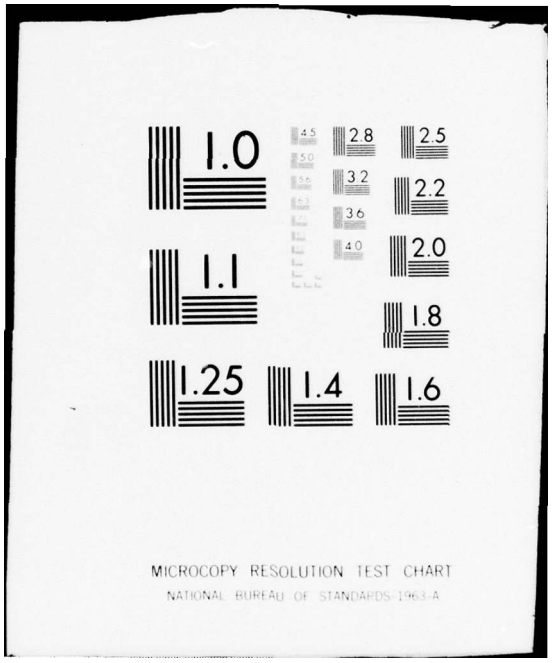
AD-A057 961 PERCEPTRONICS INC WOODLAND HILLS CALIF F/G 5/8
MAN-MACHINE COMMUNICATION IN COMPUTER-AIDED REMOTE MANIPULATION--ETC(U)
MAR 78 W H CROOKS, E SHAKET, Y ALPEROVITCH N00014-76-C-0603
UNCLASSIFIED PATR-1034-78-3 NL

1 OF 2
AD
A057961


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL II

Technical Report PATR-1034-78-3
Contract N00014-76-C-0603
NR 196-140
March 1978

(12)

MAN-MACHINE COMMUNICATION IN COMPUTER-AIDED REMOTE MANIPULATION

WILLIAM H. CROOKS
EFRAIM SHAKET
YORAM ALPEROVITCH

Prepared For:

ENGINEERING PSYCHOLOGY PROGRAMS (CODE 455)
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

PERCEPTRONICS
6271 VARIEL AVENUE • WOODLAND HILLS • CALIFORNIA 91367 • PHONE (213) 884-7470

NOTES

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any office of the United States Government.

Technical Report PATR-1034-78-3
Contract N00014-76-C-0603
NR 196-140
March 1978

126 p.

MAN-MACHINE COMMUNICATION IN COMPUTER-AIDED REMOTE MANIPULATION

Technical rept. 2 Feb 77 - 1 Feb 78

WILLIAM H. CROOKS
EFRAIM SHAKET
YORAM ALPEROVITCH

Prepared For:

ENGINEERING PSYCHOLOGY PROGRAMS (CODE 455)
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

PERCEPTRONICS
6271 VARIEL AVENUE • WOODLAND HILLS • CALIFORNIA 91367 • PHONE (213) 884-7470

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER PATR-1034-78-3	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* MAN-MACHINE COMMUNICATION IN COMPUTER-AIDED REMOTE MANIPULATION		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL REPORT 2 Feb. 1977 to 1 Feb. 1978 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) William H. Crooks Efraim Shaket Yoram Alperovitch		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0603
9. PERFORMING ORGANIZATION NAME AND ADDRESS Perceptronics, Inc. 6271 Variel Avenue Woodland Hills, California 91367		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR196-140
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research (Code 455) 800 N. Quincy Street Arlington, Virginia 22217		12. REPORT DATE March 1978 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Unclassified - Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

Approved by: — [signature]

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Man-Machine Communication
Remote Manipulators
Computer-Aided Control
Procedural Nets
Supervisory Control
Resolved Motion Control
Hierarchical Planning
Shared Man-Computer Control
Symbolic Command Language

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Computerized control offers the possibilities of improved performance times and reduced operator work loads with underwater work systems. Computers can be used at various levels of control, ranging from control augmentation where the computer performs difficult coordinate transformations which simplify operator control requirements through complete autonomy in which the computer performs all of the required activities with no intervention by the operator. However, with the introduction of computer-based control techniques, the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

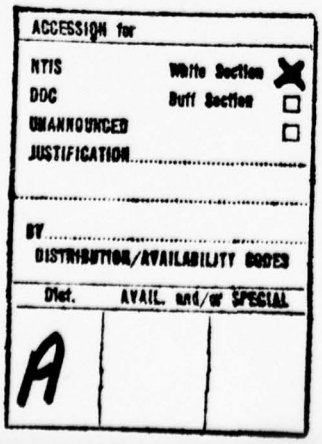
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

communication between the operator and the underwater device becomes an important determinant of work system performance. Rather than controlling directly every action of the manipulator, the functions of the operator of a computer-controlled manipulator are to plan the tasks, command goal-directed actions, monitor task performance, and intervene when appropriate. This paper describes the initial results of a research program directed toward the investigation and optimization of man-machine communication in computer-aided remote manipulation.

A laboratory study was performed with a number of operators who used a six degree-of-freedom manipulator to perform a variety of tasks which are representative of underwater maintenance tasks. The computer-aided manipulator system incorporated several computer-assistance functions, including transformation of control coordinates (Resolved Motion Control) and automatic manipulator movements. The operators performed the series of tasks in an integrated, sequential order, using the full range of available control capabilities.

The results of the experimental investigation indicated that computer aiding can significantly decrease task performance times for a number of remote manipulation tasks. The results also indicated that if higher-level computer aiding schemes are to be effective in terms of improving man-machine performance, the design of the communications language and interface must be carefully designed to achieve a man-machine interaction which is natural, simple, and understandable. Further experimental work will focus on the elements of the command language and controller interface which an operator should have to use a computer-aided remote manipulator system effectively.

TABLE OF CONTENTS



1. INTRODUCTION

Although remote manipulators have been in regular use for over 30 years, relatively little research has been done to address the question of real-time communication between the human operator and the manipulator. As a result, significant advances have been demonstrated in the construction, capabilities and reliability of manipulators, while few changes have occurred in the methods of controlling them. This paper describes the initial results of a program to develop and evaluate improved methods of communicating with and controlling a computer aided remote manipulator.

1.1 Manual Control

Remote manipulators are general purpose, dexterous, cybernetic machines (Corliss and Johnson, 1967) which allow man to extend the functions of the human arm and hand into hostile environments where he himself cannot function. Unlike other complex machines which are preprogrammed to do specific tasks, remote manipulators, also called teleoperators, are able to perform a variety of tasks in changing environments. This versatility is possible both because of the multiple and complex actions which the manipulators can make and because the human operator maintains close supervision and control. Historically, two basic manipulator control techniques have been used, (1) direct position or rate controllers, and (2) master-slave controllers. In the former case, the operator controls each of the five, or more degrees of freedom individually via switches, potentiometers, joysticks, or other such actuators (Corliss and Johnson, 1967). In most systems, the operator's direct view of the work area is used as the feedback channel. With such a control mechanism, the operator must produce smooth and coordinated analog signals, a task at which he is at best slow and inexact, and getting worse as the number of signals increases.

Master-slave control arrangements were developed as early as the late 1940s to overcome the limitations of direct controllers (Goertz, 1952). Master-slaves are bilateral teleoperators in which forces and torques at the master control, located out of the hostile area, are proportionally reproduced at the slave station and vice versa for force feedback. This control arrangement replaces the conscious control of several independent manipulator joints with the natural human capability to control the arm and hand. Operation of the master-slaves is natural, and the operator easily projects himself into the work area. Master-slave systems are among the most common teleoperators in operational use, mainly in radioactive material handling laboratories.

Direct rate or position controllers and master-slave controllers require different command inputs from the operator. However, both techniques require continuous control inputs from the operator and direct visual or force feedback from the working environment. Therein lie the limitations. Both control methods are extremely slow in comparison to direct human capabilities (Pesch, Hill, and Klepser, 1970); thus, human control not only consumes valuable time but also severely taxes an operator's attention. Attention is demanded particularly in those cases where the operator has to overcome built-in system deficiencies such as time delays, no force feedback, high or unnatural inertial forces, etc. Ferrell (1965) has shown experimentally that time delays beyond a fraction of a second substantially disrupt the control process for a simple manipulator with only visual feedback. Subjects avoid the slow and erratic movements that delays tend to induce by consistently adopting the move-and-wait strategy, resulting in very high task completion times.

For underwater manipulation, the limitations of manually controlled manipulators are compounded by the problems of visibility. Control of present day underwater manipulators depends almost entirely on visual feedback. Experience in laboratory studies and field work has shown that

such devices are virtually impossible to use when vision is degraded by turbid water, by poor angle of view, by light failure, etc. These conditions occur frequently in deep sea operations, particularly during bottom work. Expedient completion of underwater tasks, for other than the most simple and most amenable to manipulator accomplishment, is an acknowledged shortcoming of present systems (Rechnitzer and Sutter, 1973). For example, Pesch, Hill and Allen (1971) investigated the performance capabilities of underwater manipulator systems and concluded that "the tasks which were successfully performed were more the result of the ingenuity of the operator utilizing the existing hardware, than the result of the hardware augmenting the operator's basic abilities to perform the task."

1.2 Computer Aided Control

One method to improve the slow and often error prone performance of remote manipulators is to augment or automate most of the operator's control responsibilities. This approach allocates functions to the human operator and to a computer. Such an allocation retains the favorable attributes of human intelligence and foresight, and combines these attributes with the advantages of automatic, computer controlled operation.

Computers can be used at various levels of control, ranging from control augmentation through complete autonomy. In the totally automatic mode, the machine performs all of the required activities with no intervention or control by the operator. The most common method of controlling a fully autonomous manipulator is to utilize a computer-like language which is used to program sequences of actions. Manipulator motions are broken into primitives, and complex trajectories are achieved using control structures similar to those in other real time programming languages. The system described by Wang (1976) is a good example of this

approach with the language ALPHA designed for manipulator control (Nevin, Sheridan, Whitney, and Woodin, 1973). Nevin et al (1973) and Nevin and Whitney (1978) describe another successful system which is programmed by "showing" the manipulator through the detailed motions of a task. Then, using a specially designed, passively compliant gripper, the computer controlled manipulator can assemble a small production electric pump, made of 17 parts, in less than two minutes. However, all the parts must arrive through specially designed feeding stations. Preprogrammed control is the method most commonly adopted for industrial robots and, in number, far exceeds all other approaches to manipulator control. For industrial use, where the environment can be controlled and the reprogramming time is considered short, the reprogrammed manipulator provides a cost-effective alternative when compared with specially designed and built machinery.

The artificial intelligence community has tried, with limited success, to overcome the problems of preprogrammed control and controlled environments. Various systems, including LAMA by Lozamo-Perez and Winston (1977), AL by Finkel, Taylor, Bolles, Paul, and Feldman (1974), and Fikes and Nilsson's (1971) STRIPS have achieved varying degrees of success in automatically controlling manipulators and robots in semi-structured environments. Although these examples and others are promising, the state of the art in artificial intelligence is not yet capable of devising a fully autonomous control system for a robot or a manipulator. The problems lie in (1) recognition of realistic three-dimensional objects, (2) problem solving in an incompletely known environment, (3) planning, and (4) execution with proper recovery from errors, blunders or changes of the environment. Until these problems are solved, it is unlikely that intelligent, fully automatic manipulators will displace the human operator completely within the foreseeable future.

The "middle ground" between the manually-operated and the fully-autonomous manipulator includes various techniques of computer-assistance to augment the human operator. Encompassing many techniques, this middle ground has been termed "supervisory control" (Ferrell and Sheridan, 1967) because the operator "supervises" the operation of a manipulator, selecting automatic functions where available and expedient, or assuming direct manual control when required. Two basic computer-assistance functions can be identified within supervisory control: (1) augmented control, and (2) automatic control.

Augmented control includes those techniques in which the operator remains in control of the manipulator; however, the computer performs some function to facilitate the operator's performance. Resolved motion control (RMC), first described by Whitney (1969), is a form of augmented control designed to obtain coordinated end-point movement. The objective of RMC is to relieve the operator of part of the control responsibility. For example, most of the current control systems in use by the Navy are joint-by-joint rate controllers (Rechnitzer and Sutter, 1973). Rate controlled manipulators usually have one motor to power each joint. Power to each motor is usually controlled by a separate button or switch. Several motors must be operated simultaneously and at different rates to produce end-point movement along a natural coordinate (e.g., "move to the left"). In resolved motion control, the operator specifies direction and speed of the end-point along some natural coordinate axis. The computer calculates the joint angles required to move the manipulator along the commanded coordinate system. The operator is thus relieved of the tedious and time-consuming task of controlling several joint motors simultaneously (Mullen, 1973).

The second form of computer-assistance is automatic control, in which the computer assumes command responsibility for one or more subtasks.

The computer executes the subtask with little or no intervention required by the operator and returns control to the operator when the subtask has been completed or when the computer encounters difficulty that it cannot overcome. Such automatic subtasks can be preprogrammed or the computer can learn the task from the operator (Freedy, Hull, Lucaccini, and Lyman, 1971). The automatic functions can proceed "blind" once they are initiated or they can be equipped to respond to sensory feedback, including force feedback (Groome, 1973; and Woodin, Whitney, and Nevins, 1973), tactile sensing (Goto, 1972), force grip feedback (Ueda and Iwata, 1973), and proximity sensing (Bejczy, 1974).

As we extend shared control of a manipulation system to the full range of capabilities afforded by the computer element, the question of man-machine communication becomes of primary importance. In any manipulation task the operator must observe the actions of the manipulator, make judgments of the commands necessary to perform the task, and carry out those judgments in terms of manipulator control. However, the involvement of the operator in performing these tasks is strongly affected by the degree of assistance provided by the computer. When using unaided manual control, the question of communication between the operator and the machine is relatively straightforward, involving continuous operator control and observation of every machine action. Introduction of supervisory control techniques changes the character of the relationship between the operator and manipulator. In supervisory control the operator not only provides direct analog control of the manipulator's movements, but he also (1) selects any of a number of computer-assistance functions, (2) monitors the progress of automated routines, (3) is able to resume manual control, and (4) knows what control mode is currently operating. A wide range of communication modes, encompassing more than simple analog control, is required when augmented remote manipulators are used.

1.3 Command Language

The present program focused on the required structure and form of the language exchanged between an operator and remote system, i.e., a computer plus manipulator. Ferrell (1973) has shown experimentally that an artificial, constrained, and standardized language facilitates performance in manipulative tasks better than a free format, English-like language. Ferrell concluded that the advantage comes from the fact that, although entire manipulation task goals are readily and perhaps most easily described by a person using ordinary English, the complex geometrical and temporal configurations of objects and motions are not readily formulated in such a language format. A structured but flexible, task-oriented, artificial language can enable the operator to work more efficiently as well as simplify the process of machine translation.

Verplank (1967) compared two classes of commands for a manipulator, (1) symbolic commands, given through an alphanumeric keyboard, and (2) analogic commands, given through joysticks. The results suggested that to indicate direction, magnitude and time-related motions, analogic commands are much faster, easier, and more natural for the human operator. Abstract concepts, however, such as goals or tasks, are more easily specified using symbolic commands. The problem, then, is to provide a communication language which combines both the analogic and symbolic commands into one structured, task-oriented format.

A useful way to conceive of such a structured language is first to define a model of the tasks which are performed by remote manipulators. For our purposes, a hierarchical model of tasks, based on the concepts of procedural nets (Sacerdoti, 1973 and 1975), is appropriate. In this model, an overall manipulation task is seen as a tree-like structure, with individual manipulator motions combined into subtasks, which are then

combined into one overall task. Each node of the tree structure consists of (1) a goal statement - what must be accomplished by the task, (2) an object on which the action is performed, and (3) an action - the sequence of subtasks, expressed as lower "branches" of the tree structure. This information is represented as a planning procedure in each node of the tree. The planning process is activated when a global task request is made at the node in question. The activation initiates the procedures within the net node which first evaluate the current state of the environment and then generate an appropriate sequence of subtasks which, when accomplished one after the other, will satisfy the goal. These subtasks are, in turn, also activated and the procedures within each respective nodes develop the plan to finer levels of detail. At the final stage, the top nodes at the bottom of the tree are connected by sequencing links. These linked nodes represent the sequence of detailed subtasks which will accomplish the overall task. If the process is carried down to the level where each tip node is a manipulator-executable primitive action, the linked sequence of tip nodes in the tree represents the time sequence of manipulator actions which will accomplish the task.

Figure 1-1 illustrates the procedural net model of the task "Shut Valve." At the top level, the task description is independent of valve type and valve location so it can be used in many ways as a step in more general plans. The parameter that is needed at this level is an indication of which valve is to be closed. One level lower in the task, "bring gripper to valve", the valve location is needed. At the level underneath it, a specific manipulator configuration must be decided upon. At the most primitive level, individual motor actions must be specified. As the plan is developed from the top down, more concrete facts about the environment and the configuration of the manipulator in relation to it must be incorporated into the developed plan.

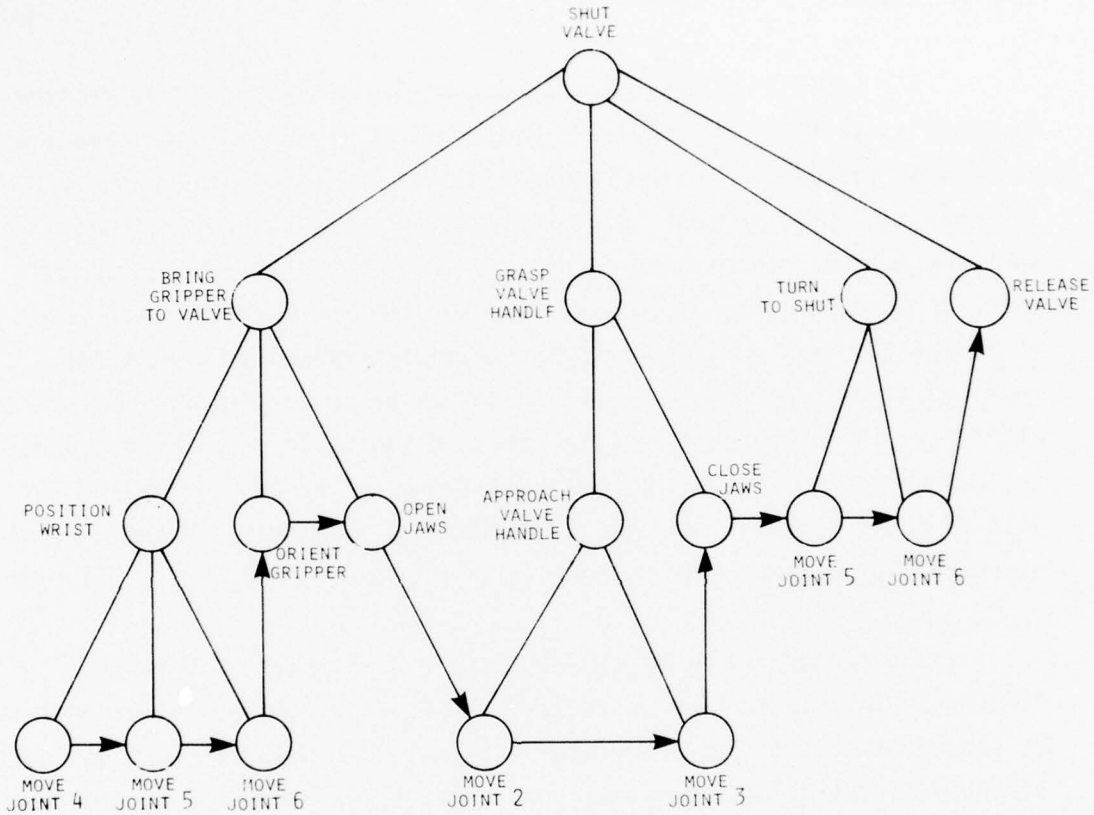


FIGURE 1-1. PROCEDURAL NET FOR "SHUT VALVE" TASK

The procedural net model can represent not only the hierarchy of tasks but also the hierarchy of commands within the communication language itself. Figure 1-2 illustrates a procedural net as a model of man-machine communication.

The operator conceives the task at a high level of abstraction and develops it to some intermediate level. Here, the task is represented as a chain of subtasks. The operator has to communicate each of these subtasks, in turn, indicating the sequencing relations between them. The subtasks are now represented internally in the computer, shown as the nodes at the top level of the machine-developed part of the net. Each node is actually a task description complete with termination conditions, possible error reports, and links to the tasks to be performed before and after it. The computer can, using the procedures in these nodes, develop the plan autonomously to the machine level. The arrows connecting the tips of the procedural net are the sequencing links between the manipulator executable primitive tasks.

The procedural net model suggests a basic structure for the language. It is a task-oriented language with provisions to define tasks as sequences of subtasks. Together with a set of primitive actions defined directly as manipulator motions, such a recursive organization provides the user with the capability to define tasks at arbitrary levels of complexity. The user is doing the high-level planning, and the computer handles the details.

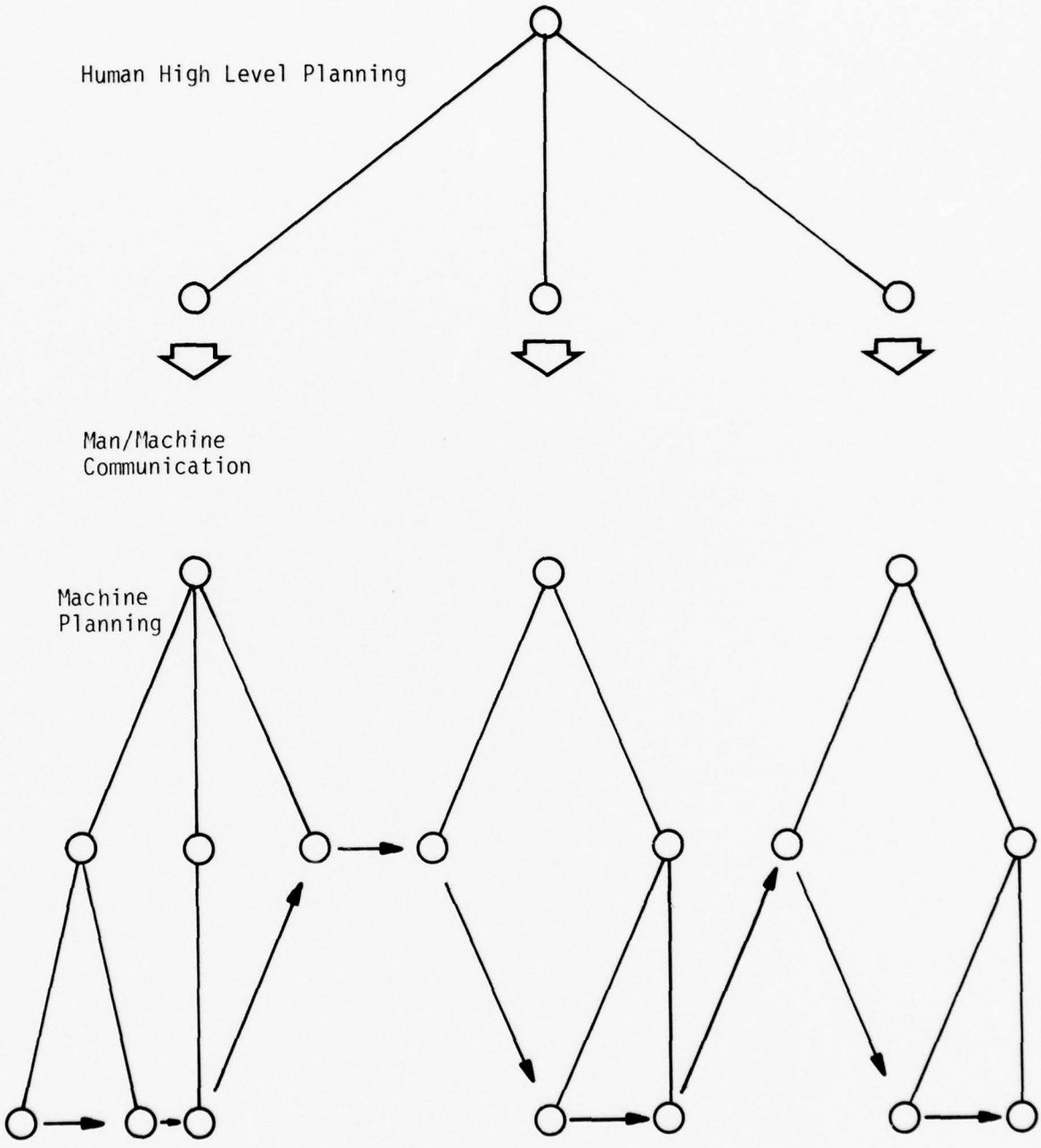


FIGURE 1-2. PROCEDURAL NET MODEL OF MAN/MACHINE COMMUNICATION

2. A COMMAND LANGUAGE FOR SUPERVISORY CONTROL

Based on the results of the previous year's experiment (Berson, Crooks, Shaket, and Weltman, 1977),and analysis of the hierarchical task model, a manipulator command language was developed which would permit an operator to specify manipulator movement in global, whole-task terms. This command language was designed not only to include the capability for symbolic and analogic control as used in the previous experiment, but also to permit the operator to define a sequence of actions and to then call this sequence with a single command string. Moreover, the newly developed command language was designed to simplify the sequence of actions required to define or execute a command.

The command language consisted of two major elements, symbolic and analogic command, which, at first glance, appear to be mutually exclusive. In the first class, symbolic commands are executed by discrete pushbutton actions, resulting in discrete and defined arm motions. Such a command would include ROTATE ⬎ DO which would cause the manipulator wrist joint to rotate clockwise 180°. In the second class, analogic commands are executed by continuous movement of joysticks, resulting in analogous manipulator movements. However, the language also included the capability to "define" manipulator configurations, called "points", and trajectories, called "paths". This "define" function permitted the operator to first move the arm using the analog joysticks, and then to repeat the same motion, or move back to the same place in the work space by using symbolic commands. Thus, the distinction between analogic and symbolic commands refers primarily to the method of controlling the manipulator movements, rather than to the resulting sequence of movements.

Within the class of symbolic commands, the newly developed command language consisted of two categories of commands: primitive and variable.

The Primitive commands are fixed in the sense that they produce unvarying manipulator motions whenever they are used. They are system-defined rather than user-defined. These commands include the conceptual unitary motions which are useful and necessary for performing a variety of tasks.

Primitives are the lowest level in the symbolic command hierarchy and provide the basic elements for higher level commands. For the tasks used in the present experiment, the following primitive commands were used:

(1) FORWARD - moves the manipulator forward along the current axis of the effector (gripper) for a distance equal to the length of the gripper.

(2) BACKWARD - reverse motion of "FORWARD".

(3) GRASP - closes the gripper.

(4) RELEASE - opens the gripper.

(5) ROTATE ↷ - rotates the gripper clockwise.

(6) ROTATE ↶ - rotates the gripper counterclockwise.

(7) MANUAL - stops automatic arm motions and waits for operator commands (symbolic or analogic).

The Variable category of commands are the user-defined commands which provide the capability to construct task specific commands. This feature of the command language is especially important in uncontrolled environments where the required programming is not known in advance.

Variable commands allow the user to record complex combinations of manipulator actions and link space positions, which can be subsequently activated by a single command. Therefore, when employing a variable command, the user conceives of the task at a high level of abstraction and the computer implements the lower level details of the pre-recorded manipulations. These variable commands include:

(1) POINT - moves the manipulator to a previously recorded position.

(2) PATH - moves the manipulator along a previously recorded trajectory.

(3) REVERSE PATH - moves the manipulator along a path from end to beginning.

(4) CHAIN - moves the manipulator according to a recorded sequence of Primitive and Variable commands.

Under the GO TO POINT command, the computer calculates the joint motions necessary to produce a straight-line motion between the current position to the recorded position. Under GO TO PATH command, the computer first calculates the motions necessary to move from the current position to the starting position of the recorded trajectory, then moves the arm along that trajectory. Thus, a PATH or REVERSE PATH can be an irregular, curving motion which can be used, for example, to avoid an obstacle. The CHAIN command is a user-defined sequence of Primitive and Variable (including other chains) commands which is defined, labelled, and used as a single unit. The chain command permits the operator to aggregate several unitary manipulator motions together into a single complex motion, thus, commanding tasks which are higher in the hierarchy of tasks. For example, the following sequence of unitary actions are necessary to close a valve:

Unitary Actions	Commands
	CHAIN 1
1. Open gripper	RELEASE
2. Rotate gripper counterclockwise	ROTATE ↶
3. Move forward with gripper over valve	FORWARD
4. Close gripper	GRASP
5. Rotate valve clockwise 180°	ROTATE ↷
6. Open gripper	RELEASE
7. Move backward away from valve	BACKWARD

The primitive commands necessary to execute this sequence are listed in the right hand column. As shown, this sequence has been labelled CHAIN 1, and these seven valve-turning steps will be executed whenever CHAIN 1 is commanded. A second, more comprehensive chain could then be defined to close the valve located at point 6 as follows:

CHAIN 2

GO TO POINT 6
GO TO CHAIN 1
GO TO CHAIN 1

This chain would cause the manipulator to move to point 6, which has previously been recorded in front of the valve, and to open the valve one complete revolution (360°).

Given these two categories of symbolic commands, the command language was designed to have a general *verb-noun-parameter-terminator* syntax. While not all command strings use all four syntax elements, the sequence order is the same for all commands. This syntax was preserved and facilitated by the keyboard which was designed to implement the command

languages. As shown in Figure 2-1 , the command word classes, i.e., verbs, nouns, parameters, and terminators, were grouped together in color-coded columns (indicated by various shadings in the figure). The flow of hand movement in using the verb-noun-parameter-terminator syntax was consistently from left to right.

