To the Graduate Council:

I am submitting herewith a thesis written by Wendy Ellen Moore entitled "The Effects of Force Reflection on Servomanipulator Task Performance." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mechanical Engineering.

Gary V. Smith, Major Professor

We have read this thesis and recommend its acceptance:

Accepted for the Council:

Vice Provost
and Dean of The Graduate School

THE EFFECTS OF FORCE REFLECTION ON SERVOMANIPULATOR TASK PERFORMANCE

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Wendy Ellen Moore

December 1986

To my grandparents.

ACKNOWLEDGEMENTS

There are many individuals at both professional and personal levels without whose efforts the completion of this degree would not have been possible.

First of all, appreciation is extended to my major professor, Dr. Gary V. Smith, whose belief in my abilities as both a student and an individual enabled me to grow both professionally and personally. It was through his encouragement that I attempted and completed this major chapter in my career. Thanks also in extended to the other members of my committee, Dr. James A. Euler and Dr. Masood Parang.

The wonderful group of engineers at ORNL have made this learning experience the ultimate. Although many of those in the Robotics group are electrical engineers, I would like to thank them for not only tolerating but tutoring yet another mechanical engineer. It was great. I would especially like to thank Steve Zimmerman and Brad Weil for helping me to learn and appreciate FORTH as well as providing encouragement along the way.

Finally, I would like to thank my family for supporting my decision to earn this degree and providing the love needed to do so. A very special thanks is also extended to Paul Stevens, who had the love and faith to stick by me throughout most of this ordeal, thus providing my link to the real world and that ounce of sanity every graduate student needs to survive.

ABSTRACT

Extensive data collection was performed on a servomanipulator system, the Sargent Industries' Central Research Laboratories Model M-2, in an effort to study the effects of force reflection on remote handling task performance. The results show the impacts of force reflection on both the environment and the operators in terms of overall forces incurred, errors committed, and the amount of time required to perform a typical task. The experiment incorporated a multi-axis force/torque measurement table as a primary data source, along with motor tachometers, accelerometers, and current taps.

This thesis reports preliminary results of the testing program. The results indicate that force reflection reduces operator errors and lessens the magnitude of forces applied during task completion. Further, it seems that force reflection can be counterproductive if it fatigues the operators. In this experiment, task completion times were longer and errors more frequent with 1:1 force reflection than with 4:1 force reflection. This supports the assumption that force reflection provides useful feedback information to the operator in remote handling and suggests the manner in which force reflection should be implemented in order to best protect both the hot cell equipment and the manipulator.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF FIGURES

FIGURE PAGE

FIGURE PAGE

INTRODUCTION

The Oak Ridge National Laboratory's (ORNL) Consolidated Fuel Reprocessing Program (CFRP) is responsible for development of systems for reprocessing commercial nuclear fuel. To cope with the problem of equipment maintenance in the high levels of radiation and atmospheric corrosive elements in future nuclear reprocessing facilities, the CFRP's Remote Control Engineering (RCE) section has been developing dexterous servomanipulators to replace humans in this dangerous role. These systems usually consist of similar master and slave manipulators. Each manipulator is composed of two multi-degree of freedom arms with end effectors. The operator moves the master arms and the slave arms follow in one-to-one real-time correspondence. Remote closed circuit television (CCTV) allows the operator to view the working environment. Force reflection, a feedback element in the manipulator controls which allows the operator to "feel" the amount of force he is exerting, whether he feels all or simply a fraction of the force, has been an added sensory channel in the recent ORNL servomanipulator developments. This study will investigate the importance of force feedback as an element of remote handling technology.

CHAPTER 1

BACKGROUND

The History of Remote Handling

The sudden rise of nuclear energy in the 1940's brought on many new ideas in the world's technology. Many of these concepts, such as remote handling, evolved out of need to protect man from hazardous nuclear environments. Remote handling stemmed from a need to project the types of manipulations previously performed by humans into a remote environment. Thus, the beginning of a series of developments which enabled the projection of the capabilities of man into hostile environments (17). These types of manipulator systems, in various stages of their development, have been used by the nuclear industry for nearly three decades (8). Much of the technology has also been enhanced and improved by space and underwater applications where there is also a need to overcome physical barriers and project man-like manipulations into a hostile environment (21,22).

In 1947, a group at Argonne National Laboratory (ANL) led by Ray Goertz was formed for the specific purpose of developing remote-handling systems. Much of the early work done in this area was performed by this group.

The first manipulators developed at ANL were velocity controlled unilateral (non-force reflecting) manipulators. These manipulators consist of a mechanical arm with seven or more independent motions. One motion is simply a pair of tongs for gripping objects. The arm, along with a support system and control box constitute the manipulator. All motions are actuated by electric motors, and motor speeds are controlled by joysticks or switches moved by human operators. Primary disadvantages of this system include the lack of dexterity and force sensing capabilities, and the fact that only one degree of freedom of the arm can be moved at a time. The control system for these manipulators is open loop, thus resulting in simple and reliable performance. Self-locking worm gear drives in addition to an open loop system result in a manipulator which exerts near maximum force when in contact with a fixed object.

The next manipulator system developed by ANL was a mechanical master-slave type manipulator which consists of a slave (working) arm to perform work in the hostile environment and a master (control) arm, similarly shaped, controlled by a human operator. These manipulators are able to perform dexterous tasks with force reflection through steel tape which connects the master and slave arms. These manipulators were very popular and are still widely used today. This type of manipulator however, has several disadvantages. Since the master and slave are directly connected, the operator is required

to remain in close proximity to the working environment. In addition, the volumetric coverage is limited because the system is mounted at a fixed pivot point. The ratio of the slave to master forces is fixed at about 1:1, which limits the slave to only exerting forces within the range of human capabilities (8).

The more recent developments in unilateral manipulation have been performed by Remote Technology Corporation (23). This unilateral manipulator is controlled by a replica master in a position mode, requiring position feedback from the remote environment. Drive mechanisms allow some compliance via backdriving in the slave arm through the use of spur and bevel gearing. This reduces the tendency of a unilateral manipulator to exert maximum force on any stationary system, as is a problem with the early ANL unilateral system discussed earlier.

Servomanipulators

Electromechanical servomanipulators were developed to overcome limitations imposed by mechanical master-slave manipulators. Although many similarities exist, the basic difference lies in the fact that connections are electrical rather than mechanical resulting in many advantages. The slave manipulator can be mounted on a mobile transporter system resulting in a greatly increased volumetric coverage. Though the cost is greater than that of a similar mechanical system, the decrease in the number required to fully cover a large area

results in an overall lower cost for the electromechanical servomanipulator. Further, the master and slave need not be located in close proximity to one another. This has resulted in many additional applications for servomanipulators including outer space, military and undersea task performance.

Another advantage is the ease in which the force feedback ratio can be changed in electromechanical systems thus allowing for comfortable force sensing for the operator and a larger range of force capabilities for the slave arm depending on the task requirements.

The concept of the force reflecting servomanipulator was developed by ANL in the mid-1950's. The original control systems relied completely on analog electronics. No memory storage devices were required for operation since the human operator provided continuous manual control. The original systems were designed to minimize friction and inertia. Force feedback was accomplished by backdriving the mechanisms to move the motor and thus produce a position error (8).

Later developments by Flatau (7) and Vertut (19) further refined the design of force reflecting servomanipulators. Vertut first developed the ability to store and repeat manipulator motions, which led to further computer techniques designed to ease the operator's burden.

Servomanipulators can be driven with electric motors or hydraulic actuators. The hydraulic systems are normally favored

in undersea or extremely heavy-duty applications. Electromechanical systems offer sensitivity of 1% peak load force through backdriveable gear trains, and are used on most modern servomanipulator systems (9).

Just as an object in space has six degrees-of-freedom, (DOF), the general purpose manipulator must have six DOF's plus an end effector for gripping objects. Any additional DOF's merely increase complexity, difficulty, and are often redundant. Modern servomanipulators usually have configurations similar to the human arm. The most common end-effector is a general-purpose two-fingered tong gripper, a very simple and reliable end-effector.

Commercially marketed complete master-slave servomanipulator systems are currently provided by only two American companies. These are Sargent Industries' Central Research Laboratories (CRL) and TeleOperator Systems Corp. (TOS) (9).

The CRL Model M-2, originally developed and installed at ORNL represents the first successful implementation of a fully digital control system for a force reflecting manipulator (16). This application of digital control results in an inherently flexible control system by virtue of software implementation. Digital control also allows for an even easier change in force reflection ratio, and eliminates the problems of drift inherent to

analog control systems. Table 1 outlines major servomanipulator technology developments.

The CRL Model M-2 was the servomanipulator used in this experiment and will be discussed in further detail later.

While many similarities exist in servomanipulator systems and artificial intelligence based autonomous robotic systems, their control techniques differ significantly (20). While robotic controls focus on feedback sensors such as vision and pattern recognition which allow for autonomous repetitive operation in structured environments, servomanipulators rely on the presence of the operator to utilize sensory perception (view, position, force reflection) to perform tasks more efficiently and with a greater level of dexterity (18). Thus, performance objectives for these two types of systems differ significantly. Recent research has placed a large amount of emphasis on robotic developments, but the research regarding servomanipulator development is limited due to their specialized applications. Recent developments include a fully remotely maintainable servomanipulator system developed at ORNL (13). ORNL has a technical exchange program with France and Japan (5,6) in order to provide a channel in which basic developments are shared. Funding is continually increasing for work in servomanipulator development due to the increase in outer space applications.

TABLE 1

SERVOMANIPULATOR DEVELOPMENT HISTORY

Year	Milestone
1948	Argonne National Laboratory develops first mechanical master-slave.
	General Mills produces unilateral manipulator using analog switch control.
1954	ANL develops first electromechanical force-reflecting servomanipulator.
1958	General Electric develops first electrohydraulic force-reflecting manipulator.
1961	General Mills adapts manipulator for underwater application.
1965	ANL develops head controlled TV camera and monitor.
1970's	NASA sends soil samplers to the moon and Mars. NASA and Spar develop Shuttle Remote Manipulator System.
1980	Supervisory manipulator control implemented by Massachusetts Institute of Technology. Universal manipulator controller developed by the Jet Propulsion Laboratory (JPL) and Stanford University.
1982	Oak Ridge National Laboratory and Sargent Industries develop the first digitally controlled servomanipulator.
1983	Odetics developed the teleoperated walking functionoid.
1984	Oak Ridge National Laboratory develops a remotely maintainable servomanipulator.

Purpose of Experiment and Review of Related Research

As has been discussed, the presence of a human in the control loop distinguishes servomanipulators from other robotic developments, therefore the performance of the operator is extremely important to the overall performance of the manipulator system. Therefore, the RCE section is concerned with the man-machine interface, and the assurance that the operator is not limited by lack of sensory feedback. A minimum requirement along this line is the use of monochromatic, monoscopic television systems which may be further enhanced by full-color scene reproduction or stereoscopic viewing.

The CFRP advocates the use of force reflection as a supplementary sensory channel. In this experiment, a comparison was made between force reflection levels of 1:1, 4:1 and 1:0. While operating in the 1:1 mode, the entire force exerted by the slave side of the manipulator is reflected to the operator through the master arms. Likewise, the 4:1 mode implies one-fourth of the slave force is "reflected" to the operator and in 1:0 the operator has no force feedback from the slave side. Force reflection is accomplished by backdriving the servo motors of the slave arms in order to develop a position offset in the control system. Although this feedback to the operator is quite crude compared to the human tactile senses, it does provide

some information to the operator about the actual "feel" of objects, impact, and spacing, especially important when viewing is under suboptimal conditions.

Important factors which must be considered when testing servomanipulator task performance include the amount of time required to perform the task, the quality of work done (i.e. errors incurred), the effects of the manipulator on the task equipment, and the effects of the manipulator on the operator. Previous research performed to study the effects of force reflection failed to study all four parameters of servomanipulator task performance (1,11,14). These results were even less meaningful since they were probably biased by using only two operators, by a lack of varying force reflection levels (often only 1:1 or 1:0), and by a lack of number of tasks performed, or even changing manipulators to accomplish different force reflection levels. This experiment utilizes several operators, three levels of force reflection, and four different tasks in order to more fully investigate all four task performance factors without undue biasing of the data in the evaluation of force versus non-force reflection conditions.

Earlier research conducted at ORNL in the course of manipulator comparison failed to yield any positive effect of force reflection on the performance of some remote handling tasks (4). However, it was concluded that the tasks and procedures used in the first experiment were not primarily

designed to evaluate force reflection and may have been insensitive to its effects, thus demonstrating a need for development of specific hypotheses concerning the effects of force reflection.

From this ORNL study, the following hypotheses were devised for further experimentation and study. First of all, force reflection is a valuable tool especially when the viewing by the operator is not under optimum conditions or when the operator is learning a new task. Secondly, force reflection is helpful in successfully performing tasks involving alignment or insertion of an object. In order to prove or disprove these hypotheses, the tasks involved in the experiment must be designed especially to show the effects of force reflection.

From this set of conclusions lies the basis for this particular study concerning the impacts of force reflection. This experiment was concerned particularly with determining the validity of the statement that force reflection is helpful when it provides information that cannot be acquired through vision and which is important to task performance.

CHAPTER 2

EXPERIMENTAL SYSTEM DESCRIPTION

Remote Operations and Maintenance Demonstration Facility

This experiment was conducted in the Remote Operations and Maintenance Demonstration (ROMD) Facility, which is operated by CFRP. The ROMD facility consists of a high-bay remote handling equipment demonstration area filled with prototypical process equipment and manipulator systems, along with a control room for the manipulator (2).

The high bay area of ROMD encases an area with a 48 foot working height and a 4312 $ft.^2$ ground floor level. A 1046 $ft.^2$ pit area is also included and extends 30 feet below the facility ground level. Large sections of the facility floor are available for remote handling experiments, such as this one, with portable equipment mock-ups and test stands. Figure 1 shows an overhead view of the ROMD high-bay area with much of its prototypical process equipment.

The master station, shown in Figure 2, is housed in the ROMD Control Room located adjacent to the ROMD facility. The station incorporates a pair of CRL Model M-2 force reflecting servomanipulator master arms to relay operator input (10). Switches on the master handles allow for slave tong lock, slave

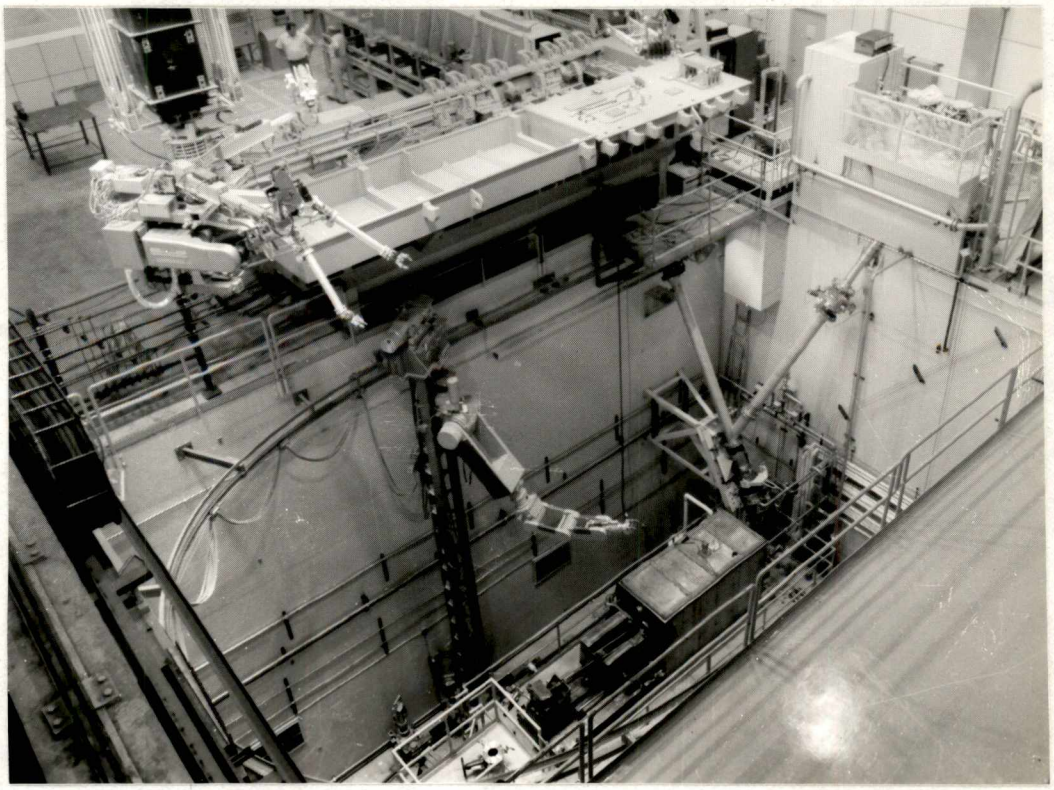


FIGURE 1. OVERHEAD OF ROMD FACILITY

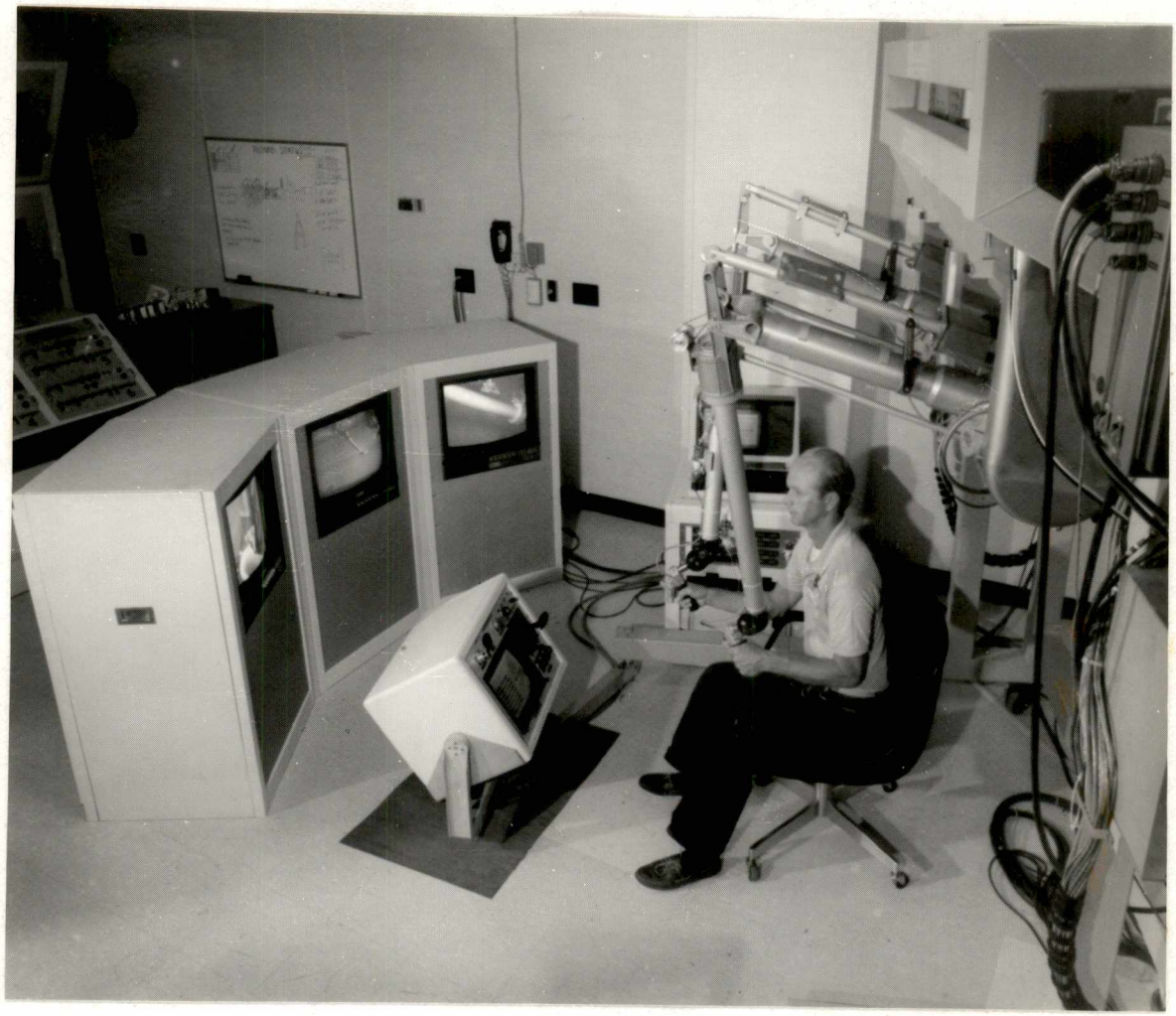


FIGURE 2. MASTER ARMS AT CONTROL STATION

arm lock, "all joint" indexing, tool control, and a unique deadman function which removes all power from the master arms when the operator's hands are removed. The operator interfaces with the control system primarily through a menu-driven touch-screen/CRT. This operator interface provides operating mode selection, force ratio selection, camera/lighting control, and the system status diagnostics. The cameras, two of which are mounted to provide four DOF overhead views and the other which is mounted at the center of the M-2, provide the operator with orthogonal views to optimize perspective and enhance depth perception, along with wide-angle view from the center camera. The two overhead cameras are equipped with zoom lens. The camera position is joystick controlled, while the focus, iris, and zoom are controlled by rotary potentiometers.

The CRL Model M-2 Servomanipulator

The Sargent Industries' Central Research Laboratories' CRL Model M-2 (Figure 3), the servomanipulator chosen for use in this experiment, represents the first successful implementation of an entirely digitally controlled servomanipulator and thus represents a major step forward in servomanipulator development (10). The system incorporates a distributed, microprocessor based digital control system. The design of the manipulator arms, developed and installed at ORNL, are based on

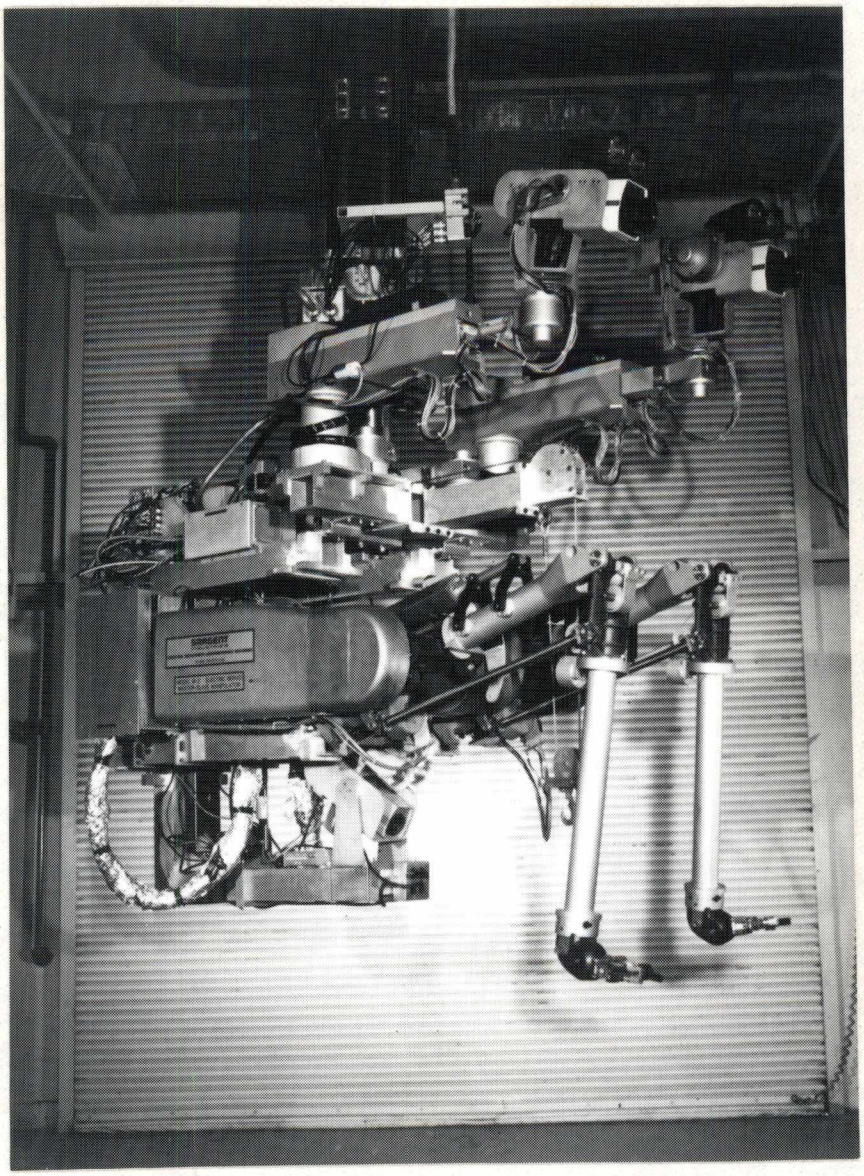


FIGURE 3. THE CRL MODEL M-2 SERVOMANIPULATOR

the CRL Model M servomanipulator developed in the early 1960's at Argonne National Laboratory as the ANL Mark E4A. The original Mark E4A kinematics, shown in Figures 4 and 5, have been retained for the most part. The master and slave arms have the standard seven degrees of freedom arranged in an "over the wall" or "elbows up" configuration, and their motions are in one-to-one correspondence. Each master and slave joint are driven by their own state-of-the-art servomotor drive units. The servomotors are brushless dc with rare-earth samarium-cobalt permanent magnets. These motors, capable of higher torque capacities, allow the servomotor drive units to be packaged with one motor each. Further, an 11% increase in continuous capacity and a 100% increase in peak capacity have been attained over the original E4A design, while the motion maximum velocities are more than double the original design. In addition to a servomotor, each servomotor drive unit incorporates encoding sensors, brakes, a cooling fan, and an enclosed gearbox. This drive unit is easily removed and replaced as an assembly to facilitate remote maintenance.

The Model M-2 master arms have maintained their original light structure design to minimize effective inertia. (10). The upper three degrees of freedom, X,Y,and Z, for both the master and slave are gear and lever driven. The lower four degrees of freedom, (azimuth, elevation, twist and squeeze), are cable driven on the slave side, and steel tape driven on the master side

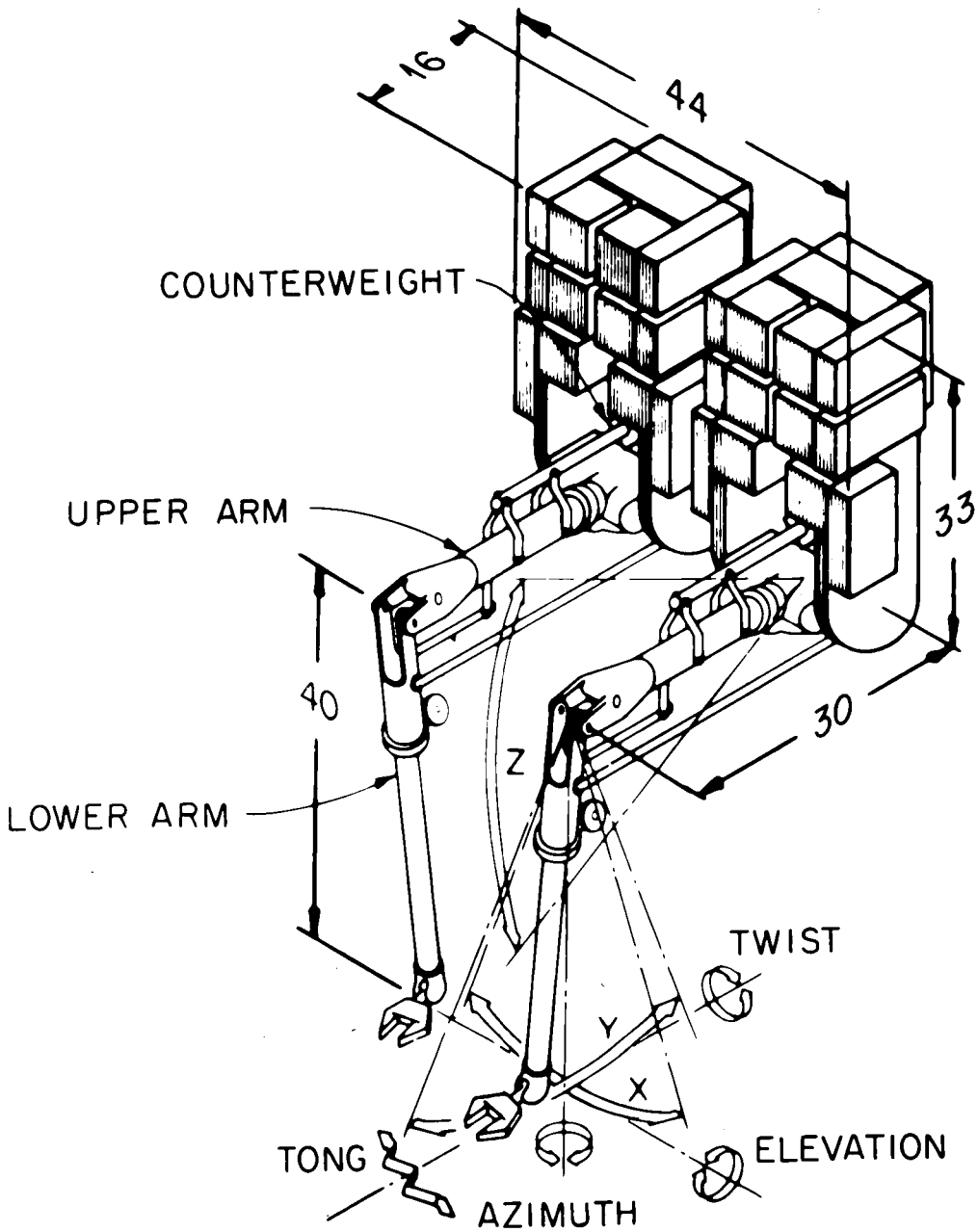


FIGURE 4. ANL MARK E4A SLAVE MANIPULATORS

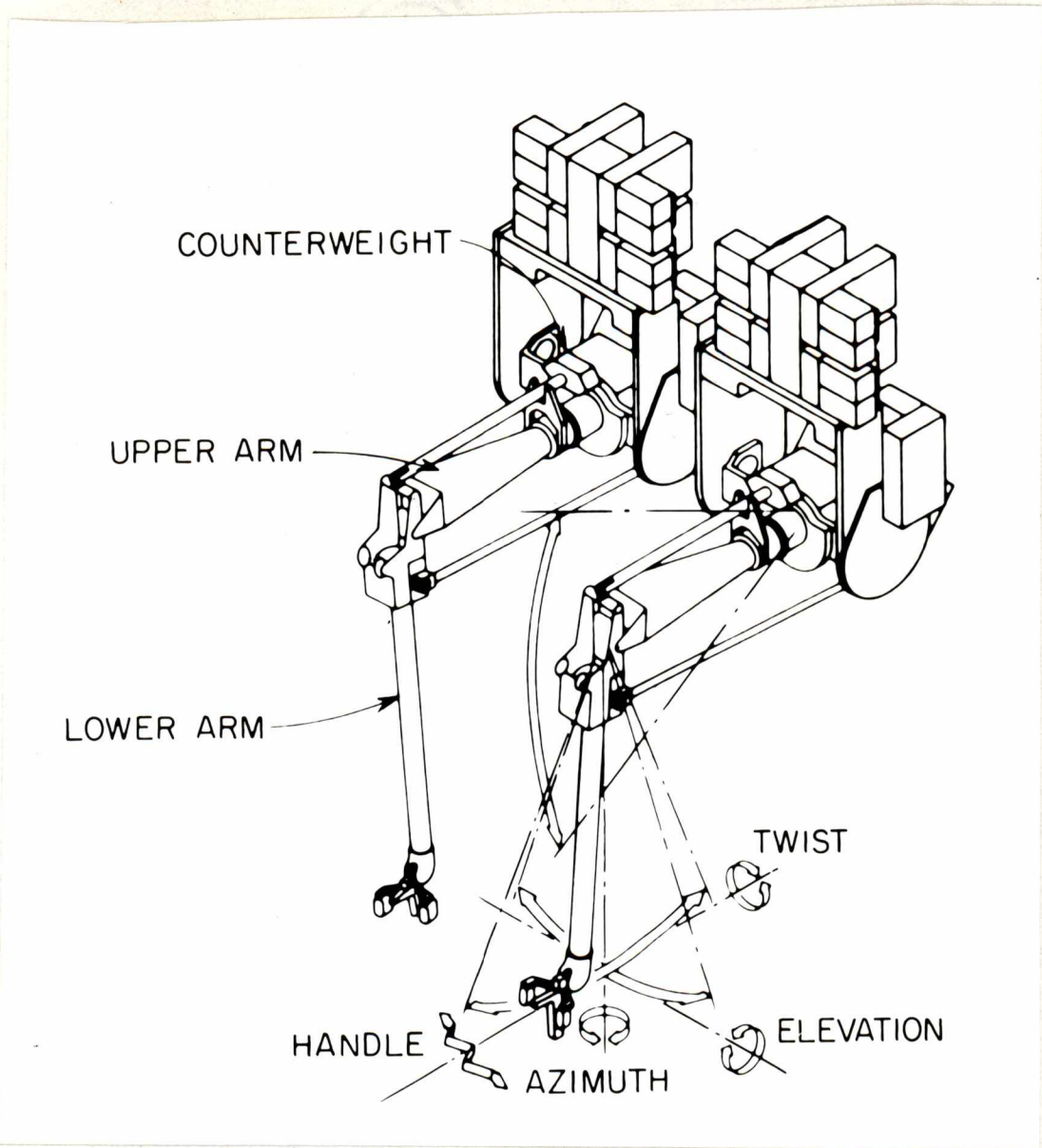


FIGURE 5. ANL MARK E4A MASTER MANIPULATORS

to reduce friction and enhance sensitivity. An additional feature on the M-2 is the remotely detachable wrist assembly which allows removal of the wrist joint and tong without disturbing the slave cable drives. This is very important due to the fact that this is the portion of the slave arm which is most vulnerable to mechanical damage and contamination.

