AN INSTRUMENTATION LABORATORY COMPUTER SYSTEM

by

GREG DECI

B. S., Kansas State University, 1977

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
1979

Approved by

[illegible]

Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

With the maturing of computer technology there has been a growing trend to use computers as more than just "number-crunchers". In order to use computers in these alternate ways it became obvious that keypunching computer cards and running them through a card-reader was an inadequate means of entering data. Many of the applications that computers lend themselves to involve processing data from laboratory measurements, so it was advantageous to design interfaces that allow measuring devices (instrumentation) to transmit their data directly to the processing computer. With the advent of such computer compatible instrumentation and inexpensive mini- and micro-computers, computer instrumentation systems have become increasingly popular.

A very appealing feature of computer instrumentation is its ability to make faster and more reliable measurements than a system which uses a human operator. With the traditional means of testing a circuit, the operator would clip a test probe to the first test point, select the range and mode for the measuring equipment, attach the test equipment needed to provide the desired stimulus to the circuit, adjust the controls to provide the desired stimulus, and finally read the value indicated on the measurement device (often on an analog scale). After taking and recording this measurement the operator would then move the probe to the next test point and repeat the whole process. Even with an experienced operator each of these operations takes several seconds and, more importantly, each provides a possible source of error. In particular, when the measurements are repetitive the operator tends to become inattentive and may misplace a probe or misread an analog scale.

With a computer controlled system, repetitive measurements can be made with a repeatable procedure from unit to unit. Because the time the computer needs to program the test equipment to its proper settings and to take a reading is much smaller than the time a human needs to do the same operation, throughput for testing units is greatly increased. These advantages are not gained without cost. While the time to test each unit is greatly decreased, the "front end" investment in time and equipment is much larger. Test equipment that is computer compatible is more expensive than comparable manual equipment. A test fixture to interface units to the instrumentation system must be built. Finally, the software to control the sequence of events in the test procedure must be written and debugged. For these reasons automated testing currently dominates the testing of complex, mass-produced products, but is only practical for product development when very repeatable test procedures are imperative.

Automated test equipment probably accounts for a bulk of current computer instrumentation systems, but there are also other applications of such systems. Computer instrumentation systems are used as part of the feedback loop in automated control of industrial processes. These systems continually monitor various parameters of the process (temperature, humidity, chemical concentrations, etc.) and adjust equipment to keep those parameters within given limits.

A variation of such a system is a "data-logging" system, which acts as a monitor rather than a controller. An advantage of using a computer system for this job instead of a strip chart or similar device is the computer's ability to make a "value judgement" on the readings it is taking. During periods of little or normal activity the computer can slow down its sample rate to reduce the amount of data that must later be analyzed. When the computer detects abnormal or rapidly varying activity it can then increase its sampling rate to record the event more completely. The computer can also be programmed to set some sort of flag to indicate that abnormal readings were encountered.

Computerized instrumentation can also be used to transform raw data into a more presentable form without first manually transcribing the data into a computer readable form. One of the most apparent uses of this feature is in the linearization and conversion of transducer readings. For instance, a commonly used transducer for measuring temperature is the thermocouple. This device produces a voltage that is a function of the temperature of the thermocouple. However, the relationship is not linear. In the past, conversion tables or an intricate system of biasing diodes to switch in different resistance values at different voltage levels were used to linearize the output. With a computer monitoring the output of the thermocouple, the computer can mathematically manipulate the reading to provide an output in any of the common temperature scales desired.

Other data reduction that would be very useful to have the computer perform on laboratory measurements would be statistical analysis and automatic generation of plots and graphs of the data.

A final reason for computer instrumentation is that it can perform some jobs that cannot be effectively done otherwise. Any situation that requires measurements to be made many times a second can only be handled with a computerized system. Systems that can make 20,000 measurements a second are readily available and ones that make over 1 million measurements a second are not uncommon. A practical application of this type of measurement is a low frequency spectrum analyzer which can provide a far greater range of capabilities and improved performance using fast Fourier transform algorithms than similar equipment using swept band-pass filters.

Computer instrumentation offers appealing solutions to many problems encountered in developing instrumentation systems. Once the software needed to use the computer in an instrumentation system is developed, the computer can become a valuable tool in the field of instrumentation.

INSTRUMENTATION LABORATORY COMPUTER SYSTEM

In order to give students "hands-on" experience with computer instrumentation, a PDP11/03 minicomputer was purchased for the instrumentation laboratory in the Electrical Engineering Department at Kansas State University. Along with the CPU the system has the following configuration: dual floppy disk drive; 16 K words (16-bit word) read/write memory; a 16-channel, 12-bit Analog-to-Digital converter; a 4-channel, 12-bit Digital-to-Analog converter; a real-time clock; a serial interface (selectable to 300 baud for the Decwriter, or 4800 baud for the CRT display); an interface card compatible with the IEEE-488 interface standard; a Decwriter; and a CRT display/keyboard. Software purchased with this system includes two monitors (a foreground/background monitor and a single job monitor), a FORTRAN compiler, an assembler, a scientific subroutine package, and an editor for building ASCII files on disk. To use this minicomputer system in the laboratory a large amount of software and some hardware needed to be developed.

The PDP11/03 computer system was added to the instrumentation laboratory to provide students with experience in the ways that a computer can be used in instrumentation systems and the problems encountered in trying to interface analog signals to the computer system. To prevent the desired laboratory experience from being lost in a frustrating exercise in computer programming, it is imperative that the software developed for the system allow the student to control as many functions as possible with a minimum of programming. With this in mind, the software was developed in the form of a large library of FORTRAN-callable subroutines. Early in the development of the software two fundamental decisions were made which enabled the software to be developed as rapidly as possible and kept the system complexity to a minimum. The price of these decisions has been a restriction on the flexibility and future growth of the system.

The first decision was that the single user monitor (RT-11) provided by DEC would be the monitor that the students would operate under. The other possibilities were to use DEC's foreground/background monitor, a multiprogramming monitor in which a program that is not time-critical (background job) is executed when execution of the foreground job is suspended, or to write a laboratory monitor that would run under DEC's single job monitor. The foreground/background monitor would allow a resident program to take periodic measurements of an event with a long time constant and execute other programs between measurement times. The problem with this approach is that most programs that are run on the system are "real-time" programs and, as such, do not lend themselves to being run in a background environment.

Writing a new monitor directed specifically toward student use would have many advantages over using DEC's monitor. Most importantly, it would protect the system from the students. (By using DEC's general purpose monitor it is possible to modify or destroy files on the system disk which are needed to run the computer.) A special monitor would also allow a less complicated procedure for developing and running a student's program than the general purpose monitor requires. Finally, using DEC's monitor means that the students will have to gain at least a minimal understanding of DEC's operating system to use the computer. While the advantages that would be gained by writing a custom monitor appear to be very strong, the overhead required to implement that option (both time and money) make it unreasonable.

The second decision was that the computer's interrupt capability would not be used. Since only one job will be running at a time it was decided that system complexity would be greatly reduced at small performance cost by not using interrupts. Instead, when the program needs to know if an event has occurred yet, it goes into a loop and waits until an appropriate bit is set or cleared. The drawbacks to this method are: much CPU time is wasted in wait loops, and the time until the computer responds to an event could be as long as the time needed to execute the wait loop once. Since there is not much else the CPU can be doing while waiting for an event in a single job environment, the first drawback is acceptable in this system. For occasions where the response time is critical (i.e. reading an A/D conversion before another is completed), the second drawback is overcome by writing the wait loop in assembler, thereby providing as fast a response as an interrupt routine would.

The software written for the laboratory can be grouped under three catagories: software to use the features of the PDP11/03 system, software to interface with IEEE-488 instrument bus compatible devices, and the versatile miscellaneous category A description of each of the subroutines is given in appendix B. All of the subroutines are collected in a single library along with several of DEC's FORTRAN-callable system subroutines. This library is on the system disk along with the system monitor and FORTRAN compiler. (For a more complete description of the system disk see appendix A.)

The subroutines that access the peripherals of the PDP11/03 system do the bookkeeping necessary to use the ADV11-A Analog-to-Digital Converter[1], the AAV11-A 4-channel 12-bit D/A Converter[2], and the KWV11-A Programmable Real-time Clock[3]. Three of the subroutines are written in assembler (DAC, RDAD, and RDBLK) in order to minimize their execution time. Sampling rates of up to 4 KHz. (using RDAD for each reading) or 12.5 KHz. (reading a block of measurements with RDBLK) can be obtained using these subroutines.

The second group of subroutines are those which allow the user to interface with the IEEE-488 bus. These routines do the bookkeeping needed to use DEC's IBV11-A LSI-11/Instrument Bus Interface[4]. Since the IEEE-488 bus standard is growing in

---

[1]For further description of the ADV11-A see Microcomputer Handbook, Digital Equipment Corp., 2nd Edition, sections I-4.15 and I-5.15.

[2]For further description of the AAV11-A see Microcomputer Handbook, Digital Equipment Corp., 2nd Edition, sections I-4.14 and I-5.14.

[3]For further description of the KWV11-A see Microcomputer Handbook, Digital Equipment Corp., 2nd Edition, sections I-4.18 and I-5.17.

[4]For further description of the IBV11-A see Microcomputer Handbook, Digital Equipment Corp., 2nd Edition, sections I-4.13 and I-5.13.

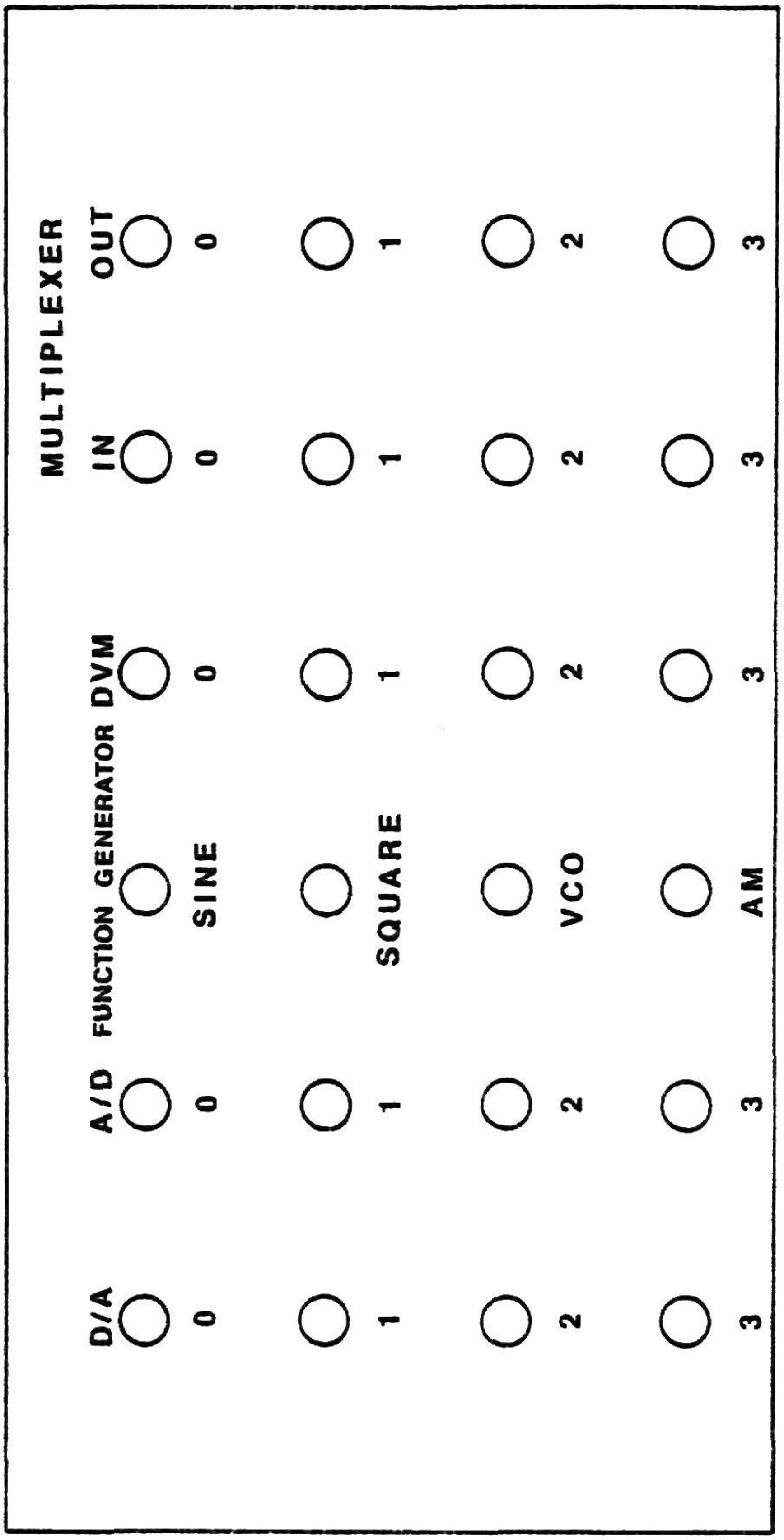
acceptance and is extremely useful in configuring a computerized instrumentation system, it will be emphasized in the lectures that accompany the instrumentation laboratory. (For a description of the IEEE-488 bus standard see appendix C.) In order to give the student an understanding of the step-by-step working of the bus, most of the subroutines were written to allow individual commands to be sent on the bus rather than combining them into functional blocks. The one exception to this is the routine SNDMSG which sends a string of characters to a specified listener.

The final group of subroutines are those written to make life easier for the student and to allow him to use the computer earlier in the semester. Currently, the only commercial IEEE-488 bus compatible instrument in the laboratory is an HP3455 voltmeter donated by the Hewlett-Packard Company. Two of the subroutines in the library allow students to take readings with this voltmeter without understanding anything about the IEEE-488 bus. In this way simple measurements can be made using the computer and the voltmeter before the IEEE-488 bus is covered in lecture. In addition to these subroutines there is a stand-alone plot routine that will take a file of data points and generate a computer printout graph.

Along with the software written for the laboratory, a few items of hardware were needed. The most obvious need was for a convenient, low-noise means of connecting the A/D and D/A converters to the desired points of the circuit being examined. BNC connectors and the associated co-axial cabling were finally selected. A patch panel with 24 BNC connectors (6x4) was built (see Fig. 1) and provides connections to other circuitry as well. Between each A/D channel's BNC connector and the A/D converter there is a unity gain differential amplifier and a 2-pole, active, low-pass filter with a cut-off frequency of 10 KHz. There is also a 2-pole, active, low-pass filter between the D/A converters and the D/A connectors on the patch panel. (For further detail on the interfacing to the A/D and D/A converters see appendix D.)

As mentioned earlier, the only commercial IEEE-488 compatible test equipment currently in the laboratory is an HP3455 voltmeter. Until further equipment is purchased, several test functions will be performed by student-built IEEE-488 compatible test equipment. This equipment includes a function generator, a frequency counter, and a multiplexer. Further information on these items is given in appendices E and F. The current configuration of the laboratory's computer system and associated equipment is shown in Fig. 2.

The computer instrumentation system that is currently being used is sufficient to demonstrate that it is possible to use computers in solving instrumentation problems. However, some improvement is needed for it to be more representative of a practical system. To upgrade the system the following modifications are desirable. First, the "home-built" test equipment needs to be replaced with quality, commercial IEEE-488 bus compatible equivalents. Power supplies that are compatible with the IEEE-488 bus are needed to examine the effects of power supply fluctuations on the devices being tested. Less crit-



PATCH PANEL

FIGURE 1

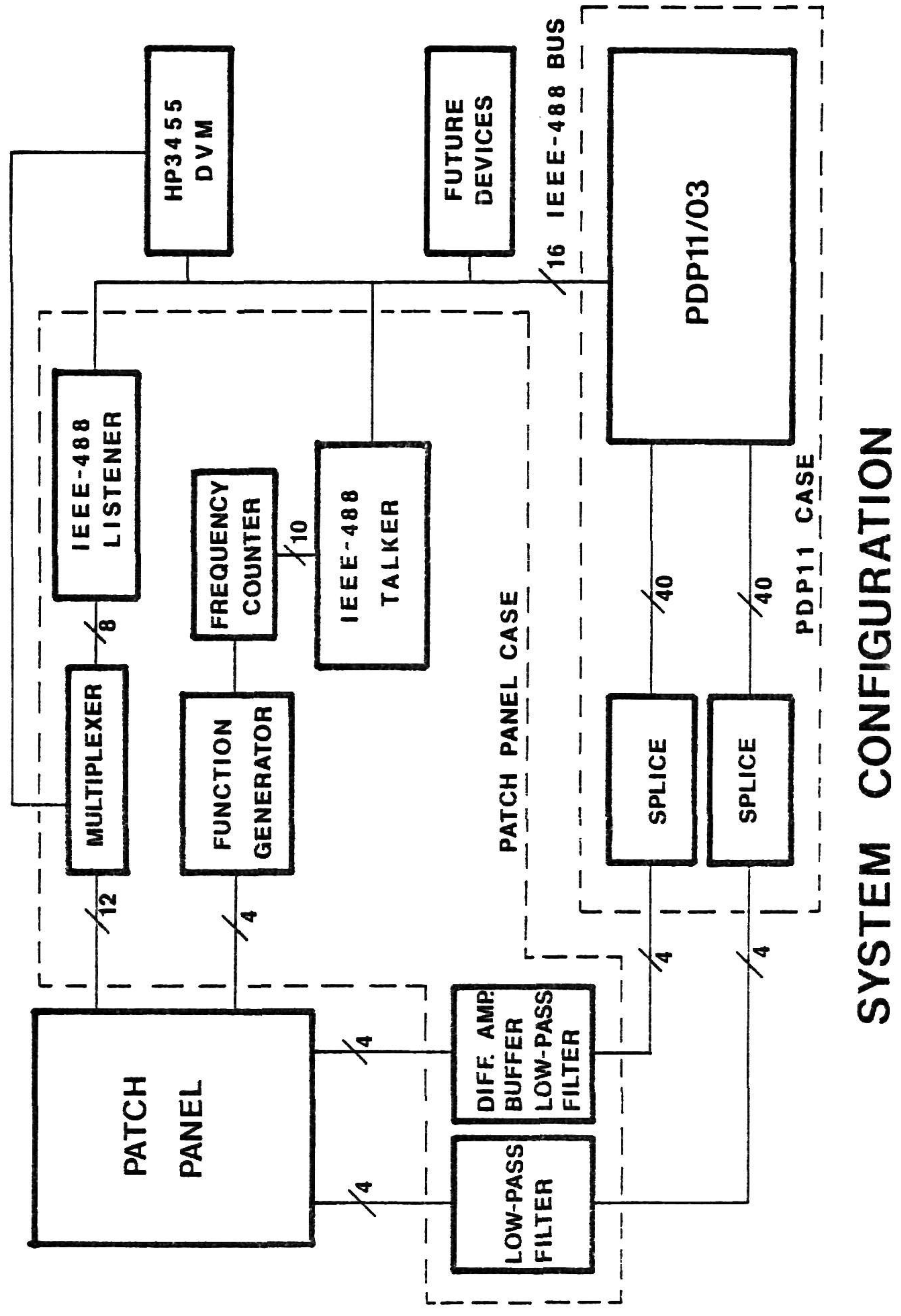

SYSTEM CONFIGURATION
HP3455 DVM
FUTURE DEVICES
IEEE-488 BUS
16
PDP11/03
IEEE-488 LISTENER
8
MULTIPLEXER
12
FREQUENCY COUNTER
10
IEEE-488 TALKER
FUNCTION GENERATOR
4
PATCH PANEL CASE
PATCH PANEL
4
4
DIFF. AMP. BUFFER LOW-PASS FILTER
LOW-PASS FILTER
SPLICE
SPLICE
40
40
4
4
PDP11 CASE

FIGURE 2

ical, but desirable, is a redesign of the filter/buffer circuits for the D/A and A/D converters. The current design is functional, but the error due to the offset voltages of the buffers nullifies the accuracy of the converters. Finally, a second computer/controller with a monitor and an architecture more friendly toward program development is needed.

