A11101 565621

NBS
PUBLICATIONS

ROBOTICS SUPPORT PROJECT
FOR THE AIR FORCE
ICAM PROGRAM

National Bureau of Standards
National Engineering Laboratory, CME
Industrial Systems Division
Programmable Automation
Washington, DC 20234

FINAL REPORT
August 1981

NBSIR 81-2387

For Early Domestic Dissemination

Because of its significant early commercial potential, this information, which has been developed under a U.S. Government program, is being disseminated within the United States in advance of general publication. This information may be duplicated and used by the recipient with the expressed limitations that it not be published nor released to foreign parties without appropriate licenses. Release of this information to other domestic parties by the recipient shall be made subject to these limitations. This legend shall be marked on any reproduction of this data in whole or part. These limitations shall be considered void two years after the release date on the data.

Manufacturing Technology Division
Air Force Materials Laboratory
Air Force Systems Command
Wright-Patterson Air Force Base, Ohio 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

NATIONAL BUREAU
OF STANDARDS
LIBRARY
JUN 20 198
notacc-circ.
QC100
.U56
81-2387
1981
c.2

MIPR SY1455-78-00003

ROBOTICS SUPPORT PROJECT FOR THE AIR FORCE ICAM PROGRAM

FINAL REPORT

James S. Albus
Anthony J. Barbera
M. L. Fitzgerald
Robert Haar
Roger D. Kilmer
Marilyn Nashman
Louis Palombo
Thomas Wheatley

National Bureau of Standards
National Engineering Laboratory, CME
Industrial Systems Division
Programmable Automation
Washington, DC 20234

Prepared for
Manufacturing Technology Division
Air Force Material Laboratory
Air Force Systems Command
Wright-Patterson Air Force Base, Ohio 45433

TABLE OF CONTENTS

Page

EXECUTIVE SUMMARY 1

GUIDELINES FOR SELECTION OF ROBOTIC SYSTEMS (4.1).............. 7

REVIEW OF ROBOTICS GUIDE (4.1.1)................................ 8

PRACTICES FOR DEVELOPING ROBOT SOFTWARE (4.1.3) 9

INTERFACE STANDARDIZATION (4.1.4) 10

MANAGEMENT SUMMARY OF LEVELS OF CONTROL (4.1.4) 11

GLOSSARY OF TERMS FOR PROCUREMENT (4.1.5) 12

EXPECTED FUTURE DEVELOPMENTS (4.1.6) 13

RECOMMENDED IMPLEMENTATION CONCEPTS (4.1.7) 30

PROVIDE TECHNICAL SUPPORT TO CONTRACTORS (4.2) 44

TECHNICAL CONSULTING (4.2.1) 45

FIELD TRIPS (4.2.2) 46

IMPROVED ROBOT SYSTEM PERFORMANCE AND SAFETY (4.3) 47

NBS ROBOT CONTROL SYSTEM DEVELOPMENT (4.3.1) 48
INTRODUCTION 49
DESIGN REQUIREMENTS 51
SYSTEM PARTITIONING 56
IMPLEMENTATION 82
SUMMARY 110

IMAGING SENSOR DEVELOPMENT (4.3.2) 112
VISION SYSTEM OVERVIEW 113
IMAGE PROCESSING SOFTWARE 115
CAMERA INTERFACE DESIGN 131

INHERENTLY SAFE SYSTEMS (4.3.3) 136

LIST OF FIGURES

Section 4.1.5

1. Three Aspects of a Sensory-Control System 32
2. NBS Microcomputer Network 37
3. Hierarchical Architecture for Factory Control 39

Section 4.3.1

III.1 The Modules of the Control System 56
III.2 Off-line Programming Module 57
III.3 Function of Task Description Segment 57
III.4 Format of an Entry in the Program Table 58
III.5 Valid Entries in the Program Table 58
III.6 Function of the Data Description Segment 61
III.7 Methods of Entering Data into Location Tables 62
III.8 Format of an Entry in the Location Table 63
III.9 Format of an Entry in the Object Description Table 64
III.10 Partitioning of the Robot Control System 64
III.11 Information Flow for a Level in the Control Hierarchy 66
III.12 Two Parallel Hierarchies - Control and Sensory Processing 69
III.13 Sensor Data Input to Different Levels of the Hierarchy 70
III.14 Interactions between Sensory and Control Hierarchies 71
III.15 Programming Interfaces 71
III.16 The Four Level Hierarchy for Real-Time Control Module 73
III.17 Primitive - Trajectory Interface 80
III.18 Trajectory - Coordinated Joint Motion Interface 81
III.19 Coordinated Joint Motion - Servo Interface 81

IV.1 Layout of 16-bit Microprocessor Board and Bus Interface 87
IV.2 Dual-Port RAM 87
IV.3 Priority Circuit Diagram 89
IV.4 Block vs. Single Data Transfer 91
IV.5 Common Memory Layout 92
IV.6 Protocol Logic 95
IV.7 On-board RAM Layout 97
IV.8 Data Transfers among 3 Processors 97
IV.9 Status Checking Algorithm 99
IV.10 Interaction in a 2 Processor System 99
IV.11 Buffer Allocation Table Format 102
IV.12 Protocol Flags 103
IV.13 Data Collection Interface Hardware 104
IV.14 DTP-DIB Interface 104
IV.15 DTP-DIB Protocol 106
IV.16 Data Transfer Timing 106
IV.17 DIB-Device Interface 107
IV.18 Example of Asynchronous Transfer 108

Section 4.3.2

1. Camera and Flash Mounted on Robot 113
2. Structured Light 114
3. NBS Vision Interface 114
4 - 7. Object Rotations 117
8. Height Computation 121
9. Calculation of Object Distance 122
10. Defining Plane of Light 122
11. Transition Value Data 123
12. Plane of Light Picture 129
13 - 22 Circuit Diagrams - Vision Hardware 133

Section 4.3.3

1. Shapes used to Model Robot 138
2. Side View of Stanford Arm 139
3. Top View of Stanford Arm 139
4. Stanford Arm with all Joints at Zero Position 140
5. Stanford Arm identifing Wrist, Finger and Tool Point 140
6. Vectors to Define Arm Configuration 141
7. Ray Tracing Technique 141
8. Graphic Simulation of Stanford Arm 142
9. Stanford Arm in NBS Laboratory 152
10. Floor Plan of NBS Laboratory 152
11. Safety Mat Surrounding Robot 155
12. View of Electrostatic Transducer 158
13. Instrumentation for Acoustic Characteristic Measurement 158
14. Time History Envelop for Single Ultrasonic Pulse 159
15. Directional Characteristics of Radiated Sound Waves 160
16. Narrow Band Spectral Analysis of Single Pulse 161
17. Analysis of Individual Components of Pulse 161
18. Output Signals from UCB 163
19. Setup for Investigation of Target Surfaces 165
20. Manikin 167
21. Robot with Five Electrostatic Transducers 169
22. Areas of Detection Coverage 170

List of Tables

Section 4.3.3

1. Spatial Coordinates of Stanford Arm 139
2. Comparison of Intrusion Alarm Systems 151
3. Instrumentation Settings for Acoustic Measurements 159
4. Effect of Targer-Sensor Distance on Echo 166
5. Comparison of Target Echo Strengths for Plywood 166
6. Comparison of Target Echo Strengths for Manikin 167

INTRODUCTION

The National Bureau of Standards Programmable Automation Group was chosen by the Air Force ICAM office to perform a wide variety of tasks in support of the project entitled "Robotic Systems for Aerospace Batch Manufacturing." The scope of work was a 26 month effort involving approximately two man years per year of in-house work at NBS. Work was begun in October 1978 and was completed in December 1980. Three major tasks were addressed:

1. To develop a guidleine for the selection and procurement of robots, robot computer languages and robot control systems for aerospace batch manufacturing.

2. To provide technical support to the Air Force and to Air Force contractors on the implementation of robot systems.

3. To directly support computer control, programming, and sensor technology to substantially improve robot system performance and safety.

These three major tasks were further broken down into thirteen subtasks, three of which were to result in documents published separately from the normal quarterly reports and this final report.

The thirteen subtasks cover a wide variety of topics from the compilation of a Glossary of Robotic Terms, to the development of Recommended Practices for Robot Control Software, to research on Robot Imaging Sensors. The entire list of subtasks and a short description of what was accomplished under each is contained in the Executive Summary. The main body of this report consists principally of a discussion of Expected Future Developments (Task 4.1.6), a Recommended Aerospace Robot System Implementation Concept (Task 4.1.7), and a description of NBS work in Hierarchical Control (Task 4.3.1), Imaging Sensor Development (Task 4.3.2), and Inherently Safe Systems (Task 4.3.3). NBS recommendations on Practices for Developing Robot Control Software (Task 4.1.2), Recommended Standard Interfaces (Task 4.1.3), and a Glossary of Terms for Robotics (Task 4.1.5) are being published as separate documents and are included with this final report as addenda.

EXECUTIVE SUMMARY

The NBS effort on MIPR SY14757-78-00003, "Robotics Support Project for the Air Force ICAM Program", is described in this final report. Our work on all the major task areas is summarized by subtask in what follows.

TASK 4.1 DEVELOP GUIDELINES FOR SELECTION AND PROCUREMENT OF ROBOTIC SYSTEMS FOR BATCH MANUFACTURING

Task 4.1.1 Review of the "Robotics Guide"

The draft copy of the "Robotics Application Guide" (RAG) was received and an extensive review conducted. The review included a global analysis, suggestions for reordering of topics, and a detailed commentary on sections that could benefit from additional work. This review was given to General Dynamics and a finalized RAG was generated.

Task 4.1.2 Recommended Practices for Developing Robot Control Software

The specifications, design and implementation of the system control software of the Task B contractor were reviewed. In addition, based upon the concept of software as the control and integrating mechanism of future robot control systems, a philosophy was developed to guide the creation of software for Aerospace Batch Manufacturing robot control. A set of recommended robot software development procedures is being published as a separate report as specified by the above referenced contract. That document is attached as an addendum to this final report.

Task 4.1.3. Recommended Standard Interfaces

A workshop on potential robot interface standards was held at the National Bureau of Standards June 4 through June 6, 1980. One hardware and five software interfaces were identified as candidates for standardization. Proceedings of this workshop were published as a separate report and are attached as an addendum to this report.

Task 4.1.4 Management Summary of Levels of Control

NBS reviewed the hierarchical control system proposed during Task B of the referenced contract. An overview was prepared concerning the essential role of computer software and sensor technology in making effective use of industrial robots in Aerospace Batch Manufacturing. Potential system implementation problems were discussed and implemetation procedures recommended.

Task 4.1.5 Glossary of Terms for Robotics

A glossary of robotic terms has been prepared and is being published as a separate report as specified. That document is attached as an addendum to this report.

Task 4.1.6 Expected Future Developments

Based upon the analysis of the efforts conducted under the above referenced contract, industry trends in robot hardware and software development have been identified and extrapolated to assess their impact in the Aerospace Industry. A number of potential future Aerospace applications were analyzed, and the enabling technologies needed to make robots practical for these applications were identified. An attempt was made to estimate the dollar and manpower costs and time needed to develop these enabling technologies.

Task 4.1.7 Recommended Aerospace Robot System Implementation Concept

A scenario for the best use of robot systems in the next few years is hypothesized in this section of the report. The suggested scenario draws upon the efforts conducted in the referenced contract, as well as on other programs at NBS and trends in control software and sensor systems and potential future developments in the robotics industry. Recommendations are made for a system design that will not become obsolete with the expected developments in the industry. A design is described which integrates sensory-interactive robots into a modular hierarchical control system for a totally automated factory of the future.

TASK 4.2 PROVIDE TECHNICAL SUPPORT TO THE AIR FORCE AND TO AIR FORCE CONTRACTORS

Task 4.2.1 Technical Consulting

In addition to reviewing contractor reports referenced in 4.1, NBS was available for consulting with contractors on proposed plans and specifications related to robotic technology.

Task 4.2.2 Field Trips

Several field trips were made to contractor facilities and plants for review and consulting.

Task 4.2.3 NBS Reports

NBS supplied copies of related reports and papers to interested users and suppliers as related to Air Force needs.

TASK 4.3 IMPROVE ROBOT SYSTEM PERFORMANCE AND SAFETY

Task 4.3.1 Robot Control System

NBS work on hierarchical control systems is reported under this section of this final report. A technical description of the NBS microcomputer system architecture is provided as well as a discussion of the specific hardware and bus priority circuitry used to implement the present version of this system.

Task 4.3.2 Imaging Sensor Development

Work performed under this task investigated techniques for making effective use of imaging sensors and for incorporating interpretation of image processing into robot control systems on a real-time basis. The vision hardware and attributed lighting techniques used to acquire images are described. Software for determining binary image orientation and analyzing the 3-dimensional shape of binary images is presented. Circuit diagrams for the NBS vision system camera interface electronics are also included.

Task 4.3.3 Inherently Safe Systems

An ultrasonic ranging sensor and data processing system was developed and tested. Experiments were performed using this sensor system as a safety device. Whenever an intruder or unexpected obstacle is detected in the working envelope of the robot, a warning flag to the control system causes the robot to stop and wait for the intruder to leave before continuing the task. Additional types of safety sensors are also analyzed for possible future systems.

FINAL REPORT

NBS ROBOTICS SUPPORT PROJECT

FOR THE

AIR FORCE ICAM PROJECT

GUIDELINES FOR SELECTION AND PROCUREMENT OF ROBOT SYSTEMS FOR BATCH MANUFACTURING 4.1

Task 4.1.1 Review of the "Robotics Guide"

The draft copy of the "Robotics Application Guide" (RAG) was received and an extensive review conducted. The review included a global analysis, suggestions for reordering of topics, and a detailed commentary on sections that could benefit from additional work. This review was given to General Dynamics and a finalized RAG was generated.

Work performed on this section of the contract was reported in its entirety in the Combined First and Second Reports covering the period from October 1, 1978 to April 1,1979. That report will not be duplicated in this final report.

Task 4.1.2 Recommended Practices for Developing Robot Control Software

The specifications, design and implementation of the system control software of the Task B contractor was reviewed. In addition, based upon the concept of software as the control and integrating mechanism of future robot control systems, a philosophy was developed to guide the creation of software for Aerospace Batch Manufacturing robot control.

This task specified that a separate document titled "Recommended Procedures for the Design and Implementation of Real-time Control Software" be published under separate cover. That document is attached as an addendum to this final report.

Task 4.1.3. Recommended Standard Interfaces

A workshop on potential robot interface standards was held at the National Bureau on Standards June 4 through June 6, 1980. One hardware and five software interfaces were identified as candidates for standardization. Proceedings of this workshop were published as a separate report. That document is attached as an addendum to this final report.

Task 4.1.4 Management Summary of Levels of Control

NBS reviewed the hierarchical control system proposed during Task B of the referenced contract. An overview was prepared concerning the essential role of computer software and sensor technology in making effective use of industrial robots in Aerospace Batch Manufacturing. Potential system implementation problems were discussed, and implemetation procedures recommended.

The Combined First and Second Interim Report presented issues in robot control systems, functional requirements, and an overview of a hierarchical structured control system. That report will not be duplicated here.

Task 4.1.5 Glossary of Terms for Robotics

A glossary of robotic terms has been prepared and is being published as a separate report as specified. That document is attached as an addendum to this final report.

Task 4.1.6 Expected Future Developments

Abstract

Based upon the analysis of the efforts conducted under the contract referenced above, industry trends in robot hardware and software development have been identified and extrapolated to assess their impact in the Aerospace Industry. A number of potential future Aerospace applications are analyzed, and the enabling technologies needed to make robots practical for these applications are identified. An attempt is made to estimate the costs and time needed to develop these enabling technologies.

EXPECTED FUTURE DEVELOPMENTS

The results of the Air Force ICAM project "Robotic Systems for Aerospace Batch Manufacturing" make it clear that much remains to be done before robots can have a significant impact on productivity in aerospace production. Each of the three contractors selected by the Air Force for Tasks, A, B, and C performed a series of experiments using robots in a variety of applications. In every case, the results were interesting and suggestive of many potential applications. The drilling and routing experiments at General Dynamics provided valuable insights into how robots might be used on the shop floor to automate these routine yet costly tasks. The McDonnell-Douglas riveting demonstration was an impressive accomplishment which integrated elements of off-line programming, visual processing, and hierarchical control. The experimental and theoretical studies performed by Lockheed Georgia suggest a number of task areas which may benefit from robotics technology.

However, all these contract tasks were more enlightening from the standpoint of how much remains to be done than from what was actually accomplished. The General Dynamics drilling and routing tasks were chosen because they were considered to be doable with presently available robot technologies. Yet the final demonstration clearly showed that more sensory information and more sophisticated control capabilities will be required before robots can do even these simple tasks reliably without constant human supervision. The McDonnell-Douglas riveting demonstration showed that despite a great amount of capital equipment supporting the robot, even the simplest riveting tasks were painfully slow and far from being ready for the small batch production shop. None of the experiments indicated that robots are ready to leave the laboratory in large numbers to begin work on the factory floor -- at least not in the next few years. Robots are still too cumbersome, expensive, slow, stupid, and insensitive to what is happening in their environment to compete seriously with human labor in most aerospace applications, particularly in assembly. Furthermore, the range of tasks that were represented by the ICAM demonstrations make up only a tiny subset of the total number of jobs in airframe assembly.

These results make it clear that, at best, it will be several more years before robots can produce significant cost savings in a substantial fraction of aerospace manufacturing applications. In short, the robot revolution is not yet ready to sweep the aerospace industry into a new era.

Technical Problem Areas

The ICAM robotics systems effort has demonstrated that there are a number of difficult technical problems that still need to be solved.

First of all, robot positioning accuracy needs to be improved. Although the repeatability of most robots is on the order of 0.050 inch over its working volume, the absolute positioning accuracy may be off as much as 0.250 inch, or even 0.500 inch in some regions of the reach envelope. Thus, it is not possible to program a robot to go to an arbitrary mathematically defined point in a coordinate space and have any assurance that the robot will come within a half of an inch. This difficulty creates major problems for off-line programming, particularly where the same program is to be used on several different robots. Presumably, this accuracy problem could be solved through closer manufacturing tolerances, although not without cost. Alternatively, calibration procedures might allow each robot to offset its off-line program points to compensate for its mechanical inaccuracies. However, no efficient methods of robot calibration have yet been developed, and robot control software is not presently designed to use calibration tables for improving absolute positioning accuracy. Until this absolute positioning accuracy problem is solved, robot assembly in the small batch environment will be uneconomical. Teaching a robot every point in the trajectory of a complex assembly task is a time-consuming job which may take many times longer than would be required to perform the same task by hand. Thus, using a robot for small lot batch assembly cannot be economical until software can be efficiently produced by off-line programming.

Second, dynamic performance must be improved. Present robots are too slow and clumsy to compete effectively with human labor in assembly. Two possible exceptions to this are in arc welding, where speed is not a factor, and spot welding, where the task corresponds to a simple string of points in space -- a procedure which the robot is particularly adept at executing. However, if robots are to perform other types of assembly tasks, they must be able to execute much more complex routines with much greater grace, dexterity, and speed than they are now capable of. Servo systems must be designed to take into consideration the changing inertial configuration of the robot under different loads and different positions of the robot arm. Servos need to be alternately stiff and compliant along different axes in space (which do not generally coincide with joint coordinates). This requires much more sophisticated cross-coupled servo computations than are presently employed.

Third, sensors of many different types must be developed. Robots must become able to see, feel, and sense the position of objects in a number of different ways. Processing of visual data must become faster and be able to detect 3-dimensional shapes and relationships. Robot grippers must become able to feel the presence of objects and sense the forces developed on those objects. Proximity sensors are needed on robot fingertips to enable the robot to measure the final few millimeters before contacting objects. Longer range proximity sensors are needed on the robot arm to avoid colliding with unexpected obstacles. Force and touch sensors are needed to detect and measure contact forces. A variety of acoustic, electromagnetic, optical, x-ray, and particle detectors are needed to sense the presence of various materials such as metals, ferromagnetics, plastics, fluids, and limp goods, and to detect various types of flaws in parts and assemblies. Both the sensing devices and the software for analyzing sensory data represent research and development problems of enormous magnitude.

Fourth, control systems are needed which can take advantage of sophisticated sensory data from a large number of different types of sensors simultaneously. Present control systems are severely limited in their ability to modify a robot's behavior in response to sensed conditions. Robot control systems need to be able to accept feedback data at a variety of levels of abstraction and have control loops with a variety of loop delays and predictive intervals. (See for example Section 4.3.1 of this report). Sensory data used in tight servo loops for high speed or high precision motions must be processed and introduced into the control system with delays of no more than a few milliseconds. Sensory data used for detecting the position and orientation of objects to be approached must be available within hundreds of milliseconds. Sensory data needed for recognizing the identity of objects or the relationship between groups of objects can take seconds. Control systems that are properly organized in a hierarchical fashion so that they can accommodate a variety of sensory delays of this type are not available on any commercial robot.

Fifth, robot control systems need to have much more sophisticated internal models of the environment in which they work. Robot control systems should have data bases similar to those generated by Computer-Aided-Design (CAD) systems, and used for computer graphics displays. These can describe the three dimensional relationships of both the workplace and the workpieces. Such data bases are needed to generate expectations as to what parts should look like to the vision system, or what they should feel like to the touch sensors, or where hidden or occluded features are located. Eventually, such internal models might be used in the automatic generation of robot software - for example, by describing how a finished assembly should look, or even how each stage of an assembly or construction task should appear in sequence.

Sixth, techniques for developing robot software must be vastly improved. Programming-by-teaching is impractical for small lot production, especially for complex tasks where sensory interaction is involved. Shop floor personnel unskilled in computers must be able to instruct robots in what to do and what to look for in making sensory decisions. Eventually, it will be necessary to have a whole range of programming languages and debugging tools at each level of the sensory-control hierarchy. The development of compilers and interpreters and other software development tools, as well as techniques for making use of knowledge of the environment derived from a number of different sensors and CAD data bases are research topics that will require hundreds of person-years of highly skilled software talent.

Seventh, interfaces need to be defined in some standardized way, so that large numbers of robots, machine tools, sensors, and control computers can be connected together in integrated systems. (See for example Section 4.1.3 of this report).

For the most part, these are profound problems which will require much more research and development. It may be possible to improve the mechanical accuracy of robots, and to improve servo performance with little more than careful engineering. But much more fundamental research and development will be required before the sensor, control, internal modeling, software generation, and systems interface problems are solved. Much remains to be done in sensor technology to improve the performance, reliability, and cost effectiveness of all types of sensory transducers. Even more remains to be done in improving the speed and sophistication of sensory processing algorithms and special purpose hardware for recognizing features and analyzing patterns both in space and time. The computing power that is required for high speed processing of visual and acoustic patterns may even require new types of computer architecture.

Sensory-interactive control systems that can respond to various kinds of sensory data at many different levels of abstraction are still very much in the research phase. Current commercial robot control systems do not even allow six-axis incremental movements in response to sensory data. None have convenient interfaces by which sensory data of many different kinds can be introduced into the servo loops on a millisecond time scale for true real-time sensory interaction. None of the commercial robot control systems have anything approximating CAD data bases or computer graphics models of the environment and workpieces. Finally, current programming techniques are time consuming and not capable of dealing with internal knowledge or sophisticated sensory interactions.

These are very complex problems that will require many person-years of research effort. It is thus not suprising that the robotic systems work carried out under the first phase of the Air Force ICAM project demonstrated the inadequacy of present robot technology significantly to affect productivity in aerospace manufacturing.

What Lies in the Future?

All of the problems listed above are capable of being solved. It is only a matter of time and expenditure of resources before sensors and control systems are developed that can produce dexterous, graceful, skilled behavior in robots. Eventually, robots will be able to store and recall knowledge about the world that will enable them to behave intelligently and even to show a measure of insight regarding the spatial and temporal relationships inherent in the workplace. High order languages and sophisticated control systems will eventually make it possible to instruct robots using much the same vocabulary and syntax that one might use in talking to a skilled worker.

There is no question that, given enough time and resources, robotics will eventually become a significant factor in increasing productivity in aerospace production. The question is: How much time and how many resources will be required before such increases occur?

The NBS ICAM robotics research team thinks that more than a few tens of millions, and probably less than a few hundreds of millions of dollars will be required to make robots capable of dramatically improving aircraft manufacturing. More than a few hundred and probably less than a few thousand person-years of high level scientific and engineering talent will be needed before robot software of sufficient complexity can be generated economically for small lot batch production. In other words, a research and development effort of at least one, and perhaps two, orders of magnitude greater than the initial Air Force ICAM Robotics project will be required to produce a significant impact on productivity in the aerospace industry.

Recommendations for Future Programs

If this analysis of the magnitude of the problem is correct, the question then arises as to what the Air Force can (or should) do in supporting future robotics research. The following suggestions are offered:

First, a long term research and development program should be formulated that would foster centers of scientific and engineering competence in the basic sciences of artificial intelligence and robotics.

Second, short term application areas should be chosen that are suitable to present and near-term robot technology. In particular, assembly was and probably still is the wrong area on which to concentrate. Although it is a glamorous goal, robot assembly is very difficult. Results in assembly will come slowly -- too slowly to satisfy funding managers who require short term results. A viable robotics R&D program should therefore at least include, if not concentrate on, tasks where robots are relatively good already, and where solid evolutionary progress can be demonstrated at regular intervals.

In particular, robotics research for aerospace manufacturing should focus on N/C machine tool loading and unloading, spot welding, arc welding, spray painting, sanding, and placement of fixtures for drilling, counter sinking, riveting, deriveting, screwing, and inspecting wing skins and fuselage panels. These are areas where current robot technology can be applied in the near term and economic payoff can be achieved in the short run. Longer term research in off-line programming, sensory interactive control systems, and use of CAD data models can initially be focused on these more simple tasks before being applied to the more difficult problems of assembly.

Third, adequate funding should be provided on a long-term basis. A goal of $20 to $50 million per year is realistic in terms of the dimensions of the technical problems to be solved, and is quite small compared to the potential payback to be gained from enhanced productivity just in the manufacture of products for the Air Force. Such a funding level represents only the cost of one or two airplanes per year. Of course, it is not necessary that the Air Force provide this entire amount. Other funding sources are available, and efforts should be made to coordinate funding for research and development. Much of the basic control theory, sensors, data processing, programming languages, and knowledge storage and recall technologies are generic and useful in all types of robotics. Thus, coordination in funding of research between the Air Force, DARPA, ONR, NBS, NASA, and NSF, plus a number of private corporations, is a practical option.

One typically invests in a new technology with the expectation of something between ten and fifty percent annual return on investment. If one believes that robot technology has the potential significantly (e.g. by greater than 10%) to affect productivity in aerospace manufacturing, then the proper level of investment would be at least an order of magnitude greater than it has been.

Even a few million per year invested in robotics research and development would begin to bring rapid progress in robot accuracy and dynamic performance. One or two million could easily produce calibration techniques and error correcting control systems with sufficient accuracy to permit off-line programming. Several million more could produce advanced servo systems that could execute swift and dexterous movements that approach or even exceed the capabilities of the human hand and arm. A few million more would produce ultra high level programming languages and man-machine interface techniques that would make robot programming economically practical for small lot production.

Ten to $20 million per year would support sufficient research into vision and tactile sensing to make robots able to see and feel well enough to perform a wide variety of tasks. Programmers would develop software that gives robots significant intellectual capacities for planning and scheduling and imparts a high level of manufacturing expertise and craftsman-like skills. Significant research into light weight structures and exotic types of actuators would produce robot tip velocities of hundreds of inches per second and accelerations of several "g's". Over a decade, this level of funding would produce sensors and sensory processing systems that would give robots perceptual capabilities approaching that of human assembly workers in the limited domain of many aerospace jobs. Sensory interactive control systems using tens, or even hundreds, of microcomputers could be built. These would be capable of producing entire manufacturing cells with skills, coordination, and intellectual capabilities almost like those of a person. Robots, machine tools, materials transport systems, and inspection machines could be operated unsupervised and unassisted for long periods of time (i.e., hours, days, even weeks). Internal data models could be developed to the degree necessary so that robots could be instructed to perform tasks, and to shift from one task to another, with little more effort than currently required for dealing with skilled human workers. Robot mobility systems could be developed that would permit robots to move freely about an entire plant and to work in teams of two and three when required by specific tasks.

In short, several tens of millions per year spent on robot R&D could revolutionize aerospace manufacturing within a decade. In two decades the results would spill over into all of industrial manufacturing.

The spin-off for the civilian economy would be rapid and dramatic. In the long run, the development of the technology needed to make robots effective in the batch production environment of aerospace manufacturing would directly affect productivity in the robot industry itself. Eventually, the use of robots to automate the manufacture of robots would become practical. The results of this might enable the robot industry to duplicate the cost/performance record of the computer industry. Once that happens, the productive capabilities of constant-cost robots might escalate by 20% each year.

By the year 2000 totally automated factories could produce annual savings of billions of dollars annually in the aerospace industries alone. The benefits to the civilian economy could exceed this by an order of magnitude.

If this analysis is even close to the mark, a much higher rate of investment in robotics by the Air Force would appear to be easily justified. The results are certain to be large and positive. All the technological problems are reasonably well understood. No further fundamental scientific breakthroughs are needed. Each of the technical problems mentioned above is slowly yielding to solution in the few research laboratories in this country that have more than a "critical mass" level of effort.

Unfortunately, there is not a widespread appreciation of the nature of the problems that remain, and especially not of the potential benefits that will result from the solution of these problems. Highly skilled, dexterous, inexpensive robots with sufficient intelligence to perform many industrial tasks will completely change the economic structure, not only for aerospace manufacturing, but for the entire industrial system.

Unfortunately, domestic political considerations preclude the civilian branch of government from taking a lead role in investing in robot technology on a large scale. Robots are widely perceived by average citizens as a threat to jobs and the potential harbinger of unemployment. It is not clear to the people on the street how robots would benefit them personally, or why the government should subsidize robot development. Many persons are fearful of robots as competitors in the job market. Thus, there is not, and most likely will not be, a ground swell of demand for government involvement in the development ,of robots for the civilian economy. In fact, most popular opinion is negative, or at best neutral, assuming that such development is the proper domain of the private sector.

Massive investments in robot technology may also be slow in coming from private industry. This technology is still considered futuristic, and therefore risky, by most industrial managers. The amount of research that still needs doing looks awesome to the average executive, even for the corporate giants. More important, the returns on investments in robotics technology cannot be exclusively guaranteed to the corporations that make the investments. Much of robot technology is generic. Many of the important developments cannot be protected by patents. There are thus few incentives for private industry to invest any more than is necessary simply to remain abreast of the competition. The conclusion is that unless the military takes the lead in funding large investments in robot technology, it probably will not occur in America for a number of years.

Task 4.1.7 Recommended Aerospace Robot System Implementation Concept

Abstract

A scenario for the best use of robot systems in the next few years is hypothesized in this section of the report. The suggested scenario draws upon the efforts conducted in the referenced contract, as well as on other programs at NBS, trends in control software and sensor systems, and potential future developments in the robotics industry. Recommendations are made for a system design that will not become obsolete with the expected developments in the industry. This design integrates sensory-interactive robots into a modular hierarchical control system for a totally automated factory of the future.

RECOMMENDED AEROSPACE ROBOT SYSTEM IMPLEMENTATION CONCEPT

Based on the efforts conducted under the present contract, recent trends in control software and sensor systems, and expected future developments in the robotics industry, NBS makes the following technical recommendations for robot systems that should not become obsolete with the expected developments in the industry.

It is clear that future robots will be much more sophisticated in the use of both sensory data and prior knowledge of the work environment. Furthermore, most of them will be integrated into a much larger control system which will direct many other types of machine tools, material transport systems, and inspection, testing, and inventory control systems throughout the entire plant. Eventually, robots will become integral components in the totally automated factory.

The problem of producing control software for totally automatic factories incorporating hundreds of robots and other types of machines is overwhelmingly complex unless some methodology is developed which can partition the overall problem into manageable subproblems. Of course, large systems such as factories (and even more complex systems such as armies, governments, businesses, and biological organisms) can often be controlled quite effectively. The command and control structure for successful organizations of great complexity is usually hierarchical. Goals or tasks selected at the highest level are decomposed into sequences of subtasks which are passed to one or more operational units at the next lower level in the hierarchy. Each of these lower level units decomposes its input command in the context of feedback information obtained from other units at the same or lower levels, or from the external environment, and issues sequences of sub-subtasks to a set of subordinates at the next lower level. This same procedure is repeated at each successive hierarchical level until at the bottom of the hierarchy a set of sequences of primitive actions drive individual actuators such as motors, servo valves, hydraulic pistons, or individual muscles. This basic scheme can be seen in the organizational hierarchy on the left of Figure 1.

A single chain of command through the organizational hierarchy on the left is shown as the computational hierarchy in the center of Figure 1. This computational hierarchy consists of three parallel hierarchies: a task decomposition hierarchy, a sensory processing hierarchy, and a world model hierarchy. The sensory processing hierarchy consists of a series of computational units, each of which extracts the particular features and information patterns needed by the task decomposition unit at that level. Feedback from the sensory processing hierarchy enters each level of the task decomposition hierarchy. This feedback information comes

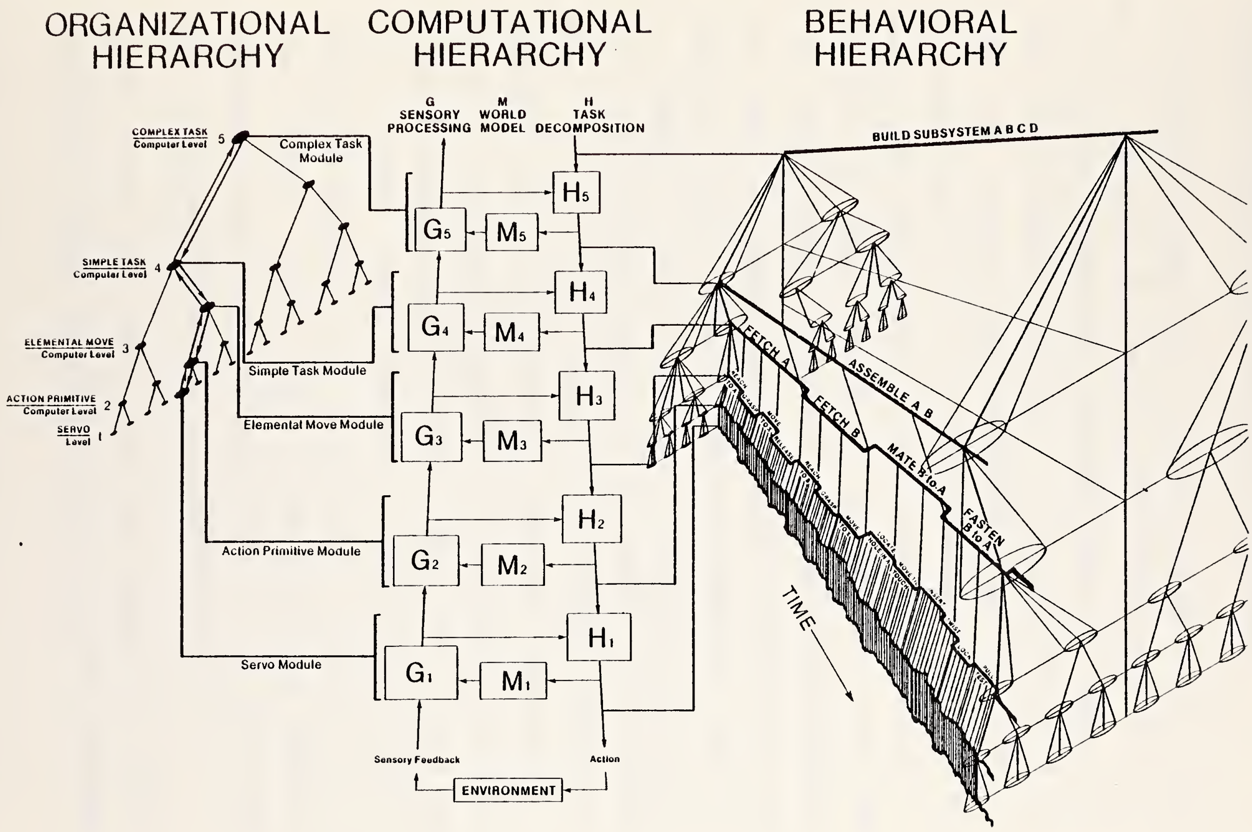

ORGANIZATIONAL HIERARCHY
COMPUTATIONAL HIERARCHY
BEHAVIORAL HIERARCHY
COMPLEX TASK Computer Level 5
SIMPLE TASK Computer Level 4
ELEMENTAL MOVE Computer Level 3
ACTION PRIMITIVE Computer Level 2
SERVO Level 1
Complex Task Module
Simple Task Module
Elemental Move Module
Action Primitive Module
Servo Module
G SENSORY PROCESSING
M WORLD MODEL
H TASK DECOMPOSITION
H5
G5
M5
H4
G4
M4
H3
G3
M3
H2
G2
M2
H1
G1
M1
Sensory Feedback
Action
ENVIRONMENT
BUILD SUBSYSTEM A B C D
ASSEMBLE A B
FETCH A
FETCH B
MATE B to A
FASTEN B to A
REACH TO A
GRASP
MOVE TO X
RELEASE
REACH TO B
GRASP
MOVE TO X
LOCATE HOLE IN A
MOVE TIL TOUCH
INSERT
TWIST
LOCK
PULL TEST
TIME

FIGURE 1. Three aspects of a sensory-control system.

computes an output with a very short time delay.

If the output of each unit in the task decomposition hierarchy is described as a vector, and plotted versus time in a vector space, a behavioral hierarchy such as is shown on the right side of Figure 1 results. In this illustration a high level goal, or task, (BUILD SUBASSEMBLY ABCD) is input to the highest level in a robot control hierarchy. The H5 task decomposition unit breaks this task down into a series of subtasks, of which (ASSEMBLE AB) is the first. This "complex" subtask command is then sent to the H4 task decomposition unit. H4 decomposes this "complex" subtask into a sequence of "simple" subtasks (FETCH A), (FETCH B), (MATE B to A), FASTEN B to A). The H3 unit, subsequently decomposes each of the "simple" subtasks into a string of "elemental moves" of the form (REACH TO A), (GRASP), (MOVE to X), (REALEASE), etc. The H2 decomposition unit then computes a string of trajectory segments in a coordinate system fixed in the work space, or in the robot hand, or in the work piece itself. These trajectory segments may include acceleration, velocity, and deceleration profiles for the robot motion. In H1, each of these trajectory segments is transformed into joint angle movements, and the joint actuators are servoed to execute the commanded motions.

At each level, the G units select the appropriate feedback information needed by the H modules in the task decomposifrom the same or lower levels of the hierarchy or from the external environment. It is used by the modules in the task decomposition hierarchy to sequence their outputs and to modify their decomposition function so as to accomplish the higher level goal in spite of perturbations and unexpected events in the environment.

The world model hierarchy consists of a set of knowledge bases that generate expectations against which the sensory processing modules can compare the observed sensory data stream. Expectations are based on stored information which is accessed by the task being executed at any particular time. The sensory processing units can use this information to select the particular processing algorithms that are appropriate to the expected sensory data and can inform the task decomposition units of whatever differences, or errors, exist between the observed and expected data. The task decomposition unit can then respond, either by altering the action so as to bring the observed sensory data into correspondence with the expectation, or by altering the input to the world model so as to bring the expectation into correspondence with the observation.

Each computational unit in the task decomposition, sensory processing, and world modeling hierarchies can be represented as a finite-state machine. At each time increment, each unit reads its input and based on its present internal state tion hierarchy. The M units generate predictions, or expected values, of the sensory data based on the stored knowledge about the environment in the context of the task being executed. A more complete description of the mathematics which describe this type of hierarchical structure is contained in the document "Theory and Practice of Hierarchical Control" included with this report.

Microcomputer Network Implementation

As a part of the ICAM contract, NBS has implemented the cross-coupled task decomposition, sensory processing, world modeling computational hierarchy of Figure 1, in a network of microcomputers shown in Figure 2.

Time is sliced into 28 millisecond increments. At the beginning of each increment each logical module reads its set of input values from the appropriate locations in common memory. It then computes its set of output values, which it writes back into the common memory before the 28 millisecond interval ends. If a logical module takes longer than the 28 milliseconds to compute an output, an on-board protocol procedure causes the processor to get back in synchronization with the reset pulse before writing out the results to common memory. The process then repeats.

Each logical module is thus a state machine whose output depends only on its present inputs and its present internal state. None of the logical modules admit any interrupts except for the reset-sync pulse that signals the beginning and end of the 28 millisecond computation intervals. This simple modular structure enormously simplifies the writing and debugging of software.

The NBS Vision System

The sensory side of the NBS hierarchical control system contains a vision system which uses active illumination to obtain depth information. A plane of light is generated by a photoflash tube and a cylindrical lens. This plane is projected into the field of view of a solid state 128x128 automation camera so that the distance to an illuminated surface can be directly computed by simple trigonometry. This camera and flash unit are fixed to the wrist of the robot manipulator.

The control hierarchy activates the vision system at specific points in the execution of a particular task. The control hierarchy also tells the vision software what type of object to expect and approximately how far away the object is expected to be. The vision software uses this information to select appropriate values for flash intensity and threshold and appropriate software algorithms for processing the visual data.

The vision processing modules either confirm the existence of the expected object and tell the control system where to move to approach it, or report that the expectation was incorrect.

At present, the NBS vision system interfaces with the control system primarily at the primitive action level for computing range and position of grip points and at the elemental move level for computing part orientation and approach paths. However, we are now in the process of adding new capabilities for part recognition at the simple task level.

Common Memory Data Transfer

All communications of data from one module to another in the NBS hierarchical control system take place via a common memory "mail drop" system as shown in Figure 2. This system has a disadvantage in that it requires two data transfers to get information from one module to another. However, we believe this disadvantage is far overshadowed by the following advantages:

1. There are no communication protocols between computing modules, because modules do not talk directly to each other. Only one processor is allowed to write into any single location in common memory. In each 28 millisecond time slice, all modules read from common

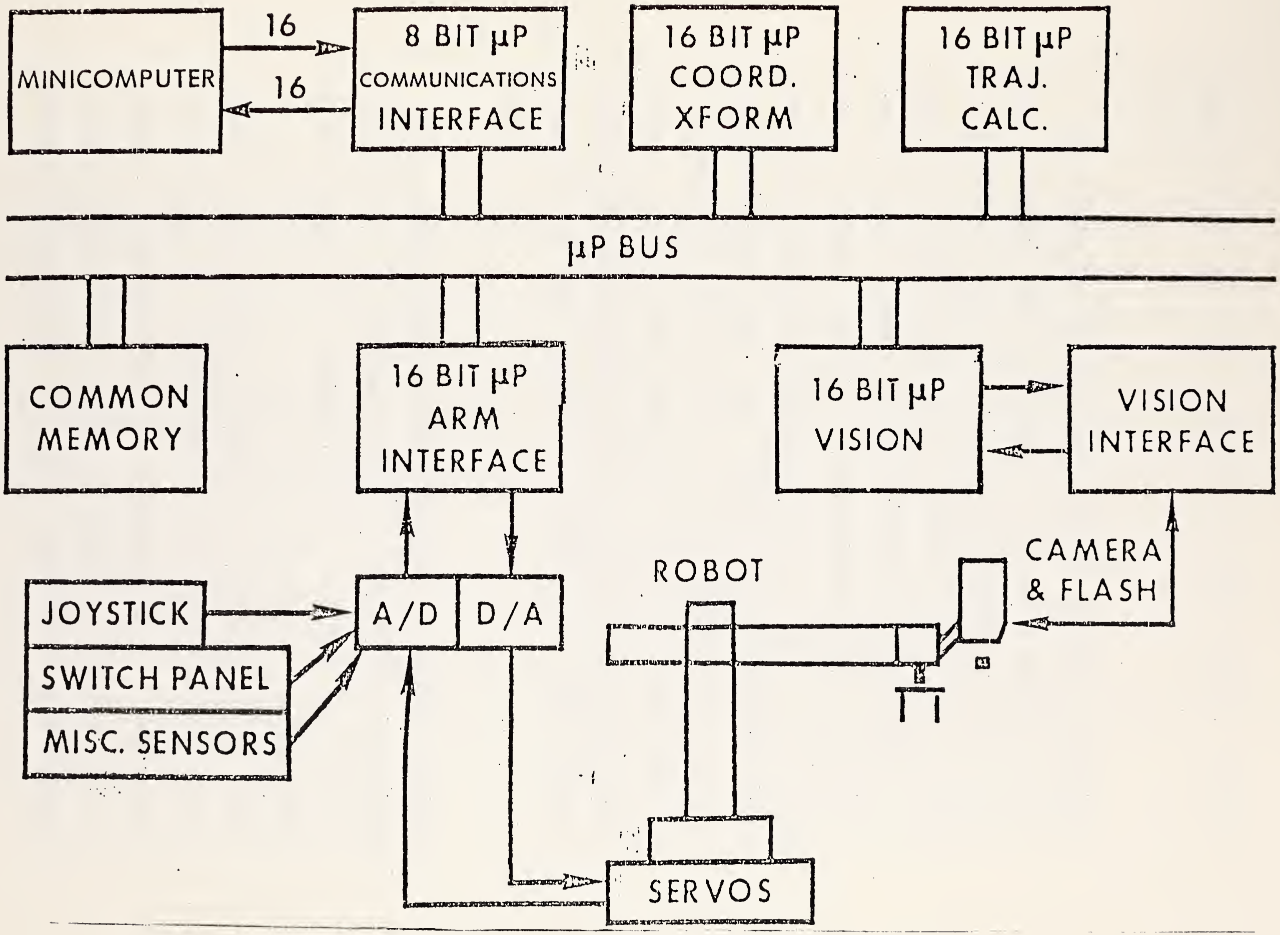

MINICOMPUTER
16
16
8 BIT µP COMMUNICATIONS INTERFACE
16 BIT µP COORD. XFORM
16 BIT µP TRAJ. CALC.
µP BUS
COMMON MEMORY
16 BIT µP ARM INTERFACE
16 BIT µP VISION
VISION INTERFACE
JOYSTICK
SWITCH PANEL
MISC. SENSORS
A/D
D/A
ROBOT
CAMERA & FLASH
SERVOS

FIGURE 2. The NBS microcomputer network architecture for implementing a hierarchical robot control system.

memory before any are allowed to write their outputs back in.

2. The addition of each new state variable requires only a definition of where it is to be located in common memory so that the module which generates it knows where to write it, and the modules which read it know where to look. Thus, new microcomputers can easily be added, logical modules can be shifted from one microcomputer to another, and new functions such as safety watchdogs and even new sensors can be included with limited effect on the rest of the system. As long as the system bus has surplus capacity, the physical structure of the system can be altered with few changes required in the software resident in the logical modules.

3. The common memory always contains a readily accessible map of the current state of the system. This map makes it easy for a system monitor to trace the history of any or all of the state variables, to set break points, and to reason backwards to the source of program errors or faulty logic. Such traceability is extremely important in a sophisticated, real-time, sensory-interactive system in which many processes are going on in parallel at many different hierarchical

levels.

Factory Control Hierarchy

As a result of successful experience with the microcomputer network described above, NBS has extended the hierarchical control concept beyond a single robot into an integrated factory. Figure 3 illustrates the computing architecture for an Automated Manufacturing Research Facility (AMRF) which is presently being designed and built at NBS. This AMRF will contain four N/C machining centers: a large and a small turning center, a horizontal and a vertical milling machine. It will also contain a coordinate measuring machine, a robot cart system for transportation of parts and tooling, and an automatic storage system for entering raw stock and tools and for storing finished parts.

The computing architecture shown in Figure 3 is intended as a generic system that can be applied to a wide variety of automatic manufacturing facilities and can be extended to much larger applications. The basic structure is hierarchical, with the computational load distributed evenly over computational units at the various different levels of the hierarchy. At the lowest level in this hierarchy are the individual robots, N/C machining centers, smart sensors, robot carts, conveyors, and automatic storage systems, each of which may have its own internal hierarchical control

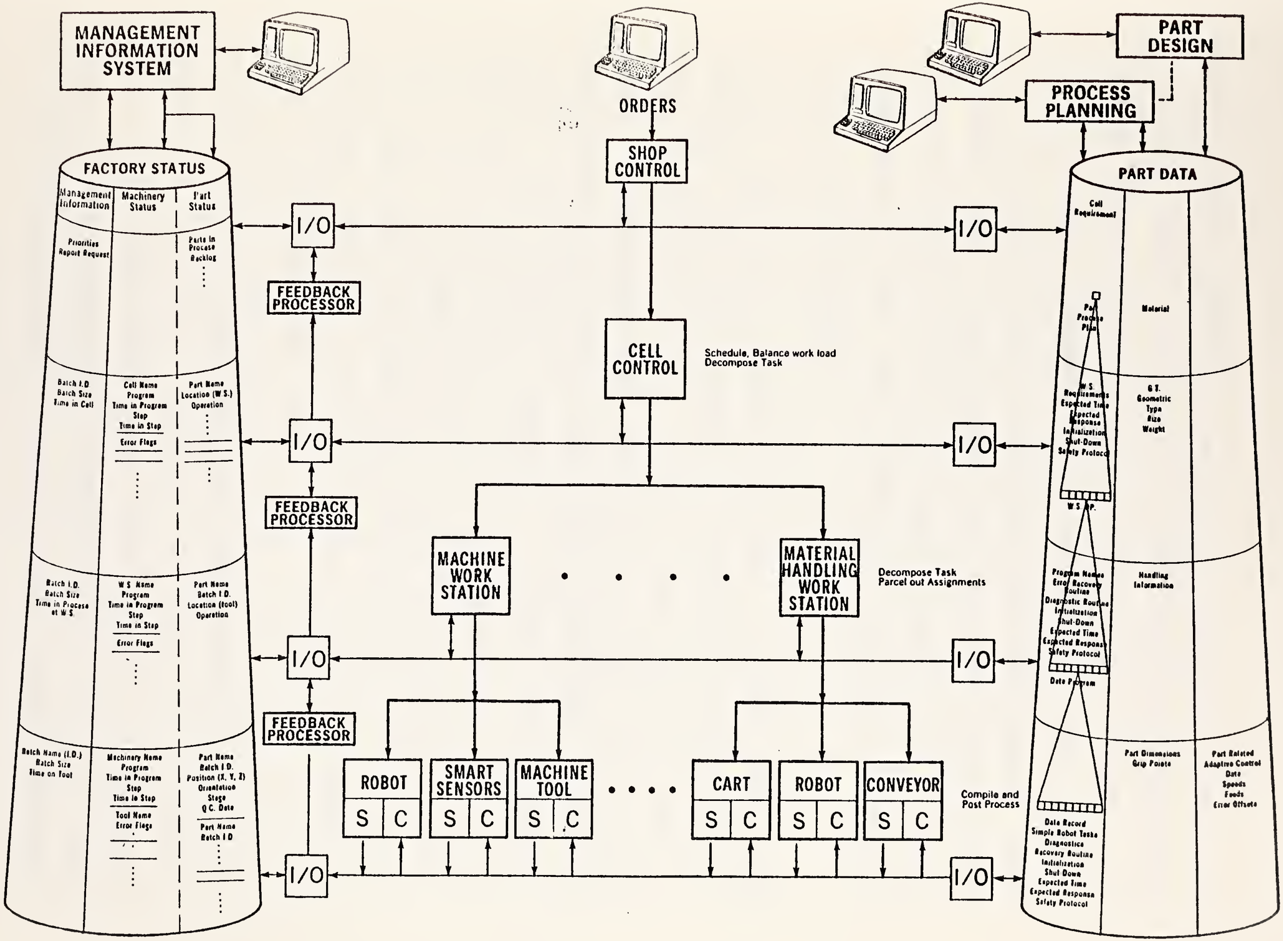

MANAGEMENT INFORMATION SYSTEM
FACTORY STATUS
Management Information
Machinery Status
Part Status
Priorities
Report Request
Parts in Process
Backlog
Batch I.D
Batch Size
Time in Cell
Cell Name
Program
Time in Program
Step
Time in Step
Error Flags
Part Name
Location (W.S.)
Operation
Batch I.D.
Batch Size
Time in Process at W.S.
W.S. Name
Program
Time in Program
Step
Time in Step
Error Flags
Part Name
Batch I.D.
Location (tool)
Operation
Batch Name (I.D.)
Batch Size
Time on Tool
Machinery Name
Program
Time in Program
Step
Time in Step
Tool Name
Error Flags
Part Name
Batch I.D.
Position (X, Y, Z)
Orientation
Stage
Q.C. Data
Part Name
Batch I.D
FEEDBACK PROCESSOR
FEEDBACK PROCESSOR
FEEDBACK PROCESSOR
I/O
I/O
I/O
I/O
ORDERS
SHOP CONTROL
CELL CONTROL
Schedule, Balance work load
Decompose Task
MACHINE WORK STATION
MATERIAL HANDLING WORK STATION
Decompose Task
Parcel out Assignments
ROBOT
S
C
SMART SENSORS
S
C
MACHINE TOOL
S
C
CART
S
C
ROBOT
S
C
CONVEYOR
S
C
Compile and Post Process
I/O
I/O
I/O
I/O
PART DESIGN
PROCESS PLANNING
PART DATA
Cell Requirement
Part Process Plan
Material
W.S. Requirements
Expected Time
Expected Response
Initialization
Shut-Down
Safety Protocol
W.S. P.P.
G.T.
Geometric
Type
Size
Weight
Program Names
Error Recovery Routine
Diagnostic Routine
Initialization
Shut-Down
Expected Time
Expected Response
Safety Protocol
Data Program
Handling Information
Part Dimensions
Grip Points
Part Related
Adaptive Control
Data
Speeds
Feeds
Error Offsets
Data Record
Simple Robot Tasks
Diagnostics
Recovery Routine
Initialization
Shut Down
Expected Time
Expected Response
Safety Protocol

FIGURE 3. A hierarchical architecture for a factory control system.

system. These individual machines are organized into work stations under the control of a work station control unit. Several work station control units are organized under, and receive input commands from, a cell control unit. Several cell control units may be organized under and receive input commands from a shop control unit, etc. This hierarchical structure can be extended to as many levels with as many modules per level as are necessary, depending on the complexity of the factory.

