DEC-14-GGZA-D

PDP-14
USER'S MANUAL

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Printing November 1969

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

ILLUSTRATIONS

ILLUSTRATIONS (Cont)

ILLUSTRATIONS (Cont)

TABLES

TABLES (Cont)

PURPOSE OF THE USER'S MANUAL

This User's Manual describes the PDP-14 Industrial Control System in three general categories:

Chapters 1 through 3 define the applications of the PDP-14, and explain the function of the system, its major components, and monitoring additions.

Chapters 4 through 10 provide complete instructions for designing control operations using simplified computer programming techniques, and describes programming for computer monitor.

Chapters 11 and 12 contain installation and maintenance concepts for the control engineer and electrician.

A glossary is also included to define unusual or special terms.

The User's Manual is intended for readers who are familiar with industrial control techniques and basic electrical terms. Using this manual, special training in electronics or computer operation is not required to operate the system. (The chapters in the manual which discuss monitoring are written for computer programmers.)

OTHER PDP-14 TECHNICAL INFORMATION

A complete PDP-14 Maintenance Manual for use by the electronic technician is available. This Manual describes the operation and servicing of all electronic circuits in the system. Information on some modules of the PDP-14 is provided in the DEC Control Handbook.

PREFACE

GENERAL

This User's Manual describes the PDP-14 Industrial Control System (Frontispiece), designed and manufactured by Digital Equipment Corporation (DEC). The PDP-14 is of a new solid-state design which provides remarkable advantages over previously available systems.

APPLICATIONS

The PDP-14 can control any operation or process which can be broken into a number of discrete steps or logical expressions, each with just two states. That is, status inputs are either on or off, and the control operation selects outputs and turns them on or off. This type of control sequence is used in most mass-production equipment and materials handling systems. It is also used in such diverse industries as automotive, steel, food processing, petrochemical, appliances, and others.

CONTROL FLEXIBILITY

The PDP-14 system uses an easily replaceable memory to direct specific control operations. Convenient computer programming techniques allow each user to design a memory to suit his unique control needs. The entire control process can be redefined by changing the memory. The memory operation is reliable, and cannot be accidentally altered by electrical means.

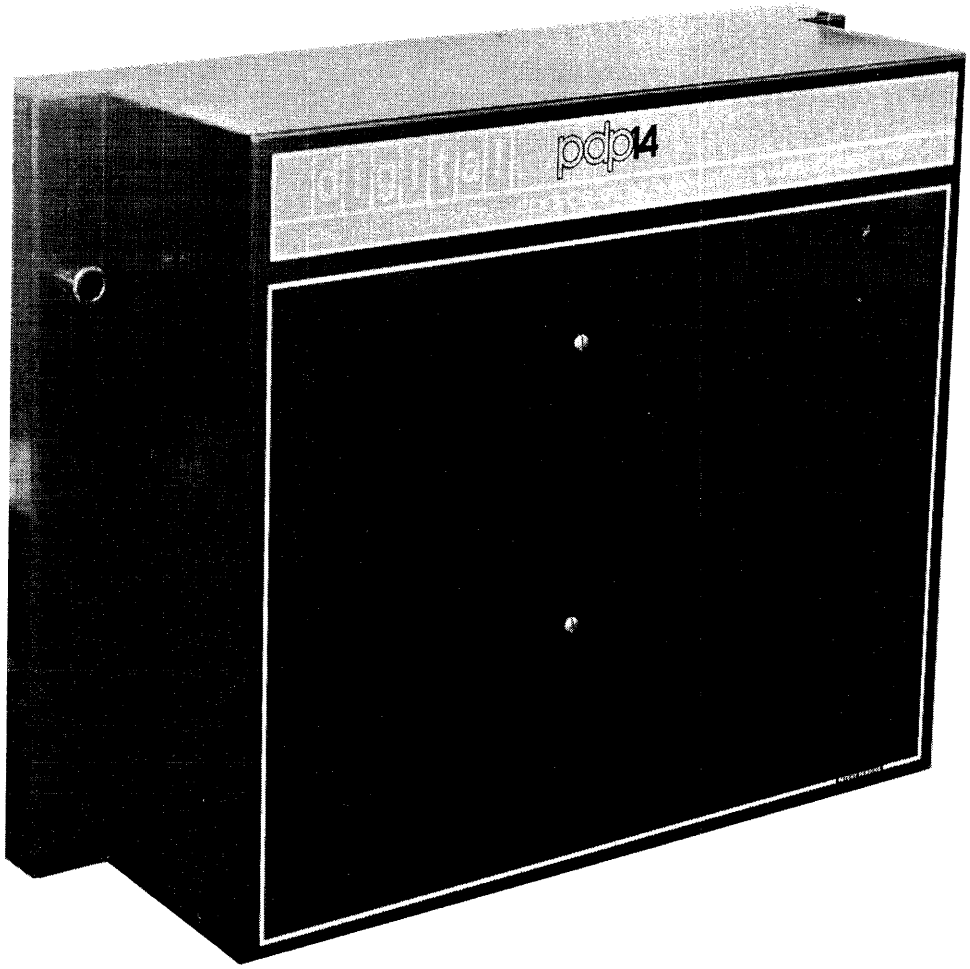
COMPUTER MONITORING

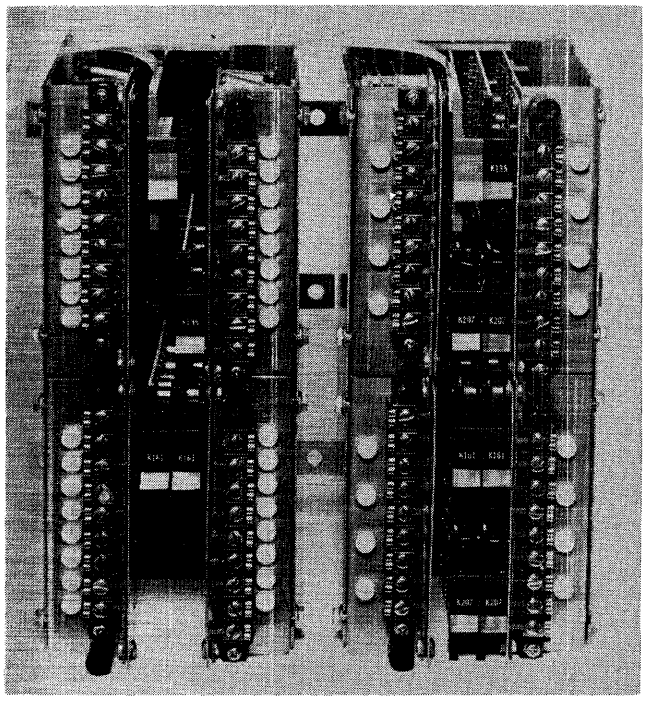
The PDP-14 system is designed to operate independently or with computer monitoring or control. Since the PDP-14 can access all control inputs and outputs, the added monitoring components are simply a general-purpose computer and a simple interface device (available from DEC). A group of PDP-14 systems can be monitored by a single computer using a multiplexer, to provide status and malfunction reports for a large control complex.

OTHER FEATURES

The PDP-14 system is extremely reliable, using no moving parts for control operations. All components are modular, and are fully inspected and tested before and after assembly. Diagnosis of system faults is highly automated, and can be accomplished by operators with only limited training in electronic systems.

The PDP-14 system is extremely compact, and requires very little operating power. Installation is quick and simple, and requires no special training.





PDP-14 Industrial Control System

CHAPTER 1

GENERAL CONCEPTS OF INDUSTRIAL CONTROL SYSTEMS

The chapter first discusses the general concepts of industrial control systems of all types. Binary and numeric control systems are described as they are ordinarily encountered in mass production or materials handling.

Secondly, the chapter focuses on binary control systems and develops criteria for judging their relative value as production equipment. Strengths and weaknesses of conventional systems are discussed, and some desirable improvements are indicated.

GENERAL CONTROL SYSTEMS

Control Systems are organized in general as shown in the following illustration.

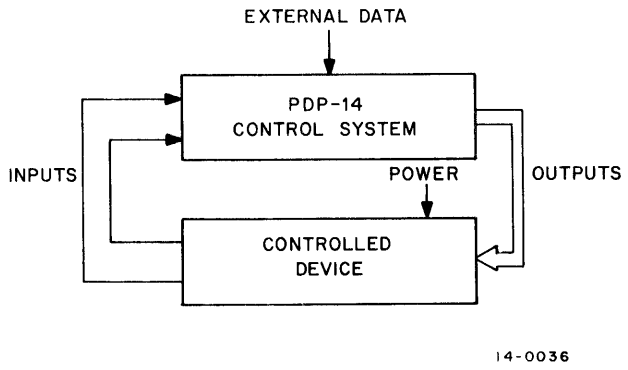


Figure 1-1 General Control Systems

The control unit senses the status (or position) of the controlled device through its inputs. Specific input conditions cause the control unit to send signals to the controlled device to effect its operation. The operation results in changes in the device status, and thus changes in the input signals to the control unit. The control unit then reacts to this new status by changing outputs, etc. This cyclic process is termed a control loop. The process of transforming a small input signal into a higher energy control operation is called control gain, and

(as in any amplification process) requires energy which, by itself, does no useful work. The external data are status information which do not result from changes in device condition (for example, from operator controls and prepared instruction commands).

CATEGORIES OF CONTROL SYSTEMS

Industrial control systems are often divided into two general categories according to the type of control signals: numeric control and binary control. (Although the PDP-14 system is a binary controller, we will briefly define numeric control.)

Numeric Control

Numeric control (NC) devices use control signals (inputs and outputs) which range over many voltage values, each defined by a number. The variation of input magnitude causes a fixed, proportional change in output magnitude. Equipment which responds precisely to varying signals is then required to transform control commands into action.

NC is a flexible method of controlling a small number of relatively intricate operations. For example, a multi-purpose, NC-controlled machine can be given a series of instructions (usually on punched or magnetic tape), and then proceed through dozens of automatic steps to transform raw blanks into complex finished pieces. The memory of required operations consists of the instructions on tape, which can be changed at any time. NC control implies:

a single multi-purpose machine
a small production lot
moderate production rates
relatively high cost, and
programmed operation

For small lots, NC techniques reduce set-up time enormously, and therefore justify the high cost.

Binary Control

Binary control systems use control signals having only two values - ON or OFF. Binary controllers are commonly used in mass production, where operations are divided into a number of simple functions, each defined by only two conditions. For example, a drill head is up or down, a moving bed is left or right, a motor is on or off. Binary control signals can be hydraulic, electric, pneumatic, fluidic, or even mechanical. Commonly, the input signals are all electric and the outputs are combinations of electrical, and electrically-activated hydraulic and pneumatic systems. Electric binary controls commonly are in the form shown in Figure 1-2.

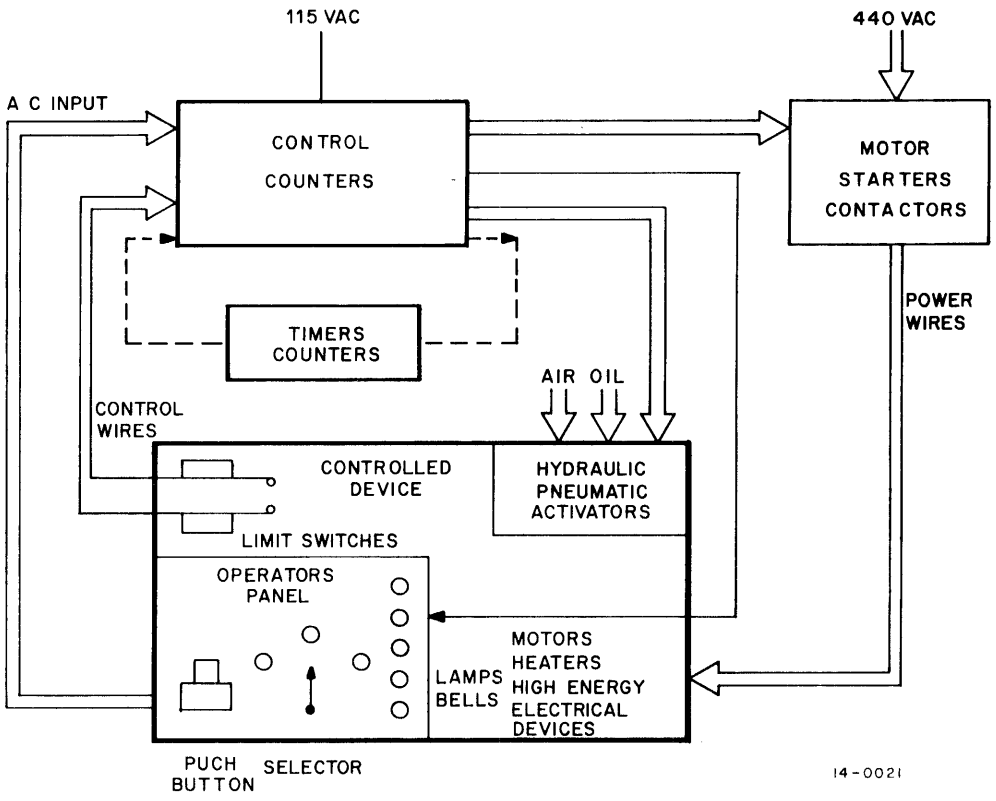


Figure 1-2 Binary Control Systems (Electric)

The control unit has three types of inputs:

1. those responding to the status or position of various sections of the controlled device
2. others established by the operator (initial conditions, manual operations, and emergency controls)
3. pre-set status signals (from timers and counters).

Because the operation is a number of simple steps, status inputs can be generated by simple devices such as limit sensors and proximity detectors. Operator commands are sensed by pushbuttons and selector switches. Each input is controlled by some variety of switch which turns power on or off.

Each unique combination of inputs stipulates which outputs will be activated or deactivated.

The control function has three general steps:

- TEST inputs for state (on or off)
- DECIDE what control function is required (which outputs must be controlled)
- SET outputs on or off (execute control decision).

In electrical binary controls, decision making is accomplished by relays and the wires which connect them. The wires actually represent the control memory, for they route the input control signals to the proper sets of intermediate relays for processing to the proper outputs. The control system must be rewired whenever a change in operation is needed.

Control system outputs are ordinarily low-power, and must be boosted to higher energy to operate most equipment (Exceptions are indicating devices and small solenoids and motors.) The control signals drive contactors, to switch high energy electrical loads, and solenoid valves, to activate hydraulic and pneumatic power. Control outputs also activate timers and counters, which are later sensed as inputs.

The controlled device responds to changing control signals by producing directed work. Input sensors determine when this operation proceeds to a new condition which requires a change in actions. The sensors deliver new electrical signals to the control unit and the control loop continues.

The "ladder diagram" is commonly used to present an electrical schematic of system operation. Each unique machine function is represented by one rung of the ladder. All inputs which determine a single required equipment action are on one side of the ladder and the output is on the other. To initiate a machine operation, the unique combination of variables (control inputs) which allow that operation must be supplied.

Binary control applied to production equipment implies:

many relatively simple machines - single purpose
often repeated cycles
low cost for control

These criteria represent the general requirements of industrial mass-production equipment, but do not necessarily establish the limits of binary control systems.

The PDP-14 control system is an example of an electrical binary control system. It is the first really new system of its type to appear in several decades. It would appear that most industrial users are satisfied with their present systems but this is often only because satisfaction has been created by familiarity. Therefore, the PDP-14 merits its own evaluation in terms of something new in control equipment.

The following discussion provides some guidelines for evaluating binary control systems in terms of something better than the status quo in control equipment. The performance of a control system can be defined in terms of efficiency (including economic), versatility, maintainability, and performance monitoring.

EFFICIENCY

The overall efficiency of a control system can be measured in terms of:

Cost - the amount of control provided for the money (dollars per input or output)
Service life of the entire system
Life of individual components (MTBF)
Cost of repair, including length of downtime
Speed of the system (efficient use of control time)
Efficient use of space (physical size of the system)
Electrical efficiency (use of energy to provide control)

Present relay control systems are reasonably reliable. In low-power signal switching for railroad use, relays have performed millions of correct actions. The cost of relays is also considered reasonable by many users. However, today's industry often demands that moderate control currents be switched at high repetition rates and, in this case the practical use of relays become limited. One operation per second is more than 30 million a year, which only special-purpose relays can withstand for any long period of time. There is also an upper limit of relay response time. Relays are somewhat bulky devices; and a complete relay control system may require a significant percentage of available manufacturing space. Any attempt to miniaturize a relay system makes servicing more difficult.

Diagnosis of faults is the most perplexing problem of conventional control systems. When one unit of a highly-interconnected system deteriorates, the real problems may appear to be at any one of several places or at several places at once. Troubleshooting often begins with an educated guess and continues through several time-consuming circuit checks. A complete monitoring system is a fine diagnostic aid, but can be quite expensive if not designed as part of the original control system. Monitoring ordinarily requires considerable additional equipment.

VERSATILITY

The versatility factor can be discussed in terms of:

Ability of a pre-packed control system to be adapted to a specific need
Ease of system design and construction (if a user must design and build his own system from supplied building blocks)
Ease of control revision or retrofit, when modifying present functions or changing entire control requirements.

System changes and adaptations must remain reasonably permanent. Neither the environment nor accidental operator error should alter proper control operation. Although it seems a contradiction in purpose, the ideal

control system should be immune to arbitrary change but easily modified in case control requirements vary. Present relay-type control signals are quite immune to accidental change except that, in corrosive environments, contacts and electrical connections may deteriorate. Since each relay systems is custom-designed, it can do its job well. However, this specialized kind of system does not easily adapt itself to desired updating, since component layouts and wiring changes must be overlaid on the existing control structure. (Just documentating system modifications can be a major effort.) When complete control requirements change, it is often more economical to discard the old system and design an entirely new one.

Digital computer technology seems to offer a handy combination of easy system design and simple system modification. A designer could provide his instructions as a program list, which the computer would learn and then provide control through an all-purpose system. Control changes would be equivalent to teaching the computer new tricks, which is quick and convenient. Unfortunately, ordinary computers can accidentally forget (and thus provide incorrect control). They can be also susceptible to electrical noise often found in industrial environments.

MAINTAINABILITY

To be easily maintained in operation, a control system must have:

a. A low failure rate of individual components (high MTBF).
b. Faults which can be quickly traced to specific components.
c. Easily and quickly replaceable parts and readily available spares.

One producer of heavy automotive manufacturing equipment estimated that 80% of control repair time is spent finding system faults and only 20% repairing them. Here is a major area for improvement. Diagnosis of faults is often the greatest single non-machinery factor contributing to downtime.

Once the difficulty is isolated, nearly every control system allows for modular parts replacement. To replace coils and contacts in relays is a relatively easy process. Even though replacement parts are industry standardized and easy to obtain, a large parts stock may sometimes be required for the many varieties of relays available.

MONITORING OF PERFORMANCE

A very useful feature of modern control systems is their ability to monitor and report equipment operations. Is the equipment operating properly or are slowdowns occurring? Has degraded machine performance indicated a need for preventive maintenance before failure can occur? Is the production schedule being met? These questions can be answered continually by computer-assisted control equipment.

Unfortunately, computer monitoring is often added as a retrofit operation. This means that an interface apparatus is added piggyback fashion to an ordinary control system to provide monitoring points for the computer. Adding this monitor requires a complete re-evaluation of the old system and a number of new parts. Monitoring and control are limited to the test points selected by the designer. Monitoring can be easily more trouble than it's worth when these limitations are imposed. The solution is an integral control/monitor package which can be augmented as production operations change.

CHAPTER 2
PDP-14 SYSTEM INTRODUCTION

PURPOSE

This chapter explains the major features of the PDP-14 control system and its function. The major units of the system are presented here and are discussed in detail in Chapter 3.

HISTORY

For several years, Digital Equipment Corporation (DEC) has been developing a large reserve of computer technology. Our industrial interest has centered around numeric control computer programs and the popular K-series control modules. We had created custom control systems of many sorts, but no general-purpose industrial equipment.

In the fall of 1968, one of our industrial customers approached us with a request for an all-purpose, solid-state industrial controller. We found that much of the technology to meet his requirements was right "on our shelf", and needed only to be combined into a useful system.

From our experience with small computers, we borrowed digital logic circuits and a modular concept. We adapted the rugged K-series industrial control modules. And finally, DEC mass production skill was concentrated on developing a reliable, low-cost system. When we finished development, we found that we could supply most industrial users with a product greatly improved over previous control systems. This is the PDP-14.

TRAINING FOR THE PDP-14

The PDP-14 is functionally equivalent to nearly all binary control systems in industrial use. This means that one should understand in general how such systems operate, and how desired control operations may be translated into electrical circuitry. Machine sequence charts and electrical ladder diagrams are two of the common design aids you should be familiar with.

The PDP-14 system is easy and straightforward to install and use since it simplifies, rather than complicates, electrical design of the control operation. This manual provides the new information required about programming

the system for specific control needs, and about installation and computer monitoring operations. DEC also offers intensive training courses on various PDP-14 topics to supplement this manual. Check with your local sales office for information on present offerings.

THE PDP-14 SYSTEM

The PDP-14 is a related group of modular units. It becomes a system when these units are properly combined to provide a control function. You buy only what you need to perform a specific control purpose.

Features

Here are some of the features of the PDP-14 control system:

It is a truly general-purpose device, adaptable to nearly every binary control requirement.

Control operations are defined through a series of flexible computer-assisted programming steps.

Control modifications and retrofit can be made at any time with minimum downtime and without rebuilding the control system.

The control system is modular, has easily replaceable parts, and can be serviced quickly and accurately, using computer based diagnostics.

Cost is comparable to systems it can replace.

The system takes less space than equivalent systems, and uses much less electric power.

The anticipated service life of the system is well over 10 years.

Reliability is very high, and all active components are solid-state (there are no active moving parts).

General PDP-14 System

The PDP-14 control system is designed to replace conventional systems directly, with a minimum of special considerations. The equipment uses standard 115 Vac input and output signals. Each of the units is divided into plug-in modules containing solid-state devices and other components, to allow for easy repair or replacement of parts.

The PDP-14 is a direct replacement for the typical binary control system shown in Figure 1-2. Figure 2-1 shows a general system layout.

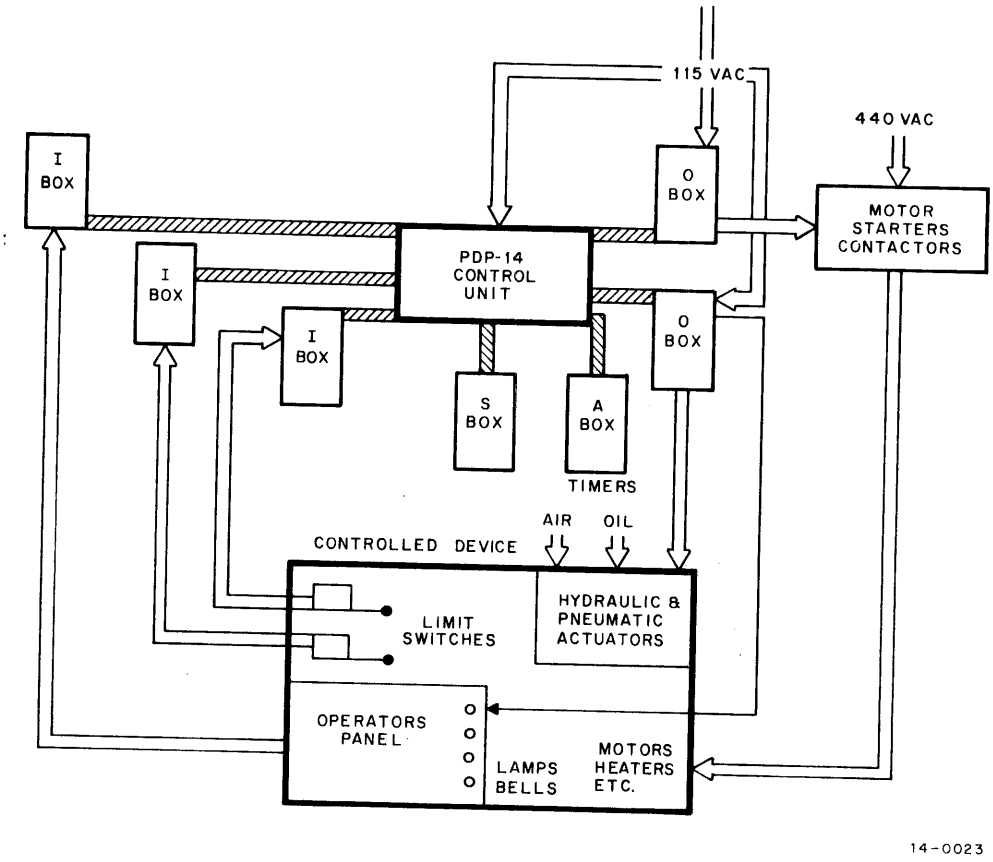
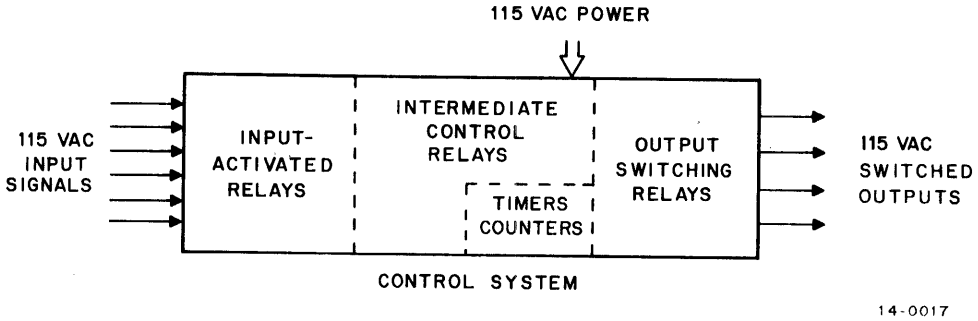
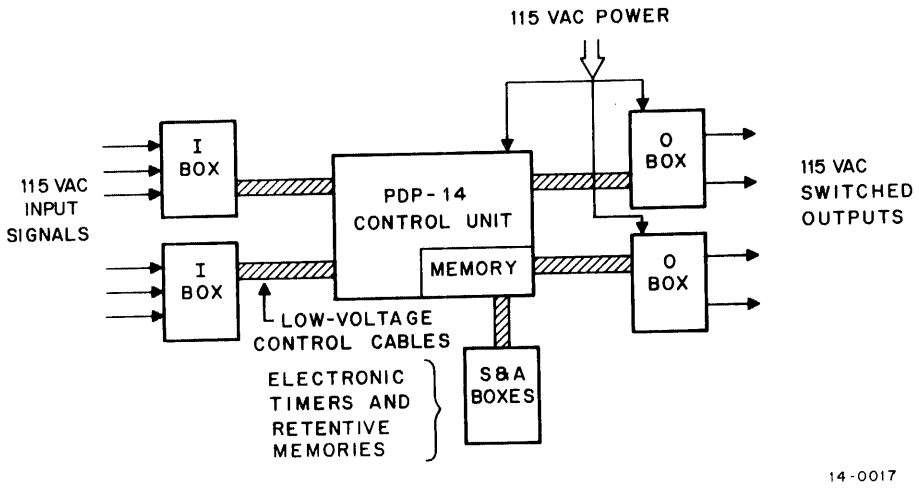


Figure 2-1 General PDP-14 Control Arrangement

The Input (I) boxes and Output (O) boxes provide an "interface" or electrical buffer between the 115 Vac control signals from the controlled device and the electronic circuits of the Control Unit. In the conventional control system we have:



And in the PDP-14 system, the equivalent parts are:



The low-voltage, solid-state circuits of the Control Unit (CU) test and direct the 115 Vac circuits of the controlled device through control cables. The I and O boxes provide effective isolation of the control circuits from the AC power supply, giving added safety when handling PDP-14 equipment. Each I or O box handles a small number of AC circuits and has its own control cable to the CU. This means that by selecting the proper number of I and O boxes, only the input and output circuits actually needed for the specific control operation, need be purchased.

The PDP-14 system is a unified assembly of five basic units consisting of a Control Unit, with its Memory, Input (I) Boxes, Output (O) Boxes, Auxilary Boxes, (timers, retentive memories, etc.), Storage Boxes.

This system does exactly what ordinary systems do; it accepts low-power AC signals at its inputs, determines which signals are necessary for a given machine function, and then turns on and off the required outputs. But it does it in a different way. The following paragraphs describe each of the PDP-14 systems separately.

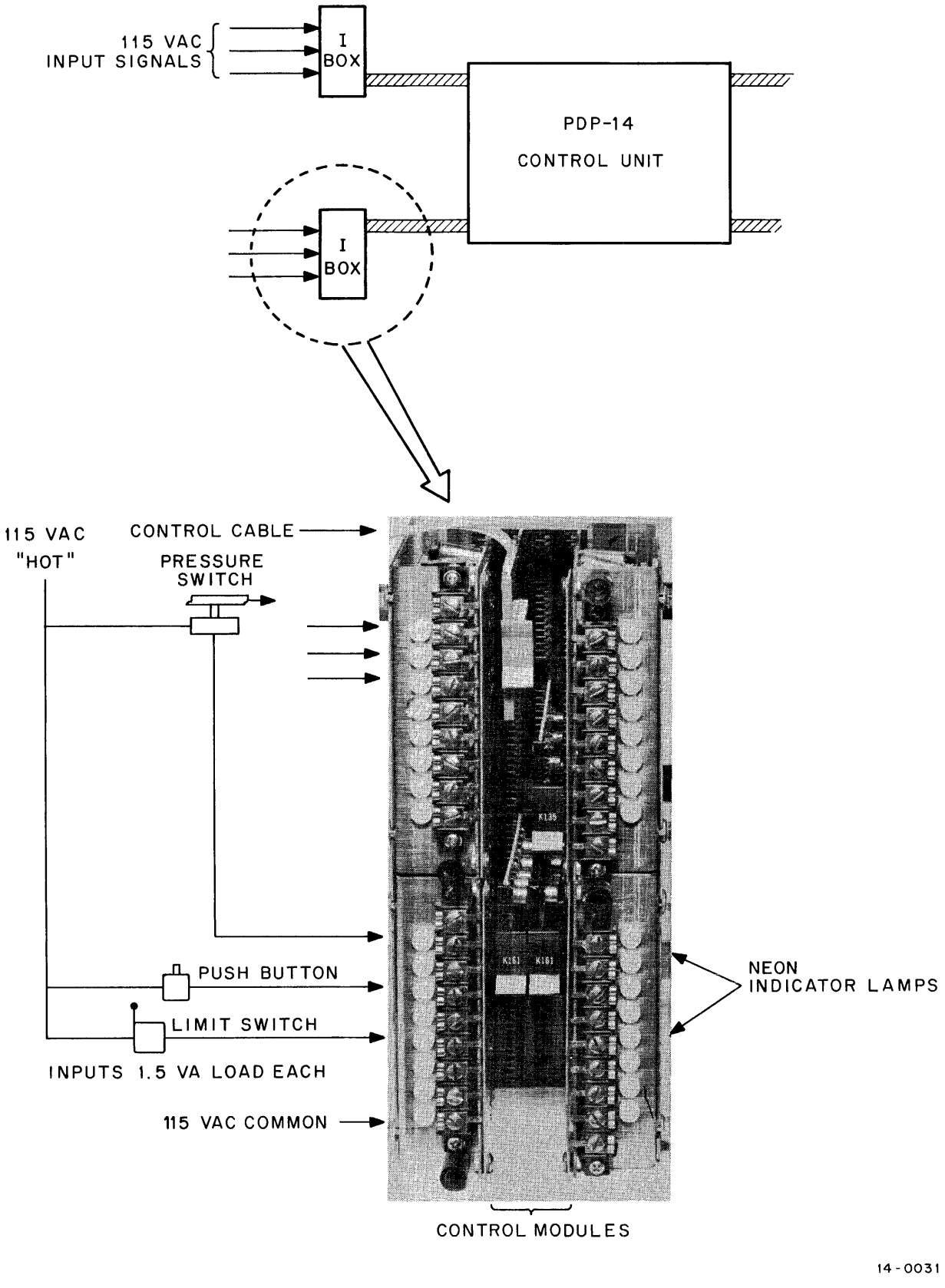


Figure 2-2 Input Box Pictorial

I Box Function (See Figure 2-3)

The I boxes are signal-conditioning devices which accept 115 Vac inputs from two-state sensing devices such as limit switches, pushbuttons, proximity switches, pressure switches, and photocells. These inputs are converted into signals which are proper for the solid-state circuits, and then pass the signals along control cables to the PDP-14 Control Unit. Each I box contains 32 independent input circuits. The Control Unit will accept control cables from a maximum of eight I-boxes (256 inputs per system). A lamp on each input circuit shows when ac power is being supplied to the box terminals.

I Box Mechanical

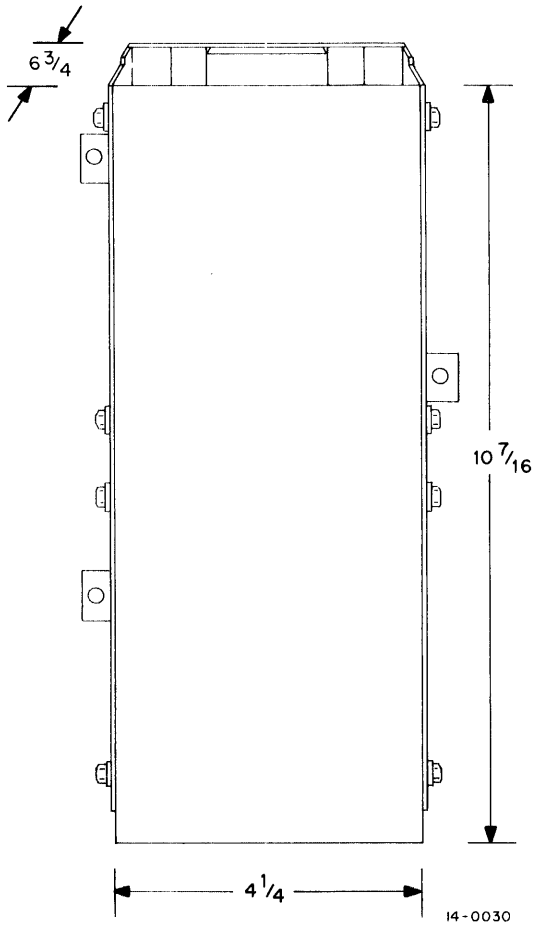


Figure 2-3 Input Box Dimensions

The I box is a strong, "U"-shaped steel channel with three mounting tabs. The box contains a wired socket on its back wall that accepts plug-in solid-state modules. Wire connections are made to screw-type terminal strips mounted on the front of the box, which accept number 14 AWG leads. A plastic shield covers the terminals and modules.

Control Unit (Figure 2-4)

The circuits of the Control Unit and the memory perform the three basic operations of any control system. They:

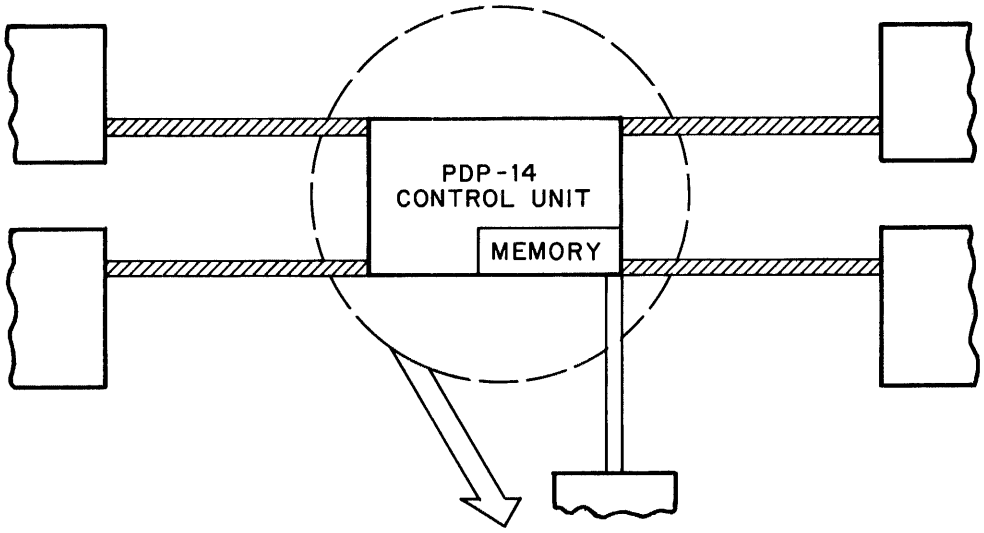
1. TEST inputs for state (ON or OFF);
2. DECIDE what control function is required; and
3. SET outputs ON or OFF (execute the control decision).

Read-Only Memory

The memory used in the PDP-14 system is called a Read-Only Memory, or ROM. It is termed "read-only" because it cannot be altered electrically (that is written on). The ROM contains a pattern or a "braid" of wires woven together. This pattern is actually a list of permanently wired electrical instructions which are read by the CU to determine system operation.

The function of memory in a conventional control system is provided by the wires which interconnect relays and inputs and outputs. They route the input control signals to the proper sets of intermediate relays for processing to the proper outputs. To modify this sort of memory, leads must be rewired and relays added or removed, an inconvenient process at best.

The ROM braid is the functional equivalent of all the wiring in a conventional system. A control designer specifies the instructions (to suit his control task) to be placed in the braid for each PDP-14 system. He first determines the control operations he requires, by referring to equipment sequence charts, and deciding conditions to be sensed (inputs) and energy to be switched (outputs). Then, in a series of computer-assisted steps, described in Chapters 4 through 10, the operations are converted into a list of coded instructions termed a program. This program is then punched into a paper tape and sent to DEC, where it is used to control an automatic weaving machine which makes a wire braid. When the braid is mounted in the ROM, and the system started, the CU will test inputs and set outputs according to the program instructions. Note that the entire control system design effort is either on paper or in a computer. This process is equivalent to designing a relay control network, including layout of all the terminals, relays, wires, and other hardware.



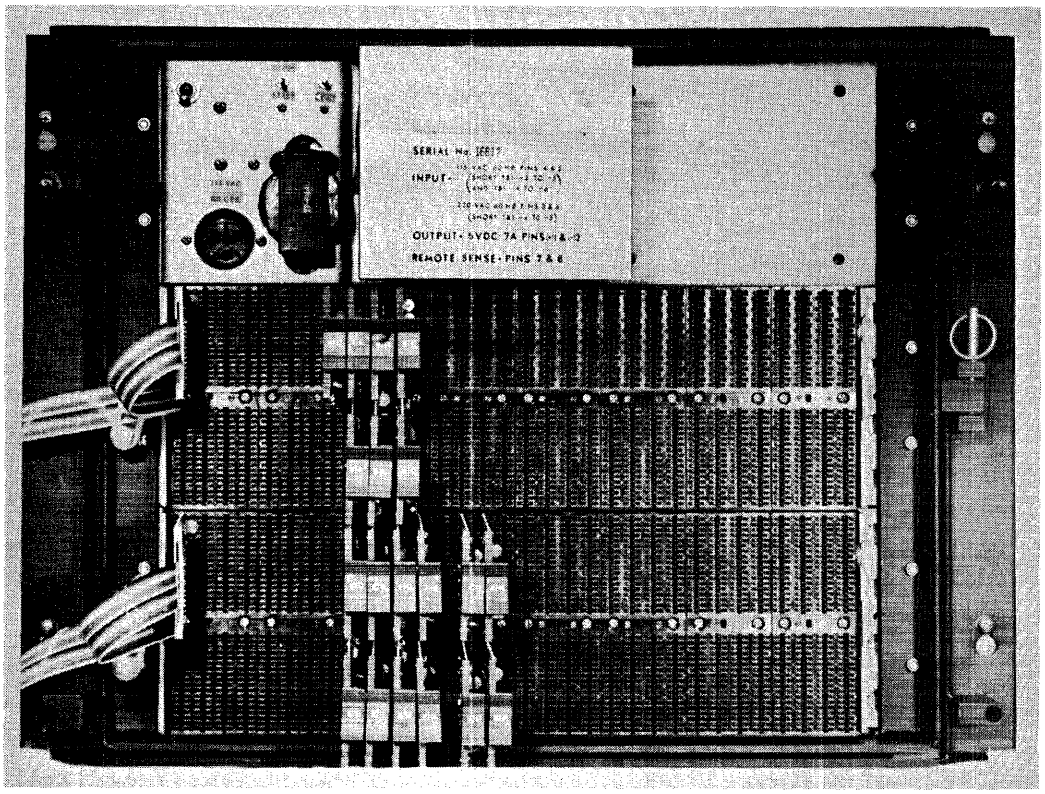
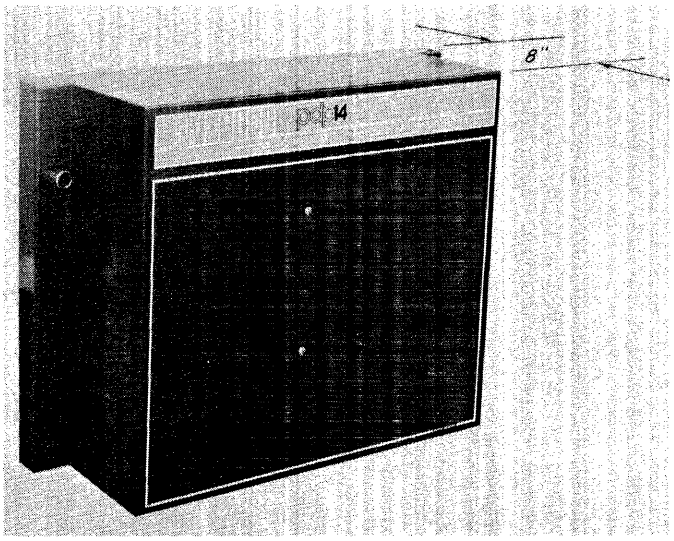


Figure 2-4 Control Unit-Pictorial

The ROM program can be changed at any time by repeating the program design steps, preparing a new braid, and installing it. This is equivalent to rewiring a complete control system panel. Minor changes to the program can also be made manually, if desired, by changing braid wires.

Control Unit Mechanical (Figure 2-5)



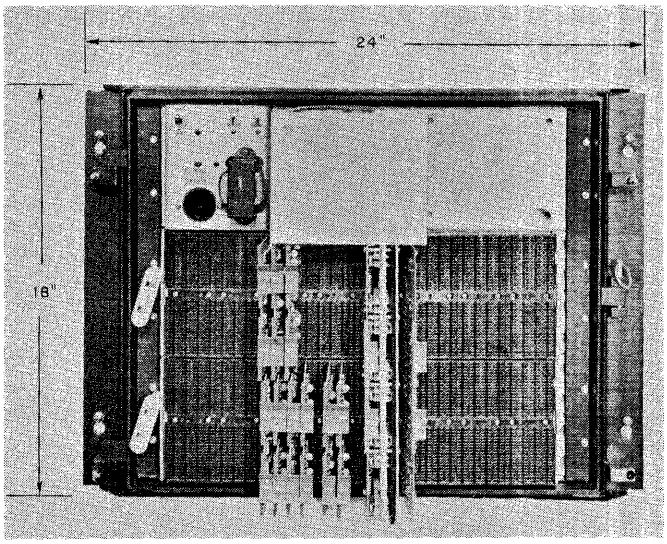


Figure 2-5 Control Unit Dimensions

The CU consists of a heavy steel mainframe containing low-voltage power supplies and module sockets, hinged to a base frame for mounting. Plug-in modules containing all control circuits, ROM assemblies and control cables, are inserted in this mainframe, and a sheet metal shield cover is fastened over the entire assembly.

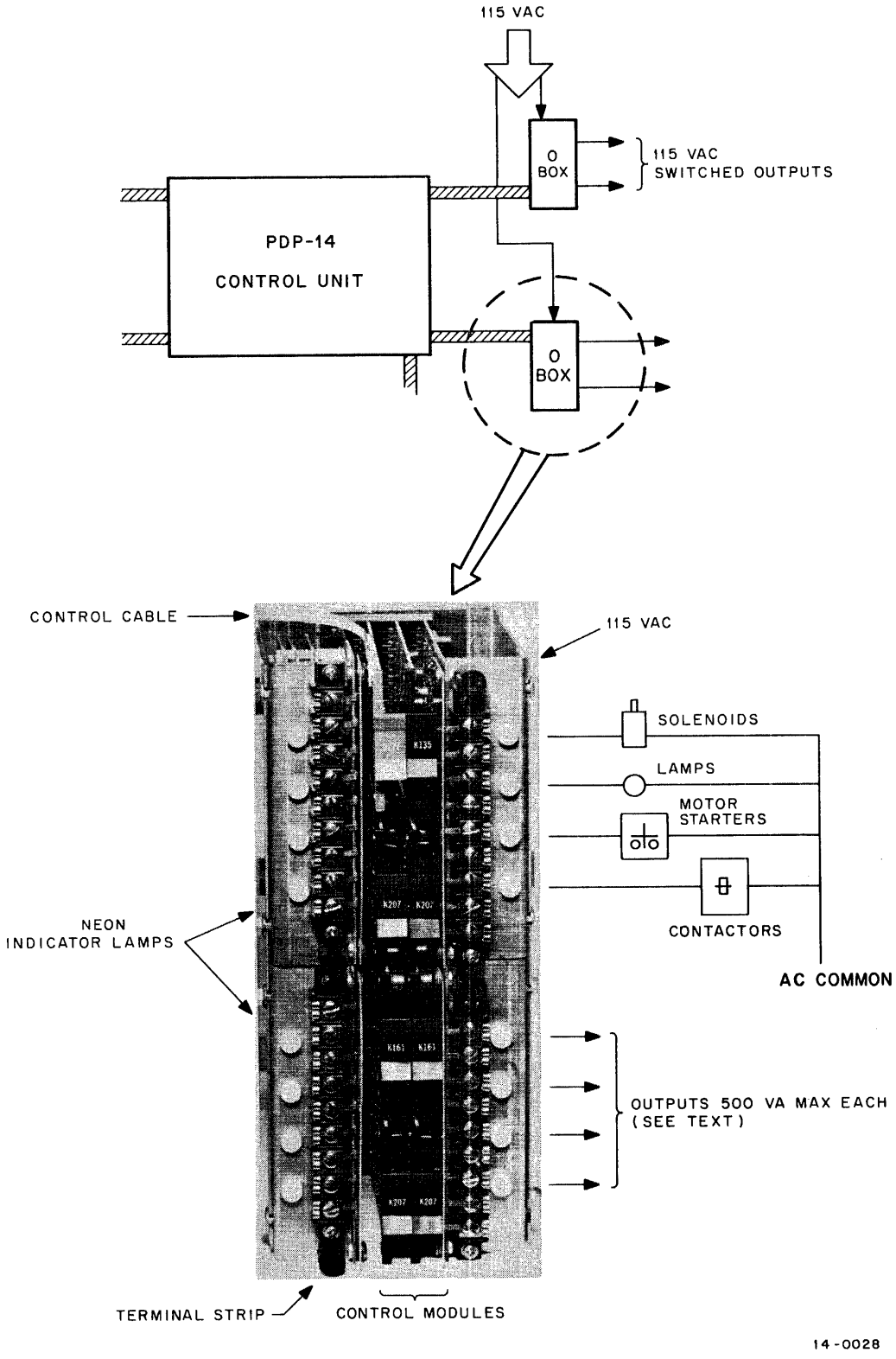


Figure 2-6 Output Box-Pictorial

O Box Function (Figure 2-6)

The O box contains isolated 115 Vac switches directed by the CU. AC power can be turned on or off, and the state sensed by the CU (as though they were inputs). These switches drive such output devices as indicators, motor starters, contactors, and any other 115 Vac load which does not exceed 500 VA (volt-amperes) per circuit. (See Chapter 3 for further specifications on output loads.) A lamp on each output switch shows when ac power is actually being supplied to a terminal strip. Each O box contains 16 independent output circuits. Command signals pass from the CU along control cables to each O box. The CU will accept as many as 16 O box cables for a total 256 outputs per system (actually, the last output is special-purpose, and is not normally used.)

O Box Mechanical (Figure 2-7)

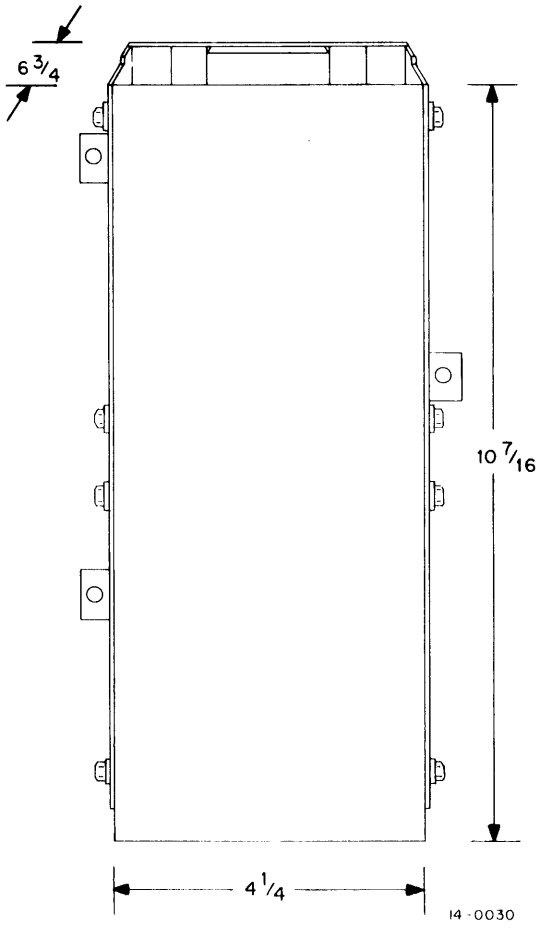


Figure 2-7 Output Box Dimensions

The O box shell is identical to the I box. It differs in the wiring of the rear module sockets, and in the modules used. The fronts of both I and O boxes are covered with a plastic safety shield.

Auxiliary Boxes (A and S) (Figure 2-8)

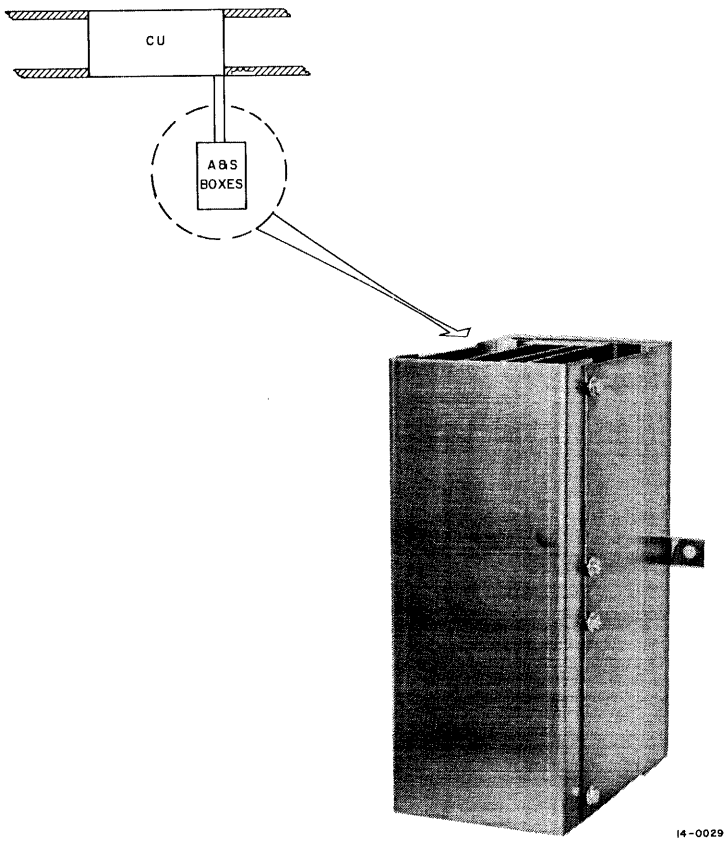


Figure 2-8 Auxiliary Boxes - Pictorial

A control system may sometimes require a group of pre-set status signals which cannot be generated by direct machine operation. For example, timers can provide duration and sequencing control for a controlled device. These functions are provided in the PDP-14 system by:

> Accessory (A) boxes, containing electronic timers with a range of 0.01 to 30 seconds (and special modifications to 4 minutes), retentive memories, which store results of selected control operations.
>
> Storage (S) boxes, providing "dummy outputs" which temporarily store the interim results of selected control operations for status indications. The S box is a temporary memory.

Each of these boxes appears to the CU electrically as an O-box, and requires some of their control cable slots. An A box contains up to 16 electronic timers, one for each output circuit, or up to four retentive memories. Using program techniques, any standard type of timer operation may be produced. The S box is available in half-size or full-size versions. The half-size box is equivalent to one O box (16 circuits). The full-size is electrically a double O box (32 circuits), although it is physically no larger than a standard box.

Auxiliary Box Mechanical (Figure 2-9)

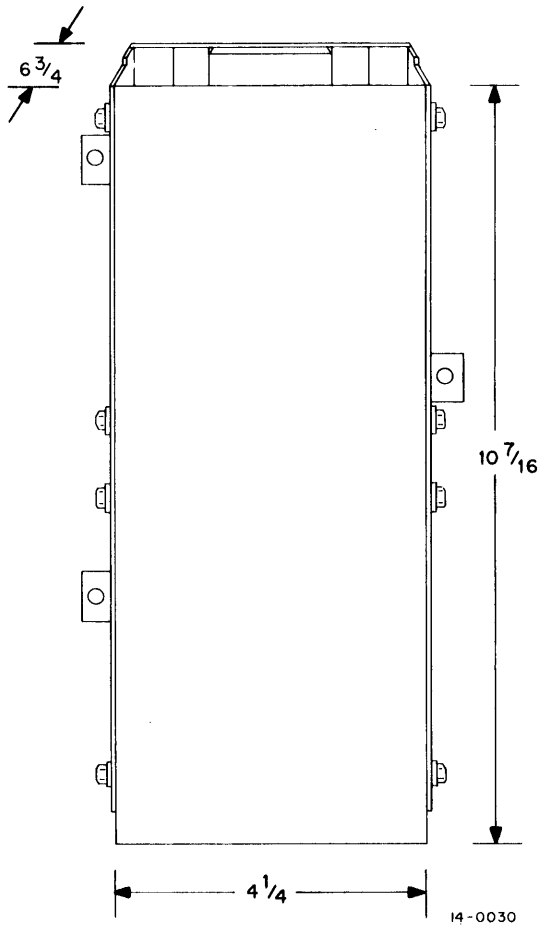


Figure 2-9 Auxiliary Box Dimensions

Both the A and S box shells are mechanically identical to the I or O box, and are mounted in the same way. Each has a complement of modules which allow it to function in its special way. The full S box has several more modules than the half box. Neither S nor A boxes has modules which contain terminal strips, since neither is intended to be connected to input or output circuits directly, but only through operation of the ROM program.