ACKNOWLEDGMENTS

I am grateful to several people for their help in developing the computer system and preparing this report:

To D. R. Palmer for the work he contributed to the development of the system; in particular, for designing and building the Function Generator/Frequency Counter and for writing the "PLOT" program.

To my wife, Sandy, for her encouragement and help in preparing this report.

To Professor M. S. P. Lucas for his sage advice whenever I got stuck.

BIBLIOGRAPHY

[1] J. Finkel, Computer-aided Experimentation: Interfacing to Minicomputers. New York: John Wiley & sons, 1975

[2] IBV-11/Instrument Bus Interface User's Manual. Maynard, Mass.: Digital Equipment Corp., 1977

[3] IEEE Standard 488-1975, "Digital Interface for Programmable Instrumentation," New York: The IEEE, Inc., 1975

[4] MCS-48 Microcomputer User's Manual. Santa Clara, CA: Intel, 1977

[5] Microcomputer Handbook. Maynard, Mass.: Digital Equipment Corp., 1975

[6] PDP-11 FORTRAN Language Reference Manual. Maynard, Mass.: Digital Equipment Corp., 1975

[7] RT-11 System Message Manual. Maynard, Mass.: Digital Equipment Corp., 1976

[8] RT-11 System Reference Manual. Maynard, Mass.: Digital Equipment Corp., 1976

[9] RT-11/RSTS/E FORTRAN IV User's Guide. Maynard, Mass.: Digital Equipment Corp., 1975

APPENDIX A

SYSTEM DISK

The system disk used for the instrumentation laboratory was configured with two considerations in mind. The first was that the system should be as easy as possible to use, and the second was that the files on the system disk should be protected from the user. Since all of the experiments that use the computer can be completed using FORTRAN programs, the assembler was omitted from the disk. All subroutines needed are included in either the "FORLIB" or "LABLIB" library file on the system disk. The stand alone program "PLOT" and two of its data files ("RECT" and "SEMI") are also on the system disk to allow the user access to the program without having to create a copy of the program on his own disk. The following list describes the purpose of the files on the system disk:

MONITR.SYS	"MONITR" is the RT-11 single-job monitor. It is the resident program that accepts commands from the teletype and translates them into actions by the computer. "TT" is the teletype device handler routine. It tells the monitor how to talk to the teletype (in this case the Decwriter).
EDIT .SAV	"EDIT" is a program that allows interactive modification or creation of ASCII data files. It is used to manipulate the FORTRAN source programs written for the laboratory.
FORTRA.SAV FORTRA.HLP	The "FORTRA" program converts a FORTRAN file into a lower level language. The file created is called an object file. "FORTRA.HLP" contains the error diagnostics used by "FORTRA.SAV".

FORLIB.OBJ	"FORLIB" is a library of the standard FORTRAN built-in functions, a few subroutines needed to access some system functions (for a more complete description of the system functions see the RT-11/RSTS/E FORTRAN IV USER'S GUIDE, Appendix B), and the code needed to convert "FORTRA"'s output file to machine executable form.
LINK .SAV	"LINK" takes object files (either from "FORTRA" or from a library file), assigns absolute memory locations to the variables, and "links" the addresses of subroutines to subroutine call statements.
PLOT .SAV RECT .DAT SEMI .DAT	The program "PLOT" will generate a rough plot from data points stored in a file on the user's disk. More information on "PLOT" can be found in appendices B and G of this report and the notebook labeled "PLOT".
LABLIB.OBJ	"LABLIB" is a library of FORTRAN callable subroutines. These subroutines come from two sources, some are part of DEC's "SYSLIB" library and the rest were written for the instrumentation laboratory. The following is a list of the subroutines in "LABLIB" and where a description of them can be found.

The following subroutines are described in appendix O of RT-11 SYSTEM REFERENCE MANUAL.

SUBROUTINE	PARAGRAPH
CVTTIM	0.3.5
GTIM	0.3.9
IPEEK	0.3.31
IPOKE	0.3.32
ISPY	0.3.44
JTIME	0.3.63
SECNDS	0.3.80
TIMASC	0.3.84
TIME	0.3.85

The following subroutines are described in appendix B of this report.

DAC	SENDC
DIGVLT	SENDM
DVMCVT	SETAD
IFC	SETCLK
POLL	SNDMSG
RCV	SYNC
RDAD	TRGAD
RDBLK	VLTDIG
RDVM	

APPENDIX B

INSTRUMENTATION LABORATORY SOFTWARE

The software written for the instrumentation laboratory consists of seventeen subroutines that are a part of the "LABLIB" library on the system disk and one stand alone program ("PLOT") that also resides on the system disk. Source listings for the subroutines can be found in the computer printout notebook labeled "LABLIB SUBROUTINE LISTINGS". Detailed descriptions of the subroutines, the parameters they use, and any subtle points about their use can be found in the computer printout notebook labeled "LABLIB SUBROUTINE DESCRIPTIONS". A detailed description and listing of "PLOT" can be found in the notebook labeled "PLOT". In the pages that follow, the routines are listed and a brief verbal description of each routine is included. For the routines whose execution time is fixed and important to the subroutine's usage it has been included in the description.

DAC: Digital to Analog Conversion: This subroutine enables the user to select one of four 12-bit D/A channels and convert an offset binary number (0=-5.12 V., 4096=5.12 V.) to an equivalent output voltage. The execution time of this subroutine is approx. 170 uSec.

DIGVLT: Digit to Voltage: This subroutine converts an offset binary number to a number that is equal to the voltage the binary number represents. This is useful in cases where A/D conversions must be used in calculations or output for examination. The execution time of this subroutine is approx. 920 uSec.

DVMCVT: Digital Voltmeter Convert: This Subroutine takes an array of 15 ASCII characters arranged in the format used by the HP3455 DVM (<+/-> D.DDDDDDE <+/-> DD <CR> <LF>) and converts it into the equivalent floating point number. The execution time for this subroutine is approx. 9.7 mSec.

IFC: Interface Clear: This subroutine puts the IEEE-488 bus into a known state. It unlistens and untalks all devices on the bus and makes the PDP11/03 the controller-in-charge.

POLL: Conduct Serial Poll: This subroutine conducts a serial poll of the current talker on the IEEE-488 bus. It returns a status byte from that talker as a parameter.

RCV: Receive: This subroutine causes the PDP11/03 to act as a listener on the IEEE-488 bus and returns one byte received from the bus.

RDAD: Read Analog to Digital Converter: This subroutine waits for an A/D conversion to be completed and returns the converted value. The channel that the A/D conversion is done on must have already been selected (with SETAD) and an A/D conversion triggered before this routine is called. The execution time of this subroutine is approximatly 208 uSec.

RDBLK: Read Block: This subroutine fills successive memory locations (a one dimensional array) with values from an A/D converter channel. The number of readings wanted is passed as a parameter. The A/D conversions must be triggered by the real-time clock and the A/D channel desired must be selected before RDBLK is called. This subroutine is used when A/D conversions must be made faster than FORTRAN programming will permit (RDBLK is written in assembler).

RDVM:	Read Digital Voltmeter: This subroutine triggers the HP3455 to take a reading and returns the floating point equivalent of the reading.
SENDC:	Send Command: This subroutine sends a one byte command on the IEEE-488 bus.
SENDM:	Send Message: This subroutine sends one byte on the IEEE-488 bus as a message byte.
SETAD:	Set Analog to Digital Converter: This subroutine selects the A/D channel that the next conversion will be made from and whether the conversion will be triggered by software or by the real-time clock. The execution time for this subroutine is approx. 1.4 mSec.
SETCLK:	Set Real-time Clock: This subroutine sets the real-time clock to produce a pulse train with a given period. The execution time of the subroutine is approx. 3.9 mSec.
SNDMSG:	Send Message: This subroutine sends an array of bytes (one at a time) on the IEEE-488 bus. This subroutine is extremely useful for configuring the HP3455 DVM.
SYNC:	Synchronize: This program is used to keep the calling program in time with the real-time clock. The subroutine does not return control to the calling program until a real-time clock output occurs.

TRGAD: Trigger an Analog to Digital Conversion: This subroutine triggers the start of an A/D conversion if the A/D converter is set for software triggering. Execution time for this subroutine is approx. 830 uSec.

VLTDIG: Voltage to Digit: This subroutine converts a number that is equivalent to a given voltage into its offset binary representation. This is useful when it is desirable to have the D/A converter output a given voltage. Execution time for this subroutine is approx. 820 uSec.

PLOT: Plot Program: This is a stand alone program that takes a data file generated by another program and makes a computer printout graph of the data. If it is wanted, the program will list the X-Y pairs contained in the data file. The Y-axis of the graph is always linear and is marked off into 5 major sections with 10 minor divisions for each major one. The X-axis can be selected as either log or linear scale. If log is selected, the scale is five decades starting from the lefthand value provided. If linear is selected, the axis is divided into 5 major sections with 20 minor divisions in each. In all cases the computer generates the graph and the major division labels based on the upper and lower bounds for both axes.

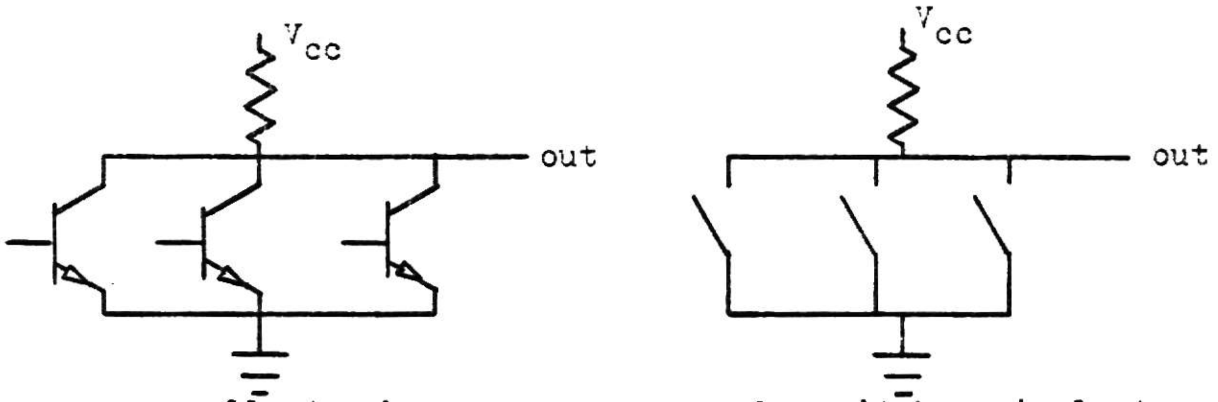
APPENDIX C

IEEE-488 INTERFACE BUS

Until recently, in order to design a computer controlled instrumentation system the designer had to build a separate interface for each piece of test equipment. Some form of addressing had to be created, the data formats of the test equipment had to be made compatible with the computer, and some form of handshaking had to be devised. The overhead required to build such an interface system was so large that automated test set-ups were only economical for systems that were to be mass-produced, or for measurements that could not be done by hand.

To alleviate this problem the IEEE-488 bus standard was developed. The standard provides a uniform protocol and bus structure to be used in instrumentation systems. The IEEE-488 bus standard is modeled on the "Hewlett-Packard Interface Bus" (HPIB) developed by the Hewlett-Packard Company. The bus structure used by the IEEE-488 standard is an open-collector bus structure.

An open-collector bus structure can be thought of as several SPST switches connected in parallel with one side tied to ground and the other side connected through a resistor to V_{cc} (Fig. C-1). When any of the switches is closed, the output is pulled low with the current drawn through the resistor causing the voltage drop of V_{cc}. If all of the switches are open, no current flows through the resistor and the output goes to V_{cc}. Since the passive output state of the bus is V_{cc}



a. open-collector bus b. switch equivalent

FIG. C-1

and the actively driven state is ground, open-collector systems are often described with negative logic. (In negative logic "1" is assigned to the lower voltage and "0" is assigned to the higher voltage.) In negative logic, the function performed by the open-collector bus is an "OR" function and the bus structure is sometimes referred to as a wire-or. The logic used in the IEEE-488 standard is negative logic. This is a source of confusion when first reading the standard.

The IEEE-488 bus standard uses the byte (8 bits) as its basic information unit. Eight data lines are used so a byte is transmitted in parallel with successive bytes transmitted serially. Since a byte is the basic information unit, it is often convenient to describe information sent on the bus in terms of ASCII characters. In addition to the eight data lines there are five general interface management lines (IFC, ATN, SRQ, REN, EOI) and three data byte transfer control (handshake) lines.

Much of the versatility of the IEEE-488 bus standard comes from the three-wire, open-collector handshake protocol defined by the standard. This handshake allows data to be transmitted as fast as the sending and receiving devices can handle it. The same handshake is used for transmission of both commands and data. Only one device is allowed to transmit at any one time, but many devices can be listening at the same time. By using an open-collector structure for the handshake lines and defining the handshakes so completion is indicated by the high state of the handshake lines, the handshake is completed by the slowest device receiving the byte (not necessarily the slowest device on the bus). In this way, data is transmitted asynchronously at the maximum rate the slowest participating device can handle. (See diagram at the end of this appendix for an example of the handshake.)

Devices on the bus can perform three functions (or some subset of the three): controller, talker, and listener. The minimum useful application of the bus has a device configured to always be a talker (i.e. a DVM) sending messages to a device configured to always be a listener (i.e. a printer). In general, the bus will be used in a computer controlled system, and the computer will be controlling the configurations of the devices attached to the bus.

The controller "directs traffic" on the bus. It selects which devices are to be listeners, which are to be talkers, and monitors the general interface management lines. The commands used by the controller are shown in Fig. C-2. The controller sends a command by forcing the attention line (ATN) low, putting the command on the data lines, then going through the handshake procedure described before. An interrupt capability is provided through the SRQ line. When SRQ goes low (indicating a device needs service) the controller conducts a

0 0 0 0 0 0 0 1	GO TO LOCAL
0 0 0 0 0 1 0 0	SELECTED DEVICE CLEAR
0 0 0 0 0 1 0 1	PARALLEL POLL CONFIGURE
0 0 0 0 1 0 0 0	GROUP EXECUTE TRIGGER
0 0 0 0 1 0 0 1	TAKE CONTROL
0 0 0 1 0 0 0 1	LOCAL LOCKOUT
0 0 0 1 0 1 0 0	DEVICE CLEAR
0 0 0 1 0 1 0 1	PARALLEL POLL UNCONFIGURE
0 0 0 1 1 0 0 0	SERIAL POLL ENABLE
0 0 0 1 1 0 0 1	SERIAL POLL DISABLE
0 0 1 X X X X X	MY LISTEN ADDRESS
0 0 1 1 1 1 1 1	UNLISTEN
0 1 0 X X X X X	MY TALK ADDRESS
0 1 0 1 1 1 1 1	UNTALK

GO TO LOCAL: Device is under the control of its manual switches

SELECTED DEVICE CLEAR: All listeners are cleared

PARALLEL POLL CONFIGURE: First step in parallel poll

GROUP EXECUTE TRIGGER: Triggers addressed listeners

TAKE CONTROL: Tells currently addressed talker to be controller

LOCAL LOCKOUT: Disables device's manual controls

DEVICE CLEAR: Clears all devices on bus

PARALLEL POLL UNCONFIGURE: Releases devices set for parallel poll

SERIAL POLL ENABLE: Causes addressed talker to enter serial poll

SERIAL POLL DISABLE: Returns device to a talker

MY LISTEN ADDRESS: Causes addressed device to become a listener

UNLISTEN: Causes all listeners to be unaddressed

MY TALK ADDRESS: Causes addressed device to be a talker

UNTALK: Causes current talker to be unaddressed

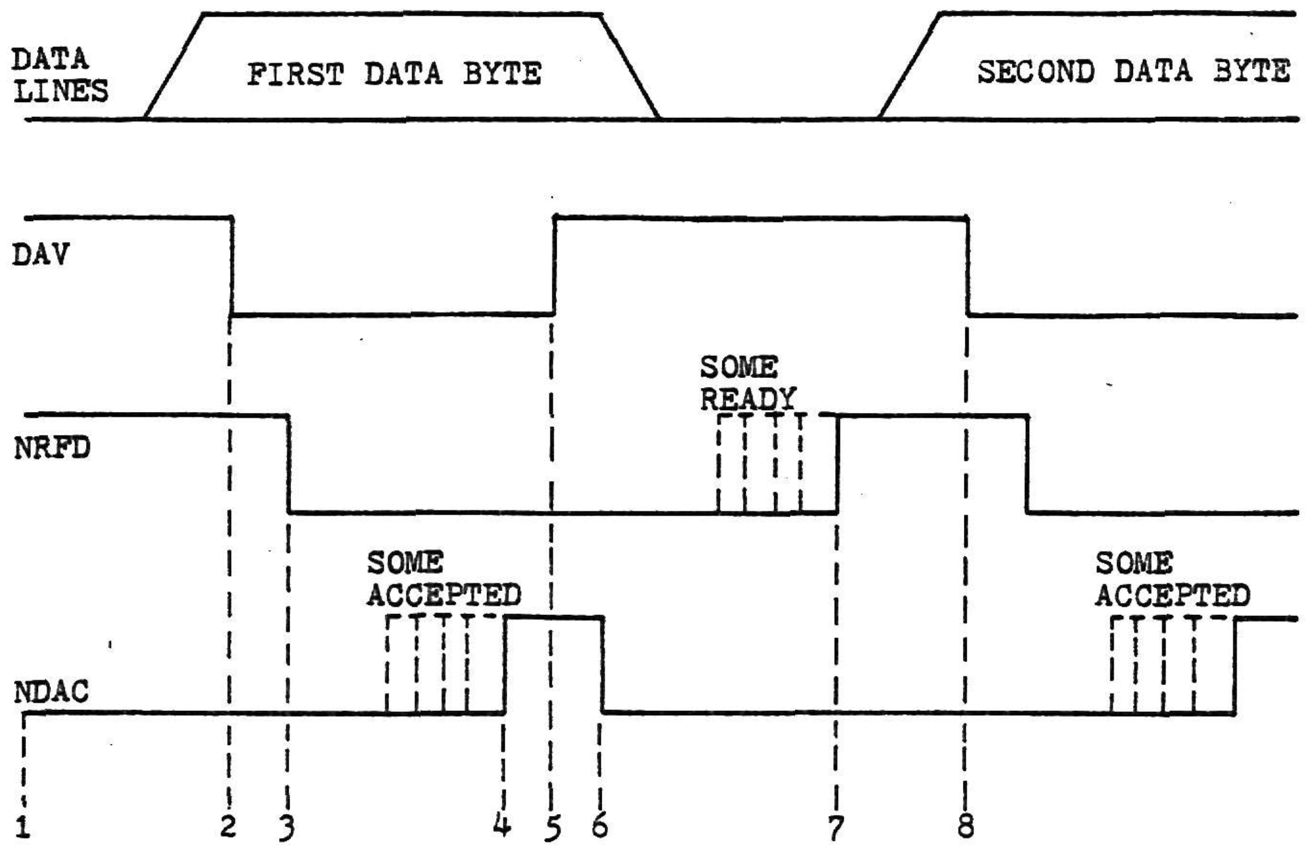
FIG. C-2
INTERFACE BUS COMMANDS

serial poll, asking each device what its status is. (A parallel poll is also possible but will not be described here.)

