NBSIR 75-973

Guide to Improving the Performance of a Manipulator System for Nuclear Fuel Handling Through Computer Controls

John M. Evans, Jr., Ph.D.
James S. Albus, Ph.D.
Anthony J. Barbera, Ph.D.
Robert Rosenthal
William B. Truitt

Office of Developmental Automation and Control Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D. C. 20234

November, 1975

Final Project Report

Prepared for
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37830

NBSIR 75-973

GUIDE TO IMPROVING THE PERFORMANCE OF A MANIPULATOR SYSTEM FOR NUCLEAR FUEL HANDLING THROUGH COMPUTER CONTROLS

John M. Evans, Jr., Ph.D.
James S. Albus, Ph.D.
Anthony J. Barbera, Ph.D.
Robert Rosenthal
William B. Truitt

Office of Developmental Automation and Control Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D. C. 20234

November, 1975

Final Project Report

Prepared for
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37830

TABLE OF CONTENTS

I. INTRODUCTION

The Office of Developmental Automation and Control Technology of the Institute for Computer Sciences and Technology of the National Bureau of Standards works to assist other Government agencies in the efficient application of computer controlled automation systems. The products of the program are consulting and advising services, standards and guidelines on interface and computer control systems, and performance specifications for the procurement and use of computer controlled manipulators and other computer based automation systems. These outputs will help other agencies and industry apply this technology to increase productivity and improve work quality by removing men from hazardous environments.

In FY 1974 personnel from the Oak Ridge National Laboratory (ORNL) visited NBS to discuss the feasibility of using computer control techniques to improve the operation of remote control manipulators which support their HTGR fuel refabrication program. Subsequent discussions led to an agreement for NBS to develop a conceptual design for a computer control system for the remote control electromechanical manipulators in the Thorium Uranium Re-cycle Facility (TURF) at ORNL (See Appendix A). Following the above agreement, an ORNL decision was made to assign the task of developing the conceptual design to ORNL personnel.

The role of NBS, in support of the ORNL conceptual design, was shifted to responsibility for development of complete computer programs for testing the ORNL computer interfaces and for controlling the ORNL manipulator as a robot in both point-to-point and continuous path modes. In addition, NBS agreed to serve as a consultant regarding the ORNL conceptual design.

ORNL has provided the required conceptual design, and this conceptual design was reviewed and endorsed by NBS. Following agreement on the conceptual design, the required computer programs to implement it were developed at NBS. These programs are written in FORTRAN to insure transportability, and they have been tested in the NBS Automation Laboratory prior to release to ORNL. The proposed modifications will result in at least a ten-fold increase in the rate of task completion for the remote manipulators with a substantial reduction in fatigue to the manipulator operator.

II. OPERATIONAL REQUIREMENTS

As a result of discussions with ORNL the following operational requirements for HTGR fuel refabrication were transmitted to NBS.

The manipulator systems servicing Cells C, D, and E of TURF are PaR Model 3000 electromechanical manipulators on overhead bridges. The standard controls are rate control switches for each separate joint on a control box operated from outside of the cell; if necessary, TV viewing is used to assist the operator.

The TURF facility already had the PaR manipulators installed when this project began. The concepts presented here are directly focused for that facility, as directed by the ERDA/NBS agreement. However, the computer control concepts discussed in this report are generally applicable to any remote control electromechanical manipulator and should this benefit other ERDA facilities and to other Government programs.

Significant improvements in the operation of such remote manipulators are possible through the addition of more advanced master-slave control systems. For example, Whitney and Nevins at the Charles Stark Draper Laboratory document a full order of magnitude improvement in task completion time with master-slave control systems compared with switch box control systems.

Operation of the High Temperature Gas-Colled Reactor (HTGR) Fuel Refabrication Pilot Plant (FRPP) includes such operations as refilling feeder bowls, moving assembled fuel elements to annealing furnaces, and changing furnace liners. ORNL personnel have indicated that these operations will be performed automatically by the manipulator system if automation demonstrates its superiority in rate of task completion.

In addition, the manipulator system will be required to perform remote maintenance operations for the fuel reprocessing equipment, including coupling disconnects, nonroutine high dexterity operations, and movement of major pieces of processing equipment to Cell B for repair. These complex, non-repetitive tasks will require a manual master/slave control mode.

The manipulator control concepts required to carry out those operations are considered next. There are two key elements in the proposed facility modifications: The addition of position servos to the manipulator joints to allow master/slave control and the addition of a small control computer to allow automatic control. The computer will be able to provide robot operation of the manipulator and automatic traversing and positioning of the overhead crane; it will also provide control of remote TV cameras and monitoring of total system operation for failure and for collision avoidance.

III. CONTROL REQUIREMENTS

The problem of controlling a manipulator system can be thought as a continuous spectrum from direct rate control of each joint (as in current remote control manipulator systems) up to complete autonomous adaptive operation under computer control. Greater flexibility and manipulative capability is gained by moving along this spectrum, augmenting man's ability to operate the remote manipulators through the addition of servo and computer controls.

A phased approach (recommended by ORNL and endorsed by NBS) to improve the operation of the manipulators for the HTGR FRPP is as follows:

Step 1: ADD POSITION SERVOS

Step 2: ADD MASTER/SLAVE CONTROL

Step 3: ADD COMPUTER INTERFACE

Step 4: ADD COMPUTER SOFTWARE FOR ROBOT CONTROL

These steps are now considered in detail.

Step 1: ADD POSITION SERVOS

Adding position servos to each joint of the manipulator is one of the most expensive and difficult steps, but it is absolutely essential to any additional improvements.

Position Sensors

Position servos require sensors that will measure the position of each joint. Candidates sensors are potentiometers, encoders, and resolvers. The criteria for selection are reliability, wires per joint, and noise immunity.

Potentiometers are the least expensive sensor, but they provide an output that will be susceptible to noise. In addition, voltage drops over the extensive cables required to connect the potentiometers with the controls external to the cell will appear as a position dependent error. Potentiometers require one lead (plus a power supply and a common ground) per joint.

Encoders may be either incremental or absolute. Incremental encoders give pulses at equally spaced angular increments. Absolute position is formed by summing the pulses in an up-down counter. Incremental encoders require two wires per joint for the two pulse trains required to determine the direction of motion (plus a common ground and common power supply lead), but they can be susceptible to pulse noise or power failure. The number of electric motors (noise sources) present on the manipulators and in the FRPP equipment dictate against incremental encoders if automatic operation is desired. Absolute encoders have good noise immunity, but require 1 lead per bit. This means that 12 bit encoders need either 12 wires per joint or a multiplexer in the radioactive environment.

Resolvers offer the best alternative. Resolvers are extremely reliable, and give good noise immunity. They require three leads per joint (plus a common ground).

Velocity Sensors

The manipulator arm positioning tolerance defined by Oak Ridge requires a position resolution of one part in ten-thousand. This high resolution requires such a high position feedback gain that compensating velocity feedback is required to achieve a stable position servo. Commercial dc servo motors are available which incorporate a dc motor tachometer to provide the required velocity feedback voltage.

Position Servos

Once position resolvers and dc servo motors with tachometers are mounted on the joints of the manipulator, the control loop can be established by feeding velocity and position voltages into a single operational amplifier. The output of the amplifier can then directly drive a power amplifier which in turn drives a dc servo motor, as shown in Figure 1.

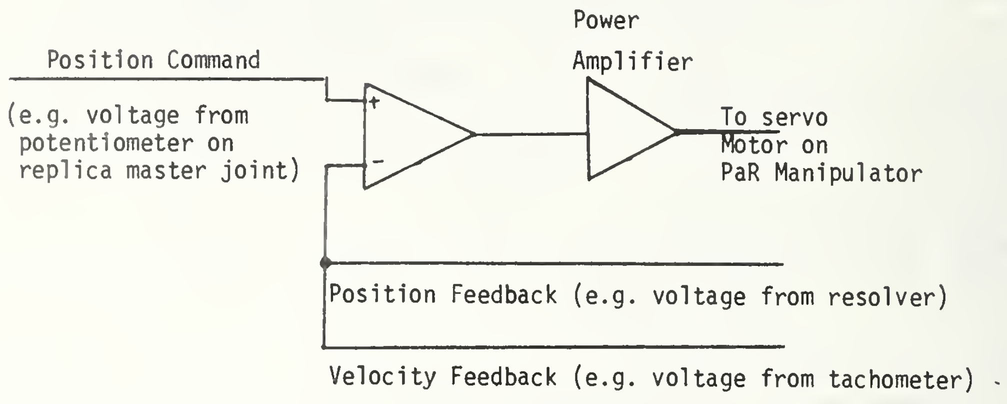
Position Command
(e.g. voltage from potentiometer on replica master joint)
+
-
Power Amplifier
To servo Motor on PaR Manipulator
Position Feedback (e.g. voltage from resolver)
Velocity Feedback (e.g. voltage from tachometer)

Figure 1 Servo Schematic

Step 2: ADD MASTER/SLAVE CONTROL

Once a closed loop control has been added, a replica master can be for control; ORNL already has such a master control with articulations matching the existing manipulator The outputs of potentiometers on each master joint provide the position commands for the manipulator position servos. The operator moves the hand of the master in a desired motion, and a PaR 3000 is servoed to follow that motion. This should result in increased operating speed by up to an order of magnitude by allowing the operator to simulataneously operate and coordinate the motions of all of the joints to achieve the desired hand motions. This addition is considered essential to expedite FRPP maintenance operations.

Step 3: ADD COMPUTER INTERFACE

The addition of a small computer to the control system allows a further increase in operational efficiency. The computer and interface circuitry is a major expense, but one that will result in clear cost benefits in allowing automatic operation of the manipulator. In particular, once the proper trajectories are recorded, the computer can move the manipulator around the cells within the accuracy of the servos and at maximum velocity without any danger of collision with FRPP equipment.

Computer Interface

The design of a computer interface depends on the specific computer hardware configuration and its location with respect to the manipulator.

The interface design used by NBS is described in Appendix C. The computer is approximately 100 feet from the manipulator. The interface consists of:

Input

1) A commercial A/D converter consisting of a 64 channel multiplexer rated at 20KH2 word rate with single-ended inputs.

2) A 16 bit parallel connection to a DR-11C general purpose interface card on the PDP-11 control computer.

Output

3) A 16 bit parallel connection from the DR-11C to the D/A circuitry.

4) A 16 channel D/A output 15 bit, sequential addressing only. 16th bit is for reset.

The interface system, utilizing a single 16 bit parallel I/O interface, O/C does not use interrupts, all I/O is handled on a polled basis. Therefore, this system design works well with any Operating System capable of supporting a "stand-alone" program. All inputs are read into the input buffer; and control calculations are made using commands which are loaded into an output buffer and then transmitted to the dedicated D/A converters for each joint servo. See Figure 2.

The important elements of an interface design recommended for the HTGR FRPP facility are:

1) Use of single channel input and output.

2) Use of minimum interrupt structures except for safety interlocks. The use of a a real-time operating system that is interrupt driven will require use of interrupts for input/output control.

3) Use of commercial D/A and A/D units that will be more reliable and less expensive than in-house designs;

4) Use of a control panel with several off/on switches and a potentiometer. (This control panel has proven to be very useful in programming and controlling computer controlled manipulators.)

Bringing up a reliable, noise-free interface is often an extensive engineering effort. Simplicity of design is very important. The test programs in Appendix D have proven helpful in debugging interface and communications circuits.

Step 4: ADD COMPUTER SOFTWARE FOR ROBOT CONTROL

The software for automatic (robot) operation of the manipulator system is straightforward. NBS has followed a hierarchical structure to software development consisting of 5 levels:

Level 1. Interface Drivers: Interface drivers are programs written in assembly language. They isolate the computer dependent I/O from the higher level software programs.

Level 2. Trajectory Control: Trajectory control of the robot, accomplished with a record-and-play-back technique, allows the robot to do both point-to-point and continuous path trajectories. Also, coordinate transformations of recorded trajectories are accomplished under program control.

Level 3. Elemental Moves: Elemental moves such as "docking" are accomplished using sensory feedback to allow adaptive control in a partially unstructured environment.

Level 4. Simple Tasks: Simple tasks, such as "replace furnace liners", are sequences of elemental moves.

Level 5. Complex Tasks: Complex tasks are the highest level control programs, such as "move broken index table to Cell B for repair". This program level calls up sequences of lower level programs.

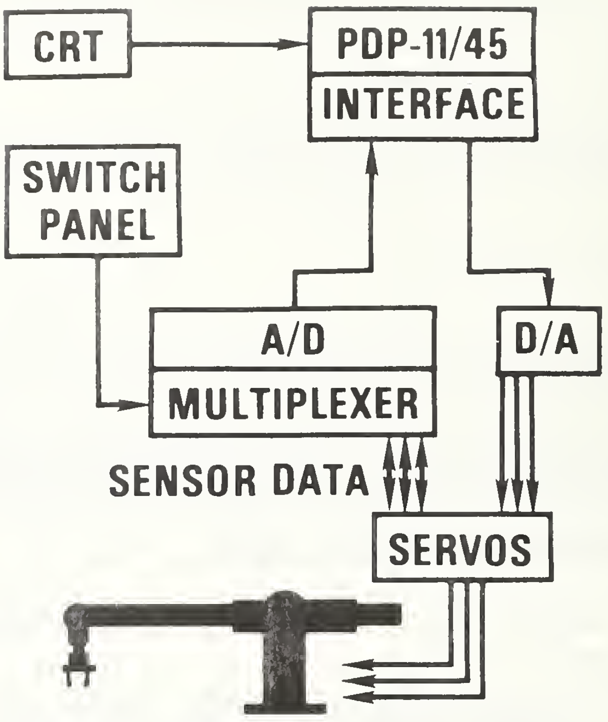

FIGURE 2. LABORATORY INTERFACE SCHEMATIC

Most of the operations required of the manipulators for the FRPP can be carried out by implementing Level 1 and Level 2. In particular, implementing the continuous path and point-to-point trajectory record and play back at Level 2 allows execution of the automatic operations required. These programs are described and listed in Appendicies B, D, and E.

For initial experiments it is recommended that only interface drivers and trajectory control programs be implemented. Testing for accuracy, repeatability, and stability of the servos and computer controls can be carried out with this level of control.