COMPUTER MONITORING

The PDP-14 system design allows for optional computer monitoring to be added at any time. A small, general-purpose computer is connected through control cables to the CU, and certain monitoring modules are also installed. The computer can then monitor any control operation of the PDP-14 system, or the status of any input or output and transfer this information into a printed record if desired. The computer can also assume command of any control functions, either on a standard or emergency basis.

Some of the best suited computers for monitoring with the PDP-14 are the DEC PDP-8/I, or PDP-8/L. They are convenient to use and install, and inexpensive. The monitoring modules to adapt these computers are available for the PDP-14.

SUMMARY OF PDP-14 SYSTEM CONTROL CIRCUITS (One System)

INPUTS	Eight I-boxes of 32 inputs each, total of 256 inputs. All inputs 115 Vac, 1.5 VA
OUTPUTS	16 O-boxes with 16 outputs each, total of 255 outputs (one output unusable). All outputs 115 Vac, 500 VA max., or total distributed load of 250 VA each. (See Chapter 3 for details.)
CONTROL UNIT	Accepts memory containing over 4000 separate instructions. Requires 115 Vac, less than 160 VA.

CHAPTER 3
PDP-14 SYSTEM PARTS AND FUNCTION

INPUT BOX OPERATION

The I box has two general functions:

a. signal conditioning and isolating, and
b. input selection for testing

As a signal conditioner, the I box reduces each 115 Vac input signal to a low, controlled voltage, and also isolates and filters it. The following is a simplified presentation of one input circuit (Figure 3-1):

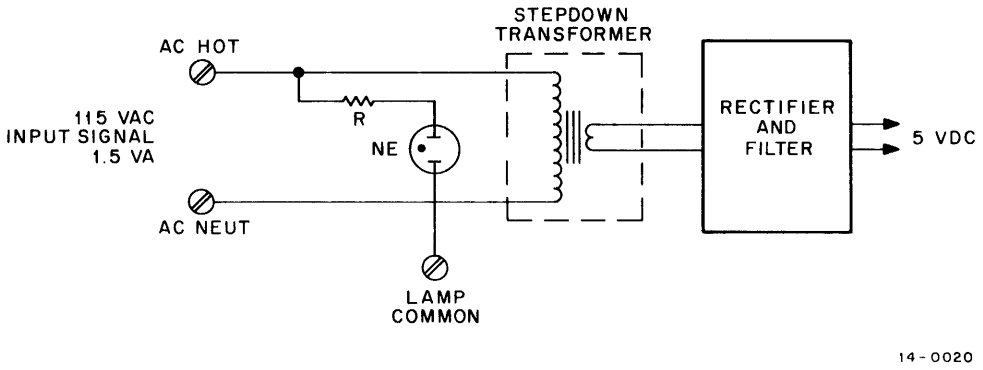


Figure 3-1 Box Input Circuit - Simplified

Each input circuit loads its 115 Vac signal to 1.5 VA, with a slight inductive surge to cause mild arcing at sensor contacts. This provides a self-cleaning function without excessive contact wear. In some cases, the switch could wear out mechanically before its contacts become unreliable.

A small neon lamp on each input line lights to show that the circuit is energized. The stepdown transformer (which provides the inductive surge) is effective isolation between input signals and the PDP-14 electronic circuits. The reduced voltage is rectified and filtered to remove electrical noise and a 5 Vdc signal is available for testing whenever an input is energized.

Eight input circuits are provided on each K578 input converter module. Four of these plug-in modules (with their screw terminal strips) are inserted in the left and right sides of an I box, for a total of 32 circuits per box. The remaining I box modules provide for the selection of one input from the 32 available. Figure 3-2 shows the general selection system for one I box.

A SELECTION CODE is sent from the Control Unit to the I box along the control cable. The PACKAGE SELECT signal is used by the Control Unit to select one I box for testing from the eight possible boxes in a complete system. The K135 GATE module extends PACKAGE SELECT to allow two K161 DECODERS to respond to the SELECTION CODE. Part of this code selects one of the K161 modules to operate for this particular input test. The rest of the code is used by the active K161 to select one of the eight inputs on two K578 INPUT CONVERTERS. This means, that for each test, two inputs have been selected by the K161, one on each of two K578 modules. Each tested input provides a SAMPLE RETURN signal. In the last step of input selection, the Control Unit checks one of the two SAMPLE RETURN lines for a signal, effectively selecting one of the 32 box inputs. An active AC input provides a 5 Vdc signal on the SAMPLE RETURN, and the absence of an input provides zero voltage. All control cable voltages are 5 Vdc with limited current for complete safety.

A summary of the selection operation (Figure 3-2).

PACKAGE SELECT	selects one box from eight
SELECTION CODE	selects one of two K161 DECODERS and (through K161) selects one of eight inputs on each of two K578 modules.

The Control Unit selects one of two SAMPLE RETURN signals for testing. This results in a selection of one input from a total of 256.

CONTROL UNIT OPERATION

The Control Unit (CU) is the most complex part of the PDP-14 system. Even so, it is much less complex than a conventional small computer. The CU and the Read Only Memory (ROM) interact in what might be called "electronic catch." CU begins the ball game by demanding that the ROM "pitch" one instruction. The ROM responds, and the instruction is "caught" by the CU, which must now "run the play." This instruction determines the operation of the CU and also in some cases, the next "pitch" the CU will demand from the ROM. The CU controls the ROM and the ROM controls the function of the CU. However, in this case, no referee can call an arbitrary "time-out."

The CU functions to:

a. Interrogate the ROM for an operating instruction
b. Generate signals that address or select specific inputs or outputs

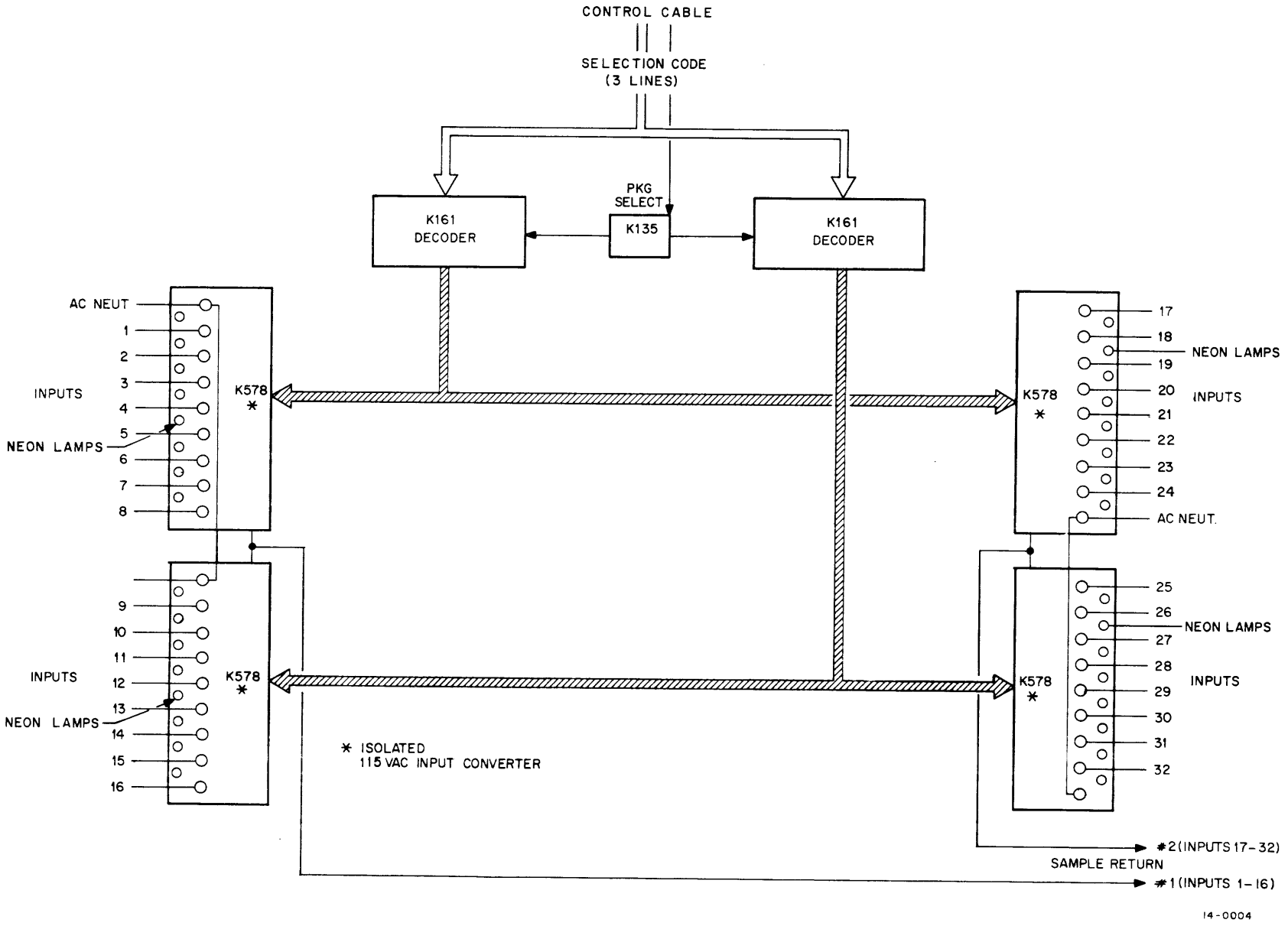


Figure 3-2 I Box Selection System

c. Test a selected input or output to determine whether it is on or off

d. Combine the results of one or more test steps

e. Turn an output on or off depending on the combined results of the testing operations

f. Interrogate the ROM for a new operating instruction.

In these operations, the ROM is functionally an electronic list of:

a. Inputs and outputs to be tested

b. Alternate commands to be followed depending on the results of the testing

c. Outputs to be turned on or off; and

d. Sometimes a special group of monitoring commands.

The ROM thus directs the CU circuits to perform the three basic operations of a control system:

Test (of inputs and outputs)
Decision
Execution (of decision)

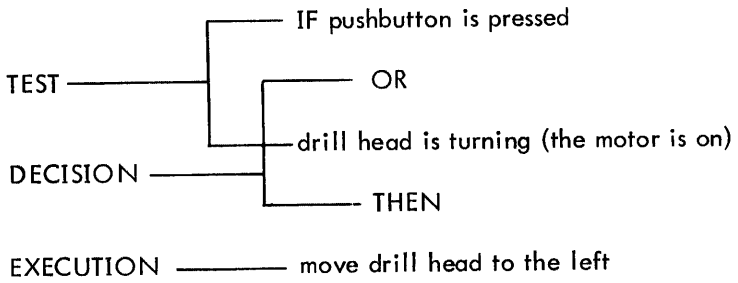
When a computer is connected to the CU for monitoring, it can functionally replace the ROM. The computer can request the condition of an input or the result of testing, or it can actually control the operation of selected outputs. In another monitoring approach, the ROM can include instructions which force the CU to send status information to a waiting computer. The computer doesn't need to make a request. The active element which temporarily stores the results of several test operations and allows alternate output control instructions to be performed is called the TEST FLAG.

Boolean Logic

The PDP-14 control system operates by a formal binary logic termed Boolean. This logic is a simple way of expressing binary equipment operation, as in this example:

> IF a pushbutton is pressed OR the drill head is turning, THEN move the drill head to the left. The operator of the machine wants to be able to move the drill head left at any point in its automatic cycle by pressing a button. In any case, the drill head should automatically move left whenever it is turning.

This simple Boolean statement has three functional parts; corresponding to the operation of any control system:



We will arbitrarily assign address numbers to the input and outputs involved.

The pushbutton is input 36

The drill head motor contactor is output 7, and the hydraulic solenoid which causes a ram to force the drill head slide to the left is output 142.

The Boolean statement is now:

IF input 36 is ON

OR

output 7 is ON

THEN

turn output 142 ON.

The statement is one of the control sequences stored in the ROM. Let's follow this step-by-step sequence through the PDP-14 system.

1. The CU requests its first instruction from the ROM. It receives the electronic command; TEST INPUT 36.

2. The CU addresses input 36 by sending the proper address code along the control cables to I box A. (Input 36 is input number 31 in I box A.) The proper SAMPLE RETURN is sent to the TEST flag circuit. The TEST flag begins the sequence in the off condition and turns on if the input tested is on.

3. The CU requests an instruction from the ROM to TEST OUTPUT 7.

4. The CU requests a SAMPLE RETURN from output 7 (in O box A) just as it did for input 36 in step 2. (In this case, no output command signals are sent. The TEST flag is turned on if output 7 is on (it may have already been turned on by input 36). If output 7 is off, the TEST flag is left unchanged.

5. The CU requests more instructions from the ROM. These instructions will tell it how to respond to the results of the testing operation (to the TEST flag). There are two basic ROM instruction, one for each possible state of the TEST flag.

At this point, if input 36 OR output 7 is on, the TEST flag would be on also, and the ROM would direct the CU to SET OUTPUT 142 ON. (The ROM would probably also contain the alternate instruction SET OUTPUT 142 OFF, so that when neither proper condition existed, the output is to be turned off.)

6. The CU now selects output 142 through the control cable for O box G. (Output 142 is output number 3 in O box G.) It also sends the ENABLE SET signal, which will turn on output 142 when it is addressed.

7. The CU now requests an instruction from the ROM for the next control sequence.

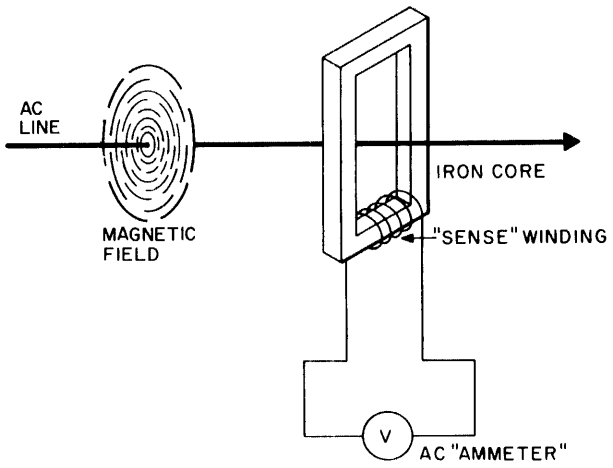
This control sequence example may seem at first to be overly complicated but these are the same logical steps you would perform if you were to control the equipment manually. This sequence can be completed in less than 100 millionths of a second.

Read-Only Memory (ROM) Operation

The Read-Only Memory (ROM) contains all PDP-14 system instructions in its pattern of wires. These wires are formed into a compact, replaceable package called a "braid", and then encapsulated to make them immune to accidental change. By replacing wires manually, all instructions can be altered to make revisions in control

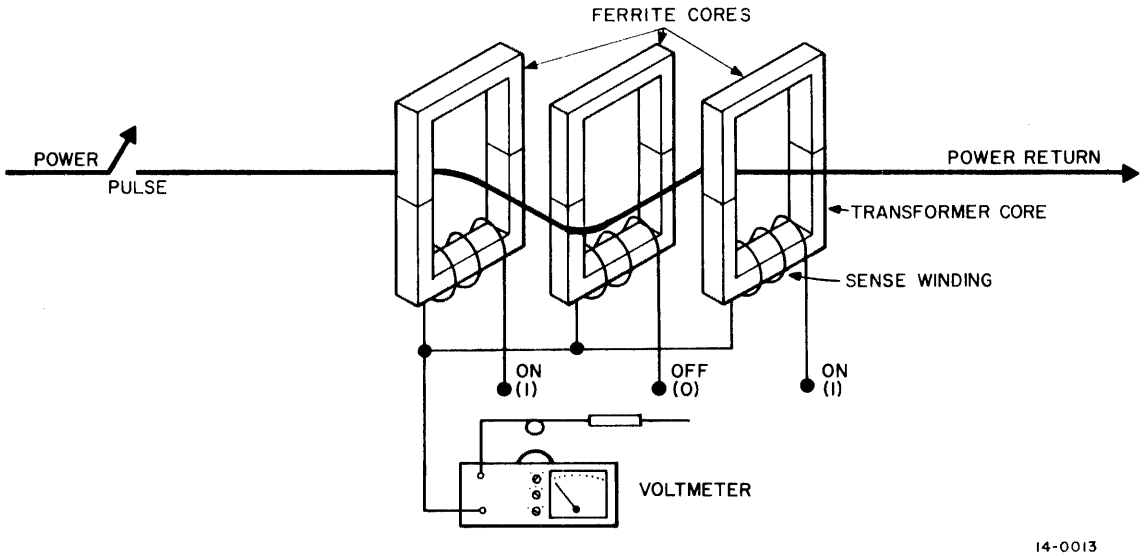
operation. The manual procedure is ordinarily used only to make minor changes involving a few wires.

The operation of the ROM depends on the same principle as the familiar current transformer used to monitor AC power leads.



Fluctuations in the AC current flow produce a varying magnetic field around the wire. This field is induced in an iron core around the wire, producing a small voltage in a sense winding. This voltage is measured by a meter to give an indirect measurement of the current in the AC lead. The lead is equivalent to a one-half turn transformer primary, and the sense winding to the secondary.

The following illustration shows three of the 12 transformers in a group



ROM circuit operation is similar to the current transformer, except that:

a. Many wires (128) are used
b. The current is checked only to be present or absent on a wire, not for actual value
c. Groups of twelve transformers are used, rather than just one (a total of 96 transformers)
d. Circuits are added to select the proper wire and to boost the sensed voltage to operate CU circuits.

To "read" the instruction wire, a pulse of current is applied to it. If a quick-acting voltmeter were applied, the left and right transformers would show voltage present at the sense winding and the middle transformer would show none. Remember that all signals used are binary, i.e., they have only two possible conditions (on or off). The presence of a voltage is defined as a data "1" and its absence as a "0" in the PDP-14 system; thus, the transformer outputs can be given as a code (in this case, "101"). Since a group of 12 transformers form a single instruction, a typical instruction code might be:

101 111 001 110

Each number of the code is termed a "bit". The 12-bit instruction is divided into four groups of three bits each, for easier reading. Each of these four groups can be expressed as an octal number (discussed in the programming sections). (For example, the binary instruction code shown above has an octal equivalent of 5716.)

The ROM is a self-contained package, consisting of three circuit boards (Figure 3-3):

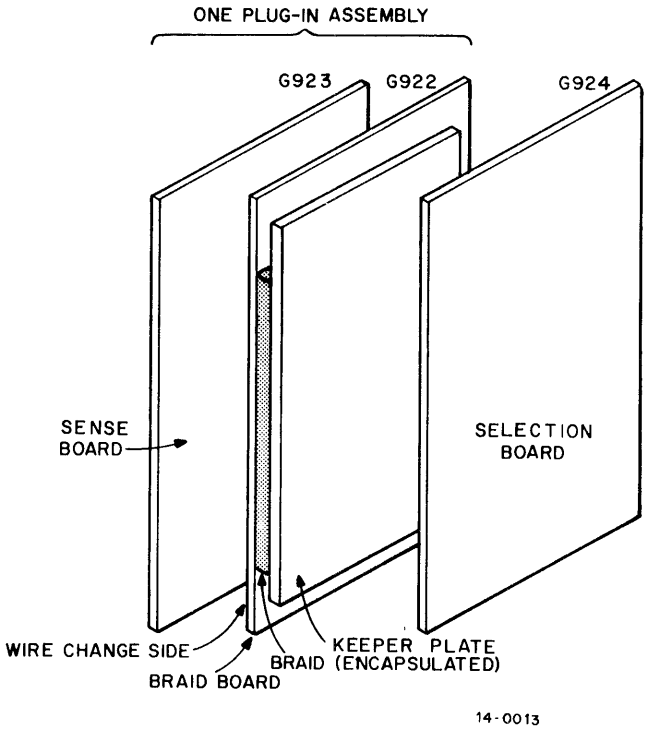


Figure 3-3 ROM Board Group-Pictorial

The boards correspond to the three general operating steps of the ROM:

a. SELECTION - The PDP-14 Control Unit provides the ROM an address code, telling the ROM which wire to select (and which group of 12 transformers).

b. READ - the Control Unit now signals the ROM to begin, and the ROM responds by placing a current pulse on the selected wire in the braid.

c. SENSE - The ROM circuits sense the 12 transformer voltage outputs, boost them to proper values, and transfer them to the Control Unit for use as operating instructions.

The Control Unit alone interprets and responds to instruction codes provided from the ROM. In this sense, the ROM does not know its own contents, and has no way of reacting to its own output codes.

Changing Instruction Codes

The position of the wires in the braid determines the instruction code which is generated in ROM operation. If a wire is placed inside a core, it produces a 1, if outside, a 0. Therefore, wires can be threaded by hand around and through the ROM transformers to produce any instruction code desired. This feature is useful to implement minor changes in control operation. The process of modifying instructions is equivalent to rewiring the control system, and is much easier and faster. The technique of ROM wire additions is discussed in Chapter 12.

The group of three circuit boards is termed a memory bank. Up to four of these banks may be added to a CU, as options, to increase the memory locations available for a control program. Each bank will contain its own braid and independent selection and sensing circuits for over 1000 locations. Thus, each memory bank is said to contain 1K units of memory. The CU can accept four memory banks, or 4K of memory. When the CU is operating, the individual memory banks are all used as one large memory in sequential program operation.

O BOX OPERATION (Figure 3-4)

The O box functions much like an I box, but with these general operations:

a. output selection (and testing),
b. output function command, and
c. isolated ac output switching.

The output selection system is nearly identical to input selection already discussed, except that only one of sixteen circuits must be selected instead of one of 32. Figure 3-4 indicates the selection sequence. PKG SELECT and part of the SELECTION CODE cause the K135 to activate the particular O box and select one of the two K161 DECODERS. These DECODERS then accept the remainder of the SELECTION CODE to select one output control circuit from eight contained on two K207 STORAGE REGISTERS. Each K207 contains four output control circuits, one for each ac output on a K614 ISOLATED AC SWITCH. At the same time the Control Unit sends its SELECTION CODE, it can also issue one of three function commands, ENABLE SET, ENABLE RESET and CLEAR.

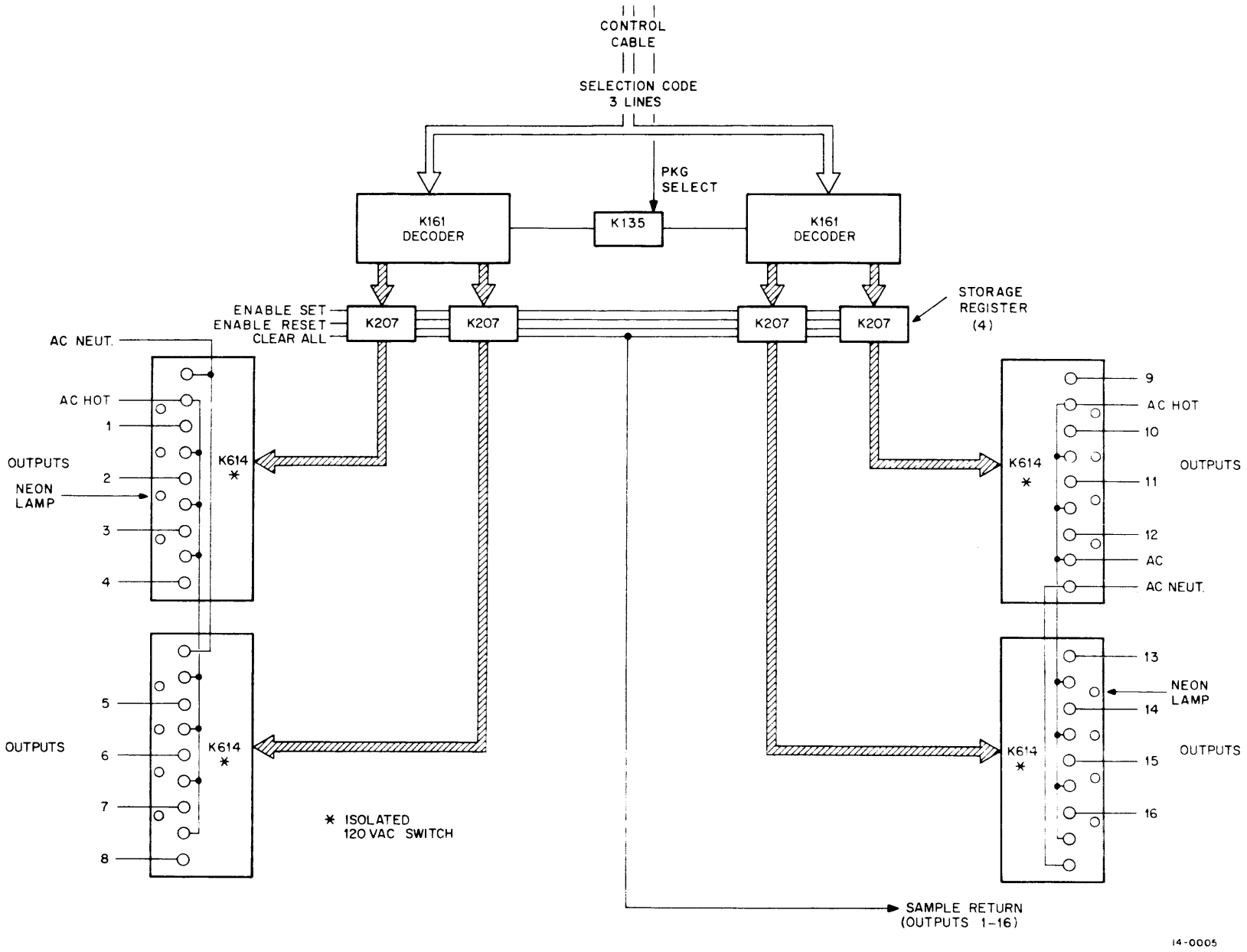


Figure 3-4 O Box Selection System

ENABLE SET will turn on a K207 control circuit corresponding to the selected output. ENABLE RESET will turn the circuit off. The CLEAR signal will turn every output circuit off regardless of selection.

The K207 output control circuits receive the CU commands which establish the condition of each output, and then maintain that condition until the CU changes it. In other words, the K207 functions as a memory device. When the PDP-14 system is stopped and then restarted, all K207 output circuits are turned OFF and wait further CU action, just as if the CLEAR command had been issued.

Whenever the SELECTION CODE has selected an output, a SAMPLE RETURN indicates the condition of the selected K207 output circuit to the Control Unit. This would occur even if a K614 ISOLATED SWITCH module were removed from the O box, because the SAMPLE RETURN indicates the status of the K207 holding circuits and not ac power conditions. If an output circuit is to be tested for status only and not changed, no command signals are sent and the selection sequence produces only a SAMPLE RETURN. If a function command was also issued, the SAMPLE RETURN shows the condition after the command has been executed.

The K207 output circuits directly control the action of the K614 115 Vac ISOLATED SWITCH modules. The four circuits in one K207 serve the four switch outputs of one K614 module.

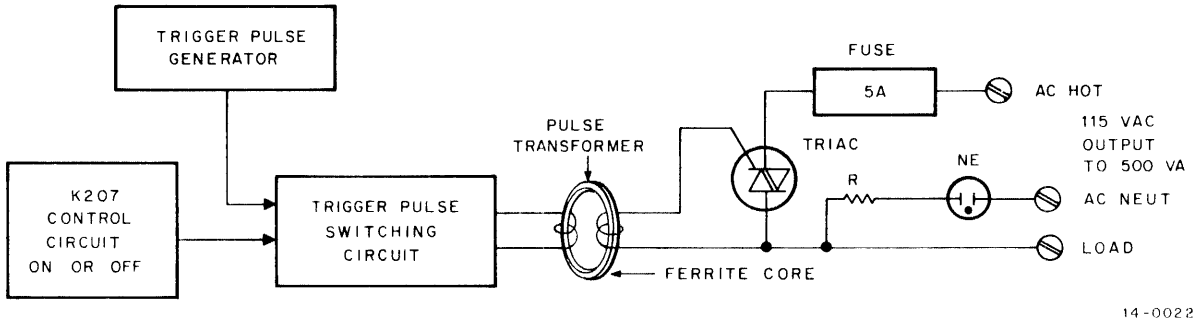


Figure 3-5 Box Output Circuit - Simplified

The active switching element is a triac, a solid-state AC control device. (The triac belongs to the family of thyristors, discussed in most modern circuit handbooks.) The triac requires a series of low-voltage pulses to turn the ac power on; otherwise it acts as a very high resistance. In other words, it is a high-voltage switch operated by a low-voltage control signal.

The K207 control circuit is first directed ON or OFF by the selection and control sequence. It now holds either condition indefinitely. When it is ON, it allows pulses from a trigger oscillator on the K614 module to enter the primary of a small pulse transformer formed of two isolated lengths of wire looped through ferrite cores. The pulses from the transformer secondary activate the triac, switching on the 115 Vac current. The transformer provides effective isolation of the low-voltage control circuits from the ac power supply.

A 5-ampere replaceable pigtail fuse is provided directly on the K614 module to protect the triac from damaging overcurrents. A neon lamp indicates when the triac is energized and ac power is being provided. This allows easy checking for actual power switching operation.

Each of the four triacs on a K614 module is mounted on its own heat sink to dissipate the slight resistance heating which is typically less than 2% of the total ac circuit load. An individual triac circuit is rated at 500 VA when operated alone, and will withstand moderate inductive surges for brief periods. However, the temperature rise caused by the collective operation of the triacs on the K614 limits the total power rating of the module to 1000 VA.

Thus, the total rating per O box is 4000 VA. Four examples of maximum loading for one K614 are as follows:

Output	Example A	Example B	Example C	Example D
1	500 VA	250 VA	400 VA	300 VA
2	500 VA	250 VA	200 VA	100 VA
3	no load	250 VA	400 VA	300 VA
4	no load	250 VA	no load	300 VA

No particular distribution must be followed, as long as no single ac circuit is loaded more than 500 VA and no single K614 module is loaded more than 1000 VA. The maximum evenly distributed load of 250 VA per circuit is more than adequate for most industrial control switching. If higher loads are required, distribute them between several K614 modules, or use a power contactor.

Each ac circuit is completely independent from every other, even on the same K614 module, and can be supplied from separate ac power sources, if necessary. Ordinarily, a common source of ac is strapped to all circuit hot terminals, as shown in Chapter 11. It is not recommended that outputs be paralleled to form higher current ratings. For 230 Vac operation, two triac circuits can be connected in series. This special arrangement and other specifications are contained in the DEC publication, "The Control Handbook", Catalog C110. Copies are available on request.

AUXILIARY BOX OPERATION

The A and S box shells are physically identical to those of the I and O boxes. Since these boxes require no ac circuits or terminal strips, the entire space may be used for electronic modules. This makes possible an S box which contains double the usual quantity of output-type circuits. The auxiliary box modules are protected by a metal cover which fastens in place of the four absent K614 ac switch modules (Figure 3-6).

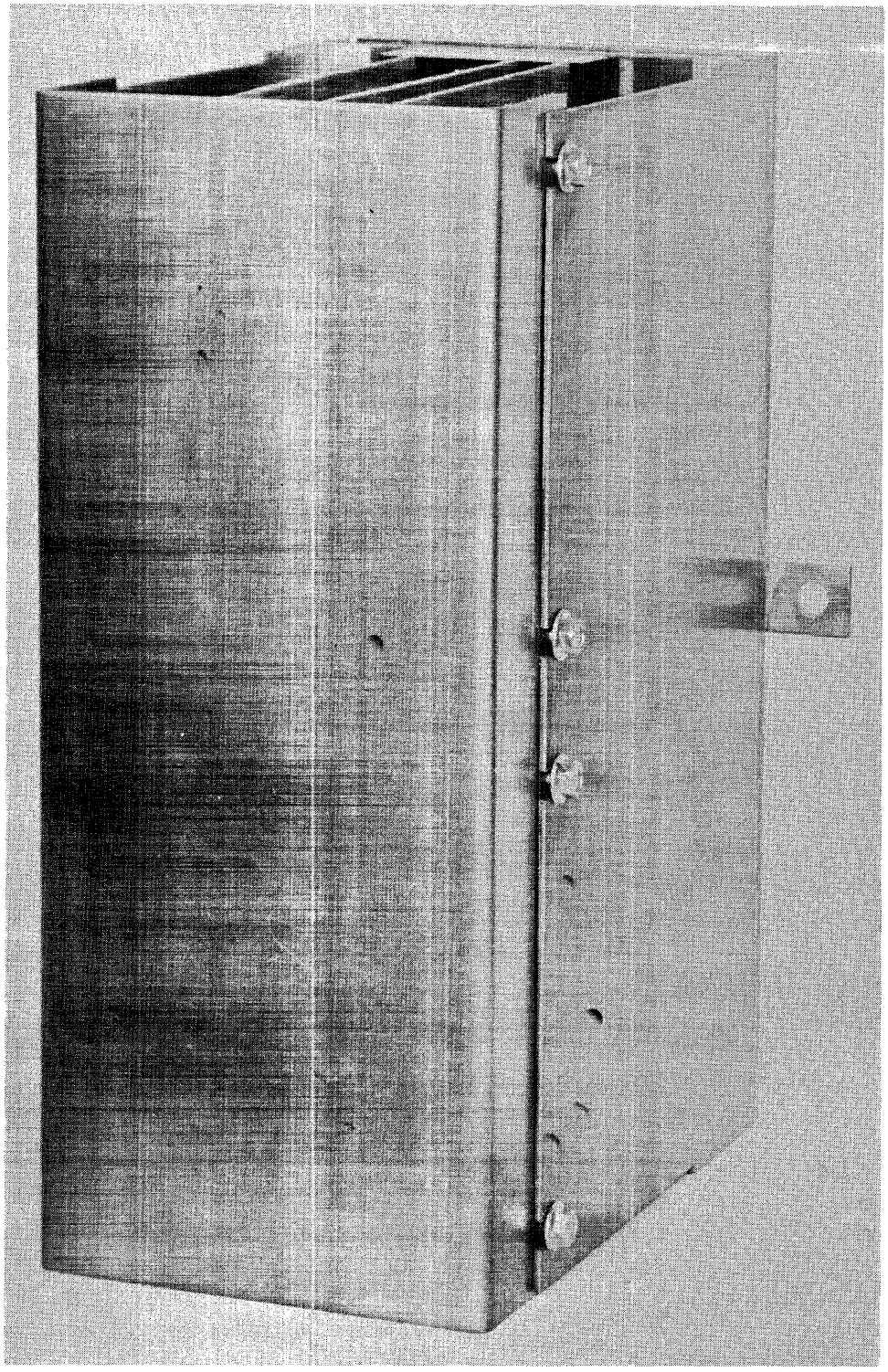


Figure 3-6 A or S Box

Each auxiliary box functions in SET, RESET, and TEST operations exactly as an O box, and thus contains groups of:

K135 GATES
K161 DECODERS
K207 STORAGE REGISTERS

The meaning of the SET and RESET commands is similar to the O box; they activate or deactivate the selected output circuit. The TEST command produces a SAMPLE RETURN which indicates the operation of the special auxiliary device in the selected circuit.

The A box may contain either timer or retentive memory modules, up to a total of 16 circuits. The K302 timer modules each contain two timer circuits, and can be inserted into any of the eight sockets along the left or right edge of the box (Figure 3-7).

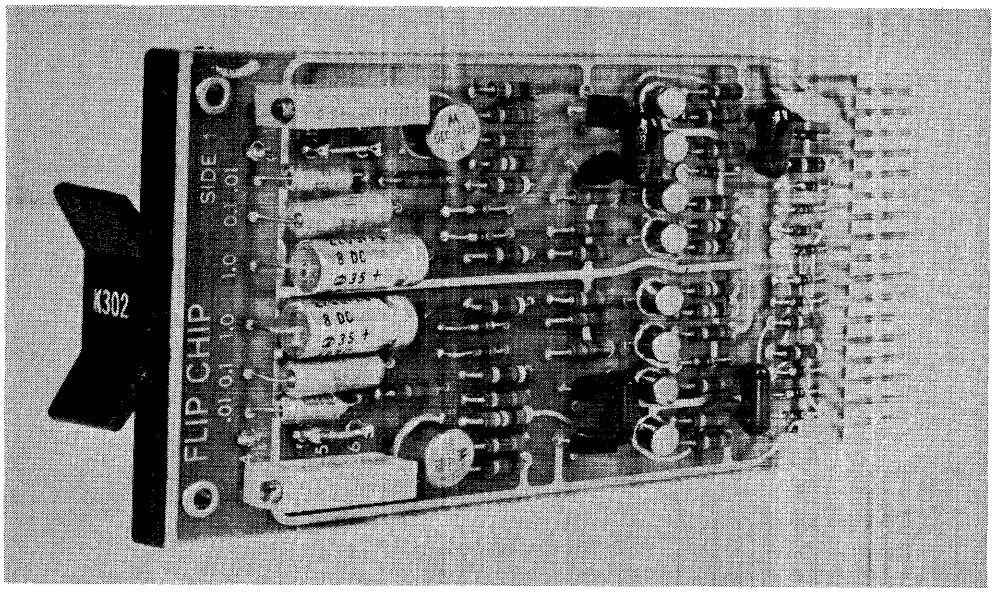


Figure 3-7 Timer Module K302

Each K302 circuit is separately adjusted for the desired timeout in two steps. The proper range is permanently selected by cutting two components from the module, providing either:

0.01 to 0.3 seconds
0.1 to 3.0 seconds
1.0 to 3.0 seconds
Up to 4 minutes by special modification (see Chapter 11).

Timing within each range is adjusted by a potentiometer, which can be turned with a small screwdriver when the module is installed. Timing error due to temperature variations is typically less than ± 1% for a 5°C (9°F) change. Repeatability is better than 95%, allowing a minimum recovery time of 0.3% of the maximum time available in the range.

The K272 retentive memory modules use a magnetically-latching mercury relay, which must be mounted in one position to keep the mercury where it belongs inside the sealed unit. Therefore, K272 modules, containing one output circuit each, are mounted only along the right side of the A box. Thus the 16 output circuits of this box may contain: Eight K302 timer modules (2 circuits) = 16 timers, or four K302 timer modules (2 circuits) = 8 timers, and four K272 retentive memory modules (1 circuit) = 4 retentive memories.

They may also contain reduced amounts of K302 or K272 modules, but never more than four K272 modules per box (Figure 3-8).

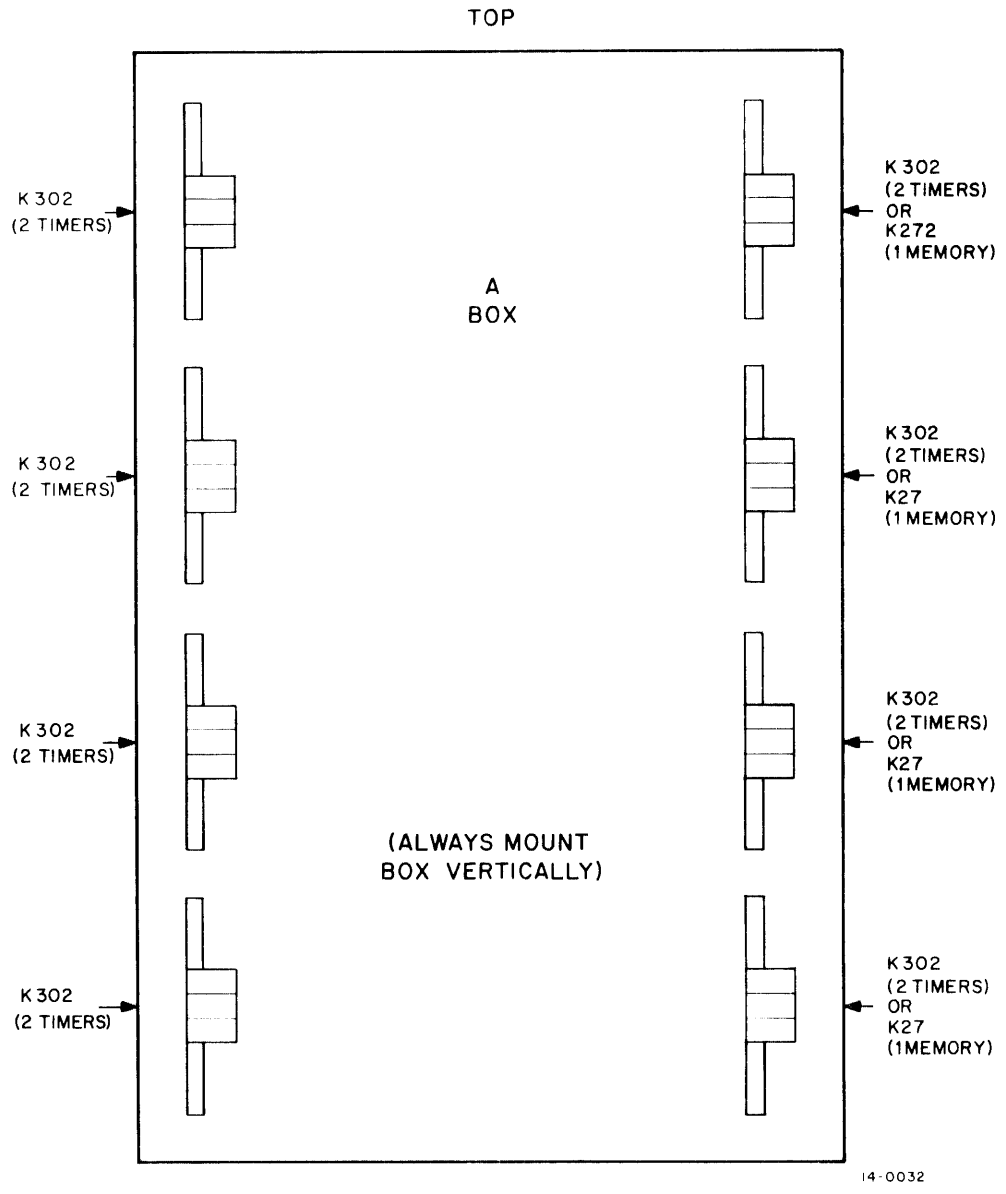


Figure 3-8 A Box Socket Functions and Allocations

As in all output circuits, a timer is normally off. A timing cycle is begun by a SET command from the CU. At the end of the timeout, the timer output turns on (indicated by an active SAMPLE RETURN). The output then remains on until a RESET command turns it off. In other words, the timer is a delayed-on type, although all types of timing cycles can be simulated by program variations.

Unlike a timer, a retentive memory circuit is not turned off by the initial startup of the PDP-14 system. The memory (relay) can be turned on or off only by a direct SET or RESET command addressed to that circuit. The state of the memory is available as a SAMPLE RETURN, just as an ordinary output circuit. The individual memory circuit may also be manually reset by a toggle switch on the module (Figure 3-9).

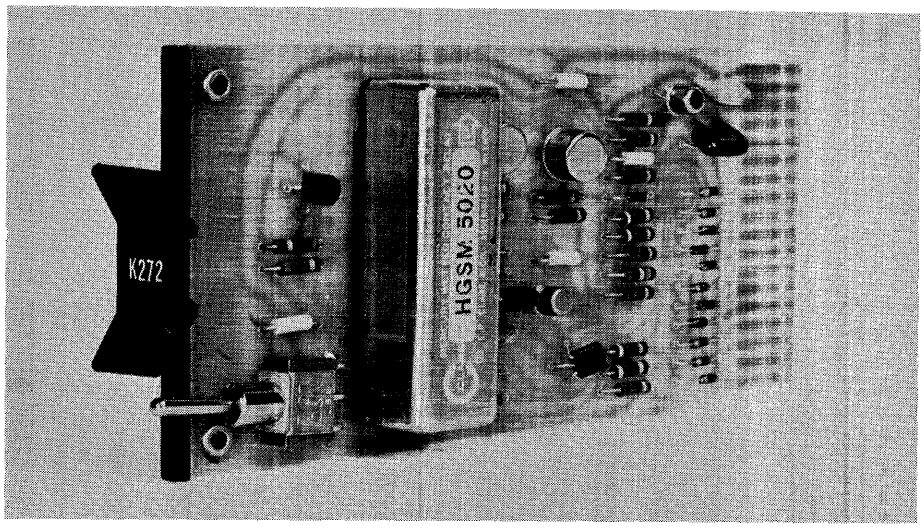


Figure 3-9 Retentive Memory Module K272

The S box, or "storage" box, contains groups of K207 output holding circuits, exactly as an O box, but without the K614 ac switch modules. The half S box contains the same number of circuits as a regular O box (sixteen), and uses one control cable. The full S box actually contains two entirely independent groups of sixteen circuits, the equivalent of two O box output circuits in one box shell. The 32 output circuits are addressed through two control cables, one at the top, the other at the bottom of the box shell.

The S box functions as a "dummy" O box - its addressing and control commands are exactly the same, but without actual ac power switching. The S box stores the outputs of intermediate testing operations which must be recalled for use later in the control operation; a sort of temporary function.

Additional Module Information

For further information about modules used in I and O boxes, consult the DEC publication, "The Control Handbook", Catalog C-110, available on request. The Handbook provides complete specifications for the K578 and K614 modules, and explains some special applications.

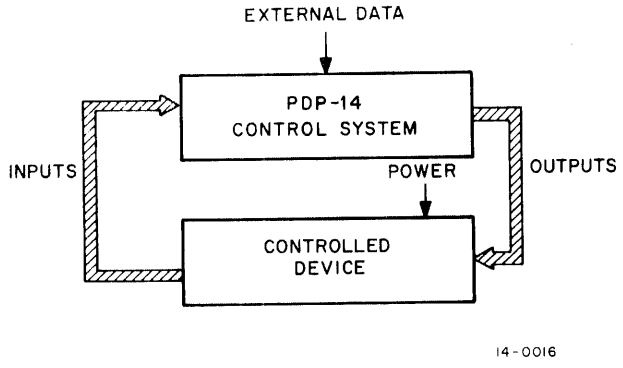
Computer Operations With the PDP-14 System

The PDP-14 system may operate in one of four general ways:

Control
Monitoring
Control and Monitoring
Shared Control

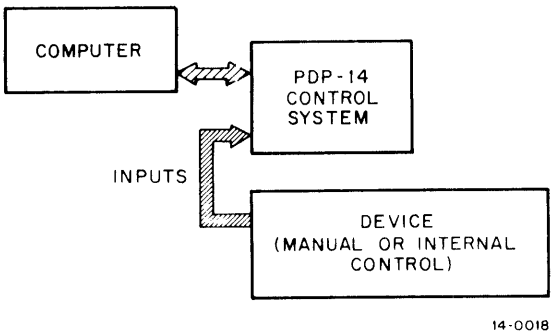
CONTROL

The control operation has been described without any monitoring function. It has been shown how the CU requests input and output samples and uses them to make a proper control decision. This same information may also be sent to a general-purpose computer for interpretation and processing in many ways. For example, the computer can prepare a complete status report of equipment operation, and signal slowdowns and other undesirable situations before they become major catastrophes. This points out the importance of preventive maintenance.



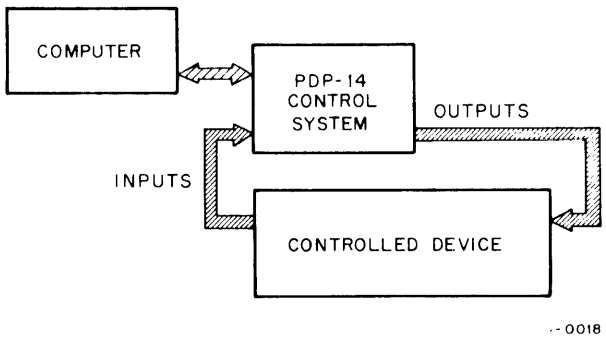
MONITORING

The PDP-14 can be used for monitoring only, if desired, with no control operations. In this mode, the ROM contains instructions for inputs and outputs to test, and the proper format to send the information to the computer. The system can also be run without an ROM; the computer controls all testing operations. That is, the PDP-14 functions as a giant "interface," providing ac coupling and addressing circuits for the computer. For extended monitoring, special options allow several PDP-14 systems to send data through long lines to a single waiting computer, where the data are assembled into a report for an entire network of equipment.



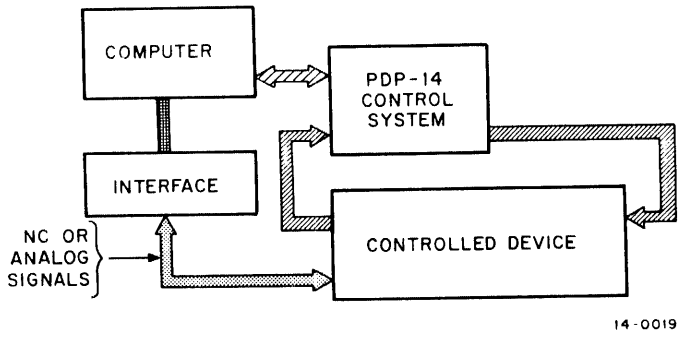
CONTROL AND MONITORING

When both monitoring and control are mixed in one installation, the PDP-14 system shows its most complete versatility. In this mode, the ROM contains the usual control instructions, including TEST and SET commands, as well as a group of special monitoring instructions. These instructions allow the CU to select specific TEST results for transfer to the computer. Part of the ROM program may be designed for emergency conditions only (to signal the computer operator that the controlled device needs special attention). In this situation, the computer may assume command of control functions until the emergency passes.



Shared Control

This approach, like control and monitoring, is a union of PDP-14 and computer. In this case, control or monitoring of analog processes or other difficult functions is handled directly by the computer. For example, our small computer can provide numeric control and also monitor a PDP-14 system supplying binary control to the same (or another) device. A computer could also provide (analog) quality-control measurements to an industrial operation being controlled by the PDP-14 system. Shared control, although requiring additional equipment, makes the most efficient use of both the PDP-14 and the computer.



Adding the Computer

The most convenient computer to use with the PDP-14 system is either the PDP-8/I or PDP-8/L. All the equipment to connect these computers is available, and additional programs and equipment for numberic control and analog operations can be quickly supplied. If you already have a general-purpose digital computer, a simple interface device may allow it to operate with the PDP-14. Contact an Applications Engineer, who can explain when it is feasible to design the proper interface.

To add any computer operations to the PDP-14 system, a small group of special modules is added to the CU, and three cables are inserted. These cables carry the data signals from the PDP-14 to the computer, and return computer control and interrogation signals. The modules allow these signals to be transmitted on command from either the computer or ROM program. The type of control or monitor operation performed, depends mostly on the program used, although some specialized applications could require a few additional modules in the CU. About 15 minutes is needed to convert a PDP-14 system used only for control to one for complete computer monitoring.

CHAPTER 4

INTRODUCTION TO PDP-14 PROGRAMMING

Anyone who has designed machine controls has actually been a programmer. Although he has used relays as his medium, he has still specified a series of actions (in a given sequence) which control machine operations. The specification of actions and their sequence is programming in a general sense. The basic difference between what is usually called programming and other forms of machine control is that the programmer uses instructions or a programming language in place of specifying the relays which control the various pneumatic, hydraulic, and electric devices.

PROGRAMMING THE PDP-14

The PDP-14 was designed specifically to control machines. The programmer need only specify the control function to be performed in a particular application. This is analogous to drafting a relay ladder diagram to specify the wiring of a relay panel. The difference is that the control designer is designing a PDP-14 memory instead of directing the actions of an electrician. Once the control function is specified, the PDP-14 will control a machine (or series of machines) just as relays would in the same application.

There are distinct advantages to programming machine control with a PDP-14 over programming with relays. Logic errors are easier to find and, once detected, easier to correct.

Programming in the PDP-14 system is simply the procedure used to generate the Read Only Memory (ROM) to control a process or machine. PDP-14 programming does not require previous computer experience although it does require experience in machine control.

PDP-14 programs provide relationships between inputs (limit switches, pushbuttons, selector switches, etc.) and outputs (solenoids, motor contactors, indicator lights, etc.). These relationships, or control functions, may be expressed as Boolean equations which, when solved for particular input values, specify the state (ON or OFF) of an output.

The following definitions are for the reader who has not been exposed to computer terminology.

Memory	the unit of the PDP-14 which contains (stores) the program.
Location	the smallest part of the memory, in which instructions are stored.

Instruction — the smallest part of a program. An instruction causes the PDP-14 to perform one specific operation, such as testing an input or setting an output. An instruction may be one-location (requires one memory location for storage) or two-location (requires two locations for storage). The actual instructions executed by the PDP-14 are binary numbers, but they are usually given symbolic names.

Operand — that which is operated upon by an instruction. For example, the operand of a test instruction is the specific input to be tested for ON or OFF. The operand of a set output instruction is the specific output which is set ON or OFF.

Program — a plan for solving a problem. A PDP-14 program is basically a series of instructions which tests inputs and sets outputs ON or OFF dependent upon the result of the tests. PDP-14 programs have three basic forms:

(1) Machine Code Program - a program which contains instructions to be executed by the PDP-14.

(2) PAL-14 Source Program - a program containing symbolic representations for instruction operands which must be translated into machine code before it can be executed by PDP-14.

(3) BOOL-14 Source Program - a program containing equations which must be translated into machine code before it can be executed by the PDP-14.

INPUT AND OUTPUT ORGANIZATION

Machine inputs and outputs must be assigned to the PDP-14 input (I) and output (O) boxes before a PDP-14 control program can be written. These assignments permit the PDP-14 instructions to test the state of specific inputs and outputs.

The technical descriptions of the I and O boxes are contained in Chapter 3. There are eight possible input boxes, lettered A through H. Each I box contains 32 inputs, thereby allowing a maximum of 256 PDP-14 program inputs. There are sixteen possible output boxes which are designated A through S (excluding I, O, and Q to prevent possible confusion). Each O box contains 16 output drivers. A total of 255 separate outputs are possible; the 256th output (the 16th output of O-box S) is reserved for other purposes and should not be used as a normal output.

PDP-14 Inputs

The inputs of the PDP-14 system are limit switches, selector switches, pushbuttons, etc. Each input has a value of either ON or OFF at any one time. These machine inputs are wired to an I-box terminal. An input is ON if there is 115 Vac present at the specific input terminal. The input is OFF to show the absence of 115 Vac. For programming purposes, all PDP-14 inputs are denoted by an "X" followed by an octal (base 8) number. This convention of identifying inputs with the letter "X" is used throughout the PDP-14 system. The number is determined by the selection of the I box and terminal to which the machine input is wired. For example, X15 is PDP-14 input 15. Specifically X15 is the PDP-14 representation of a machine input (e.g., a limit switch) which

PDP-14 INPUT ASSIGNMENT SHEET -P1

INPUT BOX B (A THROUGH H)

INPUT NUMBER	ASSIGNMENT	NORM COND	PROGRAM NUMBER
1	LS201 - Activated when slide 2 at full return position	n/c	X40
2	LS202 - Activated when slide 2 at full depth.	n/c	X41

PDP-14 INPUT ASSIGNMENT SHEET -P2

INPUT BOX B (A THROUGH H)

INPUT NUMBER	ASSIGNMENT	NORM COND.	PROGRAM NUMBER
17	PB5 - Cycle start pushbutton.	n/c	X60
18	PB12 - Start motors pushbutton.	n/c	X61

INPUT NUMBERS

Box ↓	Box Terminal Number																			
	1	2	3	4	5	6	7	8	15	16	17	18	19	20	21	22	23	24
A	0	1	2	3	4	5	6	7	16	17	20	21	22	23	24	25	26	27
B	40	41	42	43	44	45	46	47	56	57	60	61	62	63	64	65	66	67
C	100	101	102	103	104	105	106	107	6	117	120	121	122	123	124	125	126	127
D	140	141	142	143	144	145	146	147	156	157	160	161	162	163	164	165	166	167

Figure 4-1

is wired to terminal 14 of I-box A. Note that program inputs are numbered octally (there are no "8's" or "9's" used in the number system). Input screw terminals are numbered in the familiar decimal number system.