A standard position-position bilateral control technique, implemented in the digital control hardware, provides force reflection in ratios from 1:1 to 8:1 as well as operations without force reflection.

Table 2 provides a summary of performance and operating characteristics for the model M-2 Servomanipulator.

Description of Tasks

Four tasks were selected as representative of remote-handling tasks while being sensitive to force-reflection requirements. The hardware associated with these tasks was attached to a task "box", pictured in Figure 6 which in turn was attached to a three-axis orthogonal load sensing platform, as shown in Figure 7. Figure 8 shows the M-2 performing a task, and shows clearly the load platform to which it is attached. A further description of the tasks is as follows:

TABLE 2

PERFORMANCE AND OPERATING CHARACTERISTICS
FOR THE MODEL M-2 SERVOMANIPULATOR

	Gear and Lever Driven DOF's		
	X	Y	Z
Maximum Master Force	25 lb.	25 lb.	25 lb.
Maximum Slave Continuous Operating Force	50 lb.	50 lb.	50 lb.
Peak Slave Force Capacity	100 lb.	100 lb.	100 lb.
Maximum No-Load Linear Velocity	60 in./s	60 in./s	60 in./s
Typical Slave Arm Range of Motion	± 45 deg.	± 45 deg.	± 45 deg.

	Tape Driven DOF's (*Tong Handle)			
	Yaw	Pitch	Roll	Tong
Maximum Master Force or Torque	150 lb. in.	130 lb. in.	110 lb. in.	25 lb.
Max. Continuous Slave Torque	500 lb. in.	412 lb. in.	350 lb. in.	100 lb. (squeeze)
Max. No-Load Vel. Linear or Ang.	344 deg./s	400 deg./s	344 deg./s	40 in/s
Typical Slave Arm Range of Motion	± 210 deg.	+ 40 deg. - 125 deg.	± 180 deg.	3.75 in. 2.75 in.*

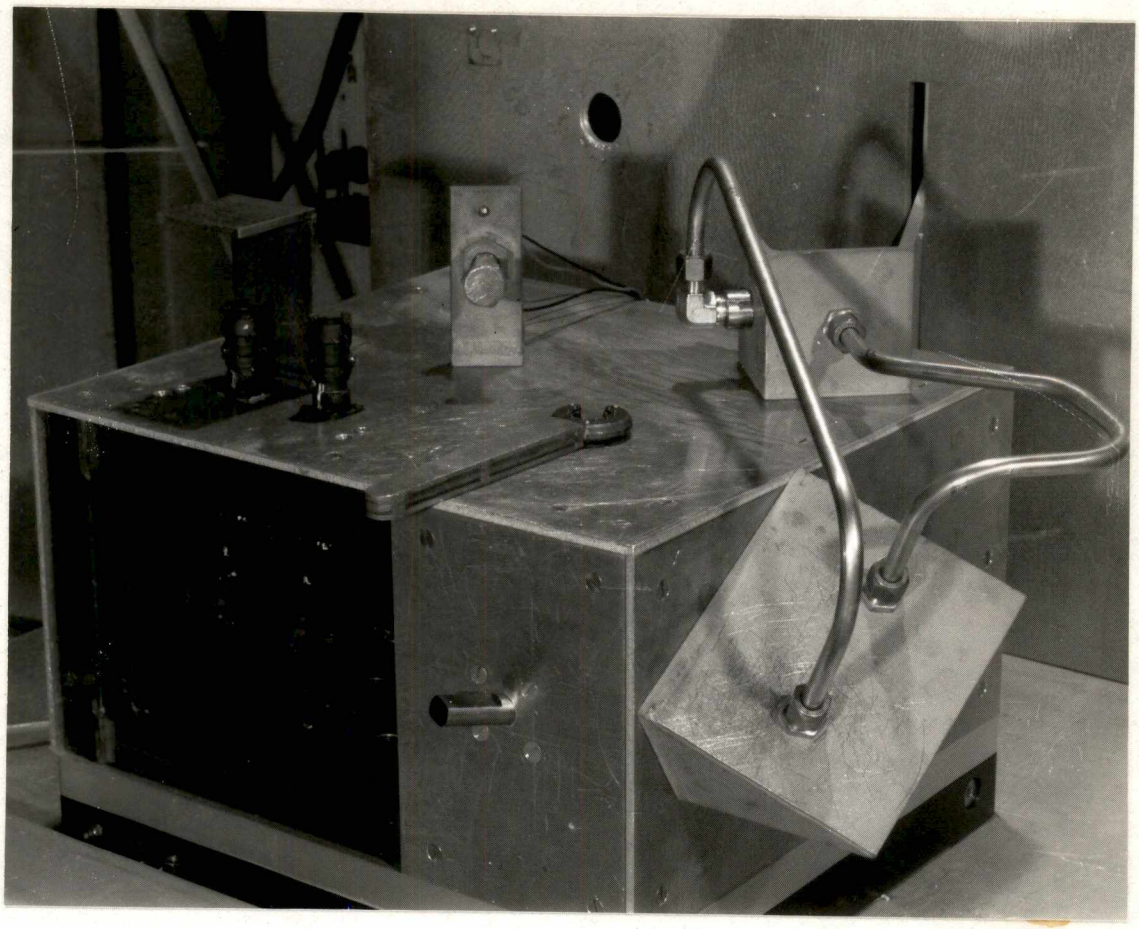


FIGURE 6. TASK "BOX" USED IN EXPERIMENT

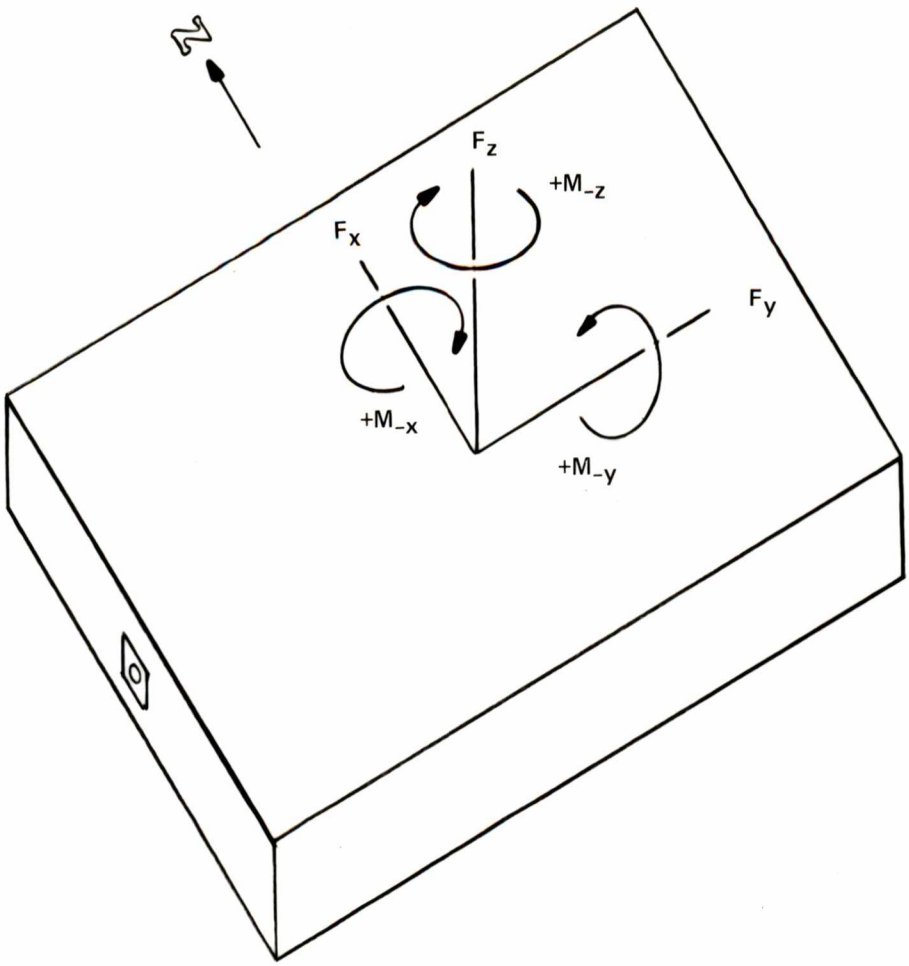


FIGURE 7. AXES REFERENCE FOR LOAD SENSING PLATFORM

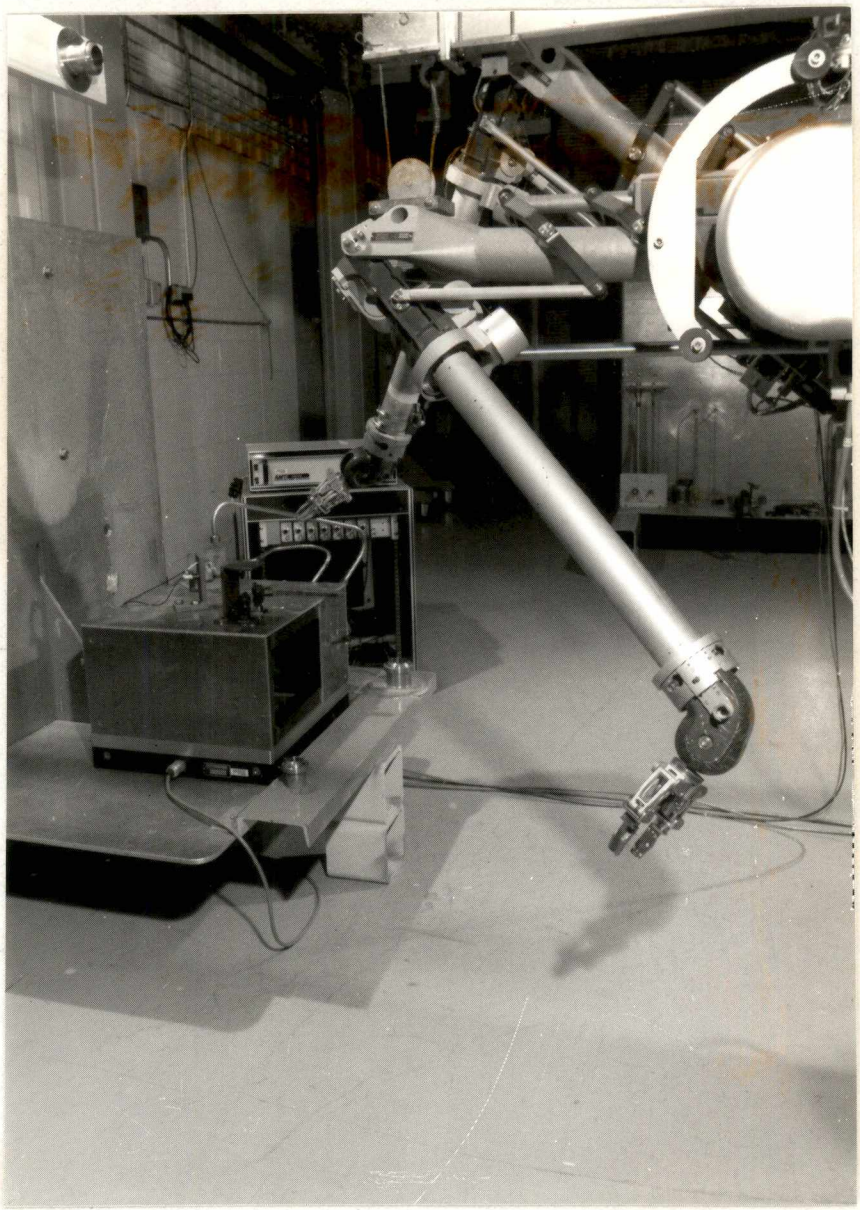


FIGURE 8. M-2 MANIPULATOR DURING PERFORMANCE OF A TYPICAL TASK

Task 1 - Electrical Connectors

This task involved the assembly of two pairs of remotely adapted (Standard Amphenol-type) electrical connectors. The operator picked up the connectors which rest on the top of the task box and plugged the ends into the appropriate sockets. After connecting all four connectors, the operator unplugged and replaced the connectors at the top of the framework (See Figure 9).

Task 2 - Peg-in-hole

To accomplish the peg-in-hole task, the operator removed a peg which was fully inserted into a hole, touched the task box above the hole with the end of the peg, then re-inserted the peg. The peg was mounted at a 15 degree elevation, offset to the left 15 degrees from the sagittal plane of the task box with the high end of the peg closest to the manipulator and canted toward the manipulator's left side as the manipulator faces the task box (See Figure 10).

Task 3 - Tubing Jumpers

This task utilized a pair of stainless steel tube jumpers with remotely adapted (Swagelock-type) pipe fittings. The socket fittings for the first jumper were mounted on vertical plates on top of the task box, while the fittings for the other jumper were mounted on a plate on the side of the task box. The

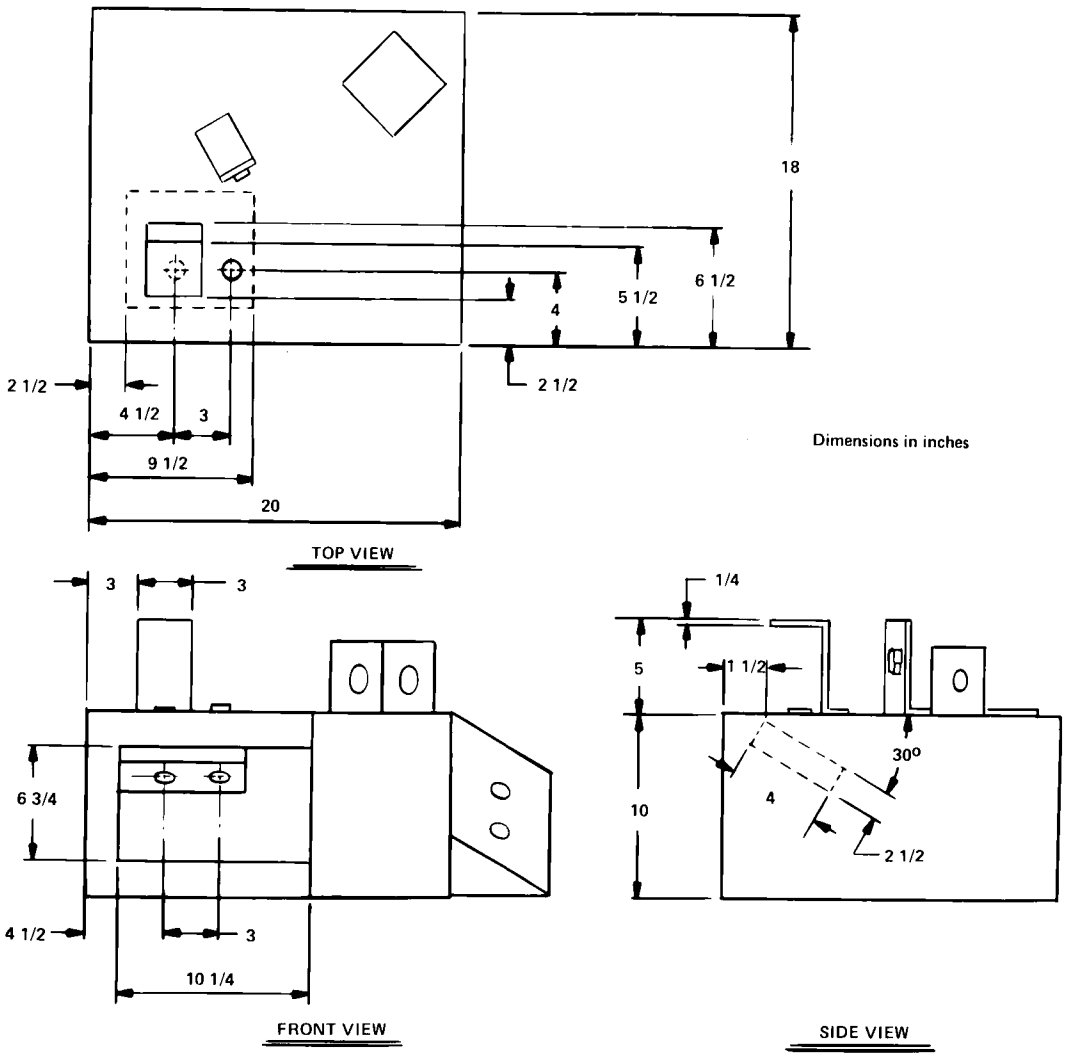


FIGURE 9. ELECTRICAL CONNECTORS TASK DIMENSIONS

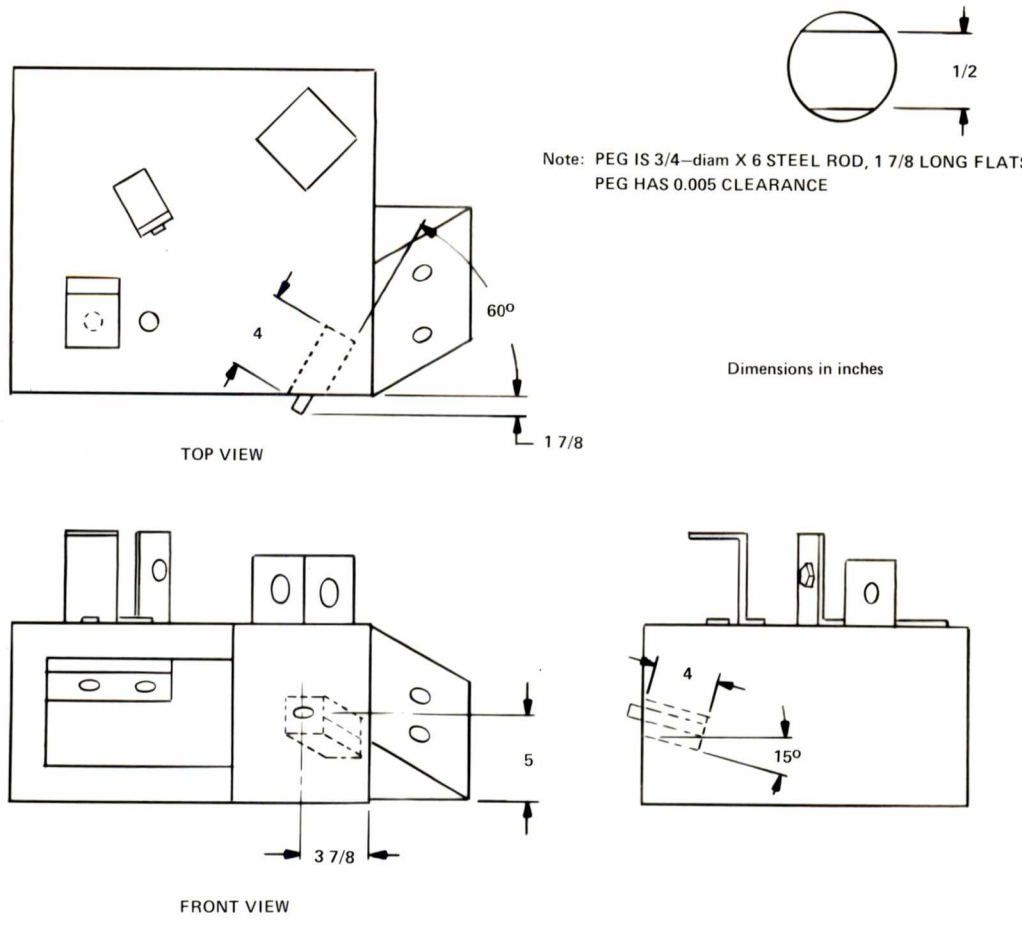


FIGURE 10. PEG-IN-HOLE TASK DIMENSIONS

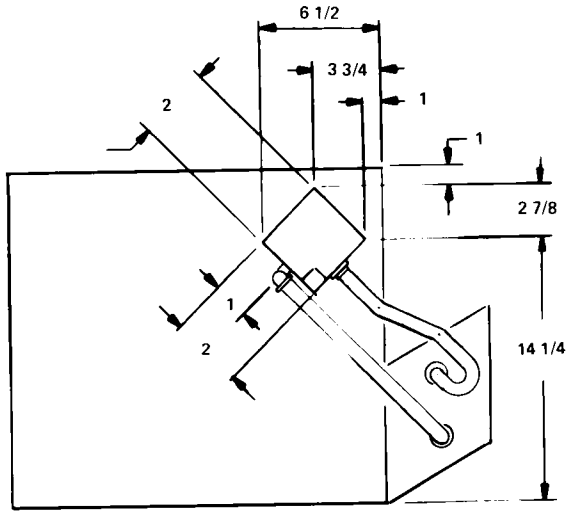
side plate was tilted 30 degrees to the baseplate along the horizontal and 30 degrees along the vertical axis. The front and outside edges of this plate were lower than the back and inside. During this task, the operator picked up the tube jumpers from the top of the task box, inserted the ends into the appropriate fittings, and tightened the fitting nuts with a wrench which he picked up from the top of the task box (See Figure 11).

Task 4 - Bolt Threading

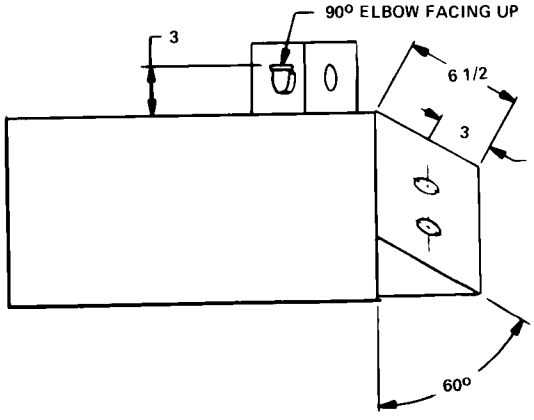
In this task, the operator picked up a 3/4 -in.-diameter, 3-in.-long bolt from the top of the task box, positioned and rotated it until it was engaged in a 3/4-in. nut welded to a plated on top of the framework. A light emitting diode on the task box indicated when the bolt was fully inserted (See Figure 12).

Work Plan Description

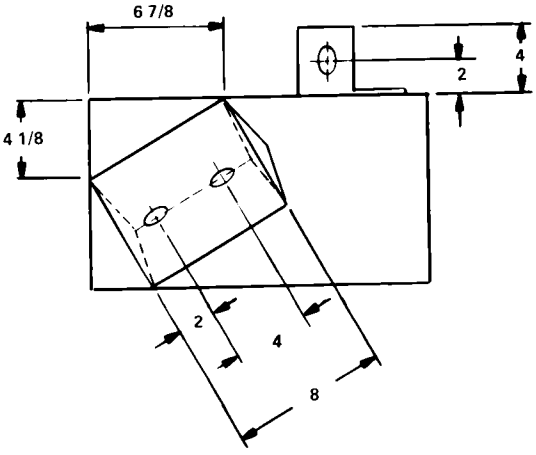
Six qualified operators participated in this experiment. Each operator completed 1.5 h of practice prior to the start of testing, 0.5 h at each force reflection level (1:1, 4:1, and 1:0). Following completion of practice sessions, each operator completed 15 testing sessions, each session consisting of six trials, two each with three experimental tasks, in two sets of one repetition per task. Within sessions, tasks were performed



TOP VIEW



FRONT VIEW



SIDE VIEW

FIGURE 11. TUBING JUMPERS TASK DIMENSIONS

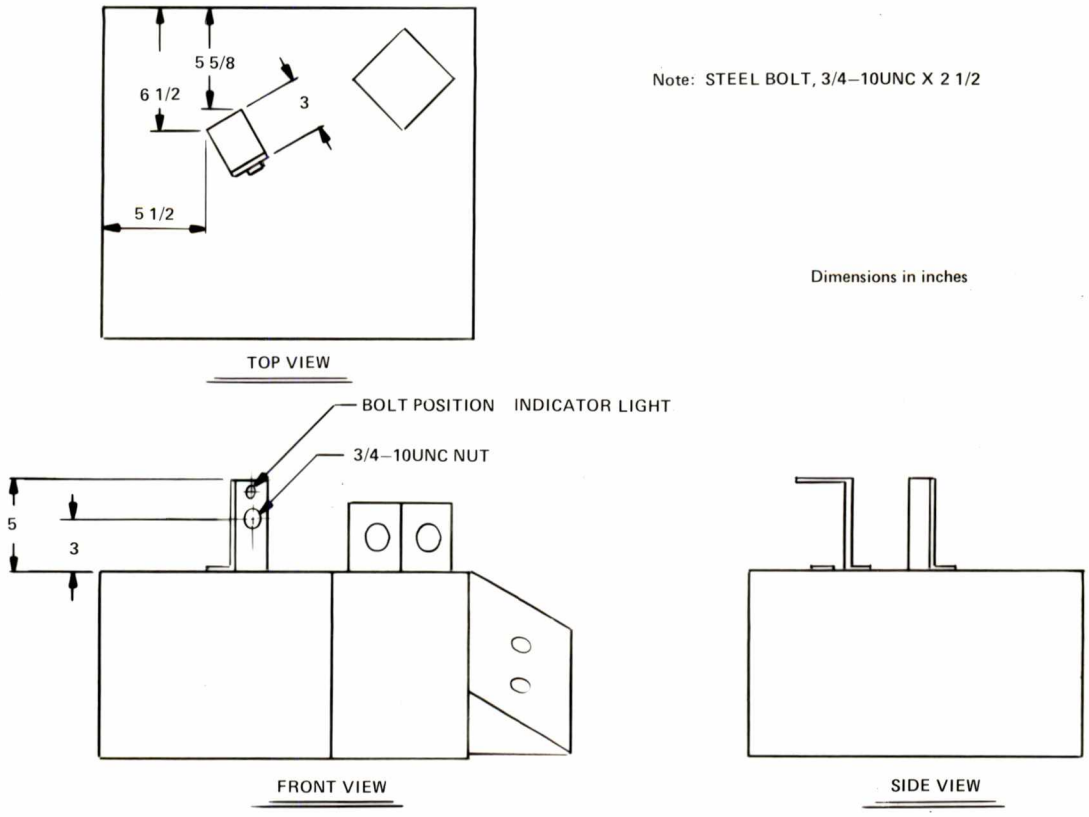


FIGURE 12. BOLT THREADING TASK DIMENSIONS

in random order, under constraints that tasks would not be completed consecutively and no task would be repeated before completion of all tasks. Six of the sessions also included one repetition of a fourth task, completed at the end of the session. A rest period of at least 1 h between consecutive sessions for the same operator was required, and no more than 6 h was allowed to elapse between testing sessions for one operator. These restrictions ameliorated operator fatigue and prevented operators from getting "out of practice."

