This Research Was Sponsored by
The Space Nuclear Propulsion Office - NASA/AEC
Under Grant NsG - 728

A COMPUTER-AIDED CONTROL TECHNIQUE FOR A REMOTE MANIPULATOR

1-67-44

by

Jon Terry Beckett

Dr. H. W. Mergler
Professor of Engineering
Principal Investigator
Grant NsG-728

1967

ABSTRACT

A technique for controlling a remote manipulator is investigated which utilizes a general purpose digital computer to assist the operator perform complex tasks with a minimum of information. An experimental computer-controlled manipulation system is described and several path and positioning control algorithms are presented. One position control algorithm minimizes the manipulator transit time and automatically directs the manipulator around all predefined obstacles. An executive program allows the operator to control the manipulator in the manual mode, to utilize a semi-automatic mode in which commands are entered through a Teletype console, or to transfer control to an automatic mode executive routine which reads a series of commands from paper tape and automatically executes them.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

Chapter I

INTRODUCTION

Remote manipulators have been designed to enable man to perform tasks in a hazardous or hostile environment. Obviously, such devices are indespensible in the handling of radioactive materials. In fact, the hazards of direct exposure to radiation are responsible for most of the development of remote manipulation devices to date. Currently, however, there is a growing interest in using manipulators for both suboceanic and extra-terrestrial research activities.

It is desirable to apply recent technical innovations to improve the design and control of these devices to make them more suitable for these new activities. The mechanical design of such a device is partially dependent upon the particular application and is also a much more straight-forward problem than the development of techniques which can be used to control it. As a result, this report primarily considers the control problem and the development of a technique which can be applied to future systems.

Manipulators currently available are generally manually controlled and are classified as either "Rate-Controlled Manipulators" or as "Master Slave Manipulators" depending upon the control scheme employed.

The master-slave type of manipulator requires a "master" unit which is controlled by the operator and which is similar in shape to the manipulator (termed the "slave"). The two units are mechanically, electrically, or hydraulically linked so that the slave follows the motions of the master. Generally this link is also used to provide force feedback in which load forces at the manipulator are reflected back to the master.

The rate-controlled technique utilizes a control unit which allows the rate of each axis of the manipulator to be independently controlled. The control unit generally contains one lever for each axis and the rate of motion of the axis is proportional to the displacement of the lever.

Even though rate-controlled manipulators are generally capable of handling heavier loads and working in a larger area than master-slave manipulators, they move at a slow rate and the necessity of independently controlling the rates of each axis makes it difficult to perform path control operations (see Section 3.3.3).

Master-slave manipulators allow path control operations to be performed with relative ease; however, the presence of force feedback can cause operator fatigue, they are awkward to use if the manipulator moves at a slow rate, and a large working area is required by the operator to manipulate the master unit.

In contrast to the manual control scheme, several manipulation devices such as the Unimate* allow fully automatic control. The manipulator motions are recorded while manually performing the task and the motions can be repeated as often as desired by using the automatic mode. Although the devices are convenient for highly repetative operations, they are of limited value for general non-repetative operations which must frequently be performed by remote manipulators.

The Case Research Arm Aid[2] also provides both a manual mode and an automatic mode; however, using the automatic mode, the operator is allowed to select one of a wide variety of pre-recorded paths. Several problems encountered with this system were that it required a very large amount of incremental pulse data recorded on magnetic tape for each task, several seconds were consumed while searching for the selected task,

*Manufactured by Unimation, Inc., Bethel, Conn.

it was necessary to initiate and halt all operations at a pre-defined reference point, and it was necessary to reprogram a task to compensate for changes in the environment (e.g., if an obstacle were placed in the path).

A control technique is presented in this report in which a general purpose digital computer assists the operator in the execution of complex remote manipulation tasks. The computer's high computation speed and ability to make decisions rapidly coupled with its highly flexible internal and input-output structure enables the following features:

1. A reduction of the amount of conscious attention required of the operator to control the path or position of the manipulator.

2. Enhancement of the performance of positioning operations by optimizing the terminal configuration of the manipulator.

3. Improvement of the speed and accuracy of path control operations.

4. Minimization of the amount of data the operator is required to specify in order to perform a task.

5. The performance of special purpose computations such as coordinate transformation.

In addition, recent reductions in the price of small computers coupled with significant improvements in both speed

and reliability make the utilization of a computer economically feasible and highly practical.

In order to facilitate the study of the computer technique, to initiate steps leading to the formulation of a practical control scheme, and to comparatively evaluate the manual and computer control techniques, an experimental computer-controlled remote manipulation facility was developed. This system is described in Chapter II. Concurrently, a set of path and positioning control algorithms and a set of executive routines providing manual, semi-automatic, and fully automatic control modes were formulated. The resulting programs, described in Chapter III, demonstrate the merits of the computer-control technique which are enumerated above.

It is imperative to stress that it is normally impossible to completely specify in advance all operations which the manipulator will be required to perform; therefore, the primary function of the computer is to assist, not replace, the operator.

Chapter II

THE EXPERIMENTAL SYSTEM

2.1 Introduction

An experimental computer controlled remote manipulator has been developed to aid the investigation of various control techniques. Photographs of the system are shown in Figures 2-1 and 2-2. Information which is essential to the user and various highlights of the system are presented in this chapter. The details of the analog control section are presented by Hammond[3] and the report by Taylor[7] describes the manipulator, the computer interface, and the manipulator digital control section.

In order to establish a practical system, a General Mills* Model 100 manipulator was modified to resemble a PaR* Model 3000 manipulator. This device, shown in Figure 2-3, has nine degrees of freedom which are denoted as bridge traverse, carriage traverse, hoist, tube (or shoulder) rotation, shoulder pivot, elbow

*Manufactured or distributed by Programmed and Remote Systems Corporation (Pa R), St. Paul, Minnesota.

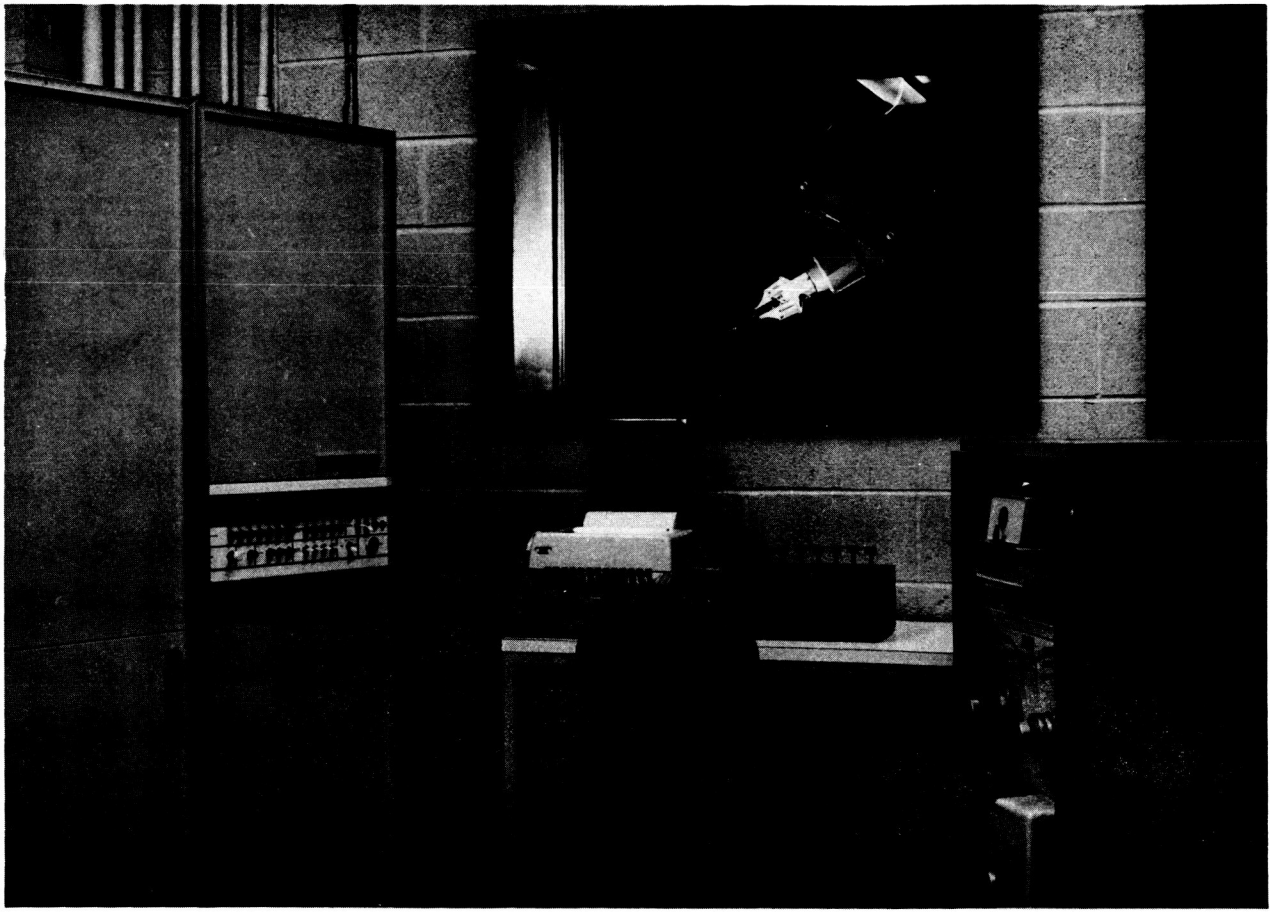


Figure 2-1. Photograph of Experimental System from Computer Room

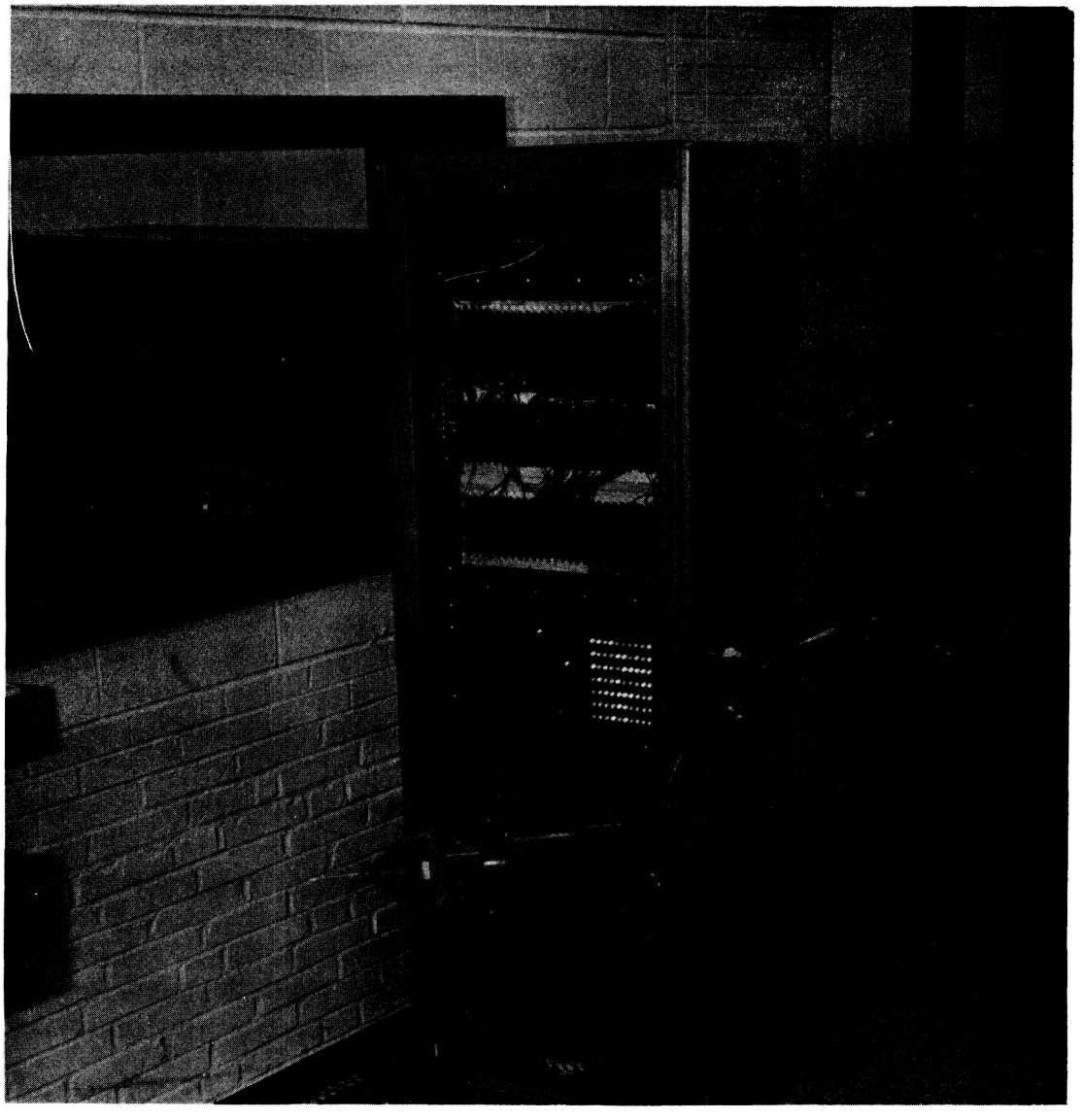


Figure 2-2. Photograph of Experimental System from Laboratory

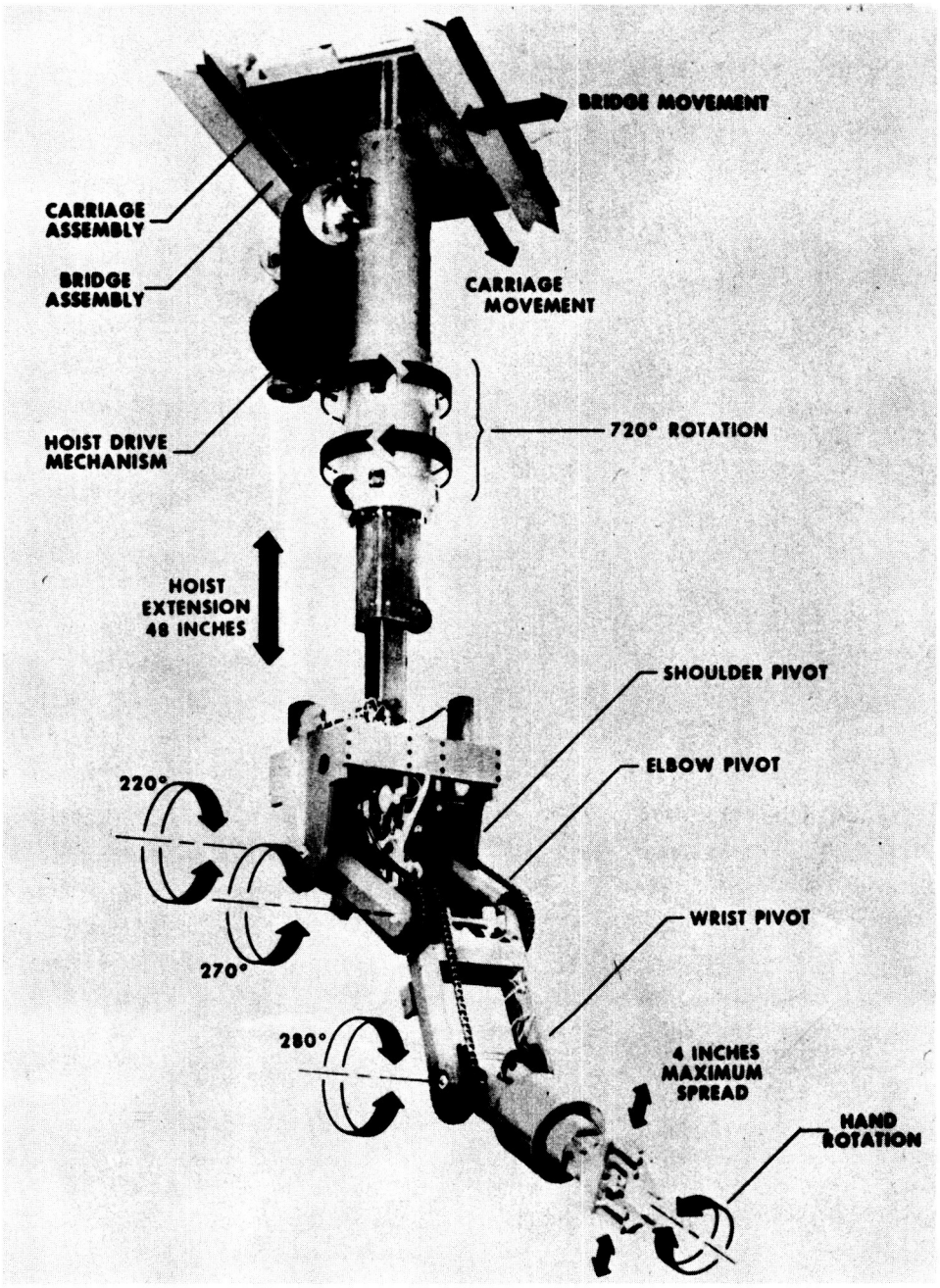


Figure 2-3. The Case Manipulator

pivot, wrist pivot, hand rotation, and hand grip. These will be abbreviated as X, Y, Z, SR, SP, EP, WP, HR, and HG in all future references.

Position feedback control loops have been added to the first seven of these axes. The last two retain the original relay controls.

A detailed block diagram showing all major sections of the system is given in Figure 2-4. Basically, the computer transfers position information to (or from) seven manipulator buffers via a General Purpose Interface Unit (abbreviated GPI) and the manipulator control logic. The digital position of a given axis is converted to an analog set point which is then compared with the signal from the potentiometer mounted on the corresponding manipulator axis. The difference, or "error", is used to drive the motor (using a SCR amplifier) until the set point is reached.

2.2 Mode Definitions

Various "modes" which are referred to in this and the following chapters are defined as follows:

INPUT MODE (of GPI): The computer is permitted to "input" information from the GPI data buffer

OUTPUT MODE (of GPI): The computer is permitted to "output" information to the GPI data buffer

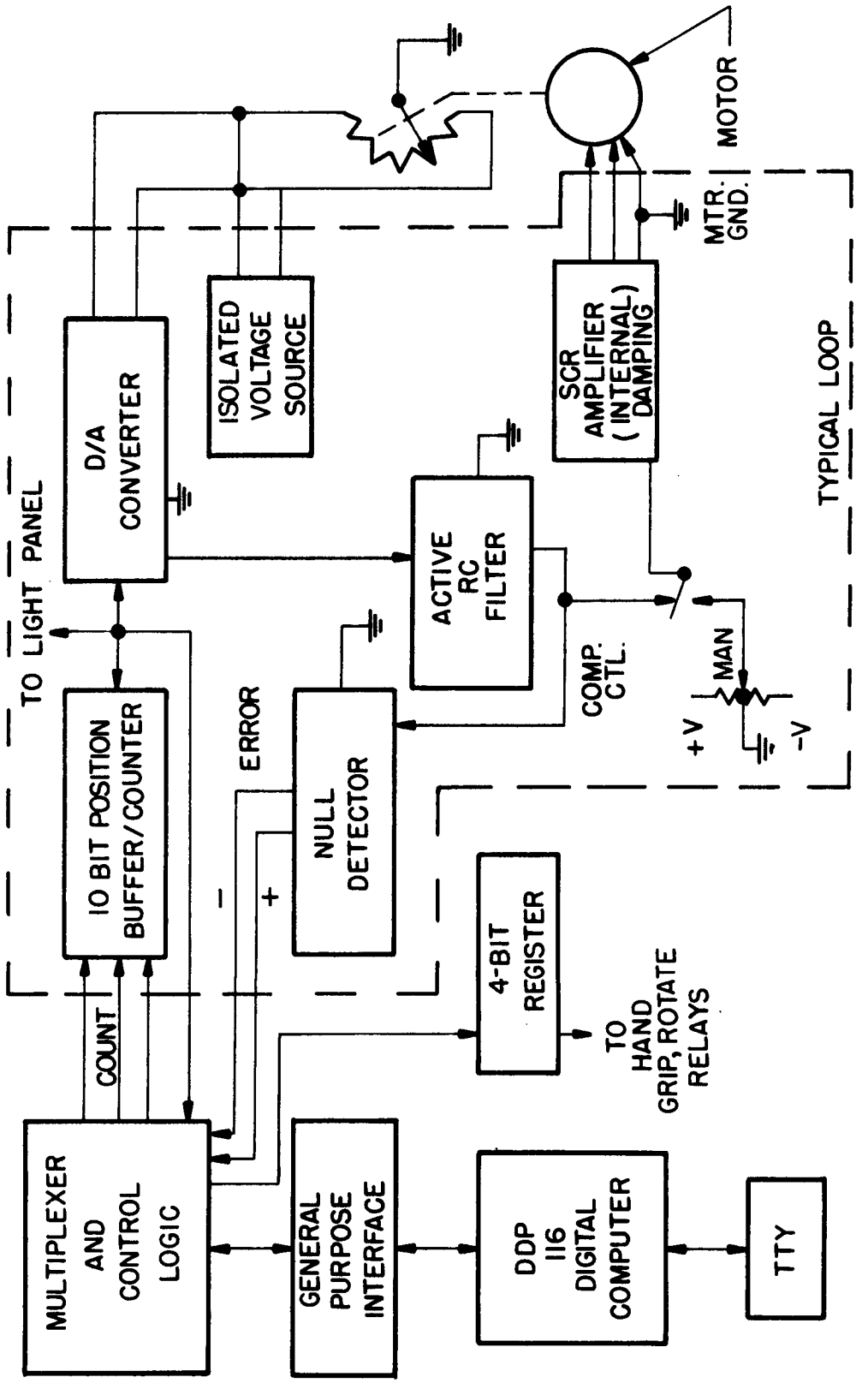


FIGURE 2-4. EXPERIMENTAL SYSTEM BLOCK DIAGRAM

MANUAL MODE (manipulator logic): The manipulator can be controlled exclusively via the manual control panel

COMPUTER-CONTROL MODE (manipulator logic): The manipulator servo-loops are enabled to drive all motors to the positions indicated by the buffers

SAD (software A/D): The clock used to increment or decrement the position buffers for hardware A/D conversion is inhibited

RATE-SIMULATED MODE: Rate control of the manipulator is simulated by the computer. The differences between the initial and final positions of all axes is divided into the same number of equal increments and these are added to the current positions at a fixed rate until the final positions are reached.

2.3 The Manipulator

In order to allow computer control and also make the General Mills manipulator resemble the Pa R Model 3000, it has been modified as follows:

1. An upper arm and forearm were added. The limbs are geared to the motors such that their absolute angles (measured relative to vertical) are controlled independently.

2. The relays used to control the X, Y, Z, EP, SP, WP, and SR motors were discarded and SCR amplifiers were designed to drive the motors.

3. The printed-circuit switches in the control panel used to activate the relays for the seven axes were discarded and replaced by potentiometers which allow continuously variable rate (and direction) control.

4. Potentiometers were mounted on the manipulator to measure the position of the seven axes listed above.

5. A repulsion motor which was used to control the hoist was replaced by a series-wound universal motor to allow the use of identical servo-loops for all seven axes.

The arm dimensions are shown in Figure 2-5 and a comparison between the Pa R and the experimental manipulator is given in Table 2-1.

2.4 Analog Control Section

2.4.1 Silicon Controlled Rectifier (SCR) Amplifier

The SCR amplifier was designed to drive a low-power series-wound universal motor, the speed and direction of which corresponds to the magnitude and sign of a DC signal. The signal controls the phase angle at which the SCR's fire. This in turn meters the amount of power supplied to the motor. In order to optimize the performance, the following features are incor - porated in the amplifier:

1. Four SCR's are used in conjunction with a 220 V, three wire AC source to drive the motors close to their rated value for maximum torque.

2. An adjustable "negative" dead-band has been added to compensate for the stiction of the motor. This is accomplished by adding a bias to the signal which

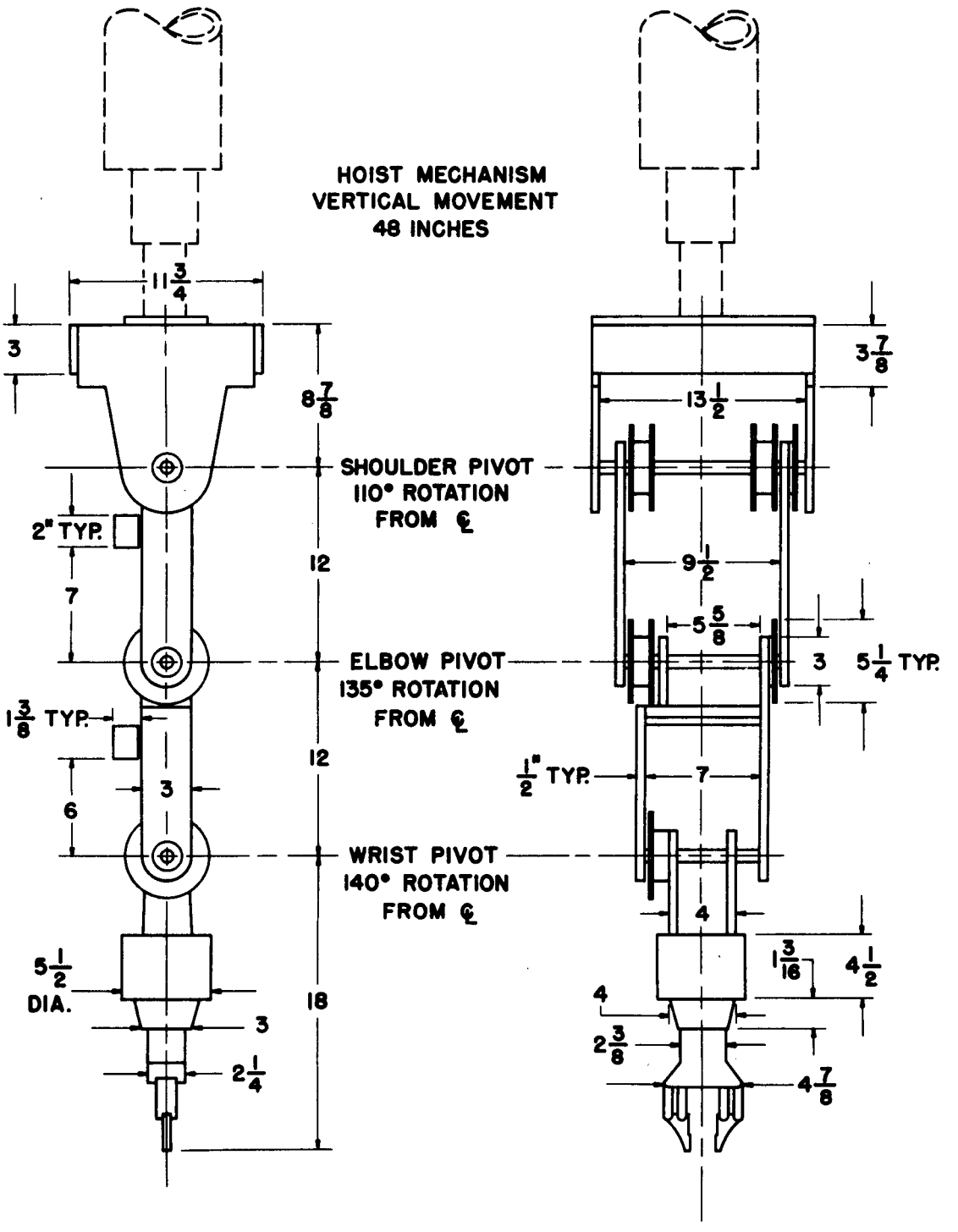


Figure 2-5. Manipulator Arm Dimensions

Table 2-1. Manipulator Specifications

Element	Case Manip.	PaR 3000
Bridge		
Travel	14.5 feet	These depend on the particular installation
Velocity	8 fpm	
Carriage		
Travel	12 feet	
Velocity	10 fpm	
Hoist		
Travel	4 feet	
Velocity	8 fpm	
Shoulder Rotation		
Travel	720°	Continuous
Velocity	5 rpm	1.5 rpm
Shoulder Pivot		
*Travel	±110°	±125°
Velocity	1.5 rpm	2.5 rpm
Elbow Pivot		
*Travel	±185°	±180°
Velocity	1.5 rpm	2.5 rpm
Wrist Pivot		
*Travel	±270°	±305°
Velocity	1.5 rpm	2.5 rpm
Hand Rotation		
Travel	Continuous	Continuous
Velocity	50 rpm	7 rpm
Grip		
Opening	4 in.	4 in.
Velocity	120 ipm	20 ipm

*Angles are measured from the center line.

produces enough power to almost overcome the stiction.. The synchronizing pulse (used to insure that all SCR's are turned off before the start of each cycle) is then extended to just delete the bias when the signal is zero. When the signal exceeds this small dead-band, the bias is added to the output signal.

3. The approximate EMF of the motor (produced by its rotation) is monitored at the amplifier and is used for first-derivative compensation to improve the response by reducing the over-shoot.

4. By mounting two diodes near the motor, it is possible to run only two wires to each motor (plus a ground wire common to all motors). The diodes are connected so that the current through the field winding always flows in the same direction. The direction of current in the armature thus controls the direction of rotation of the motor.

5. Limit switches were added to the pivots and shoulder rotation. They were mounted in series with the motor wires such that one prevents rotation in one direction and the other prevents rotation in the opposite direction. It made it possible to prohibit motion beyond the limits while allowing motion out of the limits without running separate wires back to the control logic.

Note: A 15 amp circuit breaker is used to protect the SCR's; however, it is necessary to stagger the turn-on and turn-off times of the motors because of the logic current surges which occur. In general, no more than one motor should be turned on or off each 17 m sec.

2.4.2 D/A Converter and Bridge Network

The D/A converter uses a "voltage ladder" circuit to convert the digital position to an equivalent analog signal. This is then compared with the signal from the potentiometer mounted on the manipulator by a bridge network. The D/A circuit and bridge network are shown in Figure 2-6. This arrangement was chosen because it eliminated the necessity of using an accurate reference voltage. Also, since no current flows at the null point, it is unnecessary to use a modulator and demodulator to prevent DC losses in the relatively long cables. Nevertheless, it is now necessary to use separate (but very simple) isolated voltage sources for each axis. Figure 2-7 presents a simplified diagram of the bridge arrangement. From this it is seen that the error output is zero only when the voltage between the side of the pot and the wiper, V_m equals V_{DA}, the voltage output of the D/A converter referenced to the same line which goes to the potentiometer.