The normal procedure to be followed when assigning input numbers is to complete an Input Assignment Sheet, as shown in Figure 4-1. This sheet is completed for each I box used in the PDP-14 system. The Input Assignment Sheet for one I box has two pages, each page containing 16 input assignments. The sheet records the machine input (e.g., LS 201 for limit switch 201) which is wired to a particular terminal and notes the implication of this input (e.g., "activated when slide 2 is in full return position"). A separate column records the normal condition of the contacts which are wired to the PDP-14 I box (normally open or normally closed). The last column of the sheet contains the input numbers used for PDP-14 programming purposes. These input numbers are read from the chart in Appendix A (part of which is reproduced in Figure 4-1) and are obtained by listing in the last column of the Input Assignment Sheet all the numbers which are found in the row for that particular I box in the Input Assignment Chart.

PDP-14 Outputs

The outputs of the PDP-14 system are solenoids, motor contactors, timers, small motors, indicators, signal devices, etc. These outputs are set ON or OFF by the PDP-14, dependent upon the current state of inputs or outputs. PDP-14 outputs are denoted by a "Y" followed by a number. Thus, Y123 is PDP-14 output 123; Y301 is PDP-14 output 301; etc. The convention of identifying an output with the letter "Y" is used throughout the PDP-14 system in the same way that inputs are identified by the letter "X".

The procedure followed to assign output numbers is similar to assigning input numbers. A sample Output Assignment Sheet is shown in Figure 4-2. This sheet should be completed for each O box in the PDP-14 system. The machine output (solenoid, motor contactor, etc.) which is wired to each O-box terminal is recorded with its function in the machine being controlled (e.g., "SOL A - clamps part at beginning of cycle.") The final column is the output numbers for programming purposes. These output numbers are read from the chart in Appendix A (reproduced in part in Figure 4-2).

PDP-14 Timers and retentive memories which are in A boxes are assigned to output box positions using the same procedure. Retentive memories may only be assigned to the addresses shaded in the chart of Appendix A.

BOOLEAN REPRESENTATIONS OF MACHINE CONTROL

Programming a PDP-14 requires familiarity with simple Boolean representations of control functions. These representations are comprised of operators and variables. The variables of PDP-14 control equations are inputs (X's) and outputs (Y's). The variables have two states; namely, ON and OFF. The operators in these equations are * (AND), + (OR) and / (NOT). Parentheses may be used within equations to group variables.

PDP-14 OUTPUT ASSIGNMENT SHEET

OUTPUT BOX E (A THROUGH S EXCLUDING I, O & Q)

OUTPUT NUMBER	ASSIGNMENT	PROGRAM NUMBER
1	SOL A - Clamps part at beginning of cycle.	Y120
2	SOL B - Returns head; activated by LS12.	Y121
3	SOL F - Unclamps part when LS4 is tripped.	Y122
4		Y123

OUTPUT NUMBERS

Box ↓ — Box Terminal Number — Box ↓

Box	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Box
A	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	A
B	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	B
C	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57	C
D	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77	D
E	100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117	E
F	120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137	F
G	140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157	G
H	160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177	H

Figure 4-2

For example, the equation:

$$Y10 = X23 + X21 * Y7$$

is read "output 10 is set ON when input 23 is ON, or when both output 7 and input 21 are ON." This equation instructs the PDP-14 to test input 23 and if it is ON, set output 10 ON. If input 23 is OFF, test output 7 and input 21. If they are both ON, set output 10 ON. If neither set of conditions is satisfied, set output 10 OFF.

The above equation could be represented by the following ladder diagram (Figure 4-3):

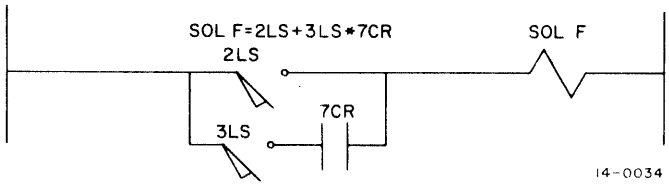


Figure 4-3

where SOL F corresponds to Y10; 2LS corresponds to X23; 7CR corresponds to Y7; and 3LS corresponds to X21. It is important to remember that input and output numbers are determined when the inputs and outputs are assigned to PDP-14 I and O boxes, respectively.

Equations Containing "NOT"

The equation operator / (NOT) is often used in control functions. For example, the equation:

$$Y27 = X5 * /X30$$

could represent the control function "solenoid A is energized if limit switch 5 is tripped and pushbutton 12 is not activated." This function could be solved with the following ladder diagram (Figure 4-4).

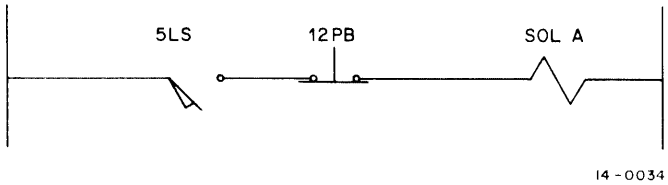


Figure 4-4

where SOL A corresponds to Y27; 5LS corresponds to X5; and X30 corresponds to 12PB. Note that a normally closed set of contacts for 12PB is shown in the ladder diagram. The NOT sign in the equation means not tripped or not pushed; it is a physical description.

The NOT sign in the PDP-14 system relates to the absence of 115 Vac at the input or output. It does not relate to physical switch positions. The PDP-14 may have either normally open or normally closed contacts wired to it, depending upon the user's choice. This is possible because the PDP-14 can test an input for the ON state or the OFF state. Thus, only one set of contacts need be wired to the PDP-14.

The choice of normal condition for input contacts must be made before writing the control equations for the PDP-14. The normal condition of the contacts should be noted on the Input Assignment Sheet. When normally open contacts are used, the equation "Y5 = X15" means that output 5 will be set ON when machine input 15 is activated (e.g., a limit switch is tripped) and 115 Vac is applied, and that output 5 will be set OFF when input 15 is not activated. Using normally closed contacts reverses the sense of the logic. When normally closed contacts are wired to the PDP-14, the equation Y5 = X15 means that output 5 will be set OFF when input 15 is activated, and will be set ON when input 15 is not activated.

Because normally closed contacts result in equations which seem contrary to common sense, it is suggested that all normally open contacts be used in the PDP-14 system. This allows standardization of programming and spare parts inventory.

Evaluating Equations

When an equation contains more than one variable, it is very important to note the manner in which the output state is determined. For example, consider the following equation:

$$Y1 = X1 * X2 + X3$$

Does the above equation represent Y1 = (X1 * X2) + X3 or Y1 = X1 * (X2 + X3)? By drawing two ladder diagrams it is obvious that the two interpretations are not equivalent (Figure 4-5).

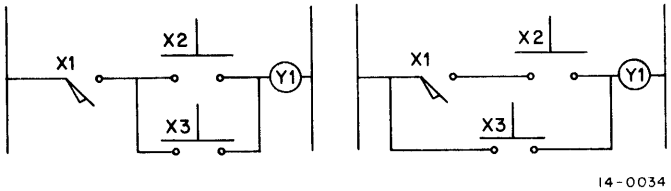


Figure 4-5

To remove possible ambiquities when evaluating equations, the variables grouped by the AND (*) operator are always combined before the variables grouped by the OR (+) operator whenever the order of combination is not clearly spelled out by parentheses. In the preceding example, therefore, (X1 * X2) + X3 (the second ladder diagram) is the correct interpretation of the expression X1 * X2 + X3. It should also be observed that Y1 = X1 * X2 + X3 is equivalent to Y1 = X3 + X1 * X2.

Examples:

X27 +/X17 * X20 = X27 + (/X17 * X20)
X11 * X23 + X31 * X14 = (X11 * X23) + (X31 * X14)
(X12 + X15) */X21 + X22 = [(X12 + X15) */X21] + X22
X21 + X22 * X23 + X24 * X25 = X21 + (X22 * X23) + (X24 * X25)

Parentheses may be added wherever necessary to define an expression more clearly.

When the NOT operator (/) precedes a single variable, the equation checks for the absence of the input (or output) at the I box or O box.

Equations with Variable Groups

Many control equations contain groups of variables which are separated from the rest of the equation by parentheses. This grouping simply means that the content of the parentheses is to be evaluated (for ON or OFF) and the result is to be treated as a single variable of the whole equation. For example, consider the following equation:

Y21 = /(X121 * X26)

The states of inputs 121 and 26 are tested and the resultant state of the quantity in the parentheses is inverted by the "NOT" to achieve the value of output 21. The following truth table summarizes the function:

X121	X26	X121*X26	Y21=/(X121*X26)
ON	ON	ON	OFF
ON	OFF	OFF	ON
OFF	ON	OFF	ON
OFF	OFF	OFF	ON

Notice that output Y21 is ON if either X121 or Y26 is OFF, or if both inputs are OFF. The following equation expresses this relationship:

Y21 = /X121 + /X26

Thus, the following replacement may always be made in control equations:

/(A*B) is equivalent to /A+/B

A similar reduction is possible for the following equation:

Y22 = /(X107 + X12)

The truth table to determine the state of output Y22 for all states of inputs follows.

X107	X12	(X107+X12)	Y22=/(X107+X12)
ON	ON	ON	OFF
ON	OFF	ON	OFF
OFF	ON	ON	OFF
OFF	OFF	OFF	ON

The reduction is possible when output Y22 is ON, only if input X107 and input X12 are both OFF. Thus, the equation may be rewritten as:

Y22 = /X107 * /X12

and the following replacement is always possible:

/(A + B) may be replaced by /A * /B

Equation Simplifications

There are several ways in which equations may be simplified before they are programmed for the PDP-14. While equations need not be in their simplest form, in many cases these simplifications conserve memory locations and lessen the time required to execute the program.

Consider the following equation:

Y5 = X1 + (X1 * X2) + (X3 * X5)

Note that the expression X1 * X2 adds no meaning to the equation. If X1 is ON, output Y5 will be set ON regardless of the state of X2. If X1 is OFF, again the state of X2 cannot affect Y5. Thus, the equation may be simplified to:

Y5 = X1 + (X3 * X5)

and the following rule is established:

A + (A * B) = A

The following equation may also be simplified:

Y6 = (X1 * X2 + X3) * X4 * /X3

Note that because of the last element of the equation (/X3), output Y6 will never be turned on when X3 is ON. Thus, the last element within the parentheses (+ X3) is meaningless since X3 may not be ON and OFF at the same time. The simplified equation is

$$Y6 = X1 * X2 * X4 * /X3$$

Thus the following rules are established:

A */A can never be true, and A +/A is always true

PDP-14 PROGRAM DESCRIPTION

After recording the input and output assignments and their functions, the programmer codes the program for the given control function. In general, a PDP-14 control program is made up of disjoint instruction groups. Each instruction group solves a Boolean equation by testing PDP-14 inputs and outputs and, at its conclusion, sets an output either ON or OFF. The final instruction group in the program ends with an unconditional jump to the start of the first instruction group. Thus the PDP-14 Program is a closed loop.

For example, if a machine control requires twenty outputs, there must be twenty instruction groups in the control program. The following diagram illustrates the construction of the program (Figure 4-6).

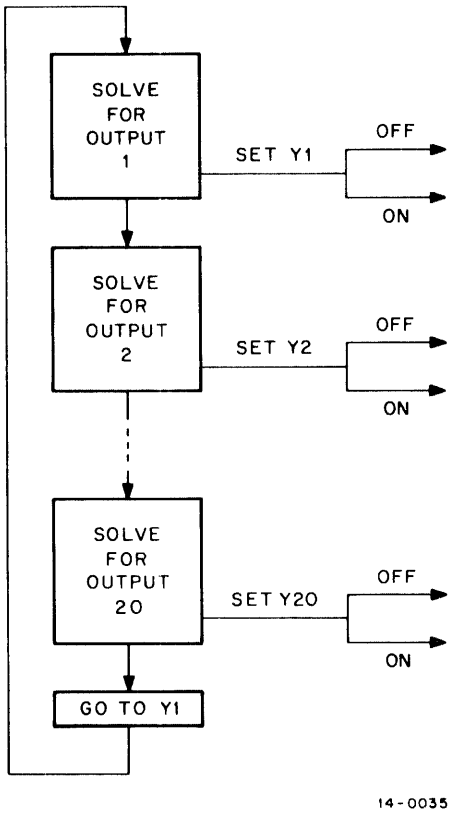


Figure 4-6

The disjoint instruction groups are often separated by several NOP's. The PDP-14 NOP instruction specifies No OPeration*. The NOP instruction is simply a space filler which leaves room for future program modifications or corrections.

The jump to the start of the program is very important. If there were no instruction to return control to the beginning of the program, the PDP-14 would stop executing instructions when it reached the end of its memory and the program would not start again.

It should be emphasized that each instruction group is independent. For example, some of the instruction groups could be controlling one device while other instruction groups control a second device. The sharing of a PDP-14 among many devices may be extended until the limits for inputs (256), outputs (255) or memory locations (4000) are exceeded.

INTRODUCTION TO THE PDP-14 SOFTWARE

"Software" describes the programs and systems which are used to operate a computer. There are three system programs in the PDP-14 Software Package. These three programs operate on PDP-8 family computers and are used to develop programs for the PDP-14. The three programs are:

BOOL-14	BOOL-14 is a compiler which translates special Boolean equations into PDP-14 machine instructions. A compiler translates expressions into machine code. However, there is no direct machine code counterpart of an element in the Boolean equation. The Boolean equation is translated into PDP-14 machine code but the translation process is not a one-for-one replacement.
PAL-14	PAL-14 is the assembler for the Program Assembly Language of the PDP-14. An assembly language is a set of instructions which are in mnemonic or symbolic form (letters) but which have a direct counterpart in the machine code of the PDP-14. In other words, the PAL-14 assembler translates the symbolic representations of the instructions of the PDP-14 into the PDP-14 machine code.
SIM-14	SIM-14 is a program which simulates PDP-14 operation in two modes of operation. The user may operate in an off-line or local mode to debug and modify his program completely within the PDP-8. When relatively certain that the program (or a particular part of the program) is correct, the user may switch to on-line mode and control the machine's operation by executing the program.

Chapters 6, 7 and 8 of this manual discuss the three foregoing software items. Before the system software may be understood, the reader must gain an understanding of the PDP-14 instruction set and its operation. The instructions used to solve control equations are described in Chapter 5. Instructions used by an external PDP-8 family computer to monitor a PDP-14 are described in Chapter 9. Chapter 10 describes operating procedures for the PDP-14 Software, and introduces a text Editor program which is useful for PDP-14 programming.

---

*NOP is the first PDP-14 instruction introduced in this manual. Others are presented in Chapters 5 and 9.

A few observations about use of the software items are helpful. The control engineer who is a computer novice wishing to use the PDP-14 will find BOOL-14 to be a practical approach to PDP-14 programming. He may write PDP-14 programs in equation form and have BOOL-14 generate the machine code instructions for him. The generated instructions are then tested or debugged with SIM-14. BOOL-14 will also allow the user to do transition checking, (one form of computer monitoring the PDP-14).

If greater flexibility in programming the PDP-14 is needed, PAL-14 may be the answer. PAL-14 is an assembler which translates symbolically written PDP-14 instructions into machine code instructions. Use of PAL-14 allows the user greater flexibility in the approach to monitoring, and more efficient use of memory. SIM-14 is also used to debug this translated program.

In general, BOOL-14 and PAL-14 are mutually exclusive; the user chooses one or the other, although this is not absolutely necessary. If the needed PDP-14 program is a combination of straightforward equations and more complicated instruction sequences, the user may compile part of the program with BOOL-14 and assemble the remainder with PAL-14. The two partial programs are merged by SIM-14. Care must be taken that the same memory locations are not used by both the BOOL-14 compiled program and the PAL-14 assembled program.

SIM-14 is a very powerful tool for exercising and testing PDP-14 machine language programs. Practically all PDP-14 users will find SIM-14 helpful regardless of whether BOOL-14 or PAL-14 is chosen to translate the program.

Chapter 10 outlines the operating procedures for PDP-14 software. Detailed descriptions of the use of BOOL-14 PAL-14 and SIM-14 are also included, as well as the operation of the PDP-8 family computer. Chapter 10 also describes the Editor, a fourth system program which will be indispensable when writing programs for BOOL-14 or PAL-14.

Paper Tape Formats

Programming a PDP-14 involves the use of several paper tapes. A paper tape uses punched holes to represent information in a similar manner to the familiar punched card. The holes in the paper tape are punched according to established codes or formats. The three types of PDP-14 paper tapes are as follows.

a. System Software Tapes - Punched in PDP-8 binary format. These tapes contain the SIM-14, PAL-14, BOOL-14 and Editor programs. They are loaded directly into a PDP-8 family computer.

b. Symbolic or Source Tapes - Punched in ASCII* format. These tapes have no meaning to the PDP-14 or to SIM-14. They are prepared by the Editor and are the input or source program representation for BOOL-14 or PAL-14. They may be listed on a Teletype to generate a readable copy of the program.

c. Binary or Machine Language Tapes - The translated form of PDP-14 programs and are punched in binary format. These tapes are punched by BOOL-14, PAL-14 or SIM-14 and may be read by SIM-14 or may be used to generate an ROM (Read-Only-Memory).

*ASCII is an abbreviation for U.S.A. Standard Code for Information Interchange.

SYSTEM DOCUMENTATION

While programming the PDP-14, many handwritten sheets of paper, Teletype listings and paper tapes will be generated. Some of these documents may be discarded but some should be saved and continuously updated. It is very important that each document be marked as it is generated. Note the date, your name, the system being programmed, the type of tape (e.g., "BOOL-14 source tape," "SIM-14 binary tape"), and other pertinent data.

The following is a partial list of forms with comments concerning their use and importance. These may not be fully understood at the present. However, when programming the PDP-14, bear this list in mind. It can save hours of time.

Forms Used During Programming of PDP-14

Form	Description
Input Assignment and Output Assignment Sheets	These sheets note which machine input or output (e.g., LS1) is wired to each PDP-14 terminal (e.g., X15). It should be constantly updated with any changes.
List of Equations	The handwritten list of equations (e.g., SOLA = LS1 * LS2) should be saved until they become part of a program listing in the form of comments. At that time, the handwritten sheets may be discarded.
Program Coding Sheets	These sheets constitute the handwritten PDP-14 program in the form of equations (e.g., Y17 = X15 * X16) or instructions to be typed into the Editor. Once a source listing is generated from the Editor, it is discarded.
Source Program Listing	This is a Teletype printout of the program in the input format to BOOL-14 or PAL-14. It should always reflect the latest program changes. The listing should contain "comments" to reflect the equations in symbolic form (e.g., SOLA = LS1 * LS2).
Source Program Paper Tape	This is a paper tape record which may be used as input to BOOL-14 or PAL-14. It corresponds to the above source listing and should also reflect latest program changes.
BOOL-14 Compiler or PAL-14 Assembler Listing	This is a Teletype printout of the translated program. It contains the symbolic and numeric contents of memory locations used by the PDP-14 program. The numeric listing may be used when wiring changes are made by hand to the ROM. It must be kept updated.
BOOL-14 or PAL-14 Binary Tape	This tape is input to SIM-14 and should correspond to the above compiler or assembly listing.
SIM-14 Listing	This is a Teletype listing of any changes or "patches" made to the PDP-14 program with SIM-14. These listings should be kept until the changes are incorporated in the source program and a new source listing and tape is generated and recompiled or re-assembled to generate a new BOOL-14 or PAL-14 listing and tape.

Forms Used During Programming of PDP-14 (Cont)

Form	Description
SIM-14 Binary Tape	This binary tape is punched in 1K segments and is the form required to generate an ROM. It is punched from SIM-14 when no errors have been detected in the BOOL-14 or PAL-14 program. It is then sent to DEC.

CHAPTER 5
PDP-14 BASIC INSTRUCTIONS

This chapter introduces the PDP-14 basic instructions which determine the state (ON or OFF) of outputs according to the changing state of inputs and outputs. They are the primary operating instructions of the PDP-14 system, and are recognized and simulated by SIM-14.

A PDP-14 program may be written either as equations for BOOL-14 or as instructions for PAL-14, but its final form always uses the basic instructions. (Additional PDP-14 instructions used for computer monitoring are introduced in Chapter 9.)

BASIC INSTRUCTION CLASSES

The three classes of basic instructions used in PDP-14 programs are listed below:

a. Test Instructions - used to sample the current state (ON or OFF) of an input or output.

b. Set Instructions - used to set outputs ON or OFF. (The state of inputs is changed by the controlled equipment, not by the PDP-14 itself.)

c. Jump Instructions - used to execute instructions out of sequence. That is, to "branch" the PDP-14 program. Instructions are always executed in sequence unless a jump occurs.

PDP-14 TEST FLAG

The TEST flag is the element of the PDP-14 system which records the results of input and output testing. The TEST flag has two values (ON and OFF). It is raised (set ON) by a true test and left unchanged by a false test. (True tests and false tests are defined below.) The condition of the TEST flag is sampled by jump instructions to determine set output operations. In this way, the basic instructions are used with the TEST flag to control the operations of a machine.

The TEST flag is sometimes referred to as the test flip-flop or simply the test flop. This terminology is most often used when describing the PDP-14 processor hardware, in which the TEST flag is represented by a solid-state flip-flop.

PDP-14 TEST INSTRUCTIONS

The PDP-14 test instructions are:

a. Test an input for the ON state.

b. Test an input for the OFF state.

c. Test an output for the ON state.

d. Test an output for the OFF state.

The symbolic and numeric forms of these instructions and their precise definitions are given in Table 5-1.

Table 5-1
PDP-14 Test Instructions

Instruction		Definition
Symbolic	Numeric	
TXN XXX	2400 + XXX	Test input XXX for the ON state. If input XXX is ON and the TEST flag is currently OFF, the TEST flag is set ON. Otherwise the flag remains unchanged.
TXF XXX	2000 + XXX	Test input XXX for the OFF state. If input XXX is OFF and the TEST flag is currently OFF, the TEST flag is set ON. Otherwise the flag remains unchanged.
TYN YYY	1400 + YYY	Test output YYY for the ON state. If output YYY is ON and the TEST flag is currently OFF, the TEST flag is set ON. Otherwise the flag remains unchanged.
TYF YYY	1000 + YYY	Test output YYY for the OFF state. If output YYY is OFF and the TEST flag is currently OFF, the TEST flag is set ON. Otherwise the flag remains unchanged.
NOTE: The expressions XXX and YYY represent numeric values of inputs (XXX) and outputs (YYY) as assigned using the Appendix A Charts. Legal values are 0-377.		

The definitions may be explained by defining the terms "true test" and "false test" as given in the following table.

	State of Input or Output	
	ON	OFF
Test for ON (TXN or TYN)	True test	False test
Test for OFF (TXF or TYF)	False test	True test

The action of the test instructions may be summarized making use of the above definitions: Test instructions which result in a true test set the TEST flag ON; those which result in a false test leave the TEST flag unchanged.

NOTE

The TEST flag is NOT cleared by a false test, as summarized below.

	Original State of TEST Flag		
	ON	OFF	
True test	ON	ON	Resultant State of TEST Flag
False test	ON	OFF	

PDP-14 SET OUTPUT INSTRUCTIONS

Set instructions turn an output ON or OFF at the end of a sequence of test instructions. The symbolic and numeric forms of these instructions and their definitions are given in Table 5-2. The SYN and SYF instructions reference output numbers (YYY) from 0 through 377 (octal), and are determined by the selection of O box sockets and the assignment of outputs within the O box.

Table 5-2
PDP-14 Set Output Instructions

Instruction		Definition
Symbolic	Numeric	
SYN YYY	3400-YYY	Set output YYY to the ON state. Output YYY remains ON until it is set OFF by a SYF or CLR instruction.
SYF YYY	3000+YYY	Set output YYY to the OFF state. Output YYY remains OFF until it is set ON by a SYN instruction.
CLR	3377	Set all PDP-14 outputs to the OFF state. This instruction sets every output OFF unconditionally (with the exception of the retentive memories) and it should not be used in normal conditions.
NOTE: The expression YYY represents the numeric value of an output as assigned from the Appendix A chart.		

The instruction CLR is equivalent to SYF 377 and causes all timers, storage outputs and regular outputs to be set OFF. Retentive memories are only set OFF when addressed individually. Thus, output 377 should not be used as a "real" output since it cannot be set OFF without setting all outputs OFF.

ADDRESSING LOCATIONS IN THE PDP-14

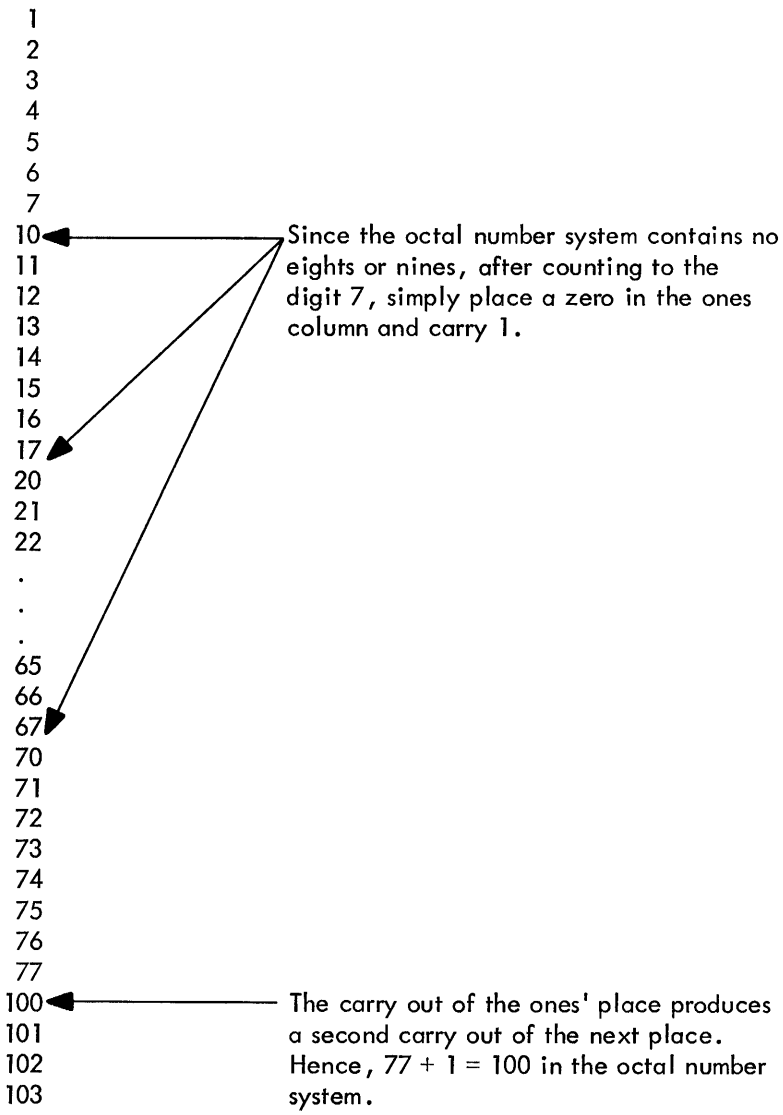
Jump instructions cause the transfer of program control to another section of the program. The jump instruction must specify the section of the program to which it is transferring control. That is, in PDP-14 terminology, jump instructions must be able to address or reference memory locations.

Every instruction must be stored in the PDP-14 memory. The storage process is not of concern at this point in the discussion. What is important here is that the instructions are stored in certain locations in the memory and that these locations have a numeric order (i.e., there is a first location, a second, a third, etc., through to the last memory location). Each of these PDP-14 memory locations has an assigned number, or address. Thus, a PDP-14 program consists of instructions which are stored in memory locations each of which has an address. This permits jump instructions to direct the PDP-14 to "jump to location 305" (begin execution of the instructions starting in location 305).

The PDP-14 memory locations are not numbered in an ordinary way, however. They are numbered in "octal", a numbering system which does not use the digits "8" or "9". Counting in octal is the same as counting in the familiar decimal number system except that any number which contains an "8" or "9" is excluded. Thus the number which follows 7 in octal is 10, and the number which follows 77 is 100.

Octal numbers are usually given a subscript (8) to avoid confusion with decimal numbers. Thus "10_8" is not the same as the decimal number "10".

The following sequence of numbers illustrates the octal counting technique.



PDP-14 Memory Organization

The locations of a PDP-14 memory are numbered in octal as previously described. A 1K PDP-14 memory contains 1024 locations in which program instructions may be stored. These locations are numbered 0_8 through 1777_8. The locations of a 2K PDP-14 memory (2048 locations) are numbered 0_8 through 3777_8; 3K memory locations (3072 locations) are numbered 0_8 through 5777_8; and 4K memory locations (4096 locations) are numbered 0_8 through 7777_8.

A PDP-14 memory is constructed of blocks of 256 locations. A 1K PDP-14 memory contains four of these blocks, called "pages". The 256 locations of each page are numbered from 0_8 through 377_8. Figure 5-1 shows the

relationship between the numbering of locations through all memory ("absolute" addresses) and the numbering of location within each page of memory ("relative" or "page" addresses). The maximum possible PDP-14 memory (4K) is diagrammed.

Absolute Addresses and Page Addresses

Each location in the PDP-14 memory has an absolute address and a relative, or page, address. For example, Figure 5-1 shows that the location with absolute address 4377 has page address 377. Note that there are many locations with the same relative or page address, but each PDP-14 location has its own unique absolute address.

The page address of a location can easily be determined from its absolute address by means of the following procedure:

a. If the absolute address is 1000 or greater, cross off the first (most significant) digit.

b. If the number resulting from step 1 is less than 400, it is the page address.

c. If the number resulting from step 1 is 400 or greater, the page address is found by subtracting 400 from the step 1 result.

The examples below illustrate the procedure.

Absolute address	Step a. result	Page address
1234	234	234
3347	347	347
2652	652	252
7521	521	121
7400	400	0
5000	0	0

PDP-14 JUMP INSTRUCTIONS

Once program execution is started, the PDP-14 normally proceeds by executing the instructions in memory in their natural sequence. For example, if execution of the PDP-14 program begins at location 200, the instruction in that location will be executed, then the instruction in location 201, followed by the instruction in location 202, etc., throughout the entire program. Jump instructions cause departures from this sequential execution of instructions.

Conditional Jump Instructions

The PDP-14 has two conditional jump instructions. The state of the TEST flag is the condition which determines whether the jump will occur. The symbolic and numeric forms of the conditional jump instructions and their definitions are given in Table 5-3.

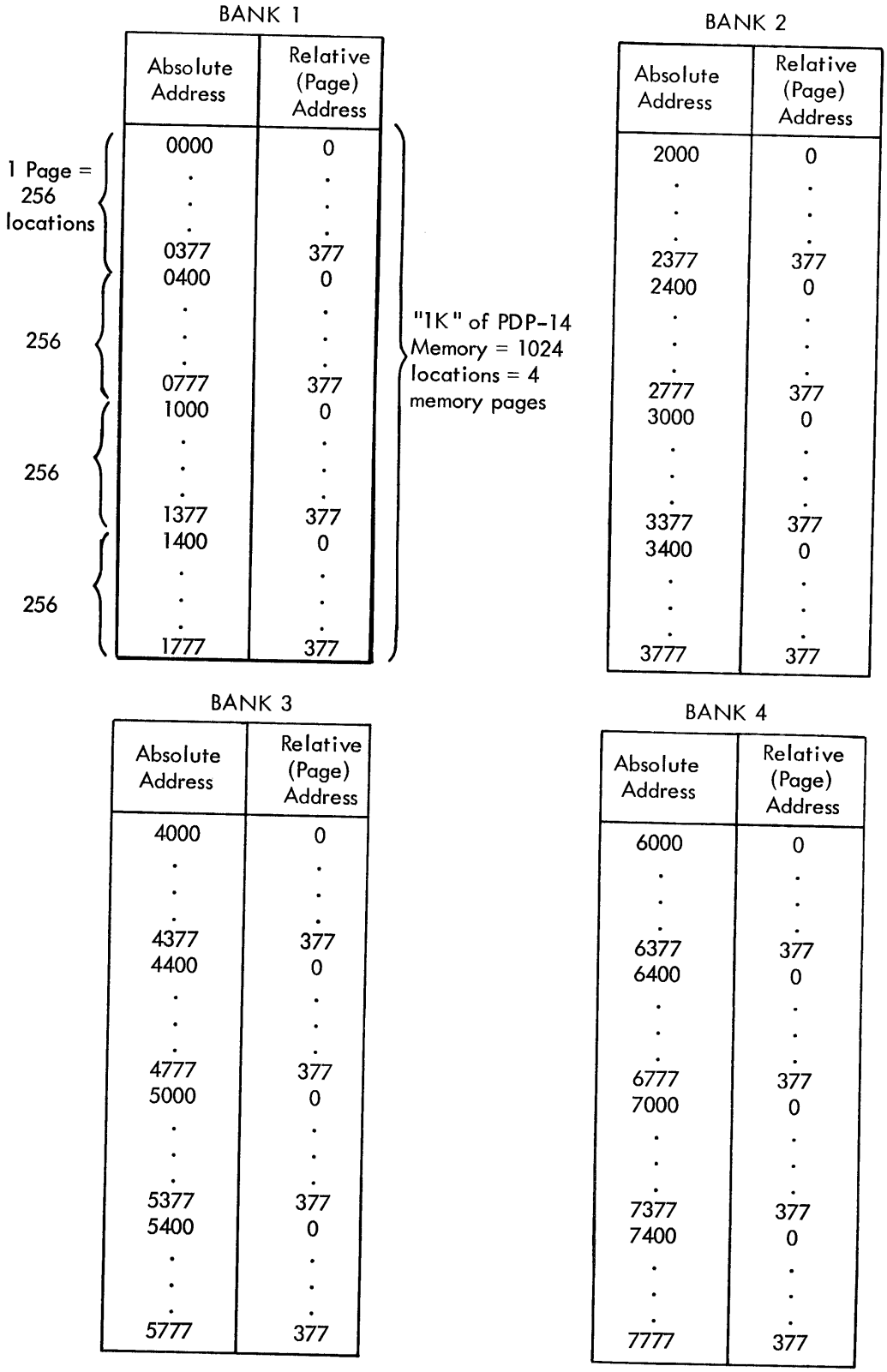

BANK 1
Absolute Address
Relative (Page) Address
1 Page = 256 locations
256
256
256
"1K" of PDP-14 Memory = 1024 locations = 4 memory pages
0000 0
0377 377
0400 0
0777 377
1000 0
1377 377
1400 0
1777 377
BANK 2
Absolute Address
Relative (Page) Address
2000 0
2377 377
2400 0
2777 377
3000 0
3377 377
3400 0
3777 377
BANK 3
Absolute Address
Relative (Page) Address
4000 0
4377 377
4400 0
4777 377
5000 0
5377 377
5400 0
5777 377
BANK 4
Absolute Address
Relative (Page) Address
6000 0
6377 377
6400 0
6777 377
7000 0
7377 377
7400 0
7777 377

Figure 5-1

Conditional jump instructions are used to cause an SYN (set output ON) instruction to occur for one state of the TEST flag and an SYF (set output OFF) for the other TEST flag state.

NOTE

The JFF and JFN instructions always set the TEST flag OFF. After execution of a JFF or JFN instruction, the TEST flag is always OFF, regardless of whether or not the jump was executed.

Table 5-3
PDP-14 Conditional Jump Instructions

Instruction		Definition
Symbolic	Numeric	
JFN NNN	5400 + NNN	Jump to location NNN if the TEST flag is now ON and execute the instructions beginning with the instruction in location NNN. (If the flag is OFF, the instructions following the JFN continue to be executed in sequence.) The TEST flag is set OFF, regardless of its original state.
JFF NNN	5000 + NNN	Jump to location NNN if the TEST flag is now OFF and execute the instructions beginning with the instruction in location NNN. (If the TEST flag is ON, the instructions following the JFF continue to be executed in sequence.) The TEST flag is set OFF, regardless of its original state.
NOTE: The expression NNN represents the relative or page address to which control may be transferred. Legal values are 0 to 377 (octal).		

Conditional jump instructions are limited. They can only reference the 256 locations (page addresses 0 through 377) on their own page (often called the current page).

Because of internal restrictions, a JFF or JFN instruction must never be stored in the last memory location of a page (e.g., locations 377, 777, 1377, 1777, etc.).

The instruction JFN 300 illustrates the use of a conditional jump instruction. This instruction means that if the TEST flag is ON, jump to relative location 300 on the current page. Recall that several absolute addresses have the same relative or page address. Thus, a conditional jump instruction has different meanings when it is stored on different pages of PDP-14 memory. Observe the examples given in the following paragraphs in order to properly understand location to be addressed.

Follow these steps to find the relative address to be used to jump to a specific absolute address:

a. Make certain that the location which contains the jump instruction is on the same page as the location to be addressed by the jump instruction.

b. Cross off the first (most significant) digit of the address if it is 1000 or greater.

c. If the result of step b is less than 400, the relative (page) address is that number.

d. If the result of step b is 400 or greater, subtract 400 to obtain the relative (page) address.

Examples:

Location of Instruction:	500
Location to be Addressed:	750
Correct Instruction:	JFN 350
Location of Instruction:	50
Location to be Addressed:	372
Correct Instruction:	JFF 372
Location of Instruction:	2005
Location to be Addressed:	2345
Correct Instruction:	JFN 345
Location of Instruction:	5500
Location to be Addressed:	5675
Correct Instruction:	JFF 275
Location of Instruction:	2311
Location to be Addressed:	2450
Correct Instruction:	Cannot be done. (Off-page reference)
Location of Instruction:	4251
Location to be Addressed:	5300
Correct Instruction:	Cannot be done. (Off-page reference)
Location of Instruction:	5631
Location to be Addressed:	5237
Correct Instruction:	Cannot be done. (Off-page reference)
Location of Instruction:	435
Location to be Addressed:	776
Correct Instruction:	JFN 376

Location of Instruction:	2722
Location to be Addressed:	3010
Correct Instruction:	Cannot be done. (Off-page reference)

NOTE

The foregoing examples which cannot be done with conditional jump instructions require modifications in the program. The program may be changed to use unconditional jumps to pass from memory page to memory page. Unconditional jump instructions are described on the following pages.

Unconditional Jump Instructions

The PDP-14 has two unconditional jump instructions. The symbolic form, numeric form and definition of each instruction is given in Table 5-4.

Table 5-4
PDP-14 Unconditional Jump Instructions

Instruction		Definition
Symbolic	Numeric	
JMP NNNN	4224 NNNN	Jump to location NNNN unconditionally. Execution of the PDP-14 program proceeds sequentially beginning with the instruction in location NNNN. JMP is a two-location instruction; the absolute address of the "jump-to" location is stored in the location following that which contains the JMP instruction.
SKP	0344	Skip the following memory location unconditionally. This instruction causes the PDP-14 to not execute the instruction which is stored in the next sequential location. Sequential execution continues with the instruction stored in the second location following the SKP instruction.
NOTE: The expression NNNN represents an absolute address for the PDP-14 memory. There is no page restriction for the JMP instruction.		

As stated in Table 5-4, the JMP is a two-location instruction. The first part of the instruction tells the PDP-14 to jump to the address given in the second part of the instruction. The first instruction executed after the jump will be the one stored in the location with that address.

The JMP instruction may reference any PDP-14 memory location using an absolute address. Thus, whenever the program must pass between memory pages, the unconditional jump is used.

For example, to transfer control to location 1523 from location 1375, use the instruction:

Location	Content
.	.
.	.
.	.
1375	JMP
1376	1523
.	.
.	.
.	.
1523	(Contains next instruction to be executed.)

The SKP instruction is an unconditional skip of one PDP-14 memory location and is very convenient to use because the address of the location to which control is passed need not be specified. The SKP instruction has no need to reference memory. Note that the SKP instruction skips one PDP-14 location only. It may therefore be used to skip any one-location PDP-14 instruction only, but may not be used to skip a two-location instruction (such as in JMP).

Subroutine Jump Instructions

Included in the PDP-14 basic instructions are jump instructions that allow programs to contain subroutines. A subroutine is simply a set of instructions to be executed more than once on each pass through PDP-14 memory. Instead of including them in more than one place in the program, these instructions are written in the form of a subroutine. Repeated execution of the same instructions is permitted via an instruction to jump to the start of the set of subroutine instructions and another instruction that jumps to return after executing the set of instructions contained in the subroutine.

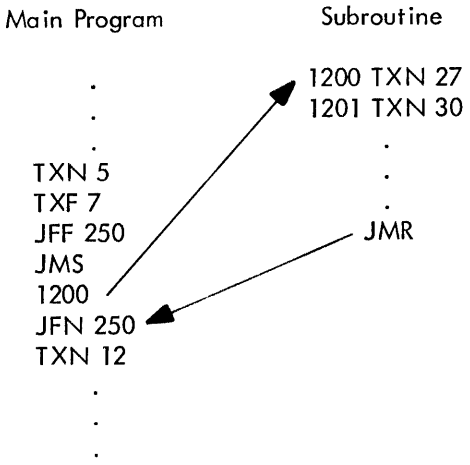
The jump instructions which are used for subroutines are given in Table 5-5. The JMS instruction is similar to the JMP instruction; both are two-location instructions which may address any PDP-14 memory location. The JMS differs from the JMP in that it allows subroutines to return to the next sequential location following the second part of the JMS instruction by saving the address of the next sequential instruction.

Table 5-5
PDP-14 Subroutine Jump Instructions

Instruction		Definition
Symbolic	Numeric	
JMS NNNN	4645 NNNN	Jump to the subroutine beginning in location NNNN. The PDP-14 executes in sequence the instructions beginning with the instruction stored in location NNNN, and terminated by a JMR instruction. The JMS instruction is a two-location instruction which may directly address all PDP-14 memory.
JMR	0354	Jump return to the location following the second part of the JMS instruction. The JMR must always be paired with a JMS instruction.
NOTE: The expression NNNN represents the absolute address of the start of a subroutine. There is no page restriction for the JMS instruction.		

The PDP-14 permits only one level of subroutines because only one return address may be stored at one time. If a subroutine were to jump to a second subroutine, the first return address would be lost, and return to the original program would be impossible.

PDP-14 subroutines are written in the following manner:



The JMS in the above example causes transfer to the subroutine which starts at location 1200. The instructions of the subroutine are then executed in sequence until the JMR instruction is encountered. Control is then returned to the instruction which follows the second part of the two-location JMS instruction.

SAMPLE PDP-14 PROGRAM AND ANALYSIS

The following equation represents a typical control requirement to be programmed:

$$Y1 = X123 + X54 * Y3$$

This equation could represent the control function "Solenoid A (Y1) is energized when pushbutton 43 (X123) is ON, or when both limit switch 12 (X54) is activated and Solenoid C (Y3) is energized."

The equation may be solved by the PDP-14 with the instructions in Figure 5-6. Notice that the user must write down the location of each instruction and he must refer to each input (X) and output (Y) by its program number as determined from the tables in Appendix A.

Table 5-6
Sample PDP-14 Program

Location	Symbolic Content	Comment
300	TXN 123	If input 123 is ON, the flag is set ON. If the flag is ON, jump to 307 and set Y1 ON. Otherwise, if either input 54 is OFF, or if output 3 is OFF, the flag is set ON. If the flag is OFF, jump to 307 and set Y1 ON. Otherwise set output 1 OFF and skip the set output ON instruction. This instruction sets Y1 ON. After determining the state for output 1 return to the start of the program.
301	JFN 307	
302	TXF 54	
303	TYF 3	
304	JFF 307	
305	SYF 1	
306	SKP	
307	SYN 1	
310	JMP	
311	300	

To demonstrate that the instructions in Figure 5-2 actually solve the equation properly for all possible values of the variables, three cases are analyzed below. In each case, the TEST flag is assumed to be OFF as the result of a previously executed JFF or JFN instruction.

Case 1 - Input 123 OFF, input 54 OFF, and output 3 OFF; Result: Y1 OFF

Location	Content	Program Execution
300	TXN 123	Input 123 is tested for the ON state. Since it is OFF, the TEST flag remains OFF, and program execution proceeds to the instruction contained in location 301.
301	JFN 307	If the TEST flag is ON at this point, the program will proceed to location 307. The flag is OFF, however, and program execution proceeds to the instruction contained in location 302.
302	TXF 54	The program tests input 54 for the OFF state. Since input 54 is OFF, the TEST flag is now set ON. Program execution proceeds to the instruction contained in location 303.

Case 1 (Cont)

Location	Content	Program Execution
303	TYF 3	The program tests output 3 for the OFF state. Since output 3 is OFF, the TEST flag would be set ON had it not been set ON already by the previous instruction. Program execution proceeds to the instruction contained in location 304.
304	JFF 307	If the TEST flag is OFF at this point, the program will proceed to location 307 to set output 1 ON. The TEST flag is ON, however. Thus program execution proceeds to the instruction contained in location 305. (The TEST flag is set OFF by the JFF before proceeding.)
305	SYF 1	Output 1 is set OFF (as it should be). Program execution proceeds to the instruction contained in location 306.
306	SKP	This instruction causes program execution to skip location 307 and proceed to the instruction contained in location 310.
310 311	JMP 300	The instruction in location 310 is recognized as the first part of a two-location unconditional jump instruction. The JMP causes program execution to proceed to location 300, which will start testing the equation again.

NOTE

The instruction in location 307 is not executed.

Case 2 - Input 123 ON, input 54 ON, and output 3 OFF; Result: Y1 ON

Location	Content	Program Execution
300	TXN 123	Input 123 is tested for the ON state. Since input 123 is ON, the TEST flag is set ON and program execution proceeds to the instruction contained in location 301.
301	JFN 307	Since the TEST flag is ON at this point, program execution proceeds to the instruction contained in location 307. (The TEST flag is set OFF by the JFN before proceeding.)
307	SYN 1	Output 1 is set ON (as it should be). Program execution proceeds to the instruction contained in location 310.
310 311	JMP 300	The instruction in location 310 is recognized as the first part of a two-location unconditional jump instruction. The JMP causes program execution to proceed to location 300, which will start testing the equation again.

NOTE

The instructions in locations 302, 303, 304, 305, and 306 are not executed.

Case 3 - Input 123 OFF, input 54 ON, and output 3 ON; Result: Y1 ON

Location	Content	Program Execution
300	TXN 123	Input 123 is tested for the ON state and since it is OFF, the TEST flag remains OFF. Program execution proceeds to location 301.
301	JFN 307	Since the TEST flag is OFF at this point, program execution does not proceed to location 307, but rather proceeds to the instruction contained in location 302.
302	TXF 54	Input 54 is tested for the OFF state. Since input 54 is ON, the TEST flag remains OFF and program execution proceeds to the instruction in location 303.
303	TYF 3	Output 3 is tested for the OFF state. Since output 3 is ON, the TEST flag again remains OFF and program execution proceeds to the instruction contained in location 304.
304	JFF 307	Since the TEST flag is OFF at this point, program execution proceeds to the instruction contained in location 307.
307	SYN 1	Output 1 is set ON (as it should be). Program execution then proceeds to the instruction contained in location 310.
310 311	JMP 300	The instruction in location 310 is recognized as the first part of a two-location unconditional jump instruction. The JMP causes program execution to proceed to location 300, which will start testing the equation again.

NOTE

The instructions in locations 305 and 306 are not executed.

PDP-14 programs may be written as the sample described above, but they may be written in formats which are easier to work with. These easier techniques for writing PDP-14 programs are fully described in Chapters 6 and 7. However, an understanding of the material described in this chapter is necessary to perform the debugging procedure required for all PDP-14 programs regardless of their format.

PROGRAM EXAMPLES

The following examples show the relationship between the circuit diagram, the Boolean representation and PDP-14 programs. Each example contains five parts.

a. A verbal explanation of the control function.

b. The ladder diagram which would be used to perform this function.

c. A Boolean representation of the control function.

d. A wiring diagram for the PDP-14 input and output boxes.

e. The PDP-14 program for the control function.

In each of the examples, it is assumed that the TEST flag is OFF initially.

Boolean "OR" Control Function - A simple control circuit in which a solenoid is energized by either one of two limit switches is presented in Example 1. To solve this problem with a PDP-14, test for either input's ON state. Since a positive test from either input will set the TEST flag to 1, the output should be turned on when the TEST flag is ON, and OFF when the TEST flag is OFF.

Notice that normally open contacts of each switch are wired to the PDP-14. Thus, testing for ON means testing for an activated switch. Testing for OFF means testing for a not activated switch.

EXAMPLE 1

Control Function

If either limit switch 1 or limit switch 2 is ON, energize Solenoid D (otherwise, Solenoid D should be OFF).

Ladder Diagram (Figure 5-2)

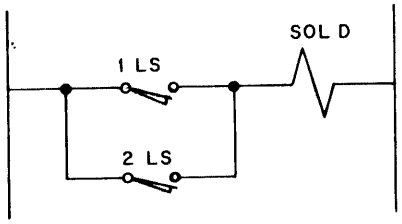


Figure 5-2

Boolean Equivalent

Inputs:	Outputs:
1LS (n/o) X11	SOL D Y7
2LS (n/o) X12	

Equation: Y7 = X11 + X12

EXAMPLE 1 (Cont)

PDP-14 Wiring (Figure 5-3)

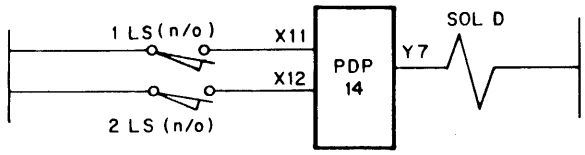


Figure 5-3

PDP-14 Program

Location	Content	Comments
0	TXN 11	Test input 11 and 12; if either is ON, set
1	TXN 12	output ON; if not, set output OFF.
2	JFN 5	
3	SYF 7	
4	SKP	
5	SYN 7	

Boolean "AND" Control Function - Example 2 illustrates a control where two inputs are in series to drive an output. In other words, both limit switch 3 and pushbutton 1 must be activated for Solenoid A to be energized. Normally open contacts are wired to the PDP-14 input box.

The PDP-14 program checks for the condition which is not wanted. If either input is OFF, Solenoid A should not be energized. Thus, the program tests both inputs, setting the TEST flag if either is OFF. The program jumps, if the TEST flag is ON, to the set output OFF instruction. If the flag remains OFF during the testing, the output for Solenoid A is turned ON, and the solenoid is energized.

EXAMPLE 2

Control Function

If pushbutton 1 and limit switch 3 are ON, energize Solenoid A (otherwise Solenoid A should be de-energized).

EXAMPLE 2 (Cont)

Ladder Diagram (Figure 5-4)

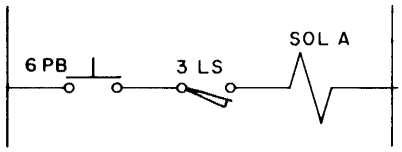


Figure 5-4

Boolean Equivalent

Inputs:		Output:	
1PB (n/o)	X1	SOL A	Y4
3LS (n/o)	X13		

Equation: Y4 = X1 * X13

PDP-14 Wiring (Figure 5-5)

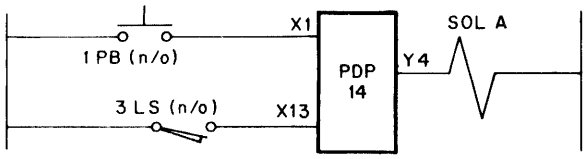


Figure 5-5

PDP-14 Program

Location	Content	Comment
6	TXF 1	Test inputs 1 and 13; if either is OFF, the output will not be set.
7	TXF 13	
10	JFF 13	
11	SYF 4	
12	SKP	
13	SYN 4	

Control Functions with Normally Open and Closed Contacts - Example 3 is a control function which (in ladder diagram form) has both normally open and normally closed contacts. The PDP-14 does not require two types of contacts for such control functions. All normally open contacts may be used, as illustrated in Example 3. Normally open contacts for both pushbutton 7 and limit switch 5 are wired to the PDP-14. Thus, an ON input means that the button or switch is activated. An OFF input means that the button or switch is not activated.

The example also illustrates the combination of AND and OR functions. Each leg is an AND function and the output results from the OR of the legs.

The PDP-14 program could also be written to test normally closed contacts. The program would merely test for the opposite state (ON or OFF) of the input. Thus, if normally closed contacts are used in the wiring, all TXF's become TXN's for the input and all TXN's become TXF's. An ON input from normally closed contacts means that the switch is not activated while an OFF input means it is activated.

EXAMPLE 3

Control Function

If pushbutton 6 and limit switch 4 are on, or if pushbutton 7 and limit switch 5 are both OFF, then Solenoid C is energized (otherwise Solenoid C is de-energized).

Ladder Diagram (Figure 5-6)

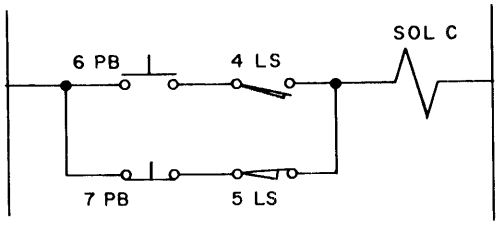


Figure 5-6

EXAMPLE 3 (Cont)

Boolean Equivalent

Inputs:		Output:
6PB (n/o)	X6	SOL C Y6
7PB (n/o)	X7	
4LS (n/o)	X14	
5LS (n/o)	X15	

Equation: $Y6 = (X6 * X14) + (\overline{X7} * \overline{X15})$

PDP-14 Wiring (Figure 5-7)

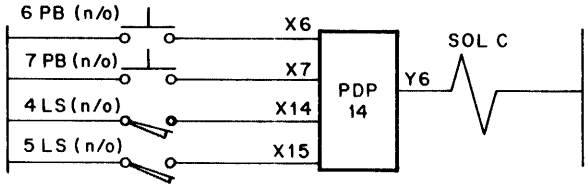


Figure 5-7

PDP-14 Program

Location	Content	Comment
14	TXF 6	If either X6 or X14 is OFF, other leg must be
15	TXF 14	tested. If both ON, output should be ON.
16	JFF 24	
17	TXN 7	If either X7 or X15 is ON and other leg
20	TXN 15	failed, set output OFF.
21	JFF 24	
22	SYF 6	Set output OFF if both legs fail.
23	SKP	
24	SYN 6	Set output ON if either leg solves.

Latching Control Function - Example 4 is a motor contactor latching function. Pushbutton 2 is the start button, and pushbutton 3 is the stop button for energizing a motor contactor. Again, normally open contacts are assumed for all inputs. However, the normally closed contact for 3 PB could be used by replacing the TXN 3 by TXF 3 in the PDP-14 program.