The Variable category of commands require the leading verb to indicate the nature of action to be taken. The verb phrase "GO TO" is used to execute any of the four commands POINT, PATH, REVERSE or CHAIN. The DEFINE command is used to record the analog motion of the POINT and PATH commands and to record the sequence of primitive commands with the CHAIN command. The DELETE command is used to erase any previously recorded command while the END command is used to terminate recording either Paths or Chains.

The resultant actions of the noun commands have been described previously. As shown in Figure 2-1, the Primitive nouns (FORWARD, BACKWARD, GRASP, RELEASE, MANUAL, and ROTATE) are grouped separately from the Variable commands, since the syntax of former commands does not require a leading verb. The third class of words, the parameters, are grouped together in the form of a 1-2-3 keypad. The manipulator controller was designed to accommodate a number of Variable commands. In the system used for the present experiment, each of ten Points, Paths and Chains could be recorded. The operator assigned a number to each recorded action while defining the action. That number could then be used as the unique identifier to refer to the recorded action.

Finally, the command language syntax requires a command terminator. This procedure allows the operator to select one of several methods of command execution. In the typical sequence, the command string would be terminated by DO which would cause the command to be executed as soon as

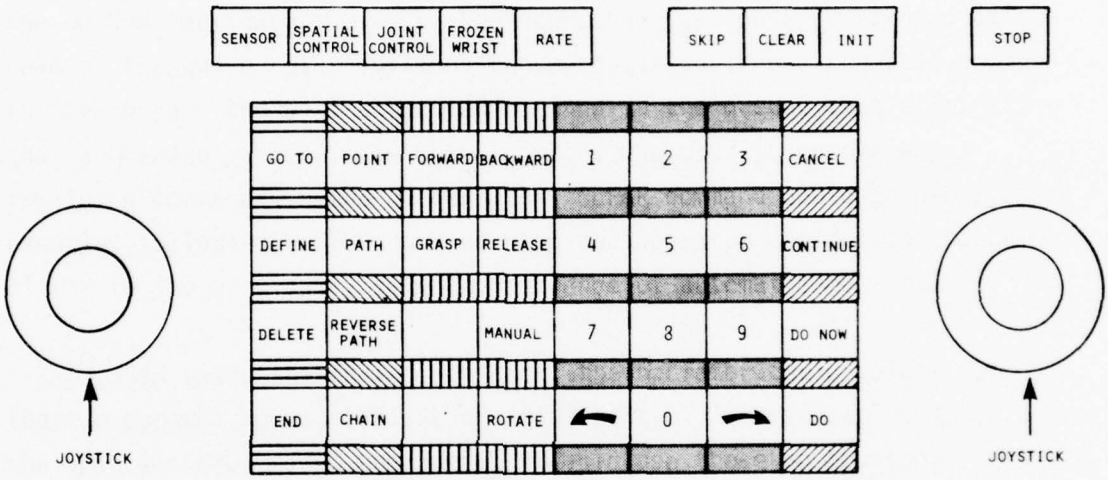


FIGURE 2-1. DEDICATED KEYBOARD ARRANGEMENT

previously requested commands have been completed. If the operator selected the DO NOW terminator, the just entered command was started immediately, prior to the completion of any previously commanded action. If the operator chose not to implement a command string that had just been entered, or if he discovered a pushbutton error, he would use the CANCEL terminator. The CONTINUE terminator was used to restart a Chain in which a MANUAL command had been inserted. For example, the following chain includes a MANUAL command to allow the operator to use the joysticks to make any necessary fine alignments at the end of path 5 before proceeding with a valve turning action. The automatic motion following the MANUAL command is initiated when the operator selects CONTINUE.

CHAIN 3

GO TO PATH 5
MANUAL
GO TO CHAIN 1
GO TO CHAIN 1
GO TO REVERSE PATH 5

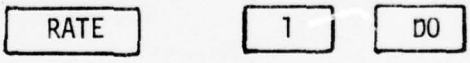
A final command pushbutton, the STOP command, required no other verb, noun, parameters, or terminator. Indeed, as shown in Figure 2-1 , the button itself was located apart from the main keyboard in a conspicuous location above the righthand joystick. The STOP command immediately halts all manipulator motion, thus giving the operator an emergency override capability. By pushing the CONTINUE key, the operator could cause the computer to resume any ongoing automatic motions.

Whenever the symbolic commands were used to control the manipulator motion, the computer first stored the commands in a queue, and then executed each command on a First-In-First-Out basis. As soon as a command

appeared in the queue, that is, as soon as the operator entered a legal verb-noun-parameter-terminator string of commands, the commanded manipulator motion was initiated. Manipulator movement continued, taking each command in turn, until the queue was depleted. The operator could enter commands at any time, with the latest entry added to the end of the queue. The operator need not wait until a previously-commanded motion was completed before entering subsequent commands. As noted previously, the DO NOW terminator was used to insert the just-entered command at the head of the queue. In addition, the operator could use the SKIP command to move beyond the next-scheduled command in the queue, thus eliminating that commanded motion. The CLEAR command was used to remove any remaining commands in the queue. This CLEAR command could be used, for example, following a STOP command when the operator decides to change or eliminate the previously selected sequence of automatic motions.

When using the analog joystick, the operator could select one of the two control modes, SPATIAL or JOINT control. For example, to select the spatial (RMC) mode, the operator would use these pushbuttons:

In addition to the mode controls, the operator could select one of four rates of motion. For example, to select the slowest rate the operator would push the following buttons:



The selected rate controlled the rate of motion for all manipulator movements, whether commanded via the symbolic pushbuttons or the analog joysticks. The operator could use the analog joystick either alone or in combination with the symbolic pushbutton commands. That is, the operator could control a manipulator motion solely by using the joysticks

or he could initiate an automatic manipulator motion via the pushbutton keyboard and then use the joysticks to modify or adjust the automatic motion during the actual manipulator movement. In this case, the resulting motion was a vector sum of the symbolically-commanded motions and the joystick-commanded motions.

3. APPARATUS

3.1 Control Station

Figure 3-1 shows a drawing of the manipulator control station consisted of the previously described keyboard with the dedicated pushbuttons and two three-degrees-of-freedom, rate control, self-centering joysticks. The control station also had a 9" CRT display which was used to provide printed information concerning the condition of the system, to respond to keyboard inputs, and to display commands in the execution queue. In the first category, the upper right hand portion of the display screen showed the control mode and rate conditions in the following manner.

CONTROL : SPATIAL
RATE : 2

Keyboard inputs were shown in the lower portion of the screen, just above the keyboard. If any keyboard entry errors were made, an "ILLEGAL ENTRY" message also appears on the lower portion of the screen. As soon as the DO or DO NOW terminator was entered, the command was entered in the queue, which was displayed in the left center portion of the screen. As the commands were executed, the displayed queue list was moved up, such that the command currently in execution was always listed at the top of the queue. When the operator selected DEFINE CHAIN, the right center portion of the screen showed the commands which had been selected for that chain. These various display fields are illustrated in Figure 3-2 . In this example, the manipulator would move at the fastest rate (RATE: 4) and if the joysticks were being used, they would control the manipulator joints directly (CONTROL; JOINT). Chain 4 had been in execution and came to a MANUAL command. During the momentary pause in the motion, the operator has begun to define Chain 5. In the course

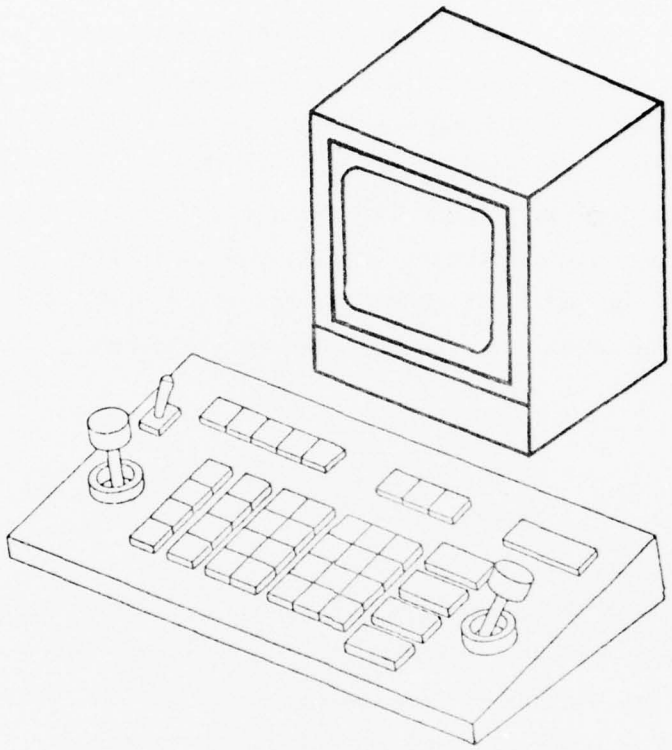


FIGURE 3-1. MANIPULATOR CONTROL STATION

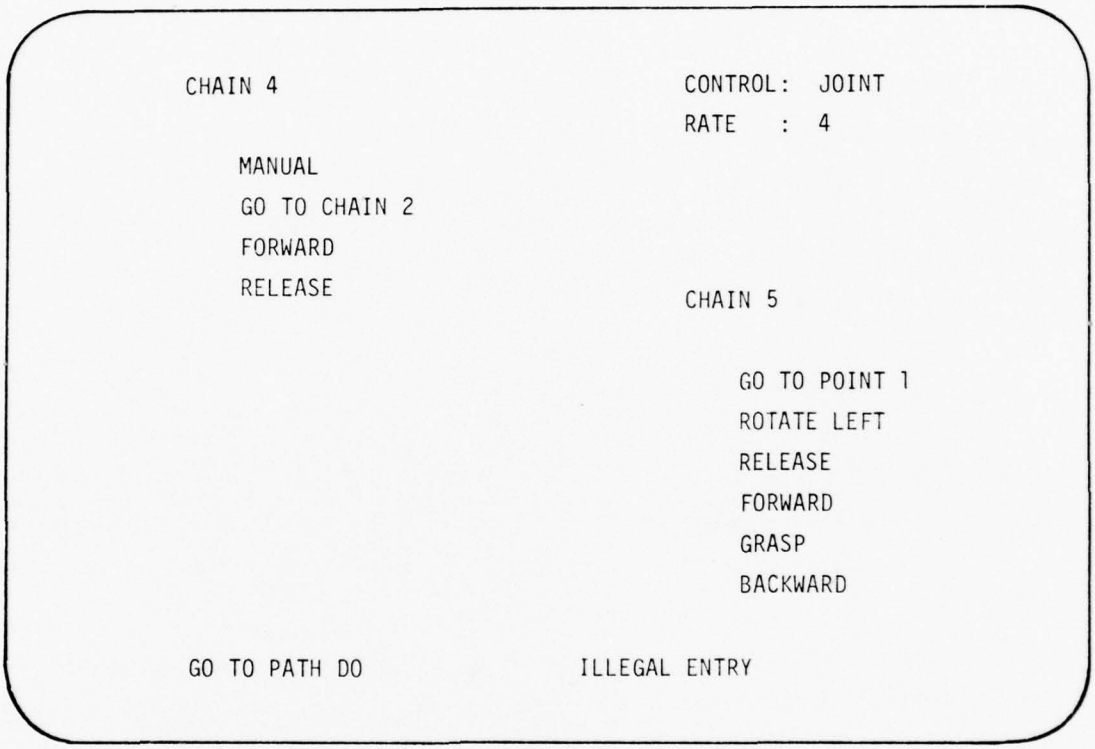


FIGURE 3-2. DISPLAY DURING EXECUTION OF CHAIN 5

of defining this chain, he has entered the command GO TO PATH DO which has been identified as an Illegal Entry since the path number was not listed.

3.2 Manipulator Facility

The Servo arm manipulator, illustrated in Figure 3-3, was used as the testbed for the present experiment. This manipulator, described previously by Berson, et al (1977) is electronically-controlled and hydraulically-powered. The manipulator has six rotating joints (each with a full 180° movement range) plus gripper closure. The arm motions and joint numbers are (in anthropomorphic notation):

(1) Shoulder rotation
(2) Shoulder elevation
(3) Elbow flexion
(4) Forearm rotation
(5) Wrist flexion
(6) Gripper rotation

These motions provide the six degrees of freedom necessary to position and orient an object in the work space. The first three joints position the gripper, while Joints 4, 5, and 6 orient the gripper with respect to the gripper axes.

The manipulator has a work envelope slightly larger than that of a stationary person. The extended manipulator arm length is about 40 inches from shoulder to gripper tip. When using the analog joysticks, the operator had two control modes available. Joint Control and Spatial Control. Joint control gave the operator direct manual control of each manipulator joint angle. Each degree-of-freedom of the joystick

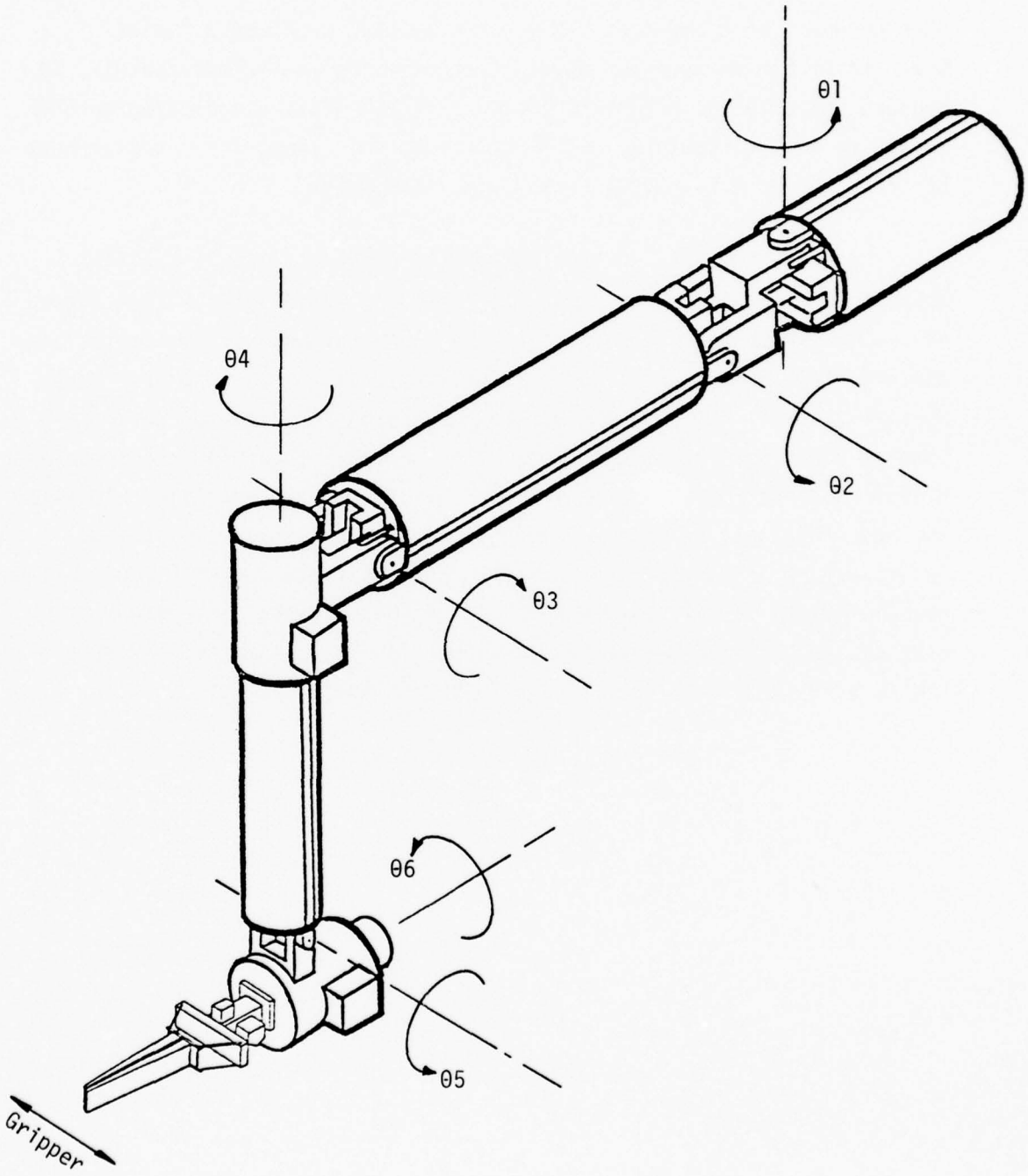
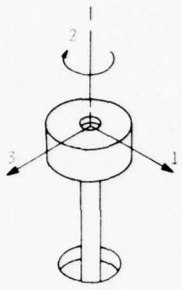


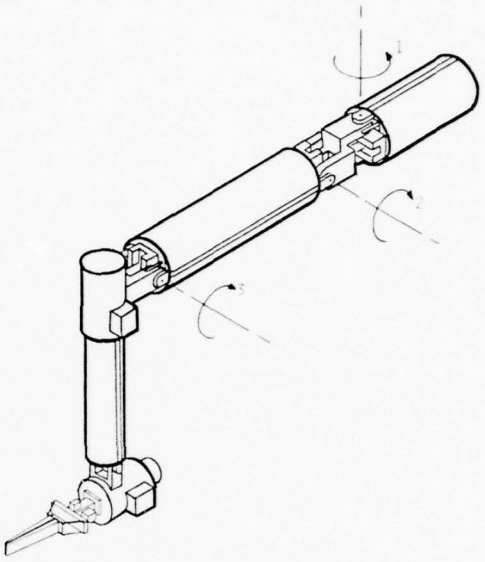
FIGURE 3-3. SERVO MANIPULATOR WITH MOTIONS OF THE SIX ROTARY JOINTS

controllers was associated with a specific manipulator joint. That is, each movement of a joystick along a single axis produced a radial movement of the manipulator about the controlled manipulator joint. To produce end-effector movement along a straight line, coordinated control of two or more manipulator joints was required. Thus, the operator must activate two or more joystick axes simultaneously.

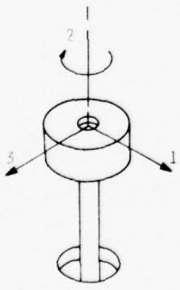
Spatial Control, or Resolved Motion Control (RMC) allowed the operator to move the wrist along task or world coordinates. Each degree of freedom of the joystick is associated with movement of the manipulator end-effector along a specific X, Y, or Z axis of the work space. Under Spatial Control, the computer assumed the responsibility for performing complex coordinate transformations. The operator specified the speed and direction of motion of the manipulator wrist, and the computer calculated the required angle of each joint and output these as commands to the individual joints of the manipulator. Spatial freed the operator from the responsibility of determining the combination of speed and motions to produce the required trajectory. Figure 3-4 illustrates the difference in manipulator motions, given similar joystick inputs.



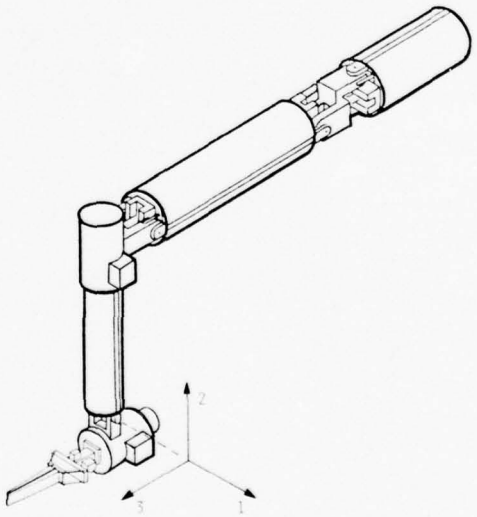
JOYSTICK



JOINT CONTROL



JOYSTICK



SPATIAL CONTROL

FIGURE 3-4. JOYSTICK MOVEMENTS AND CORRESPONDING MANIPULATOR MOVEMENTS

4. METHOD

4.1 Subjects

Six subjects participated in the experiments. They were students at California State University at Northridge; three were male and three were female. Their ages ranged from 20 to 28 years, with a mean of 24.5 years. They had a mean of 14 years of education. None had previously operated a remote manipulator or any similar devices requiring joystick controllers. All subjects were paid for participating in the experiment.

Since normal visual acuity and depth perception appear to be necessary in performing certain manipulator tasks, such as fine alignment, all subjects were screened using the Keystone Visual Survey. All six subjects were within the normal ranges for the eight subtests in the Keystone test battery. These include tests for visual acuity, stereopsis, simultaneous perception, fusion, and depth perception.

4.2 Procedure

The experiment was divided into training and experimental phases. In the training phase, the test subjects were introduced to the various manipulator control functions and they then practiced using these functions to acquire controller skills on several basic manipulator tasks. In the experimental phase, the subjects applied these skills acquired during training to perform a simulated complex maintenance task. The maintenance task integrated the elementary training subtasks into a single task which simulated realistic remote manipulator operations. Data were recorded separately for each phase.

4.2.1 Training Phase. Training was divided into two segments. In the first segment, the participants received a lecture and demonstration of the structure, function, and capabilities of the manipulator system. The objective was to introduce the participants to the manipulator in general, and to the specific procedures for controlling the manipulator. In the second segment, the participants practice basic manipulation tasks in preparation for the experimental phase. The participants received equal amounts of practice using the analog joystick controls, the Fixed and Recorded symbolic commands, and the Chain command. Total training time was approximately 10 hours for each participant, conducted in six sessions ranging from 1-1/2 to 2 hours each.

Training Tasks. During the first hour of training, the participants became familiar with manipulator control characteristics by moving the manipulator in unstructured tasks. After this initial familiarization, all training was conducted on the following structured tasks which had defined goals.

Valve Turning. The valve turning task required the participants to turn a 2" gate valve a total of five one-half revolutions. The valve turn task was used because it is amenable to preprogrammed movement. Participants performed the valve turning task using the Basic language condition or the Basic and Variable language condition. In the Basic language condition, the participant could use only the analog joystick controls plus any of the Primitive symbolic commands. With the Basic and Variable language condition, the participants were also allowed to use the Variable symbolic commands, including Points, Paths, and Chains. The following operations are required to perform the valve turning task:

(1) Align the end-effector over the valve.
(2) Open the end-effector

(3) Move end-effector forward.
(4) Close the end-effector to grasp the valve.
(5) Rotate the manipulator 180°.
(6) Open the end-effector to release the valve.
(7) Move end-effector backward.
(8) Rotate back around the valve 180°.
(9) Repeat steps 3 to 8 for each 1/2 turn.

The participants performed the valve-turning task once under each condition in each of the last five training sessions.

Ring-Placement Task. In the Ring-Placement task, the participants picked up metal rings (2" hole diameter) and placed them on horizontal and vertical posts (.75" diameter). The ring-placement task was used because it involved a variety of basic manipulation elements (i.e., gross movement, fine alignment, grasping and carrying objects, etc.). Also, the repetitive nature of this task provided the opportunity for the participants to appreciate the advantage of the Chain command with such tasks.

In the final three training sessions, the ring-placement task was conducted as an experimental study. During these sessions, the participants performed two consecutive trials of ring-placement under one of two control modes. During each trial, the participant would operate the manipulator so as to pick up each of six rings in succession, placing the first three rings on either the vertical or horizontal post and the final three rings on the other post. Within one such session, the participant might first perform two trials under the Basic language condition and then two trials under the Basic and Variable language

conditions. The order of the two conditions was counterbalanced. When using the Basic and Variable language condition, the participants were allowed to use chains constructed during the first trial when performing the second trial. This procedure makes possible an independent determination of the time to set up a chain and the time to execute a chain. Furthermore, this experimental design provides an assessment of any differences in learning rate that may occur under the two control modes.

4.2.2 Experimental Phase

Experimental Task. During the experiment, the test participants used the various control modes to operate the manipulator in performing a simulated maintenance task. The task included typical underwater manipulations such as tool usage valve activations and part insertions. The tasks were complex and took more than 15 minutes to complete. Like the task used in the previous year's experiment (Berson, et al, 1977), this maintenance task was based on a pipe structure located within the manipulator's work space. Whereas the task used in the previous experiment emphasized gross travel, contour-following, end-effector orientation, etc., the current task emphasized tool usage in addition to the previous task elements. As shown in Figure 4-1, the task structure included two gate valves with square handles, several interchangeable "caps" which were loosened (or tightened) with a screwdriver-like tool after which the manipulator could pick up the "cap", and a removable pipe section which could be removed or replaced by the manipulator. To the left of the pipe structure was a platform with receptacles for the screwdriver tools and extra "caps". To the right of the pipe structure was a similar platform with a receptacle for the removable pipe section. These side platforms were used as workboxes for storing tools and parts.

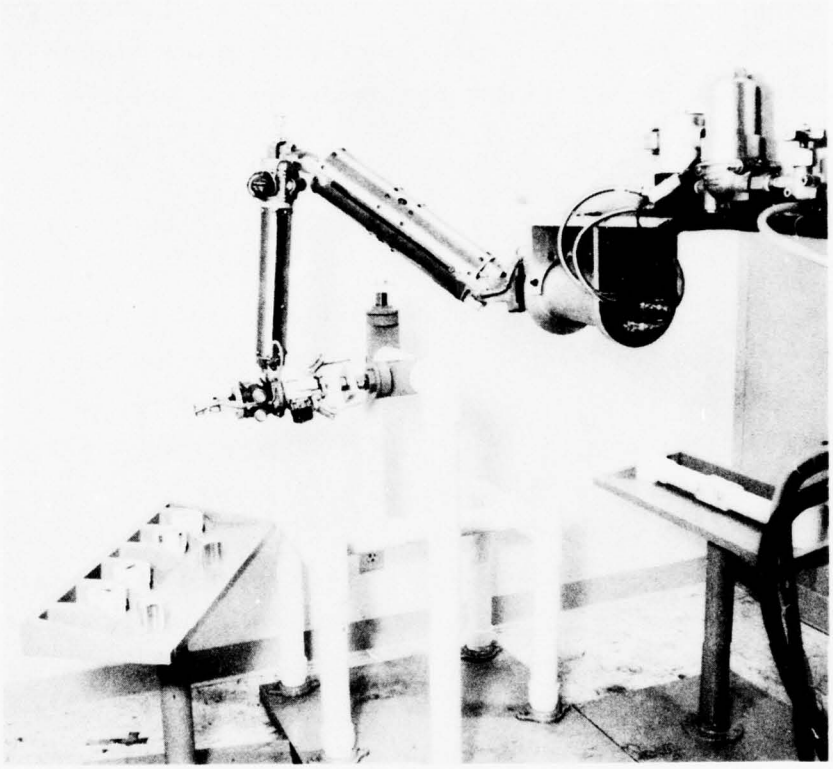


FIGURE 4-1. SIMULATED MAINTENANCE TASK

During the experiment, the participant was instructed to perform the sequence of steps shown in Table 4-1 . Together, these steps constituted a simulated maintenance task of adding a section of pipe to the pipe structure. The list of steps was posted to one side of the control console and the participant was instructed to feel free to refer to it at any time during the task.

4.3 Experimental Design

A 3 x 2 x 2, repeated measures design was used to examine three task conditions, with two levels of the language, over two trials. The three task conditions were Baseline, Increased Variability, and Increased Complexity. The list shown in Table specified the tasks which constituted the Baseline condition. The Increased Complexity condition (more repetitive subtasks) was created by increasing the number of steps in the task sequence. Specifically, this was achieved by requiring the operator to disassemble a portion of the pipe structure before beginning the steps in the Baseline condition. The Increased Variability condition (more subtasks but with no repetition among them) was created by (1) using two distinct caps, each of which could be adjusted by only one of two different screwdrivers and which were located at additional receptacles in the lefthand workbox, and (2) by altering the portion of the pipe structure components. This Increased Variability condition was designed to illustrate the potential advantages or disadvantages of higher level command language capabilities in tasks requiring movements to work-space locations which vary over time or movement to a number of different locations.

As in the training phase, the Basic and the Basic + Variable conditions were used for the two levels of language structure. In terms of the hierarchical task model, the basic language condition allowed the participants to control the task operations only at the lowest,

TABLE 4-1

SIMULATED MAINTENANCE TASK STEPS

I. Go from STOW position to top valve and rotate it five (5) times clockwise.
Define path which goes from STOW position to valve.

II. Transport to side valve and rotate it five (5) times clockwise.

III. Transport to crossbar and put into position (in the structure, bar is released.

IV. Transport to 1st cap and put into position (in crossbar).

V. Transport to tool and tighten cap (completed turn, cap in place).

VI. Return tool to position (in tool box).

VII. Transport to 2nd cap and put into position (in crossbar).

VIII. Transport to tool and tighten cap (completed turn, cap in place).

IX. Return tool to position (in tool box).

X. Transport to side valve and rotate it five (5) times counterclockwise.

XI. Transport to top valve and rotate it five (5) times counterclockwise.

XII. Return to STOW position.

elementary motion levels, whereas the Basic + Variable language condition permitted them to control the operations with commands more closely paralleling whole tasks.

The six conditions created by the two levels of language and three task conditions were conducted in six consecutive test sessions on successive days. The order of the conditions was counterbalanced to control for learning effects. Within a session, participants completed the scheduled experimental task twice in succession. During the second trial of each session, the participants were allowed to use any points, paths, or chains which had been defined during the first trial.