A talker sends information on the bus. While active, it controls the DAV handshake line and monitors the NRFD and NDAC lines to see when the handshake is completed. When a serial poll is being conducted, a talker responds to the ATN line going high by sending a status byte. (Bit 7 of the status byte is the status of the talker's service request flag.)

A listener receives information from the bus. Each listener monitors DAV and influences the state of the NRFD and NDAC lines. When the controller has ATN pulled low (indicating a command byte is being transmitted) all other devices are listeners. If commands are not being sent, only addressed listeners participate in the handshake.

Along with defining a bus protocol, the standard also defines the electrical specifications and mechanical connections needed to be compatible with the bus. The bus can support up to 15 devices attached at once, and a total transmission length of up to 20 meters. The maximum data rate that can be obtained with the bus is one million bytes/second and is typically much less than that. Although the IEEE-488 bus standard appears formidable at first glance, it tends to become reasonably "friendly" as one works with it.



1. Handshake lines are set for next data byte. DAV:high, NRFD:high, NDAC:low

2. Talker has put data byte on data lines and drives DAV low to indicate the data lines are valid.

3. After DAV goes low, all listeners drive NRFD low.

4. Last listener to read the data byte allows NDAC to go high. (All other listeners have already released NDAC.)

5. After NDAC goes high the talker releases DAV and lets it go high.

6. After DAV goes high all listeners drive NDAC low.

7. After driving NDAC low, listeners allow NRFD to go high indicating they are ready for the next data byte. (Note the handshake lines are in the same states they were in during step 1.)

8. Talker starts handshake for second data byte. Handshake cycle repeats.

IEEE-488 HANDSHAKE
FIGURE C-3

IEEE-488 INTERFACE BUS SIGNAL LINES

DI01-DI08	Data lines
DAV	Data Valid: Handshake line used by talker.
NRFD	Not Ready For Data: Handshake line used by listener.
NDAC	Not Data Accepted: Handshake line used by listener.
ATN	Attention: Used by controller to send commands and initiate parallel poll. Devices must respond to ATN going low within 200 nSec.
IFC	Interface Clear: Unaddresses all devices on the bus.
SRQ	Service Request: Line used by devices as an interrupt request.
REN	Remote Enable: Allows a device to be remotely programmed by message bytes.
EOI	End or Identify: Used to indicate the last byte of a multi-byte message and (when used with ATN) to initiate a parallel poll.

FIGURE C-4

APPENDIX D
SYSTEM PATCH PANEL

The need for a convenient means of making connections to the A/D and D/A converters became apparent shortly after starting to work with the system. A patch panel with an array of BNC connectors was selected as the solution. In addition to connectors for the A/D and D/A converters (4 channels of each) provisions were made for connections to a function generator and 3 sets of 4-input multiplexers. (One set is dedicated to the HP3455 DVM and the other two sets are arranged as a MUX/DEMUX pair.) Along with the patch panel, circuitry was necessary to act as buffers and filters for the A/D and D/A converters.

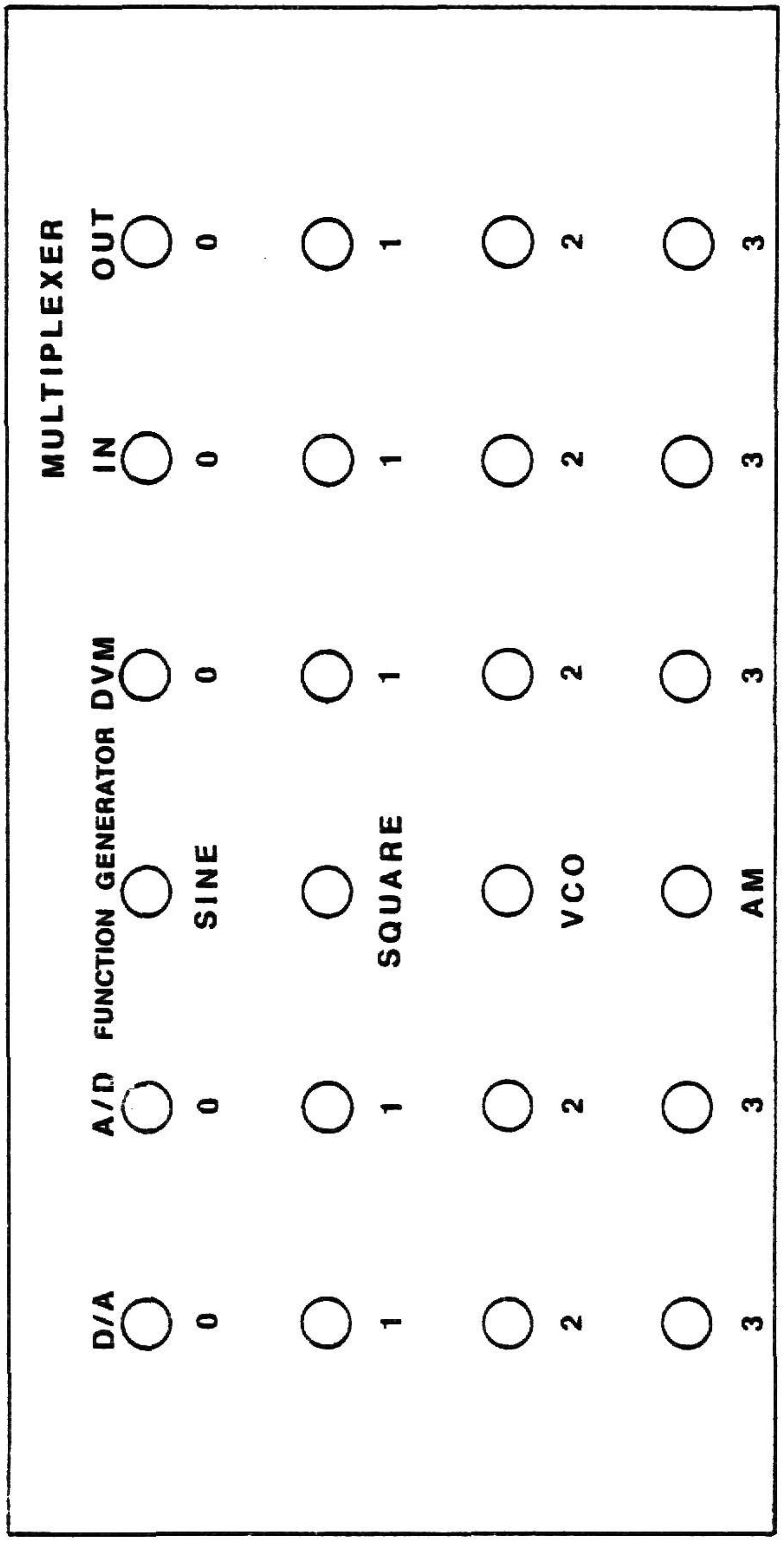
In the first attempts to use the converters, noise was a severe problem. To minimize the noise in the system that will be used in the laboratory, co-axial cables and active filters are used in the interface between the converters and the patch panel.

The A/D channels appeared to have a greater noise problem than the D/A converters did. A major source of the noise was the long length of 40-connector ribbon cable needed to bring the A/D board terminations to the back of the computer. To reduce this problem, co-axial ribbon cable was run to within about 6" of the A/D board termination, and a splice to conventional ribbon cable made at that point. (While the splice was being made, all 8 co-axial cables were spliced to the first 8 A/D channels even though only 4 would be immediately used.)

The A/D converter is single-ended with its ground tied to earth potential. To make the A/D converters more useful in a laboratory environment, a differential amplifier stage was added to each channel. The final part of the interface to the A/D converters is a 10 KHz., 2-pole low-pass filter. Since the absolute maximum sampling rate of the A/D converter is 24 KHz., little information is lost by bandlimiting the signal to 10 KHz., and high frequency noise components are significantly reduced.

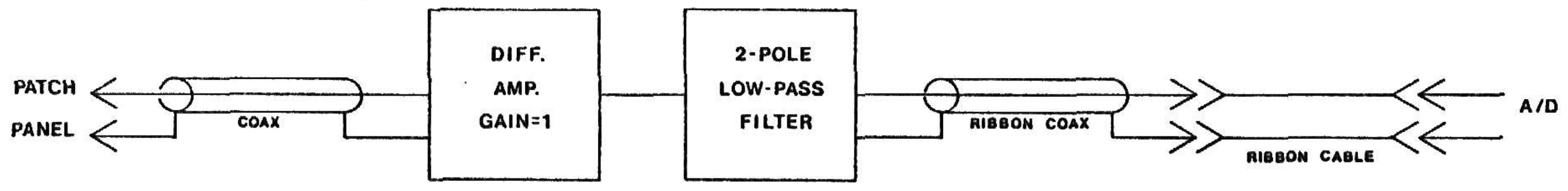
The D/A channels had a much smaller noise problem and as a result had fewer remedies applied to them. Standard ribbon cable was used from the D/A board termination to the rear of the computer and an interface to co-axial cable was made at that point. Like the A/D channels, a 10 KHz., 2-pole low-pass filter was inserted between the converter and the patch panel. In the future the pass-band of this filter might need to be increased since the risetime on a step output is fairly slow.

The following pages contain: a drawing of the patch panel, a block diagram of the cabling for the A/D and D/A converters, schematics and layouts for the buffers on the A/D and D/A converters.

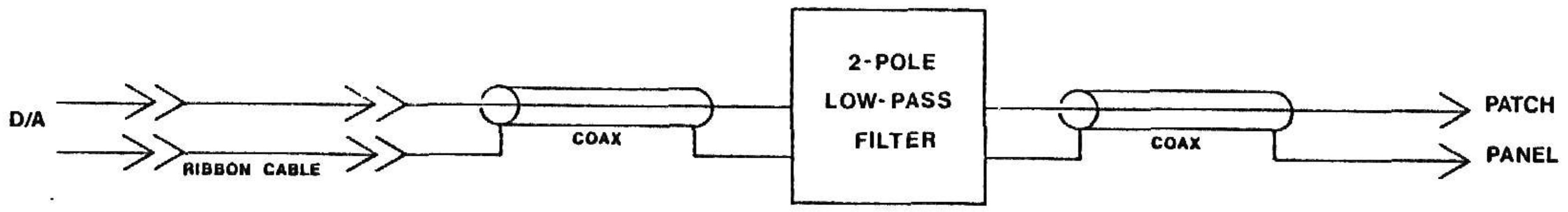


PATCH PANEL

FIGURE D-1



A/D CHANNEL INTERFACE



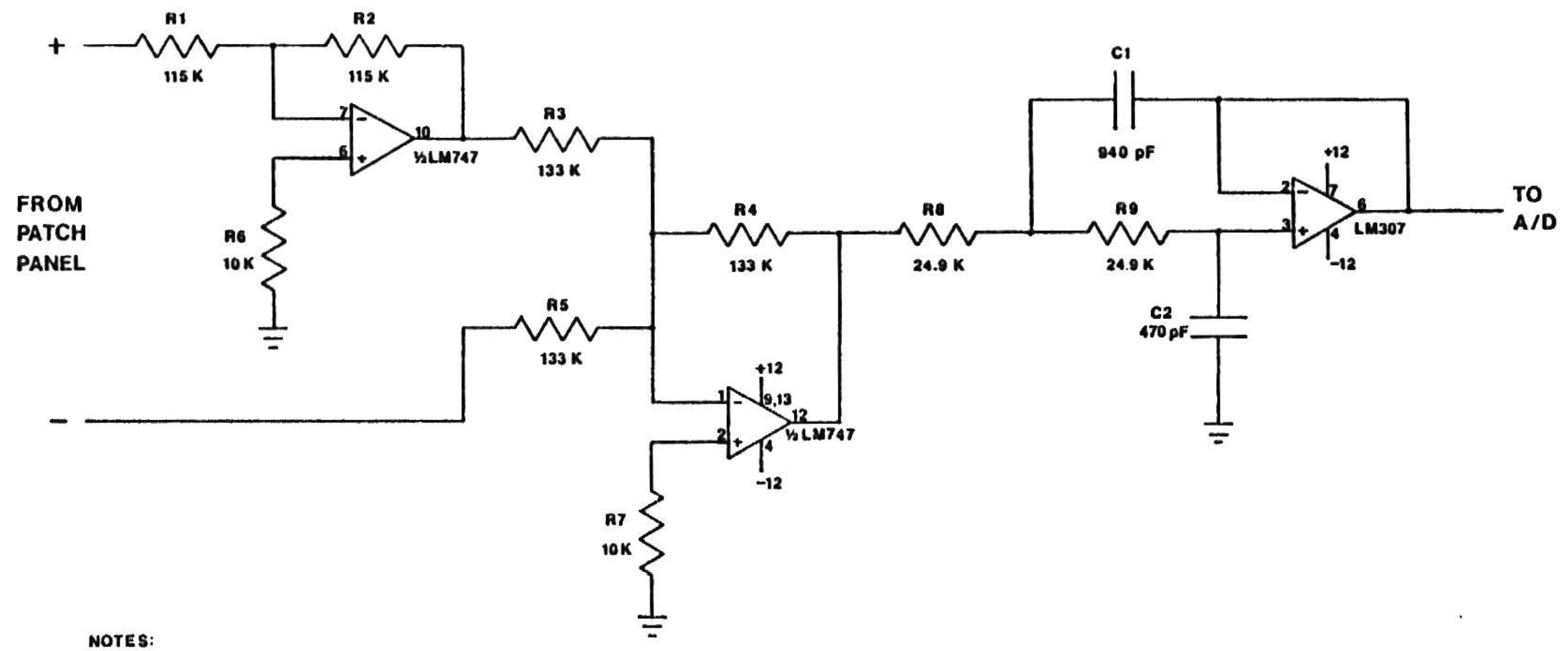
D/A CHANNEL INTERFACE

FIGURE D-2

A/D INTERFACE FEATURES

1. Differential input
2. Input impedance approximately 100 K Ohm.
3. Differential gain = 1
4. Common-mode gain = .002
5. CMRR = 54 dB
6. Low pass filter has 2-pole roll-off. f_o = 10 KHz.
7. Two channels on each printed circuit board.

TABLE D-1

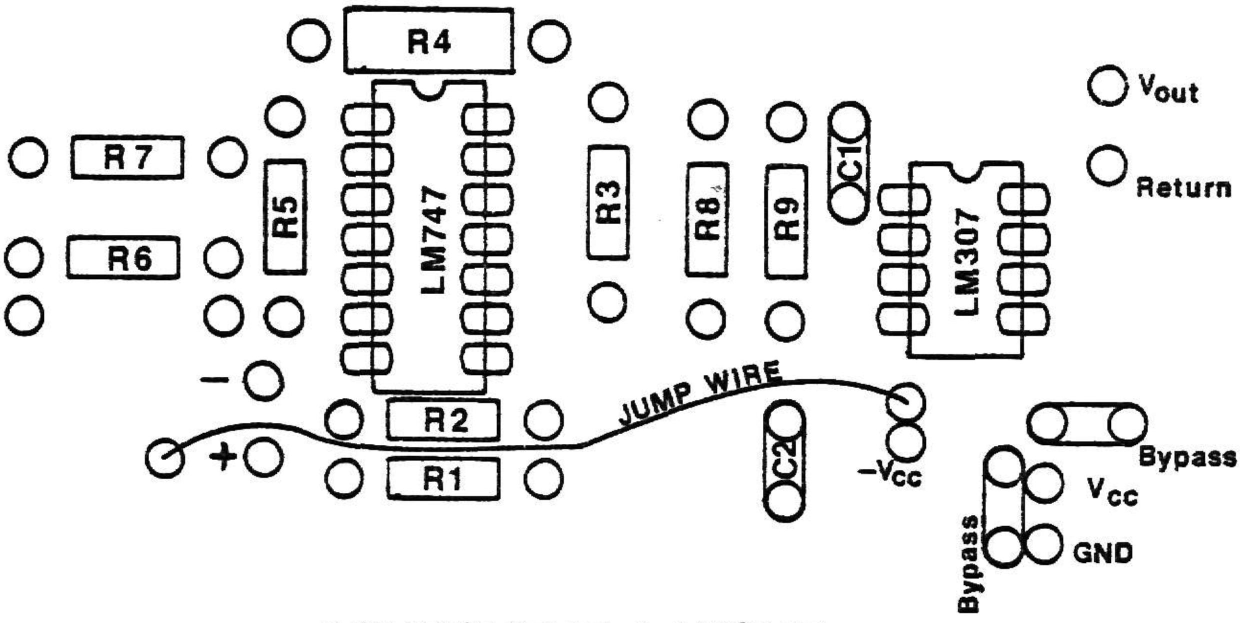


NOTES:

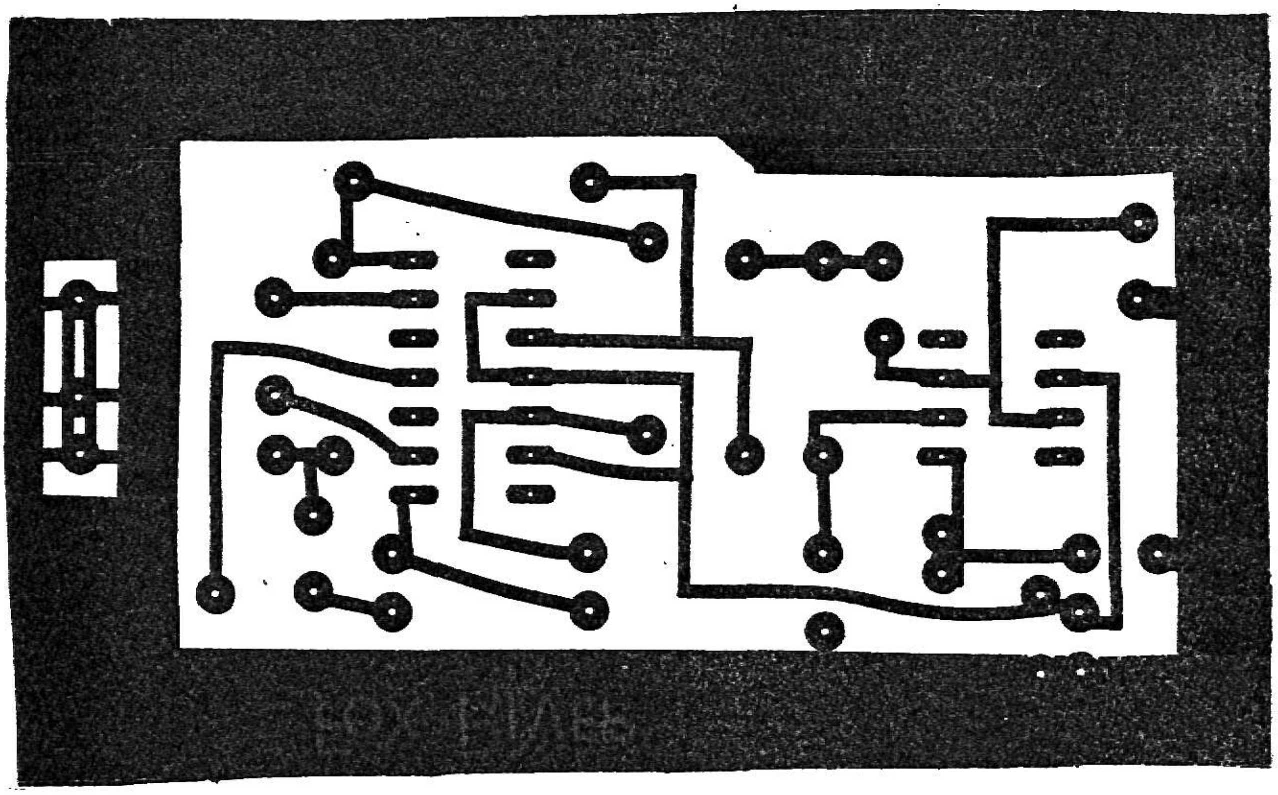
1. R1 and R2 must be closely matched. The resistors used were within .1% of each other.
2. R3, R4, and R5 must be closely matched. The resistors used were within .1% of each other.