The program uses a test output instruction, TXF 10, to latch the motor contactor on. The output will remain on as long as it is currently on and pushbutton 3 is not pressed. A PDP-14 input for the motor contacts is not required because outputs may be tested as inputs.

EXAMPLE 4

Control Function

Motor 1 is started if pushbutton 2 is pressed, and continues to operate until pushbutton 3 is pressed.

Ladder Diagram (Figure 5-8)

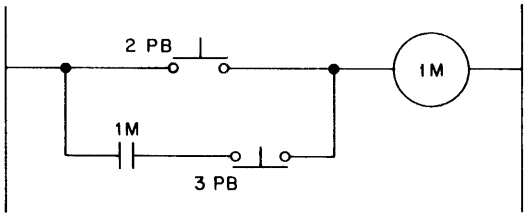


Figure 5-8

Boolean Equivalent

Inputs:		Output:
2PB (n/o)	X2	1M Y10
3PB (n/o)	X3	

Equation: $Y10 = X2 + (\overline{X3} * Y10)$

PDP-14 Wiring (Figure 5-9)

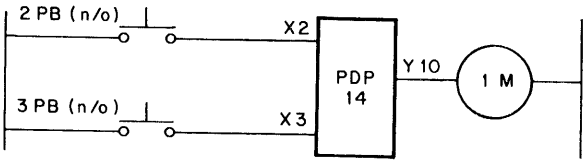


Figure 5-9

PDP-14 Program

Location	Content	Comments
25	TXN 2	If X2 is ON, jump to set output ON; otherwise, if X3 is OFF and Y10 is already ON, leave it ON; otherwise, set the output OFF.
26	JFN 34	
27	TXN 3	
30	TYF 10	
31	JFF 34	
32	SYF 10	
33	SKP	
34	SYN 10	

Relay Controlled Function - Example 5 uses a control relay to energize a solenoid. All normally open contacts are used as inputs to the PDP-14. (As explained in the two preceding examples, normally closed contacts could be used by reversing the sense of the test instructions.) Control relay 2 is energized by inputs, and latches itself in the engaged position. The output for control relay 2 is then tested to energize solenoid B.

EXAMPLE 5

Control Function

If pushbutton 4 is not ON and limit switch 2 is actuated, and either pushbutton 5 is ON or control relay 2 is ON already, then control relay 2 is set ON. If control relay 2 is ON, solenoid B is energized.

Ladder Diagram (Figure 5-10)

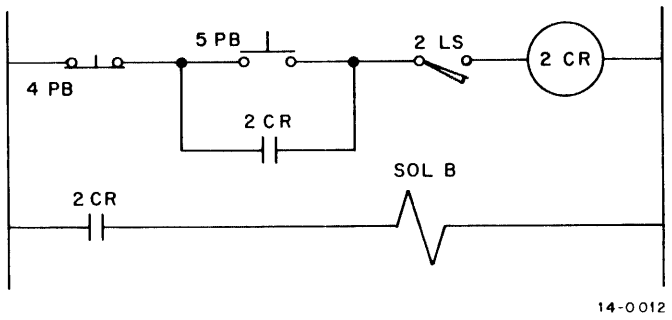


Figure 5-10

EXAMPLE 5 (Cont)

Boolean Equivalent

Inputs:		Outputs:	
4PB (n/o)	$\overline{X4}$	2CR	Y1
5PB (n/o)	X5	SOL B	Y5
2LS (n/o)	X12		

Equations: $Y1 = \overline{X4} * (X5 + Y1) * X12$

$Y5 = Y1$

PDP-14 Wiring (Figure 5-11)

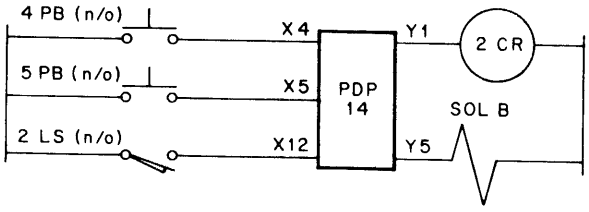


Figure 5-11

PDP-14 Program

Location	Content	Comments
35	TXN 4	If either X4 is ON or X12 is OFF, the out-
36	TXF 12	put should be OFF.
37	JFN 43	
40	TXN 5	If X4 is OFF, and X12 is ON, then either
41	TYN 1	X5 or Y1 must be ON for the output to be ON.
42	JFN 45	
43	SYF 1	
44	SKP	
45	SYN 1	
46	TYN 1	
47	JFN 52	If output 1 is ON, jump to set output 5 ON.
50	SYF 5	Otherwise, set output 5 OFF.
51	SKP	
52	SYN 5	

The Example 5 illustrates an important PDP-14 concept. Note that although the original equation was $Y1 = \overline{X4} * (X5 + Y1) * X12$, the program was written for the equation $Y1 = \overline{X4} * X12 * (X5 + Y1)$. Programming is more efficient when single variables connected with the same operator (AND or OR) are grouped together.

Simplifications - The control program for Example 5 could be greatly simplified by the observation that control relay 2 is an unnecessary output. The control relay is needed in the ladder diagram to latch the circuit on. In the PDP-14 however, the output to energize the solenoid may itself be tested as an input. Thus, as shown in Example 5 Simplified (following), output Y1 may be eliminated and the control relay may be removed. The test of output Y5 is equivalent to testing output Y1 in this case.

EXAMPLE 5, SIMPLIFIED

Control Function

If pushbutton 4 is not ON and limit switch is actuated, and either pushbutton 5 is ON or solenoid B is ON already, then solenoid B is engated.

Boolean Equivalent

Inputs:	Output:
4PB (n/o) $\overline{X4}$	SOL B Y5
5PB (n/o) X5	
2LS (n/o) X12	

Equation: $Y5 = \overline{X4} * (Y5 + Y1) * X12$

PDP-14 Wiring (Figure 5-12)

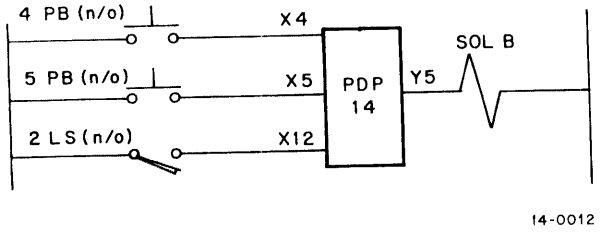


Figure 5-12

PDP-14 Program

Location	Content	Comments
35	TXN 4	If either X4 is ON or X12 is OFF, the output should be OFF.
36	TXF 12	
37	JFN 43	

Location	Content	Comments
40	TXN 5	Either X5 or Y5 itself must be ON for Y5 to
41	TYN 5	be set ON.
42	JFF 45	
43	SYF 5	Simply set output 5 ON or OFF directly,
44	SKP	without using intermediate control realy.
45	SYN 5	

TIMING CONSIDERATIONS IN A PDP-14 PROGRAM

For most PDP-14 applications, timing is of little or no consequence. For other applications; however, timing must be considered. For example, the maximum amount of time should be determined which could elapse before a momentary contact closure is checked. Otherwise, in lengthy PDP-14 programs, it could happen that the input is missed. By considering timing, this condition will be prevented.

It should be noted that a PDP-14 program is a collection of instruction sequences, where each sequence controls a single output. An average time of 20 microseconds is required to execute one PDP-14 instruction. Using this average time, the following program execution times are determined:

1K program	20 ms
2K program	40 ms
3K program	60 ms
4K program	80 ms

However, in the case of a normal PDP-14 program (e.g., as translated by BOOL-14) it is impossible to execute all instructions. For example, if the SYN instruction is executed, the SYF instruction could not be executed.

A second example is that if an output equation is of the form:

$$Y5 = Y0 * (X1 + X2 + X3),$$

and it is compiled by BOOL-14, the inputs X1, X2 and X3 are never tested whenever Y0 is OFF.

A third example of instructions which could not be executed is if an output equation is of the form:

$$Y6 = X3 * X52 + X4 * X12 + X7 * X53,$$

The last four inputs will not be tested when the first two are both ON. If either of the first two inputs are OFF, and the third and fourth inputs are ON, the last two will not be checked. Thus, all inputs will be tested only when Y6 is set to be OFF, or when Y6 is set ON by the last two inputs.

The conclusion of the previous argument is that on the average no more than 75% (and often only 50%) of the instructions are executed. Better estimates of program execution time are given in Table 5-7.

Table 5-7
Program Execution Times

	Assumption	
	75%	50%
1K program	15 ms	10 ms
2K program	30 ms	20 ms
3K program	45 ms	30 ms
4K program	60 ms	40 ms

Table 5-7 indicates, for example, that if a 3K program is being used, assuming that 50% of the instructions are being executed, each input must be present for 30 ms in order to guarantee that the input will be acknowledged. If 30 ms is not fast enough for some inputs in this particular application, the equation could be checked more than once, thereby decreasing the time between input samplings. To determine the choice of execution assumption (50% or 75%), consider the following analysis:

a. How much memory is not used (i.e., it is filled with NOP's) and is jumped around?

b. Are there many subroutines in the program? These effectively increase the memory size when they are executed.

c. Are there any sizable routines (e.g., start-up, shut-down, or manual) which are not executed under normal conditions?

d. Are the majority of equations short (1-5 elements) or long (6 or greater)? The longer the equation the less probability that all instructions will be executed.

Very detailed analyses of program execution time require the nominal execution times for each PDP-14 instruction. Although some of the following instructions are not actually introduced until Chapter 9 of this manual, they are included here for conciseness. The times given are nominal and are subject to a ±30% variation.

Operation Code	Instructions	Time (μs)
0	SKP, JMR, EEM, LEM	4.0
1	TYF, TYN	19.5
2	TXF, TXN	19.5
3	SYF, SYN	19.5
4	JMP, JMS, TRM	4.5
5	JFF, JFN	4.0
6	SKE, SKZ	4.0
7	TXD, TYD	19.5

CONCLUSION

The instructions introduced in this chapter are the instructions used by the PDP-14 to solve control functions. Although programs may be written in equation form and compiled with BOOL-14 (as described in Chapter 6), BOOL-14 actually translates the equations into the instructions introduced in this chapter. Thus, when the user debugs his program with SIM-14, he must be familiar with these instructions.

The user may choose to write his program in instruction format, using the symbolic features of PAL-14 to make addressing, input and output assignment, and other programming tasks easier. Again, however, the program which actually operates in the PDP-14 has absolute location and input/output assignments, as described herein. Thus, the user must be familiar with the instruction format given in this chapter.

Instead of using BOOL-14 or PAL-14, programs which are in the format of this chapter (as illustrated by the preceding examples) may be typed into SIM-14, the PDP-14 simulator and debugging aid. The procedure for directly typing programs is presented in Chapter 8. However, the user who chooses to directly compose programs with SIM-14 sacrifices many features of BOOL-14 or PAL-14 which greatly ease the programming task.

CHAPTER 6

BOOL-14 CONTROL EQUATION TRANSLATOR

BOOL-14 translates control equations, which are written with symbols similar to Boolean notation, into the PDP-14 machine code instructions given in Chapter 5. A series of these equations constitute a PDP-14 program. The program of control equations is prepared using the Editor and a "source program paper tape" is generated. Use of the Editor allows the source program to be corrected and revised easily during the stages of program development. BOOL-14 reads the source program paper tape and translates (compiles) the equations into a PDP-14 machine code program containing the instructions introduced in Chapter 5.

BOOL-14 will also permit equations to be typed on the keyboard and compiled. This feature is useful for quickly checking an equation. However, complete programs should be prepared in paper tape form.

The programming procedure used with BOOL-14 is diagrammed in Figure 6-1. The importance of correcting errors in the source program with the Editor cannot be overemphasized. The user should maintain a current and correct version of the program in equation form throughout the life of the system. Failure to do so may result in lost hours when changes must be made to the system and an accurate representation of the program is not available.

BOOL-14 STATEMENT

The body of BOOL-14 source programs is a series of control equations, such as the following:

```
Y10 = (X1 + X2) * X3↵
```

Each character in the control equation has special meaning to BOOL-14. For example, the non-printing carriage return character (↵) is the signal to BOOL-14 that it has read the complete source statement. The acceptable equation characters and their use is given in Table 6-1. This table includes all legal BOOL-14 characters. Functional descriptions of some characters are given later in this chapter.

The characters SPACE, TAB, LINE FEED, and FORM FEED may be included in BOOL-14 programs to format the source program. They are ignored during processing by BOOL-14. Blank leader/trailer (null) tape is also ignored by BOOL-14. All characters other than these formatting characters and the characters given in

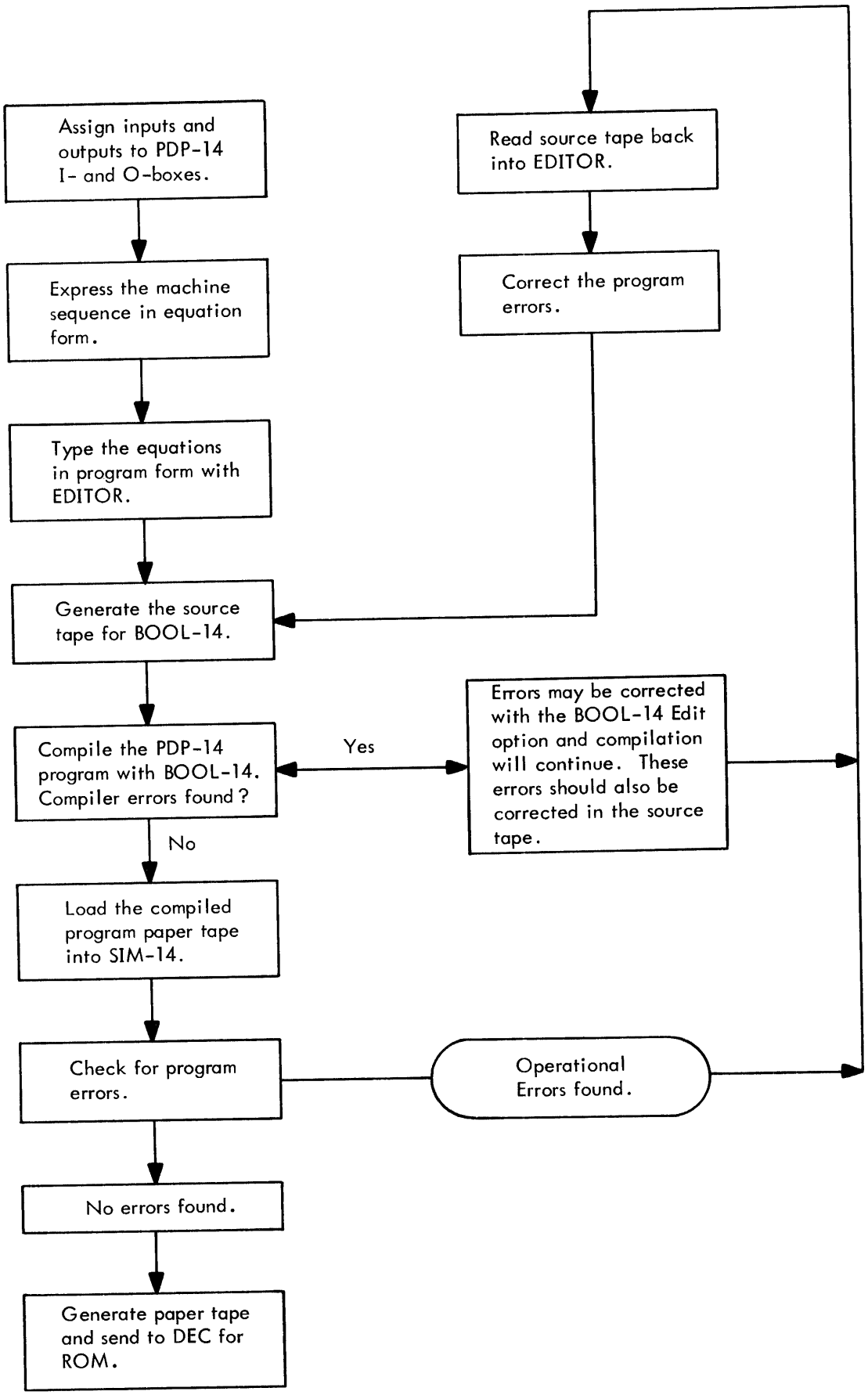

Assign inputs and outputs to PDP-14 I- and O-boxes.
Express the machine sequence in equation form.
Type the equations in program form with EDITOR.
Generate the source tape for BOOL-14.
Compile the PDP-14 program with BOOL-14. Compiler errors found?
Yes
Errors may be corrected with the BOOL-14 Edit option and compilation will continue. These errors should also be corrected in the source tape.
No
Load the compiled program paper tape into SIM-14.
Check for program errors.
Operational Errors found.
No errors found.
Generate paper tape and send to DEC for ROM.
Read source tape back into EDITOR.
Correct the program errors.

Figure 6-1

Table 6-1 are illegal and cause errors if they are included in the statement portion of the BOOL-14 source program. All illegal non-printing characters are replaced by a "?" in the BOOL-14 listing.

Table 6-1
BOOL-14 Equation Characters

X	Input indicator
Y	Output indicator
0-7	Identify specific inputs and outputs
=	Equation indicator
+	Boolean OR operator
*	Boolean AND operator
/	Boolean NOT (negates a single variable or a variable group)
(,[	Start of a variable group
),]	End of a variable group
R	Repetitive equation indicator
Z	Repetitive variable indicator
0-9	Identify specific repetitive equations or variables
:	Indicates start of repetitive equation
;	Comment indicator
$	Line break indicator
.	Control statement indicator
(RETURN)	End of statement (often represented by " ❯ ")
(TAB)	Formatter, converted to 8 spaces or less
CTRL/D	Search for error character in Editing Option
CTRL/U	Delete complete statement line, when typed from keyboard
(RUBOUT)	Delete one character to left, when typed from keyboard.

Input and Output Specification

The numbers which follow the "X" or "Y" are input and output numbers respectively, as obtained from Appendix A charts. Particular input and output numbers are determined by the selection of the input (or output) box and are octal numbers between 0 and 377. The state of an output is determined by the state of inputs and outputs. That is, a single output number (e.g., Y17) is the left hand member of an equation, while the right hand member of the equation is a combination of inputs and outputs. These inputs and outputs are combined as variables of an equation using the following operators.

Operators

The BOOL-14 operators are * (Boolean AND), + (Boolean OR, inclusive), and / (Boolean NOT). These operators combine input and output values to determine the value of an output. Thus the equation:

```
Y17 = Y17 * /X4 + X23
```

means that output 17 is set ON if (a) Y17 is already ON and X4 is not ON, or (b) X23 is ON; otherwise, Y17 is set OFF.

Note that the equation specifies the ON and OFF conditions for an output. Namely, the output is set ON if the conditions are met (true) and it is set OFF otherwise (the conditions are false). The fact that an output is set OFF when the conditions are not met is not always expressed explicitly, but it is always implied in a BOOL-14 equation.

The NOT operator (/) may negate the sense of single variables or variable groups. It may also negate the left member of an equation, in which case the equation specifies the set OFF conditions rather than the set ON conditions. For example:

```
/Y21 = X12 + /X31
```

means that output Y21 is set OFF if (a) X12 is ON, or (b) X31 is OFF; otherwise output Y21 is set ON.

Variable Groups

Input and output variables may be grouped with either parentheses or square brackets. (The square bracket "[" is SHIFT/K and "]" is SHIFT/M on the Teletype keyboard.) For example:

```
Y12 = (X12 + X13) * [X27 + Y15]
```

There is no difference in the treatment of square brackets and parentheses by BOOL-14. They are available for the user's convenience in distinguishing between variable groups. The only requirement by BOOL-14 is that these symbols must be used in pairs. Any grouping which is opened with a square bracket must also be closed with a square bracket.

For example:

```
Y12 = (X12 + X13] * [X27 + Y15)
```

will generate an error because both groupings are closed with the wrong bracket type.

However, the equations:

```
        Y12 = (X12 + X13) * [X27 + Y15]
and     Y13 = (X22 + Y17 * [X42 + X3]) * X15
```

are in proper form.

Statement Continuation

Extremely long equations which will not fit on one line in the source program may be broken into two or more parts using "$" as the statement continuation character. Each incomplete line is terminated with a dollar sign. If the equation is being typed directly to BOOL-14 from the keyboard, BOOL-14 will automatically provide a carriage return after the "$". If the equation is part of a source program paper tape generated with the Editor, the user has typed a carriage return after the "$" on the source tape. Whenever a dollar sign precedes a carriage return, BOOL-14 will not treat the carriage return as an end of statement indicator. It simply provides a new input line and accepts the additional characters until it encounters a carriage return which is not preceded by a dollar sign.

Example:

```
Y21 = /(X1 + X2 + X3) * ( (X4 + [X5 * X6]) * /X7$
) + Y12 * (X27 + X5 +/X21)
```

The dollar sign/carriage return combination may be anywhere within the statement line; otherwise, all other statement rules are followed.

Comments

Program comments are text lines included in the BOOL-14 source program to clarify the function of an equation or to document the program in general. Comments are merely typed by BOOL-14 as they appear in the source program. The comments have no effect upon the machine code which is generated for an equation.

BOOL-14 comments must be preceded by a semicolon (;). When BOOL-14 encounters a semicolon, it treats all characters to the right as a comment. Any Teletype character may be included in the comment except dollar sign ($). This character may be used to write comments on more than one line. Any characters to the right of the $ will be ignored by BOOL-14.

The following are comment examples:

```
Y15 = X1 + X2 * (X3 + /X4)              ;SOLA = PB1 + PB2 * (LS13$
+ / LS14)
;LS14 TRIPPED WHEN SLIDE IS FULL FORWARD.
```

Notice that the first comment is broken into two lines using the dollar sign.

NOTE

BOOL-14 may handle approximately 160 input characters (i.e., equation and comment) in one statement. If this maximum is exceeded, BOOL-14 will type the message:

BUFFER OVERFLOW

The remaining characters in the statement will be typed but are not part of the statement compilation. If the buffer overflow occurs within a comment it should be ignored. If it occurs within an equation, the equation must be divided using a Z-function as described later in this chapter, and recompiled.

SUBROUTINES AND STORAGE OUTPUTS

Subroutines serve two functions in BOOL-14 equations:

1. Z-functions solve for an intermediate value or result. A Z-function does not directly turn an output ON or OFF; rather, it is part of a larger equation which does.
2. R-functions solve for an output state. R-functions are the same as the normal output equation except that provision is made to solve for the output more than once on each pass through the program.

A BOOL-14 program may contain a maximum of 64 subroutines (viz., R- and Z-functions).

Intermediate Results

There are many cases in PDP-14 programs which require the "storage" of an intermediate result. For example, one set of inputs energizes or de-energizes control relay 1; a second set of inputs energizes or de-energizes control relay 2; and a third set of inputs energizes or de-energizes control relay 3. These control relays are then used in various combinations with other inputs and outputs to control solenoids A, B, and C. There are three possible approaches to such control functions:

Complete coding	Control equations are written which substitute the variables which determine the state of the control relay for the control relay itself. Thus, the equation for each solenoid will separately test the control relay inputs to determine the state of the solenoid. The actual control relays are eliminated from the system.
Z-functions	A Z-function is written to solve for the state previously represented by a control relay. (No PDP-14 output is set by the Z-function.) The solenoid equations may then contain Z-functions as variables. Only one set of PDP-14 instructions is needed to solve for each Z-function. These same instructions are used to solve for every output which contains the Z as an equation variable.

Storage outputs

A normal output equation is written which sets an output in an S-box ON or OFF. These outputs may then be used as variables in other equations.

NOTE

Before proceeding, the reader should review the section on Subroutine Jump Instructions, pages 5-11 and 5-12.

Z-functions

Z-functions are repetitive variables which are solved whenever they are needed in a normal output equation. They do not set an output to record the result of the testing; instead the result is recorded by setting the PDP-14 TEST flag ON or OFF. Z-functions usually represent values which must be solved in many different equations. For example, suppose the following equations were compiled by BOOL-14:

```
Y1 = X1 * (X21 + X13 * /X42 + Y17 * X11)
Y2 = X2 * (X21 + X13 * /X42 + Y17 * X11)
Y3 = X3 * (X21 + X13 * /X42 + Y17 * X11)
Y4 = X4 * (X21 + X13 * /X42 + Y17 * X11)
```

If given this set of equations, BOOL-14 would generate PDP-14 machine code to test each variable for each equation. Notice that many of the instructions will be the same for all four outputs, namely those instructions which test the state of the variables within parentheses.

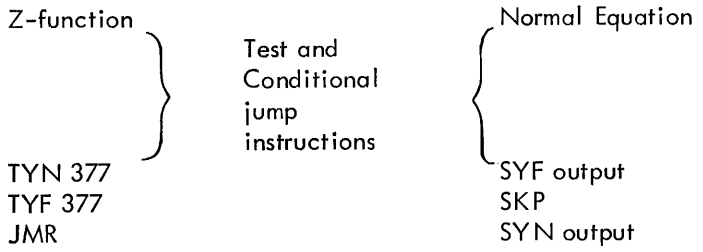
The same machine control function could be represented by the following group of equations with a considerable saving in the number of memory locations required for the program.

```
Z1 = X21 + X13 * /X42 + Y17 * X11
Y1 = X1 * Z1
Y2 = X2 * Z1
Y3 = X3 * Z1
Y4 = X4 * Z1
```

The new form is analagous to a control relay in a ladder network which represents the combined state of a group of inputs (Z1). The contacts from the relay are then included in the circuits for solenoids, motor starters, etc. The number following Z is arbitrary and may be any decimal number between 0 and 2047. The Z function must only be defined once. It may be referenced as many times as necessary.

If given this group of equations, BOOL-14 will generate the series of instructions to establish Z1 as a subroutine. The Z1 instruction will test the inputs just as a normal output equation would; however, no actual output will be set. The subroutine will set the TEST flag ON if the equation is true; otherwise it will set the TEST flag OFF. BOOL-14 then tests the state of the TEST flag to determine the result of the subroutine.

The subroutine instructions for a Z-function are terminated in the following way (compared to the termination of a normal equation).



If the result of the Z-function is not true, BOOL-14 generates a JFF or JFN instruction that jumps past the TYN 377 and TYF 377 to the JMR, thereby returning with the TEST flag OFF. If the result of the Z-function is true, the TYN 377 and TYF 377 instructions will be executed. Since output 377 must be in one of the two states, one of the tests must be true and the TEST flag will thus be set before the JMR is executed.

The instructions to solve the equations which contain Z-functions differ greatly from the original equations. In place of the instructions to test the variables in the Z-function, BOOL-14 will generate a JMS (jump to subroutine) to the beginning location of the subroutine that solves the Z-function. The subroutine will return to the main program with the result reflected in the TEST flag. The main program will test the result with a JFF or JFN instruction. Thus the JMS instruction serves a purpose similar to a test instruction.

Instead of testing a single variable, the logical combination of a group of variables is tested and the result is reflected in the TEST flag.

The PDP-14 will only accommodate one level of subroutines. This means that a subroutine may not contain a JMS instruction. In terms of BOOL-14, this means that a Z-function may not contain another Z-function as a variable of its equation. If a second level is needed, it must be written as a storage output and tested with a TYN or TYF instruction in the subroutine, as described later in this chapter.

R-functions

R- functions (repetitive equations) are subroutines which do set an output ON or OFF (as opposed to Z-functions which set the TEST flag). Characteristically, R-functions are outputs which must be checked very often and cannot wait for a complete memory cycle. The state of these outputs could be determined simply by compiling the complete instruction sequence to solve the equation in several places in the source program. This technique is inefficient if the same equation is solved more than once, however. Defining an R-function causes BOOL-14 to generate the equation as a subroutine. Whenever the output state is to be determined, BOOL-14 generates a JMS (jump to subroutine) instruction to the R-function to set the output ON or OFF. Thus the instructions to solve the equation are stored in only one place in memory, but they are executed as many times as necessary.

An R-function is defined in the following form:

```
Rn: Ym = X1 * ....
```

where n is any decimal number between 0 and 2047. Ym specifies a specific output to set by the subroutine.

The reference to the subroutine is made by writing the R-function designation. For example, the following statement defines an R-function to set output Y53.

```
R81: Y53 = X32 * (X41 + X12 * Y13) + X21
```

BOOL-14 generates the machine code instructions to solve the equation for Y53 in the same form as a normal output equation. After the set output instructions, BOOL-14 compiles a JMR instruction:

```
.
.
.
SYF 053
SKP
SYN 053
JMR
```

Whenever the R-function should be solved in the main program, the user writes the following statement in the source program:

```
R81 )
```

BOOL-14 will compile a JMS instruction to the subroutine which solves for Y53.

As seen in the following examples, all R- and Z-functions must be defined at the beginning of the source program and not following an output equation. Any R- or Z-function which is defined following an output equation (e.g., Y1 = X4) will generate an error. Each R- or Z-function must have a unique decimal identifying number.

Subroutine Examples

```
R13: Y6 = X2 + X4                ; DEFINES THE SUBROUTINE
     .
     .
     .
R13 )                            ; REFERENCES THE SUBROUTINE, EQUIVALENT
                                 : TO WRITING Y6 = X2 + X4
```

Z8 = X2 + X4 * Y7 Z9 = X31 + X4 + X12 Z8 = X21 * Y7	Error because the Z8 function has already been defined.
Y1 = X21 + R8	Error because R-functions cannot be variables in equations. (Only Z-functions may.)
R21: Y10 = Z1 * X17	Error because an R-function may not contain a Z-function as a variable. (A subroutine may not jump to another subroutine.)
Z1 = X1 + X31 * Y21 Z2 = X3 * (X21 + X31) + Z1 . . . Y12 = X12 * (X13 + X15) R13: Y12 = X15 + Y13	Error because the equation for a Z-function may not contain another Z-function. (A subroutine may not jump to another subroutine.) Error because definition of all R- and Z-functions must precede any output equations.

Storage Outputs

Storage outputs are outputs in an S-box of the PDP-14. They are programmed exactly like other Y-functions. The only difference is that the output does not drive a solenoid, motor starter, or such. Once a storage output is set, however, it may be tested with the TYN and TYF instructions of the PDP-14.

The advantage of storage outputs is that the instructions to solve for the result as represented by a storage output are executed only once for each pass through the program. The storage output itself may then be tested whenever it is needed. However, subroutines must be executed every time the result is needed in an output equation.

Storage outputs may also be tested by a subroutine. Thus if one intermediate result is a function of another, a storage output could be used to record the second result. As previously stated, a subroutine may not jump to a second subroutine.

A storage output must be used in place of a subroutine whenever the control function has itself as an element of the equation, i.e., if it is a latched function.

The equation to establish a storage output is of the form:

Y33 = X23 * X21 + Y17

where output Y33 is contained in an S-box which is connected to O-box socket "B". The result is tested by including the output as a variable in an equation.

Y16 = Y33 + (X3 + X5 + X7)

BOOL-14 will generate a TYN 033 instruction as required to solve the equation for Y16.

CONTROL STATEMENTS

Control statements are directions to BOOL-14 which affect the machine code instructions generated during compilation. Specifically, control statements affect the allocation of memory locations to the PDP-14 program instructions generated by BOOL-14 and mark the end of source program tapes.

End of Program - .END or .ENDN

Each BOOL-14 source program must be terminated with an .END or .ENDN statement to indicate the end of compilation. The .END statement completes the BOOL-14 program by compiling a JMP instruction. The address of the JMP is specified by the user such that the program is a "closed loop". The address of the JMP is written following the .END. Any octal number between 0 and 7777 may be specified as the value of the .END statement, or a default value of 0 is assumed. The value is often the same as the value of the .LOC statement.

The .ENDN (end-no jump) statement may be used to terminate an incomplete program where no JMP instruction is needed. This statement marks the end of the source program but generates no machine code instruction.

.END and .ENDN Examples

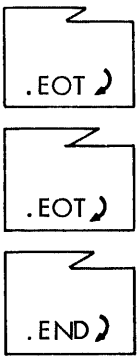
Statement	Instruction Generated
.END 50	JMP 50
.END 2000	JMP 2000
.END	JMP 0
.END 2008	error-octal numbers only
.ENDN	non generated
.ENDN 50	error-no value allowed

End of Tape - .EOT

The BOOL-14 source tape may be physically segmented using the .EOT (end of tape) statement to terminate all but the last paper tape in a series. The last segment must be terminated with an .END or .ENDN statement.

When BOOL-14 encounters an .EOT statement it will type "EOT" and halt the PDP-8 system allowing the user to load the paper tape reader with the next tape. Pressing the PDP-8 CONT switch permits the compilation to proceed.

For example, if the program is on three tapes, the end of the tapes will be



Tape 1 is terminated by an .EOT statement

Tape 2 is also terminated by an .EOT statement.

Tape 3 is terminated by an .END, .END with value, or an ENDN statement.

Memory Allocation

BOOL-14 compiles all subroutines (R- and Z-functions) as the first elements of the program. This is necessary to allow BOOL-14 to generate JMS instructions to these subroutine locations from the main program. BOOL-14 will place immediately before the subroutines a JMP to the start of the main program which follows the subroutines.

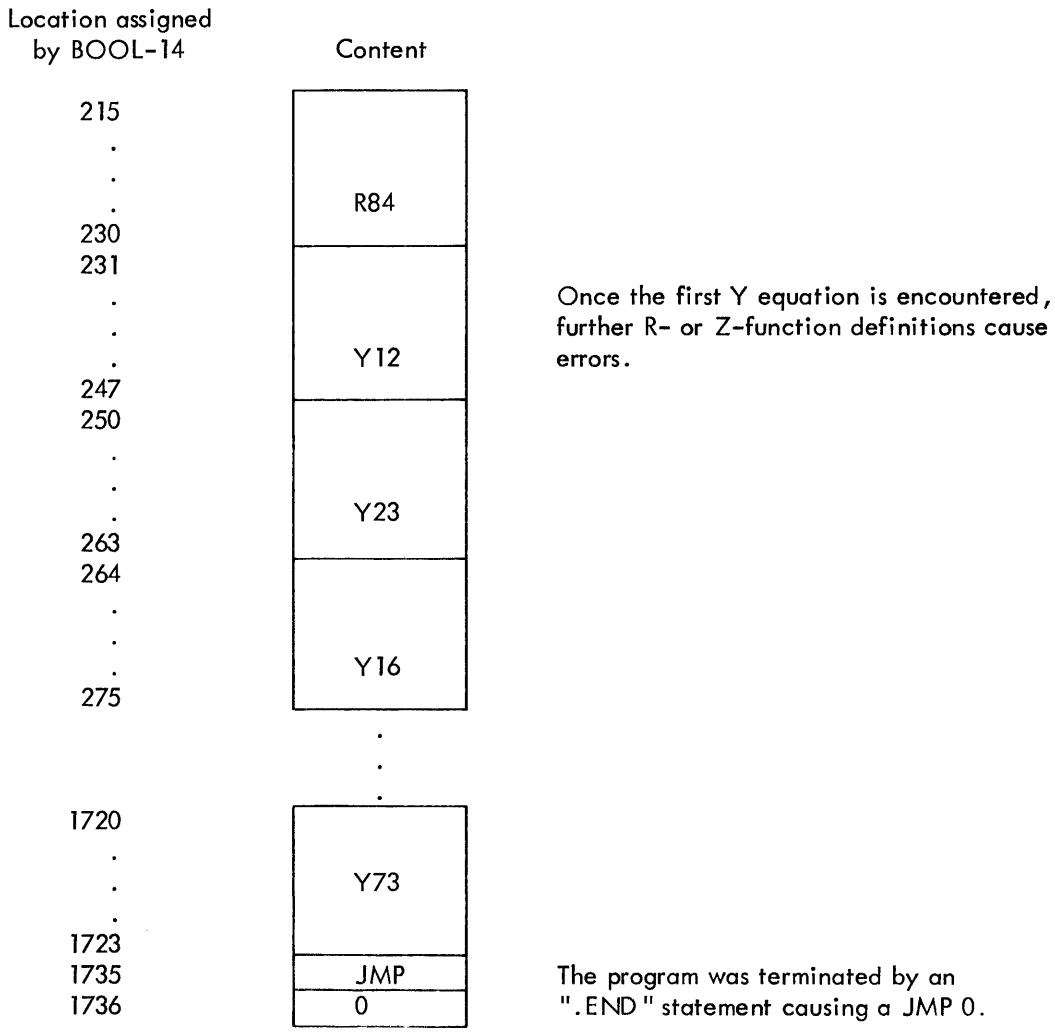
An example of memory assignment is:

Location assigned by BOOL-14	Content
0	JMP
1	231
2	
.	
.	
.	Z15
15	
16	
.	
.	
.	R29
31	
32	
.	
.	
.	Z5
57	

The JMP instruction goes to the first instruction of the first output equation (in this case Y12).

The Z- and R-function definitions may be in any order and may be assigned any identifying number less than or equal to 2047.

Example (Cont)



BOOL-14 will assign generated machine code instructions to consecutive memory locations beginning with location 0. The first instruction of a new equation is assigned to the next sequential memory location after the previous equation. This sequential assignment does not provide room for changes with SIM-14. The .LOC, .FIXS and .VARS control statements, described later, permit the user to start the program at a location other than 0 and to leave space after each instruction sequence.

When compiling an equation, BOOL-14 will check the number of remaining locations on the current page. If there are not enough PDP-14 memory locations for the machine code instructions to solve the equation, BOOL-14 will compile NOP's in the remaining locations of the current memory page, and start the equation in the first location of the next page. The NOP's thus generated are not typed as part of the listing.

The user should remember that PDP-14 program execution starts in location 0. The content of the locations beginning with 0 must control "start up", either with instructions to initialize the system, or with a JMP to an initialize or start up routine.

Start of Program - .LOC

The initial location of a program is specified by the .LOC (location) statement. BOOL-14 will generate machine code instructions for the locations beginning with the value specified by .LOC. For example:

```
.LOC 50
Y1 = X1 + X2
```

yields the compiled instructions:

```
0050    2401    TXN  001
0051    2402    TXN  002
0052    5455    JFN  055
0053    3001    SYF  001
0054    0344    SKP
0055    3401    SYN  001
```

Note that the instructions begin in location 50. If no .LOC statement were given, a .LOC 0 would be assumed and the first instruction would be placed in location 0.

Only one .LOC statement is permitted in a BOOL-14 program. It must precede the first subroutine or equation of the program. The value of .LOC may be any octal number between 0 and 7777. (Remember that the memory units are: MB1, 0 to 1777; MB2, 2000 to 3777; MB3, 4000 to 5777; MB4, 6000 to 7777.)

Spacing Between Equations - .FIXS or .VARS

BOOL-14 is directed to compile a specified number of NOP (no operation) instructions after equations by the .FIXS (fix spacing) and .VARS (vary spacing) control statements. The NOP instructions allow for program changes in SIM-14 during debugging. For example, the statement:

```
.FIXS 5
```

will result in five NOP instructions at the end of all subsequent instruction sequences. The value of the .FIXS statement may be any decimal number between 0 and 2047. As many .FIXS statements as needed may be placed in a program.

The .VARS (vary spacing) control statement only affects the equation which immediately follows it and overrides for the one equation any .FIXS statement currently in effect. The .FIXS statement is reinstated after each .VAR statement and remains in effect until overridden by another .VARS statement or replaced by a new .FIXS statement.

The value of .VARS is a decimal number between 0 and 2047. The .VARS statement may be used to suppress the output of NOP instructions after an equation by specifying a value of 0:

```
.VARS 0
```

Spacing Examples

Equation and Control Statements	Number of NOP's after The Equation
Z1 = (X27 * Y0) * X13 + X4	none
.FIXS 5	
R12: Y1 = X23 * X14 + X0	5
.VARS 2	
Y1 = X23 * Z1	2
R12	5
.FIXS 10	
Y2 = X27 * X3 + X30 * X2	10

The NOP instructions are not part of the compiler listing, although they are punched on the binary output tape when binary is requested.

MONITORING PROVISIONS IN BOOL-14

Using control statements, BOOL-14 can be directed to automatically supply monitoring instructions if the operation of the PDP-14 system is to be monitored by an external computer. The instructions used and the techniques involved are fully described in Chapter 9. The procedure involved in BOOL-14 to generate these instructions is described below.

BOOL-14 may be directed to add instructions to monitor the transition in the state of an output. Specifically, instructions may be added to monitor transitions to ON (make), transitions to OFF (break), or both, for any PDP-14 output, storage output or R-function.

The choice of the TYD or TRM instruction for monitoring is left to the user. The TYD instruction outputs a specific 12-bit word to the monitoring computer; the word is determined by the PDP-14 (see Chapter 9). The TRM is a two location instruction which outputs a 12-bit word specified by the user.

Monitor Transitions to ON - .MN

The .MN control statement will cause BOOL-14 to add instructions to the first output equation which follows it to monitor transitions from OFF to ON. These instructions will load a 12-bit word into the PDP-14 output register whenever the output goes from OFF to ON. The output register is not loaded as long as there is no transition, or when the transition is from ON to OFF.

Example:

```
.MN
Y5 = /X2 * (X3 + Y5)
(the instructions generated will include a TYD 005)
```

Only the immediately following equation will contain the sequence of monitoring instructions. The instructions generated are described in Chapter 9.

If the output is to be monitored with a TRM instruction, the user must supply an octal number (four digits or less) after the ".MN".

Example:

```
.MN 4123
Y5 = /X2 * (X3 + Y5)
```
(the instructions generated will include a TRM 4123)

If the user supplies less then four digits after the control statement, BOOL-14 will assume leading zeros.

Monitor Transitions to OFF - .MF

The .MF control statement allows the user to monitor the transition in an output from ON to OFF. Instructions are added which will load a 12-bit word into the output register whenever the output is changed from ON to OFF. The output register is not loaded as long as there is no transition or if the transition is from OFF to ON.

Example:

```
.MF
Y6 = X3 * (X4 * X5 + /X10 * X11)
```
(the instructions generated will include a TYD 006)

The .MF control statement above only has effect for the first equation which follows it (i.e., Y6).

If the output is to be monitored using a TRM instruction, the user must supply an octal number (four digits or less) after the ".MF".

Example:

```
.MF 103
Y6 = X3 * (X4 * X5 + /X10 * X11)
```
(the instructions generated will include a TRM 0103)

Monitor All Transitions - .MFN

The .MFN control statement is used to monitor all transitions in output state (viz., both the ON to OFF and OFF to ON). Instructions will be added which will load a 12-bit word into the PDP-14 output register on all transitions in the state of the output. The output register is not loaded as long as the output remains ON or OFF.

Example:

```
.MFN
YN = X21 + X15 * Y5
```

(the instructions generated will include two TYD 007's)

If the output is to be monitored with TRM instructions, the user must supply two octal numbers after the ".MFN"

Example:

```
.MFN 2007, 1007
```

(the instructions generated will include:

TRM 2007 - executed on transition to OFF
TRM 1007 - executed on transition to ON)

It is important to remember that the first number is used in transitions to OFF (.MFN) and the second number is used in transitions to ON (.MFN).

Monitoring Input Transitions

As stated above, BOOL-14 allows monitoring of output states. If the user wishes to monitor the transitions in state of an input, he must assign a storage output to record the previous state of the input. He may then monitor the transitions in the state of the storage output to provide monitoring information concerning the input.

Example:

```
.MF
Y40 = X1
```

(the instructions generated include a TYD 040)

In the above example, the transitions in state of output Y40 are used to monitor the transitions in state of input X1.

Table 6-2
Summary of Control Statements

Command	Control Operation
.LOC 250	Start the program compilation at PDP-14 location 250.
.FIXS 5	Leave 5 (decimal) NOP's after all equations beginning with the next equation.
.VARS 30	Ignore any .FIXS statement currently in effect and leave 30 (decimal) NOP's after the next equation only.

Table 6-2 (Cont)
Summary of Control Statements

Command	Control Operation
.MN	Monitor the next equation for transitions to the ON state.
.MF	Monitor the next equation for transitions to the OFF state.
.MFN	Monitor the next equation for all transitions
.EOT	End of tape - interrupt the compilation while a new tape is loaded.
.END	End of program - compile a JMP 0 instruction.
.END 50	End of program - compile a JMP 50 instruction.
.ENDN	End of program - do not compile any JMP instruction.

BOOL-14 OPTIONS

The options available to the user are requested through a series of four queries as described below. These queries are typed to start the BOOL-14 compilation process. If the user types other than a legal response to a query, BOOL-14 will retype the query.

Example:

```
*SRC-KLH? N )        N is not a legal response to the SRC query. BOOL-14
*SRC-KLH?            retypes the query.
```

If the user types more than one character in response to a query, BOOL-14 will recognize only the last character typed before the carriage return. Thus if a wrong response is typed, the user may correct it by typing the correct character prior to the carriage return.

Binary Output

The binary paper tape output from BOOL-14 which is input to SIM-14 is requested in response to the query:

```
V2
*BIN-NLH?
```

Where N, L, and H are the possible user-supplied responses with the following meanings: The V2 is the BOOL-14 version designation which is always typed before the BIN query.

N	No binary output is wanted.
L	The binary paper tape is requested on the low-speed paper tape punch associated with the Teletype unit.
H	The binary paper tape is requested on the high-speed paper tape punch.

The user types one of the possible responses followed by a carriage return. For example, the user may not want a binary tape if he expects errors in his program. By typing N, he may compile the program and get an error listing without wasting time generating an incomplete binary tape. The other two possible responses request the binary output and specify the device to be used.

If the user requests binary output from a 4K PDP-8 system and the compiled program requires more than 1K of PDP-14 memory, the message "BINARY OVERFLOW" is typed and no binary will be punched. The compilation will continue, however.

Compiler Listing

The compiler listing, which contains error messages and generated machine code instructions for solving the equations of the source program, is output according to the following query:

```
*LST - NYW?
```

where N, Y, and W have the following meanings:

N	No listing output is requested. BOOL-14 will only type the lines which contain errors and the applicable error number.
Y	The complete listing of source statements, compiled machine code instructions, error messages, and any equation reductions will be typed on the Teletype.
W	The complete listing will be typed exclusive of the reduced form of equations.

The user types one of the above responses followed by a carriage return. If only an error listing is wanted, the user types N. BOOL-14 will type only those lines of the source program which contain errors. For example, BOOL-14 might type:

```
Y10 = X27 (Y3 + X21) * X4          Note that an operator is missing
E011                               after "X27".
```

The user could then correct this and any other error detected during compilation and generate an updated and correct source program tape with the Editor. When all the errors have been corrected, the program is recompiled with BOOL-14 and a complete compiler listing should be requested.

The Y response causes BOOL-14 to generate the complete compiler listing. All lines of the source program are typed (followed by an error number if the line contains an error). If there is no error, the generated PDP-14 machine code instructions are listed below the equation.

BOOL-14 converts some equation forms to equivalent equations which it can solve more readily. In these cases BOOL-14 types the converted form to assist the user in debugging his program. This reduction is to a form in which the NOT (/) operator applies to single variables only. For example:

```
Y21 = / (X2 + Y31)
CONVERTED TO:
Y21 = (/X2) * (/Y31)
```

BOOL-14 reduces the equation and types the converted form, followed by the machine code instructions to solve the equation.

If the user does not find the converted equation form useful, he may suppress the converted form output by responding with the letter W (listing without converted forms) to the *LST-NYW? query.

The listing will contain the PDP-14 generated machine code instructions following each equation. These instructions are typed in three columns. The first column is the octal absolute address in PDP-memory where the instruction will be stored. The second column is the numeric form of the instruction. The third column is the symbolic form of the instruction. For example:

```
Y1=X1+(X32*X3+Y12)

0000 2401 TXN 001
0001 5411 JFN 011
0002 2032 TXF 032
0003 2003 TXF 003
0004 5011 JFF 011
0005 1412 TYN 012
0006 5411 JFN 011
0007 3001 SYF 001
0010 0344 SKP
0011 3401 SYN 001
```

```
Y13=X2*X4+Y13*(/X4+/X5)

0012 2002 TXF 002
0013 2004 TXF 004
0014 5024 JFF 024
0015 1013 TYF 013
0016 5422 JFN 022
0017 2004 TXF 004
0020 2005 TXF 005
0021 5424 JFN 024
0022 3013 SYF 013
0023 0344 SKP
0024 3413 SYN 013
```

At the end of the program listing, BOOL-14 will type the following messages:

ERROR LINES:	The decimal number of lines in the source program which contained errors.
EDITED LINES:	The decimal number of error lines which were subsequently edited successfully.
PROGRAM BREAK:	The highest absolute PDP-14 memory address used by the compiled program.

Program Source

The source program may be supplied to BOOL-14 from one of three devices: Teletype keyboard, low-speed paper tape reader or, if the PDP-8 system is so equipped, the high-speed paper tape reader. The source is specified in response to the query:

*SRC-KLH?

where the K, L and H are the possible responses with the following meanings:

K	The source program will be typed from the Teletype keyboard.
L	The source program is in paper tape form and will be read with the low-speed reader of the Teletype unit.
H	The source program is in paper tape form and will be read with the high-speed reader.

If the keyboard is selected as the source device, BOOL-14 will type a quotation mark (") when it is waiting for the user to type an equation.

The user then types his equation terminating it with a carriage (the dollar sign ($) may be used to break a statement line into two or more parts as previously described).

When the keyboard is source device, the RUBOUT, CTRL/U and TAB may be used. Characters on the current line of the source program may be erased with the RUBOUT key which will erase one character to the left for each striking of the key up to the beginning of the equation. The RUBOUT key is recorded by BOOL-14 as a back slash (\). CTRL/U* may be used to delete a complete statement line from the source program. The TAB key may be used to format the program. TAB positions are set at 8 character widths. Striking TAB advances the printer to the next Tab position.

Editing

The user may correct error lines in the program by requesting the editing option. The query is:

*EDT-NY?

where the possible responses, N and Y, mean No and Yes respectively.

If the editing option is requested, BOOL-14 will type a ">" after each error line and its error number. This allows the user to retype the equation.

*CTRL/U is formed by holding the CTRL key depressed while striking the letter U.

For example:

```
"Y1 = X1 + X9
E002
>Y1 = X1 + X10                    Retyped by the user
    TXN 1
    TXN 10                        Generated machine code
      .
      .
      .
```

If the source program is read from the low-speed reader and the editing option is requested, BOOL-14 will halt after typing the ">". The user must turn the low-speed reader to STOP, and press the PDP-8 CONT switch. When editing is complete, the reader should be set to START, and compilation will continue.

If the user, having requested the edit option, wishes to ignore the error line and proceed with the compilation, he types a carriage return after the ">" symbol. In this case, the source equation containing the error will be completely ignored.

If the editing option is requested, the user may type CTRL/D* to direct BOOL-14 to type all characters in the equation up to the character which caused the error. The remainder of the equation may then be typed by the user, thereby correcting the error.

If more than one error is contained in a single line of the source program, only the first encountered error will be detected. Repeatedly use the CTRL/D and the editing option will find and correct all errors.

ERROR MESSAGES

The following is a list of the possible errors which are detected and diagnosed by BOOL-14. The error numbers are typed under the equation which caused them in the form "Ennn" where nnn is the error number. All errors stop compilation of the erroneous equation. (BOOL-14 will either proceed to the next equation or, if the editing option is requested, wait for the user to correct the error.) No machine code instructions are generated for erroneous equations, unless they are first corrected.

Error Number	Cause
1	A required number is missing after an input (X), output (Y), Z-function, R-function or control statement. Examples: Y1 = X Y = X1 Y15 = X2 + Y

* CTRL/D is formed by holding the CRTL key depressed while striking the letter D.

Error Number	Cause
1 (Cont)	Z = X3 + Y1 Y17 = X5 * Z R: Y12 = X27 * X1 .FIXS
2	A non-octal digit is encountered after an X, Y, .END, or .LOC.
3	Illegal number error caused by: an input or output number greater than 377; a decimal number greater than 2047 following an R, Z, .VARS, or FIXS; an octal number greater than 7777 following a .LOC or .END statement.
4	The first or second character of an equation is not valid. The first character may be a NOT sign (/), or a "Y". If the first character is a "/", the second character must be a "Y". Examples: B = X1 /G14 = X27 * Y3
5	The equals sign (=) is either missing or not in the proper place within an equation, Z-function, or R-function. Examples: Y1 X1 * X2 Y15/ = X3
6	A variable grouping is not closed with the character type with which it was opened. Examples: [X1 + X2) [X3 + (X5 * X6] + Y7)
7	The character following a negation sign (/) is not valid. Example: Y23 = /11 * X2
8	The character following a left parenthesis or left bracket is not valid. Example: Y10 = (R5 * X2)
9	The character following an AND (*) or OR (+) sign is not valid. Examples: Y11 = X1 + * Y12 = X10 * R1
10	The character following an equals sign (=) is not valid. Example: Y21 = R1

Error Number	Cause
11	The character following a number is not valid. Example: `Y32 = X1/* X2`
12	The character following a right parenthesis or right bracket is not valid. Example: `Y7 = X12 * (X5 + X7) / * X2`

13	A Z-function is contained as a variable in an R-function or Z-function.
	Examples:
	`Z29 = X3 + Z50`
	`R10: Y73 = X1 * Z12`

14	An R-function or Z-function is referenced which has not been defined previously.
15	An equation contains more right parentheses and brackets than left parentheses and brackets. Example: `Y37 = X1 * (X2 * X24 + X25] * X4)`
16	An equation contains less right parentheses and brackets than left parentheses and brackets. Example: `Y42 = X12 * X2 * (X2 * [23 + X17] * X14`

17	A control statement is not recognized.
	Examples:
	`.MZN`
	`.MNF`
	`.SAZ`

18	A control statement contains an illegal character.
	Examples:
	`.M#N`
	`.M.N`

19	There is no control statement after the period (.).

20	The .LOC statement must precede all equations, R-functions and Z-functions.
	Examples:
	`Z12 = X1 * X2`
	`.LOC 50`

21.	A subroutine definition is encountered after at least one equation has been compiled.
	Examples:
	`Y12 = X3 * X15`
	`R12: Y5 = X27 * /Y51`

Error Number	Cause
22	The statement defines an R-function or Z-function which has been defined previously. Examples: Z129 = X25 + X3 * X12 Z129 = Y3 * X21 + X5 * X12
23	The source program contains more than 64 R-functions and Z-functions combined, the maximum that BOOL-14 can handle.
24	The character following a colon (:) is not valid. Examples: R15: Z5 = X27 * X12 R29: R5 = X121 + Y13 * X122

OPTIMUM FORM FOR BOOL-14 EQUATIONS

Although equations may be written in many forms, the optimum form for BOOL-14 equations is achieved by following the following rules. The first two rules affect the number of memory locations used to solve the equation. The other rules affect the amount of time required to execute the program. Adherence to these rules is not mandatory for proper PDP-14 operation. The rules only serve to increase the efficiency and speed of program execution in the PDP-14.