Coordinate transformations are useful in conjunction with sensor inputs for Level 3. Coordinate transformation capability will be needed if proximity, force, or touch sensors are required for docking, moving fuel elements to the annealing furnace, inserting furnace liners, or similar operations. Sensors on the hand of the manipulator or on external objects in the environment of the manipulator generate data that is in the coordinate system of the sensors themselves. For example, force sensors in the manipulator wrist will generate Fx, Fy, and Fz data referred to the wrist orientation. Closed loop control using this data inplies that motions of the manipulator must be carried out in the same coordinate system as the sensor data (e.g. inserting a furnace liner requires forces orthogonal to the axis of the furnace should be zero and the manipulator must therefore move exactly along the axis of the furnace). This closed loop control in turn requires transforming x y z motions into coordinated motions of the manipulator joints. Coordinate transformation capability can be added later, after the need for sensory feedback has been determined.

It should be noted that jigging and compliance can replace the need for Level 3 control programs in some cases. In particular, robot manufacturers in both the U.S. and Japan have used compliance and vibration with open loop positioning for complex assembly tasks.

IV. SUMMARY AND CONCLUSIONS

The operational requirements described in Part II may be achieved by implementing the following steps:

Step 1: Add resolvers and tachometers to each joint of the manipulator. Then close servo loops with position and velocity feedback.

Step 2: Add master/slave control with a replica master using position transducers to drive the manipulator position servo loops.

Step 3: Add mini-computer with appropriate interfaces.

Step 4: Implement robot control programs for point-to-point and continuous trajectory control.

Steps 1 and 2 together give the capability of master/slave control for maintenance and repair and other non-repetitive operations. Steps 3 and 4 allows automatic (robot) operations for routine repetitive tasks.

Testing of these modifications should be carried out before additional change are made. The results of testing Steps 1 through 4 will determine the need for force, touch or proximity sensors and Level 3 control programs to carry out routine operations such as docking, filling feeder bowls, and replacing furnace liners.

Implementation of these proposed modifications will result in increased speed of operation of the manipulator and overhead bridge by at least a factor of 10 and will allow routine transfer and service operations to be accomplished automatically.

Appendix A:

Letter from ERDA (then AEC) Oak Ridge Operations Office to NBS, November 13, 1974, requesting a conceptual design study of a programmable control system for an electro-mechanical manipulator system.

Appendix B: DESCRIPTION OF NBS COMPUTER CONTROL SOFTWARE

1. Introduction

Recently a series of experiments was conducted in the ICST Automation Technology Laboratory in which the computer control of a mechanical manipulator was studied. In this experiment, a manipulator was interfaced to a mini-computer.

The lowest levels of computer software for controlling this manipulator as a robot are described in this report. The discussion will describe software strategy and will examine the general problem of specifying a mini-computer for program development and real time control.

The system described is a PDP-11 computer with D/A and A/D interfaces, a hydraulic power anthropomorphic manipulator system with full hardware servos, and an exoskeleton harness control for master/slave operation. The techniques discussed are generally applicable to the computer control of manipulators.

a. Problem Statement

While the main function of the laboratory mini-computer is to execute the manipulator control programs, program preparation, compilation, linking and loading must also be performed. In general, a dedicated laboratory mini-computer must have more than just a central processing unit (CPU) and a program store (core) to carry out these program development steps. The issues of what is required in the mini-computer and its associated hardware are discussed as they relate to the problem of sizing a machine (the mini-computer) to do the necessary control functions. First, a description of the software system developed in the ICST laboratory is discussed. Then, a description of the necessary elements in the mini-computer architecture as well as a general description of a possible mini-computer configuration is presented.

b. System Overview

The laboratory mini-computer is the central coordinating point for all of the manipulator hardware. A 16 bit general purpose parallel interface connects the computer to the manipulator. The interface output buffer is connected to a D/A converter designed by NBS. The 16 bit parallel output is switched sequentially to buffer registers. These buffer registers drive digital to analog (D/A) converters which drive the servo valves on the joints of the manipulator. There is one D/A converter for each joint of the manipulator.

Input to the computer from the manipulator hardware is from a 64 channel analog to digital (A/D) multiplexer and converter. The signals to the multiplexer are derived from potentiometers and other sensors connected to the manipulator, the exoskeleton, and a special panel containing function switches and potentiometers. The A/D converter is connected to the input buffer of the general purpose interface board.

In the system described here, several software programs were prepared and executed on the mini-computer. The programs interact with the operator through the operators's console which also provides system status information. The system software was developed in a modular fashion so that new applications could easily be built by incorporating existing software.

2. Software System

The ICST software system for mini-computer control appears as a hierarchy of modular control functions easily organized into a comprehensive applications package for controlling a manipulator. Three hierarchical levels of software are defined. At the lowest level are assembly language input and output drivers for the interface to the manipulator. At the next level are programs for teaching the manipulator to carry out various trajectories. At the highest level are programs for coordinate transformations which allow trajectories to be defined in an experimental (laboratory) coordinate system. The first two levels are discussed here.

Appropriate sensor input to predefined A/D converter channels enable the mini-computer to control the manipulator in response to the sensory stimuli. Acoustic positioning feedback sensors, force sensors and pressure sensors have already been used in the laboratory experiments. The hierarchy of software programs used to provide the control functions is described.

a. Hierarchy

1. Lowest Level

At this level, two driver modules are specified. ARMIN is the driver which transfers data from the A/D multiplexer into a core buffer and ARMOUT transfers data from a core buffer to D/A converters which are connected to the servo valves which control the manipulator joint positions.

2. Second Level

The first application program in the system is called SANDF. SANDF is a mnemonic for "store and forward". The program uses the driver program ARMIN to read the input channels. Included in the input channel data are the positions of the exoskeleton joint angles. These exoskeleton channel datum are scaled by SANDF after which SANDF calls the driver ARMOUT. ARMOUT writes the scaled exoskeleton positions to the servo valves of the manipulator. In this way, SANDF is able to control the manipulator by causing it to track the exoskeleton. Successive calls to SANDF will cause the manipulator to follow the trajectory of the exoskeleton.

Building upon this existing software, an application program called ARM was designed to behave much like a tape recorder. In the ARM program, the exoskeleton positions are stored; then forwarded to the manipulator with the SANDF program. At the operator's request, the manipulator coordinates are recorded by the program and stored in the mini-computer. Much like a tape recorder, the operator can play back these coordinates and cause the manipulator to traverse the pre-recorded trajectory. ARM allows us to operate at the sophistication level of commercially available industrial robots.

Embellishments to this program are worth noting. By utilizing the file system of the mini-computer, sets of coordinates can be saved, and later retrieved by name for play back. Also, during play back, a special function button panel can be accessed. This panel has a potentiometer that can be manually set by the operator. The program ARM uses the potentiometer setting to indicate the speed at which the play back is to occur. While the maximum play back is limited by the execution time for the program, delays in play back are introduced by reference to a routine called TIMOUT which accepts, as an argument, a delay specified in 60ths of a second. The potentiometer value is appropriately scaled to indicate a suitable delay. Full speed play back is significantly faster than the recording speed so that the play back speed ranges from almost twice as fast as normal, to very very slow -- where each recorded point in the trajectory is easily noticeable.

A second program, PNTPNT, allows point-to-point trajectories to be recorded and executed. Points are recorded when the carriage return is typed on the control terminal rather than at equally spaced time intervals.

A third program, NEWARM, combines these two programs and adds interpolation between stored points along the trajectory during play back.

3. Software Requirements

While the main function of the mini-computer is to execute the control programs for the manipulator, program preparation, compilation, linking and loading must also be performed. The basic operating system usually supplied with the mini-computer should provide these functions.

a. Program Preparation

Once conceived by the programmer, the program must be entered into the computer system. Typically, the mini-computer is supplied with a keyboard control terminal and an operating system containing an editing system for entering new programs or modifying existing ones. If the mini-computer has a disc based operating system (DOS), the program source files are easily maintained as text files on the system disc. DOS can also be used to support the compilers and/or assemblers used to transform the source language statements into machine instructions. If no disc is available, some other means of storing the source program and compilers is required. Usually this is done with paper tape. Such systems, while less expensive to procure, generally require significant amounts of time just to do simple operations like compiling programs. Thus, the initial cost savings of a paper tape system are quickly absorbed in non-productive labor costs.

b. Program Compilation (Assembly)

Once the programmer is satisfied that the text of his source file is correct, he can compile the program; if the text of his source file is an assembly language program, he can assemble his program. Higher level language programs, which require compilation, are usually easier to read, easier to write, and easier to debug; but, inefficiencies in machine code are generally introduced. Many systems will therefore, allow the main programs and many subroutines to be written in a higher level language which is compiled, and allow the routines which require extreme efficiency to be coded in assembly language. In order to put together the compiled programs and the assembly language programs in preparation for executing the system, a processes called linking is performed.

c. Program Linking and Loading

To link programs together into a complete system requires a special operating system routine called a "Linker". This program assigns core locations to the variables defined in the user programs and assigns space in main memory for the actual programs to be executed. The output of the linker program is typically a file containing a core image of the linked program, or a file containing an image of the linked program along with special loader instructions specifying how the linked program is to be loaded. In any case, once the program is linked, a loader is capable of preparing a core image which can be executed by the CPU.

d. Program Execution

Once the loader is finished preparing a core image of the linked program, the loader transfers control of the CPU to the start address of the loaded program.

4. Hardware Configuration

The hardware configuration for a mini-computer used to support program preparation is often significantly different from that of a mini-computer used to execute a dedicated user program. In one case a disc based operating system used to edit, compile, assemble, link and load, might require a high speed line printer for listings of source programs, a disc pack peripheral for mounting user program libraries of commonly used debugging aids and much more. The dedicated user program on the other hand will only require a few peripherals specific to the needs of the intended application, for example, an interface to a manipulator.

a. Program Preparation

Experience indicates that in order to do program preparation on a mini-computer a disc based operating system is prefered to a paper tape system. Also at least some hard copy printing device is necessary to get listings of the programs written as well as any data the program may generate. While a high speed line printer is necessary on a machine dedicated to program development, the high cost is not justified for intermittent program development sessions. Finally, a mountable secondary storage medium such as magnetic tapes or disc packs is necessary when installing the manufacture's operating system. Mountable storage is also very useful whenever large volumes of data are collected by the application or user program.

b. Program Execution

The mini-computer configuration requirements for program execution are much different than for program perparation. The only peripheral devices required are those which are accessed by the program. The normal peripherals used in program preparation however, can often serve dual purposes. The operator's console can be used to list information regarding the status of the user program. Secondary storage is also used if the user program generates large volumes of data. Also, in systems where core size is limited, provisions are often made to overlay portions of the user program. This technique allows programs to be written which use more core than is available at any one time. The user program is organized in blocks or segments which can easily be swapped between core storage and disc storage. While this technique usually slows down the execution of the user program, the benefits gained in available program size are usually justified.

The special peripheral requirements used in the manipulator control programs include the parallel interface to the A/D and D/A converter systems. The special function button panel connected to the input A/D multiplexer provides an excellent means to control the manipulator software. Also, the visual operator feedback from the blinking lights on the multiplexer provides assurances to the operator that the mini-computer is functioning properly. This has proved to be a valuable aid during the debugging phase of the software system.

5. Conclusions

The intent of this report is to focus on the computer power and software techniques required to control a complex mechanical manipulator. Experiments in the ICST Automation Technology Laboratory indicate that sophisticated control functions can be performed using a mini-computer equipped with the proper peripherals and equipped with the proper software tools to develop the application programs.

In the ICST Laboratory, a DEC PDP-11/20 was originally used to control the manipulator. In preparing programs, a disc operating system equipped with a program editor, a FORTRAN compiler, a macro assembler and a linking loader were used. Control programs were later transferred to a PDP 11/45 again using DOS. All software described here could be prepared and run on a PDP 11/20 size mini-computer.

Hardware peripherals unique to the requirements of the laboratory were utilized by the applications programs. A special function button panel connected to several input channels of the A/D converter proved to be a very useful and desirable operator convenience in controlling which software module was to be executed. Also, the visual feedback from the blinking lights on the A/D converter reassured the operator that the programming system was operational.

Appendix C: INTERFACE DESIGN

Data transfer between the laboratory hardware and the PDP-11/45 computer is handled via a DR11-C interface card in the computer. This card has 16 data input lines, 16 data output lines, two interrupt lines and two sync lines, one (data transmitted) for input synchronization, and one (data present) for output synchronization.

The DR11-C is connected to the laboratory interface by forty parallel wires powered by 7406 line drivers. These are open collector circuits with resistors connected to the power supply voltage at the receiving end.

The laboratory interface hardware consists of two parts: an input section and an output section.

Input:

A sixty four channel analog multiplexer followed by a 14 bit analog to digital converter forms the heart of the input section. This is range -10 volts to +10 volts. Negative voltages are converted to 2's complement form.

Output from the A/D converter is carried to the computer on the 16 data input lines. Since the A/D converter has only a 14 bit output, the most significant (sign) bit is connected to the three most significant computer input lines. This causes the computer to read the correct values for both positive and negative (2's complement) numbers.

Data transfer between the A/D converter and the computer is signaled by a pulse on the data transmitted line. This pulse, initiated by the computer, indicates that the data lines have been sampled. This data transmitted pulse is used to step the analog multiplexer to the next input and to initiate the analog to digital conversion cycle on the new input voltage. The input software is timed to sample the input data at a rate slower than the conversion rate of the A/D system. Thus, a "conversion completed" signed is not required by the computer, and only one control line is needed to advance the converter to the next channel.

The multiplexer is operated in a sequential mode. It is reset to the zeroth channel by a reset pulse derived from the output section. Each data transmitted pulse steps the multiplexer to the next sequential channel. The input section is this completely under computer control.

Output:

A sixteen channel digital demultiplexer routes output signals from the DR11-C to sixteen holdings registers. These provide either digital output directly, or are connected to digital to analog convert so as to provide analog voltabes. Datatransfer between the DR11-C and the output section is initiated by a pulse on the data present line. The data present pulse occurs simultaneously with the appearance of data on the DR11-C output lines This pulse is delayed in the laboratory interface hardware for two microseconds before actuating the digital demultiplexer so as to allow the data lines time to settle. The delayed data present pulse gates the output data into the selected holding register and then steps the address counter to the next address.