DIFFERENTIAL AMPLIFIER AND LOW-PASS FILTER

FIGURE D-3



COMPONENT LAYOUT



FOIL PATTERN (SEEN THROUGH BOARD)

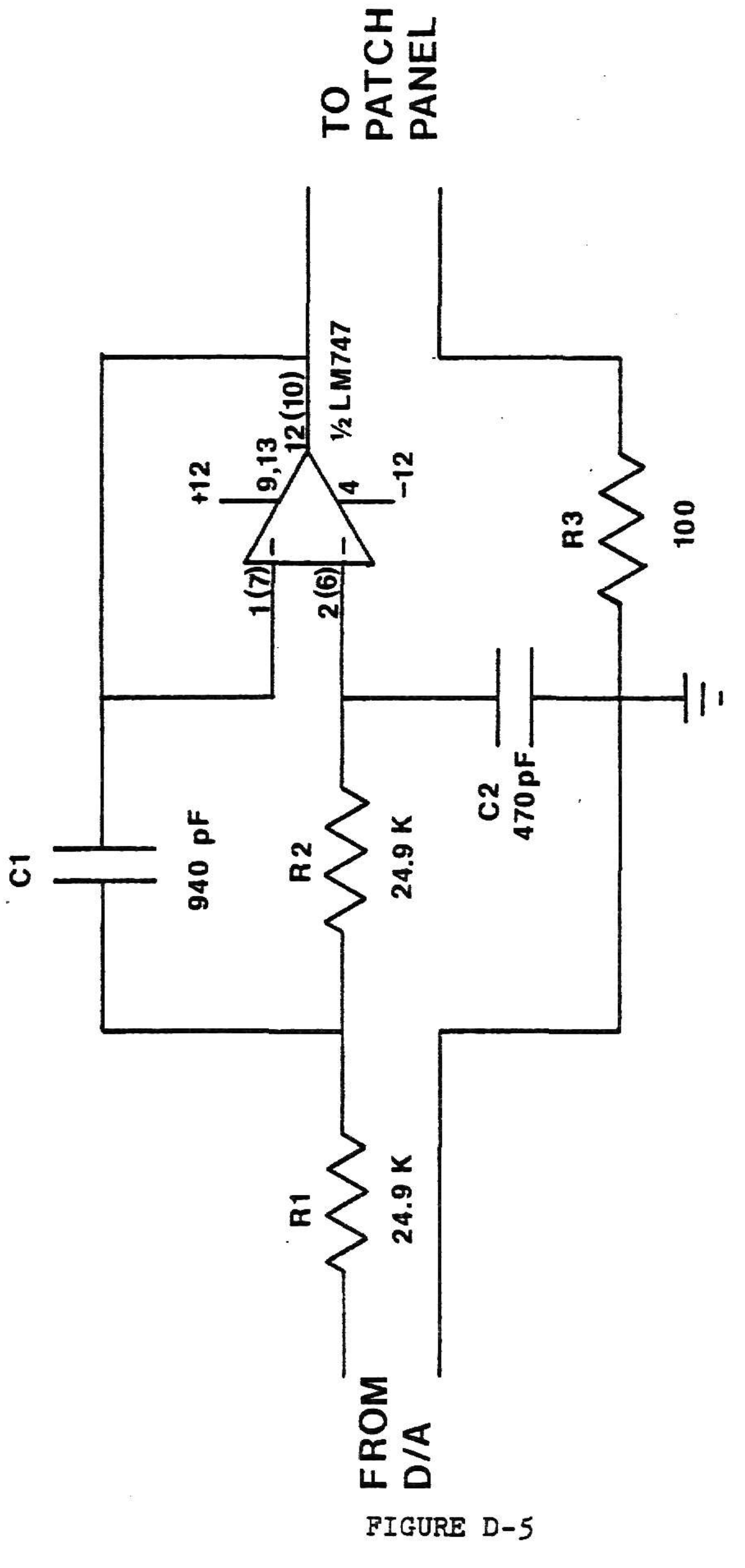
DIFF. AMP. AND LOW-PASS FILTER

FIGURE D-4

D/A INTERFACE FEATURES

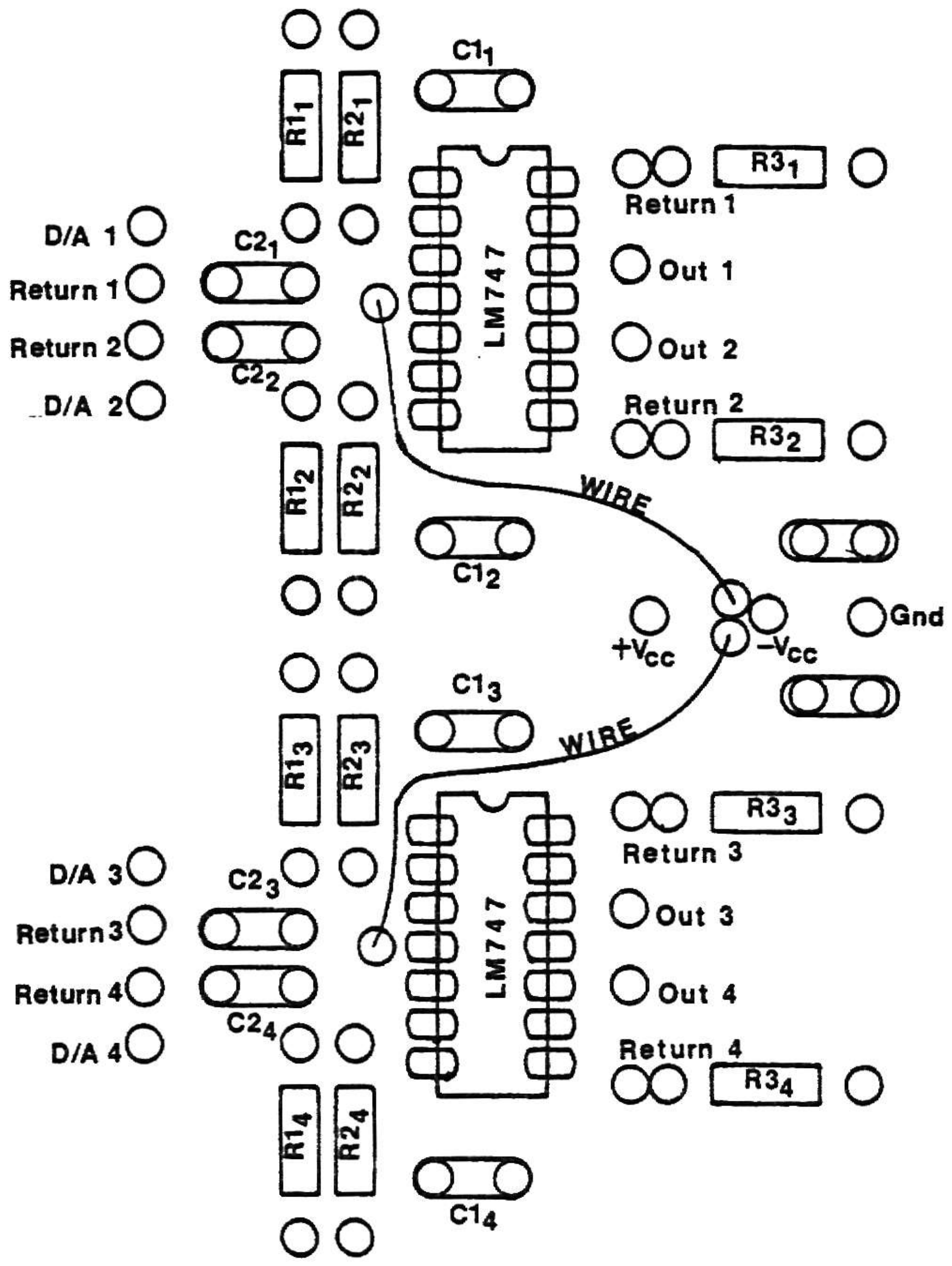
1. Output impedance approximately 200 Ohm.
2. Output ground is at earth potential. (grounded through the PDP-11's ground connection)
3. There is a 100 Ohm resistor in series in the output ground to protect the PDP-11.
4. 2-pole roll-off low-pass filter. f_o = 10 KHz.
5. Four channels on each printed circuit board.

TABLE D-2



LOW-PASS FILTER

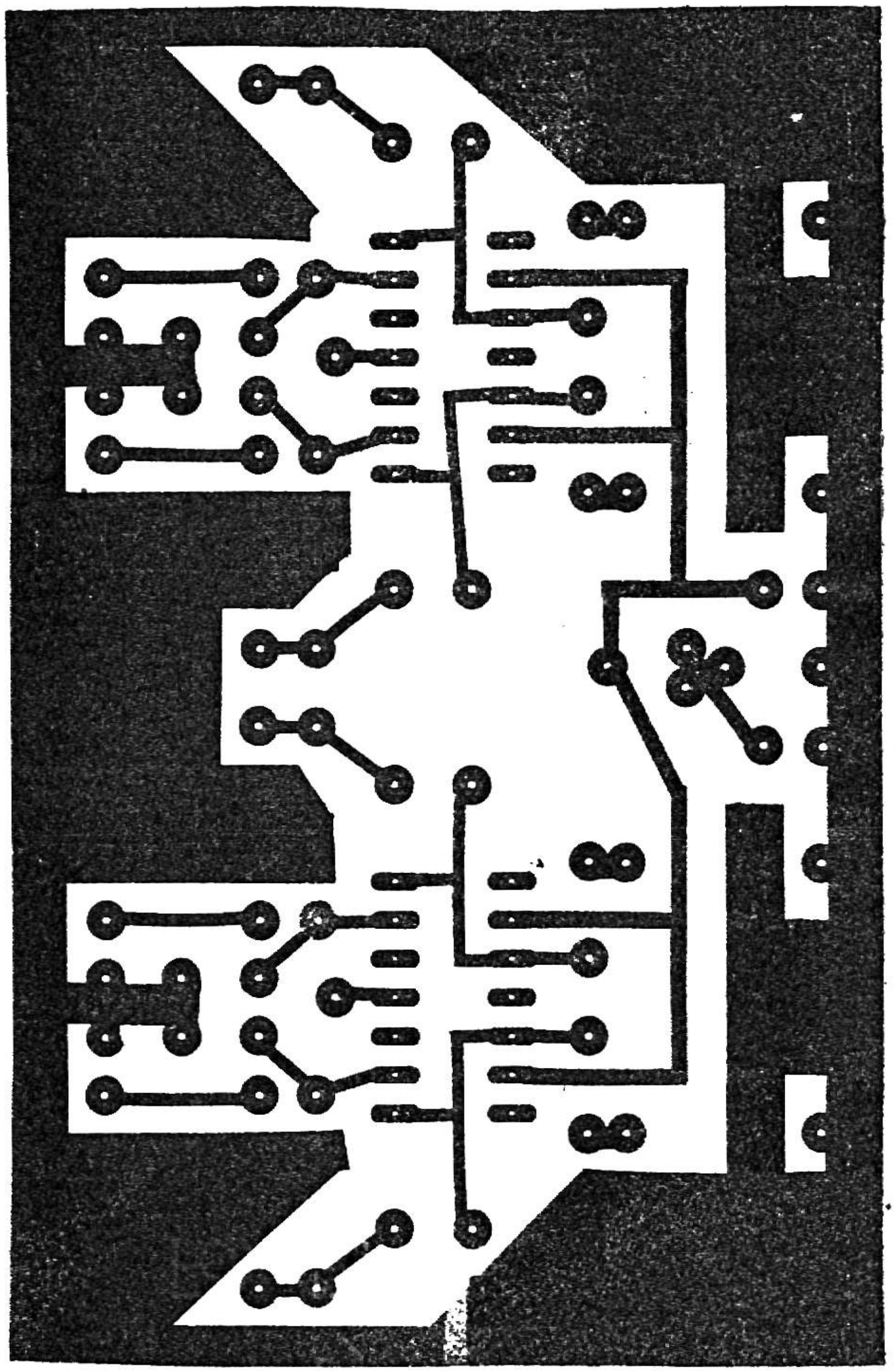
FIGURE D-5



COMPONENT LAYOUT

LOW-PASS FILTER BOARD

FIGURE D-6



FOIL PATTERN (SEEN THROUGH BOARD)
LOW-PASS FILTER BOARD

FIGURE D-7

APPENDIX E

MULTIPLEXER

The IEEE-488 compatible multiplexer is located inside the patch panel box. It requires power supply voltages of +5 V., -12 V., and Ground. It is connected to the three right-hand columns of BNC connectors on the patch panel. In addition to the patch panel connections, there is a connection to the IEEE-488 bus and a co-axial cable to the rear input of the HP3455 DVM. The IEEE-488 device number of the multiplexer is selected by the DIP switches on th 8748 "listener" card. Switches 3 (MSB) through 7 (LSB) are used to select the number. If the slide switch is pushed toward the switch number printed on the package a "0" is selected for that bit. If the switch is pushed away from the number a "1" is selected. The device number currently assigned to the multiplexer is "5". For this device number the MLA command is an ASCII '%'. The multiplexer channels are selected by sending one message byte to the multiplexer over the IEEE-488 bus. The message byte to be sent is given by the following formula.

```
IDVM="DVM" channel selected
IIN = "IN" channel selected
IOUT="out" channel selected

IBYTE=IDVM+4*IIN+16*IOUT  (IBYTE is message byte)
```

For example: The following code selects DVM1,IN2,OUT0

```
IDVM=1
IIN=2
IOUT=0
IBYTE=IDVM+4*IIN+16*IOUT
CALL SENDC('%')
CALL SENDM(IBYTE)
```

DO NOT apply voltages greater than ± 5 V. to any multiplexer input.

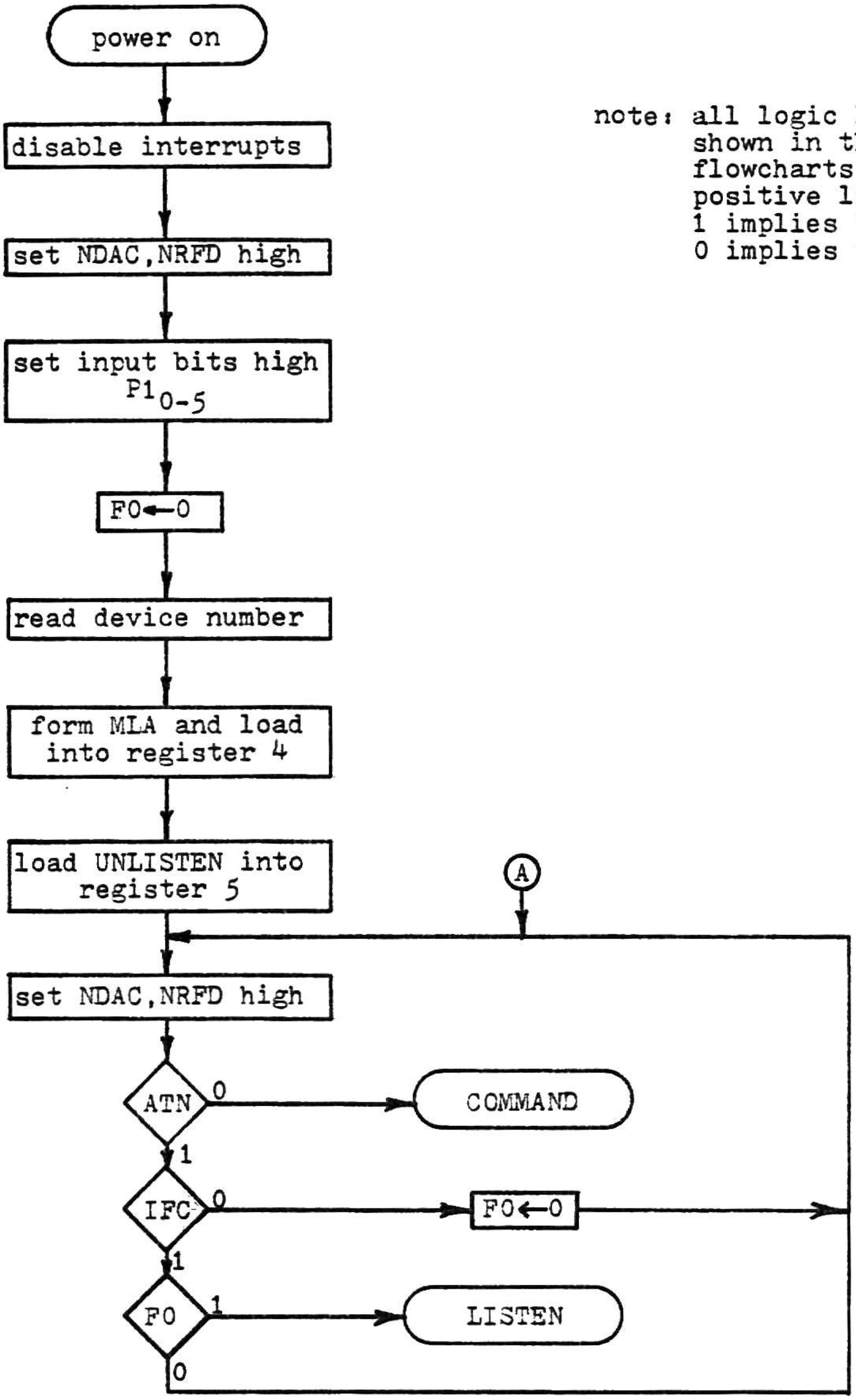
The description of the IEEE-488 compatible multiplexer is in two parts. The first deals with the "listener" card and the second with the "multiplexer" card.

LISTENER

The listener card of the IEEE-488 compatible multiplexer is a general purpose IEEE-488 listener. An INTEL 8748 single chip microprocessor was used to fulfill the protocol requirements of the bus. The message byte received by the 8748 is output on 8 data lines. (Note: if desired, a pulse can be obtained every time a new byte is received by "anding" T0, $P1_6$, and $\overline{P1_7}$ from the 8748.)

The IEEE-488 standard requires that all devices respond within 100 nSec. to ATN rising or falling. This requirement ensures that all devices that should be participating in a handshake are in fact participating. The 8748 cannot directly meet this requirement. The desired result is obtained by having each change of state on ATN trigger a TTL one-shot. This one-shot then drives NRFD and NDAC low, stopping any handshake for over 100 uSec. By the end of the 100 uSec. pulse, the 8748 has had enough time to configure itself to participate in the handshake. An MC3448 is used to provide the drive needed for the outputs to the bus (NRFD, NDAC).