On the right side of Figure 3 is shown a data base which contains the part programs for the machine tools, the part handling programs for the robots, the materials requirements, dimensions, and tolerances derived from the part design data base, and the algorithms and process plans required for routing, scheduling, tooling, and fixturing. This data is generated by a Computer-Aided-Design (CAD) system and a Computer-Aided-Process-Planning (CAPP) system. This data base is hierarchically structured so that the information required at the different hierarchical levels is readily available when needed.

On the left is a second data base which contains the current status of the factory. Each part in process in the factory has a file in this data base which contains information about the position and orientation of that part, its stage of completion, the batch of parts that it is with, and qual-

ity control. This data base is also hierarchically structured. At the lowest level, the position of each part is referenced to a particular tray or table top. At the next higher level, the work station, the position of each part refers to which tray the part is in. At the cell level, position refers to which work station the part is in. The feedback processors on the left scan each level of the data base and extract the information of interest to the next higher level. A management information system makes it possible to query this data base at any level and determine the status of any part or job in the shop. It can also set or alter priorities on various jobs.

As in the microcomputer network robot control system, all information is passed from one computational unit to another via the factory status data base. Each computing unit is a finite state machine. At each time increment, it reads its input command and its feedback, computes an output which it writes in the data base, and then waits for the next time increment. Of course, in the factory control system, the time increments need not be on 28 millisecond intervals. Several seconds are adequate at the work station, and a minute or more may be adequate for updates at the cell control level.

One advantage of using data bases for communication between modules is that communication interfaces can be standardized while allowing any type of computing hardware and any variety of programming langugage to be used in the different computational modules. As long as the computational unit can read from and write to the data base, there is no other restriction on its characteristics. Thus, the system is completely modular. Different types of software and different kinds of hardware can be readily interchanged. A new robot or machine tool, or even a new work station or cell, can be added or deleted with a minimum of impact on the rest of the system.

A second advantage is that the status data base always contains a complete state description of the entire factory at all times. Activities of various modules and of the plant variables themselves can be traced and analyzed for debugging or optimization. In the event of a system crash, the factory operation can easily be restarted, because the complete factory state description is available in non-volatile storage.

This type of architecture is sufficiently modular so that the complexity of any computing unit can be kept within reasonable bounds regardless of the complexity of the overall system. If the complexity of any module grows beyond a specified limit, the task of that module can be split into two modules at the same level, or that level can be split into two levels. This type of partitioning of the control problem strongly parallels the division of labor in a factory operated by human beings. Thus, it should be possible to have hybrid factories which sometimes operate completely automatically, sometimes are partially under manual control, and sometimes are completely manual.

Much of the development of the algorithms and heuristics which reside in the individual modules of Figure 3 can be accomplished via the techniques of "expert systems." This methodology has proven very successful in a number of areas that are are comparable in complexity to manufacturing (e.g., medical diagnosis, organic chemical mass spectrometry, and analysis of acoustic signitures from geological seismology.)

While still preliminary, the modularity and extensibility of the NBS approach to the integration of robot systems into the automatic factory appears to offer great promise for practical implementation of the automatic factory. We believe that this approach will lead to designs that will not become obsolete with the expected development in the aerospace industry in the foreseeable future.

PROVIDE TECHNICAL SUPPORT TO CONTRACTORS 4.2

Task 4.2.1 Technical Consulting

In addition to reviewing contractor reports referenced in 4.1, NBS was available for consulting with contractors on proposed plans and specifications related to robotic technology.

Task 4.2.2 Field Trips

Two field trips were made to contractor facilities and plants for review and consulting.

On February 21, 1979 Dr. James Albus and Dr. Roger Nagel visited the Lockheed-Georgia Company.

On April 25-26, 1979 Dr. James Albus, Dr. Anthony Barbera, Dr. Roger Nagel, and Dr. Gordon VanderBrug visited the McDonnel Douglas Aircraft Company in St. Louis.

IMPROVE ROBOT SYSTEM PERFORMANCE AND SAFETY 4.3

Task 4.3.1 Robot Control System Development

Abstract

NBS work on hierarchical control systems is reported in this document. This report contains a technical description of the NBS microcomputer system architecture as well as a discussion of the specific hardware and bus priority circuitry used to implement the present version of this system.

NBS ROBOT CONTROL SYSTEM DEVELOPMENT

I. INTRODUCTION

For robots to operate easily and effectively in the partially unconstrained environments of manufacturing facilities, they must be equipped with control systems that have measurement and sensory capabilities as well as a straightforward user interface. This document describes an architecture for such a real-time sensory interactive control system, its user interface modules, and its implementation on a system of microprocessors.

Industrial robots are presently controlled in a manner similar to numerically controlled (NC) machine tools. Each axis or degree of freedom is programmed to move in a coordinated fashion by a set of prestored numbers representing positional values sent to servoed actuators. This type of control severely limits the potential use of those machines to a constrained work environment. Since the robot will perform a task by moving to the same locations in space each time, the positions and orientations of the workpieces must always be maintained precisely. Only if the environment is so constrained, can the robot satisfactorily perform its tasks moving through the recorded set of sequential points in space. Program points are generated by leading the robot through the task and recording the joint position values at each location. This teach method is a tedious and time-consuming operation which hampers applications in small lot batch productions.

The capabilities of industrial robots can be greatly increased by the addition of a sensory-interactive computer control system. This type of system provides real-time coordinate transformation and sensory data processing capabilities to react effectively to an unstructured environment. An off-line programming ability in a higher level English-like language enables simplification of the previously tedious task description process.

Until recently the amount of computer processing required to provide these capabilities was not economically practical for most industrial applications. However, the advent of powerful microcomputers designed from large scale integrated (LSI) circuits has provided enormous increases in computational capabilities at a very low cost. Prior to this major technological advancement, the hardware for a sophisticated robot control system would have consisted of a large, high speed processor using complex software such as real-time multi-tasking operating systems. This software was required to simulate the parallel processing of sensory data and control algorithms. Now, these processes can be carried out in parallel by a system of microcomputers.

There are several steps involved in setting up the overall structure or architecture of the system. Section II describes the design requirements that the control system satisfies. The system is partitioned into a number of simple, well-defined modules that will meet these requirements. Section III describes the two modules that make up the NBS robot control system. These modules enable separation of the different functions into logically related groups and describe the communication interfaces between them. Each of these groups can be further structured by partitioning into simpler and smaller functional blocks until it is easy to write the algorithms for each one. Section IV presents the implementation of the system, including the hardware and interfaces to external devices.

II. DESIGN REQUIREMENTS

This section will list the design requirements for developing a robot control system.

II.1 Sensory Interactive in Real-Time

First and foremost, the control system should provide sensory-interactive, goal-directed behavior that permits the performing mechanism to adapt to perturbations in the environment. To accomplish this, the system has to measure the state of the environment with sensors, process this sensory data to identify misalignments and error conditions are identified, and use this information to modify the task execution in an appropriate manner. This interaction has to be accomplished in real-time so that changes in inputs, either from sensors or from user intervention, will be responded to immediately by the control system, allowing timely modification of the behavior.

II.2 English-like Command Language

The second requirement is to provide a method of programming the robot task through a simple, English-like command language. The programmer should be able to input a procedural description of the task in much the same way he would instruct a human worker. The language should be sufficiently robot-independent to permit use with any robot capable of accomplishing the task. Further, the entry of the task description should be as interactive as possible to allow debugging and modification while testing on the robot. These abilities will allow for fast and easy programming and make it reasonable to use the robot for small batch operations where changes and new programs are frequently required. An off-line programming capability will allow for the development of robot tasks without tying up the robot hardware systems in addition to aiding in the integration of the robot into a total computer-aided manufacturing (CAM) system.

II.3 Task and Data Independence

The third requirement of a control system is that the values that specify the location points and object descriptions that are used by the robot should be separate from the program and supplied to it only at execution time. A task description, since it is a specification of a procedure, should not change from workstation to workstation, unless the configuration of the workstation or robot is different. Therefore, in the programming of the control system, the sensory and error recovery algorithms should be in a form that is independent of the robot, workstation, and computer hardware. The advantage of this approach is the transferability of a large part of the control structure to each workstation. This "portability" minimizes duplication in creating a control structure for each robot and permits the control system programmer to expend greater effort on improving a generalized control structure instead of regenerating identical control algorithms for each new robot.

The separation of task description and data can be accomplished by the symbolic naming of locations and objects which will be assigned numerical values from a corresponding data base before or at execution time. The symbolic naming does much to ease the task programmer's job. Providing named variables like VISE, DRILL or HOLE not only makes the task description more comprehensible, but relieves the task programmer of the burden of supplying numerical values when he should be specifying only a procedure. In a number of applications, there already exists a precise description of the location points that could be used by a robot program. For example, the same N/C part program that was used to cut the die for stamping out a car fender could also provide the relative location points to allow a robot to spot weld points along the flange of the fender or to spray paint the fender. Presently, this data has to be duplicated for the robot programs.

The control structure should also allow the task programmer to enter a coordinate system description of points or use the teach method if desired. However, once these points are in the data base of a robot, they should be maintained in a general format that may be used by any other robot. That is, these values should be stored as some relative coordinate reference frame values, not as the joint values of a particular robot. Thus, if one robot is replaced by another at a work station, the same data base of points should be usable and are, therefore, independent of the particular robot.

II.4 Extensibility

Due to the desired general nature of industrial robots, all of the possible control algorithms, input commands, sensors, error conditions, etc., cannot be foreseen. Therefore, the system, to be effective, must easily permit additions or deletions of functions, as well as changes to be made in existing functions. Modular design of the control structure should greatly enhance the ease and speed with which the systems programmer can incorporate changes while keeping the high degree of reliability that is an absolute requirement of the system.

II.5 Reliability

Implicit in the discussion of an effective system has been the notion of reliability. Reliability of capital intensive equipment such as industrial robots is essential to their productivity since their payback is dependent on full use. An unreliable control structure, even with a well-defined user interface and with behavior that is responsive to the environment, is useless in the manufacturing world. The overall design or architecture is important in developing a reliable control system.

A control system architecture should provide the control system programmer with a framework necessary to implement the above features in the simplest manner possible and in a way that allows the system to be easily extended. The system architecture should also provide the underlying organization to allow the control system programmer to view the overall structure and interactions of the entire control system in order to keep it understandable and comprehensible. This helps to prevent the unnecessary introduction of complexities and unknown states into the system. For example, if the visual processing of camera data were intricately interwoven with control algorithms, and a new sensor were to be incorporated, a large number of patches to the control structure would be required. This, of course, could result in an unreliable system. Thus, the goal of reliability is fundamentally impacted by the architecture of the control system.

III. SYSTEM PARTITIONING

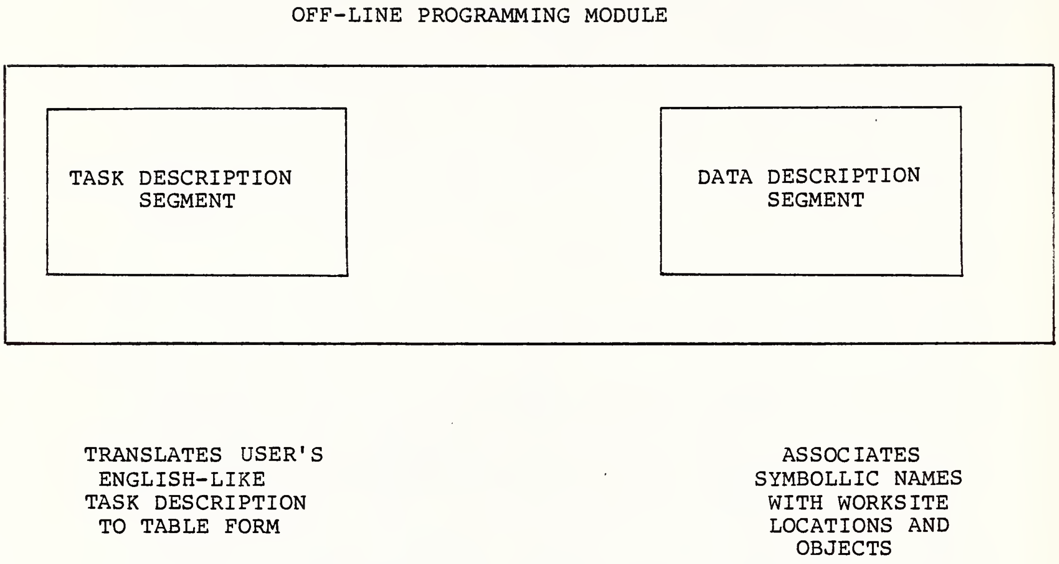
The NBS robot control system has been functionally partitioned into two modules as shown in Figure III.1. The off-line programming module provides the user interface to the control system. User inputs for task description and data specification are translated into tables that will be used by the real-time control module during execution. The real-time control module provides the actual control for the robot. User commands, worksite specifications, and sensory data are processed within this module to drive the robot to perform the desired tasks.

III.1 Off-line Programming

Through an off-line programming editor, the user describes the tasks to be done by the robot as well as the worksite environment. A procedural description of the task to be done is entered and converted into a table format by a task description segment. The data description segment is made up of two sections. The various locations within the worksite such as pallets and pieces of equipment are defined using the location section. The objects that will be manipulated during the task are described by the object description section. Figure III.2 shows the structure of the off-line programming module. All high level user interaction with the control system occurs through the task description segment and the data description segment.

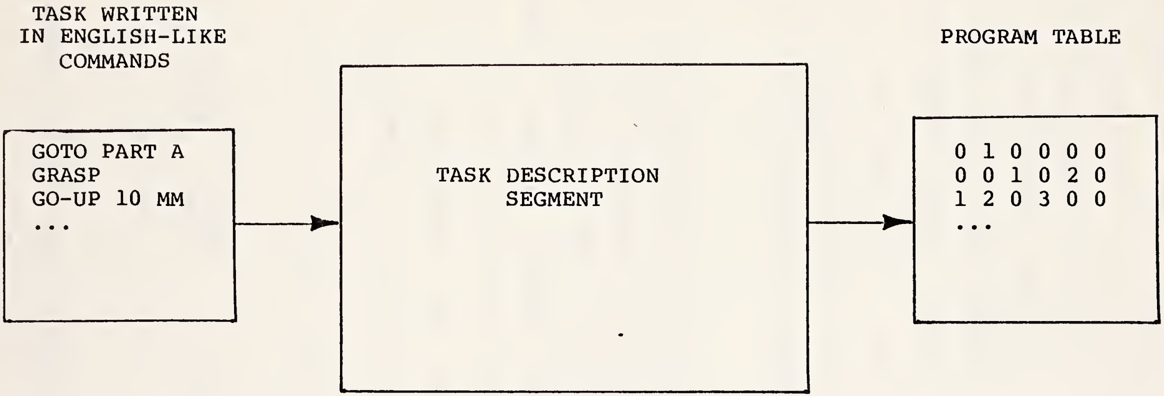
III.1.1 The Task-Description Segment

The task description segment supplies an off-line programming capability. The user enters a procedural description of the task using English-like commands. These commands can be motion statements such as GOTO or operations such as GRASP or INSERT, and can include symbolically named data such as VISE or PALLET. The program generated will be independent of the particular robot that will execute it (provided that the robot has sufficient capabilities such as weight capacity, degrees of freedom, size, etc., to perform the task.) This module can work interactively with the system in on-line and off-line modes. The task description module allows interactive debugging so that a program can be changed and immediately executed. Alternatively, the robot may be halted at some point, the program modified, and then robot execu-


OFF-LINE PROGRAMMING MODULE
TASK DESCRIPTION SEGMENT
DATA DESCRIPTION SEGMENT
TRANSLATES USER'S ENGLISH-LIKE TASK DESCRIPTION TO TABLE FORM
ASSOCIATES SYMBOLLIC NAMES WITH WORKSITE LOCATIONS AND OBJECTS

FUNCTIONS OF THE OFF-LINE PROGRAMMING MODULE

FIGURE III.2


TASK WRITTEN IN ENGLISH-LIKE COMMANDS
GOTO PART A
GRASP
GO-UP 10 MM
...
TASK DESCRIPTION SEGMENT
PROGRAM TABLE
0 1 0 0 0 0
0 0 1 0 2 0
1 2 0 3 0 0
...

TASK DESCRIPTION SEGMENT TRANSLATES A TASK WRITTEN IN ENGLISH-LIKE COMMANDS INTO A PROGRAM TABLE

FIGURE III.3

tion resumed from that or any other point in the program.

A high level user task is entered into the task description module through the off-line programming editor. This editor takes an input stream of English like commands and converts them to a program table. Execution of this table of commands will be done line by line through an interpreter within the real-time control module. Figure III.3 shows the functioning of the task description segment. In addition, the user has the ability to display the program table, edit it, and then immediately execute it through the real-time control module.

Entries into the table at the primitive-move level presently running in the hierarchy consist of motion commands (GOTO), logical location names (PALLET), object description (CYLINDER), function commands (GRASP), sensor control relative to this level in the hierarchy (FLASH-CAMERA), and other control information. The resulting program table entry for a given command has the format as shown in Figure III.4. Figure III.5 is a table of the currently implemented program table values and the meaning of each.

Location Table Pointer	Motion Command	Delta Move	Gripper Control	Camera Control	Offset Value

FORMAT FOR AN ENTRY IN THE PROGRAM TABLE

FIGURE III.4

MOTION COMMANDS

0	NULL	No Operation
1	GOTO	Move to the Location and stop there
2	GO-THRU	Move through point at maximum velocity and do not decelerate
3	P-GOTO	Move to a location within a pallet and stop there. Will automatically index to next element
4	P-GO-THRU	Move through next pallet element
5	P-GOBACK	Move to previous location within the pallet without incrementing element
6	DELTA-MOVE	Move relatively from the current position according to the next parameter type and offset amount
7	CAMERA-MOVE	Use relative offset information from camera to determine next goal. (pantograph present orientation)
8	CAMERA-ORIENT	Same as 7 except uses the orientation from the camera to rotate match the orientation of the object

DELTA-MOVES (used when motion command = 6)

0	NULL	No operation
1	GO-UP	Move up from current position
2	GO-DOWN	Move down from current position
3	FORWARD	Move forward from current position along the hand-wrist axis
4	BACKWARD	Move backward

CAMERA CONTROL		
0	NULL	No operation
1	FAR-FLASH	Long range picture-returns relative position and orientation of object
2	NEAR-FLASH	Close range picture-returns more precise position and object identification
3	ACQUIRE-FLASH	Move to a closer position based on camera data from previous flash and take NEAR-FLASH picture

GRIPPER COMMANDS	
0	NULL
1	GRASP
2	RELEASE

LEGAL VALUES FOR ENTRIES IN A PROGRAM TABLE

FIGURE III.5

The following example shows how a program table will be built for a specific application. The workspace contains 3 predefined locations within the location table.

LOCATION NAME	LOCATION POINTER
HOME	1
PICTURE-1	2
PALLET	3

where HOME is the start location for the robot; PICTURE-1 is the initial viewing location to begin searching for a part. PALLET is the ordered location where the robot should place the part after acquisition.

A sample task will be: send the robot to the start location, open the grippers so it can view the worksite, then send it to a location where it will scan for a part using a camera. If a part is within range, the robot will move closer based on the received vision data, after which it will take another picture of the part to determine its exact location. Upon successfully finding the part, the robot will acquire the part, raise it above the worksite 10 mm, and move it to the location where it will be placed. The user commands entered into the task description module to define this task will be:

```
GOTO HOME, RELEASE
GOTO PICTURE-1,  FAR-FLASH
ACQUIRE-FLASH
CAMERA-MOVE,  GRASP
GO-UP 10
P-GOTO PALLET, RELEASE
```

The editor will produce the following program table from this task description.

LOCATION POINTER	MOTION COMMAND	DELTA MOVE	GRIPPER	CAMERA	OFFSET-VAL
1	1	0	2	0	0
2	1	0	0	1	0
0	0	0	0	3	0
0	7	0	1	0	0
0	6	1	0	0	10
3	3	0	2	0	0
0	0	0	0	0	0

III.1.2 The Data Description Segment

The use of symbolic names in the robot task program defers until execution time the need to supply actual values for location points or object descriptions. This keeps the task description program independent of the robot work site. At execution time the control system module uses these symbolic location and object names as pointers into the data description tables. These tables will contain the data that specifies the actual locations and object description data relevant to the particular sensors for this worksite. The location table contains the Cartesian coordinate values that will specify the relative positions and orientations required of the robot for all of the location points in the worksite. This Cartesian description becomes the format for supplying location points to the control system module. Values from other data bases can be converted into this table form. In the object description table, all the characteristics of a given object, necessary for the particular sensors to perform object identification, such as size, grasp point or current location, are maintained and updated as the object is manipulated during a task. Figure III.6 shows the functions of the data description segment. Note that these data bases may also be obtained from other sources such as CAD data files. Separating the task description from the data specification may make the control algorithms directly transferable to many different workstations and applications merely by a change in the data base description of the worksite.

Through the data description editor, the user may define the different location points in the system as well as specify

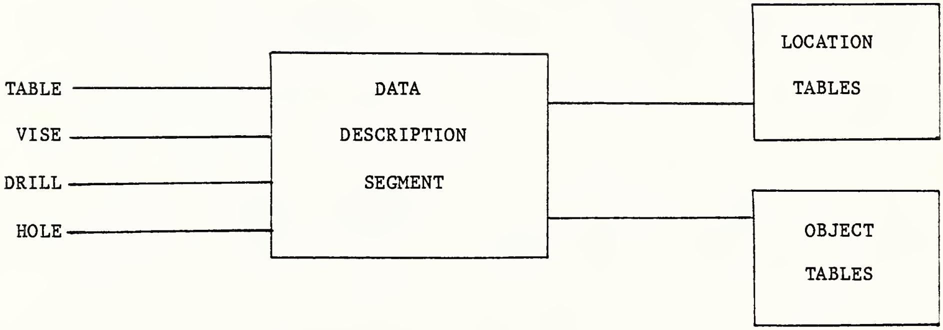


TABLE
VISE
DRILL
HOLE
DATA DESCRIPTION SEGMENT
LOCATION TABLES
OBJECT TABLES

THROUGH THE DATA DESCRIPTION SEGMENT SYMBOLIC NAMES
FOR WORKSITE LOCATIONS AND OBJECTS ARE ASSOCIATED
WITH THE ABSOLUTE DESCRIPTIONS THAT ARE KEPT IN
LOCATION TABLES AND OJBECT DESCRIPTION TABLES

DATA DESCRIPTION SEGMENT

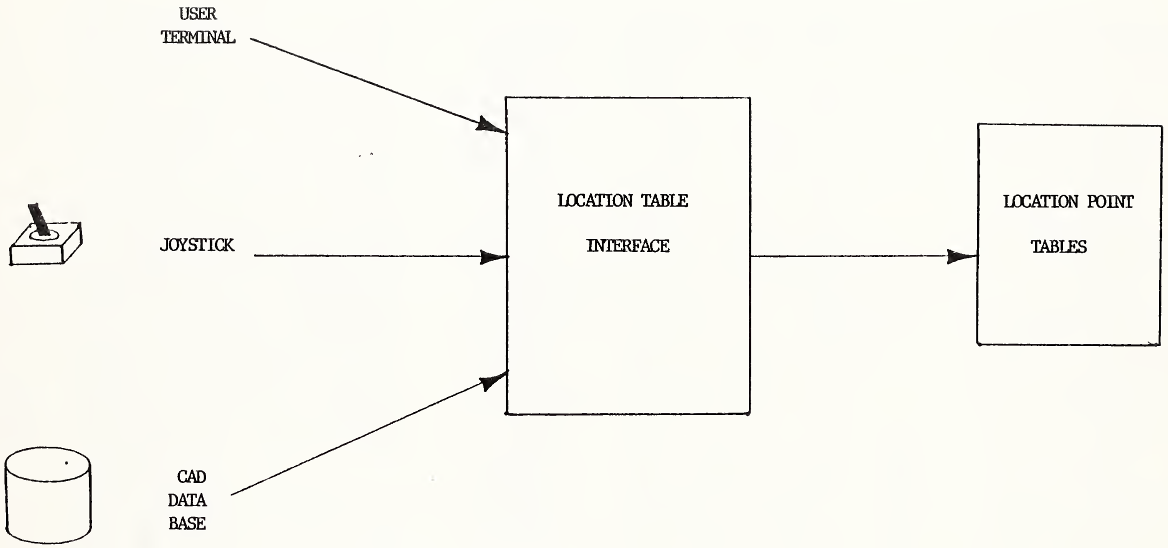
FIGURE III.6

the object characteristics that the sensor processing will need to identify that object. The data description module allows the programmer to enter the values through a terminal and assign a name to them. In addition, the location points can be generated in the traditional way by driving the robot to the location and recording the joint values, transforming them into the correct set of Cartesian values, and storing these in the table.

III.1.2.1. The Location Table

By interacting with the data description editor, a user may enter location point data by several methods: teach, direct coordinate entry, coordinate offset and array specification as shown in Figure III.7. For the teach method, the user moves the robot to the desired location through a joystick or through some sort of joint actuator control and then records the point into the location point table with a simple command. For direct coordinate entry, the Cartesian coordinates are entered by the user for that location.

Coordinate offsets may be used to define the orientation of the robot relative to its current location. For example, if the robot hand must be perpendicular to the work surface to acquire a certain object, the offsets required to move the hand from the current position to the desired position are entered as coordinate offsets in the location table. Then, when an object is found, the system will use this offset


USER TERMINAL
JOYSTICK
CAD DATA BASE
LOCATION TABLE INTERFACE
LOCATION POINT TABLES

LOCATION POINT DATA MAY BE ENTERED BY A USER AT A TERMINAL, THROUGH A JOYSTICK, OR COME FROM EXTERNAL SOURCES SUCH AS CLDATA FILES

FIGURE III.7

value to orient the hand for the acquisition.

In addition to the coordinates of the location, the location table carries other pertinent data. A flag marks the point as a pallet or a single location. For pallet points, there will be a pointer to a routine that will calculate the various elements of that pallet from the current index. There is also a pointer to a routine that will generate any associated approach path to this location. For example, if the robot must be 15 cm above the work surface and in a certain orientation in order to reach location 4, the system will access the approach path routine when the user requests GOTO 4 and generate the proper trajectory along the approach path to reach that location. The location table entry may also contain flags that will indicate if an object has been placed at that location. Figure III.8 shows the format of a location table entry and describes the entries.

III.1.2.2. The Object Description Table

The data description editor also allows the user to specify information about objects. The characteristics of the object that are stored in the table are those required by the various sensors to identify or acquire that object. For example, a vision system would need the dimensions of the object to determine if something in its field of vision could be identified as an object. A touch sensor module may re-

LOCATION TABLE ENTRY	
BYTE OFFSET	ENTRY
0-11	Name of Location (up to 12 characters)
12-29	Nine 16-bit values representing the Cartesian coordinates of the point, wrist x-y-z hand x-y-z, finger x-y-z
30	Object Flag 0 = object not there 1 = object there
32	Pointer to name of object at this location
34	Pallet Flag 0 = single location 1 = pallet location
36	16-bit pointer to routine that will generate offsets to the pallet elements
38	16-bit index indicating which pallet element is being used
40	16-bit pointer to routine to generate approach path to this location
42	16-bit pointer to routine to generate departure path from this location

FIGURE III.8

quire the maximum force that can be applied to the object, while an infrared proximity sensor module may need information regarding the reflectivity of the surface of the object. In addition, the object description table carries information such as the tool length offset for the object if the object will be used as a tool, and a pointer to the location table entry of the present and last positions of the object. If the object is one of an array of objects, the object description entry is generic and carries the value of the number of the element in the array associated with the current object. Other information that defines an object may be contained in routines that would generate pickup procedures or would define a grasp point. Figure III.9 shows the current format for object table entries.

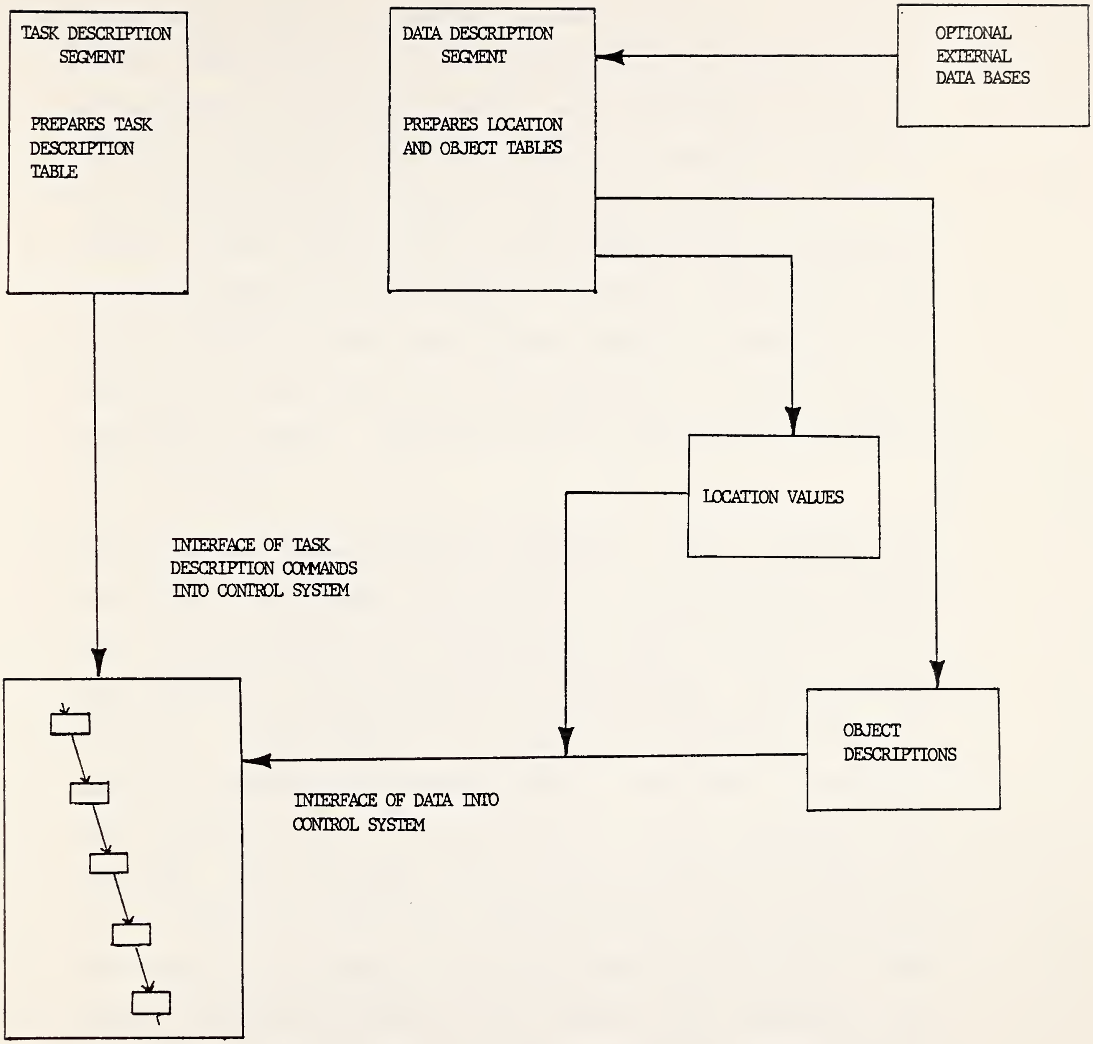
In addition to allowing the user to enter information about location points or objects, the data description editor displays the data and permits the user to modify any entries as required.

III.2 THE HIERARCHICAL REAL-TIME CONTROL MODULE

Actual execution of the robot task defined by the user is done within the real-time control module as shown in Figure III.10. This module receives the input commands through the program table and accesses the Cartesian descriptions from the location table and object descriptions from the object

OBJECT DESCRIPTION TABLE ENTRY	
BYTE OFFSET	ENTRY
0-11	Object name
12	16-bit length (or relevant dimensions)
14	16-bit height
16	16-bit width
18	Tool length offset
20	16-bit pointer to pick-up procedure
22	16-bit pointer to location table entry of where object is
24	Index of element number if in pallet
26	16-bit pointer to destination of object
28	Index of element in destination pallet
30	Number of objects transferred of this type

FIGURE III.9


TASK DESCRIPTION SEGMENT
PREPARES TASK DESCRIPTION TABLE
DATA DESCRIPTION SEGMENT
PREPARES LOCATION AND OBJECT TABLES
OPTIONAL EXTERNAL DATA BASES
LOCATION VALUES
INTERFACE OF TASK DESCRIPTION COMMANDS INTO CONTROL SYSTEM
OBJECT DESCRIPTIONS
INTERFACE OF DATA INTO CONTROL SYSTEM

OVERALL SYSTEM PARTITIONED INTO FUNCTIONALLY SEPARATE GROUPS. THE PROGRAM MODULE TO CONTROL SYSTEM INTERFACE IS IN THE FORM OF A TABLE OF TASK DESCRIPTION COMMANDS THAT FORM THE INPUT TO THE HIGHEST LEVEL OF THE REAL TIME CONTROL SYSTEM. THE LOCATION AND OBJECT DATA ARE PASSED IN THE FORM OF TABLE ENTRIES SPECIFIED IN A STANDARDIZED FORMAT.

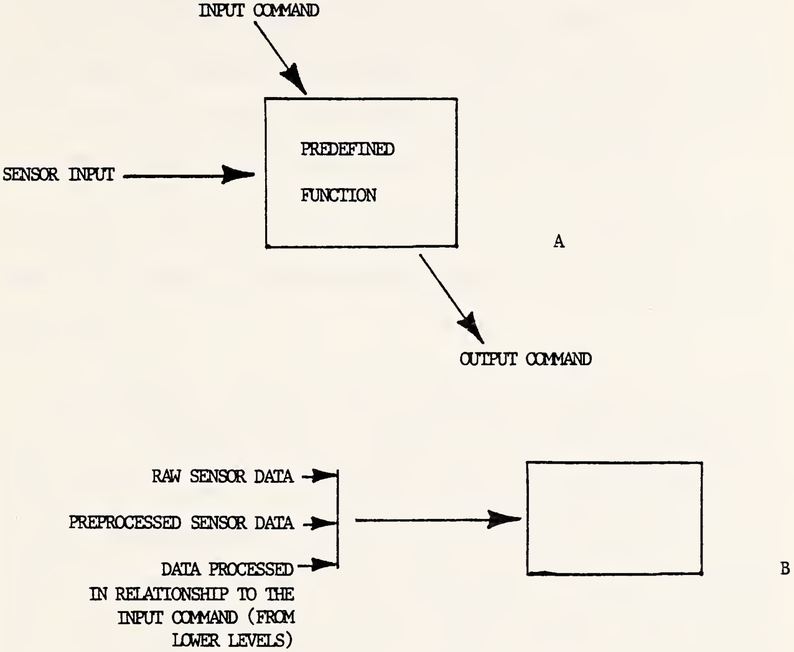
FIGURE III.10

description table. A hierarchy has been used as the fundamental architecture of this executing control system. Commands input at the highest level are decomposed into sequences of subtasks which are passed as commands to the next lower level in the hierarchy. This same procedure is repeated at each level until, at the bottom of the hierarchy, a set of outputs is generated to actuators, interlocks and signal lines to cause the necessary external response. The complexity at any level in the hierarchy is held within manageable limits, regardless of the complexity of the entire structure.

All levels of the hierarchical control system execute in real-time. This means that any input at any instant in time will cause an output in a sufficiently short time so that the stability of the system can be maintained. At each time interval, a level samples its inputs and provides a set of outputs computed from those inputs. Therefore, if sensory data indicates that a modification in a robot trajectory is required, the new output position will reflect that modification within the next time interval rather than finishing the current trajectory. The operation at each level of the control system architecture can readily be visualized by considering each level of the hierarchy as a simple finite state machine. For each level at each instant in time there exists a finite set of values for input variables to be tested and a predetermined output state to be generated upon the occurance of any particular input state. At each level inputs consist of commands from the next higher level, processed sensory feedback data from the environment, and status flags indicating the state of processing of the next lower level. Outputs from each level consist of commands to the next lower level, predictions or processing commands to the sensory-processing hierarchy and status flag feedback to the next upper level. Figure III.11 is a diagram indicating the flow of the inputs and outputs for a level in the hierarchy.

Since a robot operates under conditions that are susceptible to perturbation, the sensors must measure the environment often and the control system must process that data in order to output the correct set of actions to compensate for the conditions observed. This decision branching must occur often in order for the system to operate in real-time. At each instant or clock-tick, every level in the control system independently uses its own set of inputs (commands and feedback data) to generate the correct set of outputs (commands to a lower level, predictions to the sensory processing algorithms, and status flags.)

This organization permits each level of the hierarchy to be independent of the execution times of other levels. As long as the time increment between each sampling of the input states is short enough to avoid instability, then the levels


INPUT COMMAND
SENSOR INPUT
PREDEFINED FUNCTION
A
OUTPUT COMMAND
RAW SENSOR DATA
PREPROCESSED SENSOR DATA
DATA PROCESSED IN RELATIONSHIP TO THE INPUT COMMAND (FROM LOWER LEVELS)
B

Each level in the hierarchy has information flowing into and out of it. The output is a function of the inputs (A).

Each level's sensory input is composed of three types (B). There is raw sensor data such as the voltage levels from proximity sensors indicating relative reflected intensity.

There is preprocessed sensor data such as might come from a vision system where sophisticated data manipulation and pattern recognition is performed by some sensor unit and its output is a sensor input to the control system.

The other feedback is information from the lower levels of the hierarchy that is a reporting of their effectiveness in completing their input task. This takes the form of an interpretation of their sensor data in light of the input command. These are additional outputs from each level not as commands to the next lower level, but as inputs to the higher levels. As with the other outputs they are also predetermined functions.

FIGURE III.11

of the hierarchy can interact very simply by passing variables through a common interface. If a level changes its command to the level below, it is not critical whether the lower level picks up that change instantaneously or merely the next time it samples its input. Since every level samples its input at each time tick, there will never be more than one tick delay before the information is passed.

This real-time implementation design of a hierarchy of levels acting as simple finite state machines provides a number of advantages:

1. The highest level in the hierarchy always has control. A change in its input at any time can redirect the control system. Since each level is programmed to finish computations within a short interval in order to read its inputs at the next interval, a new command will be recognized within a short period. This real-time implementation always allows a high level reaction to any situation that might arise without the complexity of multilevel interrupts.

2. Since the outputs are updated at each interval on the basis of the condition of the set of the inputs, the system is sensory interactive in real time.

3. In this type of real-time hierarchical control system, the types of decisions each level has to make at

any instant are relatively simple. Each level merely has to provide outputs that continue to move toward the goal state. At any instant, the system does not have to plan out the entire course of action, only the correct action for the next instant.

4. Timing and interfacing constraints between levels are greatly simplified. The only timing requirement on the system is that each level of the control hierarchy sample its input parameters and generate outputs fast enough to ensure effective responses to measured events. Communication between levels is through common variables only. A level does not have to coordinate itself with another level even if processing at that level might take a substantial length of time, such as that needed by a complex sensory processing routine, like vision analysis.

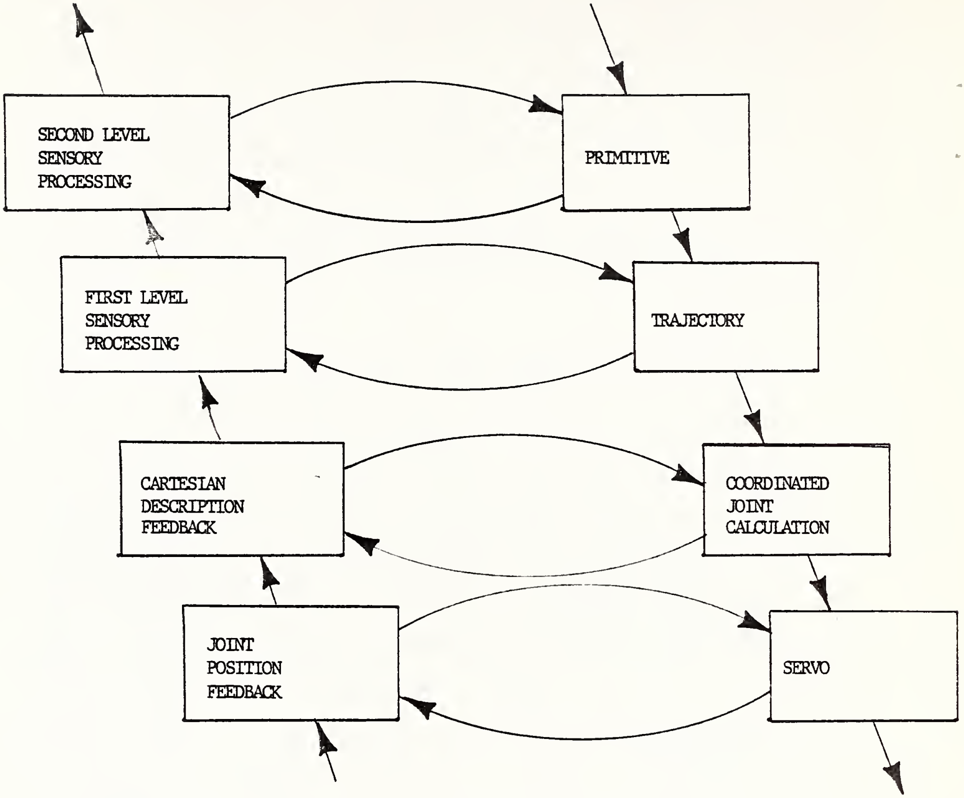
The remainder of this section examines several aspects of the real-time control module. Section III.2.1 discusses the coupling of the control hierarchy and the sensory processing hierarchies. Section III.2.2 discusses how the hierarchical decomposition allows the levels of the module to be easily visualized as a finite state machine that can be programmed by the system programmer in a straightforward manner. These programming techniques will be discussed detail. Finally, section III.2.3 describes the actual execution of the real-time control module, addressing the functions performed at each level and the interfaces between the levels.

III.2.1 Sensory Processing and Control

The real-time control system module has been separated into two parts: the control side and the sensory processing side. Each of these has been structured as a hierarchy as shown in Figure III.12.

The control hierarchy generates the proper set of drive motions which are sent to the robot to perform the task. It accomplishes this through a task decomposition. Each level decomposes its input command into a string of simpler output commands. In general, this decomposition is feedback dependent. At each instant of time, the subcommand sent to the next lower level is dependent upon the set of feedback data available at that instant. The result is that the highest level input command provides an overall goal for the system that is reached through a lengthy sequence of detailed subgoals. Each sublevel generates its outputs dependent both on the command being processed and on the present state of the environment as indicated by the sensory feedback data. Thus, the system is capable of altering its goals at each level in real-time in order to respond to changes and error states in the environment.

Sensory data is sampled at each instant in time to ensure

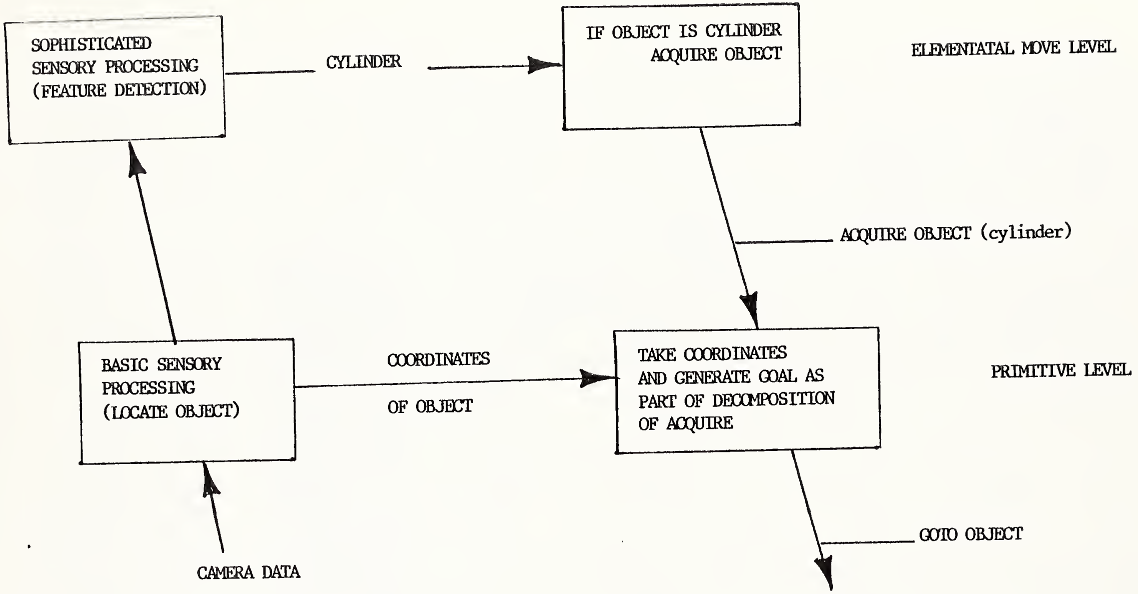

SECOND LEVEL SENSORY PROCESSING
PRIMITIVE
FIRST LEVEL SENSORY PROCESSING
TRAJECTORY
CARTESIAN DESCRIPTION FEEDBACK
COORDINATED JOINT CALCULATION
JOINT POSITION FEEDBACK
SERVO

TWO PARALLEL HIERARCHIES: ONE, A SENSORY PROCESSING HIERARCHY THAT PERFORMS DATA REDUCTION OF THE SENSORY INFORMATION IN A FORM USABLE BY THE CONTROL ALGORITHMS; TWO, A CONTROL HIERARCHY THAT PROVIDES A REAL-TIME UPDATE OF ITS OUTPUTS BASED UPON THE FEEDBACK FROM THE SENSORY PROCESSING HIERARCHY

FIGURE III.12

that the system will be responsive to the environment in real-time. The resulting values from the sensors are input to the control system module at the level in the hierarchy where their information will be acted upon. For example, the simplest camera data provides the coordinates of the location of an object in the worksite. More highly processed camera data, such as object features, provides information about the types of objects in the worksite. ACQUIRE object is a command that may be generated by the primitive level if the object is recognized as a cylinder. Therefore, the highly processed feature data should be input to this level. The control system generates new goals for the robot in terms of the coordinates that define the locations of objects at a lower level in the hierarchy. Therefore, the simple position data obtained from the camera about the cylinder should be input to that level of the control hierarchy where it will be used to generate a new goal point for the robot as shown in Figure III.13. The partitioning of the real-time control module makes it easier to program appropriate responses to sensor data because the data is input at the level at which it should be acted upon. Thus, the system will respond to its environment.

The sensory processing hierarchy also receives input at various levels from the control hierarchy. This input defines the types of sensory processing to be done and expected results. There is, therefore, a two-way exchange of


SOPHISTICATED SENSORY PROCESSING (FEATURE DETECTION)
CYLINDER
IF OBJECT IS CYLINDER ACQUIRE OBJECT
ELEMENTATAL MOVE LEVEL
ACQUIRE OBJECT (cylinder)
BASIC SENSORY PROCESSING (LOCATE OBJECT)
COORDINATES
OF OBJECT
TAKE COORDINATES AND GENERATE GOAL AS PART OF DECOMPOSITION OF ACQUIRE
PRIMITIVE LEVEL
GOTO OBJECT
CAMERA DATA

CAMERA DATA PROCESSED MORE AND MORE AS PASSED UP THROUGH THE SENSORY HIERARCHY. SIMPLE PROCESSING PROVIDES THE COORDINATES OF THE OBJECT WHICH ARE PASSED TO THE PRIMITIVE LEVEL THAT WILL USE THOSE COORDINATES TO GENERATE A GOAL POINT. FURTHER UP THE SENSORY PROCESSING HIERARCHY THE CAMERA DATA IS PROCESSED TO PROVIDE PART RECOGNITION. THE ELEMENTAL MOVE LEVEL (AS YET UNIMPLEMETED IN OUR SYSTEM) USES THAT INFORMATION TO GENERATE 'ACQUIRE' THE PART ONCE IT IS RECOGNIZED.

FIGURE III.13

information between these two hierarchies at all levels as shown in Figure III.14.

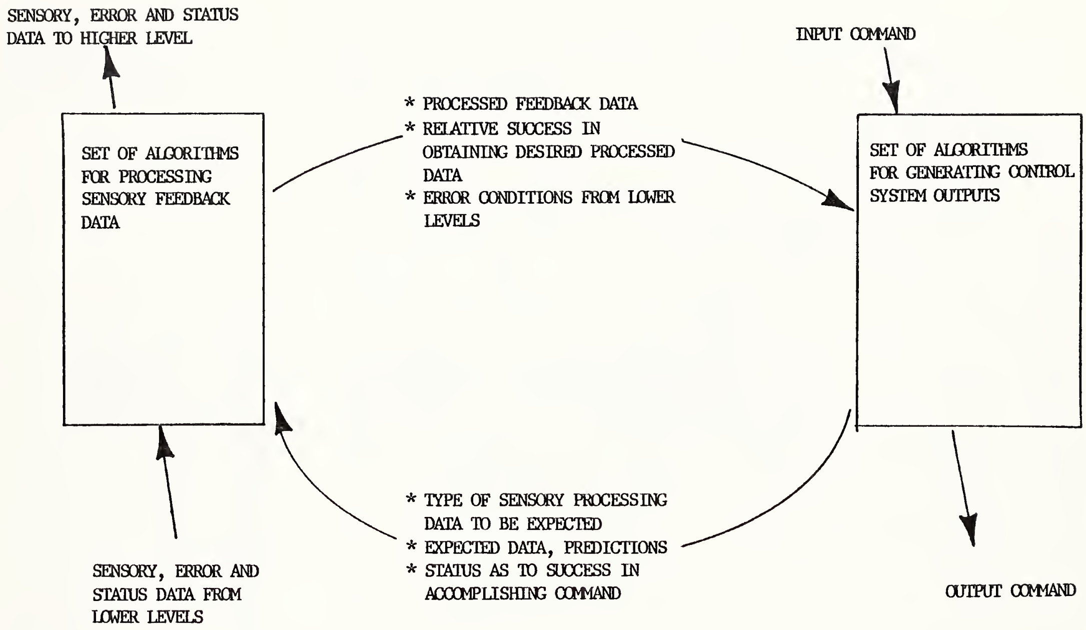
III.2.2 Programming the Real-Time Control Module

In order that the task described by the user may be executed, the real-time control module must have been programmed to interpret and execute the commands sent in the program table and to extract the location and object data associated with them. Another off-line programming interface with the system is required in order that a systems programmer may specify commands that the task programmer will execute. Figure III.15 shows where in the system this programming interface will occur.

Since each level in the hierarchy acts like a finite state machine, programming the real-time control module is actually just specifying the production rules for a finite state machine. The system programmer's interface, therefore, is a means of entering the rules for each state within the hierarchy.