The most significant (sign) bit of the output is decoded as a reset signal and is used to set the demultiplexer address counter to Channel 1 as well as to reset the analog multiplexer address in the input section.

Using the sign bit for control means that only 15 bits of output are available for data. Numerical is restricted to the range 0 to 32,767. The digital to analog converts are adjusted such that numerical 0 corresponds to +10 volts; 16,384 corresponds to zero volts, and 32,767 corresponds to -10 volts.

Appendix D: <u>INTERFACE DRIVERS AND TEST SOFTWARE</u>

The following programs are included in this Appendix:

1.	ARMIN:	An Assembly Language input driver for the DR-11C interface
2.	ARMOUT:	An Assembly Language output driver for the DR-11C interface
3.	TEST:	A FORTRAN program for displaying and testing the values of the A/D input circuits.
4.	SWTH:	A staircase function generator for testing the D/A output circuits by generating a sawtooth wave form.
5.	INTFC:	Interface tester for testing loop from D/A through A/D functions which is looped back to input and displayed. Also displays neighboring channels to test crosstalk.
6.	SAAR:	Sine wave generator with step size limiting for safety. Exercises all joints at once for servo and total system testing.

Notes: These programs are written in FORTRAN and operate on PDP-11/45 under DOS. They should also operate successfully on smaller members of the PDP-11 family. Operation of these programs under RSX, UNIX, ELF or other operating systems, or operation of other mini-computers will require rewriting ARMIN and ARMOUT and review of the I/O calls in the programs.

```
        .TITLE ARMIN FOR LOW LEVEL ARM INPUT
        .IDENT /RMR01/
;
; ARMIN AND ARMOUT HAVE SIMILAR DOCUMENTATION.  THE TWO PROGRAMS
; SHOULD BE READ TOGETHER AS SOME DOCUMENTATION EXISTS IN
; ONE AND NOT THE OTHER.
;
        .CSECT
;
; THE ARM IS CONTROLLED BY A DR11-C INTERFACE TO THE PDP-11/45
; UNIBUS.  ALTHOUGH ALL THE INTERRUPT FACILITY AVAILABLE WITH
; THE DR11-C IS INTACT, NO INTERRUPTS ARE USED WITH THIS VERSION
; OF THE ARM SOFTWARE.  THE GENERAL PHILOSOPHY OF THE SOFTWARE
; IS SUCH THAT DATA TRANSFER TO AND FROM THE ARM WILL OCCURE
; WHENEVER A PROGRAM WANTS AND/OR HAS THE TIME.
;
;       THE DR11-C HAS A UNIBUS ADDRESS OF 167770
; THIS IS THE CONTROL REGISTER. 167772 IS THE TRANSMIT REGISTER
; AND 167774 IS THE RECEIVE REGISTER.  THE A/D CONVERTER
; IS RESET WHENEVER IT SEES BIT 15 SET.  AT LOCATION 24, THE A/D CONVERTER
; IS RESET.
;     ALL 64 CHANNELS ARE READ INTO THE COMMON BLOCK NAMED ARMBUF
; THIS IS FORTRAN COMPATABLE COMMON.  ALSO THE SUBROUTINE LINKAGE
; IS COMPATABLE.
;
VAL::   .WORD   25                 ;TIME OUT VALUE
                                   ;IS SET AT 10 WHICH CAUSES THIS
                                   ;ROUTINE TO WAIT (SEE LOOP 1$)
                                   ;LONG ENOUGH FOR THE A/D CONVERTER
                                   ;TO SETTLE DOWN BETWEEN READS.
ARMIN::
        MOV     VAL,R0             ;THE 11/45 RUNS TOO FAST
                                   ;SO WE WILL USE R0 TO
                                   ;TIMEOUT BETWEEN READS.
        MOV     #167772,R2         ;LOAD THE CSR OUTPUT REG
        MOV     #-1,(R2)+          ;WITH A RESET AND POINT
                                   ;TO THE INPUT SIDE
        MOV     #64.,R4            ;COUNT UP TO 64 POINTS
        MOV     #INBUF,R3          ;POINT TO THE FIRST WORD
1$:     DEC     R0                 ;COUNT DOWN FOR 11/45 TIMEOUT
        BPL     1$
        MOV     VAL,R0
        MOV     @R2,(R3)+          ;LOAD THE INBUF WITH 64 POINTS
        DEC     R4
        BGT     1$
        RTS     R5                 ;THEN RETURN
;
;       IMPLICIT INTEGER(A-Z)
;       COMMON /ARMBUF/INBUF(64),OUTBUF(64)
;
; WHEN USED WITH FORTRN, COMPILE WITH THE /ON SWITCH
        .CSECT ARMBUF
INBUF=.
OUTBUF=.+200
.=.+400
        .CSECT
        .END
        .TITLE ARMOUT FOR LOW LEVEL ARM OUTPUT
        .IDENT /RMR01/
;
;       CALLING SEQUENCE IS:
;
;       CALL ARMOUT(COUNT,[BUFFER])
;                      WHERE [BUFFER] IS OPTIONAL
;                      IF NOT THERE, OUTBUF IS USED.
        .GLOBL  VAL             ;THE 11/45 IS TOO FAST, ARMIN AND
                                ;ARMOUT MUST SLOW DOWN, VAL IS A
                                ;WORD TO HELP US DO THAT.
ARMOUT::
        MOV     VAL,R0
        MOVB    @R5,R4          ;GET THE NUMBER OF ARGUMENTS
                                ;IN THE CALLING SEQUENCE
        MOV     @2(R5),R3       ;GET THE NUMBER OF POINTS TO
                                ;SEND TO THE ARM.
        MOV     #OUTBUF,R2      ;ASSUME OUTBUF IS USED
                                ;FOR ARM OUTPUT
        DEC     R4
        BEQ     1$              ;THE NUMBER OF ARGS WAS ONE SO
                                ;USE OUTBUF, OTHERWISE
        MOV     4(R5),R2        ;USE THE ARRAY SPECIFIED.
1$:     MOV     #167772,R4      ;POINT TO THE ARM INTERFACE
        MOV     #-1,@R4         ;AND RESET IT
2$:     DEC     R0
        BPL     2$
        MOV     VAL,R0          ;11/45 IS TOO FAST, SO, TIMEOUT
        MOV     (R2)+,@R4       ;OUTPUT FROM THE BUFFER
        DEC     R3              ;COUNT DOWN ON NUMBER OF POINTS
                                ;TO SEND TO THE ARM
        BGT     2$              ;THEN SEND THEM
        MOV     #-1,@R4         ;RESET AFTER
                                ;THE WRITE (6-MAY-74)
                                ;THIS CAUSES LESS CURRENT
                                ;DRAIN ON THE POWER SUPPLIES IN THE
                                ;D/A CONVERTER.
        RTS     R5              ;THEN GO HOME
;
;       IMPLICIT INTEGER(A-Z)
;       COMMON /ARMBUF/INBUF(64),OUTBUF(64)
;
        .CSECT ARMBUF
INBUF=.
OUTBUF=.+200
.=.+400
        .CSECT
        .END
```

```
C
C        ***************************************
C
C TEST
C        THIS PROGRAM IS USED TO TEST THE A/D PORTION
C        OF THE INTERFACE.  IT REQUESTS AN INPUT
C        CHANNEL NUMBER(M).  THE VALUE ON THIS CHANNEL
C        IS PRINTED OUT AS A BINARY NUMBER(BIN(15)).
C        THIS NUMBER CAN BE COMPARED TO THE BINARY
C        NUMBER FROM THE A/D CONVERTER DISPLAY PANEL
C        TO ASCERTAIN IF THE COMPUTER IS RECEIVING
C        THE CORRECT VALUE AND IF IT IS ACCESSING
C        THE PROPER INPUT CHANNEL.  IF A BINARY
C        OUTPUT ON CHANNEL M IS NOT DESIRED, THEN
C        THE VALUE OF M (M=0,M<0) IS USED AS A
C        FLAG TO (1) OMIT THE CALCULATIONS TO
C        CONVERT TO A BINARY NUMBER AND (2) CAUSE
C        THE PROGRAM TO LOOP CONTINUOUSLY (AS
C        OPPOSED TO STOPPING AFTER 100 OUTPUTS).  THE
C        PROGRAM ALSO DISPLAYS THE DECIMAL EQUIVALENTS
C        OF THE VALUES ON A SEQUENCE OF INPUT
C        CHANNELS.  THE PROGRAM, THEREFORE, REQUESTS
C        THE STARTING CHANNEL NUMBER(LL), AND THE
C        ENDING CHANNEL NUMBER(KK), THEN DISPLAYS
C        THE VALUES ON ALL THE CHANNELS BETWEEN THEM.
C
C        ***************************************
C

         IMPLICIT INTEGER (A-Z)
         COMMON /ARMBUF/INBUF(64),OUTBUF(64)
         DIMENSION BIN(15)
C
C        **********
C THIS SECTION OF CODE REQUESTS THE INPUT
C        CHANNEL NUMBER(M) FOR THE BINARY DISPLAY,
C        AND THE START(LL) AND STOP(KK) FOR THE SERIES
C        DISPLAY.  IF ONLY INPUT CHANNEL M IS TO
C        HAVE ITS DECIMAL EUIVALENT DISPLAYED, THEN
C        BOTH LL AND KK ARE SET EQUAL TO M.
C        **********
C
80       WRITE (6,198)
198      FORMAT ( '  CHANNEL NUMBER = ')
         READ(6,197) M
197      FORMAT (I2)
         WRITE(6,200)
200      FORMAT(2X'STARTING CHANNEL=           , ENDING CHANNEL= ')
         READ(6,201)LL,KK
201      FORMAT(I3,I3)
C
C        **********
C THIS 'DO LOOP' (FROM STATEMENT 81 TO 50)
C        CAUSES THE PROGRAM TO CYCLE THROUGH 100
C        OUTPUTS BEFORE REQUESTING ANOTHER CHANNEL
C        NUMBER.
C        **********
C
81       DO 50 J=1,100
10       CALL ARMIN
C
```

```
C        **********
C THIS SECTION OF CODE PRINTS OUT THE DECIMAL
C        EQUIVALENTS OF THE VALUES FROM THE A/D
C        CONVERTER FOR THAT SEQUENCE OF REQUESTED
C        CHANNELS.
C        **********
C
         WRITE (6,100) (INBUF(I),I=LL,KK)
100      FORMAT (20X,10I5)
C
C        **********
C THIS 'DO LOOP' (TO STATEMENT 13) PROPERLY
C        WEIGHTS THE VALUES ON THE INPUT CHANNELS
C        AND LOADS THEM IN THE OUTBUF REGISTERS(OUTBUF(I)).
C        THE VOLTAGE THAT NOW APPEARS ON THE OUTPUT
C        CHANNELS OF THE D/A CAN BE COMPARED TO THE
C        VOLTAGE LEVELS ON THE INPUT CHANNELS AS WELL
C        AS THE BINARY REPPRESENTATION OF THIS VOLTAGE
C        IN THE COMPUTER AS A MEASURE OF THE ACCURACY
C        OF THE A/D CONVERSION BY ITSELF, AND
C        THE D/A CONVERSION BY ITSELF.  THIS IS IN
C        CONTRAST TO THE INTFC PROGRAM WHICH MEASURES
C        THE TOTAL CONVERSION ERROR WITH AND WITHOUT
C        THE MANIPULATOR THEREBY ALLOWING A MEASURE
C        OF THE ERROR IN THE HARDWARE SERVOS.
C        **********
C
         DO 13 I=1,64
13       OUTBUF(I)=2*INBUF(I)+16384
         CALL ARMOUT(16)
C
C        **********
C THE VALUE OF M IS TESTED HERE TO SEE IF THE
C        BINARY CONVERSION SHOULD BE CARRIED OUT.
C        **********
C
         IF (M) 50,50,22
C
C        **********
C A 15 BIT BINARY NUMBER IS CALCULATED FOR
C        INPUT CHANNEL M BY THE FOLLOWING CODE.
C        BIN(1) IS THE MOST SIGNIFICANT BIT.  IT
C        IS USED HERE AS A SIGN BIT(0 FOR -, 1 FOR +).
C        IF THE INPUT VALUE IS A NEGATIVE NUMBER, THEN
C        BIN (1) IS SET EQUAL TO ZERO AND THE 'DO LOOP'
C        THROUGH STATEMENT 5 IS USED TO DETERMINE
C        THE REMAINING BITS.  IF THE INPUT VALUE IS A POSITIVE
C        NUMBER, THEN BIN(1) IS SET EQUAL TO ONE AND THE
C        'DO LOOP' THROUGH STATEMENT 8 IS USED TO
C        DETERMINE THE REMAINING BITS.
C        **********
C
22       X=8192
         N=-INBUF(M)
         IF (N) 6,7,7
C
C        **********
C THIS IS THE BINARY CONVERSION FOR A NEGATIVE
C        NUMBER ON INPUT CHANNEL M
C        **********
C
```

```
7         BIN(1)=0
          DO 5 I=1,14
          IF (N-X)1,2,2
1         BIN(I+1)=0
          GO TO 5
2         BIN(I+1)=1
          N=N-X
5         X=X/2
          GO TO 19
C
C         **********
C THIS IS THE BINARY CONVERSION FOR A POSITIVE
C         NUMBER ON INPUT CHANNEL M.