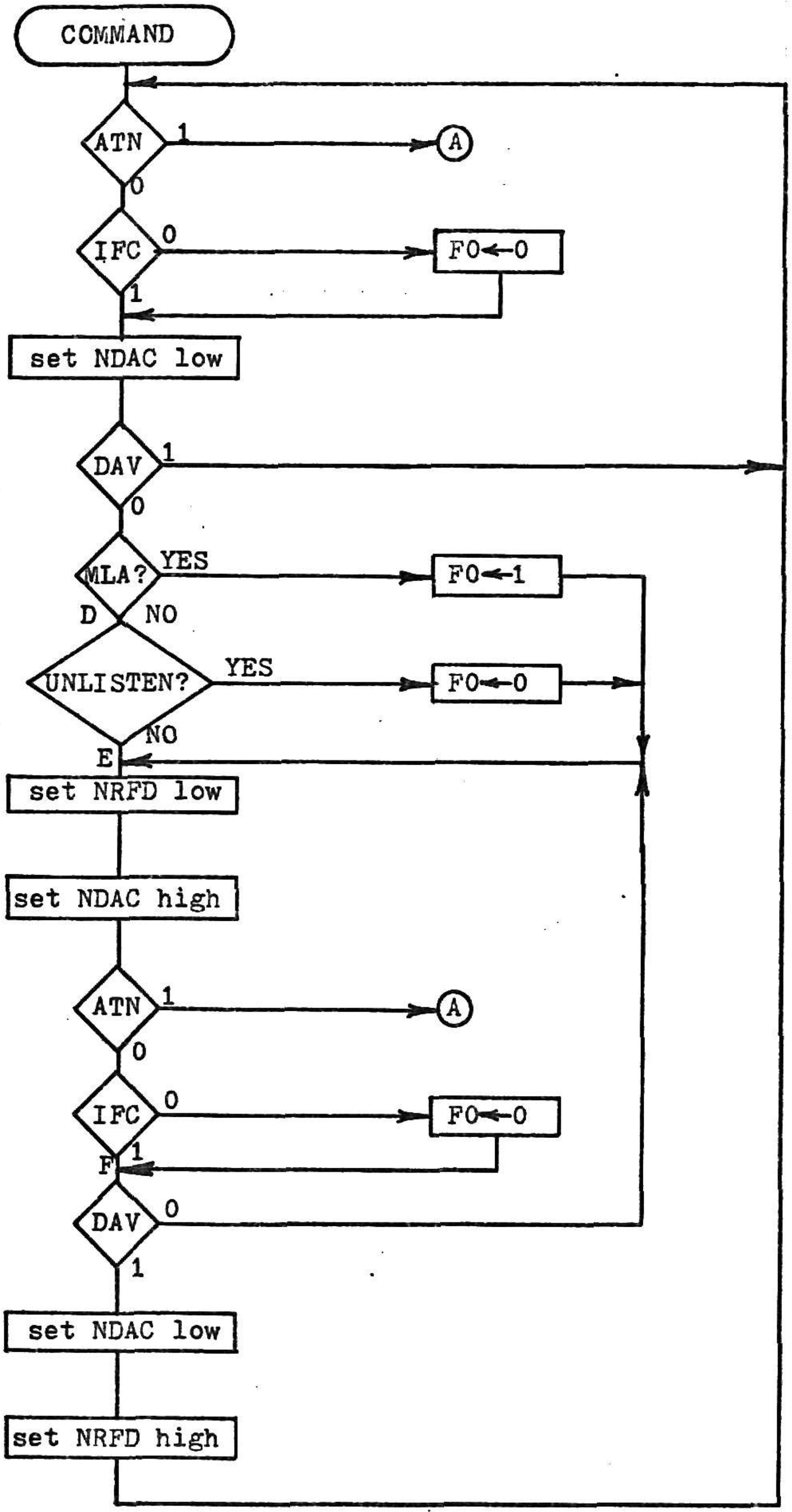
The following pages contain: a flowchart for the 8748 program, program variable description, edge-connector pin-out, a schematic, and circuit board layout.

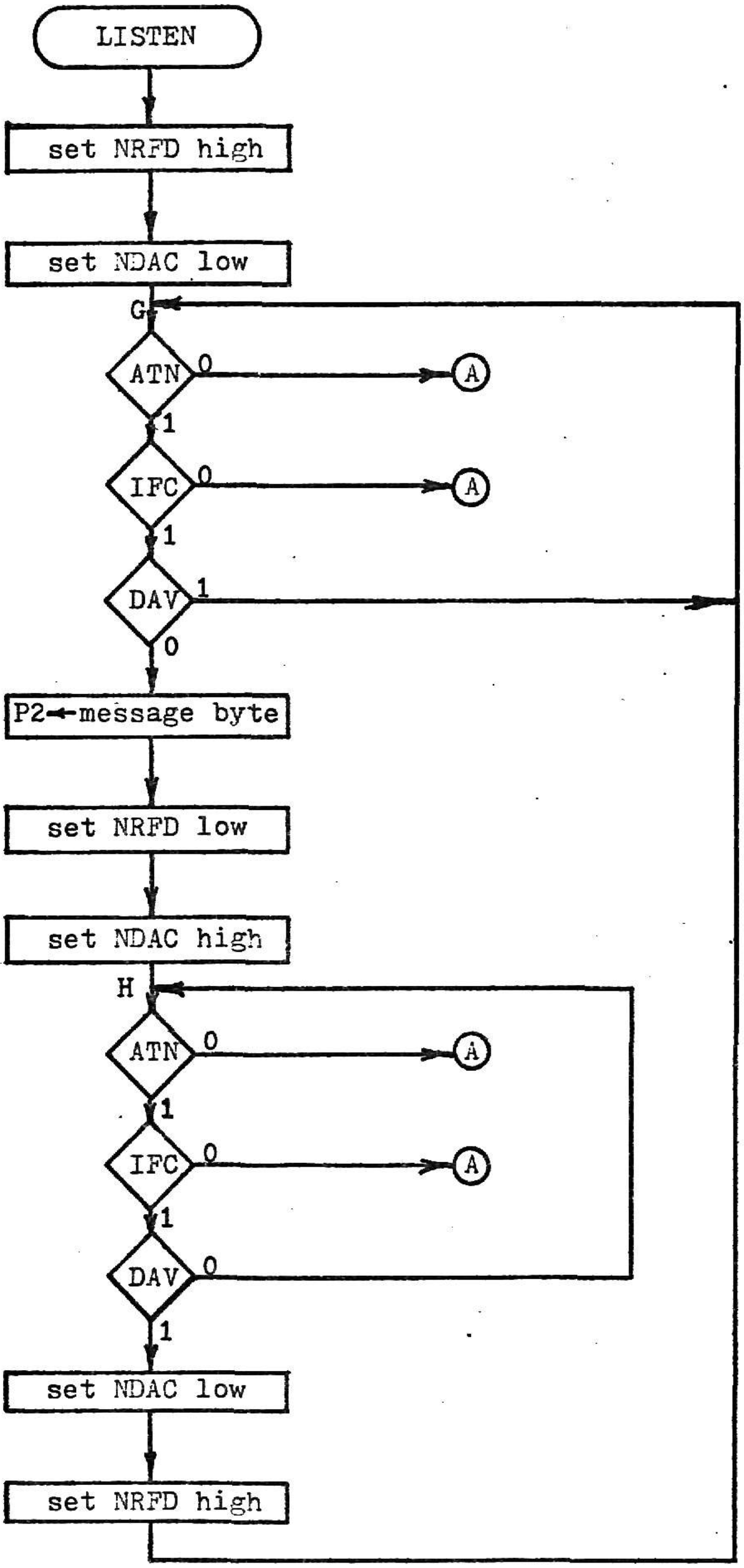


note: all logic levels shown in these flowcharts are positive logic. 1 implies "high" 0 implies "low"

FLOWCHART, IEEE-488 LISTENER

FIGURE E-1


COMMAND
ATN
1
A
0
IFC
0
F0←0
1
set NDAC low
DAV
1
0
MLA?
YES
F0←1
D
NO
UNLISTEN?
YES
F0←0
NO
E
set NRFD low
set NDAC high
ATN
1
A
0
IFC
0
F0←0
F
1
DAV
0
1
set NDAC low
set NRFD high


LISTEN
set NRFD high
set NDAC low
G
ATN
0
1
A
IFC
0
1
A
DAV
1
0
P2←message byte
set NRFD low
set NDAC high
H
ATN
0
1
A
IFC
0
1
A
DAV
0
1
set NDAC low
set NRFD high

8748 PROGRAM

```
00          DIS   TCNTI  DISABLE INTERRUPTS                    35
            DIS   I                                            15
            ORL   P1,#FF NRFD,NDAC SET HIGH.  OTHER BITS       89FF
                         SET AS INPUTS
            CLR   F0     CLEAR MLA FLAG                        85
            IN    A,P1   READ DEVICE NUMBER                    09
            ANL   A,#1F  MASK OFF NRFD,NDAC,IFC                531F
            ORL   A,#20  MAKE MLA FROM DEVICE NUMBER           4320
            MOV   R4,A   R4=MLA                                AC
            MOV   R5,#3F R5=UNLISTEN                           BD3F
            ANL   P2,#00 CLEAR OUTPUT PORT                     9A00
0F A:       ORL   P1,#FF RESET PORT 1 TO ALL 1's               89FF
            JNTO  CMMD   IF ATN IS LOW GO TO COMMAND MODE      261F
            IN    A,P1   LOOK AT IFC                           09
            ANL   A,#20                                        5320
            JNZ   B                                            961B
            CLR   F0     IF IFC IS LOW, CLEAR MLA FLAG         85
            JMP   A                                            040F
1B B:       JF0   LSTN   IF MLA FLAG IS SET, GO TO LISTEN      B64E
            JMP   A                                            040F
1F CMMD:    JTO   A      IF ATN IS HIGH GO TO WAIT MODE        B64E
            IN    A,P1   LOOK AT IFC                           09
            ANL   A,#20                                        5320
            JNZ   C                                            9627
            CLR   F0     IF IFC IS LOW, CLR MLA FLAG           85
27 C:       ANL   P1,#BF SET NDAC LOW                          993F
            JT1   CMMD   WAIT FOR DAV TO GO LOW                561F
            INS   A,BUS  READ COMMAND                          08
            CPL   A                                            37
            XRL   A,R4   COMPARE TO MLA                        DC
            JNZ   D                                            9634
            CLR   F0     IF MLA, SET MLA FLAG                  85
            CPL   F0                                           95
            JMP   E                                            043A
34 D:       INS   A,BUS  READ COMMAND AGAIN                    08
            CPL   A                                            37
            XRL   A,R5   COMPARE TO UNLISTEN                   DD
            JNZ   E                                            963A
            CLR   F0     IF UNLISTEN, CLEAR MLA FLAG           85
3A E:       ANL   P1,#7F SET NRFD LOW                          997F
            ORL   P1,#7F SET NDAC HIGH                         897F
            JTO   A      IF ATN IS HIGH, GO TO WAIT MODE       360F
            IN    A,P1   LOOK AT IFC                           09
            ANL   A,#20                                        5320
            JNZ   F                                            9646
            CLR   F0     IF IFC IS LOW, CLEAR MLA FLAG         85
46 F:       JNT1  E      WAIT FOR DAV TO GO HIGH               463A
            ANL   P1,#BF SET NDAC LOW                          993F
            ORL   P1,#BF SET NRFD HIGH                         893F
            JMP   CMMD   LOOK FOR NEXT COMMAND                 041F
```

TABLE E-1

```
4E LSTN: ANL  P1,#BF SET NDAC LOW                        99BF
         ORL  P1,#BF SET NRFD HIGH                       89BF
52 G:    JNTO A      IF ATN IS LOW, GO TO WAIT ROUTINE  260F
         IN   A,P1   LOOK AT IFC                         09
         ANL  A,#20                                      5320
         JZ   A      IF IFC IS LOW, GO TO WAIT ROUTINE   C60F
         JT1  G      WAIT FOR DAV TO GO LOW              5652
         INS  A,BUS  READ MESSAGE BYTE                   08
         CPL  A                                          37
         OUTL P2,A   OUTPUT MESSAGE BYTE                 3A
         ANL  P1,#7F SET NRFD LOW                        997F
         ORL  P1,#7F SET NDAC HIGH                       897F
62 H:    JNTO A      IF ATN IS LOW, GO TO WAIT ROUTINE  260F
         IN   A,P1   LOOK AT IFC                         09
         ANL  A,#20                                      5320
         JZ   A      IF IFC IS LOW, GO TO WAIT ROUTINE   C60F
         JNT1 H      WAIT FOR DAV TO GO HIGH             4662
         ANL  P1,#BF SET NDAC LOW                        99BF
         ORL  P1,#BF SET NRFD HIGH                       89BF
         JMP  G      LOOK FOR NEXT MESSAGE BYTE          0452
```

The following assignments were made for the 8748.

BUS: IEEE-488 data lines (negative logic)

$P1_7$: NRFD (output)

$P1_6$: NDAC (output)

$P1_5$: IFC (input)

$P1_{0-4}$: device number from DIP switches (input)

$P2_{0-7}$: message byte received (output)

T0: ATN (input)

T1: DAV (input)

F0 (internal): MLA flag

Register 4 (internal): bit pattern for MLA command

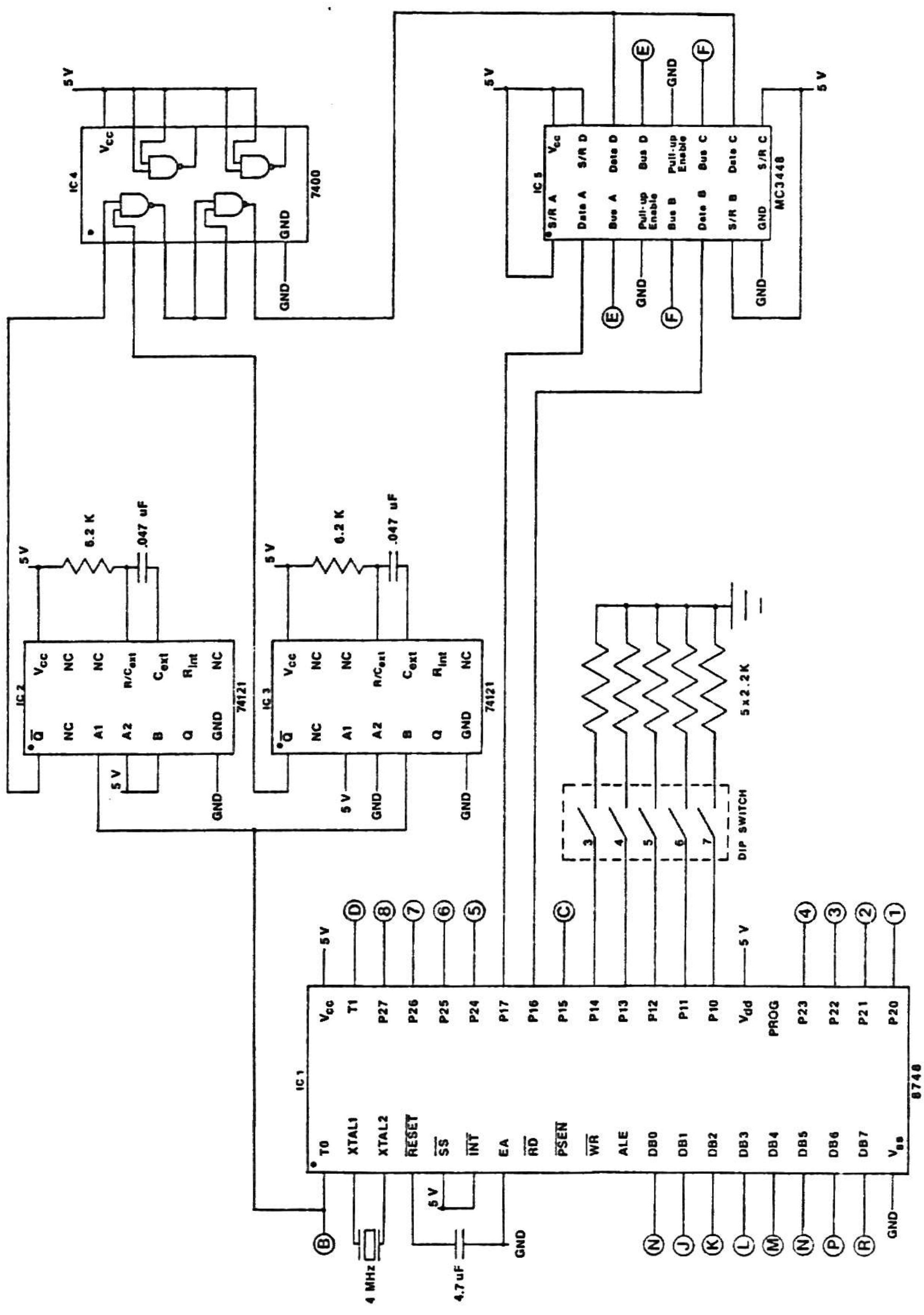
Register 5 (internal): bit pattern for UNLISTEN command

VARIABLES LIST
TABLE E-2

LISTENER EDGE-CONNECTOR ASSIGNMENTS

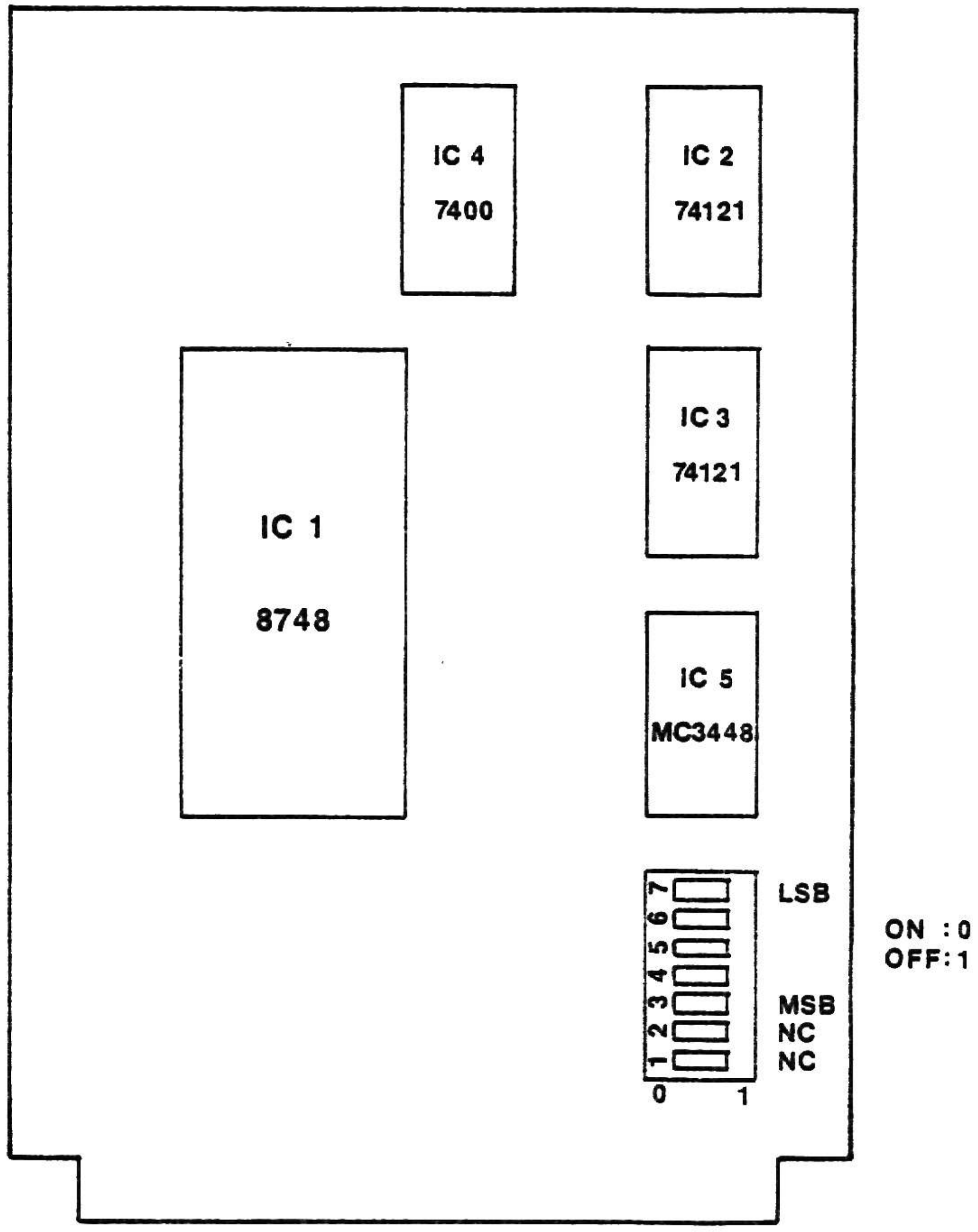
A	GND
B	ATN
C	IFC
D	DAV
E	NRFD
F	NDAC
H	DIO 1
J	DIO 2
K	DIO 3
L	DIO 4
M	DIO 5
N	DIO 6
P	DIO 7
R	DIO 8
S	
T	
U	
V	
W	
X	
Y	
Z	+5 V.
1	D0
2	D1
3	D2
4	D3
5	D4
6	D5
7	D6
8	D7
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	