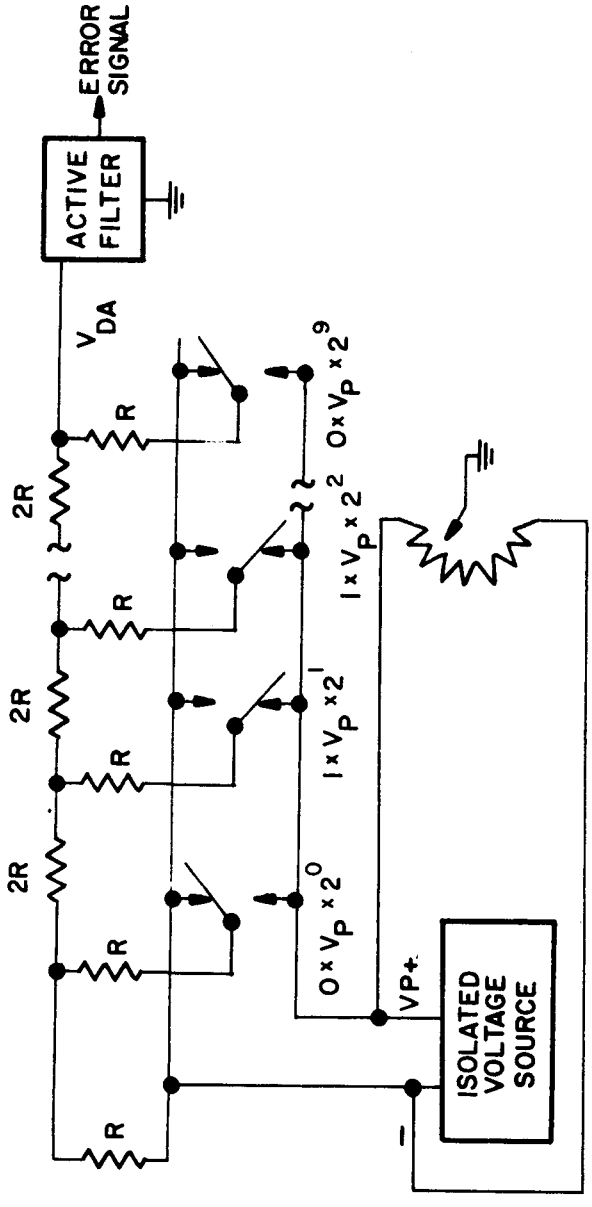


Figure 2-6. D/A Converter and Bridge Network

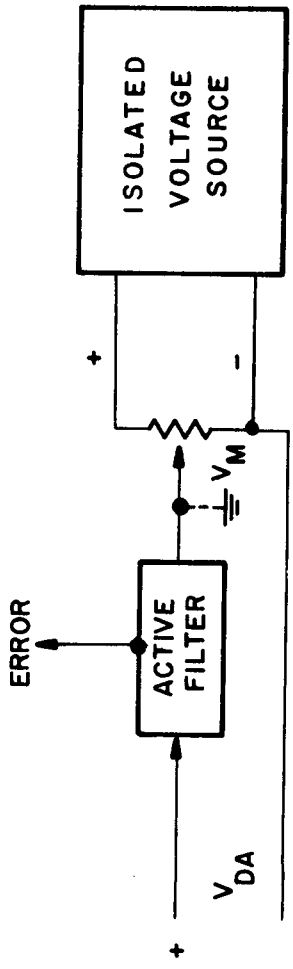


Figure 2-7. Simplified Bridge Network

2.4.3 Active Filter

The SCR amplifier is essentially a sampling amplifier. As a result, it is sensitive to synchronous noise -- particularly if the frequency is equal to or a harmonic of the sampling frequency. Unfortunately, this case is encountered in this system because the noise created by the firing of the SCR's can be picked up by the lines between the controls and the potentiometers. As a result, the filter was designed to attenuate signals above 20 cps. It also is used to amplify the error signal slightly for the SCR amplifier.

2.4.4 Null Detector

The null detector grounds either the "positive" or "negative" error lines when the output from the active filter exceeds a small "dead-zone". These signals are then used by the manipulator digital control logic.

2.5 The Computer

A DDP-116 low-cost digital computer manufactured by the Computer Control Division (3C) of Honeywell is used to control the manipulator. The memory contains 8,192 words, the word length is 16 bits, the memory cycle time is 1.72 μ sec and typical

instruction execution times are 3.4 μ sec. for addition, 9.5 μ sec. for multiplication, and 17 μ sec. for division (all fixed point). The Input-Output equipment available includes a magnetic tape transport, a line printer, a card reader, high-speed paper tape equipment and a teletypewriter. Only the last two units have been used during the execution of the manipulator control algorithms.

2.6 The General Purpose Interface Unit (GPI)

The GPI controls the flow of data and commands between the manipulator logic and the computer. It contains a 16 bit data buffer, control flip-flops and the decoding logic for five external output command pulses (OCP) and for six external status sense lines (SKS). Two of the control flip-flops are of prime concern to both the programmer and to the external device (i.e., the manipulator control logic). These indicate the "mode" of the GPI (INPUT or OUTPUT) and the status (READY or NOT READY). Data transfers between the computer register and the GPI data buffer are allowed only when READY is enabled (set). The status of READY is modified by the following operations:

READY SET -- a. By the computer on a OCP Ø 1 (set GPI OUTPUT mode).

b. By the manipulator logic when additional data transfers are anticipated.

READY RESET -- a. By the computer on a OCPØØ (set GPI INPUT mode)

b. By the computer after data is transferred by the computer to or from the GPI data buffer during the execution of an INA (input) or OTA (output) instruction.

Whenever READY is reset, it in effect signals the external device that the computer has transferred data to the GPI buffer and it is now available to the external device, or else that the external device can transfer data to the GPI buffer. As soon as the device is finished with the data or has filled the buffer, it must transmit a pulse to set READY.

2.7 The Manipulator Control Logic

The manipulator logic contains seven ten-bit buffers, one four-bit register, control logic for A/D conversion, and multiplexing logic to allow data transfers with the computer via the single GPI channel.

All sense (SKS) and command (OCP) instructions recognized by the GPI and the manipulator logic are listed in Table 2-2. In addition, the formats for all data transfers are shown in Figure 2-8.

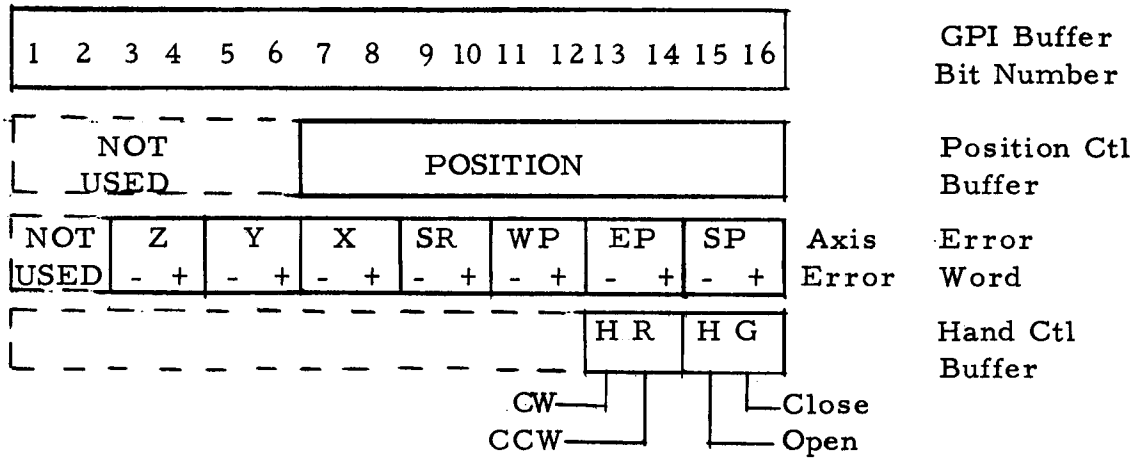
Table 2-2. OCP and SKS Instructions

OCPØØ	Enable GPI INPUT Mode, Reset READY Immediately force Manual Mode
OCPØ1	Enable GPI OUTPUT Mode, set READY Enable Computer Control Mode if no error (20 cps clock)
OCPØ2	Set SAD Mode Clear AXIS Counter, Reset START Set Computer Control Mode if GPI in OUTPUT Mode
OCPØ3	Enable DMC Mode
OCPØ4	Set AXIS Counter to one, Set START Caution: First clear AXIS counter, set Manual Mode or Computer Control Mode
OCPØ5	Reset SAD Mode
OCPØ6	If GPI in INPUT Mode, transfer ERROR STATUS to GPI data buffer If GPI in OUTPUT Mode, transfer GPI data to the four-bit register
OCPØ7	Reset DMC End of Transmission
SKSØØ	Sense READY
SKSØ1	Sense no error
SKSØ2	Sense Computer-Control Mode
SKSØ4	Sense not interrupting

Note:* G.P. INTERFACE ADDRESS = '40
MASK BIT = 8
DMC CHANNEL = 3

*See Programmer's Reference Manual[4] for explanation.

Word Formats



Data Transferred

Axis Ctr	(Input mode) to GPI Buff	(Input mode) from GPI Buff
0	Error	Hand Control
1	SP	SP
2	EP	EP
3	WP	WP
4	SR	SR
5	X	X
6	Y	Y
7	Z	Z

Figure 2-8. Manipulator - GPI Buffer Data Formats

2.7.1 Buffer Registers

One ten-bit buffer is used for each axis under closed loop control. These buffers form the main link between the digital and analog portions of the manipulator controls. In the automatic mode, the computer outputs the desired position of each axis to the buffers via the GPI and multiplexing logic. The analog controls convert this to an analog set-point, compare it with the position indicated by the manipulator potentiometer and use the resulting error signal to drive the motor until the error vanishes. The buffers are also used in the A/D conversion (see Section 2.7.2).

The two motors which control the hand grip and rotation are actuated by a set of relays. Two relays for each motor control the direction. The speed is fixed. If both relays controlling a motor are set, no motion is allowed.

In the Computer Control mode, the appropriate relay is energized whenever a flip-flop in the four bit buffer register is set. The flip flops are set or reset by an OCP Ø6 command when the AXIS counter is zero and the Computer Control mode is enabled. The register is always reset by a DC level whenever the Manual mode is enabled.

2.7.2 Position Measurement (A/D Conversion)

The potentiometers which are connected to the manipulator axes for position feedback can also be used to measure the current positions of the axes. Two techniques have been implemented to perform the measurement. Basically the error signal which indicates the sign of any difference between the axis buffer and the position is used to modify the buffer contents instead of driving the manipulator. The difference between the two techniques is the method used to modify the buffer contents.

The first method demands that the buffers also function as bidirectional counters. Whenever the control logic is in the manual mode, the buffer contents are incremented when the error is negative and are decremented when the error is positive. Only when the error is within the dead band of the null detector does counting cease.

The second technique requires the use of a computer program to perform the conversion. In this case, the buffer contents are modified strictly by data transfers from the computer. In order to perform the "software" conversion, the computer must be permitted to examine the error status of each axis. An algorithm, such as that described in Section 3.5 , first outputs a set of trial values

to the buffers. The resulting error status of each axis is examined and is used to output a second trial. The process is repeated until the error of all axes reaches zero. A conventional binary search technique requires only one guess for each bit in the position word; therefore, a maximum of ten trials are necessary in the experimental system.

Both methods have been implemented in the experimental system. In order to use the software technique it is necessary to enable the SAD mode to prevent counting the buffer.

The conversion speed in the experimental system is limited because of a lag introduced by the active filter. The attenuation beyond 20 cps results in a delay of approximately 50 m sec. after each output before the error signal is meaningful.

The hardward technique is of value because of its ability to "track" the position of each axis as it is driven under manual control; however, for large errors, up to 20 seconds can be consumed before a null is reached. Conversely, the software mode requires a maximum of ten trials; thus, the conversion speed of approximately 0.5 sec is independent of the magnitude of the error.

The tracking mode has been convenient to use in the

experimental system, however, the time consumed when the manipulator is first turned on and when the operator interupts the Computer Control mode becomes annoying. As a result, the SAD mode appears to be most practical because of savings in digital logic and the fixed conversion time. With improvement in the conversion speed, it will become even more appealing.

2.7.3 Multiplexing Technique

All data transmitted between the computer and the manipulator buffers is multiplexed and passes through a single computer I/O unit -- the General Purpose Interface (GPI). A three-bit counter is used to gate the information between the GPI buffer and each of the buffers in the control logic.

Consider a typical operation in which the computer outputs information to the seven position buffers. The computer must first output an OCP Ø 2 to clear the axis counter and initialize the multiplex logic; next an OCP Ø 1 is used to set the output mode in the GPI; and then an OCP Ø 4 sets the axis counter to "1". When the computer outputs the first word, this is automatically gated into buffer "1" and then the axis counter is incremented. The process continues until finally the seventh word is transferred. It is gated

into buffer "7", the axis counter is reset, and the multiplexing logic is conditioned to ignore all further position information until an OCP Ø 4 is executed again.

The data always appears in a fixed sequence. Currently, the order is SP, EP, WP, SR, X, Y, and Z as shown in Figure 2-8. Although it is possible to transfer a partial set of information, only the data at the end can be ignored. For example, to change the value of SR, it is necessary to output values of SP, EP, and WP as well as SR but the X, Y and Z values can be ignored. For such a partial transfer, it is mandatory that an OCP Ø2 be executed as described in Section 2.8.

Seven states (1-7) of the axis counter are used to control data transfers with the position buffers and the remaining state is used to output data transfers to the four bit buffer and to input the error status of each axis depending upon the mode of the GPI.

The general sequence of operations required to input and output information is shown in Figures 2-9 and 2-10.

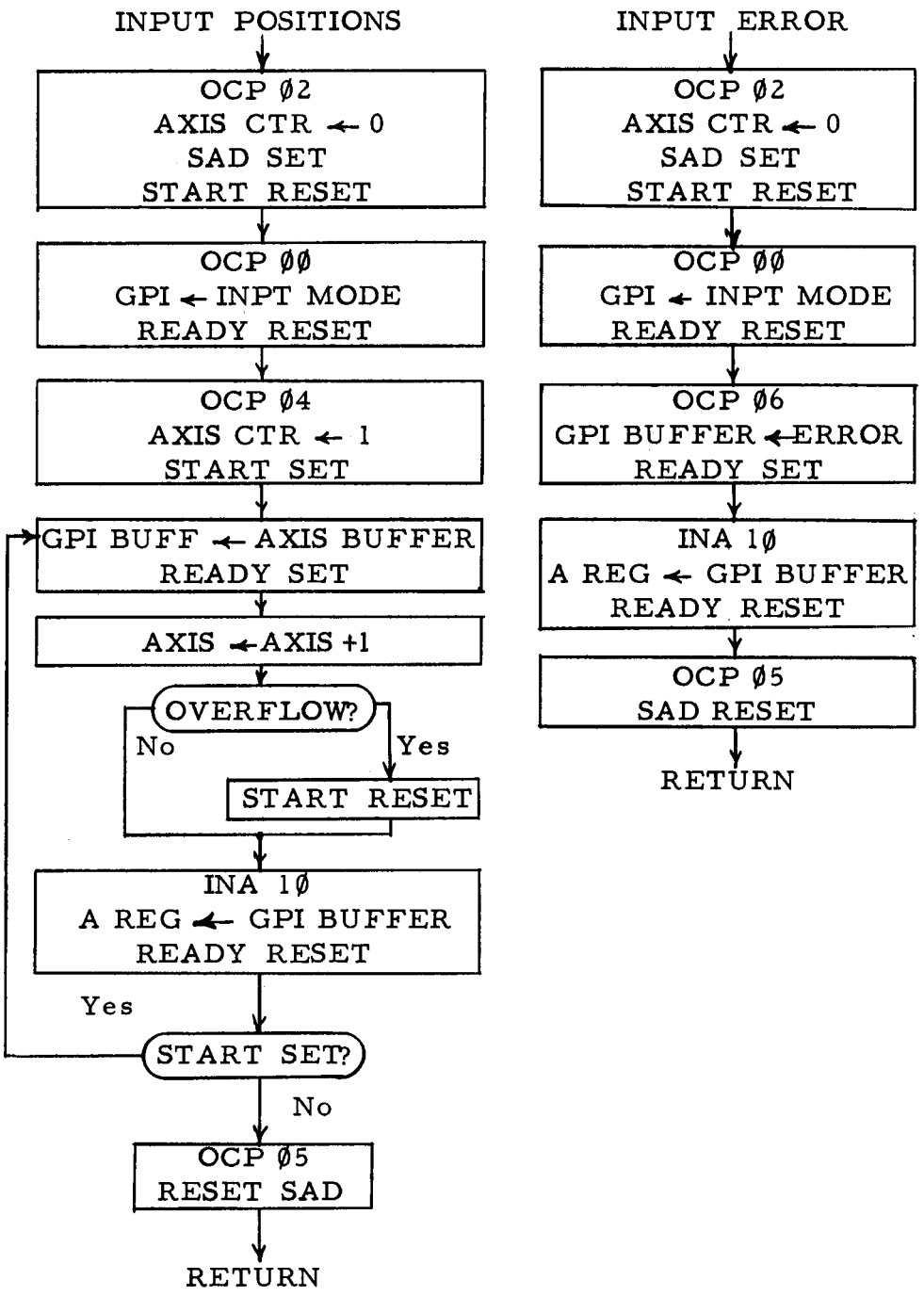


Figure 2-9. Typical Information Flow -- Manipulator to Computer

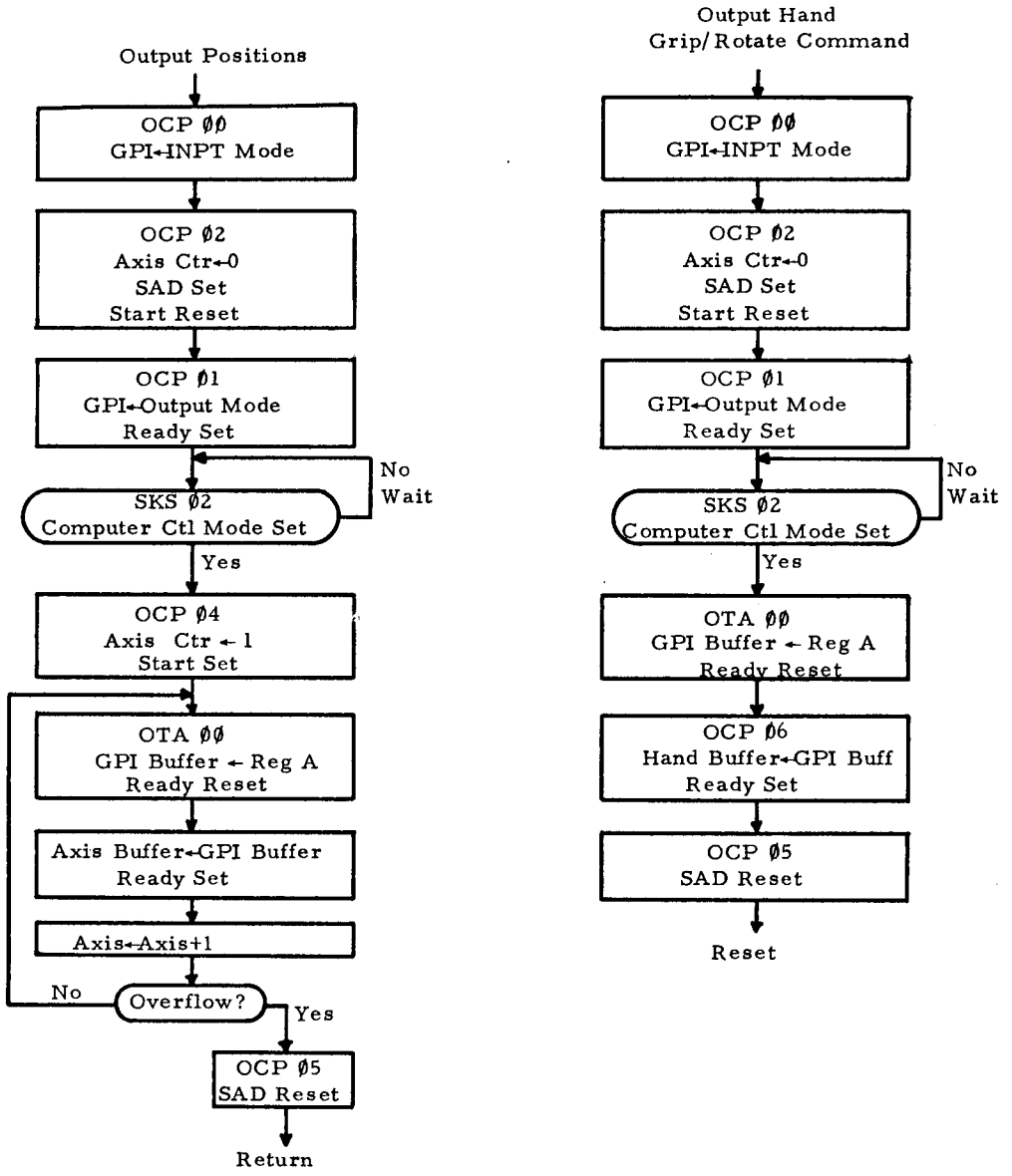


Figure 2-10. Typical Flow of Information
From Computer to Manipulator Logic

2.8 Programming Notes

The SKS and OCP instructions utilized by the manipulator logic and interface unit are listed in Table 2-2. Due to the limited number of external OCP commands available, the OCP Ø2 instruction performs several functions. As a result, it is occasionally necessary to follow or precede this command with one or two instructions to disable any undesirable modes. For example, to clear the AXIS counter, it may be necessary to execute an OCP ØØ first, then the OCP Ø2, and finally an OCP Ø5. The first instruction insures that the Computer Control mode will not be forced, the second instruction clears the AXIS counter and sets SAD, and the third instruction resets the SAD mode.

Several side effects arise from the multiplexing technique and the programmer must exercise caution to prevent any damage to the system and to insure that the proper information is transferred between the computer and the manipulator control logic. In general, the following rules should be observed:

1. After an INA or OTA instruction, wait at least ten μ sec before using an OCP instruction.

2. Before changing modes in the GPI, execute an OCP Ø2 to clear the AXIS counter.

3. After executing an OCP Ø1 (GPI Output mode), wait until the Computer Control mode is established (test with an SKS Ø2) before executing an OCP Ø4 to initialize the counter. Note that the Computer Control Mode is set by a 20 cps clock.

The side effects which will occur can be predicted from a brief consideration of the multiplexing technique. A "START" flip-flop in the multiplexing logic is set when the AXIS counter is initialized (OCP Ø4) and is reset when the counter overflows or by an OCP Ø2. If the GPI READY flip-flop is reset and START is set, the following operations will occur on the next clock pulse (200 KC clock):

1. Leading edge of clock pulse
   a. Data is transferred to (from) manipulator buffer (selected by current value of AXIS counter) from (to) GPI data buffer
   b. READY is set.
2. After a slight delay, the axis counter is incremented by one (if value was seven, it is cleared and START is reset).

The above operations require up to ten μ sec and the sequence occurs after all INA and OTA instructions (unless the AXIS counter is zero). It is imperative to wait for their completion before executing any instructions which change the mode or the axis counter contents (OCP ØØ, OCP Ø1, OCP Ø2, OCP Ø4).

Two examples of problems which will occur if the above rules are not observed are shown in Figure 2-11.

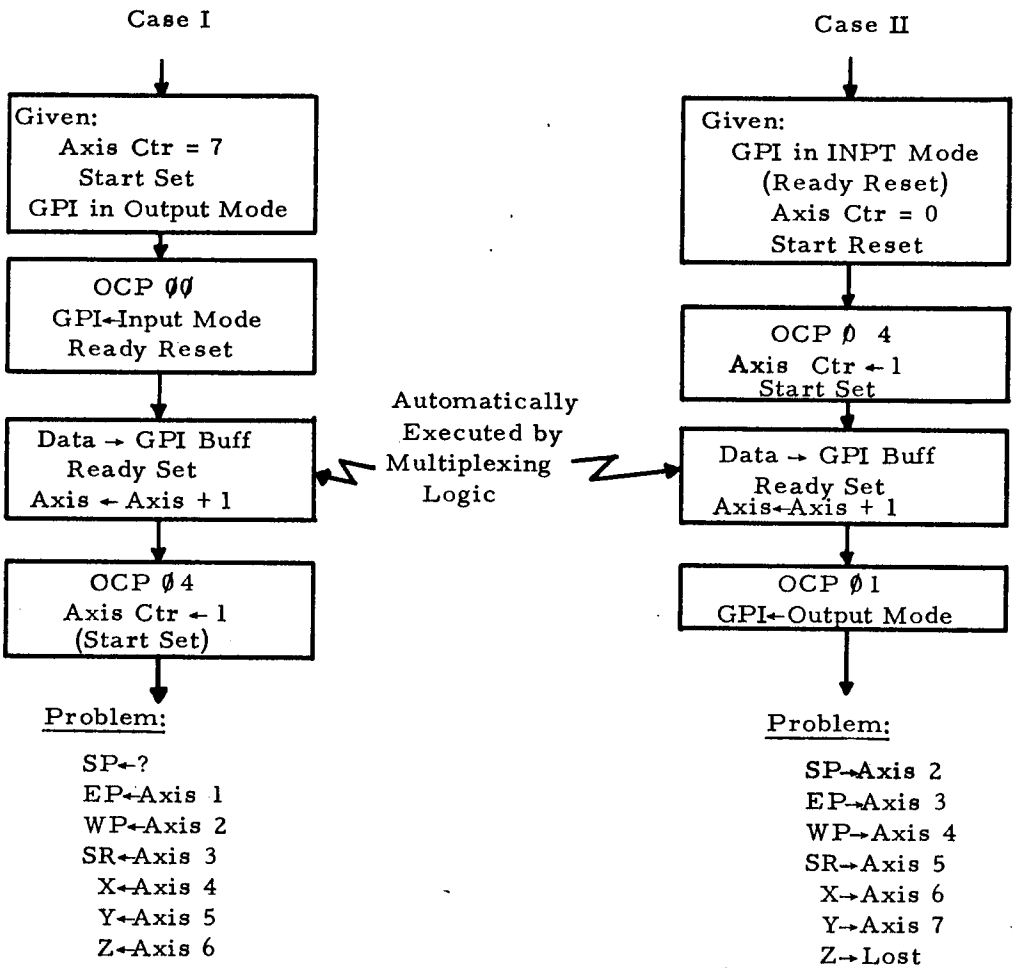


Figure 2-11. Two Programming Hazards

Chapter III

THE MANIPULATOR CONTROL ALGORITHMS

3.1 Introduction

Several position and path control algorithms have been developed and are presented in this chapter. These algorithms have been implemented in the experimental system, and they demonstrate the ability of a computer to effectively control a manipulator by only a few strategic input commands together with a minimum of accompanying data.

Several special programs have also been written to execute miscellaneous functions such as typing the contents of the seven manipulator buffers, performing software analog to digital conversion, and enabling various modes to facilitate system maintenance.

System versatility is insured by an executive routine which allows the operator to select any of the control routines or special functions at random via the teletype console. In addition, a second executive routine allows pre-recorded operations to be executed automatically.

The coordinate system used in the position and path control algorithms is defined in Appendix I.

All subroutines available are listed in Appendix III. The flow diagrams and program listings appear in a separate document[1].

3.2.1 Teletype Executive Control Program

The "Teletype Executive Control Program" (abbreviated TTY Exec) provides the basic link between the computer and the operator. All subroutines are initially loaded into the memory of the computer and program execution is started in the TTY Exec. This program waits for the operator to specify the operation desired. Upon receipt of a command, control is transferred to the appropriate subroutine. Generally, the subroutine accepts additional required information and executes the command. If one of the basic positioning or path control algorithms is specified, however, a subroutine is first called which accepts all additional data required by the algorithm and stores it in the appropriate buffers. Then control is transferred to another subroutine which actually executes the command. This feature is included to allow control from both the TTY Exec and the AUTO Exec (described in the next section).

The following two examples illustrate the selection technique:

Example I: Instruct the computer to type the seven current manipulator variables (scaled).

1. The operator first types a "T" -- the TTY Exec transfers control to the TYPE subroutine.

2. The operator types a "5" -- this instructs the TYPE subroutine to type the current manipulator variables.

3. The TYPE routine then inputs the current manipulator variables, scales them into inches and degrees to conform to the coordinate system, and types the resulting values.

Example II: To direct the manipulator 7.5 inches in a straight line along a vector formed by the hand orientation.

1. The operator types a "V" -- the TTY Exec first calls the "IDIST" subroutine.

2. The IDIST routine types DIST: then waits for the operator to specify the distance (in inches).

3. The operator types 7.5 followed by a carriage return.

4. The IDIST routine stores this in the DIST memory buffer.

5. The TTY Exec calls the VECTOR subroutine which outputs a series of increments to drive the manipulator the specified distance.

Figure 3-1 presents a list of the major functions controlled by the TTY Exec. In addition, Appendix II contains a flow diagram of the TTY Exec, an exhaustive list of all options available, and a list of all additional information required by each option.

Another important feature of the TTY Exec is that it permits the operator to interrupt the execution of any subroutine in case of an emergency. After a command is accepted by the TTY Exec, but before transferring control to the proper subroutine, the hardware interrupt feature of the computer is conditioned to immediately transfer program execution to the TTY Exec if the operator depresses any key on the console. If this occurs, the GPI Input Mode is enabled, further interrupts are inhibited, and the TTY Exec awaits the next command. All manipulator motion ceases when the GPI Input Mode forces the Manual Mode in the manipulator control logic. In order to allow additional information to be requested and accepted, the teletype input and output subroutines temporarily suspend TTY interrupts; however, in case erroneous information was typed, the interrupt feature can be simulated by setting Sense Switch four on the computer console.