C         **********
C
6         BIN(1)=1
          N=N
          DO 8 I=1,14
          IF (N+X)11,11,12
12        BIN(I+1)=0
          GO TO 8
11        BIN(I+1)=1
          N=N+X
8         X=X/2
C
C         **********
C THE BINARY EQUIVALENT OF THE VALUE OF INPUT
C         CHANNEL M IS PRINTED HERE.
C         **********
C
19        WRITE (6,199) BIN
199       FORMAT (5(1X,3I1))
50        CONTINUE
          IF (M) 81,81,80
          END
```

```
C        ****************************************
C
C  SWTH
C        THIS PROGRAM GENERATES A STAIRCASE TEST FUNCTION.
C        BOTH THE LOWER AND UPPER BOUNDARIES ARE ENTERED
C        IN RESPONSE TO QUERIES BY THE PROGRAM.  THESE SET
C        THE BEGINNING AND ENDING VALUES OF A SIMPLE 'DO
C        LOOP'.  THE STEP SIZE IS ALSO ENTERED AND BECOMES
C        THE INCREMENTAL STEP OF THE 'DO LOOP'.
C        THIS PROGRAM CALLS THE SUBROUTINES :
C        ARMOUT : SETERR :
C        SETERR IS A FORTRAN LIBRARY SUBROUTINE WITH TWO
C        ARGUMENTS(SETERR(C,M)).
C        C= AN INTEGER INDICATING THE ERROR CLASS
C           AFFECTED.  C=3 IS THE VALUE FOR ARITHMETIC
C           OVERFLOW.  THIS IS USED IN CASE THE INTEGER
C           IN THE 'DO LOOP' EXCEEDS 32,767 BEFORE THE
C           LOOP IS TERMINATED.
C        M= AN INTEGER THAT INDICATES DIFFERENT INSTRUCTIONS.
C           M=-1 TELLS THE SYSTEM NOT TO LOG MESSAGES, AND
C           TO IGNORE THE ERROR COUNT FOR THE SPECIFIED
C           ERROR CLASS.
C
C        ****************************************
C
         INTEGER ARMBUF,INBUF,OUTBUF
         COMMON/ARMBUF/INBUF(64),OUTBUF(64)
         CALL SETERR(3,-1)
C
C        **********
C  PROGRAM ASKS FOR UPPER BOUNDARY(L), THE STEP SIZE(M),
C        AND THE LOWER BOUNDARY(K) OF THE FUNCTION.
C        **********
C
         WRITE (6,101)
101      FORMAT (' UPPER LIMIT OF SAWTOOTH = '/
         1 2X(0 TO 32767 OUTPUTS +10 TO -10 VOLTS)
         READ (6,100) L
100      FORMAT(I5)
         WRITE (6,102)
102      FORMAT (' SIZE OF STEPS = ')
         READ (6,100)M
         WRITE (6,103)
103      FORMAT (' LOWER LIMIT OF SAWTOOTH = ')
         READ (6,100)K
C
C        **********
C  HERE, THE NESTED LOOP LOADS 16 OUTBUF
C        REGISTERS WITH THE APPROPRIATE VALUES.
C        THE OUTER LOOP CALLS ARMOUT(16) TO OUTPUT
C        THESE VALUES AFTER EACH STEP INCREMENT.
C        **********
C
11       DO 20 I=K,L,M
         DO 10 J=1,16
10       OUTBUF(J)=I
         CALL ARMOUT(16)
20       CONTINUE
C
```

```
C         ***********
C THIS GOTO STATEMENT CAUSES THE STAIRCASE
C         FUNCTION TO REPEAT UNTIL THE PROGRAM IS
C         STOPPED BY THE USER
C         ***********
C
          GO TO 11
          END
```

```
C
C       ****************************************
C
C SNW
C       THIS PROGRAM, WITH ITS SUBROUTINE SAAR, IS
C       PART OF AN ACCEPTANCE PROCEDURE FOR THE MANIPULATOR.
C       IT CAUSES ALL OF THE JOINTS TO BE EXERCISED
C       SIMULTANEOUSLY IN A RYTHMIC FASHION.
C         ALL OF THE OUTPUTS ARE SINE WAVES.  THE
C       FREQUENCY OF EVERY JOINT CAN BE THE SAME OR
C       DIFFERENT FROM EVERY OTHER JOINT.  IF THE
C       FREQUENCIES ARE KEPT CONSTANT, THE PHASE SHIFT
C       OF EACH JOINT CAN BE DIFFERENT.  THE AMPLITUDE
C       OF THE MOTION OF EACH JOINT IS CONTROLLED BY
C       ASSIGNING LIMITING VALUES THROUGH THE SUBROUTINE
C       SAAR.  THE OVERALL FREQUENCY AT WHICH THE SYSTEM RUNS
C       IS DETERMINED BY THE VOLTAGE LEVEL(A POTENTIOMETER)
C       ON A DESIGNATED INPUT CHANNEL.
C
C       ****************************************
C


        IMPLICIT INTEGER (B-R)
        IMPLICIT INTEGER (T-Z)
        COMMON/Z/ZZ
        COMMON/ARMBUF/INBUF(64),OUTBUF(64)
        DIMENSION ASIN(400),K(10),AB(10),B(10),ABB(10),L(10)
C
C       **********
C ZZ IS A INTEGER THAT, WHEN ITS VALUE IS ZERO, CAUSES
C       THE PROGRAM TO WAIT AFTER THE INITIAL VALUES HAVE
C       BEEN SENT TO ALL THE JOINTS.  THE PROGRAM PRINTS OUT
C       A STATEMENT TO THIS EFFECT AND WAITS FOR THE USER
C       TO ENTER A +1 IN ORDER TO CONTINUE.
C       **********
C
        ZZ=0
C
C       **********
C THIS SECTION OF CODE SETS UP A TABLE OF SINE VALUES
C       FROM 1 TO 360 DEGREES TO BE USED IN CALCULATING
C       THE OUTPUTS.
C       **********
C
        DO 35 R=1,360
        AR=R
        AX=(6.28/360.)*AR
35      ASIN(R)=SIN(AX)
C
C        **********
C REQUESTS THE INPUT CHANNEL NUMBER(Q) WHICH WILL
C       CONTROL THE OVERALL FREQUENCY OF THE SYSTEM.
C       **********
C
88      WRITE(6,90)
90      FORMAT(3X'THE FREQ OF THE OUTPUT CHANNELS WILL BE CONTROLLED'/
        1 3X'BY THE VOLTAGE ON A REF INPUT CHANNEL'/3X'REF INPUT
        1 CHANNEL NUMBER =')
        READ(6,93) Q
93      FORMAT(I3)
C
```

```
C        **********
C ASKS WHETHER ALL OF THE OUTPUTS SHOULD BE A CONSTANT
C        FREQUENCY WITH DIFFERENT PHASE SHIFTS(Z=-1), OR IF
C        ALL OF THE OUTPUTS SHOULD BE A DIFFERENT FREQUENCY
C        (Z=+1).
C        **********
C
         WRITE(6,100)
100      FORMAT('   IF ALL OUTPUTS ARE TO BE THE SAME FREQ,'/
         1 3X'BUT HAVE DIFFERENT PHASE SHIFTS, TYPE  -1'/
         1 3X'IF EACH OUTPUT IS TO HAVE A DIFFERENT FREQ, TYPE  +1')
         READ(6,103)Z
103      FORMAT(I3)
C
C        **********
C BRANCHES TO 70 FOR A CONSTANT FREQENCY WITH DIFFERING
C        PHASE SHIFTS, AND TO 72 FOR DIFFERENT FREQUENCIES.
C        **********
C
         IF (Z)70,70,72
C
C        **********
C THIS IS THE BRANCH FOR ALL OUTPUTS HAVING
C        DIFFERENT FREQUENCIES.  THIS SECTION OF CODE REQUESTS
C        EACH OUTPUT CHANNEL NUMBER(K(J)) WITH ITS CORRESPONDING
C        FREQUENCY MULTIPLICATION FACTOR(AB(J)).
C        **********
C
72       DO 220 J=1,7
         WRITE(6,216)
216      FORMAT('   OUTPUT CHANNEL NUMBER =')
         READ(6,217)K(J)
217      FORMAT(I3)
         WRITE(6,218)K(J)
218      FORMAT(30X' FREQ FACTOR FOR CHANNEL',I2,'=' /
         1 30X'(ANY FLOATING POINT NUMBER FROM 0.000 TO 99.000)')
         READ(6,219)AB(J)
219      FORMAT(F6.3)
C
C        **********
C B(J) IS THE INTEGER VALUE OF THE ANGLE FOR
C        JOINT J.  ABB(J) IS THE FLOATING POINT VALUE
C        OF THE ANGLE FOR JOINT J.  INITIALIZING THESE
C        TO A VALUE OF 1 FOR ALL OF THE JOINTS CAUSES
C        ALL OF THE ANGLES TO BEGIN TOGETHER AT I DEGREE.
C        **********
C
         B(J)=1
220      ABB(J)=1
         GO TO 700
C
C        **********
C THIS IS THE BRANCH FOR ALL OUTPUTS HAVING THE
C        SAME FREQUENCY BUT DIFFERENT PHASE SHIFTS.
C        THIS SECTION OF CODE REQUESTS EACH OUTPUT CHANNEL
C        NUMBER(K(JJ)), AND ITS RELATIVE PHASE SHIFT
C        VALUE(B(JJ)).
C        **********
C
```

```
70      DO 115 JJ=1,7
        WRITE(6,111)
111     FORMAT('   OUTPUT CHANNEL NUMBER =')
        READ(6,112)K(JJ)
112     FORMAT(I3)
        WRITE(6,113)K(JJ)
113     FORMAT(30X'PHASE SHIFT OF CHANNEL ',I2,'='/
        1 30X'(ANY ANGLE FROM 0 TO +360)')
        READ(6,114)B(JJ)
114     FORMAT(I4)
C
C       **********
C THE FLOATING POINT VALUE(ABB(J)) OF THE ANGLE
C       IS SET EQUAL TO THE INTEGER VALUE(B(JJ)) OF THE
C       ANGLE WHICH IS ALSO THE PHASE SHIFT ANGLE OF JOINT JJ.
C       THE FREQUENCY MULTIPLICATION FACTOR(AB(JJ)) OF
C       EACH JOINT JJ IS SET EQUAL TO 1 SINCE IN THIS
C       PART OF THE PROGRAM, ALL OF THE OUTPUTS HAVE
C       THE SAME FREQUENCY.
C       **********
C
        ABB(JJ)=B(JJ)
115     AB(JJ)=1
C
C       **********
C THIS 'DO LOOP', FROM STATEMENT 700 TO 400,
C       CALCULATES THE APPROPRIATE OUTPUT VALUES
C       FOR EACH OF THE SEVEN JOINTS(JK).
C       **********
C
700     DO 400 JK=1,7
C
C       **********
C D= THE ANGLE IN DEGREES WHOSE SINE WILL BE
C       THE OUTPUT.  CONVERTED HERE FROM A SUBSCRIPTED
C       TO A NON-SUBSCRIPTED INTEGER.  THIS IS DONE
C       SINCE THIS INTEGER VALUE WILL ITSELF BE USED
C       AS A SUBSCRIPT FORv THE 'ASIN' TABLE.
C       **********
C
        D=B(JK)
C
C       **********
C CHECKS TO SEE IF THE ANGLE(D) HAS EXCEEDED
C       360 DEGREES.  IF SO, 360 IS SUBTRACTED FROM
C       THE ANGLE(BOTH THE INTEGER VALUE(D) AND THE
C       FLOATING POINT VALUE(ABB(JK)).  THIS, THEN,
C       BECOMES THE NEW VALUE OF THE ANGLE.
C       **********
C
        IF(360-D)404,404,405
404     D=D-360
        ABB(JK)=ABB(JK)-360.
C
C       **********
C L BECOMES THE OUTPUT VALUE FOR JOINT JK.  THE
C       VALUE OF THE SINE IS WEIGHTED TO PUT IT IN
C       THE PROPER OPERATING RANGE.
C       **********
C
405     L(JK)=ASIN(D)*16383.+16383.
```

```
C
C         **********
C ABB(JK) IS THE CURRENT FLOATING POINT VALUE
C         OF THE JOINT(JK) ANGLE.  A FLOATING POINT
C         VALUE IS USED SINCE THE ANGLE MAY SOMETIMES
C         BE INCREMENTED BY A VALUE LESS THAN ONE.
C         THIS OCCURS WHEN THE PRODUCT OF AB(JK)*AX
C         IS LESS THAN ONE.  AB(JK) MAY HAVE BEEN
C         ASSIGNED A VALUE LESS THAN ONE BY THE USER(SINCE
C         AB(JK) IS THE FREQUENCY MULTIPLIER FACTOR).  THE
C         VALUE OF AX IS DETERMINED BY THE VOLTAGE LEVEL
C         ON THE DESIGNATED INPUT CHANNEL(Q) WHICH
C         CONTROLS THE OVERALL FREQUENCY OF THE SYSTEM.
C         THUS AX MAY BE <,=, OR > ONE.
C         **********
C
          ABB(JK)=ABB(JK)+AB(JK)*AX
          B(JK)=ABB(JK)
C
C         **********
C THE OUTPUT CHANNEL NUMBER(K(JK)) IS CHANGED
C         FROM A SUBSCRIPTED TO A NON-SUBSCRIPTED(N)
C         INTEGER.
C         **********
C
          N=K(JK)
400       OUTBUF(N)=L(JK)
          CALL SAAR
C
C         **********
C ARMIN IS CALLED TO BRING IN THE VALUE OF
C         THE REFERENCE INPUT CHANNEL(Q) TO CONTROL
C         THE OVERALL FREQUENCY OF THE SYSTEM.
C         **********
C
          CALL ARMIN
C
C         **********
C THE VALUE OF THE REFERENCE INPUT CHANNEL(Q) GOES
C         FROM 0 TO +10 VOLTS PROVIDING INPUT VALUES OF
C         0 TO -8192 RESPECTIVELY.
C            THE FOLLOWING CODE IS USED TO GENERATE A
C         FLOATING POINT NUMBER(AX) THAT TAKES ON
C         VALUES OF 0.012 TO 11.24.  THIS NUMBER IS
C         MULTIPLIED BY THE DELTA INCREASE(AB(JK)) OF
C         THE ANGLE FOR EACH JOINT TO PROVIDE AN OVERALL
C         VARIATION IN THE SPEED THAT THE SYSTEM RUNS.
C         THAT IS, THE SPEED OF THE SYSTEM IS VARIED
C         BY HAVING THE POINTS COME OUT OF THE COMPUTER
C         AT A CONSTANT RATE, WHILE THE SIZE OF THE
C         INCREASE OF EACH JOINT IS CONTROLLED BY THE