TABLE E-3



8748 LISTENER

FIGURE E-2



8748 LISTENER LAYOUT

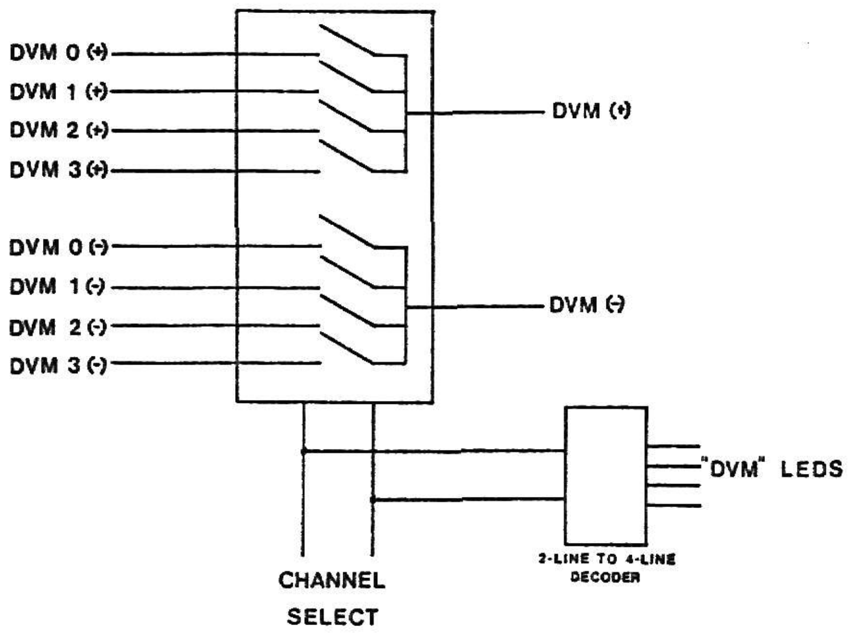
FIGURE E-3

MULTIPLEXER CARD

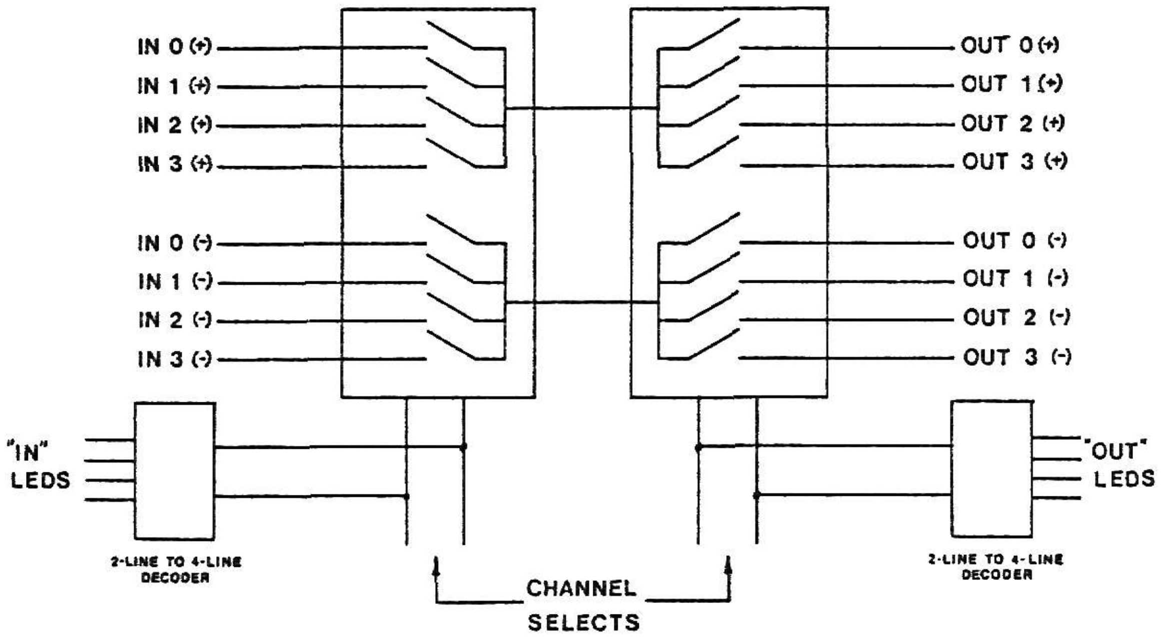
The multiplexer card of the IEEE-488 compatible multiplexer contains three, dual 4-channel multiplexers (dual since both signal and return are switched), and the logic needed to indicate which channel has been selected on each multiplexer. The multiplexer used was the CD4052 Dual 4-channel Analog Multiplexer/Demultiplexer. One multiplexer is used to select one of 4 channels on the patch panel as the input to the HP3455 DVM. The other two multiplexers are arranged as a MUX/DEMUX pair. In addition to the multiplexers, three TTL 2-line to 4-line decoders are used to light an LED next to each channel selected.

The Power supply voltages used for the CD4052 multiplexers are $\pm$ 5 V. Since input voltages that exceed the CD4052's supply will destroy the device, it is important that signals applied to the multiplexers be less than $\pm$ 5 V. The "on resistance" of the CD4052 is approximately 200 Ohm. The channel select lines are brought in on a DIP connector that plugs into a socket on the multiplexer board.

The following pages contain: an edge-connector pin-out, a schematic, a wiring diagram, and circuit board layout.



DVM MULTIPLEXER



MUX/DEMUX PAIR

FIGURE E-4

EDGE-CONNECTOR

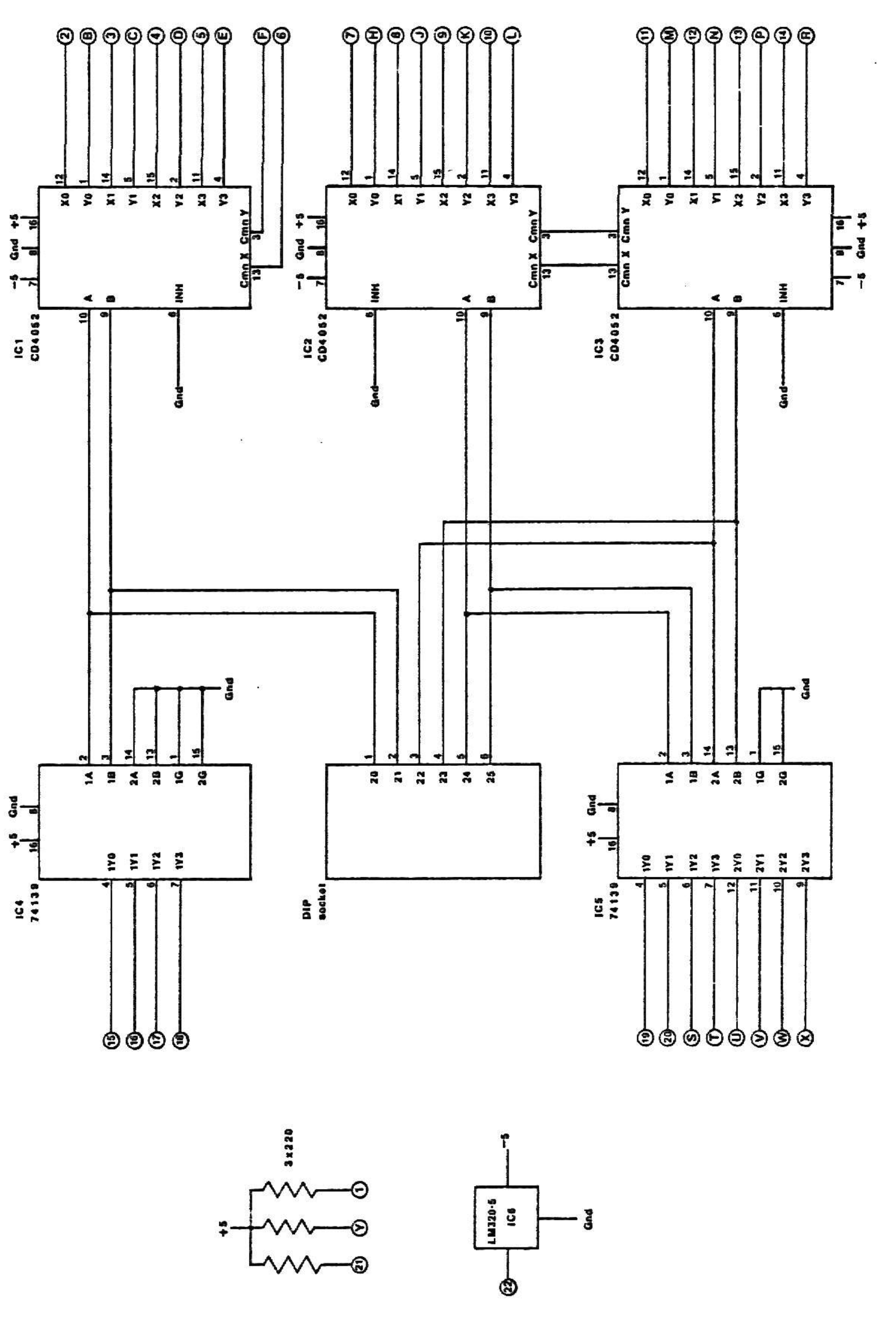
1	DVM LED SOURCE	A	GND.
2	DVM0	B	DVM0 RET
3	DVM1	C	DVM1 RET
4	DVM2	D	DVM2 RET
5	DVM3	E	DVM3 RET
6	DVM(+)	F	DVM(-)
7	OUT0	H	OUT0 RET
8	OUT1	J	OUT1 RET
9	OUT2	K	OUT2 RET
10	OUT3	L	OUT3 RET
11	IN0	M	IN0 RET
12	IN1	N	IN1 RET
13	IN2	P	IN2 RET
14	IN3	R	IN3 RET
15	LED DVM0	S	LED OUT2
16	LED DVM1	T	LED OUT3
17	LED DVM2	U	LED IN0
18	LED DVM3	V	LED IN1
19	LED OUT0	W	LED IN2
20	LED OUT1	X	LED IN3
21	OUT LED SOURCE	Y	IN LED SOURCE
22	-12 V.	Z	+5 V.

DIP CONNECTOR

1	D0
2	D1
3	D2
4	D3
5	D4
6	D5

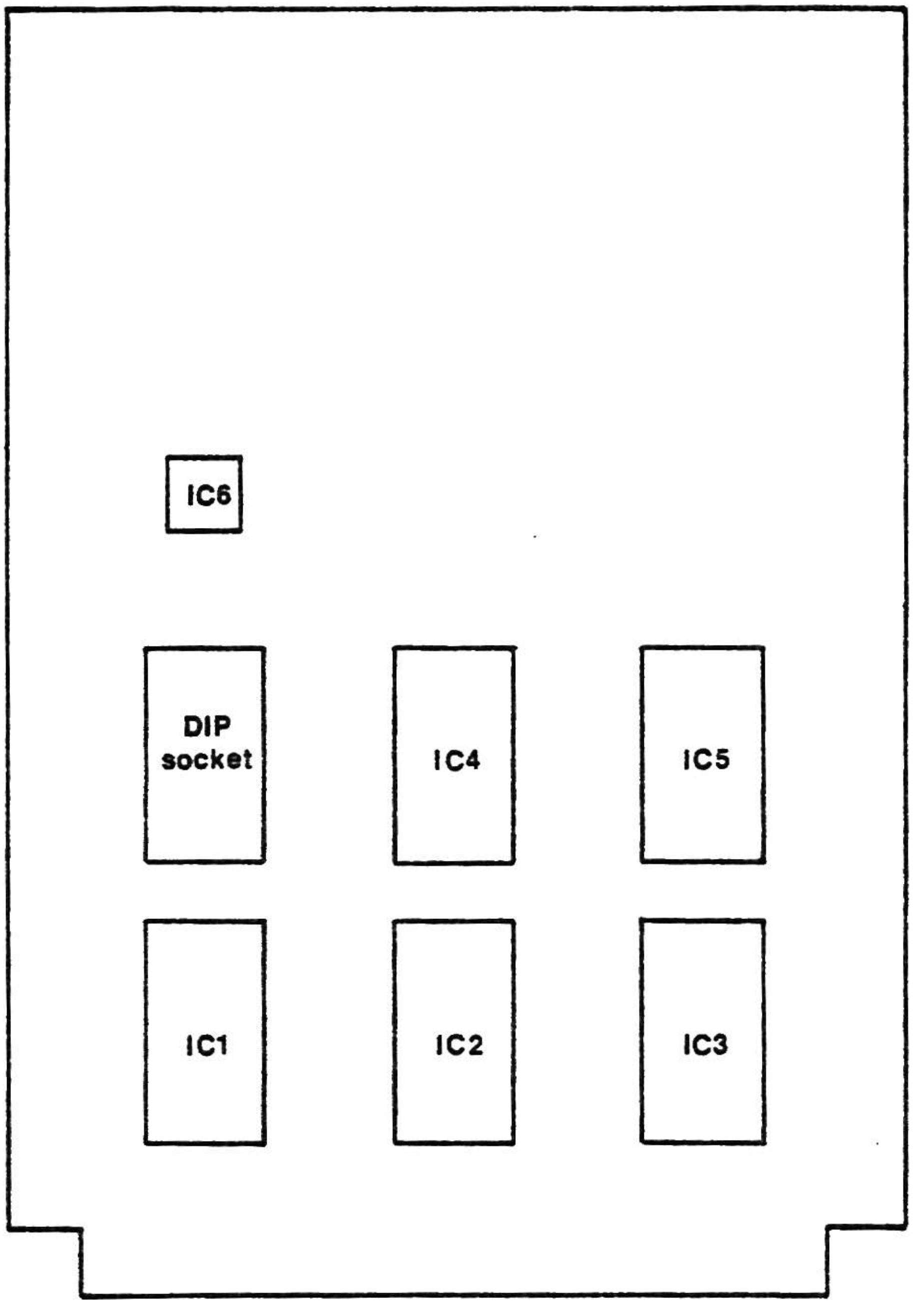
(From message byte lines of listener.)

TABLE E-4



MULTIPLEXER

FIGURE E-5



MULTIPLEXER LAYOUT

FIGURE E-6

APPENDIX F

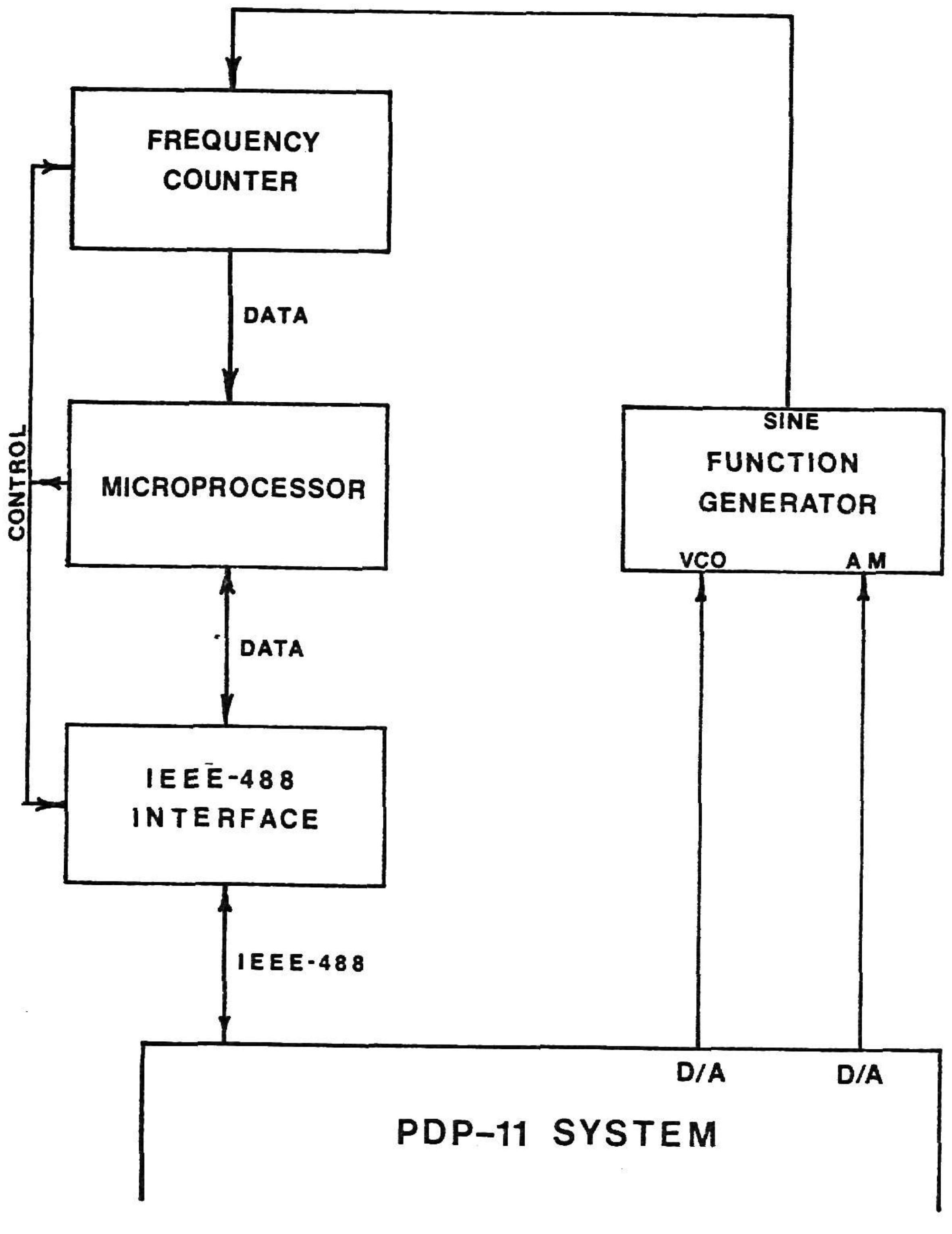
FUNCTION GENERATOR/FREQUENCY COUNTER

The IEEE-488 compatible frequency counter and function generator are located inside the patch panel case. They require power supply voltages of +5 V., ±12 V., and ground. The function generator is connected to the column of BNC connectors on the patch panel labeled "FUNCTION GENERATOR" (third column from the left). The frequency counter is connected to the IEEE-488 bus, and to the BNC connector on the right side of the case. The IEEE-488 device number of the frequency counter is "1" which corresponds to an MTA command of ASCII 'A'.