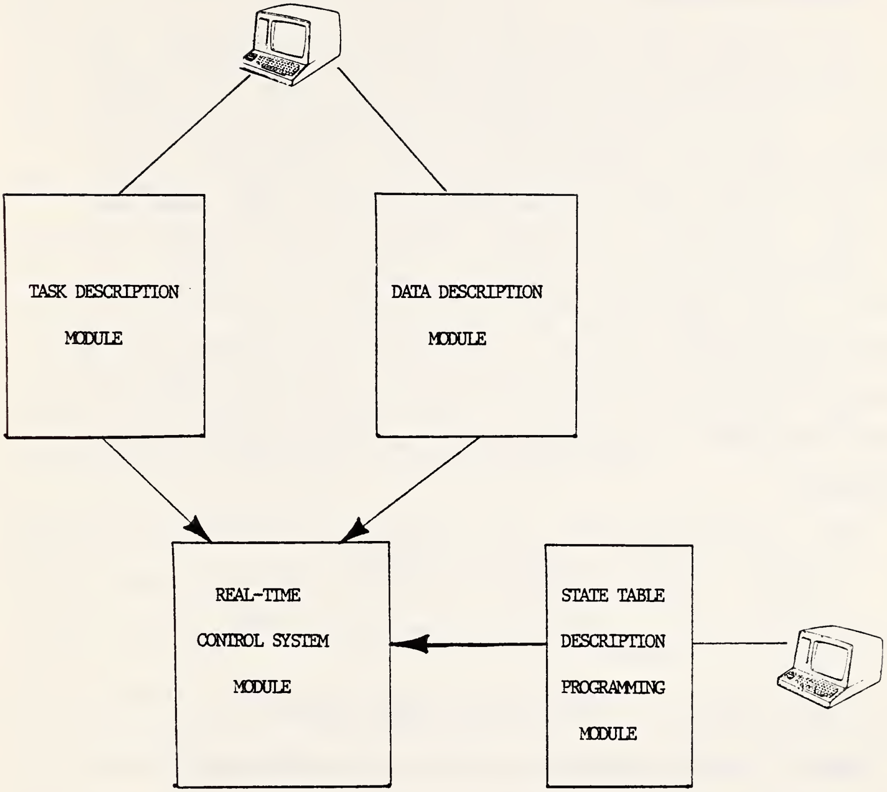
Production rules are usually of the form

```
IF (condition 1) THEN DO A
ELSE
        IF (condition 2) THEN DO B
        ELSE ...
```


SENSORY, ERROR AND STATUS DATA TO HIGHER LEVEL
INPUT COMMAND
SET OF ALGORITHMS FOR PROCESSING SENSORY FEEDBACK DATA
* PROCESSED FEEDBACK DATA
* RELATIVE SUCCESS IN OBTAINING DESIRED PROCESSED DATA
* ERROR CONDITIONS FROM LOWER LEVELS
SET OF ALGORITHMS FOR GENERATING CONTROL SYSTEM OUTPUTS
* TYPE OF SENSORY PROCESSING DATA TO BE EXPECTED
* EXPECTED DATA, PREDICTIONS
* STATUS AS TO SUCCESS IN ACCOMPLISHING COMMAND
SENSORY, ERROR AND STATUS DATA FROM LOWER LEVELS
OUTPUT COMMAND

DETAILED VIEW OF THE TYPES OF INTERACTIONS BETWEEN THE SENSORY AND CONTROL HIERARCHIES AT ONE LEVEL

FIGURE III.14


TASK DESCRIPTION MODULE
DATA DESCRIPTION MODULE
REAL-TIME CONTROL SYSTEM MODULE
STATE TABLE DESCRIPTION PROGRAMMING MODULE

TASK PROGRAMMERS ENTER ROBOT TASK DESCRIPTIONS THROUGH THE TASK DESCRIPTION AND DATA DESCRIPTION MODULES. SYSTEM PROGRAMMERS ENTER THE PROGRAMMING FOR THE REAL-TIME CONTROL SYSTEM THROUGH THE STATE DESCRIPTION MODULE.

FIGURE III.15

They usually involve some sort of language and thus may be thought of as a program. Changing a condition, or adding a new production, usually implies some sort of recompilation of existing programs.

The system programmer interface may be greatly simplified by using a table-driven method for generating the production rules or functional outputs for a level. Instead of entering the productions or commands in the IF, THEN, ELSE format, the conditions to be tested become relative table entries. The resulting outputs generated for a given set of conditions are also entries in the table. For example, for a system that must check conditions X, Y, and Z and react with output commands A and B the following table would represent a program.

	X	Y	Z	OUTPUT
1.	< 0	= 1	< 0	A
2.	> 0	= 0	< 10	B

The equivalent production rules would be

```
IF (X < 0) AND (Y = 1) AND (Z < 0) THEN OUTPUT = A
   ELSE IF (X > 0) AND (Y = 0) AND (Z <10) THEN OUTPUT = B
```

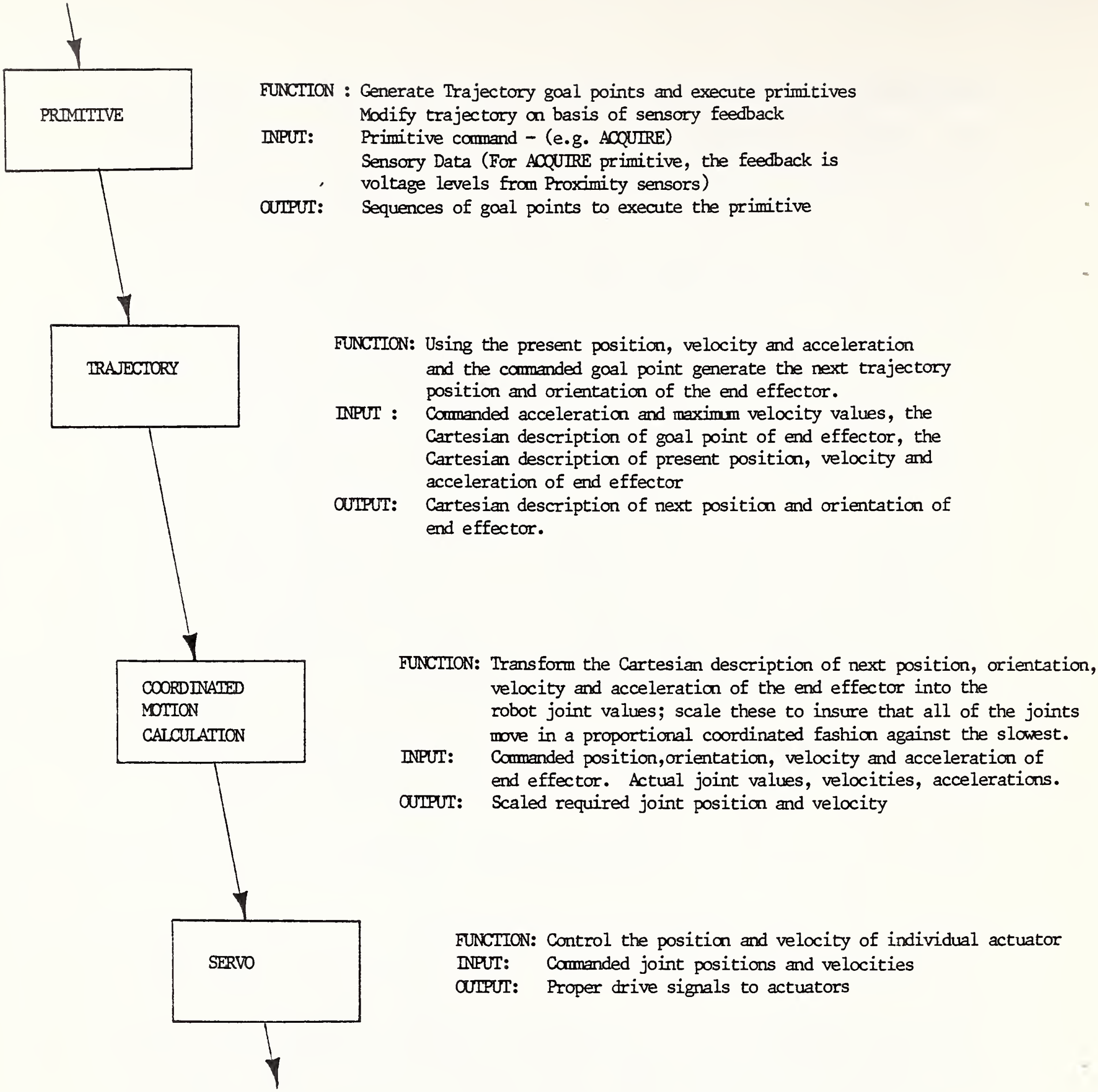
At each level of the control system module, a monitor will execute the entries in the table once per cycle, testing each of the specified conditions line by line and generating the corresponding outputs when all conditions are met on a given line.

Programming a level is straightforward. An editor prompts the system programmer with the identified variables to be tested. The programmer enters the state for the variables and then the values for the outputs when those conditions are met. Adding new variables, either input or output, to the system is done using the editor to add new columns to the table. The programming or reprogramming of a level is the process that builds the table and defines the order of condition testing and the proper outputs to generate. Adding new input or output variables means simple table modification. Since the system has been decomposed to comprehensible modules through the hierarchy, the number of inputs and outputs at each level is small. Therefore the table size is also small and easy to specify.

Adding new sensors to the system is straightforward. The sensor data provides a new condition to be tested and a new output may be added to respond to this data.

III.2.3 Real-Time Control Module Execution

The current implementation of the real-time control module has been decomposed into 4 levels as shown in Figure III.16. The first sub-section describes the function of each level in the hierarchy and the second specifies the interfaces between the levels.



FUNCTION : Generate Trajectory goal points and execute primitives
Modify trajectory on basis of sensory feedback
INPUT: Primitive command - (e.g. ACQUIRE)
Sensory Data (For ACQUIRE primitive, the feedback is voltage levels from Proximity sensors)
OUTPUT: Sequences of goal points to execute the primitive

FUNCTION: Using the present position, velocity and acceleration and the commanded goal point generate the next trajectory position and orientation of the end effector.
INPUT : Commanded acceleration and maximum velocity values, the Cartesian description of goal point of end effector, the Cartesian description of present position, velocity and acceleration of end effector
OUTPUT: Cartesian description of next position and orientation of end effector.

FUNCTION: Transform the Cartesian description of next position, orientation, velocity and acceleration of the end effector into the robot joint values; scale these to insure that all of the joints move in a proportional coordinated fashion against the slowest.
INPUT: Commanded position,orientation, velocity and acceleration of end effector. Actual joint values, velocities, accelerations.
OUTPUT: Scaled required joint position and velocity

FUNCTION: Control the position and velocity of individual actuator
INPUT: Commanded joint positions and velocities
OUTPUT: Proper drive signals to actuators

THE 4 LEVEL CONTROL HIERARCHY

FIGURE III.16

III.2.3.1 Goal Decomposition

The Primitive Level (4)

The primitive level receives commands which are simple motion specifications and a pointer to the location description of the point associated with that motion. Using the various sensor data inputs, this level constructs the goal point required to execute the command.

The goal point is built by establishing a Cartesian representation of a reference point and describing a set of offsets that may be added to that point to produce the position and orientation that will be the goal point.

For example, a command to put an item in a particular pallet location would build a goal point that would specify the origin of the pallet plus the relative coordinate offsets required to go from this point to the element within the pallet to be filled. Imbedded in the location table description for a pallet would be the coordinates of the point which is the origin of the pallet, the index value of the position (which element in the pallet is being used at this time), and values that describe the relative offsets to be used to calculate the coordinates of the indexed element. When a pallet location is specified in a command, the primitive level passes the coordinates of the origin of the pallet, and using the index value, builds an offset table that

gives the offsets to reach the required pallet element.

Sensors on the robot or joystick input may provide relative offset values to describe movement. For these motions, the present position of the robot is used as the reference position and a new goal point position is calculated as the sum of the reference position and the offset data values.

The Cartesian coordinates of a start location and the table description of offsets in any of several reference frames are used to generate the Cartesian coordinates of the goal point. This is done by transforming the offset descriptions from the given frames of reference into the equivalent offset in a standard frame of reference (the robot worktable) and adding these offsets to the start location coordinates. These final coordinates represent the intended goal point. Thus the start location and sequences of offsets can be used to describe any goal point in the robot workspace.

Entries in the offset table consist of an index to the coordinate system in which the motion is described and the magnitude and direction of motion in that system. The different coordinate systems are:

1 - the worksite (standard reference system)
2 - hand Cartesian
3 - hand spherical
4 - wrist spherical
5 - finger cylindrical

Worksite and hand Cartesian motions may be in x, y, or z directions. Hand or wrist spherical motions are rotations about axes either parallel or perpendicular to their origin. Finger cylindrical is the rotation of the hand about the hand-wrist axis. Thus, offset table entries will have the form:

1	x-dist	y-dist	z-dist
2	x-dist	y-dist	z-dist
3	degrees parallel	degrees perpendicular	
4	degrees parallel	degrees perpendicular	
5	degrees		

For example, the start location may be the origin of a pallet. The offset table values to get to the nth element of the pallet may describe the offset in terms of a rotation with respect to the wrist origin. The offset is translated into the equivalent worksite offset coordinate and is added to the absolute worksite coordinates of the origin. This, then, will be the position of the nth element of the pallet.

In addition to calculating the new goal point from the offset table, this level controls when that goal point is sent to the robot by interpreting the flags sent as status from the next lower level and by determining if the current goal point has been reached. Joystick inputs are sent out immediately to alter the current motion of the arm. Most control system inputs are held until the previous goal point is reached, then the new goal point is sent.

In addition to the goal point, output data from this level also includes trajectory generation flag information such as velocity and acceleration specifiations, and gripper control flags.

The Trajectory Level (3)

The next level in the hierarchy is trajectory generation. Given the input parameters of a goal point, velocity and acceleration information from the primitive level, and the current position of the robot determined from servo input, the trajectory level calculates an incremental move along a straight line path between the current location and the goal. The move calculated is that to be made by the end point of the robot hand. In addition, this level calculates the motion to be made by the robot wrist and fingers to cause a smooth transition from the present orientation to that of the goal point. Since all of the parameters are input each cycle, the robot can be made responsive to different goal points immediately and can also have its motion controlled within a trajectory by modifying the velocity and acceleration factors from upper levels. The motion is also controlled through a smoothing function, which according to the size of a limit variable, controls the degree of sharpness of trajectory change. The output from this level is a

Cartesian representation of the next position of the robot along a path from the current position toward the input goal position. Appendix A gives a detailed discussion of the trajectory calculations and the pseudo-code representation of the currently running programs for this level.

The Coordinated Joint Motion Calculations (2)

This is the first level in the system that is robot dependent. The Cartesian representation of the next position of the robot generated by the trajectory level is taken and transformed into the actual joint position representations of that point.

Next, given this new joint position data and the joint position data from the last cycle, the requested motion is tested to make sure that joint velocities and accelerations do not exceed a maximum amount while maintaining coordinated movement. That is, each joint should be making an appropriate delta move so that all appear to move toward the goal smoothly, and no one joint has to 'catch-up' after the rest have reached their final orientation at the goal point. All joint motion will be scaled down to permit the slowest responding joint to perform in a coordinated fashion with the other joints.

The final calculated joint position is then interpolated into 4 intermediate positions so that data may be output to the joint servos at a frequency sufficient to maintain stability. Appendix B shows the pseudo-code representation of the programming for the coordinated motion part of this level in the current implementation.

The I/O (Robot Interface or Servo) Level (1)

The lowest level in the hierarchy is the actual interface to the servos. Inputs to this level are 4 sets of interpolated joint position data which are output to the robot in 7.5 msec intervals. In addition this level reads in the current position of the robot as well as switches and other sensory data. These data are scaled as required and output for use by the other levels of the hierarchy. It is important to note that all the servo and sensory data is read in and output to the system each time interval and all levels have access to the sensory data that influences their processing. If software servoing is to be done for the system, it will be done at this level.

III.2.3.2 Data Interfaces

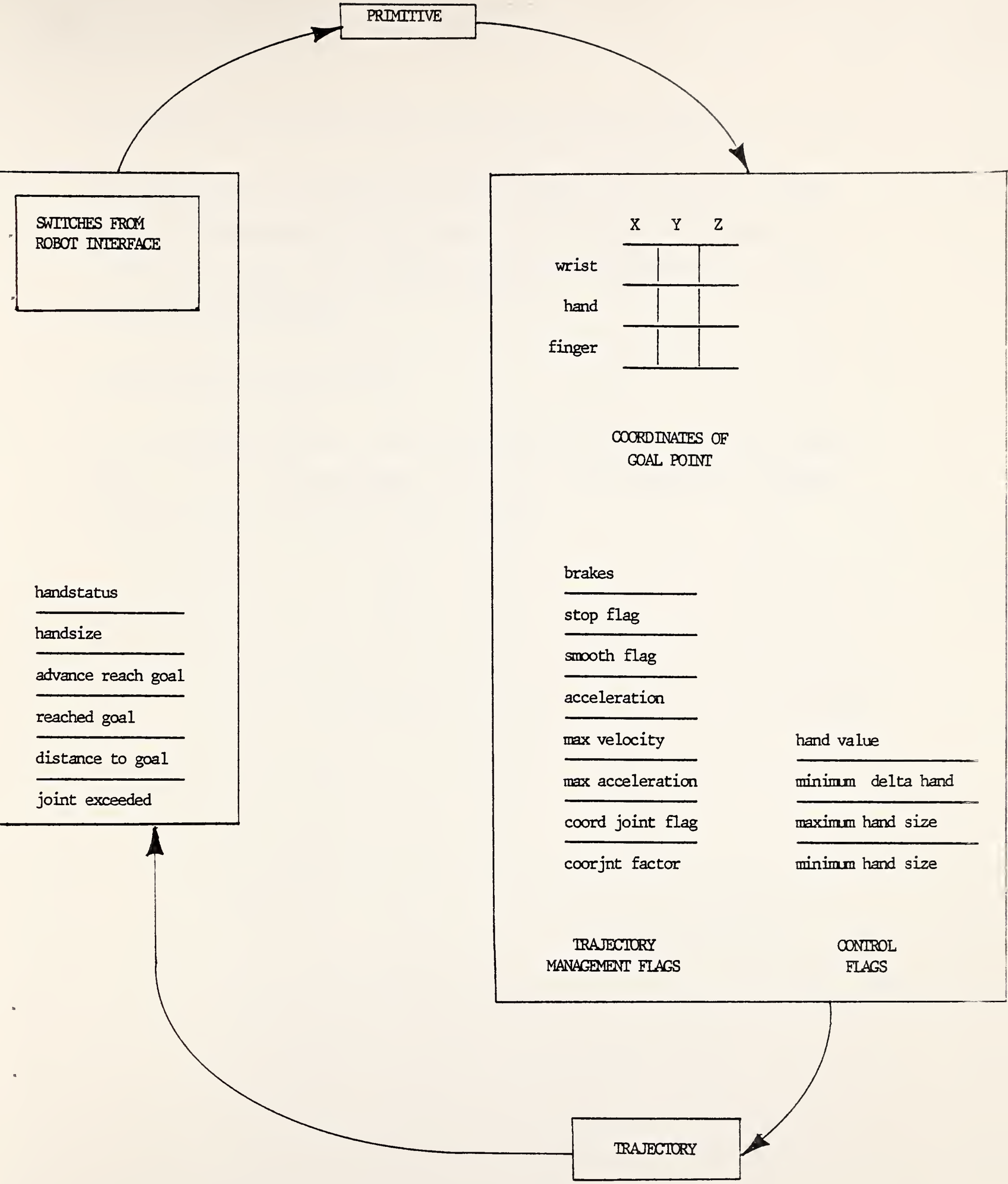
This section describes the data that is passed between the levels, that is, the interfaces.

Task Description to Primitive

Inputs to a primitive level consist of a program table entry which contains pointers to entries in the location and data description tables. The formats for entries in these tables have been shown in Figures III.8 and III.9. In addition sensor information from the joystick, vision system, touch, proximity and force sensors also is input to this level. Status received from the trajectory level includes a flag indicating that the goal point has been reached, gripper status, switch values from the robot interface.

Primitive to Trajectory

The interface into the trajectory level contains the Cartesian coordinates in the worksite reference frame of the goal point generated by the primitive level and the flag data for trajectory control. In addition this level receives the Cartesian representation and orientation of the current position of the robot end effector determined by coordinate transformations of the current joint positions. The status information passed back to the transform level consists of flags indicating that the robot is either almost at the goal, or at the goal. Figure III.17 shows this data.


PRIMITIVE
SWITCHES FROM ROBOT INTERFACE
handstatus
handsize
advance reach goal
reached goal
distance to goal
joint exceeded
X
Y
Z
wrist
hand
finger
COORDINATES OF GOAL POINT
brakes
stop flag
smooth flag
acceleration
max velocity
max acceleration
coord joint flag
coorjnt factor
TRAJECTORY MANAGEMENT FLAGS
hand value
minimum delta hand
maximum hand size
minimum hand size
CONTROL FLAGS
TRAJECTORY

DATA INTERFACE BETWEEN PRIMITIVE AND TRAJECTORY LEVELS

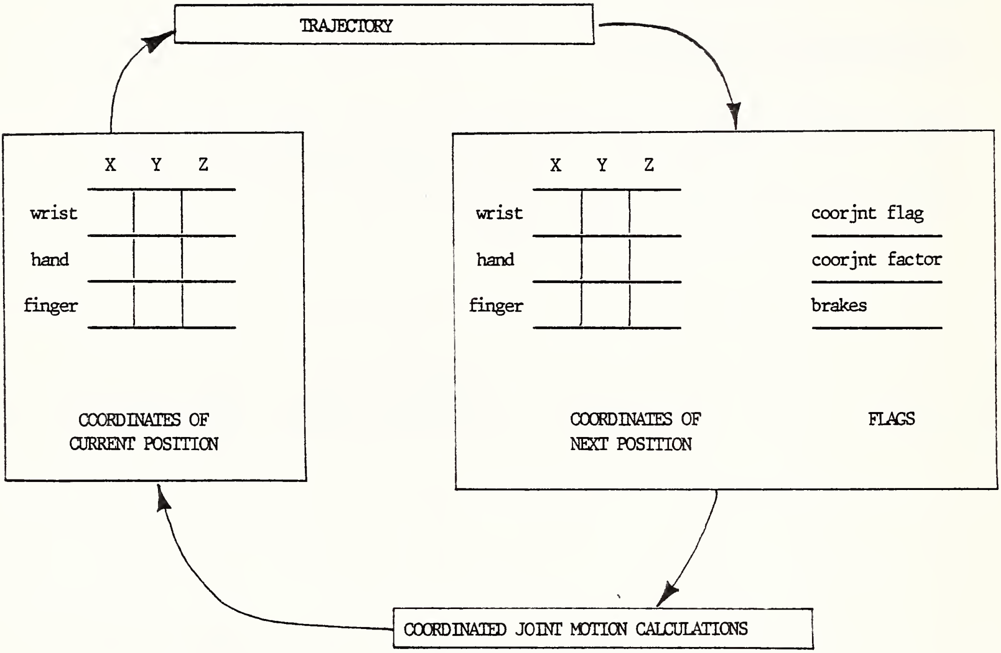
FIGURE III.17

Trajectory to Coordinated Joint Motion Calculation

The input to this level is the Cartesian work site coordinate representation of the goal to be output and flags indicating the amount of scaling required. Figure III.18 shows this data interface.

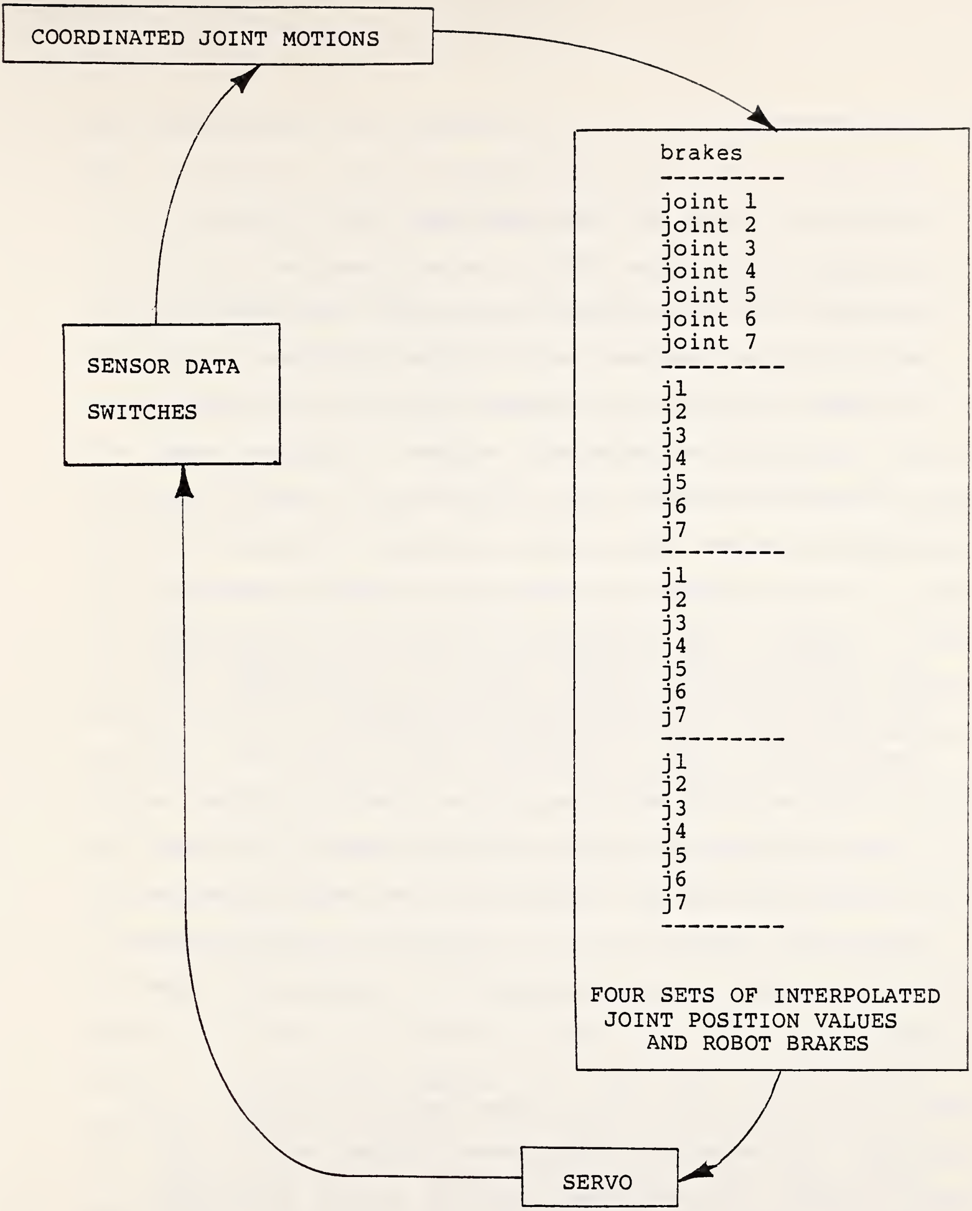
Coordinated Joint to I/O

The interface between the coordinated joint level and the servo is 4 interpolated sets of 7 joint position values. Figure III.19 shows the data layout.


TRAJECTORY
X Y Z
wrist
hand
finger
COORDINATES OF CURRENT POSITION
X Y Z
wrist
hand
finger
coorjnt flag
coorjnt factor
brakes
COORDINATES OF NEXT POSITION
FLAGS
COORDINATED JOINT MOTION CALCULATIONS

DATA INTERFACE BETWEEN TRAJECTORY AND COORDINATED JOINT MOTION CALCULATION LEVELS

FIGURE III.18


COORDINATED JOINT MOTIONS
brakes
joint 1
joint 2
joint 3
joint 4
joint 5
joint 6
joint 7
j1
j2
j3
j4
j5
j6
j7
j1
j2
j3
j4
j5
j6
j7
j1
j2
j3
j4
j5
j6
j7
FOUR SETS OF INTERPOLATED JOINT POSITION VALUES AND ROBOT BRAKES
SENSOR DATA
SWITCHES
SERVO

DATA INTERFACE BETWEEN COORDINATED JOINT MOTION CALCULATION AND SERVO LEVELS

FIGURE III.19

IV. IMPLEMENTATION

A robot control system has been described in terms of a hierarchical architecture which can be viewed as a set of finite state machines processing in parallel and communicating with each other through a well-defined interface of common variables. These discrete processing modules are made to execute on separate processors coupled through a common memory communication link. The availablity of 16-bit microprocessors along with a bus structure that allows a number of these microprocessors to access common resources, such as memory, has presented the option of a relatively low-cost multi-processor system meeting the hardware requirements of the control system architecture.

Implicit in the design of the real-time control system is the concept of responsiveness to the environment. The control of the robot and its interaction with the enviroment is critically sensitive to. A response must be supplied quickly for a given input, especially for reactions to safety sensors.

There are two ways to react to these external events. In one case, the occurrence of an event triggers an interrupt into the system. The interrupt handling routine determines the priority of the event and either records it with relevant data or else branches to routines to process the data and generate the response at that time. Control is then returned to whatever program was executing at the time of the event. Since the presence of data is totally unpredictable, branching to an interrupt handler may occur anywhere within the control system processing. It becomes difficult to maintain sequencing of events within the system especially as more and more sensors are added. In addition, the more interrupt routines that execute, the higher the probability that the state of the system will not be exactly restored after the event is responded to. The system loses its comprehensibility and may become unreliable.

The other method uses a polling technique that requires the system to read the state of external devices at periodic intervals. This data is then processed in some predetermined manner. If an event has occurred since the last polling, its changed data will be read in during the present poll. The procesing algorithms will use this new data to calculate a proper response. Unlike the interrupt method, this method processes all data in a totally predictive manner and sequence. Therefore all possible states of the system are known. The system responds to the data in an orderly fashion, rather than having the response driven by the data. The main requirement of this technique is that the system must sample the data space sufficiently soon after the occurrence of an event and be able to finish calculations in a short enough time interval to maintain system stability and ensure an effective response.

The present system is implemented using a time interval of 28 milliseconds. Every 28 milliseconds all levels of the control hierarchy will sample their input space and respond to those inputs. Algorithms to process the data collected during each time interval execute within that interval. If the processing time for any module exceeds this limit, the algorithms may be divided over several processors to get the job done.

The system is implemented on several microprocessors connected to a common bus. The structure of these processors and the way they interface through the bus is described in section IV.1. A software protocol synchronizes the passing of data between processors and regulates the processing done within the system. This data transfer and process control mechanism for the multiprocessor system is presented in section IV.2. Since the implementation results in a synchronous processing system, a hardware data collection interface has been designed which will buffer asynchronous data from external devices in a separate memory area accessible to the system. This interface will manage all data transfers between these devices and then present the data to the control system in a synchronous manner. This interface is described in section IV.3.

IV.1 THE HARDWARE IMPLEMENTATION

All of the microcomputer boards used in this implementation are Intel iSBC * 86/12-A's utilizing the 8086 16-bit microprocessor chip. The Intel MULTIBUS * is used as the common bus structure. The primary features of this hardware and their relationship to the control system and communications protocol software are described in the following sections.

IV.1.1. The 16-bit Microprocessor Board

The 16-bit processor used in this implementation has hardware multiply and divide instructions, as well as instructions for string operations, and an extended range of addressing modes. It is capable of directly addressing 1 megabyte of memory (20 bits of address). It has a cycle time of 200 nsec., with a typical memory read or write cycle taking 1.8 μsec.

---

* iSBC and MULTIBUS are trademarks of the Intel Corporation. This equipment is identifed by brand name in order to adequately describe the functions performed. In no case does such identification imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that these products are necessarily the best available for the purpose.

Each 16-bit microcomputer board has a a RS232 compatible serial I/O port, three programmable timers, circuitry for up to eight levels of vectored priority interrupts, and 24 programmable I/O lines. The 24 lines of I/O are configured as 16 lines of data, and eight lines for handshaking signals and miscellaneous I/O. An add-on board has been built to expand the I/O capabilities of this microcomputer board to 48 lines. This extra board allows separate 16-bit data input and 16-bit data output from one board. These lines are used for communications with external devices such as sensors, the robot, and vision systems, either separately or through the hardware data collection interface.

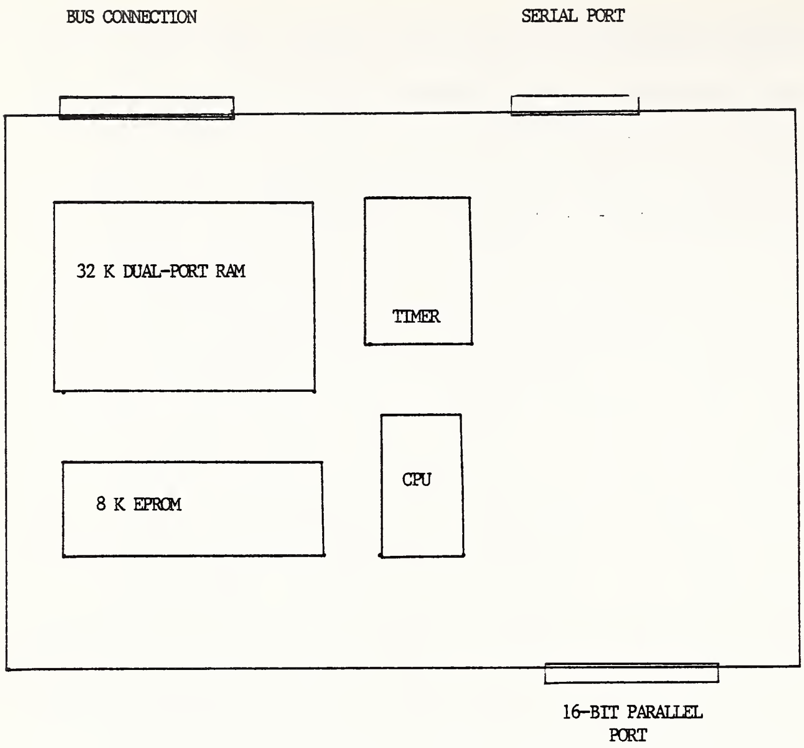
The serial I/O port is used to provide a separate interface to each computer board. Through this interface, object code can be downloaded from other computer systems or a terminal can be connected to make use of onboard monitor programs for debugging. The baud rate for this port originates from one of the programmable timers. A software wait instruction is available which suspends computation until an external signal is applied to one of the computer input lines. This suspension is used to provide a mechanism for system synchronization which will be described in section IV.2.

A 16-bit data bus enables the CPU to access any other board in the system as if it were an extension of its own memory space. When an off-board data transfer is required, the CPU requests use of the bus and is put into a wait state until the bus is available. Once the bus is acquired, the CPU resumes execution and performs the data transfer. I/O addressing is similiar, with the on-board addresses fixed and all others generating a bus request. As with memory, the CPU enters a wait state until the I/O device responds. A time-out feature can be implemented to avoid the situation of locking the CPU in an infinite wait for a nonexistent or failed device. After a specified time, an interrupt is sent from a hardware timing circuit to the CPU, which clears the wait state and starts execution of appropriate action. Figure IV.1 shows the logical layout of a microprocessor board and the interconnection of several boards through the system bus.

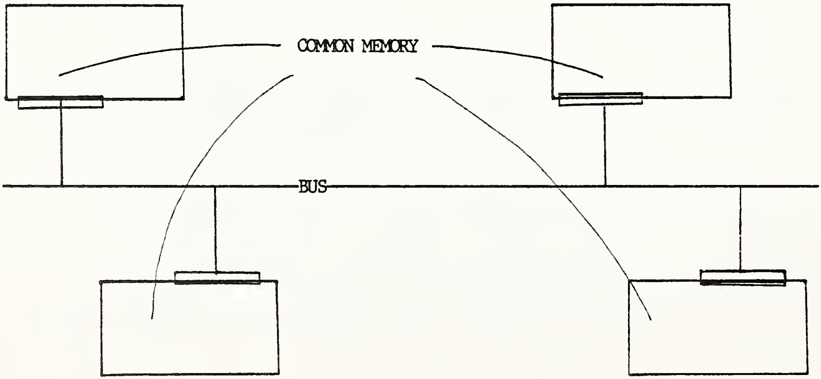
This single board computer has 8K bytes of fixed address erasable programmable read-only memory (EPROM) and 32K bytes of dual-port dynamic random access memory (RAM). Both RAM and ROM are word and byte addressable.

IV.1.2. Dual-Port RAM

The dual-port RAM in Figure IV.2 is accessible to the CPU of the board on which it is resident and, through jumpers, may be made accessible to external processors on the common bus. The addressing is fixed with respect to the CPU and is selectable to the other processors on the bus in 8K segments anywhere in the 1 megabyte range. When access to an RAM is

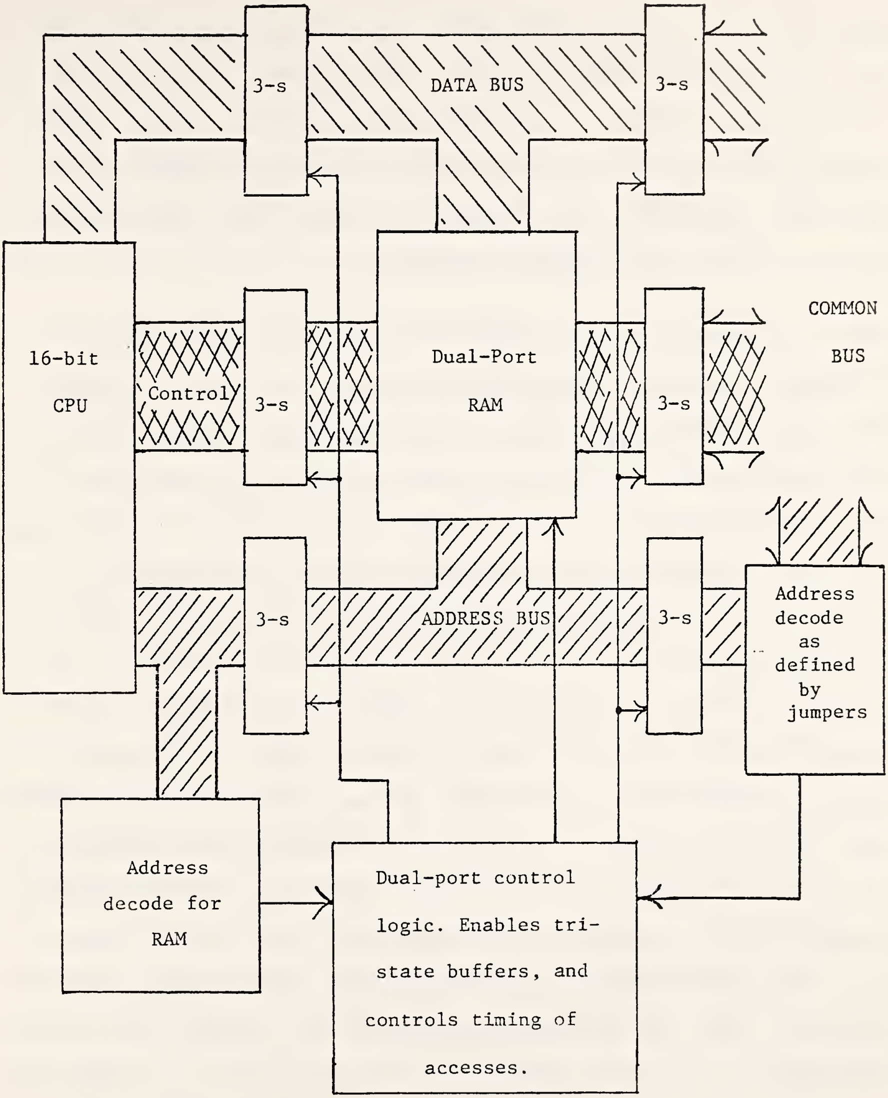

BUS CONNECTION
SERIAL PORT
32 K DUAL-PORT RAM
TIMER
8 K EPROM
CPU
16-BIT PARALLEL PORT

LAYOUT OF A 16-BIT MICROPROCESSOR BOARD SHOWING RAM AND EPROM MEMORY SPACE, PARALLEL AND SERIAL I/O PORTS.


COMMON MEMORY
BUS

SEVERAL 16-BIT MICROPROCESSOR BOARDS CONNECTED TO A COMMON BUS STRUCTURE

FIGURE IV.1


3-s
DATA BUS
3-s
COMMON
BUS
16-bit
CPU
Control
3-s
Dual-Port
RAM
3-s
3-s
ADDRESS BUS
3-s
Address
decode
as
defined
by
jumpers
Address
decode for
RAM
Dual-port control
logic. Enables tri-
state buffers, and
controls timing of
accesses.

IV. 2 DUAL-PORT RAM

granted to the external processor, the CPU continues execution from ROM or EPROM while the external CPU is reading or writing to the RAM. If the external processor is not in the middle of a transfer, the host CPU has immediate access when required. Otherwise, it is placed in a wait state. The same holds true for the external processor.

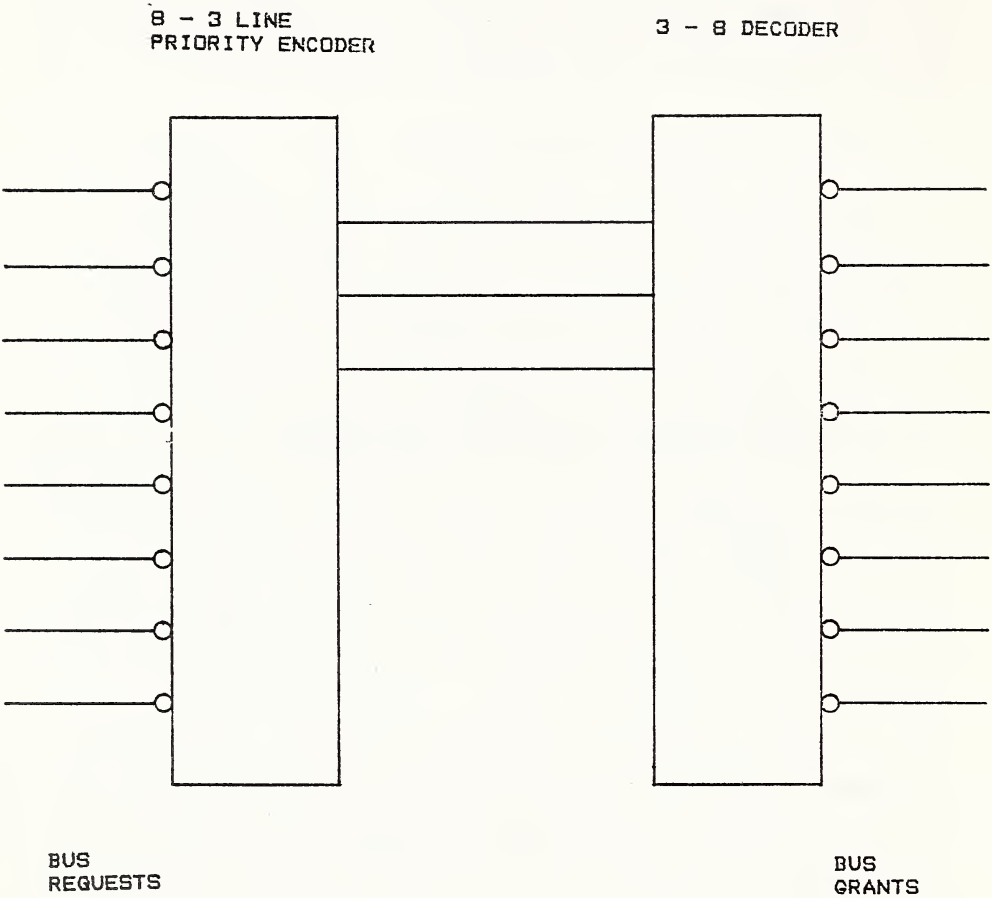
The turn around time for memory access on the 16-bit microcomputer board is 5 μsec. during block transfers. The total time that the dual-port RAM is actually being accessed during this transfer operation is about 1 μsec. The remaining 4 μsec are required for internal operations of the processor that are necessary for the execution of the transfer instruction. During this part (4μsec) of the transfer, the dual-port RAM can be accessed by an off-board processor. As a result, the worst case transfer time for an external processor during a block transfer is about 6 μsec. between each memory accesss. This corresponds to the case of having to wait the full time for the host CPU to complete its transfer before being granted the use of the RAM. This worst case response is an increase of 20% more time spent on the common bus. The host CPU will also experience some delay while waiting for the external processor to finish with a transfer.

The main disadvantages of the dual-port RAM are the possibility of the wait states slowing down the computations of the host CPU and increasing the time spent by the external processor on the common bus. Another disadvantage is the possibility that an external processor will write into an incorrect segment of memory thereby destroying programs or data. This may be partially eliminated by segmenting the memory into private and externally accessible sections.

IV.1.3. Parallel Bus Request Mechanism

In any multi-processor system, the method of accessing and sharing the common resources is of utmost importance. In addition to the actual procedure of acquiring the bus, there must be an order in which the processors are allowed access. Bus contention in this implementation is resolved using a priority scheme with parallel bus requests.

The parallel request mechanism is similar to a first-in, first-out (FIFO) buffer with some minor modifications. Each processor synchronizes its bus request to a common bus clock, and presents the request to an external priority bus resolution circuit. Such a circuit is shown in Figure IV.3. Thus, all bus requests are received in parallel. Whichever processor requests the use of the common resource first will receive it first. If two (or more) processors are requesting the resource at the same time, the highest priority processor will receive the resource first. The priority of a processor is determined by its slot position in the backplane, with the top slot having the highest priority. Processors


8 – 3 LINE PRIORITY ENCODER
3 – 8 DECODER
BUS REQUESTS
BUS GRANTS

IV. 3 PRIORITY CIRCUIT

can be "bumped off" the bus by a higher priority request or retain the resource for their entire transfer period. This is a function of design philosophy and will be discussed later.

A priority encoder receives the synchronized requests and encodes the address of the highest request. This address is then decoded and passed back to the processors, thereby granting the processor with the highest priority the bus. This scheme allows up to 16 bus requests on a common bus with all processors being looked at in parallel.

IV.1.4. Data Transfer

Transfers over a common bus can involve either a single byte (or word) or a block. During single-byte transfers, the processor allows a higher priority processor to gain access to the bus after completing one transfer on the bus. If there are no higher priority requests, the processor then transfers another byte of data. When the amount of data to be traded is small, this method works well.

For block transfers of data, a processor acquires the bus and asserts a lock mechanism that allows it to continue holding the bus until it finishes all of its interactions with the common resource. It then removes its lock and frees the bus to other requests. In a structured system, this locking mechanism helps to ensure the integrity of all

of the data transferred to or from that processor. The major disadvantage is the problem of a processor locking the bus and, because of some error condition, not releasing the bus again, thereby stopping all interprocessor communication. This disadvantage, however, may be eliminated by adding a hardware time-out mechanism that will free the bus if any processor holds it more than 6 msec. The timer will ensure that all other communication may continue even if an error occurs on one board. Figure IV.4 shows the bus activity for block transfers versus single transfers.

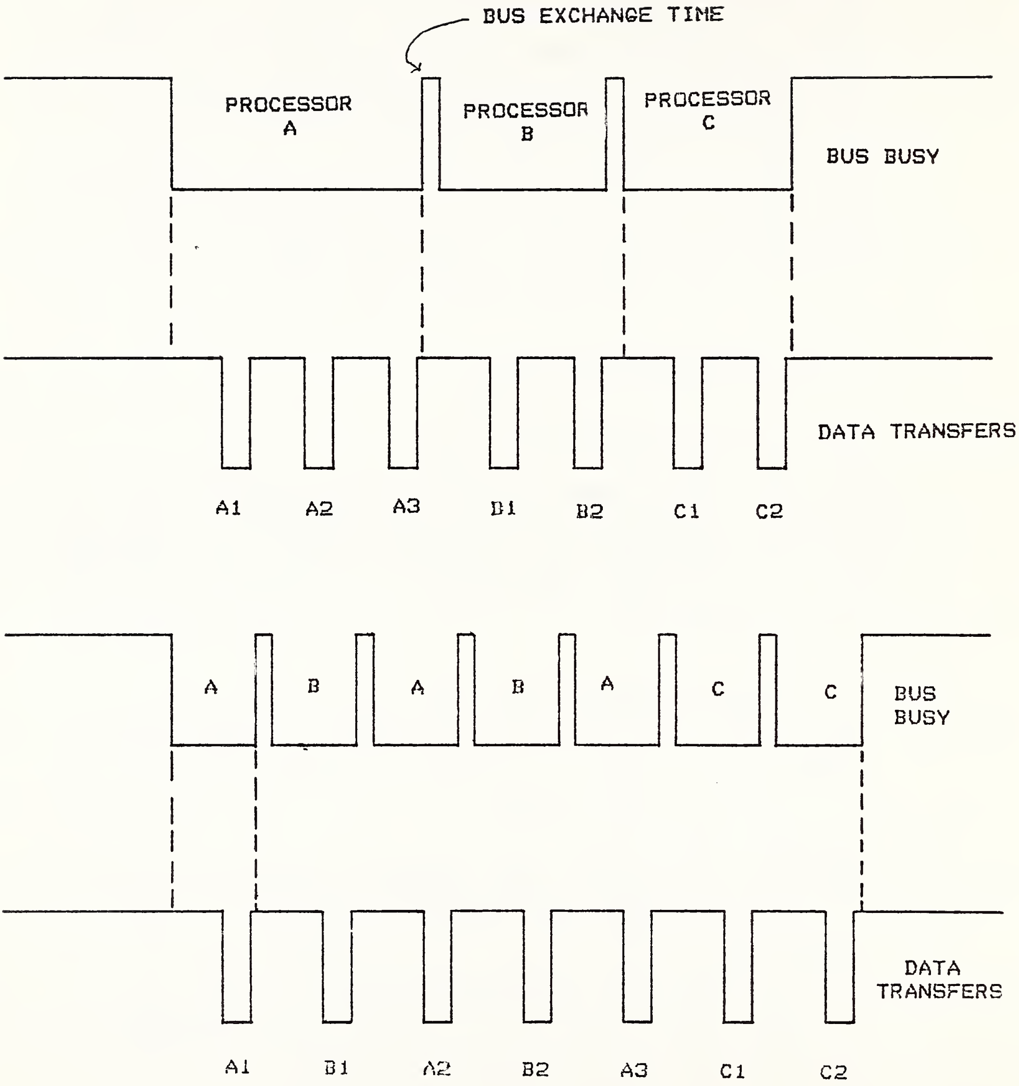
IV.2 THE COMMUNICATION PROTOCOL

Given the hierarchical architecture of the real-time control system and the hardware system described above, a software protocol was developed to regulate information transfer among the processors. This protocol was designed using the following guidelines:

1. All data transfers and interprocessor communication will occur through an interface of common variables in common memory buffer areas.

2. Only one process may update a given variable, although any number may read that variable.

3. No processor may interrupt the execution of another.


BUS EXCHANGE TIME
PROCESSOR A
PROCESSOR B
PROCESSOR C
BUS BUSY
DATA TRANSFERS
A1
A2
A3
B1
B2
C1
C2
A
B
A
B
A
C
C
BUS BUSY
DATA TRANSFERS
A1
B1
A2
B2
A3
C1
C2

IV. 4 BLOCK VS. SINGLE BUS TRANSFERS

4. The fundamental logical structure for each processor's function will resenble a simple finite state machine: input, compute, output.

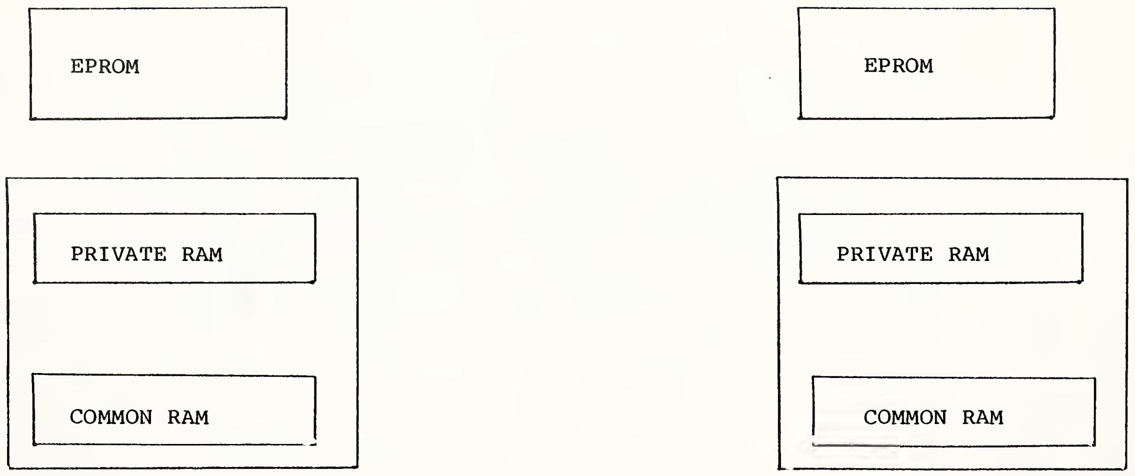
5. Processors will compute only when they receive new input data.

6. The system deals with all input data in a synchronous fashion, bringing in data at periodic intervals from a memory area that has buffered the asynchronous incoming data.

The implementation of the protocol is based upon the ability to provide a simple common memory accessible to all the processors in the system. This common memory and a synchronization mechanism to govern the communications protocol to this common memory comprise the framework for the control of the whole multiprocessor system.