Each of the six operators performed each task ten times for each force reflection level, with the exception of the "novel" task, bolt threading, which each operator only performed twice for each force reflection level. These numbers total 576 task completions.

Competitiveness between operators was avoided by preventing them from knowing any details of operator performance (task completion time, errors committed, etc.). Further, operators were not allowed to view other operators' testing sessions.

During task performance, television views were restricted to those available from cameras on board the M-2 package. The two cameras on arms having four degrees of freedom were set up to give views from approximately 45 degrees to either side of the tasks (outboard of the sagittal plane of the manipulator).

Operators were allowed to use either of these views and/or a view from the camera positioned between the manipulator arms. Figures 13 through 16 show more clearly the manipulator positioning and camera alignment. Operators were not allowed to see views provided by cameras other than those on board the M-2 package during trials, and were not allowed to change the aim, magnification, or position of the cameras. This procedure simulated the restricted viewing conditions which may be encountered in actual maintenance operations, where equipment may occlude the line of sight of television cameras (3).

Data Acquisition

Data characterizing several performance criteria were collected during testing sessions. The data taken during experimentation included task completion time, a tally of errors committed during task completion, the forces and moments applied during the tasks (logged by a three axis orthogonal load sensing platform onto which the task hardware was mounted), velocities of each joint in the manipulator right slave arm, acceleration of the right slave arm end effector, and motor currents at four important right slave arm joint motors. A complete list of the hardware used in this experiment is found in Appendix A. Each of these variables (except error occurrence) was recorded at 20 samples per second by a Hewlett-Packard

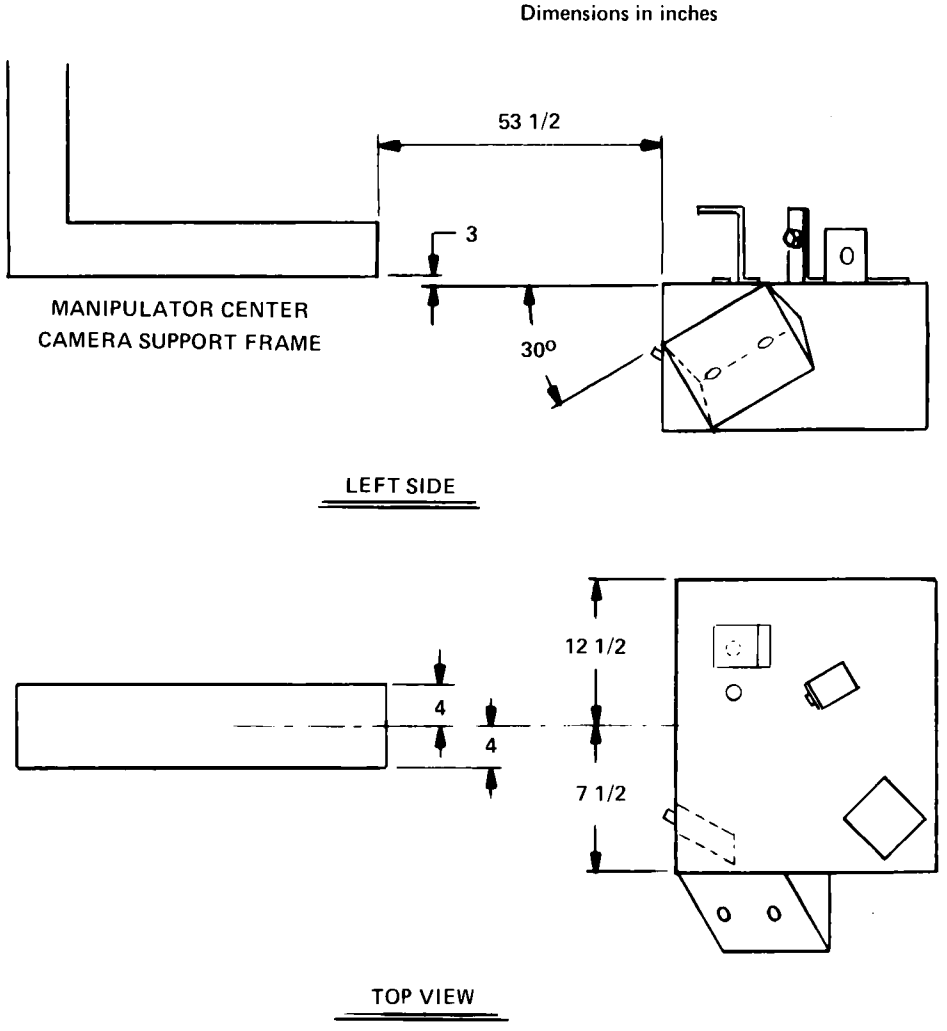


FIGURE 13. MANIPULATOR POSITIONING

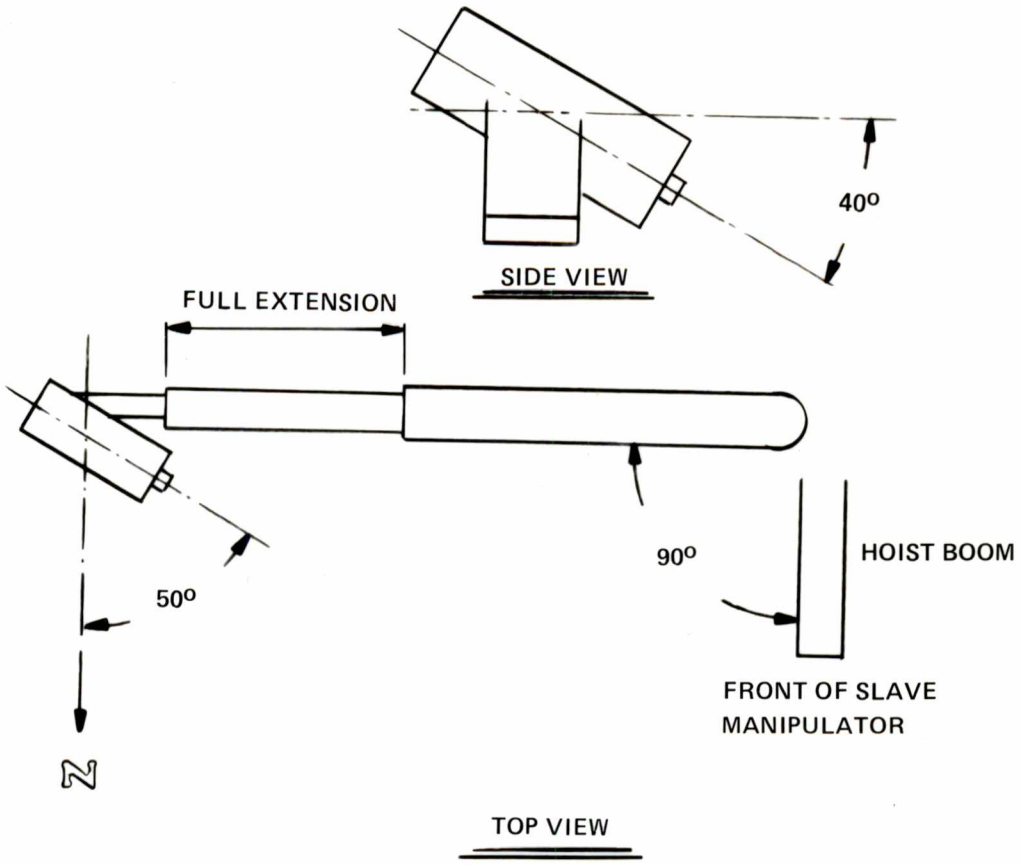


FIGURE 14. MANIPULATOR CAMERA POSITION
(RIGHT OVERHEAD)

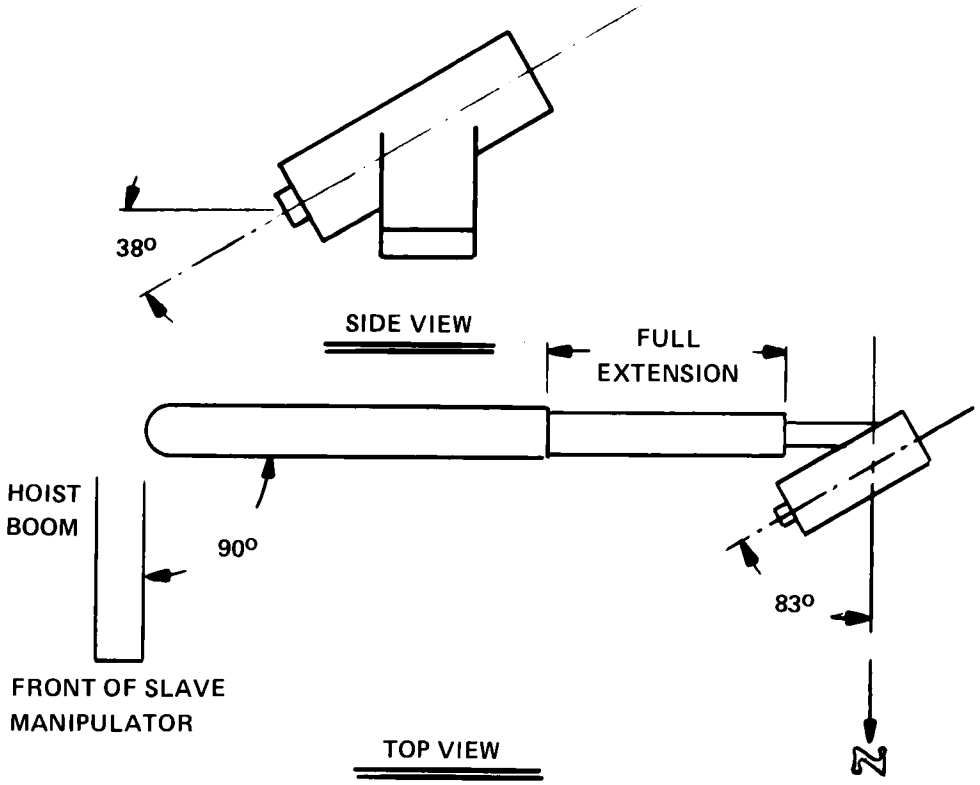


FIGURE 15. MANIPULATOR CAMERA POSITION
(LEFT OVERHEAD)

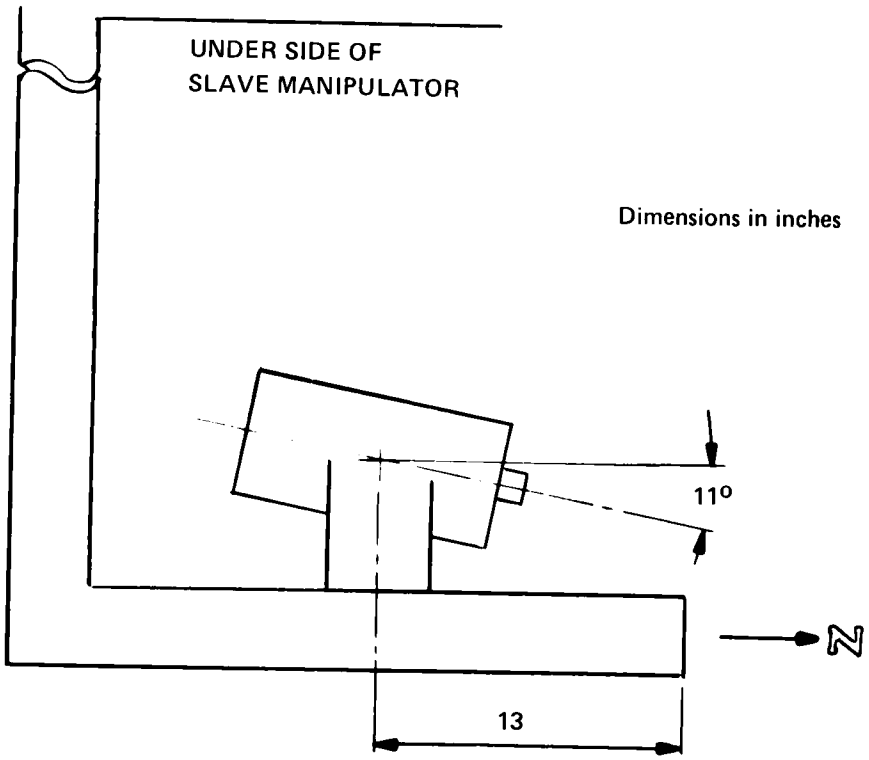


FIGURE 16. MANIPULATOR CAMERA POSITION
(CENTER LOWER)

9236 computer programmed in Multi-FORTH to scan the 21 channels of A/D information and store the data to a Hewlett-Packard 24 megabyte hard disk/tape drive system, allowing permanent data storage to a tape file. For further reference, each task was videotaped with the video clock synchronized with the remote time clock on the data acquisition system. The importance of FORTH as a data capture/data analysis software system will be discussed in more detail in the Data Reduction Chapter of this thesis.

CHAPTER 3

DATA REDUCTION

FORTH Background

The Hewlett-Packard 9236 Computer System incorporates a Motorola 68000 microprocessor, a very fast and powerful processor which complements the use of FORTH as a software tool. Multi-FORTH is a 32-bit implementation of FORTH which is developed by Creative Solutions, Inc. A bit of background on the theory behind the FORTH system will better explain its role in this application.

FORTH is not merely a programming language. In addition to the language, it consists of an operating system for disk-based development systems, an editor, an assembler, a compiler, a run-time interpreter, and a basic vocabulary of what are called verbs or primitives (operators) and nouns (constants and addresses). Words, the smallest unit of FORTH code, can act like verbs or nouns depending on whether they are naming actions to be performed or naming locations or addresses within memory. These words are defined in a dictionary. The FORTH programmer actually defines words and adds them to the dictionary. Each word is made up of previously defined words. Therefore, it is easy to see how the language builds upon itself and can be easily checked at each step, or word, to monitor the correctness. The

execution of a command is controlled by an interpreter. The interpreter translates the string of words in a command into computer instructions. The FORTH interpreter, however, operates on two levels, text and address. This enables a much faster run time than other interpretive languages. When the text interpreter finds a word, it executes that word by starting the address interpreter which starts at the most recently defined word and works backwards until it locates the address of the word in the dictionary. For this reason, FORTH is called a threaded interpretive language. The fundamental difference between developing a program in FORTH and developing programs in other conventional languages such as BASIC or FORTRAN concerns the novel dictionary structure. At run time, only a small interpreter is needed to execute the application program. This approach allows FORTH programs to run 10 to 50 times faster than BASIC. Only a small portion of the total FORTH code is required to perform the run process which results in a low overhead for real time tasks (12).

Because of the ideal suitability of FORTH to real-time applications, it is used solely in the control of the latest ORNL servomanipulator development, the Advanced Servomanipulator, or ASM. The RCE advocated its use in servomanipulator control systems, and as a result, it also followed that FORTH would be the instrumental system in both the data capture and the data analysis for this servomanipulator performance test. FORTH was

an essential tool in the data capture in order to scan 21 channels of A/D in 50 milliseconds and store to a hard disk. Because the data were formatted in FORTH, and there was no working software available at the time of data analysis to convert formatted FORTH to BASIC files, the data analysis therefore was also performed in FORTH, which constituted a major part of this research effort. The data analysis software, as will be described in more detail later, allows for fast and accurate viewing of data from related experiments in the future.

Data Analysis Software

Data from this experiment, stored on cartridge tapes, totaled approximately 480 megabytes. Because of this massive amount of data, much time was spent writing software, in Multi-FORTH, to examine the data and determine the best approach to a logical presentation. Software was developed to (1) plot normalized histogram arrays for each task; (2) plot variables versus time, including resultant force and moment vectors; (3) find maximum and average resolved forces and moments; (4) break down errors into the 12 categories tallied; and (5) evaluate sum of squares and cross products matrices for further statistical analysis. Forces less than one pound and moments less than one lb-in. were not included in the averaging process. These were below the threshold of the the manipulator system sensitivity and were considered random noise.

All maximum and average force and moment data, operator error data, task completion times, and matrix information was then entered into a file to enable a study using Statistical Analysis Software in order to more fully analyze the results of the experiment. A copy of the fully documented data analysis software is provided in Appendix B.

Figures 17 and 18 are representative graphics performed with the FORTH data analysis software. The samples are taken from data recorded by the same operator performing the same task within the three different force reflection modes. The plots of force versus time as shown in Figure 17 demonstrate the ability to take any of the variables measured during the experiment and plot them as a function of time in order to more fully study the various differences in modes of operation between operators, tasks, or force reflection levels. In this example, not only are the impact forces apparent, but also a trend for the forces to increase towards the end of the task. This could be a sign of operator frustration and/or fatigue.

The normalized histograms, as demonstrated in Figure 18, are tools which allow for a much faster comparison of data sets and thus a quicker general analysis of the behavior of a variable within a data set. Again, the histograms shown here represent data from the same type of task, for the same operator, while varying force reflection. Here, it is important to note the

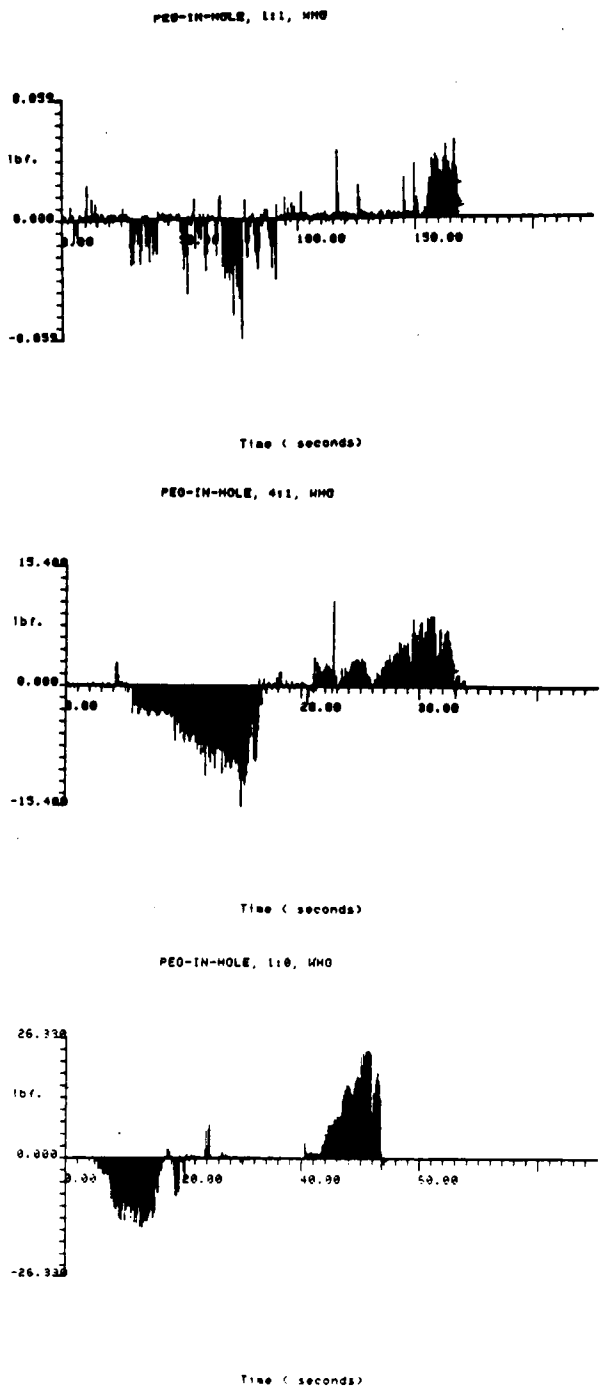


FIGURE 17. REPRESENTATIVE VARIABLE VERSUS TIME PLOTS

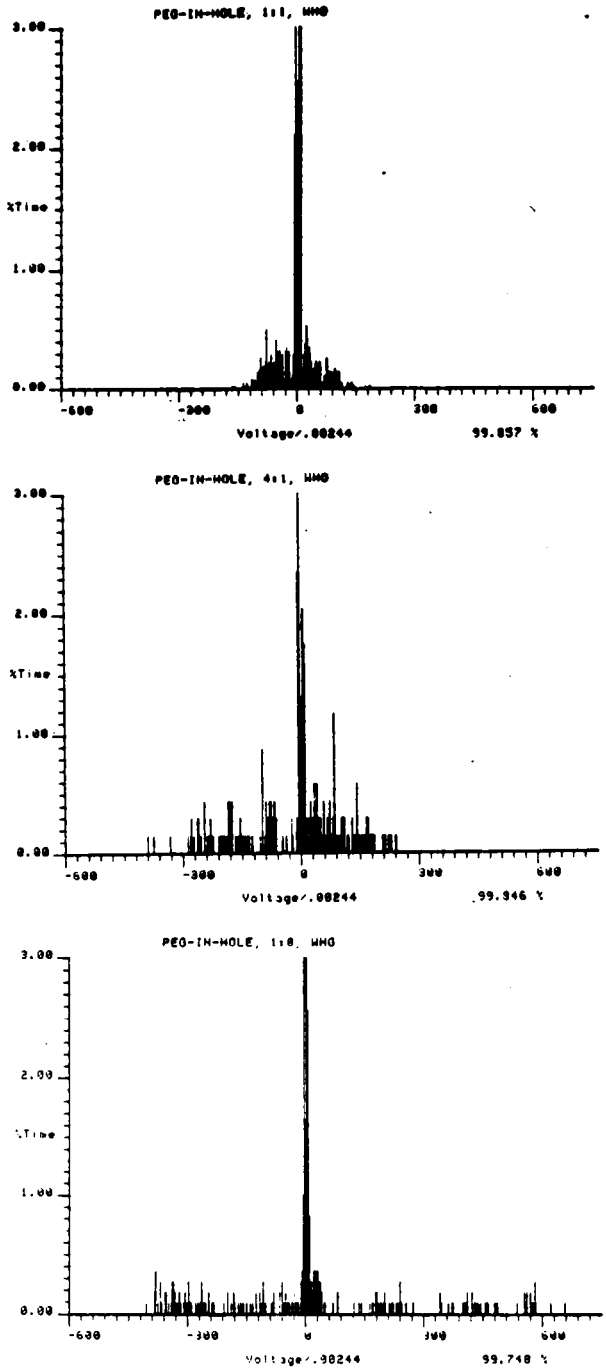


FIGURE 18. REPRESENTATIVE NORMALIZED HISTOGRAM PLOTS

decrease in "%Time" with increase in the force becomes much sharper as the level of force reflection increases.

Much of the information concerning general trends in manipulator use gathered from the use of the histograms and the variable versus time plots echo results obtained through statistical analysis. However, the FORTH data analysis software can be implemented using the same hardware as was used for data capture and can provide very quick results. Much of the original intent when purchasing the experimental hardware involved here was to provide a portable test unit which could easily be used on different manipulator systems and yield quick answers to questions concerning the manipulator mode of operation. This data analysis software provides a basic tool in allowing such a portable experimental unit.

CHAPTER 4

DISCUSSION OF RESULTS

Although several types of information were recorded during testing, the results discussed in this thesis will primarily focus on the force/moment data analysis, accompanied by the number of operator errors and task completion times. By taking this approach, the four factors of manipulator task performance, rate, quality, effects to the operator and manipulator, can be studied.

Task Completion Time

The amount of time required for each operator to perform each task (576 total tasks) was recorded during the course of the experiment. Each time value, in seconds, was converted to its common log value before averaging in order to lessen the effects of those values which varied greatly from the mean (15). This statistical approach has been utilized in previous ORNL research regarding manipulator testing (4). Therefore, it should be noted that each geometric mean time value listed in related figures is the antilog of the average of the individual common log values.

Several important results stemmed from this task time analysis Figure 19 shows the task time as a function of tasks with force reflection as a parameter. These numbers are an

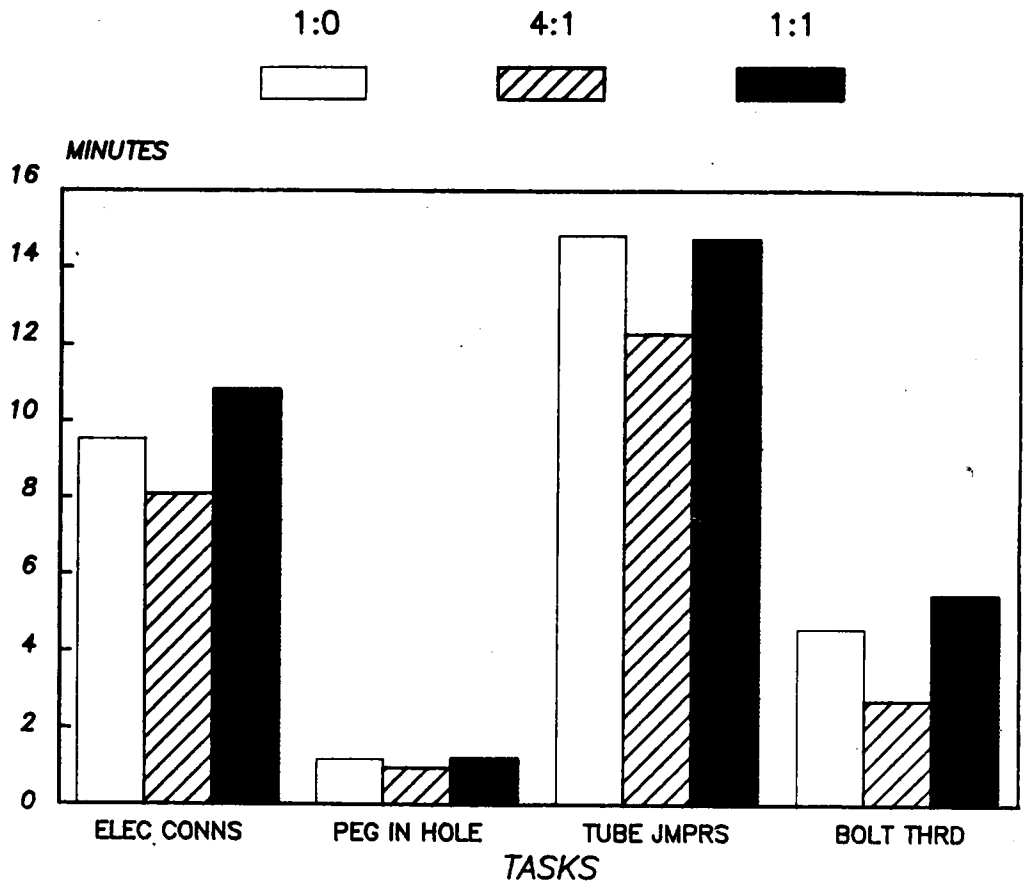


FIGURE 19. AVERAGE TASK COMPLETION TIME
(AVERAGED WITHIN EACH TASK)

average across all operators. Each bar represents the average of 60 task times, except for the bolt threading task, for which each bar is the average of 12 task times. It appears that the 4:1 level of force reflection allows for the quickest task completion times, with the 1:1 and 1:0 being quite similar in effects on task completion time. For three of the four tasks, the non-force reflection level results in a faster task completion than 1:1 levels. The tedious work involved in the tubing jumpers task, and a poor camera/lighting situation for the electrical connectors task could be viewed as possible causes for their increased task time over the other two tasks. For the electrical connectors, tubing jumpers, and bolt threading tasks, the difference between 4:1 and the other levels of force reflection was approximately 1.5 to 2.5 minutes, a high percentage of the total time for the bolt threading task. There was essentially no difference in the peg in hole task performance time as a function of force reflection level.

Figure 20 presents the task performance time data as a function of individual operators to reveal the effects of operator variation on task time. The figure represents the average of 32 task times for each force reflection for each operator. The results show that the 4:1 level was best for all but one operator. Four of the six operators performed better with 1:1 force reflection than in the absence of force reflection, although these values are close. It should be noted, also, that operator 4 , the

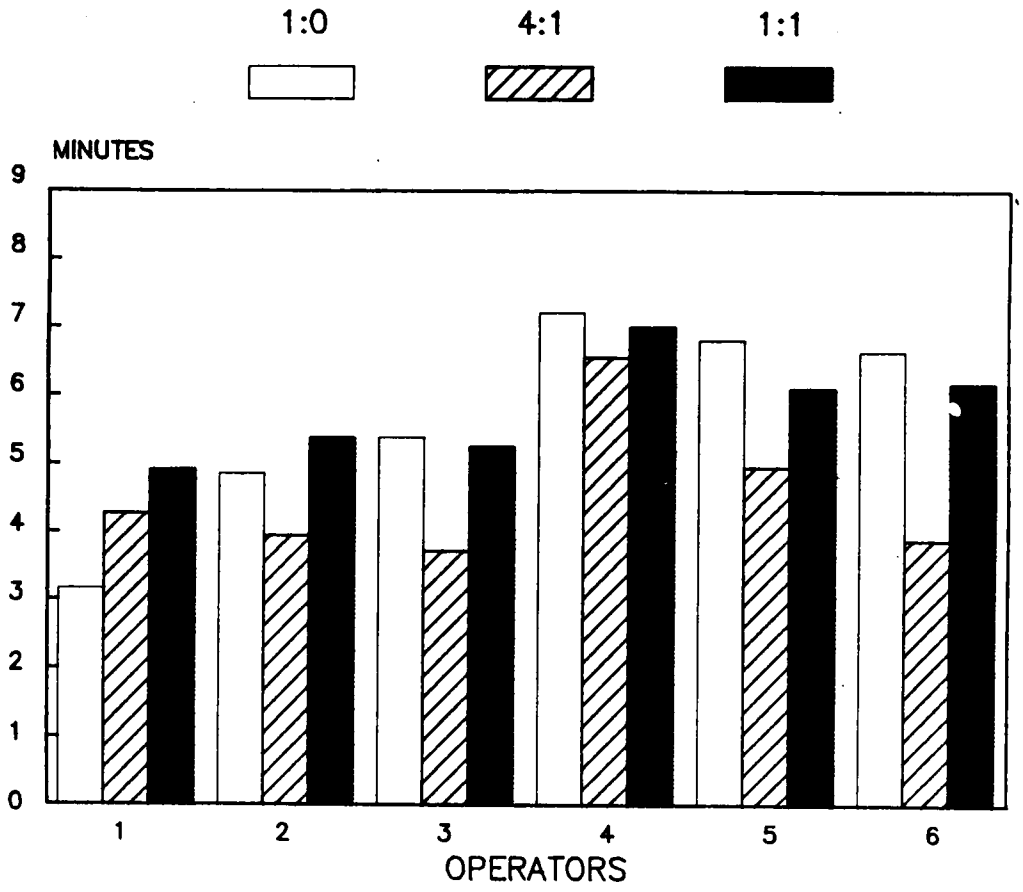


FIGURE 20. AVERAGE TASK COMPLETION TIME
(AVERAGED FOR EACH OPERATOR)

least experienced operator, averaged longer completion times at each level than did any other operator.