The frequency counter and function generator are essentially separate devices. The function generator supplies a sine wave output (4 Volt maximum amplitude) to the BNC connector labeled "SINE". The frequency of this signal is determined by the voltage input to the "VCO" connector of the patch panel, and the position of the switch on the right side of the patch panel case. The "up" position selects the high range (5 KHz. - 700 KHz.), while the "down" position selects the low range (30 Hz. - 5 KHz.). The range of voltages allowed on the "VCO" input is -5.12 V. to +5.12 V. (the range of a D/A converter channel). The frequency is a linear function of the "VCO" input, with -5.12 V. corresponding to the high end of the frequency range and +5.12 V. resulting in the lowest value. The amplitude of the sine wave is controllable through the "AM" input. It too is a linear function of the voltage at the input. Zero Volts input (or no connection) results in the maximum amplitude.

The function generator frequency cannot be set accurately from the "VCO" input. However, the frequency being produced may be accurately measured with the IEEE-488 compatible frequency counter. When it receives its MTA, the counter measures the frequency of the signal on its input. When the reading is complete (after about one second), the device generates a Service Request on the interface bus. After a Serial Poll is completed, it sends six bytes on the bus, each containing a BCD digit in the lower four bits. The digits are sent in descending order, most significant digit first.

To operate the devices, first connect one of the D/A outputs to the "VCO" input and another to the "AM" input (if amplitude control is desired). Connect the sine wave output on the patch panel to the frequency counter input on the right side of the case. Assign appropriate voltages to the D/A converters to select the desired frequency and amplitude. Be certain the range switch is in the correct position. To read the frequency being produced, address device "1" to talk. When a Service Request occurs, the reading is complete. Conduct a Serial Poll of device "1", then receive the six bytes of the frequency value. These bytes can be manipulated to yield a single number representing the frequency.



FUNCTION GENERATOR/FREQUENCY COUNTER

FIGURE F-1

APPENDIX G

USER'S GUIDE

This appendix is the user's guide used by the students in the instrumentation laboratory.

USER'S GUIDE

INSTRUMENTATION LAB

PDP11-03

TABLE OF CONTENTS

* Read First-- This section clarifies some points not fully explained in the manual.

GLOSSARY

< >	angle brackets indicate a single character that needs more than one letter to describe it.

<CR> means the carriage return key

<ESC> means the escape key

<control C> means to hold down the control key while typing a "C"

<TAB> means the tab key

UPPER CASE LETTERS: in a typing instruction mean letters are to be typed exactly as shown.

lower case letters: in a typing instruction mean the position is to be filled with characters selected by the user.

EX: If a program called TEST1 was being entered and the user's guide listed the following as a typing instruction:

```
      DX1:filename.FOR
```

the following should be typed:

```
      DX1:TEST1.FOR
```

OPERATING PROCEDURE

All programs to be written for the instrumentation lab PDP11-03 should use FORTRAN as the source language. This guide will assume a working knowledge of FORTRAN. If this is not the case, you might want to refresh yourself with a FORTRAN manual. The lab has a reference manual on DEC's (Digital Equipment Corporation) implementation of FORTRAN entitled PDP-11 FORTRAN LANGUAGE REFERENCE MANUAL with the manuals for the computer (none of which should be taken out of the lab!) and the library has many other FORTRAN manuals available.

The first step in using the lab's computer is to write a program. One feature that is different from FORTRAN implementations you might have used before are the device numbers for input and output. Since the Decwriter is both the input and output device, input/output statements will take the form:

```
WRITE(5,format number)variable list
     -or-
READ(5,format number)variable list
```

The example program below asks for a single digit integer, calculates the square of the number and prints the integer and its square.

```
C  PROGRAM TO CALCULATE THE SQUARE OF A ONE DIGIT INTEGER
      WRITE(5,100)               !EXPLAIN INPUT NEEDED
100   FORMAT(' ENTER A SINGLE DIGIT INTEGER')
      READ(5,101)K               !ENTER NUMBER
101   FORMAT(I1)
      J=K**2                     !CALCULATE THE SQUARE
      WRITE(5,102)K,J            !OUTPUT RESULTS
102   FORMAT(1X,I1,' SQUARED EQUALS ',I2)
      STOP
      END
```

Notice the following points about the previous program.

1) Comments can be made with a line started with a "C" (see the first line of the program) or an "!" following a FORTRAN statement. (lines 2,4,6, and 7)

2) Each output line requires a carriage control (for instance the first blank after the apostrophe in format 100 and the 1X in format 102). Input formats do not need carriage controls.

Now that the program is written it needs to be transferred to floppy disk. First, go through the turn-on procedure on page 20. You should now be in monitor and a '.' should be the last thing the computer typed out. If so, read the section on the system editor (page 6). If not, try going through the turn-on procedure again. After reading the section on the system editor enter the program onto your group's floppy disk.

To run this program, first make sure you are in monitor (indicated by the prompting character'.'). If not, you are probably still in editor and need to type:

```
EX<ESC><ESC>
```

To get in monitor if you aren't in editor, type:

```
<control C> <control C>
```

Now that you are in monitor, type the following to compile your program. The computer responses are shown underlined.

```
. R FORTRAN<CR>
* DX1:filename,TT:/L:1/W=DX1:filename <CR>
     The computer will type your program and
     any warnings or errors.*
* <control C>
. R LINK<CR>
* DX1:filename=DX1:filename,SY:LABLIB/F<CR>
* <control C>
```

* Explanation of errors and warnings can be found in DEC's RT-11/RSTS/E FORTRAN IV User's Guide Appendix C.

To run this program (or any program that you haven't modified since you compiled it last) type:

```
        RUN DX1:filename<CR>
        -Your program is now running-
```

If you want to abort program execution you can return to monitor by typing:

```
        <control C><control C>
```

When you are finished using the computer go through the turn-off procedure on page 21.

SYSTEM EDITOR

The FORTRAN programs are stored on your group's floppy disk and arranged in units called "files". Each file must have a unique (to that disk) filename of up to six characters long. Make sure you remember the filenames written on your group's disk since you can't get a listing of the disk's contents. (The lab instructor should be able to.) The editor keeps the file it is operating on in memory and keeps track of which lines are being operated on by moving a pointer to indicate the section being manipulated. (You never "see" the pointer, it is just a variable in the editor program.) The instructions explained in this guide will enable you to do any editing you will need to do. If you are interested in a complete description of the editor commands, see the lab instructor.

When in editor, the prompting character from the computer is an "*" (as opposed to a "." when in monitor). Since ⟨CR⟩ is a character that the editor must operate on, it is not used to cause the computer to execute the commands given. Instead, the "escape" key is used. Typing ⟨ESC⟩ will separate commands, and typing ⟨ESC⟩⟨ESC⟩ will cause the whole string of commands to be executed. The commands that follow this description will allow you to perform any necessary modification to your file. In the explanation of the commands "n" represents a number selected by the user.

To run editor you must first be in monitor (indicated by a "." prompting character). If you are in monitor type:

```
        R EDIT<CR>
```

If you are not in monitor type:

```
         <control C><control C>
        .R EDIT<CR>
```

You should now be in editor and the prompting character should be an "*". To create a new file type:

```
        EWDX1:filename.FOR<ESC><ESC>
```

To modify an existing file type:

```
        EBDX1:filename.FOR<ESC>R<ESC><ESC>
```

After you are finished creating and/or modifying a file, type:

```
        EX<ESC><ESC>
```

This will write the file onto your disk and return you to monitor. Do not return to monitor with <control C>. If you do, nothing you did in editor will be written onto disk and ALL modifications made will be lost.

1) FORTRAN statements will be correctly aligned for you by the editor if you type <TAB> before the start of the statement part of a line.

```
EX:     <TAB>GO TO 100
        100<TAB>WRITE(5,200)A,B
        <TAB>*C,D             (this line is interpreted
                               as a continuation line)
```

2) Try to "recycle" filenames by destroying an old file with "/K" and inserting your new program in the resulting blank file. Your disk can only hold a finite number of files. If your latest program would exceed that number you won't be able to write it out to your disk when you try and you will have lost all the time spent to create it in editor.

EDITOR COMMANDS

nA	advances the pointer to the start of the nth line from the current line (n can be positive or negative).
V	verifies the current position of the pointer by typing the line that the pointer is on.
Gchar	finds the first occurrence of the character string "char" after the current pointer location and moves the pointer to point immediately after the last character in the string.*
=Cchar	replaces the character string just found (using the "G" command) with character string "char" i.e. GWRTE<ESC>=CWRITE<ESC><ESC> will replace the first occurrence of WRTE with WRITE.
nL	lists n lines counting from current pointer location. /L will list all lines from the current pointer location to the end of the file.
nK	deletes ("kills") n lines counting from the current pointer location. /K deletes all lines from the current pointer location to the end of the file.
B	moves the pointer to the beginning of the file.
Ichar	inserts the character string "char" immediately before the current pointer location. The character string can be many lines long and is delimited by the <ESC> that separates this command from the next. This command is used to enter a program into editor the first time.
*NOTE:	If the editor doesn't find the character string "char" it will type ?*SRCH FAIL*? and the pointer will be pointing at the end of your file. To use the "G" command again it will be necessary to move the pointer in front of the string you are looking for. The "B" command will move the pointer to the top of your file.

SOFTWARE DESCRIPTION

A number of subroutines have been written to enable you to use the instrumentation peripherals of the PDP11-03. The next two pages give a brief description of these subroutines. A more detailed description can be found in the computer printout notebook labeled "LABLIB SUBROUTINE DESCRIPTIONS". The following notes should help avoid some problems in using the subroutines.

1) To remotely set the functions on the HP3455 pass '6' (with apostrophes) for IMLA in subroutine SNDMSG.

   EX: SNDMSG(IARRAY,N,'6')

2) The D/A and A/D converters use the following scale:

```
  -5.12 V = 0
   0.00 V = 2048          each step = 2.5 mV
   5.12 V = 4095
```

   These conversions can be done with DIGVLT and VLTDIG

3) Maximum sampling rate using RDAD is 4KHz.
   Maximum sampling rate using RDBLK is 12.5KHz.

INSTRUMENTATION LAB SUBROUTINES

The following is a collection of abbreviated descriptions of the subroutines in LABLIB. Complete descriptions can be found in computer output labeled "LABLIB SUBROUTINE DESCRIPTIONS".

```
DAC      Perform a D/A conversion
         CALL DAC(I,ICH)
              I           INTEGER        number to be converted
              ICH         INTEGER        D/A channel to be used


DIGVLT   Convert an offset binary number to equivalent voltage
         CALL DIGVLT(I,V)
              I           INTEGER        offset binary number
              V           REAL           equivalent voltage


DVMCVT   Convert output from HP3455 to floating point number
         CALL DVMCVT(IARRAY,VALUE)
              IARRAY(15)  INTEGER        output from HP3455
              VALUE       REAL           floating point number


IFC      Put IEEE-488 bus into a known state
         CALL IFC


POLL     Returns status from current talker on IEEE-488 bus
         CALL POLL(IBYTE)
              IBYTE       INTEGER        status of current talker


RCV      Reads a byte from the IEEE-488 bus
         CALL RCV(IDATA)
              IDATA       INTEGER        byte read from bus


RDAD     Waits for current A/D conversion to finish and reads value
         CALL RDAD(I,IERR)
              I           INTEGER        value of converted voltage
              IERR        INTEGER        error code


RDBLK    Fills an array with A/D conversions
         CALL RDBLK(IARRAY,N,IERR)
              IARRAY(N)   INTEGER        array to be filled
              N           INTEGER        number of readings
              IERR        INTEGER        error code
```

```
RDVM    Triggers a reading on the HP3455 and returns a reading
        CALL RDVM(VALUE,IERR)
             VALUE       REAL           reading returned
             IERR        INTEGER        error code


SENDC   Sends a command byte on the IEEE-488 bus
        CALL SENDC(IDATA)
             IDATA       INTEGER        command byte to be sent


SENDM   Sends a message byte on the IEEE-488 bus
        CALL SENDM(IDATA)
        IDATA            INTEGER        message byte to be sent


SETAD   Set A/D converter to selected channel and trigger option
        CALL SETAD(ICH,ICLK)
             ICH         INTEGER        A/D channel selected
             ICLK        INTEGER        trigger option


SETCLK  Set period of real-time clock
             CALL SETCLK(T)
             T           REAL           period in seconds


SNDMSG  Send message to a device on IEEE-488 bus
        CALL SNDMSG(IARRAY,N,IMLA)
             IARRAY(N)   INTEGER        message
             N           INTEGER        number of bytes in message
             IMLA        INTEGER        device to receive message


SYNC    Wait until next ouput of real-time clock to return
        CALL SYNC(IERR)
             IERR        INTEGER        error code


TRGAD   Triggers the start of an A/D conversion
        CALL TRGAD(IERR)
             IERR        INTEGER        error code


VLTDIG  Convert a voltage value to equivalent offset binary
        CALL VLTDIG(I,V)
             I           INTEGER        offset binary representation
             V           REAL           voltage value
```

"PLOT" DESCRIPTION

The plot program takes a data file produced in another program, and produces a rough, computer printout plot. To create a data file that is compatible with the plot program the following must be added to the program that gathers the data. Variable names can be changed to avoid conflicts with existing variable names in the program but the data types must stay the same (i.e. integer or real).

The following variables are needed to specify the plot:

LOG	Selects logarithmic or linear scale for the x-axis. 1=log, 0=linear. The y-axis is always linear.
BOTTOM	Lower bound of the y-axis
TOP	Upper bound of the y-axis
ALEFT	Lower bound of the x-axis. Must be a power of 10 if semi-log plot is selected.
RIGHT	Upper bound of the x-axis. If semi-log plot is selected this value is ignored and 100,000 * ALEFT is used instead.
N	The number of X,Y pairs in the data file.

NOTE: Since the linear scales are separated into five sections, if (upper bound) - (lower bound) = 5n then each major division is n units from the next. In other words, the graph labeling turns out neater if the upper and lower bounds are round numbers and the difference between them is a multiple of five.

The easiest way to form the data file is to create an array (N x 2) that contains the data pairs. If such an array exists (say it is called ARRAY(N,2)) and the above variables have been assigned values, then the following statements in your program will create the desired data file.

```
          CALL ASSIGN(1,'DX1:PLOT.DAT')
          WRITE(1,1000)LOG,BOTTOM,TOP,ALEFT,RIGHT,N
     1000 FORMAT(I1,4F10.2,I4)
          WRITE(1,3000)((ARRAY(J,K),K=1,2),J=1,N)
     3000 FORMAT(2F20.10)
```

Once the data file has been created (and the creating program's execution has terminated) the following steps will cause the plot to be made:

1) Make sure you are in monitor

2) Type: R PLOT<CR>

3) Follow the directions typed by the computer

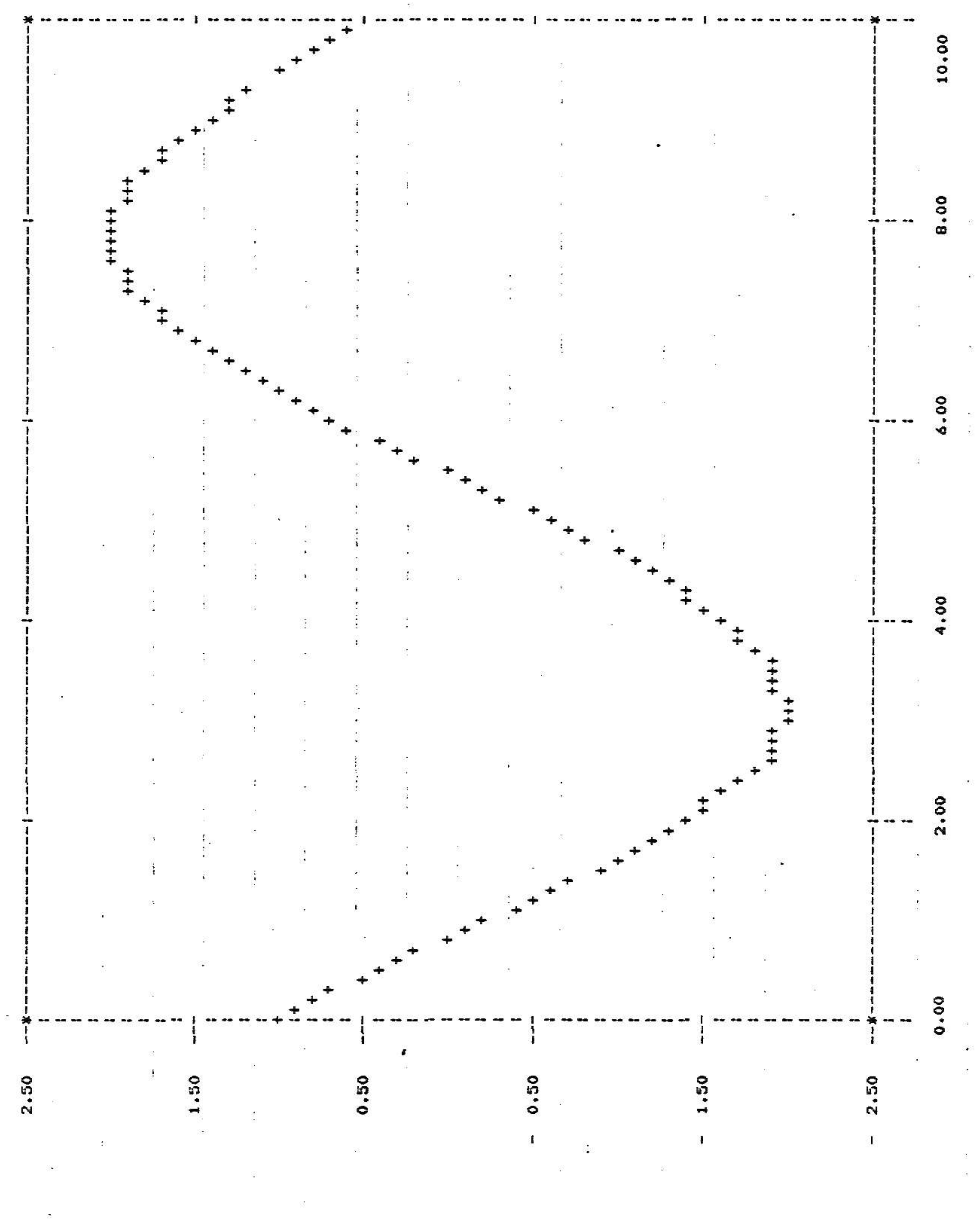
SAMPLE PROGRAM

In the pages that follow, a sample program is taken through the steps of its development. The program takes 100 A/D conversions at a sample rate of 10 KHz and creates a data file of those values vs. time for the plot routine. The use of several of the subroutines and the plot program is illustrated in this program. The following pages were written by entering and running the program on the PDP11 system. Occasionally there will be a note marked by a circled number (i.e. ③). The note will be explained on the page following the listing. Output typed by the computer is shown underlined.

```
$DX
RT-11SJ    V02C-02

.DAT 8-FEB-79

.ASSIGN DX1:DK

.R EDIT
*EWDX1:SMPLE.FOR$$①
*I      REAL ARRAY(100,2)
     ②  INTEGER IARRAY(100)
        N=100
        T=100.E-6
C
C SAMPLE RATE OF 10 KHZ
C
        IERR=0
        ICH=0
        ICLK=1
        CALL SETCLK(T)
C
C SET REAL TIME CLOCK
C
        CALL SETAD(ICH,ICLK)
C
C SET A/D CONVERTER TO CHANNEL 0, AND TRIGGERING OPTION TO
C REAL-TIME CLOCK TRIGGERING
C
        CALL RDBLK(IARRAY,N,IERR)
C
C TAKE 100 READINGS
C
        T=.1
        TIME=0
        DO 10 I=1,100
        ARRAY(I,1)=TIME
C
C X-AXIS IS TIME IN MILLI-SECONDS
C
        CALL DIGVLT(IARRAY(I),V)
C
C CONVERT A/D READINGS TO EQUIVALENT VOLTAGE
C
        ARRAY(I,2)=V
        TIME=TIME+T
10      CONTINUE
        LOG=0
        BOTTOM=-2.5
        TOP=2.5
        ALEFT=0.
        RIGHT=10.
C
C SET UP PARAMETERS FOR PLOT AND CREATE DATA FILE
C
        CALL ASSIGN(1,'DX1:PLOT.DAT')
        WRITE(1,1000)LOG,BOTTOM,TOP,ALEFT,RIGHT,N
1000    FORMAT(I1,4F10.2,I4)
        WRITE(1,3000)((ARRAY(J,K),K=1,2),J=1,N)
3000    FORMAT(2F20.10)
        STOP
        END
$EX$$
```

```
.R FORTRAN                                ③
*DX1:SMPLE,TT:/L:1/W=DX1:SMPLE:\:\
FORTRAN IV        V01C-03A    THU 08-FEB-79 00:00:00                     PAGE 001

            REAL ARRAY(100,2)
            INTEGER IARRAY(100)
            N=100
            T=100.E-6
      C
      C SAMPLE RATE OF 10 KHZ
      C
            IERR=0
            ICH=0
            ICLK=1
            CALL SETCLK(T)
      C
      C SET REAL TIME CLOCK
      C
            CALL SETAD(ICH,ICLK)
      C
      C SET A/D CONVERTER TO CHANNEL 0, AND TRIGGERING OPTION TO
      C REAL-TIME CLOCK TRIGGERING
      C
            CALL RDBLK(IARRAY,N,IERR)
      C
      C TAKE 100 READINGS
      C
            T=.1
            TIME=0
            DO 10 I=1,100
            ARRAY(I,1)=TIME
      C
      C X-AXIS IS TIME IN MILLI-SECONDS
      C
            CALL DIGVLT(IARRAY(I),V)
      C
      C CONVERT A/D READINGS TO EQUIVALENT VOLTAGE
      C
            ARRAY(I,2)=V
            TIME=TIME+T
      10    CONTINUE
            LOG=0
            BOTTOM=-2.5
            TOP=2.5
            ALEFT=0.
            RIGHT=10.
      C
      C SET UP PARAMETERS FOR PLOT AND CREATE DATA FILE
      C
            CALL ASSIGN(1,'DX1:PLOT.DAT')
            WRITE(1,1000)LOG,BOTTOM,TOP,ALEFT,RIGHT,N
      1000  FORMAT(I1,4F10.2,I4)
            WRITE(1,3000)((ARRAY(J,K),K=1,2),J=1,N)
      3000  FORMAT(2F20.10)
            STOP
            END
*^C

.R LINK
*DX1:SMPLE=DX1:SMPLE,SY:LABLIB/F

*^C④
```

```
.RUN SMPLE

STOP --

.R PLOT

DO YOU WISH TO ALTER PLOT PARAMETERS? (1=YES, 0=NO)
0

PRINT DATA POINTS? (1=YES, 0=NO)
0

ADVANCE TO BOTTOM OF PAGE AND TYPE 'GO<CR>'
```



```
STOP --
```

① <ESC> is printed as a $ by the computer

② Correct FORTRAN alignment is generated by the computer if <TAB> is typed before each statement (if the statement has a label number, type <TAB> between the label number and the statement).

③ Errors can be deleted by typing delete once for each character to be deleted. The computer will type a backslash "\", echo print each character deleted, then type another backslash when characters are entered again.

④ <control C> is printed as a ^C by the computer.

TURN-ON PROCEDURE

1. Make sure all three switches on the front panel of the computer are in their down position (this turns the computer off).

2. Turn on power strip on the back of the large wooden bench above the computer if it isn't already on.

3. Make sure the floppy disk labeled "*INSTRUMENTATION* LAB SYSTEM DISK" is in the left hand disk drive. (When inserting a floppy disk, the label should be up and towards you.)

4. Make sure your group's floppy disk is in the right hand disk drive.

5. Make sure the Decwriter is turned on.

6. Flip the two leftmost front panel switches to their up position. (Both must be flipped at the same time.) The computer will type $.

7. Type: DX <CR>
   After a short pause, the computer will respond:
   ```
       RT-11SJ  V02C-02
   ```

8. Type: DAT dd-mmm-yy <CR>
   Where dd is the day of the month, mmm is the first three letters of the current month, and yy is the last two digits of the year.
   The computer will respond with ".".

   EX: DAT 23-MAR-79 <CR>

9. Type: ASSIGN DX1:DK <CR>

If you did everything right you are now in monitor and the computer should have typed the prompting character ".".

TURN-OFF PROCEDURE

1. Place all three front panel switches in their down position.

2. Turn off the power strip on the back of the wooden bench above the computer.

HELPFUL HINTS

In the course of developing this user's guide it became evident that there were several "tricks" to using the system that weren't obvious from the explanation in the manual.

1. When entering a program for the first time use the "I" editor command after typing EWDX1:filename.FOR<ESC> to "insert" the program.

2. When you are through inserting characters with the "I" editor command, make sure you terminate your input string with <ESC>.

3. Typing mistakes can be deleted by typing <delete> once for each character to be deleted.

4. The manual contains a sample program entered on the PDP-11. Looking at that sample program might clarify confusing instructions in the manual.

5. To get a listing of your program while in editor, type:

   ```
   B<ESC>/L<ESC><ESC>
   ```

   (this also moves the pointer to the top of the file.)

6. Liberal use of the "V" editor command helps prevent errors due to not knowing where the pointer is.

7. To move the editor pointer to the bottom of a file, search for a character that isn't in the file.

8. If you have done the turn-on procedure properly, you can omit the "DX1:" part of command lines.

APPENDIX H

LABORATORY ASSISTANT MANUAL

In order to protect the system software from the students a powerful user program was omitted from the system disk used by the students. This program is on the disk labeled "INSTRUMENTATION LAB SYSTEM DISK MASTER". By using this disk in the left disk drive instead of the system disk used by the students, the laboratory assistant can perform several operations on a user disk that a student cannot.

Erasing a user disk (also used to make a blank disk a user disk).

1. Go through the turn-on procedure in the laboratory user's guide with the disk labeled "INSTRUMENTATION LAB SYSTEM DISK MASTER" in the left disk drive, and the disk to be erased in the right disk drive.
2. Type: R PIP<CR>
3. Type: DX1:/Z<CR>
   The computer will respond with DX1:/Z ARE YOU SURE?
4. If sure (this will destroy all files on the disk).
   Type: Y<CR>
5. To erase another disk, replace the disk in the right disk drive with the next disk to be erased and go to step 3.
6. If done, go through the turn-off procedure in the laboratory user's guide, then replace the disk in the left disk drive with the system disk used by the students.

"Cleaning up" a student user disk.

1. Go through the turn on procedure in the laboratory user's guide with the disk labeled "INSTRUMENTATION LAB SYSTEM DISK MASTER" in the left disk drive, and the disk to be "cleaned up" in the right disk drive.
2. Type: R PIP<CR>

3. To list the files on the disk:
   Type: DX1:/E<CR>

   To delete a file from the disk:
   Type: DX1:filename.FOR/D<CR>

4. Type: DX1:*.BAK/D<CR>

5. Type: DX1:*.BAK/D<CR>

6. Type: DX1:/S<CR>

7. Go through the turn-off procedure in the laboratory user's guide, then replace the disk in the left disk drive with the system disk used by the students.

LABLIB DOCUMENTATION

The source listings and documentation files for the LABLIB subroutines and PLOT are on the disk labeled "INSTRUMENTATION LAB SOFTWARE DOCUMENTATION".

To get a copy of a file.

1. Follow the turn-on procedure in the user's guide with the disk labeled "INSTRUMENTATION LAB SOFTWARE DOCUMENTATION" in the right disk drive.

2. Type: R EDIT<CR>

3. To get a copy of a source listing:
   Type: ERDX1:subroutine.FOR<ESC>R<ESC>/L<ESC>/K<ESC><ESC>

   To get a copy of a description:
   Type: ERDX1:subroutine.DOC<ESC>R<ESC>/L<ESC>/K<ESC><ESC>

   Repeat step 3 for each copy desired.

4. Go through the turn-off procedure in the user's guide.

5. Return disk labeled "INSTRUMENTATION LAB SOFTWARE DOCUMENTATION" to the disk box.

To modify a subroutine in LABLIB the following steps must be done.

1. Using EDIT
   a) make the modification.
   b) increase version number.
   c) update "date of last modification" in heading.
   d) describe modification in heading.
   e) get a clean copy on a new page.

2. Put new version in notebook. Keep old version in notebook.

3. If necessary, correct the description using EDIT.

4. Compile new version.

5. Using LIBR, replace subroutine in LABLIB with new version.

6. Using PIP, replace all copies of LABLIB with new version.

APPENDIX I

COMPLETE SYSTEM

The computer system used in the laboratory is set-up to only use a part of its capabilities. This was done to make the laboratory system easier to use. Other capabilities, which require a more complete understanding of the PDP11/03 system, can be accessed by using a different system disk ("COMPLETE SYSTEM DISK MASTER").

This appendix describes some of the other capabilities, explains a little more about the PDP11/03, and indicates where to look in DEC's reference manuals for the information most needed by the user.

FILENAMES

To use the more general system , the user must understand the system used by DEC to specify a file. The file specification takes the form ddd:ffffff.eee where "ddd" is the device where the file is located, "ffffff" is the filename, and "eee" is the type of file.

The device ("ddd") can be specified several ways. The device can be directly named ("DX0" is the left disk drive, "DX1" is the right disk drive), the device can be indirectly named ("SY" is the system, which is always the left disk drive), or the device specification (and its ":" delimiter) can be omitted. If the device specification is omitted, the monitor uses the current default device. The PDP11/03 turns on with "DX0" the default device and the line "ASSIGN DX1:DK" in the turn-on procedure assigns DX1 to be the default device.

The filename "ffffff" is a name (1 to 6 characters) assigned to the file. The extension "eee" of the file indicates what type of file it is. Possible extensions are:

FOR	Fortran Source File
MAC	Assembler Source File
BAK	Back-up Source File
OBJ	Object File
SAV	Executable File
DAT	Data File
SYS	System File

In many instances (whenever the type of file specified is obvious by usage) the extension may be omitted. For instance, "FORTRA" always uses an "FOR" file as a source and creates an "OBJ" file, so those extensions may be omitted. When using editor, the extensions must be specified.

A "BAK" file is created by EDIT when a file to be modified is loaded into editor with an "EBddd.ffffff.eee" command. After the modifications are made and editor is terminated with the "EX" command, the editor renames the unchanged, original file with a "BAK" extension and stores the modified file under the filename specified. If you do not want "BAK" files, enter files to be modified into editor with these commands:

```
ERddd:ffffff.eee<ESC>R<ESC>EWddd:ffffff.eee<ESC><ESC>
  source file               destination file
```

A data file can be written or read by a FORTRAN program by assigning a FORTRAN device number to the data file using the subroutines described in paragraph B.2 of RT-11/RSTS/E FORTRAN IV USER'S GUIDE.

COMPLETE SYSTEM DISK

To make room for "LABLIB" and "PLOT" several system programs were omitted form the instrumentation laboratory system disk. These programs are on the disk labeled "COMPLETE SYSTEM DISK MASTER". The files on this disk that are not on the laboratory system disk are:

PIP
LIBR
MACRO
SYSMAC
PATCH
ODT
SYSLIB

PIP: The Peripheral Interchange Program is used to manipulate files. The features of this program are described in Chapter 4, <u>RT-11 SYSTEM REFERENCE MANUAL.</u> The commands found to be most useful are the following:

ddd:/E Lists the directory of the indicated device.

ddd:ffffff.eee/D Deletes the specified file.

ddd:/S Compresses the files on the device. It combines the unused blocks into one large block.

ddd:ffffff.eee=ddd:ffffff.eee Creates a copy of the file specified on the right side of the "=" under filename given on the left side of the "=".

LIBR: Used to make a subroutine library like LABLIB. This program is described in Chapter 7, RT-11 <u>SYSTEM REFERENCE MANUAL</u>.

MACRO:
SYSMAC: "MACRO" and "SYSMAC" are the files needed to assemble programs written in assembler. Descriptions that will help in writing assembler subroutines and programs can be found in the following sections:

Section 3 (pages 479-581) <u>MICROCOMPUTER HANDBOOK</u>
Chapter 5 <u>RT-11 SYSTEM REFERENCE MANUAL</u>
Chapter 9 <u>RT-11 SYSTEM REFERENCE MANUAL</u>
Appendix C <u>RT-11 SYSTEM REFERENCE MANUAL</u>
Appendix E <u>RT-11 SYSTEM REFERENCE MANUAL</u>
Sections 2.3 and 2.4 <u>RT-11/RSTS/E FORTRAN IV GUIDE</u>

PATCH:	Described in Appendix L, <u>RT-11 SYSTEM REFERENCE MANUAL</u>.
ODT:	Described in Chapter 8, <u>RT-11 SYSTEM REFERENCE MANUAL</u>.
SYSLIB:	Library of useful FORTRAN-callable subroutines that allow some access to system functions from a FORTRAN program. Described in Appendix O, <u>RT-11 SYSTEM REFERENCE MANUAL</u>.

CRT TERMINAL

To use the CRT terminal instead of the Decwriter, use the following procedure:

1. Unplug the Decwriter from the rear of the PDP11/03 and plug in the CRT terminal.
2. Remove the top cover-plate on the PDP11/03.
3. Set the small slide switch on the bottom right printed circuit board to 4800.
4. Turn on the CRT terminal.
5. After you are finished be sure to return the slide switch to 300 and reconnect the Decwriter.

Note: Since the screen only displays 24 lines, a listing of over 24 lines causes the top lines of the listing to go off the top of the screen before they can be read. To control the rate that new lines are sent to the CRT:

With the computer in monitor, type:

```
SET TTY HOLD<CR>
```

Now, after the screen is filled, typing <SCROLL> allows one new line to appear and typing <SCROLL> while depressing <SHIFT> allows 24 new lines to appear. To return to the normal mode:

With the computer in monitor, type:

```
SET TTY NOHOLD<CR>
```

MAKING A COPYABLE USER DISK

In general, it is a good practice to have each user disk contain the bootstrap loader, monitor, and PIP. If this is done, a copy of the disk can be made by treating it as a system disk and following the directions for copying a disk.

To make a user disk with the above files, the following procedure must be used when the disk is initialized.

1. Follow the turn-on procedure in the laboratory user's guide with the disk labeled "COMPLETE SYSTEM DISK MASTER" in the left disk drive, and a blank disk in the right disk drive.

2. Type: R PIP<CR>
   DX1:/Z <CR>
   The computer will respond: DX1:/Z ARE YOU SURE?
   Type: Y<CR>

3. Type:MONITR.SYS=SY:MONITR.SYS/X/Y<CR>
   Ignore the ?REBOOT? message returned.
   Type: TT.SYS=SY:TT.SYS/X/Y<CR>
   PIP.SAV=SY:PIP.SAV/X<CR>
   DUMMY=MONITR.SYS/U<CR>

TO COPY A DISK

1. Follow the turn-on procedure in the laboratory user's guide with the disk to be copied in the left disk drive, and a blank disk in the right disk drive.

2. Type: R PIP<CR>

   ```
   DX1:/Z<CR>
   ```

   The computer will respond: DX1:/Z ARE YOU SURE?

   Type: Y<CR>

   ```
   DX1:*.*/X=SY:*.*/X/Y<CR>
   DX1:A=MONITR.SYS/U<CR>
   ```

SUBROUTINES

To use a subroutine with a FORTRAN program, the subroutine must first be created as a separate file and compiled (or assembled).* When linking the main program include the subroutine(s) filename(s), separated by commas, immediately after the main program filename on the right-hand side of the "=".

EXAMPLE: "MAIN" is a program that calls a subroutine "SUB". "MAIN" and "SUB" have already been written and compiled. To link "MAIN" the following command will be typed:

```
DX1:MAIN=DX1:MAIN,DX1:SUB,SY:LABLIB/F<CR>
```

* For a description of the parameter passing convention used by this FORTRAN implementation see sections 2.3 and 2.4 of the RT-11/RSTS/E FORTRAN IV USER'S GUIDE. This information is needed to write an assembler subroutine.

WHERE TO FIND INFORMATION

<u>DECLAB-03 HANDBOOK</u>: (little green book)

How to copy a disk	page 14
Creating a user disk	page 15
CRT scroll feature	page 24
FORTRAN error messages	pages 65-84

<u>PDP-11 FORTRAN LANGUAGE REFERENCE MANUAL</u>:

<u>RT-11/RSTS/E FORTRAN IV USER'S GUIDE</u>:

Filenames	1.1.1
FORTRAN compiler	1.2
Link	1.3
Subroutine linkage	2.3
Subroutine register usage	2.4
Device/file default assignments	3.6
System subroutines	Appendix B
Error diagnostics	Appendix C

<u>MICROCOMPUTER HANDBOOK</u>

Module descriptions:	
A/D	p186-192,328-339
D/A	p184-186,322-328
IEEE-488 interface	p180-184,314-322
Real-time clock	p211-215,349-365
Serial output	p147-153,259-270
Architecture and Assembler	p505-583
Instruction timing	Appendix B

<u>RT-11 SYSTEM REFERENCE MANUAL</u>

Editor	Chapter 3
PIP	Chapter 4
Assembler	Chapters 5 and 9
	Appendices C,D, and E
Link	Chapter 6
Librarian	Chapter 7
SYSLIB	Appendix 0

AN INSTRUMENTATION LABORATORY COMPUTER SYSTEM

by

GREG DEGI

B. S., Kansas State University, 1977

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
1979

ABSTRACT

A computer system based on a PDP11/03 minicomputer has been developed for use in the KSU Electrical Engineering instrumentation laboratory. System software includes a simplified user's guide and a library of FORTRAN-callable subroutines. The subroutine library consists of subroutines which control the PDP11/03 laboratory peripherals: real-time clock, A/D converter, D/A converter, and IEEE-488 bus interface. In addition to the description of the system software, an appendix which describes some capabilities of the system which were omitted from the user's guide is included.

Several items of hardware designed and built for the system are described. These items include: buffers for the D/A and A/D converters, a patch panel, an IEEE-488 compatible frequency counter, and an IEEE-488 compatible multiplexer. The multiplexer uses an INTEL 8748 microprocessor to interface with the bus. This interface is documented in an appendix.