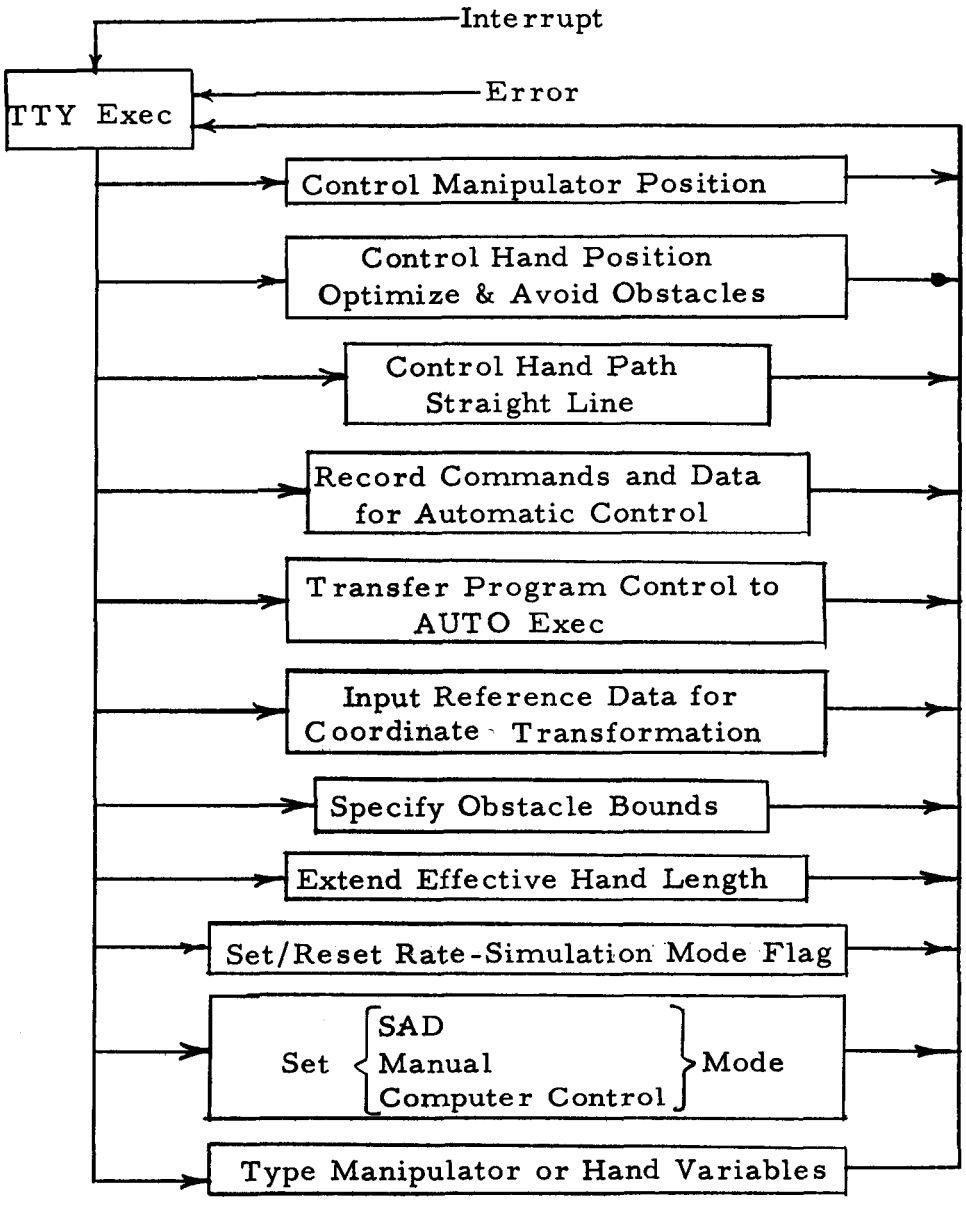


Figure 3-1. TTY Exec Options

3.2.2 Automatic Mode Executive Program

The operation of the Automatic Mode Executive Program (abbreviated AUTO Exec) is quite similar to that of the TTY Exec. In fact, the primary difference is that the AUTO Exec reads the basic commands and accompanying information from paper tape rather than accepting data directly from the operator via the Teletype. The control algorithms available via the AUTO Exec are listed in Figure 3-2. This figure also indicates how the program is called from the TTY Exec. Once called, it retains control until one of the five following conditions occurs:

1. The "TTY Return" Code is encountered.
2. An "ORIGIN" Code is encountered (see Sect. 3.4.3).
3. The operator forces a program interrupt.
4. An error condition is encountered during execution of the algorithms.
5. Sense Switch three on the computer console is set.

In order to use the AUTO Exec, it is necessary to prepare the paper tape in advance as described in Section 3.4.1. The tape is punched in blocks consisting of a flag which specifies the control algorithm and all data required by the algorithm.

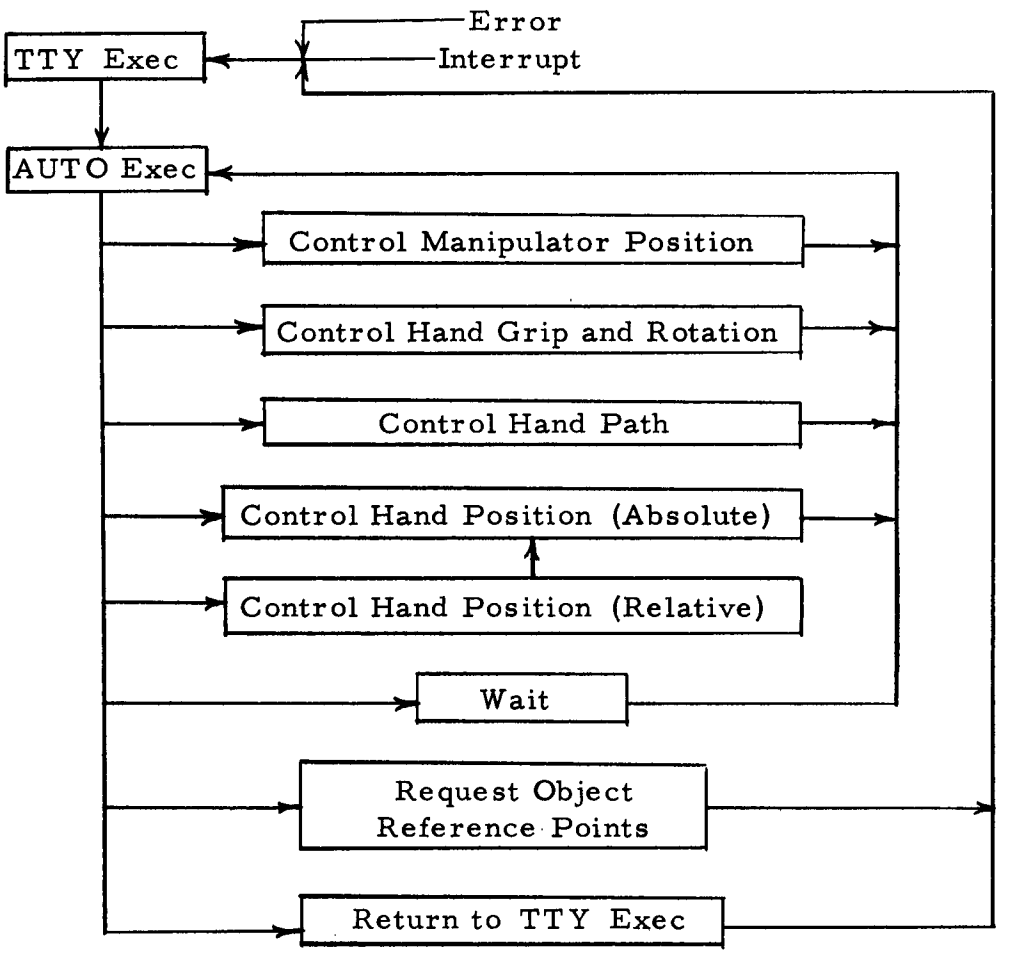


Figure 3-2. AUTO Exec Options

When the AUTO Exec is called from the TTY Exec, it reads a block of data from the tape. The command is decoded, the information is stored in the appropriate buffers, and the specified subroutine is executed. All subsequent blocks of data are similarly processed without operator intervention until one of the conditions listed above relinquishes control to the TTY Exec.

3.3 Control Algorithms

3.3.1. Manipulator Position Control

The most basic control algorithm allows the direct application and control of the X, Y, Z, SP, EP, WP, and SR positions. The desired positions are stored (in scaled form) in the FMV (final manipulator variables) buffer. The "FBUF" subroutine converts the scaled values to ten-bit manipulator position values and stores them in an output buffer (OBUF). The "OMANIP" subroutine then outputs the seven variables from the OBUF buffer to the manipulator buffers, allowing about 17 msec between each output to prevent turning all motors on simultaneously. The "ETST" subroutine then tests the error status to determine when the final positions have been reached by all axes.

The AUTO Exec routine fills the information from tape into

the FMV buffer and then calls these three subroutines. When the error reaches zero, it automatically reads and processes the next block.

The TTY Exec calls the "ITMNIP" to input the terminal manipulator position and fill the FMV buffer. It then calls the three control subroutines listed above to output the buffer contents.

In addition, the three control subroutines are used by the VECTOR and Hand Position Control subroutines described in the following sections.

The flow diagram for the ITMNIP subroutine appears in Figure 3-3. The operator first selects either an absolute or an incremental mode in which all subsequent position data will represent either the absolute position or an increment relative to the present position. The operator then specifies one or more axes followed by the desired position or change in position. A special command terminates the list and all unspecified axes will retain their current values.

Unless the operator desires to control the exact position of the manipulator, this algorithm is of limited value via the TTY Exec because of the time required for the operator to specify the

desired variables in comparison with the time required to control them directly with the manual controls. Obviously, the algorithm's value is much greater when used in conjunction with the more complex positioning and path control algorithms and also when used by the AUTO Exec.

3.3.2 Hand Grip and Rotate Control

This hand grip and rotation control routine demonstrates the method by which the two "open-loop" variables are controlled by the computer. The grip and rotate commands are outputed to the manipulator logic as described in Section 2.7.1. After the time specified by the operator has elapsed, the manipulator is forced into the Manual mode, thereby halting all motion.

The teletype input routine, "IGRIP," is shown in Figure 3-4 and the control routine appears in Figure 3-5.

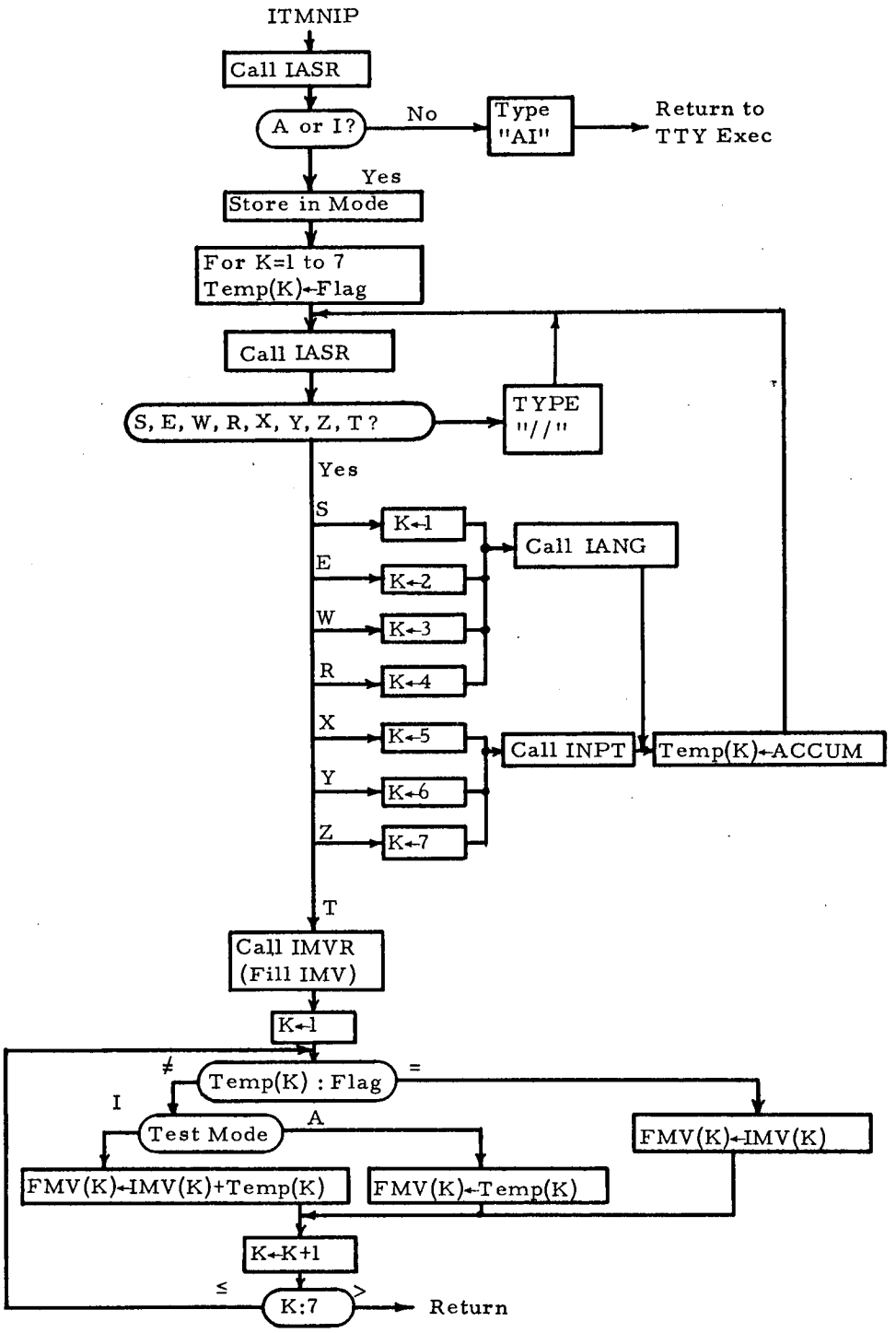


Figure 3.3 ITMNIP Flow Diagram

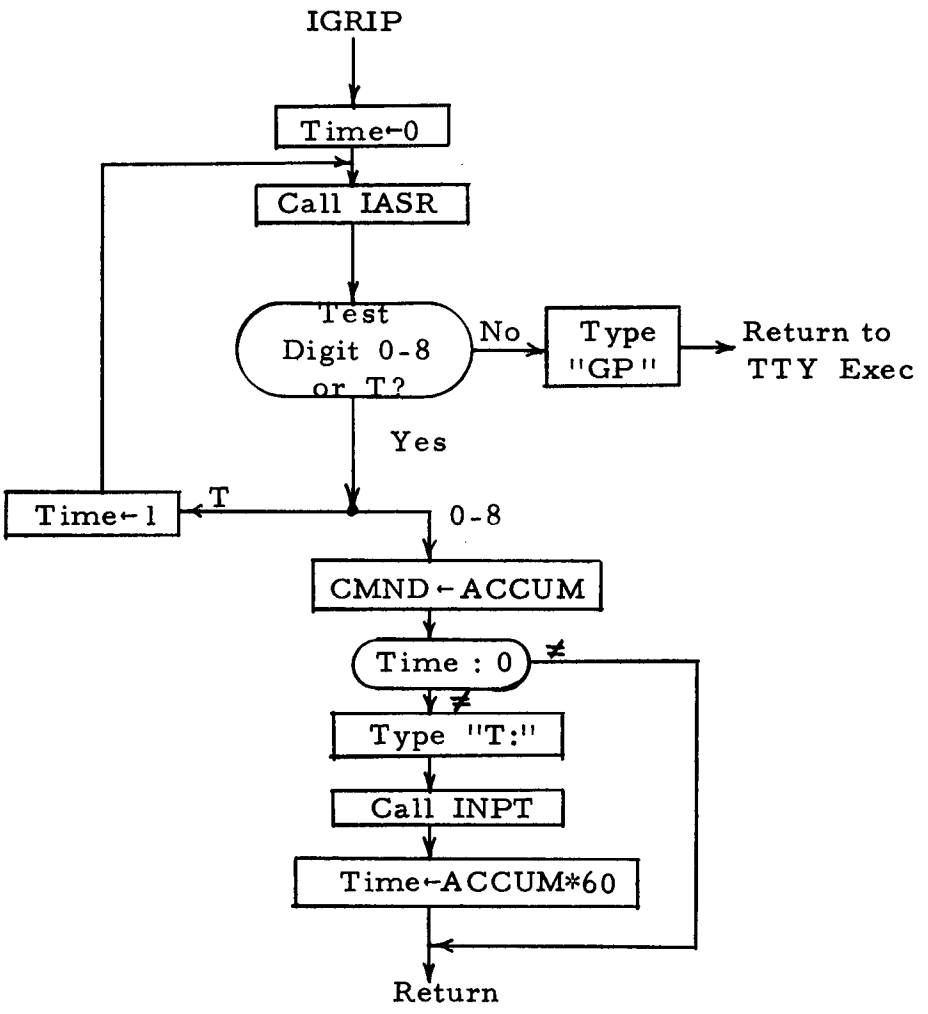


Figure 3-4. IGRIP Flow Diagram

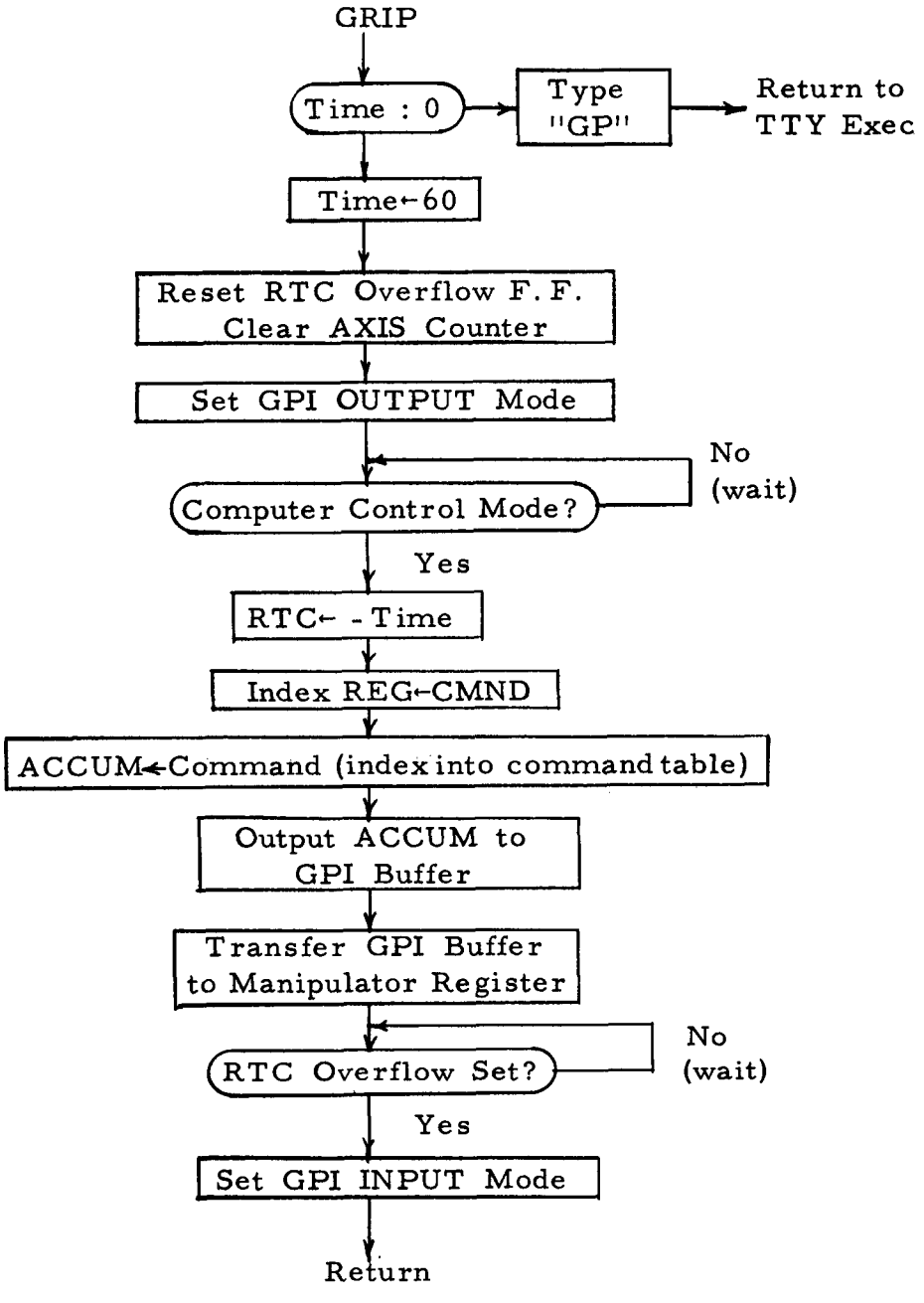


Figure 3-5. GRIP Flow Diagram

3.3.3. Path Control -- Straight Line

The path control algorithm demonstrates the relative ease with which the computer can precisely control the path of the manipulator. This forms a sharp contrast with the problems encountered by the operator in manually controlling the rate of each axis of the manipulator to form the desired path.

The algorithm which has been implemented allows straight-line path control along a vector formed by the orientation of the "hand" of the manipulator. The only input data required by the routine is the traverse distance. The TTY Exec calls the "IDIST" subroutine, Figure 3-7, which requests the distance, accepts it, and stores it in the DIST buffer. The AUTO Exec transfers the distance information from the tape to the DIST buffer. Both executive routines then call "VECTOR" to execute the algorithm.

From Figure 3-6, the relationship among the specified distance, the hand orientation, and the linear manipulator axis are:

$$\Delta X = D \sin WP \cos SR$$

$$\Delta Y = D \sin WP \sin SR$$

$$\Delta Z = -D \cos WP \qquad (1)$$

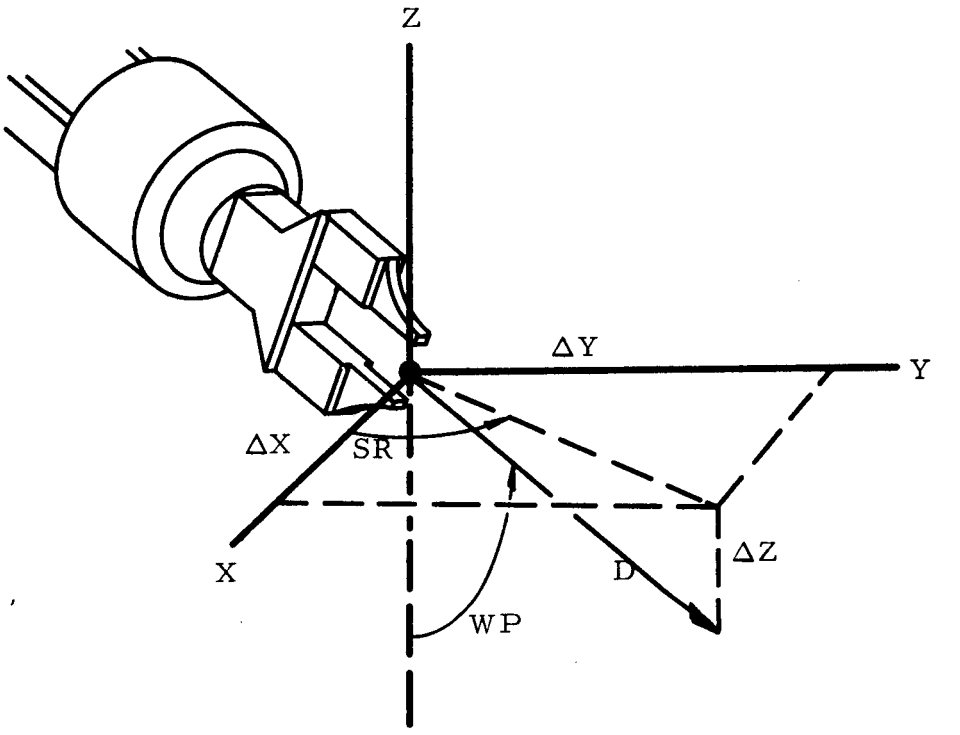


Figure 3-6. Vector and Hand Orientation Relationships

To form a straight line path, ΔX, ΔY, and ΔZ are all divided into an equal number of relatively fine increments. As shown in the flow diagram, Figure 3-8, these increments are added to the current manipulator position at a constant rate. The resultant path will be a close approximation to a straight line if the increments are sufficiently small; moreover, only the maximum speed of the motors limits the rate at which the manipulator is driven.

To manually generate a straight-line path, it is necessary to adjust the rates RX, RY, and RZ, such that the equation

$$\frac{\Delta X}{RX} = \frac{\Delta Y}{RY} = \frac{\Delta Z}{RZ} \qquad (2)$$

is satisfied. Consequently, the operator must control the rates of three axes simultaneously. To do so with any degree of precision even at relatively slow rates is nearly impossible. The generation of a more complex path (such as a circular arc) would be far more difficult because of the need to simultaneously control the rates and/or changes in rates of several axes.

It is far easier to control complex paths using a master-slave manipulator; nevertheless, considering the difficulty encountered by most people in drawing a straight line or a circle free hand, it is unlikely that a master-slave unit can be used for precise path control operations.

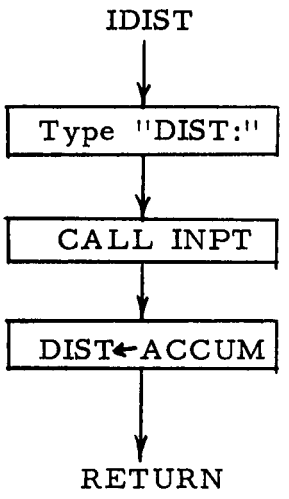


Figure 3-7. IDIST Flow Diagram

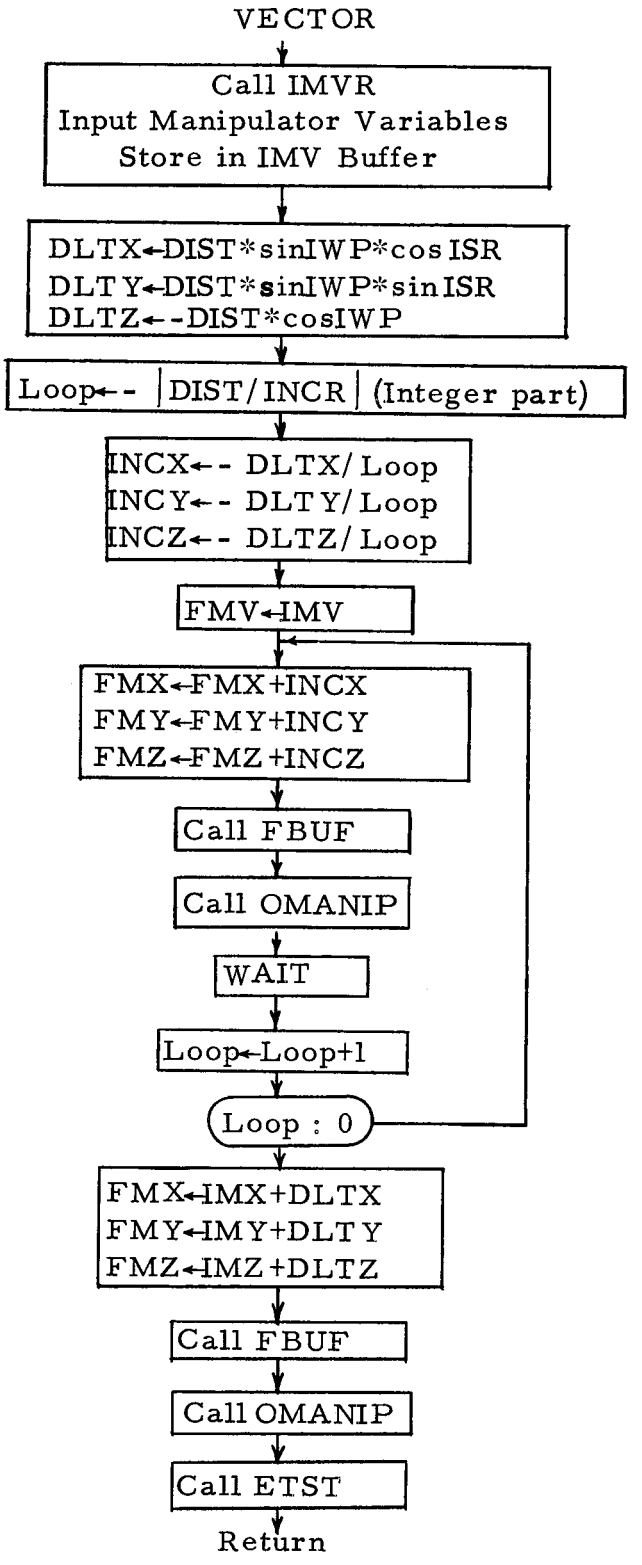


Figure 3-8. VECTOR Flow Diagram

3.3.4 Hand Position Control

The Hand Position Control algorithm most effectively demonstrates the ability of the computer to perform a complex operation with only one command and a minimum of accompanying data. The operator must specify only five variables which indicate the desired hand position (FHX, FHY, and FHZ) and orientation (Azimuth, Elevation).

A terminal manipulator position is computed which satisfied the five specified variables. Generally, it is the one requiring the least transit time; however, the projected path is tested by a special obstacle avoidance algorithm to insure that the manipulator will not collide with any predefined obstacles. If a potential collision is detected, a new configuration is specified or, if necessary, a set of intermediate positions are automatically selected to avert the collision.

The flow diagram of the "HOPOUT" (Hand Optimization, Obstacle Avoidance, and Output Routine) is shown in Figure 3-9.

3.3.4.1 Buffer Initialization

Three buffers must be initialized before calling the "HOPOUT"

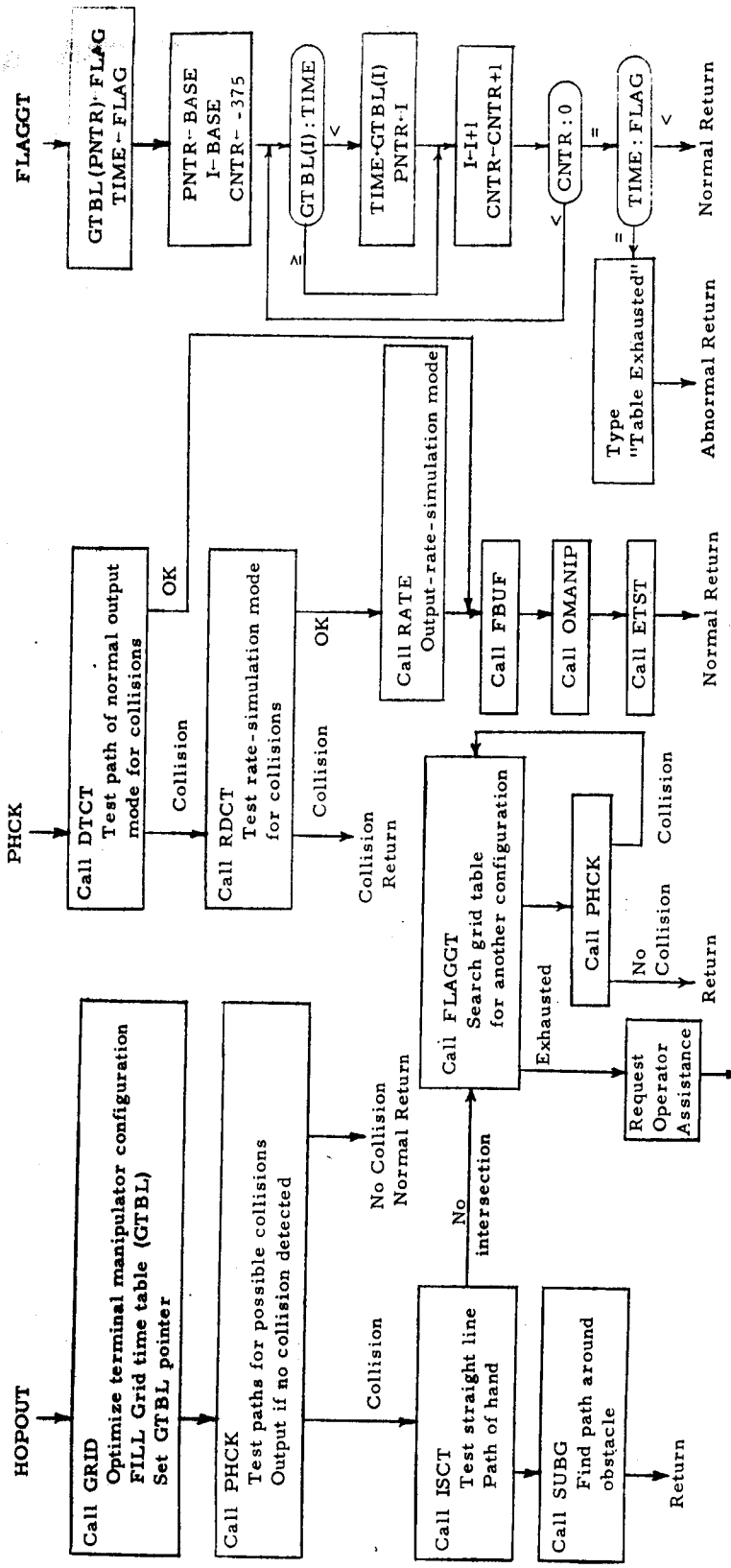


Figure 3-9. HOPOUT, PHCK, and FLAGGT Flow Diagrams

subroutine. These indicate the present manipulator variables (IMV), the present hand variable (IHV), and the desired hand variables (FHV).