The effects of force reflection on overall task completion time, shown in Figure 21, are not surprising based on the results presented in Figures 19 and 20. This figure represents averages across all tasks and all operators, as a function of force reflection. Again, the intermediate force reflection level of 4:1 appears to be the best operating level of those studied.

Rate of Error Occurrence

Due to the large variation in task completion times between different tasks, a rate of error occurrence, as opposed to a pure number of errors per task, was implemented to better represent the accuracy or precision at which a task was performed. A list of errors tallied and their definitions can be found in Appendix C.

The effects of force reflection on error rate within each task is given in Figure 22. The numbers in this figure are averaged across all operators. This figure shows that the highest error rate occurs in the absence of force reflection, with varying effects in the 4:1 and 1:1 modes depending on the task. It is interesting to note that the task with the shortest average completion time, the peg-in-hole task, resulted in the highest rate of errors, while the longest task times, tubing jumpers,

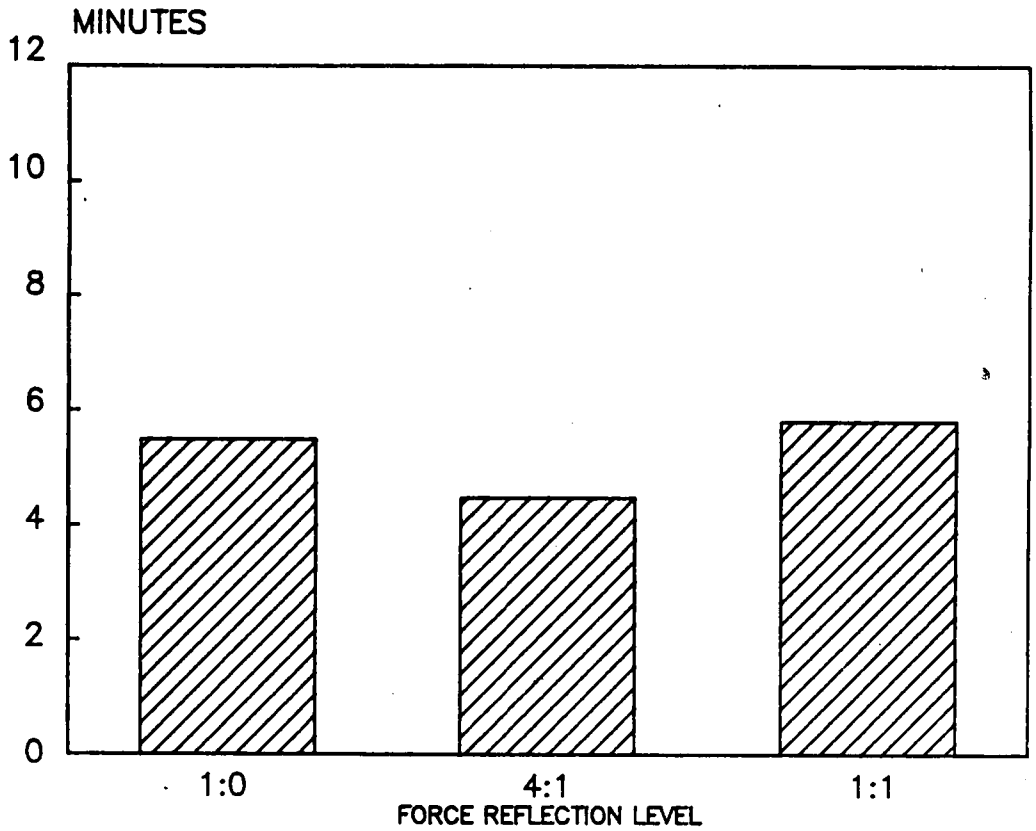


FIGURE 21. OVERALL TASK COMPLETION TIME
AS A FUNCTION OF FORCE REFLECTION LEVEL

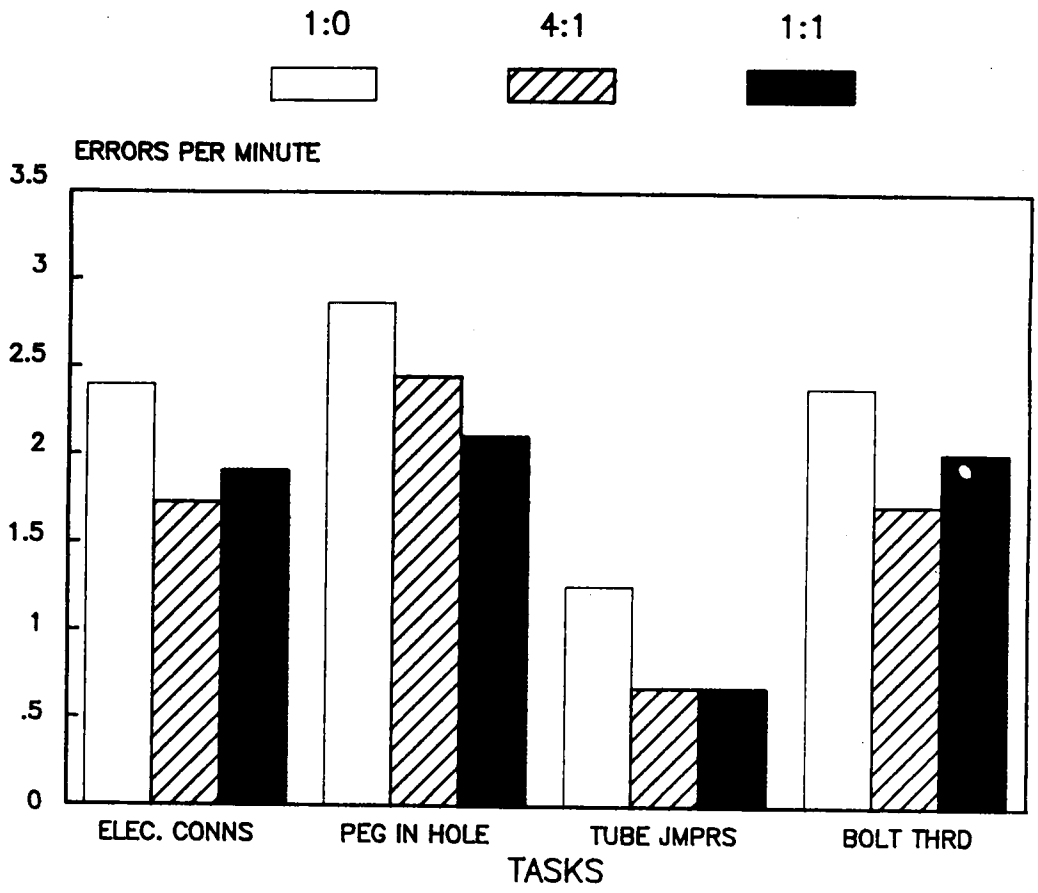


FIGURE 22. AVERAGE RATE OF ERROR OCCURRENCE
(AVERAGED WITHIN EACH TASK)

yielded the lowest error rate. It also appears that the differences are very small between 4:1 and 1:1 force reflection levels in the tubing jumpers task.

Figure 23 shows the operators reaction, in the form of error rate, to force reflection as an average across all tasks. While five of the six operators experienced their highest error rate with no force reflection, only two experienced their lowest error rate when operating at the 4:1 level. Again, the 4:1 and 1:1 levels are close to the same values with the 1:0 level data appearing much different. Operator 4, the least experienced with the longest task completion times, did not have the highest error rate at any level.

The overall effects of force reflection as averaged across all operators and tasks shown in Figure 24, repeat the general trend with the highest rate of errors occurring when there was no force reflection implemented. The 4:1 and 1:1 force reflection level error rate values are extremely close (1.63 errors/minute and 1.60 errors/minute for the 4:1 and 1:1 levels respectively).

Force and Moment Data

The data recorded from the load sensing platform were divided up in the same manner as that of the time and error rate data in order to present the effects of force reflection on forces and moments incurred for each task and again for individual

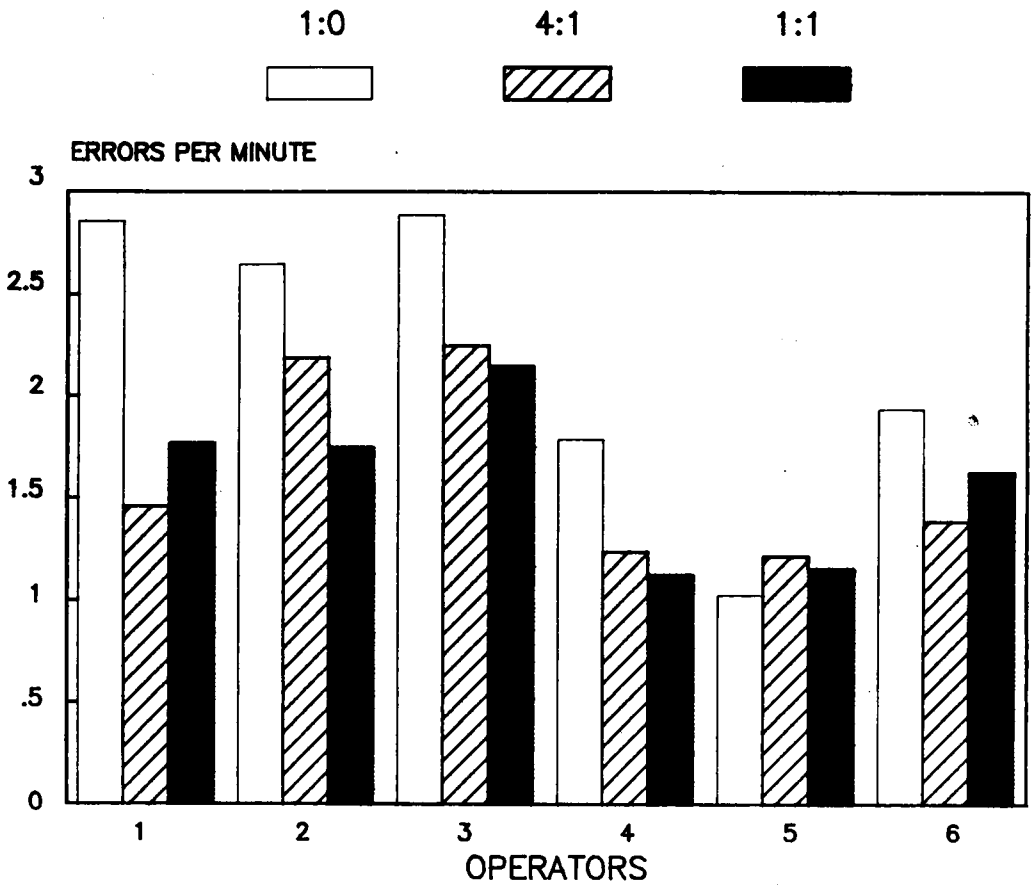


FIGURE 23. AVERAGE RATE OF ERROR OCCURRENCE
(AS OPERATOR AVERAGES)

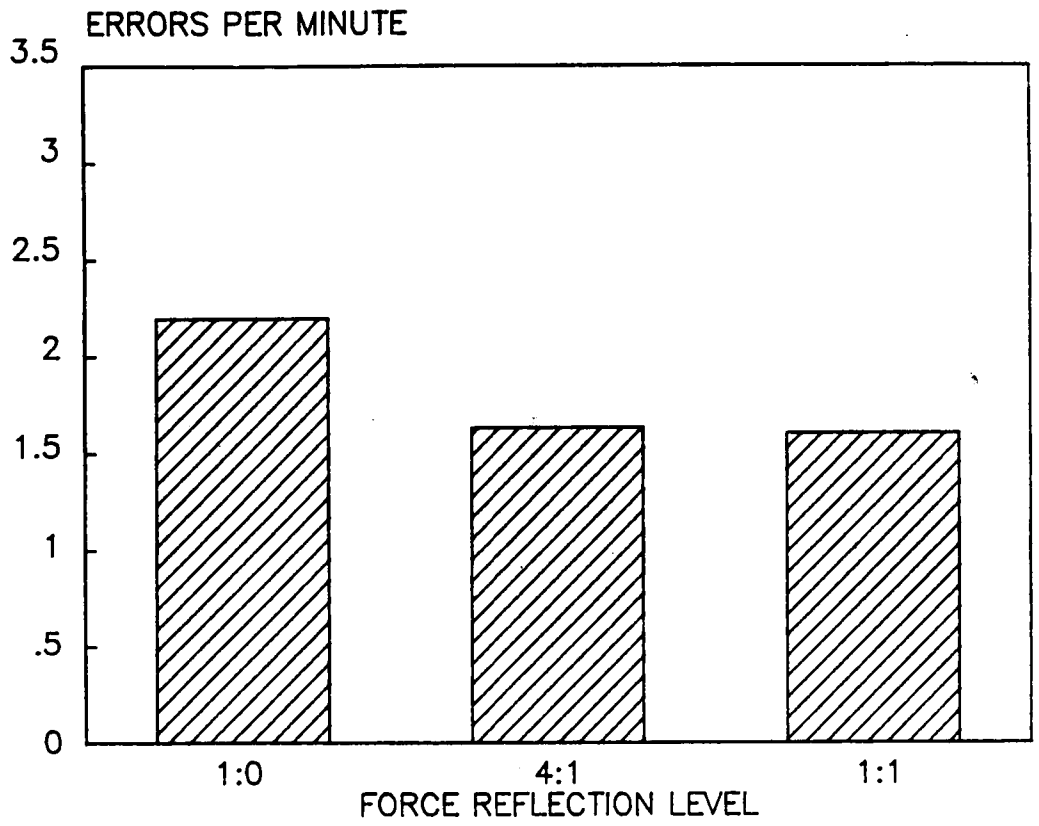


FIGURE 24. OVERALL RATE OF ERRORS AS A FUNCTION OF FORCE REFLECTION LEVEL

operator performances. For the ease of presentation, both the force and moment data are presented in terms of the magnitude of their resultants.

The force values shown in Figure 25 reflect the average of maximum force values averaged across all operators, within each task with force reflection as a parameter. The highest values of maximum force were the result of lack of force reflection for each task, while the lowest maximum forces were incurred when the operators were in a 1:1 force reflection mode. The tubing jumpers tasks yielded the highest average maximum forces at each level of force reflection, with a very small amount of difference (five pounds or less) within each force reflection level for the remaining three tasks. The peg-in-hole task and bolt threading task, quite similar in concept as they are both insertion as opposed to alignment tasks, show a great deal of similarity in the maximum forces incurred.

The data presented in Figure 26 emphasize these previously discussed trends. All six operators experienced the same general trend, though some were more drastic than others, averaging their highest maximum forces when there was no force reflection present and their lowest in the 1:1 force reflection condition. Operator 4, the least experienced, averaged the highest maximum force at each of the three force reflection levels.

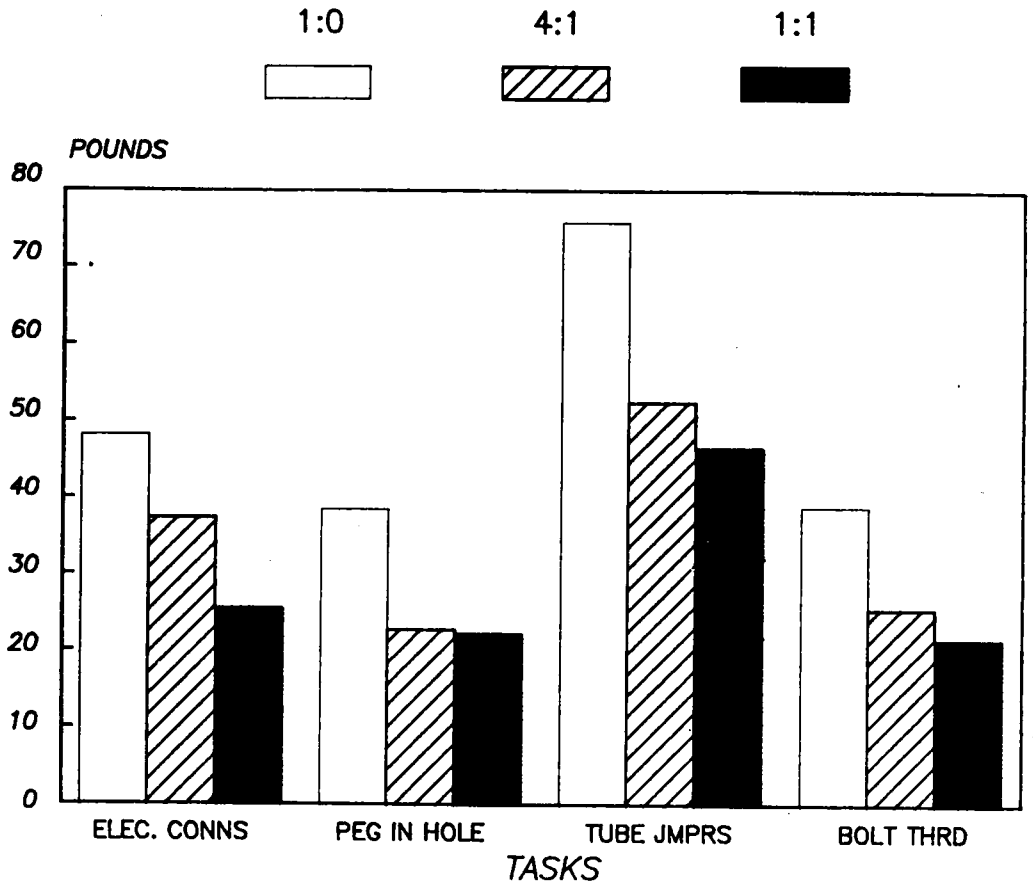


FIGURE 25. MAXIMUM FORCES
(AVERAGED WITHIN EACH TASK)

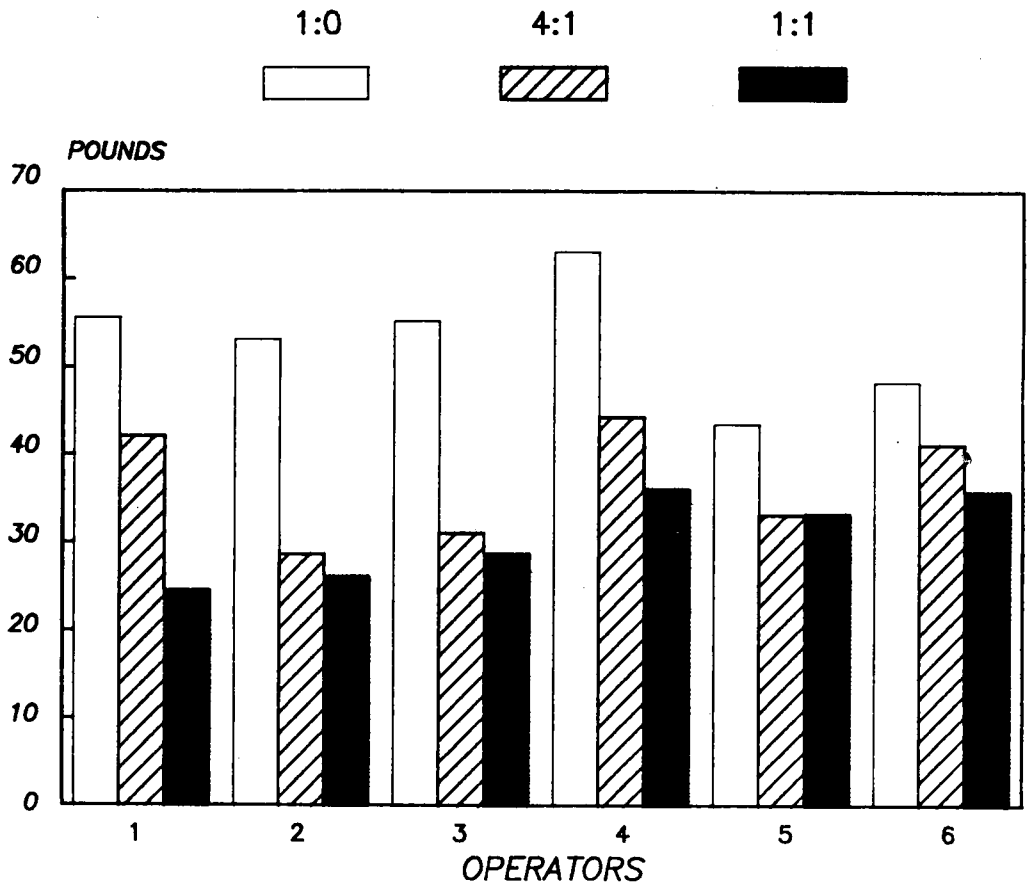


FIGURE 26. MAXIMUM FORCES
(AS OPERATOR AVERAGES)

At this point it is interesting to note that the absolute maximum force values (this is the absolute highest force incurred as opposed to the average maximum force discussed above) when broken into the same categories as presented in Figures 25 and 26, indicate almost identical trends. Only a slight shift upwards in scale (10 to 15 pounds) reflects the difference in the data.

Figures 27 and 28 reveal average force data, as averaged only while the manipulator was performing the task at the load table, not over the entire time of the task. These figures repeat the findings of the force reflection trends discovered in the maximum force figures. In Figure 27, the highest average forces occur in the absence of force reflection, while the lowest occur in a 1:1 force reflection mode. Figure 28 shows that each operator averaged their highest average forces with no force reflection present, and all but one operator performed best with a 1:1 force reflection ratio.

As expected, the moment data results echo those results obtained from the force data. Again, the highest moments in each task are generated when no force reflection is present, as seen in Figure 29. The 1:1 level offers the lowest maximum moments. Just as the highest forces were found in the tubing jumpers task, the highest moments are also found in that particular task.

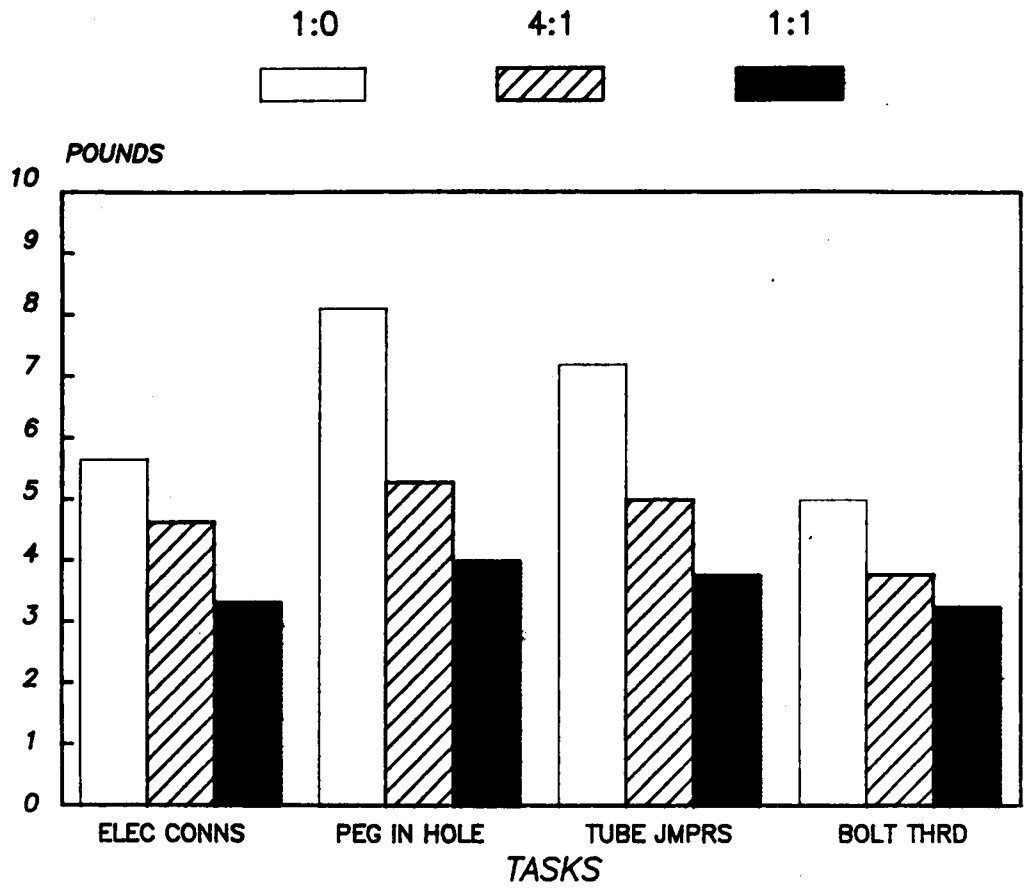


FIGURE 27. AVERAGE FORCES
(AVERAGED WITHIN EACH TASK)

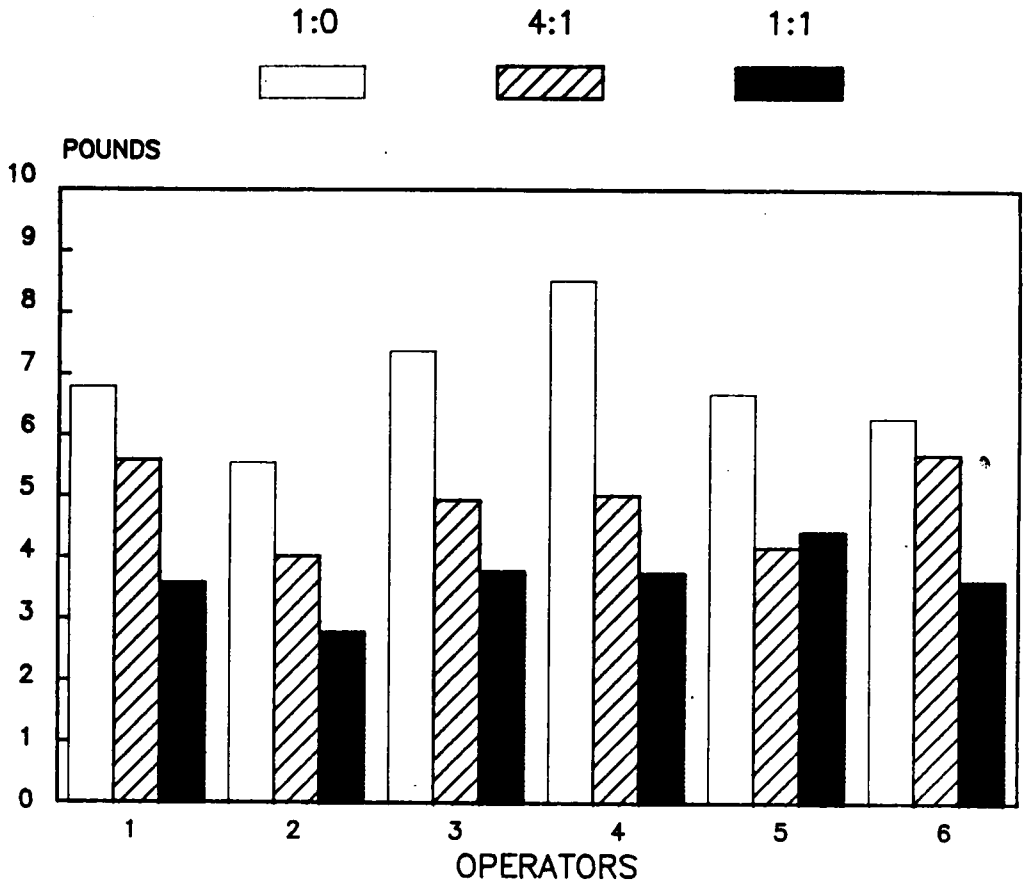


FIGURE 28. AVERAGE FORCES
(AS OPERATOR AVERAGES)

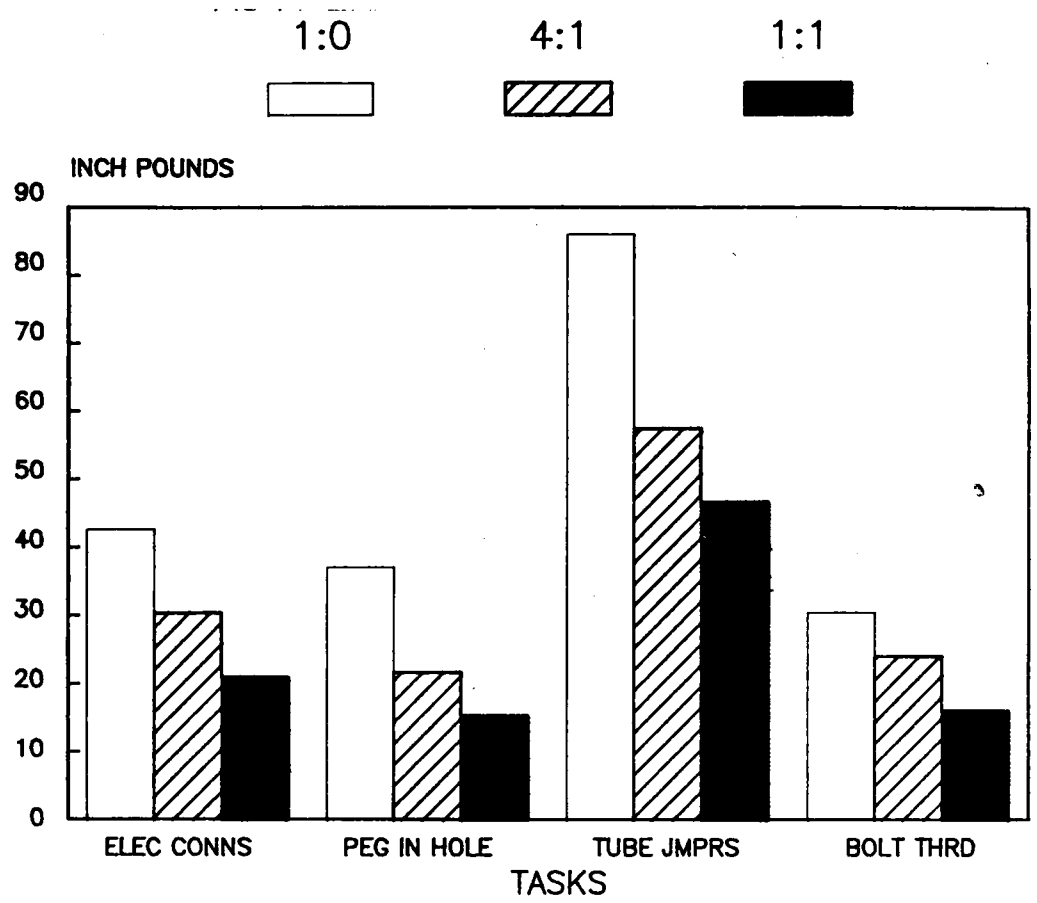


FIGURE 29. MAXIMUM MOMENTS
(AVERAGED WITHIN EACH TASK)

The effects of force reflection on the operator, shown in Figure 30, generally repeat the results found when comparing forces as a function of operator and force reflection. Generally, Operator 4 again had the highest average maximum moment results, although there is not an extreme difference at each level when comparing operators.

