFD
200 MFD

4. The Tactile Interface Unit (TIU)

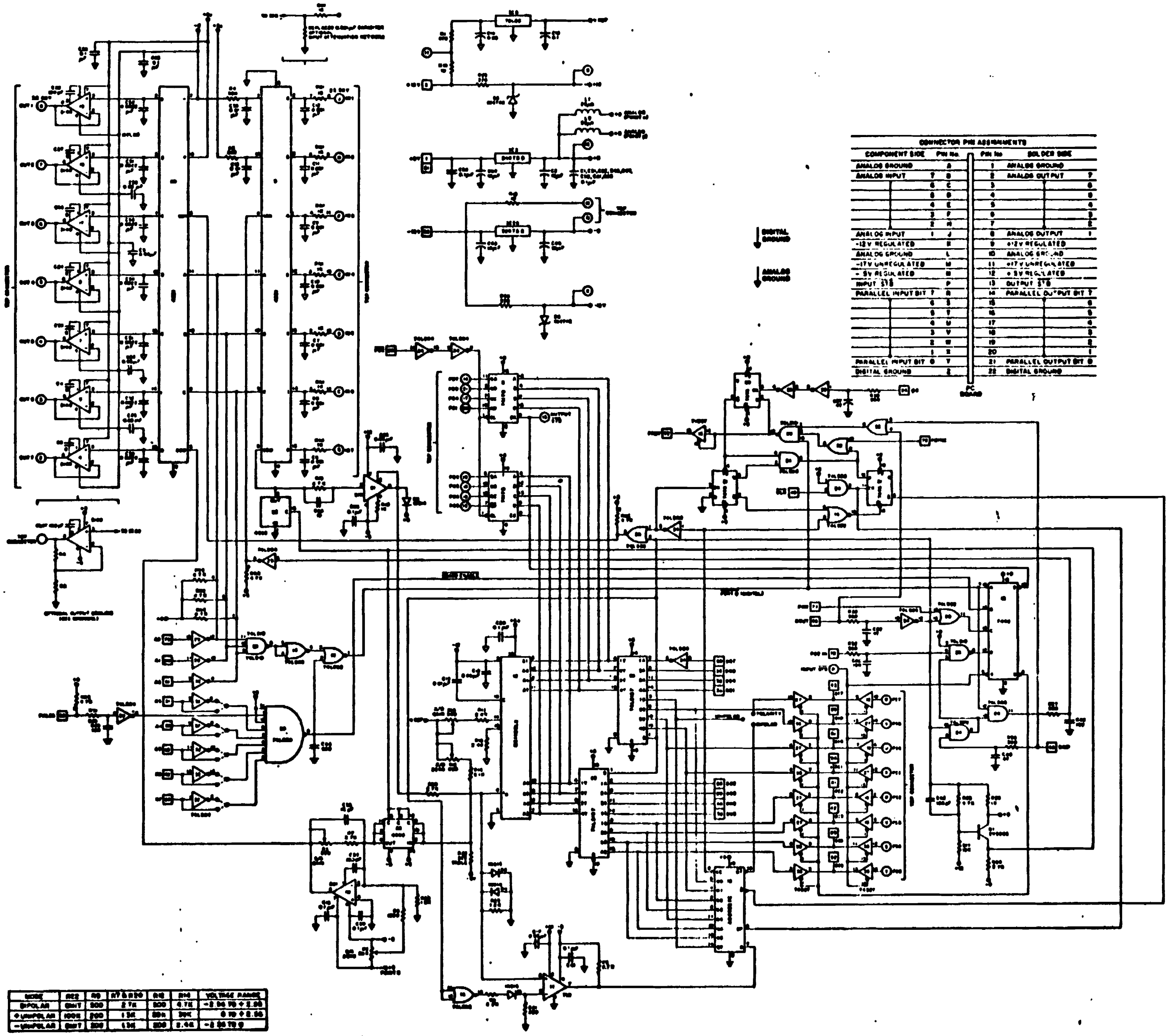
The TIU that is proposed, consists of an 8 bit analog to digital converter, capable of rapid conversion (about 200 ns) of any of 133 sampled analog input on the "finger". An 8 bit address selects the sensor to be sampled. This tactile interface will arrive in conjunction with the "finger" being developed at L.A.A.S. in Toulouse France, by J. Clot (see Appendix A).