5. RESULTS

5.1 Training Results

Figure 5-1 illustrates the performance times required to complete the ring placement task, shown as functions of the training session and language conditions. An analysis of variance, summarized in Table 5-1 , confirmed the significant improvement in successive sessions. This result reflected the expected improvement with increased practice. The Trials and Control Mode-by-Trials (A x B) interaction effects shows that the participants were significantly faster ($p<.01$) in performing the second trial, particularly when the Variable language commands (i.e., Points, Paths, and Chains) were used. When the points and chains have already been defined, as in the second trial of the Basic + Variable condition, the overall task performance time was improved by 39%. However, the cost of defining the paths or chains was to increase the overall, first-trial performance times by 11%. This result illustrates the potential benefit to be realized by recording repetitive motions with Variable commands for tasks requiring a number of repeated actions. The significant Trials-by Session (B x C) interactions demonstrates the greater practice improvement when using the Variable commands. This result suggests that with extended practice, the time cost of defining the Variable commands may be reduced considerably.

In summary, these training data demonstrate the significant improvement in performance time to be gained when pre-established Variable commands are available to use with repetitive manipulation tasks. Furthermore, these data indicate not only that the use of chains and points significantly improves the performance during the initial introduction to the manipulator system but also that the benefit of the operator-defined Variable commands increases with practice.

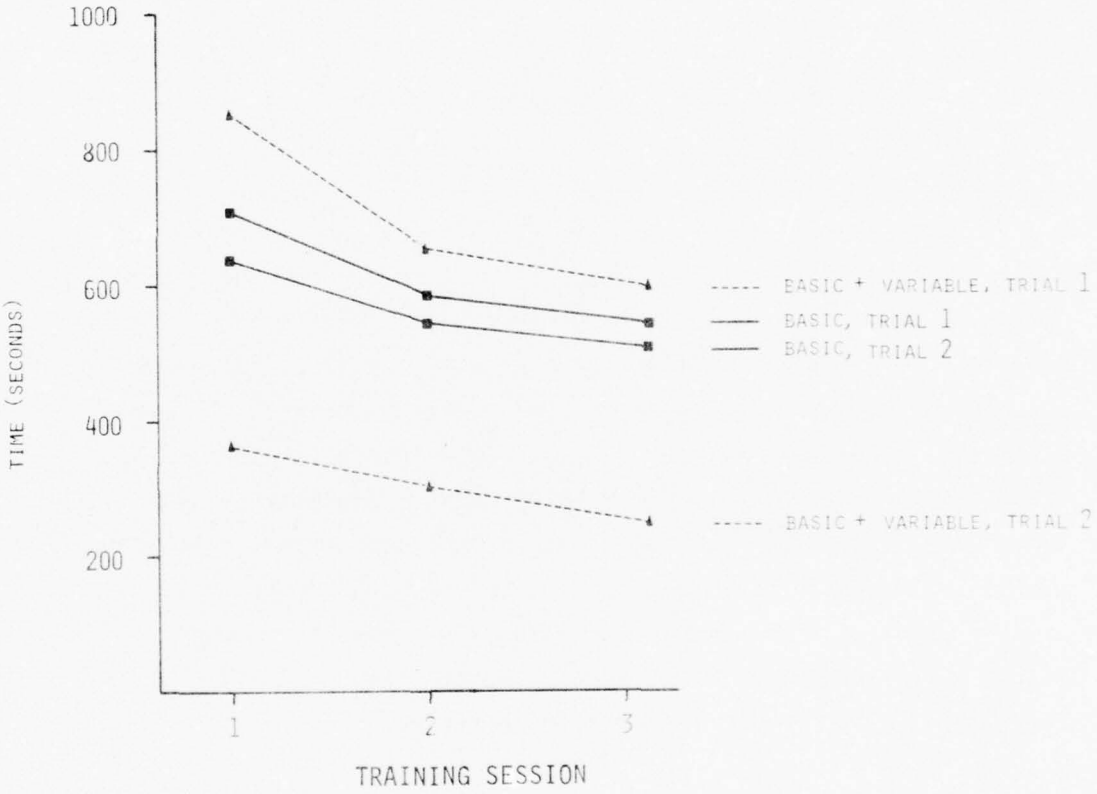


FIGURE 5-1. PERFORMANCE TIME ON RING PLACEMENT TASK AS A FUNCTION OF CONTROL MODE AND TRAINING SESSION

TABLE 5-1. ANALYSIS OF VARIANCE SUMMARY TABLE FOR PERFORMANCE TIME IN RING PLACEMENT

Source	df	MS	F
Control Mode (A)	1	64,440.50	3.81
Trials (B)	1	703,298.00	289.94*
Session (C)	2	240,637.79	11.05*
Subjects (S)	5	98,975.57	
A x B	1	377,870.22	184.91*
A x C	2	807.55	.04
A x S	5	16,923.57	
B x C	2	25,469.05	17.36*
B x S	5	2,425.67	
C x S	10	21,779.76	
A x B x C	2	11,696.69	3.59
A x B x S	5	2,043.49	
A x C x S	10	18,664.51	
B x C x S	10	1,467.41	
A x B x C x S	10	3,253.65	
TOTAL	71		

*$p<.01$

5.2 Test Results

The experimental results in terms of mean time to completion as a function of task, control mode, and trial are shown in Figure 5-2 . The analysis of variance, summarized in Table 5-2, showed that the effects of task and trial number were significant ($P<0.01$). The time relations between the first and second trial demonstrate (again as in the training phase) the definite advantage of the chaining capability, but the percentage impact is much smaller than before in the simple repetitive task. This is representative of effects observed in actual undersea systems where advanced features of the language or the system get partially buried in the varied actions that the operator does during a long and complex task, of which language use is only a small part (i.e., he sits and thinks about what to do next).

As could be expected, the tasks with increased variability and increased complexity each take longer than the base line task. The task with increased variability takes longer because there are a greater number of different subtasks in it. The high complexity task is longer still because it has more subtasks altogether, including repetitiveness.

In addition to recording performance time, contact errors committed during the performance of the maintenance tasks were also recorded. The number of contact errors for each task under both control modes is shown in Figure 5-3. The analysis of variance, summarized in Table 5-3 , yielded a significant Control Mode-by-Task (A x B) interaction. This result illustrates that operator-defined Variable commands are most useful in reducing contact errors in routine tasks or difficult tasks, but that they are not as effective in reducing errors in highly variable tasks in which the manipulator must be moved to a number of positions which change during the course of the task.

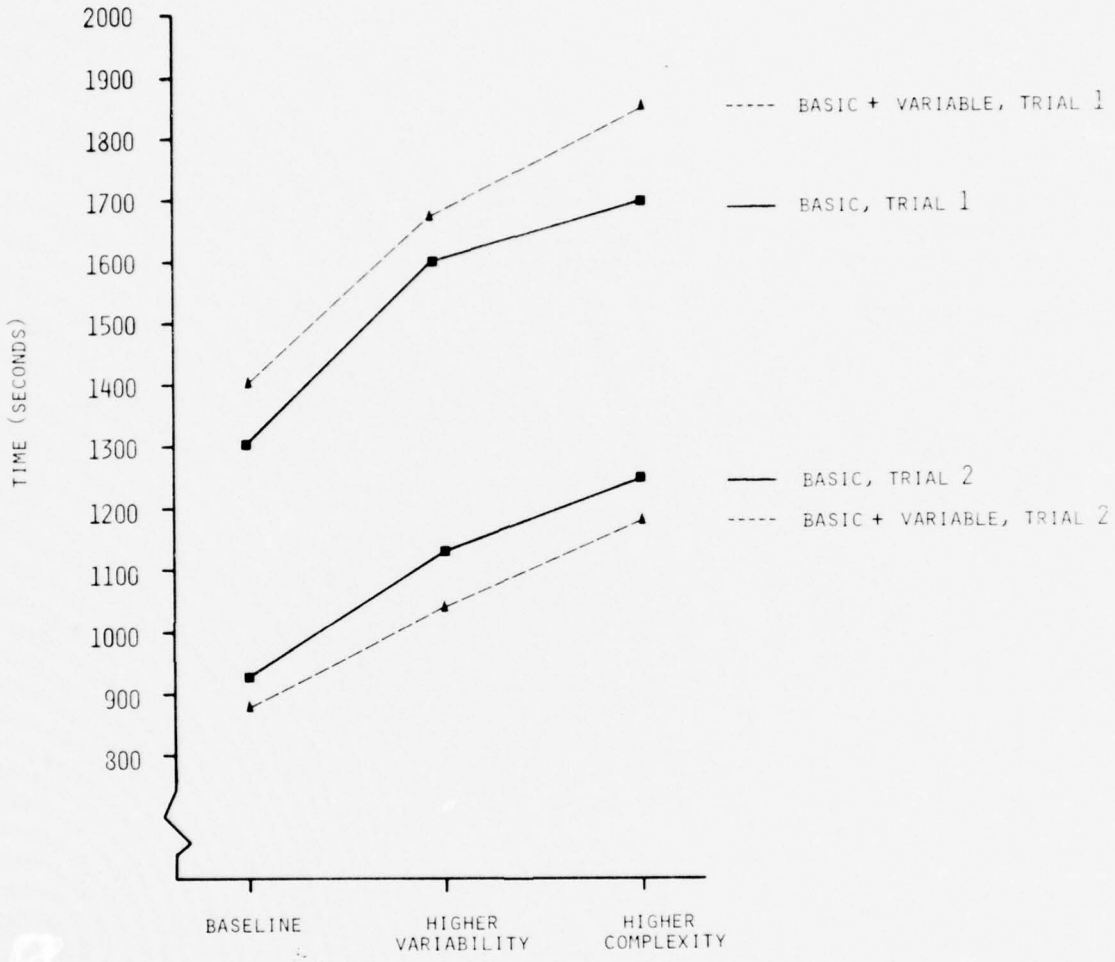


FIGURE 5-2. MEAN PERFORMANCE TIME ON EXPERIMENTAL TASK AS A FUNCTION OF EXPERIMENTAL TASK, CONTROL MODE, AND TRIAL

TABLE 5-2. ANALYSIS OF VARIANCE SUMMARY TABLE FOR PERFORMANCE TIME IN EXPERIMENTAL TASK

Source	df	MS	F
Control Mode (A)	1	2,837.56	.08
Experimental Task (B)	2	906,716.10	13.16*
Trial (C)	1	4,572,288.00	91.89*
Subjects (S)	5	426,007.30	
A x B	2	2,312.72	.09
A x C	1	91,449.39	22.43*
A x S	5	33,925.02	
B x C	2	23,133.50	3.26
B x S	10	68,879.54	
C x S	5	49,759.27	
A x B x C	2	980.39	.07
A x B x S	10	26,917.14	
A x C x S	5	4,076.46	
B x C x S	10	7,098.92	
A x B x C x S	10	15,032.21	
TOTAL	71		

*$p<.01$

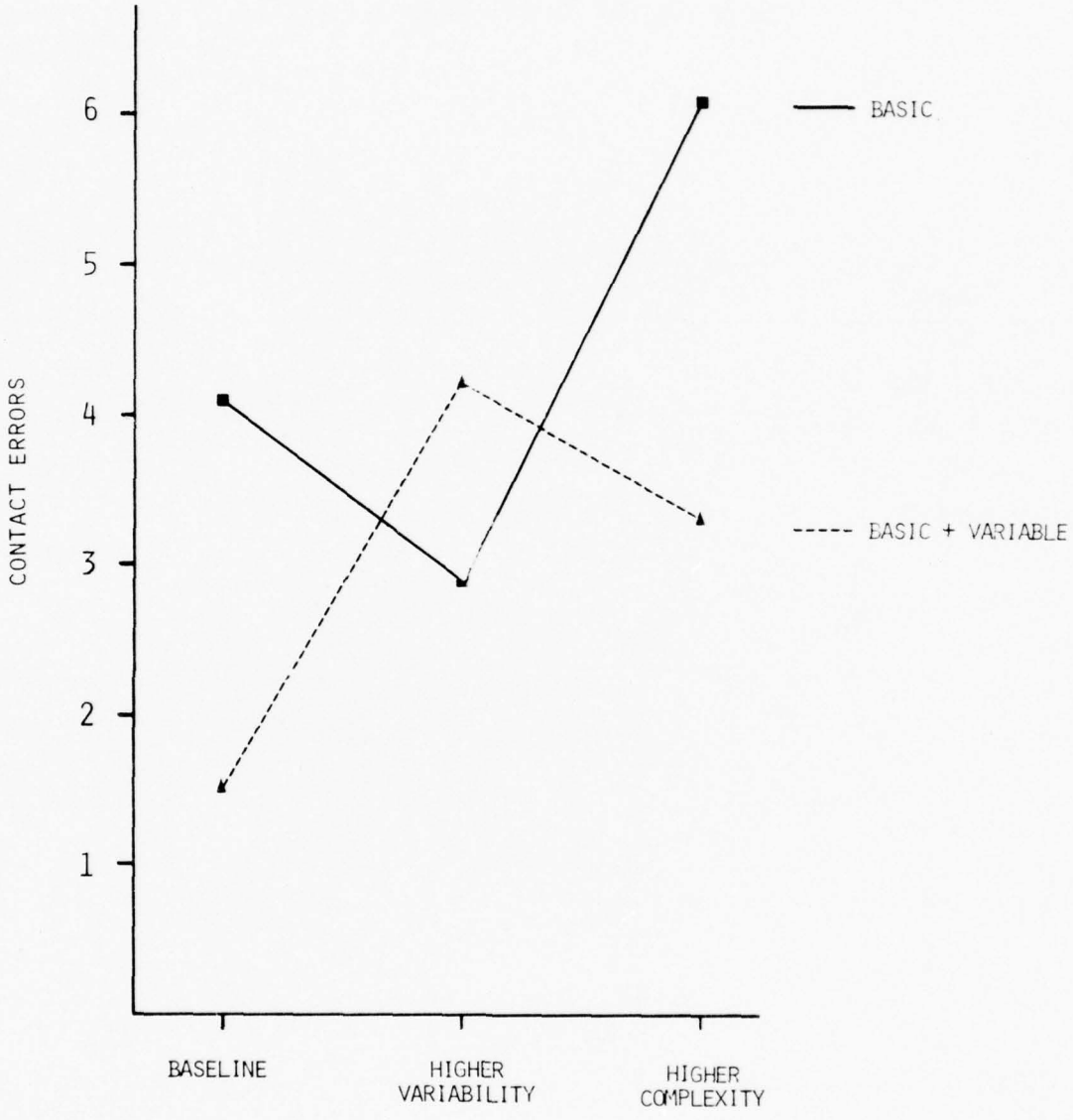


FIGURE 5-3. CONTACT ERRORS ON EXPERIMENTAL TASK AS A FUNCTION OF EXPERIMENTAL TASK, AND CONTROL MODE

TABLE 5-3. ANALYSIS OF VARIANCE SUMMARY TABLE FOR CONTACT ERRORS IN EXPERIMENTAL TASK

Source	df	MS	F
Control Mode (A)	1	15.13	1.46
Experimental Task (B)	2	15.50	1.67
Trial (C)	1	.68	.14
Subjects (S)	5	16.29	
A x B	2	26.00	5.07*
A x C	1	1.13	.32
A x S	5	10.36	
B x C	2	.05	.05
B x S	10	9.27	
C x S	5	4.98	
A x B x C	2	6.50	4.15
A x B x S	10	5.13	
A x C x S	5	3.49	
B x C x S	10	1.16	
A x B x C x S	10	1.57	
TOTAL	71		

*$p<.05$

In a more refined evaluation of the task performance time, the performance of the valve turning positions of the maintenance task were analyzed. This subtask is characterized by its highly repetitive nature. Also, when using the appropriate chain, an operator is able to perform this rather complex operation by using a single symbolic command, thus providing a classic example of a command which is matched to the task name, rather than to the elementary task motions. As shown in Figure 5-4 and Table 5-4 , the performance of the valve-turning subtask closely parallels the overall task performance time. However, in this subtask, the operators were able to define the points and chains during the first trial with essentially no time penalty. This subtask was particularly sensitive to variations in the work space locations. Greatly increased performance time was required in the Higher Variability condition which required the operator to move the manipulator to several different valve locations.

5.3 Operator Interview

After completion of the training and test phases, the participants answered specific and open-ended questions concerning their experience with the two control modes they used. All participants stated that they preferred a combination of analog (RMC) and symbolic (primitive and chained commands) control over analog control alone. Most experienced less fatigue with the chaining capability and claimed it made their task more "interesting". Finally, several operators reported that the procedures required to complete a task were easier to remember when some level of symbolic control was used. This last finding suggests that one function of symbolic control is to "chunk" a task into discrete units which can be more easily verbalized and recalled.

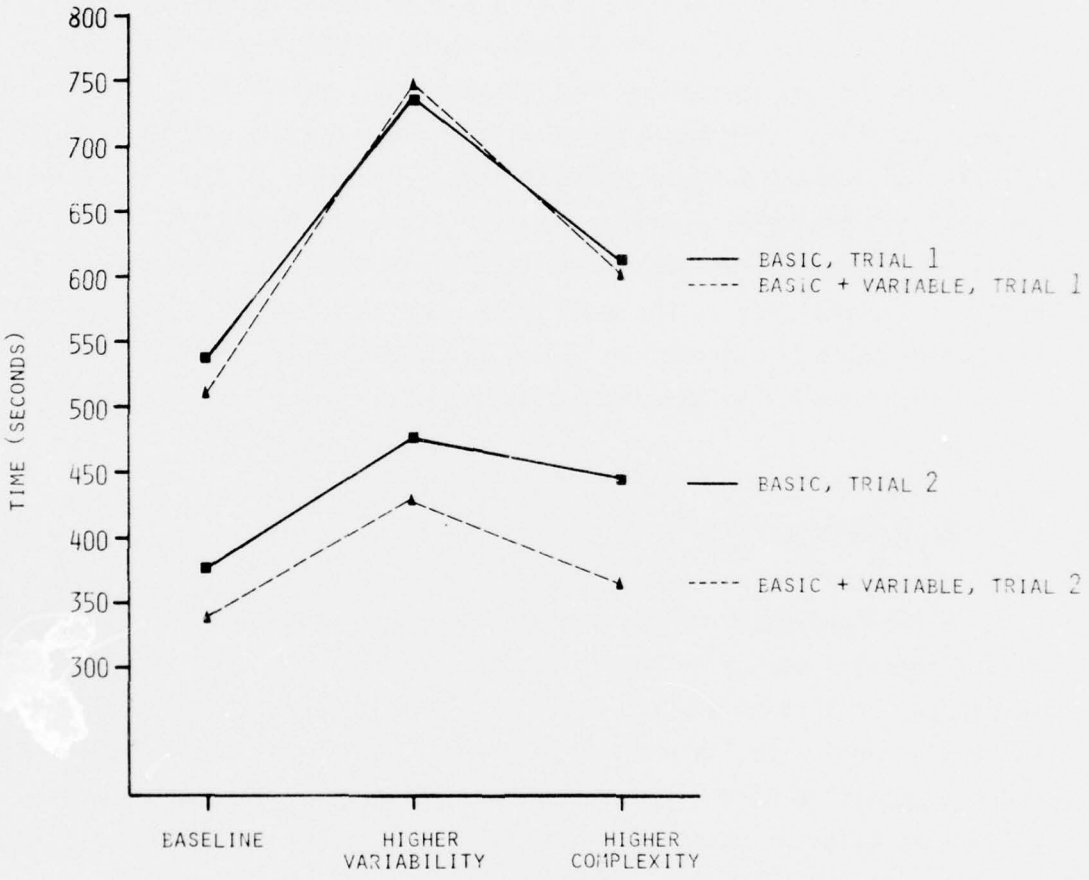


FIGURE 5-4. PERFORMANCE TIME ON VALVE TURN SUBTASK AS A FUNCTION OF EXPERIMENTAL TASK, CONTROL MODE, AND TRIAL.

TABLE 5-4. ANALYSIS OF VARIANCE SUMMARY TABLE FOR PERFORMANCE TIME IN VALVE TURNING

Source	df	MS	F
Control Mode (A)	1	15,138.00	3.45
Experimental Task (B)	2	144,405.50	12.50*
Trial (C)	1	831,190.20	72.96*
Subjects (S)	5	40,090.12	
A x B	2	264.04	.06
A x C	1	10,320.06	12.61**
A x S	5	4,382.00	
B x C	2	23,640.10	4.43**
B x S	10	11,553.71	
C x S	5	11,392.62	
A x B x C	2	847.68	.26
A x B x S	10	4,510.34	
A x C x S	5	818.66	
B x C x S	10	5,336.60	
A x B x C x S	10	3,200.28	
TOTAL	71		

* $p<.01$

** $p<.05$

6. DISCUSSION

The results of the training and experimental phases demonstrate the potential benefits to be gained in manipulator control by having symbolic commands which are user-defined. Specifically, when performing tasks which must be repeated one or more times, a user-defined command (particularly a chain command) will significantly improve performance time, and in most cases will reduce the number of movement errors. Moreover, the training data demonstrate that performance is significantly faster during the initial introduction to the system when the user-defined commands are available. Finally, the training data also suggest that any initial time cost arising from the period required to define the commands will decrease with extended practice.

Based on the theoretical development of the hierarchical task model, upon the related work of other researchers, and upon the results of the experimental results of this year and last (Berson, et al, 1977), a preliminary set of guidelines can be formulated for the design of man-computer-manipulator communication languages.

(1) Use Task- and Concept-Oriented Commands. The human operator thinks in terms of tasks and complete concepts and these are the basic units in his natural language. Organizing the command language around such units provides a natural mental framework for the user, allowing him to think about the task at hand rather than the mechanics of expressing his intentions in manipulator-oriented terms. The principle is related to the idea of *sentence structure* as discussed by Foley and Wallace (1974). "An action language is *sentence-structured* if, within a given phase or subdomain of discourse, each complete user thought can be expressed in a continuous sequence of input device

manipulations with standard patterns of beginning and termination. Upon termination, the machine returns to a state from which similar action sequences, other sentences, can begin". The structure is enhanced if the verb-noun format of natural language is utilized.

(2) Use a Hierarchical Task Organization. The language should have facilities to define a task in terms of its subtasks. The user will be able to express his intentions in a comfortable level of detail. Giving commands in too much detail would cause the user to loose sight of the overall task while dealing with the details. On the other hand, giving a command at a gross level does not allow adaptation of the command to the peculiarities of the situation at hand. The communication should take place at the highest symbolic level comfortable to the user and he should have the facility to choose this level. There are, however, cases where it is necessary for an operator to manually control the manipulator motion, i.e., for fine adjustments, and a manual facility should be incorporated into the hierarchical system.

(3) Use a Constrained, Standardized Language. As shown by Ferrell (1972),an artificial, constrained, and standardized language facilities performance in manipulative tasks better than a free-format, English-like language. The advantage comes from the fact that, although entire manipulation task goals are readily and perhaps most easily described by a person using ordinary English, the complex geometrical and temporal configurations of objects and motions are not readily formulated in such a way. A structured but flexible, task-oriented, artificial command language can enable the

operator to work more efficiently as well as simplify the process of machine translation.

(4) Provide for Tactile, Visual, and Contextual Continuity. Foley and Wallace (1974),emphasize the importance of continuity in the operator sensory and conceptual interaction with the system. Tactile continuity refers to natural grouping and flow of motion required for the tactile input devices such as keyboards, joysticks, etc. Visual continuity refers to the arrangement of information, so that within a given sentence (i.e., one conceptual command), the eye should focus on a single area on the control panel or move in a continuous manner throughout the expression of the sentence. Contextual continuity refers to providing immediately perceivable responses to reinforce the effect of every step in the action sequence and giving standard feedback information in dedicated, fixed positions in the visual field.

(5) Provide Feedback on System State. In dealing with man-machine display interfaces, Engle and Granada (1975) emphasize the importance of providing continuous feedback to the user about the state of the computer system he is dealing with. The information alleviates the frustration generated in the operator when dealing with a complex black box.

(6) Use a Consistent Structure of Language and Feedback. Consistent structure reduces training time, error rate, and user memory requirements and thus is a basic feature of a command language. It is related to the Foley and Wallace (1975) idea of sentence structure.

(7) Provide Mechanisms to Undo Army Errors. Human operators, especially under time pressure are error-prone. A system accepting operator commands must have easy "error recovery capability". Engel and Granada (1975) see this as an important feature of any computer command language.

7. REFERENCES

Bejczy, A.K. Environment-Sensitive Manipulator Control, Proceedings of the IEEE Decision and Control Conference Symposium on Adaptive Processes. Phoenix, Arizona, November 20-22, 1974.

Berson, B.L., Crooks, W.H., Shaket, E., and Weltman, G. Man-Machine Communication in Computer-Aided Remote Manipulation. Perceptronics, Inc. (Woodland Hills, CA) Technical Report PATR-1034-77-3/1, Contract No. N00014-76-C-0603, Office of Naval Research, Code 455, March 1, 1977.

Corliss, W.R. and Johnsen, E.G. Teleoperator Controls. NASA Technical Report SP-5070, Washington, D.C. 1967.

Engel, S.E. and Granada, R.E. Guidelines for Man/Display Interfaces. Poughkeepsie Laboratory, IBM Technical Report TR002720, December 1975.

Ferrell, W.R. Remote Manipulation with Transmission Delay. IEEE Trans. Human Factors in Electronics, 1965, Vol. HFE-6, pp. 24-32.

Ferrell, W.R. and Sheridan, T.B. Supervisory Control of Remote Manipulation. IEEE Spectrum, October 1967, pp. 81-88.

Ferrell, W.R. Command Language for Supervisory Control of Remote Manipulation, in Remotely Manned Systems, (E. Heer, Ed.) California Institute of Technology, 1973.

Fikes, R.E. and Nilsson, N.J. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, 1971, Vol. 2, pp. 189-208.

Finkel, R., Taylor, R., Bolles, R., Paul, R., and Feldman, J. AL, A Programming System for Automation. Stanford Artificial Intelligence Laboratory (Stanford, CA) Memo AIM-243 (STAN-CS-74-456), November 1974.

Foley, J.D. and Wallace, V.L. The Art of Natural Graphic Man-Machine Conversation. Proceedings of the IEEE, April 1974, 62(4).

Freedy, A., Hull, F., Lucaccini, L., and Lyman, J. A Computer-Based Learning System for Manipulator Control. IEEE Transactions on Systems, Man and Cybernetics, Vol. 1(4), 1971.

Goertz, R.C. Mechanical Master-Slave Manipulator. Nucleonics. November 1954, 12:45-46.

Goto, T. Compact Packaging by Robot with Tactile Sensors, Proceedings Second International Symposium on Industrial Robots, pp. 149-159, Chicago, Ill., May 1972.

Groome, R.C. Force Feedback Steering of a Teleoperator System. M.S. Thesis, Massachusetts Institute of Technology, August 1972.

Lozamo Perez, T. and Winston, P.H. LAMA: A Language for Automatic Mechanical Assembly. Fifth International Joint Conference on Artificial Intelligence, Boston, Massachusetts, pp. 710-716, August 1977.

Mullen, D.P. An Evaluation of Resolved Motion Rate Control for Remote Manipulators. Charles Stark Draper Laboratory, Massachusetts Institute of Technology, Cambridge, Mass, 1973.

Nevin, J.L. and Whitney, D.C. Computer Controlled Assembly. Scientific American, January 1978.

Nevins, J.L., Sheridan, T.B., Whitney, D.C. and Woodin, A.E. The Multi-Moded Remote Manipulator System. Proceedings First National Conference on Remotely Manned Systems, ed. by Ewald Heer, California Institute of Technology, 1973.

Pesch, A.J., Hill, R.G., and Klepser, W.F. Capabilities of Operators as Divers vs. Submersible Manipulator Controllers in Undersea Tasks. General Dynamics, Electric Boat Division, Report No. U417-70-043, 1970.

Pesh, A.J., Hill, R.G., and Allen, F.L. At-Sea Operator Performance of Small Performance of Small Sumbersible Manipulators. General Dynamics Electric Boat Division Technical Report No. U-413-71-031, 1971.

Rechnitzer, A.B. and Sutter, W. Naval Applications of Remote Manipulation. Proceedings of the First National Conference of Remotely Manned Systems for Exploration and Operation in Space. Heer, E., Ed., California Institute of Technology, Pasadena, California, 1973.

Sacerdoti, E.D. Planning in a Hierarchy of Abstraction Spaces. Third International Joint Conference on AI, 1973, pp. 412-422.

Sacerdoti, E.D. A Structure for Plans and Behavior. Stanford Research Institute Artificial Intelligence Center, Technical Note 109, 1975.

Verplank, W.L. Symbolic and Analogic Command Hardware for Computer-Aided Manipulation. M.S. Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1966.

Whitney, D.E. Resolved Rate Control of Manipulators and Human Prostheses. IEEE Transactions on Man-Machine Systems, June 1969.

Woodin, A., Whitney, D.E., and Nevins, J.L. Annual Progress Report No. 2 for the Development of Multi-Moded Remote Manipulator Systems. Massachusetts Institute of Technology, Cambridge, Massachusetts, 1973.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

APPENDIX A

SOFTWARE DOCUMENTATION

Yoram Alperovitch

1. INTRODUCTION

1.1 Old System Functional Overview

The ONR-1976 Computer-Aided Manipulator Facility includes a hydraulic servo manipulator, 3-dimensional display, man-machine interface and a minicomputer.

The operator controlled the manipulator through joysti and pushbuttons on the control-console and observed the manipulator's activities through direct viewing or via 3-D display.

The operator's input was processed by the minicomputer which, in turn, controlled the manipulator's servo electronics.

Control Modes - three control modes were available

(1) Direct Control - which enabled direct manual control of each joint.

(2) Resolved Motion Control - RMC enabled manipulation of the arm in a cartesian space.