IV.2.1. The Common Memory

The dual-port RAM allows one or more external processors to access memory space located on any given processor board. Allowing some segments of each processor's memory space to be accessed by other processors creates a common memory. Figure IV.5 shows the memory layout of several processors in the system. On each board, part of the memory is available only to the processor on that board; the rest may be accessed by the other processors in the system. Therefore, a


EPROM
PRIVATE RAM
COMMON RAM
EPROM
PRIVATE RAM
COMMON RAM

EACH MICROPROCESSOR BOARD HAS EPROM SPACE AND RAM SPACE. THE RAM SPACE IS SEGMENTED INTO PRIVATE MEMORY, ACCESSIBLE ONLY TO THE ONBOARD PROCESSOR, AND COMMON MEMORY, ACCESSIBLE TO ALL PROCESSORS IN THE SYSTEM.

FIGURE IV.5

processor has access to its private memory, its own common memory area and the common memory areas of the other processor boards in the system.

All interprocessor communication is done by passing data through the common memory. Common memory buffers define the interfaces between the processors. Each processor is assigned specific buffers in common memory for its input and output data. If new processors are added to the system, they too are assigned buffers. Any processors that will use their data are programmed to address those data buffers.

Any task execution status information for a processor that may be required by another processor is written as data to a buffer assigned as an interface between those processors. Likewise, processing results to be used by another processor are transferred through the buffers in common memory. Since this common memory is the communication link, the rigid timing constraints of both processors exchanging data in a "handshake" procedure are not required. One processor can deposit its data in the common memory area totally independent of when the other processor will read it.

Data intended for use by another processor is put into a buffer in the common memory area that resides on the processor that generated that data. When the external processor is ready for new data, it requests the system bus, moves a copy of the data to its own private memory and processes it.

(It should be noted that the transfer of data from the onboard private RAM to the onboard allocated common memory area of a processor does not require a bus access. Only when the data is transferred between different processor boards is there any bus activity.)

The common memory construct provides the framework for all intercomputer communication in the real-time control system when used with the protocol described below.

IV.2.2. Intercomputer Communication Protocol

All communication among the processors occurs within the common memory buffers. Each computer is assigned one or more buffers in the common memory for its output data. Any number of processors may read a buffer, but to ensure that no two processes will overwrite the same data, only one process may write to a specific buffer. Output buffers are onboard for the processors that generate the data but require a bus access when read by other processors.

The software that monitors the buffers in the system and controls all data transfers is referred to as the protocol. This protocol is responsible for checking the status of input data buffers to see if the data has been updated, reading in the new data, initiating execution of the processes resident on that processor, sending output results from private memory to common memory at termination of computations, and updating the status for those buffers on each processor that will use the data. A copy of the protocol is resident on each processor board. Figure IV.6 shows the program logic for the protocol.

During protocol initialization, at system startup or by operator request, the processes to be executed on each computer are loaded into RAM along with a buffer allocation table that specifies where input buffers will be found and where output data buffers are to be written. This buffer allocation table is defined by the system programmer. The programs and this table identify the onboard processing that will be monitored by the protocol for each processor. Section IV.2.3 will further explain the buffer allocation table.

This protocol for accessing data in common memory buffers and executing processes executes optimally once each time interval. An external bus signal indicates the start of a time interval to each of the protocols in the system. During each interval, the protocols in the system check their common memory status buffer to see if the input data have been updated since last computations. If all data is ready, each protocol will transfer input data from common memory buffers to private memory buffers. Once all required data has been input, computations are executed. The results of those computations are written to onboard common memory, and

```
IF   ALL DATA HAS BEEN UPDATED    THEN
     TRANSFER INPUT DATA TO PRIVATE RAM
     EXECUTE PROCESSES
     TRANSFER RESULTS TO COMMON MEMORY BUFFERS
     UPDATE STATUS ON OTHER PROCESSORS
ENDIF
```

A PROTOCOL PROGRAM ON EACH PROCESSOR MONITORS DATA TRANSFERS BETWEEN THE COMMON MEMORY BUFFERS IN THE SYSTEM AND EXECUTION OF PROCESSES.

PROTOCOL LOGIC

FIGURE IV.6

status is updated on other processors. The protocols then execute a wait instruction which suspends processing until the start of the next time interval. Whether that signal comes immediately, or whether some additional time elapses is transparent to each protocol. Time intervals are initiated by the signal and are independent of any external intervention. In this manner, computations are only done on complete and current data buffers and the logic on each processor is restricted to the sequence of: check status,input, process, output results.

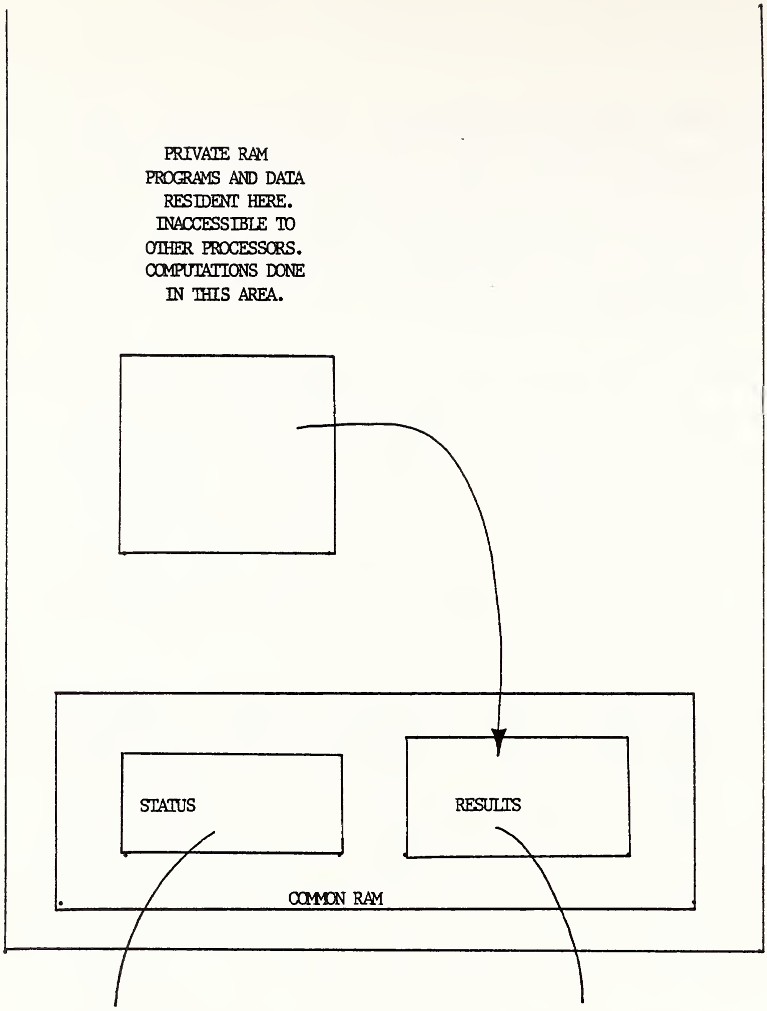
On each processor as described above, there is private memory space where the programs and data reside. There is also common memory space that contains two types of data buffers. One set of common memory buffers is the buffers used to store output results and other data that will be used by the other external processors. When the computations are finished on input data, the protocol transfers the results from the private memory buffers to the common memory buffers where they will be read by the processors that use that data. The other set of buffers within the common memory space contain status flags for the common memory buffers on other processors where this protocol will get its input data. The status flag identifies when the associated data buffer was last updated. When a protocol is ready to restart computations, it refers to the status buffer in its common memory area to see if the status has changed since

the last time the status was accessed. If the status indicates that the data buffers have not been updated since the last time this protocol initiated computations, it will not read its data or start computations. Thus, processing is only done on new input data.

After a protocol has transferred the computation results to its common memory buffers, it updates the status word associated with each output buffer on each processor that will use that data. In this manner, those processors will be informed that they have new data available and may proceed with their computations. Figure IV.7 shows the layout of the buffers on a processor board.

As an example, consider a three processor system. Protocol P1 reads buffers 2A and 3B. P1 generates the buffers 1A, 1B and 1C. Protocol P2 reads buffers 1A and 1B and generates buffer 2A. Protocol P3 reads buffers 1C and 2A and generates buffers 3A and 3B. Figure IV.8a is a table showing the buffers and the associated protocols.

In the onboard common memory area for P1, there will be three data buffers 1A, 1B and 1C and an additional buffer that will contain the status words for the input buffers, 2Astat and 3Bstat. Likewise P2 has the data buffer 2A in its onboard common memory and the buffer containing 1Astat and 1Bstat. P3 has output buffers 3A and 3B and the status words 1Cstat and 2Astat. See figure IV.8b.


PRIVATE RAM
PROGRAMS AND DATA
RESIDENT HERE.
INACCESSIBLE TO
OTHER PROCESSORS.
COMPUTATIONS DONE
IN THIS AREA.
STATUS
RESULTS
COMMON RAM

RESULTS MOVED FROM
PRIVATE RAM TO COMMON
RAM TO BE ACCESSED BY
PROCESSORS WHO WILL
USE THEM

STATUS OF INPUTS TO
BE READ BY THIS
PROCESSOR. WRITTEN
INTO BY PROTOCOL OF
PROCESSOR WHO GENERATES
THE DATA.

COMMON MEMORY. ACCESSIBLE
TO THIS PROCESSOR AND OTHER
PROCESSORS THROUGH THE BUS.

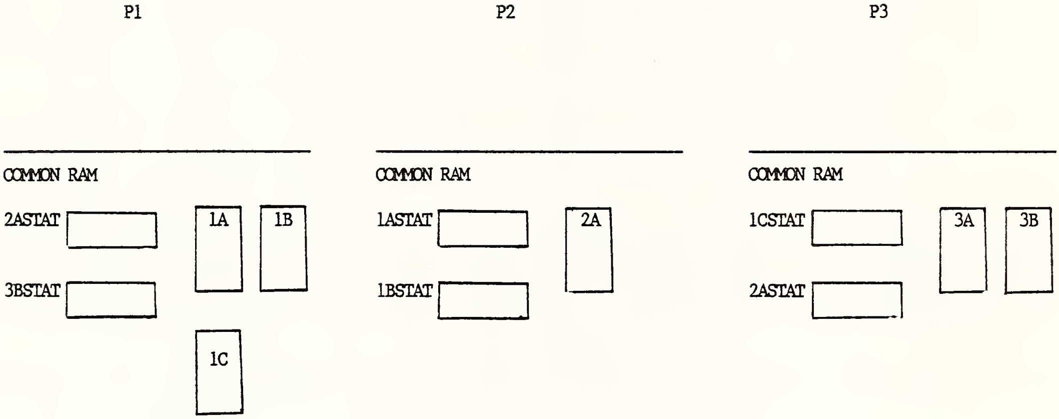
ON-BOARD RAM LAYOUT

FIGURE IV.7

PROCESSOR	INPUT BUFFERS	OUTPUT BUFFERS
P1	2A 3B	1A 1B 1C
P2	1A 1B	2A
P3	1C 2A	3A 3B

INPUT AND OUTPUT BUFFER ASSIGNMENTS FOR A 3 PROCESSOR SYSTEM

FIGURE IV.8a

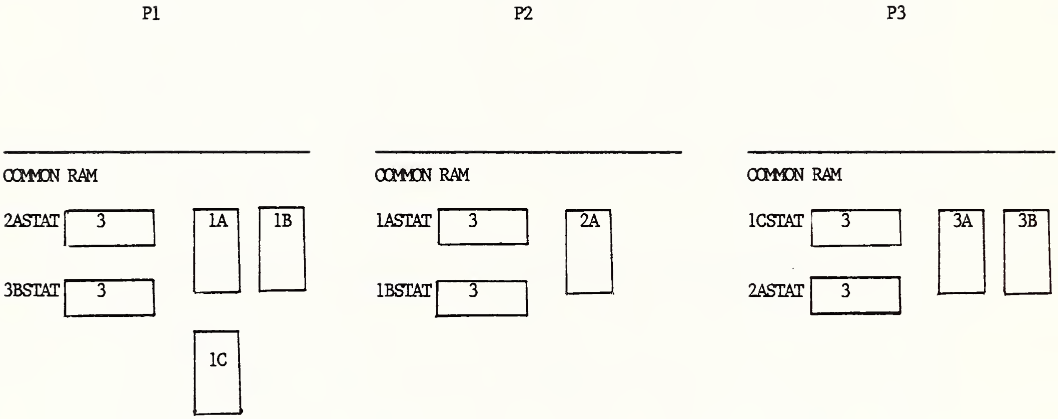

P1
P2
P3
COMMON RAM
2ASTAT
3BSTAT
1A
1B
1C
COMMON RAM
1ASTAT
1BSTAT
2A
COMMON RAM
1CSTAT
2ASTAT
3A
3B

COMMON MEMORY BUFFER AREAS ON EACH OF THE THREE PROCESSORS. EACH CONTAINS BUFFERS FOR OUTPUTS AND A STATUS WORD ASSOCIATED WITH EACH BUFFER THAT IS USED AS INPUT FOR THAT PROCESSOR. NOTE: BUFFER 2A IS READ BY P1 AND P3. THEREFORE, THERE IS A 2ASTAT ON BOTH PROCESSORS.

FIGURE IV.8b

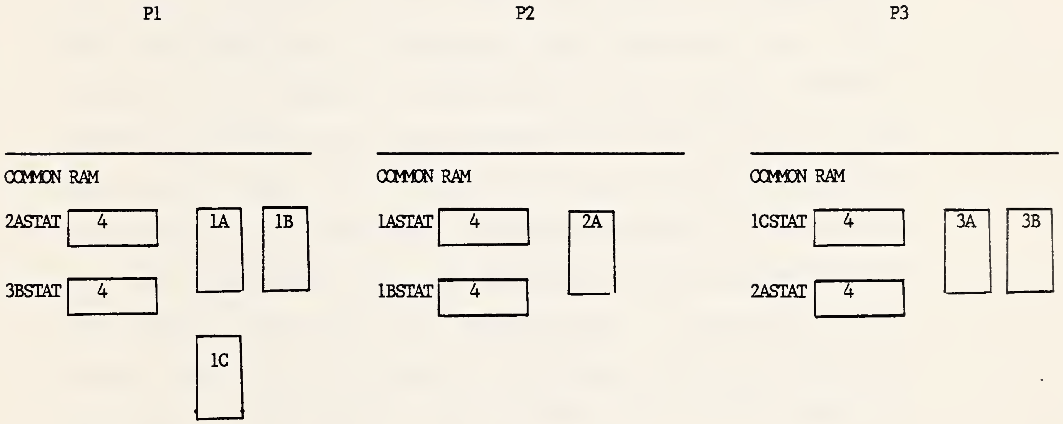
After P1 updates its three onboard output buffers in its common memory area, it also updates the status words associated with each of its output buffers in the memory areas of all the processors that will use those buffers as input. That is, P1 will update 1Astat and 1Bstat in P2's common area, and 1Bstat and 1Cstat in P3's common area. Similarly, P2 will move its results to its output buffer and update 2Astat on P1 and P3, and processor P3 will generate its outputs and update 3Bstat on P1. Figure IV.8c shows the common buffers for the protocols P1, P2 and P3 at cycle 3. Figure IV.8d shows the contents of the protocols' common buffer areas and status words after new outputs have been generated during cycle 4.

Presently, the status value associated with a current buffer is the cycle count of the system. A tally of the number of elapsed time intervals is maintained and is written to a common memory buffer along with other system execution parameters (see section IV.2.3). Each protocol reads in the count immediately at the start of a time interval and then writes it in each status word associated with that output buffer on external processors. The status of a buffer is compared with the last status for that buffer. If the values are different, indicating new data, the new status is also checked with the current cycle count to see if the data is recent or if some error has occurred. If the status of the buffer is the same as the last status, it is compared


P1
P2
P3
COMMON RAM
2ASTAT
3
3BSTAT
3
1A
1B
1C
COMMON RAM
1ASTAT
3
1BSTAT
3
2A
COMMON RAM
1CSTAT
3
2ASTAT
3
3A
3B

AT TIME INTERVAL 3, THE STATUS FOR BUFFERS UPDATED IS 3.
THIS INDICATES THAT ALL BUFFERS ARE CURRENT.

FIGURE IV.8c


P1
P2
P3
COMMON RAM
2ASTAT
4
1A
1B
3BSTAT
4
1C
COMMON RAM
1ASTAT
4
2A
1BSTAT
4
COMMON RAM
1CSTAT
4
3A
3B
2ASTAT
4

AT TIME INTERVAL 4, THE STATUS FOR BUFFERS UPDATED IS 4.
THIS STATUS IS DIFFERENT THAN THE LAST INTERVAL, THEREFORE
ALL DATA IS NEW, AND COMPUTATIONS WILL PROCEED.

FIGURE IV.8d

with the cycle count to see how many cycles have elapsed since the buffer was updated. If a predetermined minimum of elapsed cycles is exceeded, the processor or process that supplies this data is probably malfunctioning, and the operator will be informed of this occurrence. Figure IV.9 shows the status-checking algorithm.

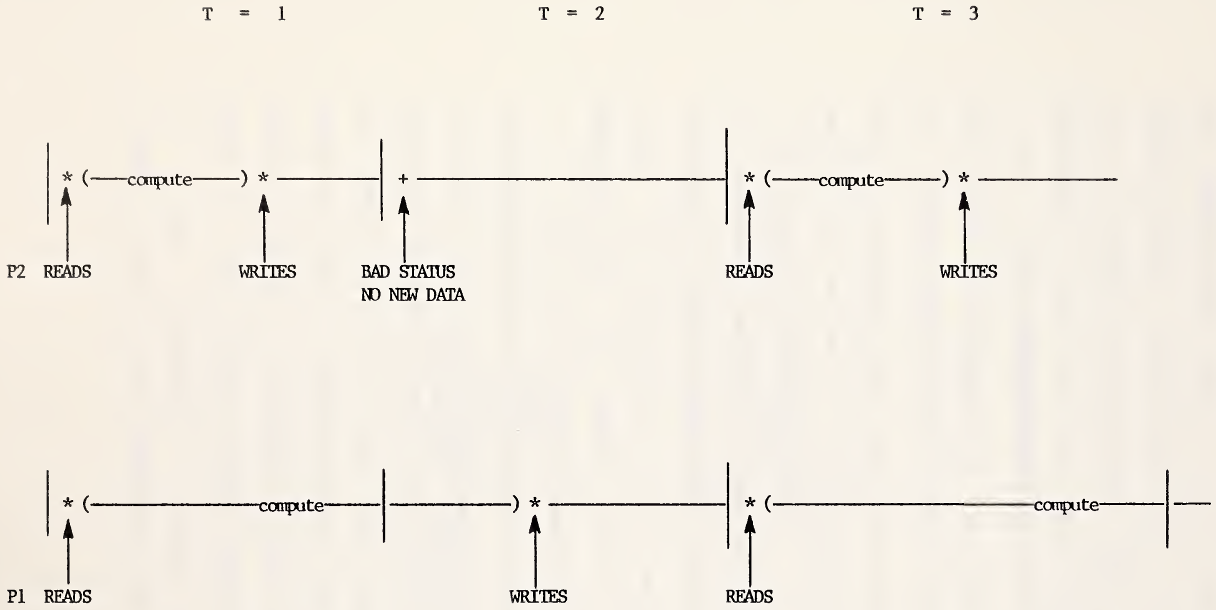
If computations are not completed within a time interval, execution continues until the computation is completed. When task execution is finished, the protocol will then transfer its results to the onboard common memory buffers, update the status words associated with output buffers, and wait until the next interval to restart processing again. Any computations on another processor dependent on this data will be suspended by that protocol if the status has not been updated since the last computations. Processing will resume when this new data is supplied. Each processor in the system is designed to function when there is new data to be processed, not according to some time constraint. The time interval merely synchronizes the data transfer process and fosters data integrity by regulating read access to common memory through the bus.

Figure IV.10a shows the timing cycle and data accesses for a system of two processors, one with an execution cycle time greater than the timing interval of 28 milliseconds. Processing on P1 takes 40 msec to execute, and on P2 10 msec.

```
IF    STATUS NOT EQUAL TO LAST STATUS  THEN
          IF  | STATUS - CURRENT COUNT |  <=  MAXIMUM  THEN
                  READ DATA
          ELSE
                  POSSIBLE BAD DATA
          ENDIF
ELSE
          IF | CURRENT COUNT  - STATUS |   >  MAXIMUM   THEN
                  POSSIBLE PROCESSOR MALFUNCTION
          ENDIF
ENDIF
LAST STATUS = STATUS
```

ALGORITHM FOR CHECKING STATUS OF AN INPUT BUFFER

FIGURE IV.9


T = 1
T = 2
T = 3
* (——compute——) *
+
* (——compute——) *
P2 READS
WRITES
BAD STATUS
NO NEW DATA
READS
WRITES
* (——compute——) *
* (——compute——
P1 READS
WRITES
READS

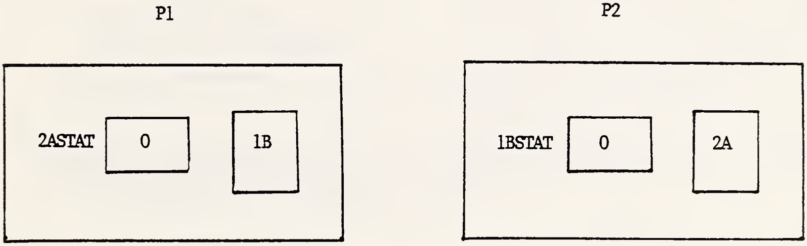
THIS DIAGRAMS THE ACTIVITY ON THE BUS FOR THE EXAMPLE.

AT T = 1 P2 COMPUTES AND WRITES DATA. P1 STARTS EXECUTION.
AT T = 2 P2 CAN'T COMPUTE BECAUSE THERE IS NO NEW DATA.
P1 FINISHES COMPUTATIONS AND WRITES RESULTS OUT.
AT T = 3 P2 RESTARTS
P1 RESTARTS

FIGURE IV.10a

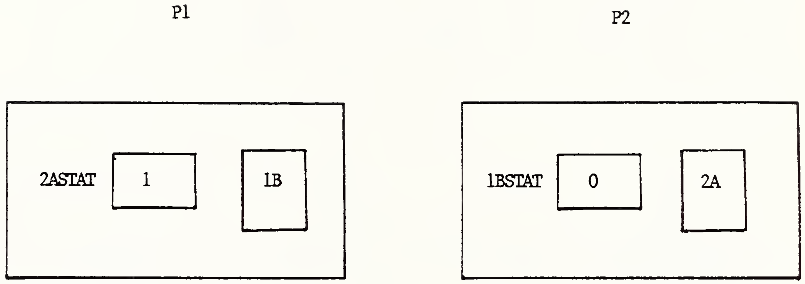
P1 reads buffer 2A and generates buffer 1B as shown in Figure IV.10b. Processor P2 reads buffer 1B and generates buffer 2A. The status words associated with all input buffers will be 0. At system startup, the cycle count of the system will be 1. Both protocols read in the cycle count and check the status of the input buffers to be read. Each finds new data and starts its computations. The protocol on P2 will write out the data and will be ready to restart at the second signal. The status of output buffer 2A will be 1. P1 has not finished processing and therefore continues execution into the second interval as shown in Figure IV.10c. At the start of the second interval, the cycle count is 2. The protocol on P2 reads the new cycle count and checks the status for its input data. Since the status is the same as it was last cycle, P2 does no inputs, no computation, and writes out no data. P1 finishes processing during the second cycle. P1 then transfers its results to the onboard common memory buffer and updates the status for this buffer on processor P2. This new status is 1. (Since P1 was computing when the trigger for the second cycle was generated, it did not read in the current cycle count which is 2. It uses the cycle count that is available as shown in Figure IV.10d. P1 now waits until pulse to get back into the computation cycle. At the third interval, P2 reads a different status than the last cycle, finds that the data is current, reads the data in and processes it. P1 also reads

PROCESSOR	INPUT BUFFER	OUTPUT BUFFER	EXECUTION TIME
P1	2A	1B	40 MSEC
P2	1B	2A	10 MSEC


P1
P2
2ASTAT
0
1B
1BSTAT
0
2A

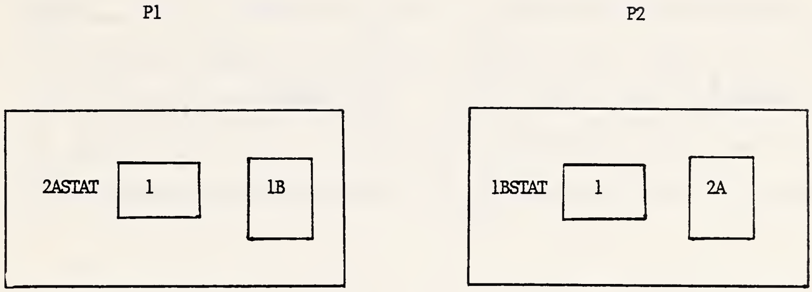
2 PROCESSOR SYSTEM AT TIME = 0. STATUS BUFFERS CONTAIN 0
CYCLE = 0.

FIGURE IV.10b


P1
P2
2ASTAT
1
1B
1BSTAT
0
2A

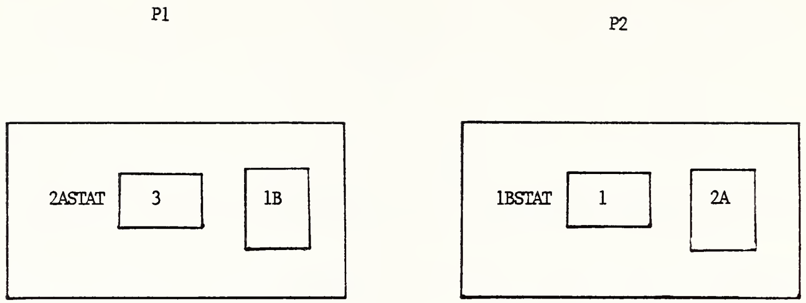
AT THE END OF TIME INTERVAL 1, P2 HAS UPDATED BUFFER 2A AND 2ASTAT.
P1 STILL PROCESSING. CYCLE = 1.

FIGURE IV.10c


P1
P2
2ASTAT
1
1B
1BSTAT
1
2A

AT THE END OF TIME INTERVAL 2 , P1 HAS UPDATED BUFFER 1B AND 1BSTAT. P2 DID NO PROCESSING THIS TIME INTERVAL. CYCLE = 2.

FIGURE IV.10d


P1
P2
2ASTAT
3
1B
1BSTAT
1
2A

AT THE END OF TIME INTERVAL 3, P2 READ NEW DATA AND UPDATED 2A AND 2ASTAT AGAIN.
P1 STARTED SECOND PROCESSING. CYCLE = 3.

FIGURE IV.10e

in newly updated data and so execution continues as before as shown in Figure IV.10e.

Issuing the status word of the buffer at the end of the transfer of the output data to common memory will eliminate the necessity of reading the buffer before all of its data is updated. If a buffer is only partially updated at the time of the pulse, the protocol that reads that data will not find an updated status word. A new status word implies that the whole buffer has been updated, thus ensuring that partially updated data will never be processed.

IV.2.3. External Communications

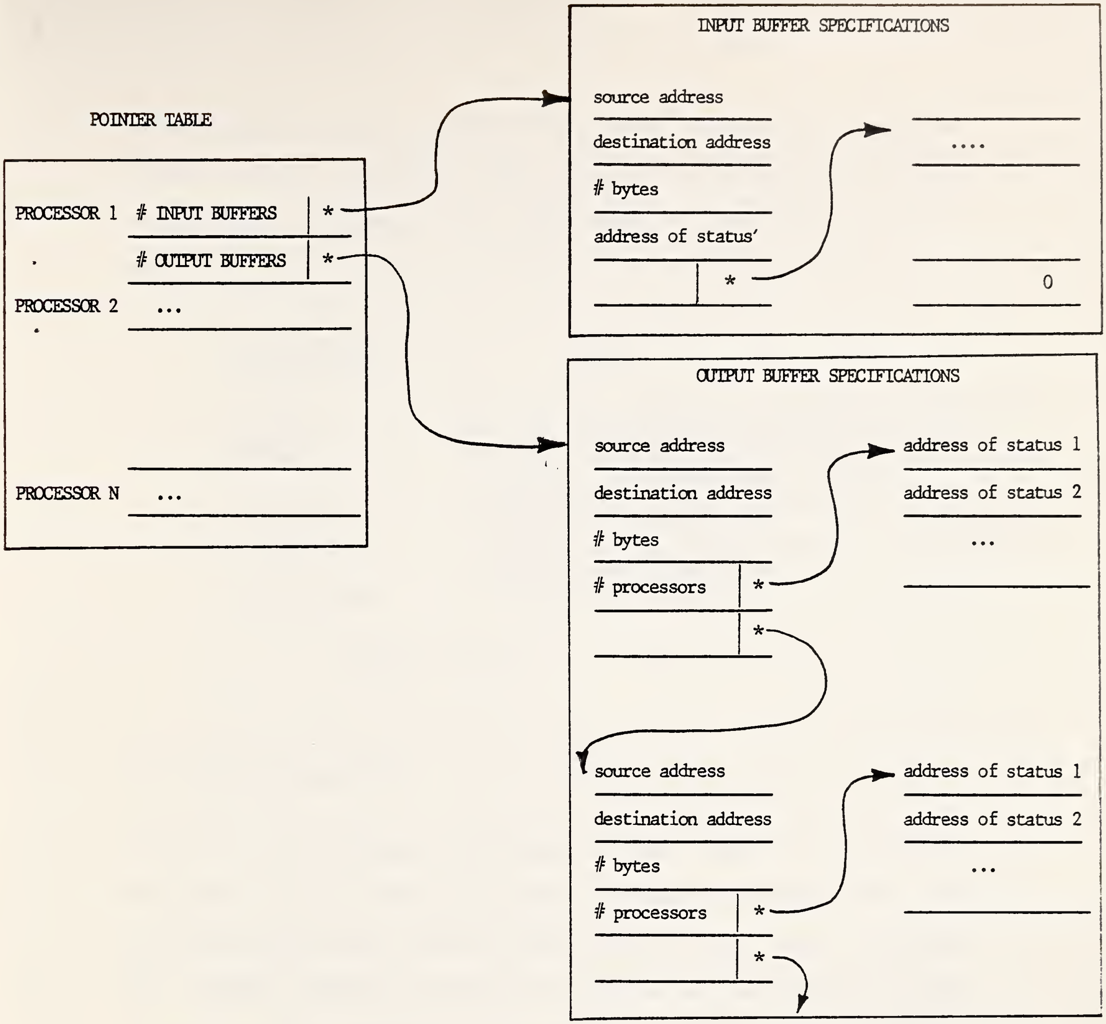
Most of the processors execute the algorithms that comprise the robot control system. One processor, in addition, provides a user interface for system control purposes and supplies the synchronization pulse mentioned above. This section will describe the functions of this processor.

The communication processor maintains a clock for determining the timing interval of the system. Presently, the clock on this processor is set for an interval of 28 milliseconds. The clock interrupts the processor at the end of each period. The interrupt handler routine resets the timer for another interval and then sends an external signal to each of the other processors in the system. This external signal is tied to a pin in each processor. When a protocol is finished with all its functions, it executes a software wait instruction. This suspends all processing on that processor until the signal from the communication processor is detected. This is the signal that will start the next execution interval.

In addition to synchronizing the system, the communication processor provides an interface that enables the user to define the parameters of execution. At system startup, this processor places in common memory the buffer allocation tables that specify the source, destination and number of bytes for each common memory transfer. They indicate for each protocol which buffers are to be accessed for input data and which will be written with the computation results. In addition each table gives the address of the status areas for any outputs. Figure IV.11 shows the data structure for the buffer allocation tables.

Other execution parameters include a set of flags to set the mode of execution, including the current cycle count. The protocol on each processor reads in these flags along with the processor data after each synchronization pulse. Figure IV.12 gives a list of the current flags available, and the function of each.

In addition to the flag data, the user may also modify the size of the time interval. If monitoring system performance


POINTER TABLE
PROCESSOR 1
# INPUT BUFFERS
*
# OUTPUT BUFFERS
*
PROCESSOR 2
...
PROCESSOR N
...
INPUT BUFFER SPECIFICATIONS
source address
destination address
# bytes
address of status'
*
....
0
OUTPUT BUFFER SPECIFICATIONS
source address
destination address
# bytes
# processors
*
*
address of status 1
address of status 2
...
source address
destination address
# bytes
# processors
*
*
address of status 1
address of status 2
...

DATA STRUCTURE FOR THE BUFFER ALLOCATION TABLES

FIGURE IV.11

shows that too many processes are not completing computations within a interval, the user may enlarge the interval rather than reallocate the algorithms to different processors.

FLAG/VALUE	USE
read	set if that processor should read its data
verify	set if input data should be verified
debug	set if private variables should be output along with computation results
reinit	reinitialize local variables and start at beginning of program
restart	reinitialize protocol and all programs and start execution anew
cycle count	current cycle number

Figure IV.12

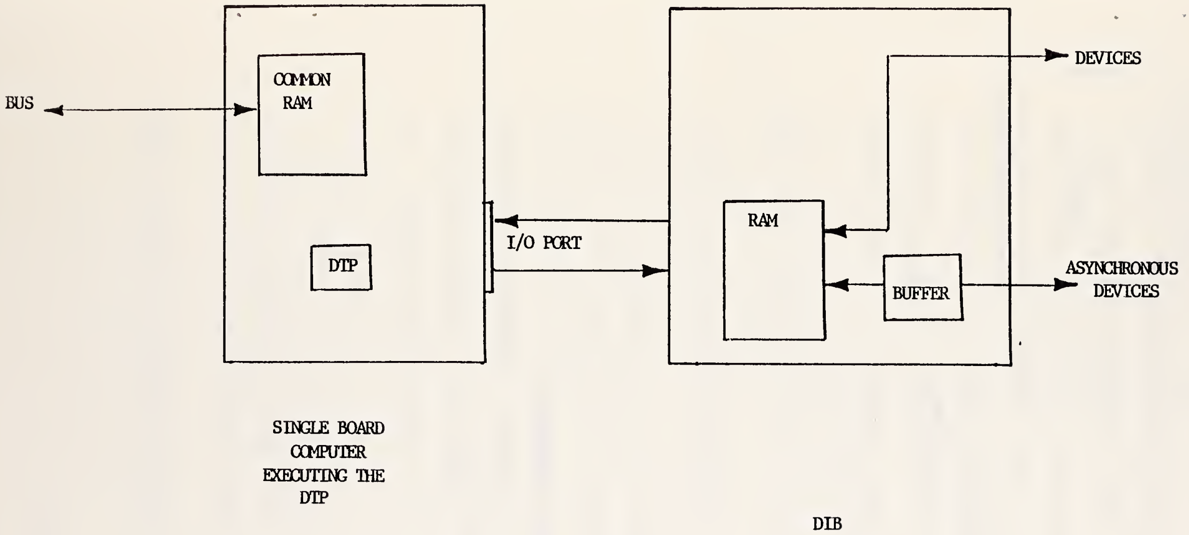
IV.3 DATA COLLECTION

Currently, the system interfaces to two types of external devices. First there are simple sensors, such as touch sensors and the robot itself, that return data upon request. The simple sensors are now interfaced to the system through a dedicated microcomputer that polls each in turn, scales that data if necessary, and passes it to the control system through common memory buffers. Secondly, there are complex sensing devices such as the vision system or other computer systems. These may require data or make requests at any time and may involve a handshake procedure to exchange data. These complex devices are generally asynchronous in nature.

The vision system has its own dedicated computer on the common bus of the control system that is programmed to input the vision data, process it, and pass it to the control system with the synchronization supplied by the protocol. As more processing for complex vision is required, additional processors will create the need for a separate vision system. This vision system will reside on a separate bus system and will communicate with the control system over a high-speed data link. In order to provide an interface into the control system for this type of vision module and other complex external devices, as well as the simple devices mentioned above, a specialized data collection interface is being implemented.

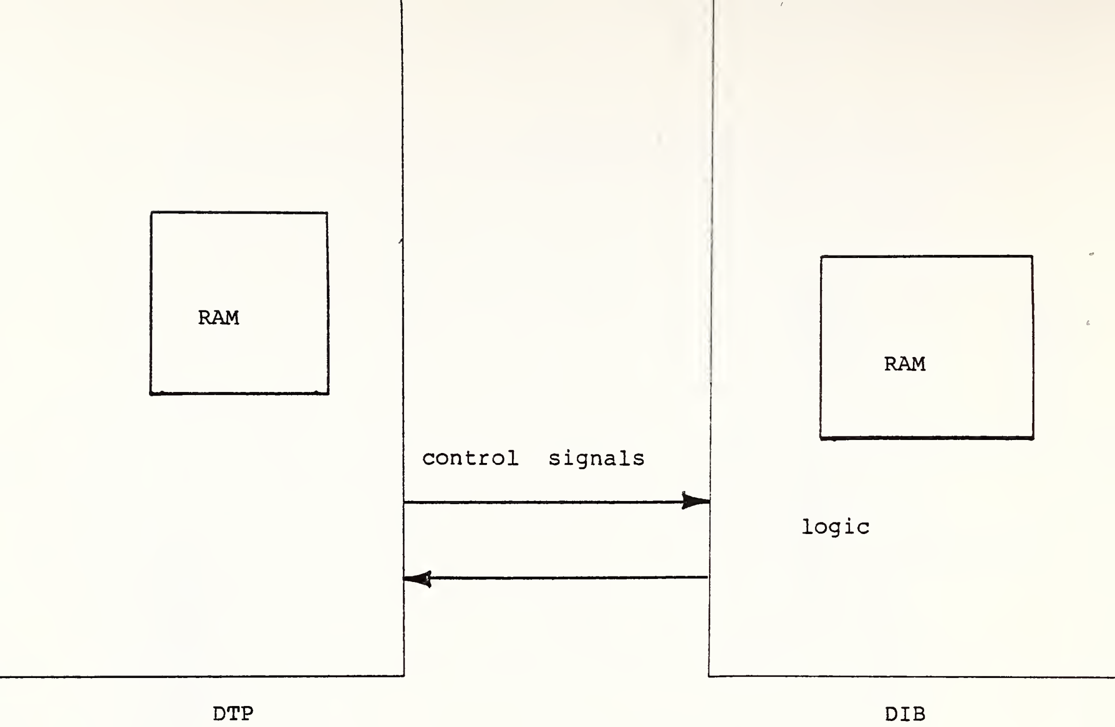
IV.3.1. Data Collection Interface Design

Data collection is accomplished with a two component interface. A single board microcomputer containing a data transfer program (DTP) is coupled with a special external device interface (DIB) board designed at NBS. (Figure IV.13) Using the protocol described above, data from the control system that is intended for external devices is input to the DTP. The DTP further processes this data if required and transfers it to the buffer area onboard the DIB through the I/O port. (Figure IV.14) Then the DTP transfers from the DIB any data collected from the devices. This device data is scaled by the DTP and finally tranferred to on-


BUS
COMMON RAM
DTP
I/O PORT
RAM
BUFFER
DEVICES
ASYNCHRONOUS DEVICES
SINGLE BOARD COMPUTER EXECUTING THE DTP
DIB

DATA COLLECTION INTERFACE. A SINGLE BOARD COMPUTER EXECUTING A DATA TRANSFER PROGRAM IS INTERFACED TO A DATA INTERFACE BOARD THROUGH AN I/O PORT. THE DIB IS THE INTERFACE TO THE EXTERNAL DEVICES.

FIGURE IV.13


RAM
RAM
control signals
logic
DTP
DIB

DTP TO DIB INTERFACE

FIGURE IV.14

board common memory buffers, ready for use by the control system.

To the DTP, the DIB appears to be a buffer area. The DTP supplies an address and then sequentially transfers data to or from the DIB buffers. When the buffers are not being accessed by the DTP, circuitry on the DIB transfers data between the DIB buffers and the external devices. For simple devices, a polling exchange suffices for the data transfer. For asynchronous devices, more elaborate handshaking is involved and double buffering is employed. Thus, whenever the DIB buffers are not being accessed by the DTP, they are being used to collect data from the devices.

IV.3.2. DTP to DIB Protocol

The DTP is interfaced to the DIB through 3 8-bit parallel ports. The first is for input data, the second for output data, and the third for control communication signals. Presently, several control flags are used. The Addr flag specifies whether the output to the DIB is an address or data. The IBF (input buffer full) flag is set by the DIB when it has completed its cycle of transfering data between the buffers and the devices. The Transfer flag switches the DIB from data collection mode when it is accessing the devices, to transfer mode when the DTP is accessing the DIB buffers.

When the DTP is ready to access the DIB buffers, it polls the IBF to make sure the DIB has completed all interactions with the external devices. When the IBF is set, the DTP sets the Transfer flag indicating that it will have control over the DIB buffers. Since the DIB is just a slave device to the DTP, all elements of the data transfer to and from the DIB buffers is controlled by the DTP. The data transfer is initiated when the DTP sets the Addr flag to identify that the next data sent to the DIB will be an address. The address of the start location in the DIB buffer space for the next data transfer is then sent. This address may be anywhere in the buffer area on an 8-byte boundary. The DTP resets the Addr flag and then accesses the data sequentially from that address until done. The DTP continues this sequence of sending an address and accessing the data from sequential address locations until it has completed all necessary data transfers between the DTP and the DIB buffer space. It then resets the Transfer flag to free the DIB buffers. After the DTP finishes the computations on the data it collected from the DIB, it sends the data to the control system and restarts the cycle.

Figure IV.15 shows the logic flow of the DTP-DIB interactions. Figure IV.16 shows the timing and flag status as the DTP transfers data to and from the DIB buffers.

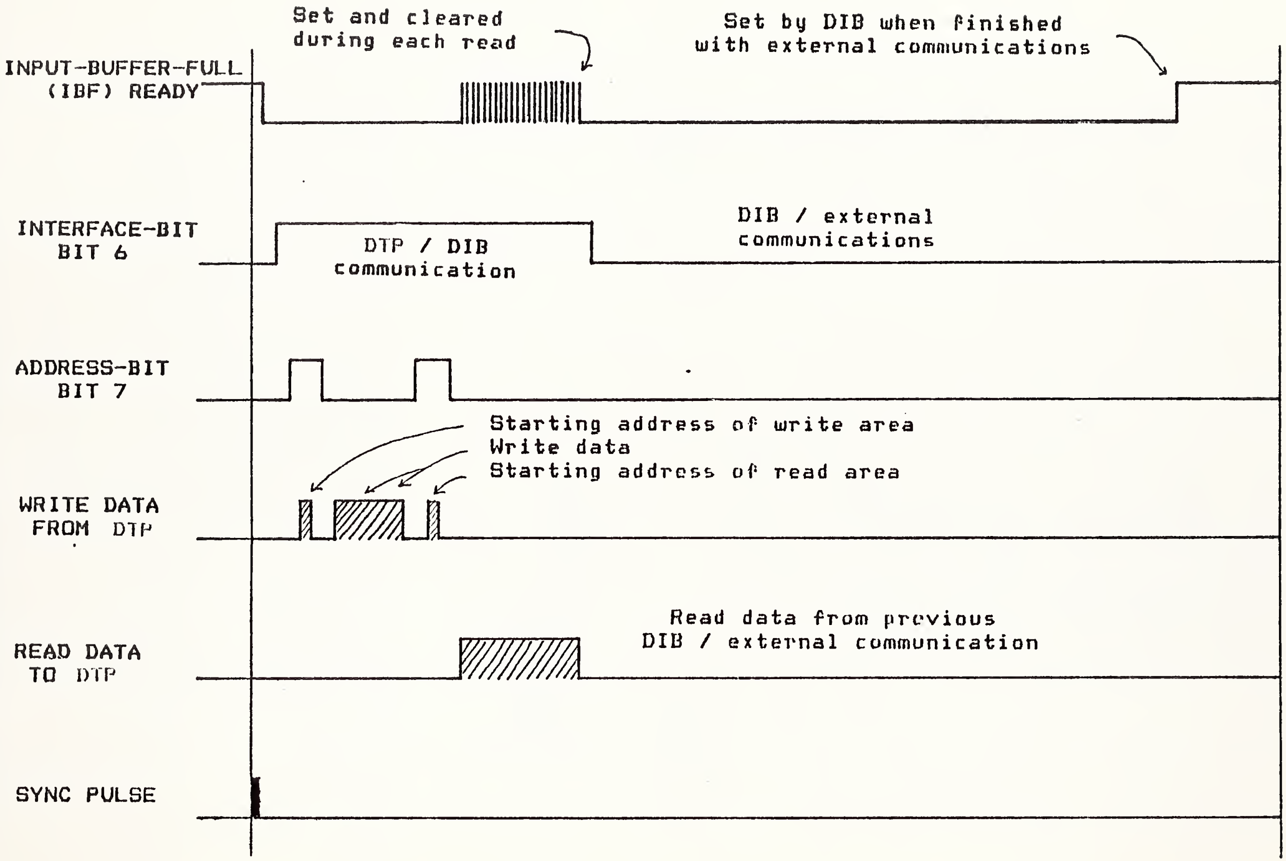
IV.3.3. DIB to Device Interfaces

When the DTP resets the Transfer flag indicating that it

```
   IF    IBF UP   THEN
         Set Transfer
         Set Addr
         Send Address
         Reset Addr
         Send data
         ...
         Set Addr
         Send Address
         Reset Addr
         Receive Data
         ....
         Reset Transfer
ENDIF
```

DTP - DIB HANDSHAKE

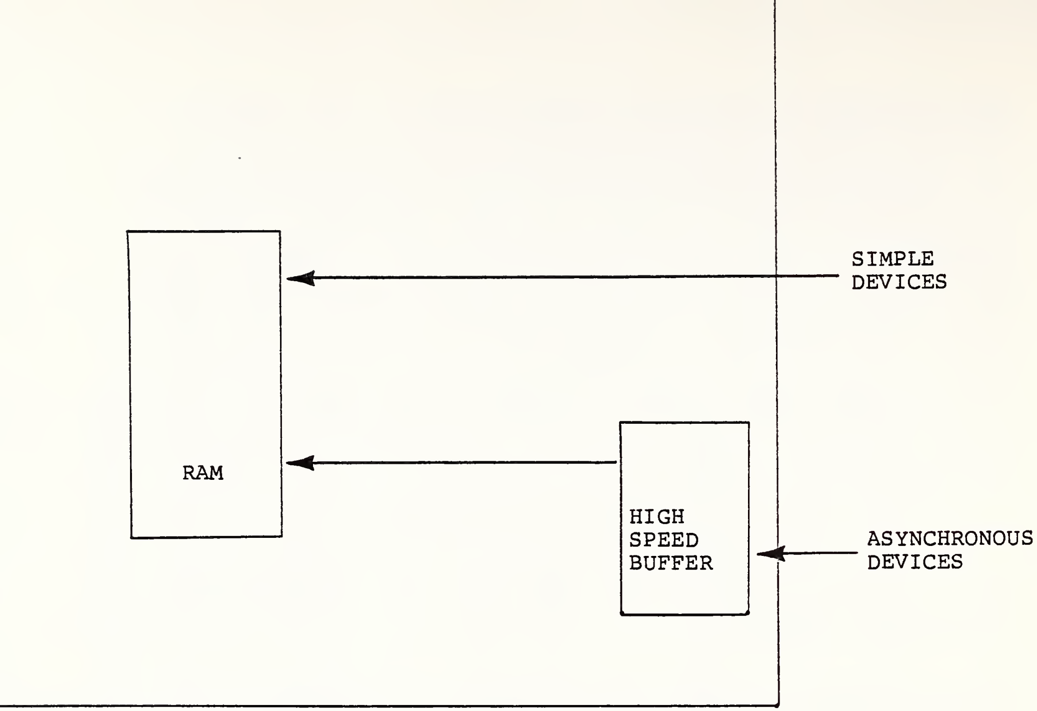
FIGURE IV.15


Set and cleared
during each read
Set by DIB when finished
with external communications
INPUT-BUFFER-FULL
(IBF) READY
INTERFACE-BIT
BIT 6
DTP / DIB
communication
DIB / external
communications
ADDRESS-BIT
BIT 7
Starting address of write area
Write data
Starting address of read area
WRITE DATA
FROM DTP
Read data from previous
DIB / external communication
READ DATA
TO DTP
SYNC PULSE

IV. 16 TIMING DIAGRAM

will no longer be accessing the DIB buffers, the circuitry on the DIB begins communication with all the external devices. The DIB, under automatic operation, performs a global update of all simple sensors connected to it. A simple sensor, in this context, is a device or sensor which only requires or provides data when accessed. Its data is available within a short time period (100 µsec. or less), and is acquired through a simple polling interface. The data is transferred directly between the device and the DIB buffers. Since the DIB polls each sensor or device in turn until all are read in or written to, they can be classified as synchronous in nature.

Complex sensors and external processors are asynchronous in nature. The DIB does not know when the data will be available. One method being implemented at NBS to handle asynchronous transfers is to use additional buffering. As shown in Figure IV.17, each external device (computer or complex sensor) has some form of high-speed data buffer associated with it for its input and output data. Additional circuitry enables these high speed buffers to be constantly ready to transfer data to or from the complex devices performing the required handshaking. Data intended for these devices is transferred from the DIB buffer area to the high speed buffers. Data that will be transferred to the DTP is transferred from the high speed buffers when the DIB is doing data collection. In this manner, the DTP will be able


SIMPLE DEVICES
RAM
HIGH SPEED BUFFER
ASYNCHRONOUS DEVICES

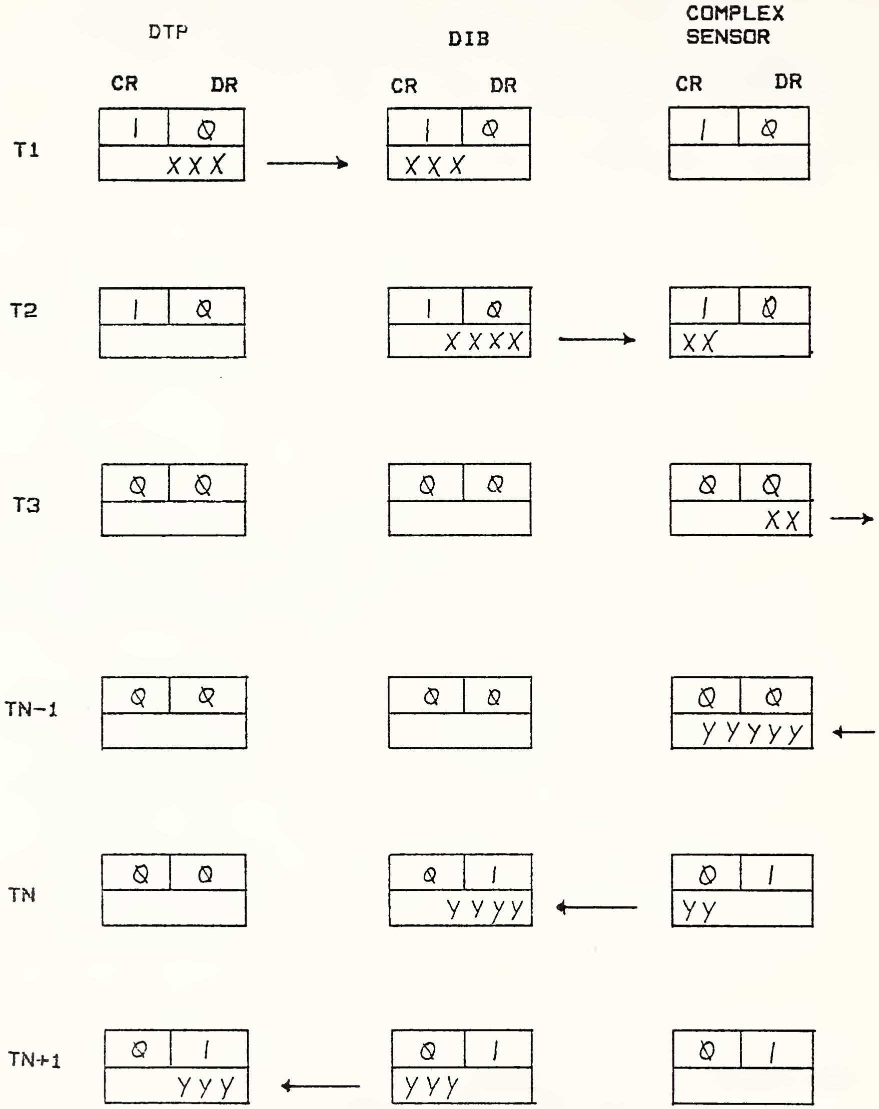
DIB PASSED SIMPLE DEVICE DATA DIRECTLY INTO RAM SPACE. ASYNCHRONOUS DATA ARE FIRST COLLECTED IN HIGH SPEED BUFFERS AND THEN PASSED INTO RAM.

FIGURE IV.17

to access complex device data in the same synchronous manner as it does the simple data. All the uncertainity of the data collection from these devices is removed through the DIB buffering mechanism.