In lieu of the arrival of the multisensing finger of Clot, a simpler verion has been used to demonstrate the viability of the TSS. At the present time, a high precision joystick has been placed at the end of the z-axis arm of the mechanical positioner in inverted (upside down) position. The joystick has springs which cause a return to center (x,y origin) after joystick deflection. In this fashion, the joystick acts as a simple single digit (finger), providing two analog inputs (x and y axis deflection).

The joystick may be used for both contour description and even limited texture decription. Once the joystick is in contact with a surface, it is moved along the surface. Surface contour can be described based on the degree of deflection of the joystick as it is moved over the surface. Limited textural analysis can also be performed based on the less macroscopic perturbation of the joystick as it is moved along a suface. It should be possible to distinguish between reasonably smooth surfaces, finely grained surfaces, and several other gradations of coarseness, by observing the extent to which the joystick is deflected (jitters) and the rate at which this occurs.

The A to D unit presently being used in conjunction with the joystick

finger is a Cromemco D+7A I/O board. This board contains seven 8 bit A to D channels, seven D to A, and one digital bidirectional 8 bit parallel port, and is S-100 bus compatible. Two of the A to D inputs are used for digitization of the joystick x and y deflection. The D+7A I/O is neatly interfaced to the Tactile Sensing Processor along its local S-100 bus. This allows the TSP to gain access to the digitized x and y joystick values through two i/o ports along the S-100 bus. The Schematic diagram of the D+7A I/O is figure #25, and full detail of this device is given in the Cromemco D+7A I/O Reference Manual.



CONNECTOR PIN ASSIGNMENTS					
COMPONENT SIDE		PIN No.	PIN No.	SOLDER SIDE	
ANALOG GROUND		A	1	ANALOG GROUND	
ANALOG INPUT	7	B	2	ANALOG OUTPUT	7
	6	C	3		6
	5	D	4		5
	4	E	5		4
	3	F	6		3
	2	H	7		2
ANALOG INPUT	1	J	8	ANALOG OUTPUT	1
-12 V REGULATED		K	9	+12 V REGULATED	
ANALOG GROUND		L	10	ANALOG GROUND	
-17 V UNREGULATED		M	11	+17 V UNREGULATED	
+ 5V REGULATED		N	12	+ 5 V REGULATED	
INPUT STB		P	13	OUTPUT STB	
PARALLEL INPUT BIT	7	R	14	PARALLEL OUTPUT BIT	7
	6	S	15		6
	5	T	16		5
	4	U	17		4
	3	V	18		3
	2	W	19		2
	1	X	20		1
PARALLEL INPUT BIT	0	Y	21	PARALLEL OUTPUT BIT	0
DIGITAL GROUND		Z	22	DIGITAL GROUND	

MODE	R22	R[illegible]	R7 & R20	R[illegible]	R14	VOLTAGE RANGE
BIPOLAR	OMIT	200	2.7K	200	4.7K	-2.56 TO +2.56
+ UNIPOLAR	100K	200	1.3K	20K	39K	0 TO +2.56
- UNIPOLAR	OMIT	200	1.3K	200	2.4K	-2.56 TO 0

5. Mechanical Positioning System

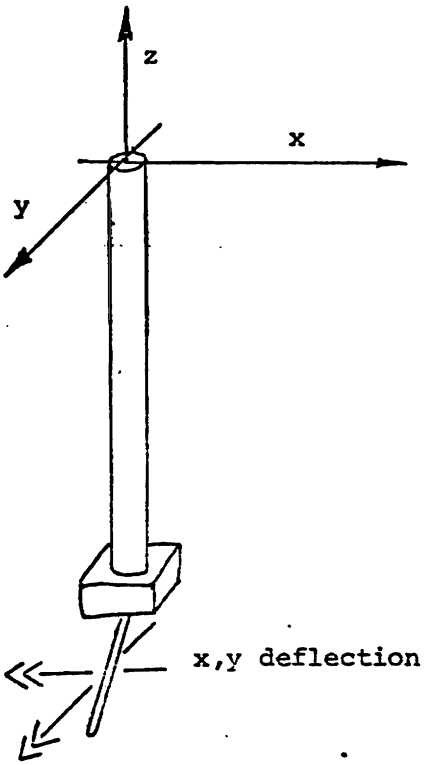
The mechanical positioning system resides at the lowest level of the TSS. It provides the mechanical hardware necessary to manipulate the sensor in a constrained observation region.