C         MAGNITUDE OF THE FACTOR AX.
C         **********
C
          IF (INBUF(Q)+4096)67,67,68
67        AY=(-1)*(INBUF(Q)+4096)
          AX=(AY/400.)+1.
          GO TO 307
68        AW=INBUF(Q)+4096
          AX=1./(((AW/50.)*(AW/4096.))+1.)
307       IF(ZZ)308,308,700
308       WRITE(6,301)
301       FORMAT(10X'THE INITIAL VALUES ARE ON THE OUTPUT CHANNELS'/
         1  10X'IF YOU WISH THE PROGRAM TO CONTINUE, TYPE  +1')
            READ(6,302)ZZ
  302       FORMAT(I3)
            IF(ZZ)88,700,700
            END
```

```
C
C       ****************************************
C
C INTFC
C       THIS IS AN INTERFACE TESTER PROGRAM.  ITS PURPOSE
C       IS TO (1) MEASURE THE AMOUNT OF CROSSTALK BETWEEN
C       ADJACENT INPUT CHANNELS AND (2) MEASURE THE
C       ACCURACY OF THE DIGITAL TO ANALOG (D/A), AND THE
C       ANALOG TO DIGITAL (A/D) CONVERSIONS IN THE INTERFACE.
C         THE PROGRAM REQUESTS AN OUTPUT CHANNEL NUMBER.
C       THROUGH THIS CHANNEL, THE USER HAS THE OPTION OF
C       SENDING EITHER A CONSTANT VALUE, OR A STAIRCASE
C       FUNCTION.  THIS ENTERED VALUE PASSES THROUGH
C       THE D/A ON THE SPECIFIED OUTPUT CHANNEL, WHICH
C       IS CONNECTED TO A SPECIFIED INPUT CHANNEL, INTO
C       THE A/D AND BACK TO THE COMPUTER.  THIS RETURNED
C       NUMBER IS COMPARED TO THE NUMBER THAT WAS SENT
C       AND A MEASURE OF THE ACCURACY OF THE D/A AND
C       A/D CONVERSIONS IS OBTAINED.  IN ADDITION,
C       THE VALUES ON THE INPUT CHANNELS ON EITHER
C       SIDE OF THE SPECIFIED INPUT CHANNEL ARE
C       READ.  THIS ALLOWS A MEASURE OF THE CROSSTALK
C       BETWEEN THESE CHANNELS.
C         THIS PROGRAM CALLS THE SUBROUTINES : ARMIN :
C       : ARMOUT :
C
C       ****************************************
C
        IMPLICIT INTEGER (A-Z)
        COMMON/ARMBUF/INBUF(64),OUTBUF(64)
C
C       **********
C THIS SECTION OF CODE REQUESTS THE OUTPUT
C       CHANNEL NUMBER(R), THE INPUT CHANNEL NUMBER(K),
C       AND WHETHER THE OUTPUT IS TO BE A CONSTANT
C       VALUE(U)(S=1), OR A STAIRCASE FUNCTION(S=0).
C       IF A STAIRCASE FUNCTION, IT ASKS FOR THE
C       STARTING VALUE(T), AND THE STEP SIZE(L).
C       **********
C
20      WRITE (6,100)
100     FORMAT ( ´ OUTBUF  CHANNEL NUMBER= ´ )
        READ (6,101) R
101     FORMAT (I2)
        WRITE (6,115)
115     FORMAT( ´ INBUF CHANNEL NUMBER=´)
        READ (6,125)K
125     FORMAT(I2)
        WRITE(6,150)
150     FORMAT(´ IF OUTPUT CHANNEL IS TO HAVE A CONSTANT VALUE, TYPE 1´)
        WRITE(6,151)
151     FORMAT(´ IF IT IS TO STEP THROUGH A RANGE OF VALUES, TYPE 0´)
        READ(6,152) S
152     FORMAT(I4)
        IF(S)110,110,140
110     WRITE(6,178)
178     FORMAT(´ STARTING VALUE=´)
        READ(6,179) T
179     FORMAT(I7)
        WRITE (6,109)
109     FORMAT ( ´    STEP SIZE= ´)
```

```
         READ (6,105) L
105      FORMAT(I7)
         GO TO 27
140      WRITE(6,159)
159      FORMAT('   CONSTANT VALUE=')
         READ(6,167) U
167      FORMAT(I7)
27       J=0
63       IF(S)75,75,76
75       X=T-L*J
         GO TO 77
76       X=U
C
C        **********
C HERE, J IS USED AS A COUNTING INDEX THAT
C        ALLOWS THE PROGRAM TO CYCLE 100 TIMES.
C        THE PROGRAM THEN STARTS OVER AT THE
C        BEGINNING.
C        **********
C
77       J=J+1
         IF(100-J)64,63,63
C
C        **********
C THE SPECIFIED OUTBUF REGISTER IS LOADED
C        WITH THE PROPER VALUE
C        **********
C
63       OUTBUF(K)=X
         CALL ARMOUT(16)
         WRITE (6,102) K,OUTBUF(K)
102      FORMAT(22X 'OUTBUF(',I2,')=',I7)
         CALL ARMIN
         C=K-1
         D=K+1
C
C        **********
C THE VALUE ON THE SPECIFIED INPUT CHANNEL AS
C        WELL AS THE VALUES ON THE CHANNELS ON EITHER
C        SIDE ARE READ IN AS INBUF(K), INBUF(C), AND
C        INBUF(D).  THESE VALUES ARE THEN
C        WEIGHTED TO YIELD THE INTEGERS N,M,P, WHICH
C        CAN BE COMPARED TO THE NUMBER SENT ON THE
C        OUTPUT CHANNEL.
C        **********
C
         N=2*INBUF(K) + 16384
         M=2*INBUF(C) + 16384
         P=2*INBUF(D) + 16384
C
C        **********
C THIS SECTION WRITES OUT THE INPUT CHANNEL NUMBERS(C,K,D),
C        AND THEIR WEIGHTED VALUES(M,N,P).
C        **********
C
         WRITE(6,103) C,M,K,N,D,P
103      FORMAT(3( '   INBUF(',I2,')=',I7))
C
```

```
C        **********
C THE DIFFERENCE BETWEEN THE OUTPUT VALUE(X)
C        SENT TO THE D/A AND THE VALUE(N) RETURNED
C        FROM THE A/D IS COMPUTED AND PRINTED OUT
C        AS THE ERROR(ERR).  THIS IS A MEASURE OF
C        THE ACCURACY OF THE CONVERSION PROCESS.
C        A MEASURE OF THE NOISE LEVEL OF THE SYSTEM
C        IS OBTAINED BY OBSERVING THE VARIATION IN
C        THE ERROR SIGNAL OVER A NUMBER OF TRIALS.
C          THE MAGNITUDE OF THIS ERROR CAN BE COMPARED
C        TO THE ERROR GENERATED WHEN THE MANIPULATOR
C        IS IN THE LOOP, THEREBY, PROVIDING A MEASURE
C        OF THE ACCURACY OF THE HARDWARE SERVOS OF
C        THE ARM.
C        **********
C
         ERR= N - X
         WRITE(6,130) ERR
130      FORMAT(55X 'ERRØR=',I7,///)
         GO TO 68
64       WRITE(6,200)
200      FORMAT(3X'TO CONTINUE, TYPE [CR].'/
         1 3X'TO SELECT ANOTHER INPUT CHANNEL,'/
         1 3X'TYPE +1.')
         READ(6,201)J
201      FORMAT(I3)
         IF(J)77,77,20
         END
```

```
        SUBROUTINE SAAR
C
C       ****************************************
C
C THIS SUBROUTINE IS USED WITH THE PROGRAM SNW
C       TO GENERATE SINE WAVE OUTPUTS TO THE JOINTS
C       OF THE MANIPULATOR.  THE PURPOSE OF THIS
C       SUBROUTINE IS  (1) TO PREVENT THE ARM FROM
C       TRAVELING TOO GREAT A DISTANCE IN A SINGLE
C       STEP BY PRESCRIBING THE LARGEST AMOUNT
C       THAT THE OUTPUT VALUE IS ALLOWED TO CHANGE,
C       AND (2) TO SET LIMITS ON THE AMPLITUDE OF THE
C       SINE WAVE TO EACH JOINT.
C
C       ****************************************
C
        IMPLICIT INTEGER (A-V)
        IMPLICIT INTEGER (X-Z)
        COMMON/Z/ZZ
        COMMON/ARMBUF/INBUF(64),OUTBUF(64)
        DIMENSION AOTBF(64), CHAN(20,2)
C
C       **********
C ZZ IS THE FLAG FROM THE SNW PROGRAM THAT
C       HAS A VALUE OF 0 FOR THE INITIAL OUTPUT,
C       AND +1 FOR THE REMAINDER OF THE PROGRAM RUN.
C       ZZ IS USED HERE TO PREVENT AN ERROR
C       SIGNAL FROM OCCURRING WITH THE INITIAL
C       OUTPUT.  THE ERROR SIGNAL ARISES WHENEVER
C       THE DIFFERENCE BETWEEN THE PRESENT OUTPUT AND
C       THE PREVIOUS OUTPUT IS GREATER THAN THE
C       SPECIFIED ALLOWABLE CHANGE(B).  FOR
C       THE INITIAL OUTPUT, THE PREVIOUS OUTPUT
C       IS ZERO, THEREFORE, THE DIFFERENCE WILL
C       PROBABLY BE GREATER THAN THE SPECIFIED VALUE(B).
C       THUS, FOR ZZ=0, THE PREVIOUS OUTPUT IS
C       SET EQUAL TO THE INITIAL OUTPUT BEFORE
C       THE DIFFERENCE BETWEEN THEM IS CALCULATED.
C       **********
C
        IF(ZZ)4,4,5
C
C       **********
C THIS BLOCK OF CODE (STATEMENT 4 THROUGH 10) IS
C       EXECUTED FOR THE INITIAL OUTPUTS ONLY.  THIS
C       SECTION OF CODE REQUESTS (1) THE SIZE OF THE
C       ALLOWABLE OUTPUT CHANGE(B), (2) IF LIMITS
C       ARE TO BE SET ON THE AMPLITUDES OF THE SIGNALS
C       TO ANY OF THE JOINTS (YES(LIMIT=-1), NO(LIMIT=+1))
C       AND (3) IF LIMITS ARE TO BE SET, IT ASKS FOR
C       THE LOWER LIMIT(CHAN(IJ,1)) AND THE UPPER
C       LIMIT(CHAN(IJ,2)) FOR EACH JOINT(IJ).
C         THIS SECTION OF CODE ALSO SETS THE PREVIOUS
C       OUTPUT VALUES(AOTBF(J)) EQUAL TO THE PRESENT
C       INITIAL VALUES(OUTBUF(J)) FOR EACH JOINT(J)
C       FOR THE REASON DESCRIBED ABOVE.
C       **********
C
4       WRITE(6,227)
227     FORMAT(3X,'SIZE OF LARGEST ALLOWED OUTPUT CHANGE=',/,
        1 3X,'(0 TO 32767 GIVES +10 TO -10 VOLTS)')
        READ(6,228) B
228     FORMAT(I5)
C
```

```
C       **********
C THIS ´DO LOOP´ SETS THE LOWER(CHAN(KL,1) AND
C       THE UPPER(CHANN(KL,2) LIMITS OF EACH CHANNEL(KL)
C       TO THEIR NORMAL FULL RANGE VALUES
C       OF Ø AND 32,767 RESPECTIVELY.
C       **********
C
        DO 13 KL = 1,2Ø
        CHAN(KL,1)=Ø
13      CHAN(KL,2)=32767
        WRITE(6,111)
111     FORMAT(3X,´ARE ANY CHANNELS TO BE LIMITED ?´,/,
        1 3X,´IF "YES" TYPE -1,  IF "NO" TYPE +1´)
        READ(6,112) LIMIT
112     FORMAT(I5)
        IF(LIMIT.NE.-1) GO TO 5
        WRITE(6,999)
999     FORMAT(3X,´SET LIMITS BETWEEN Ø AND 32767´,/,
        1 3X,´GIVE THE LOWER LIMIT FIRST´)
        DO 9 IJ = 1,7
        WRITE(6,113) IJ
113     FORMAT(3X,´LIMITS ON CHANNEL(´,I1,´) ARE´)
        READ(6,114) CHAN(IJ,1), CHAN(IJ,2)
114     FORMAT(I5)
C
C       **********
C THIS SECTION OF CODE MODIFIES THE SINE WAVE
C       OUTPUT SO THAT THE FULL SWEEP OCCURS
C       WITHIN THE LIMITS SET FOR EACH JOINT.
C       W IS SET EQUAL TO THE SINE WAVE OUTPUT
C       VALUE(OUTBUF(IJ) FOR JOINT IJ.  WI IS SET
C       EQUAL TO THE LOWER LIMIT(CHAN(IJ,1) FOR
C       JOINT IJ. WIL IS SET EQUAL TO THE ALLOWED
C       AMPLITUDE RANGE BETWEEN THE LOWER AND
C       UPPER LIMIT FOR JOINT IJ.  THE
C       NEW OUTPUT IS THEN CALCULATED SO THAT THE
C       FULL AMPLITUDE OF THE SINE WAVE OCCURS
C       BETWEEN THESE SPECIFIED LIMITS.
C       **********
C
        W=OUTBUF(IJ)
        WI=CHAN(IJ,1)
        WIL=CHAN(IJ,2)-CHAN(IJ,1)
        OUTBUF(IJ)=W*WIL/32767.+WI
9       CONTINUE
        DO 1Ø J = 1,16
1Ø      AOTBF(J)=OUTBUF(J)
C
C       **********
C THIS BLOCK OF CODE (FROM STATEMENT 5 TO THE END)
C       IS THE PART OF THE SUBROUTINE CALLED AFTER
C       THE INITIAL VALUES HAVE BEEN SENT.  IT
C       CORRECTS THE AMPLITUDE OF THE SIGNAL TO
C       EACH JOINT IN ACCORD WITH THE LIMITS,
C       AND LOOKS FOR AN ERROR IN THE SIZE OF THE
C       OUTPUT CHANGE.  IF AN ERROR DOES OCCUR,
C       THE PROGRAM STOPS, PRINTS THE SIZE
C       OF THE ERROR AND THE CHANNEL NUMBER IT
C       IS ON AND WAITS FOR A [CR] BEFORE CONTINUING.
C       **********
```

```
C
5        DO 5Ø J=1,16
         W=OUTBUF(J)
         WI=CHAN(J,1)
         WIL=CHAN(J,2)-CHAN(J,1)
         OUTBUF(J)=W*WIL/32767.+WI
         X=AOTBF(J)-OUTBUF(J)
         IF (X)25,5Ø,17
25       X=-X
17       IF (B-X)1,2,2
1        WRITE(6,2ØØ)J,X
2ØØ      FORMAT('  DELTA(',I2,')= ',I6)
         WRITE(6,2Ø1)
2Ø1      FORMAT('  TO CONTINUE, PUSH THE CARRIAGE RETURN')
         READ(6,2Ø2)M
2Ø2      FORMAT(I6)
2        AOTBF(J)=OUTBUF(J)
5Ø       CONTINUE
         CALL ARMOUT(16)
         END
```