The AUTO Exec reads the desired hand variables from tape stores them in the FHV buffer, then calls "CIHV" to fill the IMV and IHV buffers.

The TTY Exec allows two input modes. The "RTHAND" subroutine, Figure 3-10, requests all variables, accepts the terminal positions, and stores them in the FHV buffer. "CIHV" is called to fill the IMV and IHV buffers. The "ITHAND" subroutine is nearly identical to "ITMNIP" and is shown in Figure 3-11. This allows the operator to specify a partial set of variables in either an absolute or an incremental mode and fills the three buffers.

The "RELHND" subroutine described in Section 3.4.3 also calls the "HOPOUT" subroutine, but fills the buffers before calling it.

3.3.4.2 Optimization Algorithm

During the execution of many tasks, the operator is primarily

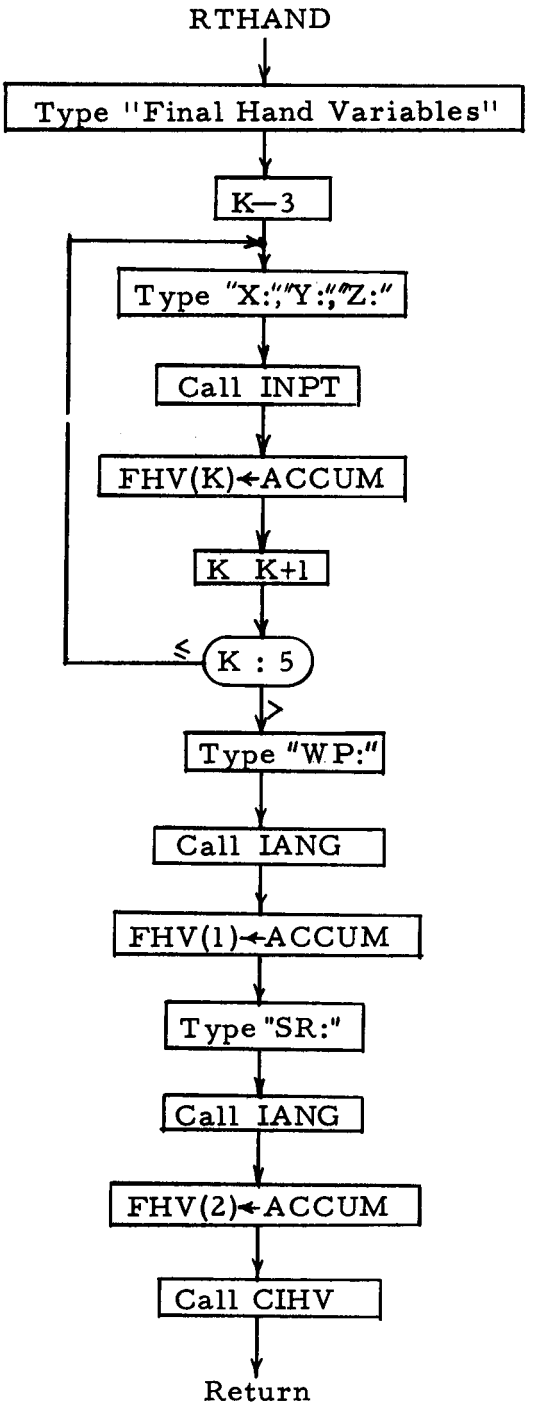


Figure 3-10. RTHAND Flow Diagram

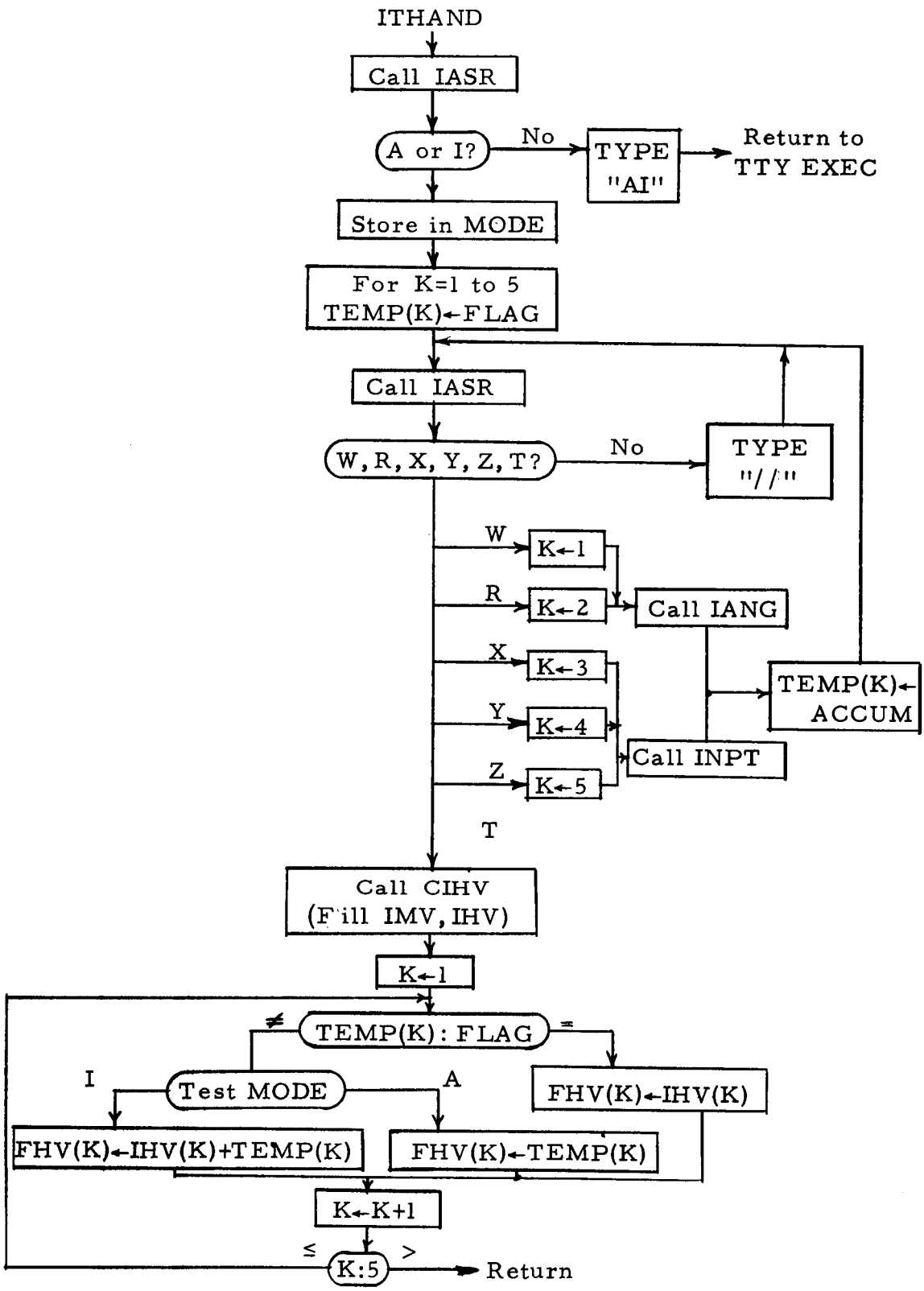


Figure 3-11. ITHAND Flow Diagram

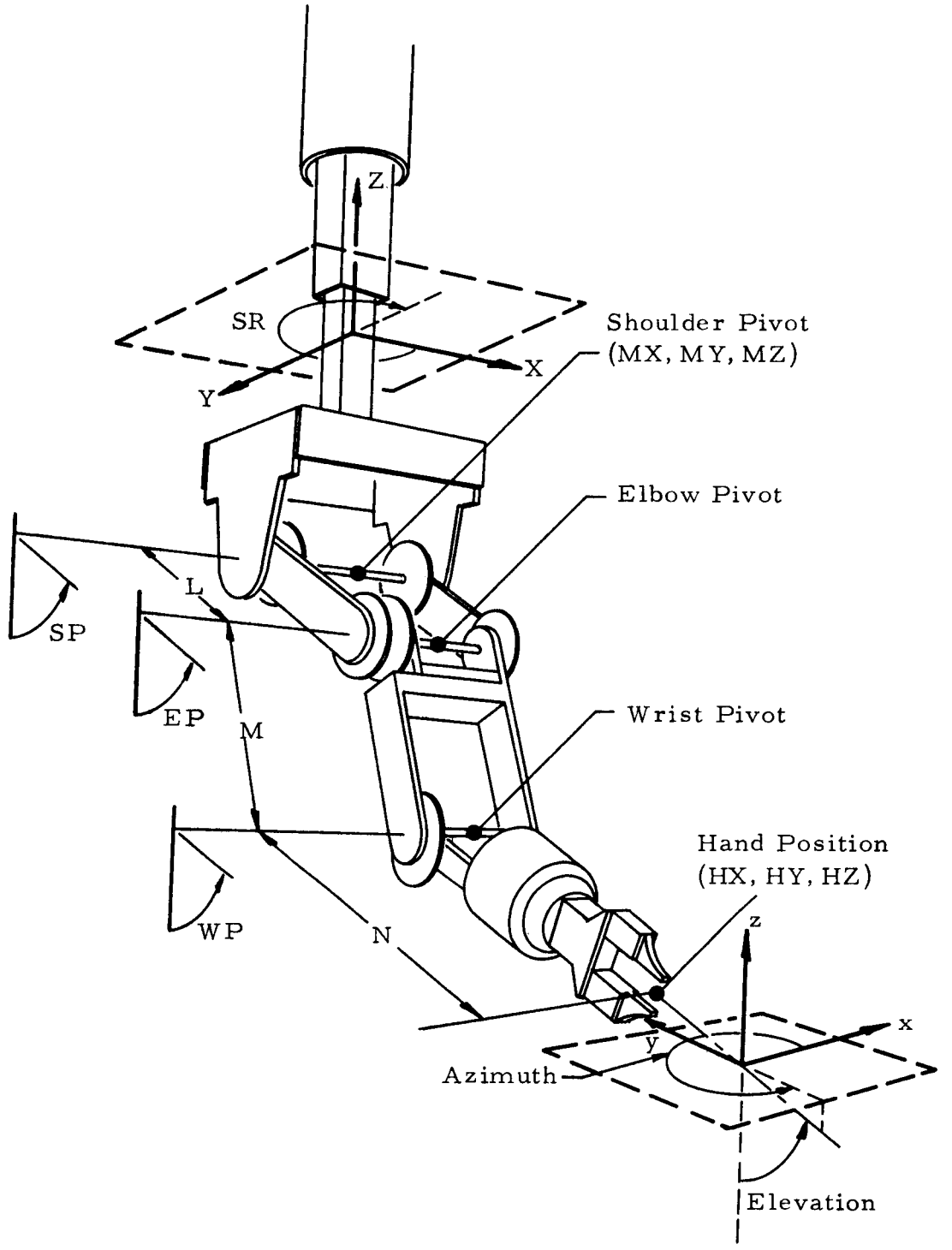
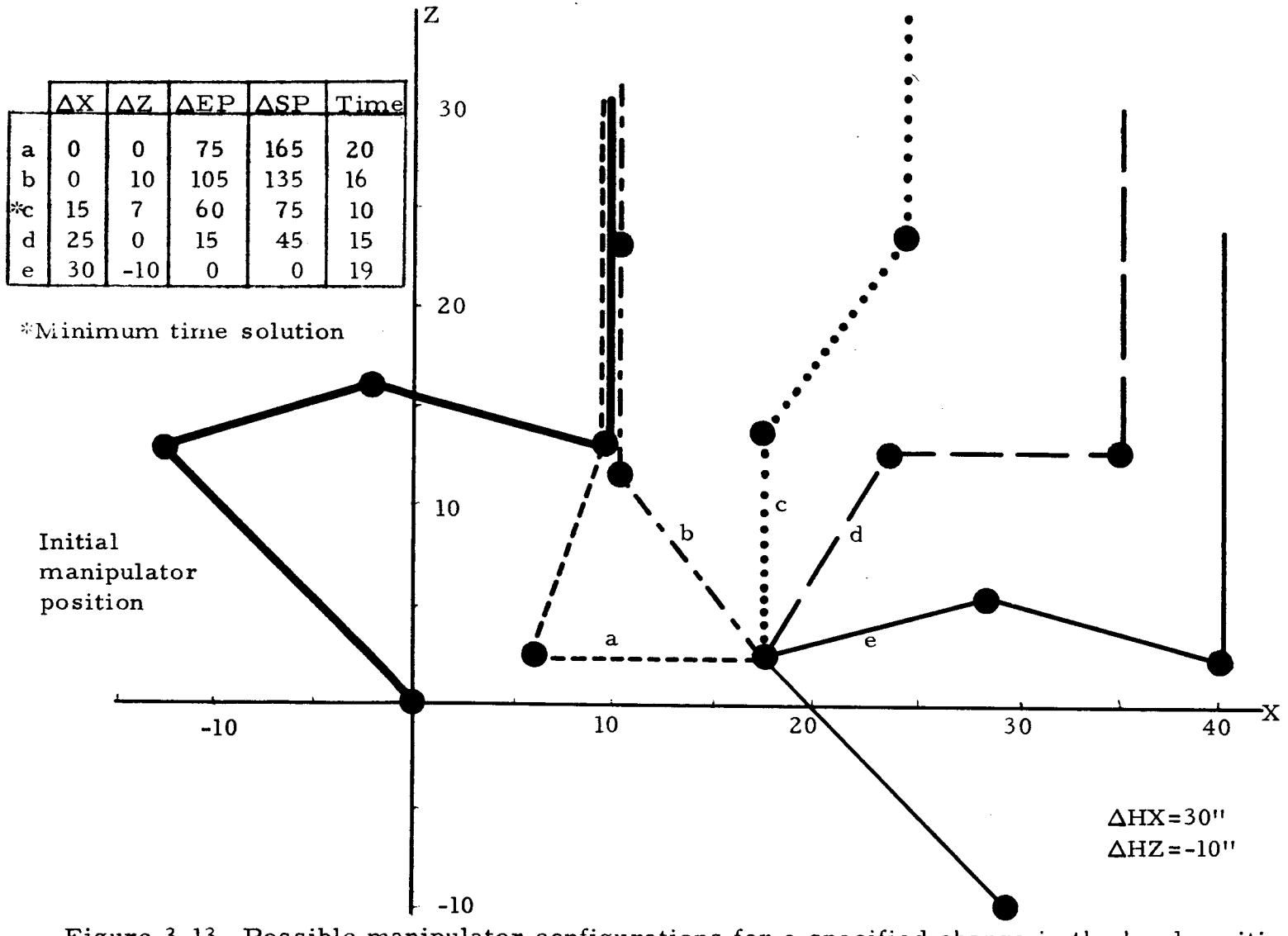


Figure 3-12. Manipulator and Hand Variable Relationships

problem to the specification of two independent variables belonging to the set MX, MY, MZ, EP, and SP and then using Equation (3) to compute the remaining three variables.

In order to enhance the performance of the system, it is possible to couple the specification of the two independent variables to a set of optimizing criteria (e.g., minimum time, minimum momentum, minimum energy, etc.). The algorithm implemented in the experimental system is based upon minimum time considerations. Figure 3-13 shows a few of the manipualtor configurations which can be assumed for a specified set of hand variables and the time required to reach each configuration from the initial position. Note that (c) is the minimum time solution computed by the algorithm. In effect, the principal concern of the algorithm is to find values for SP and EP such that the resulting ΔMX, ΔMY, or ΔMZ variable which requires the longest transit time is reduced to the point that it is equal to or exceeded by the transit time of another axis.

Since it is not imperative to compute the absolute minimum time solution, two basic simplifying assumptions have been made. The first one is that the manipulator motion is so slow that the dynamics of the system can be neglected. The second is that



	ΔX	ΔZ	ΔEP	ΔSP	Time
a	0	0	75	165	20
b	0	10	105	135	16
*c	15	7	60	75	10
d	25	0	15	45	15
e	30	-10	0	0	19

*Minimum time solution

Figure 3-13. Possible manipulator configurations for a specified change in the hand position

the motors drive the axes at a constant rate. Of the two, the second will produce the greatest error. This is due to the fact that series-would universal motors are used to drive the manipulator and that their speed is strongly dependent upon the load. Experimentally, this assumption has produced no adverse effects; however, if necessary, it would be possible to modify the algorithm to select one of several rates for each axis dependent upon the load, the initial manipulator position, and the final manipulator position.

Based upon the two assumptions, the time required by each axis of the manipulator is given by

$$\begin{aligned} TMX &\cong |(FMX-IMX)/RX| \\ TMY &\cong |(FMY-IMY)/RY| \\ TMZ &\cong |(FMZ-IMZ)/RZ| \end{aligned} \tag{5}$$

and

$$\begin{aligned} TSP &\cong |(FSP-ISP)/RSP| \\ TEP &\cong |(FEP-IEP)/REP| \end{aligned} \tag{6}$$

where the prefixes T, R, F, and I denote time, rate, final values, and initial values respectively. The WP and SR times are not considered since the terminal values of these angles are unique.

From Equation (5), Equation (3) can be rewritten as

$$TMX \cong |K_0 - K_3(L \sin FSP + M \sin FEP)|$$

$$TMY \cong |K_1 - K_4(L \sin FSP + M \sin FEP)|$$

$$TMZ \cong |K_2 + K_5(L \cos FSP + M \cos FEP)| \qquad (7)$$

where

$$K_0 = (FHX - IMX - N \sin FWP \cos FSR)/RX$$

$$K_1 = (FHY - IMY - N \sin FWP \sin FSR)/RY$$

$$K_2 = (FHZ - IMZ + N \cos FWP)/RZ$$

$$K_3 = (\cos FSR)/RX$$

$$K_4 = (\sin FSR)/RY$$

$$K_5 = 1/RZ \qquad (8)$$

The optimization algorithm utilizes a grid technique to find the minimum time solution. This method was chosen because it avoids problems with local minimum values, is easily implemented, allows pivot and boundary constraint tests, and is compatible with the obstacle avoidance routines described in the next section.

The EP and SP angles were selected as the two independent variables since they already appear as functions of MX, MY, and MZ in Equation (3). A grid is formed by restricting the values of EP and SP to an integral multiple of 15 degrees throughout their

entire range. The grid for the experimental system is shown in Figure 3-14. For each intersection on the grid, the times required by the X, Y, Z, EP, and SP axes to travel from their initial positions to those which satisfy Equation (3) for the EP, SP values are computed. The greatest of these times represents the time required to reach the terminal point (excluding TWP and TSR). As each of the intersections is considered, the transit time is stored in a "grid" table for future use by the obstacle avoidance routine. The time is also compared with the smallest time previously encountered. If equal or larger, it is ignored; otherwise, its value and the grid intersection pointer are preserved. The combination which remains after all intersections have been tested is then chosen as the "minimum time" solution.

During the computation and selection sequence, two special constraint tests are performed. The first one compares the MX, MY, and MZ positions resulting from the SP and EP values with the physical bounds of the manipulator working area. If they are exceeded, the time value is flagged by setting it to the maximum value permitted by the computer. The second test is used to insure that the relative angles between each limb of the arm are within the allowable range and also that the configuration

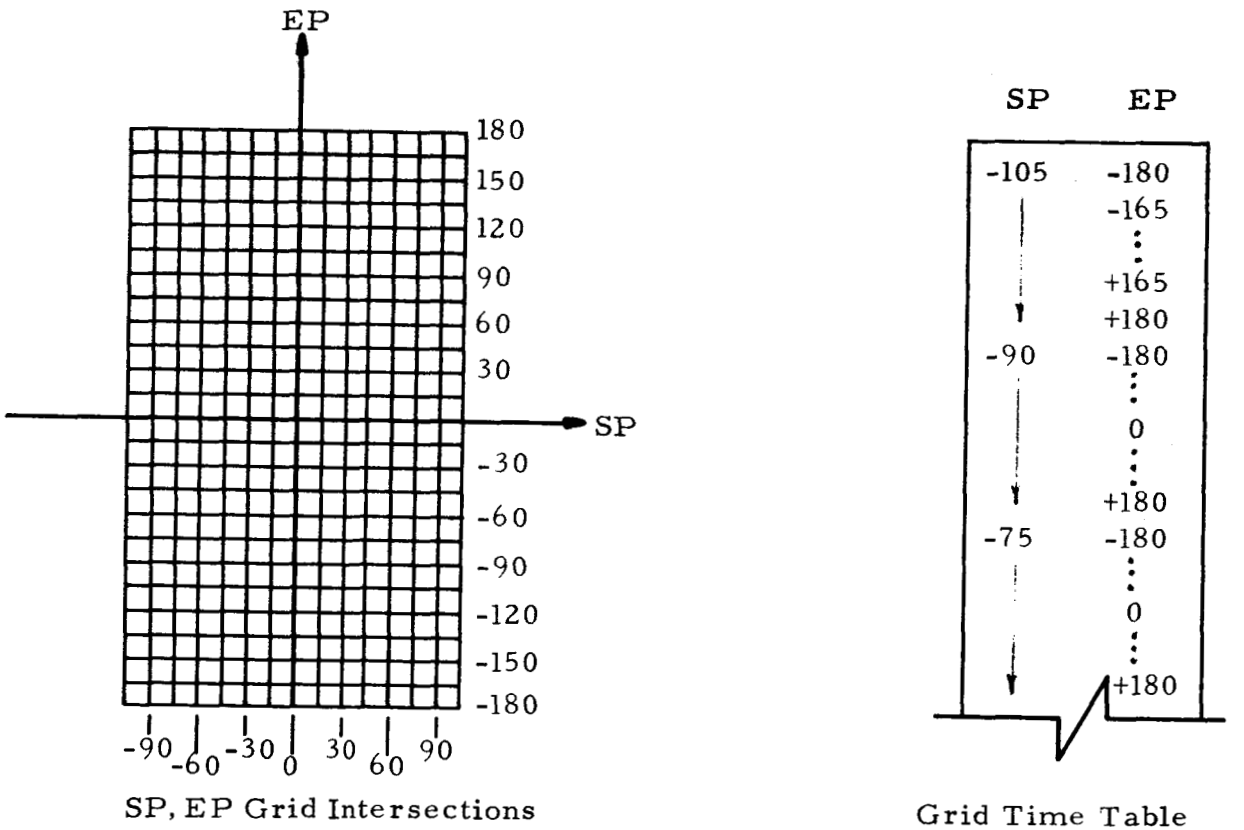


Figure 3-14. Example of SP, EP Grid and the Grid Time Table

will not cause damage to the hoist assembly or to any part of the motor platform. Examples of these constraints are shown in Figure 3-15. In order to indicate the permissible pivot combinations, a special table has been formulated to indicate upper and lower wrist pivot limits for each EP, SP grid intersection. If the wrist pivot is not within these limits, the transit time is flagged and the next intersection point is tested. Under a few circumstances (e.g., when the relative angle is less than 45°), the EP, SP combination is not permitted for any WP value. In these cases, a special zero flag is stored in the constraint table. This table is prepared in advance and is loaded into the memory of the computer with the optimization program.

Figures 3-16 and 3-18 demonstrate several terminal manipulator configurations computed by the optimization routine for various increments in the X and Z hand positions respectively. In addition the transit times required by the manipulator are listed and are compared with the times which would be required if EP and SP were not incremented.

Figures 3-17 and 3-19 show the paths followed by each pivot and the hand position for the examples shown in Figures 3-16 and 3-18. Note that there is an abrupt change

in the paths as each axis reaches the terminal positions.

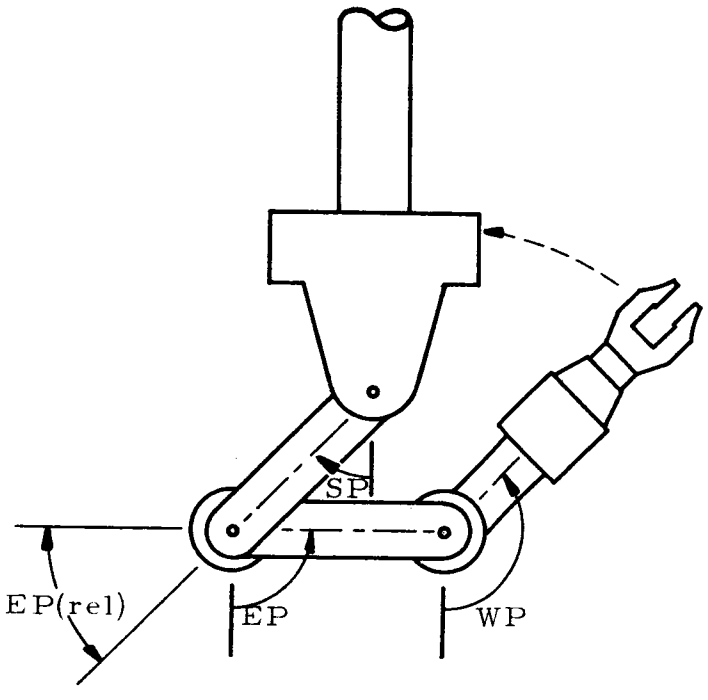
Before calling the optimization routine, it is necessary to fill the IMV, IHV, and FHV buffers. In addition, the following buffers are loaded into computer's memory with the optimization routine·

1. The constraint table.
2. The X, Y, Z manipulator boundary table.
3. Tables containing the values of L sin FSP, L cos FSP, M sin FEP, M cos FEP (for allowed values of FSP, FEP).

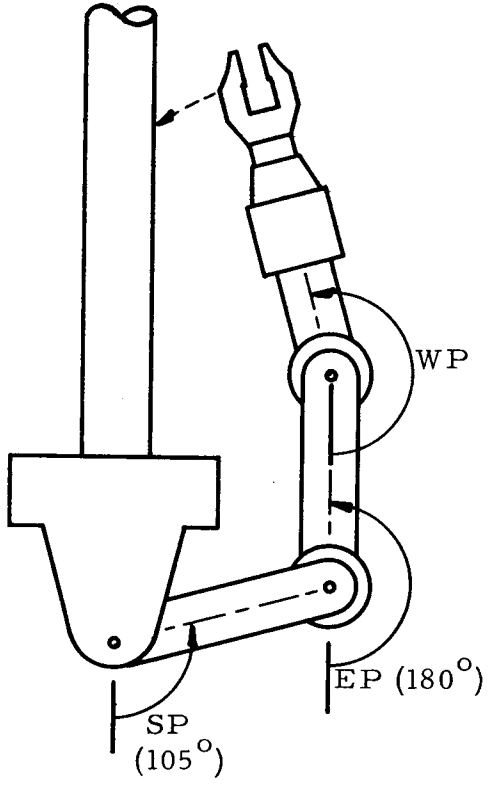
The sine and cosine terms were determined in advance to reduce the computation time required by the algorithm (approximately 80 msec).

The algorithm fills the grid time table, the grid intersection pointer, the FMV buffer, and the transit time, TMAX.