5.1 Mechanical Provision for Movement

The mechanical assembly is a three axis, rectilinear system employing stepper motors to drive x,y and z carriages. The carriages each move along a linear track established by two parallel guides. Guides consist of either a Unislide mechanical slide and case hardened guide rod, or two guide rods. Carriages are mounted on ball bushing which slide with limited friction along guide rods. Stepping motors each rotate a worm gear (lead screw), clockwise or counterclockwise in order to drive the carriage back and forth along the track. Lead screws are 7 mm. in diameter, with 1 mm. thread pitch, and are each roughly one half a meter in length. Since the stepping motors has 90 degree step resolution, the precision of the system is 0.25 mm, or one step. However the physical flexiblity of the mechanism exceeds this precision, allowing only about a 1 mm. accuracy (1 shaft revolution or 4 steps).

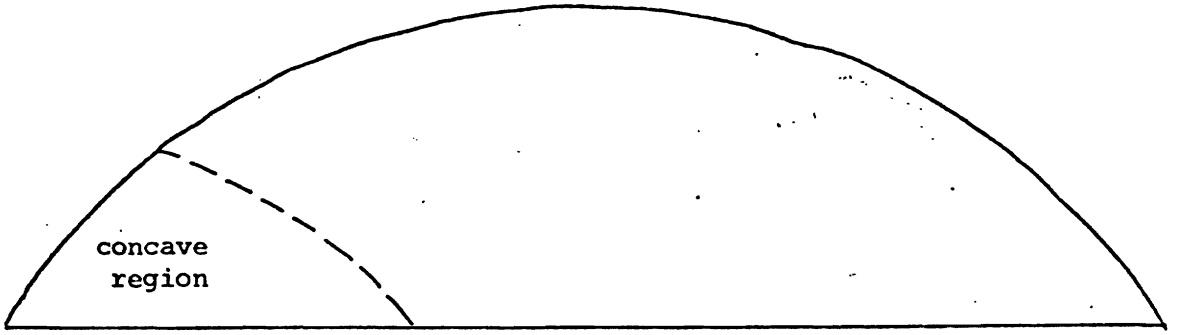
5.2 Arm and Sensor

Along the z carriage (depthwise degree of freedom) is a tubular shaft. It is at the end of this shaft or "arm" that the sensing "finger" resides. Figure #26, below illustrates this apparatus.

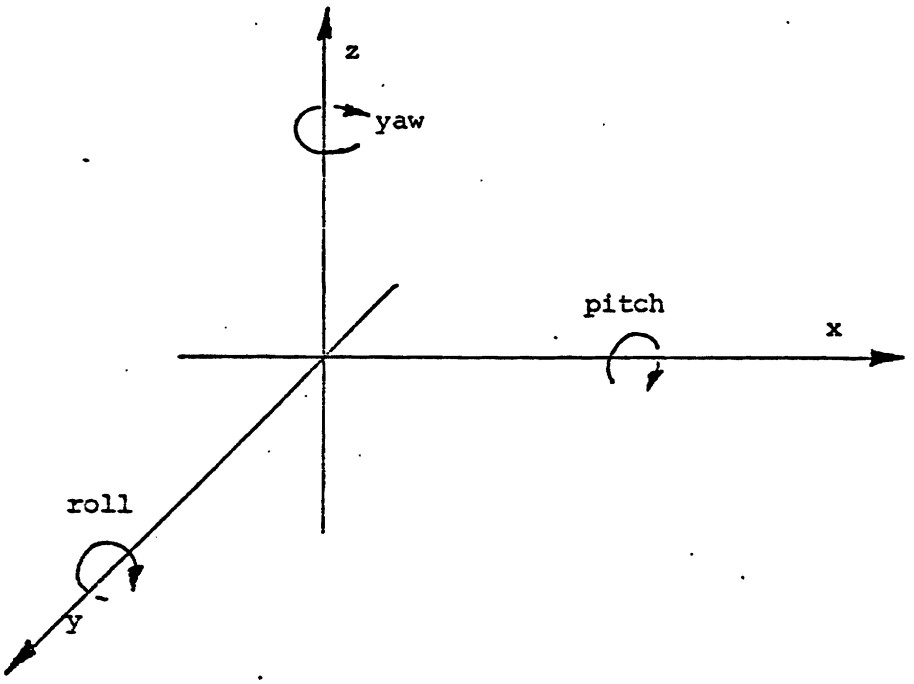


The sensor is placed in contact with the object by extending the arm into the observation region and/or sweeping the region in x or y directions until contact is made. It is subsequently possible to "follow" surfaces with the sensor as described in the interprocessor interface discussion (section 2).