1. Group together all single variables (inputs or outputs) which are combined with the same operator.

   Examples:

   Y1 = X1 + X2 + (X3 * X4 * X5) + Y12 + X15
   should be rewritten:

   Y1 = X1 + X2 + Y12 + X15 + (X3 * X4 * X5) and,

   Y12 = X3 * Y7 * (X4 + X27 + X21) * Y5 * X32
   should be rewritten:

   Y12 = X3 * Y7 * Y5 * X32 * (X4 + X27 + X21).

2. BOOL-14 will generate a test instruction for every variable occurrence on the right side of the equation. Therefore, factor equations to eliminate repeated variable occurrences.

   Examples:

   Y15 = X3 * Y5 + X6 * Y5 + X12 * Y5 + Y17 * Y5
   should be rewritten:

   Y15 = Y5 * (X3 + X6 + X12 + Y17)

3. Where a single variable is ANDed with an expression, preceed the expression with the single variable.

   Example:

   Y6 = (X31 + X4 * X52) * X27
   should be rewritten:
   Y6 = X27 * (X31 + X4 * X52)

4. Group variables and expressions in order of their complexity, with single variables at the beginning of equations and large expressions, and the more complex expressions at the end.

Example:

Y31 + [(X5 * X7 + X3) * X21 + (Y52 * Y12) + X13 * Y17 * / Y5)] * X22
should be rewritten:

Y31 = X22 * [(Y52 * Y12) + (X13 * Y17 * / Y5) + (X3 + X5 * X17) * X21]

5. When expressions of approximately the same complexity are combined using either the OR operator or the AND operator, the expression which represents the set of conditions which is more often true should preceed the other expression.

Example:

Y5 = (X12 * X4 + / X27) + (X21 * (X3 + / X51)) + (Y7 * / X23)
(X12 * X4 + / X27): true 30% of time; (X21 * (X3 + / X51)): true 20% of time; (Y7 * / X23): true 50% of time

This equation should be rewritten:

Y5 = (Y7 * / X23) + (/ X27 + X12 * X4) + (X21 * (X3 + / X51)

6. Try to group subroutines near the end of equations or expressions. Remember that subroutines take longer to test than a single variable.

Example:

Y11 = (Z3 * / X21) + ((Z7 + X4) * X3 * X5)) + X1 * X2
should be rewritten:

Y11 = X1 * X2 + (/ X21 * Z3) + (X3 * X5 * (X4 + Z7))

The foregoing rules are sometimes contradictory. However, the guiding rule is to write the equations in the form which will permit the PDP-14 to determine as quickly as possible if the output is to be set ON or OFF. The rules attempt to limit the amount of testing which the PDP-14 will perform before it is able to make the ON/OFF decision.

SAMPLE BOOL-14 PROGRAM

A BOOL-14 program representing a simple control circuit is given on the following pages. The reader should note the extensive use of comments to indicate the physical input or output which is represented by an X or Y. The control statements .LOC, .FIXS, .VARS, and .END are illustrated. Z-functions and R-functions are included with the regular output equations. The output listing from BOOL-14 follows the program.

```
;BOOL-14 SAMPLE PROGRAM
;COMMENTS TO IDENTIFY THE PROGRAM WOULD APPEAR HERE.
;CONTOL STATEMENTS WHICH AFFECT THE WHOLE PROGRAM FOLLOW.
.LOC25Ø
.FIXS5
;SUBROUTINES PRCEED THE MAIN PROGRAM.
;2CR=/6PB*/7PB*/1PS*/1FS*/5CR
Z2=/X13*/X14*/X6*/X7*/Y11
;SOLE=3CR=(8PB*CRH+CRA)*7CR*8CR*9CR*/3LS*/4LS*/5LS
R1:Y4=(X15*Y7+Y6)*Y12*Y13*/Y14*/X2*/X3*/X4
;SOLF=4CR=/7CR*(CRA*5CR+/9CR+9PB*CRH)
R2:Y5=/Y12*(Y6*Y11+Y14+X16*Y7)
;NORMAL EQUATIONS FOLLOW.
;2LT=CRA=RESET*/CRH*(AUTO+CRA)
Y6=X1Ø*Y7*(X11+Y6)
;CALL TO REPETITIVE EQUATIONS.
.VARSØ
R1
R2
;NORMAL EQUATIONS CONTINUE BELOW.
;3LT=CRH=RESET*(MAN+CRH)
Y7=X1Ø*(X12+Y7)
;SOLA=SOLC=CR1=/6CR*(6PB*7PB*2CR*/1LS*/2LS*5LS*6LS*7CR+1CR)
YØ=/Y3*(X13*X14*Z2*/XØ*/X1*X4*X5*Y12+YØ)
Y2=XØ
;4LT=/6PB*/7PB*(1PS+1FS)
Y1Ø=/X13*X14*(X6+X7)
;5LT=5CR=3LS*/4LS+5CR*8CR
Y11=X2*/X3+Y11*Y13
;SOLB=SOLD=6CR=7CR*(CRA*/5LS*/6LS*(6CR+5CR*8CR)+PB1Ø*CRH)
Y2=Y12*(Y6*/X4*/X5*(Y2+Y11*Y13)+X17*Y7)
;6LT=7CR=4LS
Y12=X3
;7LT=8CR=1LS*2LS
Y13=XØ*X1
;/8LT=9CR=/11PB*(9CR+12PB)*(7CR+8CR)
/Y14=/X2Ø*(Y14+X21)*(Y12+Y13)
;CALL TO REPETITIVE EQUATIONS.
.VARSØ
R1
R2
.END 25Ø
```

```
V1
*BIN-NLH?L
*LST-NYW?Y
*SRC-KLH?L
*EDT-NY?N


;BOOL-14 SAMPLE PROGRAM
;COMMENTS TO IDENTIFY THE PROGRAM WOULD APPEAR HERE.
;CONTOL STATEMENTS WHICH AFFECT THE WHOLE PROGRAM FOLLOW.
.LOC250
.FIXS5
;SUBROUTINES PRCEED THE MAIN PROGRAM.
;2CR=/6PB*/7PB*/1PS*/1FS*/5CR
Z2=/X13*/X14*/X6*/X7*/Y11

   0252 2413 TXN 013
   0253 2414 TXN 014
   0254 2406 TXN 006
   0255 2407 TXN 007
   0256 1411 TYN 011
   0257 5662 JFN 262
   0260 1377 TYF 377
   0261 1777 TYN 377
   0262 0354 JMR

;SOLE=3CR=(8PB*CRH+CRA)*7CR*8CR*9CR*/3LS*/4LS*/5LS
R1:Y4=(X15*Y7+Y6)*Y12*Y13*/Y14*/X2*/X3*/X4

   0270 2015 TXF 015
   0271 1007 TYF 007
   0272 5275 JFF 275
   0273 1406 TYN 006
   0274 5304 JFF 304
   0275 1012 TYF 012
   0276 1013 TYF 013
   0277 1414 TYN 014
   0300 2402 TXN 002
   0301 2403 TXN 003
   0302 2404 TXN 004
   0303 5306 JFF 306
   0304 3004 SYF 004
   0305 0344 SKP
   0306 3404 SYN 004
   0307 0354 JMR

;SOLF=4CR=/7CR*(CRA*5CR+/9CR+9PB*CRH)
R2:Y5=/Y12*(Y6*Y11+Y14+X16*Y7)

   0315 1412 TYN 012
   0316 5727 JFN 327
   0317 1006 TYF 006
   0320 1011 TYF 011
   0321 5331 JFF 331
   0322 1414 TYN 014
   0323 5731 JFN 331
   0324 2016 TXF 016
   0325 1007 TYF 007
   0326 5331 JFF 331
   0327 3005 SYF 005
   0330 0344 SKP
   0331 3405 SYN 005
   0332 0354 JMR
```

```
;NORMAL EQUATIONS FOLLOW.
;2LT=CRA=RESET*/CRH*(AUTO+CRA)
Y6=X10*Y7*(X11+Y6)


   0250 4224 JMP
   0251 0340 0340

   0340 2010 TXF 010
   0341 1007 TYF 007
   0342 5746 JFN 346
   0343 2411 TXN 011
   0344 1406 TYN 006
   0345 5750 JFN 350
   0346 3006 SYF 006
   0347 0344 SKP
   0350 3406 SYN 006

;CALL TO REPETITIVE EQUATIONS.
.VARS0
R1


   0356 4645 JMS
   0357 0270 0270


R2


   0360 4645 JMS
   0361 0315 0315


;NORMAL EQUATIONS CONTINUE BELOW.
;3LT=CRH=RESET*(MAN+CRH)
Y7=X10*(X12+Y7)

   0367 2010 TXF 010
   0370 5774 JFN 374
   0371 2412 TXN 012
   0372 1407 TYN 007
   0373 5776 JFN 376
   0374 3007 SYF 007
   0375 0344 SKP
   0376 3407 SYN 007

;SOLA=SOLC=CR1=/6CR*(6PB*7PB*2CR*/1LS*/2LS*5LS*6LS*7CR+1CR)
Y0=/Y3*(X13*X14*Z2*/X0*/X1*X4*X5*Y12+Y0)

   0404 1403 TYN 003
   0405 5424 JFN 024
   0406 2013 TXF 013
   0407 2014 TXF 014
   0410 5422 JFN 022
   0411 4645 JMS
   0412 0252 0252
   0413 5022 JFF 022
   0414 2400 TXN 000
   0415 2401 TXN 001
   0416 2004 TXF 004
   0417 2005 TXF 005
```

```
   0420 1012 TYF 012
   0421 5026 JFF 026
   0422 1400 TYN 000
   0423 5426 JFN 026
   0424 3000 SYF 000
   0425 0344 SKP
   0426 3400 SYN 000

Y2=X0

   0434 2400 TXN 000
   0435 5440 JFN 040
   0436 3002 SYF 002
   0437 0344 SKP
   0440 3402 SYN 002

;4LT=/6PB*/7PB*(1PS+1FS)
Y10=/X13*X14*(X6+X7)

   0446 2413 TXN 013
   0447 2014 TXF 014
   0450 5454 JFN 054
   0451 2406 TXN 006
   0452 2407 TXN 007
   0453 5456 JFN 056
   0454 3010 SYF 010
   0455 0344 SKP
   0456 3410 SYN 010

;5LT=5CR=3LS*/4LS+5CR*8CR
Y11=X2*/X3+Y11*Y13

   0464 2002 TXF 002
   0465 2403 TXN 003
   0466 5074 JFF 074
   0467 1011 TYF 011
   0470 1013 TYF 013
   0471 5074 JFF 074
   0472 3011 SYF 011
   0473 0344 SKP
   0474 3411 SYN 011

;SOLB=SOLD=6CR=7CR*(CRA*/5LS*/6LS*(6CR+5CR*8CR)+PB10*CRH)
Y2=Y12*(Y6*/X4*/X5*(Y2+Y11*Y13)+X17*Y7)

   0502 1012 TYF 012
   0503 5520 JFN 120
   0504 1006 TYF 006
   0505 2404 TXN 004
   0506 2405 TXN 005
   0507 5515 JFN 115
   0510 1402 TYN 002
   0511 5522 JFN 122
   0512 1011 TYF 011
   0513 1013 TYF 013
   0514 5122 JFF 122
   0515 2017 TXF 017
   0516 1007 TYF 007
   0517 5122 JFF 122
   0520 3002 SYF 002
   0521 0344 SKP
   0522 3402 SYN 002
```

```
;6LT=7CR=4LS
Y12=X3

   0530 2403 TXN 003
   0531 5534 JFN 134
   0532 3012 SYF 012
   0533 0344 SKP
   0534 3412 SYN 012

;7LT=8CR=1LS*2LS
Y13=X0*X1

   0542 2000 TXF 000
   0543 2001 TXF 001
   0544 5147 JFF 147
   0545 3013 SYF 013
   0546 0344 SKP
   0547 3413 SYN 013

;/8LT=9CR=/11PB*(9CR+12PB)*(7CR+8CR)
/Y14=/X20*(Y14+X21)*(Y12+Y13)

   0555 2420 TXN 020
   0556 5565 JFN 165
   0557 1414 TYN 014
   0560 2421 TXN 021
   0561 5165 JFF 165
   0562 1412 TYN 012
   0563 1413 TYN 013
   0564 5567 JFN 167
   0565 3414 SYN 014
   0566 0344 SKP
   0567 3014 SYF 014

;CALL TO REPETITIVE EQUATIONS.
.VARS0
R1
   0575 4645 JMS
   0576 0270 0270

R2

   0577 4645 JMS
   0600 0315 0315

.END 250

   0606 4224 JMP
   0607 0250 0250

ERROR LINES: 0000

EDITED LINES: 0000

PROGRAM BREAK: 0607

TURN PUNCH ON

B€¶ .2??,
/<  +$,/+/+$,/&/($&%8&%
  />$,,&%*(
            (,$, $,,,.$(<(<$
-))$-                          --
     $
       )'$-5)5
"((%          -7$&%8&%
```

SPURIOUS CHARACTERS ARE TYPED WHILE BINARY IS PUNCHED.

CHAPTER 7

PAL-14

PDP-14 SYMBOLIC PROGRAM ASSEMBLER

The PAL-14 symbolic assembler translates programs into the binary machine code which is input to SIM-14, or which may be used to generate an ROM for the PDP-14. PDP-14 programs may be written with the Editor and "assembled" (translated into machine code) by PAL-14, thereby enabling the user to correct program errors with the Editor and to reassemble with PAL-14. This capability considerably eases the debugging procedure because sections of the source program may be moved, or new lines inserted, without retyping the remainder of the program. Figure 7-1 outlines the procedure for developing programs using PAL-14.

PAL-14 GENERAL FEATURES

Chapter 5 described the machine code instructions of the PDP-14. Remember that test instructions referred to specific input and output numbers and that the jump instructions referenced locations by an address number (viz., an absolute address for a JMP or JMS; a relative or page address for a JFF or JFN). Consider the following PDP-14 program segment:

```
      .
      .
      .
227  TXN 15
230  JFN 233
231  SYF 12
232  SKP
233  SYN 12
234  JMP
235  437
```

NOTES

1. Each instruction is assigned to a numbered location.
2. The JFN and JMP instructions refer to locations by specific address numbers.
3. The TXN, SYF, and SYN instructions refer to input or outputs by specific number.

Changing PDP-14 programs written with numeric addresses and references is very cumbersome. For example, assume that there should be a "TYN 27" instruction after the "TXN 15" stored in location 227 in the above program. To correct the program with SIM-14, each instruction is moved to the next sequential location (by typing it into that location) thereby allowing the "TYN 27" to be inserted at location 230.

However, simply moving each instruction to the next successive location does not properly solve the problem. The result of simply moving the instructions is:

```
        .
        .
        .
227 TXN  15
230 TYN  27
231 JFN  233
232 SYF  12
233 SKP
234 SYN  12
235 JMP
236 437
        .
        .
        .
```

Note that by relocating the program segment, the sense of the JFN 233 instruction has changed. Originally the jump caused transfer to the SYN 12 instruction; now it causes transfer to the SKP. The JFN instruction must be changed to a JFN 234 to preserve the original sense of the program. Thus, there are dangers in simply relocating programs with SIM-14. These dangers can be removed by using PAL-14 and reassembling, as will be seen.

Location Assignment

Note that PDP-14 programs are stored in consecutive memory locations. The need to specify the address of each instruction is therefore unnecessary. The locations may be assigned by PAL-14 using an origin statement for the sequence of instructions. The statement is "LOC NNNN" where NNNN is an absolute PDP-14 address. This statement instructs PAL-14 to assign the instructions which follow to consecutive PDP-14 locations beginning with location NNNN.

```
LOC 227
TXN 15
TYN 27
JFN 234
SYF 12
SKP
SYN 12
JMP
437
```

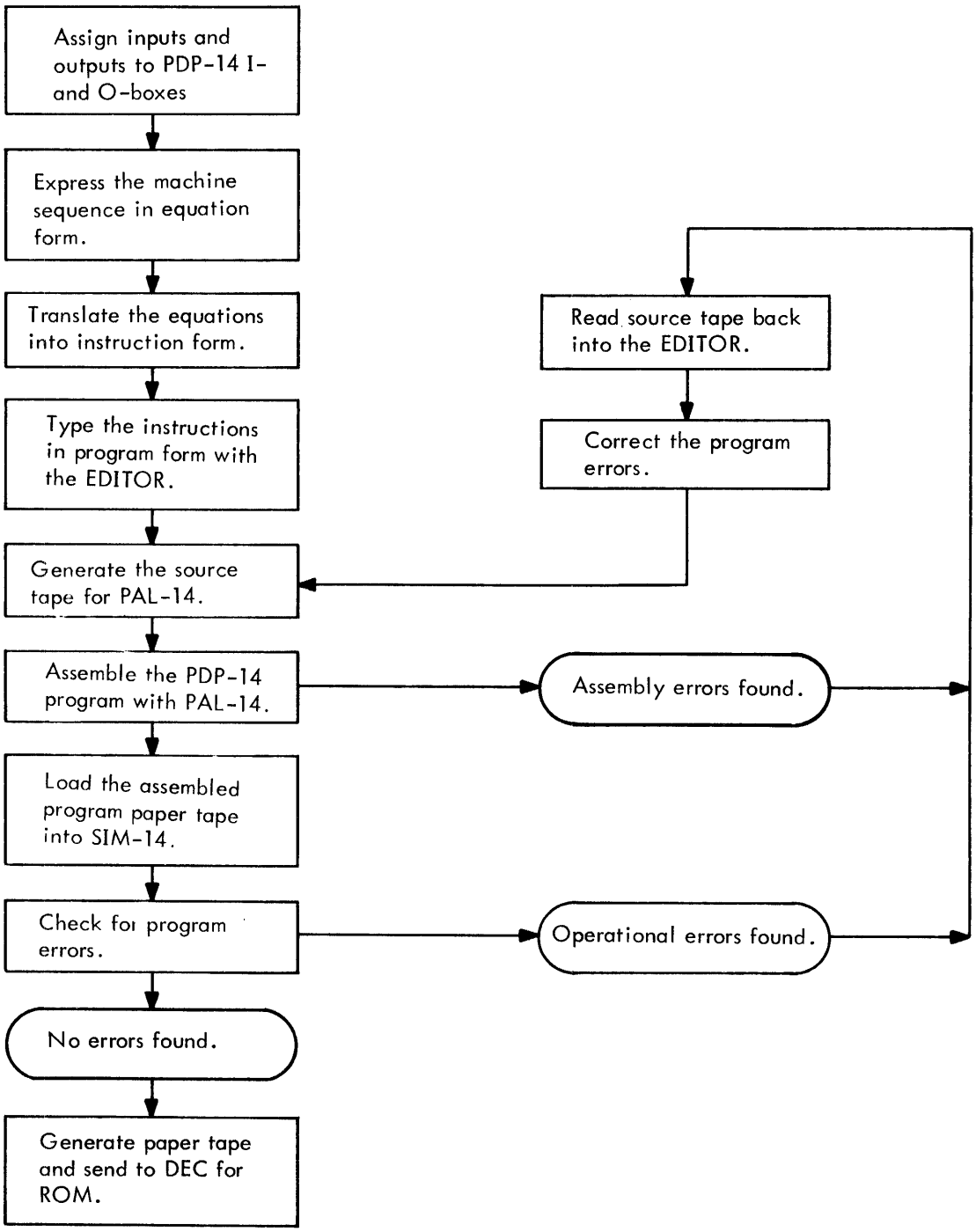

Assign inputs and outputs to PDP-14 I- and O-boxes
Express the machine sequence in equation form.
Translate the equations into instruction form.
Type the instructions in program form with the EDITOR.
Generate the source tape for PAL-14.
Assemble the PDP-14 program with PAL-14.
Load the assembled program paper tape into SIM-14.
Check for program errors.
No errors found.
Generate paper tape and send to DEC for ROM.
Read source tape back into the EDITOR.
Correct the program errors.
Assembly errors found.
Operational errors found.

Figure 7-1

PAL-14 recognizes the LOC 227 as an origin statement, and assigns the instructions which follow to sequential locations, beginning with location 227. Hence, the TXN 15 is stored in 227; the TYN 27 is stored in 230; the JFN 234 is stored in 231; etc. Thus, the need to write location numbers is eliminated by PAL-14.

Symbolic Addresses

The LOC statement by itself does not eliminate the problems arising when PDP-14 programs are altered. However, automatic location assignment using the LOC statement permits symbolic addresses which do eliminate some of the problems. Symbolic addresses are simply name tags or labels used to identify memory locations. Instead of writing jump instructions with numeric addresses, jump instructions are written with symbols for addresses. These symbols are then identified (defined) elsewhere in the program. The symbol is assigned a numeric value during the assembly of the program by PAL-14.

The instruction JFN 234, for example, may be written JFN TAG . The location to be jumped to is assigned the address label "TAG", by writing TAG followed by a comma, preceding the referenced instruction.

```
        .
        .
        .
    JFN TAG
        .
        .
        .
TAG, SYN 12
        .
        .
        .
```

The JFN instruction above references the symbolic address "TAG", while the statement "TAG, SYN 12" defines the symbolic address.

PAL-14 will assign address values to each instruction of the program using a "location counter" (LC). The LOC statement sets the LC to a value (e.g., LOC 200 sets the LC to 200). PAL-14 then assigns each successive instruction of the source program to the locations specified by the LC. The LC is incremented for each instruction, thereby storing the program in sequential locations of PDP-14 memory. Whenever a symbolic address (recognized because it is followed by a comma) is encountered, it is assigned the value of the LC. Thus, if the LC is equal to 234 when the line "TAG, SYN 12" is encountered in the source program, the SYN 12 is stored in PDP-14 memory location 234, and TAG is assigned the value 234. The instruction JFN TAG thus becomes JFN 234 when assembled by PAL-14.

Note that programs in this form may readily be changed to eliminate bugs. The program may be completely relocated by changing the LOC statement. Parts of the program may be inserted or deleted without affecting address references made by jump instructions. Upon reassembly, PAL-14 will correctly assign all symbolic addresses and any references to them.

Symbolic addressing does not eliminate the need to consider paging for JFF and JFN instructions. An address error will result if a PAL-14 program contains such instructions which reference symbolic addresses that are not defined within the same PDP-14 memory page. Only the JMP or JMS instruction may directly cross page boundaries.

The selection of the symbol "TAG" is arbitrary and could be any group of letters and numbers subject to the rules for symbols which are given in Table 7-1. For example, SETY1ON is a valid PAL-14 symbol. Since it contains 7 characters, however, only the first six are recognized by PAL-14. Thus if the symbols SETY1ON and SET1OF were both used as symbolic addresses in a PAL-14 program, multiple definition error would occur. PAL-14 only recognizes the first six characters, and therefore cannot distinguish between two symbols which are identical in these first six characters. The error results because the same symbol is defined twice in the program. Note that the symbols SOLAON and SOLAOFF are perfectly acceptable, since they differ in the first six characters.

Table 7-1
PAL-14 Symbol Rules

1.	The first character of a symbol must be a letter.
2.	The successive characters in the symbol may be only letters or numbers. No special characters (e.g., *, +, -, /, $) are permitted within symbols.
3.	A symbol must only be defined at only one point in a program. Symbols may be referenced as many times as necessary.
4.	Only the first six characters of the symbol are meaningful to PAL-14. Although more characters may be used, they are not meaningful to PAL-14.
5.	No spaces or tabs are allowed within the symbol.

Assignment Statements

PAL-14 assignment statements equate a six character symbol with a numeric value. Symbols used in assignment statements are subject to the rules given in Table 7-1. Assigning a value to a symbol which is also used as a symbolic address is illegal in PAL-14. The same symbol may, however, be used in more than one assignment statement; the last assigned value is used whenever a symbol has been redefined in this way.

The most common use of assignment statements in PAL-14 programs is to identify input and output assignments. It is more convenient to remember that "LS1 is limit switch 1" as opposed to "17 is limit switch 1". Since PDP-14 programs must refer to inputs and outputs by the numbers which are determined by the I- and O-box assignments (e.g., 17), the following PAL-14 assignment statement is used.

```
LS1=17
```

Instructions may then test "LS1" in place of testing "17". The PDP-14 program may include, usually at its beginning, a list of such assignments which are used throughout the program. This allows the program to be written in a completely symbolic form. For example:

```
        LS1=17
        LS2=30
        SOLA=11
        SOLB=12
        PB1=10
        PB2=11
LOC 250
Y11,    TXN LS1
        TXF LS2
        JFN SOLAN
        TXF PB1
        TYN SOLB
        JFF SOLAF
SOLAN,  SYN SOLA
        SKP
SOLAF,  SYF SOLA
END
```

The END statement is necessary to mark the end of the program for PAL-14. Remember that assignment statements need only be made at one point in the program and the symbols thus assigned may then be used throughout the program. (The above example only controls one output.) Also note that LS1 = "17", not "X17". The X and Y designations are not used in PAL-14.

PAL-14 PROGRAM CONVENTIONS

PAL-14 programs are free-form; spaces and/or tabs may be used to organize and format the program as desired. At least one space (or a TAB) must be used between an instruction (e.g., TXN or SYF) and the referenced symbol or operand (e.g., LS1 or SOLB). Tabs and spaces may not be within a symbol or instruction, but only between symbolic addresses, instructions, operands, and comments. Otherwise these characters have no effect upon the machine code generated by PAL-14; they are only used to format the source program. For example:

```
SOLA            TXF             LS1
```

generates the same machine code as

```
SOLA,TXF LS1
```

where the comma is needed to identify "SOLA" as a symbolic address.

Comments

Program comments are lines of text included in the PAL-14 program to clarify the function of an instruction or group of instructions. Comments do not affect the machine code generated by PAL-14; they only exist as a reminder for later reference to the program listing. Comments may be added within the PAL-14 program by preceding them with a slash (/). Comments may either conclude lines of a program, or may be on separate lines by themselves. PAL-14 instructions or assignment statements may not be included on a line started with a comment because all information would be treated as part of the comment.

The following are examples of PAL-14 comments.

```
/THE FOLLOWING PROGRAM CONTROLS SOLENOID A
      LOC 200
Y1,   TXF  LS1      /LS1  ACTIVATED WHEN SLIDE 2 IS FULL FORWARD.
      TXN PB1       /PB1  IS SHUT DOWN BUTTON.
        .
        .
        .
```

Two-Location Instructions

The PDP-14 two-location instructions (e.g., JMP, JMS, TRM) are written in PAL-14 source programs with both parts of the instruction on one line. These instructions will require two PDP-14 memory locations when they are translated into machine code instructions. The following example illustrates the two-location translation proces The input to PAL-14 appears at left; the translated output in machine code is given at right. The symbols Y5 and NEXT are symbolic addresses which have been assigned the values 1323 and 1401 respectively by the curren location counter of PAL-14.

```
      STA1N=2002          200  TRM
      LOC 200             201  2002
      TRM  STA1N          202  JMS
      JMS  Y5             203  1323
      JMP  NEXT           204  JMP
          .               205  1401
          .                     .
          .                     .
Y5,   TXN  LS16                 .
          .
          .
          .
```

```
NEXT,  TYN SOLB
        .
        .
        .
```

Permanent Symbol Table

PAL-14 must "know" what PDP-14 machine code to generate for each symbol of a source program before it can translate that program into a complete PDP-14 machine code program which is acceptable to SIM-14 or to the ROM generator. Numbers are the only elements of PAL-14 source programs which need no definition; all symbols must be defined in terms of these numbers. As we have seen, definitions may be in the form of assignment statements (e.g., LS1=27) and symbolic addresses (whose values are defined by the location counter and LOC statements).

In addition to user defined symbols, PAL-14 has a table of permanently defined symbols and their numeric equivalents. The permanent symbol table allows the user to write statements such as "TXN 15" without having previously defined the symbol TXN. This symbol and all others given in Table 7-2 are permanently defined within PAL-14. The list includes the basic PDP-14 instructions introduced in Chapter 5 as well as the extended instructions introduced in Chapter 9. Also included are internal register names and pseudo-instructions which are described later in this chapter.

Each of these symbols may be used in PAL-14 source programs without definition by the user; furthermore, these symbols must not be used in PAL-14 source programs as symbolic addresses or in assignment statements.

The PAL-14 symbol table is finite in size. When loaded in a 4K PDP-8, 310 symbols may be accommodated. When loaded in an 8K PDP-8 (in field 1), 792 symbols may be accommodated. When its capacity is exceeded by user defined symbols, the following message is typed:

```
*SMB OFLW
```

Recovery is described in the PAL-14 Operation section of Chapter 10 (pages 10-21).

Table 7-2
PAL-14 Permanent Symbol Table

PDP-14 Instructions				
Symbolic	Octal	Meaning	Argument	Max. Arg. Value in Octal
TXF	2000	Test input for OFF	Input Number	377
TXN	2400	Test input for ON	Input Number	377
TYF	1000	Test output for OFF	Output Number	377
TYN	1400	Test output for ON	Output Number	377
SYF	3000	Set output OFF	Output Number	376
CLR	3377	Clear all outputs	None	---
SYN	3400	Set output ON	Output Number	377
JFF	5000	Jump if test flag OFF	Page Address	377
JFN	5400	Jump if test flag ON	Page Address	377
JMP	4224	Jump unconditional	Absolute Address	7777
SKP	0344	Skip next location	None	---
JMS	4645	Jump to subroutine	Absolute Address	7777
JMR	0354	Jump return from subroutine	None	---
NOP	0000	No operation	None	---
HLT	0007	Halt the PDP-14	None	---
TXD	7000	Test input and display	Input Number	377
TYD	7400	Test output and display	Output Number	377
EEM	0600	Enter external mode	None	---
EES	0645	Enter external mode, store PC1	None	---
LEM	0400	Leave external mode	None	---
LER	0454	Leave external mode, restore PC1	None	---
TRM	4226	Transfer memory to Output	Any data word	7777
TRS	4225	Transfer memory to Storage in PC2	Any data word	7777
TRR	0200	Transfer register to register	Src.R Dest.R*	77
SKE	6704	Skip if register equal to PC2	R*	7
SKZ	6304	Skip if register is zero	R*	7

*R is one of the internal registers given below.

PDP-14 Internal Registers			
Register Name	Octal Code	Register Name	Internal Code
DUMMY	0000	PC1	0004
IR	0001	PC2	0005
MB	0002	INPUT	0006
SPARE	0003	OUTPUT	0006

Pseudo- Instructions	
END	End of source program
EOT	End of tape in a segmented program
LOC	Set location counter
PAGEJ	Page jump to start of next page

Special Characters and Operators

Several characters of the Teletype keyboard have special meaning to PAL-14. An example is the slash (/) which signals the start of a comment. Another example is the carriage return which terminates a statement line Table 7-3 contains all characters which are legal in the operation portion of a PAL-14 statement (viz., the statement exclusive of any comment field). If any other character appears in the operation part of a PAL-14 statement, the "I" error (illegal character) will be generated.

Table 7-3
PAL-14 Special Characters

Character	Use in PAL-14
+ (plus)	combines symbols or values by two complement addition
- (minus)	combines symbols or values by two complement subtraction
! (logical OR)	combines symbols or values by a logical OR
> (shift)	shifts a value one octal digit left
␣ (space)	separates instructions from operands and otherwise formats the source program
→| (tab)	separates instructions from operands and otherwise formats the source program
↵ (carriage return)	terminates a statement
/ (slash)	terminates the operation part of a statement and starts a comment
, (comma)	identifies symbols used as symbolic addresses
= (equals)	assigns values to symbols directly
. (period)	A symbol which, when encountered in the program, is assigned a value equal to the present value of the PAL-14 location counter

The following characters are ignored in PAL-14 source programs but do not generate an error.

line feed
form feed
rubout
leader/trailer (null)

Addition and Subtraction - The "+" and "-" characters may be used with symbols and octal numbers, usually for addressing purposes. The statement "JMP START +2" in the following program segment causes an unconditional jump to the second location after that labeled "START" (i.e., location 102 and the TNX PB3 instruction).

```
LOC  100
START,  TXN  PB1
        JFN  OUT
        TXN  PB3
          .
          .
          .
        JMP  START+2
          .
          .
          .
```

Likewise the minus sign may be used; for example "JFF SOLAN-2" causes a JFF to the second location before the location labeled SOLAN. The location counter of PAL-14 will assign values to all labels and then PAL-14 will compute the values of all symbolic addresses which contain operators and symbols.

Logical OR - The "!" character may be used in PAL-14 programs to combine values of symbols or octal numbers. The result is the bit-by-bit inclusive OR of the two values. The following table gives the resultant value when any two octal digits are combined with the "!" character.

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	1	3	3	5	5	7	7
2	2	3	2	3	6	7	6	7
3	3	3	3	3	7	7	7	7
4	4	5	6	7	4	5	6	7
5	5	5	7	7	5	5	7	7
6	6	7	6	7	6	7	6	7
7	7	7	7	7	7	7	7	7

When numbers which contain two or more digits are combined, each digit in the result is obtained by combining the corresponding digits in the original numbers using the above table.

For example:

```
1234 ! 2460 = 3674
```

The operator may be used in PAL-14 in the following manner:

```
TRM 15 ! 27          translates into      TRM
                                          37
```

The values of symbols may be combined with the "!" character also.

```
ON = 1
SOLA = 1500
TRM SOLA ! ON        translates to        TRM
                                          1501
```

Shift - The > character changes the value of the octal number or symbol which precedes it by shifting it one octal digit to the left. Zeros are shifted into vacated positions on the right and the most significant digit is lost. For example:

```
0231 > = 2310
0010 > = 0100
1234 > = 2340
```

The shift character will often be used with the PDP-14 instruction TRR and internal register names. PAL-14 maintains the following register assignments in its permanent symbol table.

PAL-14 Register Name	Octal Equivalent
DUMMY	0
IR	1
MB	2
SPARE	3
PC1	4
PC2	5
INPUT	6
OUTPUT	6

The above register names may be used as operands in PAL-14. For example:

```
TRR  PC2 >    OUTPUT
```

will generate the PDP-14 instruction to transfer the contents of PC2 to the output register. The octal value of PC2 (5) is shifted one octal digit, then OR'd with the octal value of OUTPUT (6) and the result is OR'd with the octal equivalent of TRR (0200), thereby generating the octal result 0256 for transfer PC2 to the output register. Note that the following two expressions are equivalent.

```
TRR   PC2 >     OUTPUT          TRR!   PC2 >!    OUTPUT
```

The shift character only affects the symbol or number which immediately precedes it. Thus

```
STATN3 = 300          translates to        TRM
TRM 1 +300 >                               3001    (not 3010)
```

Period - The period is a special symbolic address used in PAL-14. Its value changes for each instruction and is always equal to the present value of the PAL-14 location counter. Jump instruction operands may contain the period (often read "dot") and the +, -, and ! operators. For example, JFF .+12 (read, JFF dot plus 12) means JFF to the twelfth octal (tenth decimal) location after the JFF instruction. The period is most used with small (less than 10) octal numbers; large jumps are best done in PAL-14 with true symbolic addresses.

A variation of this instruction is "JFF .+1". Note that this instruction causes the next instruction to be executed for both states of the TEST flag. However, the TEST flag is cleared in both cases; thus the instruction JFF .+ may be used to unconditionally clear the TEST flag.

The period may also be used with two location instructions such as JMP, JMS or TRM. However, the value of the period is the value of the location counter when the second location is assigned to its location. Thus JMP .+2 causes transfer to the third location which follows the JMP. Assuming there are no two location instructions following the JMP, the jump is to the second instruction which followed.

```
LOC   200          translates to      200   JMP
JMP   .+2                             201   203      (201 +2)
TXN   1                               202   TXN 1
TXN   2                               203   TXN 2
```

The following instruction could be used to have a TRM instruction output its own address:

```
LOC   1400         translates to      1400   TRM
TRM   .-1                             1401   1400
```

Pseudo-Instructions

Pseudo-instructions are included in PAL-14 program listing on the source paper tape to direct the assembly process. They are not themselves translated into PDP-14 machine code; they affect the translation of other instructions into machine code. There are four pseudo-instructions in PAL-14 including the LOC and END statements described previously in this chapter.

END - The END pseudo-instruction marks the end of a source program and directs PAL-14 to terminate the current assembly pass. The complete assembly process requires either two or three "passes" (dependent upon the assignment of input/output devices) during which the source tape is read by PAL-14 and the translation to PDP-14 machine code is made. The END statement terminates the current pass and causes PAL-14 to halt while the source tape is reloaded to begin the next assembly pass.

Since the END statement halts the reading of a paper tape during the assembly, any information present on the source tape after the END statement will not be read by PAL-14. However, the comments are a permanent part of the source program and may be read with the Editor. The assembly process will be faster if lengthy comments and program documentation are written here.

EOT - The pseudo-instruction EOT (end of tape) allows a long PAL-14 program to be segmented and recorded on more than one source paper tape. The EOT is typed as the last statement on all tapes except the last one which must be concluded with the END pseudo-instruction. PAL-14 will stop upon the reading EOT at the end of each tape during an assembly pass and type "*EOT". The next source tape is then loaded and the assembly continues. This procedure is repeated for each source tape in the complete program.

LOC - The LOC pseudo-instruction directs PAL-14 to set its location counter to the operand value of the LOC. As many LOC statements as desired may be used. However, if no LOC statement is present, a "LOC 0000" statement at the beginning of the program is assumed by PAL-14.

The LOC statement may advance or reset the location counter to any desired value. If the LOC statement returns the location counter to an earlier value, any previously assembled instruction at that location will be lost. For example,

```
        LOC   50
              TXN  LS1
              TXN  LS2
              .
              .
              .
```

```
LOC   50
          SYN  SOLA
          .
          .
          .
```

results in the assembly of two instructions for the same location. The first instruction (TXN LS1) is lost from the assembly; PDP-14 location 50 will contain the assembled instruction for SYN SOLA.

The LOC statement may have an expression or symbolic operand. All terms of the expression must have been previously defined to allow PAL-14 to set the location counter to the proper value.

For example:

```
          .
          .
          .
SOLAF,  SYF  SOLA
/LEAVE 8 LOCATIONS BEFORE STARTING NEXT PROGRAM SEGMENT.
LOC SOLAF+10
          .
          .
          .
```

The value of SOLAF will be assigned at the time of assembly and the LOC statement will set the location counte to the value of SOLAF plus 10 (octal). Thus the value of a LOC statement may be determined at the time of assembly.

The pseudo-instruction "LOC .!377+1" causes PAL-14 to advance the location counter to the first location of the next PDP-14 memory page. The statement is evaluated by PAL-14 as: the current value of the location counter (.) OR'd with 377 (octal) plus 1. If the location counter is 1312 for example, the result of the OR will be 1377 and the LOC statement will set the location counter to 1400 ($1377_8 + 1_8$).

PAGEJ - The PAGEJ (page jump) pseudo-instruction causes PAL-14 to include in the assembled program whatever instruction is necessary to transfer control to the first location of the next PDP-14 memory page. The actua code generated is dependent upon the current value of the location counter.

Current Location	Instruction Generated
First location in a memory page	none
Last location in a memory page	NOP
Any other memory location	JMP .!377+1

If the next instruction would not have been assembled as the first or last instruction in the page, PAL-14 will supply a JMP instruction, the operand of which is 1 plus the OR of the current location counter value and 377. This JMP always goes to the first location of the next PDP-14 memory page.

The PAGEJ pseudo-instruction also acts as a "LOC .!377 +1" in the cases where a JMP instruction was assembled. Thus PAGEJ supplies the necessary PDP-14 instruction to transfer control to the beginning of the next memory page, and also sets the PAL-14 location counter to that location.

PAL-14 OUTPUT

The output of PAL-14 consists of an assembly listing with error messages, a symbol table and a punched binary tape. The outputs are requested by the user in response to three PAL-14 messages:

*BIN -	Does the user want binary output; if so, on what device?
*LST -	Does the user want an assembly listing; if so, on what device?
*SMB -	Does the user want a symbol table listing; if so, on what device?

The responses to the three messages are:

L	The output is requested and directed to the low-speed device (e.g., Teletype printer - LST or SMB; or paper tape punch - BIN).
H	The output is requested and directed to the high-speed paper tape punch.
(RETURN)	The output is not requested.

Binary Output

The PAL-14 binary output is a paper tape which could be used to generate an ROM for the PDP-14 controller. Normally it is used as input to SIM-14 where the program may be simulated and debugged. The paper tape has a punched code at its end (checksum) which is read by SIM-14 when the tape is read into memory to verify reading accuracy.

Assembly Listing

The assembly listing contains the original source statements, the numeric value of the generated PDP-14 machine code and error codes. PAL-14 formats the assembly listing into pages of 66 lines (approximately 11 inches long). It types the page number (in octal) at the top of each output page.

The assembly listing contains four fields:

Error field	-	contains letters to indicate all assembly errors included in the line of the source program.
Location field	-	contains the four digit octal address of the PDP-14 location which contains the assembled instruction. Only assembled instructions have entries in this field.
Code field	-	contains the numeric value of the code for assembled instructions, assignment statements or the LOC pseudo-instruction.
Source statement field	-	contains the line of user program as read from the source tape. If the source statement is longer than 54 characters it will not fit on one output line and PAL-14 will break it into two or more output lines preceding the second and later lines with an asterisk.

At the end of the program listing PAL-14 types:

ERR LINES	-	the number of lines (in decimal) in the source program which contain at least one error.
MEM BRK	-	the actual number of memory locations (in octal) occupied by the assembled program. (This is not the largest address; it is a count of the locations used.)

Symbol Table

The PAL-14 symbol table is a list in alphanumerical order of all user defined symbols in the source program (viz., all symbolic addresses and assigned values). Any undefined symbol is typed with the letter U in the column of values. Symbols which are defined by equating them with undefined symbols are assigned a value "+0". Thus, if the statement "PB1=PB0 + 1" and there is no statement which defines PB0, the symbol table will contain "PB0 -U" and "PB1 +0". The values of directly assigned symbols are typed as five digit numbers (with leading zeros) to distinguish them from symbolic address values which are typed as four digit numbers.

Error Messages

Error messages are single letters typed as the first entry of the same line as the source statement which contains the error. If more than one error is detected, a list of all error codes is typed in alphabetical order. A maximum of six error codes is typed for each line. If the same error occurs at more than one place in the source statement, the error code is only typed once.

PAL-14 stops processing the source statement when it encounters an S (syntax) error, as described in the table below. Thus the source statement may contain undetected errors since they occur after the cause of the syntax error. These errors will, of course, be detected after the cause of the syntax error is removed and the statement is reassembled.

The following is a complete table of errors recognized by PAL-14. The table includes examples of source statements which cause the particular error and states the action taken by PAL-14 when the error occurs.

Error Code	Meaning	Action Taken
A	Address Error - the page address of a JFF or JFN instruction is not on the same page as the instruction itself. Example: At location 376 A LEAVE, JFN .+3	The machine code is generated for JFF 0 or JFN 0. The code for the example is JFN 0, therefore.
D	Double Defined Symbol Error - an instruction references a symbol which has been defined more than once in the source program. A, TYN SOLA A, JFN OUT D JMP A	PAL-14 uses the first definition. The JMP A in the example would jump to the TYN SOLA instruction.
I	Illegal Character Error - The statement contains a character which is not acceptable to PAL-14. Example: I A, JMP A#BC	PAL-14 ignores the illegal character. Thus the example is interpreted: A, JMP ABC
L	Label or Assignment Error - the first character of a symbol used in assignments or labels is not a letter (A-Z) or the left hand member of an assignment statement is numeric. Example: L 50=LS1 L 47, JMP RESTART	The label or assignment is ignored by PAL-14.
M	Multiple Definition Error - a label is defined more than once in the source program. Examples: M A, TYN SOLB M A, JFN SOLBN	The first definition is used and all others are ignored by PAL-14. In the examples, A is equal to the location which contains TYN SOLB.
O	Operation Error - the source program contains an illegal operation (e.g., two instructions in the same statement). Example: O START, JMP TEST JMP	PAL-14 will ignore the illegal and all subsequent information. In the example, the machine code is generated as if the statement were: START, JMP TEST

Error Code	Meaning	Action Taken
P	Phase Error - a label has a different Pass 1 and Pass 2 value. This error is normally caused by a symbolic LOC statement the operand of which is defined after it is used. Example: P LOC CDE P XYZ, JMP .+2 CDE=50	The current value of the location counter is used to define the label for each pass. Since CDE is undefined at the start of assembly, LOC 0 is assumed, thus, CDE=0 and XYZ=0. During the first pass however, CDE is given the value 50 and on the second pass LOC CDE becomes LOC 50 and XYZ=50, thereby causing the phase errors.
R	Redefinition Error - the source program attempts to redefine (using an assignment statement) a PAL-14 permanent symbol (see Table 7-2) or a previously defined label. Examples: LOC 50 R JMP=400 ABC, TXN LS1 . . . R ABC=70	The symbols retain their original values and the assignment statement is ignored.
S	Syntax Error - the source statement is not meaningful to the assembler. Examples: S ABC, JMP .+ S LS1+LS2, S LS1= S SOLA, TXN+=3 S JFN SOLAN,	The statement is evaluated from left to right up to the point where the error occurred. Thus: ABC, JMP .+ assigns ABC the current value of the location counter and generates the machine code for "JMP .". The value of LS1 plus LS2 will be stored in the next memory location (without any symbolic address). The statement "LS1=" will not generate any machine code. The location with label "SOLA" will contain the machine code for TXN 0. The "JFN SOLAN," will be assembled as "JFN SOLAN".
T	Truncation Error - a single value exceeds 7777 octal, or the maximum argument value for the instruction (see Table 7-2) if maximum is less than 7777.	In assigned values, the excess right hand digits are ignored. Thus the result of A=40543+C is A=4054+C. If the truncated value is used as the operand of an instruction, PAL-14 substitutes the value 0 when the value is too large. Thus the value of:

Error Code	Meaning	Action Taken
	Examples: C=3 T A=40543+C ABC=375+100 T TYN ABC T SKZ 40	ABC = 375 + 100 TYN ABC is the following: ABC = 475 TYN 0 The instruction SKZ 40 becomes SKZ 0.
U	Undefined Symbol Error - an instruction references an undefined symbol. Example: XYZ = 50 U JMP XZZ +3 where XZZ is not defined	The undefined portion of the statement is set equal to 0. Thus: JMP XZZ +3 becomes JMP 0+3 or JMP 3
V	Illegal Value Error - A value given in the source program cannot be evaluated by PAL-14 Example: V ABC=8402 +5 LV 4LS, TXN LS4	The illegal digit is ignored, and the remainder of the expression is evaluated. Thus: ABC = 402+5 = 407 However, in the second case, the whole label is ignored.

Error Listing

The assembly listing as typed during pass 2 of PAL-14 includes error messages for all errored lines. However, PAL-14 also types a partial error listing of all lines containing errors which were detected during pass 1. This listing in general will not report all errors, as some errors are only detected during pass 2. Furthermore, the listing may report undefined symbols which will be properly defined during pass 2, such as the statements:

```
ABC=CDE
CDE=27
```

which result in ABC being undefined at the end of pass 1. It will, therefore, be included in the table of undefined symbols (with a "value" of +0). However, since CDE is defined later in pass 1, ABC will be defined equal to 27 on pass 2 of the assembly and will not generate an error for pass 2.

Although the pass 1 error listing may be incomplete, it does allow the user to detect errors at an early stage of the assembly and to correct them without wasting time on pass 2 of the assembly process. Although pass 1 errors were generated, if the user wishes he may still proceed to pass 2 without correcting the errors and thereby generate a complete error listing.

If the assembly listing is to be punched on the high-speed punch, all error lines detected on pass 2 are typed on the Teletype unit in addition to being punched as part of the complete listing on the high-speed punch.

The error table output is controlled by two characters typed during the query sequence which is described later. If N is typed, the separate error tables are not typed. If the assembly listing is to be output on the high-speed punch, typing E will cause pass 1 errors to be recorded there also.

Sample Output

Figure 7-2 is sample assembly output including the assembly listing and symbol table. Several errors are included intentionally to illustrate the error codes in the assembly listing and symbol table.

```
        /THE FOLLOWING PROGRAM SOLVES THE EQUATION
        /SOLA=(LS1*PB2*/PB3+SOLA*/LS2)*/PPB5
        /
        /
                SOLA=5
                LS1=415
                LS2=16
                PB2=7
                PB3=10
                PB5-12
        LOC 200
        SOLA,   TXF LS1
                TXF PB2
                TXN P*B3
                JFF CK5
                TYF SOLA
                TXN LS2
                JFN SOLAF
        5CHK,   TXN PB5
                JFF SOLAN
        SOLAF,  SYF SOLA          /TURN SOLA OFF.
                SKP
        SOLAN,  SYN SOLA          TURN SOLA ON.
        END
```

Source Program

```
*BIN-L
*LST-L
*SMB-L
*SRC-L
```

Command String

```
RT      0200   2000   SOLA,   TXF LS1
I       0202   2410          TXN P*B3
LV      0207   2400   5CHK,   TXN PB5
T       0213   3400   SOLAN,  SYN SOLA          TURN SOLA ON.
```

Pass 1 Errors

```
ERR LINES-0004  MEM BRK-0015

UNDEFINED SYMBOLS

CK5     -U
ON      -U
PB5     -U
TURN    -U
```

Pass 1 Symbol Table

```
*PAS

PAGE- 0001 - PAL-14 V1

                        /THE FOLLOWING PROGRAM SOLVES THE EQUATION
                        /SOLA=(LS1*PB2*/PB3+SOLA*/LS2)*/PPB5
                        /
                        /
R                                SOLA=5
                  0415           LS1=415
                  0016           LS2=16
                  0007           PB2=7
                  0010           PB3=10
U       0000      7766           PB5-12
                  0200  LOC 200
T       0200      2000  SOLA,    TXF LS1
        0201      2007           TXF PB2
I       0202      2413           TXN P*B3
U       0203      5000           JFF CK5
        0204      1200           TYF SOLA
        0205      2416           TXN LS2
        0206      5611           JFN SOLAF
LUV     0207      2400  SCHK,    TXN PB5
        0210      5213           JFF SOLAN
        0211      3200  SOLAF,   SYF SOLA          /TURN SOLA OFF.
        0212      0344           SKP
TU      0213      3400  SOLAN,   SYN SOLA          TURN SOLA ON.
                        END
```

Pass 2 Listing

Error Codes

```
ERR LINES-0007  MEM BRK-0015

SYMBOL TABLE


CK5    -U
LS1    -00415
LS2    -00016
ON     -U
PB2    -00007
PB3    -00010
PB5    -U
SOLAF  - 0211
SOLAN  - 0213
SOLA   - 0200
TURN   -U
```

Pass 2
Symbol Table
(Addresses are 4-digit numbers;
assigned values are 5-digit numbers.)

```
*PAS
@?6B(
     .*$

*PAS
```

Spurious characters are
typed while binary is punched.

Pressing CONT will restart PAL-14.

CHAPTER 8

SIM-14

PDP-14 PROGRAM DEBUGGING AID

SIM-14 is the PDP-8 based simulator for the PDP-14 system. Its primary use is to assist the PDP-14 user in debug ging programs which have been compiled by BOOL-14 or which have been assembled by PAL-14. It may also be used to compose programs directly by typing PDP-14 instructions under control of SIM-14.

MODES OF OPERATION

SIM-14 has two basic modes of operation (local and on-line), offering the user several approaches to locating and correcting program bugs. SIM-14 also protects the user from making mistakes which could damage valuable machinery.

Local Mode

Local mode allows the user to check his program for errors off-line, entirely within the PDP-8 family computer. All inputs and outputs are simulated by the PDP-8 computer. The program may be checked without concern for damage to equipment, since operations in local mode have no effect on the PDP-14 or the controlled machinery. In fact, it is unnecessary to have the machinery connected to the PDP-14, or even to have the PDP-14 connectec to the PDP-8.

On-Line Mode

On-line mode is used to check that a program, which has been fully debugged in local mode, will operate the actual equipment when placed in a PDP-14 memory. The PDP-14 is used in on-line mode to execute the pro- gram contained within the PDP-8 computer. The program may be executed in part or in total, with the user specifying the location (or locations) where program execution may be stopped. The user may also specify a sequence of operations to be performed before program execution (and therefore control) is terminated.

RESTRICTIONS

SIM-14 can be used to simulate PDP-14 programs of 1K, 2K, 3K or 4K words. However, if more than 1K of PDP-14 program is to be simulated, SIM-14 must be loaded in a PDP-8 computer with at least 8K of core memory.

CONVENTIONS

The following conventions are used to describe the dialogue between the user and SIM-14.

a. All characters typed by SIM-14 will be identified in this manual by an underline.

Examples:

NOP	Typed by SIM-14
TXN 1	Typed by user

b. The non-printing character RETURN will be noted in this manual by a curved down arrow. No character appears on the teleprinter when this character is typed.

Example:

TXN 1 ↵	User concluded typing with a carriage return

c. The non-printing character LINE FEED will be noted in this manual by a straight down arrow.

Example:

TXN 1 ↓	User concludes typing with a line feed

PDP-14 INSTRUCTIONS RECOGNIZED BY SIM-14

SIM-14 local mode recognizes and simulates the PDP-14 instructions given in Table 8-1. Only these instructions are recognized or simulated; if the user attempts to execute any other instruction in local mode, the error message OPER (operation error) and the location of the instruction will be typed.

TXD, TYD and TRM are PDP-14 instructions used for monitoring (see Chapter 9). They are treated as NOP's during equation verification or truth table generation. They are simulated and cause appropriate messages during local mode simulated execution.

On-line mode of SIM-14 does not check for legal instructions; thus all Chapter 9 instructions will be executed in this mode. The SKE and SKZ instructions (octal codes 6000 through 6777) are exceptions which, because of the internal organization of SIM-14, may not be executed in on-line mode. (SIM-14 will change any such instruction to set an output instruction with a program stop assigned at that location for on-line execution.)

Execution in on-line mode of instructions which affect the PDP-14 output register (e.g., TXD, TYD and TRM) have no effect upon SIM-14.

Table 8-1
PDP-14 Instructions Recognized by SIM-14

Symbolic Instruction	Octal Equivalent
TYN	1400-1777
TYF	1000-1377
TXN	2400-2777
TXF	2000-2377
SYN	3400-3777
SYF	3000-3377
JFN	5400-5777
JFF	5000-5377
TXD	7000-7377
TYD	7400-7777

Symbolic Instruction	Octal Equivalent
JMP	4224 NNNN
JMS	4645 NNNN
TRM	4226 NNNN
JMR	0354
SKP	0344
NOP	0000
CLR	3377

SIM-14 COMMAND DESCRIPTION

SIM-14 has many commands to control program debugging. Some commands are recognized only in local mode; others are recognized only in on-line mode; still others are recognized in both modes. If the user types a command which is not recognized in the present mode of operation, SIM-14 types the error message M? (mode error). The command which caused the error will be ignored and the user may either change modes or type a new command. (Commands which are not recognized by the truth table mode or the equation verification mode, will also generate an M? error message when used in the wrong mode.)

All commands in SIM-14 are terminated by a carriage return. When the carriage return is typed, SIM-14 will act upon the command. Any spaces typed by the user are ignored.