The flow diagram of the optimization algorithm is shown in Figure 3-20 and the individual subroutines appear in Figure 3-21.



a. EP (rel) at limit
WP must be constrained to prevent hand from hitting motor



b. SP and EP at absolute limits
WP must be constrained to prevent hand from hitting tube

Figure 3-15. Examples of SP , EP , and WP constraints

	ΔX	TΔHX	TΔMX
a	-30	7	19
b	-40	10	25
c	-50	16	31

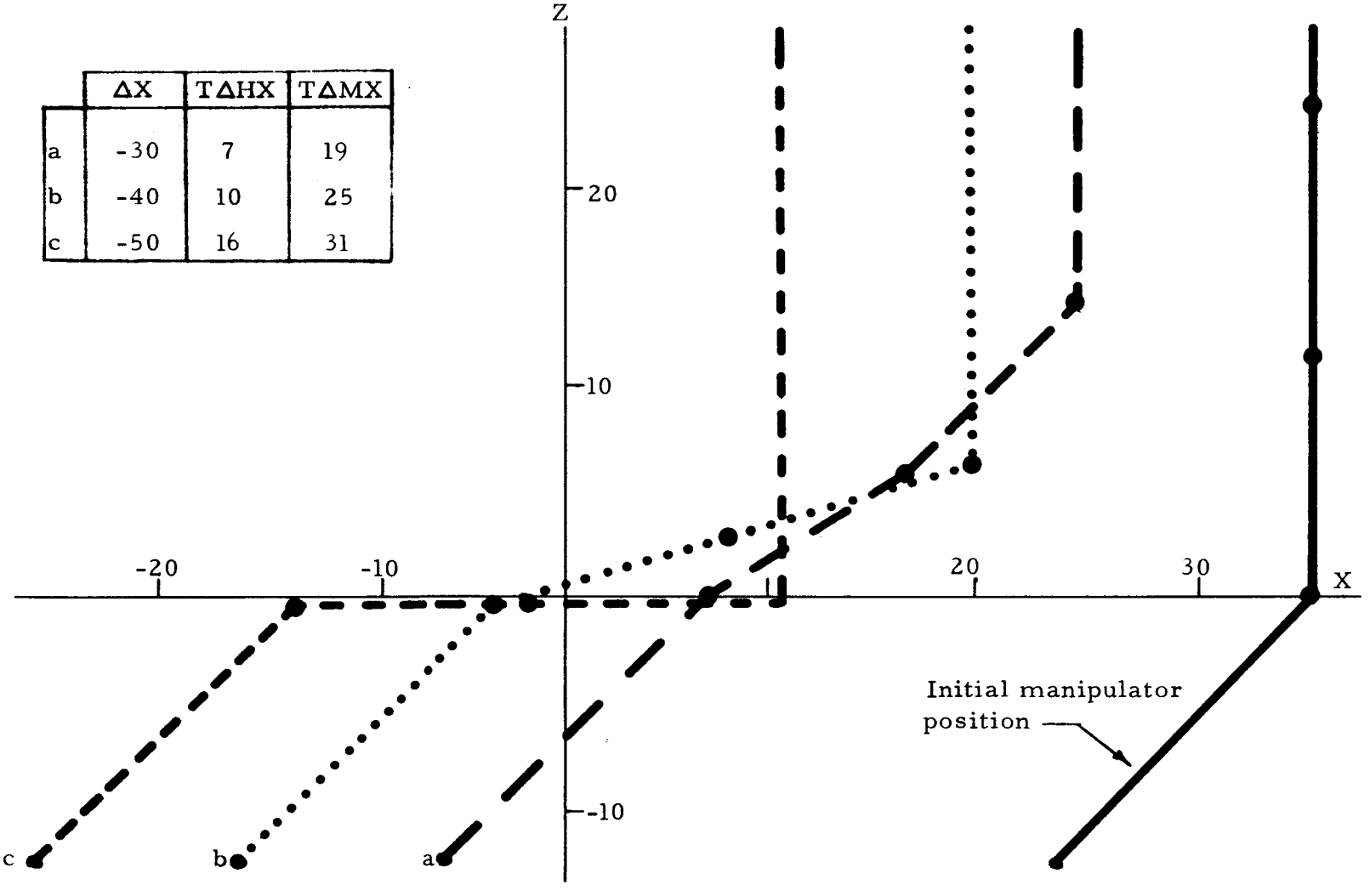
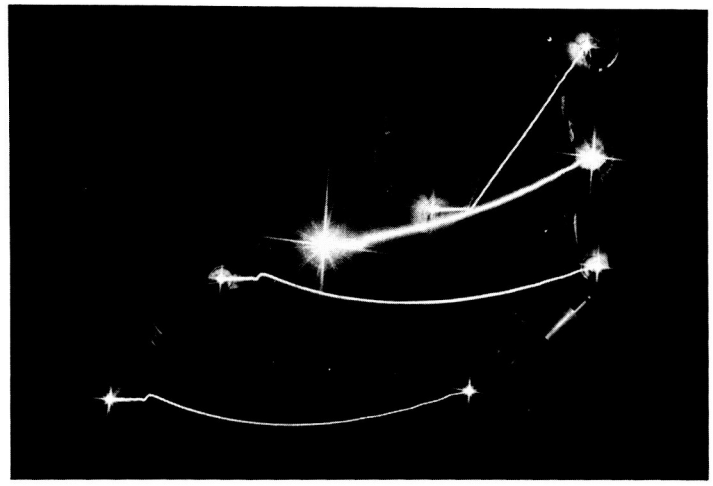
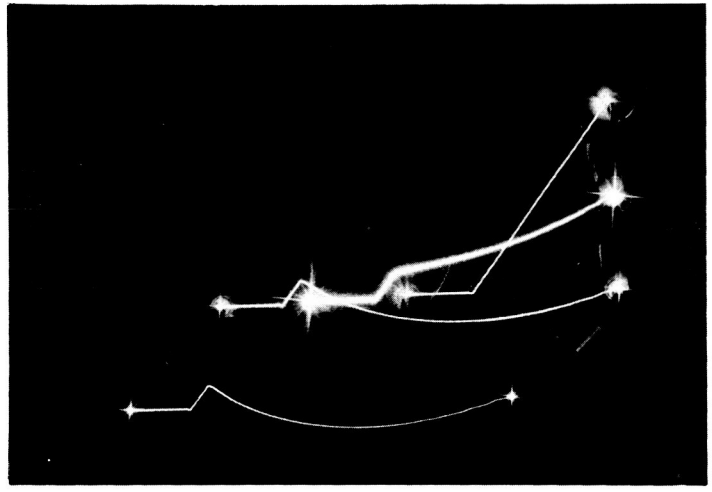


Figure 3-16. Optimized manipulator configurations for HX increments

a. ΔHX=−30

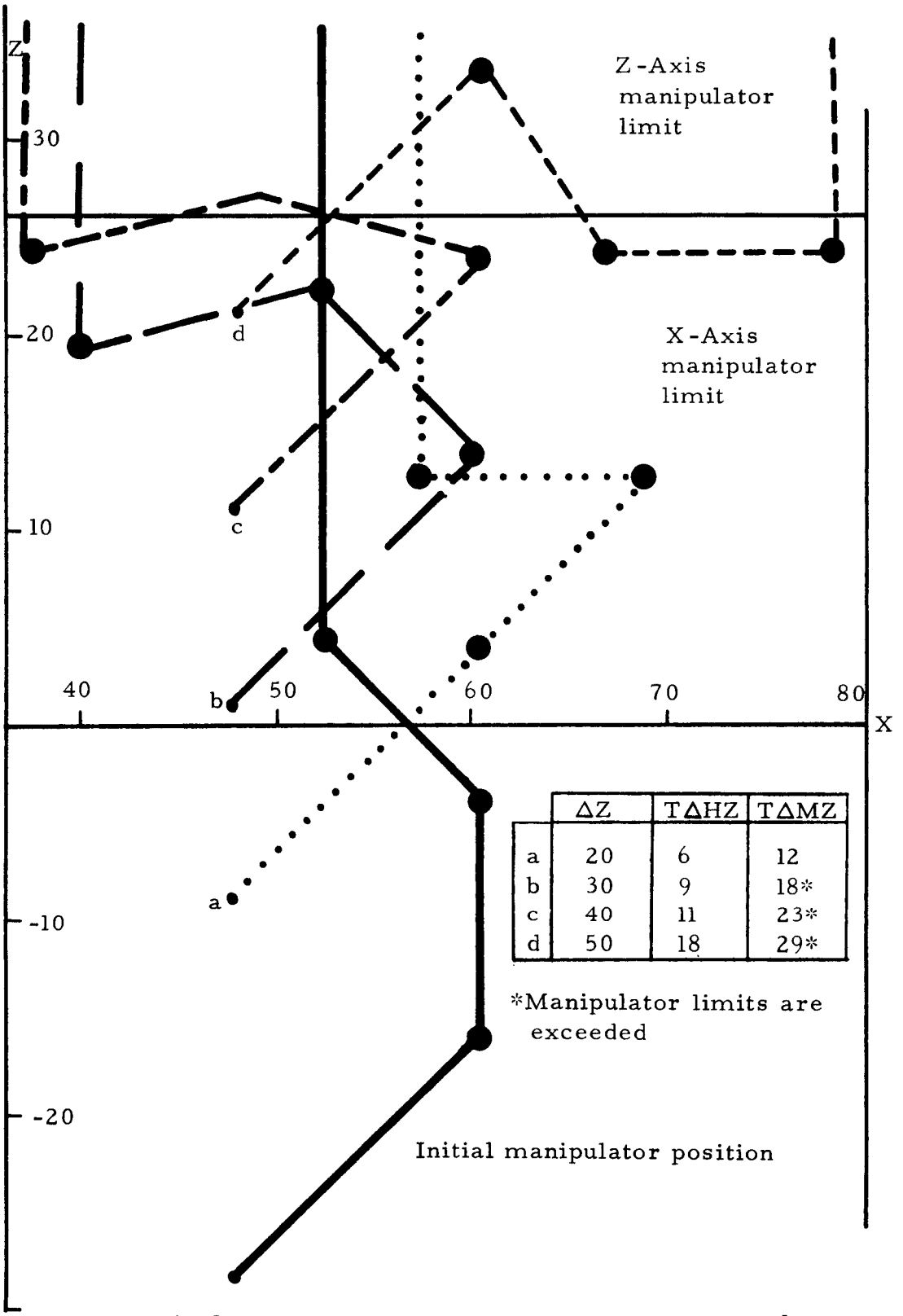


b. ΔHX=−40



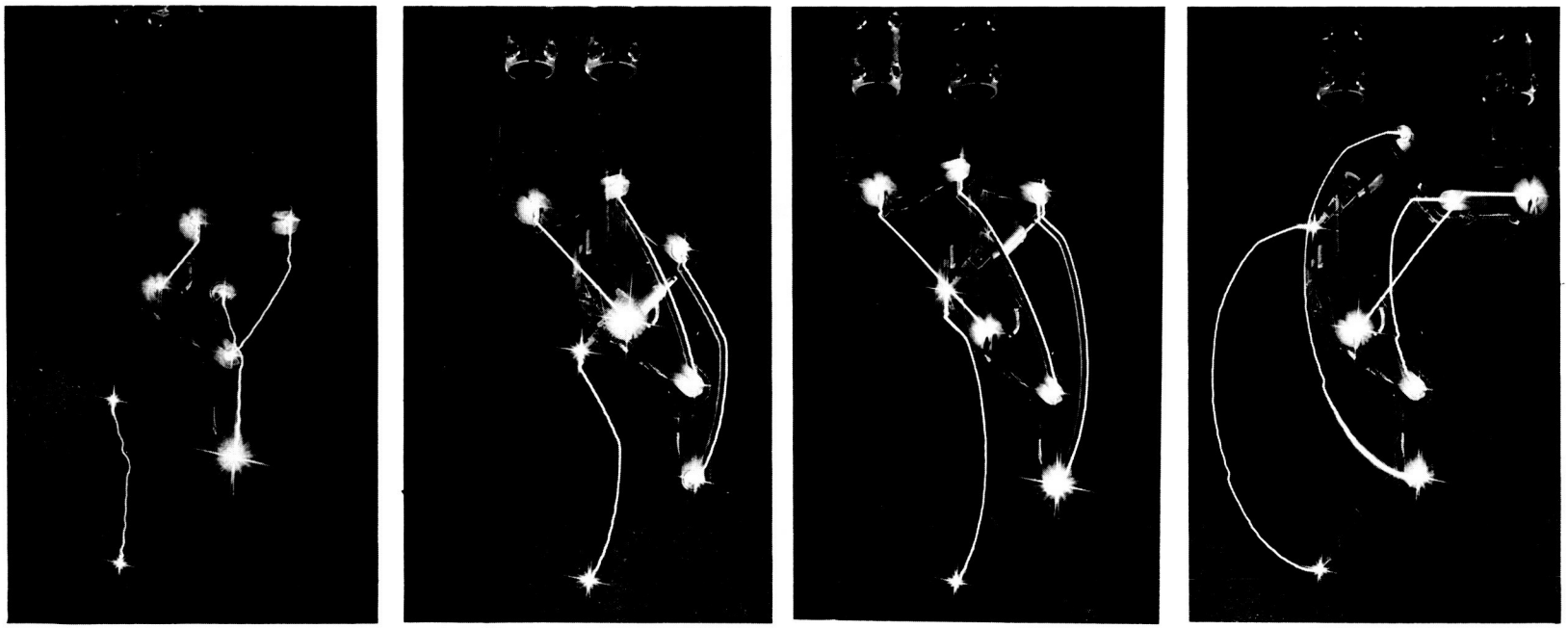
c. ΔHX=−50

FIGURE 3-17. PATHS OF MANIPULATOR FOR EXAMPLES IN FIGURE 3-16



	ΔZ	TΔHZ	TΔMZ
a	20	6	12
b	30	9	18*
c	40	11	23*
d	50	18	29*

Figure 3-18. Optimized manipulator configurations for HZ increments



a. ΔHZ=20 b. ΔHZ=30 c. ΔHZ=40 d. ΔHZ=50

FIGURE 3-19. PATHS OF MANIPULATOR FOR EXAMPLES IN FIGURE 3-18

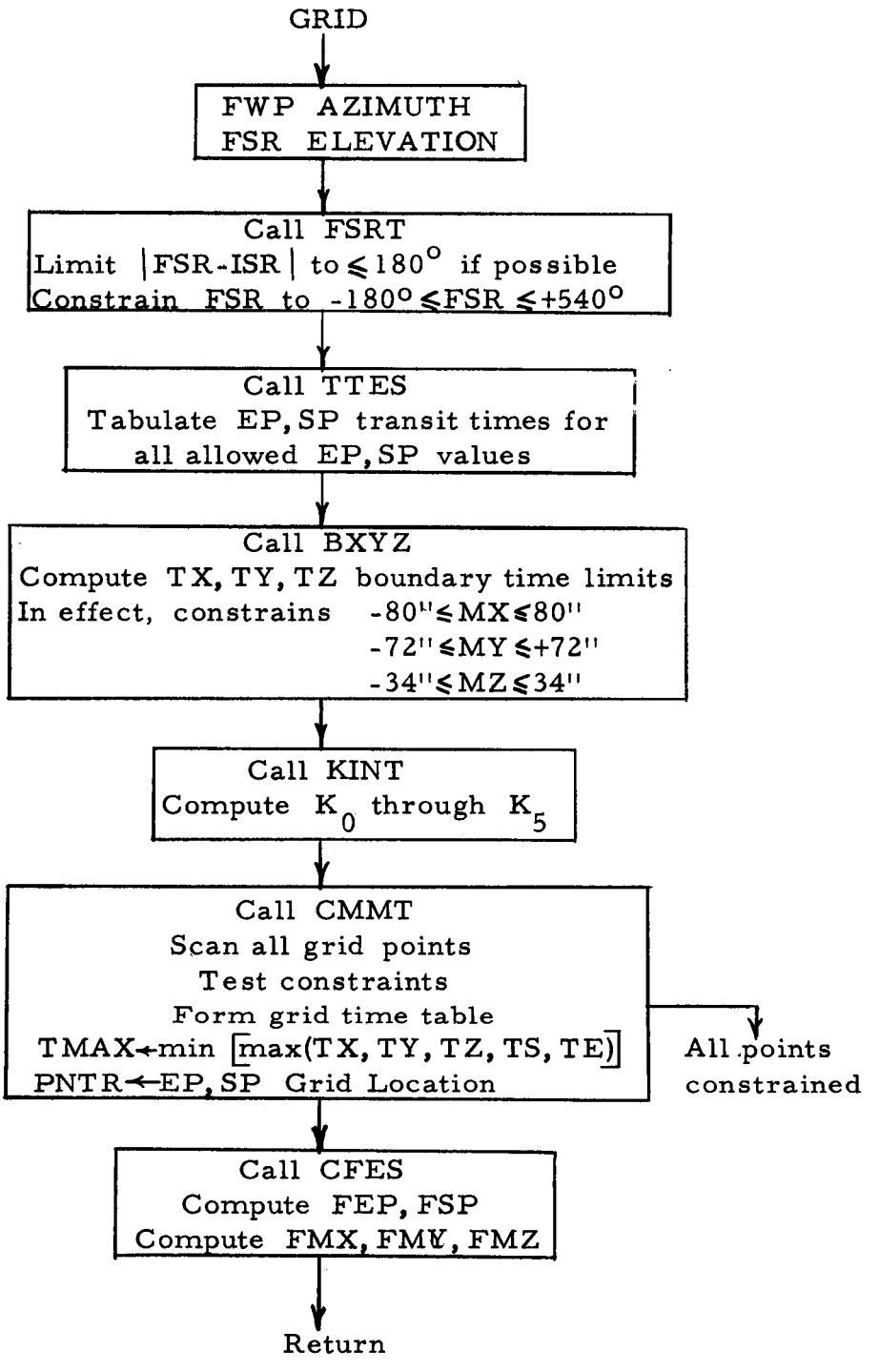


Figure 3-20. GRID Flow Diagram

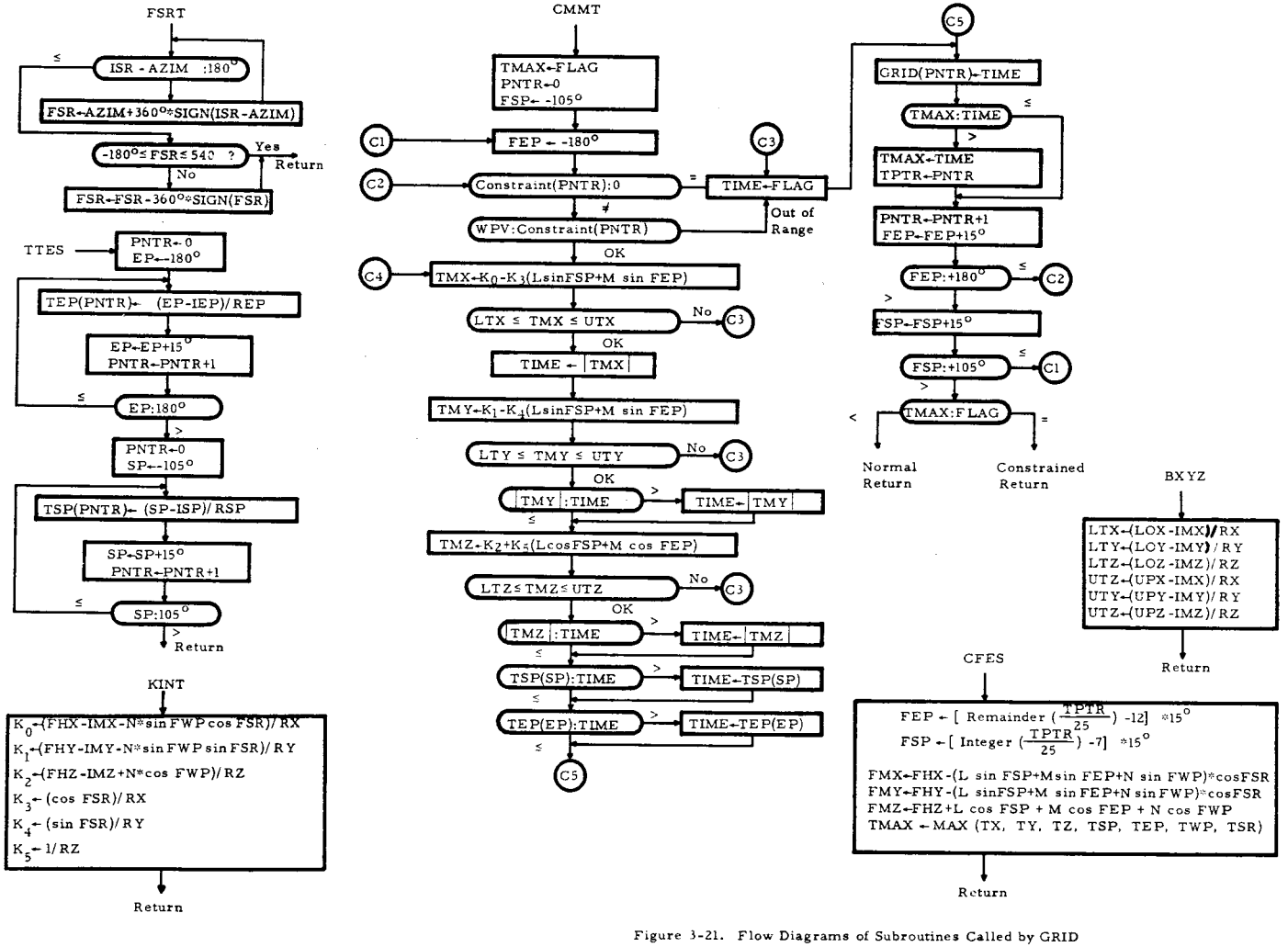
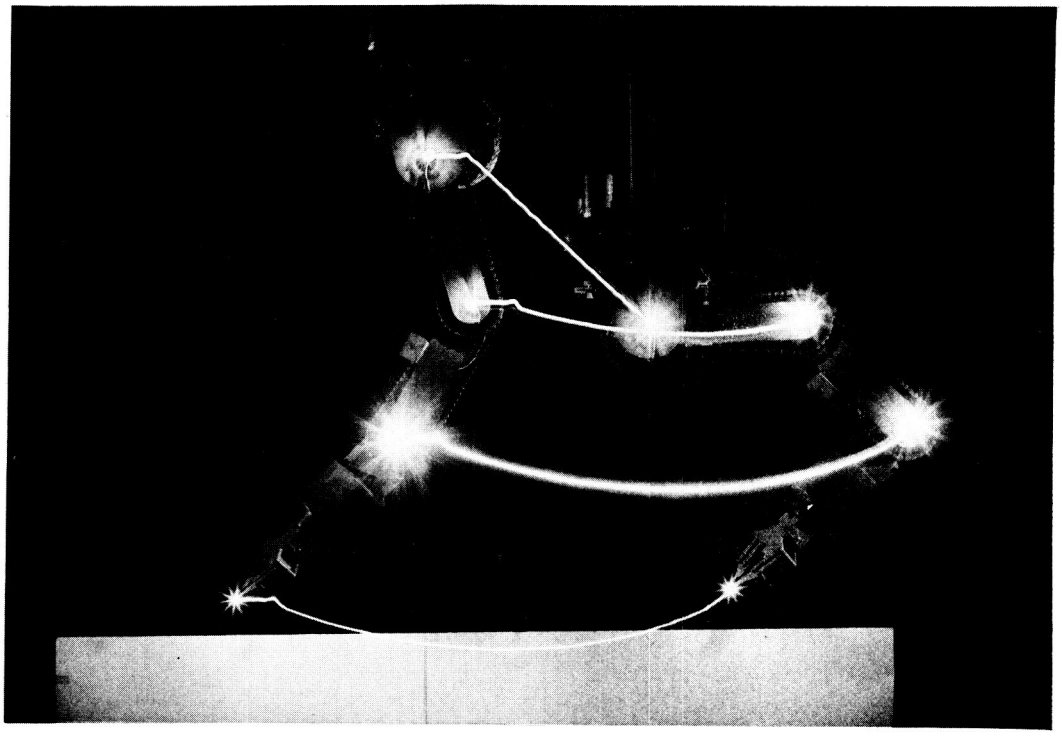


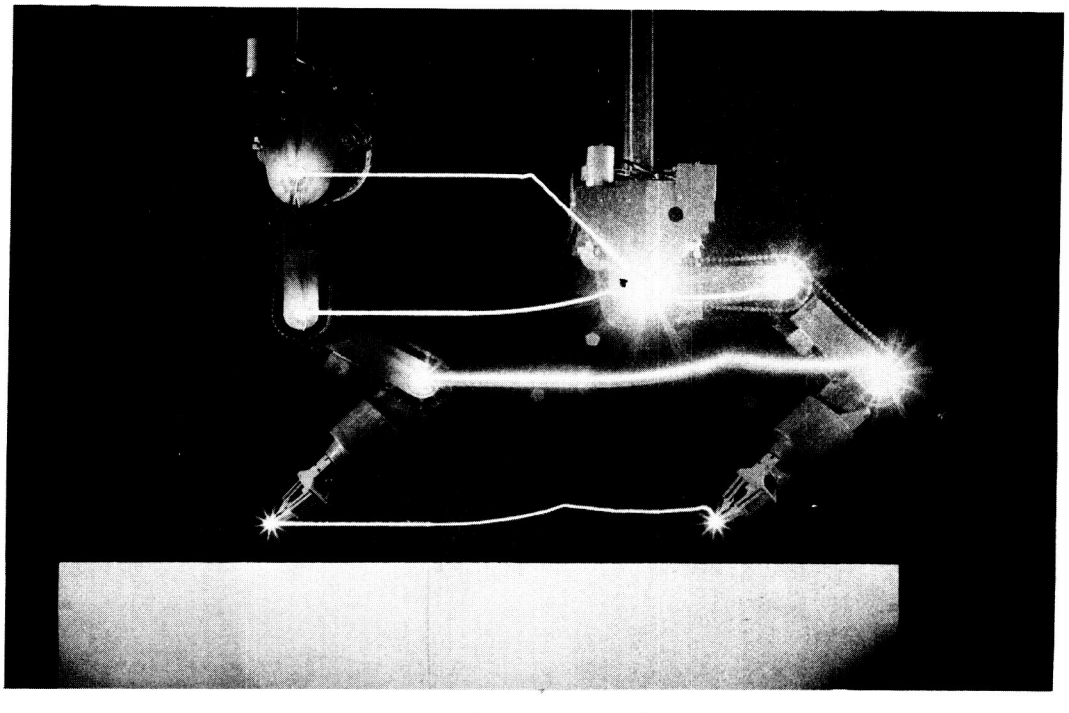
Figure 3-21. Flow Diagrams of Subroutines Called by GRID

3.3.4.3 Obstacle Avoidance

One of the major tasks which was deferred until the completion of the initial phase of the experimental system was the inclusion of a feedback path to inform the computer when the manipulator was within close proximity to an obstacle. Without this feedback, the operator was required to carefully observe the manipulator's motions when under computer control to insure that no part of the manipulator would collide with any obstacles. Unfortunately, this task was considerably more difficult than when using the manual control mode for two reasons. First, with computer control, the seven independent servosystems drive all motors simultaneously and at their maximum rate until the desired coordinates are reached -- manually, this is rarely done. Second, the operator did not always know the terminal configuration of the manipulator and even when it was known, the exact path of the manipulator could not always be accurately visualized in advance; hence, a number of collisions occurred before the operator was aware of the danger. For example, Figure 3-22a demonstrates the path of the manipulator when the hand position optimization algorithm was directed to move the hand to a location 40 inches to the left of the initial position. Note that the table would have been hit if



a. First path generated by Optimization routine



b. Path allowed by Obstacle Avoidance routine

Figure 3-22. Example of a possible collision between the manipulator and an obstacle

it had been located directly under the hand even though both the initial and terminal manipulator configurations would be acceptable.

The collision detection subroutines were formulated to recognize potential hazards such as that described above. In addition, a subroutine was formulated which tries to automatically select an acceptable path over or around the obstacle. Figure 3-22b demonstrates the path generated to prevent the manipulator from striking the table.

In order to facilitate the development of the subroutines, the following assumptions were made:

1. All obstacles have the same shape and orientation (see next section).
2. All obstacles remain stationary (due to lack of sensory feedback).
3. The table accommodates only four obstacles (but can easily be enlarged).
4. All obstacle dimensions specified by the operator are enlarged (see next section).
5. Additional operator assistance is required if more than one obstacle is encountered en route to the specified destination (see Section 3.3.4.3.3)
6. The routines are called only by the Hand Position Control algorithm (see below).

In spite of the restrictions, the subroutines have increased the value of the computer control technique by a significant amount. The collision detection routine has drastically reduced the number of collisions encountered by the experimental manipulator and the obstacle evasion capability dramatically illustrates how the system performance can be enhanced. In the semi-automatic mode, it reduces the amount of information which the operator must specify to direct the manipulator around an object. In the automatic mode, it allows obstacles to be placed into or removed from the manipulator task area without modifying any pre-recorded data.

Since the obstacle bounds are enlarged, the routines are primarily used to supervise the gross motions of the manipulator and they are called only by the Hand Position Control Algorithm. The remaining path and position control algorithms must be used (cautiously) to work within the obstacle bounds under computer control. In the future, the experimental system will be modified to provide a feedback path which will inform the computer when the manipulator is in close proximity to an obstacle. A set of routines can then be formulated which allow the manipulator to work within the gross obstacle bounds but which do not require detailed operator supervision to avert a collision. These can be

coupled with the obstacle avoidance routines presented below to form a comprehensive obstacle detection and avoidance scheme.

3.3.4.3.1 Specifications of Obstacle Bounds

Due to the lack of sensory feedback from the manipulator to the computer, it is necessary for the operator to specify all obstacle bounds via the TTY console. The bounds must conform to the following restrictions:

1. The shape must be reduced to a rectangular parallelopiped.
2. The sides extend to the floor.
3. The sides are parallel to the X and Y axes.
4. The effective bounds will be enlarged five to ten inches beyond those specified by the operator.

Figure 3-23 demonstrates how several arbitrary geometrical shapes can be bounded to satisfy the first three restrictions. The second item was included for programming ease but is justified by the fact that the manipulator travels on overhead rails. The third item was included to reduce the programming complexity during the formulation and evaluation of the obstacle avoidance routines.

The last restriction most severely limits the utility of the

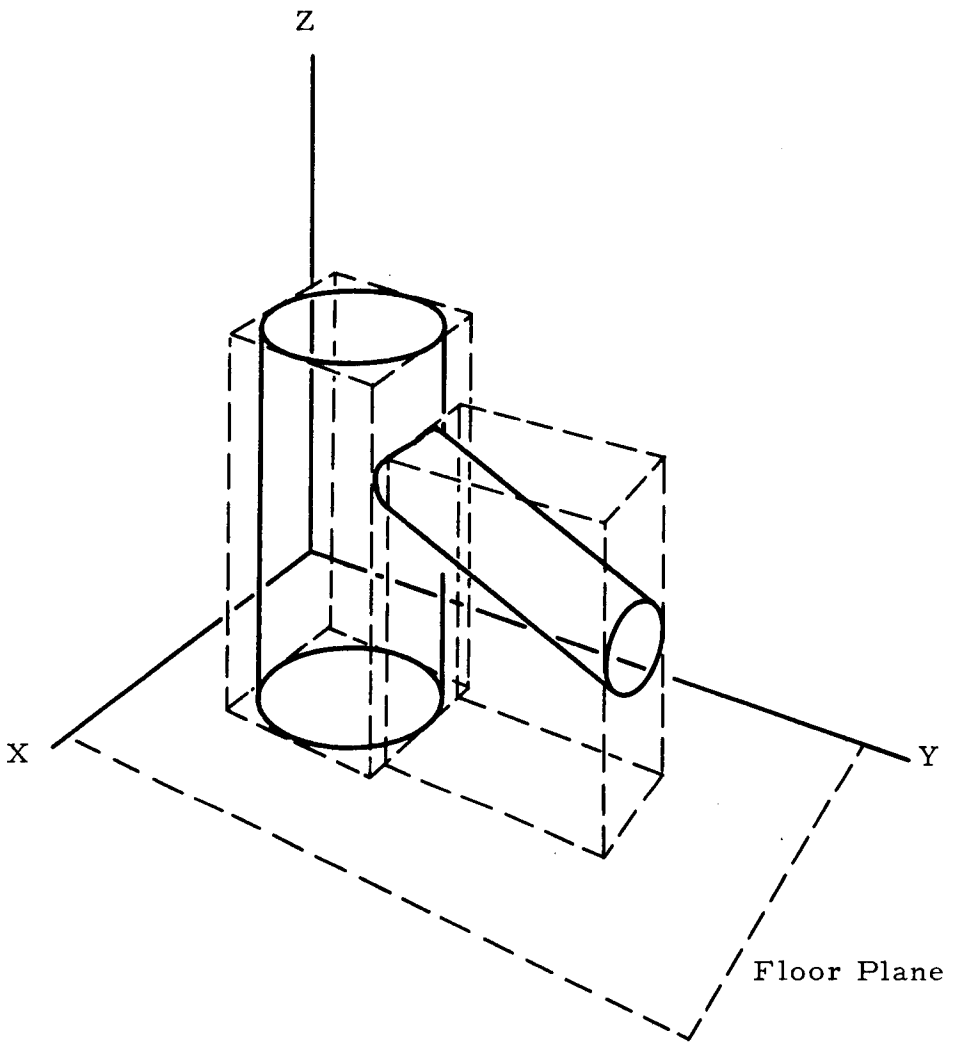


Figure 3-23. Typical obstacle bounds

algorithm but is necessary because of two problems. First, the collision detection scheme considers the projected path of the manipulator. This path is only an approximation and the load-speed characteristics of the motors can cause the actual path to deviate from the simulated one. It would be possible to eliminate this problem by transmitting a series of small path increments to the manipulator in a manner similar to the path control algorithm. Second, it is both difficult and time consuming to insure that all parts of the manipulator will not hit the obstacle. In order to minimize the computation time and reduce the complexity of the algorithm, only the positions of the shoulder pivot, elbow pivot, wrist pivot, and hand are compared with a set of "effective obstacle bounds". The effective bounds are the bounds specified by the operator plus five to ten inches to conservatively accommodate the area occupied by the motor platform and the limbs of the arm.