One limitation of the present mechanical positioner is its inability to probe certain concave object regions. For example, if an object such as figure #23 illustrates has a concavity (dotted lines), the finger cannot be oriented properly in order to investigate the surface in the concavity.



A device capable of probing concave regions requires the ability to orient as well as position the sensor. Orientation of the sensor requires three additional degrees of freedom: pitch, roll and yaw. These are the respective rotations about the three positioning axes: x, y and z.



5.3 Enhanced Tactile Control

Presently it is not intended that the TSS affect the environment that it observes. Ultimately, however, it may be desirable to grasp objects to help identify them. To this extent a multi-digit tactile "hand" is requisite. Such a Tactile Manipulator can re-orient objects in order to acquire additional information about them. Furthermore, multiple digits afford the ability to test object compliance. This can be performed by "squeezing" the object between two or more digits, and measuring force required as digits get closer together.

Ultimately one or more multijointed arms such as the Unimation Puma 500 should be used in conjunction with multidigit hands. Important to the extended tactile process are the ability to grasp and manipulate objects, and garner tensile strength and compliance information based on forces applied by a hand or hands. Tensility may be examined by pulling or stretching between two arms as opposed to the former pressing or squeezing for compliance measurement. Investigation into manipulators and their control has been going on for several years. However, the realization of aspirations for such an extended capability tactile manipulation system is still a distant objective.

6. Summary and Results

The general intention of this work has been to consider the design of computer object recognition systems. In particular, attention has been focused upon the architictural and organizational aspects of a system which effectively realizes the following needs and desires:

* The integration of responses from many different sensors in the recognition process
* Multilevel Planning and Goal Oriented Processing
* Sensor Adaptive Control
* Cooperation between distributed processes to achieve common higher level goals.

The notion of a stratified architecture embodying many levels of virtual machines, was asserted to be an important conceptual framework for the solution of these needs and desires. Such a hierarchical and/or distributed system was said to afford the advantages of:

* multiple localized independent control of plans and goals
* virtual machines specialized to their task
* modular design
* intrinsic verification through multisensing
* exploitation of concurrent operation of low level tasks

An architecture of stratified nature was proposed for an object recognition system embodying the above intentions.

6.1 Results

The specific result of this work was the design and implementation of a system for tactile sensing which is congruous with the considerations we have given to the design of an object recognition system. The TSS was designed and implemented, embodying these fundamental principles, to which extent it is a subsystem which epitomizes the more general aspirations for the object recognition system. It includes the notions of multilevel control, concurrent low level operations, and interprocessor cooperation for sensor adaptive control and object description, within a stratified architecture.

Preliminary results from the associated work of Wolfeld include the completion of motor control routines within the MCP for positioning and movement of the sensor, and the development of primitives for interprocessor communication between the MCP and TSP within the Control Unit.

Bibliography

[1] Tanenbaum, Andrew S.: Structured Computer Organization, Prentice Hall, New Jersey 1976.

[2] Nilsson, Nils J.: Principles of Artificial Intelligence, Tioga Publishing Company, Palo Alto, Ca. 1979.

[3] Winston, Patrick Henry: Artificial Intelligence, Addison Wesley, Reading, Ma. 1977.

[4] Randall Davis and Jonathan King, "An Overview of Production Systems," AIM-271, The Artificial Intelligence Laboratory, Stanford University, Stanford, Ca., 1975.

[5] Erman L. D., and Lesser, V. R.: "A Multi-level Organization for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge", Advance papers of the Fourth Joint Conference on A. I., M.I.T. Artificial Intelligence Laboratory, Cambridge Ma., 1975.

[6] Minsky, Marvin: "A Framework for Representing Knowledge", in The psychology of Computer Vision, McGraw Hill, New York, 1975.

[7] Zadeh, Lotfi A.: "Fuzzy Algorithms", Information and Control, 8, 338-353, 1968.

[8] Gupta, M. M., Saridis, G. N., and Gaines, B. R. (eds),: Fuzzy Automata North Holland Publishing, New York,1977.

[9] Walker, Michael William: "Manipulator Control", Ph.D. Dissertation, Purdue University, 1978.

[10] Geschke, Clifford C.: "A System for Programming and Controlling Sensor-Based Robot Manipulators", Ph.D. Dissertation, University of Illinois at Champaign-Urbana, 1979.