There are three types of commands in SIM-14. If a command is not typed in its proper form the error message S? (syntax error) will be typed. The forms of the three SIM-14 command types are:

a. One or two letters which stand alone.

b. One or two letters followed by a single number.

c. One or two letters followed by two numbers separated by a minus sign (-). The first number must be less than the second number.

Any command which is of the last form may also be typed in form b. Specific examples of these command forms will be covered in the following pages. It is important that the reader understand that there are three forms and that a command must be typed in the form expected by SIM-14.

Commands in this manual are presented as specific examples without giving the general form. Note that commands which have limits (e.g., form c above) may be typed in form b if only one entity is affected by the command.

The user may substitute any PDP-14 program address, PDP-14 input or output program number, or other parameter for those given for illustration purposes in this manual.

PDP-14 GENERAL COMMANDS

Once SIM-14 has been loaded into the PDP-8 computer and started (see Chapter 10 for these operating procedures), the user may type commands to SIM-14 and begin program debugging. SIM-14 is initially in local mode when loaded, and it signals readiness to accept commands by typing its version number (e.g., V2 is version 2) and then a period at the left margin of the Teletype paper.

LM Command

If SIM-14 is in on-line mode, or if the user is in doubt as to the current mode of operation, local mode can be entered by typing the command LM, followed by a carriage return.

```
.LM )
                SIM-14 is now awaiting a local mode command.
.
```

OM Command

When the user has completely debugged his program in local mode and wants to test the program with the machinery to be controlled, he may enter on-line mode by typing the command OM, followed by a carriage return.

```
.OM )
                SIM-14 is now awaiting an on-line mode command.
.
```

On-line mode execution is possible only when the PDP-14 has been completely installed, with the machine inputs and outputs wired to the I and O boxes of the PDP-14. The ROM for the PDP-14 is not in place, however, and the PDP-8 interface modules and cable must be installed. If any of these steps has not been taken, or if the PDP-14 is not running, SIM-14 will type NR? and will not enter on-line mode.

RUBOUT

If the user types a command incorrectly or wishes to have SIM-14 disregard a command for any reason, he may strike the teletype RUBOUT key, if he has not already typed a carriage return to terminate the command. SIM-14 will record the RUBOUT as the character @.

RUBOUT

```
.MB2@ )
.MB3 )
.
```

The user meant to type MB3 and therefore struck the RUBOUT key at this point. SIM-14 ignored the command and typed the @, the carriage return and period to signal readiness to accept a further command. The user then typed MB3, followed by a carriage return.

Note that the RUBOUT key can be used only to ignore commands which have not been acted upon by SIM-14 (viz., those which have not yet been followed by a carriage return).

LOCAL MODE PROGRAM DEBUGGING

Figure 8-1 illustrates the suggested sequence for using the local mode features to debug programs. It is strongly suggested that the user pass through all stages of the debugging procedure to insure a methodical approach to program testing.

MB Command

The user should specify the maximum size of the PDP-14 program to be debugged so that SIM-14 may check for jumps which transfer control out of the simulated program, as well as attempt to store instructions in nonexistent memory. The command used in local mode of SIM-14 to specify program size is MB (memory banks) followed by a 1, 2, 3, or 4, to specify the size of the PDP-14 program (1K, 2K, 3K or 4K).

```
.MB3 )
.
```

SIM-14 will allow up to a maximum 3K program (i.e., addresses 0 through 5777).

Unless otherwise specified, SIM-14 will assume four memory banks if an 8K PDP-8 is used for simulation, and one memory bank if a 4K PDP-8 is used. Furthermore, if SIM-14 is loaded in a PDP-8 computer with 4K of core memory, only the MB1 command is legal (viz., only 1K PDP-8 programs may be simulated).

The memory addresses which may be used in PDP-14 programs of any number of memory banks are summarized in the following.

No. of Memory Banks	Memory Addresses
1	0000 - 1777
2	0000 - 3777
3	0000 - 5777
4	0000 - 7777

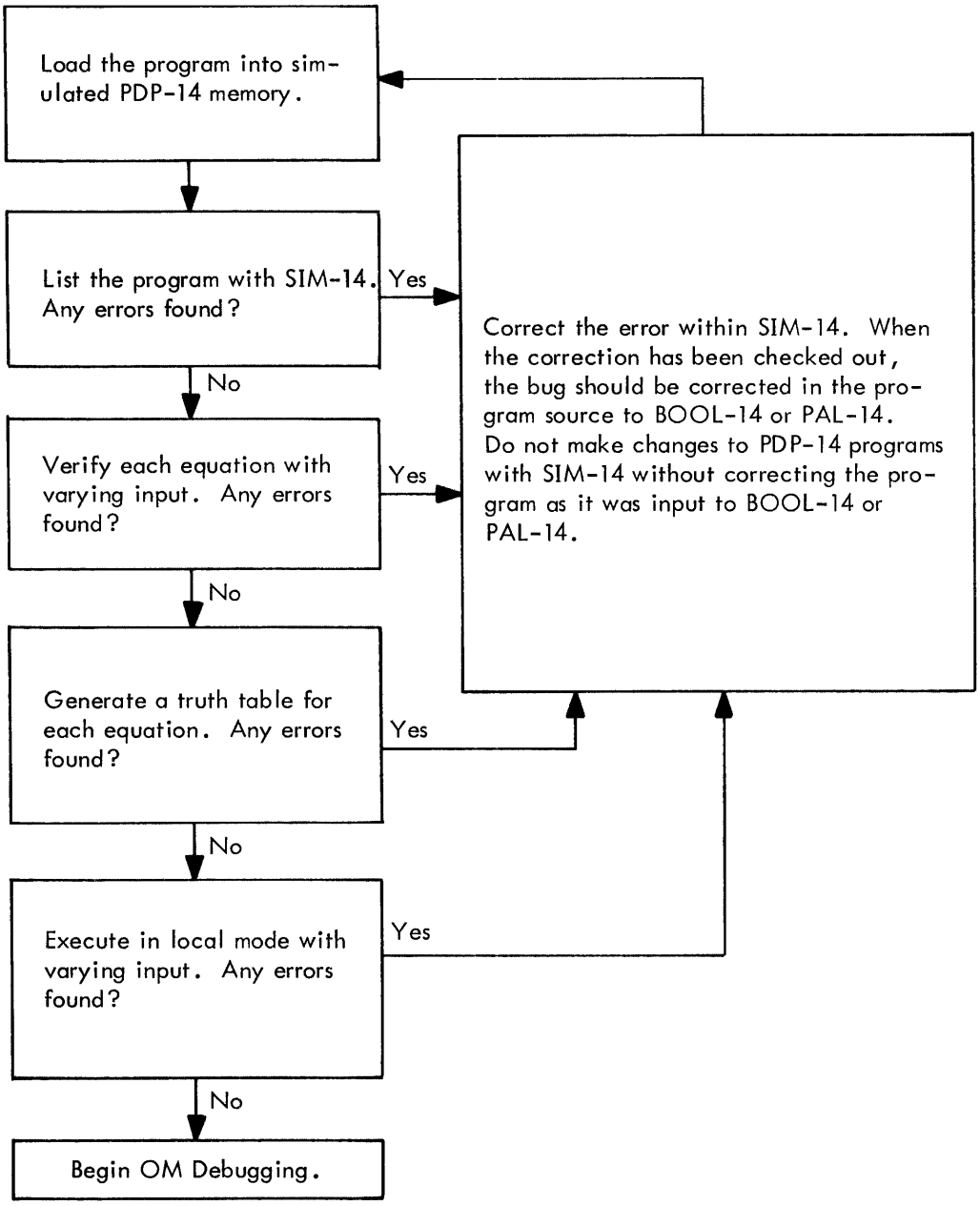


Figure 8-1 Local Mode Debugging Procedure

PROGRAM MANIPULATION AND MODIFICATION COMMANDS

The following set of SIM-14 commands enables the user to read paper tape programs, make changes to these programs, list these changed programs, and punch new paper-tape records of the changed program. These commands are most often used in local mode, but may also be used in on-line mode.

Since changes may readily be made to PDP-14 programs with SIM-14, the user must keep an accurate record of the current state of his program. This may be in the form of SIM-14 generated listings; however, it is strongly recommended that the user maintain a correct and current version of his program in its original "source" form (i.e., BOOL-14 or PAL-14 input). If program changes of large magnitude are needed at a later date, the user will want a correct and debugged version of his original program in the format of the BOOL-14 compiler, or the PAL-14 assembler, so that he may make changes with the PDP-8 Editor (see Chapter 10). A correct and current program listing and paper tape record of each PDP-14 program in BOOL-14 or PAL-14 source form is essential if changes are to be made in the future.

R Command

The user types R followed by a carriage return to direct SIM-14 to read a paper tape from the low-speed reader unit of the Teletype console. The paper tape must be in binary form as output from the BOOL-14 compiler, the PAL-14 assembler, or SIM-14 itself.

The loading procedure for paper tapes with the low-speed reader in SIM-14 is:

a. Type R followed by RETURN key.

b. Place paper tape in the low-speed reader of the Teletype unit. The leader section of the tape must be positioned over the reading head.

c. Switch the reader to START.

d. The tape is read by SIM-14.

e. When the tape stops, switch the reader to STOP.

f. Remove the paper tape from the reader.

g. Press the CONT switch on the PDP-8 computer console.

```
.R )
.
```

The user types R followed by a carriage return, then places a binary program tape in the low-speed reader, and switches the reader to START. After the tape is read, the user switches the reader to OFF, removes the tape, and presses the PDP-8 console switch marked CONT. SIM-14 then types the period (.) to signal readiness to accept further commands.

RH Command

The RH command is used to read paper tapes with the high-speed reader, if the PDP-8 computer used for simulation is equipped with this device. The binary paper tape (output from SIM-14, BOOL-14, or PAL-14) is first placed in the reader unit; the user then types RH followed by a carriage return. SIM-14 will read the paper tape record into memory, and type the period at the left margin to signal readiness for a new command. (The user does not need to press the CONT switch of the PDP-8 console as in the R command.)

.RH ↵	After placing the paper tape in the high-speed reader, the user types RH followed by a carriage return. The tape is read and SIM-14 types a period when it is ready for a new command.
.	

The leader section of the paper tape must be positioned over the reader head before the RH command is typed.

When the R or RH command is used to read a SIM-14 generated paper tape for a PDP-14 program requiring more than one memory bank, the command must be retyped for each successive memory bank after the first until the whole tape is read. (For a full 4K program punched by SIM-14, the R or RH command must be typed four times.)

When either the R or the RH command reads a paper tape for a program which requires more PDP-14 memory than was reserved by the user with the MB command, SIM-14 types the error message MOV (memory overflow). The user must either reorganize his program to fit within the allotted number of memory banks or he must add more memory banks to the simulated memory with the MB command.

SIM-14 types the error message RDER (reader error) when a checksum error occurs while reading a paper tape with the R or the RH command. This error message indicates that the tape was incorrectly read and should be reread. If the same tape generates this error consistently, the paper tape is bad and should be repunched from BOOL-14 or PAL-14.

Open Location Commands

Open a Location Symbolically - A colon (:) typed following a legal PDP-14 address will cause the content of that location to be typed symbolically (i.e., with its instruction mnemonic). The user may then alter the content of that location by typing a legal PDP-14 instruction next to the content typed by SIM-14, and terminating the line with a carriage return. If he does not wish to change the content of the location, he simply types a carriage return after the instruction typed by SIM-14.

.10:NOP TXN 7 ↵	The user changed the content of location 10 from NOP to TXN 7; modification was terminated by a carriage return and SIM-14 awaits a new command.
.	

Alternatively, the user may terminate a line which is typed by SIM-14 (and which he may or may not have modified) by a line feed. Lines so terminated will cause the next sequential location to be automatically opened for modification. If a line which was opened symbolically is closed with a line feed, the next location will also be opened symbolically. This feature may be used to scan a series of PDP-14 instructions by continually typing line feed.

```
.10:TXN 007 ↓
0011:NOP TXN10 )
.
```

The user opened location 10 but did not modify the content. Termination by a line feed caused the next sequential location to be opened for modification. The user terminated this second line with a carriage return and SIM-14 responded by typing a period to request a new command.

The following example shows the way in which a PDP-14 program could be completely typed. Each line is terminated by the line feed key to automatically open the next sequential location. If a line is terminated by a carriage return, the line feed may be typed at any time to open the next location, as illustrated in the example below.

```
.60:NOP TXN2
.
0061:NOP JFN65
0062:NOP TXN3
0063:NOP TYF10
0064:NOP JFN67
0065:NOP SYN10
0066:NOP SKP
0067:NOP SYF10
.
```

The user typed a carriage return after modifying location 60; SIM-14 responded with a period. Since the user really wanted to enter a group of instructions, he typed a line feed and the next location (61) was opened. The user continued to type program instructions, terminating each line with a line feed.

Open a Location Numerically - A slash (/) typed following a legal PDP-14 address will cause the content of that location to be typed numerically. The user may alter the content, close the location unaltered, or close the location (altered or unaltered) and open the next sequential location. This function may also be used with the colon (:) previously described to open locations in both symbolic and numeric form.

```
.65/3410 ↓
0066/4224 ↓
0067/0100 )
.
```

The user opens three sequential locations numerically by following each but the last line with a line feed. No modifications were performed although the user does have this option.

The following example illustrates both the symbolical and numerical opening of registers. The same register may be opened in both forms on the same line. No modifications were performed although they could have beer made at any time.

```
.60/2402
0061/5465
0062/2403
0063/1010 :TYF 010
0064:JFN 070
0065:SYN 010 /3410
0066/4224
0067/0100
0070/3010
0071/4224
.
```

The user opened location 63 symbolically, after it had been opened numerically, by simply typing the colon (:). By following the line with a line feed, the next location is opened in the same form. Thus, locations 64 and 65 are opened symbolically. At location 65, the user reverts to opening locations numerically. Line 71 is terminated by a carriage return.

LS Command

Once a program is loaded into memory, the LS command is used to list the program symbolically. The program is listed with instruction mnemonics for all locations within the specified limits. The second part of two word instructions (viz., JMS, JMP and TRM) are typed numerically. Any location within the limits which contains a number other than those given in Table 8-1 is also typed numerically. The user specifies the limits of the listing by following the LS with two numbers. For illustration purposes, the listing is shown in two columns.

```
.LS200-230
0200:TXN 011
0201:JFF 207
0202:TXN 001
0203:TXN 002
0204:JFF 207
0205:SYN 001
0206:SKP
0207:SYF 001
0210:JFF 215
0211:NOP
0212:NOP
0213:NOP
0214:NOP
0215:JMS
0216:1000
0217:JFF 227
0220:JMS
0221:1012
```

The user requests SIM-14 to list the locations from 200 to 230 inclusive. If the user has not altered the content of M and W, all locations within these limits are typed.

The user lists the complete set of instructions by not altering the initial values of M and W (namely M=0000 and W=NOP).

```
0222:JFF 207
0223:TYN 001
0224:JFF 227
0225:SYN 005
0226:SKP
0227:SYF 005
0230:JFF 234
.
```

The user may modify the function of this command to list only certain program instructions, or he may list all program instructions within the limits by not exercising the option. The user controls the locations listed by changing the contents of two special registers, W (word) and M (mask).

When these locations are modified, the LS command will type only certain locations within the limits specified, thereby searching for particular instructions. The following examples illustrate the use and power of this feature.

The user wishes to list:	He changes M to:	He changes W to:
all SYF or SYN instructions	7000	SYF or 3000
all SYF instructions	7400	SYF or 3000
all SYN instructions	7400	SYN or 3400
all JMP instructions	7777	JMP or 4224
all JMS instructions	7777	JMS or 4645
all JMR instructions	7777	JMR or 0354
all JFF or JFN instructions	7000	JFF or 5000
all JFF instructions	7400	JFF or 5000
all JFN instructions	7400	JFN or 5400
all TXN or TXF instructions	7000	TXF or 2000
all TXF instructions	7400	TXF or 2000
all TXN instructions	7400	TXN or 2400
any particular instruction	7777	the instruction
all locations within limits	0000	NOP or 0000
a particular operand	0377	the operand

The user alters the locations M and W in the same way that he alters PDP-14 program locations. He uses the colon or slash to open the location and then changes the content and closes the location with a carriage return.

```
.M/000 7000 ↵
.
```

The user changes the mask to 7000.

```
.W:NOP SYF ↵
.
```

The user changes the word to SYF.

The following examples list different instructions within the same limits by changing the M and W locations.

```
.M/0000 7000
.W:NOP SYF
.LS230-267
0235:SYN 001
0237:SYF 001
0225:SYN 005
0227:SYF 005
0244:SYN 006
0246:SYF 006
0264:SYN 007
0266:SYF 007
.
```

The user changes M and W to list all set output instructions within the limits of the command.

```
.M/7000 7400
.W:SYF 000 SYN
.LS200-267
0205:SYN 001
0225:SYN 005
0244:SYN 006
0264:SYN 007
.
```

The user changes M and W so that only the SYN instructions are listed.

```
.M/7400 7777
.W:SYN 000 JMP
.LS200-267
0267:JMP
.
```

The user changes M and W to list only the JMP instructions within the limits of the command.

```
.M/7777
.W:JMP JFF264
.LS200-267
.
```

The user changes the content of M and W to have SIM-14 type all the JFF 264 instructions within the limits. There are no such instructions, however, and SIM-14 types the period to request a new command.

```
.M/7777
.W:JFF 264 JFN264
.LS200-267
0256:JFN 264
.
```

The user changes the content of W to list the JFN 264 instructions and SIM-14 types the single such instruction within the specified limits.

LN Command

If the user wishes to list a section of his PDP-14 program in numeric form, he uses the LN command. All locations within the specified limits will be typed. The LN command is subject to the M and W locations described in the LS command.

```
.LN200-235
0200/2411
0201/5207
0202/2401
0203/2402
0204/5207
0205/3401
0206/0344
0207/3001
0210/5215
0211/0000
0212/0000
0213/0000
0214/0000
0215/4645
0216/1000
0217/5227
```

The user requests SIM-14 to list the locations from 200 to 235, inclusive. All locations will be typed (in numeric form) if the user has not altered the content of M and W. Otherwise, each instruction within the limits will be taken from memory, masked with the content of M and then compared with W. The original content of all matches are then typed numerically on the Teletype.

```
0220/4645
0221/1012
0222/5227
0223/1401
0224/5227
0225/3405
0226/0344
0227/3005
0230/5234
0231/0000
0232/0000
0233/0000
0234/4645
0235/1000
.
```

```
.M/0000 7777
.W:NOP JMS
.LN200-235
0215/4645
0220/4645
0234/4645
.
```

The user requests SIM-14 to type numerically all JMS instructions within the limits.

```
.M/7777
.W:JMS SKP
.LN200-235
0216/0344
0226/0344
.
```

The user requests SIM-14 to type all SKP instructions numerically.

The LN command is useful to obtain the necessary octal values needed to modify a wired ROM braid.

P Command

The user may generate a paper tape record of his PDP-14 program on the low-speed punch by using the following SIM-14 procedure.

a. Type P followed by the limits of the program section to be punched.
b. Turn the paper-tape punch unit of the teletype ON.
c. Press the CONT switch of the PDP-8 computer console.
d. After the tape is punched, turn the punch unit OFF and remove the punched tape.
e. Press the PDP-8 CONT switch.

.P0-200 ↵	The user requests SIM-14 to punch a paper tape record of the program instructions in locations 0 through 200, inclusive. When the tape has been punched (by following the procedure above), SIM-14 types the period (.) to request a new command.
.	

PH Command

To punch a paper tape record on the high-speed punch, type PH (punch high) followed by the limits of the program section to be recorded. SIM-14 punches the paper tape without further operation by the user (viz., he need not follow the procedure above).

.PH300-1500 ↵	The user types PH followed by the limits to be punched. The command is terminated by a carriage return. SIM-14 punches the tape, and types a period (.) when it is ready for a new command.

The high-speed punch button must be activated to punch the paper tape. It may be shut off after the tape is punched.

The tape will be punched by the P or PH command in 1K segments. The leader for the first 1K segment (addresses 0000 to 1777) will have a single row of punched holes on the right edge of the paper tape; the second 1K (2000-3777) will have two punched rows; the third 1K (4000-5777) will have three punched rows; and the leader for the fourth 1K (6000-7777) will have four punched rows.

VERIFICATION OF A PAPER TAPE PROGRAM RECORD

Once the user has punched a paper tape record of his program, he may use the following commands to check that no punching errors have occurred.

.V ↵	Verify the paper tape in the low-speed reader by comparing it with PDP-14 simulated memory and noting any discrepancies.
.VH ↵	Verify the paper tape in the high-speed reader by comparing it with PDP-14 simulated memory and noting any discrepancies.

The first command above uses the low-speed reader unit of the teletype console. The following procedure should be followed.

a. Type the command V, followed by a carriage return.
b. Place the paper tape in the low-speed reader.
c. Switch the low-speed reader to START.

d. When the tape stops reading, switch the low-speed reader to FREE, and remove the paper tape.

e. Press the PDP-8 CONT switch.

When using the VH command it is not necessary to follow the above procedure. Simply place the paper tape in the reader and then type VH, followed by a carriage return. The tape will be read and return to SIM-14 will occur without pressing the CONT switch.

The V or VH command must be used for each 1K segment in the paper tape to be verified.

The tape verify commands compare the tape being read with the simulated memory within SIM-14. If the content of the tape does not represent the content of memory, SIM-14 types the following message:

1234 3774 2774

where the first number typed is the address of the discrepancy, the second number is the numeric representation of the content of the location in the simulated memory, and the last number is the numeric representation of the content of the location on the paper tape. Thus, the above message means simulated memory location 1234 contains the PDP-14 instruction 3774, while the tape being verified has contents 2774 for location 1234. A new paper tape should be punched and the old tape should be destroyed whenever there is a tape verification error.

If the error message RDER (read error) is typed, the paper tape has a bad checksum punched at its end, and a new tape should be punched.

Before sending a program paper tape to DEC to have an ROM generated, the user must verify the tape to be certain that a punching error did not occur. It is also good practice to verify the paper tapes generated during the debugging stage to be certain that they are accurate. Punching errors which cause verify errors do not occur often, but such errors could change the user's program with dangerous results.

EQUATION VERIFICATION COMMANDS (LOCAL MODE)

Once the PDP-14 instructions for solving a particular output have been stored in the SIM-14 memory, the user begins debugging the program, using the equation verification feature. The user supplies the identity of the variables of the equation, their state (ON or OFF), and the starting address of the equation to be verified. SIM-14 responds by typing the value of the output, based on the supplied inputs. SIM-14 also notes any supplied variables which were not actually part of the solved equation for the output (listed, not encountered), and any unsupplied variables which were part of the solved equation (encountered, not listed).

The commands for verifying an equation are:

VY20	Verify the equation for output 20. SIM-14 will now await user specification of the variables used in the equation.
XN5-7	User specifies that inputs 5 through 7 are in the equation, and that their state is ON.
XF 10-13	User specifies that inputs 10 through 13 are in the equation, and that their state is OFF.
YN 5-6	User specifies that outputs 5 and 6 are in the equation, and that their state is ON.
YF 11	User specifies that output 11 is in the equation and that its state is OFF.
S200	User instructs SIM-14 to begin execution of the program at location 200. This should be the starting address for the equation to be verified.

All of the assignment commands above (XN, XF, YN, or YF) can be typed in either the multiple variable form (e.g., XN 5-7) or the single variable form (e.g., YF 11).

The messages that may be typed by SIM-14 are:

Y020	1	Output 20 is turned ON by the user-supplied variables.
Y020	0	Output 20 is turned OFF by the user-supplied variables.
X005	E	Input 5 was encountered by SIM-14 when solving the equation, but it was not listed by the user.
Y011	L	Output 11 was listed by the user but it was not encountered by SIM-14 when solving the equation.
Y100	1023 ?	A set output command for Y100 was encountered at location 1023 when SIM-14 was verifying a different output. Indicates that the user supplied the wrong starting address.

A maximum of 63 variables may be included in one verification. If this number is exceeded, SIM-14 types the message TOV (table overflow). This limit should be of little consequence because few user programs contain equations which have as many as 63 variables.

Continue Verification

The user may repeatedly verify the same equation with varying states of variables by using the CV (continue verification) command. After one verification, the user may change the states of variables with the XN, XF, YN or YF commands. The last entered state of a variable is the state used by SIM-14 for verification; thus, the user may change variable states at will. Those variables which are not re-entered after the CV command is typed, retain their previous state.

If an E error (a variable encountered but not listed) occurred on the previous verification, the user may enter new variables after the CV command has been typed. When a bug is encountered in a program which necessitates changing the PDP-14 instructions, the user should restart verification with the VY command. (The CV command should not be used to continue verification after a change is made to the program being debugged.)

To illustrate the use of the program verification feature, consider the following equation to be solved by the PDP-14.

$$Y1=(X1*X2+X3)*/X4*X5$$

As an example, the user generates the following PDP-14 instructions to solve the equation (errors are included by intent).

```
•LS350-364
0350:TXF 011
0351:TXF 002
0352:JFF 355
0353:TXN 003
0354:JFN 362
0355:TXN 004
0356:TXN 005
0357:JFN 362
0360:SYN 001
0361:SKP
0362:SYN 002
0363:JMP
0364:0400
•
```

From analyzing the equation, the user knows that there are two sets of conditions which will cause the output to be set ON; namely, when

X1 is ON and X2 is ON and X4 is OFF and X5 is ON	OR	X3 is ON and X4 is OFF and X5 is ON

The user then checks the program with the equation verification feature to determine whether the program will correctly solve for the output with varying input.

```
•VY1
•XN1-3
•XF4
•XN5
•S350
```

The user begins the verification by typing the initial input and the starting address.

```
Y002 0362 ?
.362:SYN 002 SYN1
.VY1
.XN1-3
.XF4
.XN5
.S350
```

SIM-14 encountered a SYN2 instruction at location 362 when verifying output 1. The user changes the instruction to a SYN1, and restarts verification.

```
Y001 1
X001 L
X011 E
.350:TXF 011 TXF1
.VY1
.XN1
.XN2-3
.XF4
.XN5
.S350
Y001 1
.CV
.XN4
.S350
Y001 1
.M/0000 7777
.W:NOP SYN1
.LS350-365
0360:SYN 001
0362:SYN 001
.362:SYN 001 SYF1
```

SIM-14 reports that with the supplied input, Y1 is now ON; however, X1 was listed by the user and not encountered in the program. X11 was encountered in the program, but was not listed by the user. The user thus discovers that the TXF11 instruction at location 350 should be a TXF1. He then restarts verification. (He does not continue the verification because variable 11 was entered in the input table by SIM-14 when it was previously encountered. To delete any variables, the verification must be restarted.) SIM-14 reports that the output is ON with the given input. The user continues verification, changing the value of X4 to ON. The user finds that Y1 remains ON when it should be OFF. He decides that there must be an extra SYN1 instruction in the program and changes the mask (M) and word (W) to type all such instructions. The SYN1 instruction at location 362 should be an SYF1.

```
.VY1
.XN3-5
.XF4
.S350
```

The user then restarts verification. (Assigning input 4 ON and then OFF is the same as assigning it OFF since only the last value is used by SIM-14.)

```
Y001 0
X001 E
X002 E
.
```

The user did not list inputs 1 or 2 and the output is OFF with the supplied input. The program still contains at least one bug since the output should have been ON when inputs 3 and 5 are ON and 4 is OFF.

```
.M:7777 0
.W:SYN 001 NOP
.LS350-362
0350:TXF 001
0351:TXF 002
0352:JFF 355
0353:TXN 003
0354:JFN 362
0355:TXN 004
0356:TXN 005
```

The user decides that he should have a listing of his program with its changes at this point. Since he has changed the values of M and W, he must now return them to the original values of 0000 (or NOP). The listing continues below.

```
0357:JFN 362
0360:SYN 001
0361:SKP
0362:SYF 001
```

```
•354:JFN 362 JFF362
•VY1
•XF1-2
•XN3
•XF4
•XN5
•S350
Y001 0
•CV
•XN1-2
•XF3
•XF4
•XN5
•S350
Y001 0
```

After scanning the program, the user sees the bug which caused the output to be set OFF instead of ON during the last verify case. He changes the JFN to a JFF at location 354, since the outputs should be OFF only if the flop is OFF (not ON). He then restarts the verification and finds that the output remains OFF. Confused at this point, he tries to solve the equation with the original set of input, and discovers that again the output is OFF.

```
•356:TXN 005 TXF5
•VY1
•XN1-2
•XF3
•XF4
•XN5
•S350
Y001 1
```

Examining the program again, the user notices that the instruction in location 356 should be TXF5 and not TXN5, since the JFN instruction jumps to the set output OFF. With this final bug removed, the program solves all input supplied to it.

```
•CV
•XF1-2
•XN3
•S350
Y001 1
•CV
•XF4
•S350
Y001 1
•CV
•XN4
•S350
Y001 0
•CV
•XF4
•XN5
•S350
Y001 1
•CV
•XF5
•S350
Y001 0
•
```

The user continues to supply varying input to test the program until he is satisfied that all bugs are removed.

The user should realize that most of the bugs detected in the preceding example could have been removed by carefully reading the program. The user can save valuable debugging time by carefully proofing his program before entering the debugging stage. The important point to realize, however, is that SIM-14 allows the user to completely check and find bugs in his program using the equation verification feature without endangering the valuable equipment to be controlled.

Equation Verification for Subroutines (Z-Functions)

When the user program contains subroutines, the equation verification process of SIM-14 may also be used to debug the program.

Repetitive equations (subroutines which actually set an output ON or OFF) may be debugged as a regular PDP-14 output verification. (This type of subroutine is called an R-function in BOOL-14.)

Repetitive variables or Z-functions (subroutines which do not set an output) may be debugged with the equation verification feature of SIM-14 using the VZ command, in the same way that regular outputs are verified with the VY command. The VZ command is typed, followed by the input states of the variables (using the XN, XF, YN or YF commands). The starting address of the subroutine is specified with the S command and SIM-14 types out the resultant state of the subroutine as recorded in the TEST flag when the JMR instruction is executed. The subroutine (Z function) is specified by its starting address. Thus, to verify the subroutine starting at location 1000, the user types VZ1000.

It is important to remember that the numbers used to identify R- and Z-functions in SIM-14 are not the same as used in BOOL-14. In BOOL-14, for example, an R-function would be written as:

```
R28:    Y15=X12 + Y23 */X13
```

To verify this function in SIM-14, simply verify output Y15 (i.e., VY15).

In BOOL-14 a Z-function would be written as:

```
Z19=X123 * (Y5 + X2*X3)
```

When the Z-function is compiled by BOOL-14 it is assigned to memory locations. Assuming that the first address of the instruction sequence generated by BOOL-14 is 35, the above Z-function would be verified with the command VZ35 (not VZ19).

Suppose the user has a subroutine to solve the repetitive variable (Z-function)

```
Z1000 = X4 *X12 * (X5 + Y1)
```

The subroutine is listed with the LS command below.

```
.LS1000-1010
1000:TXF 004
1001:TXF 002
1002:JFN 010
1003:TXN 005
1004:TYN 001
1005:JFF 010
1006:TXF 010
1007:TXN 010
1010:JMR
.
```

```
.VZ1000
.XN4-5
.YF1
.XN12
.S1000
```

The user begins verification specifying states for the inputs to the equation (X4, X12, X5 and Y1). X10 is used as a dummy variable and is not listed.

```
Z1000 0
X012 L
X002 E
X010 E
.1001:TXF 002 TXF12
.VZ1000
.XN4-5
.YF1
.XN12
.S1000
Z1000 1
X010 E
```

SIM-14 computes the state of Z1000 and notes that input 12 was listed, not encountered, and input 2 was encountered, not listed. The program was in error and the instruction is changed. The X010 E error was ignored since input 10 is the dummy variable used to set the test flop.

```
.VZ1000
.XN4
.XF5
.YN1
.XN12
.S1000
Z1000 1
X010 E
.VZ1000
.XF4-5
.XN12
.YN1
.S1000
Z1000 0
X010 E
.VZ1000
.XN4-5
.YN1
.XF12
.S1000
Z1000 0
X010 E
.
```

Continued verifications indicate that the subroutine is now fully debugged. The encountered X10 error continues to be typed, but is ignored since it does not indicate an error in the program.

The continue verification command (CV) can be used to verify a subroutine in the same way that it is used to verify an output.

Verifying an Equation which Contains Subroutines

The user program which contains subroutines (Z-functions) as elements of an equation may be verified in two ways. The user may arbitrarily assign the result of the subroutine ON or OFF, or he may enter the states of all the variables which determine the state of the subroutine. In the first case, the subroutine is not solved by SIM-14; it is arbitrarily considered ON (or OFF) to determine the verification result. In the second case, the subroutine will be solved with the user supplied input to determine the verification result.

The commands used to include subroutines in an equation verification are:

Z1400	A subroutine (Z-function) which begins at location 1400 is part of the equation being verified. (SIM-14 types any variables in the subroutine which have not been previously listed by the user. All variables listed by this command are considered OFF unless they are later assigned ON by either the XN, or the YN instruction.)
ZN1450	A subroutine (Z-function) which begins at location 1450 is part of the equation being verified and is considered ON. (The actual subroutine is not solved; the result of the subroutine is arbitrarily considered ON for equation verification purposes.)
ZF1500	A subroutine (Z-function) which begins at location 1500 is part of the equation being verified and is considered OFF. (The actual subroutine is not solved; the result of the subroutine is arbitrarily considered OFF for equation verification purposes.)

The following debugging example corrects the PDP-14 program to solve the equation:

$$Y6 = Z1012 + (/Y1 * Z1000)$$

where the subroutines are defined as:

$$Z1000 = X4 * X12 * (X5 + Y1)$$
$$Z1012 = X15 * (X3 + X4)$$

The user types the program and the subroutines with the LS command (shown in two columns below).

```
•LS254-270
0254:JMS
0255:1012
0256:JFN 264
0257:TYN 001
0260:JFF 266
0261:JMS
0262:1000
0263:JFF 266
0264:SYN 007
0265:SKP
0266:SYF 007
0267:JMP
0270:0000
•LS1000-1010
1000:TXF 004
1001:TXF 012
1002:JFN 010
1003:TXN 005
1004:TYN 001
1005:JFF 010
1006:TXF 010
1007:TXN 010
1010:JMR
•LS1012-1021
1012:TXN 015
1013:JFF 021
1014:TXN 003
1015:TXN 004
1016:JFF 021
1017:TXF 010
1020:TXN 010
1021:JMR
•
```

```
•VY6
•ZN1012
•ZF1000
•YN1
•S254
Y007 0264 ?
•264:SYN 007 SYN6
0265:SKP
0266:SYF 007 SYF6
```

At the first stage of debugging, the user assigns arbitrary values to the subroutines. The command ZN1012 tells SIM-14 that the equation being verified includes a subroutine beginning at location 1012 which should be considered ON when determining the state of output 6. SIM-14 responds by saying that output 7 was discovered at location 264. The user finds that the program should be setting output 6 not 7 and therefore changes it.

```
•VY6
•ZN1012
•ZF1000
•YF1
•S254
Y006 1
```

Since the program was changed, the user restarts the verification. Y6 is ON when Z1012 is ON by verification.

```
•CV
•ZF1012
•ZN1000
•S254
Y006 0
•257:TYN 001 TYF1
•VY6
•ZN1000
•ZF1012
•YF1
•S254
```

The user continues verification and finds that the output is OFF when Z1000 is ON, and Y1 is OFF (it should be ON). Examination of the program shows that output 1 is tested for ON when it should be tested for OFF. The program is changed and the user restarts verification.

```
Y006 1
.CV
.YN1
.S254
Y006 0
.CV
.YF1
.ZF1000
.S254
Y006 0
```

The output is now ON as it should be. Continued verification indicates that all bugs have been removed.

```
.ZN1012

M?
.CV
.ZN1012
.S254
Y006 1
.
```

The user forgot to type the CV command before he supplied input to the verification. Since SIM-14 does not recognize the ZN command outside of the verification mode, a mode error (M?) was typed. The user then supplied the necessary CV and proceeded with the verification.

Note that assigning arbitrary values to the subroutines could cause erroneous solutions for Y6. The user could arbitrarily assign Z1000 ON, representing X4 ON, X12 ON, X5 OFF, and Y1 ON. The user could now verify Y6 by assigning Y1 OFF and Z1012 OFF. Using these assignments SIM-14 concludes that Y6 is ON. Notice that this result is misleading. The state assigned is an impossible state, since Y1 is required to be ON and OFF at the same time. Thus, if Z1012 is OFF, the only way Y6 may be ON is to have X4 ON, X12 ON, X5 ON and Y1 OFF.

```
.VY6
.Z1000
X004
X012
X005
Y001
X010
.Z1012
X015
X003
.YN1
.XN3-4
.XN15
.S254
Y006 1
.CV
.XF15
.S254
Y006 0
```

After removing program bugs by arbitrarily assigning states to subroutines, the user begins to debug the program by allowing SIM-14 to solve the subroutine with user supplied input. The user types Z followed by the address of the subroutine. SIM-14 types all variables needed to solve the subroutine, and enters them in the OFF state. (The user may re-enter them with an XN command if he wants them to be considered ON.) Only those variables which have not been listed previously for the verification, are listed when the user enters a subroutine in the verification.

The user begins verification and SIM-14 replies that the output is ON when computed with the supplied input.

```
.CV
.YF1
.XN4-5
.XN12
.S254
Y006 1
.CV
.YN1
.S254
Y006 0
.CV
.YF1
.XF5
.S254
Y006 0
.CV
.XN5
.S254
Y006 1
.CV
.XF4
.XN15
```

The user continues verification with varying input until he is convinced that the program is working properly. Verification of the output continues below.

```
.XN3
.S254
Y006 1
.CV
.XF3
.XN4
.S254
Y006 1
.
```

TRUTH TABLE GENERATION COMMANDS (LOCAL MODE)

Once the user has verified an equation of his program with the preceding commands, he may generate a truth table for all possible values of input as a permanent record of the equation. SIM-14 will record the resultant state of an output for all combinations of variable states. SIM-14 types an array of 0s and 1s to indicate the OFF or ON state, respectively, of an input or output. The maximum number of variables in the truth table is 63.

The user typed commands are:

TA5	Generate the truth table for output 5 and print all cases; i.e., both ON and OFF output results.
TN6	Generate the truth table for output 6 and print only the ON cases.
TF20	Generate the truth table for output 20 and print only the OFF cases.
X5-11	Inputs 5 through 11 are variables of the equation whose truth table is to be generated.
Y6-7	Outputs 6 and 7 are variables of the equation whose truth table is to be generated.
S230	Start execution of the PDP-14 program at location 230. (Used in truth table generation to specify the starting location of the equation whose truth table is to be generated.)

Generating a Truth Table without Subroutines

The following truth tables are generated from the PDP-14 program instructions listed below by an LS command. There is no command in SIM-14 which will enable the user to type the truth table of a subroutine which does not set an output itself (Z-function). The user may, however, generate the truth table for the equation which contains the subroutine and thus check out the subroutine itself. The generation of truth tables containing subroutines is described in the next section.

The program for which the following truth tables are generated is:

```
.LS600-613
0600:TXN 050
0601:TXF 023
0602:TXN 017
0603:JFF 210
0604:TXN 032
0605:TYN 004
0606:TYF 057
0607:JFN 212
0610:SYN 073
0611:SKP
0612:SYF 073
0613:JFF 220
.
```

SIM-14 types a heading for the truth table which associates each variable with a letter in a list, and then types the letters at the head of the truth table columns. The first column of the truth table contains the values for the variable listed beside the letter A; the second column corresponds to the variable which is next to B; etc. The variables are listed in the order in which they were typed by the user. Thus, the user may arrange his truth table in the form most meaningful to him by typing the variables in a specific order.

If the user begins the generation of a truth table and later wants to prematurely terminate the table being generated, he may do so by setting switch 5 of the PDP-8 console switch register ON. This switch may be used to prematurely terminate any typing being done by SIM-14. The user must set it ON and then OFF. SIM-14 will then type a period, and await a new command. If the user types C (continue), the remaining part of the interrupted truth table will be typed.

```
.TA73
.X50
.X23
.X17
.X32
.Y4
.Y57
.S600
```

The user requests that all values of the truth table for output 73 be typed. He then supplies the variables of the equation. The truth table will be typed in the order that the variables are supplied. The first column of the truth table will be the values of the first supplied variable. When the user has specified the starting address of the equation to be solved, SIM-14 types out the heading and the lines of the truth table. For illustration purposes, the truth table has been shown in three columns; it is actually typed in one long column on the Teletype.

```
A X050
B X023
C X017
D X032
E Y004
F Y057

ABCDEF
000000=0
000001=1
000010=0
000011=0
000100=0
000101=0
000110=0
000111=0
001000=0
001001=1
001010=0
001011=0
001100=0
001101=0
001110=0
001111=0
010000=1
010001=1
010010=1
010011=1
010100=1
010101=1
010110=1
010111=1
011000=0
011001=1
011010=0
011011=0
011100=0
011101=0
011110=0
011111=0
100000=0
100001=1
100010=0
100011=0
100100=0
100101=0
100110=0
100111=0
101000=0
101001=1
101010=0
101011=0
101100=0
101101=0
101110=0
101111=0
110000=0
110001=1
110010=0
110011=0
110100=0
110101=0
110110=0
110111=0
111000=0
111001=1
111010=0
111011=0
111100=0
111101=0
111110=0
111111=0
.
```

The long truth table shown in the previous example is sometimes cumbersome and it takes a relatively long time to output the entries on the Teletype. The user may use the TN or TF commands to generate only the ON or OFF cases, respectively, as seen below. A truth table generated for only the ON (or OFF) cases is just as useful as that which contains all cases, since the user need only remember that all cases not typed must therefore be of the opposite state. The TF truth table on the right below is not complete, but is broken in the middle for illustration purposes.

```
.TN73
.Y4
.Y57
.X17
.X23
.X50
.X32
.S601

A Y004
B Y057
C X017
D X023
E X050
F X032
```

```
.TF73
.Y4
.Y57
.X17
.X23
.X2
.X50
.X32
.S601

A Y004
B Y057
C X017
D X023
E X050
F X032
```

```
ABCDEF
000100=1
000101=1
010000=1
010001=1
010100=1
010101=1
010110=1
011000=1
011010=1
011100=1
011110=1
100100=1
100101=1
110100=1
110101=1
.
```

```
ABCDEF
000000=0
000001=0
000010=0
000011=0
000110=0
000111=0
001000=0
001001=0
001010=0
001011=0
   .
   .
   .
111011=0
111100=0
111101=0
111110=0
111111=0
.
```

Generating Truth Tables for Equations Containing Subroutines

As previously stated, the user may not directly generate a subroutine truth table if the subroutine does not set an output directly. However, the following commands may be used in SIM-14 to generate truth tables for equations which contain subroutines.

Z1400	The subroutine (Z-function) which starts at location 1400 is a variable of the truth table. (All variables in the subroutine not previously listed by the user are added to the truth table.)
ZU1450	The subroutine (Z-function) which starts at location 1450 is to be considered as a unit variable of the truth table. The separate variables which determine the value of Z1450 are not included in the truth table (as in the Z1400 command above). The value of the Z function is arbitrarily considered ON or OFF without actually solving the subroutine.

With the first command, the user may generate for an equation a truth table which contains all the variables which comprise the subroutine. With the second command, individual variables are not considered, and the state of the whole subroutine is varied arbitrarily. The resultant state of the output is determined. This second command is likely to include impossible states in the truth table if there are variables which are both in the subroutine and in the main equation.

The truth tables generated are for the program which was debugged with the equation verification feature previously, namely:

$$Y6 = Z1012 + (/Y1 * Z1000)$$

where

$$Z1000 = X4 * X12 * (X5 + Y1)$$

$$Z1012 = X15 * (X3 + X4)$$

```
•TN6
•ZU1000
•ZU1012
•Y1
•S254

A Z1000
B Z1012
C Y001

ABC
010=1
011=1
100=1
110=1
111=1
```

The user generates a truth table in which the two subroutines are considered unit variables. The values of the subroutines are arbitrarily varied without actually solving the subroutines for specific variables.

```
•TN6
•Z1000
X004
X012
X005
Y001
X010
•Z1012
X015
X003
•Y1
•S254

A X004
B X012
C X005
D Y001
E X010
F X015
G X003

ABCDEFG
0000011=1
0000111=1
0001011=1
0001111=1
0010011=1
0010111=1
0011011=1
0011111=1
0100011=1
0100111=1
0101011=1
0101111=1
1010110=1
1010111=1
1011010=1
1011011=1
1011110=1
1011111=1
1100010=1
1100011=1
1100110=1
1100111=1
1101010=1
1101011=1
1101110=1
1101111=1
1110000=1
1110001=1
1110010=1
1110011=1
1110100=1
1110101=1
```

The user generates a truth table for the equation which considers the varying states of the variables which are represented by the subroutine. The truth table is generated for the ON cases only. The table has been cut in half and the lower part has been moved to the right of the upper part. The complete table appears as one column when typed by the teletype.

```
0110011=1
0110111=1
0111011=1
0111111=1
1000010=1
1000011=1
1000110=1
1000111=1
1001010=1
1001011=1
1001110=1
1001111=1
1010010=1
1010011=1
```

```
1110110=1
1110111=1
1111010=1
1111011=1
1111110=1
1111111=1
.
```

SIMULATED PROGRAM EXECUTION IN LOCAL MODE

Once the user has debugged his program with the equation verification feature and has generated a truth table for each equation solved by the program, he should use the simulated execution feature to test out the program in total. The user supplies input values and starts the execution of the program. SIM-14 reports any changes in the state of outputs.

LM Execution Commands

Simulated execution of programs with SIM-14 is initiated by the following commands.

AS50	Assign an address stop at location 50. (Stop is performed only when switch register 6 is on.) Only one stop is allowed at any one time.
XN1-5	Assign inputs 1 through 5 ON for program execution.
XF6-12	Assign inputs 6 through 12 OFF for program execution.
YN1-4	Assign outputs 1 through 4 ON for program execution.
YF5-15	Assign outputs 5 through 15 OFF for program execution.
CX	Clear all inputs; i.e., set all inputs OFF.
CY	Clear all outputs; i.e., set all outputs OFF.
IX5-11	Interrogate the state of simulated inputs 5 through 11 (i.e., are these inputs ON or OFF).
IY1-20	Interrogate the state of simulated outputs 1 through 20 (i.e., are these outputs ON or OFF).
S400	Begin program execution at location 400. The change of state of any output will be announced by SIM-14 by typing the output and its new state (0 or 1).
C	Continue PDP-14 program execution from the point at which it was interrupted (by either switch register 5, 6, or 9).

SIM-14 maintains two tables of input and output values. The first table is used for equation verification purposes, and the second table is used for simulated execution. The same commands (XN, XF, YN, and YF) are used to enter variables for both features. The table of values to be affected by a particular command is determined by which feature is currently being used. If the user has typed a VY, CV, or VZ command, the equation verification table is affected by the command. If the user has not typed one of the verify commands, the simulated execution table is affected. Thus, the same variable may have two different values at the same time in local mode (e.g., it may be ON for equation verification, but OFF for simulated execution). The reader should also remember that the same input has a real state (ON or OFF) which is used in on-line execution.

The YN and YF commands above allow the user to specify the states of outputs. While these commands may be used in simulated execution, the user is advised to restrict himself to the XN and XF commands. By specifying the states of inputs, the program will specify the state of same outputs. These resultant outputs with other inputs may in turn specify other output states. In brief, since the equipment to be controlled can only specify input states, the user should also restrict himself to specifying input states when testing the program.

Switch Register Options in Simulated Execution

The execution of a program in local mode is controlled by five switches on the PDP-8 computer console. The use of each switch is summarized below.

Switch	Function
5	Terminate Typing - Switch 5 may be used to prematurely terminate the output currently being typed by SIM-14. When this switch is ON, typing is terminated; control returns to SIM-14 when the switch is turned OFF, thereby allowing the user to type further commands to SIM-14.
6	Enable Address Stop - Switch 6 controls the address stop feature of local mode execution, honoring the address stop only if switch 6 is ON.
7	Report Address, Contents, and Test Flag - When switch 7 is ON during program execution, each return to SIM-14 will cause the following information to be typed: the address of the last instruction, the last instruction and the resultant state of the TEST flag. If switch 7 is OFF, only the address of the previously executed instruction is typed.
8	Automatic Return to PDP-14 Program - When switch 8 is ON and a return to SIM-14 occurs during program execution, information is typed according to switch 7 and PDP-14 program execution automatically continues. When switch 8 is OFF, program execution must be continued with an S or C command.

Switch	Function
9	Return to SIM-14 - Control returns to SIM-14 after executing the current instruction, if switch 9 is ON. Returns are affected by switches 7 and 8 as above. Thus, if switches 8 and 9 are ON, the PDP-14 program will be executed one instruction at a time, while reporting information to the user on the Teletype.

Figure 8-2 illustrates the 12 switches which comprise the PDP-8 switch register. The function of switches 10 and 11 (used in on-line mode only) will be described later in this chapter. It is suggested that the user affix labels to the PDP-8 console (above the switches) when using SIM-14. These labels should contain the information given in Figure 8-2 (the use of each switch and the modes in which they are recognized). A useful card incorporating similar information is attached at the back of this manual.

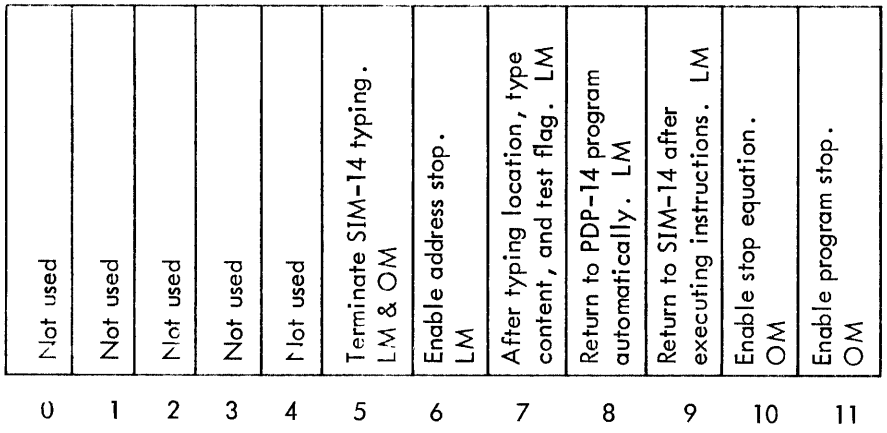


Figure 8-2 Switch Register Functions in SIM-14

Types of Simulated Execution

The command AS allows the user to specify an address at which he wants to have the simulated program execution stopped. The stop is honored only when switch 6 is ON. If the address stop is encountered and switch 6 is ON, the instruction in that location will not be executed, and control will return to SIM-14 which will type the location from which it returned. After typing the address, SIM-14 types a period and awaits further instructions from the user. If the user types C (continue), the instructions beginning with the address stop instruction will be executed, until the address stop is again encountered. The user may change the input and continue execution after each address stop. SIM-14 will type all changes of state in the output values, by typing a message of the form:

	Y123 1	(Output 123 changed from OFF to ON.)
or	Y123 0	(Output 123 changed from ON to OFF.)

Thus, the user may monitor the results achieved by his program for varying input.

When TXD, TYD, or TRM instructions are encountered during simulated execution, SIM-14 types the following messages:

TXD 023 0	meaning TXD 23 encountered and X23 is OFF
TYD 125 1	meaning TYD 125 encountered and Y125 is ON.
TRM 4015	meaning TRM 4015 encountered.

As state above, SIM-14 types the address from which it returned when PDP-14 simulated execution was interrupted. If switch 7 is ON, SIM-14 also types the instruction contained in that location, and the current state of the TEST Flag (i.e., before execution of the typed instruction). If switch 7 is OFF, only the address is typed.

The message typed when switch 7 is OFF is:

1234	(Program execution was interrupted at location 1234)

The message typed when switch 7 is ON is:

	1234 TXN 027 1	(Program execution was interrupted at location 1234 which contains the instruction TXN 27. The TEST flag was ON (1) when the interruption occurred.)
or	0013 SYN 142 0	(Program execution was interrupted at location 13 which contains the instruction SYN 142. The TEST flag was OFF (0) when the interruption occurred.)

Note that the information typed above concerns the next instruction to be executed, if the interruption was caused by an address stop (enabled by switch 6). However, if the interruption is caused by switch 9, as described below, the information typed above concerns the instruction just executed, not the instruction to be executed. This distinction is crucial when one is examining the state of the TEST flag. If execution is stopped by switch 6 and an address stop, the TEST flag value typed is for before instruction execution. If the execution is stopped by switch 9, the TEST flag value is for after instruction execution.

The user may monitor program execution without using program stops by using switch 9. When switch 9 is ON during simulated execution, control is returned to SIM-14 after the execution of the next instruction. The address typed (or the address, instruction and TEST flag value typed) are for the instruction just executed (not the instruction to be executed as in switch 6 and address stops). The user may use switch 9 in two ways: to stop the current execution of a PDP-14 program unconditionally, or to allow the program to be executed one instruction at a time. The second case is often used with switch 8 to achieve a trace of the PDP-14 program.

Switch 8 is used to allow immediate return to the PDP-14 after an interruption (by switch 9, usually). When switch 8 is ON and a return to SIM-14 occurs, the address causing the return is typed, and control is returned immediately to the PDP-14 program. (It is equivalent to the user typing C after each return. It does not permit the user to change any input values.)

The following is a list of commonly used switch combinations.

Switch	Function
6,7	Switches 6 and 7 ON cause the program to be stopped at the address stop location. The instruction and TEST flag value is typed in addition to the address that caused the return. Any changes in output states are also typed. This allows the user to change variables for each complete pass through the program.
8,9	Switches 8 and 9 ON cause the program to be executed one instruction at a time. The address of the last executed instruction is typed, as well as any changes in output states. This allows the user to generate a complete list of the locations encountered during the execution of the program. The user terminates this form of execution by changing switch 8 to the OFF position.
9	Only switch 9 ON, causes the program to execute one instruction for each time the user types the C command. The user may thus change the values of variables at any point in the program. This switch may be used to return control to SIM-14 at any time.
7,8,9	Switches 7, 8 and 9 ON cause the program to be executed one instruction at a time. The address, instruction and resultant TEST flag value of each instruction executed will be typed. The user may stop this form of execution by changing switch 8 to the OFF position.

```
.IX1-3
001 1
002 0
003 1
.IX4-10
004 1
005 0
006 0
007 1
010 0
.IY1-3
001 1
002 1
003 0
```

The user checks the current state of inputs and outputs with the IX and IY commands. SIM-14 types the number of the input or output followed by its state (1 for ON and 0 for OFF).

```
•ASØ
•XF1-3
•XN4-10
•YFØ-376
•SØ

ØØØØ
•C
Y101  1
Y105  1
Y200  1
YØØ3  1
```

The user assigns an address stop at location 0 and supplies values for the variables used in the program. He then directs execution to start at location 0, after setting switch 6 ON.