When comparing average moments, Figures 31 and 32, with the average force data discussed earlier, the electrical connectors and bolt threading tasks compare quite favorably. However, the peg-in-hole task which had the greatest forces at all levels takes a second place in-so-far as highest moments applied during the the tubing jumpers task. The tubing jumpers task required much more alignment and turning of the fittings into place which might explain the higher moments here.

The effects on the operator, Figure 32, reveal the same trends as the force data, with the highest moments occurring at a 1:0 force reflection level and the same five of six operators applying smaller moments when the force reflection was at a 1:1 level.

As previously discussed, the multi-sampled data presented in Figures 19 through 32 are to be subjected to a rather thorough statistical analysis through the use of two large statistical programs, MANOVA and ANOVA. Although this analysis, to be carried out by statisticians, is only in its initial stages of development, some of the preliminary results are of interest.

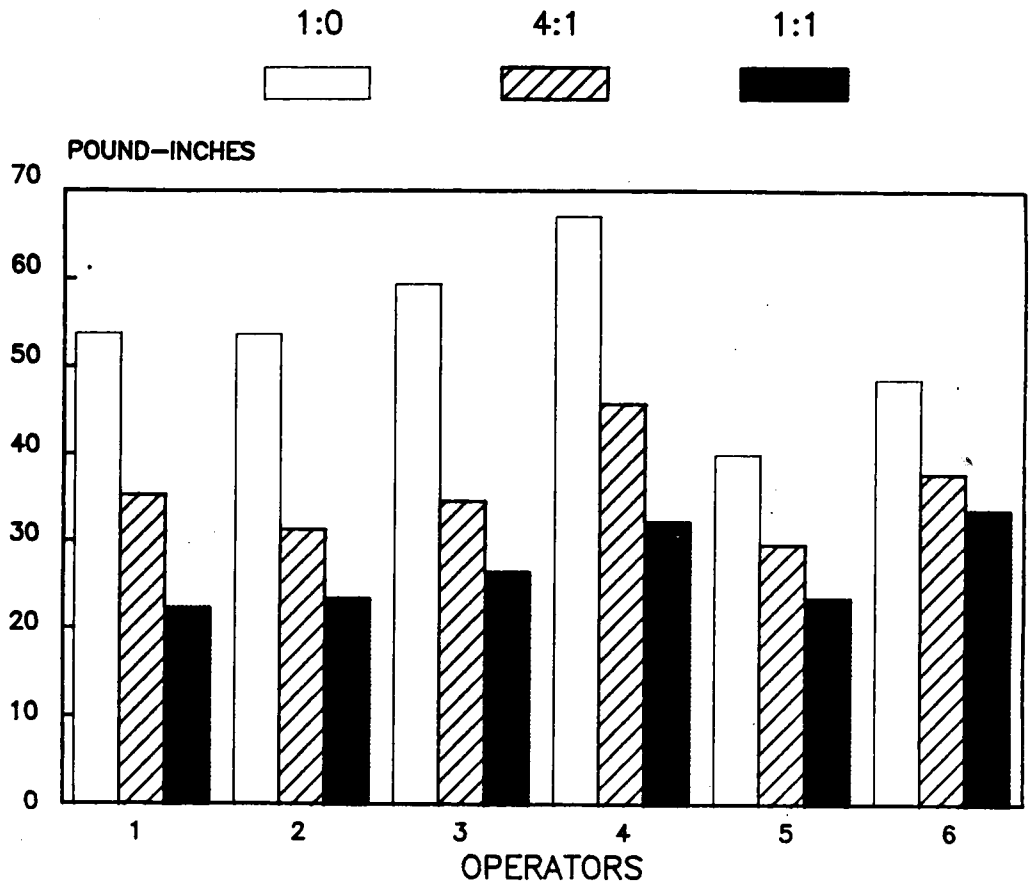


FIGURE 30. MAXIMUM MOMENTS
(AS OPERATOR AVERAGES)

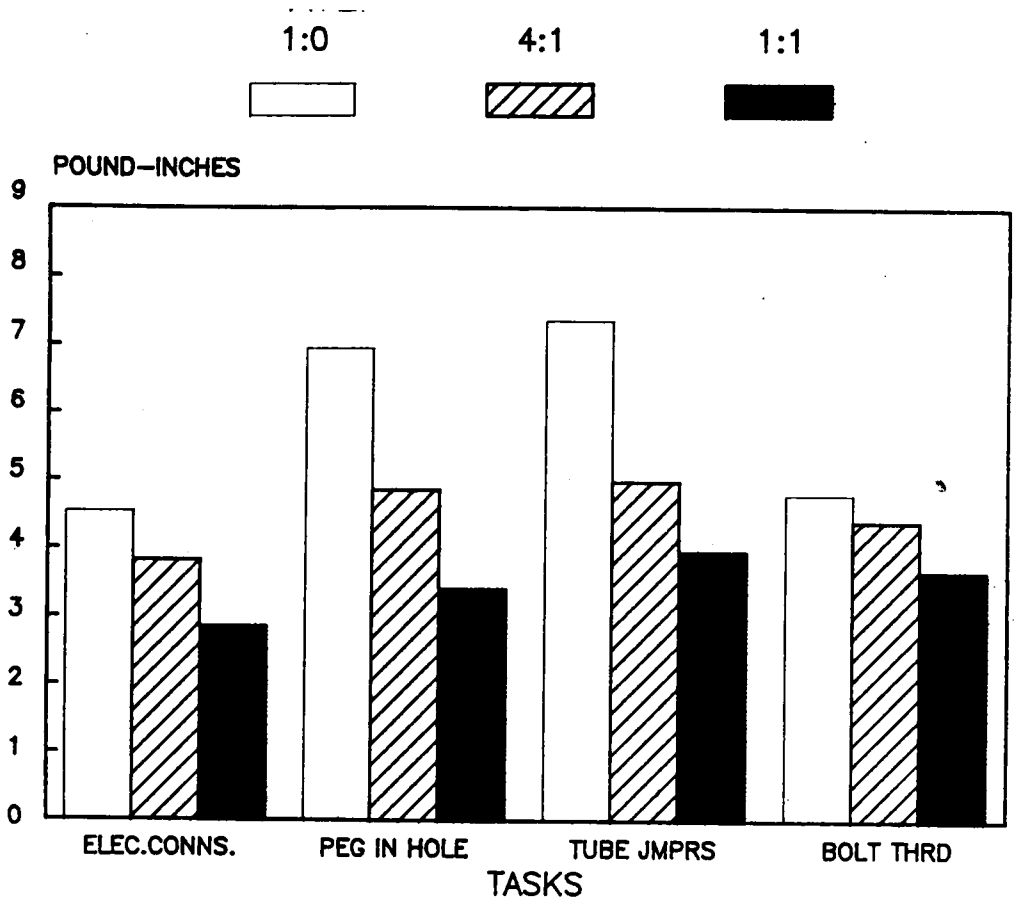


FIGURE 31. AVERAGE MOMENTS
(AVERAGED WITHIN EACH TASK)

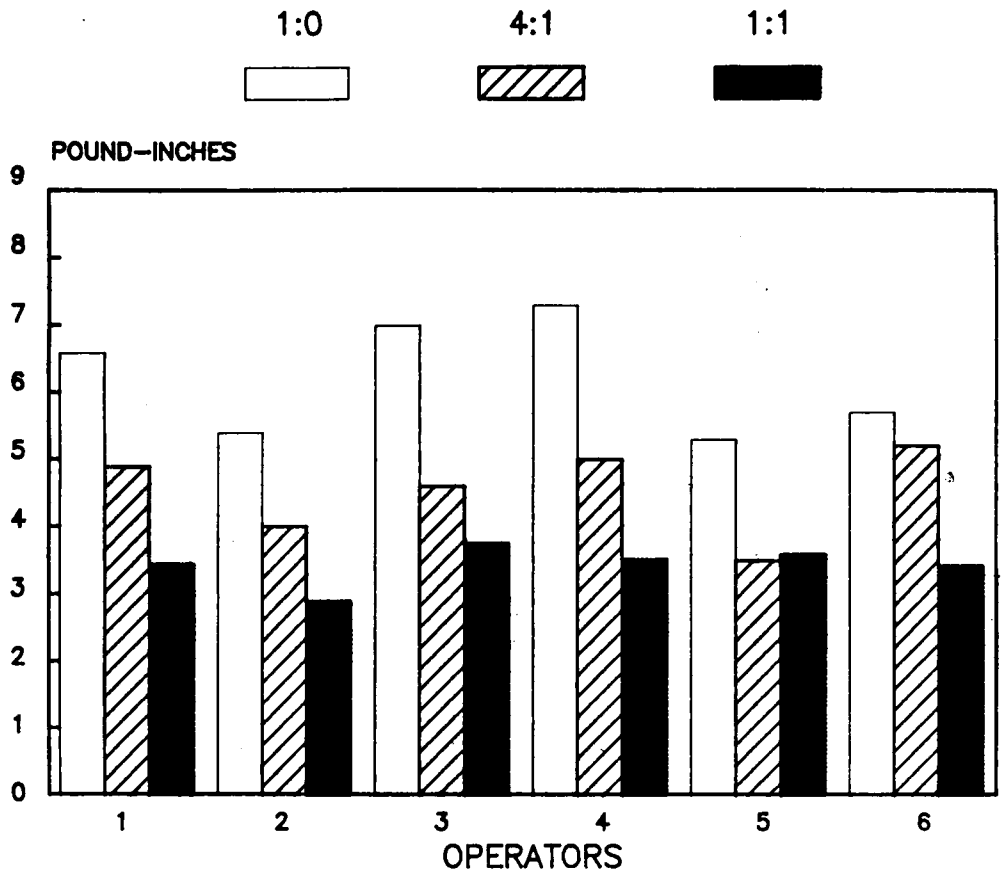


FIGURE 32. AVERAGE MOMENTS
(AS OPERATOR AVERAGES)

Figures 33 and 34 present the standard deviation of the average force incurred during a task for a particular force reflection level. Figure 34 presents this data as a function of the different types of tasks performed averaged across all operators, while Figure 33 yields average force standard deviation information averaged across all tasks as a function of individual operators.

Inspection of Figure 33 reveals that the highest standard deviation resulted from the zero force reflection mode of operation, while, for 3 of the 4 tasks, the 4:1 force reflection mode yielded the second highest standard deviation. Figure 34 indicates that the individual operators all experienced the highest standard deviation in their average force data for the non-force reflection mode, while all but one operator had their second highest standard deviation for the 4:1 force reflection condition. Information contained in these figures indicates that no force reflection to the operators results in a significantly larger variation of average forces applied to the task hardware while unity force reflection yields a much smaller root-mean-square deviation from the average force applied. These data indicate that the "best" force control is achieved for 1:1 force reflection with decreasing force control with decreasing force feedback.

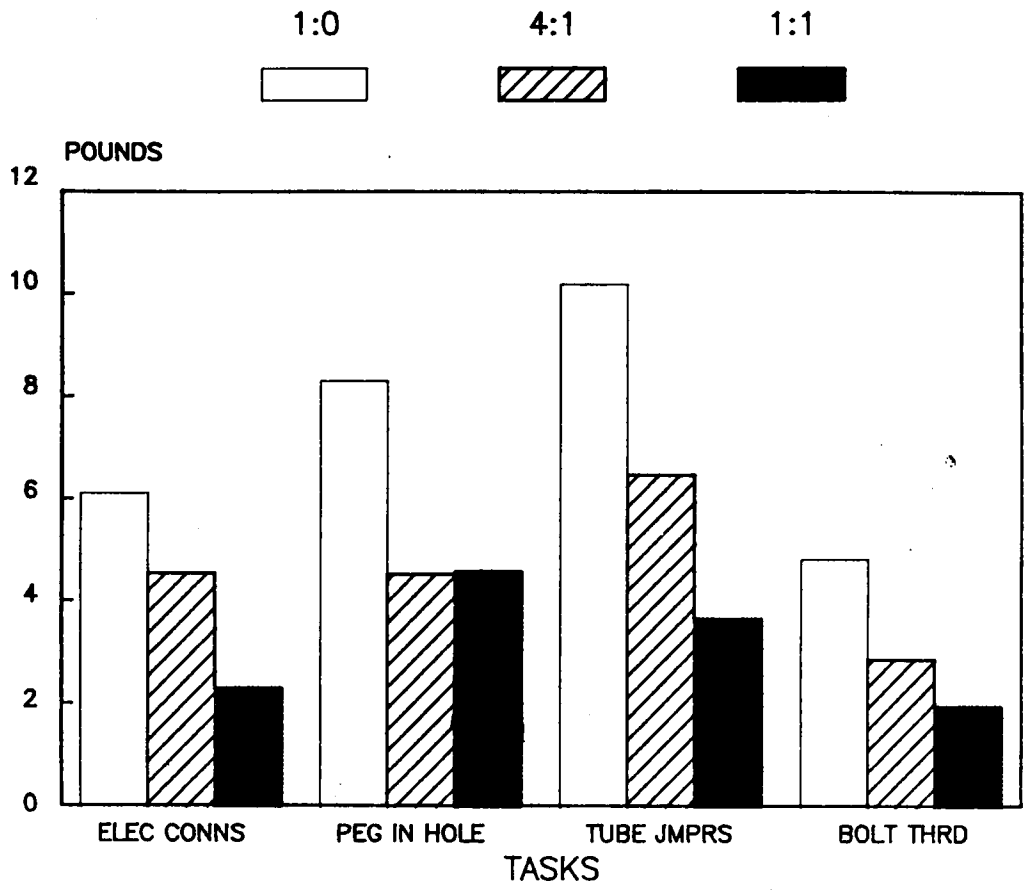


FIGURE 33. FORCE STANDARD DEVIATION
(AVERAGED WITHIN EACH TASK)

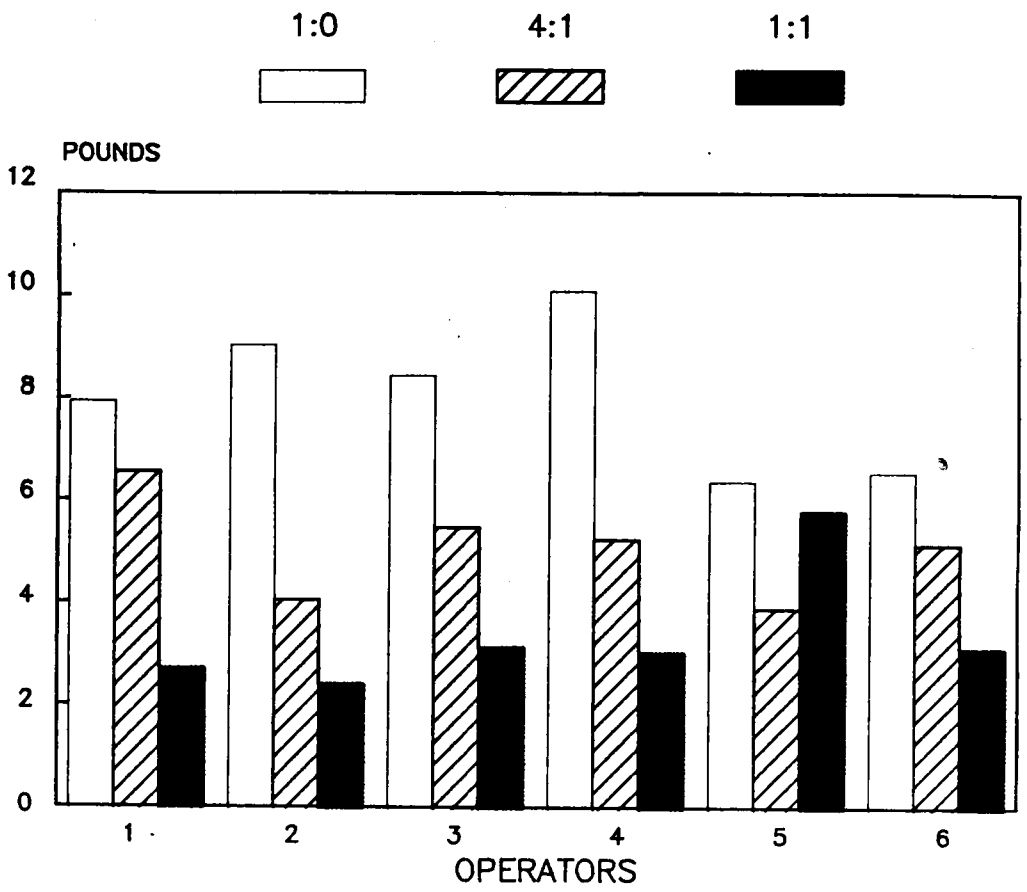


FIGURE 34. FORCE STANDARD DEVIATION
(AVERAGED FOR EACH OPERATOR)

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

These data support the hypothesis that force reflection can be beneficial for operators performing remote handling tasks. However, perfect correspondence between input forces and output forces, as in the 1:1 force reflection condition, may not be necessary. For these data, the forces fed back to the operator scaled down to one-fourth of the forces applied by the slave manipulator produced the shortest task completion time. This may be due to two related manipulator characteristics: the amount of information provided to the operator through force reflection and ease of use. In discussing some of the final results with the operators who performed the experiment, the response was that they were not surprised. Several operators complained during the test that the 1:1 level was extremely tiring, especially for the tubing jumpers task which proved to be somewhat long and difficult to perform. In 1:1 force feedback, as the operator tires, the data provide evidence to support the theory that the job takes longer and thus more mistakes are made than in an intermediate force reflection level, which could be critical in certain remote handling situations.

Given the relatively low sensitivity of the manipulators to forces (approximately one pound of force is necessary to overcome friction and inertia of the slave arms), the ability of

human operators to extract force information from the master controllers in the 4:1 condition is not greatly different from the 1:1 condition. This is based on the fact that the threshold sensitivity of the manipulator slave arms to force change is roughly one pound and the minImum force difference displayed by the master controller operating at the 4:1 level was one-fourth of a pound. This is well above the minimum human sensitivity to force changes, so there was no significant reduction in the overall (combined manipulator and human) force sensitivity in the 4:1 condition compared to the 1:1 condition. Complete absence of force reflection deprived the operators of all information about forces applied by the slave to the tasks. The presence of force information at the 4:1 level led to lower completion times and fewer errors accompanied by maximum and average forces which were much smaller than those with no force reflection information present.

Ease of use was best in the case of no force reflection, since the absence of reflected forces eliminated master friction and inertia, eliminated the need for operators to move loads, and thus reduced the potential for fatigue. Compared to the other force reflection levels in the study, friction, inertia, loads and fatigue potential were highest in the 1:1 condition. For the 4:1 condition, these effects were not high enough to offset the advantage of force information when compared to non-force reflection conditions, but they were reduced enough from the 1:1

condition to produce the performance advantage observed in the data. Comparing the 1:1 condition to the non-force condition, ease of use and force information differences between the two conditions were apparently equivalent in terms of their impact on task time, since no significant differences were observed between these conditions. For error rate, some operators were able to perform more effectively in the presence of force information, regardless of differences in ease of use. Other operators were sensitive to ease of use and force reflection on this criterion, and performed best in the 4:1 condition. In comparing the impact of force reflection when considering forces incurred, thus taking into account the environment and the forces which the task components themselves can withstand without damage, it is obvious that the lowest forces are uniformly a function of a 1:1 force reflection condition. Likewise, the highest forces are a factor when the operators are benefiting from the ease of use encountered with no force reflection. The 4:1 level seems to strike a happy medium, unless the lowest possible forces on the environment is a prime concern when dealing with fragile task elements.

These data indicate that force reflection is beneficial when it provides useful information to manipulator operators, but this advantage may be offset by degradation in manipulator ease of use. The data here seem to support the original hypotheses made about the importance of force reflection. First

of all, since the four tasks chosen were alignment and insertion tasks, it appears that force reflection is a valuable tool for these types of tasks. Secondly, the viewing was intentionally limited during the course of the experiment, which yielded substantial evidence supporting the importance of force reflection under suboptimal viewing conditions. The hypothesis relating the importance of force reflection to an operator learning a new task cannot be gathered from this data. However, it has been observed that operators working with the Model M-2 servomanipulator tend to implement a force reflection of 8:1 when they are faced with learning a new task. After becoming comfortable with the task at hand, the tendency is to then move to a non-force reflection condition, obviously for the ease of use.

For further study regarding the impact of force reflection in remote handling situation, it is recommended that other intermediate levels of force reflection be implemented in the experiment, perhaps 8:1 since it is a common force reflection level. More detailed study should also be performed in order to more fully understand the usefulness of force reflection when the operator is learning a task which is totally new, or when the operator is very inexperienced with servomanipulator control in general. A study of this nature would further refine the role of force reflection in remote handling.

LIST OF REFERENCES

LIST OF REFERENCES

1. Bertsche, W. R. and A. J. Pesch, "Investigation of Operator Performance and Related Design Variables in Undersea Force Feedback Manipulator Systems," Unpublished Manuscript, Prepared for the Office of Naval Research, June 1976.

2. Burgess, T. W., " The Remote Operation and Maintenance Demonstration Facility at the Oak Ridge National Laboratory," *Spectrum '86* American Nuclear Society International Topical Meeting, September, 1986.

3. Draper, J. V., " The Impact of Force Reflection on Remote Handling Task Performance, Test Plan," Oak Ridge National Laboratory, Fuel Recycle Division, Operating Procedures TP-FRT-001.

4. Draper, J. V. et al., "Final Report: Manipulator Comparative Testing Program," Oak Ridge National Laboratory Report ORNL/TM-10109, December 1986.

5. Feldman, M. J., el al., "European Foreign Trip Report," Oak Ridge National Laboratory Report, ORNL-STR-2103, 1985.

6. Feldman, M. J., et al., "Japan Foreign Trip Report," Oak Ridge National Laboratory Report, ORNL-STR-1813, 1984.

7. Flatau, C. R., "SM-229 - A New Compact ServoMaster-Slave Manipulator," *Proc. 25th Conf. Remote Systems Technology*, American Nuclear Society, p. 169, 1977.

8. Goertz, R. C., "Some Work on Manipulator Systems at ANL; Past Present, and a Look at the Future," *Proc .1964 Seminars on Remotely Operated Special Equipment*, USAEC Report CONF-641120, May 1964.

9. Hamel, W R., and H. L. Martin, "Robotics-Related Technology in the Nuclear Industry, " *Proc. of SPIE*, vol. 442, Robotics and Robot Sensing Systems, August 1983.

10. Herndon, J. N., et al., "The State-of-the-Art Model M-2 Maintenance System," *Proc. 1984 Topical Meeting on Robotics and Remote Handling in Hostile Environments*, American Nuclear Society, pp. 147-54, April, 1984.

11. Hill, J. W. and J. K. Salisbury, Jr., "Study to Design and Develop Remote Manipulator Systems," NASA Report NAS2-8652, November 1977.

12. Jackson, A. S., "FORTH Finds Favor in Real-Time Control," Machine Design, September 1986.

13. Kuban, D. P. and H. L. Martin, "An Advanced Remotely Maintainable Force Reflecting Servomanipulator Concept," *Proc. 1984 Topical Meeting on Robotics and Remote Handling in Hostile Environments*, American Nuclear Society, pp. 407-15, 1984.

14. Magdeleno, R. E. and D. T. McRuer, " Effects of Manipulator Restraints on Human Operator Performance," Air Force Flight Dynamics Laboratory Technical Report, AFFDL-TR-66-72, December, 1966.

15. Neter, J. and W. Wasserman, Applied Linear Statistical Models, Richard D. Irwin, 1974.

16. Martin, H. L.,et al., "Distributed Digital Processing for Servomanipulator Control," *Proc. 30th Conf. Remote Systems Technology*, American Nuclear Society, 1982.

17. Martin, H. L. and D. P. Kuban, ed., Teleoperated Robotics in Hostile Environments, Society of Manufacturing Engineers, 1985.

18. Rooks,Bryan, Ed., Developments in Robotics in 1983, IFS Publications Ltd., North Holland Publishing Company, 1983.

19. Vertut, J., et al., "The MA-23 Bilateral Servomanipulator System," *Proc. 24th Conf. Remote Systems Technology*, American Nuclear Society, pp. 125-133, 1976.

20. Vertut, J. and P. Coiffet, Teleoperations and Robotics: Evolution and Development, Robot Technology Vol. 3A, Prentice Hall, 1986.

21. Walters, S., "Synergy in Space-Man-Robot Cooperation," *Mechanical Engineering*, pp.26-37, Jan., 1985.

22. Wernli, R., "The Silent World of the Undersea Robot," *Decade of Robotics, Industrial Robot*, pp. 100-102, 1983.

23. White, J., et al., "Surveillance Robot for Nuclear Power Plants," *Proc. 33rd Conf. Remote Systems Technology*, American Nuclear Society, November, 1985.

APPENDICES

APPENDIX A
LIST OF EXPERIMENTAL HARDWARE

APPENDIX A

LIST OF EXPERIMENTAL HARDWARE

1. Controller: Hewlett-Packard 9836CS Data Acquisition System Controller. Color CR and Drivers, 8 slot backplane with addended 1 Megabyte memory. Serial# 2441A04316

2. Hard Disk: Hewlett-Packard 7942A Hard Disk/Tape System 24 Megabyte capacity plus cartridge tape drive. Serial# 2517A00494

3. Force/Torque Sensing Platform: Advanced Mechanical Technology, Inc. Model MC20-6-1000. 6 axis (3 orthogonal forces, 3 orthogonal torques) 1000 lb. capacity, axis cross talk less than 2%.

4. Signal Conditioner: Advanced Mechanical Technology, Inc. MCA-6, 6 channel strain measurement amplifiers, ± 10 Volt output to 2K ohm load. Sample Hold capability on all channels.

5. Accelerometers: Endevco Model 2233 Piezoelectric. Sensitivity 60 pC/g, nominal. Frequency response (± 5%) charge: 4 to 8000 Hz., nominal.