(3) Autmoatic Motion Control - AMC gives complete control to the computer. It enabled the operator to record seven different configurations (points) and move the arm to any previously recorded point under complete computer control.

Joystick - Joysticks were inoperable under AMC mode. Speed adjustment allowed the operator to select 4 different rates of arm motion.

The 3-dimensional display enabled the operator to generate a "stick-figure" image of the arm.

1.2 New System Functional Overview

A whole new array of functional requirements were introduced in the ONR-1977 R&D activity plan. An additional degree of freedom to enable wrist flexion, a set of new primitive and non-primitive commands, and a continuous, control-feedback visual system were just a part of the new system requirements. Consequently, a new hardware/software system was designed and developed.

Hardware changes included reconstruction of the manipulator, development of a new dedicated control keyboard and a new CRT for visual control feedback. The new keyboard and CRT replaced the old integrated control-console.

Required software modifications and additions were so drastic that an overall redesign and development were necessary.

Modularity and an hierarchical software structure were the guide lines in the process of new system design. Current and future requirements were analyzed as to enable design of an easy modifiable and expandable software system.

2. NEW SYSTEM FUNCTIONAL DESCRIPTION

2.1 Command Language Design

2.1.1 Overview. The manipulator command language was designed, based on the procedural net model analysis of command language requirements and the principles identified by the analysis and by review of related work. This language consists of a set of primitive commands and "chains" or sequences of commands.

The primitive commands are the conceptual unit tasks useful in manipulations which were automated and can be called by one user command. Primitive calling commands have the syntex of *verb-noun-paramater-terminator* but not all the syntactic elements are present in all the commands as we will see. Chains are user defined sequences of primitives and/or chains which provide the facility to define new commands higher in the hierarchy of tasks. Figure 2-1 shows the complete keyboard arrangement used in implementing the language. The different command word types such as verbs and nouns are grouped together by colors indicated by different shading in the figure. The flow of hand movement using the verb-noun-parameter-terminator is consistent from left to right.

2.1.2 Points. Named points correspond to points in link space, (i.e., specific configurations of the manipulator). The points can be defined by bringing the manipulator to the required position. They can be executed and then erased or renamed. The snytax of these commands is:

DEFINE	POINT	7	DO
GOTO	POINT	7	DO
DELETE	POINT	7	DO

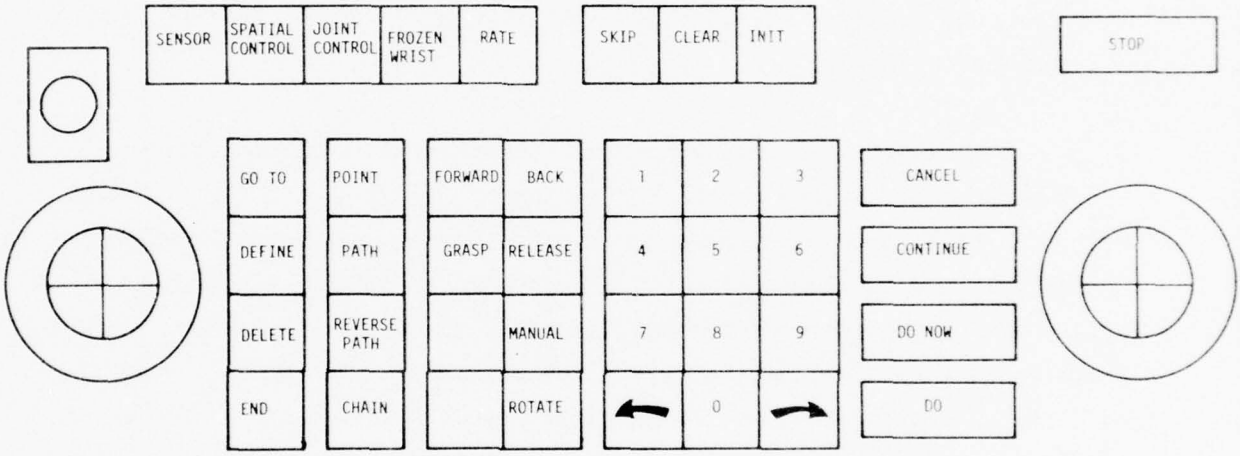


FIGURE 2-1. DEDICATED KEYBOARD ARRANGEMENT

The verb-noun structure is evident where each square is a single keystroke on the dedication keyboard. "DO" is the standard terminator of each command. If the user notices an error while entering a command he can abort the current command by pressing CANCEL instead of DO. We used numbers as point names instead of alphanumeric names and only single-digit names are allowed. A point name can be assigned to a new manipulator position by redefining it.

2.1.3 Paths. Complete paths in link space can be defined, traversed forward or backward and deleted, using a similar set of commands. The syntax of these commands are as follows:

DEFINE	PATH	3	DO
END	PATH		DO
GOTO	PATH	3	DO
GOTO	REVERSE PATH	3	DO
DELETE	PATH	3	DO

A path is defined as a sequence of points from the initiation command to the END command. All motions of the manipulator are stored, both those controlled manually or those controlled by the computer while executing a previously defined chain. The provision of running both forward or backward on a path is useful to move from one area of the workspace to another and back.

2.1.4 Other Primitives. Other primitives are provided to do ROTATion of the wrist, GRASP and RELEASE and short movement in the direction of the gripper; FORWARD and BACKWARD.

2.1.5 Chains. Chains are named sequences of specific primitive commands which were defined and labeled as a unit. For example, the following is a complete chain which closes a value located at the end of path 4.

```
CHAIN 9
     GO TO PATH 4
     MANUAL
     GRASP
     GO TO CHAIN 2
     GO TO CHAIN 2
     RELEASE
     GO TO REVERSE PATH 4
```

"CHAIN 9" is the name of the chain. PATH 4 is traversed forward at the beginning and backward at the end of CHAIN 9. "MANUAL" is a special command by which the computer temporarily gives control back to the user so he can fine-adjust the manipulator gripper in front of the valve to be closed. After the GRASP, CHAIN 2 is called twice to perform the rotation of the valve. This is an example of using a chian within a chain. When such recursion is allowed by any level, a hierarchy of chains, (i.e., task units) can be defined by the user. RELEASE opens the gripper and the last command brings the manipulator to the initial position. A PATH is used in this example to bring the manipulator to and from the valve, rather than a GO TO POINT, making the manipulator go around obstacles which it may collide with if moved by a GO TO POINT.

Chains are defined by going into a special EDIT mode in which no execution of commands takes place. The chain name is specified and then the sequence of primitives that make up the chain is given. The user then goes out of edit mode into the regular execution mode and he can call up the chain for execution by giving a command of the following type:

```
GOTO      CHAIN      9      DO
```

2.2 Controller Interface

The operator controls the computer-aided manipulator through a specially-designed keyboard and joysticks. A dedicated keyboard, shown in Figure 2-1, allows the operator to select any of the command language primitives and to change from manual operation (via the joysticks) to automatic operation (via the keyboard) and vice versa. Feedback information generated by the computer is displayed on a CRT screen.

The system was designed to provide as much feedback information as possible. The keys provide tactile and visual feedback through a soft "click" and a monentary lights that are activated when each key is depressed. An additional visual feedback is given on the screen when the completely spelled-out command is shown immediately for user verification. Extensive additional information about the system state and the processes going on is shown on the display using a consistent format.

2.3 Software Requirements

2.3.1 General. The requirements for the revised software include:

(1) Additional Primitive Command - like "GOTO PATH," "MOVE FORWARD," "ROTATE LEFT," "GRASP," etc.

(2) Chain Execution - sequential execution of primitives in chain.

(3) Commands Queuing - which should be performed asynchronously with any other ongoing process.

(4) Continuous Visual Feedback - should reflect the actual state of the system.

(5) Chain Definition Mode - which should halt any execution activity and enable operator to define new sequence of commands.

(6) Additional Link - seventh degree of freedom in the manipulator which will enable wrist rotation.

The new software was designed and developed under strict adherence to principles of modularity and hierarchical design. Detailed monitoring of present and future requirements enabled the design of a flexible and expandable modular system. Utilization of the look-ahead approach into the present system design makes many future requirements and modification suitable for easy and rapid implementation.

2.3.2 Primitive Commands. Ten different primitive tasks can be executed as a response to the operator's command. These include:

(1) "GOTO POINT" is the only primitive which was used on the old system.

(2) "GOTO PATH" - manipulator traverses along a predefined and stored trajectory.

(3) "GOTO REVERSE PATH" - traverse a path from end to beginning.

(4) "FORWARD" moves manipulator small distance forward along the axis of the gripper in constant orientation.

(5) "BACKWARD" - moves manipulator backward in constant orientation.

(6) "GRASP" - closes gripper.

(7) "RELEASE" - opens gripper.

(8) "ROTATE LEFT" - rotates the gripper counterclockwise.

(9) "ROTATE RIGHT" - rotates the gripper clockwise.

(10) "MANUAL" - stops execution and waits for operator's response.

These primitives can be executed in a sequence such that a new command is invoked after a previous command is finished. They can also be keyed in while execution of previous primitives is going on. The new primitives are pushed into a pending-commands queue and as execution of the current command ends, the next command in the queue is executed. This process is repeated until the queue is empty.

2.3.3 Chained Command. *"GOTO CHAIN X" command is a higher level command* whose operand is a sequence of primitives described in the previous section. A chain of primitive commands can be defined in a special editing mode and later used as an operand. Each chain can have up to ten primitives. Chains can be queued like any other primitive.

2.3.4 Definition Commands. Some of the primitives can not be executed unless their operands have been defined. The proper definition commands are:

(1) DEFINE POINT X - records present position of the arm.

(2) DEFINE PATH X - records an entire trajectory traversed by manipulator until the "END PATH" command is given.

(3) DEFINE CHAIN X - records all the subsequent primitives into chain X. "END CHAIN" command stops chain definition.

The operator can define ten different points, five paths and five chains.

2.3.5 System States. The operator can utilize various control options included in the software system. These options are invoked by State Command which include the following:

(1) Spatial Control - enables/disables the cartesian control mode of the arm.

(2) Joint Control - enables/disables the joint-by-joint control mode of the arm.

(3) Wrist Invar - enables/disables wrist invariance function. The initial orientation of the wrist is maintained while the arm is maneuvered by primitives, joysticks, or both.

(4) Rate X - changes the rate of motion of the arm. Three different rates are available.

The states described above can be viewed functionally as higher level functions which are prevailing in the system while some lower level functions (primitives, non-primitives) are started, executed, and finished. For example, joystick manipulation in spatial or joint state can be performed while some primitive execution is going on. The invariance can be invoked too, and then the orientation of the gripper remains constant.

2.3.6 System Visual Feedback. Visual console feedback serves three different functions.

(a) Continuous display of operational control information.

(b) Editing mode feedback which displays chains definition and modification.

(c) Instant feedback to keys pressed by the operator on the dedicated keyboard.

(1) System Control Feedback

(a) Sensor, Wrist Invariance, 3-D Path-Definition are indicated by on and off. Joysticks control is indicated by spatial or joint control and Rate shows the current rate of motion.

(b) The Current command in execution is shown on top of a queue which is popped up every time a new command enters execution. If a chain is an appending command, it is shown like another primitive. When it enters execution, a special header indicates "CHAIN IN EXECUTION" and chains primitives are passed so that the operator will see what comes next into execution. A maximum of 10 commands can be shown at a time.

(c) When execution stops a STOP is displayed at the center of the console. The same message is generated when the operator stops the system.

When MANUAL command is executed, a "YOUR TURN" message is displayed

(2) Editing Mode Feedback

The Edit Mode is involved by the Define Chain X Command. If the chain exists already, chain primitives are displayed. The operator can modify or generate a new chain by typing valid primitive commands which are displayed under the chains name.

(3) Keyboard Feedback

Whenever a key is pressed, its name is displayed on the console. This enables the operator to verify that he generates a valid sequence and to correct errors. Whenever a typing error occurs, an "Invalid Command" message is displayed.

2.3.7 System Special Features. To enable smooth operation of the system three options were defined as follows

(1) Skip - skips on the current command in execution. It enables the operator to stop any primitive in the middle of its execution and proceed with next command in queue.

(2) Clear - skips on the current command in execution and clears the stack and the queue of all the commands.

(3) Initialize - clears all commands and deletes all the defined points, chains, and paths. It also erases the screen and displays current system states.

3. HARDWARE

3.1 Hardware Configuration

Hardware components of the system are shown in Figure 3-1. A detailed explanation of each one of them is given in the following sections.

3.1.1 Control Processor. The supporting processor for the manipulator system is an Interdata Model 70 minicomputer. This machine, which is microprogrammed to accept an imitation of the IBM system 360 assembly language set, has a memory cycle time of 1 μs and basic instruction execution times averaging between 1 and 3 μs. As presently configured, the processor system includes 48 kilo bytes of core memory, a high speed paper tape reader/punch, a line printer, a CRT alphanumeric terminal, a selector channel, a disk memory and re-settable precision interval clock. The disk drive, CRT, and other preipherals are used to support program development work and are not part of the real time manipulator control system.

3.1.2 +I/O Interface. Data transfer between the computer and manipulator servo-electronics and between the computer and the control console is performed by a Perceptronics +I/O Programmable Interface. Besides providing all analog-to-digital (A/D), digital-to-analog (D/A), and digital-to-digital (D/D) conversions among the system components, this interface allows the outputs of all controlled devices to be treated by the processor as if they were the product of only one device, thus simplifying the software arrangements at the processor.

The +I/O Interface contains a number of functional modules arranged along a transfer buss by which commands, data and status signals are communicated. These modules perform such individual functions as standardizing communication with the processor, sequencing data transfers across the buss, and performing D/A conversion and output.

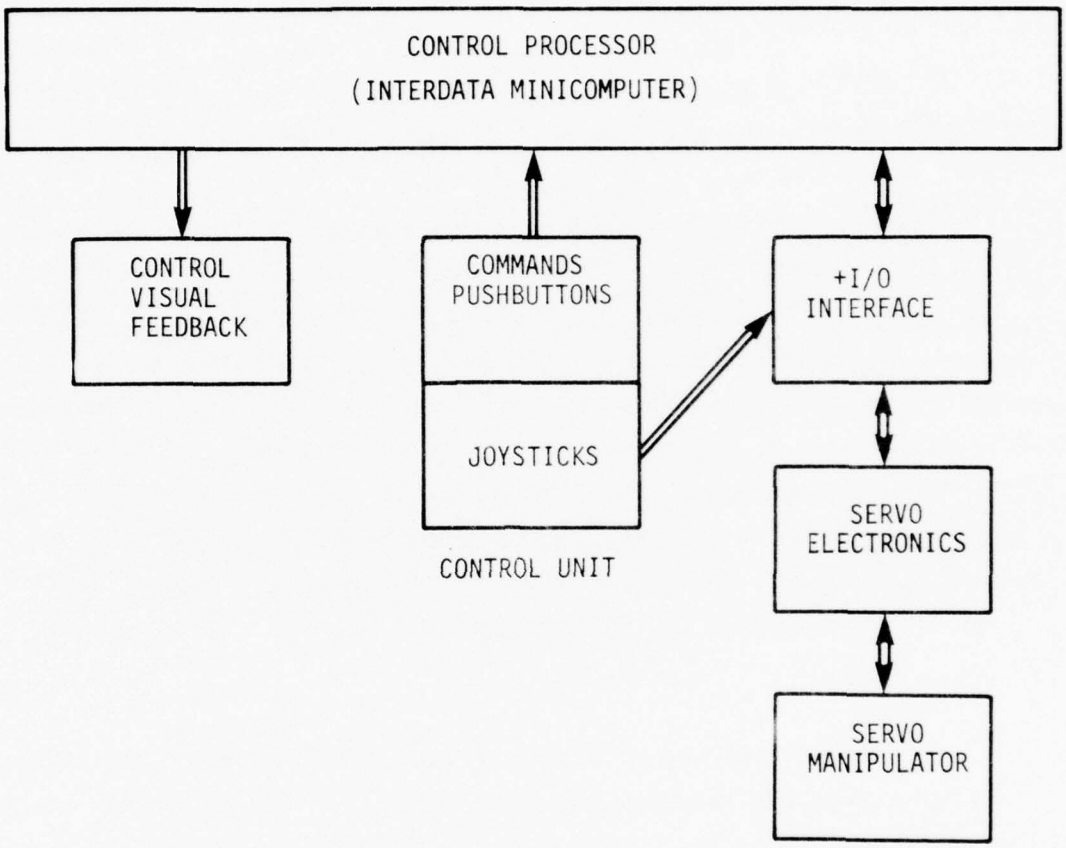


FIGURE 3-1. HARDWARE CONFIGURATION

In addition to an interface module between the processor and interface, the +I/O includes an A/D module that is used to interface with the servo position potentiometers and control console joysticks. Thirty-two individually addressable input channels are provided. An eight-channel D/A module is used for converting and sending position commands (voltages) to the control inputs of the servos. Finally, a D/D module is used to provide the 16 input and output channels for the button and lamp arrays of the control console.

3.1.3 Servo Electronics. The electronic position-feedback servo amplifier generates a control voltage to the valves according to the difference between an input command voltage and the output voltage of the position sensing potentiometers attached to the actuators; thus, the motion of the servo manipulator is controlled by the D.C. input voltage to each actuator servo amplifier. The command voltages to the servo amplifiers are output from the computer via digital to analog converters.

3.1.4 Servo Manipulator. The manipulator shown in Figure 3-2 is electronically controlled and hydraulically powered. It has six rotating joints and the seventh is gripper closure. The arm motions and joint number are:

(1) Shoulder Rotation
(2) Shoulder Elevation
(3) Elbow Flexion
(4) Forearm Rotation
(5) Wrist Flexion
(6) Gripper Rotation
(7) Gripper Closure

The links are moved by activators. Each activator has a pair of pistons and cylinders controlled by a single servo value. The servo values are controlled by electronic, position-feedback servo amplifiers.

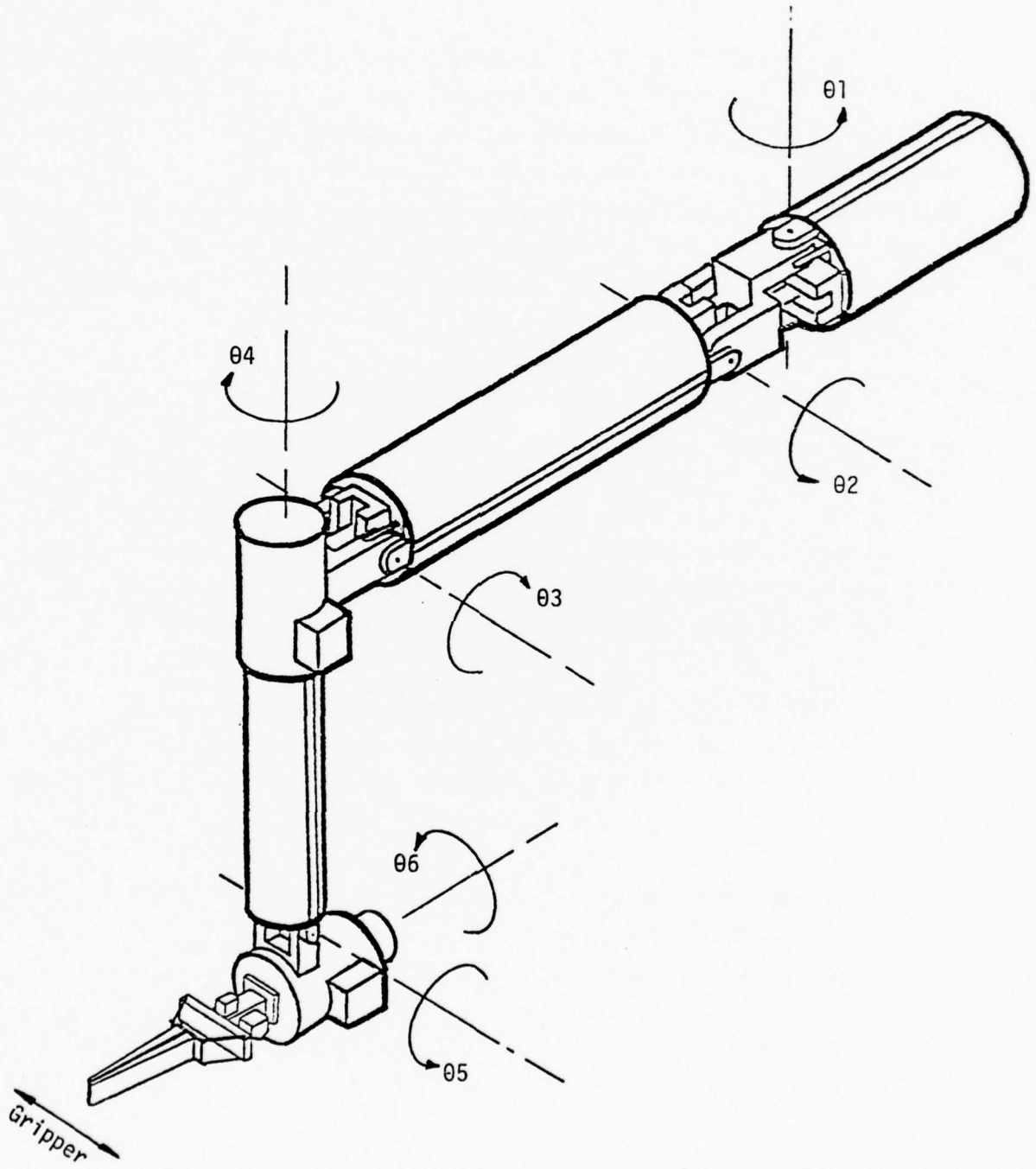


FIGURE 3-2. SERVO MANIPULATOR WITH MOTIONS OF THE SIX ROTARY JOINTS

3.1.5 Control Unit. It has a panel of pushdown buttons and two joysticks. Operator's control commands are generated by pushing valid sequences of buttons. Every button generates a different ASCII code. The codes are transmitted to the control processor, which performs the required functions. The joysticks are used to manually control the arm. Each joystick has three degrees of freedom which corresponds to certain joints in the arm. The seventh link, the gripper, is controlled by an additional switch on the keyboard. The analog signals generated by the joysticks are transmitted to the A/D converter in the +I/O interface which then sends digital values to the control processor.

3.1.6 Control Visual Feedback Unit - CRT. The console displays characters sent to it by the control processor. Two types of information are displayed

(1) immediate visual feedback to keys pushed by the operator.
(2) control information - which is functional feedback of the system.

3.1.7 3-Dimensional Display. This device produces a three-dimensional image. Designed specifically for use as a computer output terminal, the display permits direct user interaction with the 3-D image.

During operation with the manipulator, the computer calculates the image coordinates of the arm and generates a "stick figure" image of the arm on the 3-D display. As the arm moves within the work space, the displayed arm figure moves in a corresponding manner within the 3-D image volume. With a force sensor attached to the manipulator wrist, an image of objects in the work space can be created by recording and displaying those points in space where the end-effector touched the object.

3.2 System Data Flow

3.2.1 Manipulator. A continuous flow of information to and from the manipulator is performed. Every 4 ms. "read manipulators current position" and "write to manipulator required position" are performed. If the difference between two values of any of the seven links is greater than the "hardware sensitivity threshold" the servos are activated and links are moved toward the required position.

The "read" activity transmits seven analog values measured by the potentiometers of each link to the A/D converter in the +I/O which then sends digital values to the control processor. The "write" activity sends digital values from control processor to the D/A converter in the +I/O. Analog values are sent then to the servo "control electronics" which generates required activation signals to the manipulator servos. The read and write activities are invoked by clock interrupts generated every 4 ms. When control is given to such an interrupt it calls the "read" and "write" routines which takes about 1 ms and then control is returned to the interrupted program.

3.2.2 Control Unit (Keyboard and Joysticks). There are two different routes of data at the control unit. Every pushbutton touch generates a special keyboard-interrupt which diverts control to programs which read and analyze the code generated at the keyboard. The rate of keyboard-interrupts is a function of the key-pressing rate. The joysticks generate analog signals which are sent to the control processor through the +I/O. The +I/O converts the analog signals and the digital data is "read" into the control processor every 4 ms together with the manipulators' seven links data.

3.2.3 CRT - Visual Feedback Control. Information displayed on the CRT is generated by the Main Control Process and the Teletype (read-keyboard) process. The first generates the current state of system information

and updates the CRT with every functional event in the system. The latter generates visual pushbutton feedback and error messages in case of lexical or syntactical errors.

Same clock interrupts which are invoking the +I/O "read" and "write" routines invoke the CRT feedback routine which sends characters from output buffers to the CRT. One single character is sent every time this routine is invoked and so 250 characters can be displayed on the CRT every second.

4. SOFTWARE

4.1 Software Modular Structure

The software system combines three main modules shown in Figure 4-1. Module 1 is the main control process. It contains the main central loop which executes most of the software functions. All the programs of primitive and non-primitive functions, joystick control and state command programs are included in this module. Module 2 is the teletype process module. It contains the programs which read the keyboard codes analyzes and interprets them Module 3 is the I/O process module. It contains programs which "read" and "write" and send characters to the CRT.

4.2 Software Hierarchical Structure

In Figures 4-2, 4-3, and 4-4 programs organization of each module is shown. The diagrams depict the hierarchical structure organization and not the functional flow of control. Some of the commonly shared routines are not shown but will appear later in the flowchart diagrams.

4.3 Real Time System Processes.

One of the basic system requirements was to generate and produce smooth motions of the arm which can be corrected and modified while performing automatic tasks. Another requirement was to assure an immeidate reflex response to any obstacle the arm may bump into. A continuous control of the system through the dedicated keyboard must be available to allow immediate intervention by the operator. These requirements dictated a real-time system design which consists of three parallel processes.