The "IBOUND" subroutine is called by the TTY Exec and allows the operator to specify the obstacle bounds. As shown in the flow diagram, Figure 3-24, it requests and accepts the X and Y bounds and the top (Z) of the obstacle.

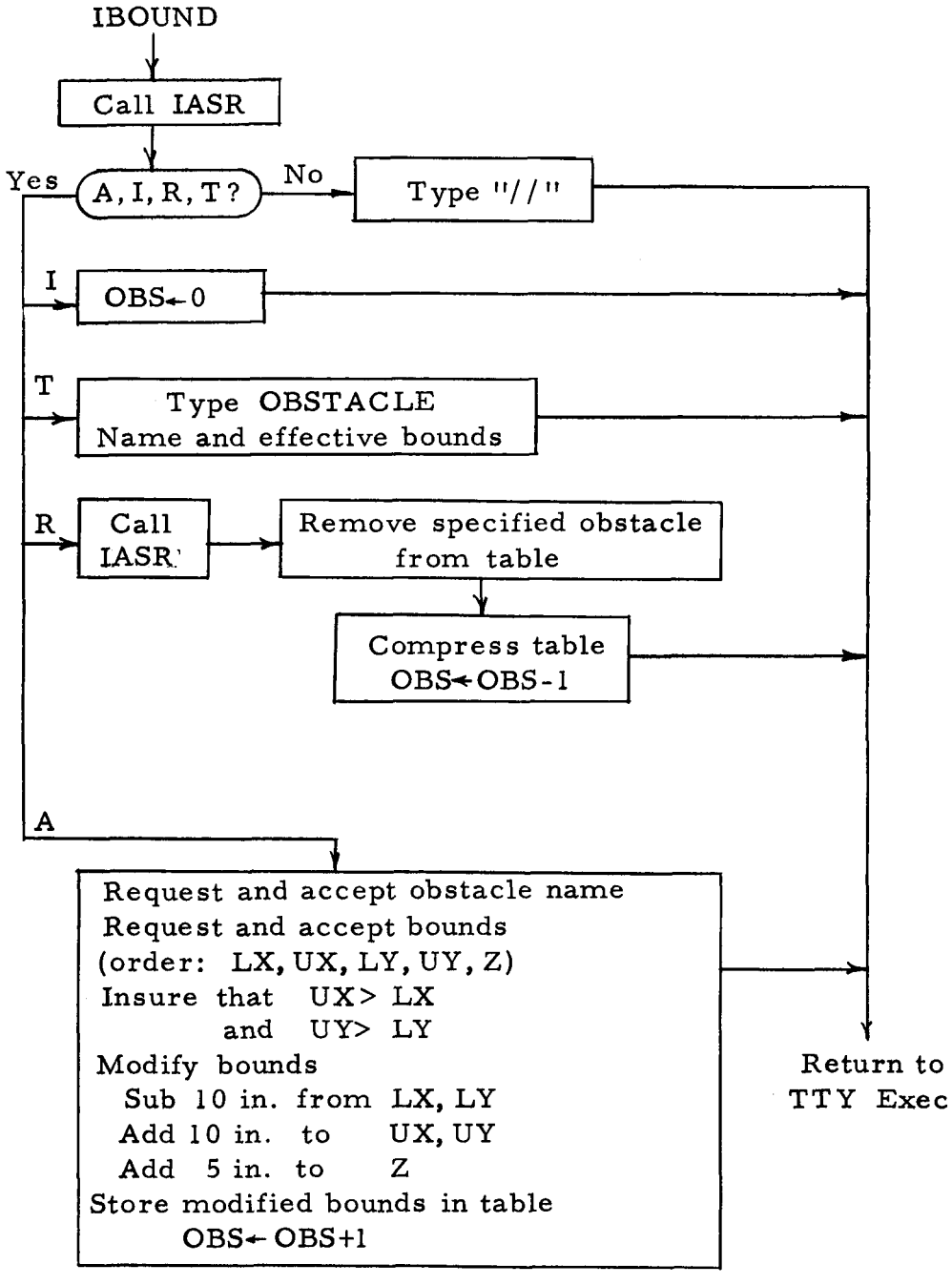


Figure 3-24. IBOUND Flow Diagram

3.3.4.3.2 Path Simulation and Collision Detection

The collision detection subroutines accept the initial and final manipulator configurations and project the path of the three pivot points and the hand. The paths are divided into increments of up to one inch and, at each increment, the positions of the hand and the pivots are compared with the effective obstacle bounds. If any point is within the constrained area, a special "collision" exit is taken to indicate that the FMV values are not acceptable. The manipulator can be driven in both the normal and rate-simulated modes, and since they frequently produce quite different paths, the two collision detection subroutines shown in Figures 3-25 and 3-26 have been written to test each mode.

When an obstacle is encountered, it is possible for these routines to be executed many times for a single command. Thus, in order to minimize the effective computation time when an obstacle is likely to be encountered, the testing of the path is staggered such that the routine first checks the mid-point, next the 1/4 and 3/4 points, then the 1/8, 3/8, 5/8, and 7/8 points, etc. until an obstacle is encountered or until the path has been sufficiently segmented to insure that no obstacles will be hit.

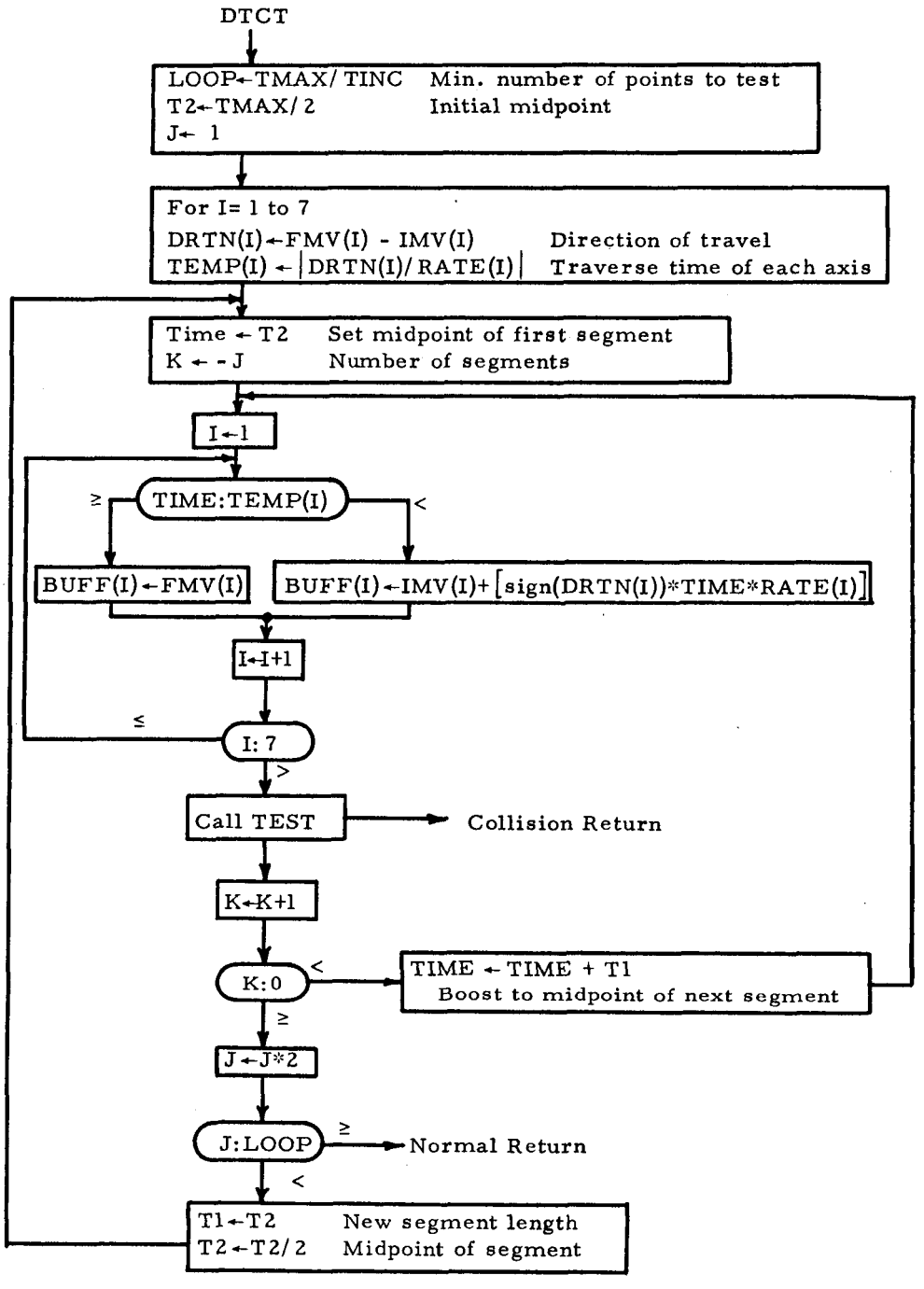


Figure 3-25. DTCT Flow Diagram

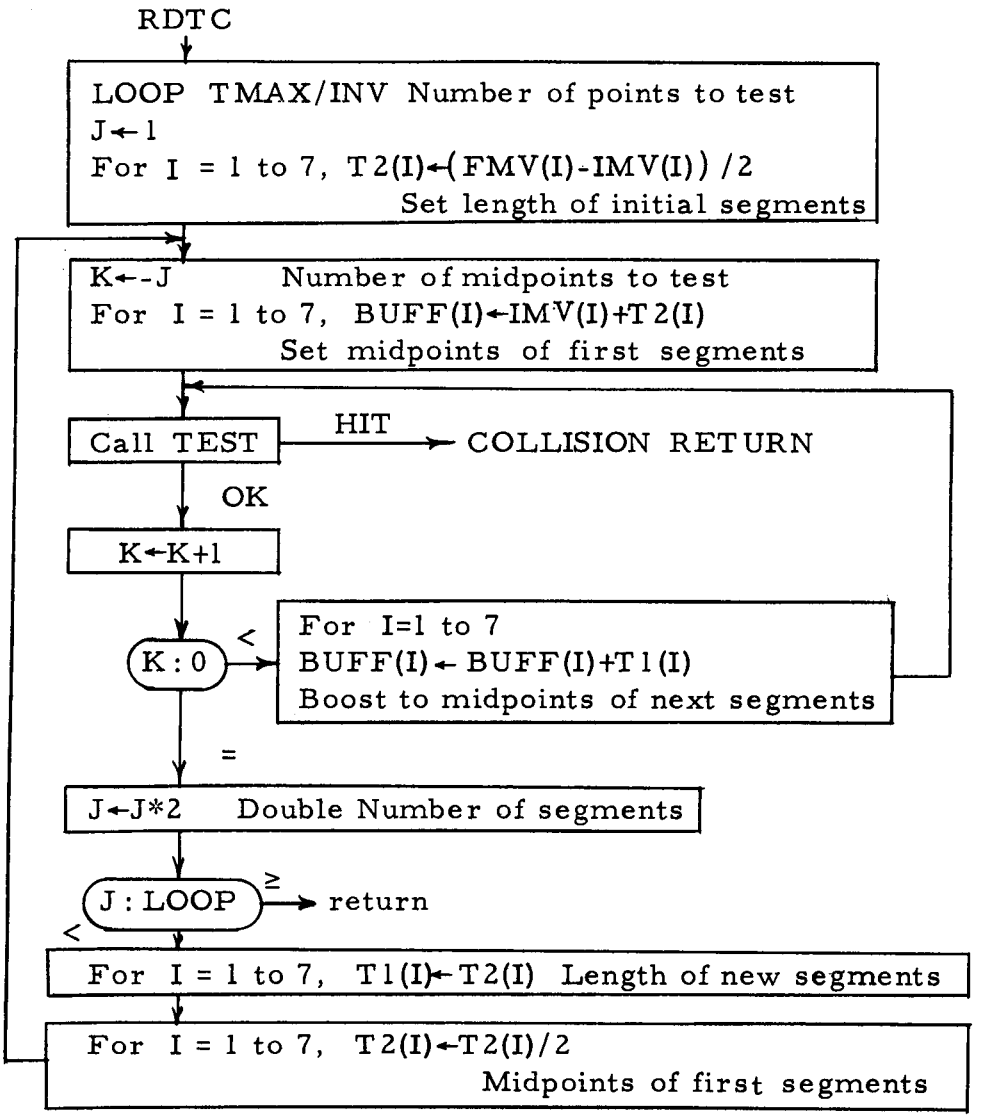


Figure 3-26. RDCT Flow Diagram

3.3.4.3.3 Obstacle Evasion

If a collision is detected, several efforts are made to evade the obstacle without requesting assistance from the operator. First, the straight line connecting the initial and terminal hand positions is tested for an intersection with the X and Y effective bounds of all obstacles. If no intersection is detected, "FLAGGT" is called to search the Grid Time Table for other possible terminal manipulator configurations, then "PHCK" is used to test the corresponding paths for obstacles. The table search continues until the table is exhausted or until a configuration is found which does not result in a collision. If an acceptable set of variables are found, they are used to actuate the manipulator; otherwise, the operator's assistance is requested. If the straight line connecting the initial and terminal hand positions intersected with a set of obstacle bounds, the "SUBG" routine is called.

The SUBG routine can direct the manipulator around only one obstacle at the present time and requests assistance if more than one obstacle was intersected. Further extensions of the routines were deferred until a sensory feedback loop is added to the system and a more comprehensive obstacle evasion technique can be developed.

When only one obstacle is encountered, the SUBG routine determines a set of intermediate hand variables which direct the manipulator over or around the obstacle, computes the approximate transit time required by each path, and selects the most promising one to output to the manipulator. A sketch of the three possible paths is shown in Figure 3-27. The hand variables at each of the intermediate points are selected as follows:

1. Over the obstacle

   a. First intermediate goal
      HX, HY = X, Y values of first intersection point
      HZ = top of obstacle plus one inch
      WP = IWP+(FWP-IWP)/3
      SR = ISR+(FSR-ISR)/3

   b. Second intermediate goal
      HX, HY = X, Y values of second intersection point
      HZ = top of obstacle plus one inch
      WP = IWP+2(FWP-IWP)/3
      SR = ISR+2(FSR-ISR)/3

2. Around the obstacle
   Paths are tested in both a clockwise and a counter-clockwise direction around the obstacle. The manipulator is directed to each corner in turn until the path to the desired hand position is clear. If N intermediate goals must be specified, the values of WP and Z at the M^{th} point will be

   $$Z = IHZ+M(FHZ-IHZ)/N$$
   $$WP = IWP+M(FWP-IWP)/N$$

   The values of HZ, HY and HZ are dependent upon the corner and are as shown in Figure 3-28.

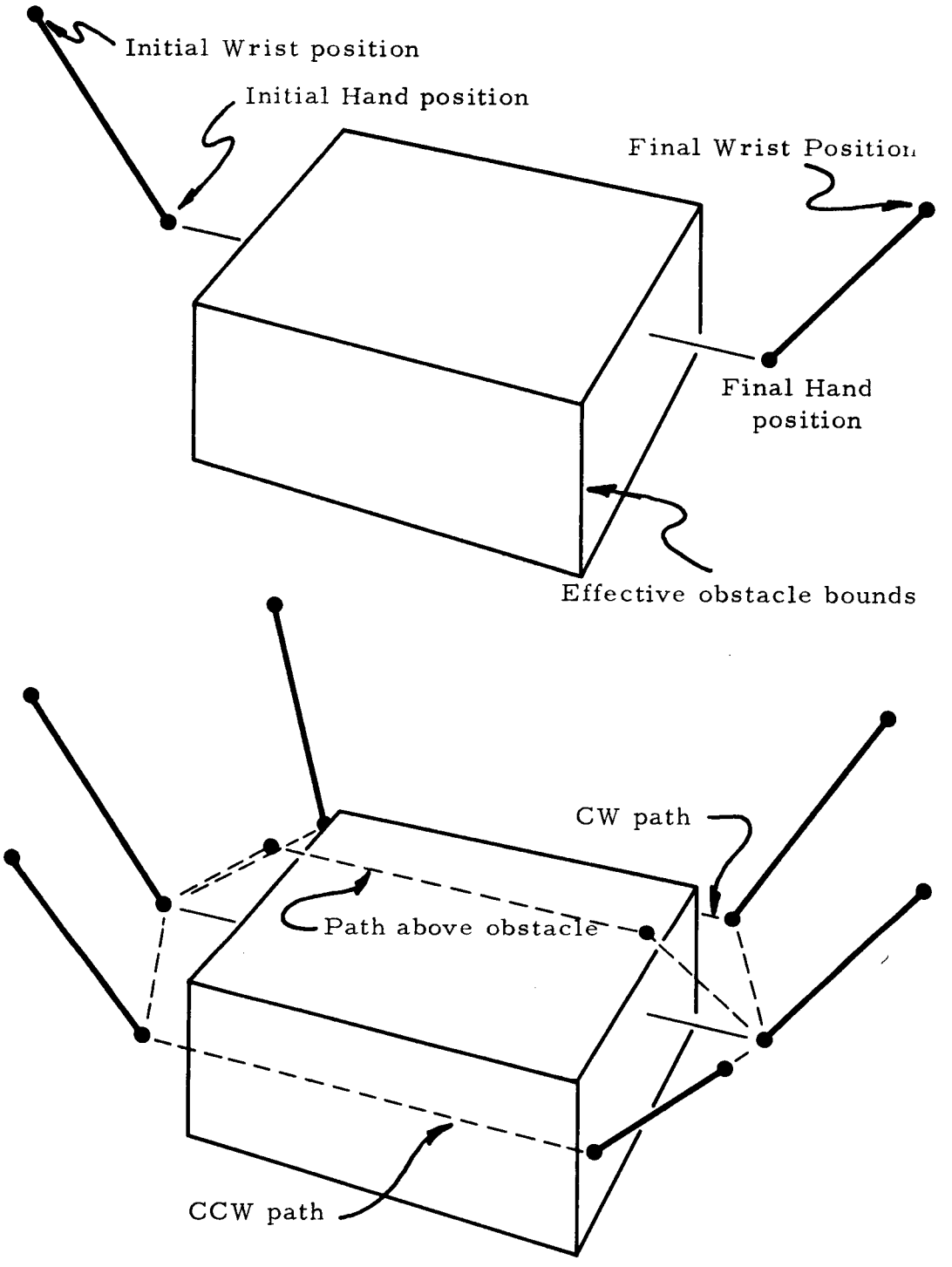


Figure 3-27. Obstacle evasion paths tested by SUBG

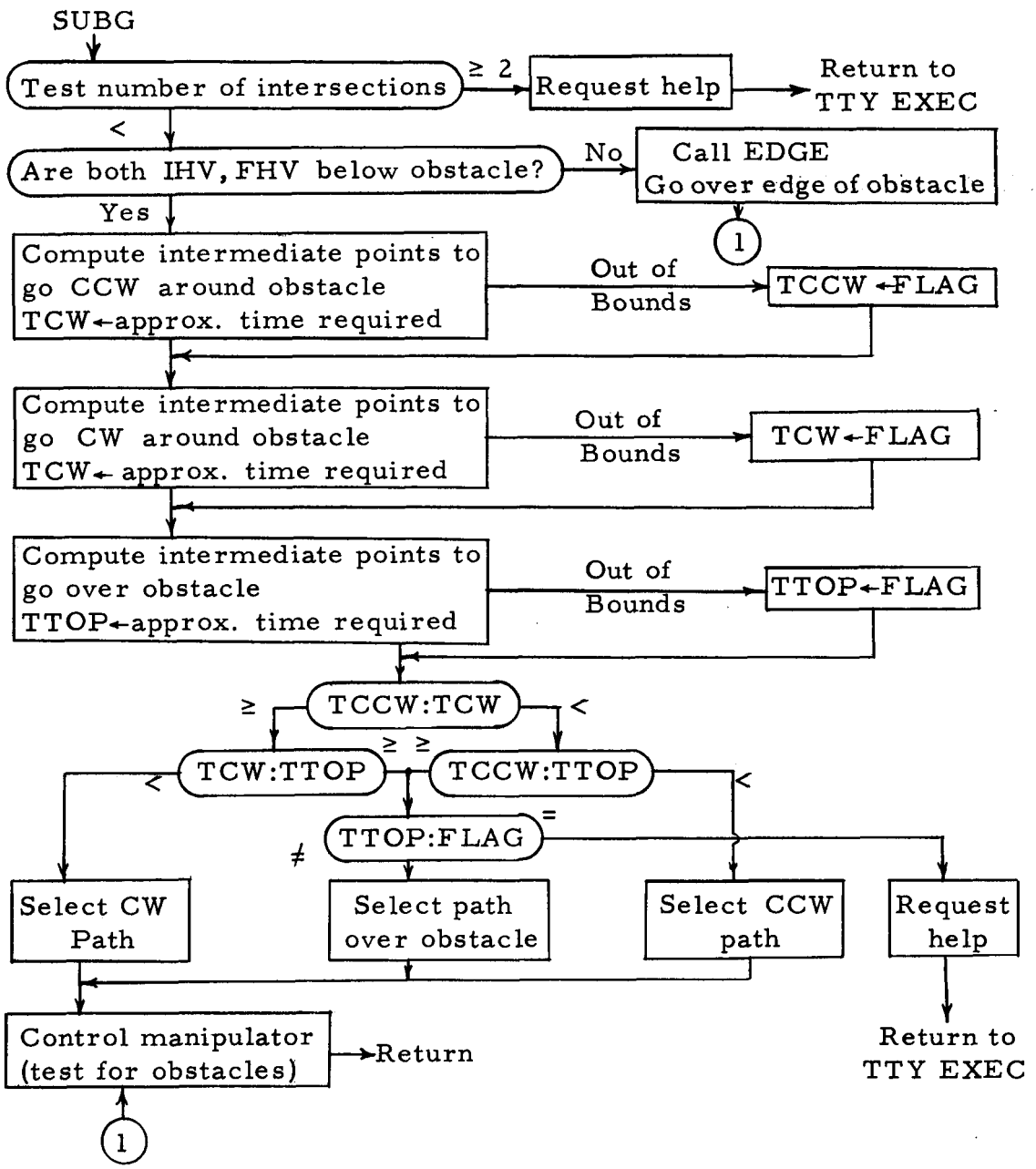
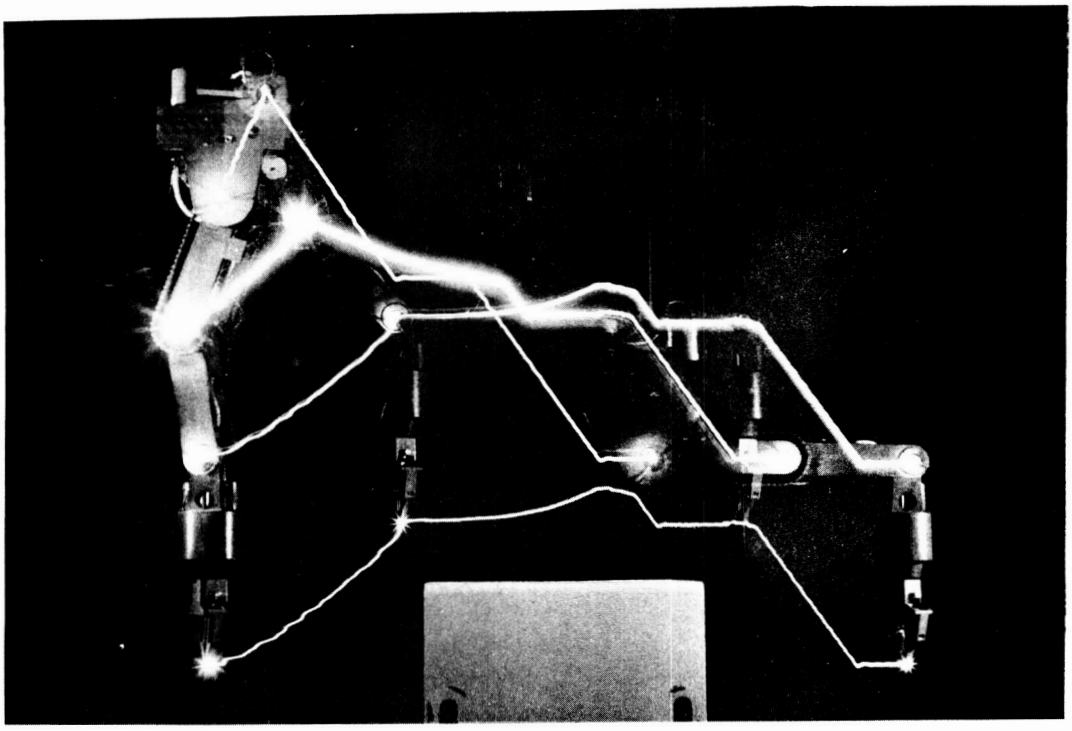
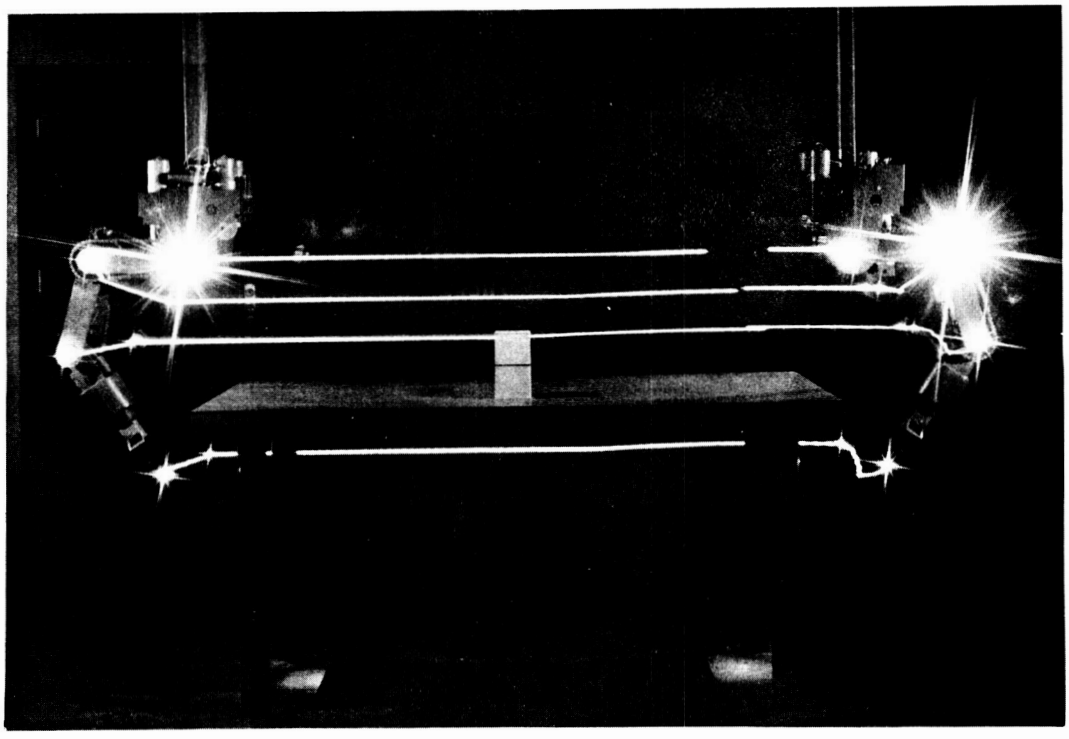


Figure 3-28. SUBG Flow Diagram

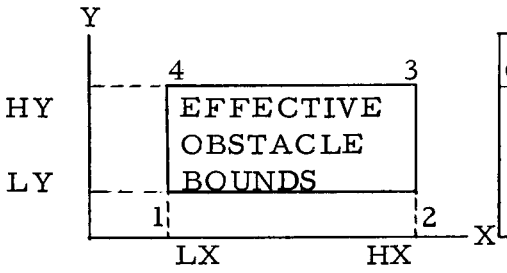


a. Path over obstacle



b. Path around obstacle

Figure 3-29. Photographs of manipulator paths generated to avoid obstacles



Corner	HX	HY	SR
1	LOX-1	LOY-1	45
2	HIX+1	LOY-1	135
3	HIX+1	HIY+1	225
4	LOX-1	HIY+1	315

Figure 3-30 Values of HX, HY, and HZ for Directing the Manipulator Over or Around an Obstacle

A flow diagram of the "SUBG" routine appears in Figure 3-28 and the photographs in Figure 3-29 demonstrate the path followed by the manipulator when directed over and around obstacles by the SUBG routine.

3.4 Automatic Mode

The Automatic Mode allows the operator to record or pre-program a complete task, segments of a task, and highly repetative operations. When it is desired to execute one of these tasks, program control is transferred from the TTY Exec to the AUTO Exec. The AUTO Exec then executes the recorded operations without any further operator assistance.

As shown in Figure 3-2, the AUTO Exec is permitted to call the four control algorithms described in Section 3.3 after transferring all information required by the algorithm to the proper

buffers. The "WAIT" command requires the AUTO Exec to wait for a specified time before executing the next command, and the TTY Return command transfers computer control back to the TTY Exec.