[11] Sacerdoti, Earl J.: "Plan Generation and Execution for Robotics", Proceedings of the Workshop on Robotics Research, Newport, R.I. April 1980.

[12] Agin, Gerald J.: "Issues Involving Sensory Control", Proceedings of the Workshop on Robotics Research, Newport, R.I., April 1980.

[13] Nitzan, David: "Assesment of Robotic Sensors", Proceedings of the Workshop on Robotics Research, Newport, R.I., April 1980.

[14] Takase, Kunikatsu: "Skill of Intelligent Robots" Interantional Joint Conference on Artificial Intelligence, Tokyo, 1979.

[15] Okada, Tokiju: "Computer Control of a Multijointed Finger System" Proceedings of the Workshop on Robotics Research, Newport, R.I., April 1980.

[16] Nii, H.P., and Aiello, N.: "AGE, A Knowledge-Based Program for Building Knowledge-Based Programs, Proceedings of the Workshop on Robotics Research, Newport, R.I., April 1980.

[17] Lesser, V. R., and Corkill, D. D.: "The Application of Artificial Intelligence Techniques to Cooperative Distributed Processing" Proceedings of the Workshop on Robotics Research, Newport, R.I., April 1980.

[18] Hill, J. W., and Sword, A. J.: "Touch Sensors and Control", Stanford Research Institute, Menlo Park, Ca. 1972.

[19] Margolin, Bob: "Stepping Motors, the Precision Positioners", Electonic Product Magazine, September 1979.

[20] Giacomo, Paul: "A Stepping Motor Primer" Byte Magazine, February and March 1979.

[21] Wolfeld, J.: Forthcoming Master's Thesis, University of Pennsylvania, 1981.

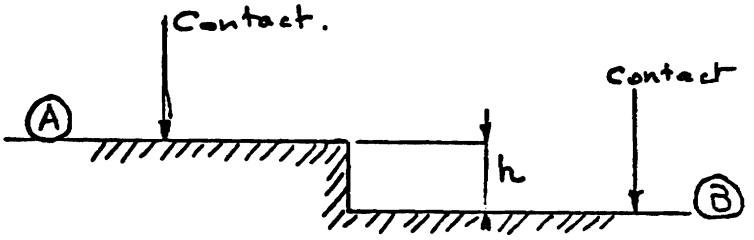
APPENDIX A

A FEW COMMENTS ABOUT INSTRUMENTATION PROBLEMS IN THE FIELD OF TACTILE SENSING (WITH OR WITHOUT VISION)

To recognize an object by "touching" supposes that the main objective is basically the determination of its surface texture. If one adds to this parameter the measure of thermal conductibility, acoustic impedance, etc... it is then possible to determine the structural nature of the object (wood, steel, ..., rough, smooth....).

However, if the only objective is pattern recognition, the concept of "contact" seems to be sufficient.

The difference between "touching" and "contact" can easily be shown in figure 1. Indeed, two planes A and B can be separated by contact only if the value of"h"is superior to the resolution of the mechanical measuring system. If h is inferior to that limit, it is necessary to use methods involving the "touching" concept.



- Figure 1 -

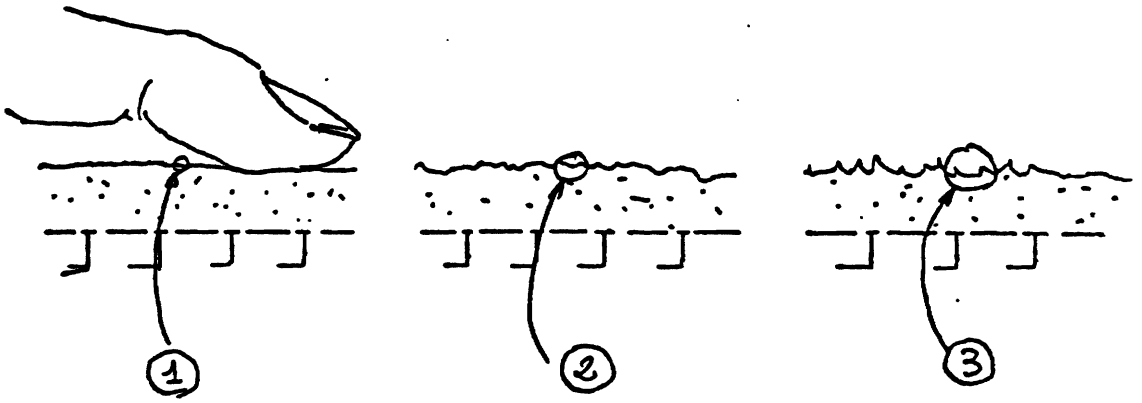
The problem of estimating a surface texture may obviously be solved by using the artificial skin.