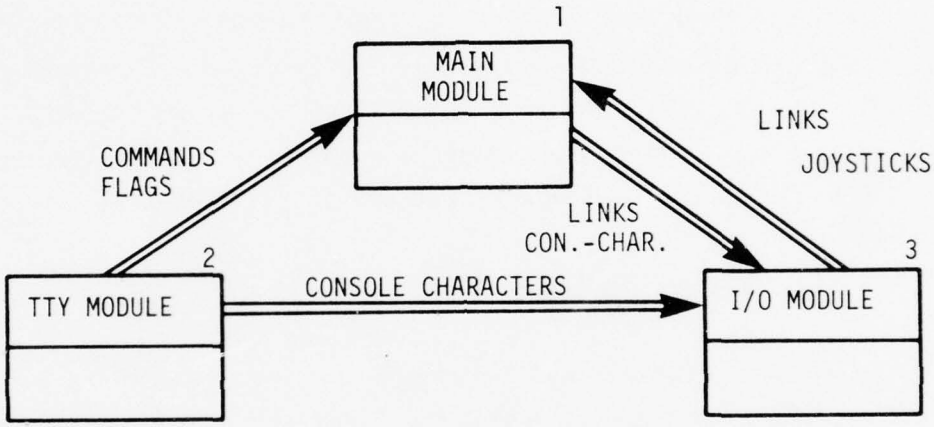


FIGURE 4-1. DATA CONNECTION DIAGRAM

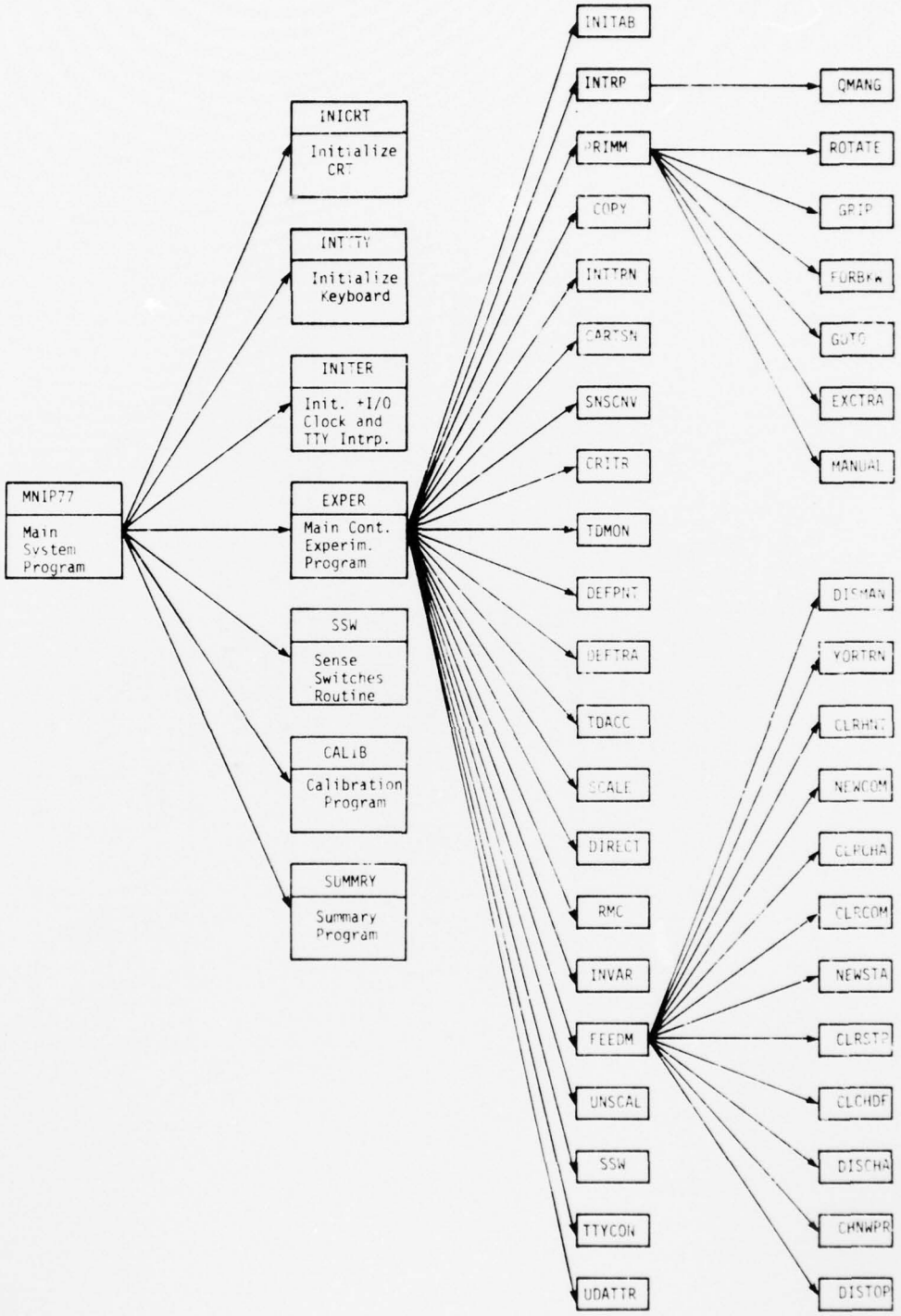


FIGURE 4-2. MAIN MODULE HIERARCHICAL TIER STRUCTURE

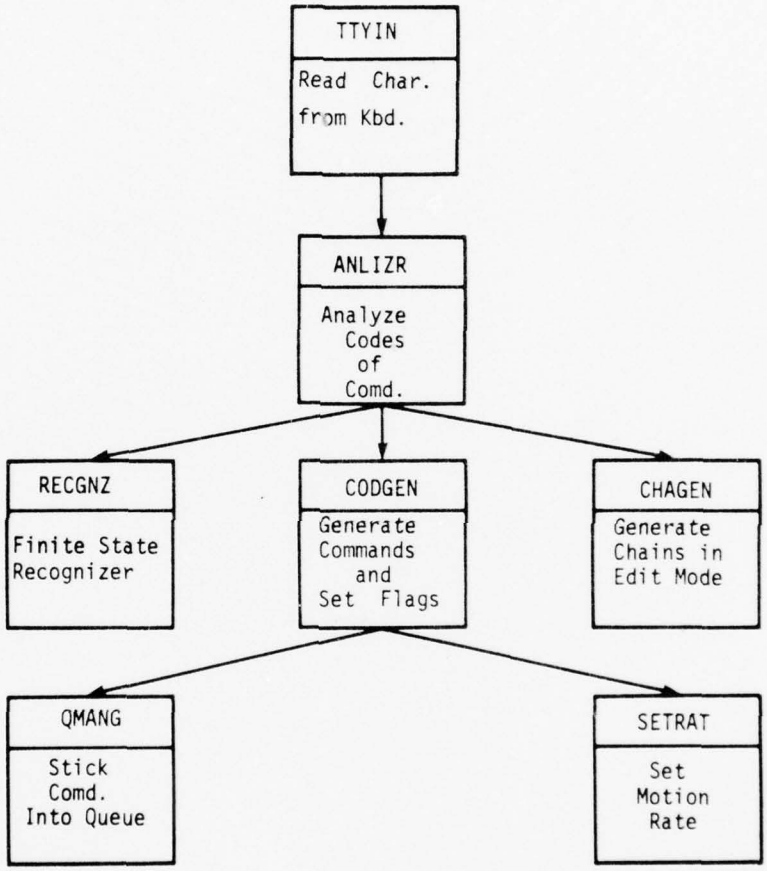


FIGURE 4-3. TTY MODULE STRUCTURE

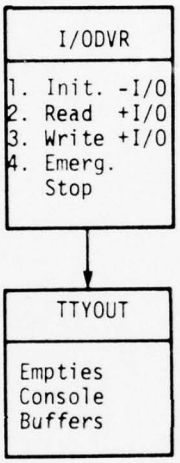


FIGURE 4-4. I/O MODULE STRUCTURE

(1) Main Control Process

This process performs most of the system functions. It executes primitives and chains taken out of the queue. If joysticks are operated, it adjusts the arm position accordingly. If invariance is enables, it keeps the wrist orientation constant. Since spatial control and wrist invariance require complicated computations the main control process calls the appropriate routines to perform this job before sending required data to the manipulator.

(2) Teletype Process

This process is invoked whenever a key is pressed. It records the code generated by the key and when a terminal key is pressed it sends the command (sequence of codes) to the interpreter. The interpreter recognizes commands and interprets them into codes of action. When a primitive command is received, it is transformed into a code of two digits and the code is pushed into the queue. State commands are transformed into specific flags which indicate to the main process what control function to perform. The modular structure of the interpreting programs is flexible to allow additional changes and extensions of the system user language.

(3) I/O Process

Continuous data flow is required between the computer and the manipulator to enable continuous and accurate control of the manipulator position. The I/O process reads current values of the seven links and writes new required values. This activity is repeated about 100 times per second which ensures smooth control of the arm.

4.4 Real Time System Constraints

In the real time environment, operations of the three processes interleave in time so they are concurrent. The processes interact among themselves by using same data structures. In our system these are buffers which are filled by one process and emptied by another. The queue is filled by the TTY process and emptied by the main control process and flags are set and reset in the same way. The shared resources must not be accessed simultaneously by two processes and some provisions must be made to assure mutual exclusion. The access-restricted code sections are called critical sections (or protected sequence). Whenever critical section execution starts, it must not be interrupted until it is finished. The execution of critical sections must be synchronized by a mechanism which will enable to complete execution of one process at a time.

In Figure 4-5 the three concurrent processes are shown. Each process corresponds to the module shown on Figure 4-1.

Average execution of the main control loop takes about 60 ms. and the loop is executed every 100 ms. This specification was devised to satisfy the constraint of smooth arm motion described in Section 4.3.

The clock interrupts are generated every 4 ms and CPU control is transferred to TRAP1 until execution of the I/O module is finished and control is returned to the main module. Execution of an I/O interrupt takes about 1 ms.

The I/O module is regarded as a critical section so that any other interrupts are disabled.

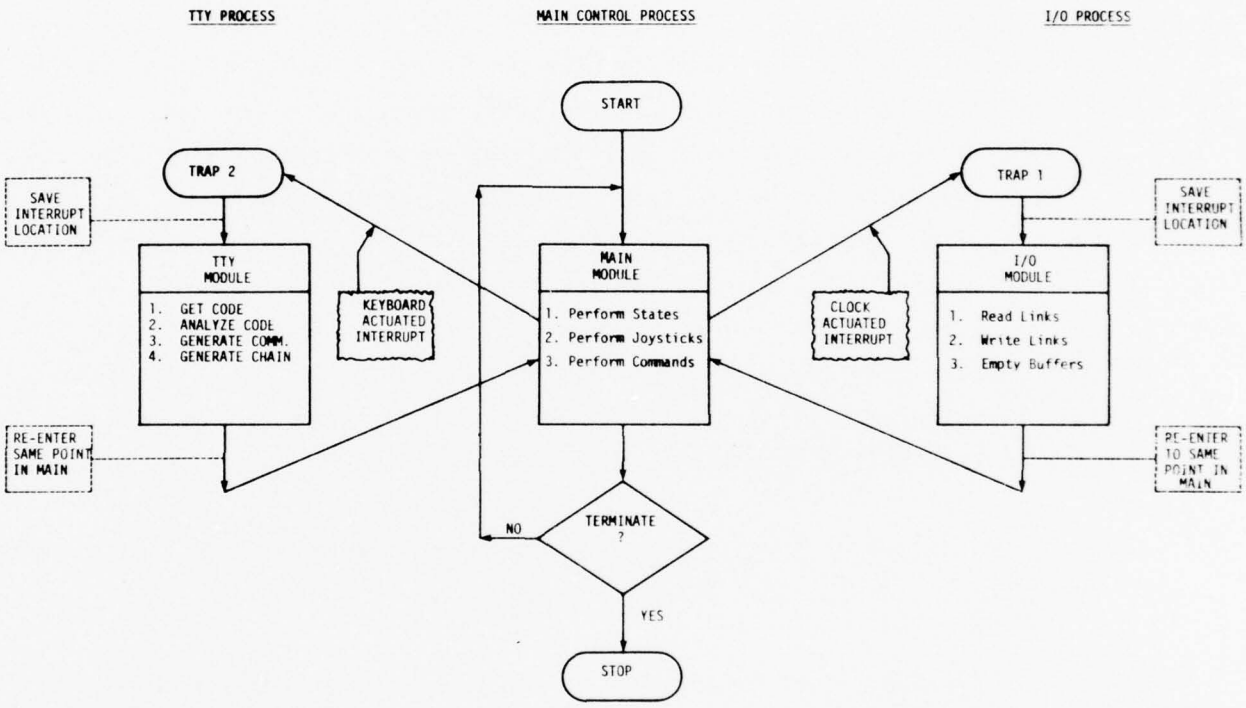


FIGURE 4-5. REAL-TIME PROCESSES

The TTY module is invoked by the keyboard generated interrupts. Control is transferred to TRAP2 and the TTY module is executed while any other interrupts are disabled. An average execution time takes about 1 ms. In order to respond to coming interrupts with minimal delay the main process must not disable CPU interrupts for long periods of time. This requires an even distribution of critical sections along the main process control flow so that an I/O or TTY interrupt will not get lost.

We can visualize the concurrent processes in the system in the following way. There is a basic cycle every 100 ms. For about 60 ms. the CPU works on main control loop. Clock interrupts are coming every 4 ms. so that control is diverted to the I/O process. One TTY interrupt can occur during the cycle and its time consumption is negligable. Overall in about 85-90 ms. of 100 ms. cycle the CPU is really busy.

4.5 Software Functional Structure

This section will include the flowcharts, and functional explanations of the system programs. Any additional detailed documentation can be found in program source listings.

4.6 Documentation Methodology

In our further hierarchical documentation we'll use the following documentation methodology (reference 1).

"The ANSI technique used for denoting hierarchic flowchart expansion is striping the box to be expanded, as shown in Figure 4-6. The striped module is given a procedural name, NAME, a cross-reference identifier, x, and a number, n, on its current flowchart. I shall augment that method as follows. If the current flowchart identifier is m, then the box can be uniquely identified as the Dewey-decimal number m.n, and this number can be used for cross-referencing as long as no ambiguity arises. In such a case x need not appear at the point of striping.

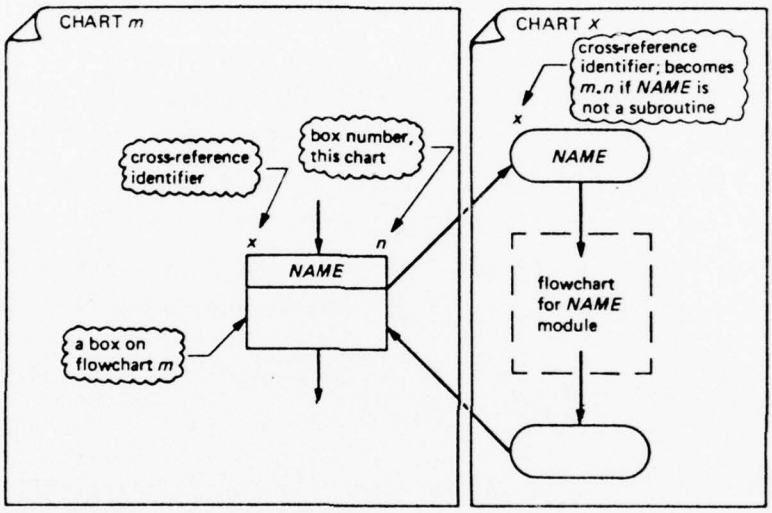


FIGURE 4-6. HIERARCHIC EXPANSION OF STRIPED FLOWCHART SYMBOL

Striped symbols can refer to hierarchic expansion in one of three ways: (a) subprograms, which can either be segments of in-line code or procedures that, on normal termination, continue execution always at the same point in the program; (b) internal subroutines, which are segments of code invoked at several places in the program, which always return, upon completion, to the point of call, and which are part of the body of the program; and (c) external subroutines, which are subroutines (returning to the point of call) whose designs are external to the program (e.g., library subroutines) and not described in this set of documentation. I have previously referred such program segments as striped or named modules. Notations for these three cases are illustrated in Figure 4-7.

The hierarchic place that a module occupies in a design is denoted by its Dewey-decimal cross-reference. For example, suppose that on a flowchart numbered m, a box numbered n refers to a procedure (not subroutine) to be expanded later in the design process. Then the flowchart for that later expansion is made Chart No. m.n. One reading the flowchart wishing to trace out how the function in box n of flowchart m is achieved, merely has to locate Chart No. m.n to proceed.

More specifically, suppose a stripped module appears on Chart 1.2.6, and has the number 5. Then one can state that box number 2 on Chart 1 was expanded as Chart 1.2; on that chart, box 6 was expanded as Chart 1.2.6; and module number 5 may appear expanded later as Chart 1.2.6.5.

The reference to a flowchart, however, cannot always be cross-referenced this way because subroutines, which can be called from many places, would not then possess a unique chart number. Therefore, each subroutine is assigned its own unique level-one chart number. One convenient way of distinguishing procedures from subroutines is by assigning an alphanumeric chart number for subroutines; for example, S6 refers to

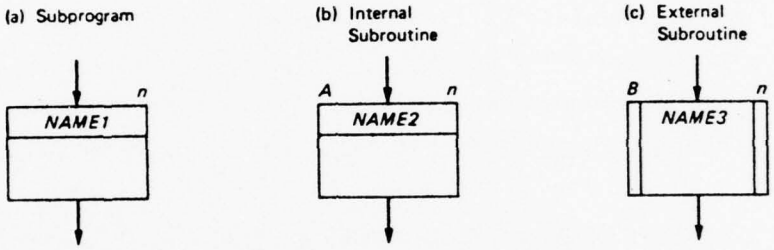


FIGURE 4-7. STRIPED SYMBOLS: n is the module number on this chart; A is a numeric or alphanumeric chart number where the hierarchic expansion of that subroutine begins; and B is a designation that indicates where interface information can be found.

Subroutine 6, T4 to Trap routine 4, etc. The choice of an alphanumeric designator can be used to group subroutines with common properties together in documentation. Expansions within subroutine flowcharts follow the normal numbering: for example, S6.4.2 refers to the box numbered 2 on Chart S6.4."

4.7 Main Control Process Functional Structure

MNIP77 -

It is the highest level program of the system - see chart 1. It invokes a sequence of routines which are initializing the system. It then senses (see SSW routine) the switches of the computer's panel and if all of them are reset then system calibrating program is invoked followed by termination. If at least one of the switches is set the EXPER program is invoked and an experimental session can be conducted. When the experiment is concluded a summary program is invoked and the data which was monitored during the session is processed and printed as a report. The CALIB and SUMMARY programs are not written yet and are resolved as dummy routines.

EXPER -

This is the central program of the system - see chart 2 - by which most of other programs are invoked. Basically the program is a one "DO WHILE" loop executed every 100 ms. The while condition is the same as in MNIP77. Whenever all the switches are reset program terminates, otherwise it loops on.

The main cycle is a long sequence of programs which are invoked in certain contingencies. Generally each program has a flag which indicates whether it should be invoked or not. A short description of the program, according to their sequence in chart 2, is given below.

INITAB - Initializes screen and core where points chains and paths are stored.

TDACC - Three dimensional display activation routine. It should be only invoked when the 3-D is hooked on. The program is not written yet. The 3-D display will be incorporated into the system in further stages of the development. The next is synchronization "While Do" loop which starts the main cycle every 100 ms. All the next programs are invoked only by the main cycle.

SCALE - Scales input integer values which are read from the +I/O into radious floating values.

COPY - Copies input floating values into output floating values.

INTTRN & UDATTR & CARTSN - Transformation routines which are transforming values in link space into cartesian space.

SNSCVV - Sensors conversion routine.

CRITER - Sensors stress and torque criteria routine which protects the manipulator from damage.

TDMON - 3-D monitoring routine.

DEFPNT - Define point routine - see chart 3.

DEFTRA - Define trajectory routine. It is invoked whenever the trajectory flag is on and specific synchornization conditions are prevailing. See Chart 4.

MONITOR CURRENT PRIMITIVE - In Chart 5 explicit flowchart is shown. It is the automatic commands monitoring program. When a primitive command code is in execution area it invokes the PRIMM - see chart 9 - which monitors primitives execution. If primitive execution is finished INTRP - see chart 8 - is invoked and it fetches a new primitive into the execution area. The right hand branch of floowchart 5 is similar except that it starts when no primitive node is residing in the execution area.

Now, continuing down the main cycle loop, we see:

DIRECT - Joysticks direct control program which changes manipulator links according to values received from joysticks.

RMC - Joysticks resolved motion control program which enables manipulation in cartesiam space.

INVAR - Grippers constant orientation perserving routine - see chart 6.

FEEDM - CRT feedback generating routine - see chart 7.

UNSCALE - Current floating point values are unscaled into integer values which are written into the I/O.

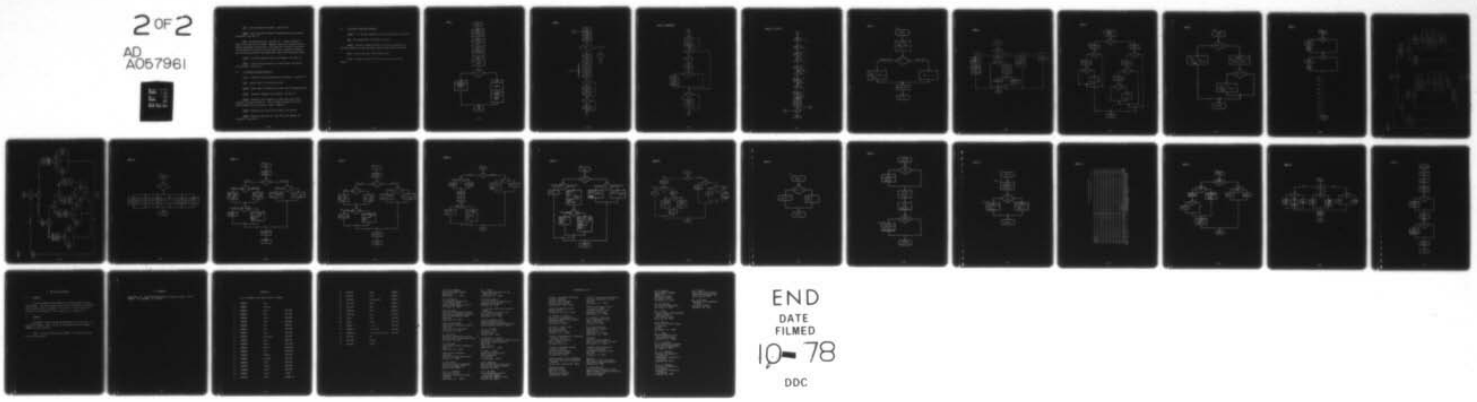
SSW - Sense switches routine.

BRANCH - This routine is invoked by the PRIMM to transfer control to the required primitive routines - see chart 10.

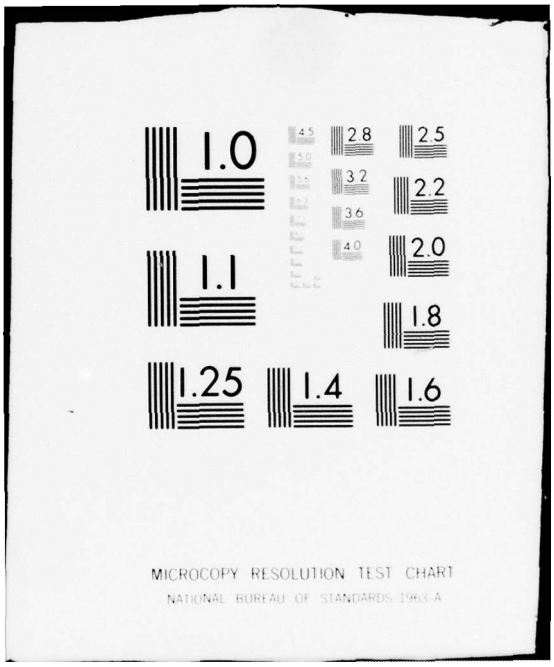
ROTATE - Performs end gripper rotation - see chart 11.

AD-A057 961 PERCEPTRONICS INC WOODLAND HILLS CALIF F/G 5/8
MAN-MACHINE COMMUNICATION IN COMPUTER-AIDED REMOTE MANIPULATION--ETC(U)
MAR 78 W H CROOKS, E SHAKET, Y ALPEROVITCH N00014-76-C-0603
UNCLASSIFIED PATR-1034-78-3 NL

2 OF 2
AD
A057961



END
DATE
FILMED
10-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

GRIP - Closes and opens the gripper - see chart 12.

FORBKW - Moves the gripper forward or backward preserving constant orientation - see chart 13.

GOTO - Performs goto point - see chart 14. A special feature is added to the "goto point" function. Whenever "goto point" executes and the operator manipulates the joysticks in the same time, in order to change the final manipulator's position, the final goal point is adjusted accordingly and the function terminates when manipulator reaches the adjusted goal point.

EXCTRA - Executes trajectory forward and backward - see chart 15.

MANUAL - stops primitives execution to enable manual intervention with joysticks - see chart 16.

4.8 TTY Process Functional Structure

TTYIN - It handles the codes generated at the keyboard - see chart 17.

XFER - Transfer codes to low priority buffer.

GETCHR - Reads codes from keyboard and sticks them into command buffer

ANLIZR - Interprets commands read by GETCHR - see chart 18.

RECGNZ - Recognizes valid sequences of codes using finite state recognizer described in chart 19. The LL1 grammer from which the finite state recognizer was constructed is shown in Appendix A.

CHAGEN - Generates chain under the edit mode - see chart 20.

CODGEN - Generates codes and sets flags when valid commands are recognized - see chart 21.

4.9 I/O Process Functional Structure

I/ODVR - It is the Main program of the I/O process shown in chart 22.

READ - Read manipulators link values into core.

COMPAR - Emergency compare provided to stop the arm whenever the difference between the input and output values exceeds certain threshold.

WRITE - Writes new values into the manipulator.

TTYOUT - Outputs characters to CRT from high and low priority buffers.

CHART 1

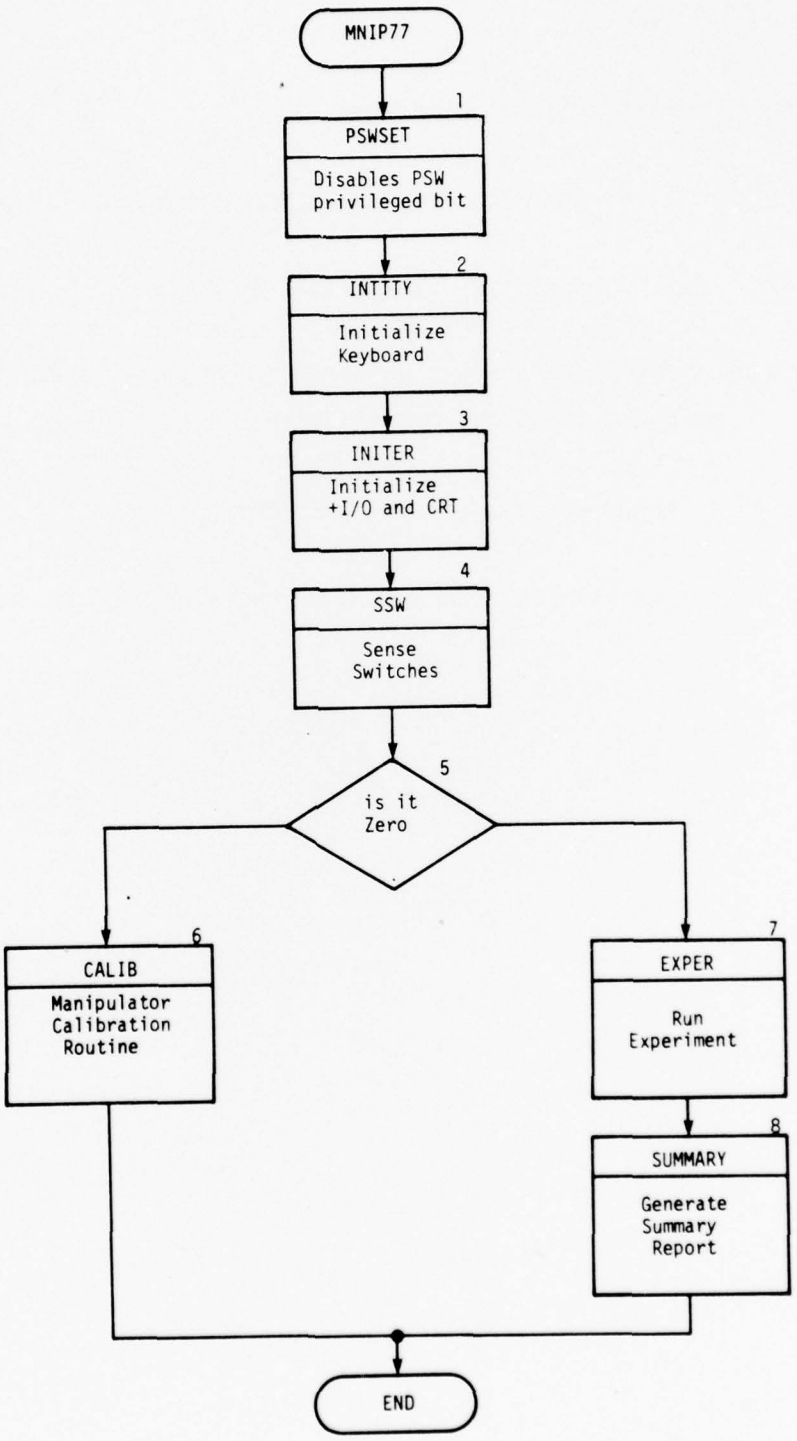

MNIP77
1
PSWSET
Disables PSW privileged bit
2
INTTTY
Initialize Keyboard
3
INITER
Initialize +I/O and CRT
4
SSW
Sense Switches
5
is it Zero
6
CALIB
Manipulator Calibration Routine
7
EXPER
Run Experiment
8
SUMMARY
Generate Summary Report
END

CHART 2

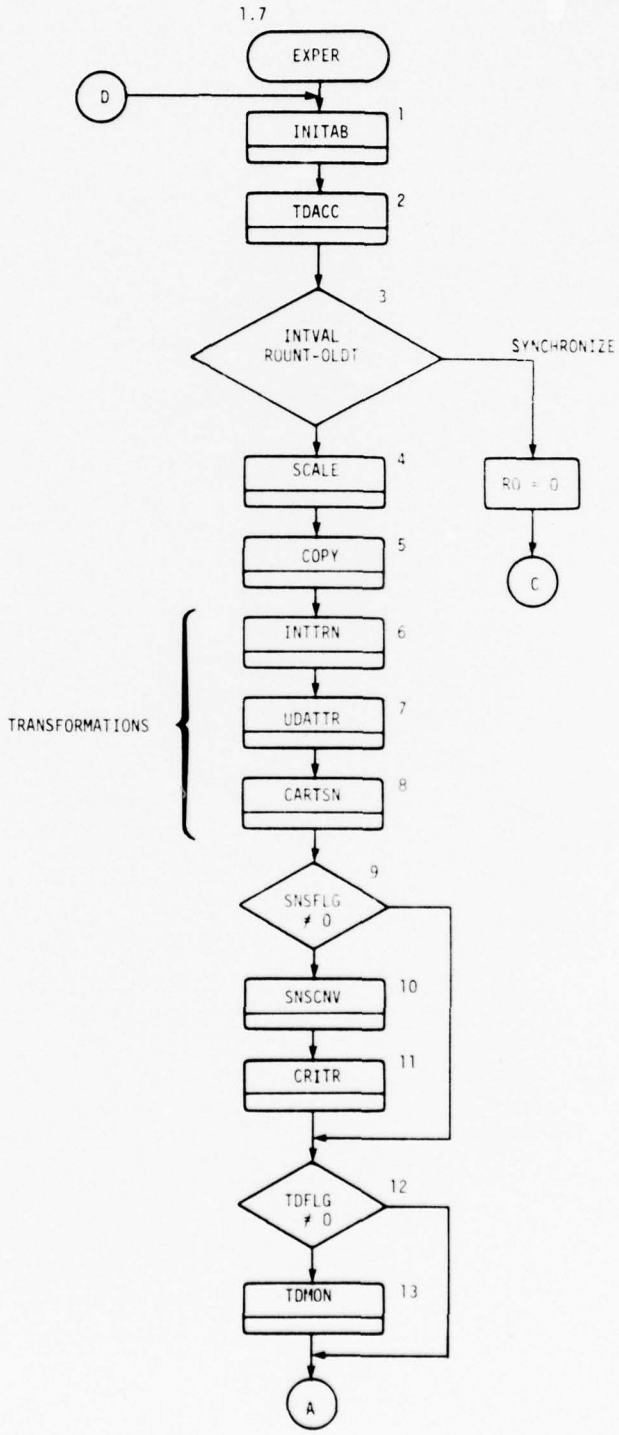

1.7
EXPER
D
INITAB
1
TDACC
2
INTVAL
ROUNT-OLDT
3
SYNCHRONIZE
SCALE
4
R0 = 0
COPY
5
C
INTTRN
6
TRANSFORMATIONS
UDATTR
7
CARTSN
8
SNSFLG
≠ 0
9
SNSCNV
10
CRITR
11
TDFLG
≠ 0
12
TDMON
13
A