A typical transfer of a command from the DTP to the complex sensor, with the sensor then returning data, is shown in Figure IV.18. There are two flags associated with this transfer: the Command Ready flag (CR) and the Data Ready flag (DR). The DTP writes a command to the DIB at time T1 and sets the CR flag. When the Transfer flag is reset at time T2, the DIB reads the CR flag, and, if it is set, transfers the CR flag and the command to the complex sensor input buffer. The DIB then resets the CR flag internally. At times T3 - TN-1, the complex sensor interprets the command and places the data into its output buffer. During this time, the DIB polls the DR flag until it is set. At time TN, the complex sensors finish, and set the DR flag. The DIB then loads the output buffer in its own buffer area. When the Transfer flag is set at time TN+1, the DTP reads the set DR flag, and reads in the data. After all the data is read, the DTP resets the DR flag to complete the transaction.

As mentioned before, block sizes are variable for asynchronous devices. This variablity is accomplished by having the external device (or the complex sensor) write a status word


DTP
DIB
COMPLEX SENSOR
CR
DR
CR
DR
CR
DR
T1
1
0
XXX
1
0
XXX
1
0
T2
1
0
1
0
XXXX
1
0
XX
T3
0
0
0
0
0
0
XX
TN-1
0
0
0
0
0
0
YYYYY
TN
0
0
0
1
YYYY
0
1
YY
TN+1
0
1
YYY
0
1
YYY
0
1

IV. 18 DIB COMMAND AND DATA TRANSFER

containing a byte count at the end of its transmission. This status word also contains flags which signal a ready-for-access to the DIB to transfer the data from the device to the buffer (or vice versa). Variable block size allows shorter cycle times and increased flexibility.

V. SUMMARY

The control system implementation described above is a first attempt to meet the requirements for a real-time sensory interactive robot control system.

The system has the ability to sample external devices and provide different levels of processing of this data to the control system through the sensory-processing hierarchy. The control hierarchy enables the system to respond to that data in a timely manner, making it indeed sensory-interactive in real-time.

A high level task programmer interface provided by the off-line programming module enables the user to describe the robot task as a sequence of English-like commands using symbolic names for locations and objects in the worksite. These commands greatly simplify the job of task description and keep it in a form independent of robot and worksite.

Task and data description have been separated. The English-like task description using symbolic names is independent of the data base values that are supplied when the tasks are executed. At this time, the values are assigned for the names specified in the program. In this manner locations and object descriptions are separate from task descriptions.

At the system programming level, the system is extensible through the state table programming technique used. New input and output variables may be supplied to any level in the control or sensory processing hierarchies by entering additional table entries that assign new output states for the given inputs. In addition, the modularity of the system makes it easy to determine at which level the system should be extended.

This architecture should make for a reliable system. Since it has been structured and modular, programmers at all levels have simple interface requirements to meet in programming the system. Ease in programming and ease of overall system comprehension increase reliablity because it decreases the possiblity of programming error. In addition, error conditions are easily recognized since the explicitly specified interfaces help isolate a malfunctioning module.

In addition, the software procedures help provide a well-defined control structure for the executing algorithms. This control structure generates only in-line sequential paths through the code, eliminating looping and multi-entry paths that can be caused by interrupt handling mechanisms. The resulting code is deterministic and comprehensible. The state of the system is reflected in the code at any time, and the programmer may easily view the state and generate correct and reliable responses for it.

Task 4.3.2 Imaging Sensor Development

Abstract

Work performed under this task investigated techniques for making effective use of imaging sensors and for incorporating interpretation of image processing into robot control systems on a real-time basis. The vision hardware and attributed lighting techniques used to acquire images are described. Software for determining binary image orientation and analyzing the 3-dimensional shape of binary images is presented. Circuit diagrams for the NBS vision system camera interface electronics are also included.

IMAGING SENSOR DEVELOPMENT

Task 4.3.2 is to investigate techniques for making effective use of imaging sensors and for incorporating interpretation of image processing into control systems on a real-time basis. The following sections discuss the software and hardware aspects of this problem and the experimental results obtained in this study. Section 4.3.2.1 is an overview of the vision hardware in use with the N.B.S. robots. The software and algorithms developed to perform image processing are discussed in 4.3.2.2. The hardware interfaces are described in section 4.3.2.3.

Vision System Overview

The sensory side of the NBS hierarchical control system contains a vision system which uses active illumination to obtain depth information. The vision system uses a solid state camera, computer-controlled light sources (flood flash and plane of light), and interface hardware accepting commands from and communicating data to a microcomputer. The camera and flash unit are fixed to the wrist of the robot manipulator (Figure 1).

The plane of light is generated by a photoflash tube and a cylindrical lens. This plane is projected into the field of view of a solid state camera so that the distance to an illuminated surface can be directly computed by simple tri-

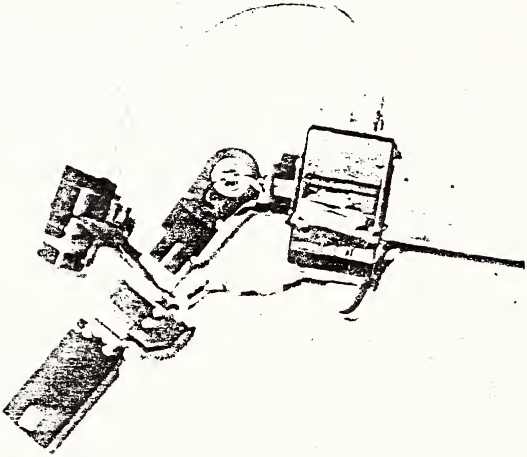


FIGURE 1

gonometry. Where the plane of light strikes a surface, a line (straight or curved) appears in the camera image. The distance to an illuminated point is directly related to the vertical position of the reflected light in the camera field. Figure 2 is a diagram of the geometry of this configuration. This type of illumination is called structured light.

The hardware interface between the camera and the microcomputer system not only provides the electrical interconnections, but also performs some pre-processing of the camera data. Figure 3 is a block diagram of this interface. First, the image is tested against a program selected threshold value to produce a binary image. Then this binary image data is compressed using the technique of run-length encoding. This compression technique takes advantage of the property that the raster scan lines typically contain long sequences or runs of constant value. Each such run is encoded by its length. This encoding process greatly reduces the volume of data that must be communicated to the microcomputer portion of the vision system.

The camera is mounted so that the raster scans lines run vertically, from bottom to top. Thus the vertical position of an illuminated spot in the camera image is its dot position in the scan line. This is precisely the information given in the run-length encoded data.

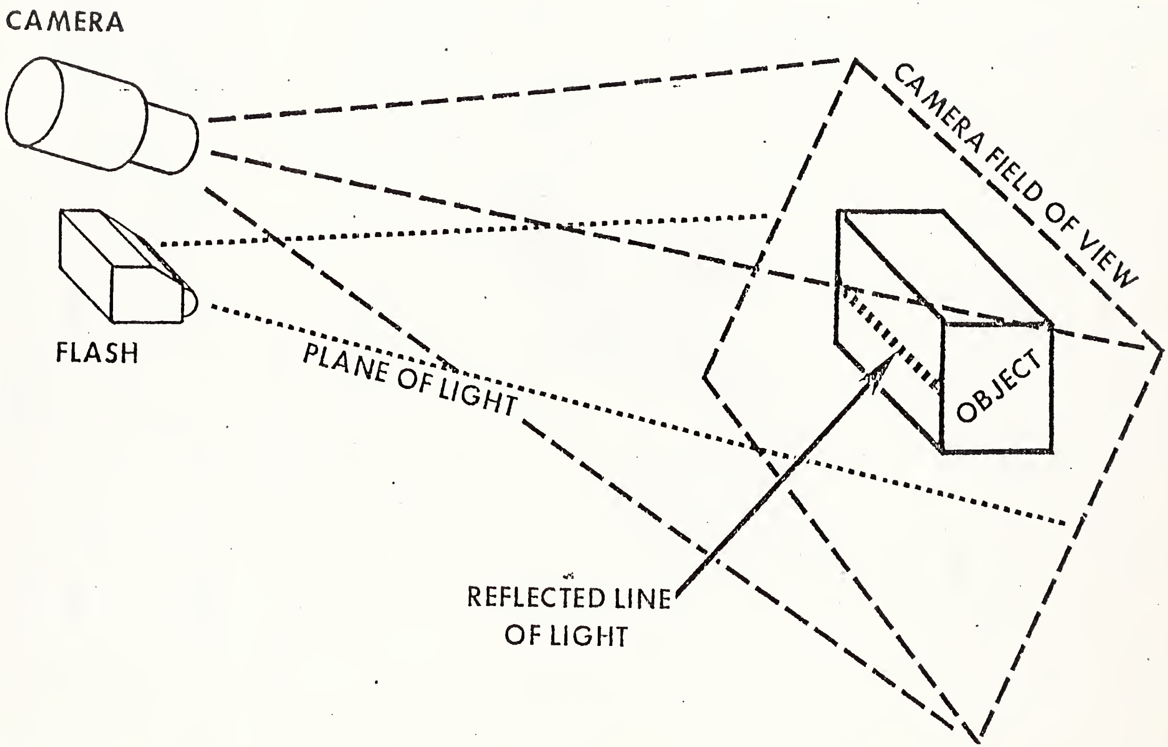

CAMERA
FLASH
CAMERA FIELD OF VIEW
OBJECT
PLANE OF LIGHT
REFLECTED LINE OF LIGHT

FIGURE 2. The plane of light forms a line segment image of an object in the camera field of view.

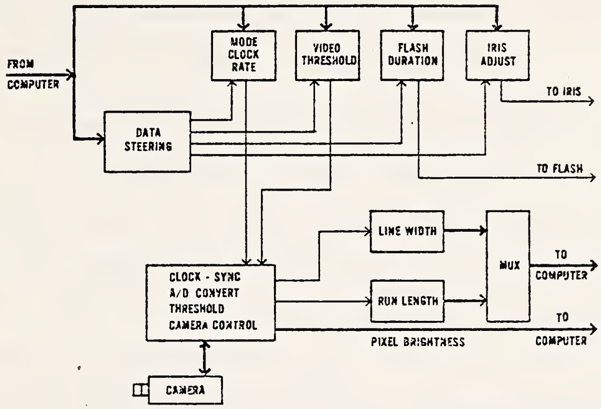

FROM COMPUTER
MODE CLOCK RATE
VIDEO THRESHOLD
FLASH DURATION
IRIS ADJUST
TO IRIS
DATA STEERING
TO FLASH
LINE WIDTH
MUX
TO COMPUTER
CLOCK - SYNC
A/D CONVERT
THRESHOLD
CAMERA CONTROL
RUN LENGTH
TO COMPUTER
PIXEL BRIGHTNESS
CAMERA

FIGURE 3. A block diagram of the NBS vision system interface hardware.

The following notation will be used for referring to image data scanned in the manner just described.

a) line i is a vertical column i spaces from the left, and

b) pixel j of line i is a picture element j spaces from the bottom of line i

The control hierarchy activates the vision system at specific points in a particular task execution. The control hierarchy also tells the vision software what type of object to expect and approximately how far away the object is expected to be. The vision software uses this information to select appropriate values for flash intensity and threshold and appropriate software algorithms for processing the visual data.

The vision-processing modules either confirm the existence of the expected object and tell the control system where to move to approach it, or report that the expectation was incorrect.

Image-Processing Software

Two major problems were addressed under this task. One is computing the orientation of two dimensional images. The second is computing the shape and orientation of a 3-dimensional part by the combination of information from two images taken from two separate illumination systems.

Two-Dimensional Orientation

An orientation study was done to determine the usefulness of the hole-blob program output (described in detail in the October 1979 Fourth Quarterly Report--"Robotics Support Project for the Air Force ICAM Program") in computing the orientation of an object. The study involved testing the accuracy of three different methods described below to examine the relationship between the center of area and the center of perimeter of a given object in different known positions. Knowledge of these centers enables one to determine the relative orientation of the object.

The center of area of an object is defined as (xbar,ybar) where:

xbar=the sum of the x coordinates of every point in the blob or hole divided by the total area of the space.

ybar=the sum of the y coordinates divided by the total area.

The center of perimeter of an object is defined as (xp,yp), where:

xp=the sum of the x coordinates of every perimeter point divided by the perimeter of the object.

yp=the sum of the y coordinates of every perimeter point divided by the perimeter of the object.

In figures 4, 5, 6 and 7, the center of area of the hole and blob are denoted as points A and C respectively. The centers of perimeter of the hole and blob are labeled points B and D.

Data for the study was collected by taking grey scale pictures of the same object in four different orientations--starting with a base position defined to be at the zero degree rotation. The object was rotated 15 degrees clockwise from the origin, and 30 degrees and 90 degrees counter-clockwise from the origin (Figures 4 thru 7). All measurements were approximated with a simple protractor.

Three different methods were used to determine the relative orientations of one object position to another. All methods rely on the assumption that a vector can be drawn between the points of interest (center of perimeter to center of area in Method 1; center of area of blob to center of area of hole in Method 2, and center of perimeter of blob to center of perimeter of hole in Method 3). Knowing the position of the vector at an origin position, and measuring the vector position after a rotation, one can subtract these values to determine the degree of rotation of the vector. The measured vector rotation corresponds to the actual rotation of the object. These methods are effective only when the two points being considered are not coincident. Method 1 cannot be used if the object has more than one axis of

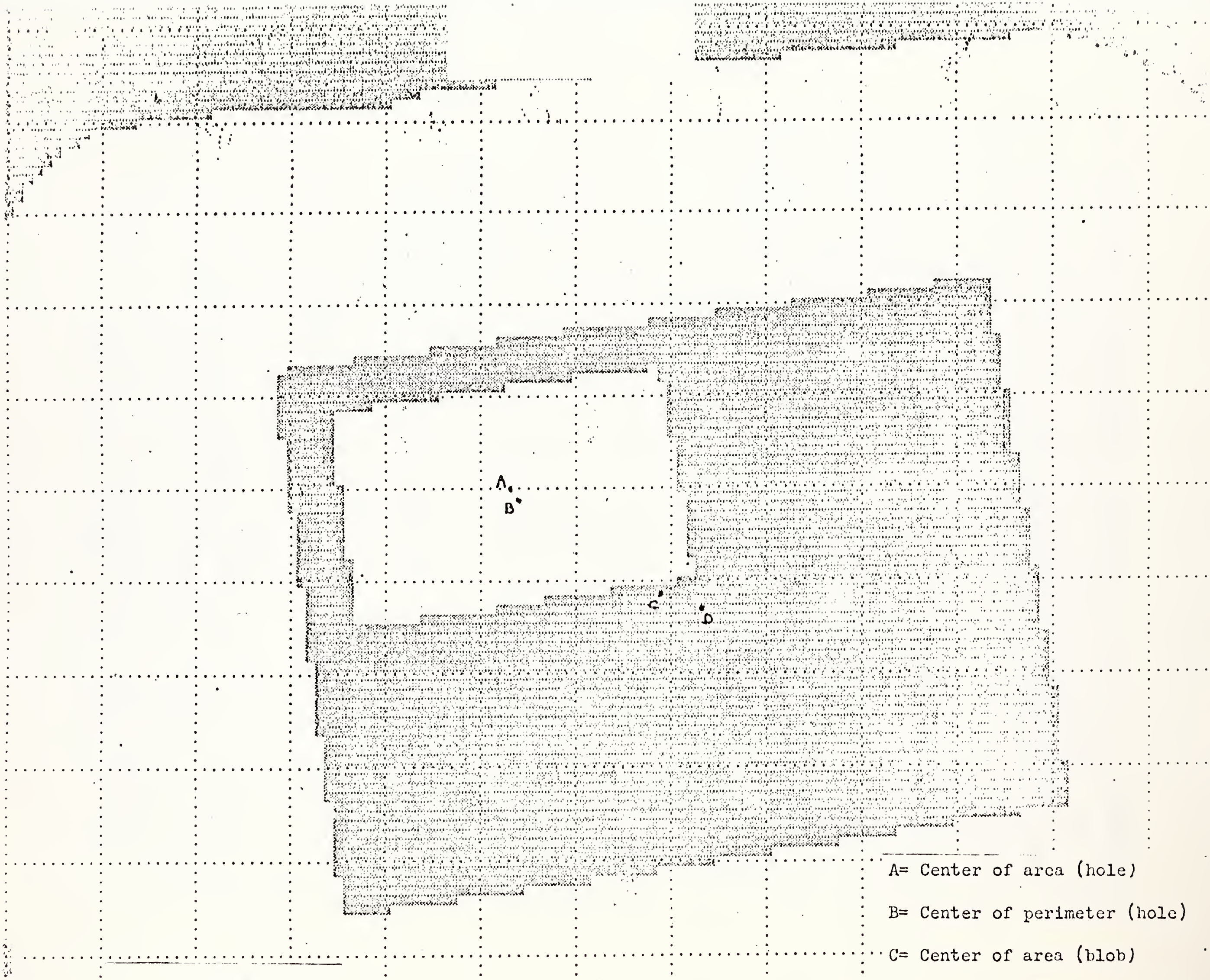

A
B
C
D
A= Center of area (hole)
B= Center of perimeter (hole)
C= Center of area (blob)
D= Center of perimeter (blob)
Origin

FIGURE 4

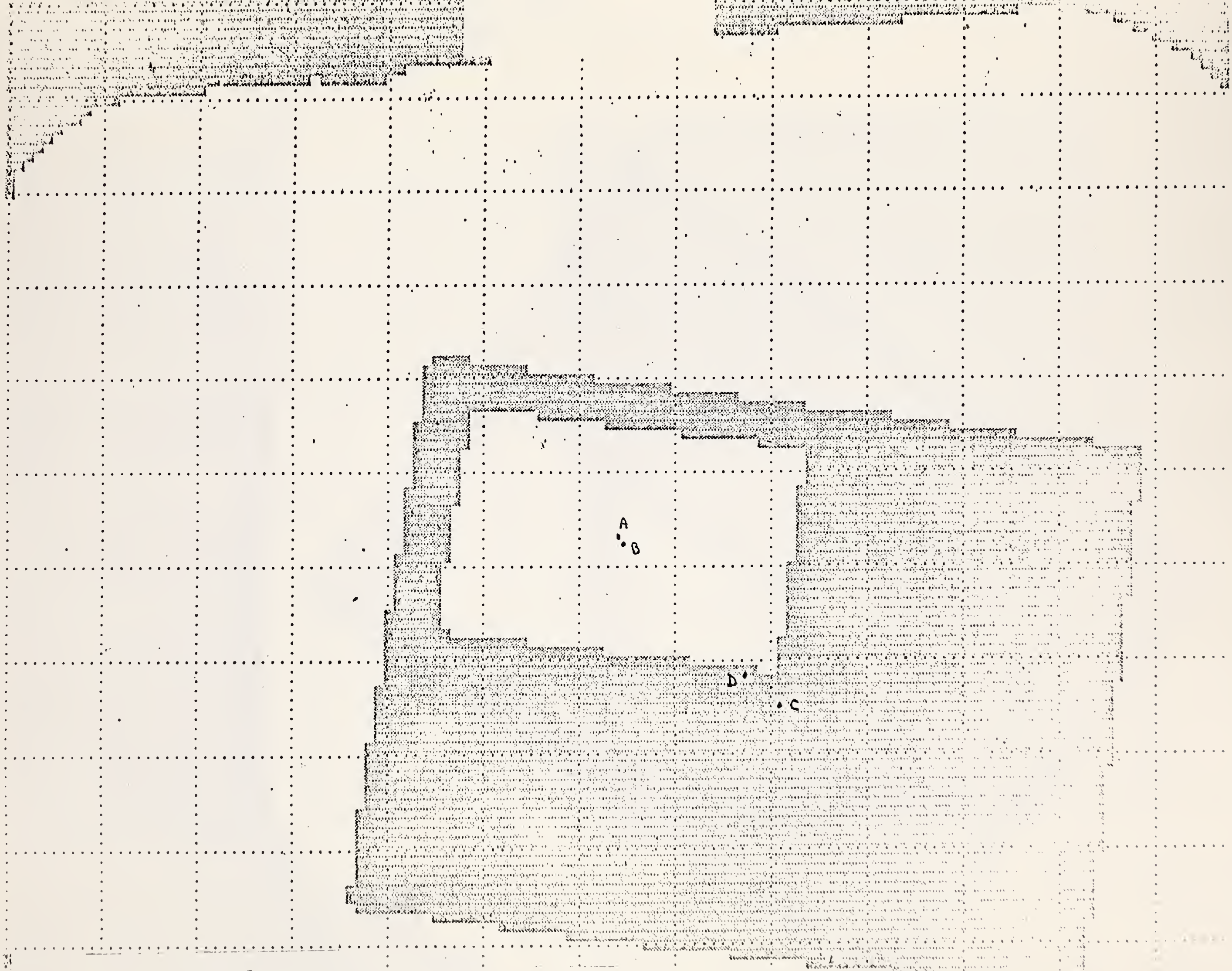

A
B
C
D
15 degrees

FIGURE 5

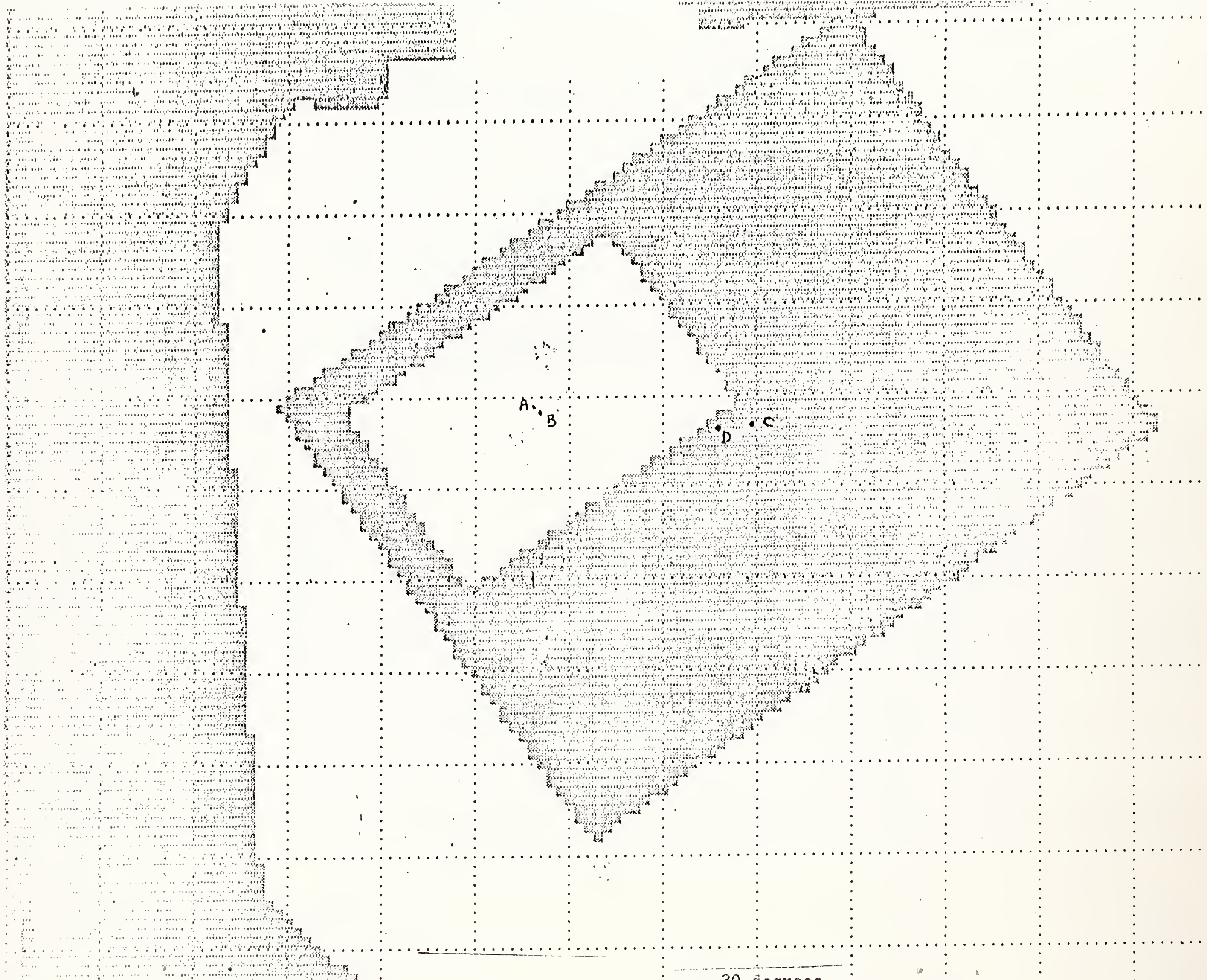


FIGURE 6

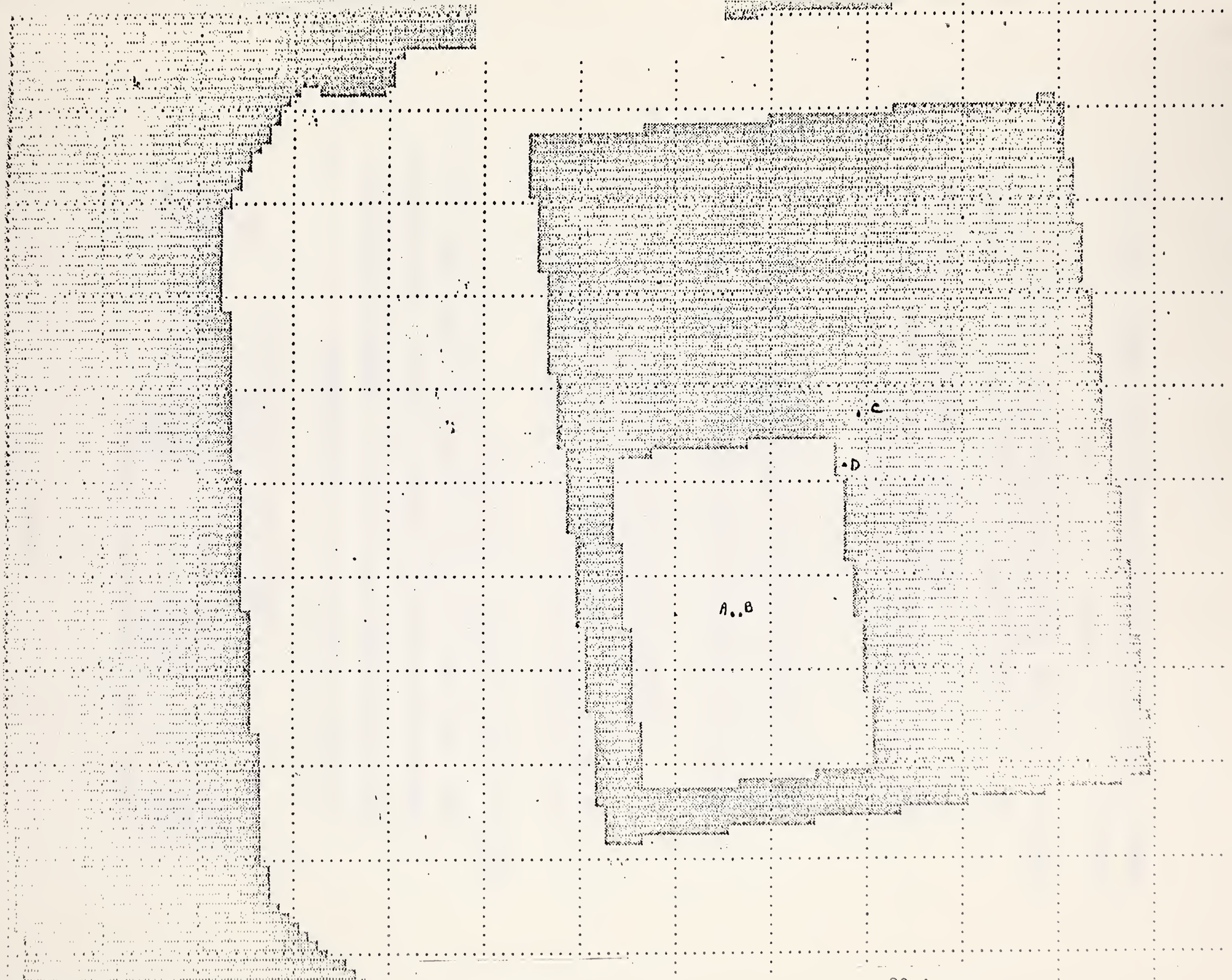


FIGURE 7

90 degrees

symmetry because that would mean the two points were not distinct.

Method 1: Center of Perimeter-Center of Area Blobs only

Given the coordinates of the center of perimeter (xp,yp) and the center of area (xbar,ybar) of the same blob, the direction theta1 of the vector from the center of perimeter to the center of area is defined as:

theta1=arctan((yp-ybar)/xp-xbar)) for View 1

The direction theta2 is computed in the same way

theta2=arctan((yp-ybar)/(xp-xbar)) for View 2

The angle of orientation is then computed to be theta1-theta2.

Method 2: Center of Area Blobs-Holes

Given the center of area coordinates for the blob and hole in one orientation, and the center of area coordinates of the same blob and hole in another orientation, the change of position can be computed as:

arctan(y1/x1)-arctan(y2/x2)

where y1,x1 are the deltas in y and x values between blobs and holes in View 1, and y2,x2 are the deltas between blobs and holes in View 2.

E.g. y1=y1bar blob -y1bar hole ;
x1=x1bar blob -x1bar hole ;
y2=y2bar blob -y2bar hole ;
x2=x2bar blob -x2bar hole .

Method 3: Center of Perimeter Blobs-Holes

Same as Method 2 except that coordinates for the center of perimeter are used in calculations.

Thus y1,x1 are the deltas in the centers of perimeter between blobs and holes in View 1, and y2,x2 are the deltas in centers of perimeter between blobs and holes in View 2.

Then the angle of orientation is defined as :

Theta=arctan (y1/x1)- arctan(y2/x2).

Below is a chart summarizing the results obtained from each of the methods. Column 1 represents the true measurement and columns 2-4 represent the results obtained by using methods 1, 2, and 3 respectively.

ANGLE	CP-CA	CA	CP
345	346.47	344.43	344.20
30	30.43	27.04	26.97
90	91.93	90.56	91.15
105	105.46	106.23	106.94
45	43.97	42.71	42.77
60	61.50	63.52	64.18

Method 1 has an average error of 1.5 percent while Methods 2 and 3 both have average errors in the 4 percent range. Based on these preliminary investigations, the best orientation information is based on the comparison of center of area to center of perimeter of blobs.

Three Dimensional Shape and Orientation

As described in the Third Quarterly Interim Technical Report (July 1979), the camera being used with the robot has the capability of taking thresholded video images and run-length encoding the thresholded picture. See figures 11a and 11b for examples of run-length coding. Two modes of flash intensity can be used in taking pictures: line flash or flood flash. The line flash is a plane of light which intersects an object in a relatively narrow band. Thus it is necessary in most cases to take more than one line flash image of an object from varied positions, and to analyze each of these flash lines, before determining the shape of the object being examined.

The flood flash mode has an effect similar to that of a commercial camera flash attachment, i.e. it bathes the object in light and thus makes it possible to "see" the entire shape of the object in one view. As with the line flash, run length coding is used to define the pixel numbers where transitions from white-to-black and black-to-white occur.

Using the flood-flash capability however, the first transition defines the bottom of the object, and the second transition defines the top of the object for most interior lines. As illustrated in Figure 8, the edge effects of a computer image are very irregular and thus cannot reliably be used in defining bottom and top.

Using a line flash to determine an object's distance from the camera (y distance), followed by a flood flash of the same object, one can determine the "computed" height and the angle of tilt of that object. The angle of tilt is defined as the measure of an object away from the vertical. Calibration charts have been prepared which, in conjunction with trigonometric properties of the camera system and its environment, allow line and pixel units (column and row in image) to be associated with the x,y,z coordinates of a point in space. The height of an object is computed by first determining the coordinates (x1,y,z1) of the line at its first transition, and then determining the coordinates of the point (x2,y,z2) at the second transition. "Computed" height is defined as the difference in z measurements, z2-z1.

In Figure 8, computations for height were arbitrarily chosen at line 61. Calibration charts based on the trigonometry of the camera were computed such that for every picture point

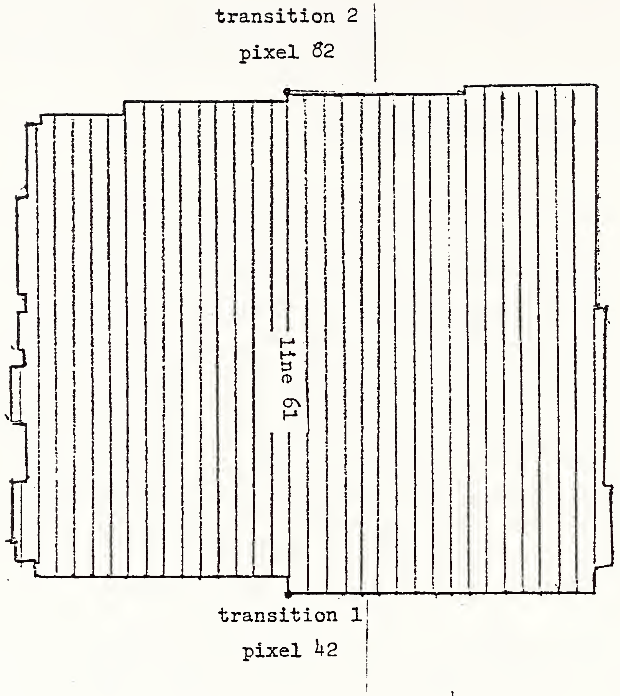

transition 2
pixel 82
line 61
transition 1
pixel 42

FIGURE 8

whose coordinates are (line,pixel) and which is a known distance y from the camera origin, values for x and z can be computed as functions of the pixel position. In particular,

$$x = (y + k0) * f1(pixel) * (line - 64)/64$$

$$z = k2 - (y + k0) * f2(pixel)/k3$$

where all constants are fixed measured distances relevant to the geometry of the camera on the robot wrist. Using the calibration charts and the above formulas, the following information was obtained:

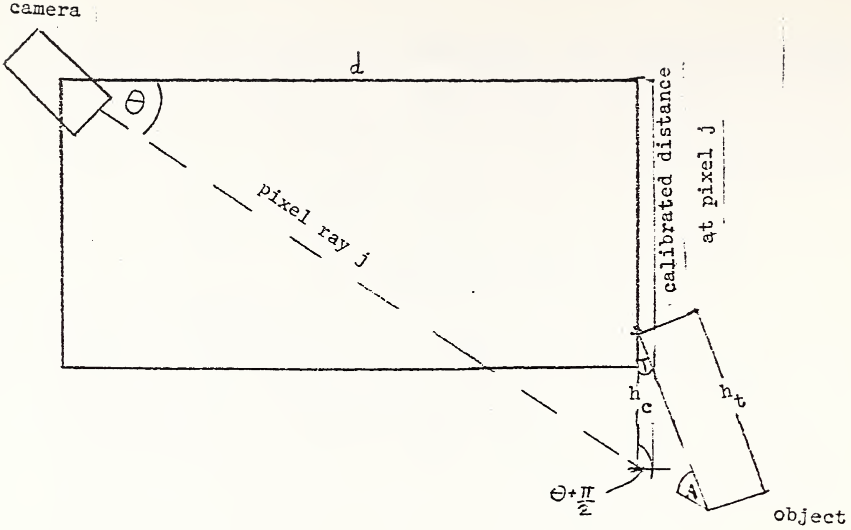
line	pixel	x	y	z
61	42	.45	16.	-2.45
61	82	.42	16.	3.72

The computed height of the object is 3.72 - (-2.45) or 6.17cm.

Knowing the true height of an object and its computed height, the angle of tilt can be computed. Refer to Figure 9 for the following discussion. Physical distances have been calibrated to every pixel value, and thus the value of angle theta can be computed :

$$theta = \arctan(calibrated\ distance / d)$$

Using the law of sines, the value of angle A can be evaluated :


camera
d
θ
pixel ray j
calibrated distance
at pixel j
h_c
h_t
θ+π/2
A
T
object

h_c = computed height

h_t = true height

T = tilt angle

$$\sin A = \frac{h_c \cdot \sin(\theta + \pi/2)}{h_t}$$

$$T = \pi - (A + \theta + \pi/2)$$

FIGURE 9

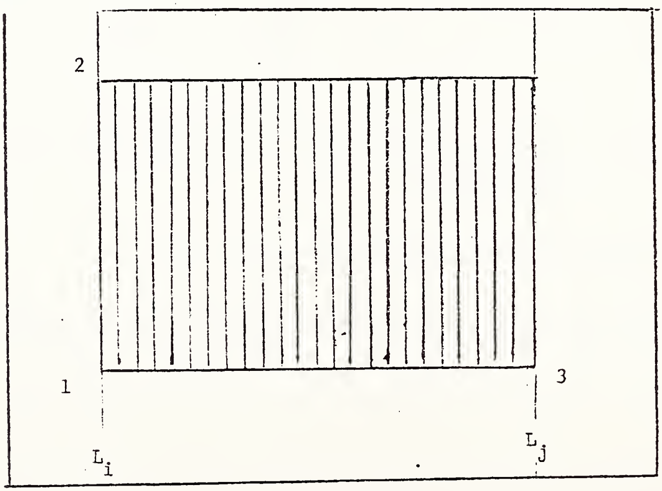


FIGURE 10

computed height/sin A = true height/sin(theta+pi/2).

The angle of tilt is then defined as :

T= pi-(A+ theta +pi/2).

Corner Detector

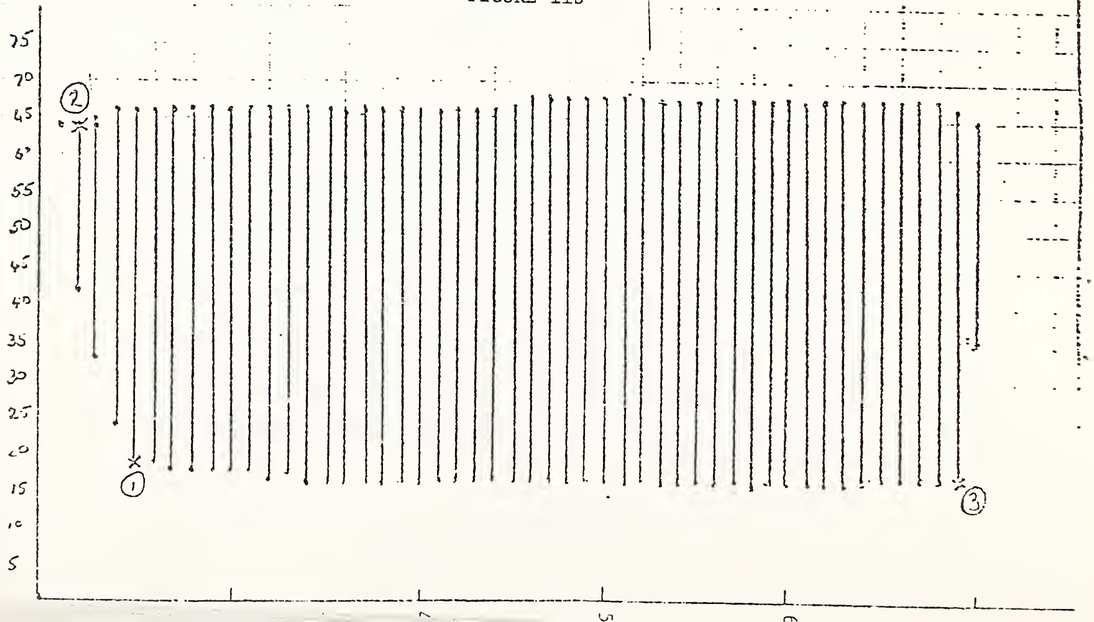
The ability to determine the coordinates of the corners of a square or rectanglar object is helpful in determining the orientation of that object. Toward that end, a corner detection algorithm has been written. The window containing an object is defined as those lines in an image containing continuous information about the object. The window is terminated by a single row containing no information. (See figure 10) The window of interest in Figure 10 would be defined as li to lj.

Knowledge of the coordinates of three corners of an object is sufficient to define the plane in which that object lies; therefore, this algorithm computes the line and pixel numbers of corners 1, 2 and 3 only (Figure 10). Input information received from the camera is in run-length coded form, i.e. transitions from white-to-black and black-to-white are recorded by the pixel value at which the transition occurs. The first transition will be denoted by rl, and the second transition will be denoted by chg2. Figure 11a is a computer printout of rl and chg2 vaues representing

FIGURE 11a

LINE	RL	CHG2
20	0	
21	64	64
22	42	64
23	33	65
24	24	66
25	19	66
26	19	66
27	18	66
28	18	66
29	18	66
30	18	66
31	18	66
32	17	66
33	18	66
34	17	66
35	17	66
36	17	66
37	17	66
38	17	66
39	17	65
40	17	66
41	17	66
42	17	66
43	17	66
44	17	65
45	17	66
46	17	67
47	17	67
48	17	67
49	17	67
50	17	67
51	17	67
52	17	67
53	17	67
54	17	67
55	17	67
56	17	67
57	17	67
58	17	67
59	17	67
60	17	67
61	17	67
62	17	67
63	16	67
64	17	67
65	17	67
66	17	67
67	17	67
68	17	67
69	17	66
70	36	65
71	-0	
72	0	

FIGURE 11b



a camera image of a rectangle. Figure 11b illustrates graphically this representation: the lower points are r1 (first transition) and the upper points are chg2 (second transition).

The algorithm defines the coordinates of corner 2 upper-left corner as line=i, pixel position= chg2(i) which is the pixel position of chg2 in line i such that

$$chg2(i)-chg2(i+1) < 1$$

The use of a range of acceptable values in defining any corner is necessitated by the fact that computerized images have very irregular edges.

The coordinates of corner 1 (lower-left corner) are defined as line =j, pixel position=r1(j) such that:

$$r1(j+1)-r1(j) < 1$$

Lastly, the coordinates of corner 3 are defined as line= k, pixel value =r1(k) such that:

$$r1(k+1)-r1(k) > 2$$

The corners detected by this algorithm in figure 9b are represented by 1 , 2 , and 3.

Model-Based Vision

To deal with complicated machine parts in a real environment, the vision system must have models of the objects involved. Three-dimensional models of the objects, in conjunction with structured lighting, are used to determine the positions and orientations of the objects in space. The models enhance the robustness of the vision system by enabling various kinds of noise to be ignored, and by supplying information not visible to the camera.

The automation control system is defined in a hierarchical manner, and the vision system must supply feedback of an appropriate kind to various levels in the hierarchy. This hierarchical structure leads to the definition of models at several levels, and it is intended that lower level models be generated from higher level models using, for example, projections of the three-dimensional model onto a two-dimensional plane.

Object Models

As an initial step, three-dimensional models of objects are constructed by hand, based on edge and corner information (wire frame models). Two automatic methods of constructing models are possible. The first method takes the output of a computer-aided design system and uses it to construct three-dimensional models for use in the manipulation system. The second method uses the vision system itself to construct the models.

The object models are intended to be convenient descriptions of the geometric, visual, and functional aspects of the types of objects encountered in an industrial environment. The functional information will be used in task planning and high level control but is not particularly important for the research described here. The geometric data is a representation of the three-dimensional shape of the object, while the visual component contains essentially two-dimensional information about the appearance of the object.

The three-dimensional geometry is assumed to have been extracted from a computer aided design (CAD) database having content similar to the IGES preliminary standard (Nagel, et al. 3). The object's shape is represented in a hierarchical, linked structure of graphic elements in three-dimensional Euclidean space. Each element is constructed around a template consisting of fields for the element's name, a typecode, a body that defines the element, and a list of attributes. The attribute list contains auxiliary information about the object. Some of this - such as surface texture, geometric data like area or length that are not in the body field, and relationships with other objects - are of use in the vision processing. Other attributes may be functional information such as grasp points for use in task planning.

The form of the body field of a model depends on its typecode. A composite object is defined by its component elements, which may be joined together at virtual edges or surfaces (i.e. edges or surfaces created for descriptive convienence without having physical existence). The component set of a composite object may include elements that are subtracted from the others to form holes. Non-composite objects are defined in terms of elements of lower dimensionality. For example, a polyhedron is defined by the collection of its polygonal faces. The polygons are defined by the line segments which are its edges, etc. The relationships between elements can be given explicitly in the attribute lists or can be derived from their common sub-elements. At the most fundamental level, this decomposition stops with point objects defined by their coordinates. No other elements have their own coordinates. This reduces the effort involved in applying coordinate transforms to objects; only the point coordinates need be changed because everything else refers to them.

The visual modelling information can be thought of as providing predictions about the appearance of an object or cues to the vision system as to what features should be looked for in the matching process. This information can be derived from the geometric shape data in the same way that a graphics system generates display data, i.e. by projecting onto an image plane and applying any of well known hidden line elimination algortihms. From this two-dimensional data, corner and edge lists are obtained as the features to be used in the shape matching step. To provide a starting point for the matching process, some typical views are computed and included in the object models. Other views from specific viewpoints are produced only after an initial match has hypothesized position and orientation.

The second way of constructing three-dimensional models is to use the camera system itself. This process involves integrating several views of an object into a single model. Also, unless structured light or shape from shading or texture is used, it is necessary to map from the two-dimensional pictures to the three-dimensional object model. Both the plane of light and the flood light can be used to construct the models. Surface shading and texture are not available because the images are thresholded.

While the plane of light can be used by itself to form models of objects (Agin and Binford 1 , Nevatia and Binford 4 , Shneier 8 , Sugihara 9), it has two disadvantages. It is very slow because multiple pictures are needed to scan the side of an object, and it is able to see only those parts of the object that are actually illuminated by the plane of light. Similarly, while a flood light can be used to construct a set of views of an object from all sides, it is not easy to reconstruct the three-dimensional shape from the set of views (Falk 2 , Roberts 5 , and Underwood and Coates 10). It seems, however, that a joint approach could give the advantages of both techniques without their disadvantages.

A set of floodlit pictures can be taken that encompasses all sides of the object. Associated with each flood flash picture are one or more plane-of-light pictures (Figure 12). The plane of light supplies depth and curvature information, while the flood light supplies shape information. The depth and shape information complement each other, in that depth enables the various pictures to be fitted together more easily, and gives information about the number of surfaces visible in each picture, as well as their relative orientations. The shape information supplies details about the outlines of surfaces and the presence of holes. Together, they give all the information needed to construct useful models of three-dimensional objects.

Initially, the models serve only to handle noise and missing data in the images. It is intended that they will be extended to provide information about invisible surfaces, to aid in quality control, and to supply non-visual information to

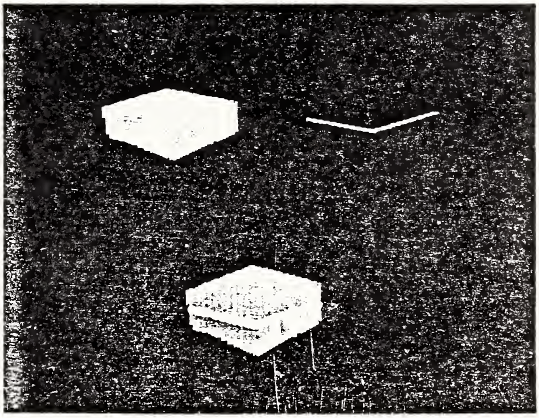


FIGURE 12. a. The thresholded image of a cube as seen under flood lighting.

b. The cube as seen using the plane of light.

c. The two images superimposed.

the control system, such as prespecified grasp points.

Matching a picture of an object with a model is a relaxation-like process (6) that involves models of several sorts. Initially, only two kinds of models are used, structural, three-dimensional models, and two-dimensional image-like models constructed by projection from the three-dimensional models. To construct the two-dimensional models, an initial match with a three-dimensional model must be found, and the correct projection calculated.

The matching process starts when a picture is taken. A set of features, including corners and edges and the distances (in millimeters) between corners, is calculated for the largest connected component in the picture.

The features are matched with the three-dimensional model or set of models, using a relational-structure matching algorithm (7). The best match is tentatively chosen as identifying the object. Since a hypothesized match includes the position of the object, a two-dimensional image model can be calculated from the structural model by geometric projection.

When the image model is overlayed on the original picture, an error measure can be calculated. Before that is done, however, the connected component analysis is performed again, but this time accepting feedback from the image

model. Thus, when a new component is found, a check is made to see whether or not it is part of the current object by seeing if it is also overlayed by the image model. Small dropouts can be ignored because the model fills in the gaps, and objects whose projection consists of more than one component can be correctly handled. The result of this process is a reduction in the noise of the image, and a more informed component analysis.

Features of the new component are again calculated and matched with the three-dimensional model. If the match is worse, the model can be rejected, while if it improves, the process is repeated until an error condition is met or no further improvement occurs. At this stage, the model can be used to supply position and orientation information to the manipulator, even in the presence of noise or when the object is partially out of the field of view of the camera.

The models could also contain information about properties of the surfaces of the object, as well as suggesting ways of distinguishing between apparently similar objects. For example, model predictions of highlights and shadows would be useful in analyzing shiny objects using multiple point sources.

Camera Interface Design

In the Third Quarterly Interim Technical Report dated July

1979 a vision system camera interface design was described with the following features:

1. The threshold of the video image may determined by external hardware.
2. Run-length encoding of the thresholded picture.
3. Access to sufficient grey-scale information to optimize light intensity and thereshhold settings in the creation of binary images.
4. Software control of structured lighting, threshold settings, and camera parameters.
5. Software control of windowing of areas of interest.

During the periods since then, the following work was accomplished:

1. The hardware interface electronics were completed and sucessfully tested.
2. The vision interface hardware was interfaced with a single board microcomputer which was itself integrated into the NBS microcomputer network architecture.
3. A series of tests were run including the NBS demonstration of the robot using structured light vision to find, pick up, and sort rectangular and cylindrical objects from a random pile of such objects.

4. The same design was used to interface a new camera with the system. The new camera has a resolution of 244 by 244 pixels as compared to the 128x128 resolution of the camera used in the earlier system. In addition, the new camera has a greater tolerance of intensity overloads.

Figures 13 through 23 are circuit diagrams of the vision interface hardware. Figure 13 is a block diagram of the vision interface hardware. Figure 14 is a circuit diagram of the manual data input section and the control logic decoding circuitry. Figure 15 is a diagram of the clock rate and read mode circuitry. Figure 16 is a diagram of the flash duration control circuitry. Figure 17 is a diagram of the run length and run length buffer circuitry. Figure 18 is a diagram of the threshold, digitized video buffer, and computer handshaking control circuitry. Figure 19 is a diagram of the iris adjust circuitry. Figure 20 is a diagram of the flash trigger isolation circuitry.