Appendix E: <u>ROBOT CONTROL PROGRAMS</u>

These programs allow robot control of a manipulator by a mini-computer. Input commands are read from a master (referred to as the exoskeleton in the documentation) and output commands drive the servos of the slave manipulators.

1.	SANDF:	"Store and forward" for master/slave control. Reads input commands, scales them, and outputs them to the slave manipulator.
2.	ARM:	Basic robot program with continuous path teaching and playback.
3.	PNTPNT:	Point-to-point version of ARM.
4.	NEWARM:	Combines ARM and PNTPNT and adds interpolation if desired on playback.

Note: SANDF needs to know which input channels are which and what scaling factors, if any, are needed to produce the proper output commands to the servos (e.g. in the NBS system the D/A converters require all positive numbers; the value +16,384 is zero volts).

Note: NEWARM is the recommended robot program for general use.

```
        SUBROUTINE SANDF
C
C       SANDF STANDS FOR STORE AND FORWARD
C       IT STORES, SCALES AND FORWARDS THE VALUES READ
C       FROM THE EXOSKELETON (A/D CONVERTED VALUES) TO
C       THE SLAVE ARM (D/A CONVERTED VALUES).
C       ALL VALUES MUST BE SCALED TO POSITIVE QUANTITIES
C       AS THE SIGN BIT OF A NUMBER SENT TO THE D/A CONVERTER
C       WILL RESET THE CHANNEL ADDRESS TO Ø.
C
C THIS PROGRAM IS USED TO GATHER ALL 64 CHANNELS INTO THE COMMON
C AREA ARMBUF IN ARRAY INBUF, THEN OUTPUT THE SCALED VALUES
C FROM THE MASTER EXOSKELETON. THERE ARE 7 DEGREES OF FREEDOM.
C THE 64 CHANNELS INCLUDE MASTER POSITION POTS, SLAVE POSITION
C POTS SENSORS, AND SENSE SWITCHS.
C
        IMPLICIT INTEGER(A-Z)
        COMMON /ARMBUF/INBUF(64),OUTBUF(64)
C
        CALL ARMIN
C
C       **********
C       THE FOLLOWING TWO ´DO LOOPS´ ARE USED TO
C       GENERATE THE CORRECT INPUT CHANNEL NUMBERS
C       (5,8,11,13,15,17,19) FOR THE SEVEN JOINT
C       POTENTIOMETERS.
C       **********
C
        K=2
        DO 5Ø I=1,2
        K=K+3
        OUTBUF(I)=IABS(INBUF(K)*2+16383)
5Ø      CONTINUE
        K=9
        DO 51 I=3,7
        K=K+2
        OUTBUF(I)=IABS(INBUF(K)*2+16383)
51      CONTINUE
C
        RETURN
        END
```

```
C                    ARM
C
C
C    ARM IS A PROGRAM TO STORE AND FORWARD DATA BETWEEN
C        AN INPUT CONTROL DEVICE (EXO-SKELETON) AND A MECHANICAL
C        MANIPULATOR.  AT THE OPERATOR'S REQUEST, ARM RECORDS
C        THE INPUT DATA FOR SUBSEQUENT PLAYBACK -- MUCH LIKE
C        A TAPE RECORDER.
C
         IMPLICIT INTEGER (A-Z)
C
C    DATA IS PASSED BETWEEN THE MAIN PROGRAM AND THE LOWER
C        LEVEL INTERFACE DRIVER ROUTINES THROUGH THE COMMON
C        DATA AREA NAMED ARMBUF.
C
         COMMON/ARMBUF/INBUF(64),OUTBUF(64)
C
C        THE LOCAL BUFFER INCBUF STORES THE INPUT COORDINATES DURING
C        THE RECORDING SESSION.  AT THE OPERATOR'S REQUEST, THIS
C        RECORDED DATA CAN BE SAVED ON DISK OR TAPE.
C
         DIMENSION INCBUF(2000,7)
C
C    FOR CONVENIENCE, A VARIABLE NAMED SWITCH IS EQUIVALENT TO THE
C        INPUT BUFFER ELEMENT ASSIGNED TO CHANNEL NUMBER 27.
C        THIS CHANNEL ENABLES THE OPERATOR TO COMMUNICATE TO
C        THE RUNNING PROGRAM VIA A FUNCTION BUTTON LOCATED NEAR THE
C        INPUT EXO-SKELETON.
C
         EQUIVALENCE(SWITCH,INBUF(27))
C
C    FIRST, THE PROGRAM ACCEPTS A REQUEST FROM THE OPERATOR
C        TO PLAY BACK A PREVIOUSLY RECORDED TRAJECTORY.
C
         WRITE(6,1000)
         READ(6,1100)FILFLG
         IF (FILFLG  .NE. 1HY)GOTO 10
C
C    IF THE OPERATOR DID REQUEST A PREVIOUSLY RECORDED TRAJECTORY
C        THE OLD DATA VALUES ARE READ IN AND CONTROL TRANSFERS
C        TO THE PLAYBACK PORTION OF THE PROGRAM.
C
         WRITE(6,1010)
         DO 2 I=1,2000
2        READ(2,1080,END=3)(INCBUF(I,J),J=1,7)
3        IINC=I-1
         GOTO 300
C
C    IF THE OPERATOR INTENDS TO CREATE A NEW TRAJECTORY, THE PROGRAM
C        FIRST ENTERS A LOOP IN WHICH ALL OF THE JOINT ANGLES
C        FROM THE EXO-SKELETON ARE FORWARDED TO THE MANIPULATOR.
C        ALSO, THE SWITCH (CHANNEL 27) USED BY THE OPERATOR IS
C        SAMPLED ON EACH PASS THROUGH THE LOOP.
C        A COMBINATION OF THE SWITCH VALUE AND THE VARIABLE LATCH
C        IS USED TO CONTROL THE PROGRAM FLOW.
C
10       LATCH=1
         IINC=1
         EOTHR=20
         EOTHRS=EOTHR
         WRITE(6,1020)
```

```
C
C    THE SUBROUTINE SANDF IS CALLED TO STORE AND FORWARD THE JOINT
C        ANGLES FROM THE INPUT EXO-SKELETON TO THE MANIPULATOR.  THIS
C        CALL IS REPEATED UNTIL SWITCH CHANGES STATE.
C
11       CALL SANDF
C
C    A COMBINATION OF SWITCH AND LATCH IS USED TO DECIDE
C        THE NEXT COURSE OF ACTION.
C
         IF(SWITCH .GT. 1000) GOTO (110,120,130,140,150),LATCH
         GOTO (210,220,230,240,250),LATCH
C
C    WHEN RECORDING BEGINS, THE OPERATOR IS INFORMED THAT EOTHR
C        (EVERY OTHER) POINT WILL BE STORED AS A RECORDED POINT
C        IN THE TRAJECTORY.
C
110      WRITE(6,1030)EOTHR
         LATCH=2
120      EOTHRS=EOTHRS-1
         IF(EOTHRS) 121,121,11
121      DO 122 JINC=1,7
122      INCBUF(IINC,JINC)=OUTBUF(JINC)
         EOTHRS=EOTHR
         IINC=IINC+1
         IF(IINC .LE. 2000) GOTO 11
C
C    IF THE LOCAL BUFFER USED TO STORE THE RECORDED TRAJECTORY
C        SHOULD OVERFLOW, THE OPERATOR IS INFORMED.
C
         LATCH = 3
         WRITE(6,1040)
         IINC=IINC-1
130      GOTO 11
140      LATCH = 5
C
C    WHEN THE TRAJECTORY IS FINISHED THE OPERATOR IS INFORMED AND
C        ASKED IF HE WOULD LIKE TO SAVE THE RECORDED POINTS.
C
         WRITE(6,1050)
         READ(6,1100)FILFLG
150      GOTO 11
C
210      GOTO 11
220      IINC=IINC-1
         WRITE(6,1060)
230      LATCH = 4
240      GOTO 11
250      IF(FILFLG .NE. 1HY)GOTO 300
         WRITE(6,1070)
         DO 251 I=1,IINC
251      WRITE(2,1080)(INCBUF(I,J),J=1,7)
C
300      WRITE(6,1090)(INCBUF(1,J),J=1,7),IINC
         READ(6,1100)FILFLG
310      DO 320 I=1,IINC
C
         DO 321 J=1,7
321      OUTBUF(J)=INCBUF(I,J)
         CALL ARMOUT(7)
C
```

```
      CALL ARMIN
C
C   INPUT CHANNEL 26 IS CONNECTED TO A POTENTIOMETER WHICH IS
C       USED TO CONTROL THE SPEED AT WHICH RECORDED TRAJECTORIES
C       ARE PLAYED BACK.  TIMOUT DELAYS EXECUTION OF THE PROGRAM
C       FOR THE SPECIFIED TIME.
C
      CALL TIMOUT((IABS((INBUF(26)/2000)+4)+1))
320   CONTINUE
      GOTO 310
C
1000  FORMAT (' ARM TRAJECTORY RECORD/PLAYBACK DEMON
     1STRATION PROGRAM.'/' DO YOU WANT AN OLD TRAJECTORY?')
1010  FORMAT(' READING DATA VALUES, PLEASE WAIT')
1020  FORMAT(' TOGGLE SWITCH TO RECORD TRAJECTORY')
1030  FORMAT(' BEGIN RECORDING EVERY ',I2,' POINTS')
1040  FORMAT(' FILE OVERFLOW,'/' TRUNCATION OF RECORDED TRAJECTORY
     1 OCCURED: BE EXTREMELY CAREFUL ON PLAYBACK!')
1050  FORMAT(' DO YOU WANT TO SAVE THIS TRAJECTORY?')
1060  FORMAT(' END RECORDING TRAJECTORY. TOGGLE SWITCH TO
     1PLAY BACK')
1070  FORMAT(' PLEASE WAIT WHILE FILE IS BEING SAVED')
1080  FORMAT(7(I6,2X))
1090  FORMAT(' THE INITIAL ARM POSITION IS',/,(7(I6,2X)),/
     1,' THERE ARE',I5,' POSITIONS IN THIS TRAJECTORY',/
     2,' TYPE CARRIAGE RETURN [CR] WHEN READY.')
C
1100  FORMAT(A1)
      END
```

```
C                            PNTPNT
C
C    PNTPNT IS A PROGRAM TO GENERATE TRAJECTORIES BY ALLOWING THE
C        OPERATOR TO POSITION THE MANIPULATOR WITH THE INPUT CONTROL
C        DEVICE -- THE EXO-SKELETON.  ONCE POSITIONED BY THE
C        OPERATOR, THE MANIPULATOR COORDINATES ARE RECORDED WHEN
C        THE OPERATOR PRESSES A FUNCTION BUTTON LOCATED NEAR THE
C        EXO-SKELETON.
C
        IMPLICIT INTEGER (A-Z)
C
C    DATA IS PASSED BETWEEN THE MAIN PROGRAM AND THE LOWER
C        LEVEL INTERFACE DRIVER ROUTINES THROUGH THE COMMON
C        DATA AREA NAMED ARMBUF.
C
        COMMON/ARMBUF/INBUF(64),OUTBUF(64)
C
C    THE LOCAL BUFFER INCBUF STORES THE INPUT COORDINATES DURING
C        THE RECORDING SESSION.  AT THE OPERATOR´S REQUEST, THIS
C        RECORDED DATA CAN BE SAVED ON DISK OR TAPE.
C
        DIMENSION INCBUF(2000,7)
C
C    FOR CONVENIENCE, A VARIABLE NAMED SWITCH IS EQUIVALENT TO THE
C        INPUT BUFFER ELEMENT ASSIGNED TO CHANNEL NUMBER 27.
C        THIS CHANNEL ENABLES THE OPERATOR TO COMMUNICATE TO
C        THE RUNNING PROGRAM VIA A FUNCTION BUTTON LOCATED NEAR THE
C        INPUT EXO-SKELETON.
C        FUNCTION BUTTON 2 IS ASSIGNED TO INPUT CHANNEL NUMBER 28.
C        THIS FUNCTION BUTTON EQUIVALENT TO POINT, IS USED TO
C        IDENTIFY A PARTICUAR SET OF COORDINATES TO BE RECORDED.
C
        EQUIVALENCE(SWITCH,INBUF(27))
        EQUIVALENCE(POINT,INBUF(28))
C
C    FIRST, THE PROGRAM ACCEPTS A REQUEST FROM THE OPERATOR
C        TO PLAY BACK A PREVIOUSLY RECORDED TRAJECTORY.
C
        WRITE(6,1000)
        READ(6,1100) FILFLG
        IF (FILFLG .NE. 1HY)GOTO 10
C
C    IF THE OPERATOR DID REQUEST A PREVIOUSLY RECORDED TRAJECTORY
C        THE OLD DATA VALUES ARE READY IN AND CONTROL TRANSFERS
C        TO THE PLAYBACK PORTION OF THE PROGRAM.
C
        WRITE(6,1010)
        DO 2 I=1,2000
2       READ(2,1080,END=3)(INCBUF(I,J),J=1,7)
3       IINC=I-1
        GOTO 300
C
C    IF THE OPERATOR INTENDS TO CREATE A NEW TRAJECTORY, THE PROGRAM
C        FIRST ENTERS A LOOP IN WHICH ALL OF THE JOINT ANGLES
C        FROM THE EXO-SKELETON ARE FORWARDED TO THE MANIPULATOR.
C        ALSO, THE SWITCH (CHSNNEL 27) USED BY THE OPERATOR IS
C        SAMPLED ON EACH PASS THROUGH THE LOOP.
C        A COMBINATION OF THE SWITCH VALUE AND THE VARIABLE LATCH
C        IS USED TO CONTROL THE PROGRAM FLOW.
C
```

```
10        LATCH=1
          IINC=1
          WRITE(6,1020)
C
C    THE SUBROUTINE SANDF IS CALLED TO STORE AND FORWARD THE JOINT
C         ANGLES FROM THE INPUT EXO-SKELETON TO THE MANIPULATOR.  THIS
C         CALL IS REPEATED UNTIL SWITCH CHANGES STATE.
C
11        CALL SANDF
C
C    A COMBINATION OF SWITCH AND LATCH IS USED TO DECIDE
C         THE NEXT COURSE OF ACTION.
C
          IF(SWITCH .GT. 1000) GOTO (110,120,130,140,150),LATCH
          GOTO (210,220,230,240,250),LATCH
C
C    WHEN RECORDING BEGINS, THE OPERATOR IS INFORMED THAT FUNCTION
C         BUTTON NUMBER 2, (INPUT ON CHANNEL NUMBER 28) IS USED TO
C         RECORD THE LOCATION OF THE MANIPULATOR AT ITS CURRENT
C         POSITION.
C
110       WRITE(6,1030)
          LATHC=2
120       IF(POINT .LT. 1000)GOTO 11
121       DO 122 JINC=1,7
122       INCBUF(IINC,JINC)=OUTBUF(JINC)
          EOTHRS=EOTHR
          IINC=IINC+1
          IF(IINC .LE. 2000) GOTO 11
C
C    IF THE LOCAL BUFFER USED TO STORE THE RECORDED TRAJECTORY
C         SHOULD OVERFLOW, THE OPERATOR IS INFORMED.
C
          LATCH = 3
          WRITE(6,1040)
          IINC=IINC-1
130       GOTO 11
140       LATCH = 5
C
C    WHEN THE TRAJECTORY IS FINISHED THE OPERATOR IS INFORMED AND
C         ASKED IF HE WOULD LIKE TO SAVE THE RECORDED POINTS.
C
          WRITE(6,1050)
          READ(6,1100)FILFLG
150       GOTO 11
C
210       GOTO 11
220       IINC=IINC-1
          WRITE(6,1060)
230       LATCH = 4
240       GOTO 11
250       IF(FILFLG .NE. 1HY)GOTO 300
          WRITE(6,1070)
          DO 251 I=1,IINC
251       WRITE(2,1080)(INCBUF(I,J),J=1,7)
C
300       WRITE(6,1090)(INCBUF(1,J),J=1,7),IINC
          READ(6,1100)FILFLG
310       DO 320 I=1,IINC
C
```

```
        DO 321 J=1,7
321     OUTBUF(J)=INCBUF(I,J)
        CALL ARMOUT(7)
C
        CALL ARMIN
C
C    INPUT CHANNEL 26 IS CONNECTED TO A POTENTIOMETER WHICH IS
C       USED TO CONTROL THE SPEED AT WHICH RECORDED TRAJECTORIES
C       ARE PLAYED BACK.  TIMOUT DELAYS EXECUTION OF THE PROGRAM
C       FOR THE SPECIFIED TIME.
C
        CALL TIMOUT((IABS((INBUF(26)/2000)+4)+1))
320     CONTINUE
        GOTO 310
C
1000    FORMAT(' ARM TRAJECTORY RECORD/PLAYBACK DEMON
        1STRATION PROGRAM.'/' DO YOU WANT AN OLD TRAJECTORY?')
1010    FORMAT(' READING DATA VALUES, PLEASE WAIT')
1020    FORMAT(' TOGGLE SWITCH TO RECORD TRAJECTORY')
1030    FORMAT(' POSITION THE MANIPULATOR TO THE DESIRED COORDINATES',
        1' THEN, PUSH BUTTON 2 TO RECORD THE POINT.')
1040    FORMAT(' FILE OVERFLOW,'/' TRUNCATION OF RECORDED TRAJECTORY
        1 OCCURED:BE EXTREMELY CAREFUL ON PLAYBACK!')
1050    FORMAT(' DO YOU WANT TO SAVE THIS TRAJECTORY?')
1060    FORMAT(' END RECORDING TRAJECTORY.  TOGGLE SWITCH TO
        1PLAY BACK')
1070    FORMAT(' PLEASE WAIT WHILE FILE IS BEING SAVED')
1080    FORMAT(7(I6,2X))
1090    FORMAT(' THE INITIAL ARM POSITION IS',/,(7(I6,2X)),/
        1,' THERE ARE',I5,' POSITIONS IN THIS TRAJECTORY',/
        2,' TYPE CARRIAGE RETURN [CR] WHEN READY.')
C
1100    FORMAT(A1)
        END
```

```
C
C       ****************************************
C
C NEWARM
C       THIS PROGRAM IS A MODIFIED VERSION OF THE ARM
C       PROGRAM.  A LARGE PART OF THE CODE, THEREFORE,
C       IS IDENTICAL AND WILL NOT BE DOCUMENTED HERE.
C         LIKE ARM, THIS PROGRAM ALLOWS
C       OPERATOR TO RECORD A DESIRED TRAJECTORY AND
C       EITHER PLAY IT BACK IMMEDIATELY OR STORE IT
C       ON DISC TO BE PLAYED BACK AT A LATER TIME.
C         RECORDING THE TRAJECTORY CONSISTS IN STORING
C       THE VALUES OF THE POTENTIOMETERS ON EACH OF THE
C       JOINTS AS THE MANIPULATOR MOVES THROUGH A SEQUENCE
C       OF MOVEMENTS.  TO PLAYBACK, THESE VALUES ARE SENT
C       IN THE SAME SEQUENCE TO THE SERVOS OF ALL
C       OF THE JOINTS.
C         THIS PROGRAM PROVIDES THE OPERATOR WITH THREE
C       OPTIONS WITH REGARDS TO THE MANNER OF RECORDING
C       THE TRAJECTORY.
C               (1) A CONTINUOUS PATH RECORDING IDENTICAL
C       TO THE ARM PROGRAM.  BY CONTINUOUS PATH IS MEANT
C       THAT THE POTENTIOMETER VALUES ARE CLOCKED IN