CHART 2 (CONTINUED)

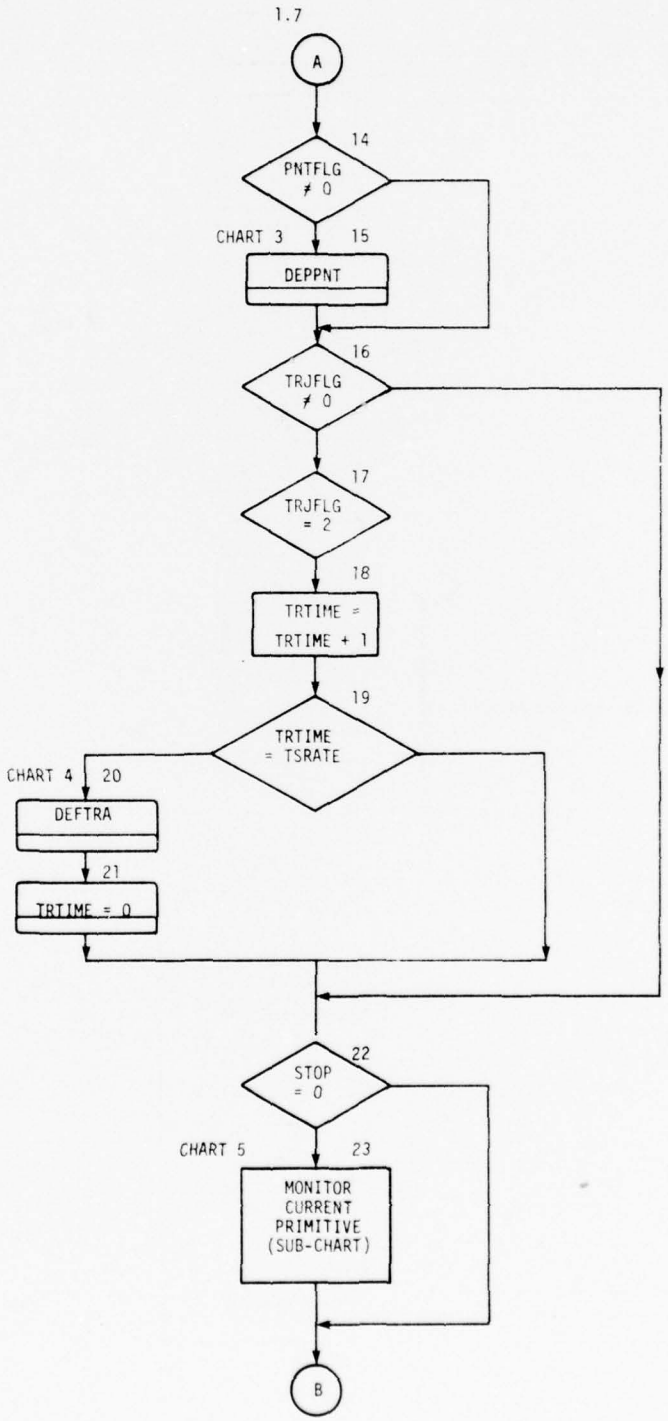


CHART 2 (CONTINUED)

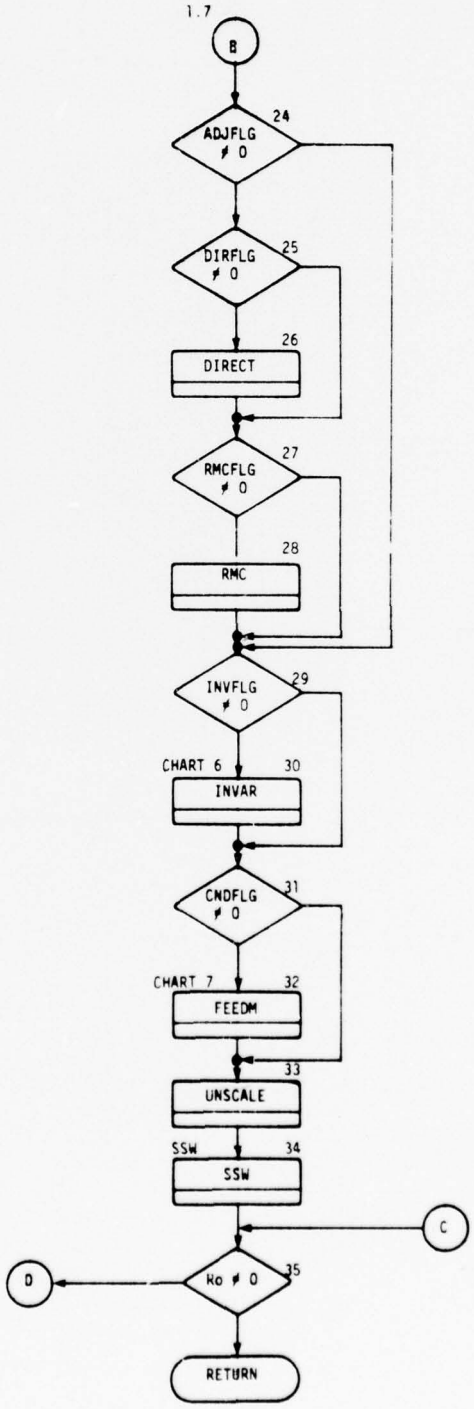


CHART 3

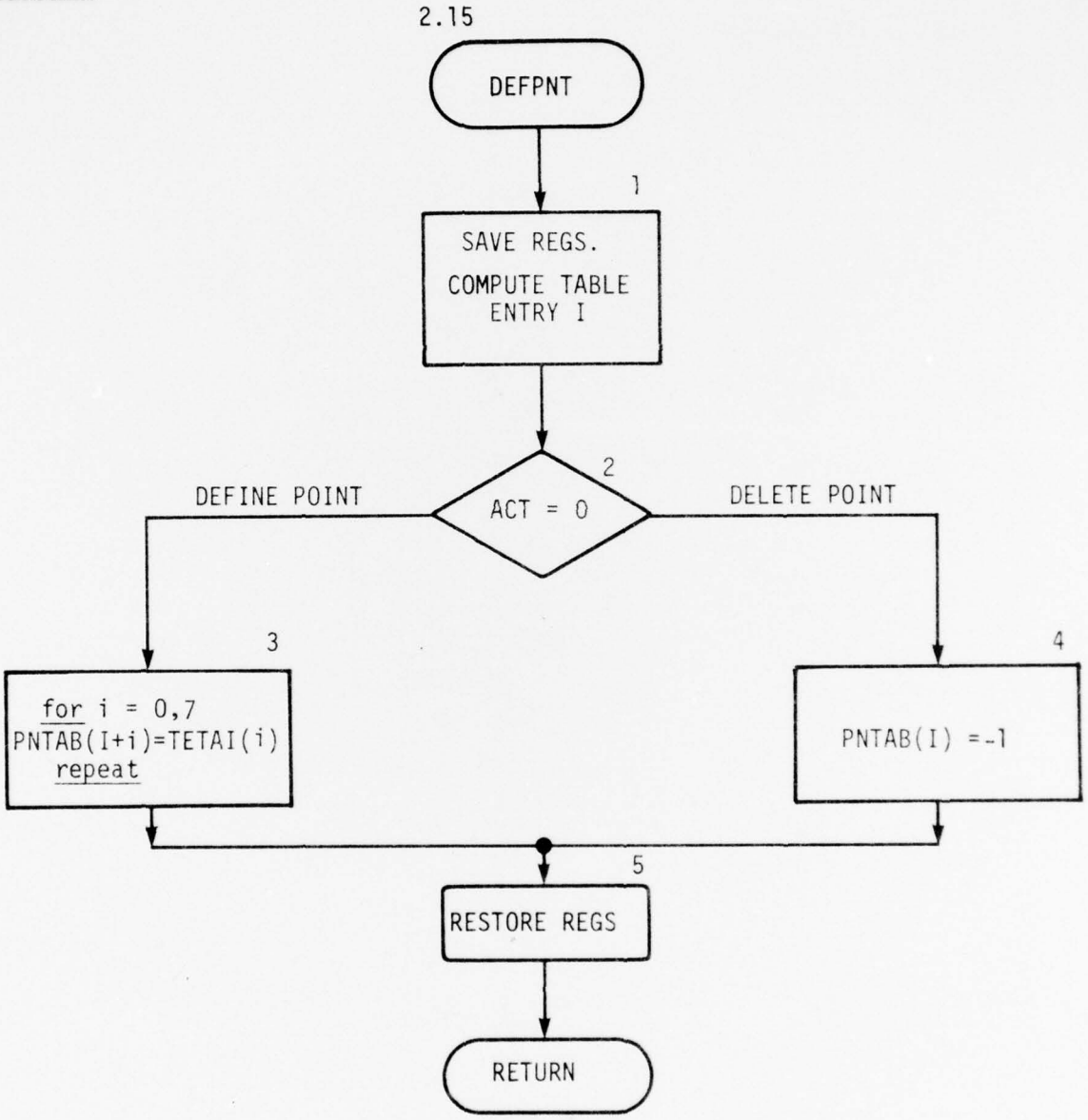

2.15
DEFPNT
1
SAVE REGS.
COMPUTE TABLE ENTRY I
2
ACT = 0
DEFINE POINT
DELETE POINT
3
for i = 0,7
PNTAB(I+i)=TETAI(i)
repeat
4
PNTAB(I) =-1
5
RESTORE REGS
RETURN

CHART 4

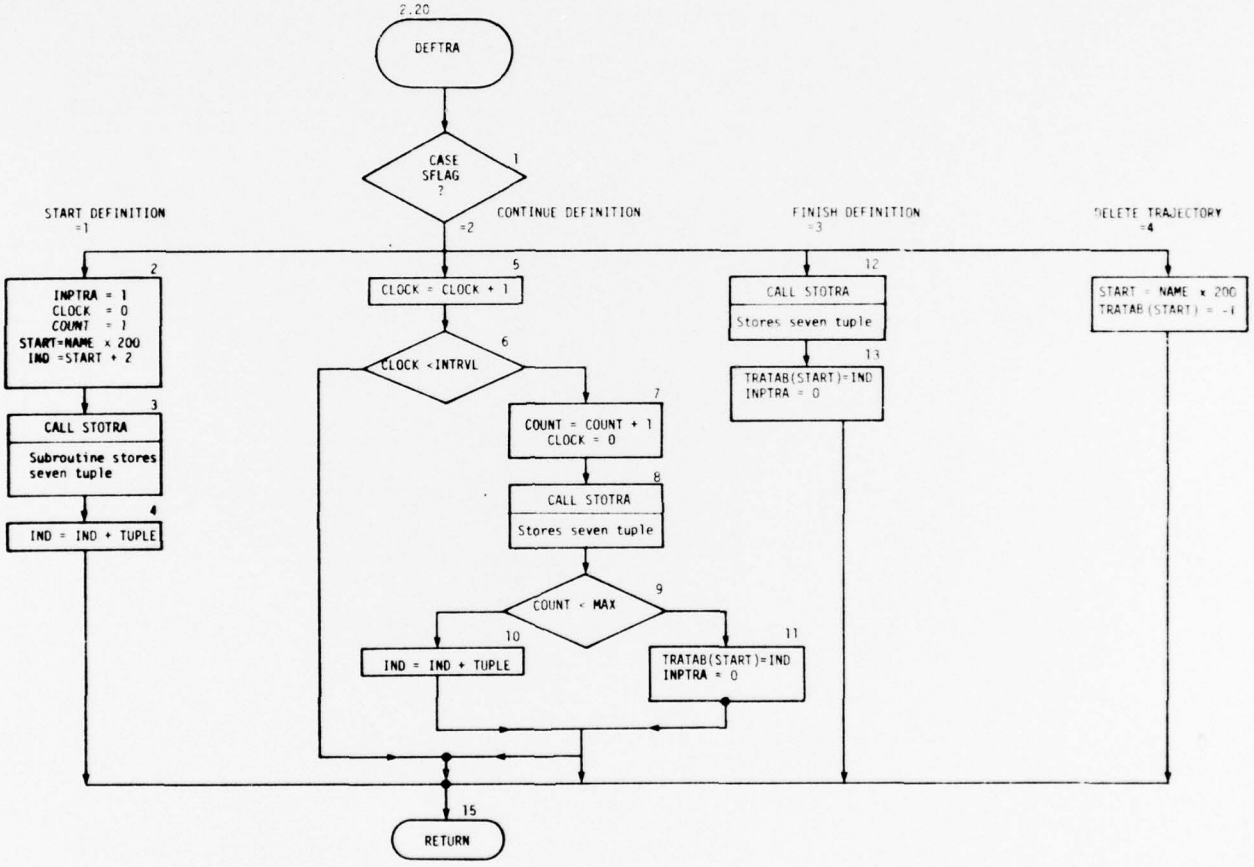

2.20
DEFTRA
CASE SFLAG ?
1
START DEFINITION =1
CONTINUE DEFINITION =2
FINISH DEFINITION =3
DELETE TRAJECTORY =4
2
INPTRA = 1
CLOCK = 0
COUNT = 1
START=NAME x 200
IND =START + 2
3
CALL STOTRA
Subroutine stores seven tuple
4
IND = IND + TUPLE
5
CLOCK = CLOCK + 1
6
CLOCK <INTRVL
7
COUNT = COUNT + 1
CLOCK = 0
8
CALL STOTRA
Stores seven tuple
9
COUNT < MAX
10
IND = IND + TUPLE
11
TRATAB(START)=IND
INPTRA = 0
12
CALL STOTRA
Stores seven tuple
13
TRATAB(START)=IND
INPTRA = 0
START = NAME x 200
TRATAB (START) = -1
15
RETURN

CHART 5

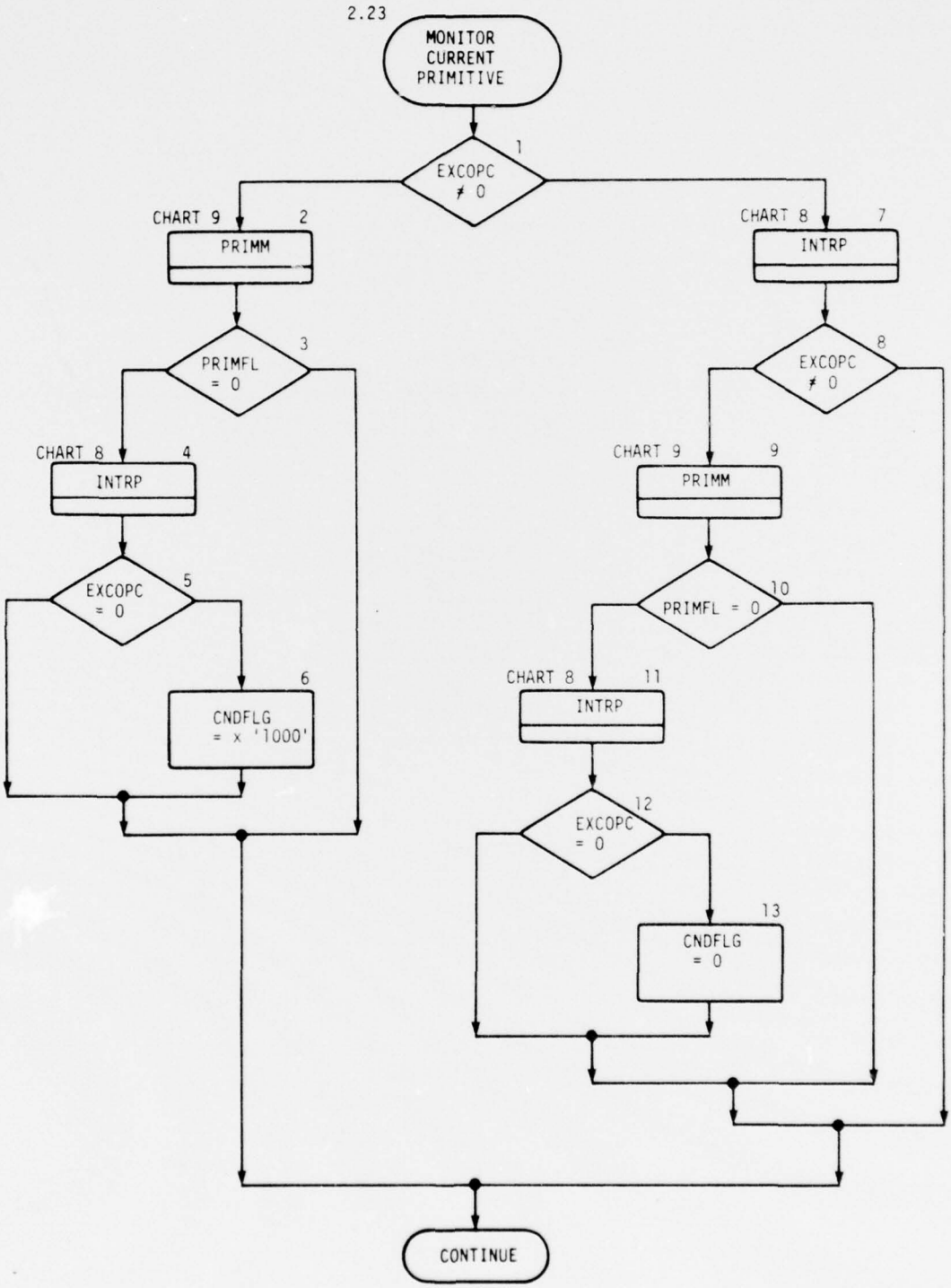

2.23
MONITOR CURRENT PRIMITIVE
1
EXCOPC ≠ 0
CHART 9
2
PRIMM
3
PRIMFL = 0
CHART 8
4
INTRP
5
EXCOPC = 0
6
CNDFLG = x '1000'
CHART 8
7
INTRP
8
EXCOPC ≠ 0
CHART 9
9
PRIMM
10
PRIMFL = 0
CHART 8
11
INTRP
12
EXCOPC = 0
13
CNDFLG = 0
CONTINUE

CHART 6

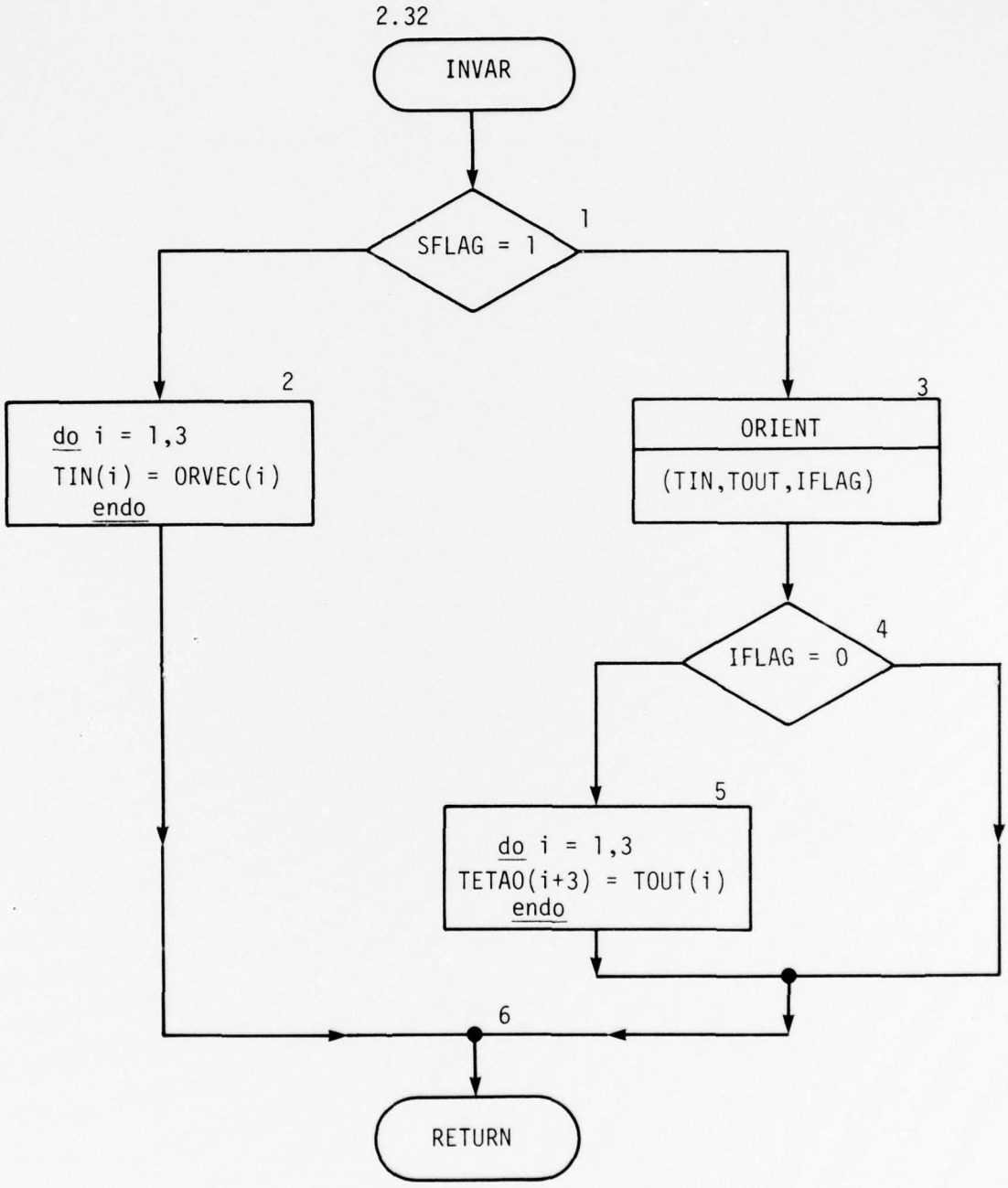

2.32
INVAR
1
SFLAG = 1
2
do i = 1,3
TIN(i) = ORVEC(i)
endo
3
ORIENT
(TIN,TOUT,IFLAG)
4
IFLAG = 0
5
do i = 1,3
TETAO(i+3) = TOUT(i)
endo
6
RETURN

CHART 7

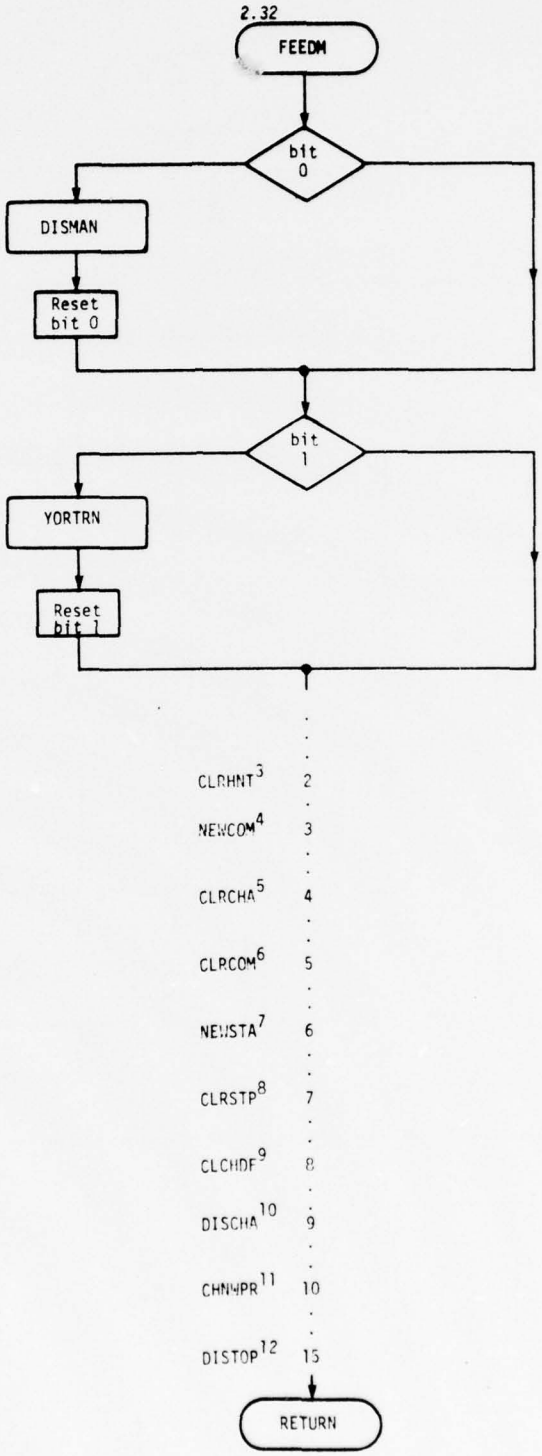

2.32
FEEDM
bit 0
DISMAN
Reset bit 0
bit 1
YORTRN
Reset bit 1
CLRHNT[3] 2
NEWCOM[4] 3
CLRCHA[5] 4
CLRCOM[6] 5
NEWSTA[7] 6
CLRSTP[8] 7
CLCHDF[9] 8
DISCHA[10] 9
CHNWPR[11] 10
DISTOP[12] 15
RETURN

CHART 8

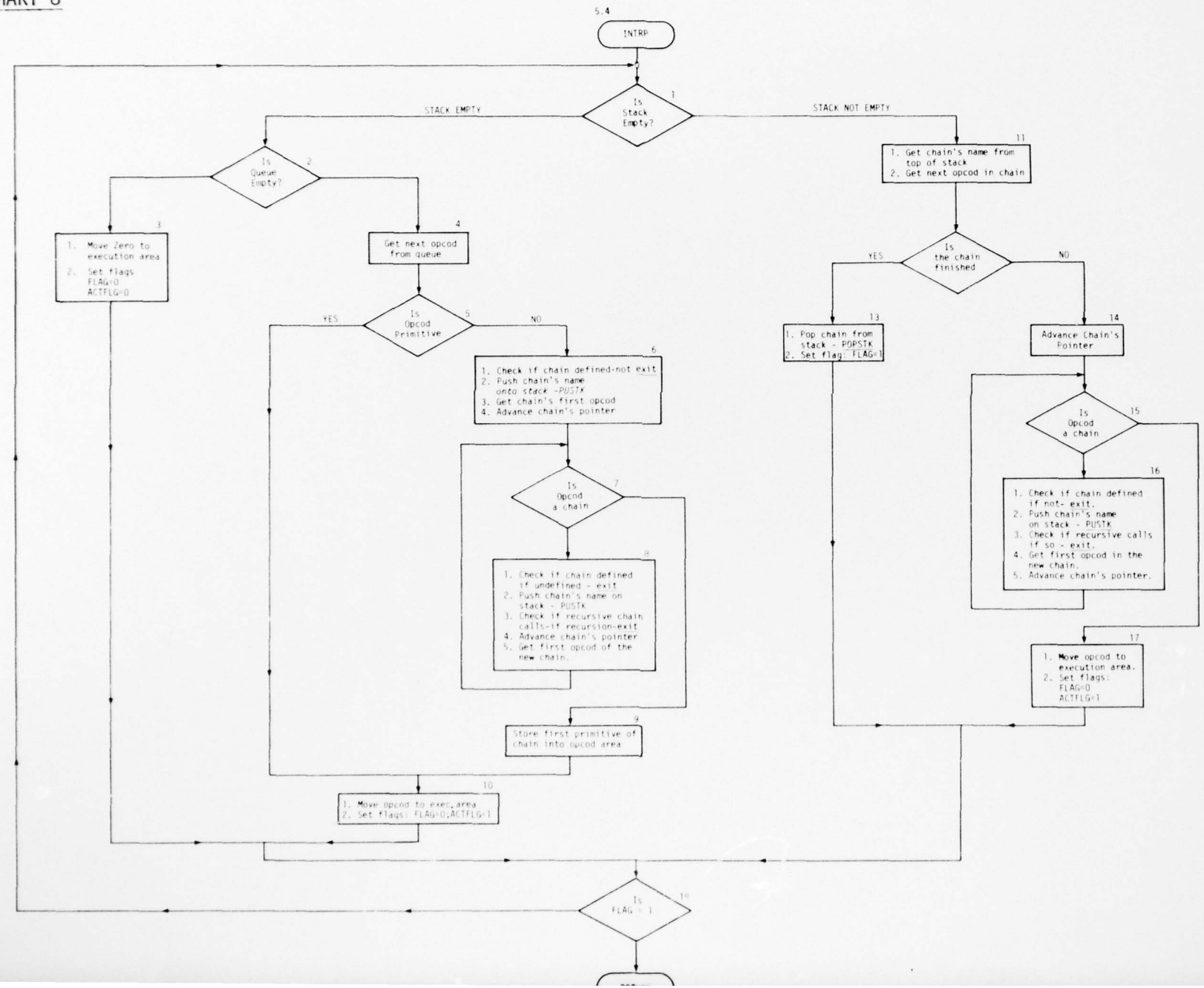

5.4
INTRP
1
Is Stack Empty?
STACK EMPTY
STACK NOT EMPTY
2
Is Queue Empty?
3
1. Move Zero to execution area
2. Set flags FLAG=0 ACTFLG=0
4
Get next opcod from queue
5
Is Opcod Primitive
YES
NO
6
1. Check if chain defined-not exit
2. Push chain's name onto stack -PUSTK
3. Get chain's first opcod
4. Advance chain's pointer
7
Is Opcnd a chain
8
1. Check if chain defined if undefined - exit
2. Push chain's name on stack - PUSTK
3. Check if recursive chain calls-if recursion-exit
4. Advance chain's pointer
5. Get first opcod of the new chain.
9
Store first primitive of chain into opcod area
10
1. Move opcod to exec.area
2. Set flags: FLAG=0;ACTFLG=1
11
1. Get chain's name from top of stack
2. Get next opcod in chain
Is the chain finished
YES
NO
13
1. Pop chain from stack - POPSTK
2. Set flag: FLAG=1
14
Advance Chain's Pointer
15
Is Opcod a chain
16
1. Check if chain defined if not- exit.
2. Push chain's name on stack - PUSTK
3. Check if recursive calls if so - exit.
4. Get first opcod in the new chain.
5. Advance chain's pointer.
17
1. Move opcod to execution area.
2. Set flags: FLAG=0 ACTFLG=1
18
Is FLAG = 1