ROBOT VISION
INTERFACE
BLOCK DIAGRAM

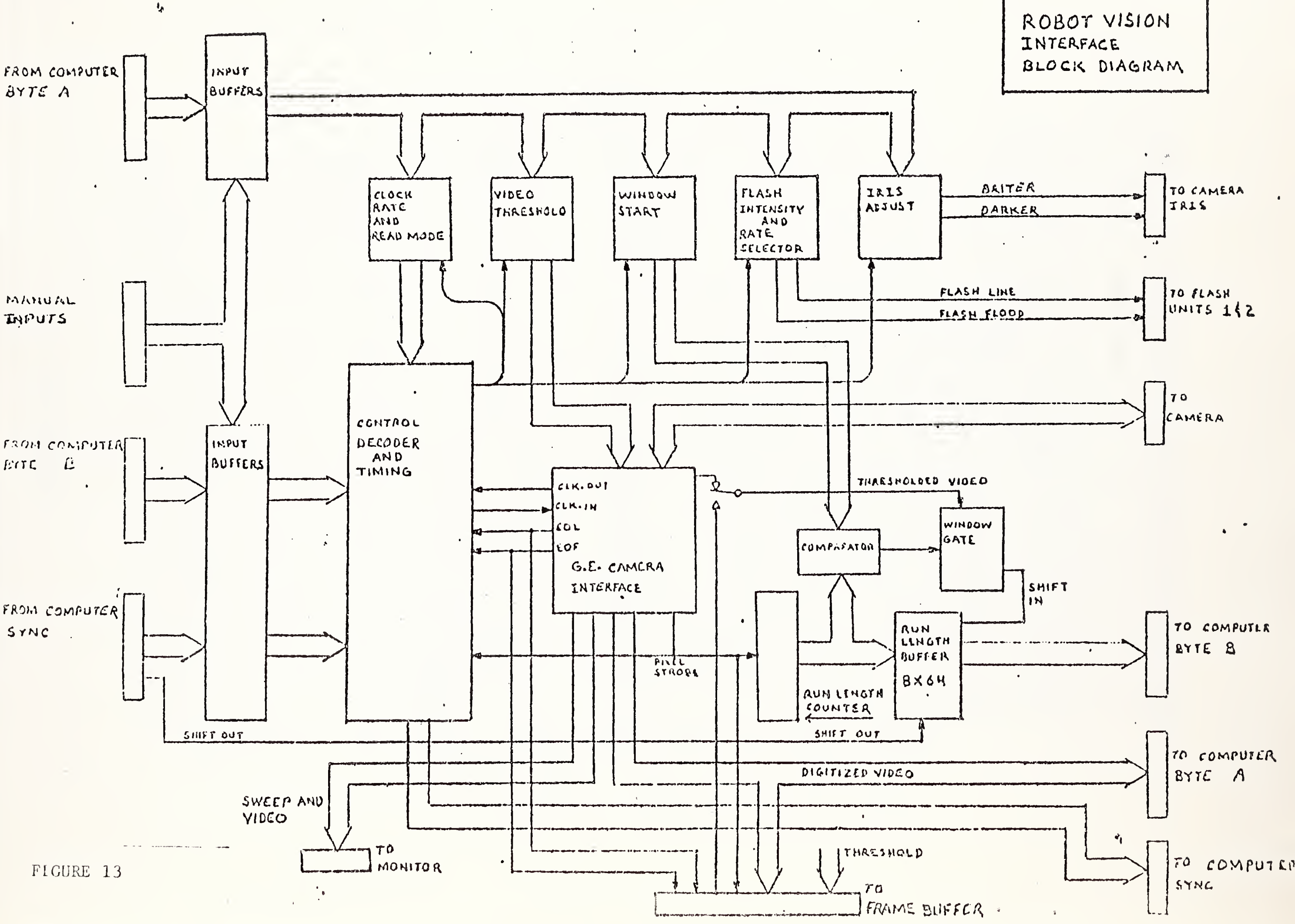

FROM COMPUTER BYTE A
MANUAL INPUTS
FROM COMPUTER BYTE B
FROM COMPUTER SYNC
INPUT BUFFERS
INPUT BUFFERS
CLOCK RATE AND READ MODE
VIDEO THRESHOLD
WINDOW START
FLASH INTENSITY AND RATE SELECTOR
IRIS ADJUST
BRITER
DARKER
TO CAMERA IRIS
FLASH LINE
FLASH FLOOD
TO FLASH UNITS 1&2
TO CAMERA
CONTROL DECODER AND TIMING
CLK. OUT
CLK. IN
EOL
EOF
G.E. CAMERA INTERFACE
THRESHOLDED VIDEO
COMPARATOR
WINDOW GATE
SHIFT IN
RUN LENGTH BUFFER 8X64
RUN LENGTH COUNTER
PIXEL STROBE
SHIFT OUT
SHIFT OUT
TO COMPUTER BYTE B
TO COMPUTER BYTE A
DIGITIZED VIDEO
TO COMPUTER SYNC
SWEEP AND VIDEO
TO MONITOR
THRESHOLD
TO FRAME BUFFER

FIGURE 13

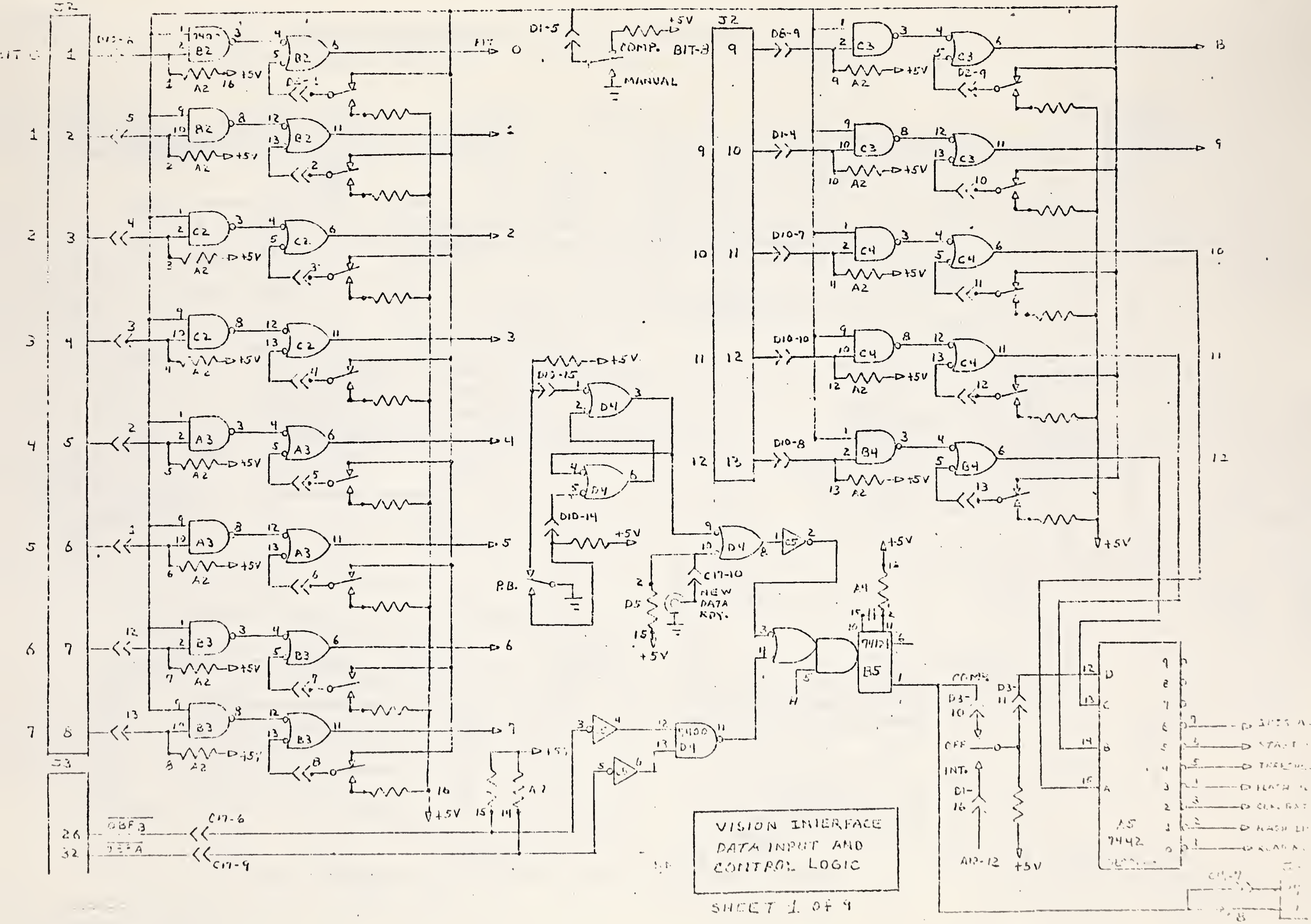

VISION INTERFACE
DATA INPUT AND
CONTROL LOGIC
SHEET 1 of 9
J2
J3
COMP. BIT-8
MANUAL
P.B.
NEW DATA RDY.
C17-10
C17-6
C17-9
OBF B
IBF A
D1-5
D6-9
D1-4
D10-7
D10-10
D10-8
D10-14
D13-15
D5
A1
A2
7400
74121
B5
A5
7442
COMP.
OFF
INT.
D1-16
A10-12
+5V

FIGURE 14

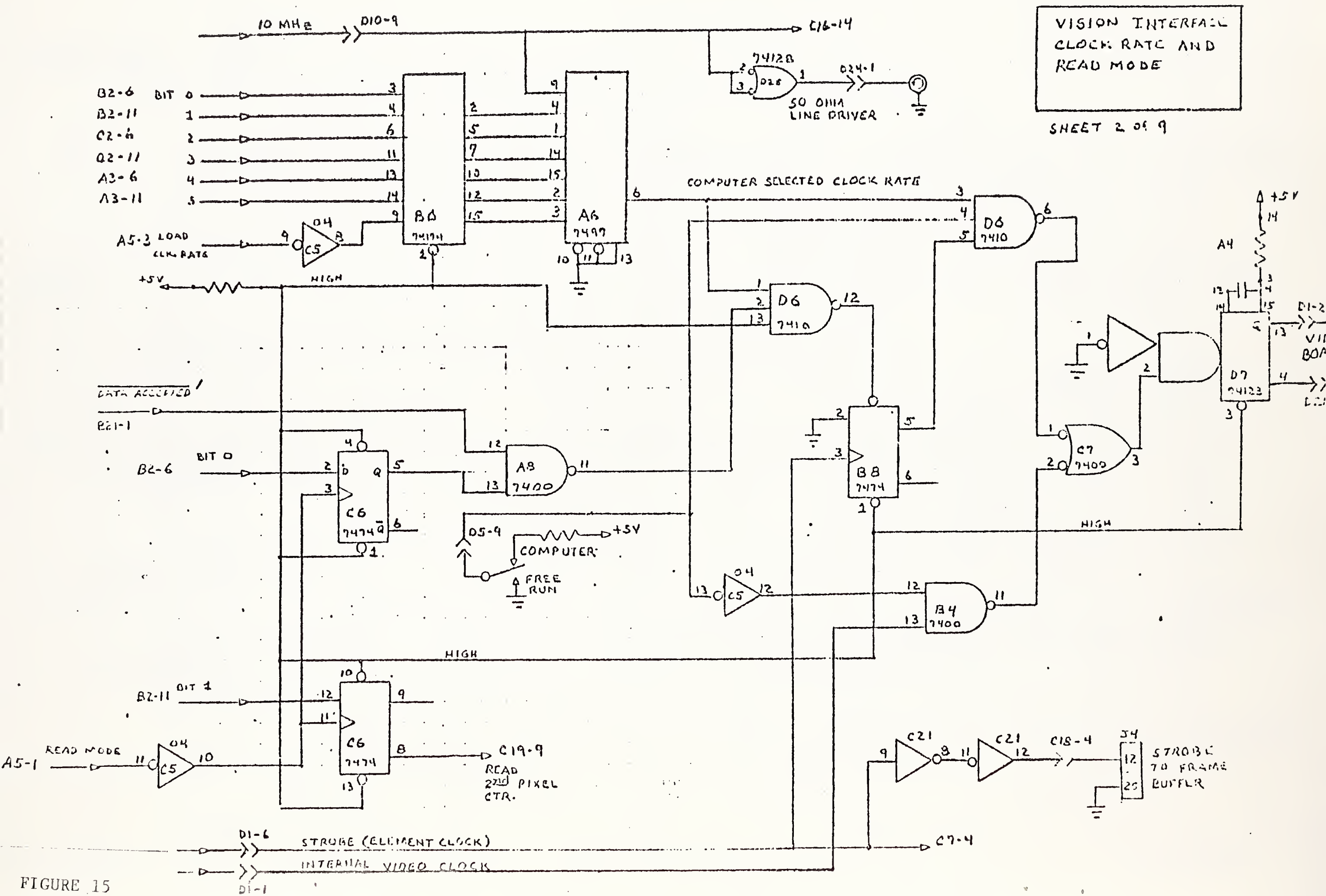

VISION INTERFACE
CLOCK RATE AND
READ MODE
SHEET 2 of 9
10 MHz
D10-9
C16-14
74128
D28
D24-1
50 OHM
LINE DRIVER
B2-6 BIT 0
B2-11 1
C2-6 2
Q2-11 3
A3-6 4
A3-11 5
A5-3 LOAD
CLK RATE
04
C5
B4
74194
A6
7497
HIGH
+5V
COMPUTER SELECTED CLOCK RATE
D6
7410
A4
+5V
D7
74123
D1-2
VI
BOA
DATA ACCEPTED
B21-1
BIT 0
B2-6
C6
7474
A8
7400
B8
7474
C7
7400
D5-9
COMPUTER
FREE
RUN
+5V
B4
7400
HIGH
B2-11
BIT 1
C6
7474
C19-9
READ
2nd PIXEL
CTR.
A5-1
READ MODE
C21
C21
C18-4
J4
STROBE
TO FRAME
BUFFER
D1-6
STROBE (ELEMENT CLOCK)
C7-4
INTERNAL VIDEO CLOCK
D1-1

FIGURE 15

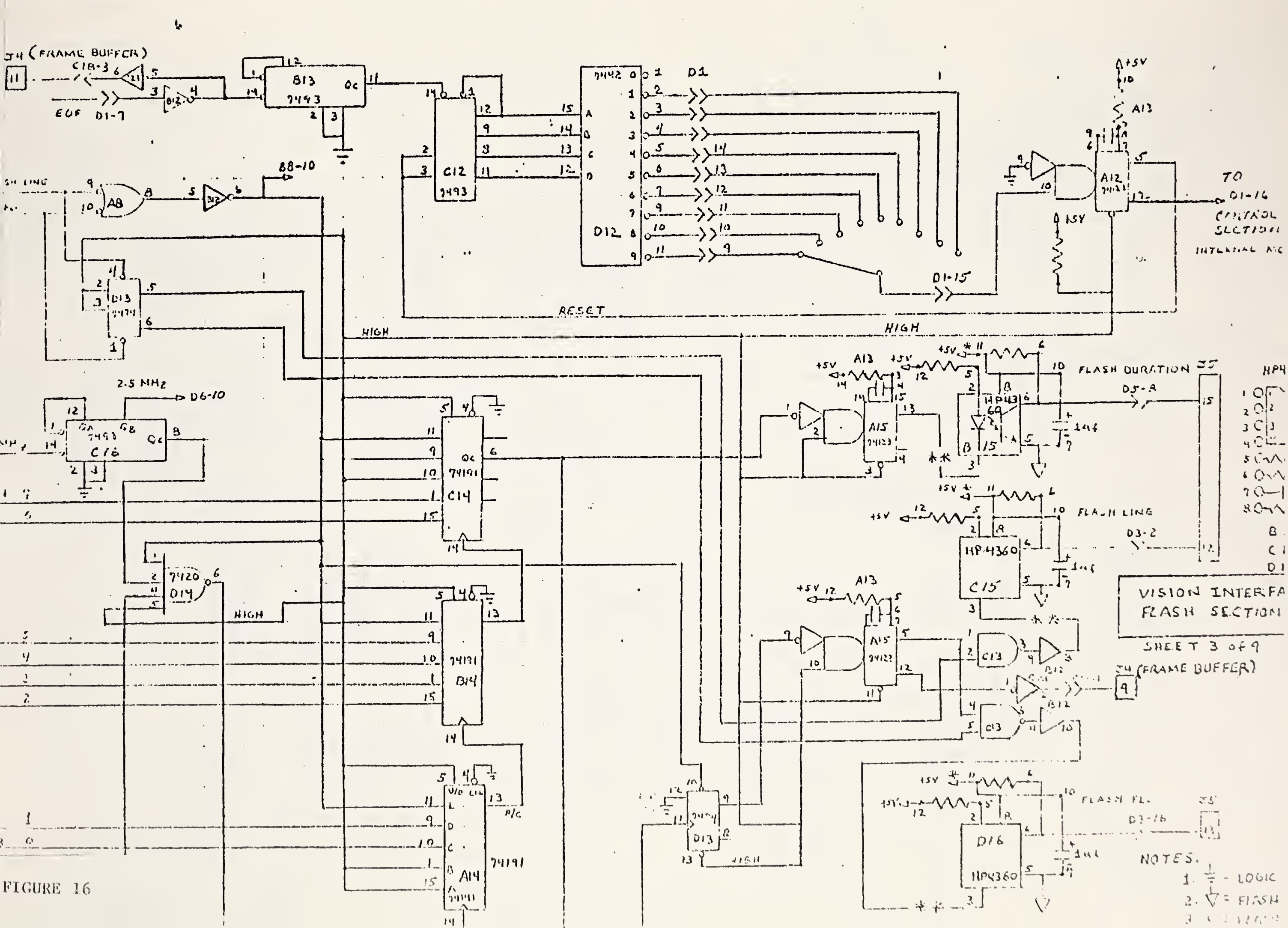

J4 (FRAME BUFFER)
C18-3
EOF D1-7
B13 7493
C12 7493
7442 D12
D1
A13
+5V
A12 74121
TO D1-16 CONTROL SECTION INTERNAL MC
88-10
A8
B12
D13 7474
RESET
HIGH
D1-15
2.5 MHz
D6-10
C16 7493
C14 74191
B14 74191
A14 74191
7420 D14
A15 74123
HP4360
C15
D16
C13
FLASH DURATION
D5-8
FLASH LINE
D3-2
FLASH FL.
D3-16
J5
VISION INTERFA
FLASH SECTION
SHEET 3 of 9
J4 (FRAME BUFFER)
NOTES.
1. = LOGIC
2. = FLASH

FIGURE 16

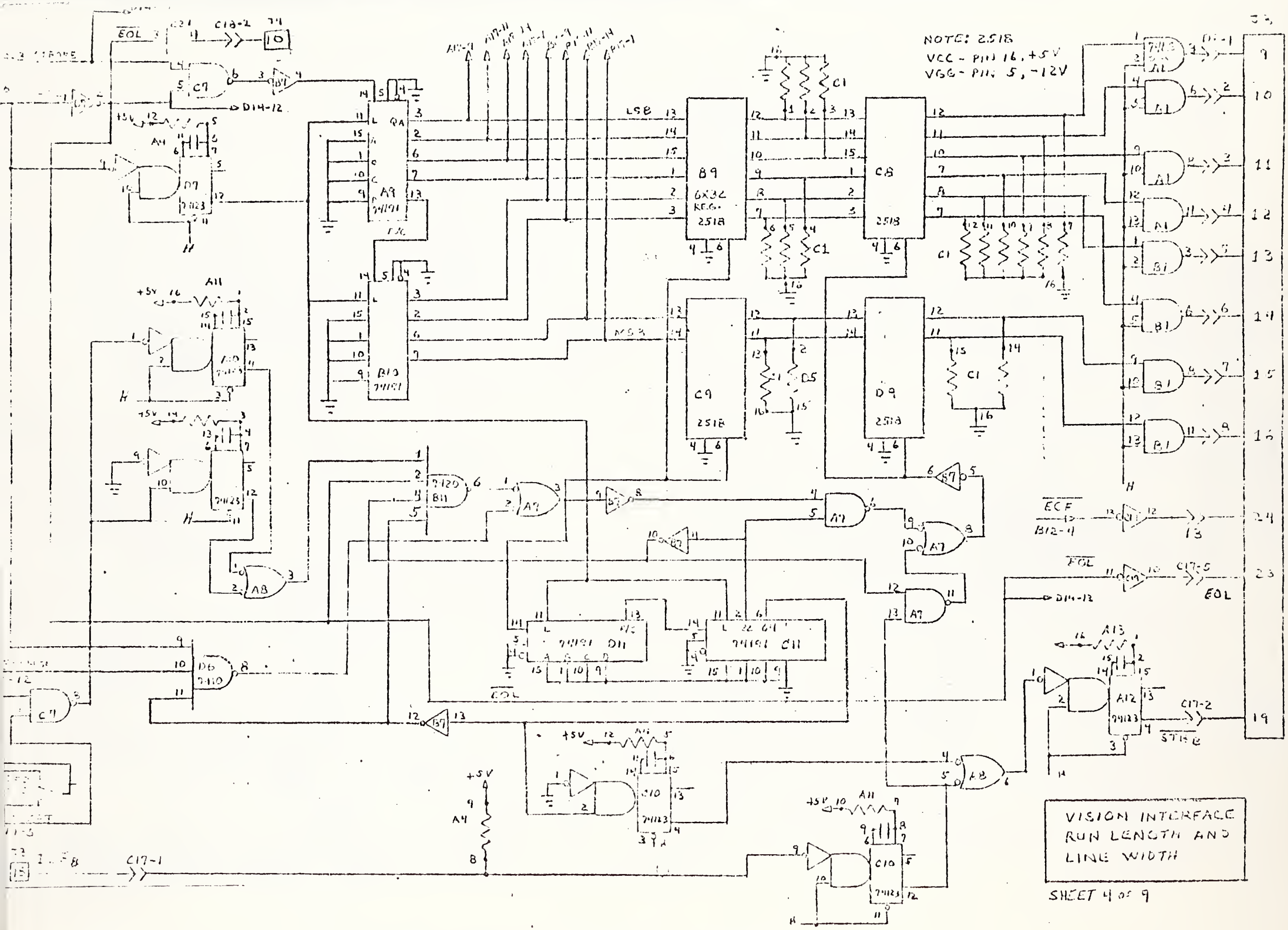

NOTE: 2518
VCC - PIN 16, +5V
VGG - PIN 5, -12V
VISION INTERFACE
RUN LENGTH AND
LINE WIDTH
SHEET 4 of 9
LSB
MSB
B9
6X32
REG.
2518
C8
2518
C9
2518
D9
2518
A9
74191
B10
74191
D11
74191
C11
74191
A12
74123
C10
74123
D7
74123
EOL
ECF
B12-4
EOL
C17-5
C17-2
STB
C17-1
C18-2
D14-12
+5V
A4
A11
A13
C1
D5
H

FIGURE 17

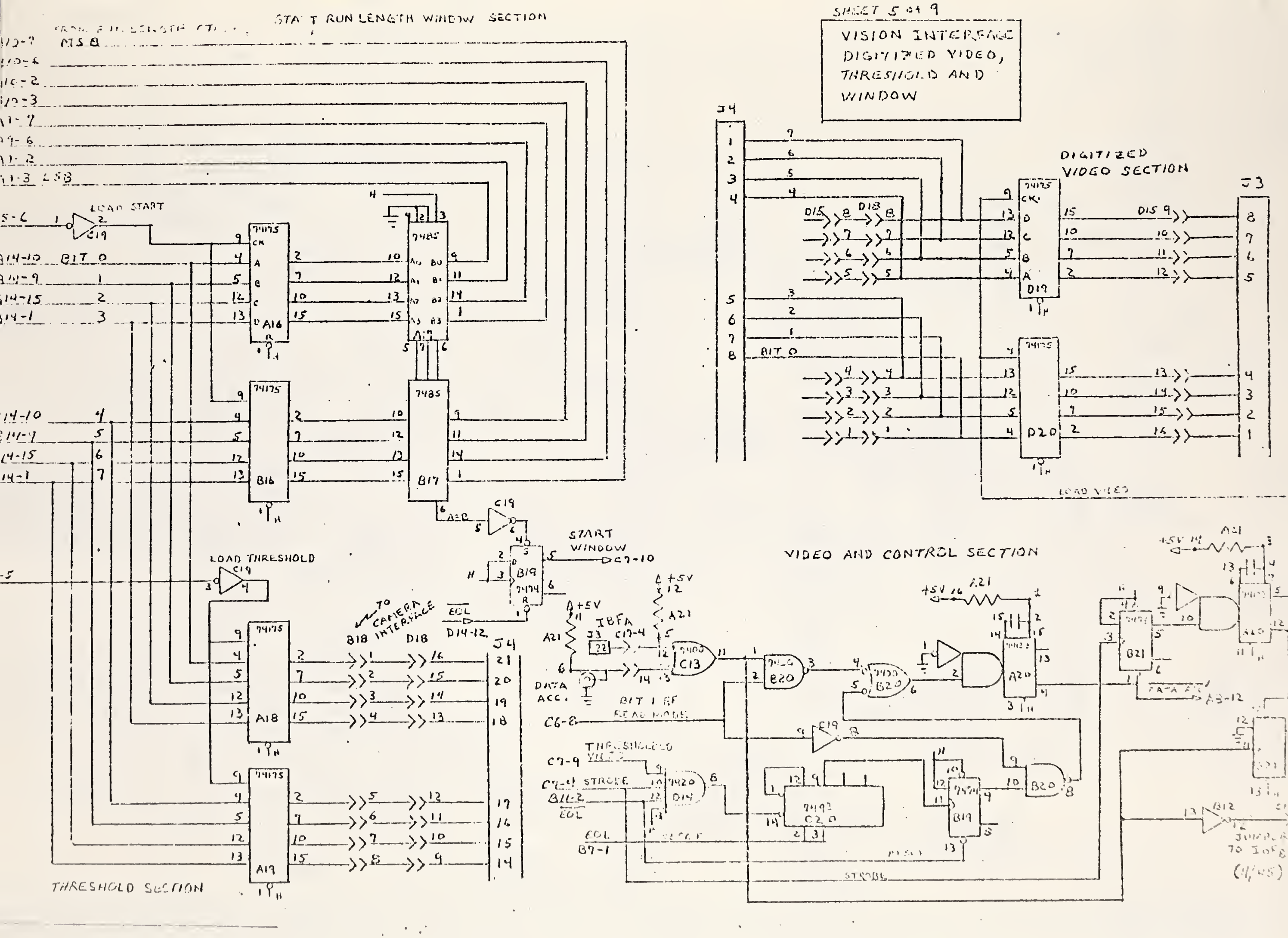
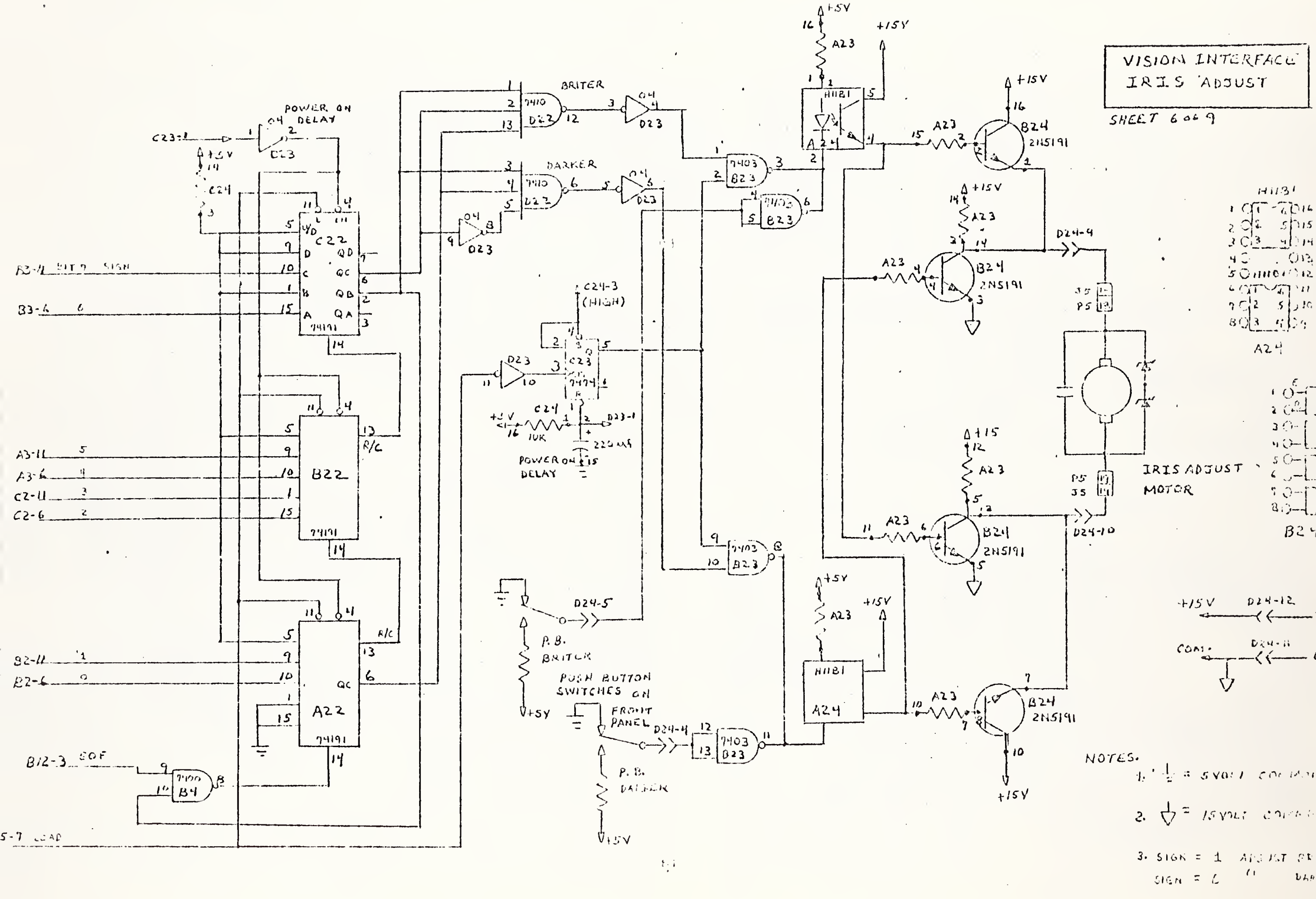
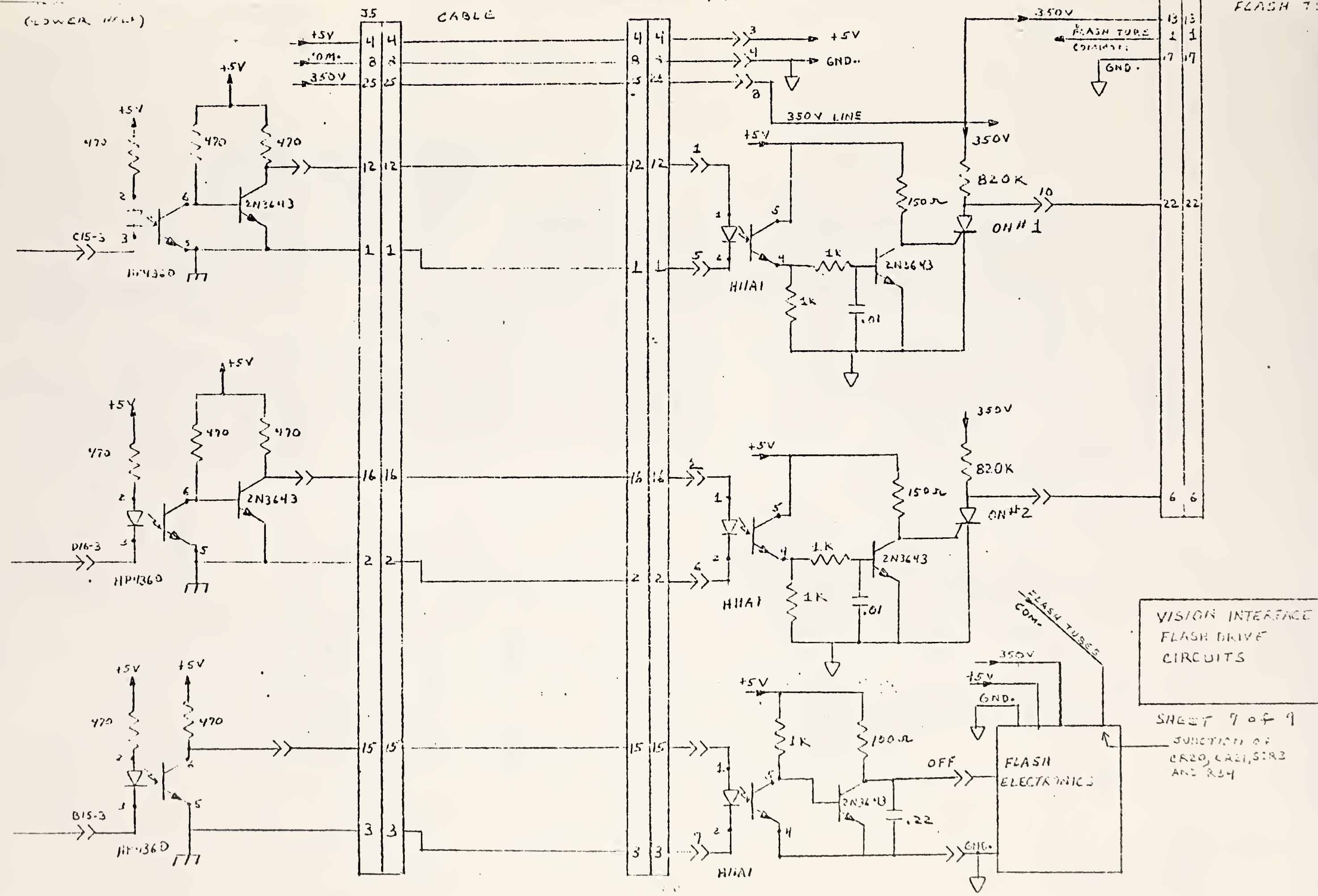

SHEET 5 OF 9
VISION INTERFACE DIGITIZED VIDEO, THRESHOLD AND WINDOW
START RUN LENGTH WINDOW SECTION
MSB
LSB
LOAD START
BIT 0
7485
74175
A16
B16
B17
A18
A19
C19
START WINDOW
LOAD THRESHOLD
TO CAMERA INTERFACE
B18
D18
J4
J3
EOL
D14-12
DIGITIZED VIDEO SECTION
D19
D20
D15
LOAD VIDEO
VIDEO AND CONTROL SECTION
+5V
A21
IBFA
C17-4
DATA ACC.
BIT 1 OF READ MODE
C6-8
C7-9
THRESHOLDED VIDEO
STROBE
B11-2
B7-1
7400
C13
B20
A20
B21
7420
D14
7492
C20
7474
B19
B12
THRESHOLD SECTION

FIGURE 18

VISION INTERFACE
IRIS ADJUST

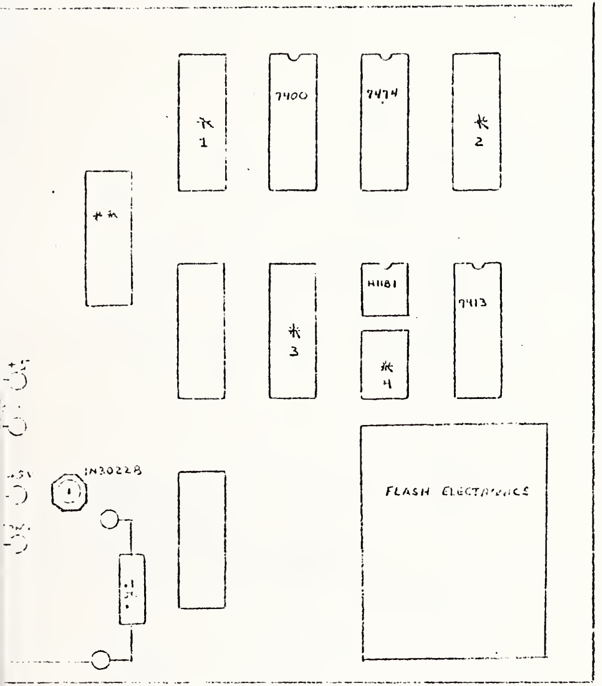
SHEET 6 of 9


POWER ON DELAY
C23-1
+5V
C24
D23
BRITER
DARKER
7410
D22
04
7403
B23
H11B1
A24
A23
+5V
+15V
B24
2N5191
D24-9
D24-10
J5
P5
IRIS ADJUST MOTOR
C22
74191
QD
QC
QB
QA
U/D
B22
R/C
A22
B3-4 BIT 7 SIGN
B3-6
A3-11
A3-6
C2-11
C2-6
B2-11
B2-6
B12-3 EOF
7400
B4
LOAD
C24-3 (HIGH)
7474
10K
220 μF
POWER ON DELAY
D23-1
D24-5
P.B. BRITER
PUSH BUTTON SWITCHES ON FRONT PANEL
D24-4
P.B. DARKER
B24
D24-12
D24-11
COM.
NOTES.
1. = 5 VOLT
2. = 15 VOLT
3. SIGN = 1
SIGN = 0


(LOWER HALF)
FLASH CHASSIS
FLASH 7
J5
CABLE
+5V
COM.
350V
GND.
350V LINE
FLASH TUBE COMMON
470
2N3643
C15-3
D16-3
B15-3
HP4360
HP1360
HP136D
H11A1
150Ω
100Ω
820K
1K
.01
.22
ON #1
ON #2
OFF
FLASH TUBES COM.
FLASH ELECTRONICS
JUNCTION OF CR20, CR21, SCR3 AND R34

VISION INTERFACE
FLASH DRIVE
CIRCUITS

SHEET 7 of 9

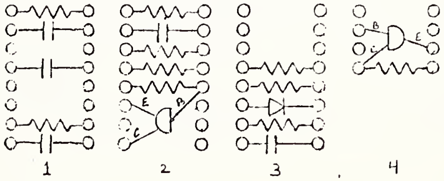


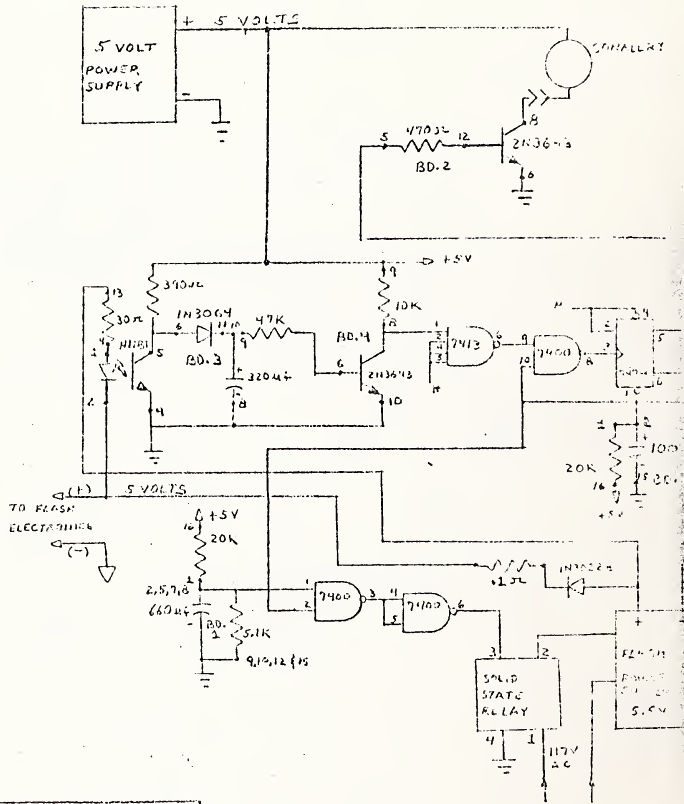
7400
7474
7413
H11B1
1
2
3
4
**
1N3022B
FLASH ELECTRONICS

COMPONENT LOCATIONS

* COMPONENT BOARDS

** INPUT SOCKET

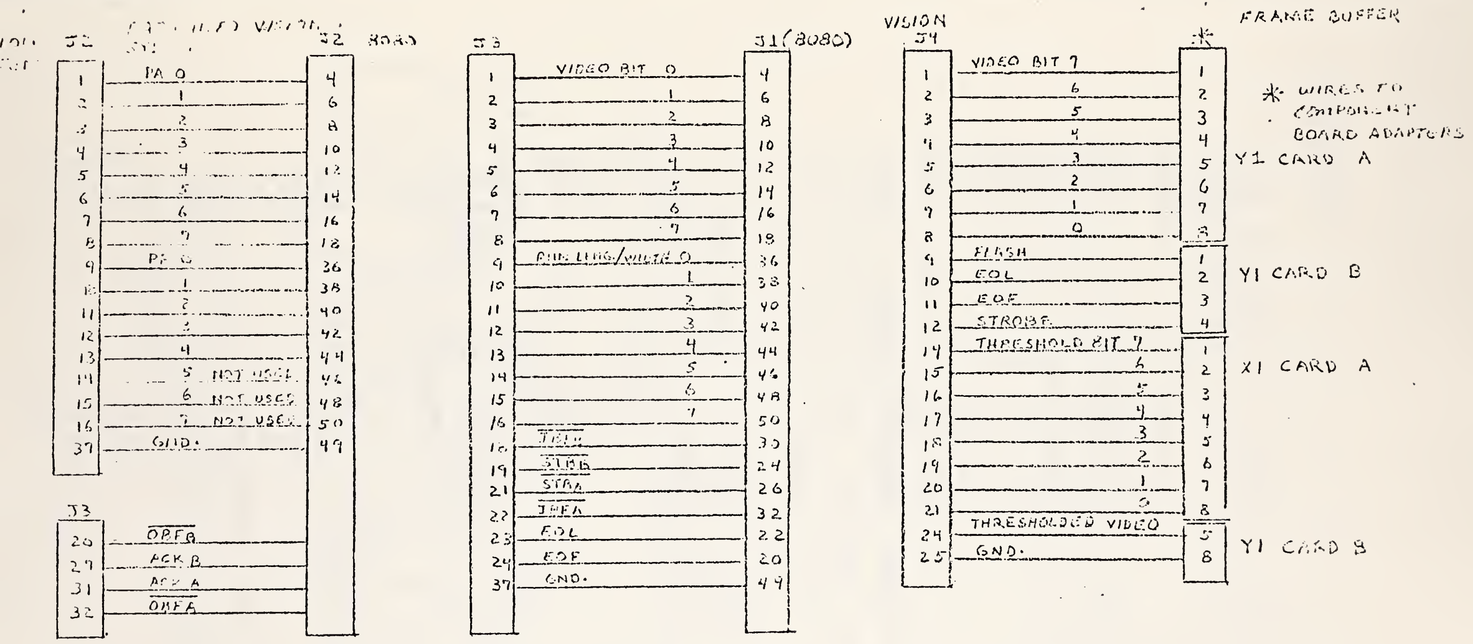




5 VOLT POWER SUPPLY
+
5 VOLTS
−
47Ω
BD.2
2N3643
390Ω
30Ω
H11B1
1N3064
BD.3
47K
320μf
10K
BD.4
2N3643
+5V
7413
7400
TO FLASH ELECTRONICS
(+)
(−)
5 VOLTS
20K
2,5,7,8
660μf
BD. 1
5.1K
9,10,12 & 15
.1Ω
1N3022B
SOLID STATE RELAY
117V AC
20K
100
+5V

VISION INTERFACE
FLASH CURRENT
MONITOR CIRCUIT

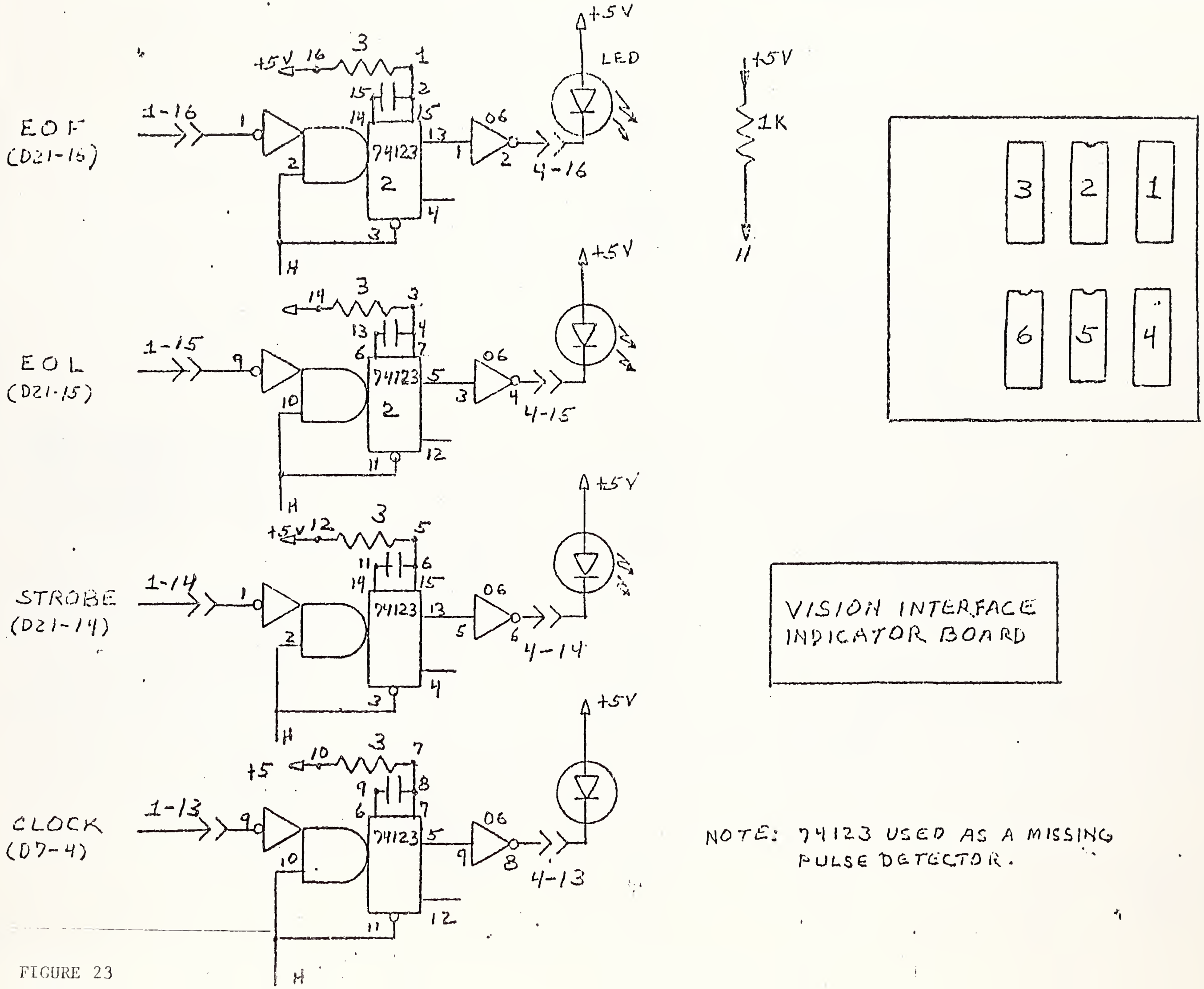
SHEET 8 of 9


J2
J2
BOARD
PA 0
1
2
3
4
5
6
7
PB 0
1
2
3
4
5 NOT USED
6 NOT USED
7 NOT USED
GND.
J3
OBFB
ACK B
ACK A
OBFA
J3
J1 (8080)
VIDEO BIT 0
RUN LENG/WIDTH 0
TBLH
STBB
STBA
IBFA
EOL
EOF
GND.
VISION
J4
VIDEO BIT 7
FLASH
EOL
EOF
STROBE
THRESHOLD BIT 7
THRESHOLDED VIDEO
GND.
FRAME BUFFER
*
* WIRES TO COMPONENT BOARD ADAPTORS
Y1 CARD A
Y1 CARD B
X1 CARD A
Y1 CARD B

VISION INTERFACE
CABLES.

SHEET 9 of 9

FIGURE 22


EOF
(D21-16)
1-16
EOL
(D21-15)
1-15
STROBE
(D21-14)
1-14
CLOCK
(D7-4)
1-13
74123
06
+5V
LED
4-16
4-15
4-14
4-13
H
1K
3
2
1
6
5
4

VISION INTERFACE
INDICATOR BOARD

NOTE: 74123 USED AS A MISSING PULSE DETECTOR.

FIGURE 23

References

1. G. J. Agin and T. O. Binford, "Computer description of curved objects." Proc. 3rd IJCAI, Stanford, 1973, pp. 629-640.

2. G. Falk, Interpretation of line data as a three-dimensional scene. Artificial Intelligence, 3 2, 1972, pp. 101-144.

3. R. N. Nagel, W. W. Braithwaite, and P. R. Kennicott, Initial Graphics Exchange Specification IGES, Version 1.0, NBSIR 80-1978(R), National Bureau of Standards, U.S. Department of Commerce, March 1980.

4. R. Nevatia and T. O. Binford, Description and recognition of curved objects. Artificial Intelligence 8 1, 1977, pp. 77-98.

5. L. G. Roberts, Machine perception of three-dimensional solids. Optical and Electro-Optical Information Processing (ed. Tippett et al.), MIT Press, Cambridge, Mass. 1965, pp159-197.

6. A. R. Rosenfeld and A Kak, Digital Picture Processing, Academic Press, New York, 1976.

7. A. R. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labelling by relaxation operations. " IEEE Trans SMC-6, 1976, pp420-433.

8. M. Shneier, "A compact relational structure representation" Proc. Workshop on the Representation of Three-Dimensional Objects, University of Pennsylvania, 1979.

9. K. Sugihara, "Dictionary-guided scene analysis based on depth information." PIPS-R-No 13, Electrotechnical Laboratory, Tokyo, 1977.

10. S. A. Underwood and C. L. Coates, "Visual learning and recognition by computer". TR123, Information Systems Research Laboratory, University of Texas, Austin, 1972.

Task 4.3.3 Inherently Safe Systems

Abstract

An ultrasonic ranging sensor and data processing system were developed and tested. Experiments were performed using this sensor system as a safety device. Whenever an intruder or unexpected obstacle is detected in the working envelope of the robot, a warning flag to the control system causes the robot to stop and wait for the intruder to leave before continuing the task. Additional types of safety sensors are also analyzed for possible future systems.

ROBOT SAFETY

As the use of robots in industrial applications becomes more widespread, the safety of both personnel and other machinery, including the robot, is an increasingly important concern. Thus, the development of techniques for ensuring safe operations is an important factor in applications of industrial robots.

The NBS work on robot safety for the Air Force ICAM Program has been in two areas: graphic simulation of robot motion and development of sensors for preventing robot collisions with personnel or other equipment. The following sections give a brief overview of this work and the major accomplishments.

Graphic Simulation

Graphic simulation of robot motion permits the off-line checking of the software without the danger of an accident because of a programming error. As the tasks robots are designed to perform become more complex and varied, the percentage of off-line versus teach-mode programming is likely to increase. As a consequence, the number of possible accidents resulting from programming errors is also likely to increase. The possibility of accidents will add emphasis to the need for graphics simulation.

The graphics simulation of robot motion developed by NBS involves three distinct steps. The first step is to create a 3-dimensional model of the robot. This model is constructed by representing the major components of the robot by appropriate geometric solids such as spheres, cylinders, and rectangular parallelepipeds (rectangular solids). Also included in this first step is the specification of all possible joint movements. The second step in the simulation is to configure the robot model in a specific set of joint positions. The third step is to generate a graphical image of the robot corresponding to the joint positions specified in Step 2. This graphical image is produced by ray-tracing techniques which are described in more detail later in this section.

This simulation technique was applied to the Stanford Arm robot used in the NBS robotics laboratory. As described above, the first step is to create a physical model of the robot. For the Stanford Arm, this is done by approximating each component of the robot as either a cylinder or rectangular solid. To describe mathematically these two basic geometric shapes, the following quantities, illustrated in Figure 4.3.3-1, are required:

Cylinder	o vertex
	o height vector
	o scalar radius

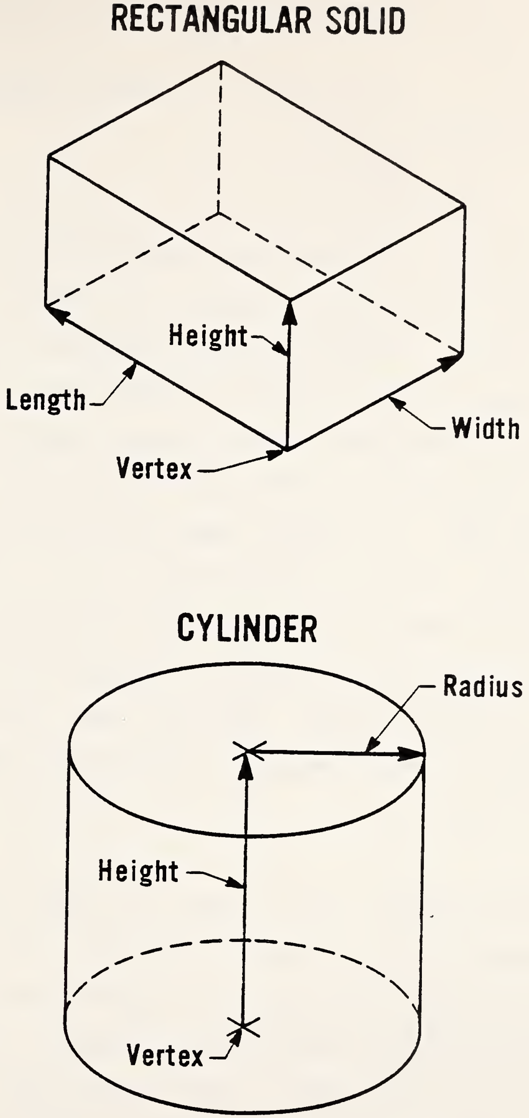

RECTANGULAR SOLID
Height
Length
Width
Vertex
CYLINDER
Radius
Height
Vertex

Figure 4.3.3-1. Basic geometric shapes used to model the robot. The cylinder requires the specification of a vertex, height vector, and a scalar radius. The rectangular parallelepiped (or rectangular solid) requires a vertex and three vectors for the length, width and height.

Rectangular Solids	o vertex o length, height and width vectors.

The model of the Stanford Arm, as seen from the side and top, is shown in Figures 4.3.3-2 and -3, respectively. This model consists of seven rectangular solids and five cylinders. The location of each of these elements in space is established by setting up a coordinate system relative to some fixed point on the robot, in this case the center of the bottom of the base (Element 1). The fixed point is given the (x, y, z) coordinates (0, 0, 0). Any other point on the robot is represented as a set of coordinates, measured in centimeters, from the (0, 0, 0) point. Table 4.3.3-1 lists the vertices, space vectors, and scalar radii, when appropriate, for each of the 12 elements of the model.