The major drawback of such a method is to involve a relatively large area for just one measure.

So, the conventional magnetic cartridge is generally prefered, and that is right. The cartridge method is far more sensitive but perhaps gives unfortunately a "pin-point information", and is quite fragile.

All the above concepts are not really interesting for Robotics. They are studied in our Laboratory in the domain of Biomedical Engineering following two different kinds of application :

1°) After surgery, one is interested in determining the recovering of tactile sensations by the patient. The method consists in making several skins with different surface texture, as shown in figure 2.

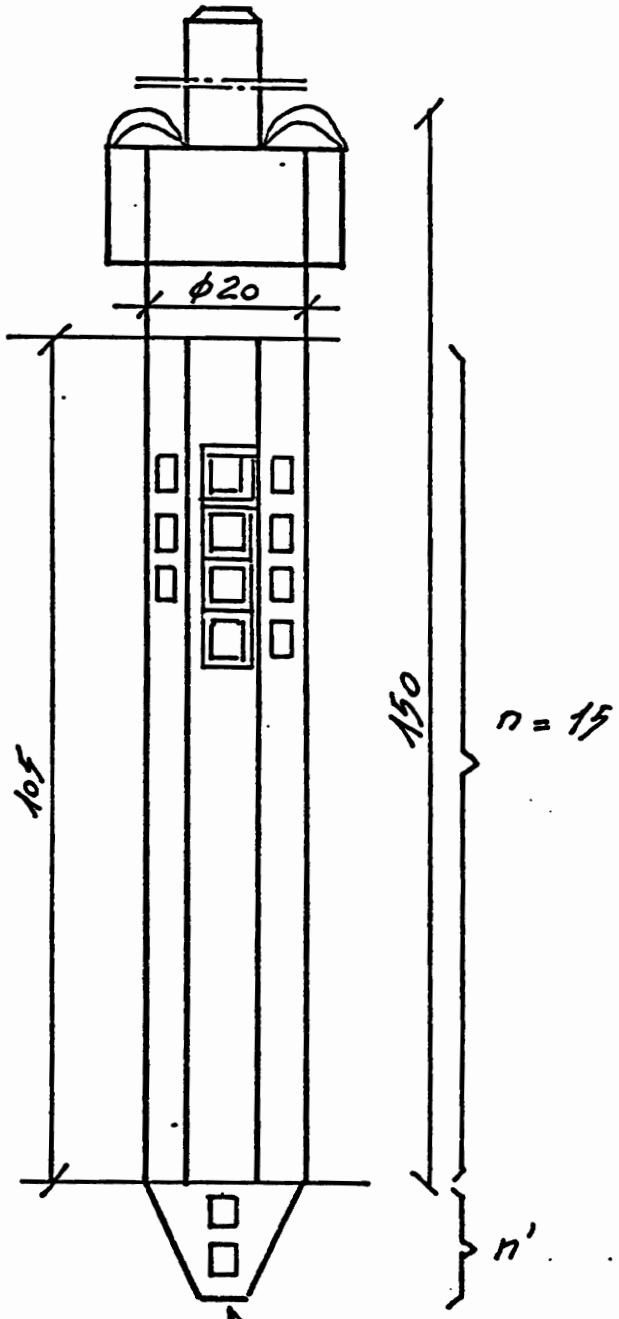


- Figure 2 -

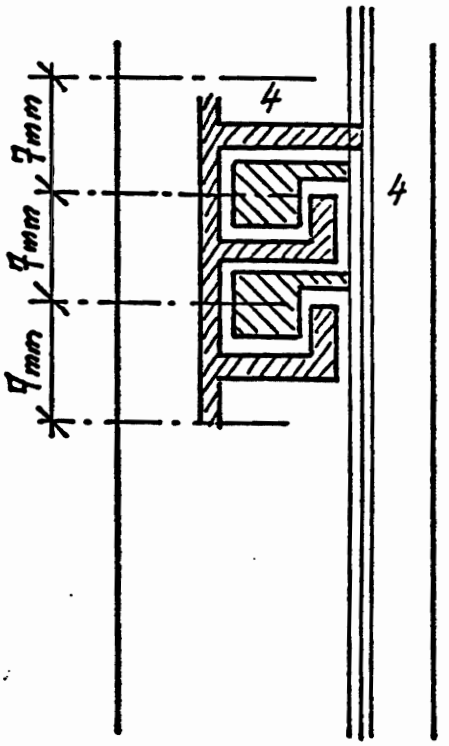
2°) For persons having lost all sensations of touching and force feedback, they can only see the object and determine its volume and roughly its weight. There is, in that case, a clear relationship between vision, touching and forces. The objective of the study is then, to replace the force feedback by the pitch of a musical sound. The goal is to give the person a training in order he/she later adapts automatically the force to the object he/she recognizes.