6. Charge Amplifiers: Endevco Model 2721A

7. Power Supply: Endevco Model 4222

8. Servomanipulator: Central Research Laboratories' Model M-2.

APPENDIX B
DOCUMENTED FORTH CODE FOR DATA REDUCTION AND ANALYSIS

APPENDIX B

DOCUMENTED FORTH CODE FOR DATA REDUCTION AND ANALYSIS

```
1 35 INDEX


 1
 2
 3 ( Documentation for Normalized Histogram Routine) ( 082Z86 WEM)
 4 ( Documentation for Normalized Histogram Routine) ( 082286 WEM)
 5 ( Normalized Histograms Load Screen)              ( 081986 WEM)
 6 ( Voltage Data Analysis "Constants")              ( 081986 WEM)
 7 ( Voltage Data Analysis "Constants")              ( 081986 WEM)
 8 ( Loading, calibrating, normalizing data array)   ( 081986 WEM)
 9 ( Histogram axis setup)   DECIMAL                 ( 082186 WEM)
10 ( Histogram Plotting)   DECIMAL                   ( 081986 WEM)
11
12
13 ( Documentation for Volts vs. Time Routines)      ( 082286 WEM)
14 ( Documentation for Volts vs. Time Routines)      ( 082286 WEM)
15 ( Load screen for volts vs. time plots)           ( 081986 WEM)
16 ( Voltage Data Constants for versus time array)   ( 081986 WEM)
17 ( Loading calib.data array for versus time plots) ( 082186 WEM)
18 ( Variable versus time axis setup)   DECIMAL      ( 081986 WEM)
19 ( Variable versus time plotting)   DECIMAL        ( 082186 WEM)
20 ( Variable versus time plotting)   DECIMAL        ( 040286 WEM)
21
22
23
24 ( Documentation Correct Units vs.Time Routines)   ( 082286 WEM)
25 ( Load screen, correct units var. vs. time plots) ( 081986 WEM)
26 ( Load and Tachometer Resistance constants)       ( 081986 WEM)
27 ( Correct units of force, moment, ang. velocity)  ( 052186 WEM)
28 ( Correct units of acceleration and current)      ( 042386 WEM)
29 ( Store corrected variable units in UNITSARRAY)   ( 091186 WEM)
30 ( Correct units versus time axis setup)           ( 081986 WEM)
31 ( Correct units versus time axis setup)   DECIMAL ( 081986 WEM)
32 ( Correct units variable versus time axis setup)  ( 082186 WEM)
33 ( Correct units variable versus time plotting)    ( 081986 WEM)
34
35
```

```
73 120 INDEX
            WEDNESDAY NOVEMBER 12 1986    3:16:14 AM

 73 ( Array and matrix filling load screen)                ( 080686 WEM)
 74 ( SIGNED 16-BIT.and SQRT Routines)                     ( 060386 WEM)
 75 ( Voltage Data Constants for versus time array)        ( 060386 WEM)
 76 ( Loading calibrated counts data array)                ( 081886 WEM)
 77 ( Load and Tachometer Resistance constants)            ( 051486 WEM)
 78 ( Correct units of force & moment)                     ( 082786 WEM)
 79 ( Correct units of acceleration and current)           ( 052186 WEM)
 80 ( Create all arrays for correct units variables)       ( 080686 WEM)
 81 ( Store corrected variable units)                      ( 072186 WEM)
 82 ( Store corrected variable units)                      ( 071686 WEM)
 83 ( Store corrected variable units in UNITSARRAY)        ( 072186 WEM)
 84 ( Words for filling resultant arrays)                  ( 071686 WEM)
 85 ( Commands for filling arrays)                         ( 082786 WEM)
 86 ( Setting array pointers)                              ( 081186 WEM)
 87 ( Setting pointers continued)                          ( 080686 WEM)
 88 ( Arrays of data)                                      ( 080686 WEM)
 89 ( Two Dim Array defined and zeroed)                    ( 080686 WEM)
 90 ( Matrix filling words)                                ( 061286 WEM)
 91 ( More matrix filling words)                           ( 080686 WEM)
 92 ( Do everything screen)                                ( 082786 WEM)
 93 ( Setting array pointers)                              ( 071686 WEM)
 94 ( Load screen for analysis of resultants only)         ( 082786 WEM)
 95 ( Begin of analysis of data arrays only)               ( 071686 WEM)
 96 ( Resultants only continued)                           ( 082786 WEM)
 97 ( Errorcode Array defined and zeroed)                  ( 071186 WEM)
 98 ( Resultants only analysis continued)                  ( 071086 WEM)
 99 ( Resultants analysis continued)                       ( 082786 WEM)
100 ( ** VIRTUAL Memory load screen              ** )      ( 040786 BSW)
101 ( ** VIRTUAL Memory load screen              ** )      ( 040786 BSW)
102 ( Virtual Memory Definition)                           ( 121385 BSW)
103 ( Virtual fetch & store primitives)                    ( 103185 BSW)
104 ( Virtual fetch & store primitives)                    ( 041686 WEM)
105 ( Virtual Memory Access and Comma)                     ( 110485 BSW)
106 ( Virtual text & move words)                           ( 060486 WEM)
107
108
109
110
111
112
113
114
115 ( Individual task listings w/addresses)                ( 091286 WEM)
116 ( Individual task listings w/addresses)                ( 081886 WEM)
117 ( Individual task listings w/addresses)                ( 081886 WEM)
118 ( Individual task listings w/addresses)                ( 081886 WEM)
119 ( Listing of items to be analyzed, in order)          ( 091286 WEM)
120 ( Listing of items to be analyzed, in order)          ( 081886 WEM) ok
```

```
36 72 INDEX
           WEDNESDAY NOVEMBER 12 1986    3:15:31 AM

 36
 37
 38
 39
 40
 41
 42
 43
 44
 45 ( Load screen for Resultants vs. Time Routines)   ( 090786 WEM)
 46 ( Voltage Data Constants for versus time array)   ( 091286 WEM)
 47 ( Correct units of force & moment)                ( 091186 WEM)
 48 ( Store corrected variable units)                 ( 090786 WEM)
 49 ( Words for filling resultant arrays)             ( 090786 WEM)
 50 ( Loading calibrated counts data array)           ( 091186 WEM)
 51 ( Setting array pointers)                         ( 091286 WEM)
 52 ( Resultants only continued)                      ( 091286 WEM)
 53 ( Resultants versus time axis setup)              ( 091186 WEM)
 54 ( Resultants versus time axis setup)   DECIMAL    ( 090986 WEM)
 55 ( Resultants versus time axis setup)              ( 091286 WEM)
 56 ( Resultants versus time plotting)                ( 091286 WEM)
 57
 58
 59
 60 ( 3D Histogram Plotting) DECIMAL                  ( 050786 WEM)
 61
 62
 63
 64
 65
 66 ( Documentation: Resultants, Errors, and Matrix)  ( 082786 WEM)
 67 ( Documentation: Resultants, Errors, and Matrix)  ( 082786 WEM)
 68 ( Documentation: Resultants, Errors, and Matrix)  ( 082286 WEM)
 69 ( Documentation: Resultants, Errors, and Matrix)  ( 082286 WEM)
 70 ( Documentation: Resultants, Errors, and Matrix)  ( 082786 WEM)
 71
 72   ok
```

```
3 LIST
SCR # 3     WEDNESDAY NOVEMBER 12 1986     3:19:00 AM
 0 ( Documentation for Normalized Histogram Routine) ( 082286 WEM)
 1
 2      Screens 5 thru 10 contain software written for
 3  obtaining normalized histogram plots for any of the variables
 4  measured during testing.  The fact that the plots have been
 5  normalized keeps the actual task time independent of the
 6  final outcome of the histograms, keeping %time as an indepen-
 7  dent variable. Key words include DATA.IN which reads, calib-
 8  rates, and normalizes the data.  This word requires a variable
 9  constant, a starting address, and an ending address from the
10  hard disk.  These addressses can be found in the directory of
11  tasks at the beginning of every tape.
12       Histogram axes run from -4095 to +4095 on the horizontal
13  axis with %time on the vertical axis.  The variable SCALE
14  allows scaling of the %time axis.  A value of 10 stored in
15  SCALE will show a max %time of .3%, while a value -->CONT.
ok
```

```
4 LIST
SCR # 4     WEDNESDAY NOVEMBER 12 1986     3:19:27 AM
 0 ( Documentation for Normalized Histogram Routine) ( 082286 WEM)
 1
 2   of 100 stored in SCALE will provide a maximum %time of 3.0%,
 3   etc.
 4       Another key word is NEW.PLOT.  This word requires 2 values
 5   on the stack.  The first, low, refers to the value on the
 6   far left end of the horizontal axis.  The second, inc, is the
 7   increment between the horizontal axis ticks.  There are both
 8   major and minor axis ticks on the horiz. axis.  With an inc of
 9   1, the increment between major ticks is 100.  An inc of 2 will
10   show the gap to be 200, while an inc. of 3 yields a gap of 300
11   counts.
12       The label "Voltage/.00244" is a reminder that the horiz.
13   axis is merely a representation of calibrated counts corres-
14   ponding to the signal from the A/D converter. For information
15   concerning conversion to proper units, refer to documentation
ok
```

```
5 LIST
SCR # 5     WEDNESDAY NOVEMBER 12 1986    3:20:13 AM
  0 ( Normalized Histograms Load Screen)              ( 081986 WEM)
  1
  2    100 LOAD          ( Virtual memory defining words)
  3      6 LOAD          ( "Constants" for variable access and calib.)
  4      7 8 THRU        ( Load, Calibrate, and Normalize data array)
  5      9 LOAD          ( Histogram axis setup)
  6     10 LOAD          ( Plotting histograms)
  7
  8
  9
 10
 11
 12
 13
 14
 15
 ok
6 LIST
SCR # 6     WEDNESDAY NOVEMBER 12 1986    3:20:38 AM
  0 ( Voltage Data Analysis "Constants")              ( 081986 WEM)
  1
  2 : VELX 10 6 8 ;     : VELY 12 6 8 ;    : VELZ 14 6 8 ;
  3 : VELWR 16 6 8 ;    : VELWL 18 6 8 ;   : VELAZ 20 6 8 ;
  4 : VELT 22 6 8 ;
  5
  6 : FX 28 24 26 ;     : FY 30 24 26 ;    : FZ 32 24 26 ;
  7 : MX 34 24 26 ;     : MY 36 24 26 ;    : MZ 38 24 26 ;
  8
  9 : ACCX 46 42 44 ;       : ACCY 48 42 44 ;    : ACCZ 50 42 44 ;
 10 : CURRZ 52 42 44 ;      : CURRY 54 42 44 ;
 11 : CURRWL 56 42 44 ;     : CURRWR 58 42 44 ;
 12
 13
 14
 15
 ok
7 LIST
SCR # 7     WEDNESDAY NOVEMBER 12 1986    3:20:52 AM
  0 ( Voltage Data Analysis "Constants")              ( 081986 WEM)
  1 VARIABLE SUM
  2
  3 CREATE READARRAY 8192 4* ALLOT
  4 CREATE NORMARRAY 8192 4* ALLOT
  5
  6 : ZERRO 8192 0 DO 0 READARRAY I 4* + ! LOOP ;
  7 : ZR 8192 0 DO 0 NORMARRAY I 4* + ! LOOP ;
  8
  9
 10 : U.R ( d n) >R  <# #S #> R> OVER - SPACES TYPE ;
 11 : REPORT 8191 0 DO I 4* READARRAY  + @ 5 U.R CR 20 +LOOP ;
 12
 13 : SUMMING 0 SUM ! 8192 0 DO I 4* READARRAY + @ SUM +! LOOP ;
 14
 15
```

```
8 LIST
SCR # 8     WEDNESDAY NOVEMBER 12 1986    3:21:23 AM
  0 ( Loading, calibrating, normalizing data array)   ( 081986 WEM)
  1 CODE CALIB ( hi lo n  -- n' ) SP )+ D0 LONG MOVE, SP )+ D1
  2    LONG MOVE,  SP )+ D2 LONG MOVE, 4095 # D3 LONG MOVE,
  3    D3 D4 LONG MOVE,  D3 TO D1 LONG AND,  D2 TO D3 LONG AND,
  4    D1 TO D3 LONG SUB,  D2 4095 LONG CMPI,  GT IF, D3 LONG NEG,
  5    THEN, D3 08 # LONG ASR,  D3 01 # LONG ASR, D0 TO D4 LONG AND,
  6    D1 TO D4 LONG SUB,  D0 4095 LONG CMPI,  GT IF,  D4 LONG NEG,
  7    THEN, D3 TO D4 LONG SUB, D4 SP -) LONG MOVE,  NEXT  END-CODE
  8
  9 : READ  ( constant from scr#6, s f --) 1024 -   ZERRO
 10    SWAP DO DUP I + VW@ 2OVER I + VW@ SWAP I + VW@ CALIB
 11    DUP 0< IF NEGATE 4096 + THEN 8191 MIN 4* READARRAY +
 12    1 SWAP +! 60 +LOOP DROP   2DROP   7 EMIT ;
 13 : NORMALIZE ZR SUMMING SUM @ 8192 0 DO DUP I 4* READARRAY + @
 14    100000 ROT */ I 4* NORMARRAY + ! LOOP DROP 7 EMIT 7 EMIT ;
 15 : DATA.IN ( const. from scr#6, s,f--)   READ   NORMALIZE ;
ok
9 LIST
SCR # 9     WEDNESDAY NOVEMBER 12 1986    3:24:02 AM
  0 ( Histogram axis setup)   DECIMAL                ( 082186 WEM)
  1 50 CONSTANT X0 50 CONSTANT Y0 VARIABLE PEN-VAL  15 PEN-VAL !
  2 VARIABLE XPOS  VARIABLE SCALE 100 SCALE ! VARIABLE TOTAL
  3 : ORIG X0 Y0 START ;
  4 : LAB-X PROP Y0 20 - Y.CURSOR ! 5 0 DO I 100 * X0 + X.CURSOR !
  5   OVER + DUP . LOOP 2DROP REG ;
  6 : LAB-Y PROP 4 0 DO 10 X.CURSOR ! I SCALE @ * I 100 * Y0
  7   + Y.CURSOR ! <# # # 46 HOLD #S #> TYPE LOOP REG ;
  8 : TICK ORIG 45 0 DO I 10 * DUP 0 RSTART DUP 100 /MOD DROP
  9   0= IF -10 ELSE -5 THEN RDRAW LOOP
 10   31 0 DO I 10 * 0 OVER RSTART DUP 100 /MOD DROP 0= IF -10
 11   ELSE -5 THEN SWAP RDRAW LOOP ;
 12 : AXIS 10 PEN ORIG 500 Y0 DRAW X0 Y0 START X0 350 DRAW ;
 13 : %TOTAL 0 0 TOTAL ! 8192 0 DO I 4* NORMARRAY + @ + LOOP
 14   TOTAL ! ;
 15 : .TOTAL TOTAL @ <# 37 HOLD 32 HOLD # # # 46 HOLD #S #> TYPE ;
ok
10 LIST
SCR # 10    WEDNESDAY NOVEMBER 12 1986    3:24:20 AM
  0 ( Histogram Plotting)   DECIMAL                  ( 081986 WEM)
  1 CREATE G-LABEL 40 ALLOT
  2 : +DOTS ( val) PEN-VAL @ PEN XPOS @ DUP Y0 START   SWAP
  3   Y0 + DRAW  1 XPOS +! ;
  4 : @VAL ( n-n) DUP 0< IF NEGATE 4095 + THEN 4* NORMARRAY + @ ;
  5 : PLOTHIST ( low inc) SWAP 450 0 DO OVER + DUP @VAL
  6   10 SCALE @  */ 300 MIN +DOTS LOOP 2DROP ;
  7 : PLOT.DATA ( low inc) 2DUP X0 XPOS ! AXIS PLOTHIST AXIS TICK
  8   100 * SWAP OVER -   LAB-X   LAB-Y ;
  9 : NEW.PLOT ( low inc)  %TOTAL
 10   G-LABEL 40 32 FILL CR ." Name of graph: " G-LABEL 40 EXPECT
 11   XALPHA CLR.GRAPH PLOT.DATA 10 PEN PROP
 12   5 X.CURSOR ! 200 Y.CURSOR ! ." %Time"
 13   10 Y.CURSOR ! 200 X.CURSOR ! ." Voltage/.00244 "
 14   130 X.CURSOR ! 360 Y.CURSOR ! G-LABEL
 15   40 TYPE 10 Y.CURSOR ! 400 X.CURSOR ! .TOTAL   REG ;
```

```
13 LIST
SCR # 13    WEDNESDAY NOVEMBER 12 1986    3:24:56 AM
  0 ( Documentation for Volts vs. Time Routines)      ( 082286 WEM)
  1
  2     The following routines, screens 15 thru 20, enable the
  3  user to view the behavior of a variable as a function of time.
  4  In order to do this, the values taken on by a given variable
  5  within a starting and ending address are read from the hard
  6  disk, calibrated, and stored in an array.  The reading is still
  7  in "volts" at this time.  The key word VTIME is responsible for
  8  filling the array with the calibrated data.  VTIME requires 3
  9  values on the stack.  First of all, a constant from screen 15,
 10  which determines which variable will be read.  Second and third
 11  on the stack are the starting and ending addresses on the
 12  hard disk from which the data will be read.  These addresses
 13  can be found in the directory of the tape (screens 2272, 2273,
 14  etc.).   (continued)
 15
 ok
```

```
14 LIST
SCR # 14    WEDNESDAY NOVEMBER 12 1986    3:25:25 AM
  0 ( Documentation for Volts vs. Time Routines)      ( 082286 WEM)
  1
  2    Following filling the data array, the word NEW.PLOT
  3  enables the user to view the behavior on the monitor.  This
  4  command requires 2 items on the stack.  First, a beginning time
  5  and secondly, and increment for the time axis.  Since the data
  6  was recorded at 20 hz., the beginning time for the plot should
  7  be (#seconds)x(20 scans/sec) which will index into the data
  8  file the proper amount.  Two variables, XINC and SCALE, allow
  9  proper scaling.  SCALE is responsible for scaling the vertical
 10  axis ( #volts max) while XINC scales the time or horizontal
 11  axis.  Note that (XINC)*(Plotting increment, inc.)*(.05)= the
 12  amount of time in seconds between 2 major horizontal axis tick
 13  marks. In order to printout the graphics, hit the DUMP GRAPHICS
 14  command (shift, GRAPHICS).  To clear the plot, type CLR.GRAPH.
 15
 ok
```

```
15 LIST
SCR # 15    WEDNESDAY NOVEMBER 12 1986    3:25:58 AM
  0 ( Load screen for volts vs. time plots)          ( 081986 WEM)
  1
  2     100 LOAD          ( Virtual memory defining.words)
  3      16 LOAD          ( Voltage Data Analysis "Constants")
  4      17 LOAD          ( Loading Calib. data for vs time plots)
  5      18  LOAD         ( Variable versus time axis setup)
  6      19 20 THRU       ( Variable versus time plotting)
  7
  8
  9
 10
 11
 12
 13
 14
 15
 ok
16 LIST
SCR # 16    WEDNESDAY NOVEMBER 12 1986    3:26:18 AM
  0 ( Voltage Data Constants for versus time array)   ( 081986 WEM)
  1
  2 : Velx 10 6 8 ;     : Vely 12 6 8 ;    : Velz 14 6 8 ;
  3 : Velwr 16 6 8 ;    : Velwl 18 6 8 ;   : Velaz 20 6 8 ;
  4 : Velt 22 6 8 ;
  5
  6 : Fx 28 24 26 ;     : Fy 30 24 26 ;    : Fz 32 24 26 ;
  7 : Mx 34 24 26 ;     : My 36 24 26 ;    : Mz 38 24 26 ;
  8
  9 : Accx 46 42 44 ;       : Accy 48 42 44 ;    : Accz 50 42 44 ;
 10 : Currz 52 42 44 ;      : Curry 54 42 44 ;
 11 : Currwl 56 42 44 ;     : Currwr 58 42 44 ;
 12
 13
 14
 15
 ok
17 LIST
SCR # 17    WEDNESDAY NOVEMBER 12 1986    3:26:34 AM
  0 ( Loading calib.data array for versus time plots) ( 082186 WEM)
  1  CREATE VTIMEARRAY 80000 4* ALLOT
  2 : ZZZ 80000 0 DO 0 VTIMEARRAY I 4* + ! LOOP ;
  3 CODE CALIBRATE ( hi lo n  -- n' ) SP )+ D0 LONG MOVE,
  4      SP )+ D1 LONG MOVE,  SP )+ D2 LONG MOVE,
  5      4095 # D3 LONG MOVE,  D3 D4 LONG MOVE,  D3 TO D1 LONG AND,
  6      D2 TO D3 LONG AND,  D1 TO D3 LONG SUB,  D2 4095 LONG CMPI,
  7      GT IF,  D3 LONG NEG,  THEN,  D3 08 # LONG ASR,
  8      D0 TO D4 LONG AND, D1 TO D4 LONG SUB,  D0 4095 LONG CMPI,
  9      GT IF,  D4 LONG NEG,  THEN,  D3 TO D4 LONG SUB,
 10      D4 SP -) LONG MOVE,  NEXT  END-CODE
 11 VARIABLE VCOUNT
 12 : VTIME (  constant from scr#16,s,f  --) 1024 -   0 VCOUNT !
 13    ZZZ   SWAP DO DUP I + VW@ 2OVER I + VW@ SWAP I + VW@
 14    CALIBRATE VCOUNT @ 4* VTIMEARRAY + ! 1 VCOUNT +! 60 +LOOP
 15    DROP  2DROP   7 EMIT ;
```

```
18 LIST
SCR # 18    WEDNESDAY NOVEMBER 12 1986    3:27:07 AM
  0 ( Variable versus time axis setup)   DECIMAL       ( 081986 WEM)
  1 50 CONSTANT X0 200 CONSTANT Y0 VARIABLE PEN-VAL  15 PEN-VAL !
  2 VARIABLE XPOS  VARIABLE SCALE 1000 SCALE !  100 CONSTANT XINC
  3 ( Note: [XINC]*[Plotting increment, inc]*[.050]= Amount of time
  4     in seconds between 2 major x-axis tick marks.)
  5
  6 : ORIG X0 Y0 START ;
  7
  8 : LAB-X PROP Y0 20 - Y.CURSOR ! 450 XINC / 0 DO I XINC * X0
  9       + X.CURSOR ! OVER + DUP 10 / <# # # 46 HOLD #S #>
 10     TYPE LOOP   2DROP   REG ;
 11 : LAB-Y PROP 2 0 DO 10 X.CURSOR ! I SCALE @ * 10 1000 */
 12   I 100 * Y0 + Y.CURSOR ! <# # 46 HOLD #S #> TYPE LOOP
 13   -1 -1 DO 5 X.CURSOR ! I SCALE @ * 10 1000 */ I 100 * Y0 +
 14   Y.CURSOR ! DUP ABS <# # 46 HOLD #S  DROP SIGN #> TYPE
 15   -1 +LOOP    DROP   REG ;
 ok
19 LIST
SCR # 19    WEDNESDAY NOVEMBER 12 1986    3:27:25 AM
  0 ( Variable versus time plotting)   DECIMAL        ( 082186 WEM)
  1 CREATE G-LABEL 40 ALLOT
  2 : TICK ORIG 45 0 DO I 10 * DUP 0 RSTART DUP XINC /MOD DROP
  3    0 = IF -10 ELSE -5 THEN RDRAW LOOP
  4    11 0 DO I 10 * 0 OVER RSTART 10 I = IF -10 ELSE -5
  5    THEN SWAP RDRAW LOOP
  6    11 0 DO I 10 * 0 OVER NEGATE RSTART 10 I = IF -10 ELSE -5
  7    THEN SWAP NEGATE RDRAW LOOP ;
  8
  9  : AXIS 10 PEN ORIG 500 Y0 DRAW ORIG X0 100 DRAW
 10     ORIG X0 300 DRAW ;
 11
 12 : +DOTS ( val) PEN-VAL @ PEN XPOS @ DUP Y0 START SWAP
 13   Y0 + DRAW  1 XPOS +! ;
 14
 15 : @VAL ( n-n) 4* VTIMEARRAY + @ ;
 ok
20 LIST
SCR # 20    WEDNESDAY NOVEMBER 12 1986    3:27:43 AM
  0 ( Variable versus time plotting)   DECIMAL        ( 040286 WEM)
  1
  2 : PLOTVTIME ( low inc) SWAP 450 0 DO OVER + DUP @VAL
  3    244 SCALE @ */ -100 MAX 100 MIN +DOTS LOOP 2DROP ;
  4
  5 : PLOT.DATA ( low inc) 2DUP X0 XPOS ! AXIS PLOTVTIME AXIS TICK
  6    50 * XINC * SWAP 50 * OVER - LAB-X LAB-Y ;
  7
  8 : NEW.PLOT ( low inc) G-LABEL 40 32 FILL CR ." Name of graph: "
  9     G-LABEL 40 EXPECT XALPHA CLR.GRAPH PLOT.DATA 10 PEN PROP
 10     5 X.CURSOR ! 250 Y.CURSOR ! ." Volts"
 11     10 Y.CURSOR ! 200 X.CURSOR ! ." Time ( seconds) "
 12     130 X.CURSOR ! 360 Y.CURSOR ! G-LABEL
 13     40 TYPE 10 Y.CURSOR ! 400 X.CURSOR ! REG ;
 14
 15
```

```
24 LIST
SCR # 24    WEDNESDAY NOVEMBER 12 1986    3:28:15 AM
  0 ( Documentation Correct Units vs.Time Routines)   ( 082286 WEM)
  1
  2    The following routines allow the user to view the behavior
  3  of a variable as a function of time, with the correct units
  4  of the variable available.  This software is essentially the
  5  same as the routines for volts vs. time plots, with a few minor
  6  additions.  Please refer to screens 13 and 14 as an addition to
  7  the documentation found here.  The key word here is DATA.IN
  8  which requires the same stack as VTIME.  The word TRUEUNITS
  9  must be edited prior to loading to ensure that the proper units
 10  conversion is made.  The conversion words are found on screens
 11  27 and 28.  The word which must be updated is in small letters
 12  within the definition of TRUEUNITS.  The other key word,
 13  NEW.PLOT works exactly as it does in the volts vs. time
 14  routine.  Again refer to screens 13 and 14.
 15
 ok
```

```
25 LIST
SCR # 25    WEDNESDAY NOVEMBER 12 1986    3:29:04 AM
  0 ( Load screen, correct units var. vs. time plots) ( 081986 WEM)
  1
  2    100 LOAD          ( Virtual memory defining words)
  3     16 LOAD          ( Voltage Data Constants)
  4     17 LOAD          ( Loading Calib. Data, volts vs.time)
  5     26 LOAD          ( Load and Tachometer Resistance constants)
  6     27 28 THRU       ( Unit correction factors)
  7     29 LOAD          ( Store corrected variables in UNITSARRAY)
  8     30 32 THRU       ( Correct units versus time axis setup)
  9     33 LOAD          ( Correct units versus time plotting)
 10
 11
 12
 13
 14
 15
 ok
26 LIST
SCR # 26    WEDNESDAY NOVEMBER 12 1986    3:29:17 AM
  0 ( Load and Tachometer Resistance constants)       ( 081986 WEM)
  1 ( Values to be used for calculating angular velocity from tach)
  2
  3     ( 10 * Load Resistances)
  4 54700 CONSTANT XRL   54100 CONSTANT YRL   49000 CONSTANT ZRL
  5 54700 CONSTANT WRRL  54800 CONSTANT WLRL  54600 CONSTANT AZRL
  6 50600 CONSTANT TRL
  7     ( 10 * Tachometer Resistances)
  8 1672 CONSTANT XRG   1661 CONSTANT YRG   1623 CONSTANT ZRG
  9 1477 CONSTANT WRRG   1739 CONSTANT WLRG   1552 CONSTANT AZRG
 10 1693 CONSTANT TRG
 11
 12      ( Other constants for tachometer calculations)
 13 ( Voltage sensitivity * 10 in volts per rad/sec) 2 CONSTANT KG
 14
 15
 ok
27 LIST
SCR # 27    WEDNESDAY NOVEMBER 12 1986    3:29:39 AM
  0 ( Correct units of force, moment, ang. velocity)  ( 052186 WEM)
  1 : FORCEX ( n-force) 159576 4000 */ ;
  2 : FORCEY ( n-force) 158600 4000 */ ;
  3 : FORCEZ ( n-force) 631228 4000 */ ;
  4 : MOMENTX ( n-moment) 279380 4000 */ ;
  5 : MOMENTY ( n-moment) 275964 4000 */ ;
  6 : MOMENTZ ( n-moment) 132980 4000 */ ;
  7 ( NOTE: All above force & mom calc. are 1000 * actual value)
  8 : OMEGAX ( n-vel) 244 * 10 KG * / XRL XRG + 10 / * XRL / 10 * ;
  9 : OMEGAY ( n-vel) 244 * 10 KG * / YRL YRG + 10 / * YRL / 10 * ;
 10 : OMEGAZ ( n-vel) 244 * 10 KG * / ZRL ZRG + 10 / * ZRL / 10 * ;
 11 : OMEGAWR 244 * 10 KG * / WRRL WRRG + 10 / * WRRL / 10 * ;
 12 : OMEGAWL 244 * 10 KG * / WLRL WLRG + 10 / * WLRL / 10 * ;
 13 : OMEGAAZ 244 * 10 KG * / AZRL AZRG + 10 / * AZRL / 10 * ;
 14 : OMEGAT 244 * 10 KG * / TRL TRG + 10 / * TRL / 10 * ;
 15 ( Note: All above ang.velocity calcs. are 1000 * actual value)
```

```
28 LIST
SCR # 28    WEDNESDAY NOVEMBER 12 1986    3:30:11 AM
  0 ( Correct units of acceleration and current)      ( 042386 WEM)
  1   : ACX ( n- g) 1000 * 244 100000 */ ;
  2   : ACY ( n- g) 1000 * 244 100000 */ ;
  3   : ACZ ( n- g) 1000 * 244 100000 */ ;
  4
  5   : CURZ ( n- amp's) 1000 * 244 100000 */ ;
  6   : CURY ( n- amps) 1000 * 244 100000 */ ;
  7   : CURWL ( n- amps) 1000 * 244 100000 */ ;
  8   : CURWR  ( n- amps) 1000 * 244 100000 */ ;
  9
 10   ( Note: All above acc. and curr. calc are 1000 * true value.)
 11
 12
 13
 14
 15
 ok
29 LIST
SCR # 29    WEDNESDAY NOVEMBER 12 1986    3:30:28 AM
  0 ( Store corrected variable units in UNITSARRAY)   ( 091186 WEM)
  1 ( Variables retrieved from VTIMEARRAY, manipulated, then stored)
  2  CREATE UNITSARRAY 84000 4*  ALLOT
  3  VARIABLE MAXC 0 MAXC !    VARIABLE TMAX 0 TMAX !
  4 : MAXV ( - max) 0 VCOUNT @ 0 DO I 4* VTIMEARRAY + @ ABS
  5    MAX LOOP ;
  6 : ZE VCOUNT @ 10000 +  0 DO 0 UNITSARRAY I 4* + ! LOOP ;
  7 ( Note: Small lettered word in TRUEUNITS varies depending
  8   on variable being analyzed.)
  9 : TRUEUNITS ( --) ZE VCOUNT @ 0 DO I 4* VTIMEARRAY + @
 10       forcex I 4* UNITSARRAY + ! LOOP  0 0 VCOUNT @ 0 DO OVER
 11       I 4* UNITSARRAY + @ ABS 2DUP < IF >R DROP 2DROP R> I
 12       ELSE 2DROP THEN LOOP TMAX ! MAXC ! 7 EMIT 7 EMIT ;
 13 ( Note: MAXC gives 1000 * max value)
 14
 15 : DATA.IN ( const.from scr#16,s,f--)  VTIME   TRUEUNITS ;
 ok
30 LIST
SCR # 30    WEDNESDAY NOVEMBER 12 1986    3:30:40 AM
  0 ( Correct units versus time axis setup)          ( 091986 WEM)
  1
  2 50 CONSTANT X0      200 CONSTANT Y0
  3
  4 VARIABLE PEN-VAL    15 PEN-VAL !
  5
  6 VARIABLE XPOS     100 CONSTANT XINC
  7 ( Note: [XINC]*[Plotting increment, inc]*[.050]= Amount of time
  8     in seconds between 2 major x-axis tick marks.)
  9
 10 : ORIG X0 Y0 START ;
 11
 12 : LAB-X PROP Y0 20 - Y.CURSOR ! 450 XINC / 0 DO I XINC * X0
 13      + X.CURSOR ! OVER + DUP 10 / <# # # 46 HOLD #S #>
 14      TYPE LOOP   2DROP   REG ;
 15
```

```
31 LIST
SCR # 31    WEDNESDAY NOVEMBER 12 1986    3:31:04 AM
  0 ( Correct units versus time axis setup)  DECIMAL ( 081986 WEM)
  1 CREATE G-LABEL 40 ALLOT
  2
  3 : TICK ORIG 45 0 DO I 10 * DUP 0 RSTART DUP XINC /MOD DROP
  4    0 = IF -10 ELSE -5 THEN RDRAW LOOP
  5    11 0 DO I 10 * 0 OVER RSTART 10 I = IF -10 ELSE -5
  6    THEN SWAP RDRAW LOOP
  7    11 0 DO I 10 * 0 OVER NEGATE RSTART 10 I = IF -10 ELSE -5
  8    THEN SWAP NEGATE RDRAW LOOP ;
  9
 10  : AXIS 10 PEN ORIG 500 Y0 DRAW ORIG X0 100 DRAW
 11     ORIG X0 300 DRAW ;
 12
 13 : +DOTS ( val) PEN-VAL @ PEN XPOS @ DUP Y0 START SWAP
 14   Y0 + DRAW  1 XPOS +! ;
 15
 ok
32 LIST
SCR # 32    WEDNESDAY NOVEMBER 12 1986    3:31:21 AM
  0 ( Correct units variable versus time axis setup)  ( 082186 WEM)
  1  VARIABLE SCALE  1 SCALE !
  2
  3 : LAB-Y PROP 2 0 DO 10 X.CURSOR ! I MAXC @ * SCALE @ /
  4    I 100 * Y0 + Y.CURSOR ! <# # # # 46 HOLD #S #> TYPE LOOP
  5    -1 -1 DO 5 X.CURSOR ! I MAXC @ * SCALE @ / I 100 *
  6    Y0 + Y.CURSOR ! DUP ABS <# # # # 46 HOLD #S DROP SIGN #>
  7    TYPE -1 +LOOP DROP REG ;
  8
  9  : @VAL ( n-n) 4* UNITSARRAY + @ ;
 10
 11
 12
 13
 14
 15
 ok
33 LIST
SCR # 33    WEDNESDAY NOVEMBER 12 1986    3:31:38 AM
  0 ( Correct units variable versus time plotting)    ( 081986 WEM)
  1
  2 : PLOTVTIME  ( low inc) SWAP 450 0 DO OVER + DUP @VAL 100 *
  3     SCALE @ * MAXC @ / -100 MAX 100 MIN +DOTS LOOP 2DROP ;
  4
  5 : PLOT.DATA ( low inc) 2DUP X0 XPOS ! AXIS  PLOTVTIME  AXIS TICK
  6    50 * XINC * SWAP 50 * OVER -   LAB-X   LAB-Y ;
  7
  8 : NEW.PLOT ( low inc) G-LABEL 40 32 FILL CR ." Name of graph: "
  9     G-LABEL 40 EXPECT XALPHA  CLR.GRAPH  PLOT.DATA 10 PEN PROP
 10     5 X.CURSOR ! 250 Y.CURSOR ! ." lbf."
 11     10 Y.CURSOR ! 200 X.CURSOR ! ." Time ( seconds) "
 12     130 X.CURSOR ! 360 Y.CURSOR ! G-LABEL
 13     40 TYPE 10 Y.CURSOR ! 400 X.CURSOR ! REG ;
 14
 15
```

```
45 LIST
SCR # 45    WEDNESDAY NOVEMBER 12 1986    3:32:03 AM
  0 ( Load screen for Resultants vs. Time Routines)   ( 090786 WEM)
  1
  2     100 LOAD       ( Virtual Memory Defining Words)
  3     74 LOAD        ( CODE SW@,  CODE SQRT)
  4     46 LOAD        ( Variables, Calibration "Constants")
  5     50 LOAD        ( Vtimearray, Code Calibrate)
  6     47 LOAD        ( Correct units)
  7     48 49 THRU     ( Correct units values stored in arrays)
  8     115 120 THRU   ( Task array and data arrays "Itemized)
  9     51 52 THRU     ( Set array pointers, fill arrays)
 10     53 56 THRU     ( Commands for plotting)
 11
 12
 13
 14
 15
 ok
46 LIST
SCR # 46    WEDNESDAY NOVEMBER 12 1986    3:32:28 AM
  0 ( Voltage Data Constants for versus time array)   ( 091286 WEM)
  1
  2 : Fx 28 24 26 ;    : Fy 30 24 26 ;    : Fz 32 24 26 ;
  3 : Mx 34 24 26 ;    : My 36 24 26 ;    : Mz 38 24 26 ;
  4
  5       VARIABLE FXARRAY          VARIABLE FYARRAY
  6       VARIABLE FZARRAY          VARIABLE RFARRAY
  7       VARIABLE MXARRAY          VARIABLE MYARRAY
  8       VARIABLE MZARRAY          VARIABLE RMARRAY
  9       VARIABLE ZR1ARRAY         VARIABLE ZR2ARRAY
 10
 11       VARIABLE VCOUNT           VARIABLE TN
 12       VARIABLE TMAX             VARIABLE MAXC
 13
 14
 15       0 MAXC !      0 TMAX !
 ok
47 LIST
SCR # 47    WEDNESDAY NOVEMBER 12 1986    3:32:44 AM
  0 ( Correct units of force & moment)                ( 091186 WEM)
  1
  2 : FORCEX ( n-force) 159576 4000 */ 10 / ;
  3 : FORCEY ( n-force) 158600 4000 */ 10 / ;
  4 : FORCEZ ( n-force) 631228 4000 */ 10 / ;
  5
  6 : MOMENTX ( n-moment) 279380 4000 */ 10 / ;
  7 : MOMENTY ( n-moment) 275964 4000 */ 10 / ;
  8 : MOMENTZ ( n-moment) 132080 4000 */ 10 / ;
  9
 10 ( Note: All above force/moment calc. are 100 * actual value)
 11
 12
 13
 14
 15
```

```
48 LIST
SCR # 48    WEDNESDAY NOVEMBER 12 1986    3:33:08 AM
  0 ( Store corrected variable units)                ( 090786 WEM)
  1 ( Variables retrieved from VTIMEARRAY, manipulated,then stored)
  2
  3 : FXTRUE ( --) VCOUNT @ 0 DO I 2*
  4     VTIMEARRAY + SW@ forcex I 2* FXARRAY @  + W! LOOP ;
  5 : FYTRUE ( --) VCOUNT @ 0 DO I 2*
  6     VTIMEARRAY + SW@ forcey I 2* FYARRAY @ + W! LOOP ;
  7 : FZTRUE ( --) VCOUNT @ 0 DO I 2*
  8     VTIMEARRAY + SW@ forcez I 2* FZARRAY @ + W! LOOP ;
  9 : MXTRUE ( --) VCOUNT @ 0 DO I 2*
 10     VTIMEARRAY + SW@ momentx I 2* MXARRAY @ + W! LOOP ;
 11 : MYTRUE ( --) VCOUNT @ 0 DO I 2*
 12     VTIMEARRAY + SW@ momenty I 2* MYARRAY @ + W! LOOP ;
 13 : MZTRUE ( --) VCOUNT @ 0 DO I 2*
 14     VTIMEARRAY + SW@ momentz I 2* MZARRAY @ + W! LOOP ;
 15
 ok
49 LIST
SCR # 49    WEDNESDAY NOVEMBER 12 1986    3:33:27 AM
  0 ( Words for filling resultant arrays)            ( 090786 WEM)
  1
  2 : FRESULT ( --) VCOUNT @ 0 DO I 2*
  3     FXARRAY @ + SW@ DUP * I 2* FYARRAY @ + SW@ DUP * I 2*
  4     FZARRAY @ + SW@ DUP * + +  SQRT I 2* RFARRAY @
  5     + W! LOOP ;
  6
  7 : MRESULT ( --) VCOUNT @ 0 DO I 2*
  8     MXARRAY @ + SW@ DUP * I 2* MYARRAY @ + SW@ DUP * I 2*
  9     MZARRAY @ + SW@ DUP * + +  SQRT I 2* RMARRAY @
 10     + W! LOOP ;
 11
 12
 13 : TASK CREATE SWAP , , DOES> DUP @ SWAP 4+ @ ;
 14
 15
 ok
50 LIST
SCR # 50    WEDNESDAY NOVEMBER 12 1986    3:33:43 AM
  0 ( Loading calibrated counts data array)          ( 091186 WEM)
  1 DECIMAL CREATE VTIMEARRAY 13200 2* ALLOT
  2 : ZZZ 13200 0 DO 0 VTIMEARRAY I 2* + W! LOOP ;
  3 CODE CALIBRATE  ( hi lo n -- n')
  4    SP )+ D0 LONG MOVE,  SP )+ D1 LONG MOVE,
  5    SP )+ D2 LONG MOVE,   4095 # D3 LONG MOVE,  D3 D4 LONG MOVE,
  6    D3 TO D1 LONG AND,   D2 TO D3 LONG AND,   D1 TO D3 LONG SUB,
  7    D2 4095 LONG CMPI,   GT IF, D3 LONG NEG, THEN,
  8    D3 08 # LONG ASR,   D3 01 # LONG ASR,   D0 TO D4 LONG AND,
  9    D1 TO D4 LONG SUB,  D0 4095 LONG CMPI,
 10    GT IF,  D4 LONG NEG,   THEN,   D3 TO D4 LONG SUB,
 11    D4 SP -) LONG MOVE,   NEXT   END-CODE
 12 : VTIME ( constant from scr#46, s, f) 1024 - 0 VCOUNT !
 13    ZZZ SWAP DO DUP I + VW@ 2OVER I + VW@ SWAP I + VW@
 14    CALIBRATE VCOUNT @ 2* VTIMEARRAY + W! 1 VCOUNT +!
 15    120 +LOOP DROP 2DROP ;
```

```
51 LIST
SCR # 51    WEDNESDAY NOVEMBER 12 1986    3:34:10 AM
  0 ( Setting array pointers)                          ( 091286 WEM)
  1
  2 CREATE BEGIN.POINT  13200 2* 10 * ALLOT
  3 : ZR1 13200 10 * 0 DO 0 BEGIN.POINT I 2* + W! LOOP ;
  4
  5 : POINTERS.SET
  6    BEGIN.POINT  VCOUNT @ 2* + FYARRAY !
  7    BEGIN.POINT  VCOUNT @ 4* + FZARRAY !
  8    BEGIN.POINT  VCOUNT @ 6 * + MXARRAY !
  9    BEGIN.POINT  VCOUNT @ 8* + MYARRAY !
 10    BEGIN.POINT  VCOUNT @ 10 * + MZARRAY !
 11    BEGIN.POINT  VCOUNT @ 12 * + RFARRAY !
 12    BEGIN.POINT  VCOUNT @ 14 * + ZR1ARRAY !
 13    BEGIN.POINT  VCOUNT @ 16 * + RMARRAY !
 14    BEGIN.POINT  VCOUNT @ 18 * + ZR2ARRAY ! ;
 15
 ok
52 LIST
SCR # 52    WEDNESDAY NOVEMBER 12 1986    3:34:30 AM
  0 ( Resultants only continued)                       ( 091286 WEM)
  1
  2  : ENDS ( --) TN @ TAPE# VTIME ;
  3
  4  : FILLER ( --) BEGIN.POINT  FXARRAY ! ZR1
  5     FX ENDS FXTRUE   POINTERS.SET
  6     FY ENDS FYTRUE FZ ENDS FZTRUE
  7     MX ENDS MXTRUE   MY ENDS MYTRUE   MZ ENDS MZTRUE