CHART 9

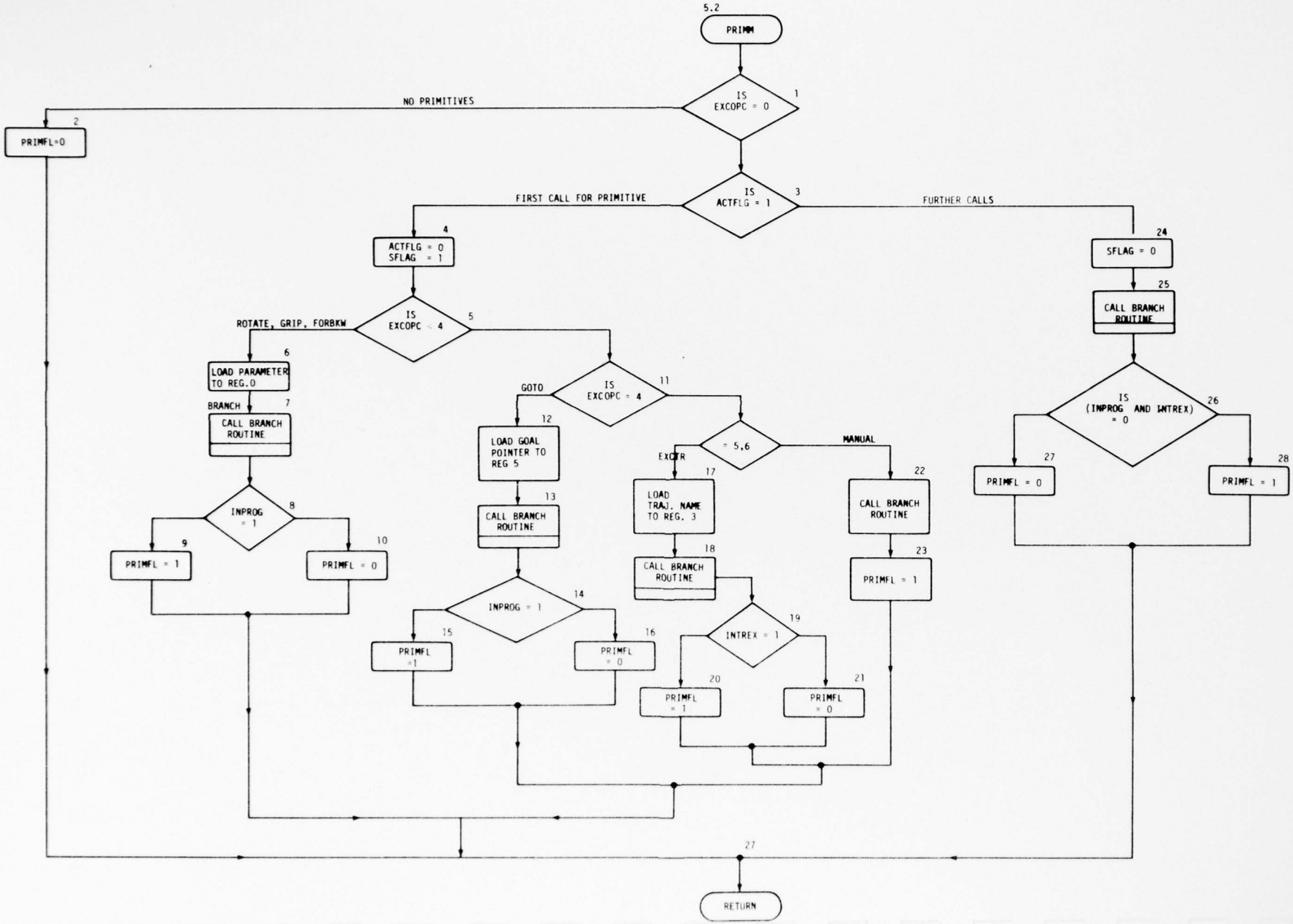

5.2
PRIMM
IS EXCOPC = 0
1
NO PRIMITIVES
2
PRIMFL=0
IS ACTFLG = 1
3
FIRST CALL FOR PRIMITIVE
FURTHER CALLS
4
ACTFLG = 0
SFLAG = 1
IS EXCOPC < 4
5
ROTATE, GRIP, FORBKW
6
LOAD PARAMETER TO REG.0
BRANCH
7
CALL BRANCH ROUTINE
INPROG = 1
8
9
PRIMFL = 1
10
PRIMFL = 0
GOTO
IS EXCOPC = 4
11
12
LOAD GOAL POINTER TO REG 5
13
CALL BRANCH ROUTINE
INPROG = 1
14
15
PRIMFL =1
16
PRIMFL = 0
= 5,6
EXCTR
MANUAL
17
LOAD TRAJ. NAME TO REG. 3
18
CALL BRANCH ROUTINE
INTREX = 1
19
20
PRIMFL = 1
21
PRIMFL = 0
22
CALL BRANCH ROUTINE
23
PRIMFL = 1
24
SFLAG = 0
25
CALL BRANCH ROUTINE
IS (INPROG AND INTREX) = 0
26
27
PRIMFL = 0
28
PRIMFL = 1
27
RETURN

CHART 10

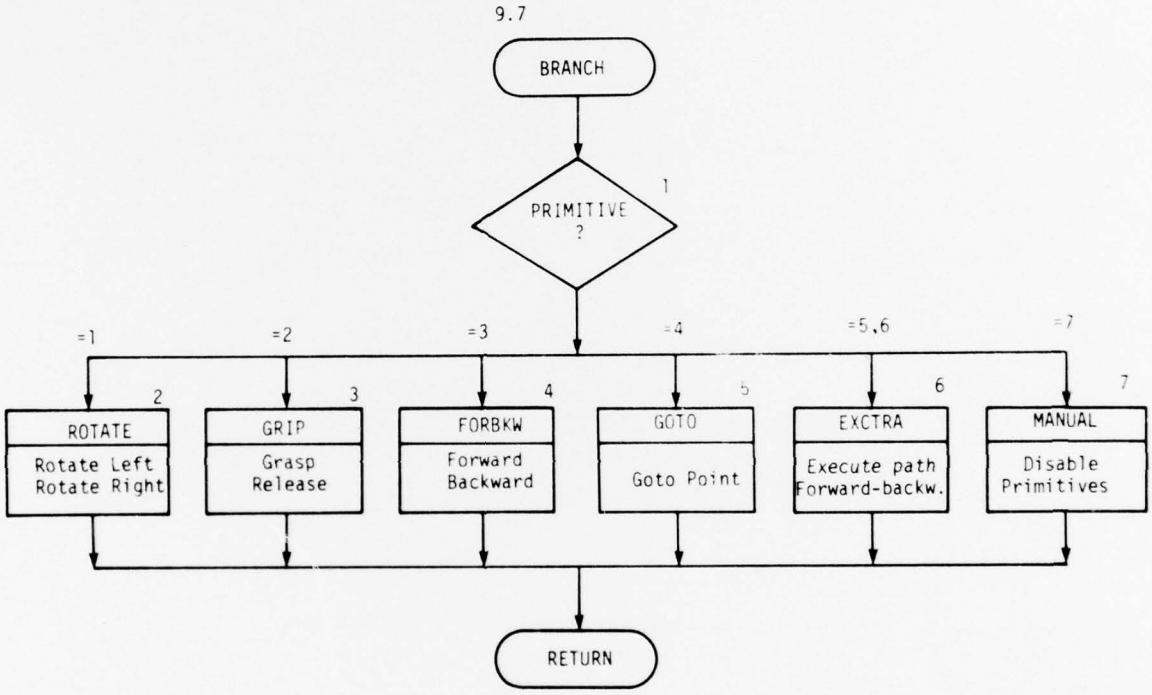

9.7
BRANCH
1
PRIMITIVE ?
=1
=2
=3
=4
=5,6
=7
2
ROTATE
Rotate Left
Rotate Right
3
GRIP
Grasp
Release
4
FORBKW
Forward
Backward
5
GOTO
Goto Point
6
EXCTRA
Execute path
Forward-backw.
7
MANUAL
Disable
Primitives
RETURN

CHART 11

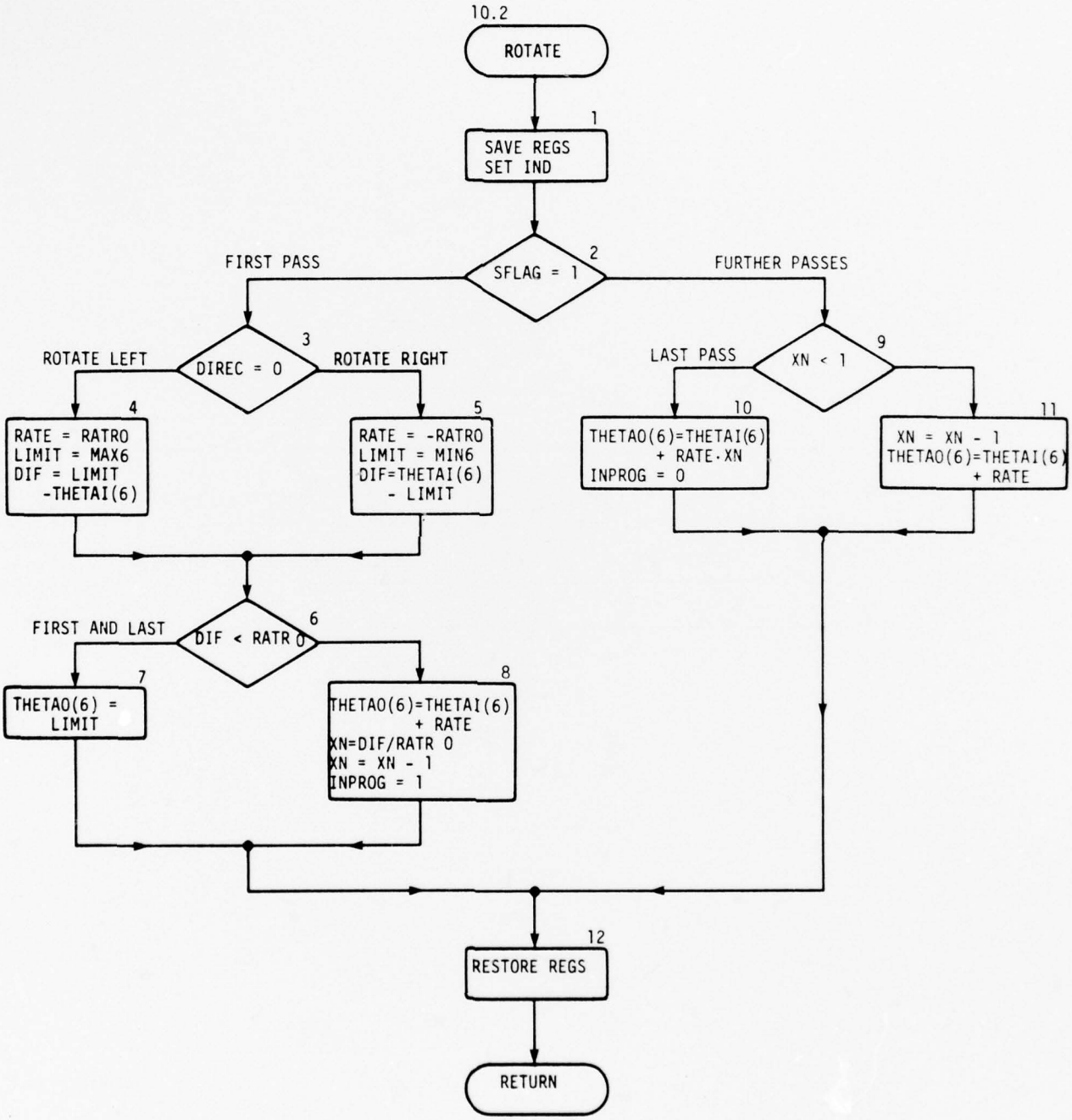

10.2
ROTATE
1
SAVE REGS
SET IND
2
SFLAG = 1
FIRST PASS
FURTHER PASSES
3
DIREC = 0
ROTATE LEFT
ROTATE RIGHT
9
XN < 1
LAST PASS
4
RATE = RATRO
LIMIT = MAX6
DIF = LIMIT
-THETAI(6)
5
RATE = -RATRO
LIMIT = MIN6
DIF=THETAI(6)
- LIMIT
10
THETAO(6)=THETAI(6)
+ RATE·XN
INPROG = 0
11
XN = XN - 1
THETAO(6)=THETAI(6)
+ RATE
6
DIF < RATR 0
FIRST AND LAST
7
THETAO(6) =
LIMIT
8
THETAO(6)=THETAI(6)
+ RATE
XN=DIF/RATR 0
XN = XN - 1
INPROG = 1
12
RESTORE REGS
RETURN

CHART 12

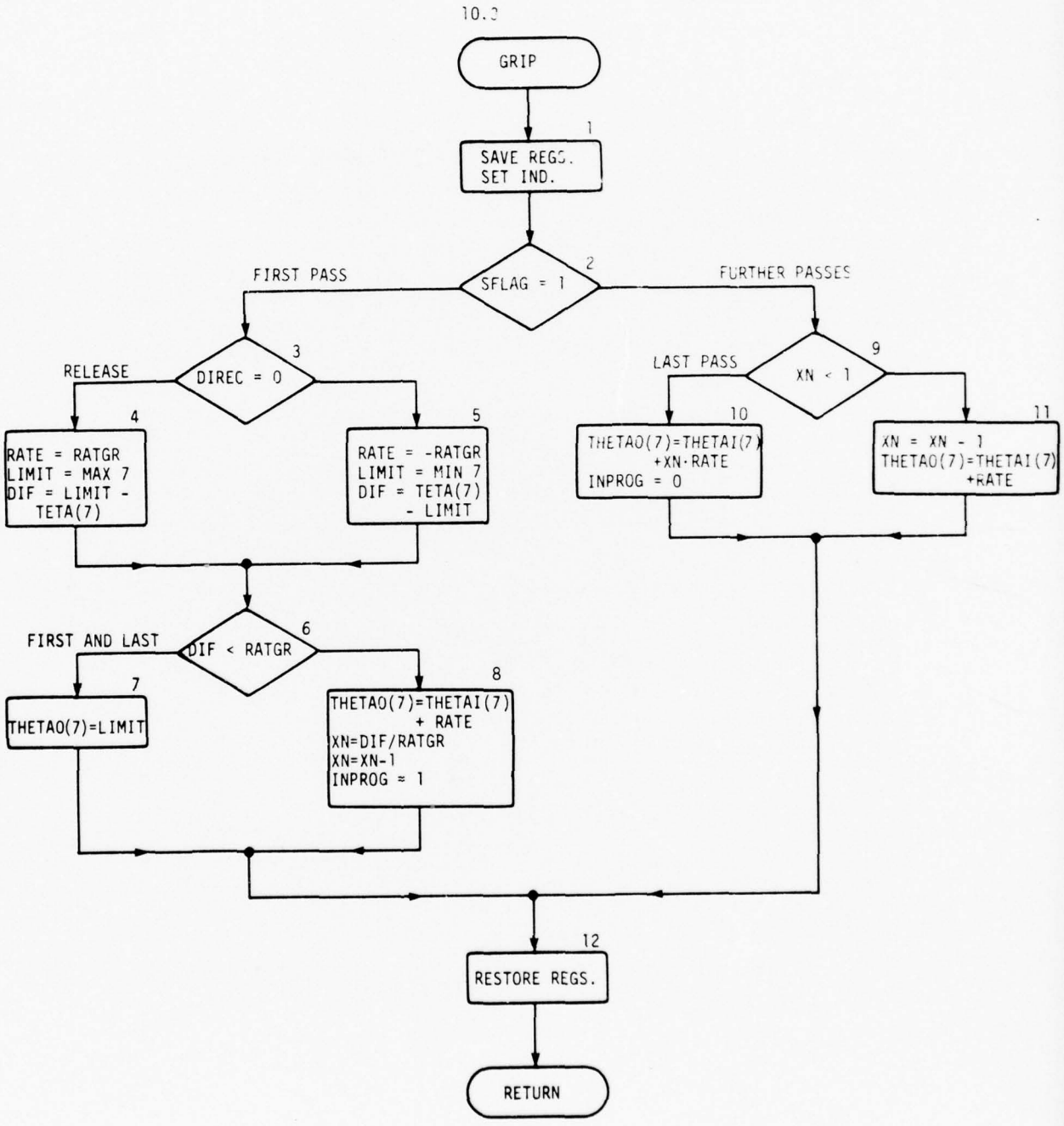

10.3
GRIP
1
SAVE REGS.
SET IND.
2
SFLAG = 1
FIRST PASS
FURTHER PASSES
3
DIREC = 0
RELEASE
4
RATE = RATGR
LIMIT = MAX 7
DIF = LIMIT -
TETA(7)
5
RATE = -RATGR
LIMIT = MIN 7
DIF = TETA(7)
- LIMIT
6
DIF < RATGR
FIRST AND LAST
7
THETAO(7)=LIMIT
8
THETAO(7)=THETAI(7)
+ RATE
XN=DIF/RATGR
XN=XN-1
INPROG = 1
9
XN < 1
LAST PASS
10
THETAO(7)=THETAI(7)
+XN·RATE
INPROG = 0
11
XN = XN - 1
THETAO(7)=THETAI(7)
+RATE
12
RESTORE REGS.
RETURN

CHART 13

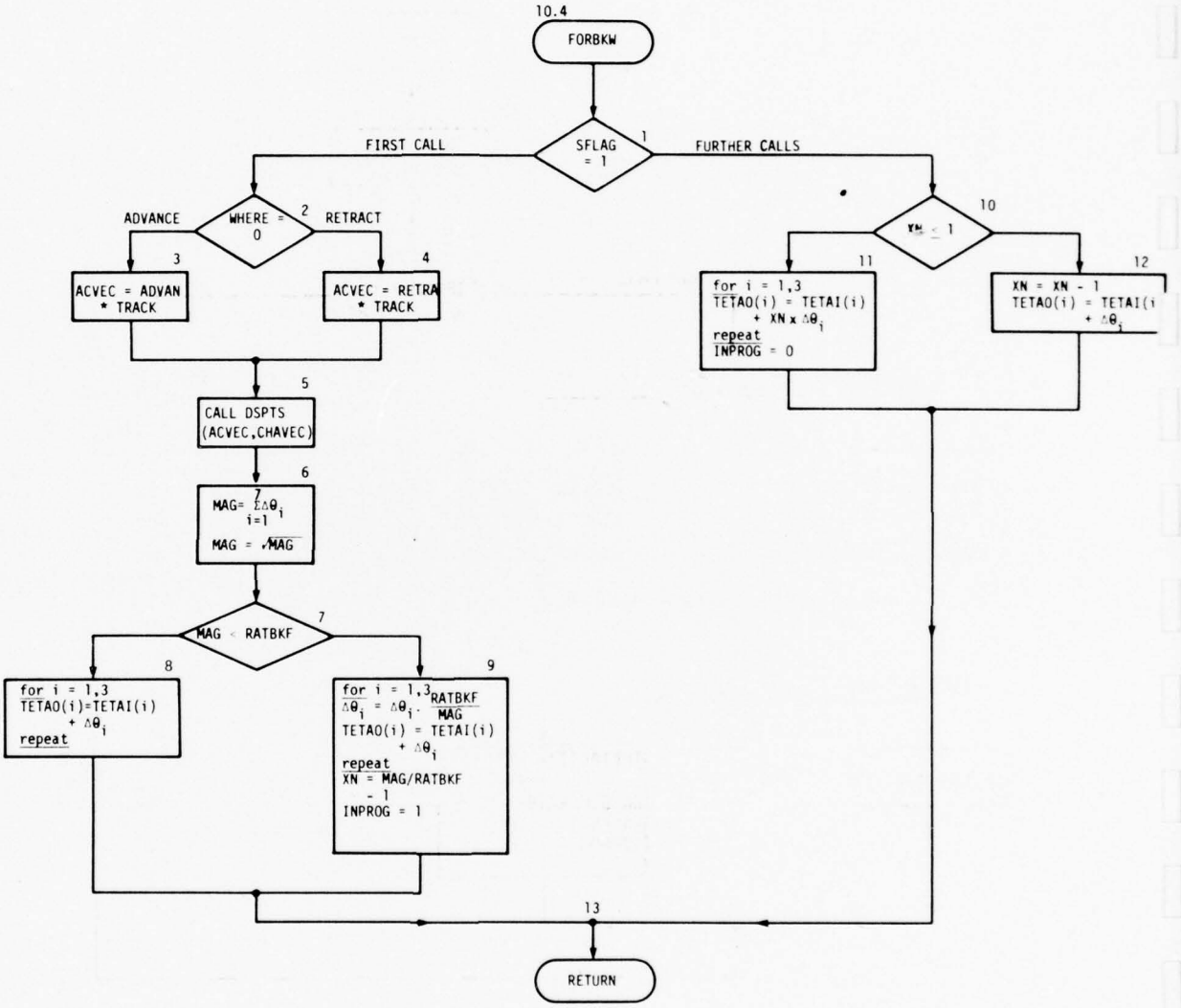

10.4
FORBKW
SFLAG = 1
1
FIRST CALL
FURTHER CALLS
WHERE = 0
2
ADVANCE
RETRACT
3
ACVEC = ADVAN * TRACK
4
ACVEC = RETRA * TRACK
5
CALL DSPTS (ACVEC,CHAVEC)
6
MAG = Σ Δθ_i
MAG = √MAG
7
MAG < RATBKF
8
for i = 1,3
TETAO(i)=TETAI(i) + Δθ_i
repeat
9
for i = 1,3
Δθ_i = Δθ_i · RATBKF / MAG
TETAO(i) = TETAI(i) + Δθ_i
repeat
XN = MAG/RATBKF - 1
INPROG = 1
10
XN ≤ 1
11
for i = 1,3
TETAO(i) = TETAI(i) + XN x Δθ_i
repeat
INPROG = 0
12
XN = XN - 1
TETAO(i) = TETAI(i) + Δθ_i
13
RETURN

CHART 14

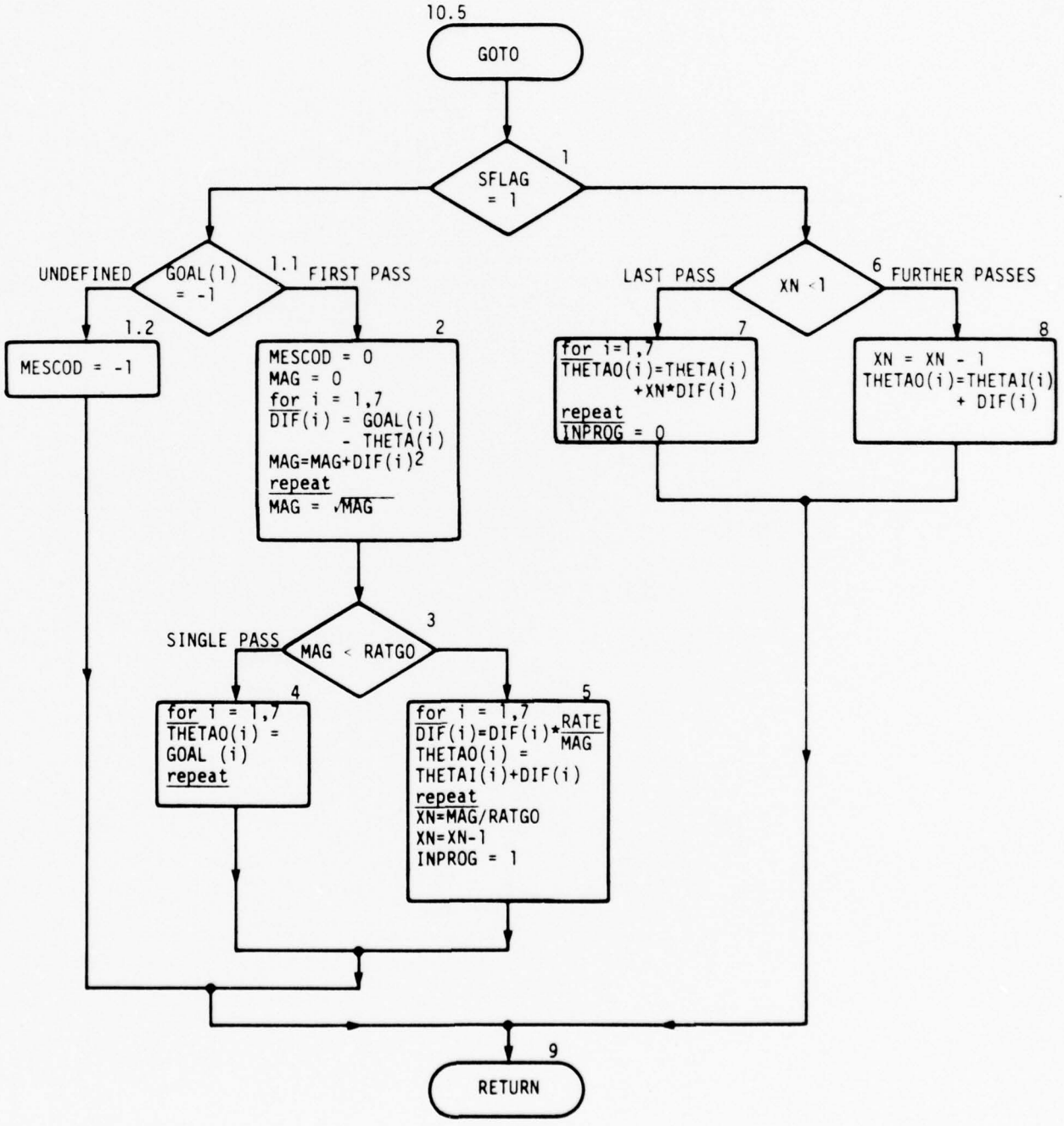

10.5
GOTO
1
SFLAG = 1
UNDEFINED
GOAL(1) = -1
1.1
FIRST PASS
LAST PASS
XN < 1
6
FURTHER PASSES
1.2
MESCOD = -1
2
MESCOD = 0
MAG = 0
for i = 1,7
DIF(i) = GOAL(i) - THETA(i)
MAG=MAG+DIF(i)2
repeat
MAG = √MAG
7
for i=1,7
THETAO(i)=THETA(i) +XN*DIF(i)
repeat
INPROG = 0
8
XN = XN - 1
THETAO(i)=THETAI(i) + DIF(i)
SINGLE PASS
3
MAG < RATGO
4
for i = 1,7
THETAO(i) = GOAL (i)
repeat
5
for i = 1,7
DIF(i)=DIF(i)*RATE/MAG
THETAO(i) = THETAI(i)+DIF(i)
repeat
XN=MAG/RATGO
XN=XN-1
INPROG = 1
9
RETURN

CHART 15

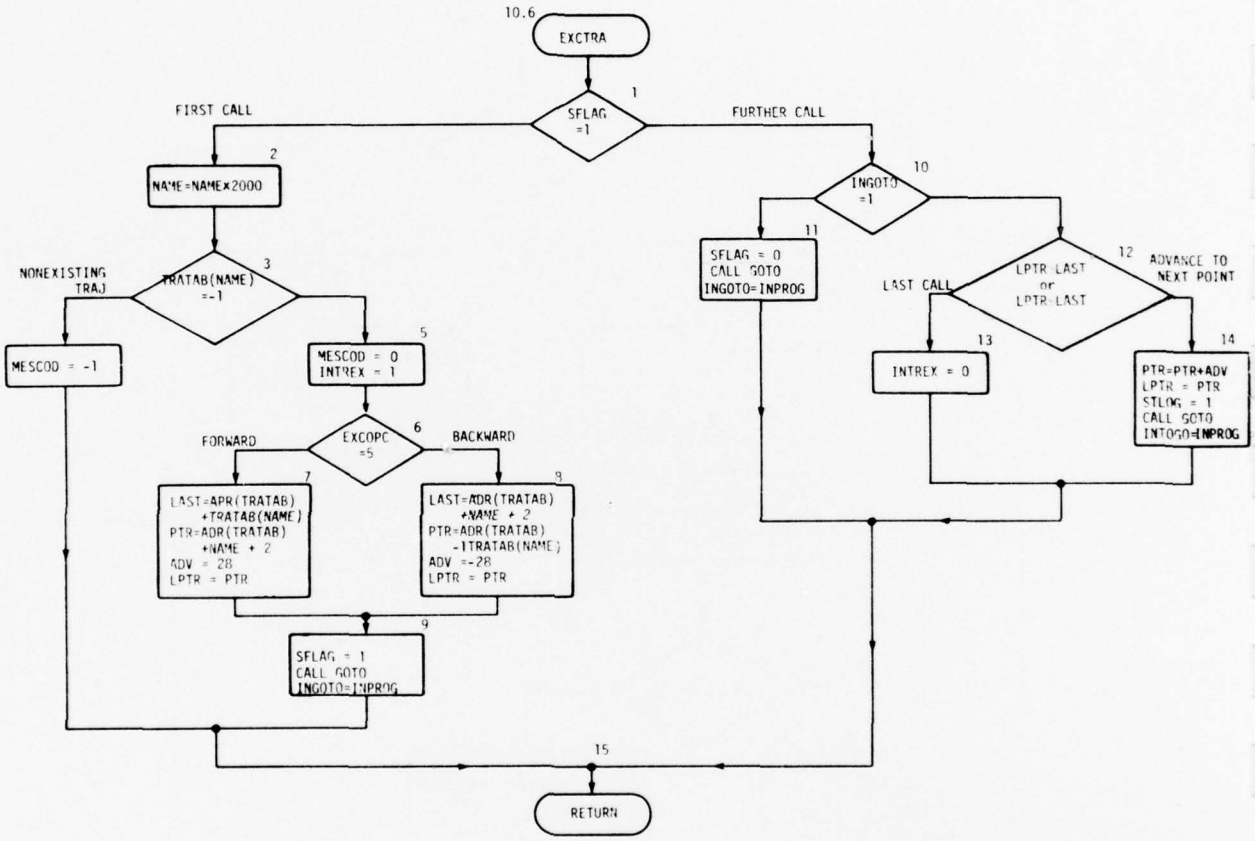

10.6
EXCTRA
1
SFLAG =1
FIRST CALL
FURTHER CALL
2
NAME=NAME×2000
3
TRATAB(NAME) =-1
NONEXISTING TRAJ
MESCOD = -1
5
MESCOD = 0
INTREX = 1
6
EXCOPC =5
FORWARD
BACKWARD
7
LAST=APR(TRATAB) +TRATAB(NAME)
PTR=ADR(TRATAB) +NAME + 2
ADV = 28
LPTR = PTR
8
LAST=ADR(TRATAB) +NAME + 2
PTR=ADR(TRATAB) -1TRATAB(NAME)
ADV =-28
LPTR = PTR
9
SFLAG = 1
CALL GOTO
INGOTO=INPROG
10
INGOTO =1
11
SFLAG = 0
CALL GOTO
INGOTO=INPROG
12
LPTR LAST or LPTR=LAST
LAST CALL
ADVANCE TO NEXT POINT
13
INTREX = 0
14
PTR=PTR+ADV
LPTR = PTR
STLOG = 1
CALL GOTO
INTOGO=INPROG
15
RETURN