PROPOSAL FOR THE CHARACTERISTICS OF A TACTILE SENSOR (FIGURE 3).

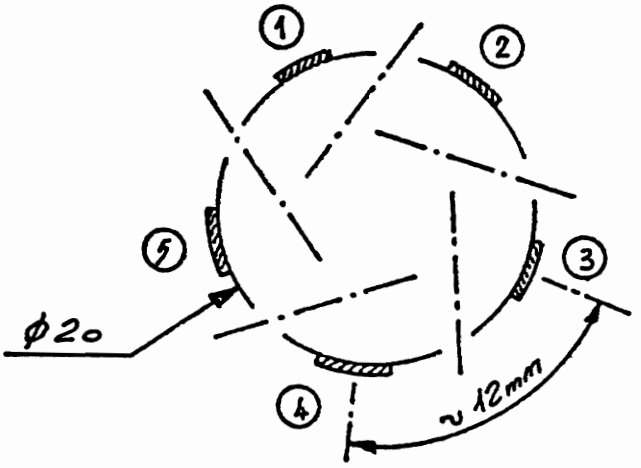
- Length of the sensitive portion : $\simeq$ 105 mm.
- Mean diameter of the probe (slightly conical) : $\simeq$ 20 mm.
- Contacts distributed upon five lines at 11 mm from each other.
- Along one line, the distance between contacts is 7 mm.
- Total number of measuring points : 86.
- For the first prototype, the implementation of the skin will be "had hoc". Later on, it might be necessary to especially cast the finger.