  8     FRESULT  MRESULT   7 EMIT ;
  9
 10  : MAXIMUM 0 0 VCOUNT @ 0 DO OVER I 2* RFARRAY @ + W@ ABS
 11      2DUP < IF >R DROP 2DROP R> I ELSE 2DROP THEN LOOP
 12      TMAX ! MAXC ! ;
 13
 14
 15
 ok
53 LIST
SCR # 53    WEDNESDAY NOVEMBER 12 1986    3:34:45 AM
  0 ( Resultants versus time axis setup)               ( 091186 WEM)
  1
  2 50 CONSTANT X0      50 CONSTANT Y0
  3
  4 VARIABLE PEN-VAL    15 PEN-VAL !
  5
  6 VARIABLE XPOS     100 CONSTANT XINC
  7 ( Note: [XINC]*[Plotting increment, inc]*[.050]= Amount of time
  8     in seconds between 2 major x-axis tick marks.)
  9
 10 : ORIG X0 Y0 START ;
 11
 12 : LAB-X PROP Y0 20 - Y.CURSOR ! 450 XINC / 0 DO I XINC * X0
 13     + X.CURSOR ! OVER + DUP 10 / <# # # 46 HOLD #S #>
 14     TYPE LOOP   2DROP   REG ;
 15
```

```
54 LIST
SCR # 54    WEDNESDAY NOVEMBER 12 1986    3:35:12 AM
  0 ( Resultants versus time axis setup)   DECIMAL     ( 090986 WEM)
  1 CREATE G-LABEL 40 ALLOT
  2
  3 : TICK ORIG 45 0 DO I 10 * DUP 0 RSTART DUP XINC /MOD DROP
  4    0 = IF -10 ELSE -5 THEN RDRAW LOOP
  5    31 0 DO I 10 * 0 OVER RSTART DUP 100 /MOD DROP 0 = IF
  6    -10 ELSE -5 THEN SWAP RDRAW LOOP ;
  7
  8
  9  : AXIS 10 PEN ORIG 500 Y0 DRAW X0 Y0 START X0 350 DRAW ;
 10
 11
 12 : +DOTS ( val) PEN-VAL @ PEN XPOS @ DUP Y0 START SWAP
 13   Y0 + DRAW  1 XPOS +! ;
 14
 15
 ok
55 LIST
SCR # 55    WEDNESDAY NOVEMBER 12 1986    3:35:31 AM
  0 ( Resultants versus time axis setup)              ( 091286 WEM)
  1
  2  VARIABLE SCALE  3 SCALE !
  3
  4 : LAB-Y PROP 4 0 DO 10 X.CURSOR ! I MAXC @ * SCALE @ /
  5    I 100 * Y0 + Y.CURSOR ! <# # # 46 HOLD #S #> TYPE LOOP ;
  6
  7
  8  : @VAL ( n-n) 2* RFARRAY @ + W@ ;
  9
 10
 11
 12
 13
 14
 15
 ok
56 LIST
SCR # 56    WEDNESDAY NOVEMBER 12 1986    3:35:41 AM
  0 ( Resultants versus time plotting)                ( 091286 WEM)
  1
  2 : PLOTVTIME  ( low inc) SWAP 450 0 DO OVER + DUP @VAL 100 *
  3     SCALE @ * MAXC @ / 300 MIN +DOTS LOOP 2DROP ;
  4
  5 : PLOT.DATA ( low inc) 2DUP X0 XPOS ! AXIS  PLOTVTIME  AXIS TICK
  6    100 * XINC * SWAP 100 * OVER -   LAB-X   LAB-Y ;
  7
  8 : NEW.PLOT ( low inc) G-LABEL 40 32 FILL CR ." Name of graph: "
  9     G-LABEL 40 EXPECT XALPHA  CLR.GRAPH  PLOT.DATA 10 PEN PROP
 10     5 X.CURSOR ! 200 Y.CURSOR ! ." lbf."
 11     10 Y.CURSOR ! 200 X.CURSOR ! ." Time ( seconds) "
 12     130 X.CURSOR ! 360 Y.CURSOR ! G-LABEL
 13     40 TYPE 10 Y.CURSOR ! 400 X.CURSOR ! REG ;
 14
 15 : KILLER   FILLER  MAXIMUM  7 EMIT  7 EMIT  0 2 NEW.PLOT ;
```

```
60 LIST
SCR # 60    WEDNESDAY NOVEMBER 12 1986    3:36:14 AM
  0 ( 3D Histogram Plotting) DECIMAL                    ( 050786 WEM)
  1  225 CONSTANT X0 150 CONSTANT Y0
  2 : ORIG X0 Y0 START ;
  3 : AXES  10 PEN ORIG 385 50 DRAW ORIG 65 50 DRAW ORIG
  4    225 350 DRAW ;
  5
  6 : TICKS 10 PEN ORIG 21 0 DO I 10 * 0 OVER RSTART DUP 100 /MOD
  7    DROP 0= IF -10 ELSE -5 THEN SWAP RDRAW LOOP  ORIG
  8    21 0 DO  I 10 * I 8 * I 5 * NEGATE 2DUP
  9    RSTART ROT 100 /MOD DROP 0= IF 10 - ELSE 5 -
 10    THEN RDRAW LOOP ORIG 21 0 DO I 10 * I 8 * NEGATE
 11    I 5 * NEGATE 2DUP RSTART ROT 100 /MOD DROP 0=
 12    IF 10 - ELSE 5 - THEN RDRAW LOOP ;
 13
 14
 15
 ok
```

```
66 LIST
SCR # 66    WEDNESDAY NOVEMBER 12 1986    3:36:57 AM
  0 ( Documentation: Resultants, Errors, and Matrix)  ( 082786 WEM)
  1
  2     The following routines enable the user to analyze data
  3  in the form of force/moment resultants, errorcode, and/or
  4  a statistical matrix.  The routines are capable of handling up
  5  to a tape of data at a time.  The basis for both routines lies
  6  in screens 115 thru 120.  On screens 115 thru 118, each task to
  7  be studied must be listed in the manner: start address  stop
  8  address, TASK name of task.  Example:
  9        21504 678564 TASK ELEC.CONNS.1
 10  Each task must have a different name. Then, on screens 119 and
 11  120, the tasks must be listed in the order to be analyzed
 12  within the word ITEMS TAPE# .
 13     If one is interested in only maximum and average resultant
 14  forces and moments and errorcode, then the load screen 94 is
 15  used.
 ok
67 LIST
SCR # 67    WEDNESDAY NOVEMBER 12 1986    3:37:15 AM
  0 ( Documentation: Resultants, Errors, and Matrix)  ( 082786 WEM)
  1
  2        The list of screens nos. from tape directory along with
  3   the number of tasks to be analyzed must be updated within the
  4   do-loop in SHABANG on scr.#99. All this should be done before
  5   loading.  The key word SHABANG will then execute each task and
  6   write to both the monitor and printer, if the printer is on.
  7   Output for each task will include, in order, task no., maximum
  8   and average resultant force, number of occurances where force
  9   exceeded 1 lbf., maximum and average moments, number of times
 10   that moment exceeded 1 lbf-in., sum of resultant forces
 11   squared, sum of resultant moments squared. All forces and
 12   moments with values less than 1 were thrown out as noise.
 13   The sums of squares are valuable from a statistical approach,
 14   so that is why they are included here.
 15
 ok
68 LIST
SCR # 68    WEDNESDAY NOVEMBER 12 1986    3:37:33 AM
  0 ( Documentation: Resultants, Errors, and Matrix)  ( 082286 WEM)
  1
  2     In order to obtain a matrix of variance and covariance for
  3 a statistical study of each of the variables and their inter-
  4 relationships, the routines should be loaded using screen 73.
  5 Again, the contents of screens 115 thru 120 should contain a
  6 listing of tasks to be analyzed and their addresses on the hard
  7 disk, as before.  This matrix has the form of a 22x22 square
  8 matrix which is alot of cross multiplication of the 20 variables
  9 measured plus the resultant force and resultant moment scalars.
 10 The remaining 2 rows of the matrix, row 23 and 24, are the rows
 11 of maximum values and average values for each variable.  Thus, a
 12 matrix 24x22 is formed.  Note, the average values for force and
 13 moment and their resultants are worthless due to the fact that
 14 noise was not eliminated in this study.  For those values, the
 15 afore mentioned routine is required. The 22x22 is formed by
```

```
69 LIST
SCR # 69    WEDNESDAY NOVEMBER 12 1986    3:38:36 AM
  0 ( Documentation: Resultants, Errors, and Matrix)  ( 082286 WEM)
  1
  2 [v]transposed times [v]  where [v] =
  3 [ Vx Vy Vz Vwr Vwl Vaz Vt Fx Fy Fz Mx My Mz Acx Acy Acz Cz
  4                                      Cy Cwl Cwr Rf Rm ]
  5 Legend:
  6   Vx,Vy,Vz, etc., are the velocities in their respective
  7                   directions.
  8   Fx,Fy,Fz are the forces in their respective directions.
  9   Mx,My,Mz are the moments in their respective directions.
 10   Acx,Acy,Acx are the accelerations in their respective
 11                   directions.
 12   Cz,Cy, etc., are the currents in their respective directions.
 13   Rf,Rm are the resultant force and moment scalar values.
 14
 15
 ok
```

```
70 LIST
SCR # 70    WEDNESDAY NOVEMBER 12 1986    3:38:59 AM
  0 ( Documentation: Resultants, Errors, and Matrix)  ( 082786 WEM)
  1
  2  The values in the matrix are the average for the entire section
  3  of data analyzed.  The key word COMPLETE on screen 92 is
  4  responsible for the entire matrix filling operation, averaging,
  5  and printing to screen and monitor the entire matrix. The do-
  6  loop within the word COMPLETE must be changed before loading
  7  the routines to reflect the number of tasks to be analyzed.
  8  The tape# must be changed within the word TITLE on screen 92.
  9
 10
 11
 12
 13
 14
 15
 ok
```

```
73 LIST
SCR # 73    WEDNESDAY NOVEMBER 12 1986    3:40:11 AM
  0 ( Array and matrix filling load screen)          ( 080686 WEM)
  1
  2  100 LOAD        ( Virtual Memory defining words)
  3  74 76 THRU      ( Code, constants, calibration, vtimearrays)
  4  77 80 THRU      ( Correct units words, variable arrays)
  5  81 83  THRU     ( Values stored in arrays)
  6  84 LOAD         ( Resultants calculated and stored)
  7  115 120 THRU    ( Task array and data arrays "itemized")
  8  86 88 THRU      ( Setting array pointers, list data arrays)
  9  85 LOAD         ( Filling data arrays)
 10  89 91 THRU      ( Matrix defined, allotted, zeroed, and filled)
 11  92 LOAD         ( Commands to analyze an entire tape)
 12
 13  ( Note: The maximum task length that can be analyzed at once
 14    is 21 minutes. Revisions must be made for longer tasks)
 15
 ok
74 LIST
SCR # 74    WEDNESDAY NOVEMBER 12 1986    3:40:23 AM
  0 ( SIGNED 16-BIT and SQRT Routines)              ( 060386 WEM)
  1
  2 CODE SW@   SP )+ A0 LONG MOVE,   A0 ( ) D0 WORD MOVE,
  3    D0 WORD EXT,   D0 SP -) LONG MOVE,   NEXT  END-CODE
  4
  5 CODE SQRT
  6    SP )+ D0 LONG MOVE,
  7    15 # D3 WORD MOVE,   D2 WORD CLR,   D1 WORD CLR,
  8    BEGIN,  D0 1 # LONG ASL,   D1 1 # LONG ROXL,
  9            D0 1 # LONG ASL,   D1 1 # LONG ROXL,
 10    D2 2 # LONG ASL,   D2 1 WORD ADDI,   D2 D1 WORD CMP,
 11    CC IF,   D2 TO D1 WORD SUB,   D2 1 LONG ADDQ,   THEN,
 12    D2 1 # LONG LSR,   D3 LOOP,   D2 SP -) LONG MOVE,
 13    NEXT  END-CODE
 14
 15
 ok
75 LIST
SCR # 75    WEDNESDAY NOVEMBER 12 1986    3:40:32 AM
  0 ( Voltage Data Constants for versus time array)   ( 060386 WEM)
  1
  2 : Velx 10 6 8 ;     : Vely 12 6 8 ;    : Velz 14 6 8 ;
  3 : Velwr 16 6 8 ;    : Velwl 18 6 8 ;   : Velaz 20 6 8 ;
  4 : Velt 22 6 8 ;
  5
  6 : Fx 28 24 26 ;     : Fy 30 24 26 ;    : Fz 32 24 26 ;
  7 : Mx 34 24 26 ;     : My 36 24 26 ;    : Mz 38 24 26 ;
  8
  9 : Accx 46 42 44 ;       : Accy 48 42 44 ;    : Accz 50 42 44 ;
 10 : Currz 52 42 44 ;      : Curry 54 42 44 ;
 11 : Currwl 56 42 44 ;     : Currwr 58 42 44 ;
 12
 13
 14
 15
```

```
76 LIST
SCR # 76    WEDNESDAY NOVEMBER 12 1986    3:41:02 AM
  0 ( Loading calibrated counts data array)          ( 081886 WEM)
  1 DECIMAL CREATE VTIMEARRAY 13200 2* ALLOT    VARIABLE VCOUNT
  2 : ZZZ 13200 0 DO 0 VTIMEARRAY I 2* + W! LOOP ;
  3 CODE CALIBRATE  ( hi lo n -- n')
  4    SP )+ D0 LONG MOVE,  SP )+ D1 LONG MOVE,
  5    SP )+ D2 LONG MOVE,   4095 # D3 LONG MOVE,  D3 D4 LONG MOVE,
  6    D3 TO D1 LONG AND,   D2 TO D3 LONG AND,   D1 TO D3 LONG SUB,
  7    D2 4095 LONG CMPI,   GT IF, D3 LONG NEG, THEN,
  8    D3 08 # LONG ASR,   D3 01 # LONG ASR,   D0 TO D4 LONG AND,
  9    D1 TO D4 LONG SUB,  D0 4095 LONG CMPI,
 10    GT IF,  D4 LONG NEG,   THEN,   D3 TO D4 LONG SUB,
 11    D4 SP -) LONG MOVE,   NEXT   END-CODE
 12 : VTIME ( constant from scr#75, s, f) 1024 - 0 VCOUNT !
 13    ZZZ SWAP DO DUP I + VW@ 2OVER I + VW@ SWAP I + VW@
 14    CALIBRATE VCOUNT @ 2* VTIMEARRAY + W! 1 VCOUNT +!
 15    120 +LOOP DROP 2DROP ;
 ok
77 LIST
SCR # 77    WEDNESDAY NOVEMBER 12 1986    3:41:11 AM
  0 ( Load and Tachometer Resistance constants)      ( 051486 WEM)
  1 ( Values to be used for calculating angular velocity from tach)
  2
  3     ( 10 * Load Resistances)
  4 54700 CONSTANT XRL   54100 CONSTANT YRL   49000 CONSTANT ZRL
  5 54700 CONSTANT WRRL  54800 CONSTANT WLRL  54600 CONSTANT AZRL
  6 50600 CONSTANT TRL
  7     ( 10 * Tachometer Resistances)
  8 1672 CONSTANT XRG   1661 CONSTANT YRG   1623 CONSTANT ZRG
  9 1477 CONSTANT WRRG   1739 CONSTANT WLRG   1552 CONSTANT AZRG
 10 1693 CONSTANT TRG
 11
 12      ( Other constants for tachometer calculations)
 13 ( Voltage sensitivity * 10 in volts per rad/sec) 2 CONSTANT KG
 14
 15
 ok
78 LIST
SCR # 78    WEDNESDAY NOVEMBER 12 1986    3:41:20 AM
                                                  ( 082786 WEM)
  0 ( Correct units of force & moment)
  1 : FORCEX ( n-force) 159576 4000 */ 10 / ;
  2 : FORCEY ( n-force) 158600 4000 */ 10 / ;
  3 : FORCEZ ( n-force) 631228 4000 */ 10 / ;
  4 : MOMENTX ( n-moment) 279380 4000 */ 10 / ;
  5 : MOMENTY ( n-moment) 275964 4000 */ 10 / ;
  6 : MOMENTZ ( n-moment) 132980 4000 */ 10 / ;
  7 ( Note: All above force/moment calc. are 100 * actual value)
  8 : OMEGAX ( n-vel) 244 * 10 KG * / XRL XRG + 10 / * XRL / ;
  9 : OMEGAY ( n-vel) 244 * 10 KG * / YRL YRG + 10 / * YRL / ;
 10 : OMEGAZ ( n-vel) 244 * 10 KG * / ZRL ZRG + 10 / * ZRL / ;
 11 : OMEGAWR ( n-vel) 244 * 10 KG * / WRRL WRRG + 10 / * WRRL / ;
 12 : OMEGAWL ( n-vel) 244 * 10 KG * / WLRL WLRG + 10 / * WLRL / ;
 13 : OMEGAAZ ( n-vel) 244 * 10 KG * / AZRL AZRG + 10 / * AZRL / ;
 14 : OMEGAT ( n-vel) 244 * 10 KG * / TRL TRG + 10 / * TRL / ;
 15 ( Note: All above ang.velocity calcs. are 100 * actual value)
```

```
79 LIST
SCR # 79    WEDNESDAY NOVEMBER 12 1986    3:41:50 AM
  0 ( Correct units of acceleration and current)      ( 052186 WEM)
  1   : ACX ( n- g) 1000 * 244 100000 */ ;
  2   : ACY ( n- g) 1000 * 244 100000 */ ;
  3   : ACZ ( n- g) 1000 * 244 100000 */ ;
  4
  5   : CURZ ( n- amps) 1000 * 244 100000 */ ;
  6   : CURY ( n- amps) 1000 * 244 100000 */ ;
  7   : CURWL ( n- amps) 1000 * 244 100000 */ ;
  8   : CURWR  ( n- amps) 1000 * 244 100000 */ ;
  9
 10   ( Note: All above acc. and curr. calc are 1000 * true value.)
 11
 12
 13
 14
 15
 ok
80 LIST
SCR # 80    WEDNESDAY NOVEMBER 12 1986    3:41:59 AM
  0 ( Create all arrays for correct units variables)  ( 080686 WEM)
  1   VARIABLE VXARRAY       VARIABLE VYARRAY
  2   VARIABLE VZARRAY       VARIABLE VWRARRAY
  3   VARIABLE VWLARRAY      VARIABLE VTARRAY
  4   VARIABLE FXARRAY       VARIABLE FYARRAY
  5   VARIABLE FZARRAY       VARIABLE MXARRAY
  6   VARIABLE MYARRAY       VARIABLE MZARRAY
  7   VARIABLE ACXARRAY      VARIABLE ACYARRAY
  8   VARIABLE ACZARRAY      VARIABLE CURZARRAY
  9   VARIABLE CURYARRAY     VARIABLE CURWLARRAY
 10   VARIABLE CURWRARRAY    VARIABLE VAZARRAY
 11   VARIABLE RFARRAY       VARIABLE RMARRAY
 12
 13
 14
 15
 ok
81 LIST
SCR # 81    WEDNESDAY NOVEMBER 12 1986    3:42:07 AM
  0 ( Store corrected variable units)                ( 072186 WEM)
  1 : VXTRUE ( --) VCOUNT @ 0 DO I 2*
  2      VTIMEARRAY + SW@ omegax I 2* VXARRAY @ + W! LOOP ;
  3 : VYTRUE ( --) VCOUNT @ 0 DO I 2*
  4     VTIMEARRAY + SW@ omegay I 2* VYARRAY @ + W! LOOP ;
  5 : VZTRUE ( --) VCOUNT @ 0 DO I 2*
  6     VTIMEARRAY + SW@ omegaz I 2* VZARRAY @ + W! LOOP ;
  7 : VWRTRUE ( --) VCOUNT @  0 DO I 2*
  8     VTIMEARRAY + SW@ omegawr I 2* VWRARRAY @ + W! LOOP ;
  9 : VWLTRUE ( --) VCOUNT @  0 DO I 2*
 10     VTIMEARRAY + SW@ omegawl I 2* VWLARRAY @ + W! LOOP ;
 11 : VAZTRUE ( --) VCOUNT @  0 DO I 2*
 12     VTIMEARRAY + SW@ omegaaz I 2* VAZARRAY @ + W! LOOP ;
 13 : VTTRUE ( --) VCOUNT @   0 DO I 2*
 14     VTIMEARRAY + SW@ omegat I 2* VTARRAY @ + W! LOOP ;
 15
```

```
82 LIST
SCR # 82    WEDNESDAY NOVEMBER 12 1986    3:42:35 AM
  0 ( Store corrected variable units)                ( 071686 WEM)
  1 ( Variables retrieved from VTIMEARRAY, manipulated,then stored)
  2 : FXTRUE ( --) VCOUNT @ 0 DO I 2*
  3     VTIMEARRAY + SW@ forcex I 2* FXARRAY @  + W! LOOP ;
  4 : FYTRUE ( --) VCOUNT @ 0 DO I 2*
  5     VTIMEARRAY + SW@ forcey I 2* FYARRAY @ + W! LOOP ;
  6 : FZTRUE ( --) VCOUNT @ 0 DO I 2*
  7     VTIMEARRAY + SW@ forcez I 2* FZARRAY @ + W! LOOP ;
  8 : MXTRUE ( --) VCOUNT @ 0 DO I 2*
  9     VTIMEARRAY + SW@ momentx I 2* MXARRAY @ + W! LOOP ;
 10 : MYTRUE ( --) VCOUNT @ 0 DO I 2*
 11     VTIMEARRAY + SW@ momenty I 2* MYARRAY @ + W! LOOP ;
 12 : MZTRUE ( --) VCOUNT @ 0 DO I 2*
 13     VTIMEARRAY + SW@ momentz I 2* MZARRAY @ + W! LOOP ;
 14
 15
ok
83 LIST
SCR # 83    WEDNESDAY NOVEMBER 12 1986    3:42:53 AM
  0 ( Store corrected variable units in UNITSARRAY)   ( 072186 WEM)
  1 ( Variables retrieved from VTIMEARRAY, manipulated, then stored)
  2 : ACXTRUE ( --) VCOUNT @ 0 DO I 2*
  3     VTIMEARRAY + SW@ acx I 2* ACXARRAY @ + W! LOOP ;
  4 : ACYTRUE ( --) VCOUNT @ 0 DO I 2*
  5     VTIMEARRAY + SW@ acy I 2* ACYARRAY @ + W! LOOP ;
  6 : ACZTRUE ( --) VCOUNT @ 0 DO I 2*
  7     VTIMEARRAY + SW@ acz I 2* ACZARRAY @ + W! LOOP ;
  8 : CURZTRUE ( --) VCOUNT @ 0 DO I 2*
  9     VTIMEARRAY + SW@ curz I 2* CURZARRAY @ + W! LOOP ;
 10 : CURYTRUE ( --) VCOUNT @ 0 DO I 2*
 11     VTIMEARRAY + SW@ cury I 2* CURYARRAY @ + W! LOOP ;
 12 : CURWLTRUE ( --) VCOUNT @ 0 DO I 2*
 13     VTIMEARRAY + SW@ curwl I 2* CURWLARRAY @ + W! LOOP ;
 14 : CURWRTRUE ( --) VCOUNT @ 0 DO I 2*
 15     VTIMEARRAY + SW@ curwr I 2* CURWRARRAY @ + W! LOOP ;
ok
84 LIST
SCR # 84    WEDNESDAY NOVEMBER 12 1986    3:43:12 AM
  0 ( Words for filling resultant arrays)            ( 071686 WEM)
  1
  2 : FRESULT ( --) VCOUNT @ 0 DO I 2*
  3     FXARRAY @ + SW@ DUP * I 2* FYARRAY @ + SW@ DUP * I 2*
  4     FZARRAY @ + SW@ DUP * + +  SQRT I 2* RFARRAY @
  5     + W! LOOP ;
  6
  7 : MRESULT ( --) VCOUNT @ 0 DO I 2*
  8     MXARRAY @ + SW@ DUP * I 2* MYARRAY @ + SW@ DUP * I 2*
  9     MZARRAY @ + SW@ DUP * + +  SQRT I 2* RMARRAY @
 10     + W! LOOP ;
 11
 12
 13 : TASK CREATE SWAP , , DOES> DUP @ SWAP 4+ @ ;
 14
 15
```

```
85 LIST
SCR # 85    WEDNESDAY NOVEMBER 12 1986    3:43:39 AM
  0 ( Commands for filling arrays)                    ( 082786 WEM)
  1  VARIABLE TN
  2
  3  : ENDS ( - -) TN @ TAPE#  VTIME ;
  4
  5  : FILLER  BEGIN.POINT VXARRAY ! ZR1 VELX ENDS VXTRUE
  6     POINTERS.SET   VELY ENDS VYTRUE
  7     VELZ ENDS VZTRUE   VELWR ENDS VWRTRUE
  8     VELWL ENDS VWLTRUE   VELAZ ENDS VAZTRUE   VELT ENDS VTTRUE
  9     FX ENDS FXTRUE   FY ENDS FYTRUE   FZ ENDS FZTRUE
 10     MX ENDS MXTRUE   MY ENDS MYTRUE   MZ ENDS MZTRUE
 11     ACCX ENDS ACXTRUE   ACCY ENDS ACYTRUE
 12     ACCZ ENDS ACZTRUE   CURRZ ENDS CURZTRUE
 13     CURRY ENDS CURYTRUE   CURRWL ENDS CURWLTRUE
 14     CURRWR ENDS CURWRTRUE   FRESULT  MRESULT
 15     7 EMIT 7 EMIT ;
 ok
86 LIST
SCR # 86    WEDNESDAY NOVEMBER 12 1986    3:43:51 AM
  0 ( Setting array pointers)                         ( 081186 WEM)
  1
  2 CREATE BEGIN.POINT  13200 2* 22 * ALLOT
  3 : ZR1 13200 22 * 0 DO 0 BEGIN.POINT I 2* + W! LOOP ;
  4
  5 : POINTERS.SET
  6    BEGIN.POINT  VCOUNT @ 2* + VYARRAY !
  7    BEGIN.POINT  VCOUNT @ 4* + VZARRAY !
  8    BEGIN.POINT  VCOUNT @ 6 * + VWRARRAY !
  9    BEGIN.POINT  VCOUNT @ 8* + VWLARRAY !
 10    BEGIN.POINT  VCOUNT @ 10 * + VAZARRAY !
 11    BEGIN.POINT  VCOUNT @ 12 * + VTARRAY !
 12    BEGIN.POINT  VCOUNT @ 14 * + FXARRAY !
 13    BEGIN.POINT VCOUNT @ 16* + FYARRAY !
 14    BEGIN.POINT VCOUNT @ 18 * + FZARRAY !
 15    BEGIN.POINT VCOUNT @ 20 * + MXARRAY !     [
 ok
87 LIST
SCR # 87    WEDNESDAY NOVEMBER 12 1986    3:44:05 AM
  0 ( Setting pointers continued)                     ( 080686 WEM)
  1  ]  BEGIN.POINT VCOUNT @ 22 * + MYARRAY !
  2     BEGIN.POINT VCOUNT @ 24 * + MZARRAY !
  3     BEGIN.POINT VCOUNT @ 26 * + ACXARRAY !
  4     BEGIN.POINT VCOUNT @ 28 * + ACYARRAY !
  5     BEGIN.POINT VCOUNT @ 30 * + ACZARRAY !
  6     BEGIN.POINT VCOUNT @ 32 * + CURZARRAY !
  7     BEGIN.POINT VCOUNT @ 34 * + CURYARRAY !
  8     BEGIN.POINT VCOUNT @ 36 * + CURWLARRAY !
  9     BEGIN.POINT VCOUNT @ 38 * + CURWRARRAY !
 10     BEGIN.POINT VCOUNT @ 40 * + RFARRAY !
 11     BEGIN.POINT VCOUNT @ 42 * + RMARRAY !  ;
 12
 13
 14
 15
```

```
88 LIST
SCR # 88    WEDNESDAY NOVEMBER 12 1986    3:44:41 AM
  0 ( Arrays of data)                                    ( 080686 WEM)
  1
  2 ITEMS DATA-ARRAY
  3    VXARRAY     VYARRAY     VZARRAY
  4    VWRARRAY    VWLARRAY    VAZARRAY  VTARRAY
  5    FXARRAY     FYARRAY     FZARRAY
  6    MXARRAY     MYARRAY     MZARRAY
  7    ACXARRAY    ACYARRAY    ACZARRAY
  8    CURZARRAY   CURYARRAY   CURWLARRAY  CURWRARRAY
  9    RFARRAY     RMARRAY
 10  END-ITEMS
 11
 12
 13
 14
 15
 ok
89 LIST
SCR # 89    WEDNESDAY NOVEMBER 12 1986    3:44:47 AM
  0 ( Two Dim Array defined and zeroed)                  ( 080686 WEM)
  1
  2 : MATRIX ( #row #col)
  3    CREATE OVER , * 4* HERE OVER ALLOT SWAP ERASE
  4    ;CODE   SP )+ D0 LONG MOVE,   D0 1 LONG SUBQ,
  5    W ( ) D1 LONG MOVE,  D1 D0 MULU,  SP )+ TO D0 LONG ADD,
  6    D0 2 # LONG ASL,  W TO D0 LONG ADD,   D0 SP -) LONG MOVE,
  7    NEXT   END-CODE
  8
  9 24 22 MATRIX STATS
 10
 11 EXIT
 12
 13
 14
 15
 ok
90 LIST
SCR # 90    WEDNESDAY NOVEMBER 12 1986    3:44:58 AM
  0 ( Matrix filling words)                              ( 061286 WEM)
  1
  2  VARIABLE #COUNT
  3
  4 : 22X22
  5       VCOUNT @ 0 DO I #COUNT !
  6       23 1 DO
  7               I DATA-ARRAY @ #COUNT @ 2* + SW@
  8       23 1 DO DUP
  9               I DATA-ARRAY @ #COUNT @ 2* + SW@
 10                  * 100 / J I STATS +!
 11            LOOP DROP LOOP LOOP  ;
 12
 13 : AVG 23 1 DO 23 1 DO J I STATS @ VCOUNT @ /
 14     J I STATS ! LOOP LOOP ;
 15
```

```
91 LIST
SCR # 91    WEDNESDAY NOVEMBER 12 1986    3:45:52 AM
  0 ( More matrix filling words)                    ( 080686 WEM)
  1
  2 ( Row of maximum values)
  3 : ROW23 ( --) 23 1 DO  VCOUNT @ 0 DO J DATA-ARRAY @
  4    I 2 * + SW@ ABS I IF MAX THEN LOOP 23 I STATS ! LOOP ;
  5
  6 ( Row of average values)
  7 : ROW24 ( --) 23 1 DO VCOUNT @ 0 DO J DATA-ARRAY @ I 2 * + SW@
  8    24 J STATS +! LOOP 24 I STATS @ VCOUNT @ / 24 I STATS !
  9    LOOP ;
 10
 11 : MATRIX.FILL ( --) 22X22 AVG 7 EMIT ROW23 7 EMIT ROW24
 12    7 EMIT ;
 13
 14 : ZROO 25 1 DO 23 1 DO 0 J I STATS ! LOOP LOOP ;
 15
 ok
92 LIST
SCR # 92    WEDNESDAY NOVEMBER 12 1986    3:46:00 AM
  0 ( Do everything screen)                         ( 082786 WEM)
  1
  2 : TITLE CR CR 10 SPACES ." MATRICES FOR TASKS FROM TAPE#20  "
  3    CR CR  ;
  4
  5 : LISTING CR CR CR CR CR 42 EMIT 2 SPACES TN @ . 2 SPACES
  6    42 EMIT CR CR 25 1 DO 23 1 DO J I STATS @ 7 .R I 11 MOD 0=
  7    IF CR THEN LOOP CR CR LOOP ;
  8
  9 : COMPLETE CR TITLE  2277 LIST   2278 LIST
 10    2 1 DO I TN ! FILLER
 11    MATRIX.FILL   LISTING  ZROO  LOOP ;
 12
 13
 14
 15
 ok
93 LIST
SCR # 93    WEDNESDAY NOVEMBER 12 1986    3:46:10 AM
  0 ( Setting array pointers)                       ( 071686 WEM)
  1
  2 CREATE BEGIN.POINT  42000 2* 8* ALLOT
  3 : ZR1 42000 8* 0 DO 0 BEGIN.POINT I 2* + W! LOOP ;
  4
  5 : POINTERS.SET
  6    BEGIN.POINT  VCOUNT @ 2* + FYARRAY !
  7    BEGIN.POINT  VCOUNT @ 4* + FZARRAY !
  8    BEGIN.POINT  VCOUNT @ 6 * + MXARRAY !
  9    BEGIN.POINT  VCOUNT @ 8* + MYARRAY !
 10    BEGIN.POINT  VCOUNT @ 10 * + MZARRAY !
 11    BEGIN.POINT  VCOUNT @ 12 * + RFARRAY !
 12    BEGIN.POINT  VCOUNT @ 14 * + RMARRAY ! ;
 13
 14
 15
```

```
94 LIST
SCR # 94    WEDNESDAY NOVEMBER 12 1986    3:48:12 AM
  0 ( Load screen for analysis of resultants only)    ( 082786 WEM)
  1
  2   100 LOAD          ( Virtual memory defining words)
  3   74 76 THRU        ( Code, constants, calibration, vtimearrays)
  4   77 78 THRU        ( Correct Units)
  5   95 LOAD           ( Variable arrays)
  6   82 LOAD           ( Variables stored in arrays)
  7   84 LOAD           ( Resultant forces and moments, task def.)
  8   115 120 THRU      ( Tasks, items found on particular tape)
  9   93 LOAD           ( Setting array pointers)
 10   96 LOAD           ( Filler, items data-array)
 11   97 LOAD           ( Errorcode array defined and zeroed)
 12   98 LOAD           ( Max, avg, errors calculated)
 13   99 LOAD           ( Format for listing to printer)
 14
 15
 ok
95 LIST
SCR # 95    WEDNESDAY NOVEMBER 12 1986    3:48:20 AM
  0 ( Begin of analysis of data arrays only)          ( 071686 WEM)
  1
  2 VARIABLE FXARRAY  VARIABLE FYARRAY  VARIABLE FZARRAY
  3 VARIABLE MXARRAY  VARIABLE MYARRAY  VARIABLE MZARRAY
  4 VARIABLE RFARRAY  VARIABLE RMARRAY
  5
  6
  7
  8 VARIABLE MAXF   VARIABLE MAXM
  9 VARIABLE AVGF   VARIABLE AVGM  VARIABLE F>1  VARIABLE M>1
 10 VARIABLE SUMF2  VARIABLE SUMM2
 11
 12
 13
 14
 15
 ok
96 LIST
SCR # 96    WEDNESDAY NOVEMBER 12 1986    3:48:27 AM
  0 ( Resultants only continued)                    ( 082786 WEM)
  1  VARIABLE TN
  2
  3  : ENDS ( --) TN @ TAPE#  VTIME ;
  4
  5  : FILLER ( --) BEGIN.POINT  FXARRAY ! ZR1
  6     FX ENDS FXTRUE   POINTERS.SET
  7     FY ENDS FYTRUE FZ ENDS FZTRUE
  8     MX ENDS MXTRUE   MY ENDS MYTRUE   MZ ENDS MZTRUE
  9     FRESULT  MRESULT   7 EMIT 7 EMIT ;
 10
 11
 12
 13
 14
 15
```

```
97 LIST
SCR # 97    WEDNESDAY NOVEMBER 12 1986    3:48:53 AM
  0 ( Errorcode Array defined and zeroed)              ( 071186 WEM)
  1
  2 : MATRIX ( #row #col)
  3    CREATE OVER , * 4* HERE OVER ALLOT SWAP ERASE
  4    ;CODE   SP )+ D0 LONG MOVE,   D0 1 LONG SUBQ,
  5    W ( ) D1 LONG MOVE,  D1 D0 MULU,  SP )+ TO D0 LONG ADD,
  6    D0 2 # LONG ASL,  W TO D0 LONG ADD,   D0 SP -) LONG MOVE,
  7    NEXT   END-CODE
  8
  9 1 18 MATRIX ERRS
 10
 11 : ZR0 19 1 DO 0 1 I ERRS ! LOOP ;
 12
 13 EXIT
 14
 15
 ok
98 LIST
SCR # 98    WEDNESDAY NOVEMBER 12 1986    3:48:59 AM
  0 ( Resultants only analysis continued)              ( 071086 WEM)
  1  : MXF ( --) VCOUNT @ 0 DO RFARRAY @ I 2* + SW@ I ( 071086 WEM)
  2      IF MAX THEN LOOP MAXF ! ;
  3  : MXM ( --) VCOUNT @ 0 DO RMARRAY @ I 2* + SW@ I
  4      IF MAX THEN LOOP MAXM ! ;
  5  : AVF ( --) 0 F>1 ! 0 AVGF ! VCOUNT @ 0 DO RFARRAY @
  6      I 2* + SW@ DUP 100 > IF AVGF +! 1 F>1 +! ELSE DROP
  7      THEN LOOP AVGF @ F>1 @ / AVGF ! ;
  8  : AVM ( --) 0 M>1 ! 0 AVGM ! VCOUNT @ 0 DO RMARRAY @
  9      I 2* + SW@ DUP 100 > IF AVGM +! 1 M>1 +! ELSE DROP
 10      THEN LOOP AVGM @ M>1 @ / AVGM ! ;
 11  : SUMSQ 0 SUMF2 ! 0 SUMM2 ! VCOUNT @ 0 DO RFARRAY @ I 2* + SW@
 12   DUP 100 > IF DUP 100 */ SUMF2 +! ELSE DROP THEN RMARRAY @ I 2*
 13   + SW@ DUP 100 > IF DUP 100 */ SUMM2 +! ELSE DROP THEN LOOP ;
 14  : ERRORS ( s,f from TN tape#) 1024 - SWAP DO 4 I + VW@ DUP
 15      IF 1 1 ROT ERRS +! ELSE DROP THEN 60 +LOOP ;
 ok
99 LIST
SCR # 99    WEDNESDAY NOVEMBER 12 1986    3:49:20 AM
  0 ( Resultants analysis continued)                   ( 082786 WEM)
  1 : ENTREE ( ) FILLER MXF MXM AVF AVM SUMSQ TN @ TAPE#  ERRORS ;
  2
  3 : HEADER ( ) 23 SPACES ." RESULTANTS" CR 13 SPACES ." FORCES" 18
  4    SPACES ." MOMENTS" CR CR ." TASK#" 2 SPACES ." MAX" 5 SPACES
  5    ." AVG" 5 SPACES ." N" 7 SPACES ." MAX" 6 SPACES ." AVG"
  6    5 SPACES ." N" 6 SPACES ." SUMF2" 7 SPACES ." SUMM2" CR CR ;
  7 : PR# ( ) <#  # # 46 HOLD #S #> TYPE ;
  8 : LUCK ( ) 4 SPACES MAXF @ PR# 4 SPACES AVGF @ PR# 3 SPACES
  9    F>1 @ . 3 SPACES MAXM @ PR# 4 SPACES AVGM @ PR# 2 SPACES
 10    M>1 @ . 2 SPACES SUMF2 @ PR# 2 SPACES SUMM2 @ PR#
 11    CR CR 8 SPACES ." ERRORS:" 1 SPACES 19 1 DO 1 I
 12    ERRS @ 5 .R I 9 MOD 0= IF CR 16 SPACES THEN LOOP CR ;
 13 : SHABANG  2275 LIST
 14     CR CR CR CR HEADER 2 1 DO I TN !
 15     ENTREE TN @ . LUCK ZR0 CR LOOP ;
```

```
100 LIST
SCR # 100   WEDNESDAY NOVEMBER 12 1986     3:53:16 AM
  0 ( ** VIRTUAL Memory load screen                ** ) ( 040786 BSW)
  1 1 +LOAD EXIT
  2
  3   Virtual memory is set up as a user-definable region of disk
  4 storage to be used as if it were actual RAM.  This component
  5 allows transparent usage of disk memory.  However, if the
  6 user is willing to pay attention, some faster access words
  7 are available, as are some elementary string operations.
  8
  9
 10
 11
 12
 13
 14
 15
 ok
101 LIST
SCR # 101   WEDNESDAY NOVEMBER 12 1986     3:53:33 AM
  0 ( ** VIRTUAL Memory load screen                ** ) ( 040786 BSW)
  1
  2     1 +load     ( Virtual Memory Definition)
  3   2 4 +thru     ( Virtual Memory Primitives)
  4     5 +load     ( Virtual String  Primitives)
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 ok
102 LIST
SCR # 102   WEDNESDAY NOVEMBER 12 1986     3:53:47 AM
  0 ( Virtual Memory Definition)                    ( 121385 BSW)
  1 decimal
  2 B/D 2000 + Constant Virtual.block.base ( 1 volume up from start)
  3 Variable VDP       ( virtual data pointer)
  4 : vaddr ( virtual addr -- buffer addr)
  5   1024 /mod              ( offset\relative block#)
  6   Virtual.block.base + ( offset\absolute block#)
  7   block + ;
  8 : VDPBLOCK@ ( -- buf addr!block where VDP is pointing)
  9   VDP @ 1024 / virtual.block.base + block ;
 10 : VDPBUF@ ( -- buf addr!block where VDP is pointing)
 11   VDP @ 1024 / virtual.block.base + buffer ;
 12
 13
 14
 15
```

```
103 LIST
SCR # 103   WEDNESDAY NOVEMBER 12 1986    3:54:14 AM
  0 ( Virtual fetch & store primitives)                ( 103185 BSW)
  1 hex
  2 : VC@ vaddr C@ ;
  3 : VC! vaddr C! update ;
  4 : vfill ( vaddr\cnt\chr--iPb!fills virtual space)
  5         rot rot over + 1+ swap do dup I VC! loop drop ;
  6 : vcmove ( virtual src addr\virtual des addr\cnt--)
  7          0 do over I+ VC@ over I+ VC! loop 2drop ;
  8 : b.split ( w -- low byte\high byte!splits word) FFFF and
  9           dup FF and swap -8 scale ;
 10 : b.merge ( low byte\high byte -- w) 8 scale or FFFF and ;
 11 : w.split dup FFFF and swap -10 scale ;
 12 : w.merge 10 scale or ;
 13 decimal
 14
 15
 ok
104 LIST
SCR # 104   WEDNESDAY NOVEMBER 12 1986    3:54:24 AM
  0 ( Virtual fetch & store primitives)                ( 041686 WEM)
  1 : VW@   ( vaddr -- w)  dup 1+ VC@ swap VC@ b.merge ;
  2 : VW!   ( w\vaddr --)  swap b.split 3 pick VC! swap 1+ VC! ;
  3 : V@    ( vaddr -- n)  dup 2+ VW@ swap VW@ w.merge ;
  4 : V!    ( n\vaddr --)  swap w.split 3 pick vaddr W! update
  5         swap 2+ vaddr W! update ; ( vaddr W! update was VW!) ?
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 ok
105 LIST
SCR # 105   WEDNESDAY NOVEMBER 12 1986    3:54:32 AM
  0 ( Virtual Memory Access and Comma)                 ( 110485 BSW)
  1 ( Word fetch & store without regard to block boundaries)
  2 : vw'@ vaddr W@ ;
  3 : vw'! vaddr W! update ;
  4 : vl'@ vaddr @ ;
  5 : vl'! vaddr ! update ;
  6
  7 : VHERE VDP @ ;
  8 : ?VALIGN VDP dup @ =cells swap ! ;
  9 : V, VDP 4 over +! @ 4- V! ;
 10 : VW, VDP 2 over +! @ 2- vw'! ; ( vw'! was VW!)
 11 : VC, VDP 1 over +! @ 1- VC! ;
 12 : VW+! ( n\va --) dup vw'@ rot + swap vw'! ;
 13 : VALLOT ( n --) VDP +! ;
 14 : vblanks ( vaddr\cnt --)  32 vfill ;
 15 : vtype ( vaddr\cnt --) 0 do dup I+ VC@ emit loop drop ;
```

```
106 LIST
SCR # 106   WEDNESDAY NOVEMBER 12 1986    3:54:59 AM
  0 ( Virtual text & move words)                          ( 060486 WEM)
  1 hex
  2 : VEXPECT ( vaddr\cnt --) pocket dup rot expect swap cnt @ 0
  3           Do over I+ C@ over I+ VC! loop 2drop ;
  4
  5 : -VTEXT ( vaddr\n\addr --- flag) rot pad 4 pick 0 do over
  6          I+ VC@ over I+ C! loop swap drop swap rot swap -text ;
  7
  8 : cmove>v   ( addr\vaddr\cnt --- )
  9             0 do over I+ C@ over I+ VC! loop 2drop ;
 10 : v>cmove   ( vaddr\addr\cnt ---)
 11             0 do over over I+ VC@ swap I+ C! loop 2drop ;
 12 : VU.R ( d vaddr n) >R >R <# #S #> R> OVER - R> + swap cmove>v ;
 13
 14 DECIMAL
 15
 ok
115 LIST
SCR # 115   WEDNESDAY NOVEMBER 12 1986    3:55:07 AM
  0 ( Individual task listings w/addresses)               ( 091286 WEM)
  1
  2  1431552 2445311 TASK TUBE.CONNS.1
  3  5463040 6256639 TASK TUBE.CONNS.2
  4  7200768 7876607 TASK TUBE.CONNS.3
  5  9160704 9336831 TASK TUBE.CONNS.4A
  6  9336832 10407935 TASK TUBE.CONNS.4B
  7
  8
  9
 10
 11
 12
 13
 14
 15
 ok
116 LIST
SCR # 116   WEDNESDAY NOVEMBER 12 1986    3:55:15 AM
  0 ( Individual task listings w/addresses)               ( 081886 WEM)
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
```

```
117 LIST
SCR # 117   WEDNESDAY NOVEMBER 12 1986    3:55:37 AM
  0 ( Individual task listings w/addresses)          ( 081886 WEM)
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 ok
118 LIST
SCR # 118   WEDNESDAY NOVEMBER 12 1986    3:55:43 AM
  0 ( Individual task listings w/addresses)          ( 081886 WEM)
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 ok
119 LIST
SCR # 119   WEDNESDAY NOVEMBER 12 1986    3:55:51 AM
  0 ( Listing of items to be analyzed, in order)     ( 091286 WEM)
  1 : ITEMS CREATE ] DOES> SWAP 1- 4* + @ EXECUTE ;
  2 : END-ITEMS 0 STATE ! ; IMMEDIATE
  3 ITEMS TAPE#
  4  TUBE.CONNS.1
  5  TUBE.CONNS.2
  6  TUBE.CONNS.3
  7  TUBE.CONNS.4A
  8  TUBE.CONNS.4B
  9 END-ITEMS
 10
 11
 12
 13
 14
 15
```