SIM-14 immediately stops at location 0 since this is the address stop. The user requests execution to continue.

```
ØØØØ
•XF4-10
•XN1-3
•C
Y100  1
Y101  Ø
Y105  Ø
Y106  1
Y200  Ø
YØ53  1
YØØ1  1
YØØ2  1
YØØ3  Ø
```

SIM-14 again stops at location 0 after typing the changes in state of four outputs. The user changes some input values and continues execution.

```
ØØØØ
•XF1
•C
Y100  Ø
Y001  Ø
Y002  Ø
Y003  1
```

Again the execution stops at location 0. The user changes the value of an input, and sets switch 7 ON (switch 6 is still ON).

```
ØØØØ TXN ØØ1  Ø
•XN1
•XF2-5
•C
ØØØØ
ØØØ1
Y100  1
ØØØ2
ØØØ3
ØØØ5
ØØØ6
Y101  1
ØØØ7
ØØ1Ø
ØØ12
ØØ13
Y105  1
```

When execution is stopped at location 0, the instruction and TEST flag value are typed. The user changes some input values. He sets switches 6 and 7 OFF, and sets switches 8 and 9 ON. Thus, each location encountered is listed. Changes in output states are also typed.

```
0014 SYN 105 0
Y106 0
0015 SYF 106 0
Y200 1
0016 SYN 200 0
Y053 0
0017
0020
1500
1501
1502
.
```

The user sets switch 7 ON to have SIM-14 type the instructions and TEST flag values as well as the address.

The user sets switch 7 OFF while execution continues.

The user sets switch 8 OFF to terminate the execution. He could now change the values and restart or continue verification.

Simulated Timer Execution

If the user program contains outputs which are timers, SIM-14 can simulate the timer action in the following manner. When the SYN instruction which turns ON a declared timer in SIM-14 is encountered during simulated execution, the timer output is not turned ON until the SYN instruction has been encountered seven times. Thus, SIM-14 simulates a time delay for the output. The delay may not be varied by the user; each assigned timer is not turned ON until its SYN instruction is encountered seven times. The following are the commands which declare and clear simulated timer assignments.

DT20-23	Define outputs 20-23 as timers. SIM-14 will treat any output so as to simulate the action of a timer. The output will not "turn ON" until the SYN instruction is encountered for the seventh time.
CT	Clear all previously assigned timer outputs.

A maximum of 63 timers may be assigned for simulated execution.

ON-LINE MODE DEBUGGING

Once the user has thoroughly debugged his program with the features of local mode, he may debug his program while actually controlling the machine by sampling true inputs and setting true outputs. Before doing so, however, the installation of the PDP-14 system must be completed. The PDP-8 to PDP-14 interface must also be installed. Only after these steps have been taken and the PDP-14 is running, can the user actually enter on-line mode. If the user types OM under any other conditions, he will receive the error message NR (not running).

When the user executes a PDP-14 program in on-line mode, SIM-14 supplies instructions which are executed by the PDP-14. The supplied program tests the actual machine inputs and sets outputs through the PDP-14 I- and O-boxes. SIM-14 serves the function in on-line mode which will be served by the ROM once the program is fully debugged. On-line mode, however, allows the user to execute the PDP-14 program one section at a time,

by placing in the program, stops which return control to SIM-14. A program stop is only executed when enabled by the user. Thus the program is executed in part or in total as controlled by the user. When a PDP-14 program is interrupted by a program stop, the machine is no longer being controlled because the PDP-14 is no longer testing inputs or setting outputs. Thus, the user must carefully select the points at which he interrupts program execution, so that the machine is not damaged when control is stopped.

The user has the option of specifying a set of operations to be performed when a program stop is encountered and before control is returned to SIM-14. This set of operations is called a stop equation and enables the user to place program stops at points in the PDP-14 program where control could not be stopped otherwise. The stop equation serves as a shut down routine for the equipment before control is terminated. The on-line mode procedure is outlined in Figure 8-3.

On-Line Mode Commands

The following is a list of commands used exclusively in on-line mode. The user may also use the program modification and manipulation commands previously introduced in on-line mode. However, the user is cautioned not to use these commands to make changes to a PDP-14 program in on-line mode without first checking out the changes with local mode debugging.

Command	Function
IX1-15	Interrogate the current state (ON or OFF) of actual machine inputs 1 through 15.
IY12-57	Interrogate the current state (ON or OFF) of the actual machine outputs 12 through 57.
SS234	Set a program stop at location 234. This location must contain a SYN or SYF instruction; otherwise, the program stop will not be set.
SE1567	Specify a stop equation at location 1567. The equation beginning with location 1567 and ending with the number 0010 will be solved (when enabled by switch 10) before any return to SIM-14 control.
TS	Type all stop locations presently specified.
RS234	Remove the stop presently set at location 234.
CS	Clear all stops presently set.
S200	Start on-line execution at location 200. (A program stop must be set before a program may be executed; the program stop need not be enabled, however.)
C	Continue on-line execution from point at which it was interrupted.

The IX and IY commands interrogate the state of the actual machine inputs and outputs, respectively. Since these same commands interrogate the state of simulated inputs and outputs when used in local mode, the user must consider the current mode whenever these commands are used.

The SS (set stop) command is used to place stop points at any SYN or SYF instruction. The user can type all currently specified stop locations with the TS command. He may remove a single program stop with the RS command, or he may clear all stops with the CS command.

On-Line Mode Switch Options

The following switch register options are used in conjunction with the on-line mode instructions to control on-line debugging.

Switch	Function
5	Terminate Typing - Switch 5 may be used to prematurely terminate the current typing of information by SIM-14. When this switch is ON, typing is terminated; control returns to SIM-14 when the switch is turned OFF, thereby allowing the user to type further commands to SIM-14. This switch may not be used to terminate on-line program execution.
10	Enable Stop Equation - Switch 10 controls the stop equation, causing its execution only when switch 10 is ON at the time a program stop is encountered in the PDP-14 program.
11	Enable Program Stop - When switch 11 is ON and a set output instruction, which is specified as a program stop is encountered, the instruction will not be executed if it will cause a change in the state of the output. If a change will occur, the instruction is not executed; control is returned to SIM-14.

Program Stops

When the user executes a PDP-14 program in SIM-14 on-line mode, he must specify at least one program stop. If the user attempts to start execution without at least one program stop set, SIM-14 will type the error message PS ? (program stop ?). The user is not required to enable the program stop, but at least one stop must be set to allow return to SIM-14 when desired by the user. If a stop is not set, PDP-14 program execution could not be terminated.

When a program stop is encountered at a set output instruction, SIM-14 checks enabling switch 11. If switch 11 is ON, the program stop is enabled. However, execution will not be stopped unless a change in the state of the output will occur. SIM-14 will only stop execution if the output is now ON and will be set OFF, or if the output is OFF and will be set ON. The stop is not executed if no change will occur, regardless of switch 11.

When an enabled program stop is encountered and a change in output state will result, the set output instruction is not executed. Control is immediately returned to SIM-14 after executing an enabled stop equation (if any). The set output instruction which caused the return to SIM-14 will be executed when the user types C (continue).

Stop Equations

The stop equation allows the PDP-14 user to perform a set of operations after interrupting program execution and before returning control to SIM-14. The stop equation must return the machine to a stable state so that no operations continue after control is terminated.

The stop equation can either be a special set of instructions which will not be part of the final program, or a section of the regular program which is a shut down routine. The only requirement for a stop equation is that it must be terminated with a location containing 0010. This number will cause SIM-14 to terminate execution of the stop equation and return control to SIM-14. (This number is executed as a NOP if it is encountered other than in the stop equation of SIM-14, or if it is included in the final PDP-14 ROM.)

The stop equation may be changed at will using the SE command. Any location in the program can be specified as the start of the stop equation. The user may specify one stop equation when testing one PDP-14 program section in on-line mode, and another stop equation for another program section.

The stop equation is enabled by switch 10. The stop equation is executed only when an enabled program stop is encountered, which would cause a change of state in the output, and when switch 10 is on.

CAUTION

Switch 10 must never be set ON when the user has not specified a stop equation. If switch 10 and 11 are ON and no stop equation is specified, SIM-14 will assume location 0 as the start of the stop equation when a program stop is encountered.

On-Line Mode Execution

When the user begins to execute his PDP-14 program in on-line mode he must:

a. Interrogate the inputs and outputs with the IX and IY commands to determine what operation should be executed next.

b. Set a program stop at the SYN or SYF which will start that operation. Other program stops may also be set.

c. Enable the program stop with switch 11.

d. Specify a stop equation, if necessary.

e. Enable the stop equation with switch 10.

f. Start program execution with the S (start) command, specifying the initial address.

g. When the program stop is encountered, the SYN or SYF which would cause a change of state will not be executed. Control will return to SIM-14.

h. Specify a new program stop at the SYN or SYF which will start the next operation.

i. If a stop equation is necessary, make certain that the stop equation is appropriate and enabled.

j. Type C (continue).

k. Proceed to check each operation by changing the program stop and stop equation as necessary.

l. Once each part of the program has been checked, turn switch 11 OFF, thus disabling all program stops. The program may then be run in total. If the user wishes to terminate execution, he turns switch 11 ON (with the stop equation enabled by switch 10, if necessary). The program is terminated on the next program stop encountered which will cause a change of state in an output.

m. If any bugs are detected, the user may change the program in on-line mode, but he should return to local mode for further debugging before again executing the program on-line.

n. Once the program has been successfully executed in on-line mode, the user sends the binary program tape generated by SIM-14 (from which the ROM will be woven) to DEC.

o. The fully tested program may be used to operate the equipment until the ROM is received. The PANEL lock feature of the PDP-8/L or PDP-8/I may be used to prevent accidentally stopping the program.

NOTE

Because of the processing time required by SIM-14 when operating in on-line mode, the PDP-14 will operate approximately three times slower than it will when it operates from the ROM stored program. Thus, momentary contact inputs which would not be missed by the PDP-14 operating with an ROM may be missed in SIM-14 on-line mode. If this happens, the user may counteract the slowdown by using subroutines to check crucial inputs more frequently when in the on-line mode.

SIM-14 SUMMARY

The following is a summary of the commands, switch options and error messages for local mode and on-line mode of SIM-14. Specific numbers for inputs, outputs, and program numbers have been used to describe the commands and error messages. It should be noted that any PDP-14 location address, input number or output number may be substituted in place of those used for illustration.

NOTES

SIM-14 commands which reference more than one address, input, or output (such as XN4-12) may be shortened when only one variable is referenced (XN5, not XN5-5).

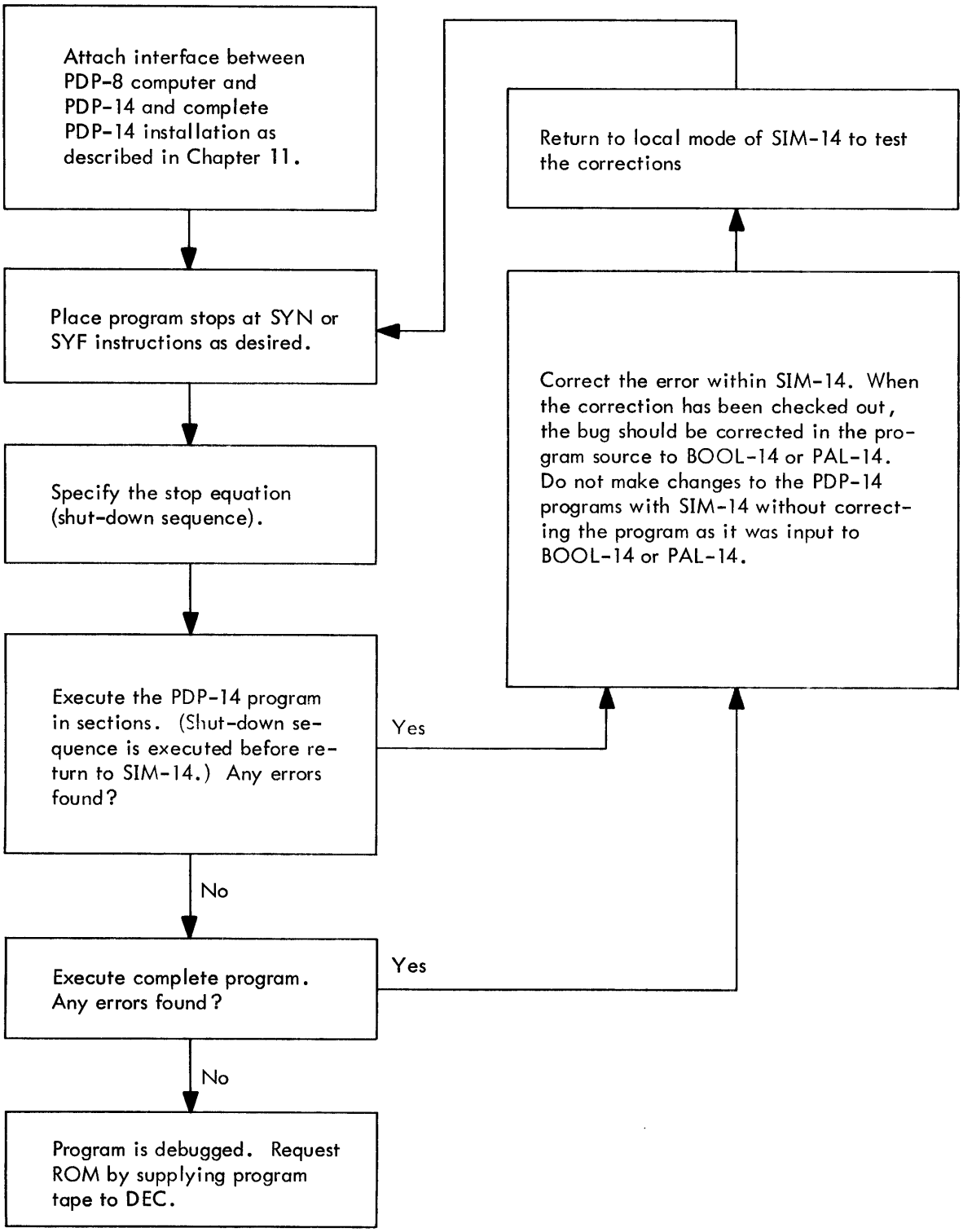


Figure 8-3 On-Line Mode Debugging Procedure

Commands which use the low-speed punch require that the user press the PDP-8 CONT switch both after typing the command, and after the tape is punched. Commands which use the low-speed reader require that the user press the CONT switch after the tape is read.

All commands must be terminated by a carriage return before they will be executed.

Commands Common to Local and On-Line Modes

General Commands

LM	Enter local mode.
OM	Enter on-line mode.
(RUBOUT)	Ignore the previously typed command (up to the last carriage return).
V	Verify the paper tape in the low-speed reader by comparing it with PDP-14 simulated memory and noting any discrepancies.
VH	Verify the paper tape in the high-speed reader by comparing it with the PDP-14 simulated memory and noting any discrepancies.

Program Manipulation and Modification

23:	Open location 23 symbolically and allow the user to modify its contents.
57/	Open location 57 numerically and allow the user to modify its contents.
(RETURN)	Close the currently open location and enter any legal modifications typed by the user.
(LINE FEED)	Close the currently open location; enter any legal modifications typed by the user; open the next sequential PDP-14 location in the same form (i.e., symbolically or numerically) and allow the user to modify its contents.
LS0-30	List the contents of locations 0 through 30 symbolically (i.e., by typing the instructions in mnemonic form).
LN0-10	List the contents of locations 0 through 10 numerically (i.e., by typing the instructions in their numeric (octal form).
P0-700	Punch a paper tape record of the contents of locations 0 through 700 on the low-speed reader.
PH40-70	Punch a paper tape record of the contents of locations 40 through 70 on the high-speed punch.
R	Read a paper tape from the low-speed reader.
RH	Read a paper tape from the high-speed reader.
M/	Open mask location and allow modification.
W:	Open word location and allow modification.

Local Mode Commands

Command	Function
MB2	The PDP-14 memory has two memory banks of 1K words. (MB must be 1 if a 4K PDP-8 is used to simulate operation. MB may equal 1, 2, 3, or 4 if an 8K PDP-8 is used.)
	Equation Verification Feature
VY5	Verify output 5, using the input which follows.
VZ1400	Verify subroutine (Z function) starting at location 1400.
XN4-12	Inputs 4 through 12 are ON for the equation being verified.
XF15-17	Inputs 15 through 17 are OFF for the equation being verified.
YN1-3	Outputs 1 through 3 are ON for the equation being verified.
YF7-11	Outputs 7 through 11 are OFF for the equation being verified.
Z1400	A subroutine (Z-function) which begins at location 1400 is part of the equation being verified. (SIM-14 types any variables in the subroutine which have not been previously listed by the user. All variables listed by this command are considered OFF unless they are later assigned ON by one of the above XN, or YN instructions.)
ZN1450	A subroutine (Z-function) which begins at location 1450 is part of the equation being verified and is considered ON. (The actual subroutine is not solved; the result of the subroutine is arbitrarily considered ON for equation verification purposes.)
ZF1500	A subroutine (Z-function) which begins at location 1500 is part of the equation being verified and is considered OFF. (The actual subroutine is not solved; the result of the subroutine is arbitrarily considered OFF for equation verification purposes.)
S50	Start PDP-14 execution at location 50. (Used with the verify feature to specify the start of the equation to be verified.)
CV	Continue verification of same output with new input. Variables may be added and states of variables may be changed with the XN, XF, YN or YF commands above. However, variables may not be deleted. The S command initiates the verification as above.
	Truth Table Feature
TA5	Generate the truth table for output 5 and print all cases; i.e., both ON and OFF output results.
TN6	Generate the truth table for output 6, and print only the ON cases.
TF20	Generate the truth table for output 20, and print only the OFF cases.
X5-11	Inputs 5 through 11 are variables of the equation whose truth table is to be generated.
Y6-7	Outputs 6 and 7 are variables of the equation whose truth table is to be generated.

Z1400	The subroutine (Z-function) which starts at location 1400 is a variable of the truth table. (All variables not previously listed by the user are added to the truth table.)
ZU1450	The subroutine (Z-function) which starts at location 1450 is to be considered as a unit variable of the truth table. The separate variables which determine the value of Z1450 are not included in the truth table (as in the Z1400 command above). The value of the Z-function is arbitrarily considered ON or OFF without actually solving the subroutine.
S230	Start execution of the PDP-14 program at location 230. (Used in truth table generation to specify the starting location of the equation whose truth table is to be generated.)

Program Execution Feature

AS50	Assign an address stop at location 50. (Stop is performed only when switch register 6 is ON.) Only one address stop is in effect at one time.
XN1-5	Assign inputs 1 through 5 ON for program execution.
XF6-12	Assign inputs 6 through 12 OFF for program execution.
YN1-4	Assign outputs 1 through 4 ON for program execution. (Use of this command should be avoided.)
YF5-15	Assign outputs 5 through 15 OFF for program execution. (Use of this command should be avoided.)
CX	Clear all inputs; i.e., set all inputs OFF.
CY	Clear all outputs; i.e., set all outputs OFF.
DT20-23	Define outputs 20 through 23 as timers. SIM-14 will specially treat any output so assigned so as to simulate the action of a timer. The output will not "turn on" until the SYN instruction is encountered for the seventh time.
CT	Clear all previously assigned timer outputs.
IX5-11	Interrogate the state of simulated inputs 5 through 11 (i.e., are these inputs ON or OFF).
IY1-20	Interrogate the state of simulated outputs 1 through 20 (i.e., are these outputs ON or OFF).
S400	Begin program execution at location 400. The change of state of any output will be announced by SIM-14 by typing the output and its new state (0 or 1).
C	Continue PDP-14 program execution from the point at which it was interrupted (by either switch register 5, 6, or 9).

On-Line Mode Commands

Command	Function
IX1-15	Interrogate the current state (ON or OFF) of actual machine inputs 1 through 15.
IY12-57	Interrogate the current state (ON or OFF) of the actual machine outputs 12 through 57.
SS234	Set a program stop at location 234. This location must contain a SYN or SYF instruction; otherwise, the program stop will not be set. The stop is enabled by switch 11.
SE1567	Specify a stop equation at location 1567. The equation beginning with location 1567 and ending with the number 0010 will be solved (when enabled by switch 10) before any return to SIM-14 control.
TS	Type all stop locations presently specified.
RS234	Remove the stop presently set at location 234.
CS	Clear all stops presently set.
S200	Start on-line execution at location 200. (A program stop must be set before a program may be executed; the program stop need not be enabled, however.)
C	Continue on-line execution from the point at which it was interrupted.

Local Mode Switch Options

Switch	Function
5	Terminate Typing - Switch 5 may be used to prematurely terminate the output currently being typed by SIM-14. When this switch is ON, typing is terminated; control returns to SIM-14 when the switch is turned OFF, thereby allowing the user to type further commands to SIM-14.
6	Enable Address Stop - Switch 6 controls the address stop feature of local mode execution, and honors the address stop only if switch 6 is ON. (Only one address stop may be in effect at one time.)
7	Report Address, Contents, and Test Flop - When switch 7 is ON during program execution, each return to SIM-14 will cause the following information to be typed: the address of the last instruction, and the resultant state of the test flop. If switch 7 is OFF, only the address of the previously executed instruction is typed.
8	Automatic Return to PDP-14 Program - When switch 8 is ON and a return to SIM-14 occurs during program execution, information is typed according to switch 7 and PDP-14 program execution automatically continues. When switch 8 is OFF, program execution must be continued with an S or C command.

Switch	Function
9	Return to SIM-14 - Return to SIM-14 after executing the current instruction if switch 9 is ON. Returns are affected by switches 7 and 8 above; thus, if switches 8 and 9 are ON, the PDP-14 program will be executed one instruction at a time, while reporting information to the user on the teletype.

On-Line Mode Switch Options

Switch	Function
5	Terminate Typing - Switch 5 may be used to prematurely terminate the current typing of information by SIM-14. When this switch is ON, typing is terminated; control returns to SIM-14 when the switch is turned OFF, thereby allowing the user to type further commands to SIM-14. This switch may not be used to terminate on-line program execution.
10	Enable Stop Equation - Switch 10 controls the stop equation, causing its execution only when switch 10 is ON when a program stop is encountered in the PDP-14 program.
11	Enable Program Stop - When switch 11 is ON, and a set output instruction which is specified as a program stop is encountered, the instruction will not be executed if it will cause a change in the state of the output. If a change will occur, the instruction is not executed; control is returned to SIM-14 which types out the location and the instruction which caused the return. If switch 11 is OFF, or if no change in the state of the output will result, the execution of the PDP-14 program will continue uninterrupted.

Error Messages

The following is a list summarizing all possible error messages typed by SIM-14 and their causes.

Error Message	Cause
S?	Syntax error - The user has violated a syntax rule for SIM-14 commands. Each command of SIM-14 is of one of the following forms: 1. One or two letters with no following numbers (e.g., C). 2. One or two letters followed by a number (e.g., S1234). 3. One or two letters followed by two numbers separated by a minus sign (e.g., XN1-23). The first number must be less than the second number. When a command is not typed in its specified format the S? error results. Commands of form 3 may be typed in form 2 without error (e.g., XN5 is the same as XN5-5).

Error Message	Cause
N?	Number error - The user has included an illegal number in his program. The following conditions produce the code N?: 1. A non-octal digit used in a program (i.e., 8 or 9). 2. A test or set instruction references an illegal input or output number (i.e., greater than 377). 3. The address of a JFF or JFN is illegal (i.e., greater than 377). 4. A JMP or JMS instruction references a location which is not in the simulated memory. 5. The second part of a two-word instruction is typed on the same line as the first part (e.g., JMP 1567).
M?	Mode error - The user has typed a command which is illegal in the present mode of SIM-14. SIM-14 has four such modes: local mode, on-line mode, truth table mode, and equation verify mode.
C?	Command error - The user has typed a command that SIM-14 does not recognize.
X123 L or Y123 L	Listed but not encountered error - Input (output) 123 was listed as a variable in an equation to be verified, but was not encountered when SIM-14 actually solved the equation.
X123 E or Y123 E	Encountered but not listed error - Input (output) 123 was not listed as a variable in an equation to be verified, but was encountered when SIM-14 actually solved the equation.
Y123?	Output error - The user has requested equation verify or truth table generation for an output other than 123, and SIM-14 has encountered a SYN 123 or a SYF 123 before the requested output has been solved.
OPER 1234?	Operation error - The instruction stored in location 1234 of the user program, if executed, would cause a transfer out of simulated memory; or SIM-14 encounters an instruction in location 1234 of the user program which it does not recognize.
RDER?	Read error - A checksum error has occurred on a tape read by SIM-14 with the R, RH, V or VH command. The tape should be reread.
PS?	Program stop error - The user has attempted on-line execution of a program without a program stop specified in the program. SIM-14 demands that there be at least one program stop present; otherwise, return to SIM-14 from the user program would be impossible.
TOV	Table overflow - The user has exceeded the highest possible number of variables in equation verify or truth table generation, or the maximum number of assigned timers for simulation.
MOV	Memory overflow - The user has attempted to load a PDP-14 program from a paper tape into memory which is not available for simulation. (The user should allow more memory with the MB command.)
NR?	Not running - The user has attempted to enter on-line mode when either there is no connected PDP-14 or the connected PDP-14 is not running.

CHAPTER 9
MONITORING THE PDP-14

This chapter introduces the instructions which allow a PDP-14 to be monitored externally by a PDP-8 family computer. These instructions are of two types: those which are executed within the PDP-14, and those which are executed within the PDP-8 (but which affect the PDP-14). The only instructions in this chapter which can be simulated by SIM-14 in on-line mode are the TXD, TYD and TRM. This chapter also describes possible approaches and techniques for monitoring the PDP-14 system with a PDP-8.

When used in a monitored or computer interfaced form of operation, the PDP-14 has three facilities for the transfer of information:

a. The Input Register - may be used to supply input data to any internal register of the PDP-14. It may only be loaded by an external computer.

b. The Output Register - may be used to hold data to be transferred to the external computer.

c. The Memory Port - may be used to directly supply instructions to the PDP-14 without passing through the Input Register. Instructions are loaded from the external computer. The PDP-14 executes the instruction and sets a flag when execution is complete.

Three general approaches may be used to monitor an operation through the PDP-14.

a. The PDP-14 could contain instructions which output information to the monitoring computer through the output register, such as TXD, TYD, or TRM. These instructions do not interrupt the programmed operation of the PDP-14. The monitoring computer (PDP-8) merely reads the information, on either a programmed or an interrupt basis, as it appears in the output register of the PDP-14

b. The second technique requires the monitoring computer to interrupt the PDP-14, and then supply an instruction to the PDP-14 for execution through the memory port. The PDP-14 executes the externally supplied instruction and then returns to its programmed operation, as stored in the ROM.

c. The third technique requires the monitoring computer to request an interrupt from the PDP-14 and supply an instruction which causes the PDP-14 to enter the external mode of operation. In this mode, as many instructions as necessary may be supplied to the PDP-14 through the memory port. The PDP-14 will suspend execution of ROM instructions until it is supplied with an instruction causing it to leave external mode.

Thus, the input and output registers may be used without interrupting the execution of the program stored in the ROM.

The memory port may be used to supply instructions to the PDP-14 on a cycle-stealing basis, in which one externally supplied instruction is executed. The memory port may also be used to allow the PDP-8 to completely take over the PDP-14's operation, by supplying all instructions to be executed.

A two-location PDP-14 instruction may not be executed using technique (b) preceding. External mode must be entered before a two-location instruction may be executed on an interrupt basis.

PDP-14 INTERNAL REGISTERS

The monitor instructions involve the PDP-14 internal registers listed in Table 9-1. The following register locks are used in the PDP-14 instructions to refer to the respective registers. The module type for each register is also noted below.

Table 9-1
PDP-14 Internal Registers

Register	Module	Octal Code	Description
PC1	M747	4	Program Counter 1 - specifies the location of the next location in sequence to be accessed. This location contains the next instruction to be executed, except when it is the second part of a two-location instruction. Transfers which use this register as destination may increment or decrement during transfer. (Decrementing PC1 can cause program loops which are improper for the PDP-14.)
PC2	M746	5	Program Counter 2 - used to store the contents of Program Counter 1 while subroutines are being executed to enable a return to the main program. PC2 may also be used to store the value of PC1 whenever the PDP-14 operates in external mode.
INPUT	M746	6	Input Register - used to accept information from a monitoring PDP-8 family computer. This register is loaded externally and its contents may be transferred to other PDP-14 registers. Information cannot be transferred to the Input Register by PDP-14 executed instructions (supplied as part of computer interface).
OUTPUT	M746	6	Output Register - used to send information to a monitoring PDP-8 family computer. This register is loaded from other PDP-14 registers, or from PDP-14 memory directly, and is read by a monitoring PDP-8 family computer. Information cannot be transferred from the Output Register by PDP-14 executed instructions (supplied as part of computer interface).
MB	M746	2	Memory Buffer - the interface between the PDP-14 memory and other internal registers. It is used to hold instructions read from the PDP-14 memory, or loaded from the monitoring computer. The memory buffer also holds the second part of a PDP-14 two-location instruction during its execution.

Table 9-1 (Cont)
PDP-14 Internal Registers

Register	Module Type	Octal Code	Description
IR	M746	1	Instruction Register - used to hold and decode PDP-14 instructions. Instructions which are brought into the Memory Buffer from PDP-14 memory, or which are supplied externally, are transferred into the IR to be decoded and executed.
SPARE	M747	3	Spare Register - used for monitoring and diagnostic functions. If the Spare Register is an incrementing type, transfers which use it as destination may increment or decrement during the transfer. (Not part of basic PDP-14.)
DUMMY	None	0	Dummy - used to transfer all 1's to any register. Dummy is not an actual register but when addressed as the source of a transfer it will act as a register which contains all 1's.

PDP-14 EXTENDED INSTRUCTIONS

The following are the PDP-14 instructions which may be used for diagnostic or monitor functions. They are not required for the basic control uses of the PDP-14 and, except where noted, are not simulated in local mode of SIM-14.

External Mode Instructions

The following instructions cause the PDP-14 to enter or leave the external mode of operation in which all instructions are supplied externally.

Instruction	Purpose
EEM (0600 octal)	Enter External Mode. The EEM instruction causes the PDP-14 to receive all instructions from the monitoring computer. The Program Counter 1, Memory Buffer, and Instruction Register will be affected during the external mode operation. Therefore, the monitoring computer must set Program Counter 1 before returning the PDP-14 to normal operation with the next instruction.
LEM (0400 octal)	Leave External Mode. The LEM instruction will return control to the PDP-14 which will then execute the instructions in its memory beginning with the instruction specified by PC1.
EES (0645 octal)	Enter External Mode and Store PC1. The EES instruction causes the PDP-14 to enter external mode and store the current value of PC1 (the address of the next instruction to be executed in the PDP-14 program) in PC2. Upon leaving external mode and returning to normal PDP-14 operation, the value of PC1 is restored by the following instruction.
LER (0454 octal)	Leave External Mode and Restore PC1. The LER instruction causes the PDP-14 to continue normal program execution at the location specified by the value of PC2. This value, which could be stored

Instruction	Purpose
LER (Cont)	in PC2 by an EES instruction of by a register transfer instruction, is transferred to PC1 before the PDP-14 returns to its normal execution.

NOTE

If the EES/LER instruction pair are used, the monitor-supplied instructions executed during external mode must not change the content of PC2 (e.g., there must be no PDP-14 JMS instructions within the monitor - supplied program).

Memory Transfer Instructions

The following instructions transfer a 12-bit word from memory to some PDP-14 internal register.

Instruction	Purpose
TRM (4226 NNNN octal)	Transfer Memory to Output. TRM is a two-location instruction which transfers the contents of the second word to the Output Register of the PDP-14 where it may be read by a monitoring PDP-8 family computer. The value of NNNN which is output to the monitor may be any octal number specified by the user at the time the PDP-14 program is assembled by PAL-14, or compiled by BOOL-14. This instruction may be simulated by SIM-14.
TRS (4225 NNNN octal)	Transfer to Storage (PC2). TRS is a two-location PDP-14 instruction which transfers the contents of the second location of the instruction to the PC2 register. It may be used to extract test patterns from memory during diagnostic work, or to alter the return address after executions of subroutines or external mode operations.

Conditional Skip Instructions

The following instructions cause a conditional skip of one PDP-14 memory location.

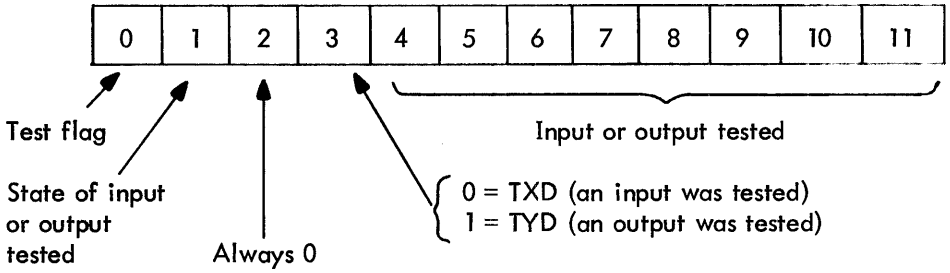
Instruction	Purpose
SKZ R (63R4 octal)	Skip on Register Zero. The next PDP-14 location will be skipped if the register specified by R has contents zero. The value of R is the code value for the previously introduced PDP-14 registers (0 through 6 octal).
SKE R (67R4 octal)	Skip on Register Equal. The contents of the register specified by octal code R is compared with the contents of PC2. If the two contents are equal, the location following the SKE is skipped. Otherwise, the following location is executed as an instruction.

The above instructions may not be part of a SIM-14 program for both on-line and local mode. They will not be recognized, simulated or executed and must not be part of a program which is executed in on-line mode of SIM-14.

Test and Display Instructions

The test and display instructions may be used to test inputs or outputs and display information in the Output Register of the PDP-14 according to the format below.

Bit Position	Information
0	Present state of the TEST Flag (0 = OFF, 1 = ON)
1	Result of the test - 0 if the output or input is OFF, 1 if the output or input is ON.
2	Presently unspecified (always a 0).
3	Type of instruction - 0 if a TXD instruction was executed, 1 if a TYD instruction was executed.
4-11	Octal number of the input or output tested by the instruction.



Instruction	Purpose
TXD (7000 + NNN octal)	Test Input and Display. The input specified by the value of NNN is tested and the information previously described is transferred to the PDP-14 Output Register, where it may be read by a monitoring PDP-8 family computer. The TEST flag is not affected by this instruction.
TYD (7400 + NNN octal)	Test Output and Display. The output specified by the value of NNN is tested and the information described above is transferred to the PDP-14 Output Register, where it may be read by a monitoring PDP-8 family computer. The TEST flag is not affected by this instruction.

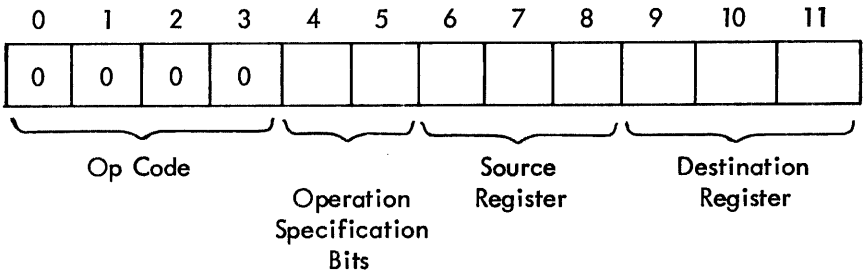
These instructions are recognized and simulated in local mode of SIM-14. They are also executed in on-line mode.

NOTE

If TRM instructions are used in the same program with TYD's or TXD's, bit 2 of the TRM transferred word should be a 1 to indicate to the monitor that it was not generated by a TXD or a TYD. Any four-digit number with an odd first digit (1, 3, 5, or 7) cannot be generated by a TXD or a TYD.

Register-To-Register Transfer Instructions

Register transfer instructions may be used to transfer information from one PDP-14 register to another. Transfers which use the PC1 or the Spare Registers as destination may also increment or decrement (by one) during the transfer. The previously introduced registers and their codes are used to construct the transfer instructions according to the following format. Not all possible transfer instructions are meaningful.



The operation code for a transfer instruction is 0000 in bits 0, 1, 2, and 3.

Bits 4 and 5 contain the operation specification. The following list gives the possible values for bits 4 and 5.

Bits	Operation
00 or 10	Transfer the information contained in the source register to the destination register.
01	Transfer the information in the source register, decremented by 1, to the destination register. The destination register must be an incrementing type register, such as PC1 or Spare (M747).
11	Transfer the information in the source register, incremented by 1, into the destination register. The destination register must be an incrementing type register, such as PC1 or Spare (M747).

The source and destination registers are specified by their respective codes given earlier in this chapter and summarized below:

Register	Source Code	Destination Code
Dummy	$000 = 0_8$	$000 = 0_8$
IR	$001 = 1_8$	---
MB	$010 = 2_8$	$010 = 2_8$
Spare	$011 = 3_8$	$011 = 3_8$
PC1	$100 = 4_8$	$100 = 4_8$
PC2	$101 = 5_8$	$101 = 5_8$
Input	$110 = 6_8$	---
Output	---	$110 = 6_8$

NOTES

a. A transfer register instruction which has a destination code (Dummy) produces a PDP-14 instruction NOP (provided bit 3 = 0).

b. The PC1 register may be cleared by incrementing Dummy while transferring it into the PC1 register. Therefore, the instruction to clear PC1 is 0304 octal. The same is true for the Spare register. Clearing Spare is accomplished by the instruction 0303 octal.

The destination register code 111 (7_8) is used to produce the HLT instruction. Any register transfer instruction which has this destination code will halt the PDP-14 program execution. The simplest form of HLT is described below.

HLT (0007 octal)	Halt the processor. The HLT instruction may be used to stop the processor until the start or continue switch of the PDP-14 is activated. This instruction should only be used for diagnostic purposes and should not be part of a user's control program.

PDP-8 INTERFACE INSTRUCTIONS

The following instructions may be used when the PDP-14 is interfaced with a PDP-8 for monitoring purposes. These instructions are executed within the PDP-8 family computer, but affect the operation of the PDP-14.

Skip Instructions

The following instructions allow the PDP-8 to determine conditions within the PDP-14 using conditional skips of one PDP-8 location.

Instruction	Purpose
SEF (6161 octal)	Skip on EXTERNAL Flag - The EXTERNAL flag is set by the PDP-14 whenever a GNI or LDE initiated operation is completed or whenever the PDP-14 executes an internally stored EEM or EES instruction. The EXTERNAL flag is on the PDP-8 program interrupt (PI) request bus. The setting of this flag indicates that the controlling computer can again interrupt the PDP-14, or enter a new external mode instruction into the memory part of the PDP-14. Execution of this instruction will cause the next PDP-8 instruction to be skipped if the EXTERNAL flag is set.
SOF (6171 octal)	Skip on OUTPUT REGISTER Flag - The OUTPUT REGISTER flag is set whenever the PDP-14 transfers information to the Output Register. The OUTPUT flag, like the EXTERNAL flag, is on the PDP-8 program interrupt (PI) request bus. This instruction enables testing the Output

Instruction	Purpose
SOF (Cont)	Register by the PDP-8 family computer and reading the register if the flag is set. The instruction following the SOF will be skipped only if the OUTPUT flag is set.
STF (6173 octal)	Skip on TEST Flag Set - This instruction will cause a program skip of the following PDP-8 instruction if the PDP-14 TEST flag is set. Thus the monitor may determine if the TEST flag is set when the PDP-14 program is interrupted and, if necessary, may return the TEST flag to the proper state when leaving external mode.
SCR (6175 octal)	Skip on Controller Run - This instruction will cause a program skip of the following PDP-8 instruction if the PDP-14 is running. When the PDP-14 is not running, it is not possible to enter external mode, or to start the PDP-14.

Input and Output Register Instructions

The following instructions allow the PDP-8 and PDP-14 to communicate through the Input and Output registers of the PDP-14.

Instruction	Purpose
LIR (6162 octal)	Load Input Register - This instruction transfers the contents of the accumulator of the controlling computer to the Input Register of the PDP-14. Data transferred in this manner is expected to be used to change the contents of internal PDP-14 registers. Register transfer instructions to accomplish this may be executed from either internal memory or the external memory port.
ROR (6176 octal)	Read Output Register - This instruction clears the PDP-14 OUTPUT flag, causes the accumulator of the monitoring PDP-8 family computer to be cleared and the contents of the PDP-14 Output Register transferred into it. The PDP-8 program may check for the presence of information in the Output Register by checking the OUTPUT flag or the OUTPUT flag could request a program interrupt. When the flag is ON, the Output Register may be read. Since ROR clears the OUTPUT flag, each new transfer of information to the output register will set the OUTPUT flag again. The monitor must read the Output Register before the PDP-14 internally transfers a new word to it or the first word will be lost.

Load PDP-14 Instructions

The following instructions are used to load the PDP-14 with an externally supplied instruction. Program examples assume a basic understanding of the PDP-8 programming.

Instruction	Purpose
GNI (6165 octal)	Generate an Interrupt - This cycle-stealing instruction interrupts the PDP-14, loads the MB of the PDP-14 with the content of the PDP-8

Instruction	Purpose
GNI (Cont)	accumulator, and clears the EXTERNAL flag. The PDP-14 takes its instruction from the accumulator of the interfaced computer through the memory port. The EXTERNAL flag is set when the PDP-14 instruction has been executed.

GNI does not increment PC1. It will modify only the MB and IR, unless otherwise specified by the instruction. The instruction to be loaded with the GNI must be held in the accumulator of the PDP-8 until the EXTERNAL flag is set.

The suggested PDP-8 instruction sequence is:

```
CLA                 /CLEAR THE ACCUMULATOR.
TAD INSTRUCTION     /GET THE INSTRUCTION INTO THE AC.
GNI                 /GENERATE THE INTERRUPT.
SEF                 /WAIT FOR THE INSTRUCTION TO
JMP .-1             /BE EXECUTED BEFORE PROCEEDING.
  .
  .
  .
```

It should be noted that the PDP-14 will execute at least one programmed instruction after each interrupt. Thus, repeated use of the GNI by the external computer will cause the PDP-14 to execute externally supplied instructions on a cycle-stealing basis with one external instruction executed followed by the execution of at least one ROM-supplied instruction.

If it is desired to inhibit the PDP-14 from executing instructions while a sequence of external instructions are supplied and executed, an interrupt should be given to enter an EEM or EES instruction. It is then possible to enter successive instructions using the LDE instruction. The LDE instruction given in the following example is usually used while the PDP-14 is in external mode. Control is returned to the PDP-14 program with a LEM or LER instruction supplied to the PDP-14 with a LDE instruction.

NOTE

The GNI instruction may be used while the PDP-14 is in external mode. The result is described immediately after the LDE instruction which follows.

LDE
(6164 octal)

Load MB and Execute Instruction While in External Mode - Clear EXTERNAL Flag - When the PDP-14 is in external mode, this instruction is used to load the instruction stored in the accumulator of the PDP-8 family computer into the PDP-14 and execute it. A new instruction can be loaded and executed every time the EXTERNAL flag is set. However, the instruction need not be held, as in GNI. Thus, the suggested PDP-8 instruction sequence is:

```
SEF                 /FLAG IS SET AFTER A GNI
JMP .-1             /INITIATED ACTION IS COMPLETED.
TAD INSTRUCTION     /GET THE INSTRUCTION INTO THE AC.
LDE                 /LOAD AND EXECUTE IT IN THE PDP-14.
```

Instruction		Purpose
LDE (Cont)	CLA	/CLEAR THE AC AND PROCEED WITH
	.	/THE PDP-8 PROCESSING. ANOTHER
	.	/PDP-14 INSTRUCTION MAY BE LOADED
	.	/AND EXECUTED AS SOON AS THE FLAG
		/IS SET.

The instructions could also be supplied on a PDP-8 program interrupt basis. Execution of LDE will not change the contents of any internal PDP-14 register except the MB, PC1, and IR, unless so specified in the instruction. (When two-part instructions such as JMP, JMS, or TRM are loaded and executed with the LDE instruction, the EXTERNAL flag is set at the completion of each part of the instruction.)

The GNI instruction, in general, is used to interrupt the PDP-14 and it must wait for the PDP-14 to finish executing its current instruction before honoring the interrupt. The LDE instruction is used after an EEM (or EES) has been executed with a GNI and the PDP-14 is, therefore, in external mode.

When in external mode, either the LDE or the GNI instruction could be used to cause the PDP-14 to execute one-location instructions (e.g., TXD, TYD and register transfers). The difference is that the LDE instruction will increment PC1 and keep track of instructions; the GNI will not increment PC1 and the monitor must keep track of the instruction sequence.

If, while in external mode, the GNI is used to execute a two-location instruction (e.g., TRM, JMP, JMS), the PDP-14 will take the second half of the instruction from its own ROM memory which will be a zero word if the ROM is not present. If a two-location instruction is loaded with an LDE, the second half will be loaded with the next LDE.

Remember that the TRM instruction causes the second half of the instruction to be transferred to the Output Register. The fact that the second half of a TRM instruction is obtained from ROM memory whenever the PDP-14 is in external mode and the GNI instruction is used to load the TRM is very useful. It allows the monitor to examine a part of or the whole ROM memory. PC1 will be incremented once each time a word is read from the PDP-14 memory, enabling a search or dump of contiguous memory. For example:

```
CLA
TAD EEM
GNI
SEF
JMP .-1
```

Put the PDP-14 into external mode.

```
CLA
TAD     NUMBER
DCA     COUNT
TAD     START
LIR
TAD     K264
GNI
```

Set up a counter, and supply the starting address transferring it to the PC1. (K264 contains 0264, transfer input to PC1.)

```
DUMP,   SEF
        JMP .-1
        CLA
        TAD     TRM
        GNI
        CLA
        SOF
        JMP .-1
        ROR
          .
          .
          .
        ISZ     COUNT
        JMP     DUMP
          .
          .
          .
```

Load the PDP-14 with a TRM and wait for the output register to be loaded. (The TRM need not be held in the AC since the PDP-14 is in external mode.)

Process the information: compare it; list it; store it; etc.

Check to see if you are done.

Using PDP-8 Program Interrupt to Monitor

A program interrupt request will be generated by the PDP-8 monitor computer if either the OUTPUT flag or the EXTERNAL flag is set. The OUTPUT flag indicates that data is present in the Output Register. The EXTERNAL flag signals that the PDP-14 has completed execution of an external mode or interrupt instruction and is ready for further instructions. Thus, the PDP-14 can be monitored on a program-shared basis, with one PDP-8 serving many PDP-14's.

The monitoring PDP-8 computer, when processing an interrupt request, must be able to clear the DEVICE flag which caused the interrupt. The instructions which follow perform this function.

CEF (6167 octal)	Clear EXTERNAL Flag - By using CEF, the flag set upon completion of PDP-14 instructions initiated by GNI or LDE can be cleared independently of GNI and LDE instructions.
CLF (6172 octal)	Clear OUTPUT Flag - The OUTPUT flag which indicates that the PDP-14 Output Register holds data for the PDP-8 can be cleared independently of the ROR instruction using CLF. This instruction also clears the PDP-8 accumulator.

It should be noted that the execution of some instructions will cause both the OUTPUT flag and the EXTERNAL flag to be set when monitoring a PDP-14 using program interrupt. The monitor should clear both flags before turning the interrupt facility back on after an interrupt caused by such an instruction. Otherwise an unwanted double interrupt will occur.

MONITOR DESCRIPTIONS

The monitor for each PDP-14 system will be somewhat unique and will require individual programming. Some general ideas and approaches for different applications are offered here.

Transition Checking With BOOL-14

The most common need in monitoring is to know when an event occurs. That is, for example, when is a solenoid energized; when is a motor started; when is a limit switch tripped? BOOL-14 provides for this type of PDP-14 monitoring.

The commands and operation of BOOL-14 were described in Chapter 6. The specific instructions added are described below. In general, BOOL-14 terminates non-monitored equations by the following instruction sequence:

```
To set ON
    To set OFF
      SYF      OUTPUT
      SKP
      SYN      OUTPUT
```

The testing instructions proceed until the state of the output is determined. A JFF or JFN instruction jumps to set the output ON, or fails to jump causing the output to be set OFF. Thus, the terminal instructions are entered at one of two points: either to turn the output OFF or to turn the output ON.

When an output is monitored, this terminating sequence of instructions is altered. TYN or TYF instructions are added to check for the occurrence of transitions. TYD or TRM instructions are added to report the transition by loading the output register and setting the OUTPUT flag.

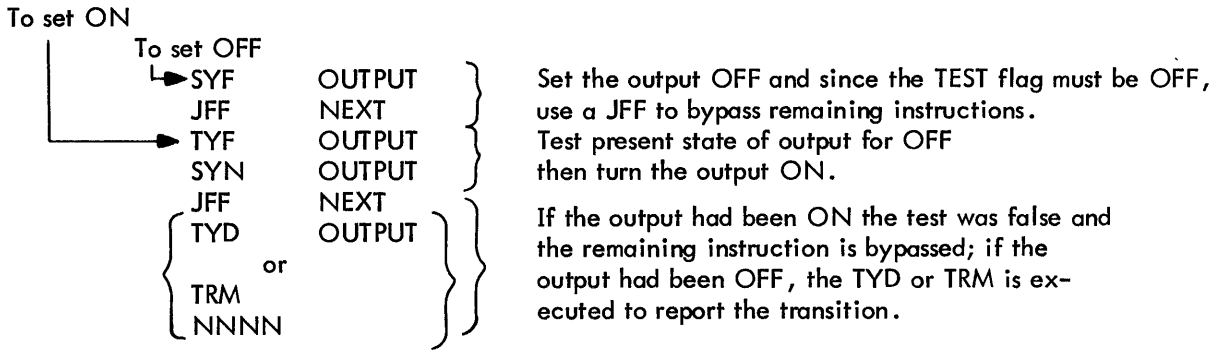
It should be noted that R-functions may be monitored as well as normal outputs or storage outputs. The only difference is that the last instruction of the terminal sequence is a JMR.

The terminating instructions for monitoring are entered in the same way as the terminal sequence of a non-monitored equation. Namely, a JFF or JFN instruction has always been executed, clearing the TEST flag. Whenever the TEST flag is known to be OFF, a JFF instruction may be used as an unconditional jump. This unconditional JFF instruction is often used in the terminating sequence for monitoring.

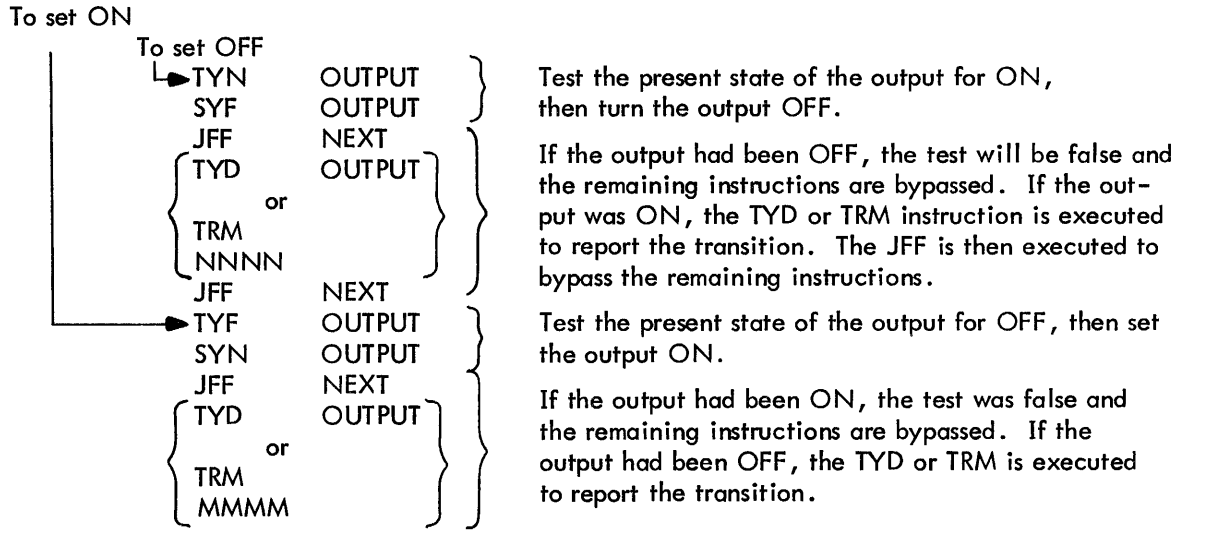
Monitoring for OFF (.MF) - The terminating sequence for .MF or .MF NNNN (where NNNN is an octal number of 4 digits or less) is:

```
To set ON
    To set OFF
      TYN      OUTPUT  }
      SYF      OUTPUT  }   Test the present state of the output for ON, then
      JFF      NEXT        turn the output OFF.
      TYD      OUTPUT  }
        or                 If the output had been OFF, the test will be false
      TRM                  and the remaining instructions are bypassed. If
      NNNN                 the output is ON, the TYD or TRM is executed to
      SKP                  report the transition. The SKP is then executed
      SYN      OUTPUT      to bypass the SYN instruction.
```

Monitoring for ON (.MN) - The terminating sequence for .MN or .MN NNNN (where NNNN is an octal number of 4 digits or less) is:



Monitoring All Transitions (.MFN) - The terminating sequence for .MFN or .MFN NNNN, MMMM (where MMMM and NNNN are octal numbers of 4 digits or less) is:



Supervisory System

There are PDP-14 applications in which a power and light company wishes to scan a set of contact inputs, all of which should be OFF. If any contact is ON, the monitor should be notified.

The following instruction sequence may be used for this monitoring application:

```
        TXN     INPUT1
        JFF     NEXT2
        TXD     INPUT1
NEXT2,  TXN     INPUT2
        JFF     NEXT3
NEXT3,  TXD     INPUT2
          .
          .
          .
```

If an input contact closes, the monitor will receive the 12-bit word sent by the TXD instruction through the output register. Notice that no control is performed in the above example, the PDP-14 simply serves a watchman function.

Decision Making

When two or more PDP-14's are monitored by the PDP-8 to form a network for material handling, each PDP-14 must interact with others. This means that the PDP-14 does not have enough information to make all necessary decisions; it does not know what other PDP-14's are doing.

A simple approach is to have the PDP-14 request help from the monitor. Each PDP-14 is assigned two device codes which differ from those assigned to other PDP-14's. This allows the monitor to "address" each PDP-14. For example, if a crate has been moved on an incremental conveyor to a point where it may be moved left or right or continue straight ahead, the PDP-8 must make the decision. The PDP-14 requests this decision by sending a distinctive word to the output register with a TXD, TYD or TRM. This word is sent whenever a crate reaches the position for decision as marked by a limit switch.

Upon receiving the direction word, the PDP-8 begins the decision-making process. Meanwhile, the PDP-14 continues its other processing duties leaving the crate unmoved. The PDP-8 determines positional information concerning other PDP-14's with the TXD and TYD instructions executed through the cycle-stealing GNI.