C       WITH A CONSTANT TIME INTERVAL THROUGHOUT THE
C       RECORDING SESSION.
C               (2) A POINT-TO-POINT RECORDING.  HERE,
C       THE POTENTIOMETER VALUES ARE READ INTO THE
C       COMPUTER ONLY ON COMMAND FROM THE OPERATOR.
C       THE [CR] IS DEPRESSED EACH TIME A POINT IS TO
C       BE READ.
C               (3)A TWO POINT TRAJECTORY.  IN THIS
C       CASE ONLY THE STARTING POINT AND THE
C       ENDING POINT ARE RECORDED.
C         DURING THE PLAYBACK OF BOTH THE CONTINUOUS
C       PATH AND THE POINT-TO-POINT TRAJECTORIES, AN
C       INTERPOLATION ROUTINE IS USED TO GENERATE
C       ADDITIONAL POINTS BETWEEN THE RECORDED POINTS.
C       THIS PROVIDES A SMOOTHER PLAYBACK TRAJECTORY,
C       AND ALSO PROVIDES A CONTROL ON THE SPEED THAT
C       THE TRAJECTORY RUNS.  THIS SPEED CONTROL IS
C       ACCOMPLISHED BY SENDING POINTS TO THE ARM AT
C       A CONSTANT RATE WHILE THE NUMBER OF ADDITIONAL
C       POINTS GENERATED BETWEEN THE RECORDED POINTS
C       IS A FUNCTION OF THE VOLTAGE LEVEL ON INPUT
C       CHANNEL 26(THIS IS THE POTENTIOMETER ON THE FUNCTION
C       KEYBOARD.  IT IS ALSO THE VALUE Q FOR THE SNW
C       PROGRAM.).
C         THE TWO POINT TRAJECTORY DOES NOT INCORPORATE
C       THIS INTERPOLATION ROUTINE.  THUS DURING
C       PLAYBACK THE ARM TRAVELS AT ITS HIGHEST
C       VELOCITY FROM THE STARTING TO THE ENDING
C       POINT.  ITS ABILITY TO PERFORM THIS TYPE
C       OF TRAJECTORY TO AN END POINT WITHIN ITS
C       SPECIFIED ACCURACY IS PART OF THE ACCEPTANCE
C       PROCEDURE.
C
C       ****************************************
C
```

```
        IMPLICIT INTEGER (A-Z)
        COMMON/ARMBUF/INBUF(64),OUTBUF(64)
        DIMENSION INCBUF(2000,7),START(10),FINI(10)
        REAL NDEL,DBUF(20)
        EQUIVALENCE(SWITCH,INBUF(27)),(SWIT2,INBUF(28))
C
C       **********
C ZZ IS USED AS A FLAG TO CAUSE THE PROGRAM
C       TO WAIT AFTER SENDING THE FIRST POINT IN THE
C       TRAJECTORY.  THE ARM SERVOS CAN NOW BE TURNED ON
C       AND THE ARM ALLOWED TO POSITION ITSELF.  IF THE
C       ARM SERVOS TO THE CORRECT INITIAL POSITION, THEN
C       A [CR] IS TYPED WHICH SETS ZZ=0 AND CAUSES THE
C       TRAJECTORY TO BE PLAYED BACK.
C       **********
C

3000    ZZ=-1
C
C       **********
C THE SUBROUTINE SETERR WAS EXPLAINED IN THE
C       PROGRAM SWTH.
C       **********
C
        CALL SETERR(3,-1)
1       WRITE(6,444)
444     FORMAT(' RECORD/PLAYBACK DEMONSTRATION PROGRAM'/3X,/
        1' IF YOU WANT A CONTINUOUS TRAJECTORY------ TYPE+1'/
        2' IF YOU WANT A POINT TO POINT TRAJECTORY-- TYPE 0'/
        3' IF YOU WANT A TWO POINT TRAJECTORY------- TYPE -1')
        READ(6,445)PC
445     FORMAT(I5)
        IF(PC)76,75,74
C
C       **********
C THIS SECTION OF CODE IS FOR THE CONTINUOUS
C       PATH RECORDING AND IS IDENTICAL TO THE ARM
C       PROGRAM.
C       **********
C
74      WRITE(6,1000)
        READ(6,1100)FILFLG
        IF (FILFLG .NE. 1HY)GOTO 10
        WRITE(6,1010)
        DO 2 I=1,2000
2       READ(2,1080,END=3)(INCBUF(I,J),J=1,7)
3       IINC=I-1
        GOTO 300
C
10      LATCH=1
        IINC=1
        EOTHR=20
        EOTHRS=EOTHR
        WRITE(6,1020)
C
11      CALL SANDF
C
C       **********
C THE ARM IS NOW POSITIONED TO THE READ VALUE
C       NOW CHECK THE SWITCH AND LATCH TO DECIDE
C       THE NEXT COURSE OF ACTION.
C       **********
```

```
C
        IF(SWITCH .LT. -2000) GOTO (110,120,130,140,150),LATCH
        GOTO (210,220,230,240,250),LATCH
C
110     WRITE(6,1030)EOTHR
        LATCH=2
120     EOTHRS=EOTHRS-1
        IF(EOTHRS) 121,121,11
121     DO 122 JINC=1,7
122     INCBUF(IINC,JINC)=OUTBUF(JINC)
        EOTHRS=EOTHR
        IINC=IINC+1
        IF(IINC .LE. 2000) GOTO 11
        LATCH = 3
        WRITE(6,1040)
        IINC=IINC-1
130     GOTO 11
140     LATCH = 5
        WRITE(6,1050)
        READ(6,1100)FILFLG
        WRITE(6,1220)
1220    FORMAT(2X'FLIP SWITCH DOWN TO INITIALIZE TRAJECTORY')
150     GOTO 11
C
210     GOTO 11
220     IINC=IINC-1
        WRITE(6,1060)
230     LATCH = 4
240     GOTO 11
250     IF(FILFLG .NE. 1HY)GOTO 300
        WRITE(6,1070)
        DO 251 I=1,IINC
251     WRITE(2,1080)(INCBUF(I,J),J=1,7)
C
300     WRITE(6,1090)(INCBUF(1,J),J=1,7),IINC
        READ(6,1100)FILFLG
310     KINC=IINC-1
C
C       **********
C THIS IS THE INTERPOLATION ROUTINE.  NDEL
C       IS SET EQUAL TO THE ABSOLUTE VALUE OF
C       INPUT CHANNEL 26 DIVIDED BY 700.  THE
C       NUMBER ONE IS ADDED TO THIS FRACTION SO
C       THAT NDEL WILL NOT BE EQUAL TO ZERO(BECAUSE
C       THE INTEGER CONVERSION OF A FRACTION LESS
C       THAN ONE WILL YIELD A ZERO.  THIS RESULT
C       IS MULTIPLIED BY 10 SO THAT THERE IS A
C       MINIMUM OF 10 POINTS GENERATED BETWEEN EACH
C       RECORDED POINT.  THAT IS, THE INTEGER
C       VALUE OF NDEL IS THE NUMBER OF POINTS THAT
C       WILL BE GENERATED BETWEEN EACH PAIR OF
C       RECORDED POINTS.
C       *********
C
        DO 320 I=1,KINC
        INBUF(26)=IABS(INBUF(26))
        NDEL=(INBUF(26)/700+1)*10
C
C       *********
C THE DIFFERENCE BETWEEN TWO SUCCESSIVE RECORDED
C       POINTS IS DIVIDED BY NDEL TO OBTAIN A DELTA
C       (DBUF(J)) FOR EACH JOINT J.
C       **********
```

```
C
        DO 321 J=1,7
321     DBUF(J)=(INCBUF(I+1,J)-INCBUF(I,J))/NDEL
        KDEL=NDEL
C
C       **********
C THIS SECTION OF CODE IS REPEATED FOR EACH
C       RECORDED POINT.  IT ADDS THE DELTA(DBUF(J)
C       TO THE RECORDED POINT NDEL TIMES,
C       THEREBY GENERATING NDEL POINTS BETWEEN
C       EACH RECORDED POINT
C       **********
C
        DO 322  K=1,KDEL
        DO 323 J=1,7
323     OUTBUF(J)=INCBUF(I,J)+DBUF(J)*(K-1)
        CALL ARMOUT(7)
322     CALL ARMIN
320     CONTINUE
        WRITE(6,1221)
1221    FORMAT(2X'TO REPEAT THIS TRAJECTORY, TYPE [CR].',
        1 2X'TO END PROGRAM, TYPE A +1',/
        1 2X'TO RETURN TO BEGINNING, TYPE A -1.')
        READ(6,1222)TE
1222    FORMAT(I3)
        IF(TE)3000,310,9300
C
1000    FORMAT(' ARM TRAJECTORY RECORD/PLAYBACK DEMON
        1STRATION PROGRAM.'/' DO YOU WANT AN OLD TRAJECTORY?')
1010    FORMAT(' READING DATA VALUES, PLEASE WAIT')
1020    FORMAT('  FLIP SWITCH TO UP POSITION TO RECORD TRAJECTORY')
1030    FORMAT(' BEGIN RECORDING EVERY ',I2,' POINTS',/
        1 '   FLIP SWITCH TO DOWN POSITION TO END RECORDING.')
1040    FORMAT(' FILE OVERFLOW,'/' TRUNCATION OF RECORDED TRAJECTORY
        1 OCCURED: BE EXTREMELY CAREFUL ON PLAYBACK!')
1050    FORMAT(' DO YOU WANT TO SAVE THIS TRAJECTORY?')
1060    FORMAT(' END RECORDING TRAJECTORY.  FLIP SWITCH TO',
        1 2X'UP POSITION TO PLAY BACK.')
1070    FORMAT(' PLEASE WAIT WHILE FILE IS BEING SAVED')
1080    FORMAT(7(I6,2X))
1090    FORMAT(' THE INITIAL ARM POSITION IS',/,(7(I6,2X)),/
        1,' THERE ARE',I5,' POSITIONS IN THIS TRAJECTORY',/
        2,' TYPE CARRIAGE RETURN [CR] WHEN READY.')
C
1100    FORMAT(A1)
C
C       **********
C THIS SECTION OF CODE IS FOR THE TWO POINT
C       RECORDED TRAJECTORY.  IT FIRST ASKS THE
C       OPERATOR IF A PREVIOUSLY RECORDED TRAJECTORY
C       IS DESIRED.  IF YES (HUH=Y), THE OLD
C       TRAJECTORY IS READ FROM THE DISC.
C       **********
C
76      WRITE(6,888)
888     FORMAT(' PROGRAM FOR TWO POINT TRAJECTORY'/
        1 ' DO YOU WANT AN OLD TRAJECTORY?')
        READ(6,446)HUH
446     FORMAT(A1)
        IF(HUH.NE.1HY) GO TO 400
        READ(2,1080)(START(I),I=1,7),(FINI(I),I=1,7)
        GO TO 432
C
C       **********
```

```
C IF A NEW TRAJECTORY IS TO BE RECORDED(HUH=N)
C       THE PROGRAM INSTRUCTS THE OPERATOR TO FLIP THE
C       SWITCH ON INPUT CHANNEL 27 TO RECORD THE
C       BEGINNING POINT.
C       **********
C
400     WRITE(6,431)
431     FORMAT(2X´FLIP SWITCH #1 TO UP POSITION TO RECORD FIRST POIN
C
C       **********
C THE PROGRAM CYCLES UNTIL THE SWITCH IS
C       FLIPPED.
C       **********
C
447     CONTINUE
        CALL SANDF
        IF(SWITCH.GT.-2000) GO TO 447
        CALL SANDF
        DO 555 I = 1,7
555     START(I)=OUTBUF(I)
C
C       **********
C THE PROGRAM INSTRUCTS THE OPERATOR TO FLIP
C       THE SWITCH ON INPUT CHANNEL 28 TO RECORD
C       THE END POINT.
C       **********
C
        WRITE(6,448)
448     FORMAT(2X´FLIP SWITCH #2 TO UP POSITION TO RECORD END POINT´
C
C       **********
C AGAIN, THE PROGRAM CYCLES UNTIL THE
C       SWITCH IS FLIPPED.
C       **********
C
449     CONTINUE
        CALL SANDF
        IF(SWIT2.GT.-2000) GO TO 449
        CALL SANDF
        DO 554 I = 1,7
554     FINI(I)=OUTBUF(I)
C
C       **********
C THE PROGRAM ASKS THE OPERATOR IF THIS
C       RECORDED TRAJECTORY IS TO BE SAVED ON THE DISC.
C       **********
C
        WRITE(6,666)
        READ(6,661)HUH
661     FORMAT(A1)
666     FORMAT(´ DO YOU WANT TO SAVE THESE POINTS ON DISC?´)
        IF(HUH.NE.1HY) GO TO 665
C
C       **********
C IF THE ANSWER IS YES(HUH=Y) THEN THE TRAJECTORY
C       IS WRITEN ON THE DISC BEFORE PLAYBACK.
C       ***********
C
        WRITE(2,1080)(START(I),I=1,7),(FINI(I),I=1,7)
665     WRITE(6,450)
450     FORMAT(´ DO YOU WANT TO PLAYBACK?´)
        READ(6,889)HUH
```

```
889     FORMAT(A1)
        IF(HUH.NE.1HY) GO TO 76
432     WRITE(6,452)
452     FORMAT(' TYPE CARRIAGE RETURN [CR] TO INITIALIZE ARM')
        READ(6,453)HUH
453     FORMAT(I5)
C
C       **********
C THE BEGINNING POINT IS LOADED INTO THE OUTBUF
C       REGISTERS, SENT TO THE ARM, AND THE PROGRAM
C       THEN WAITS FOR A [CR] TO CONTINUE.
C       **********
C
460     DO 454 I = 1,7
454     OUTBUF(I)=START(I)
        CALL ARMOUT(7)
        WRITE(6,890)
890     FORMAT(' TYPE CARRIAGE RETURN [CR] WHEN READY')
        READ(6,891)HUH
891     FORMAT(I3)
C
C       **********
C THE END POINT IS LOADED INTO THE OUTBUF REGISTERS
C       AND SENT TO THE ARM.  THE PROGRAM ASKS THE OPERATOR
C       IF HE WISHES TO REPEAT THIS TRAJECTORY.
C       **********
C
        DO 455 I= 1,7
455     OUTBUF(I)=FINI(I)
        CALL ARMOUT(7)
        WRITE(6,892)
892     FORMAT(2X'TYPE [CR] TO REPEAT THIS TRAJECTORY',/
        1 2X'TYPE +1 TO END THE PROGRAM',/
        1 2X'TYPE -1 TO RETURN TO THE BEGINNING OF THE PROGRAM.')
        READ(6,893)HUH
893     FORMAT(I3)
C
C       **********
C IF THE TRAJECTORY IS TO BE REPEATED, THE PROGRAM
C       GOES TO STATEMENT 452 AND WAITS TO  OUTPUT THE FIRST
C       POINT.  IF THE TRAJECTORY IS NOT TO BE
C       REPEATED, CONTROL RETURNS TO THE BEGINNING
C       OF THE PROGRAM (STATEMENT 3000).
C       **********
C
        IF(HUH)3000,432,9300
C
C       **********
C THIS SECTION OF CODE IS USED TO RECORD THE
C       POINT-TO-POINT TRAJECTORY.  THIS CODE IS
C       IDENTICAL TO THAT USED IN THE CONTINUOUS
C       PATH SECTION WITH THE EXCEPTION OF A READ
C       STATEMENT (AFTER STATEMENT 517) WHICH CAUSES
C       THE PROGRAM TO STOP AFTER A POINT IS READ IN.
C       TYPING THE [CR] CAUSES THE NEXT POINT TO BE
C       RECORDED.
C       **********
C
75      CALL SETERR(3,-1)
        WRITE(6,2000)
```

```
2000    FORMAT(' POINT TO POINT DEMONSTRATION PROGRAM'/
        1 ' DO YOU WANT AN OLD TRAJECTORY?')
        READ(6,1100)FILFLG
        IF (FILFLG .NE. 1HY)GOTO 500
        WRITE(6,1010)
        DO 501 I=1,2000
501     READ(2,1080,END=503)(INCBUF(I,J),J=1,7)
503     IINC=I-1
        GOTO 504
C
500     LATCH=1
        IINC=1
        EOTHR=20
        EOTHRS=EOTHR
        WRITE(6,1020)
C
505     CALL SANDF
C
C       **********
C THE ARM IS NOW POSITIONED TO THE READ VALUE
C       NOW CHECK THE SWITCH AND LATCH TO DECIDE
C       THE NEXT COURSE OF ACTION.
C       **********
C
        IF(SWITCH .LT. -2000) GOTO (506,507,508,509,510),LATCH
        GOTO (511,512,513,514,515),LATCH
C
506     WRITE(6,7270)
7270    FORMAT(2X'TYPE [CR] EVERY TIME A POINT IS TO BE RECORDED.'/
        1 2X'FLIP SWITCH TO DOWN POSITION AND TYPE A [CR]',/
        1 2X'TO END THE RECORDING.')
        LATCH=2
507     EOTHRS=EOTHRS-1
        IF(EOTHRS) 516,516,505
516     DO 517 JINC=1,7
517     INCBUF(IINC,JINC)=OUTBUF(JINC)
        READ(6,913)FF
913     FORMAT(I3)
        EOTHRS=EOTHR
        IINC=IINC+1
        IF(IINC .LE. 2000) GOTO 505
        LATCH = 3
        WRITE(6,1040)
        IINC=IINC-1
508     GOTO 505
509     LATCH = 5
        WRITE(6,1050)
        READ(6,1100)FILFLG
        WRITE(6,1220)
510     GOTO 505
C
511     GOTO 505
512     IINC=IINC-1
        WRITE(6,1060)
513     LATCH = 4
514     GOTO 505
515     IF(FILFLG .NE. 1HY)GOTO 504
        WRITE(6,1070)
        DO 518 I=1,IINC
```

```
518      WRITE(2,1080)(INCBUF(I,J),J=1,7)
C
504      WRITE(6,1090)(INCBUF(1,J),J=1,7),IINC
519      KINC=IINC-1
         ZZ=-1
C        WRITE(6,7666)
C7666    FORMAT(6X´TYPE [CR] TO RUN TRAJECTORY´)
C        READ(6,7667)JII
C7667    FORMAT(I3)
C
C        **********
C THIS IS THE INTERPOLATION ROUTINE AS
C        EXPLAINED IN THE CONTINUOUS PATH
C        SECTION.
C        **********
C
         DO 520 I=1,KINC
         INBUF(26)=IABS(INBUF(26))
         NDEL=(INBUF(26)/700+1)*10
         DO 521 J=1,7
521      DBUF(J)=(INCBUF(I+1,J)-INCBUF(I,J))/NDEL
         KDEL=NDEL
         DO 522  K=1,KDEL
         DO 523 J=1,7
523      OUTBUF(J)=INCBUF(I,J)+DBUF(J)*(K-1)
         CALL ARMOUT(7)
         IF(ZZ)830,831,831
830      READ(6,850)ZZ
850      FORMAT(I3)
831      CONTINUE
522      CALL ARMIN
520      CONTINUE
         WRITE(6,7631)
7631     FORMAT(2X´TYPE [CR] TO REPEAT THIS TRAJECTORY´,/
         1 2X´TYPE +1 TO END THE PROGRAM´,/
         1 2X´TYPE -1 TO RETURN TO THE BEGINNING OF THE PROGRAM.´)
         READ(6,7632)HUG
7632     FORMAT(I3)
         IF(HUG)3000,504,9300
C
C
9300     CONTINUE
         END
```