APPENDIX C
LIST OF ERRORS TALLIED AND THEIR DEFINITIONS

APPENDIX C

LIST OF ERRORS TALLIED AND THEIR DEFINITIONS

The operator errors, as defined below, were logged into the data file on the data acquisition unit by the operator in charge of the unit. An operator trained to note these errors was stationed in the high-bay area with the slave arms and communicated the type of incident when it happened to the data acquisition operator by means of a 2-way radio. The errors are defined as follows:

1. Miscenter. A miscenter incident occurs when the point of contact between the tong and a task component (or a grasped object and a task component) is not at the correct location.
2. Misalign. A misalign incident occurs when the axis of motion of the tongs or an object held by the manipulator is not aligned with the required motion axis in the task area.
3. Slip. A slip occurs when an item being held by the tongs slides out of position but is not released. When the item is released, the event becomes a drop. There are two levels of slip incidents. The first occurs when the position of the item does not require re-alignment by the test operator before continuing the task. If the operator

does need to realign the item, then the incident becomes the second level of a slip, a re-position item incident.

4. Collision. A collision involves accidental contact between the manipulator or a grasped object and any object in the task area.

5. Miss. A miss occurs whenever an operator fails to make contact with an object he is attempting to grasp.

6. Press. A press occurs whenever an operator fails to make contact with an object he is attempting to grasp.

7. Entangled. An entangled incident occurs when the manipulator slave becomes tangled in some manner with an object at the test site.

8. Damage. A damage incident occurs any time the manipulator or test area object is defaced or destroyed.

9. Drop. When an item which is grasped by the tongs is released accidentally, a drop is recorded.

10. Incomplete release. When the operator attempts to release a grasped object but the tongs remain in contact with the object.

11. Stop task. Any incident which requires stopping the timer before the task is completed is a stop task event.

12. Other. Any event which affected task performance and is not one of the defined incidents should be noted and described.

VITA

Wendy Ellen Moore was born January 17, 1962 in Winchester, Tennessee, to Floyd Glenn Moore, Jr., and Gracie Burrum Moore. Wendy graduated in May, 1980 from Franklin County High School and enrolled in the University of Tennessee, Knoxville on a valedictory scholarship.

While an undergraduate, Wendy was a co-op student with the U.S. Army Corps of Engineers at Arnold Engineering Development Center in Tullahoma, Tennessee. Extra-curricular activities included serving as a Vol Timette for the UT Men's Swim Team, a little sister for Alpha Gamma Rho Fraternity, a Tennessee Ambassador, and serving on the state officer team for the Future Farmers of America. Wendy also served as President of her Tau Beta Pi Pledge Class and Secretary of Tau Beta Pi her senior year. She graduated in June, 1985 with a Bachelor of Science in Mechanical Engineering with honors from the University of Tennessee.

Following graduation, Wendy accepted a research assistantship through UT with Oak Ridge National Laboratory in the area of Robotics Research and Development. It was through this work that the thesis study presented here was completed while working with the Fuel Recycle Division of the Consolidated Fuel Reprocessing Facility. She graduated in December, 1986, with a Master of Science in Mechanical Engineering. Her career

interests include working in the area of control systems, particularly in the field of robotics.