$$\Sigma = (n \times 5) + n' + 1$$
$$= 75 + 10 + 1$$
$$= 86$$

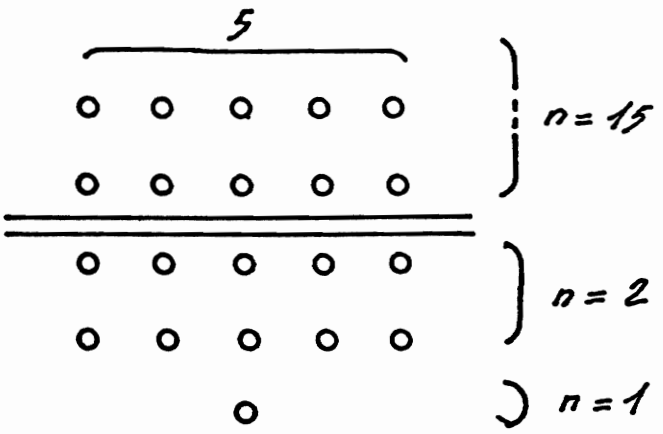


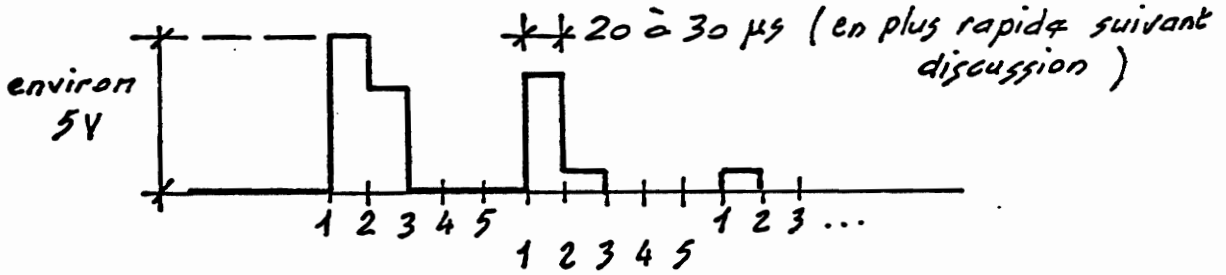
Echelle : 2

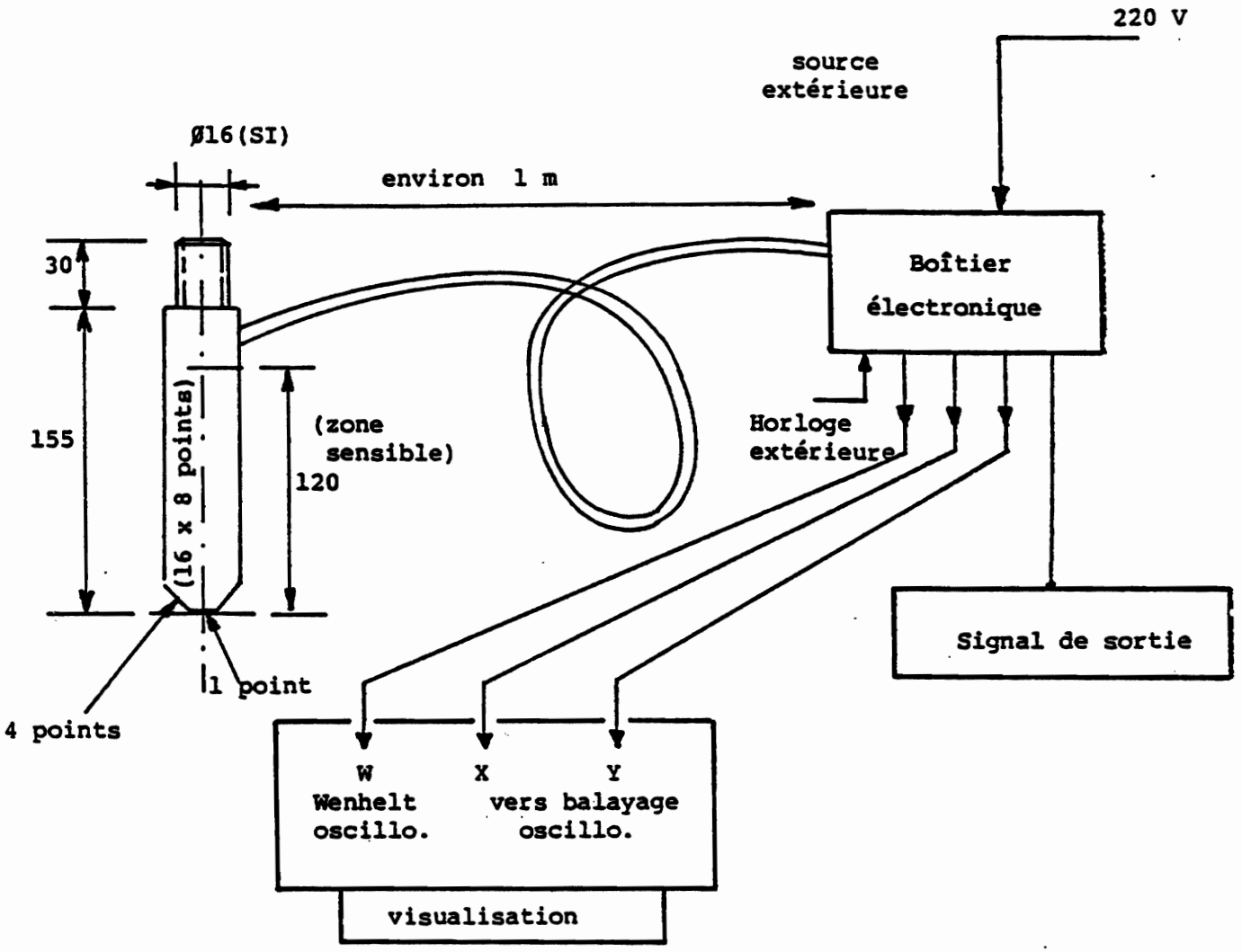


Echelle : 2

Figure 3


5
n = 15
n = 2
n = 1


environ 5 V
20 à 30 µs (en plus rapide suivant discussion)
1 2 3 4 5
1 2 3 4 5
1 2 3 ...


220 V
source
extérieure
Ø16(SI)
environ 1 m
30
155
(16 x 8 points)
(zone
sensible)
120
Boîtier
électronique
Horloge
extérieure
Signal de sortie
1 point
4 points
W
Wenhelt
oscillo.
X
Y
vers balayage
oscillo.
visualisation