The other part of Step 1 is the specification of the possible joint movements. Starting at the fixed point (the center of the bottom of the base) and using the reference coordinate system, a new coordinate system (CS) is defined for each joint in the robot. Each new CS is defined by locating it with respect to the CS defined for the previous joint. In addition to the new CS, a motion description is also specified for each joint. Both rotating and sliding joints are defined, with their motion specified in terms of

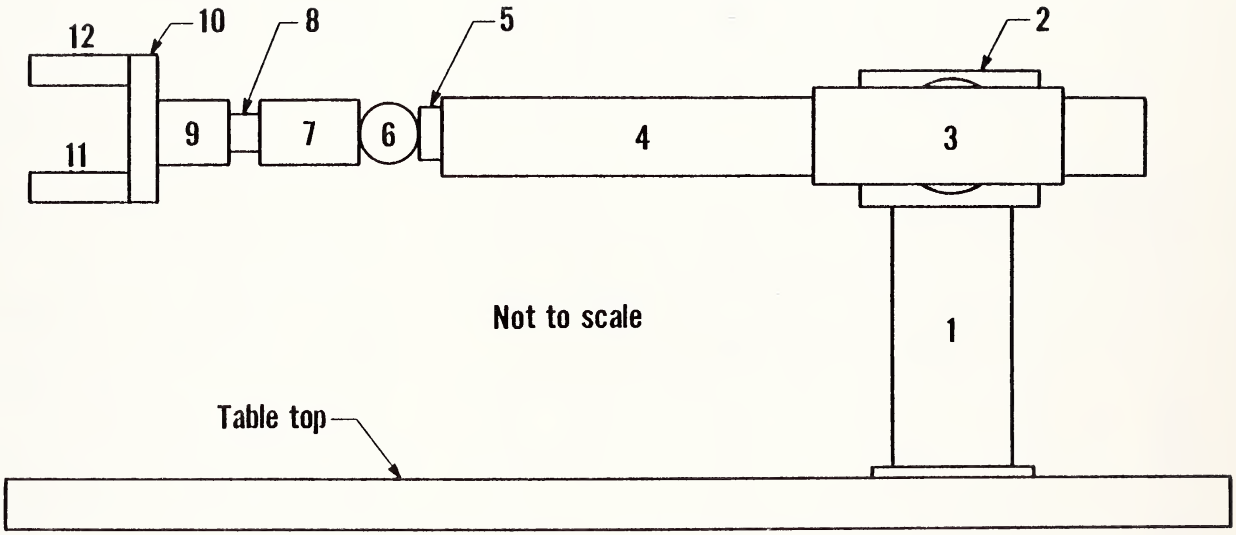

12
10
8
5
2
11
9
7
6
4
3
1
Not to scale
Table top

Figure 4.3.3-2. A side view of the Stanford Arm robot modeled with seven rectangular solids (2, 4, 7, 9, 10, 11 and 12) and five cylinders (1, 3, 5, 6 and 8).

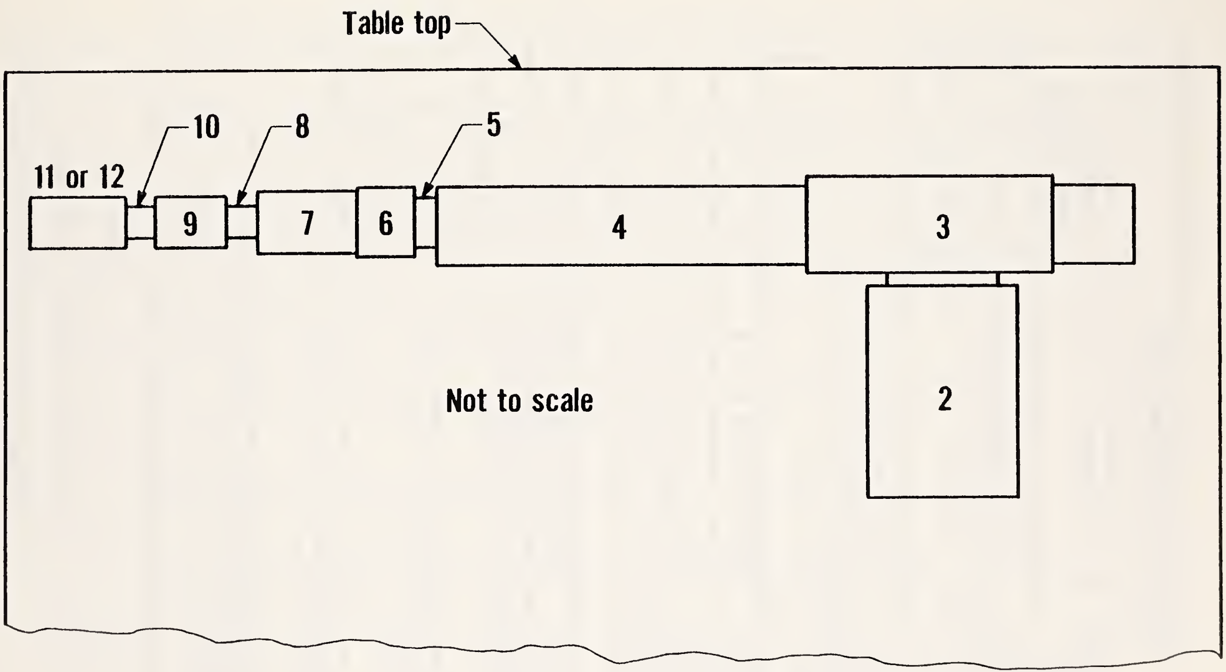


Table top
10
8
5
11 or 12
9
7
6
4
3
2
Not to scale

Figure 4.3.3-3. A top view of the Stanford Arm robot modeled with seven rectangular solids (2, 4, 7, 9, 10, 11 and 12) and five cylinders (1, 3, 5, 6 and 8). Elements 1 and 12 are hidden and therefore are not shown.

Table 4.3.3-1. Spatial coordinates for the twelve elements of the Stanford Arm robot model.

ELEMENT NUMBER	ELEMENT TYPE	VERTEX			VECTORS FROM VERTEX			SCALAR RADII
1	Cylinder	0.0	0.0	0.0	0.0	0.0	26.7	7.0
2	Rectangular Solid	7.0	-10.2	29.1	-14.0	0.0	0.0	
					0.0	-20.3	0.0	
					0.0	0.0	14.0	
3	Cylinder	0.0	-10.2	6.1	0.0	-1.3	0.0	14.0
4	Rectangular Solid	20.0	-13.2	30.6	-106.7	0.0	0.0	
					0.0	-6.4	0.0	
					0.0	0.0	6.4	
5	Cylinder	20.0	-16.4	33.8	1.6	0.0	0.0	3.2
6	Cylinder	24.4	-11.3	33.8	0.0	-10.2	0.0	3.2
7	Rectangular Solid	28.9	-11.3	33.8	-4.4	0.0	0.0	
					0.0	-10.2	0.0	
					0.0	0.0	4.4	
8	Cylinder	28.9	-16.4	33.8	2.5	0.0	0.0	1.9
9	Rectangular Solid	37.7	-12.4	35.4	-6.3	0.0	0.0	
					0.0	-8.0	0.0	
					0.0	0.0	7.0	
10	Rectangular Solid	38.3	-12.4	41.8	-0.6	0.0	0.0	
					0.0	-3.2	0.0	
					0.0	0.0	-12.7	
11	Rectangular Solid	48.5	-11.3	28.0	-10.2	0.0	0.0	
					0.0	-5.4	0.0	
					0.0	0.0	2.2	
12	Rectangular Solid	48.5	-11.3	42.9	-10.2	0.0	0.0	
					0.0	-5.4	0.0	
					0.0	0.0	-2.2	

the new CS. Thus, rotating joints always lie in a plane, while sliding joints are specified by a vector, usually in one of the three orthogonal directions of the new CS. When the joints are specified, they are assumed to be modeled in the zero position, shown in Figure 4.3.3-4. Therefore, a scheme is also required to translate joint values into rotation or translation of the model components.

The specification of the 3-dimensional model and joint definitions completes the first step of the simulation, and is performed only once per robot.

The next step is to simulate, using the robot model, the motions that the real robot would execute if it were to receive a specified set of joint values. This is done on a joint-by-joint basis starting at the hand and working back to the base. The task of updating the internal representation of the model involves the use of data extracted from the robot control system. These data consist of the coordinates of three points in 3-dimensional space and the absolute angle values of each joint at every specific instant in time. As shown in Figure 4.3.3-5, the three points used are at the interface between the boom and the wrist (labeled wrist point), at the center of the end of the fingertips (labeled tool point), and at the end of one fingertip (labeled finger point).

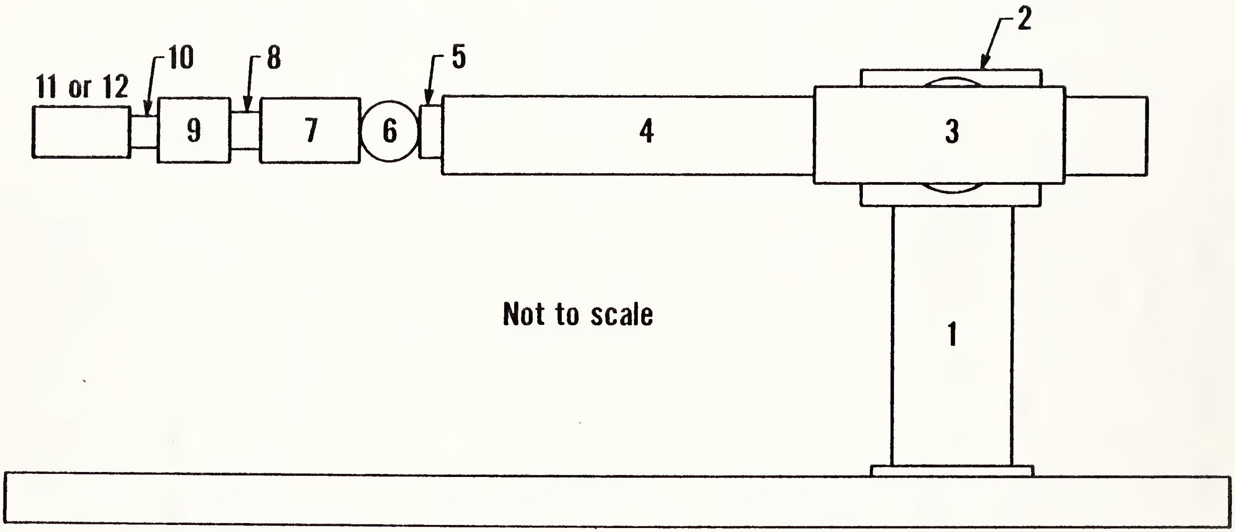

11 or 12
10
9
8
7
6
5
4
2
3
1
Not to scale

Figure 4.3.3-4. Sketch showing the Stanford Arm robot with all joints in the zero position. Element 4 is at its maximum extension.

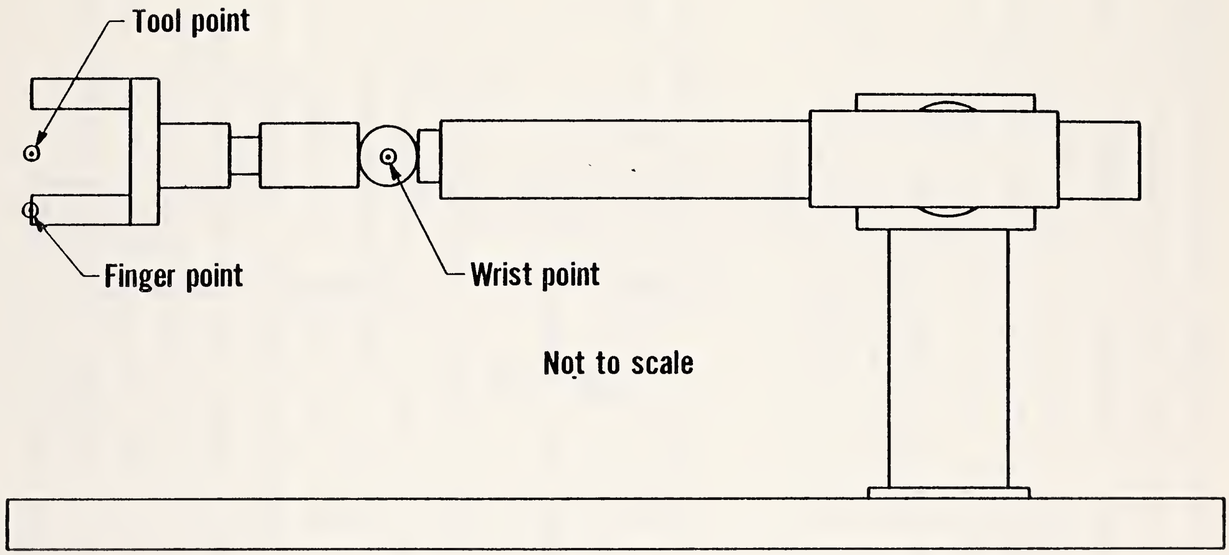

Tool point
Finger point
Wrist point
Not to scale

Figure 4.3.3-5. A side view of the Stanford Arm robot with the wrist, finger and tool points labeled.

These three points were chosen because by knowing them and the joint angles, one can determine any point on the robot. Although the robot is modeled by a number of separate geometric shapes, calculating their movements can be simplified using the relationships between the individual elements. In Figure 4.3.3-6, three vectors, originating at the three points, can be used to define almost the entire robot. Vector A defines the configuration of Elements 2, 3, 4 and 5, Vector B the Elements 6, 7, 8, 9, 10 and 11, and Vector C the rotation of the fingertips.

Using this representation, the program updates the 3-dimensional model joint-by-joint until the zero position has been appropriately changed for each joint. At that point, the second step is complete.

Step 3 is to create a graphical representation of the robot model as updated in Step 2. This graphical representation consists of a sequence of dots which corresponds to the visible outline of the robot. The graphical image is generated by ray-tracing techniques, which assume that the robot is being viewed through a fictitious camera. The rays are traced from the camera through a 128 x 128 point viewing plane and then into the space occupied by the robot as specified by the model. This hypothetical arrangement is shown in Figure 4.3.3-7.

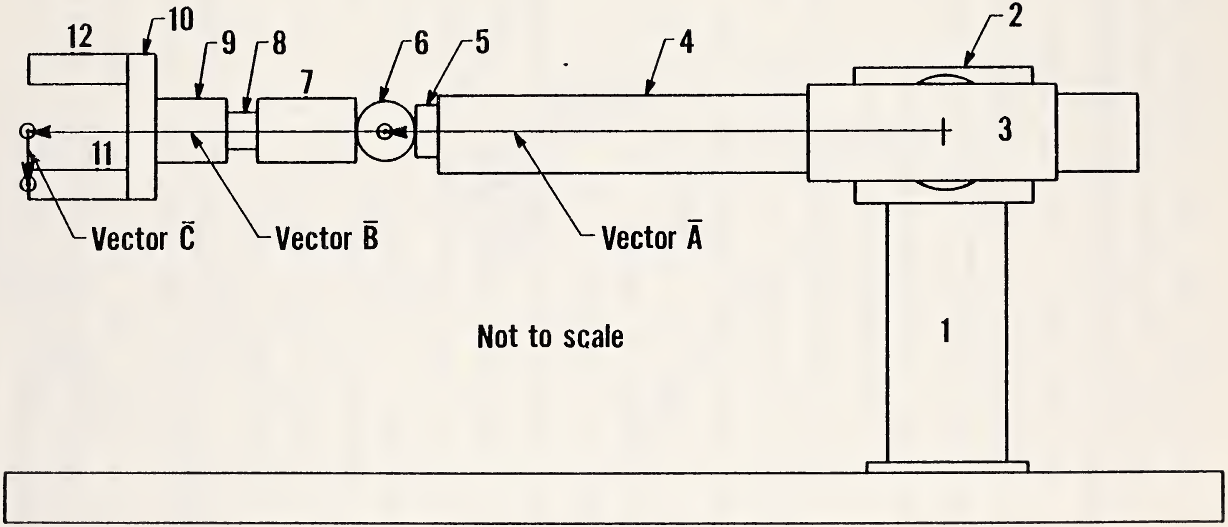

12
10
9
8
7
6
5
4
2
3
11
1
Vector $\bar{C}$
Vector $\bar{B}$
Vector $\bar{A}$
Not to scale

Figure 4.3.3-6. Illustration of the three vectors used to define the configuration of the Stanford Arm robot model.

Prior to the simulation, the user chooses the camera location by specifying the azimuth and elevation for the viewing plane and selects a scale factor for the image size of the robot. Following this, a ray is traced for each point in the viewing plane (a total of 16,384) starting in the lower right-hand corner. If the ray being traced intersects the model of the robot, the element and surface number of the first intersection is computed and stored. By treating only the first intersection, the process omits all hidden lines. For each pair of adjacent rays (both horizontal and vertical), a comparison is made. If both rays intersect the same part of the model, no dot appears on the viewing plane. However, if the rays intersect different parts of the model, the upper or left ray (depending upon whether the vertical or horizontal pair is being compared) generates a dot on the viewing plane. This technique results in images where only the edges of each element, represented by a series of dots, are shown. Repeating the process for all rays produces a rectangular dot matrix which forms the image of the robot as seen from the camera. The dot matrix is then sent to a graphic device producing a picture of the robot corresponding to the set of joint positions specified for the simulation. Figure 4.3.3-8 shows the results of a simulation of the Stanford Arm robot picking up a block on a table top. This particular graphic image is for a viewing plane azimuth of 45 degrees and elevation of 45 degrees.

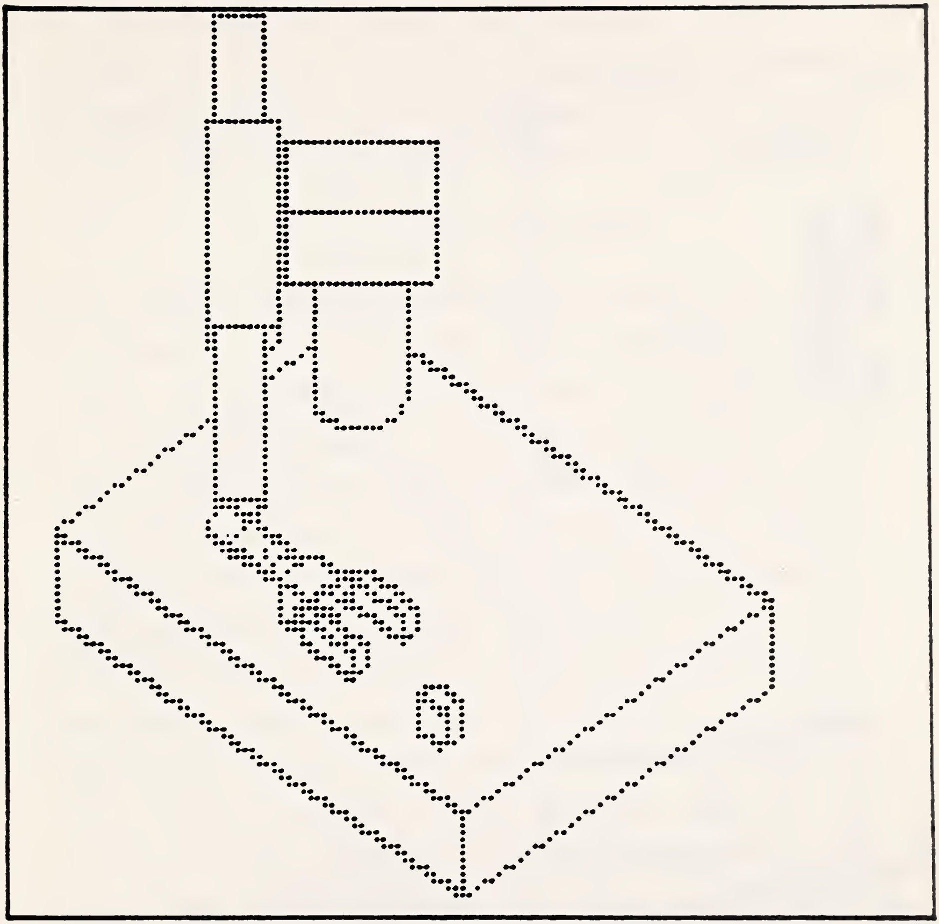
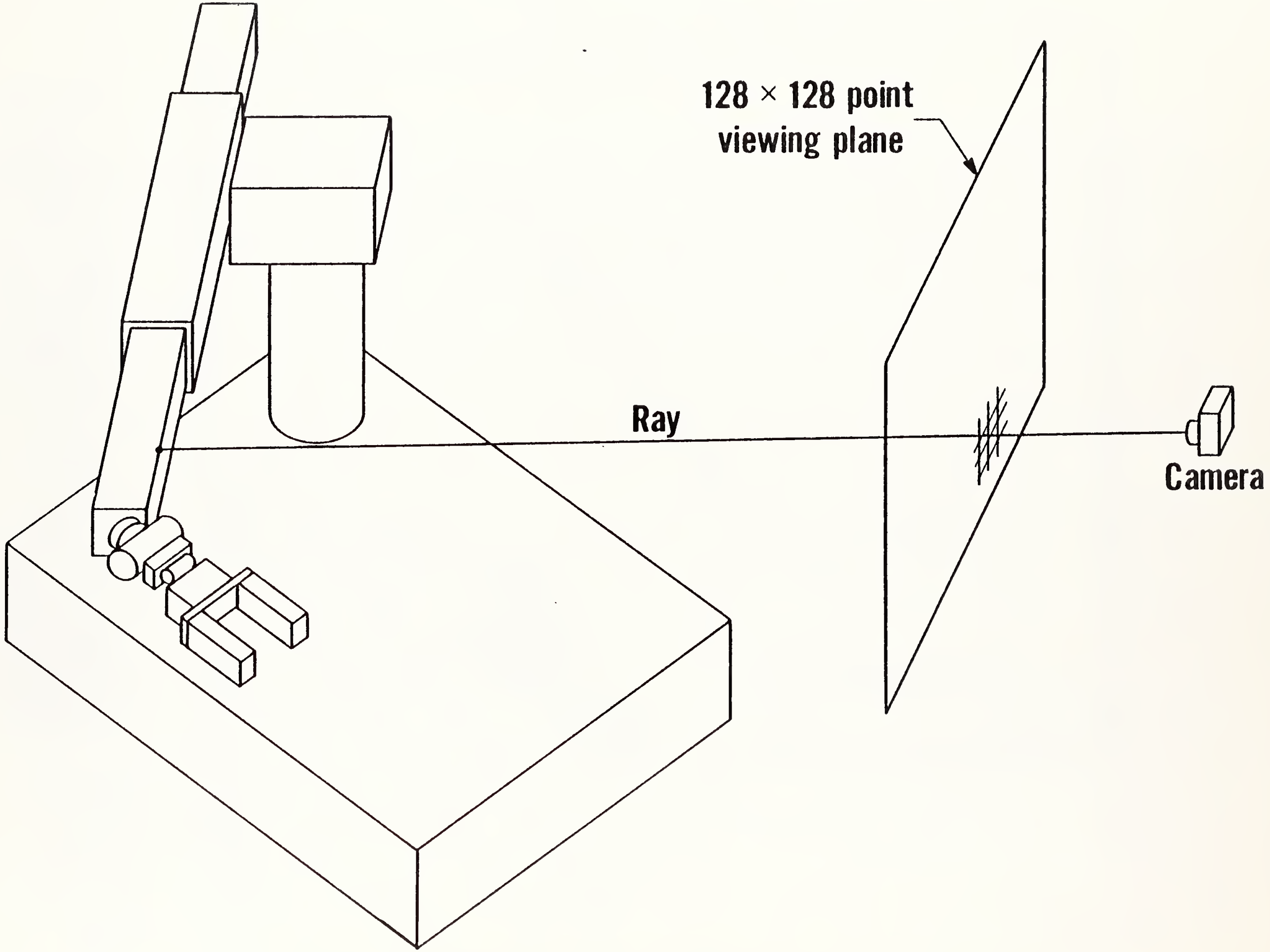


Figure 4.3.3-8. Example of the graphical simulation of the Stanford Arm robot picking up a block on a table top. The viewing plane is positioned with an azimuth of 45 degrees and an elevation of 45 degrees.



128 × 128 point
viewing plane
Ray
Camera

Figure 4.3.3-7. Conceptual view of the ray-tracing technique.

One note of caution regarding this work is that it represents an example of what can be done with simulation and was limited to an open loop program with no sensory feedback or error correction procedures. Further development would be required to include the capability to simulate sensor data and error conditions in the robot model.

This graphics simulation technique has proven to be useful for examining the software for gross programing errors. However, one drawback is the time required to generate each of these graphic images. Since this process takes on the order of 15 minutes for each image, time constraints permit only a limited number of joint positions to be checked. Although this is adequate for the intended purpose, a technique which would permit real-time, or at least near-real-time, simulation of the robot motion would be desirable. Such a technique would allow the graphic display of the simulation to be more realistic because it would run at or near the normal operating speed of the robot, and would provide the simulation user with a much faster, more efficient technique of examining the software and any changes that might be required. NBS is in the process of writing the specifications for the procurement of a graphics system with the capability of supporting such a simulation. This system will be used for several in-house, robotics-related projects, including some longer-term work on robot simulation.

Safety Sensor Systems

Graphic simulation is a viable technique for minimizing potential accidents as long as the environment around the robot does not change. However, in industrial applications this is seldom the case. Since personnel or other equipment might enter the robot workstation, interactive techniques for preventing collisions with the robot are necessary.

At present, the most commonly used technique for providing safety near industrial robots is to erect permanent barriers around the workstation, such as safety rails, fences or safety chains, in order to restrict personnel from entering these areas while the robot is operating. Although easy to implement, this approach is inadequate because no safety protection is provided for operators training the robot in teach mode or for personnel who are required to work nearby while the robot is operational. Also, the flexibility of the manufacturing facility is severely limited by erecting permanent barriers. One approach to solving this problem is to provide sensor systems which can detect intruders -- personnel and other objects -- that enter the robot workstation, and signal the robot control system so that an appropriate control action is made.

A variety of safety strategies can be developed regarding the type of intrusion and the desired response of the robot control system. There are cases when other equipment and personnel must be in the workstation while the robot is operational, e.g., during a routine maintenance check or during the operation of a robot in the teach mode. These situations obviously are to be treated differently from cases where someone who should not be there enters the workstation or some piece of equipment or hardware is left in the robot's working volume. To handle such a variety of possibilities, several distinct categories of sensor systems can be envisioned. For the NBS work on robot safety, these sensor systems have been broken-down into three levels based on the region of coverage and the associated safety strategy.

Level I systems provide perimeter penetration detection around the robot workstation. These systems provide an indication of an intruder crossing the workstation boundary, but they do not necessarily provide any information regarding the location of the intruder within the workstation. The simplest safety strategy would be to halt all operations as soon as an intruder crosses the boundary. Halting all operations, however, would severely restrict the flexibility of the workstation in much the same way a fence would. Another approach would be to use the Level I system to alert personnel that they are entering a robot workstation and that they should exercise extreme caution, or to provide a preliminary signal to the robot control system to activate or check the status of other safety sensors.

Level II systems provide detection in the region between the workstation perimeter and some point on or just inside the working volume of the robot. The actual boundaries of this region depend upon the workstation layout and the safety strategy being employed for a particular robot design and mode of operation. In some cases, it may be permissible for personnel to be inside the workstation and perhaps even inside a portion of the accessible working volume of the robot while the robot is operating. In others, it may be necessary to slow down or halt all robot movements as soon as an intruder gets within a specified distance of the robot.

These two possible strategies illustrate that there are a variety of approaches that can be taken in designing the Level II system, particularly in terms of the areas or zones of detection and the resulting robot control action.

In the design of the safety sensor system, the general approach for sensing intruders and the overall safety strategy are obviously interrelated and are two of the key design factors. Some of the general approaches for handling intruder detection are:

- Detection in a limited number of zones where the probability of a collision occurring is the highest,
- Detection at any location within a specified area around the robot (exact location of the intruder may or may not be known),
- Intruder tracking through the workstation.

Similarly, the safety strategies can be grouped as follows:

- Complete shutdown of the robot as soon as an intruder is detected (either an application of the brakes, if so equipped, or a software stop),
- Limitation of the speed of the robot when an intruder is detected and activation of appropriate warning alarms,

O Instruction of the robot to perform other tasks in another zone until the intruder leaves,

O Instruction of the robot to take an alternate path to avoid a collision -- obstacle avoidance.

These alternate approaches illustrate that there are many ways of designing the Level II safety system. Although the type of transducer and sensing system can be generalized, the final design of the safety system and robot control scheme will have to be specialized for different robot/workstation layouts and types of robot operations.

Level III systems provide detection within the robot working volume. This type of system, sometimes referred to as a "safety skin," is required for cases where personnel must work close to the robot, such as during teach-mode operations. In such cases, the robot must be operational even though someone is within the working volume. The Level III system must be capable of sensing and avoiding an imminent collision between the robot and the operator in the event of some unexpected movement. Because the distance between the robot and the operator is much less in this case, the response time of the Level III safety system must be much shorter than for the Level I or II systems. These smaller separation distances also impose a requirement for finer distance resolving capabilities in the Level III system.

The concept of three levels of sensor systems does not require that the three systems operate exclusive of one another. In fact, some overlap of the regions of detection coverage is desirable, since this could be used to provide additional checks of intruders. These additional checks would help to limit the number of false detections. However, for cases such as teach-mode operations where the Level III system is the primary system, the Level II system would have to be in a standby mode. These and other constraints would be factored into the total safety sensor system design, would be tailored to the particular robot application under consideration.

Types of Detection Sensors

There are a variety of sensing techniques currently used to detect intruders. The majority of the applications of these techniques have been used to provide security for commercial businesses, military bases, and, more recently, nuclear power generating stations. Security sensor systems can be categorized in much the same way as the Level I, II, and III breakdown of safety sensor systems described in the previous section. Although not standardized, the three general types of security intrusion detection systems are:

(1) point, spot, or object; (2) perimeter or penetration; and (3) area, space, or volumetric. As the name implies, point systems are used to detect the presence of an intruder at only a single location. Perimeter systems are used to detect penetration across a specified boundary by an intruder. Area or space systems are used to detect intruders anywhere within a selected region defined by the field of operation of the particular sensor being employed. The system can be designed so that the selected region includes the entire volume of some enclosed space such as a room.

There are some obvious differences between the design criteria for security and safety sensor systems because the intended functional operation of these systems is not the same. In general, security systems are not required to provide information about the instantaneous location of an intruder, only that a particular point, boundary, or space has been penetrated by an intruder. For efficient operation of the robot, the safety sensor system must provide additional information about the intruder's location in order to develop a system which will minimize the number of unnecessary shutdowns. Thus, not all security sensor systems are applicable to the design of robot safety systems.

As mentioned above, there are a variety of security intrusion detection sensing techniques currently in use. Table 4.3.3-2, reproduced from NBS Special Publication 480-14, Selection and Application Guide to Commercial Intrusion Alarm Systems, provides a comparison of these different techniques in terms of types of applications, relative advantages and disadvantages, resistance to defeat, and false alarm susceptibility. Considering the design of a robot safety sensor system in terms of the requirements of Level I, II, and III systems shows that many of the sensors listed in this table are not applicable. For example, switch type devices might be used for Level I perimeter detection; but once triggered, they would have to be reset to be functional again. Another example pertains to the motion detection type devices. Although these devices perform well for security purposes, they could not be used in a robot workstation, because the motion of the robot would trigger the sensor. Other limitations of some of these sensors are: susceptibility to environmental effects, such as temperature changes, extraneous noise or vibration, and dust or smoke; and to background signals from other sources. Thus, only a limited number of commercially available security intrusion sensors are applicable to robot safety system design. Of the sensors listed in Table 4.3.3-2, only the pressure-sensitive mats and the photoelectric sensors were

Table 4.3.3-2. Comparison of various types of security intrusion detection sensors.+

SENSOR	DRY CONTACT MECHANICAL SWITCHES	MAGNETIC SWITCHES	MERCURY SWITCHES	METALLIC FOIL	WIRE SCREENS	TRIP WIRES	PRESSURE MATS PRESSURE RIBBONS PRESSURE WAFERS
APPLICATIONS	DOORS, WINDOWS GATES, TRANSOMS, HATCHES, ETC., USUALLY FOR PERIMETER PROTECTION.	DOORS, WINDOWS, GATES, TRANSOMS, HATCHES, ETC., USUALLY FOR PERIMETER PROTECTION	(SAME COMMENTS AS FOR MAGNETIC SWITCHES APPLY. APPLICATION IS USUALLY FOR ACCESS POINTS THAT HAVE COVERS THAT OPEN WITH CHANGING VERTICAL ANGLE. THUS THESE SWITCHES OPERATE WHEN TILTED BEYOND A CERTAIN POINT.)	SHOW WINDOWS, OFFICE WINDOWS, GLASS DOORS, DRY WALL BOARD ETC., USUALLY FOR PERIMETER PROTECTION	ACCESS POINTS NOT SUBJECT TO EVERYDAY USE USUALLY FOR PERIMETER PROTECTION.	ENTRY WAYS SUCH AS TO CORRIDORS OR IN DOORWAYS FOR PERIMETER PROTECTION	SMALL AREAS, DOORWAYS, OR UNDER SPECIFIC OBJECTS FOR POINT PROTECTION
ADVANTAGES	LOW COST.	RELATIVELY RESISTANT TO ENVIRONMENTAL EFFECTS. RELATIVELY IMMUNE TO EFFECTS OF WEAR. LOW COST		EASILY REPAIRED. VISIBILITY SERVES AS DETERRENT	LOW DEGREE OF MAINTENANCE LOW VISIBILTIY FOR ATTRACTIVE APPEARANCE	LOW COST	LOW COST LOW DEGREE OF MAINTENANCE ADAPTABLE TO WIDE VARIETY OF SHAPES AND SIZES.
DISADVANTAGES	LOW RELIABILITY LOW SENSITIVITY SUBJECT TO ENVIRONMENTAL EFFECTS HIGH INSTALLATION COST	BECAUSE OF MOUNTING POSITION MAY BE SUBJECT TO DAMAGE IN SOME APPLICATIONS. HIGH INSTALLATION COST.		VULNERABLE TO DAMAGE BOTH INTENTIONAL AND THROUGH DAY TO DAY USE.	MUST BE REPLACED AFTER PENETRATION TO RESTORE PROTECTION	MUST BE REMOVED TO ALLOW NORMAL ACCESS, THEN REPLACED TO RESTORE PROTECTION.	SUBJECT TO WEAR IF IN PATH OF HEAVY TRAFFIC SUBJECT TO EFFECTS OF HUMIDITY AND STANDING WATER
RESISTANCE TO DEFEAT	LOW	BALANCED TYPE MORE RESISTANT TO COMPROMISE THAN SINGLE MAGNET TYPES.		LOW	MODERATE FOR CONCEALED TYPES.	LOW IF DETECTED BY INTRUDER.	RELATIVELY HIGH ONLY IF CONCEALED OR PRESENCE UNKNOWN TO INTRUDER
FALSE ALARM SUSCEPTIBILITY	HIGH IF DOOR OR WINDOW HAS LARGE AMOUNT OF PLAY LOW IF TIGHT	HIGH IF DOOR OR WINDOW HAS LARGE AMOUNT OF PLAY LOW IF TIGHT		HIGH DUE TO EFFECTS OF ENVIRONMENT.	LOW TO MODERATE.	LOW IF BUILDING IS SOLID	SUBJECT TO ENVIRONMENTAL CONDITIONS.

+ Reprinted from NBS Special Publication 480-14, "Selection and Application Guide to Commerical Intrusion Alarm Systems," published by the NBS Law Enforcement Standards Laboratory.

Table 4.3.3-2. Continued.

ACOUSTIC SENSORS	ULTRASONIC MOTION SENSORS	MICROWAVE MOTION SENSORS	INFRARED MOTION SENSORS	PHOTOELECTRIC (ACTIVE) SENSORS	PHOTOELECTRIC (PASSIVE) SENSORS	CAPACITANCE SENSORS	VIBRATION SENSORS
AREA PROTECTION OF ENCLOSED SPACES (ROOMS, VAULTS, ETC.).	AREA PROTECTION OF SMALL ENCLOSED SPACES (ROOMS, CORRIDORS, ETC.)	AREA PROTECTION OF ENCLOSED SPACES (ROOMS, CORRIDORS, ETC.) CAN COVER LARGE AREAS.	AREA PROTECTION OF ENCLOSED SPACES (ROOMS CORRIDORS ETC) CAN COVER LARGE AREAS	ACROSS DOORWAYS CORRIDORS, ETC., FOR PERIMETER PROTECTION MULTIPLE BEAM SYSTEMS FOR LIMITED AREA PROTECTION	POINT PROTECTION USING SENSORS WITH HIGH DIRECTION SENSITIVITY LIMITED AREA PROTECTION OF SMALL ROOMS OR PORTIONS OF LARGER ONES.	PRIMARILY POINT PROTECTION FOR SAFES FILING CABINETS, VALUABLE OBJECTS LIMITED AREA AND PERIMETER PROTECTION	PRIMARILY POINT PROTECTION FOR VAULTS SHOW CASES ETC. LIMITED SPACE PROTECTION WHEN INSTALLED TO PROTECT WALLS OR CEILINGS ETC
SENSITIVE. CAN USE EXISTING INTERCOM SYSTEMS. NOT AFFECTED BY AIR MOVEMENT EFFECTIVE AGAINST "STAY-BEHINDS."	USUALLY NOT DETECTABLE BY INTRUDER. EFFECTIVE AGAINST "STAY-BEHINDS." EASY PHYSICAL INSTALLATION	NOT DETECTABLE BY INTRUDER. EFFECTIVE AGAINST "STAY-BEHINDS." NOT AFFECTED BY AIR MOTION, NOISE, LIGHT OR SOUND.	RELATIVELY IMMUNE TO NOISE AND VIBRATION.	HIGH DEGREE OF FLEXIBILITY IN APPLICATION INFRARED BEAM DIFFICULT TO DETECT CAN COVER ACCESS POINTS WHERE PHYSICAL OBSTRUCTION NOT DESIRED OR CANNOT BE TOLERATED	RELATIVELY UNAFFECTED BY ENVIRONMENTAL FACTORS (EXCEPT ABRUPT CHANGE IN LIGHT LEVEL) HIGH DEGREE OF FLEXIBILITY IN APPLICATION	HIGH DEGREE OF FLEXIBILITY IN APPLICATION PROTECTIVE FIELD NOT DETECTABLE BY INTRUDER	REQUIRE LOW MAINTENANCE HIGH DEGREE OF RELIABILITY WHEN PROPERLY APPLIED
MUST BE USED IN STABLE NOISE ENVIRONMENT WHERE BACKGROUND LEVEL IS LOW	SEVERELY AFFECTED BY ENVIRONMENTAL FACTORS. AIR TURBULENCE AND MOTION RATTLING DOORS, JANGLING KEYS, BLOWING DRAPERIES, VIBRATIONS, LOUD NOISES, ETC. ESSENTIALLY LINE-OF-SIGHT OPERATION; LARGE OBJECTS COULD SHIELD INTRUDER. MAY NOT DETECT EXTREME RATES OF MOVEMENT (VERY SLOW OR VERY FAST).	COVERAGE DIFFICULT TO CONFINE TO DESIRED AREA CAN BE SET OFF BY NEARBY FLUORESCENT LIGHTS, LARGE OBJECTS OUTSIDE PROTECTED AREA, RADIO TRANSMITTER OPERATING NEAR SENSOR FREQUENCY	SOME SYSTEMS SENSITIVE TO CHANGES IN THERMAL ENVIRONMENT (e.g. CHANGES IN SUNLIGHT AND TEMPERATURE).	NARROW BEAM OF PROTECTION. LINE OF SIGHT OPERATION SMOKE OR DUST CAN HAMPER OPERATION SUBJECT TO MISALIGNMENT PROBLEMS.	NARROW BEAM OF PROTECTION. LINE-OF-SIGHT OPERATION SMOKE OR DUST CAN HAMPER OPERATION	CAN BE APPLIED ONLY TO OBJECTS NOT ELECTRICALLY GROUNDED MAY REQUIRE SPECIAL CONSTRUCTION	DETECTS ONLY FORCEFUL ATTEMPTS AT ENTRY CANNOT BE USED IN AREAS OF HIGH VIBRATION (TRAFFIC, CONSTRUCTION, ETC.)
HIGH IF PROPERLY INSTALLED.	HIGH IF PROPERLY INSTALLED.	HIGH IF PROPERLY INSTALLED	HIGH	LOW TO MODERATE WITH SYSTEMS USING MODULATED BEAMS HAVING HIGHEST RESISTANCE.	HIGH	VERY HIGH	VERY HIGH.
CAN BE HIGH BUT REDUCED USING ADDITIONAL CANCELLATION MICROPHONE.	CAN BE HIGH UNLESS ENVIRONMENTAL FACTORS ARE CONSIDERED BEFORE APPLICATION	CAN BE HIGH UNLESS PROPERLY PLACED AND CAREFULLY ADJUSTED.	HIGH FOR RECEIVE ONLY SENSORS LOW FOR TRANSMIT-RECEIVE SENSORS	CAN BE HIGH IF CERTAIN ENVIRONMENT FACTORS ARE PRESENT (SMOKE, DUST) OR POOR PLANNING RESULTS IN MISAPPLICATION.	CAN BE HIGH IF IMPROPERLY INSTALLED SO THAT HEAT OR LIGHT LEVELS ARE NOT CONSTANT IF COVERS FLOOR, RODENTS MAY SET OFF ALARM	LOW IF PROPERLY INSTALLED.	CAN BE HIGH IF ENVIRONMENTAL FACTORS ARE NOT TAKEN INTO ACCOUNT MAY BE TRIGGERED BY MINOR EARTH TREMORS SONIC BOOMS OR TRAINS.

considered suitable for use in developing a robot safety sensor system. In addition to the sensors listed in this table, various types of ultrasonic and infrared sensors (other than motion detection devices) were also evaluated for the safety system design.

Description of the Prototype Safety System

The prototype safety system was designed for the Stanford Arm robot, set up in the NBS robotics laboratory as shown in Figure 4.3.3-9. A general floor plan of the laboratory in which the robot is located is shown in Figure 4.3.3-10. In this figure, the dotted or shaded area corresponds to the region that the robot can reach with the arm at maximum extension. The cross-checked area represents the region considered to be the robot workstation for this project. The outside edge of this area defines the boundaries as far as design of the safety sensor system is concerned. The safety strategy employed in this initial system will permit personnel to be in the cross-checked area, but not in the shaded area (i.e., the working volume of the robot) while the robot is operating. When an intruder enters the cross-checked area, a warning alert (visual and/or audio) is broadcast. When an intruder is detected entering the working volume, the robot undergoes a software stop at its current position. The robot remains in this position until the intruder leaves the working volume. Once the intruder

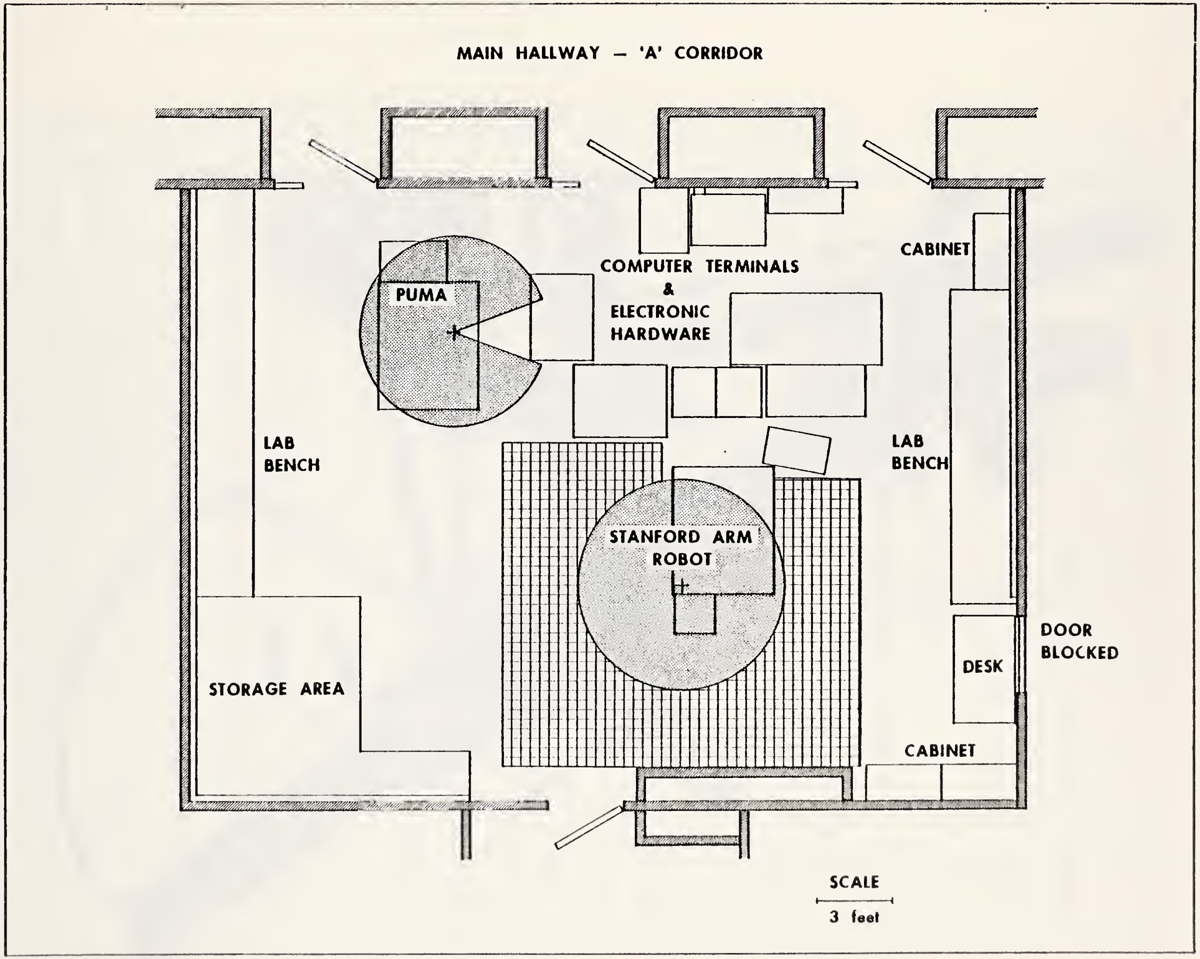

MAIN HALLWAY — 'A' CORRIDOR
PUMA
COMPUTER TERMINALS & ELECTRONIC HARDWARE
CABINET
LAB BENCH
LAB BENCH
STANFORD ARM ROBOT
STORAGE AREA
DOOR BLOCKED
DESK
CABINET
SCALE
3 feet

Figure 4.3.3-10. Floor plan of the laboratory in which the Stanford Arm robot is located.

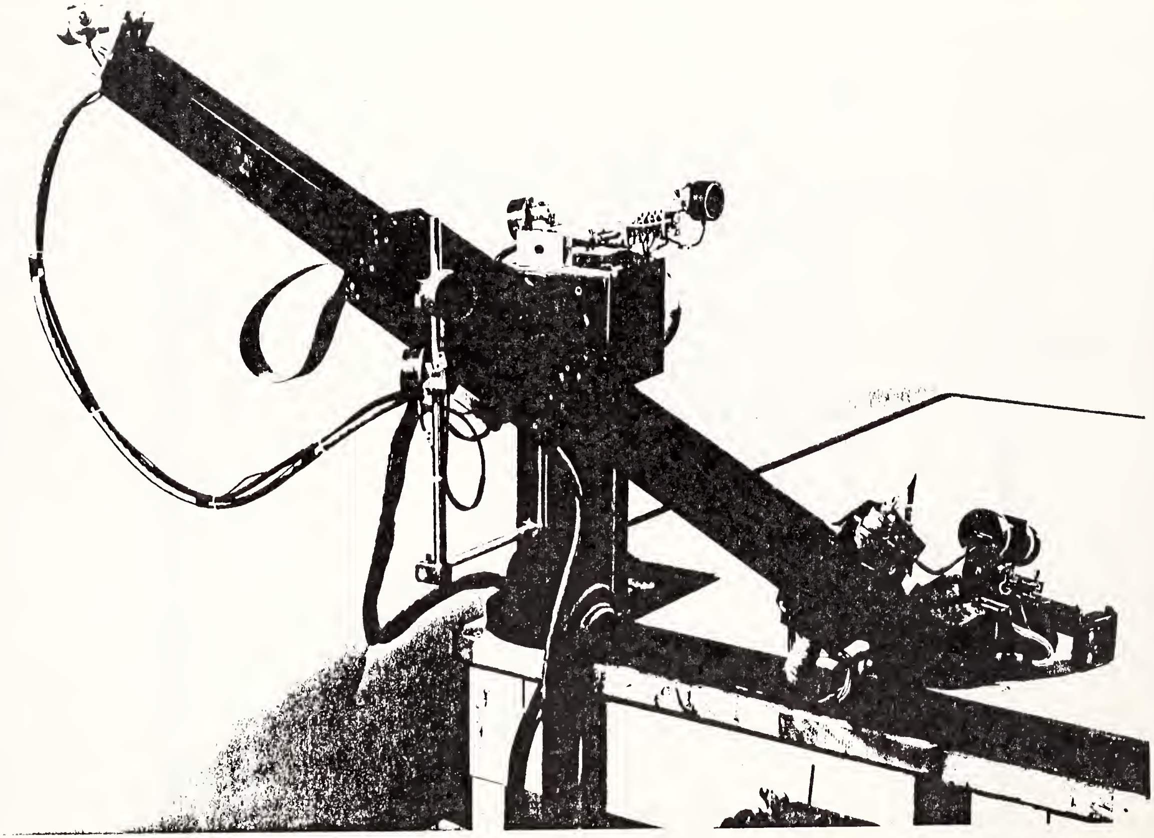


Figure 4.3.3-9. View of the Stanford Arm robot as mounted on a table in the NBS robotics laboratory.

has left this area, the robot resumes its programmed task from the point at which it was stopped. For cases where personnel must be within the working volume, a hand-operated emergency stop switch is used to halt the robot.

The prototype safety system consists of both a Level I and a Level II sensor system. In addition, an emergency stop capability exists for use in situations where personnel conducting research must be within the working volume while the robot is operating. In such cases, the safety sensor systems are put on standby and a large, easily accessible, hand-operated emergency stop switch is used to halt the robot. Hitting the emergency stop button causes the electromagnetic brakes on each joint electric motor to be applied. Since this is not a software stop, the control software must be reinitialized before the robot can continue operation.

The Level I system is composed of a set of pressure-sensitive, industrial-grade floor mats positioned in the cross-checked area around the robot. When contact is made with one of these mats (a foot pressure of 30 pounds or approximately 5 lb/sq.in. is required to activate the mat), an electrical circuit is completed which is used to turn on a warning light. Future plans include the development of a system to broadcast warning announcements to the intruder. These announcements will be generated using an electronic speech synthesis system which will give a great deal of flexibility in changing the announcement for different operational conditions in the workstation. Pressure sensitive mats are used because they do not create any obstacles to isolate the workstation from the surrounding areas. Other types of sensors, such as beam-break photoelectric detectors, would have to be mounted on a stand or pole so that they would be at a reasonable height above the floor to detect an intruder. These mounting stands would act as a hindrance to movement around the perimeter of the workstation and might, in themselves, pose a safety problem because of personnel inadvertently bumping into them. Also, since these types of sensors only detect the penetration of the workstation perimeter, they would have to be used in at least pairs at each boundary to be able to determine whether the intruder is entering or leaving the workstation. This illustrates another advantage of the mats in that the detection signal will remain on as long as the intruder is standing on the mats within the workstation.

In addition to the cross-checked area, another mat with a circular perimeter is located on the floor inside the working volume (i.e., the shaded area), except for the space occupied by the table and equipment rack, as shown by the rectangular outlines in Figure 4.3.3-10. The perimeter of this mat corresponds to the perimeter of the region reachable by the robot arm. When an intruder steps on this mat, the resulting signal is used to stop the robot (a software stop), since the intruder is within the robot working volume. In future applications, this inner mat could be used for purposes other than merely shutting down the robot, such as reducing the robot operating speed and broadcasting an audio warning instructing the intruder to leave the area or the robot will be shut down and further alarms will be sounded. Figure 4.3.3-11 shows the Stanford Arm robot mounted on a table, the circular mat in the area inside the reach of the robot arm, and a portion of the set of mats outside this inner circle in the region corresponding to the cross-checked area. Notice that for this outer area, there are a number of mats -- 11 total. For this initial safety system, these mats are all wired together to give a single output signal. It is possible in future applications to look at the output of each mat to get an indication of an intruder's general location within this area. This information could be checked against the indications from other safety sensor systems to substantiate the location of an intruder.