CHART 16

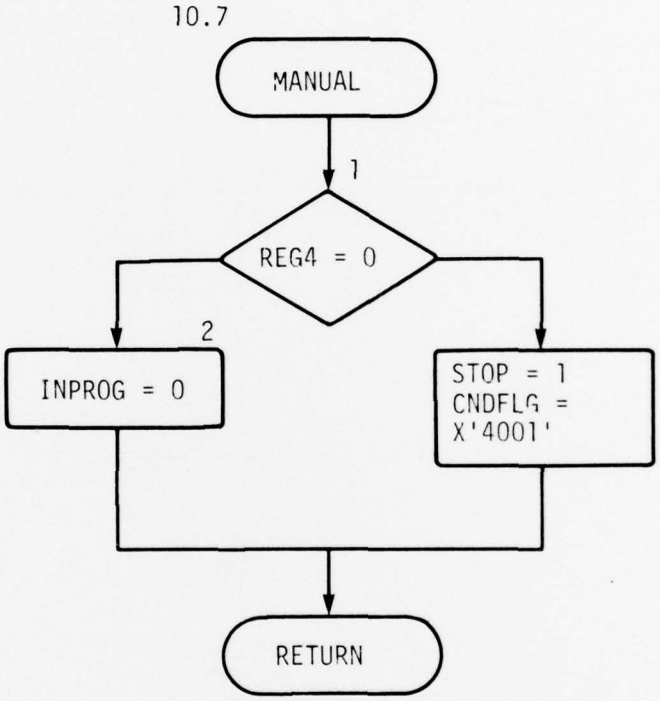


CHART 17

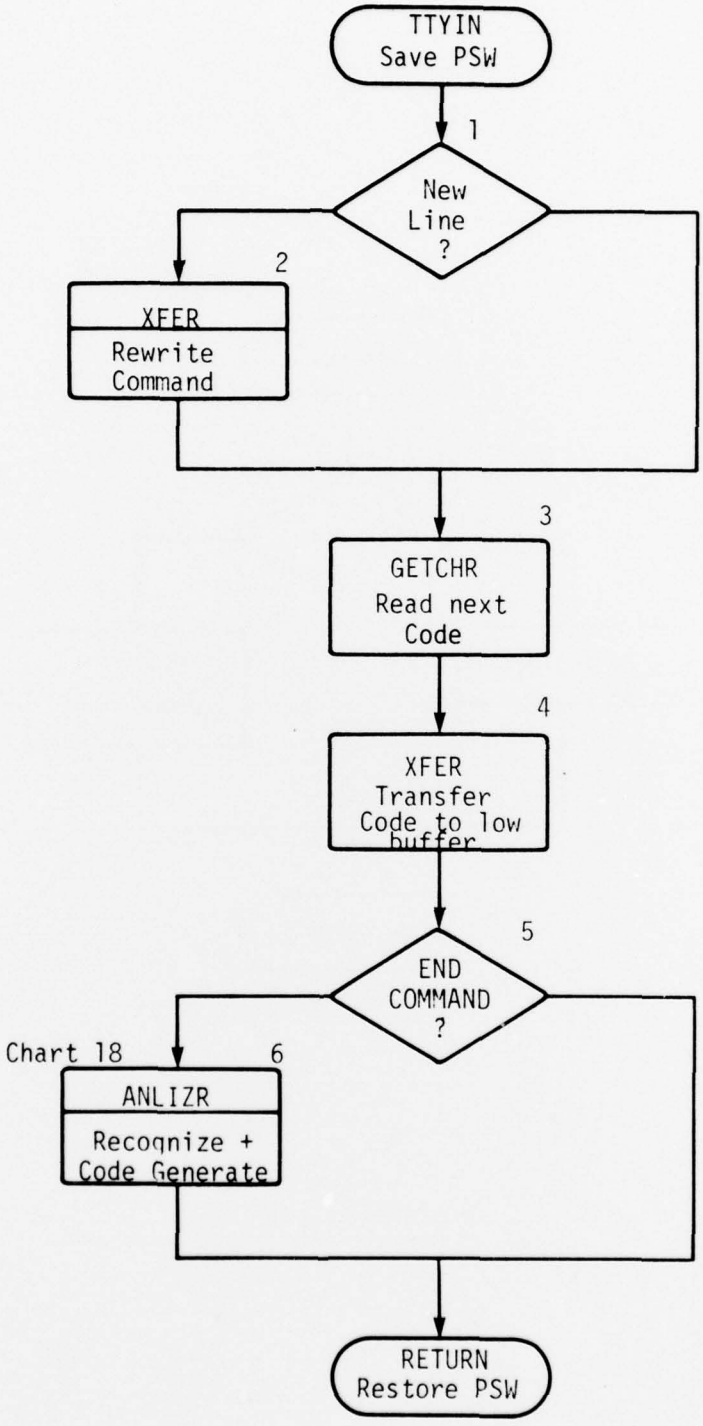

TTYIN
Save PSW
1
New
Line
?
2
XFER
Rewrite
Command
3
GETCHR
Read next
Code
4
XFER
Transfer
Code to low
buffer
5
END
COMMAND
?
Chart 18
6
ANLIZR
Recognize +
Code Generate
RETURN
Restore PSW

CHART 18

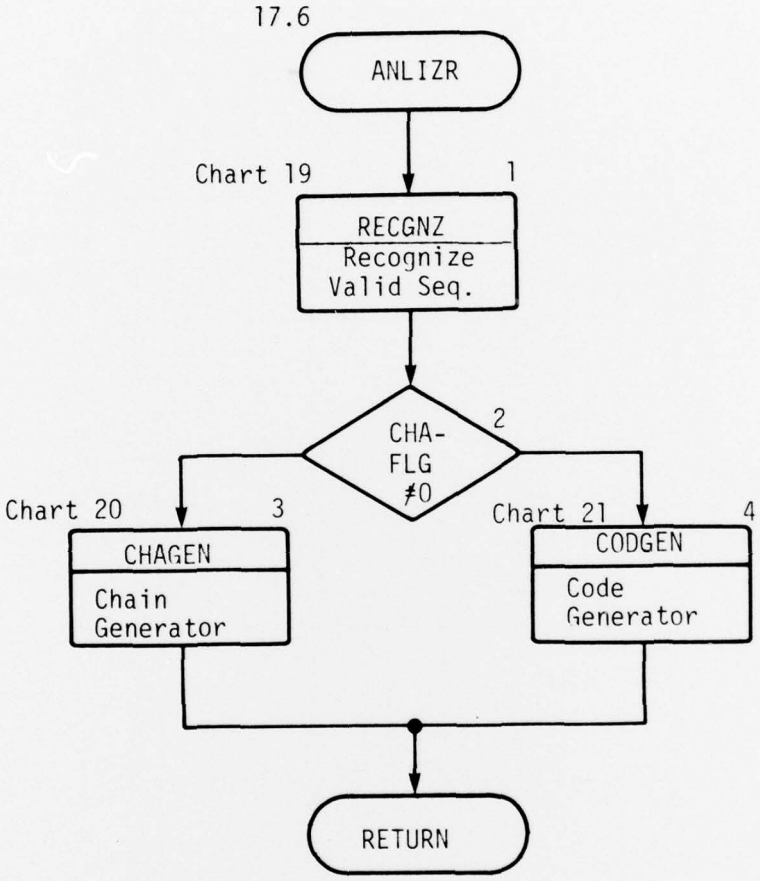
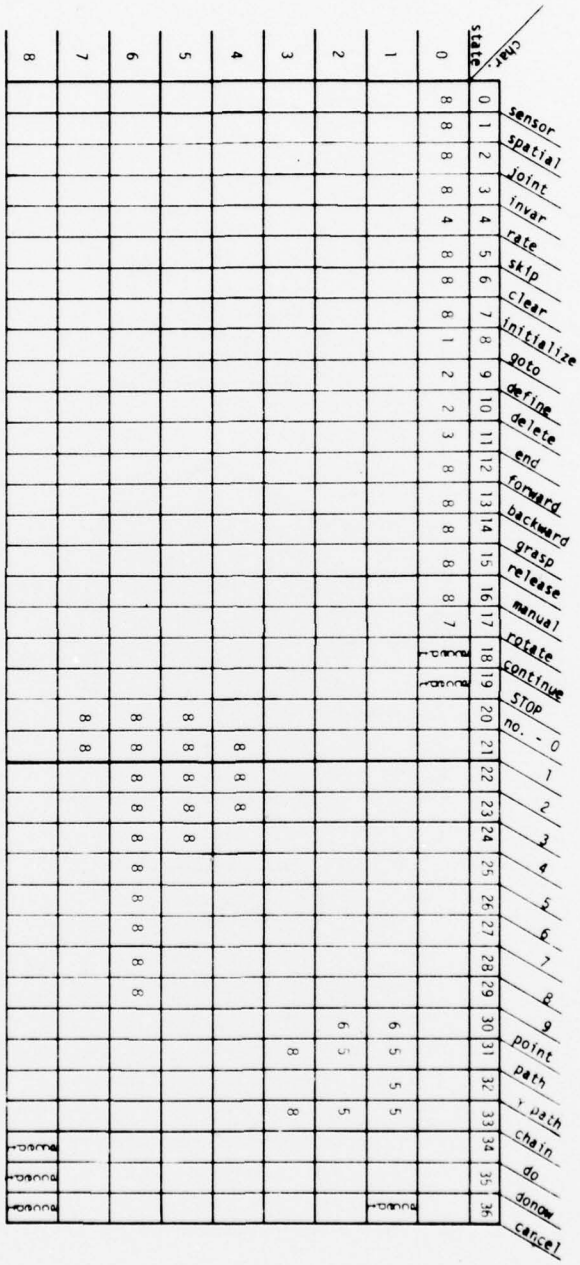

17.6
ANLIZR
Chart 19
1
RECGNZ
Recognize Valid Seq.
CHA-FLG ≠0
2
Chart 20
3
CHAGEN
Chain Generator
Chart 21
4
CODGEN
Code Generator
RETURN

CHART 19



Remarks: 1. Internal codes are shown in first line.

2. Empty entries are transitions to error routine.

CHART 20

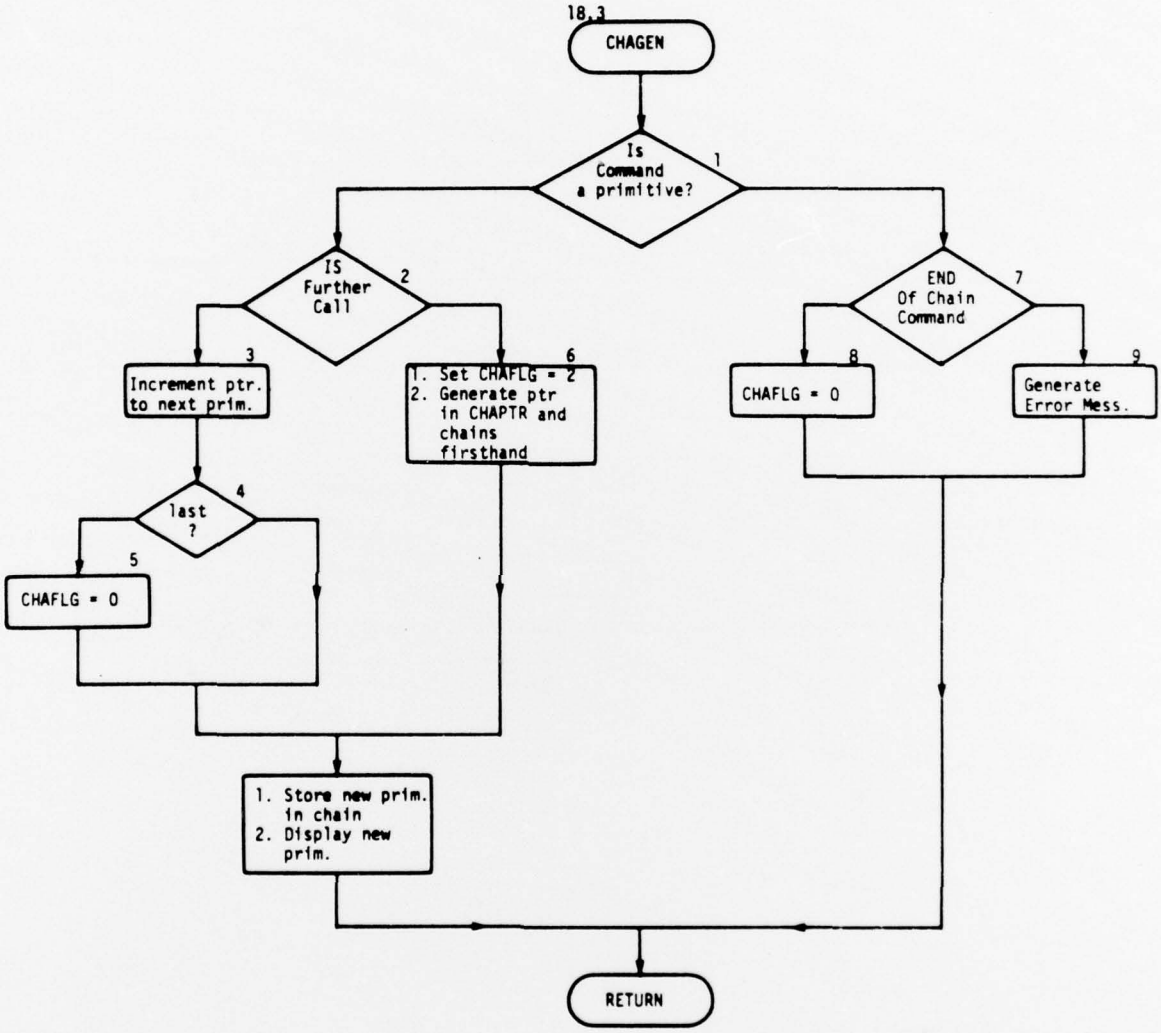

18.3
CHAGEN
Is Command a primitive?
1
IS Further Call
2
END Of Chain Command
7
Increment ptr. to next prim.
3
1. Set CHAFLG = 2
2. Generate ptr in CHAPTR and chains firsthand
6
CHAFLG = 0
8
Generate Error Mess.
9
last ?
4
CHAFLG = 0
5
1. Store new prim. in chain
2. Display new prim.
RETURN

CHART 21

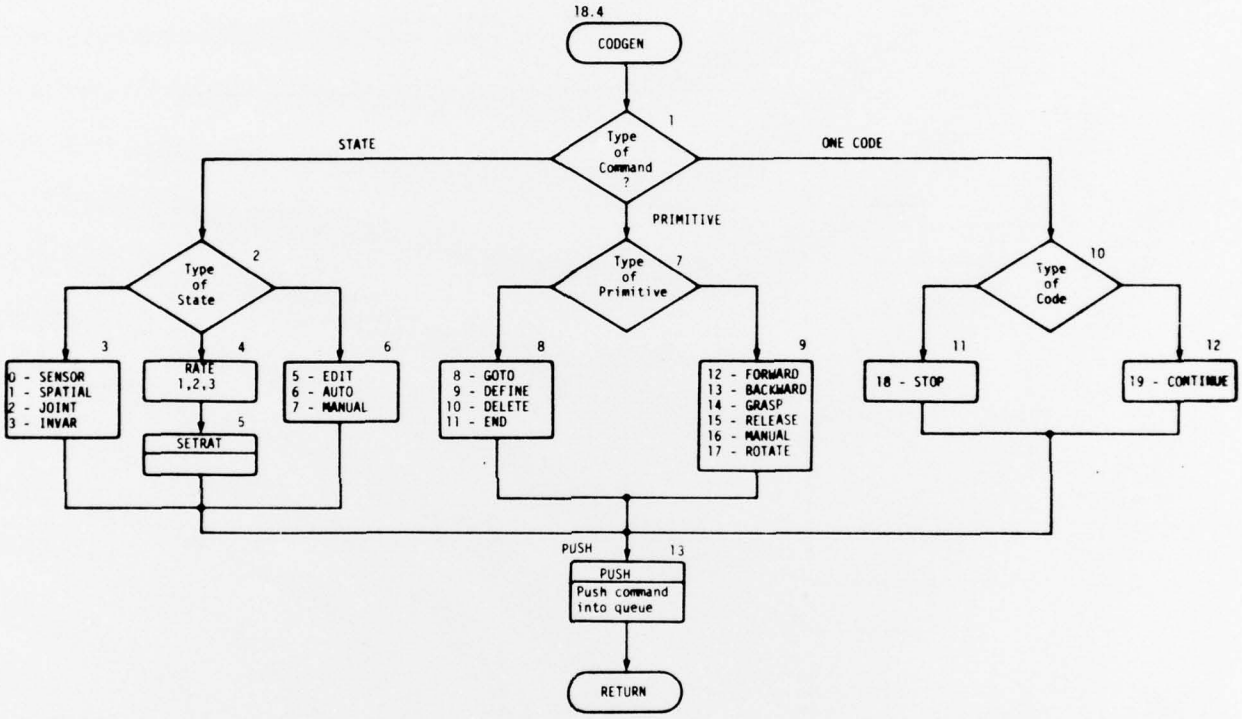
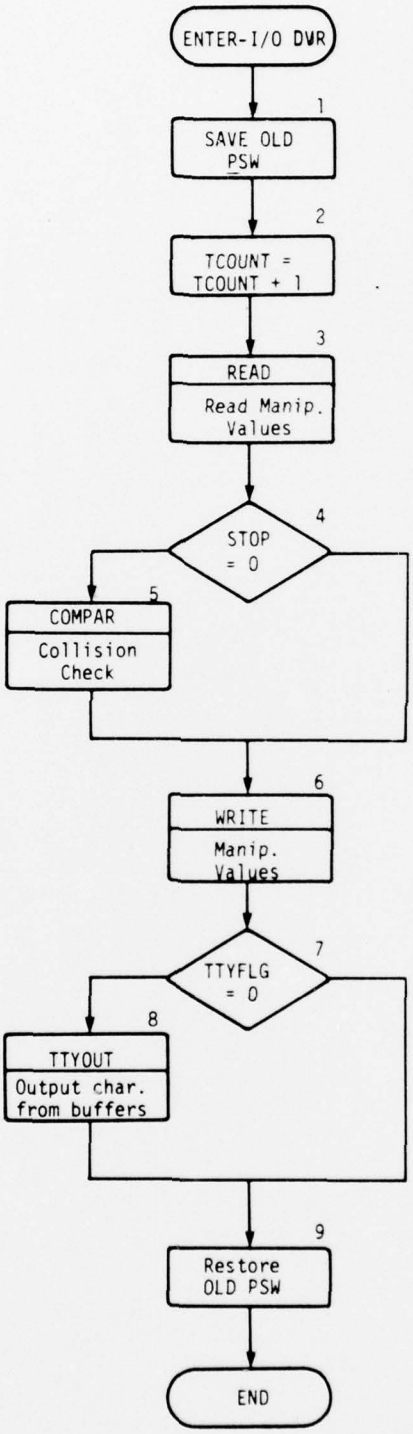

18.4
CODGEN
1
Type of Command ?
STATE
PRIMITIVE
ONE CODE
2
Type of State
7
Type of Primitive
10
Type of Code
3
0 - SENSOR
1 - SPATIAL
2 - JOINT
3 - INVAR
4
RATE 1,2,3
5
SETRAT
6
5 - EDIT
6 - AUTO
7 - MANUAL
8
8 - GOTO
9 - DEFINE
10 - DELETE
11 - END
9
12 - FORWARD
13 - BACKWARD
14 - GRASP
15 - RELEASE
16 - MANUAL
17 - ROTATE
11
18 - STOP
12
19 - CONTINUE
PUSH
13
PUSH
Push command into queue
RETURN

CHART 22


ENTER-I/O DVR
1
SAVE OLD PSW
2
TCOUNT = TCOUNT + 1
3
READ
Read Manip. Values
4
STOP = 0
5
COMPAR
Collision Check
6
WRITE
Manip. Values
7
TTYFLG = 0
8
TTYOUT
Output char. from buffers
9
Restore OLD PSW
END

5. ADDITIONS AND CHANGES

5.1 Hardware

The only hardware missing component is the end-gripper force and torque sensor. This sensor should prevent the arm from moving in directions where excessive forces are exerted. The software to handle the data is written but the development of the component is not completed yet.

5.2 Software

Development of many programs was postponed for the next phase. The calibration program - CALIB as well as the summary generating program - SUMMARY are not written yet.

TCACC - 3-D activation routine and TDMOM - 3-D monitoring routine are not written either.

6. REFERENCES

Tausworthe, C.R. Standardized Development of Computer Software - Part 1, Methods. JPL, Pasadena. pp. 107-108.

APPENDIX A

The LL1 grammer of the System Control language.

1.	<COMMAND>	stop	
2.	<COMMAND>	continue	
3.	<COMMAND>	sensor	<END-COM>
4.	<COMMAND>	R.M.C.	<END-COM>
5.	<COMMAND>	direct	<END-COM>
6.	<COMMAND>	invar	<END-COM>
7.	<COMMAND>	rate	<NUMBER3>
8.	<COMMAND>	skip	<END-COM>
9.	<COMMAND>	clear	<END-COM>
10.	<COMMAND>	initialize	<END-COM>
11.	<COMMAND>	goto	<NOUN-GO>
12.	<COMMAND>	define	<NOUN-DEF>
13.	<COMMAND>	delete	<NOUN-DEF>
14.	<COMMAND>	end	<NOUN=END>
15.	<COMMAND>	forward	<END-COM>
16.	<COMMAND>	backward	<END-COM>
17.	<COMMAND>	grasp	<END-COM>
18.	<COMMAND>	release	<END-COM>
19.	<COMMAND>	manual	<END-COM>
20.	<COMMAND>	rotate	<DIREC>
21.	<NOUN-GO>	point	<NUMBER 10>

22.	<NOUN-GO>	chain	< NUMBER 5 >
23.	<NOUN-GO>	path	< NUMBER 5 >
24.	<NOUN-GO>	reverse path	< NUMBER 5 >
25.	<NOUN-DEF>	point	< NUMBER 10 >
26.	<NOUN-DEF>	path	< NUMBER 5 >
27.	<NOUN-DEF>	chain	< NUMBER 5>
28.	<NOUN-END>	end	< END-COM >
29.	<DIREC>	left	< END-COM >
30.	<DIREC>	right	< END-COM >
31.	<NUMBER 3>	{1, 2, 3}	< END-COM >
32.	<NUMBER 5>	{0,1,2,3,4}	< END-COM >
33.	<NUMBER 10>	{0,1,2,3,4,5,6,7,8,9}	< END-COM >
34.	<END-COM>	do	
35.	<END-COM>	do now	
36.	<END-COM>	cancel	

Mr. Phillip Andrews
Naval Sea Systems Command
NAVSEA 0341
Washington, D.C. 20362

Dr. Fred Muckler
Navy Personnel Research and
Development Center
Manned Systems Design, Code 311
San Diego, CA 92152

CDR P.M. Curran
Human Factors Engineering Branch
Crew Systems Department, Code 4021
Naval Air Development Center
Johnsville
Warminster, PA 18950

LCDR William Moroney
Human Factors Engineering Branch
Code 1226
Pacific Missile Test Center
Point Mugu, CA 93042

Dr. John Silva
Man-System Interaction Division
Code 823, Naval Ocean Systems Center
San Diego, CA 92152

Mr. John Quirk
Naval Coastal Systems Laboratory
Code 712
Panama City, FL 32401

Human Factors Department
Code N215
Naval Training Equipment Center
Orlando, FL 32813

Dr. Gary Poock
Operations Research Department
Naval Postgraduate School
Monterey, CA 93940

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Mr. J. Barber
Headquarters, Department of the
Army, DAPE-PBR
Washington, D.C. 20546

Technical Director
U.S. Army Human Engineering Labs
Aberdeen Proving Ground
Aberdeen, MD 21005

U.S. Air Force Office of Scientific
Research
Life Sciences Directorate, NL
Bolling Air Force Base
Washington, D.C. 20332

Lt. Col. Joseph A. Birt
Human Engineering Division
Aerospace Medical Research Laboratory
Wright Patterson AFB, OH 45433

Dr. W. S. Vaughan
Oceanautics, Inc.
422 6th Street
Annapolis, MD 21403

Dr. Meredith P. Crawford
Department of Engineering Administration
George Washington University
Suite 805
2101 L Street
Washington, D.C. 20037

Dr. Ross L. Pepper
Naval Ocean Systems Center
Hawaii Laboratory
P.O. BOX 997
Kailua, Hawaii 96734

LCDR T. Berghage
Naval Medical Research Institute
Behavioral Sciences Department
Bethesda, MD 20014

Dr. R. Bornmann
Naval Medical Research and
Development Command
National Naval Medical Center
Bethesda, MD 20014

DISTRIBUTION LIST

Director, Engineering Psychology
Programs, Code 455
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217 (5cys)

Defense Documentation Center
Cameron Station
Alexandria, VA 22314 (12cys)

Dr. Stephen J. Andriole
Cybernetics Technology Office
Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

Col Henry L. Taylor, USAF
OAD (E&LS) ODDR&E
Pentagon, Room 3D129
Washington, D.C. 20301

Director, Electromagentics Technology
Programs, Code 221
Office of Naval Research
800 N. Quincy St.
Arlington, VA 22217

Director, Information Systems
Program, Code 437
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Program Director, Ocean Technology
Naval Ocean Research & Development
(NORDA)
Bay St. Louis, Mississippi 39529

Commanding Officer
ONR Branch Office
ATTN: Dr. E. Gloye
1030 East Green Street
Pasadena, CA 91106

Director, Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375

Office of the Chief of Naval
Operations, OP987H
Personnel Logistics Plans
Department of the Navy
Washington, D.C. 20350

Dr. Andreas B. Rechnitzer
Naval Postgraduate School
Ocean Engineering
Monterey, CA 93940

Mr. Arnold Rubinstein
Naval Material Command
NAVMAT 08T24
Department of the Navy
Washington, D.C. 20360

Commander
Naval Air Systems Command
Human Factors Programs, AIR 340F
Washington, D.C. 20361

Commander, Naval Facilities
Engineering Command
R&D Plans & Programs Division
Code 031A
Alexandria, VA 22332

Director
Behavioral Sciences Department
Naval Medical Research Institute
Bethesda, MD 20014

Dr. George Moeller
Human Factors Engineering Branch
Submarine Medical Research Laboratory
Naval Submarine Base
Groton, CT 06340

Mr. W. Greenert
Naval Material Command
NAVMAT 034
Hoffman II Building
200 Stovall Street
Alexandria, VA 22332

Mr. Paul Heckman
Naval Ocean Systems Center
San Diego, CA 92152

Dr. J. Miller
National Oceanic and Atmospheric
Administration
11400 Rockville Pike
Rockville, MD 20852

Dr. W. Mehuron
Office of the Chief of Naval
Operations
OP 009T
Washington, D.C. 20350

Mr. D. Popma
National Aeronautics and
Space Administration
Washington, D.C. 20546

Mr. H. Talkington
Ocean Engineering Department
Naval Ocean Systems Center
San Diego, CA 92152

Dr. T. B. Sheridan
Department of Mechanical
Engineering
Massachusetts Institute of
Technology
Cambridge, MA 02139

Dr. W. L. Verplank
Department of Mechanical
Engineering
Massachusetts Institute of
Technology
Cambridge, MA 02139

Mr. R. Wernli
Naval Ocean Systems Center
Ocean Technology Department
San Diego, CA 92152

Mr. J. Williams
Department of Environmental
Sciences
U.S. Naval Academy
Annapolis, MD 21402