The remaining two commands allow the hand variables to be related to a specific object in the manipulator task area rather than specifying them in terms of the manipulator coordinates. The programs, described in Section 3.4.3 demonstrate that a general purpose computer can be utilized to preprocess specified information before calling the basic control algorithms.

The programs which have been implemented require the use of paper tape for recording and executing a task. In the future, they should be expanded to allow highly repetative tasks to be named and stored in the core memory of the computer. It would also be possible to store pre-programmed information on cards or magnetic tape.

3.4.1 Record Data -- Punch Paper Tape

This routine allows the operator to program a series of operations for use by the AUTO Exec routine. The information

for the control algorithms can be specified by the operator via the TTY console or the current position of the hand or manipulator can be recorded as an operation is executed by the operator under manual or semi-automatic control.

The paper tape is punched in blocks as shown in Figure 3-31. Each block consists of a start code, a flag to indicate the operation to be performed, the number of data words following, the data, and a check sum. The next character causes the block to be ignored if it is not blank. The flow diagram of the "PUNCH" routine appears in Figure 3-32.

3.4.2 AUTO Exec -- Paper Tape Read Routine

This routine reads a block of paper tape, transfers the recorded information to the proper buffers, then calls the appropriate control algorithm or subroutine. Unless one of the exit conditions listed in section 3.2.2 occurs, a new block is read and processed upon completion of the previous command. A special entrance to the routine allows the previous operation to be repeated without reading a new block of tape in case a special condition transferred control to the TTY Exec before completing the command. A flow diagram of the routine appears in Figure 3-33.

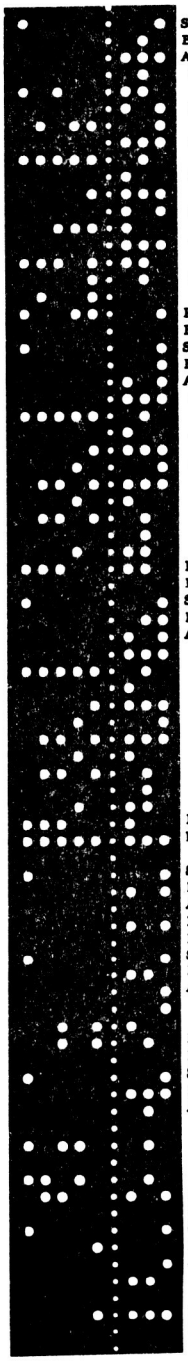


Figure 3-31. Paper Tape Block Structure

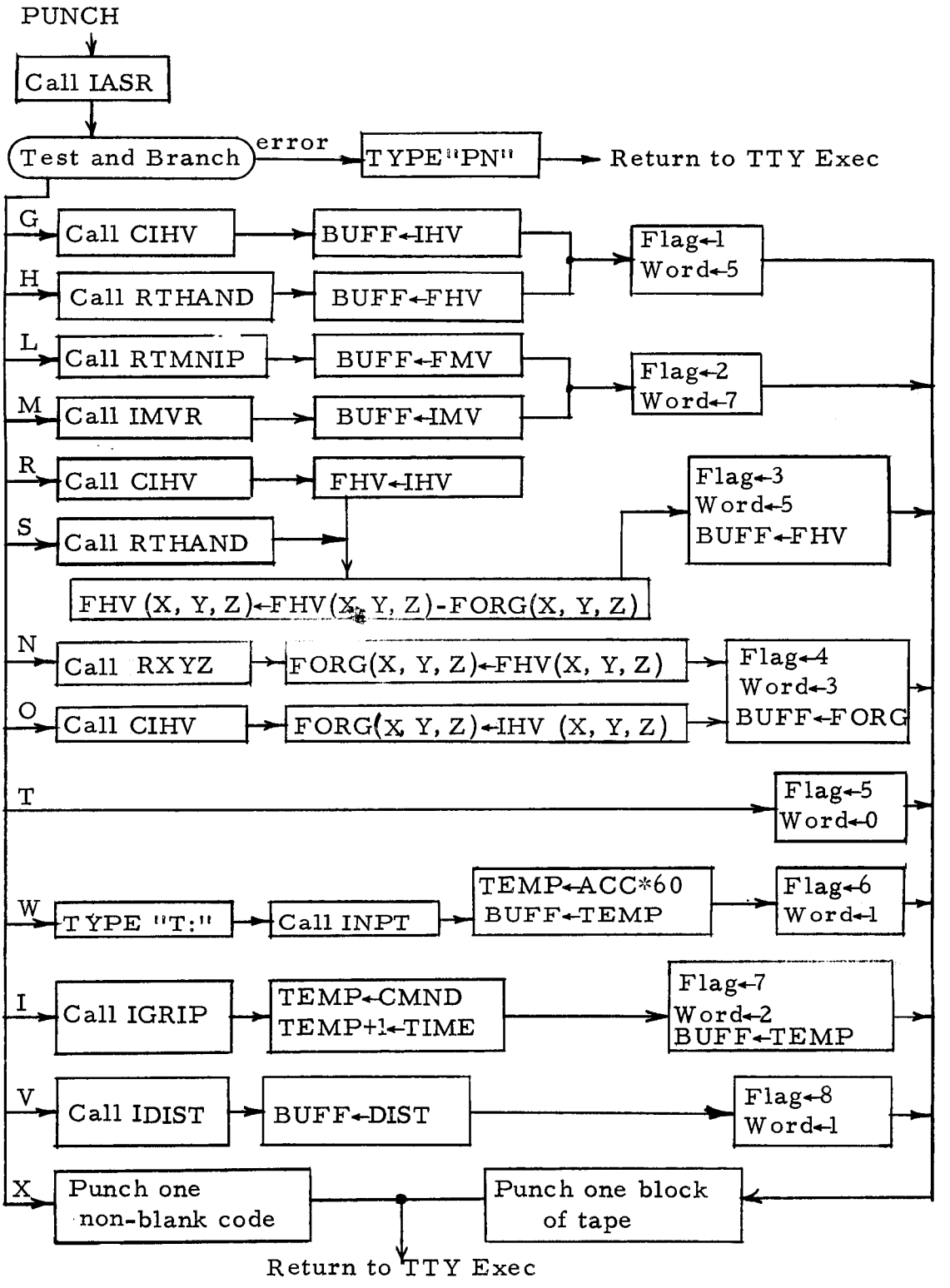


Figure 3-32. PUNCH Flow Diagram

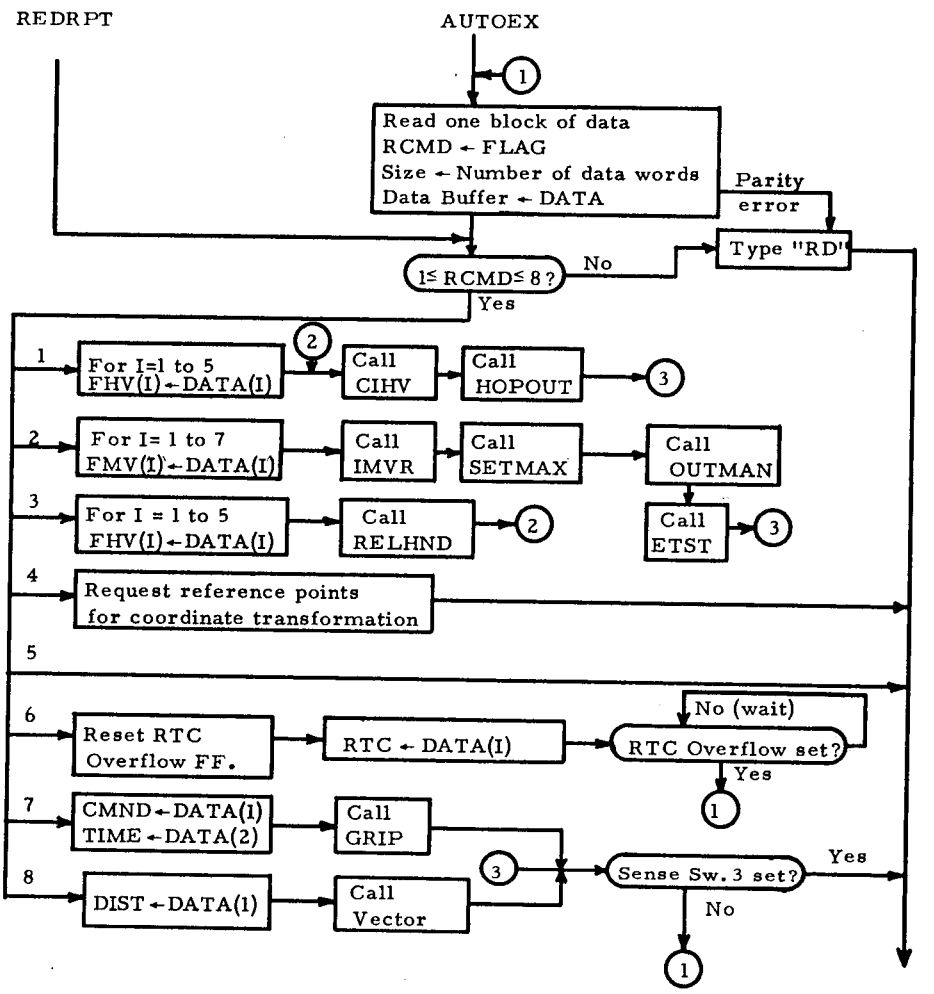


Figure 3-33. AUTOEX Flow Diagram

3.4.3 Coordinate Transformation

The coordinate transformation routines extend the utility of the automatic mode. These routines permit the specification of hand variables relative to a specific object rather than in terms of the manipulator coordinates. For example, this allows the preprogramming of tasks such as the removal of various components of a nuclear rocket engine without specifying the exact location of the engine. After the engine is placed in the disassembly area, several reference points (used in the preprogramming phase) are specified in terms of the manipulator coordinates and a transformation matrix is computed. All subsequent "relative" references to the object are automatically transformed to the manipulator coordinates. The transformation will occur only when the special "relative hand variable" block is encountered; consequently, it is permissible to intermingle all options available under fully automatic control.

The methods of recording and executing these operations are presented in the following sections.

3.4.3.1 Recording Relative Hand Variables

The relative hand variables may be programmed directly from a set of drawings or by recording the position of the manipulator as it is being controlled using the manual or semi-automatic mode. In either case, it is necessary to define a cartesian coordinate system relative to the moveable object. The location of the origin and the three reference points should be marked since their positions must be specified in terms of the manipulator coordinates after the object is placed in the manipulator task area. The subroutines written require the three reference points to be located on the positive X, Y, and Z axes eight inches from the origin. Only a slight modification would be necessary to change the distance requirement; however, as explained in Section 3.4.3.3, several additional subroutines would be required to allow arbitrary placement of the reference points.

During the recording operation, the location of the origin of the object coordinate must first be punched (specify P_o=(0, 0, 0) if entering the positions via the TTY console). When the actual manipulator positions are recorded, the X, Y, and Z axes of the object coordinate system must be parallel to those of the manipulator coordinate system.

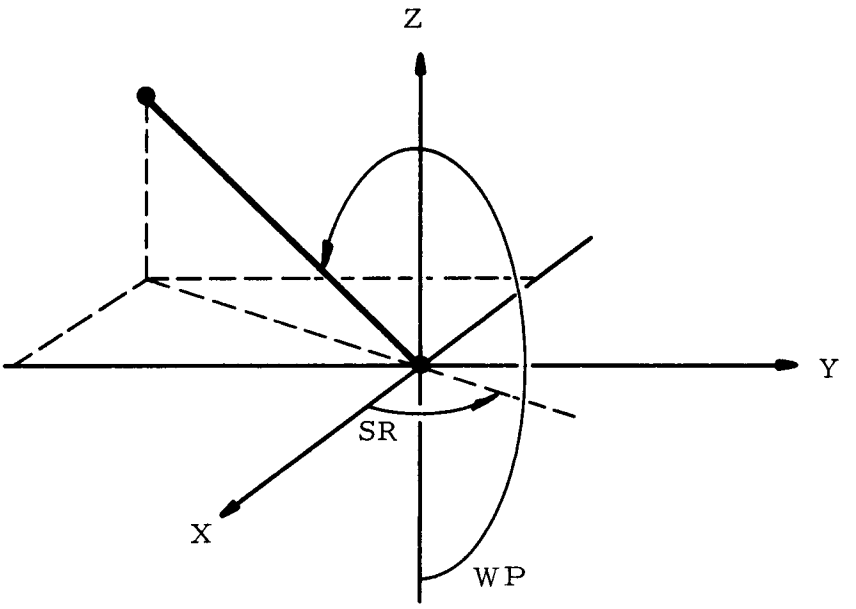
3.4.3.2 Executing "Relative Hand Data"

Before executing any relative hand variables, it is mandatory that the operator specify the positions of the origin and the three reference points of the object coordinate system in terms of the manipulator coordinate system. Then the operator must direct the computer (via the TTY Exec) to compute the coordinate transformation matrices as described in the next section.

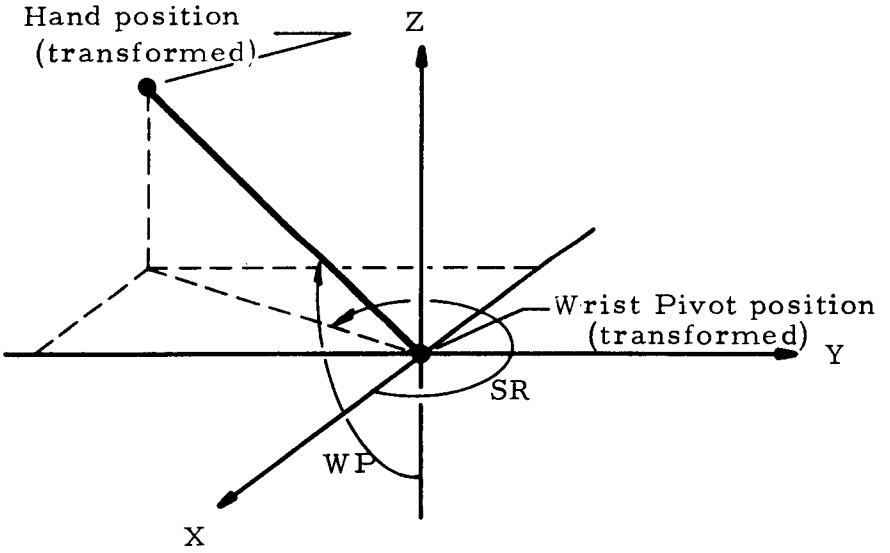
Whenever a "relative hand variable" block is subsequently encountered, the (X, Y, Z) position of the hand is transformed from the object coordinate system to the manipulator coordinate system. In order to transform the Azimuth and Elevation values, the recorded angular values are used to compute the wrist pivot position relative to the object coordinates. This position is then transformed and used to define WP and SR values in terms of the manipulator coordinates. As shown in Figure 3-34, there are two possible combinations of the angles. To form a unique solution the value of WP is restricted to

$$0 \leq WP < 180^{o} \qquad (9)$$

Flow diagrams of the transformation routines are shown in Figure 3-35 and 3-36.



a. WP > 180°



b. 0 ≤ WP ≤ 180°

Figure 3-34. Transformation of WP and SR

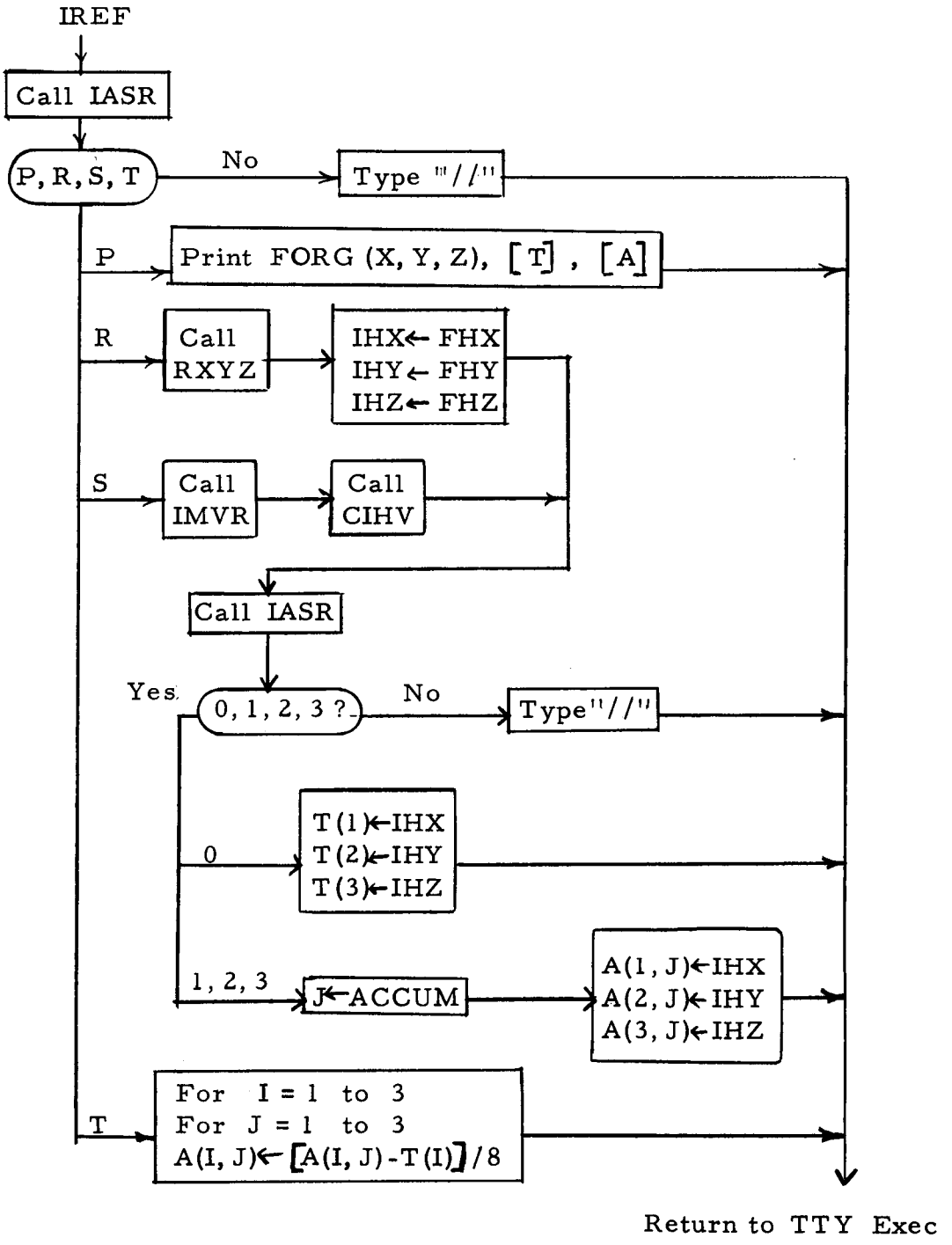


Figure 3-35. IREF Flow Diagram

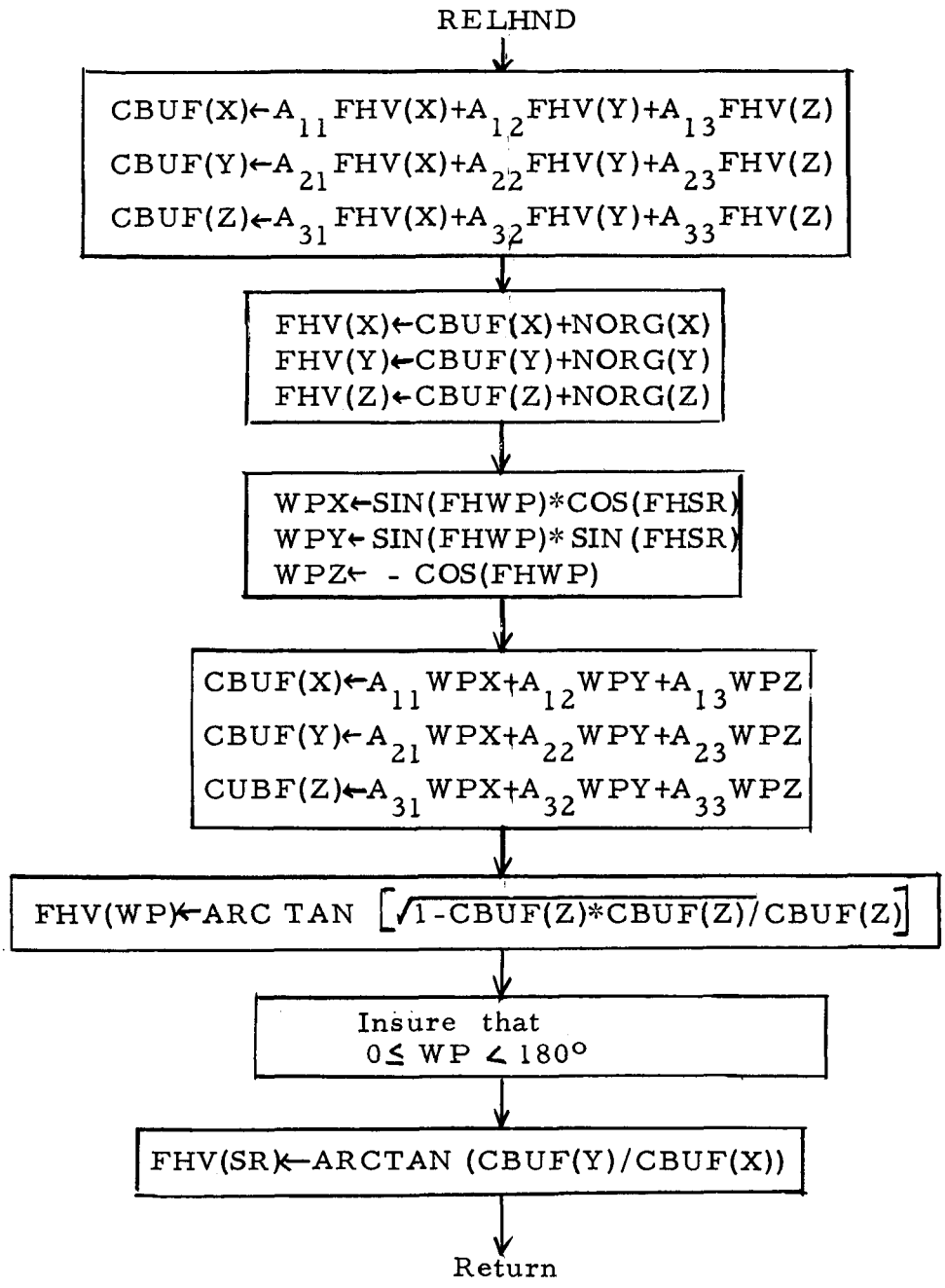


Figure 3-36. RELHAND Flow Diagram

3.4.3.3 Transformation Equations

An "object" cartesian coordinate system is defined and all "relative hand positions" are specified in terms of this coordinate system during the recording process. After the object is placed in the task area, this coordinate system is rotated and translated to coincide with the manipulator coordinate system. A point P(X, Y, Z) measured in terms of the object coordinates is then transformed to the position P'(X', Y', Z') measured in terms of the transformed (manipulator) coordinates. The standard transformation equation is

$$[P'] = [A]\,[P] + [T] \tag{10}$$

where the matrices $[A]$ and $[T]$ represent the rotation and translation of the coordinate system respectively.

In order to determine the nine coefficients of $[A]$ and the three coefficients of $[T]$, the origin, P_0, and three reference points are marked on the object. In the routines implemented these reference points in terms of the object coordinates are located at

$$\begin{aligned} P_0(X_0, Y_0, Z_0) &= (0, 0, 0) \\ P_1(X_1, Y_1, Z_1) &= (8, 0, 0) \\ P_2(X_2, Y_2, Z_2) &= (0, 8, 0) \\ P_3(X_3, Y_3, Z_3) &= (0, 0, 8) \end{aligned} \tag{11}$$

After the object is placed in the manipulator area, these four points are located and their positions are specified by the operator in terms of the manipulator coordinates. Since the values of P and P' are thus known for these four reference points, the coefficients can then be determined. From Equations (10),

$$
\begin{aligned}
[T] &= [P'_0] + [A]\ [P_0] \\
[A]\ [P_1] &= [P'_1] - [T] \\
[A]\ [P_2] &= [P'_2] - [T] \\
[A]\ [P_3] &= [P'_3] - [T]
\end{aligned}
\qquad (12)
$$

Substituting equations (11) into (12),

$$
[T] = \begin{bmatrix} X'_0 \\ Y'_0 \\ Z'_0 \end{bmatrix} \qquad (13)
$$

and

$$
[A] = 1/8 \begin{bmatrix} X'_1 - X'_0 & X'_2 - X'_0 & X'_3 - X'_0 \\ Y'_1 - Y'_0 & Y'_2 - Y'_0 & Y'_3 - Y'_0 \\ Z'_1 - Z'_0 & Z'_2 - Z'_0 & Z'_3 - Z'_0 \end{bmatrix} \qquad (14)
$$

Note that it would be possible to select any set of reference points which would produce a non-singular matrix; however, unless they are located along the axes, any other choice would require

the solution of three sets of simultaneous linear equations of three variables.

3.4.3.4 Future Extensions

Several features can be added to these routines to extend their versatility. First, it is normally inconvenient to mark the three reference points as required by Equations (11). With the addition of a program to solve simultaneous equations, it would be possible to specify an arbitrary set of reference points.

Second, when recording relative hand variables by recording the manipulator position, it would be possible to initially specify the three reference points and compute a transformation matrix for recording the data -- this would allow the object coordinates to be rotated with respect to the manipulator coordinates before recording the "relative" variables.

Third, it would be possible to create a set of tables to accommodate several objects to which the hand variables can be related. During recording and execution, it would then be necessary to specify the object to which the variables are to be related.

3.5 Software A/D Conversion

The "SAD" subroutine, Figure 3-37, demonstrates how the manipulator position can be determined with the aid of a software technique rather than using combination buffer/bidirectional counters. The program uses a binary search method. A trial value cuts the possible range in half and the value of the error indicates which half to subdivide further. Since only ten bits are used to measure the position, a maximum of ten trials is necessary to determine the exact position. As mentioned in Chapter II, the measurement speed is limited by the delay introduced by the active filters.

3.6 Miscellaneous Routines

Several routines have been written to enable the operator to test the performance of the system, control special modes, and execute special functions. A few of these are described below.

3.6.1 Type Position Information

The "TYPE" subroutine enables the operator to request one of the following types of information:

1. Present manipulator variables (scaled)
2. Present hand variables (scaled)
3. Contents of FMV (final manipulator variables)
4. Contents of FHV (final hand variables)
5. Exact contents of the manipulator buffers (unscaled -- in octal).

In the first two cases, the computer first inputs the current values of the manipulator and then scales them. In the second case, the hand variables are computed from the manipulator variables and the hand geometry.

3.6.2 Output Unscaled Manipulator Values

This routine allows the operator to type (in octal) the desired values of all seven manipulator buffers. These are then transferred to the manipulator logic. The routine is intended primarily for maintenance.

3.6.3 Adjust "Effective" Hand Length

The "Hand Position Control" algorithm actually considers an "effective" hand position which can be modified by the operator to compensate for changes in length of various objects or tools held

in the hand. This routine allows the operator to specify the distance which the object protrudes beyond the tip of the finger. The value should be specified initially and each time an object is grasped or released.

3.6.4 Control Special Modes

Several options allow the operator to force the Manual Mode, force the Automatic Mode, set the SAD Mode, or set the Rate-Simulation Mode. The first three options are primarily included for system maintenance.

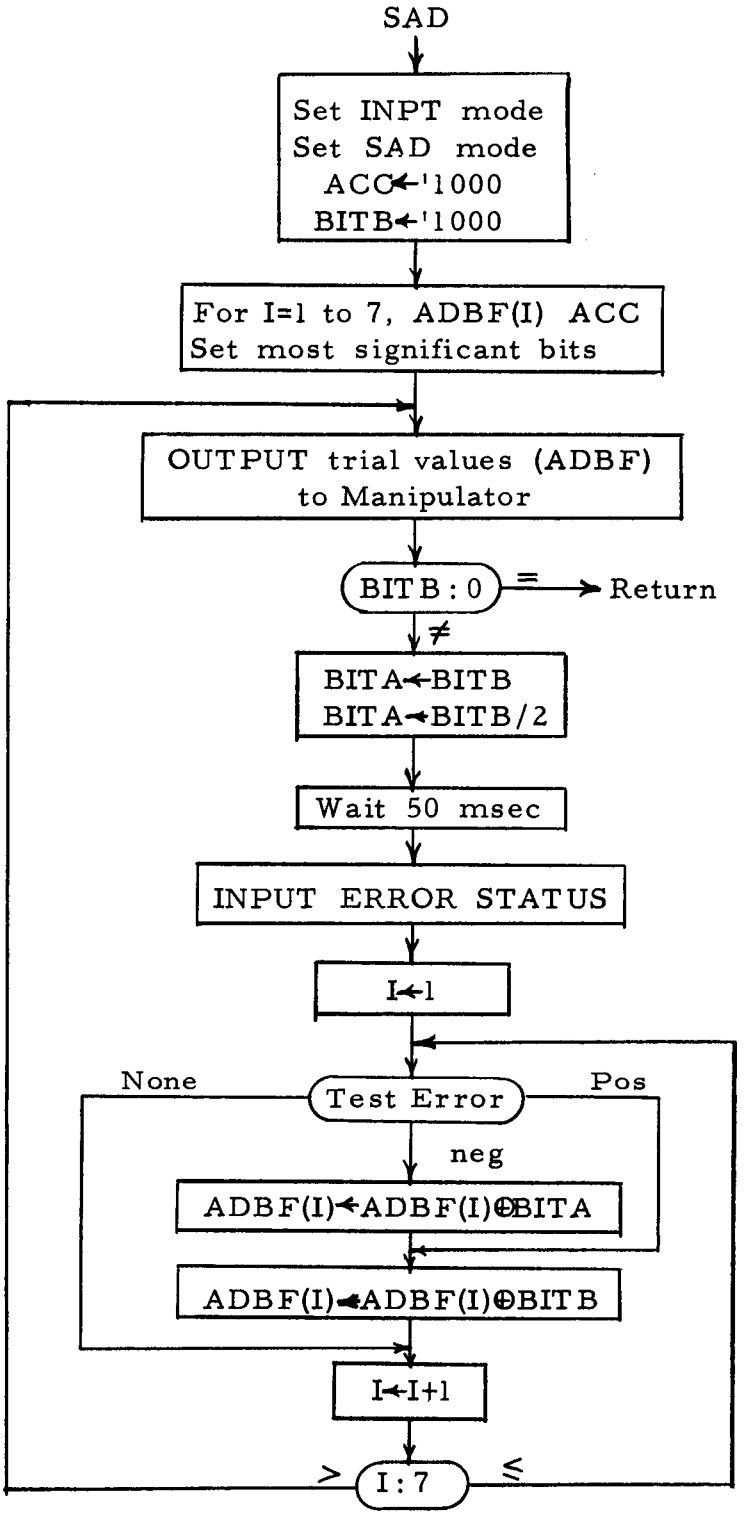


Figure 3-37. SAD Flow Diagram

Chapter IV

SUMMARY AND FUTURE TASKS

4.1 Introduction

In order to improve the controllability of a remote manipulator, a scheme was proposed in which a digital computer accepts a minimum amount of information from the operator and uses it to drive the manipulator to the proper destination or along a specified path. The experimental model was designed to allow the operator to use either the computer control mode or the manual control mode. Under computer control, the two executive programs allow the operator to execute one command at a time via the teletype console or to use the fully automatic mode which reads and executes an unlimited number of operations without operator intervention.