The Level II system used in this prototype design consists of an array of five ultrasonic echo-ranging sensors. These

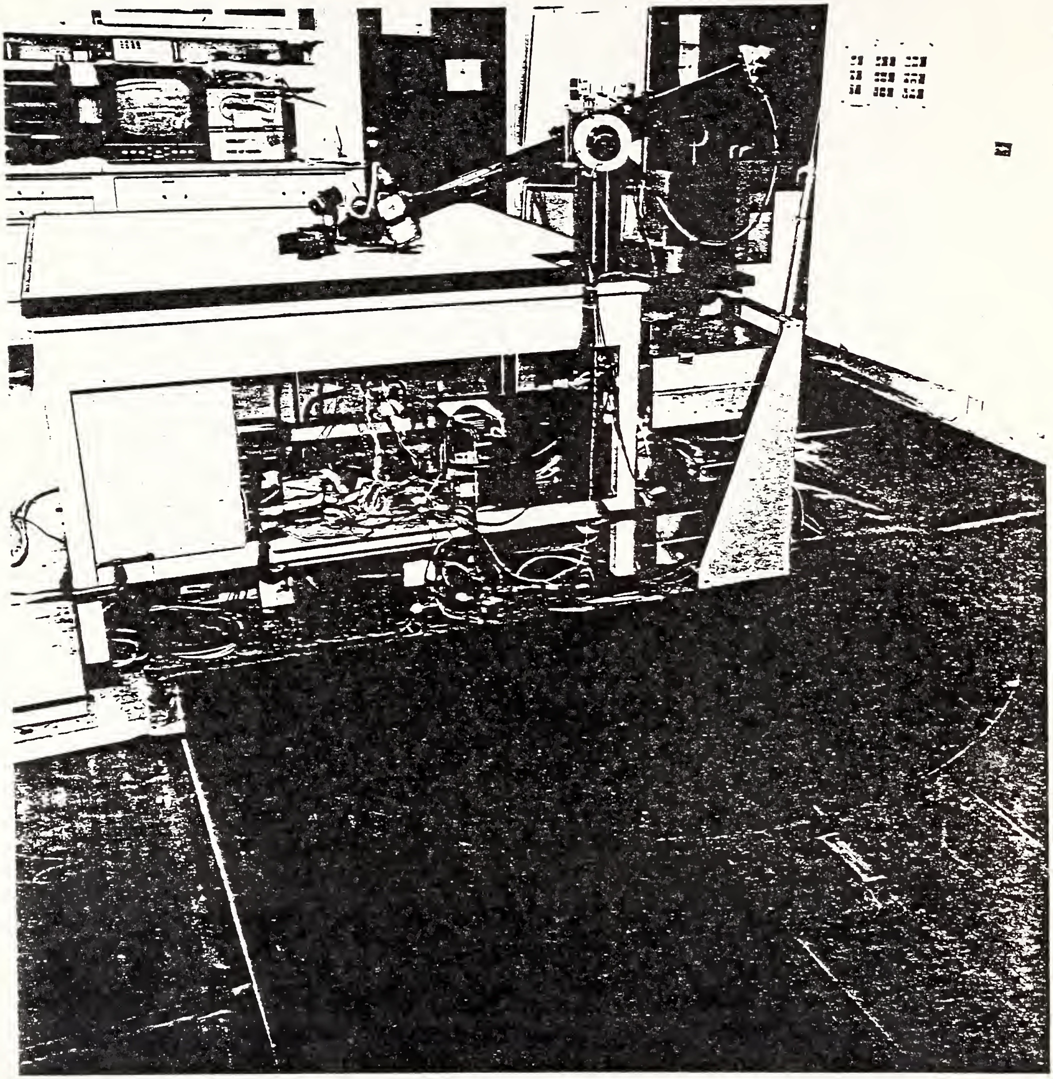


Figure 4.3.3-11. View of the Stanford Arm robot showing the inner circular mat and a portion of the set of mats covering the general workstation area.

sensors consist of an electrostatic transceiver (i.e., transmitter and receiver) and the support electronics for determining the separation distance between the transceiver and some target. The basic operation of the sensor involves: (1) transmission of an ultrasonic pulse from the electrostatic transducer; (2) reception of any reflected signals using the transducer as the receiver; (3) measurement of the time-of-flight of the ultrasonic pulse from the transducer to a target and back to the transducer; and (4) computation of the separation distance between the transducer and the target, based on the time-of-flight measurement. In this sensor, the time-of-flight is obtained by simply starting an internal clock when the ultrasonic pulse is transmitted and stopping it when an echo is received. Before describing the operation and characteristics of these sensors in detail, the following paragraph briefly outlines how they are used in the Level II safety system.

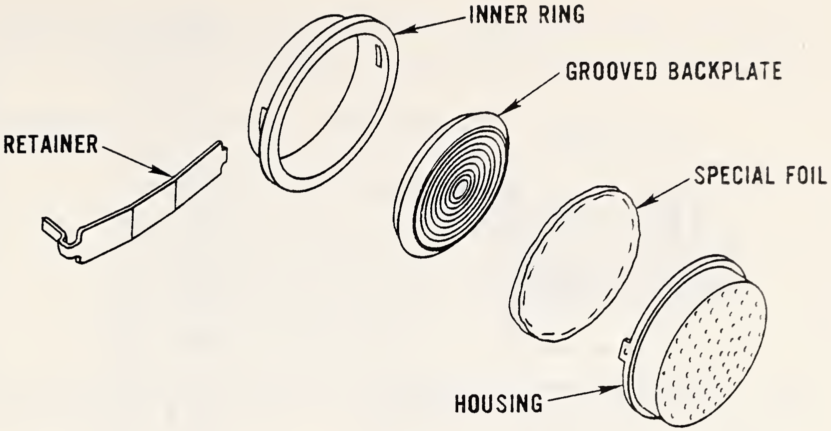
In the prototype safety system, the ultrasonic echo-ranging sensors are used to determine whether an intruder gets closer than some predetermined minimum distance from the robot. If an intruder is within this distance, a signal is sent to the robot controller to halt the robot. To determine if there is an intruder present, a comparison is made between the measured transducer-to-target separation

distance for each transducer (actually the time-of-flight is used for comparison) and a previously determined value, which corresponds to a point outside the reach of the robot arm. The reach of the robot arm when it is at full extension plus some margin of safety, which will permit the robot to be stopped before a collision can occur, determine this preset minimum distance. Three points to note about the application of the ultrasonic echo-ranging transducers in this prototype system are: (1) any signal corresponding to an echo from any permanent structure or an intruder located outside the preset minimum distance is disregarded, (2) the actual location of the intruder, i.e., the exact distance from the robot, is not used, and (3) the coverage areas can be easily selected or modified by simply changing the preset minimum distance for each transducer.

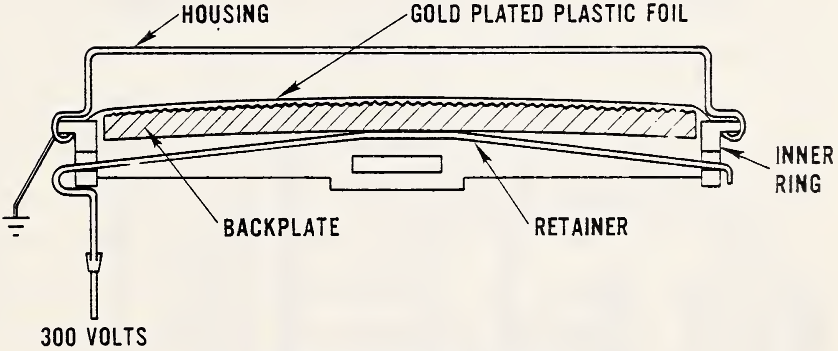
The ultrasonic echo-ranging sensors are similar to those used in a popular brand of automatic focusing camera. The sensors used in this prototype system were obtained commercially as part of an ultrasonic echo-ranging designers kit. These kits contain two electrostatic transducers, an ultrasonic circuit board (UCB) which generates the ultrasonic pulse and processes the received echo, and an experimental demonstration board which contains the clock and other electronics necessary to display the transducer-to-target separation distance in feet (resolution of tenths of feet). For applications where the experimental demonstration board is not required, it is possible to purchase the transducers and UCB's separately at a considerable savings in cost.

The physical construction of the transducer is shown in Figure 4.3.3-12. The two primary components of the transducer are a special plastic foil, which has a conductive gold coating on the front side, and an aluminum backplate, which has a series of concentric grooves and is placed against the plastic foil. These are mounted in a housing which has a perforated front and are held in an inner ring by a steel spring retainer. This retainer is used to make electrical connection with the backplate and to hold the foil under constant tension. The backplate and foil represent an electrical capacitor that, when charged, exerts an electrostatic force on the foil. Applying an AC voltage of a given frequency forces the foil to move at the same frequency and to radiate sound waves. The perforated front cover of the housing mechanically protects the foil and causes only a small loss of radiated signal strength.

In order to determine the acoustic characteristics of these sensors, a series of measurements were made in the NBS anechoic chamber. These measurements were made using the experimental setup and equipment shown in Figure 4.3.3-13

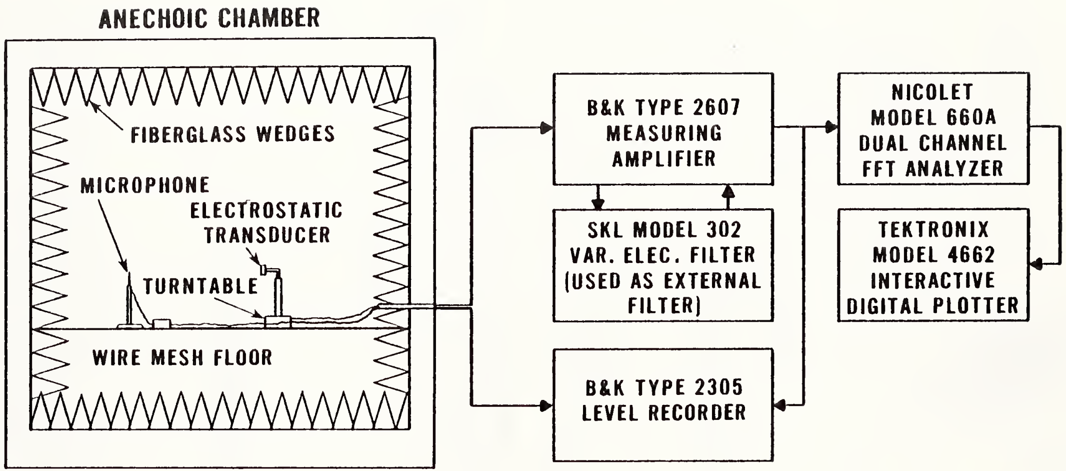

INNER RING
GROOVED BACKPLATE
RETAINER
SPECIAL FOIL
HOUSING

a) CONSTRUCTION DETAILS


HOUSING
GOLD PLATED PLASTIC FOIL
INNER RING
BACKPLATE
RETAINER
300 VOLTS

b) CROSS-SECTIONAL VIEW

Figure 4.3.3-12. Construction details and cross-sectional view of the electrostatic transducer. These drawings were reproduced, with the permission of Dr. C. Biber and the Audio Engineering Society, from: Biber, C., et al., The Polaroid Ultrasonic Ranging System, AES Paper No. 1696(A-8), Presented at the 67th Convention of the Audio Engineering Society, October 31 - November 3, 1980, New York, NY.


ANECHOIC CHAMBER
FIBERGLASS WEDGES
MICROPHONE
ELECTROSTATIC TRANSDUCER
TURNTABLE
WIRE MESH FLOOR
B&K TYPE 2607 MEASURING AMPLIFIER
SKL MODEL 302 VAR. ELEC. FILTER (USED AS EXTERNAL FILTER)
B&K TYPE 2305 LEVEL RECORDER
NICOLET MODEL 660A DUAL CHANNEL FFT ANALYZER
TEKTRONIX MODEL 4662 INTERACTIVE DIGITAL PLOTTER

LIST OF EQUIPMENT USED IN THE ANECHOIC CHAMBER

B&K Type 4138, 1/8 inch condenser microphone
B&K Type UA0160, 1/4 to 1/8 inch adaptor
B&K Type 2618, 1/4 inch preamplifier
B&K Type 2801, microphone power supply
B&K Type 3921, turntable
B&K Type 4220, pistonphone
B&K Type 4136, 1/4 inch condenser microphone

Note: These instruments are identified by brand name in order to adequately describe the tests conducted in the program. In no case does such identification imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that these products were necessarily the best available for the purpose.

Figure 4.3.3-13. Instrumentation setup for the acoustic characterization measurements of the electrostatic transducer (see Table 4.3.3-3 for instrumentation settings).

with the instrumentation settings listed in Table 4.3.3-3. The following measurements were conducted for each of the transducers used in the prototype safety system:

1) Sound level time history
2) Sound level at 1.0 meter
3) Directivity of the radiated sound waves
4) Narrow-band spectra.

Representative samples of these data are now presented and discussed.

The sensor is designed to radiate an ultrasonic pulse or chirp consisting of 56 cycles at four discrete frequencies: 8 cycles at 60 kHz, 8 cycles at 57 kHz, 16 cycles at 53 kHz, and 24 cycles at a nominal frequency of 50 kHz. This combination of frequency components gives a pulse duration of approximately 1.2 milliseconds. A typical sound level time history of a single ultrasonic pulse is shown in Figure 4.3.3-14. As expected, it is characterized by four distinct sections corresponding to the four frequency components. The sound level of this radiated pulse, as measured at 3 feet with the 1/8 inch condenser microphone at grazing incidence, ranged from 93 to 96 dB (corrected to free-field response) for the set of five transducers used in the prototype system. These sound levels were recorded from the meter on the measuring amplifier, shown in Figure 4.3.3-13, with the settings listed in Table 4.3.3-3 with

Table 4.3.3-3. Normal instrumentation settings for the equipment used to make the acoustic characterization measurements (see Figure 4.3.3-13).

INSTRUMENT	NORMAL SETTINGS
B&K Type 2607	rms, 0.1 s averaging time, external filter, linear output, ac output to Nicolet Model 660A, dc output to B&K Type 2305.
SKL Model 302	1000 Hz high-pass filter.
B&K Type 2305	50 dB logarithmic potentiometer, 50 dB potentiometer range, dc response, 20 Hz lower limiting frequency, 16 mm/s writing speed.
Nicolet Model 660A	800 line spectrum, transient-peak averaging mode, 100 kHz frequency range.

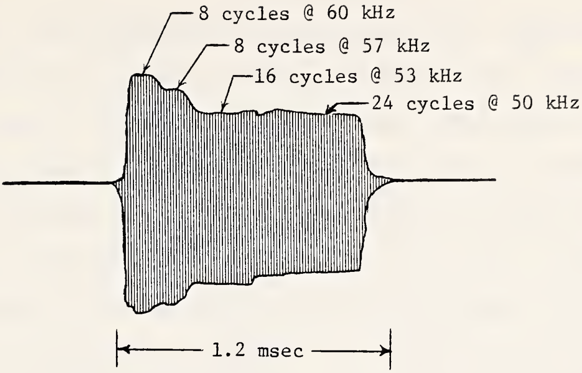


Figure 4.3.3-14. Time history envelope for a single ultrasonic pulse radiated from the electrostatic transducer.

the exception that a meter averaging time of 1.0 second rather than 0.1 second was used.

To determine the directional characteristics of these transducers, the sensors were mounted on a turntable and the sound level was recorded while the sensor was rotated 360 degrees. The sound level was recorded using the level recorder shown in Figure 4.3.3-13 with the settings listed in Table 4.3.3-3. Figure 4.3.3-15 shows a typical directivity plot for one of the electrostatic transducers. As shown by this plot, these transducers are very directional. The 10 dB down points are located at approximately ± 10 degrees off the centerline of the main forward lobe (± 9 degrees for this particular transducer). The two primary sidelobes are located at approximately ± 16 degrees, with magnitudes between 13 and 15 dB less than the main forward lobe. These directional characteristics are advantageous because with such a narrow forward beam and small sidelobes it is possible to locate intruders more accurately and also to specify the areas of detection coverage best suited for a particular application. The disadvantage is that it takes several sensors to provide adequate detection coverage around the entire robot.

The last set of characterization measurements conducted was narrow band spectral analysis of the radiated pulse. An

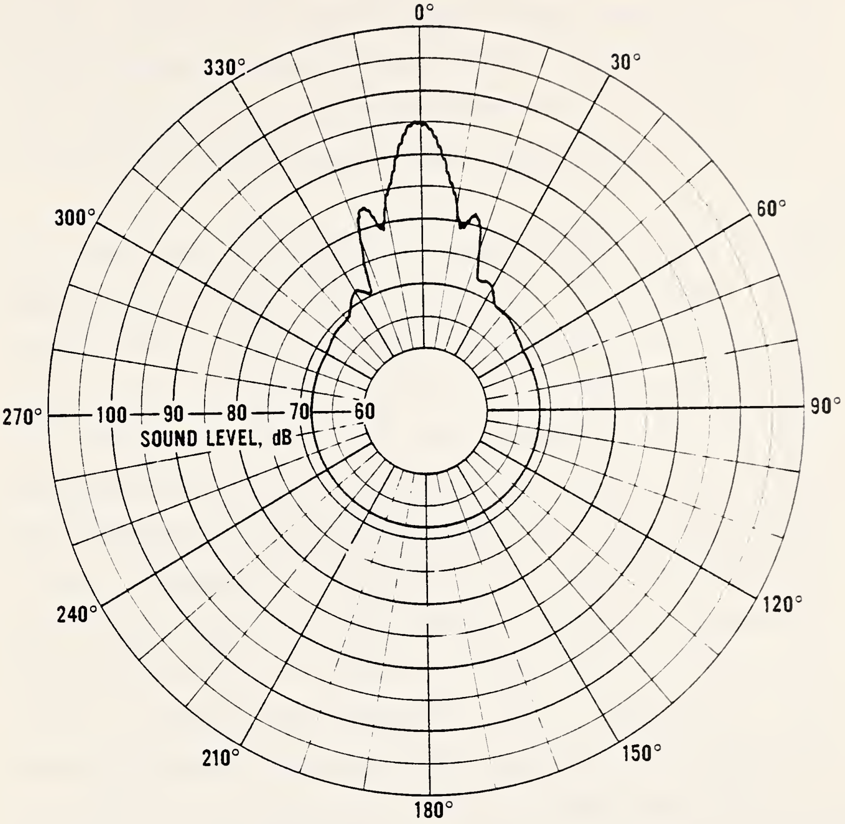


Figure 4.3.3-15. Plot of the directional characteristics of the radiated ultrasonic sound waves from the electrostatic transducer.

example of the results of these measurements is shown in Figure 4.3.3-16. This plot, typical of all the transducers tested, shows three major peaks located nominally at the design frequencies of 50, 53 and 57 kHz. The magnitudes of these spectral peaks, relative to the value at 50 kHz, are -2.2 and -4.4 dB at 53 and 57 kHz, respectively.

From this plot it is not evident that there is any component at 60 kHz. It is speculated that there is a problem with filter sidelobe interference in the FFT analyzer that effectively cancels the component at 60 kHz. The presence of a 60 kHz component was verified experimentally by using a time gating process to isolate each component of the radiated pulse, i.e., the first eight cycles, supposedly corresponding to 60 kHz, the second eight cycles (cycles nine through sixteen), supposedly corresponding to 57 kHz, etc. Figure 4.3.3-17 shows the results of these measurements for each component of the radiated pulse. These data show that there is a component at 60 kHz as originally specified for these transducers.

The receiving characteristics of the transducer were examined by testing the appropriate outputs of the ultrasonic circuit board (UCB). The sensor is designed to have an operating range of 0.9 to 35 feet. The limits of this range are a function of the timing requirements of the

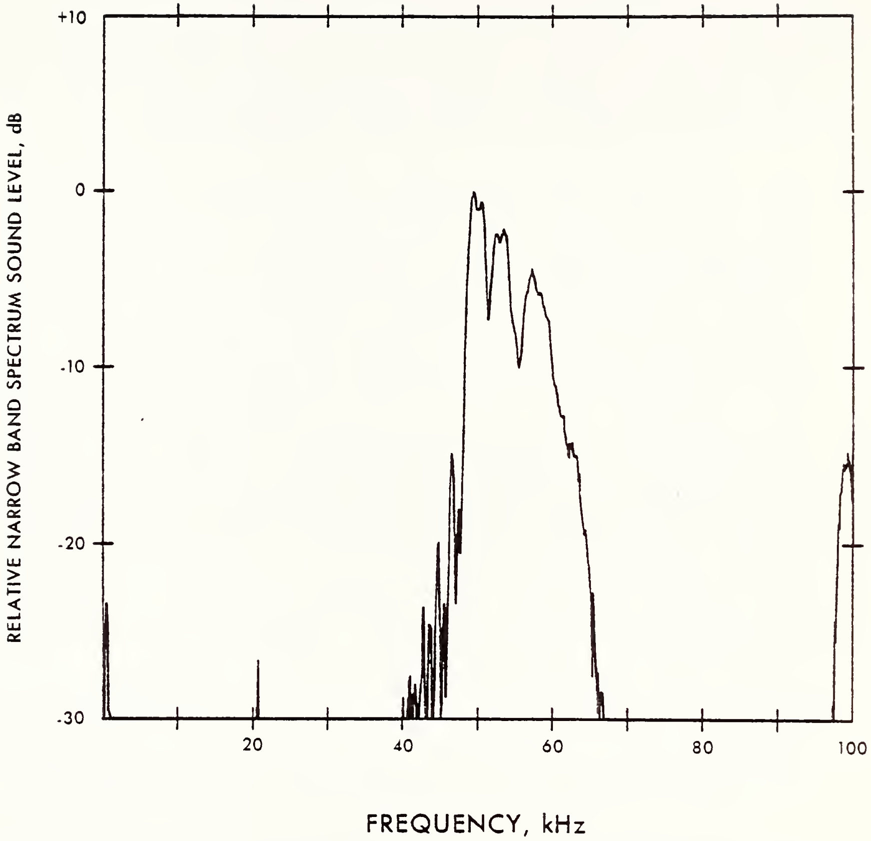
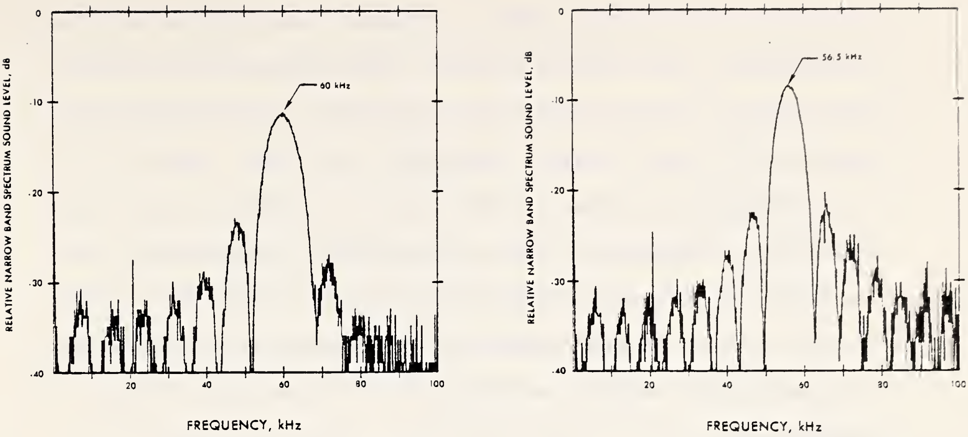
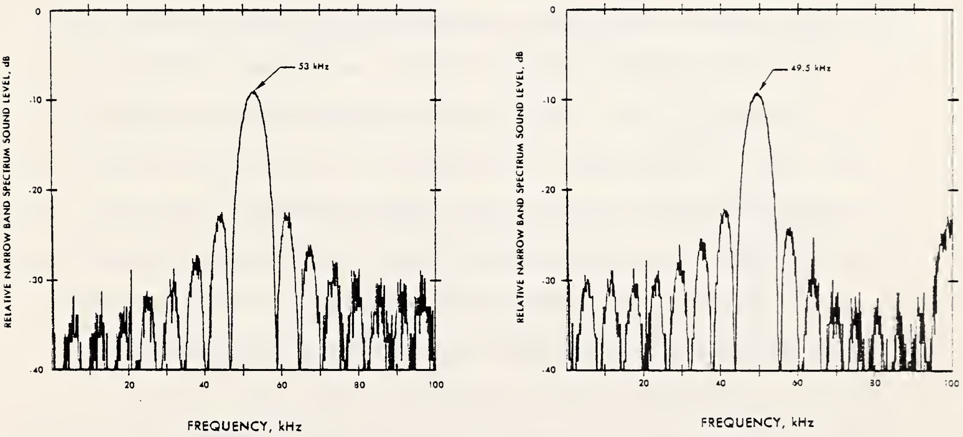


Figure 4.3.3-16. Results of narrow band spectral analysis of a single pulse radiated from the electrostatic transducer.



(a) Cycles 1 to 8.

(b) Cycles 9 to 16.



(c) Cycles 17 to 32.

(d) Cycles 33 to 56.

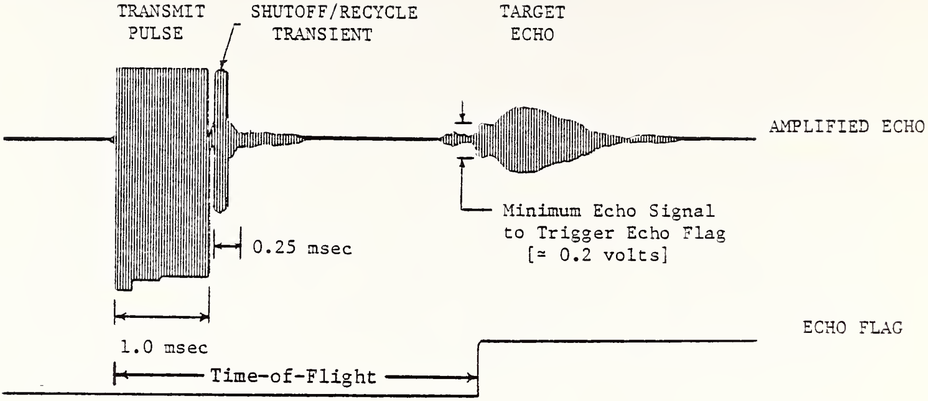
Figure 4.3.3-17. Results of narrow band spectral analysis of the individual components of the radiated ultrasonic pulse.

transmit/receive cycle and the distance attenuation of the ultrasonic pulse and its echo. The reliability of target detection by the sensor is optimized by three design features. The first feature is the use of a multi-frequency pulse. By using a pulse with four frequency components, the probability of setting up a standing wave pattern between the target and sensor so that a null occurs at the sensor (i.e., effectively no echo signal at the sensor, and therefore, no detected target) is minimized.

The second feature is the use of automatic gain control on the input amplifier to increase the signal-to-noise ratio of the received echo by increasing the amplifier gain. The reason for doing this is that the farther the pulse and echo have to travel, the smaller the received signal will be due to distance attenuation. Since this distance is proportional to the time-of-flight, the gain increase is keyed to the system clock, which is also used to measure the time-of-flight. The gain is increased in finite steps at specified timing points until an echo is received.

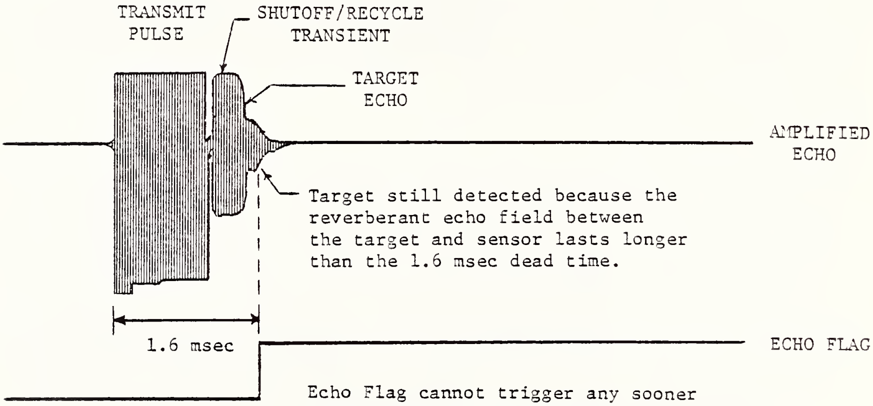
The third feature is automatic bandwidth control of the input filter. This is necessary to minimize the effects of extraneous background noise. Initially, the bandwidth is relatively wide, so that all the frequency components of the pulse can be received. Again based on the system clock, the bandwidth of this filter is decreased to a constant narrow bandwidth centered at 50 kHz. The reason for this automatic bandwidth control is that at first background noise is not a problem, because the magnitude of the echo signal will be relatively large for close targets. For targets farther from the sensor, the magnitude of the echo signal will significantly decrease as a result of wave spreading and distance attenuation so that background noise may become a problem. By narrowing the bandwidth to center on 50 kHz, the sensor can maximize the rejection of background noise and thus reduce the possibility of false detection signals. The filter is designed to center on 50 kHz rather than 60 kHz because the 50 kHz signal will be stronger, since the attenuation of sound waves in air is less for lower frequencies.

The sensor is designed to set a flag when an echo is received. It is this flag that is used to determine the time-of-flight. The detection scheme used to set this flag is based on the magnitude of the amplified echo signal. The flag is set when a threshold of approximately 0.2 volts is exceeded. This is illustrated in Figure 4.3.3-18a, which shows the amplified echo and echo flag signals from the UCB during one transmit/receive cycle. In this figure the upper trace is the amplified echo, which consists of a


TRANSMIT PULSE
SHUTOFF/RECYCLE TRANSIENT
TARGET ECHO
AMPLIFIED ECHO
0.25 msec
1.0 msec
ECHO FLAG
Time-of-Flight

Minimum Echo Signal to Trigger Echo Flag [≈ 0.2 volts]

(a) Target distance greater than 0.9 feet.


TRANSMIT PULSE
SHUTOFF/RECYCLE TRANSIENT
TARGET ECHO
AMPLIFIED ECHO
1.6 msec
ECHO FLAG

Target still detected because the reverberant echo field between the target and sensor lasts longer than the 1.6 msec dead time.

Echo Flag cannot trigger any sooner than 1.6 msec after initiation of the transmit pulse.

(b) Target distance less than 0.9 feet.

Figure 4.3.3-18. Typical output signals for the amplified echo and echo flag measured from the ultrasonic circuit board (UCB).

signal corresponding to the transmit pulse and a shutoff/recycle transient and then no signal until an echo is received. When the echo signal exceeds the 0.2 volt threshold, the echo flag is set as shown in the lower trace. The time-of-flight is then the time between the start of the transmit pulse and the echo flag as indicated in the figure. This applies to targets in the range from 0.9 to 35 feet from the transducer.

If a target is closer than 0.9 feet, it can be detected, but the sensor-to-target distance cannot be resolved. This is shown in Figure 4.3.3-18b. The sensor has a 1.6 msec dead time during the transmission of the pulse in which no received echo can be detected. This 1.6 msec dead time establishes the lower limit of the measurable range of the sensor and corresponds to the time-of-flight for a target located 0.9 feet away. However, because a reverberant sound field will exist between the target and sensor, i.e., the pulse will echo back and forth off the target and the sensor, there will still be an echo signal after 1.6 msec which will cause the echo flag to be set. Although the exact target distance cannot be determined (the sensor will indicate 0.9 feet), the target, or intruder, will be detected.

Another area of concern is the effect of different target surfaces and geometries on the echo signal and on the

target detection capabilities of the sensor. To examine this, another set of measurements was conducted in the anechoic chamber using the following targets:

(a) Plywood panel, 3' x 3', 3/4" thick

(b) Pressboard panel, 16" x 27", 1/8" thick

(c) Pressboard panel same as (b) with a 2 1/2" foam facing

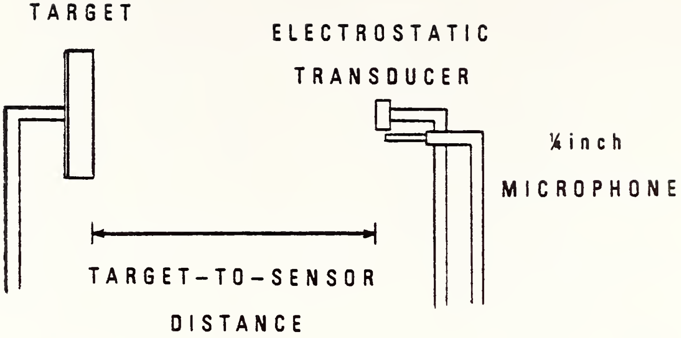
(d) Plastic manikin

(e) Plastic manikin with clothes

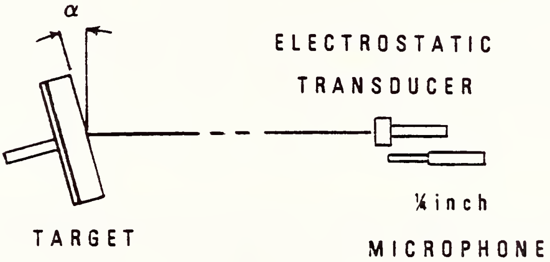
(f) Human

The echoes from these targets were measured for various sensor-to-target separation distances. Also, the effect of angling the target relative to the centerline of the sensor was investigated. The general setup for these measurements is shown in Figure 4.3.3-19.

Two types of data were recorded for the echoes from these targets: (1) acoustic measurement of the echo using a 1/4 inch microphone at normal incidence, and (2) examination of the amplified echo and echo flag signals from the UCB. In general the magnitude of the echo signals from these targets is significantly less than that of the radiated pulse. For this reason the measurements were made with a 1/4 inch microphone at normal incidence because it has a


TARGET
ELECTROSTATIC
TRANSDUCER
¼ inch
MICROPHONE
TARGET-TO-SENSOR
DISTANCE

(a) Side view showing relative positions of the hardware.


α
ELECTROSTATIC
TRANSDUCER
¼ inch
MICROPHONE
TARGET

(b) Top view showing the angular relationship between the target and sensor.

Figure 4.3.3-19. Experimental setup for the investigation of the effects of different target surfaces and geometries.

lower noise floor and greater sensitivity than the 1/8 inch microphone used for the transducer characterization measurements.

Table 4.3.3-4 lists the data for the 3' x 3' plywood panel for various target-to-sensor distances. Also listed is the maximum angle at which the panel can be turned relative to the sensor (see Figure 4.3.3-19) and still be detected and the echo flag set. These data show that the echo sound level is relatively high for reflection off this panel. For example, the sound level of the directly radiated pulse ranged from 93 to 96 dB at 3 feet for all of the transducers tested. The sound level of the echo for a target-to-sensor distance of 3 feet (actually 6 feet of total travel of the sound wave to the target and back) was 91.5 dB. This difference is about what would be expected for normal attenuation with distance (3 feet versus 6 feet). Thus as expected for a hard, flat, smooth surface, this panel has little effect on the pulse except to reflect it. The data regarding angling the panel relative to the sensor show that beyond an angle of ± 25 degrees the echo signal is not sufficient to set the echo flag; therefore, the panel would not be detected.

Table 4.3.3-5 lists similar data for the pressboard panel with and without a foam facing. Rather than list the

Table 4.3.3-4. Effect of target-to-sensor distance upon the target echo strength and the maximum angle at which the panel can be turned and the echo flag set for the plywood target.

TARGET-TO-SENSOR DISTANCE, feet	ECHO SIGNAL SOUND LEVEL, dB	MAXIMUM ANGLE FOR ECHO FLAG, degrees
1	100.5	----
2	96.5	----
3	91.5	± 23
4	87.5	± 25
6	83.5	± 18
8	79.5	± 19
10	74.5	± 20
20	65.5	----

Table 4.3.3-5. Comparison of target echo strengths for the pressboard with and without a foam facing.

TARGET-TO-SENSOR DISTANCE, feet	ECHO SIGNAL RELATIVE SOUND LEVEL, dB (see footnote)		MAXIMUM ANGLE FOR ECHO FLAG, degrees	
	PRESSBOARD	FOAM	PRESSBOARD	FOAM
1	0	-32.2	± 20	± 6
4	-11.2	-39.5	± 20	± 5

* The sound level of the echo signal is based on 20 times the log of the peak-to-peak voltage referenced to the value for pressboard at one foot.

absolute sound levels for these targets, all values are referenced to the value for pressboard at a target-to-sensor distance of 1 foot. These data show that the foam facing has a significant effect on the level of the echo signal reducing it by approximately 30 dB. This effect is also evident when the target is turned relative to the sensor. The angle at which the echo flag is set is reduced from ± 20 degrees to approximately ± 5 degrees. Thus, primarily because of absorption of the ultrasonic sound waves, the foam has much more effect on the echo than the hard, smooth surfaces of the pressboard or plywood.

When the sensors are used in the safety system to detect intruders, the targets will be human. In order to obtain information more closely related to this type of target, data were recorded for a plastic manikin with and without clothes. Also, a limited set of data was recorded for a human target to verify the manikin data. The manikin, shown in Figure 4.3.3-20, consists of the torso, head and upper arms. It does not have legs or arms below the elbow. Table 4.3.3-6 lists the data for the manikin and, for comparison, the data for the plywood panel. These data show that the sound levels of the echo signals are less for the manikin than for the plywood panel. This is expected based solely on the differences of total surface area normal to the sensor for these targets. Adding clothes to

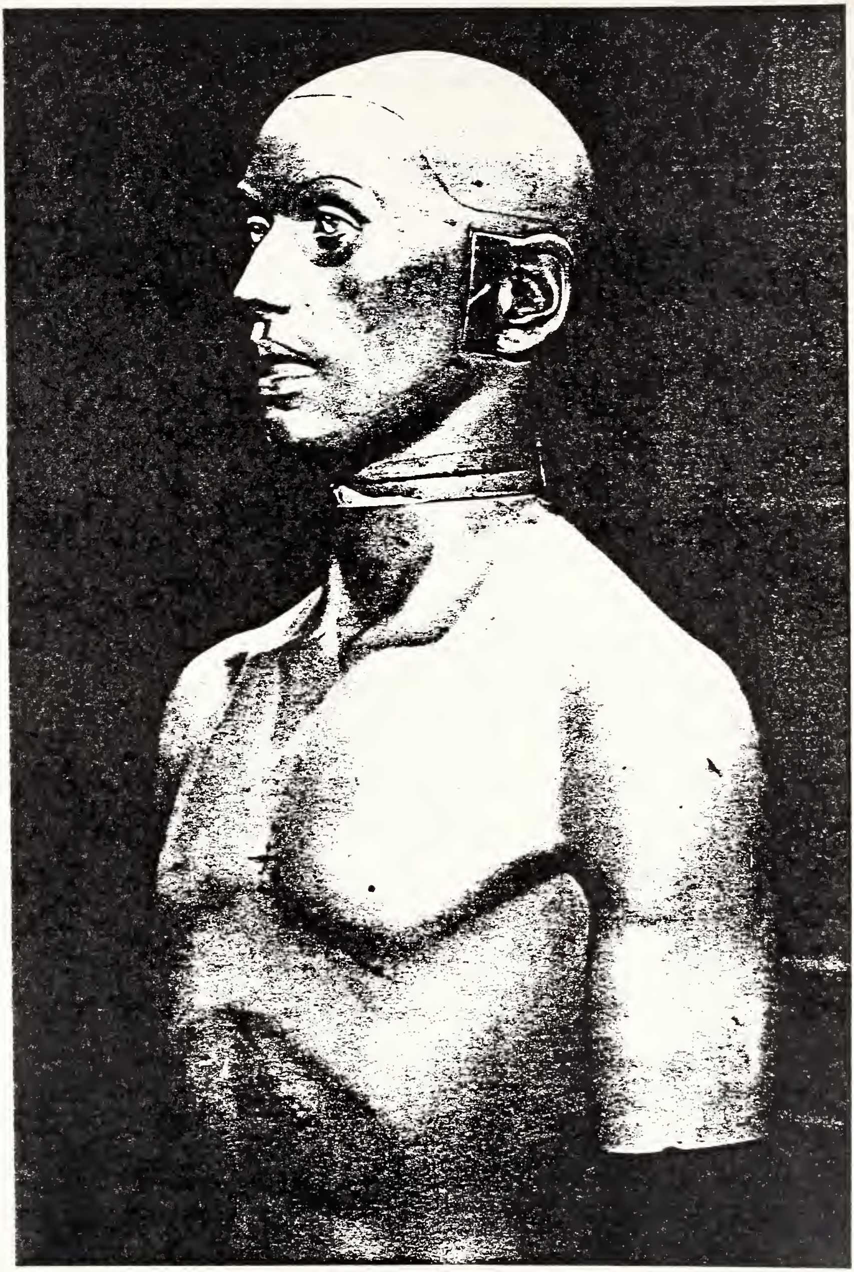


Figure 4.3.3-20. View of the plastic manikin used in the target detection measurements.

Table 4.3.3-6. Comparison of the target echo strengths as a function of target-to-sensor distance for the manikin with and without clothes and the plywood target.

TARGET-TO-SENSOR DISTANCE, feet	ECHO SIGNAL SOUND LEVEL, dB (see footnote)		
	MANIKIN (with clothes)	MANIKIN (without clothes)	PLYWOOD PANEL
1	86.5	93.5	100.5
2	82.5	86.5	96.5
3	74.5	81.5	91.5
4	70.0	78.5	87.5
6	68.0	72.0	83.5
8	62.0	68.5	79.5
10	61.0	66.0	74.5
20	42.5	41.5	65.5

* The sound level of the echo signals for target distances 4 feet and beyond are based on the level measured at one foot minus 20 times the log of the ratio of the peak-to-peak voltages of the echo pulse.

the manikin (long sleeve shirt and lab coat) reduced the sound level of the echo signals between 4 and 8.5 dB for distances out to 10 feet. For comparison to ensure that these data were representative of human targets, the sound level of the echo signal from a human at 4 feet was measured. The sound level was between 68 and 71 dB, compared to 70 dB for the manikin.

The most important fact is that the sensor, without fail, detected the manikin in all cases out to 20 feet regardless of whether the manikin was clothed or not. The manikin was rotated 360 degrees and, again, was always detected. One final test was conducted with the manikin turned sideways in front of the plywood target to determine whether the echo flag would trigger on the manikin or the plywood panel. Measurements were made for the following two sets of target-to-sensor distances:

	Manikin	Plywood Panel
(1)	4 feet	6 feet
(2)	20 feet	22 feet

In both cases the echo flag was set by the echo signal from the manikin. This is exactly the result that is desired when the sensor is used in the safety system.

Operation of the Prototype Safety System

The prototype safety system consists of the pressure-sensitive mats, shown in Figure 4.3.3-11, and an array of five of the electrostatic transducers mounted on the Stanford Arm robot, as shown in Figure 4.3.3-21. This prototype safety system is entirely hardware-based and does not have the capability of utilizing joint position information or similar data from the robot controller. Because of this, it is necessary to mount the electrostatic transducers so that they move with the robot. This is done by fastening four of the transducers to a mounting bracket, which is bolted around a section of the support for the robot arm. In this position, the sensors rotate with the robot arm while still providing clearance for the robot arm to move up and down. The fifth transducer is mounted to the rear end of the boom which moves back and forth as the robot arm is withdrawn or extended.

As seen in Figure 4.3.3-21, the transducers are put inside a phenolic case and fastened to a mounting bracket. Individual coaxial cables run from the transducers to the UCB and other electronics located under the table upon which the robot is mounted. The tranducer mounting brackets are designed to permit the transducer to be rotated right or left and up or down so that the areas of coverage can be easily set or changed as necessary.

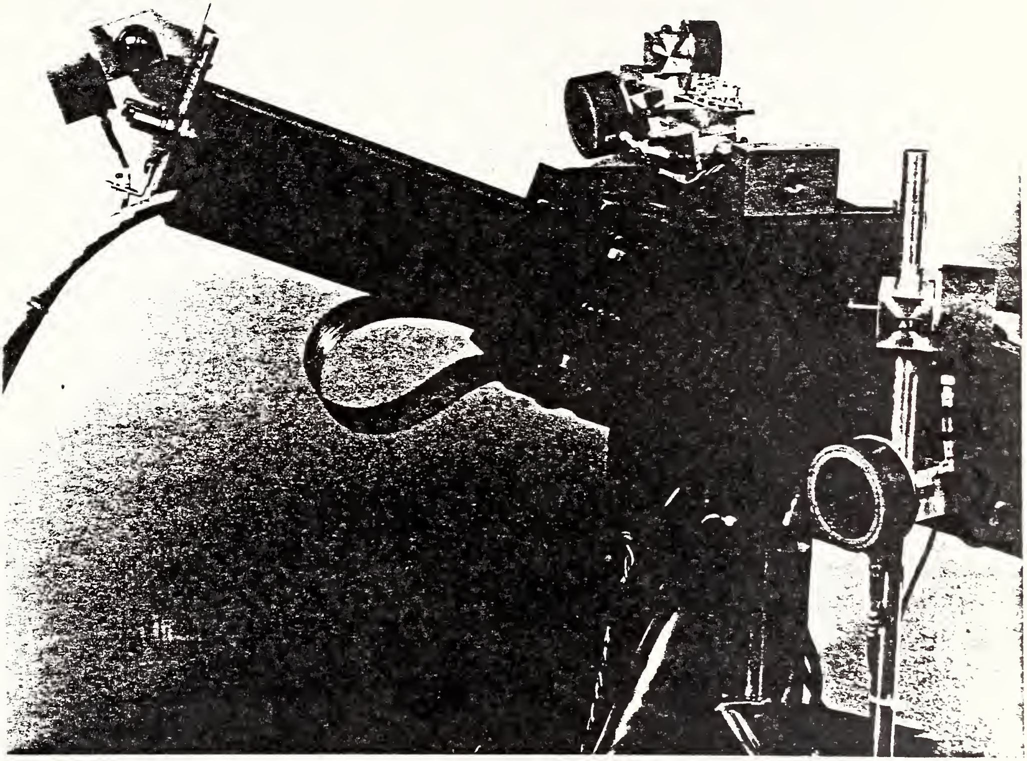


Figure 4.3.3-21. View of the Stanford Arm robot showing the locations of the five electrostatic transducers.

The two primary areas requiring coverage are those around the gripper and around the end of the boom. Because these transducers have a ± 10 degree cone in which an intruder can be detected, the positions of the transducers must be carefully selected to provide the desired coverage. One constraint to this is that the transducers cannot distinguish among an intruder, a workpiece on the table top, wire cables on the robot arm, or the robot grippers. Thus, the sensors must be positioned so that these other objects do not enter the operating cone of the five transducers.

With these design goals and constraints, the transducers were positioned to provide coverage in the areas illustrated in Figure 4.3.3-22. The robot gripper and the end of the boom are the two areas of primary coverage, because these locations have the highest potential for a collision. There is no coverage to the sides. The only way the robot can strike an intruder is by rotating to the right or left. However, before a collision could occur, the intruder would be detected by one of the transducers, since the cone sweeps across the intruder's location before the robot arm does. The other area not covered is directly in front of the robot gripper. Although this omission is not desirable and will be eliminated in more advanced safety system designs, it is not a problem in this

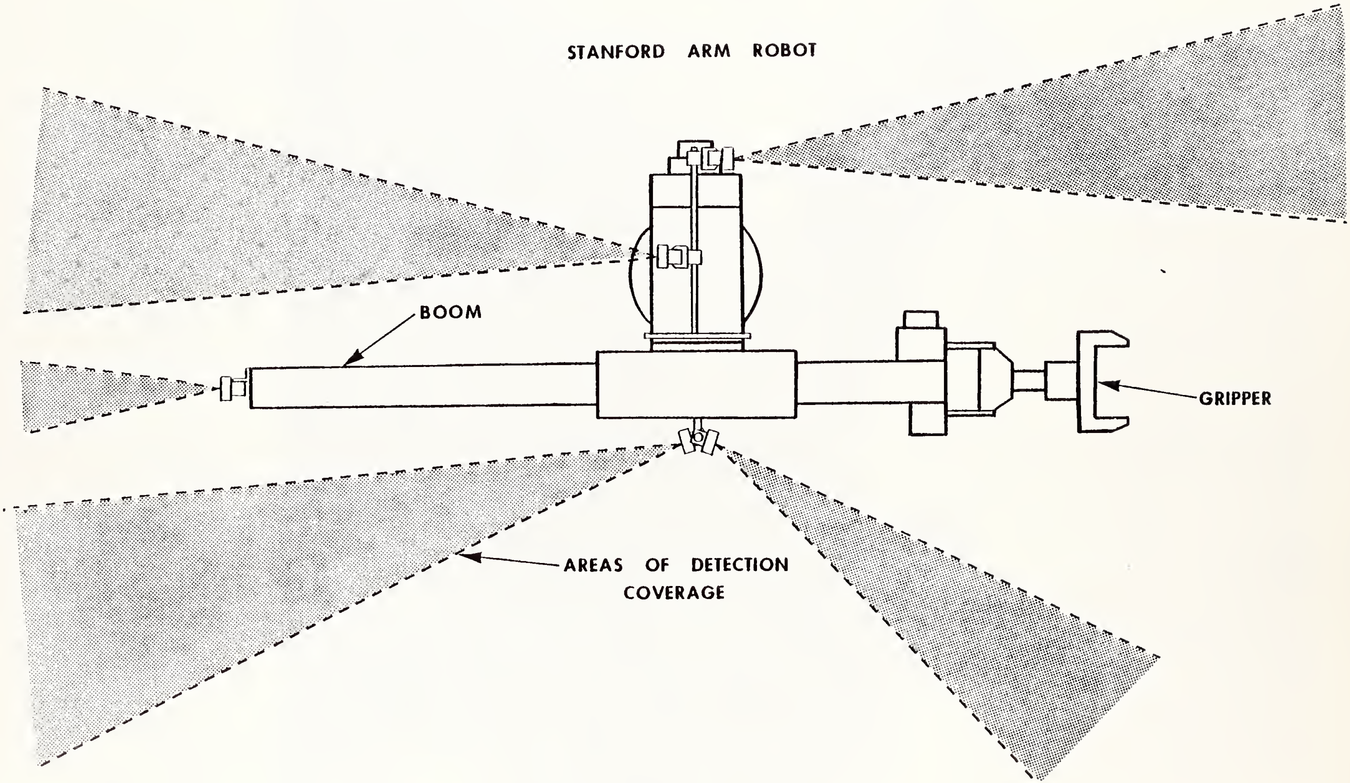

STANFORD ARM ROBOT
BOOM
GRIPPER
AREAS OF DETECTION COVERAGE

Figure 4.3.3-22. Approximate locations of the electrostatic transducers on the robot arm and the areas of detection coverage.

application, because the gripper is always operating over the table top, which extends beyond the reach of the robot arm.

The safety system electronics are designed so that any echo signal corresponding to a target (i.e., an intruder), beyond some minimum distance, is disregarded. Thus, as an intruder approaches the robot, an echo signal is received, but it is not until the intruder reaches this minimum distance that the signal is sent to the controller and the robot stopped. This minimum distance was set to be approximately one foot beyond the reach of the robot arm, so that the robot can be stopped before the intruder takes one more step and can potentially be struck by the robot. This distance was found to be more than adequate to stop the robot and prevent a collision, even with the robot operating at maximum speed and the intruder walking directly towards the robot.

In operation, the safety system electronics are configured to give a single output to the robot controller when an intruder is detected by any of the sensors. (There is also another signal input, corresponding to the circular mat, which is independently sent to the controller.) When an intruder is detected within the prescribed minimum distance and the signal sent to the controller, the robot is halted.

As long as the intruder remains, the robot stays in this position. The minimum length of time the robot is stopped is 0.5 seconds (for a 25 msec sampling rate of the output of the safety system electronics by the robot controller). After the intruder has left, the robot continues to perform the operation it had started from the position in which it was halted.

One potential problem is false detection indications which unnecessarily cause the robot to be stopped. These false detection signals can be caused by electronic interference, extraneous noise sources with frequency components in the 50 to 60 kHz range, or echo signals from one transducer being received by another transducer. This problem was handled in the electronics of the prototype system by requiring that the echo flag be set at least twice in a 0.8 second period. With an electrostatic transducer repetition rate between 7 and 8 pulses per second, this requires that two out of six consecutive pulses from the transducer result in an intruder detection indication. With this design, false detection indications were totally eliminated.

Overall, the prototype safety system performed well for all tests that were conducted. The pressure-sensitive mats operated as designed and will be used in the development of future safety systems. For the electrostatic transducers, false detection signals, unnecessarily causing the robot to be halted, were totally eliminated. More importantly, the robot was capable of stopping within an adequate distance and in time to prevent any collisions with an approaching intruder.

Future Safety System Developments

Future safety system developments will be concentrated in two areas. First, there will be further developments utilizing the electrostatic transducers. This will involve incorporating a microprocessor in the system electronics so that information from the robot controller, such as joint positions, can be used to refine the operation of the system. These refinements will permit techniques, such as difference mapping, to be used. Difference mapping involves storing a set of acceptable transducer outputs for various joint positions when no intruders are present. During operation, the safety system will compare the currently measured values with the stored values for that particular set of joint positions to determine if an intruder is present. Using difference mapping will permit sensors to be mounted on the robot near the gripper and at other locations not on the robot. Even if the robot or the workpiece triggers one of the sensors, the robot is not halted because the stored values would indicate that these conditions were acceptable for that set of joint positions. Difference mapping will provide better protection coverage without increasing unnecessary stops because of false detection indications.

The other area will be development of Level III sensor systems for detection close to the robot. As mentioned previously, there are cases, such as teach-mode operations, where an operator must be close to the robot while it is operational. The Level III system will permit this, yet still detect an imminent collision and signal the robot to stop before it can occur. Various types of sensors, such as infrared devices and piezoelectric polymers, will be evaluated. A Level III system will be developed utilizing one of these sensor techniques and will be incorporated into the overall safety system design.