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET**	1. PUBLICATION OR REPORT NO. NBSIR 75-973	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE GUIDE TO IMPROVING THE PERFORMANCE OF A MANIPULATOR SYSTEM FOR NUCLEAR FUEL HANDLING THROUGH COMPUTER CONTROLS.	5. Publication Date November 1975
	6. Performing Organization Code
7. AUTHOR(S) Drs. John M. Evans, Jr., James S. Albus & Anthony J. Barbera and Mr. Robert Ronsenthal and Mr. William B. Truitt	8. Performing Organ. Report No.
9. PERFORMING ORGANIZATION NAME AND ADDRESS NATIONAL BUREAU OF STANDARDS DEPARTMENT OF COMMERCE WASHINGTON, D.C. 20234	10. Project/Task/Work Unit No. 6006111
	11. Contract/Grant No.
12. Sponsoring Organization Name and Complete Address *(Street, City, State, ZIP)* Same as iten 9	13. Type of Report & Period Covered Final
	14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

This is an Interagency report to be presented to Oak Ridge National Laboratory, Oak Ridge, Tennessee.

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

The Office of Developmental Automation and Control Technology of the Institute for Computer Sciences and Technology of the National Bureau of Standards provides advising services, standards and guidelines on interface and computer control systems, and performance specifications for the procurement and use of computer controlled manipulators and other computer based automation systems. These outputs help other agencies and industry apply this technology to increase productivity and improve work quality by removing men from hazardous environments.

In FY 74 personnel from the Oak Ridge National Laboratory visited NBS to discuss the feasibility of using computer control technqiues to improve the operation of remote control manipulators in nuclear fuel reprocessing. Subsequent discussions led to an agreement for NBS to develop a conceptual design for such a computer control system for the PaR Model 3000 manipulator in the Thorium Uranium Recycle Facility (TURF) at ORNL. This report provides the required analysis and conceptual design. Complete computer programs are included for testing of computer interfaces and for actual robot control in both point-to-point and continuous path modes.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

Computer control of automation systems; computer interface; computer software for robot control; hierarchical control of manipulators; position servos; trajectory control.

18. AVAILABILITY [✓ Unlimited] [ ] For Official Distribution. Do Not Release to NTIS [ ] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13 [✓] Order From National Technical Information Service (NTIS) Springfield, Virginia 22151	19. SECURITY CLASS (THIS REPORT) UNCLASSIFIED 20. SECURITY CLASS (THIS PAGE) UNCLASSIFIED	21. NO. OF PAGES 54 22. Price $4.50