The initial phase of the project was to demonstrate the feasibility of the computer-aided-control scheme. The results indicate that it is desirable to continue the development of a more practical system. The next phases of the project should include

the following tasks:

1. The addition of sensory feedback.

2. The addition of a special operator console and/or a master arm unit.

3. Improvement of analog to digital conversion speeds.

4. Modification of the algorithms already developed as required by the system improvements.

5. Expansion of the algorithm library.

6. Experimental evaluation of operator's performance using (a) the manual modes and (b) the computer-control mode.

4.2 Achievements

As a side benefit, the replacement of the relay motor controls by the SCR drive amplifiers enhanced the system performance even under manual control. The principal advantage is that instead of allowing only discrete (2 or 3) motor speeds, the SCR amplifiers drive the motors at speeds proportional to the displacement of the levers on the control panel.

The addition of the computer to the control loop produces two obvious improvements over the manual "rate-control" technique. First, it is possible for the computer to actuate all motors simultaneously and at their maximum speed since it can

accurately predict the path which results. In contrast, it is not only difficult for an operator to operate seven levers simultaneously but would most likely produce catastropic results. Second, as demonstrated by the vector approach routine, it is relatively easy for the computer to control the path of the manipulator. Conversely, it is virtually impossible for the operator to perform circular, straight-line, or more complicated path control operations which require the actuation of two or more motors simultaneously.

The optimization/obstacle avoidance routines most clearly demonstrate the value of the computer-control technique. The only information which the operator must specify is a command to select this algorithm and the desired terminal hand variables. The computer then selects an "optimum" terminal configuration, insures that it does not require motion through any obstacle bounds, and, if necessary, automatically selects a set of intermediate points to direct the manipulator around the obstacle.

The AUTO Exec routine demonstrates that it is possible to record any number of commands in advance and then execute them in sequence without operator intervention. Furthermore, the amount of data to be recorded is minimized. For example, only

a minute fraction of the data required by the Case Research ARM AID to perform a task can perform an equivalent task using the computer-control scheme.

The coordinate transformation routines demonstrate that special features can easily be incorporated into the computer-control technique.

The versatility of the computer is demonstrated by the fact that the computer used in the experiment is also used for a wide variety of control experiments, for general data processing, and for the special simulation problems. As a result, the computer used to control the manipulator can also be used to perform other tasks when the manipulator is idle. For example, in space applications it would be feasible to use the same computer for navigation, data reduction, and controlling a manipulator.

Another interesting result of the investigation is that the computer consumes comparatively little time performing the computations and input-output operations necessary to execute a task. For example, the optimization routine requires less than 80 msec to compute an optimum configuration; moreoever, even when it is necessary to perform a lengthy series of

computations to select a set of intermediate points in order to avoid an obstacle, they seldom require more than two seconds. The rest of the time is simply wasted waiting for the operator to specify a task or waiting for the manipulator to reach the specified position.

As a result, it appears that it is quite possible to either time-share the computer with other tasks or to use the same computer to control several manipulators.

4.3 Future Tasks

Two tasks were deferred until the completion of the initial phase of the project. These concern the man-machine interface and sensory feedback from the manipulator to the computer.

4.3.1 Man-Machine Interface

Currently, all communications between the operator and the computer are handled by the teletype unit. Although this has been adequate for experimental purposes, it is unsatisfactory in a practical system. The principal difficulties are that it is awkward and time consuming for the operator to type position information and also that the operator is required to relate all

information to a predefined coordinate system. In addition, it is desirable to improve the manual control technique to eliminate some of the problems encountered in path control operations. The following possibilities should be considered to reduce these difficulties:

1. Master/slave unit.
2. Calibrated position control knobs on master panel.
3. Joystick positioning.
4. Voice control.

The technique used to control the position of the manipulator can easily be adapted to allow master-slave control. This would entail the construction of a master unit such that its motions are identical to those of the manipulator. A set of potentiometers can then be mounted to each axis of the master as they were mounted to the manipulator and they can be wired to the control logic as shown in Figure 4-1.

With another slight modification, the same master unit can be used to specify the desired position of all or part of the manipulator variables. In this case, the A/D conversion technique is used to measure the position of the master unit. This is demonstrated in Figure 4-2.

The second possibility enumerated above is similar to the first except that the potentiometers are calibrated and mounted on a panel rather than being attached to the master unit.

The third actually involves a combination of the first two techniques. A joystick would be used to control the linear motions and either a slave arm or a set of knobs can control the arm or hand orientations.

The last technique demands the development of a device which can recognize speech in real time. This task is the subject of another investigation in the Digital Systems Laboratory.

4.3.2 Sensory Feedback

Since there is no sensory feedback in the experimental system, there is no way the computer can recognize any obstacles. Thus, without operator intervention, it is quite possible for the manipulator to be driven into another object or even into itself. This places most of the burden on the operator if all collisions are to be prevented and requires visual observance of all operations. Since it is desired to minimize the operator's control of the manipulator, this situation is intolerable.

A partial solution to the problem has been achieved with the use of the obstacle avoidance algorithm, however, this is useful for stationary obstacles only, requires the operator to enter the bounds in advance, and must consider slightly enlarged bounds. A much better solution is to mount a set of sensors around the manipulator which will detect any obstacles in close proximity to the manipulator. If they are mounted so that the direction of the obstacle can be determined, the computer cannot only recognize the obstacles but take evasive action to avoid it without any intervention by the operator.

A second set of sensors should also be placed in the hand to control the force used to grip an object. To allow fragile articles to be handled, the force should be no greater than required to prevent it from slipping out of the hand.

With the addition of one or both sets of sensors, it is desirable to modify the current algorithms and perhaps formulate several more to take full advantage of the increased system capabilities.

4.3.3 Analog to Digital Conversion

It was noted in Section 2.7.2 that the time lag of the active filter greatly effects the A/D conversion time. Since the manipulator logic is in the manual mode at the time of the conversion and the error is not used to control the SCR amplifiers, it is likely that the active filter can be by-passed during the conversion; however, it will still be necessary to provide high frequency filtering to attenuate noise from the reed relays in the D/A converter.

4.3.4 Algorithm Modifications and Additions

The algorithm library should be expanded to include alternate optimization criteria (e.g., minimum energy or minimum momentum), additional path control routines (e.g., circular arc paths), and the automatic mode should allow highly repetative tasks to be stored and processed using core memory rather than paper tape. In addition, the adaptive techniques under development by the Cybernetics Systems Group to improve control of the Case Research ARM AID should be studied for possible application to control the remote manipulator.

4.3.5 System Evaluation

After the addition of sensory feedback and a more practical man-machine interface unit, a set of experiments should be formulated and executed to evaluate the computer-control technique and to compare an operator's performance using both the manual and the computer control techniques.

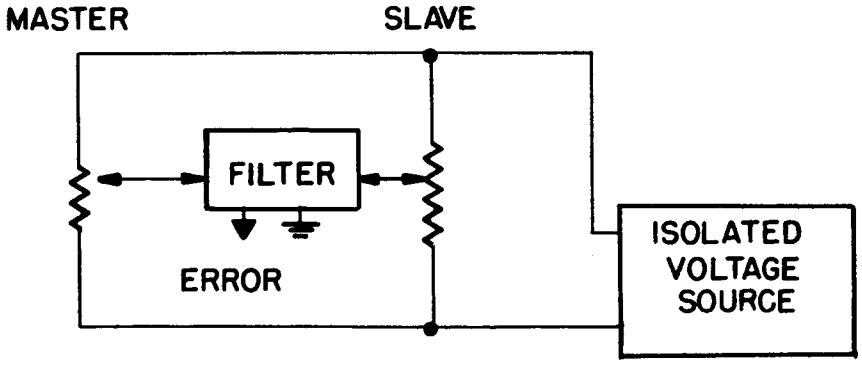


Figure 4-1. Master-Slave Position Control

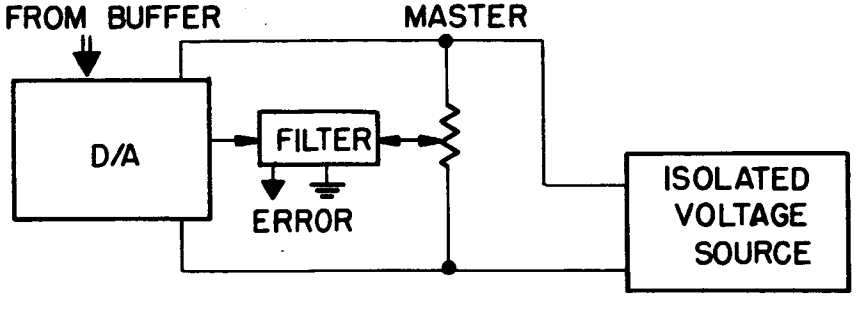


Figure 4-2. Master Position Measurement

BIBLIOGRAPHY

1. Beckett, Jon T., Detailed Documentation of the Subroutines Written for the DDP-116 to Control the Case Experimental Manipulator, Digital Systems Laboratory, Case Institute of Technology.

2. Correll, Robert W., and Maurice J. Wijnschenk, Design and Development of the Case Research Arm Aid, Report EDC 4-64-4, Engineering Design Center, Case Institute of Technology.

3. Hammond, P. W., A Hybrid Servo-Positioning System for a Computer Controlled Manipulator, Report 1-66-41, Digital Systems Laboratory, Case Institute of Technology.

4. Honeywell, Incorporated, Computer Control Division, DDP-116 Programmers Reference Manual, February, 1966.

5. Proceedings of the 1964 Seminars on Remotely Operated Special Equipment, United States Atomic Energy Commission, Reports CONF-640508 and CONF-641120.

6. Rarich, Thomas D., Development of SCM-1, A System for Investigating the Performance of a Man-Computer Supervisory Controlled Manipulator, M.S. Thesis, Massachusetts Institute of Technology, 1966.

7. Taylor, Richard J., A Digital Interface for Computer Control of a Remote Manipulator, Report 1-66-49, Digital Systems Laboratory, Case Institute of Technology.

Appendix I

MANIPULATOR COORDINATE SYSTEM AND SCALING OF MANIPULATOR VARIABLES

The programs written for the manipulator assume a cartesian coordinate system for the linear (X, Y, Z) variables. The origin is placed at the midpoint of the three manipulator axes. The SP, EP, and WP angles are measured from the center of symmetry (e.g., relative to the negative Z axis). The color coding on the manipulator is used to specify the sign of the pivot angles. The sign is positive when the dark (blue) color is on the upper half of the limb and it is negative when the light (red) color is on the upper half. The shoulder rotation angle is measured from the positive X axis to the arm when the pivots are positive (see Figure 2-3).

The operator is required to specify linear and angular variables in terms of inches and degrees respectively. Motor rates are specified in terms of feet per minute and revolutions per minute. The data is specified via the teletype console and both mixed fractions and integers are permitted. Negative values must be preceded with a minus sign. Positive numbers may be preceded with a plus sign but it is not required. For example, the numbers +2, 2,

2.0, -2, -2.0, .2, 0.2, are all permitted. The numbers are terminated by actuating the carriage return (CR) key. If any illegal numbers or characters are specified, a slash is automatically typed and the routines will wait until an acceptable number is typed.

All variables are converted to signed, scaled, 16 bit numbers. Linear distances and times contain eight integer bits and seven bits of fraction. Angular variables contain two integer bits and thirteen fraction bits (angles are converted from degrees to revolutions).

The position of each axis of the manipulator is represented by a ten bit binary number (MPB) which is slightly over zero at one extreme and slightly under $(1777)_8$ at the other extreme. These are related to the FMV and IMV buffers by Equations (A.1) and (A.2).

$$IMV(AXIS) = MPB(AXIS) - BIAS(AXIS) * SCAL(AXIS)/(1000)_8 \quad (A.1)$$

$$MPB(AXIS = FMV(AXIS) * (1000)_8/SCAL(AXIS) + BIAS(AXIS) \quad (A.2)$$

SCAL(AXIS) represents the value (in inches or fractions of a revolution) of the variable for a change of the manipulator position buffer (MPB) of $(1000)_8$. BIAS represents the zero point of each

axis and is $(1000)_8$ for all axes but SR (the SR bias is the value of the buffer when SR is zero). The BIAS and SCAL tables are loaded with the control programs.

Table A2.1
Teletype Commands

A Set GPI Output Mode

B Set GPI Input Mode

C* Modify axis ratios
(X:, Y:, Z:, SP:, EP:, WP:, SR:)

D Set SAD Mode

E Force Automatic Mode (Set GPI Output Mode, then SAD Mode)

F* Obstacle Table

I Initialize table (remove all obstacles)

T Type contents of obstacle table

A name, (LX:, HX:, LY:, HY:, Z) add an obstacle to the table

R name, remove obstacle from the table

G* Request and accept hand variables; execute hand variables control algorithm (X:, Y:, Z:, SP:, EP:, WP:, SR:)

H* Accept mode, axis name, and hand variables; execute hand variable control algorithm

I, $$X, Y, Z, W, R, T$$ Incremental mode

A, $$X, Y, Z, W, R, T$$ Absolute mode

Note: T = terminate list and R = shoulder rotation

I* Control hand grip and rotation
$T, 0, 1, 2, 3, 4, 5, 6, 7, 8$
if T specified, $0, 1, 2, 3, 4, 5, 6, 7, 8$, (TIME:)
Note: T = time duration will be specified
0 = halt motion
1 = open hand
2 = close hand
3 = rotate CCW
4 = rotate CW
5 = close hand and rotate CCW
6 = close hand and rotate CW

7 = open hand and rotate CCW
8 = open hand and rotate CW

J Set rate simulate mode flag

K Reset rate simulation mode flag

L* Specify length of object in hand (adjust effective hand length) (L:)

M* Accept mode, axis name, and manipulator variables; execute manipulator variable control algorithm
I, $$X, Y, Z, S, E, W, R, T$$ Incremental Mode
A, $$X, Y, Z, S, E, W, R, T$$ Absolute Mode
Note: T = terminate list and R = shoulder rotation.

N* Coordinate Transformation Reference Point Data
P Print coefficients of transformation matrices
R, (X:, Y:, Z:), $0, 1, 2, 3$ Specify position and reference point
S, $0, 1, 2, 3$ Specify reference point (use hand position)
T Compute transformation matrix.

P* Punch paper tape for automatic mode
G Compute present hand variables and punch (absolute)
H, (X:, Y:, Z:, W:, R:) Specify desired hand variables and punch (absolute)
M Input present manipulator variables and punch
L, (X:, Y:, Z:, W:, R:, S:, E:) Specify desired manipulator variables and punch
O Compute present hand position (HX, HY, HZ) and punch origin block
N, (X:, Y:, Z:) Specify object origin and punch
R Compute present hand variables and punch (relative)
S, (X:, Y:, Z:) Specify desired hand variables and punch (relative)
I, $T, 0, 1, 2, 3, 4, 5, 6, 7, 8$
if T, then $0, 1, 2, 3, 4, 5, 6, 7, 8$, (TIME:)
Punch hand grip and rotate block (see TTY Exec "I" command for notation)
V, (DIST:) Punch straight line path control block
W, (T:) Punch wait block (T=wait time)
T Punch TTY Exec return code
X Punch. ignore code

Q Repeat execution of lost block read from paper tape

R Transfer control to AUTO Exec

S Perform software A/D conversion

T* Type variables
$0, 1, 2, 3, 4, 5, 6, 7$
Note: 0 = Exact contents of 10 bit manipulator buffers
1 = FMV, FHV buffers, scaled
2 = IMV, IHV buffers, scaled
3 = FMV buffer, scaled
4 = FHV buffer, scaled
5 = IMV buffer, scaled
6 = IHV buffer, scaled
7 = FMV buffer and TMAX, scaled

V* Straight line path control (DIST:)

* Additional information must be specified by operator
Legend: (subroutine requests data)
$ one additional command is necessary $
$$ an axis name, then a position or increment must be specified, list is terminated only by a special command$$

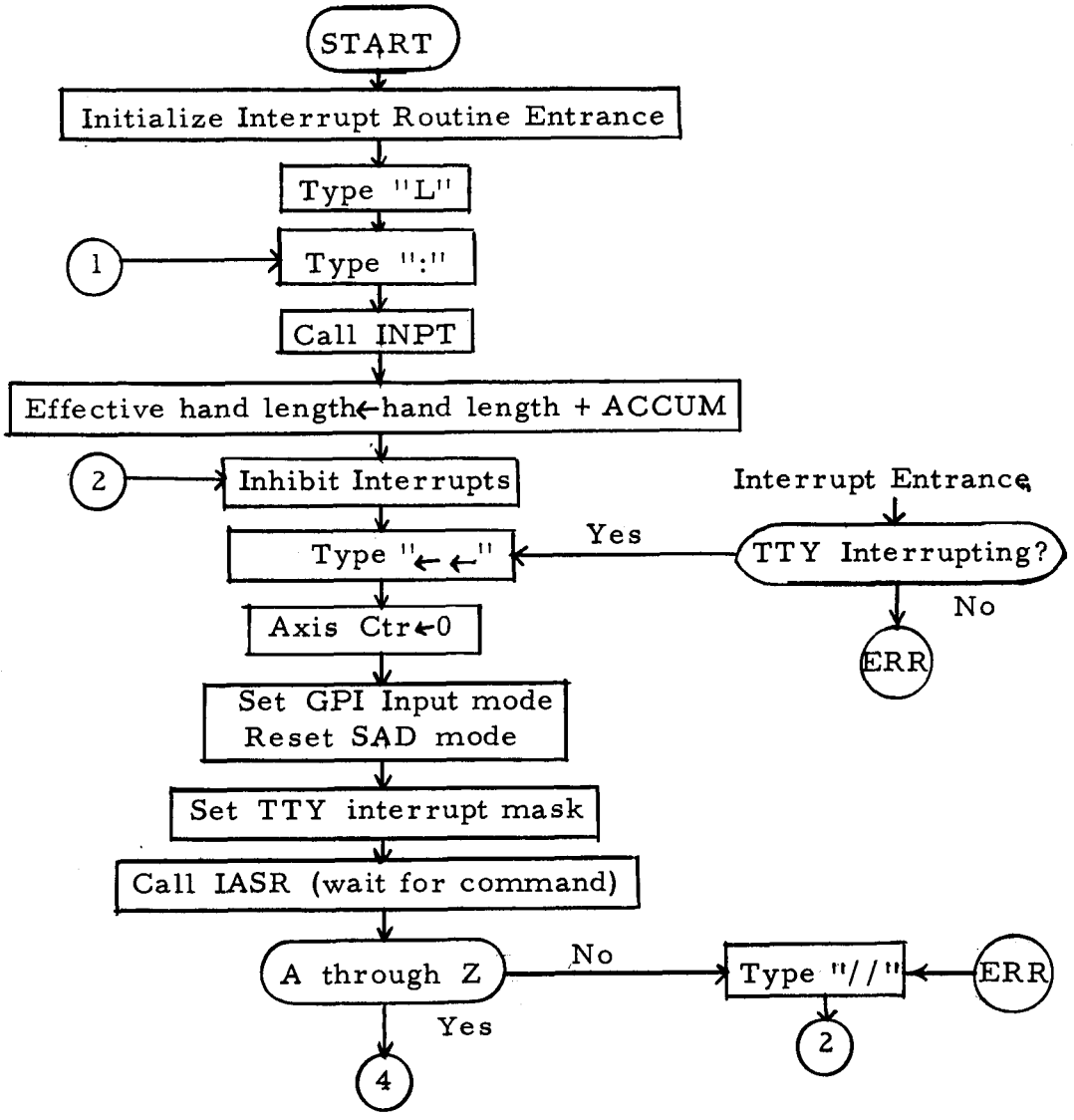
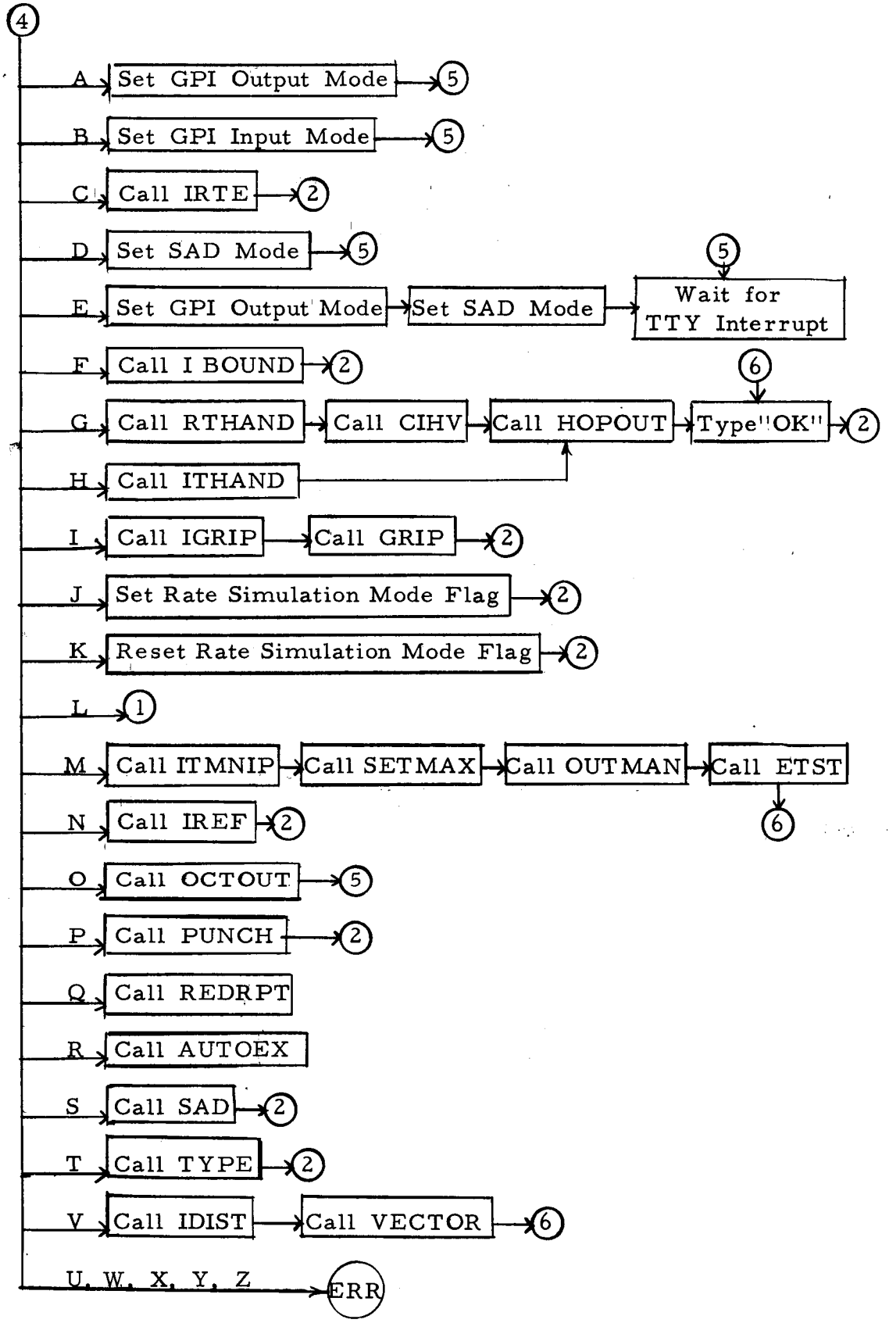


Figure A2-1. TTY Exec Flow Diagram


4
A
Set GPI Output Mode
5
B
Set GPI Input Mode
5
C
Call IRTE
2
D
Set SAD Mode
5
E
Set GPI Output Mode
Set SAD Mode
5
Wait for TTY Interrupt
F
Call I BOUND
2
6
G
Call RTHAND
Call CIHV
Call HOPOUT
Type "OK"
2
H
Call ITHAND
I
Call IGRIP
Call GRIP
2
J
Set Rate Simulation Mode Flag
2
K
Reset Rate Simulation Mode Flag
2
L
1
M
Call ITMNIP
Call SETMAX
Call OUTMAN
Call ETST
6
N
Call IREF
2
O
Call OCTOUT
5
P
Call PUNCH
2
Q
Call REDRPT
R
Call AUTOEX
S
Call SAD
2
T
Call TYPE
2
V
Call IDIST
Call VECTOR
6
U, W, X, Y, Z
ERR

Appendix III

SUBROUTINE LIBRARY FOR EXPERIMENTAL COMPUTER-CONTROLLED MANIPULATOR

The subroutines listed below have been written to implement the control functions and algorithms described in Chapter III. Note that an extensive set of Teletype input-output routines have been included to facilitate communications between the operator and the computer. The subroutines are coded in the DAP assembly language for the DDP-116 and only fixed point arithmetic operations are used; consequently, the execution times and memory capacity requirements are minimized. Approximately 5,000 words of memory are required to concurrently store all programs (including tables and extensive error diagnostic routines) in the memory of the computer. A comprehensive description of the subroutines appears in a separate report.[1]

TTY Exec:	a. Accept commands from operator via TTY console b. Enable TTY interrupts during execution of algorithms
AUTO Exec:	Control fully automatic mode a. Read paper tape b. Transfer data to proper buffers c. Transfer computer execution to proper control routine
REDRPT:	Return to AUTO Exec and re-execute last block of data read
VECTOR:	Control path of manipulator such that it approximates a straight line
GRIP:	Control hand grip and rotation (open loop)
PUNCH:	Record commands and data on paper tape for subsequent processing by the AUTO Exec
SAD:	Perform software A/D conversion to find current manipulator position.
IRTE:	a. Request and accept approximate rate of each axis b. Compute and tabulate inverse rate.
IREF:	Accept object coordinate system reference points, compute transformation matrix, or type coefficients of transformation matrix.
RELHND:	Transform relative hand variables from object coordinate system to manipulator coordinate system.
OCTOUT:	Request and accept exact manipulator positions (in octal) and transfer them directly to manipulator buffers.
IMVR:	Input exact manipulator coordinates and scale into 16 bit computer words (see Appendix I).

CIHV:	Call IMVR, then compute present hand variables
FBUF:	Scale 16 bit computer words into ten-bit manipulator positions (see Appendix I).
OMANIP:	Output the ten-bit manipulator positions to the manipulator buffers
SETMAX:	Use initial and final manipulator variables to compute approximate transit time.
RATE:	Use initial and final manipulator variables and transit time to output fine increments to manipulator for rate mode simulation.
ETST:	Wait for the manipulator to reach the final variables. If the transit time is exceeded by a significant amount, operator assistance is requested.
OUTMAN:	Call RATE if rate mode simulation flag is set; otherwise call FBUF and OMANIP.
HOPOUT:	Control hand position; optimize terminal manipulator configuration and avoid obstacles.
GRID:	Optimize terminal manipulator configuration, fill GRID time table, and compute TMAX.
CFES:	Use Grid time table pointer to compute FSP and FEP, then call CFMV.
CFMV:	Use FSP, FEP, FWP, and FSR to compute FMX, FMY, and FMZ.
FSRT:	Use azimuth and ISR and find FSR to minimize rotation yet not exceed rotation limits.
IBOUND:	Add obstacle bounds to table, initialize the table, delete an obstacle or type effective bounds of all obstacles in the table.
PHCK:	Call RDTC and DTCT to test for possible

	collisions -- if none are detected, output variables to the manipulator.
DTCT:	Test for possible collisions between manipulator and an obstacle when the normal output mode is used.
RDTC:	Test for possible collisions between manipulator and an obstacle when the rate simulation output mode is used.
TEST:	Given a set of manipulator variables, compute portions of shoulder, elbow, and wrist pivots and the hand with the obstacle bounds.
FLAGGT:	Flag current time value in Grid time table and search the table for a new minimum time value -- call CFES to find new manipulator variables.
ISCT:	Test line connecting initial and final hand positions for intersection with the X and Y obstacle bounds.
SUB G:	Test possible paths over and around an obstacle; select the one requiring the least approximate transit time.
CADS:	Use manipulator angular variables and length of arm limbs to compute the projection of the upper arm, forearm and hand on the X, Y, and Z axes.
COSX2:	Compute cosine of specified angle.
SINX2:	Compute sine of specified angle.
ATNX2:	Compute angle (between -1/8 and +1/8 revolution) which satisfies specified arc tangent.
SRND:	Round result of multiplication or shift operation.
DIV:	Divide and round the result.
RTHAND:	Request and accept terminal hand variables.

RTMNIP:	Request and accept terminal manipulator variables.
RXYZ:	Request and accept X, Y, and Z variables (stored in FHV buffer).
ITHAND:	Accept axis and absolute position or increment of hand.
ITMNIP:	Accept axis and absolute position or increment of manipulator.
IDIST:	Request and accept distance for vector subroutine.
IGRID:	Accept hand grip or rotation command. Allow a time duration to be specified (assume one second by default)
IANG:	Accept angular values in degrees, convert to fractions of a revolution and scale.
INPO:	Accept an octal number via TTY.
OUTO:	Type an octal number on TTY (suppress leading zeros).
INPT:	Accept mixed fraction or integer via TTY and scale as specified.
OUTP:	Output scaled, mixed fraction on TTY.
TANG:	Convert angular variable to degrees and type value.
TYPE:	Type contents of IMV, IHV, FMV, FHV or manipulator position buffers.
TYPF:	Type contents of FMV and TMAX buffers.
TIMV:	Type contents of IMV buffer.
TIH:	Type contents of IHV buffer.
TEMV:	Type contents of FMV buffer.

TFHV:	Type contents of FHV buffer.
IASR:	Call MAOF (inhibit TTY interrupt), input one character from TTY, then call MARS.
OUT 2:	Type two characters on TTY (inhibit TTY interrupts)
MSSG:	Type a message on the TTY.
CRLF:	Execute a carriage return and line feed.
T2CR:	Call CRLF, then type two characters (primarily for error messages).
MAOF:	Reset TTY interrupt mask to inhibit TTY interrupts.
MARS:	Set TTY interrupt mask to allow TTY interrupts.