When the monitoring PDP-8 reaches its decision, it sets one of three storage outputs in the S-box of the PDP-14. The storage outputs are variables in the equations to move right, left, or proceed. When the PDP-14 solves these equations after the PDP-8 supplied input, the crate will be moved in the proper direction. The PDP-8 will then clear the storage output and wait for the next decision request.

Input Interrogating

In machine tool applications, there are periods of time during which a set of inputs remains unchanged. While a slide moves forward, many inputs on the machine should be in a definite known state.

The monitor stores a list of the proper states for several points in the control sequence. When a suitable operation for testing begins, the PDP-14 reports to the PDP-8 monitor. The monitor determines from the word received, which operation is being performed and thus, which set of input conditions is applicable.

The PDP-8 then proceeds to check inputs with the TXD instruction executed on an interrupt basis using GNI's. If any actual input (e.g., limit switch) state differs from the predicted state, the input failure is detected by the monitor and an appropriate message is typed on the Teletype console.

CHAPTER 10
PDP-14 SYSTEM SOFTWARE OPERATION

This chapter describes the use of the five software elements of the PDP-14 system (Loader, Editor, BOOL-14, PAL-14 and SIM-14). These system programs operate on a PDP-8 family computer, usually a PDP-8/I or 8/L. The operation of the PDP-8 hardware system (computer console and ASR33 Teletype) is described in the PDP-8/I System User's Guide (Order No. DEC-08-NGCB-D).

PDP-8 SWITCH AND SWITCH REGISTER OPERATIONS

The PDP-8 is operated by a series of switches on the computer console. For PDP-14 programming purposes, the user should be familiar with the START, STOP, CONT (Continue), LOAD ADD (load address), DEP (deposit) and EXAM (examine) switches. The descriptions in this chapter will direct the user to operate each of these switches at the proper time. These switches are all spring-loaded and will return to their normal positions after release.

The PDP-14 software will not run if the SING STEP or SING INST switches are set on the PDP-8 console. (The SING INST is not present on the PDP-8/L console.) These switches should always remain off. (In the PDP-8/I, the top of these two switches should be in the OUT position, with the bottom IN. In the PDP-8/L the SING STEP switch should be in the UP position.)

The twelve switches of the PDP-8 switch register are in four groups of three switches each. Each group of switches is the same color to differentiate them from the adjacent groups.

Every possible switch pattern for a 3-switch group is pictured in Figure 10-1. There are eight such patterns, numbered 0 through 7. The chart is essentially the octal-binary equivalents for the digits 0 through 7, and their switch representations. For example, to set a 3-switch group to the number 4, the first switch is set on, and the second and third switches are set off.

NOTE

A PDP-8/I switch is on when the top is OUT and the bottom is IN. A PDP-8/L switch is on when it is in the UP position.

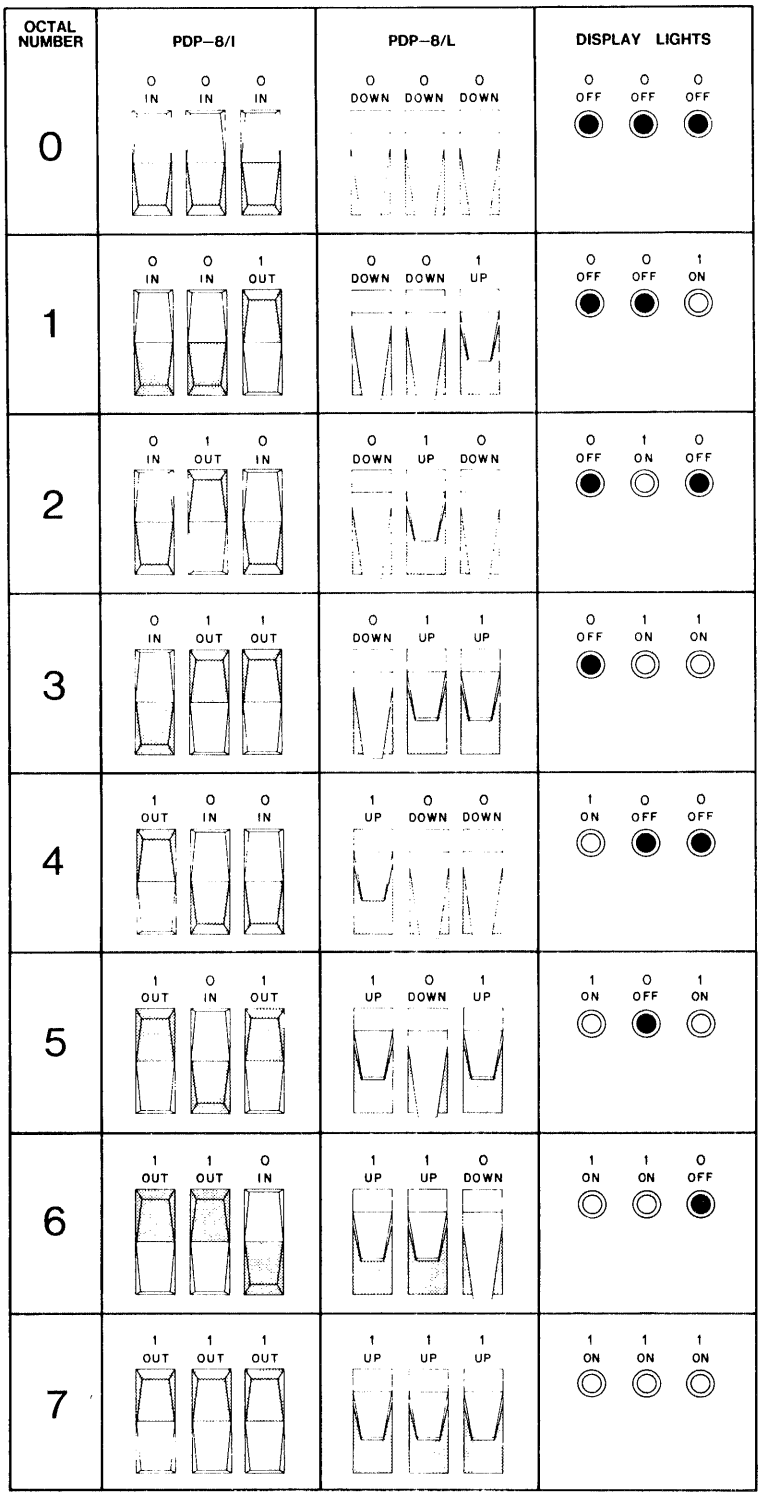

OCTAL NUMBER
PDP–8/I
PDP–8/L
DISPLAY LIGHTS
0
0 IN 0 IN 0 IN
0 DOWN 0 DOWN 0 DOWN
0 OFF 0 OFF 0 OFF
1
0 IN 0 IN 1 OUT
0 DOWN 0 DOWN 1 UP
0 OFF 0 OFF 1 ON
2
0 IN 1 OUT 0 IN
0 DOWN 1 UP 0 DOWN
0 OFF 1 ON 0 OFF
3
0 IN 1 OUT 1 OUT
0 DOWN 1 UP 1 UP
0 OFF 1 ON 1 ON
4
1 OUT 0 IN 0 IN
1 UP 0 DOWN 0 DOWN
1 ON 0 OFF 0 OFF
5
1 OUT 0 IN 1 OUT
1 UP 0 DOWN 1 UP
1 ON 0 OFF 1 ON
6
1 OUT 1 OUT 0 IN
1 UP 1 UP 0 DOWN
1 ON 1 ON 0 OFF
7
1 OUT 1 OUT 1 OUT
1 UP 1 UP 1 UP
1 ON 1 ON 1 ON
14-00014

Figure 10-1

Examine Figure 10-1 and study the equivalents. If you are not familiar with the binary and octal number systems, regard the table as a coding scheme which enables concise directions for switch settings. For example in place of stating:

SWITCH 1 = ON
SWITCH 2 = OFF
SWITCH 3 = ON

we can write "set the switches to 5", thereby specifying the following switch configurations (Figure 10-2):

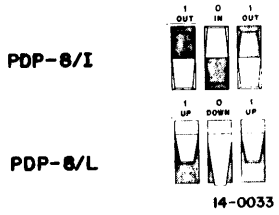


Figure 10-2

As previously stated, the switch register is a group of 12 switches in the form of four 3-switch groups. Using Figure 10-1, we may use four numbers (with a range 0 through 7) to completely define all switch register settings.

For example (Figure 10-3):

Set switch register to 1634:

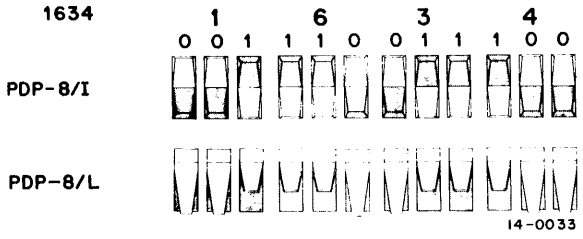
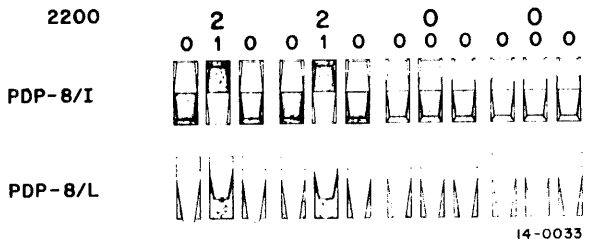


Figure 10-3 (Part 1)

Set switch register to 2200: 2200



Set switch register to 200: 200

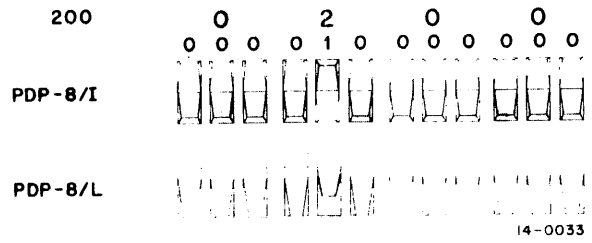


Figure 10-3 (Part 2)

PDP-8 LOADER PROGRAM

Before any PDP-8 based program can be loaded into the PDP-8 memory, a loader program must be stored which will read the binary format paper tape on which these programs are stored. The loader program is then used to bring BOOL-14, SIM-14 and all other system software into the PDP-8 memory.

NOTE

The loader program to be used with PDP-14 software is the HELP Loader. Readers who are familiar with other PDP-8 software should be aware that the HELP Loader contains the more familiar RIM and BIN loaders. Once HELP is loaded, RIM and BIN are both in memory. Binary paper tapes are actually loaded by the BIN loader, not by HELP. This distinction is not necessary for the new PDP-8 user to understand.

Loading the HELP Loader

The procedure used to store the HELP loader itself in memory is described below.

The numbers represent PDP-8 instructions which are used to load a short paper tape. This paper tape contains the loader program for binary format tapes, such as BOOL-14 or SIM-14.

The data field and instruction field switches should be set to 0 while the Loader is being stored in the PDP-8 memory. If a PDP-8/L is being used, the MEM PROT switch must be in the DOWN or off position while the loader program is being stored in memory.

The user sets the following switch register configurations, and then presses the switch noted at the right of the configuration.

Switch Register Setting	Operation Switch
0027	LOAD ADD (Load Address)
6031	DEP (Deposit)
5027	DEP (Deposit)
6036	DEP .
7450	DEP .
5027	DEP .
7012	DEP .
7010	DEP .
3007	DEP .
2036	DEP .
5027	DEP (Deposit)

The first number above is the first address where the instructions are to be stored, and the following numbers are the instructions to be deposited in the PDP-8 memory.

Once the instructions are all loaded, the user should check to see that the program was loaded correctly by examining the stored instructions. The lights of the Memory Buffer console display will light to indicate the stored instruction when the EXAM switch is pressed. The lights will have patterns corresponding to the numbers given below, if the instructions have been correctly loaded.

Switch Register Setting	Operation Switch
0027	LOAD ADD

Operation Switch	Memory Buffer Display
EXAM	6031
EXAM	5027
EXAM	6036
EXAM	7450
EXAM	5027
EXAM	7012
EXAM	7010
EXAM	3007
EXAM	2036
EXAM	5027

If the instructions have been loaded incorrectly, the user should restart the switch setting process.

When the switches are correctly loaded, the user should place the HELP Bootstrap Loader paper tape (DEC-08-LHA1-PB) in the paper tape reader of the Teletype unit, and set the reader control switch to START. The user then sets the switch register to 0027, and presses the LOAD ADD and START switches of the PDP-8 computer console. The paper tape containing the loader will then be read. Once the tape has been read, thus placing the loader program in memory, any system program (e.g., SIM-14, BOOL-14, etc.) may be loaded.

If a PDP-8/L is used for programming, the MEM PROT switch of the PDP-8/L console should be put in the UP position, once the loader is in memory. This prevents unwanted destruction of the loader program. This switch is not present on other family-of-8 computers.

Loading a Binary Tape into the PDP-8

To load a system software paper tape into the PDP-8 memory, place the tape in the paper tape reader of the Teletype unit, and set the reader control switch to START. The tape must always be positioned with the leader tape over the reading head. Then set the PDP-8 switch register to 7777, press the LOAD ADD and START switches of the PDP-8 computer console. If the preceding directions have been followed correctly, the tape will now be read into memory. When the tape reaches the end, it will stop. If any light of the accumulator display on the PDP-8 console (not LINK display) remains on, a reading error occured and the tape must be re-read. If a light remains on, a checksum error has occurred.

If the PDP-8 system is equipped with a high-speed paper tape reader, a similar procedure is followed. The paper tape is positioned in the reader with the leader tape over the reading sensors. The PDP-8 switch register is set to 7777, and the LOAD ADD switch is depressed. The first left-hand switch is then set OFF (SR = 3777), and the START switch is depressed. The binary tape will then be read.

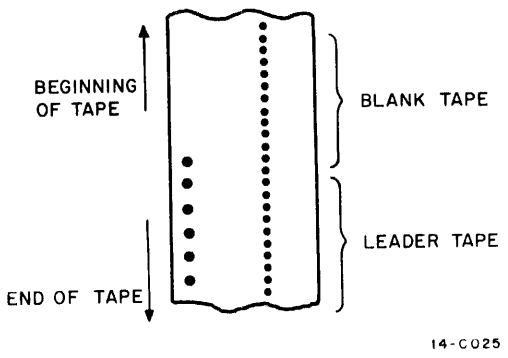


Figure 10-4

PDP-8 EDITOR

PDP-14 users of BOOL-14 and PAL-14 must prepare a paper tape record of their program. This paper tape is referred to as a program source tape or source language tape. BOOL-14 and PAL-14 translate the source tape into the machine language or binary tape used by the PDP-14 itself. Machine language is the form in which PDP-14 programs are debugged with SIM-14.

Composing Source Programs

The PDP-8 Editor program may be used to compose programs in the BOOL-14 and PAL-14 source language. The Editor allows the PDP-14 user to type the program on the Teletype keyboard and correct typing errors as they occur. Sections of the program may be moved or removed by typing single Editor commands. Program source tapes are generated by Editor which are, in turn, used as input to BOOL-14 or PAL-14.

Updating Source Programs

A second, and very important, function of the Editor is updating user programs. During the life of a PDP-14 program, many changes will be made to the original version. Bugs uncovered with SIM-14, changes in machine function, and additions of new inputs or outputs to the machine all require changes to the PDP-14 program. These changes may be made with SIM-14; however, in this case there is no permanent record of the altered program. The user should always correct the source tape of his program by reading it back into the Editor, making the necessary changes and corrections to the source program, and generating a new program listing and tape.

Text Buffer

A Buffer is a computer storage area. The text buffer stores the text of PDP-14 programs typed by the user. The text buffer is organized by lines of text. A text line is a collection of typed characters up to and including the carriage return at the end of the line. The lines of the text buffer are numbered decimally and each line is referred to by its line number. By referring to these specific line numbers, the Editor commands delete lines of text, insert lines of text before a specified line, change lines of text, list lines of text and more.

Modes of Operation

Since Editor accepts both commands and text from the Teletype keyboard, it must know whether the characters currently being typed are text of the program being created or modified by the user, or are commands directing the Editor to perform some function upon the text. The Editor makes this distinction by dividing its operation between two modes. In the command mode, all characters typed on the Teletype keyboard will be interpreted

as commands to the Editor to perform some operation on the content of the text buffer, or to allow the user to operate upon the text. In the text mode, all characters typed on the keyboard are entered into the text buffer. Typed text may replace, be inserted into, or be added at the end of the current content of the text buffer.

The current mode of the Editor must always be kept in mind. If a command is typed while the Editor is in text mode, it will not be recognized as a command by the Editor and will be entered into the text buffer. Likewise, if text is typed while the Editor is in command mode, it will be treated as a command.

Transition Between Modes

The Editor is in command mode when it is first loaded and is ready to accept the first command from the user. Only legal commands are accepted; any other typed characters cause the Editor to type a question mark (?) and wait for a legal command to be typed. Some commands cause the Editor to enter text mode, where all characters typed are entered in the text buffer. When the user has finished typing his program and wishes to leave the text mode, he simply holds the CTRL key of the Teletype keyboard depressed while he strikes the FORM key (CTRL/FORM). The Editor will respond by ringing the Teletype bell and waiting for the user to type a new command (Figure 10-5).

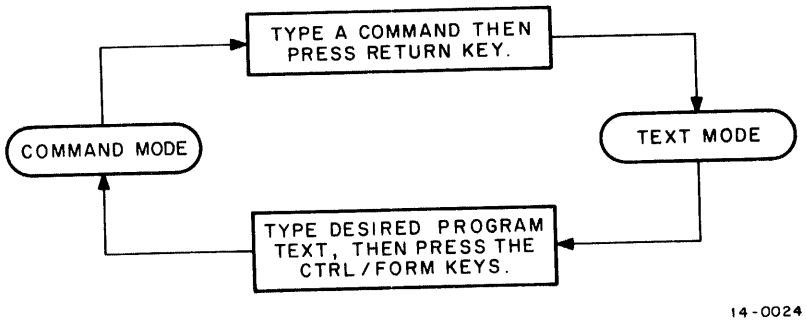


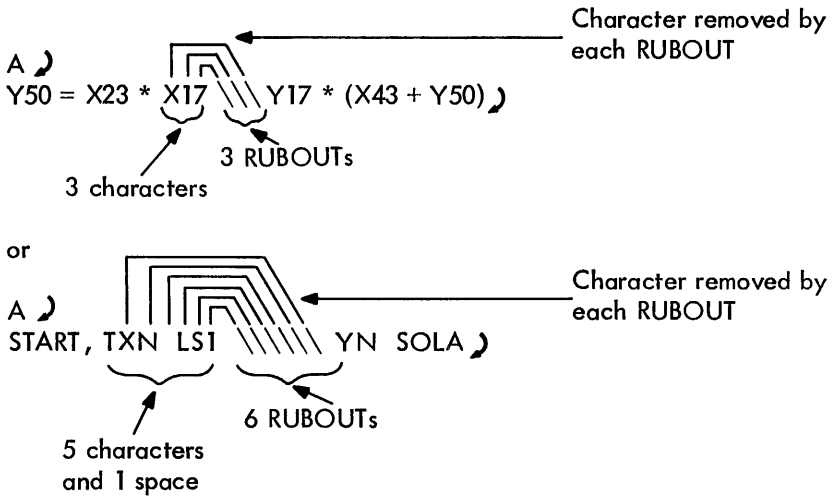
Figure 10-5

Adding Text to the Buffer

To compose PDP-14 programs with the Editor, simply type "A↵ ". The A command instructs the Editor to enter text mode and add all text typed on the keyboard to the text buffer. The new text is added at the end of any text currently in the buffer.

Editing Keys

Typing errors, which invariably occur, may be erased with the RUBOUT key. This key will delete one character or space to the left each time it is struck. The RUBOUT key can only delete characters in the same line of text. The Editor types a backslash (\) for each time the RUBOUT is struck.



In the foregoing examples, the user entered text mode with the A command and began typing a BOOL-14 program above or a PAL-14 program below. He typed the wrong information, however, and changed it by repeatedly striking the RUBOUT key and "erasing" all characters back to the "*" in the BOOL-14 program and back to the "T" in the PAL-14 program. He then typed the correct characters, terminating the line by typing RETURN. The corrected versions of the foregoing examples are now:

```
Y50 = X23 * Y17 * (X43 + Y50) ↵
or
START, TYN SOLA ↵
```

The backarrow (←) is also used to erase typed characters from the text buffer. It is typed by holding the SHIFT key while striking the letter O key. This special character deletes the complete line of characters between the left margin and itself.

```
Y5 = X2 + / (X16 * Y3) ←; THE FOLLOWING EQUATION CONTROLS SOLENOID A. ↵
```

In the above example, a complete line of text was deleted using the backarrow, and a comment was inserted in its place. The line of the text buffer is now:

```
; THE FOLLOWING EQUATION CONTROLS SOLENOID A.
```

Reading a Paper Tape

Source tapes previously generated by the Editor may be read into the text buffer for modification or addition by using the command:

```
R ↵
```

The paper tape will be read from the low-speed reader into the text buffer. If the buffer is not empty, the information will be added at the end of the previously stored text.

After the R is typed and the tape is read, the Editor will ring the Teletype bell and enter command mode.

The R command will read a paper tape from the high-speed reader when the last switch of the switch register (SR 11) is set ON. Otherwise the tape will be read from the low-speed reader of the Teletype.

Listing a Program

Once the program has been typed or read from a tape, the complete text can be typed by issuing the L command.

L ↵

The complete content of the text buffer will be typed on the Teletype printer.

The L command may be modified to type only certain lines by specifying limits for the list. The limits are typed preceding the "L" and are separated by a comma. For example:

1,15L ↵	List the content of text buffer lines 1 through 15 inclusive.
12L ↵	List line 12 of the text buffer.

Remember that the lines of the text buffer are numbered decimally starting with line 1. After the Editor has typed the text, it will enter the command mode.

Inserting a New Line of Text

PDP-14 programs may be modified with the Editor by inserting new lines of text using the I command. This command is typed, preceded by the decimal number of the line before which the new text will be placed. For example,

15I ↵

will allow one or more new lines of text to be typed before line 15 of the current text buffer.

Editor will remain in text mode and accept all characters and lines typed by the user into the text buffer. When the user has completed typing the lines to be inserted, he should type CTRL/FORM to return to command mode.

The Editor "pushes down" text in the buffer when the I command is used. The new text will be automatically numbered, thereby changing the previously assigned line numbers. Thus, if one line is inserted before line 15 (viz., 15I ↵) old line 15 will become line 16 and the newly typed line will become line 15.

Deleting Lines from the Text Buffer

PDP-14 programs may also be modified by deleting lines with the D command. The command is typed, preceded by the line or lines to be deleted. For example,

```
1,15D )
```

removes the first 15 lines of the text buffer. Line 16 immediately becomes line 1 and all other line numbers are adjusted accordingly.

After executing the D command, the Editor returns to command mode.

Remember that when a line has been deleted, the line numbers for the remaining lines are immediately adjusted. For example, the following two commands will actually delete lines 16 and 17 of the original text buffer.

```
16D )
16D )
```

The complete buffer is deleted by using the K (kill) command. It may be used after punching one program and before reading another program tape.

Changing Lines of Text

The C (change) command is a combination of the D and I commands and allows the replacement of a line or group of lines by text typed on the keyboard. For example:

```
15,17C )
```

deletes lines 15 through 17 and then causes Editor to enter text mode. The text typed by the user is then inserted as lines 15, 16, 17, 18, etc. The user may type as many lines as needed. When the new text has been typed, strike the CTRL/FORM to return to command mode.

The C command may replace one line with many lines, or many lines with only one; there need be no relation between the amount of lines deleted and the amount of lines added in their place.

NOTE

Line numbers are automatically adjusted for the new text when the user presses the CTRL/FORM to return to text mode.

Moving Lines of Program Text

Occasionally PDP-14 programs need to be completely reorganized (e.g., to better fit the "page structure" of the PDP-14). This reorganization can easily be accomplished with the M (move) command. The M command has a special command format which specifies the line or group of lines to be moved and the line number before which these moved lines will be placed. For example:

```
15,27$30M
```

will move lines 15 through 27 before line 30. The line numbers are automatically adjusted and the Editor returns to command mode.

In the foregoing example, line 30 will remain line 30; line 28 will become line 15; line 29 will become line 16; lines 15 through 27 will become lines 17 through 29.

Remember, the M command can move one or more lines before the line number which is preceded by a "$" in the command. The Editor automatically returns to command mode after an M command is executed.

Search Feature

A very helpful feature available with the Editor is the search feature, which allows the user to modify a line of text without completely retyping the line. The line number for the line to be searched is typed followed by "S" and a carriage return. The user then must type the "search character". This character may be any keyboard character (e.g., a letter, number or symbol). The search character is not printed on the teleprinter when the key is struck. Instead, the Editor begins typing the searched line up to and including the first occurrence of the designated search character. When the Editor locates and types the search character, typing stops and all, or any combination of, the following operations may be typed by the user:

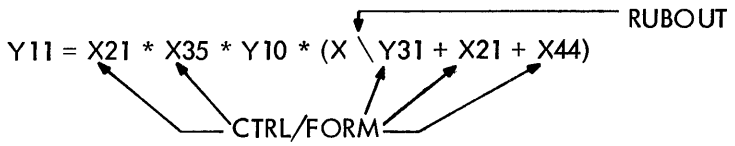
1. New text and terminate line with the RETURN key to change the entire line to the right,
2. ← to delete the entire line to the left,
3. RETURN to delete the entire line to the right,
4. RUBOUT to delete from right to left one character for each RUBOUT typed,
5. LINE FEED to insert a carriage return/line feed, thereby dividing the line into two lines,
6. CTRL/FORM to search for the next occurrence of the search character.

For example, consider the following equation:

```
Y11 = X21 * X34 * Y10 * (X31 + X21 + X44)
```

assume that the "(X31" should be "(Y31".

The letter X could be selected as the search character. However, since there are two occurrences of the letter X prior to the one to be altered, the CTRL/FORM keys must be used to pass by the two preceding letter X's. Once the third X is reached a RUBOUT is struck to delete the X and then the letter Y is struck to replace the X. The CTRL/FORM keys are used to continue the search until the end of the line, thereby completely retyping the line. If the RETURN key were used after the letter was changed, the remainder of the line would have been deleted.



If the user had selected the number 3 as the search character, he would have used one CTRL/FORM to get to the second "3". He then could type two RUBOUTs (thereby removing the 3 and the X), and type Y3 and then CTRL/FORM to repeat the remainder of the line. Thus, the intelligent choice of a search character can ease the editing task when the search feature is used.

Special Characters

The use of the Editor commands in most cases requires the user to type line numbers. These line numbers are computed by counting the number of preceding lines, or the following special characters may be used

.(period)	has a value equal to the line number of the last edited line. Thus after a line is edited, it may be listed by typing ".L" and a carriage return.
/ (slash)	has a value equal to the line number of the last line in the buffer. Thus "/D" means delete the last line.
+, - (plus, minus)	may be used with the "." or "/" characters to represent line numbers. For example, ".+5L" means list the fifth line after the last edited line.
= (equals)	may be used with the "." or "/" characters to find the value of these characters. When the user types ".=", the Editor will type the decimal line number of the current line. When the user types "/=", the Editor will type the decimal value of the last line in the buffer.

Punching a Source Tape

Once the PDP-14 program has been prepared with the Editor, a paper tape is punched with the P command. This command may punch the complete program (P) or a group of lines within the program (15,30P↵).

The procedure for generating a source tape using the low-speed punch (LSP) of the Teletype is:

1. Type T↵ . Quickly press the LSP ON. Leader/trailer tape will be punched.
2. Press the LSP OFF.
3. Type the punch command: P↵ , nP↵ , or m,nP↵ , (where m and n are decimal line numbers).
4. Turn LSP ON.
5. Press CONT switch located on the PDP-8 console. The program text will now be punched.
6. Press the LSP OFF.
7. Press F and RETURN keys. Turn LSP ON. A special character is placed to signal the end of the tape for the Editor.
8. Turn the LSP OFF.
9. Type T and RETURN keys. Turn LSP ON. Leader/trailer tape will be generated.
10. Turn the LSP OFF and remove the source tape.

The procedure for generating a source tape using the high-speed punch (HSP) is:

1. Set the next-to-last switch of the switch register (SR 10) ON.
2. Press the HSP ON.
3. Type T and RETURN keys.
4. Type the punch command: P↵ , np↵ , or m,nP↵ , (where m and n are decimal line numbers).
5. Press the PDP-8 CONT switch. The program text will now be punched.
6. Type F and RETURN keys.
7. Type T and RETURN keys.
8. Remove the generated source tape.

A program may be segmented and punched in sections using the .EOT (BOOL-14) or EOT (PAL-14) control statements to conclude all but the final segment. Thus, the above procedure may be used to punch all, or part of, a PDP-14 program. This allows the user to edit and generate tapes for programs which are not small enough to fit completely within the text buffer.

Editor Summary Tables

Tables 10-1 and 10-2 summarize the characters and symbols used with the Editor. Note that the characters have different meanings when used in command or text mode.

Table 10-1
Special Editor Keys

Key	Command Mode	Text Mode
RETURN	Execute preceding command	Enter line in text buffer
←	Cancel preceding command (Editor responds with a ? followed by a carriage return and line feed)	Cancel line to the left margin
RUBOUT	Same as ←	Delete to the left one character for each depression; a \ (backslash) is echoed (not used in Read (R) command)
CTRL/FORM	Respond with question mark and remain in command mode	Return to command mode and ring teleprinter bell
.	Value equal to decimal value of current line (used alone or with + or − and a number, e.g., .+8)	Legal text character
/	Value equal to number of last line in buffer; used as an argument	Legal text character
LINE FEED	List next line	Used in Search (S) command to insert CR/LF into line
>	List next line	
<	List previous line	
=	Used with . or / to obtain their value	
:	Same as = (gives value of legitimate argument)	
CTRL/TAB		Produces a tab which on output is interpreted as 8 spaces or a tab/rubout, depending on SR option

Table 10-2
Summary of Editor Commands

Type	Command	Function
Input	A	Append incoming text from keyboard into text buffer
	R	Append incoming text from tape reader into text buffer
Editing	L	List entire text buffer
	nL	List line n
	m,nL	List lines m through n inclusively
	nC	Change line n
	m,nC	Change lines m through n inclusively
	I	Insert before first line

Table 10-2 (Cont)
Summary of Editor Commands

Type	Command	Function
Editing (Cont)	nI	Insert before line n
	nD	Delete line n
	m,nD	Delete lines m through n inclusively
	m,n$kM	Move lines m through n to before line k
	S	Search buffer for character specified after RETURN key and allow modification (search character is not echoed on printer)
	nS	Search line n, as above
	m,nS	Search lines m through n inclusively, as above
Output	P	Punch entire text buffer
	nP	Punch line n
	m,nP	Punch lines m through n inclusively
	T	Punch about 6 inches of leader/trailer tape
	F	Punch a FORM FEED onto tape

m and n are decimal numbers, and m is smaller than n; k is a decimal number

The P command halts the Editor to allow the programmer to select I/O control; press CONT to execute these commands.

Commands are executed when the RETURN key is depressed, excluding the P and N commands.

Loading the Editor

The Editor is supplied as a binary paper tape (No. DEC-08-ESAB-PB) and is loaded with the loader program. It will operate in any field of a PDP-8.

Starting the Editor

When the Editor has been loaded and no checksum error has occurred, the PDP-8 switch register is set to 200 and the instruction field is set equal to the data field, if an 8K PDP-8 is used. The LOAD ADD and START switches are then depressed in that order. The Editor will type a carriage return and await a typed command from the user. All further operation is controlled by commands.

The paper tape output and input are controlled by the last two switches of the switch register. If the high-speed reader/punch is to be used, these switches should be set ON. If the low-speed reader/punch of the Teletype is used, these switches should remain OFF.

NOTE

Further information on the Editor, including features not described in this manual, is available in the DEC publications:
Symbolic Editor Manual (DEC-08-ESAB-D), and
System Users Guide (DEC-08-NGCB-D).

BOOL-14 OPERATION

BOOL-14 reads paper tape records of programs in an equation format, or equations typed directly on the Teletype keyboard. It translates them into a paper tape in machine code (binary) form which contains the instructions necessary to solve the equations.

Loading BOOL-14

BOOL-14 is supplied as a binary tape and is loaded into the PDP-8 memory with the loader program. Either the paper tape reader on the ASR-33 Teletype or the high-speed reader may be used to load the tape.

If programs requiring greater than 1K of PDP-14 memory are to be compiled by BOOL-14, an 8K PDP-8 system is required. In this case, BOOL-14 is loaded into field 1 of the PDP-8 (by setting the data field switches to 1).

Starting BOOL-14

When BOOL-14 has been loaded and no checksum error has occurred, the PDP-8 switch register is set to 2200 and the instruction field switches are set to the value of the data field switches. The LOAD ADD, and START switches are then pressed in that order. If the high-speed reader is to be used, the source program paper tape should be loaded in the high-speed reader before the queries are answered.

When started, BOOL-14 will respond by typing sequence of queries, for example:

Query	Meaning
*BIN-NLH?	What device is to be used for the binary output tape? Is no binary requested (N); binary on the low-speed
*LST-NYW?	What listing form is required? No listing is required (N). In this case, only error messages will be typed. Yes, a listing is requested (Y) which will contain any reduced forms determined by BOOL-14. Yes, a listing is requested, but without the reduced equation forms (w).
*SRC-KLH?	What is the source device from which the program will be supplied to BOOL-14? Will it be the keyboard (K), the low-speed reader of the Teletype (L), or the high-speed paper tape reader (H)?
*EDT-NY?	Is editing requested, No (N), or Yes (Y)?

When the *EDT-NY? query has been answered, compilation will begin. If the program is to be supplied from the keyboard, the user will type the first equation or control statement following the quotation mark ("). If the low-speed reader is used to read a source paper tape, the reader must be switched to START before the compilation will begin. If the high-speed reader is used and it was not loaded prior to answering the final query, the user must stop BOOL-14 and restart at 2200, after he has loaded the tape to be translated into the reader.

If the editing option is requested and the low-speed reader is the SRC device, BOOL-14 will halt after an error; the reader must be switched to STOP and the PDP-8 CONT switch must be pressed. After the editing is completed, the compilation will continue when the user switches the low-speed reader to START.

BOOL-14 will complete the compilation after reading the source paper tape once. However, if the program has been segmented with the .EOT statement, BOOL-14 will halt after the incomplete tape and type "EOT". The next tape should then be loaded and the PDP-8 CONT switch depressed.

When the compilation is complete, BOOL-14 will type the message "TURN PUNCH ON" if a binary tape is to be punched. The proper punch should then be turned ON and the PDP-8 CONT switch depressed. The binary output tape will then be punched.

If more than one source tape is to be assembled, the user may simply press the PDP-8 CONT switch after the binary paper tape is punched. BOOL-14 will then begin the option queries by typing "*BIN-NLH?"

PAL-14 OPERATION

PAL-14 reads paper tape program records in a symbolic form and translates them into a paper tape in machine code (binary) form which is acceptable to SIM-14. The paper tape which is input to PAL-14 is a symbolic or source tape and may be generated using a Teletype (in LOCAL mode with the punch turned ON) or by using the PDP-8 Editor program. Once this symbolic tape has been generated, the program is assembled through the following procedure.

Loading PAL-14

PAL-14 is supplied as a binary tape and is loaded into the PDP-8 memory with the loader program. Either the paper tape reader on the Teletype or the high-speed reader may be used to load the tape.

If an 8K PDP-8 is used for the assembly process, the symbol table storage capacity is expanded when PAL-14 is loaded into field 1 (by setting the data field switches to 1 during loading).

Starting PAL-14

To start PAL-14, set the PDP-8 switch register to 200, set the instruction field switches equal to the data field, and press the console switches LOAD ADD and START, in that order. PAL-14 will then type the following queries to determine the devices to be used during assembly.

*BIN	Type the letter signifying the device on which the binary tape, which is the assembly output, will be punched.
*LST	Type the letter signifying the device on which the assembly listing will be typed (or punched).
*SMB	Type the letter signifying the device on which the symbol table will be typed (or punched).
*SRC	Type the letter signifying the device on which the program source tape will be read.

The responses are:

L	The low-speed Teletype unit is to be used as the device.
H	The high-speed paper tape reader/punch is to be used as the device.
(RETURN)	The particular output is not wanted; the RETURN key is typed without being preceded by either an L or H.

If the PDP-8 system is not equipped with a high-speed reader/punch unit and all outputs are wanted, the responses to all questions will be L and the query sequence will be:

```
*BIN-L )
*LST-L )
*SMB-L )
*SRC-L )
```

To generate an error listing only, request no binary, no listing, and no symbol table, specifying only the source device.

If the user makes a mistake when typing a response to a query, he may change the response before he has typed the terminating carriage return simply by typing the correct response. Only the last character is recognized. For example:

```
*BIN-LH )
*LST-HL )
```

is the same as typing:

```
*BIN-H )
*LST-L )
```

The user may erase all previous query responses and restart the sequence of queries by typing the response X. For example:

```
*BIN-L )
*LST-L )
*SMB-X )
*BIN- )        The query sequence has been restarted.
```

Two additional responses are used to control the output of optional error listings. These character responses may be typed along with the required response to any query. They may be erased by typing the X response and restarting the query sequence.

N	No separate error listing is desired on the Teletype. PAL-14 will suppress the output of the Pass 1 error listing to the Teletype. If the high-speed punch is used to output the assembly listing, the output of Pass 2 errors to the Teletype will also be suppressed.
E	If the high-speed punch is used to output the assembly listing, typing E requests that the Pass 1 error listing be punched on the high-speed punch, also.

For example, the following sequence of queries and responses results in binary output on the high-speed punch 1) the assembly listing will be punched on the high-speed punch, 2) the symbol table will be typed on the Teletype, 3) the source tape will be read from the high-speed reader, 4) there will be no Pass 1 or Pass 2 error listings typed on the Teletype, and 5) the Pass 1 error listing will be punched on the high-speed punch.

```
*BIN-HN )
*LST-HE )
*SMB-L )
*SRC-H )
```

Assembly Passes

PAL-14 must read the program source tape two or possibly three times dependent upon the selection of assembly devices. The first pass of PAL-14 defines symbols and checks for errors. The second pass types the listing and symbol table. If separate devices are designated for the listing and the binary output, the binary will also be punched on the second pass. If the same device (e.g., the Teletype) is used for listing and binary output, a third pass will be required to punch the binary tape and thereby complete the assembly process.

PAL-14 will type the message:

```
*PAS
```

and halt when it has completed an assembly pass. If a further pass is needed, the user reloads the paper tape in the reader and presses the PDP-8 CONT switch. If the low-speed reader is used as the SRC device, the

reader must be switched to START before the pass will start. Before beginning the pass which will punch the binary tape output (either pass 2 or 3), the punch designated in the BIN query must be turned ON.

More than one program may be assembled without reloading PAL-14 by pressing CONT in response to the "*PAS" statement typed after the binary output has been punched. PAL-14 will respond by typing the initial queries (*BIN, *LST, *SMB, and *SRC), and the assembly process may be restarted.

If a PAL-14 source program is contained on two or more source tapes, the assembler will stop after reading all but the last source tape and type:

*EOT (end of tape)

The user should load the next sequential tape and press the PDP-8 CONT switch whenever this message is typed.

Error Halt

The only error which will halt the assembly is symbol table overflow. If this occurs, PAL-14 will type:

*SMB OFLW

Recovery is only possible by:

a. If the user is presently assembling using a 4K PDP-8, he will increase the symbol table area by using an 8K PDP-8 and loading PAL-14 in Field 1.

b. If an 8K PDP-8 is not available, or if such a system is presently being used, the user must segment his program and assemble it in parts.

In both of the above cases, the assembly must be restarted.

SIM-14 OPERATION

SIM-14 reads the paper tapes as output from BOOL-14 or PAL-14 and allows the user to thoroughly test the program thus generated. SIM-14 outputs a binary tape to be sent for the weaving of the Read-Only-Memory (ROM).

Loading SIM-14

SIM-14 is supplied as a binary tape and is loaded into the PDP-8 memory with the loader program. Either the paper tape reader on the ASR-33 Teletype of a high-speed reader may be used to load the tape.

SIM-14 is loaded into Field 0 if the PDP-14 will have a 1K memory. If the PDP-14 program will require greater than 1K of PDP-14 memory, a PDP-8 with 8K of memory is needed to simulate the PDP-14 memory. In this case, the SIM-14 program is loaded into Field 1 of the 8K PDP-8, by setting the data field switches on the PDP-8 equal to 1 when SIM-14 is loaded.

Starting SIM-14

When SIM-14 has been loaded and no checksum error has occurred, the PDP-8 switch register is set to 2200, and the instruction field is set equal to the data field if an 8K PDP-8 is used. The user then presses (1) the LOAD ADD switch (load address) and (2) the START switch, in that order. SIM-14 responds by typing its version number (e.g., V3 means version 3 of SIM-14), and a period to signal its readiness to accept commands from the user.

When it is started at location 2200 as above, SIM-14 will load all PDP-14 simulated memory with the instruction NOP (0000). Thus, unwanted instructions can never be in the simulated program. All instructions must be entered by the user. SIM-14 also sets the values of all inputs and outputs for local mode execution to OFF state.

SIM-14 may be restarted without destroying the user's program or assigned input and output values by setting the switch register to 2201 and then pressing the LOAD ADD and START switches.

If SIM-14 ever appears to be out of control in the local mode and the user cannot type commands, SIM-14 probably is executing the PDP-14 program and is in a closed loop. The user should set switch 9 of the PDP-8 switch register ON (with switch 8 OFF), and control will return to SIM-14. New commands may then be typed.

To prematurely terminate typing by SIM-14, simply set switch 5 of the PDP-8 switch register ON and then OFF.

SWITCH REGISTER KEY

A very useful aid for operating the PDP-8 switch register is pictured below. The key may be mounted over the 12 switches of any PDP-8 family computer. The starting address switch settings for the Loader, Editor, BOOL-14, SIM-14, and PAL-14, are identified with symbols. The switches operated when using SIM-14 are also marked (Figure 10-6).

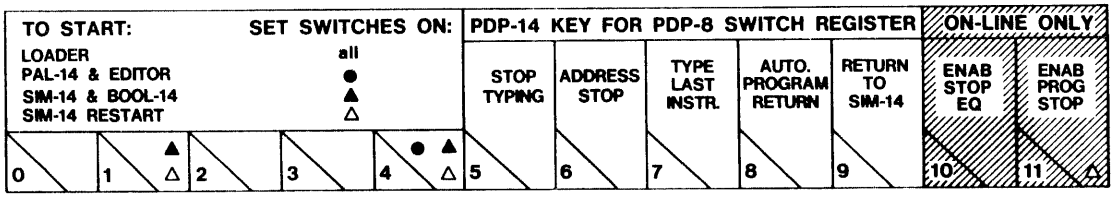


Figure 10-6

CHAPTER 11
INSTALLATION

BASIC FACTORS AFFECTING SYSTEM INSTALLATION

The three basic factors that affect the PDP-14 system installation:

a. electrical noise
b. heat dissipation
c. cabling and maintenance clearance.

The following discussion explains the importance of each of these factors, as an introduction to an orderly installation scheme.

Electrical Noise

Electrical noise abounds in many common industrial environments. Large electrical noise pulses are generated by motor controls, brush-type motors, and any other devices which abruptly switch large amounts of power and create arcing. Plasma flame cutters and high-power RF heating units are the source of the worst noise. These noise signals can travel many yards from their source, into cables and anything else in the vicinity.

Electronic switching circuits use electrical pulses in their normal operation. They must be protected against accepting electrical noises from the environment which might be accepted as valid control signals. Although the obvious place to eliminate hash is at its source, this is not always feasible, and can be quite expensive.

Early designs of solid-state control equipment were extremely sensitive to hash, and required elaborate protection from the hostile environment they were supposed to be controlling. If a computer was used, it was placed in an air-conditioned and humidity-controlled engineering area, often far from the controlled equipment. In many cases, such cumbersome solutions to hash problems just weren't practical. And so, solid-state control equipment acquired a bad reputation and was put on the shelf.

The PDP-14 AND Electrical Noise

Solid-state design has advanced greatly since the early 1960's. If the installation instructions in this chapter are complied with, you can put the PDP-14 control system in nearly any industrial environment without concern for noise problems. If the installation is to be located near absurdly high sources of electrical noise (such as plasma cutting equipment), some additional precautions may be needed, and you should consult a PDP-14 Applications Engineer.

The PDP-14 control system is protected in three ways:

a. Circuit design techniques which reject false pulses of electrical energy
b. Hash filtering or shielding on cables and other sensitive points
c. An installation arrangement which provides an overall shield protection.

Installation considerations which reduce noise problems are as follows:

a. The entire system is to be mounted in a standard NEMA 12 enclosure, and operated with the enclosure door closed (except for maintenance operations, when this is not possible). The NEMA 12 is an effective noise shield.

b. The Control Unit (CU) cover is to be in place for all normal operations, as a second level of shielding.

c. I, O, A, and S boxes are to be placed nominally between the CU and the entering AC signal cables, to reduce noise transfer.

d. An arrangement of AC and control cables, given later in the chapter, helps reduce noise transfer even further.

These requirements are similar to those of any ordinary control system.

Heat Dissipation

The PDP-14 system is designed to be cooled by natural convection and does not require special equipment. The system produces heat from two sources:

a. Current from the low-voltage power supply dissipated in the electronic circuits of the CU and I, O, S, and A units;

b. AC current dissipated in the circuits of the I and O boxes.

AC is dissipated in both I- and O-box circuits. Since each input of the I box is a 1.5 VA load, all 32 inputs active would consume 1.5 x 32, = 48 VA, or about 44 watts. A more reasonable estimate would be that only half the inputs would be active on the average, or a load of 22 watts. The total dissipation of all eight possible I boxes is, thus, about 22 watts x 8, = 176 watts.

The resistive dissipation of each O-box triac AC circuit is about 8 watts with a maximum 5 ampere load. But the maximum distributed load is 2.5 ampers per circuit, or 4 watts. Assuming that an average of half the outputs are turned on, each O box consumes 4 watts x 8 outputs, or 32 watts. All 16 O boxes dissipate about 16 x 32 watts, or 512 watts.

The CU requires from 90 to 175 watts for its low-voltage power supplies.

The basic system power dissipation is thus:

- CU power	90 Watts
- one I box	25 Watts
- one O box	32 Watts
	147 Watts

The maximum system dissipation for a full-size PDP-14 system is (rounded figures):

- control system power	175 Watts
- eight I boxes	175 Watts
- sixteen O boxes	510 Watts
Total	860 Watts

This figure will increase if you have a control requirement which allows most of the outputs to be on for an extended time. Reduce this figure for less I or O boxes.

Cooling requirements affect installation in these ways:

a. The CU is mounted at the bottom of the NEMA 12 for best cooling, and clearance is provided on all sides.

b. The I, O, A, and S boxes are mounted in vertical columns with unobstructed air flow at top and bottom.

c. The NEMA 12 is specified at a minimum size for proper convection cooling (minimum airspace and surface area).

d. A maximum enclosure surface temperature is specified. Note that the surrounding air must be somewhat cooler than the maximum enclosure temperature to allow heat transfer from the control system.

Cabling and Maintenance Clearance

A PDP-14 system installation will involve these three kinds of wiring:

a. Flexible wiring supplies 115 Vac to the CU

b. Flat ("ribbon") cables carry control signals between the CU and the various boxes.

c. No. 14 AWG leads carry ac signal and power circuits from terminals on the I and O boxes to and from the controlled device.

The CU mainframe is hinged at its left side to swing open for service. Since the AC feed and all control cables enter the CU on the left, clearance is needed for both the free swing of the mainframe and the bundle of cables. Modules plugged into the mainframe will protrude several inches to the left when the unit is fully open for service. Allowance must be made for this space.

Each I, O, or A box requires one ribbon control cable. An S box requires either one or two, depending on whether it is a half or a full box. Each cable plugs into a box, and bends either left or right, along the entire row of boxes, until it meets the CU. Thus, one end of the row will have a thick group of several control cables which must fit in the space above the end box. A minimum clearance must be left above each box (or row of boxes) for these cables. AC wiring will normally be groups of No. 14 AWG leads passing in the vertical spaces among I and O boxes. These wire groups may be stiff and bulky, and need clearance to the left and right of each box. It will be useful to leave some extra space beyond minimums in these areas.
Maintenance clearance is simply the space left around all parts of the control system to allow work to be done and parts to be changed. The CU must be able to swing out 90° from the enclosure, and also to lift from its hinges for removal. Generally speaking, a control system which is mounted in somewhat greater space than bare minimum requirements, will be much easier to install and service.

ENVIRONMENTAL SPECIFICATIONS

The entire PDP-14 System has been designed to operate reliably in rigorous industrial environments when properly installed. Table 11-1 contains the DEC System Specifications as:

Table 11-1
PDP-14 Environmental Specifications

- The entire PDP-14 system complies to J.I.C. electronic industrial standards applicable at time of manufacture.
- The PDP-14 CU, and all I, O, and Auxiliary Boxes are to be mounted in a standard NEMA 12 enclosure system. The NEMA 12 is assumed to be normally sealed - this means also that the door must be closed when the PDP-14 is operating (except for maintenance).
- Corrosive atmospheres shall be kept from the PDP-14 System by the design and proper use of the NEMA 12 enclosure. Extensive exposure to corrosive atmospheres might cause system failure.
- The surface temperature of the NEMA 12 must be between 0^{o} and 55^{o} Centigrade (32 to 130^{o}F) at all times. This temperature range assumes all thermal effect of the surrounding atmosphere and of any radiation (such as sunlight) which may strike the NEMA 12. Assuming proper free-air space within the enclosure, the PDP-14 System is designed to be cooled by closed convection, with no need for forced air or air-conditioning.
- Relative humidity shall be within 10 to 85 percent. No special moisture-proofing of individual modules is required.
- A vibration of 1.25 g's maximum at frequencies from 0 to 100 Hz sinusoidal in each of the three normal axes, can be tolerated.
- The PDP-14 System is highly noise-immune. However, absurdly strong sources of electrical noise should not be placed in the immediate area of the NEMA 12 enclosure.

TYPICAL INSTALLATION

The previous discussion of installation factors indicates a general mounting scheme. The sample installation, shown in Figure 11-1, embodies these requirements. Other layouts are of course permissable, if they follow the general considerations already given, and do not exceed the environmental specifications of Table 11-1.

ENCLOSURE

The entire PDP-14 system, including the CU and all I, O, A, and S boxes, is mounted in a standard NEMA 12 enclosure. The steel box keeps the control system clean, and provides both mechanical and electrical noise protection. The minimum size enclosure which provides enough free cooling air and mounting space for a basic system is:

48 in. high
36 in. wide
12 in. deep or greater

Since the basic system consists of one CU, one I, and one O box, you will require a larger enclosure for systems with more units. You may also wish to use a larger enclosure to make mounting and maintenance easier.

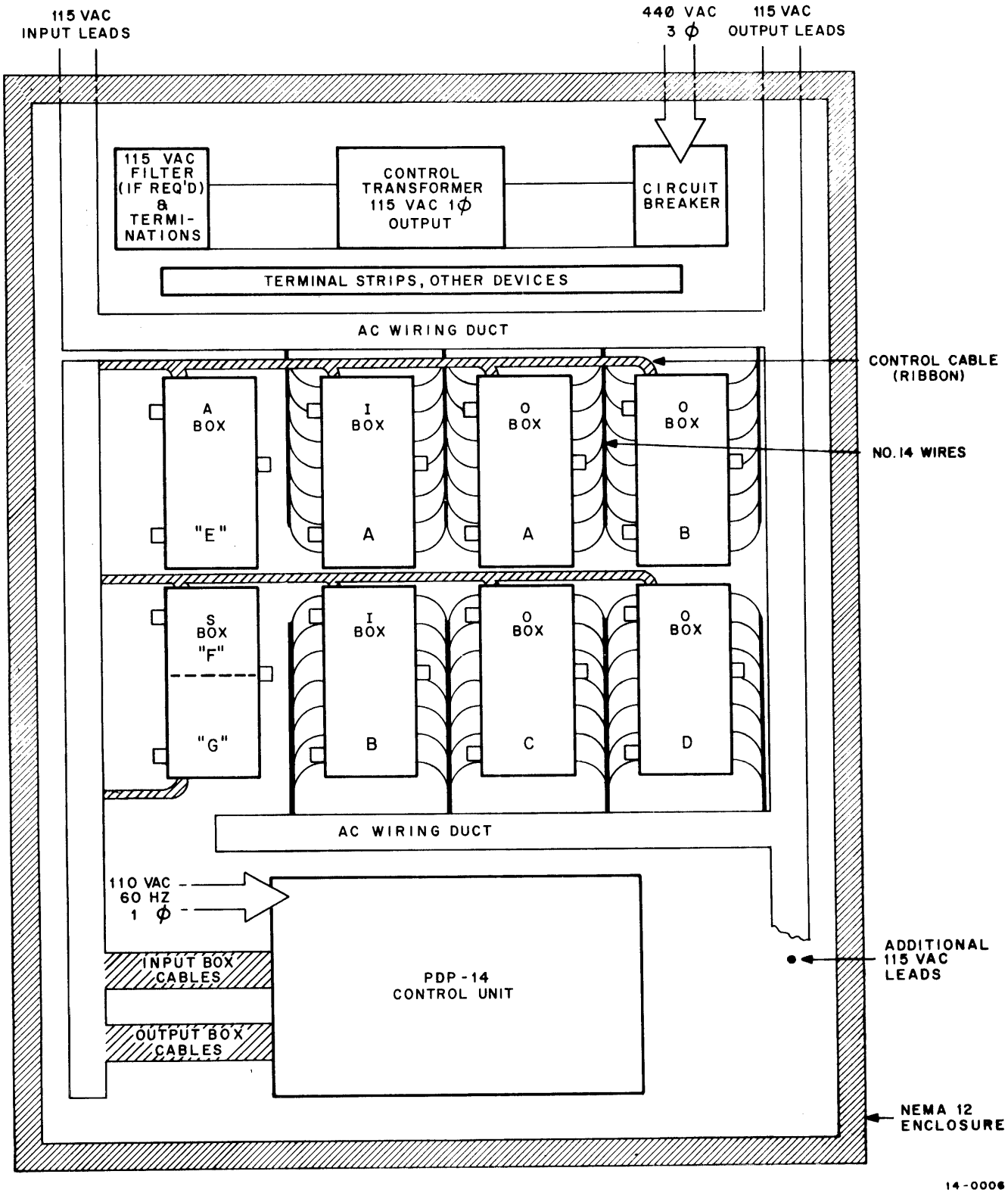


Figure 11-1 PDP-14 System Layout Example

CAUTION

Installing the PDP-14 system involves drilling operations which will produce metal and paint articles. These particles could short out electronic components and cause permanent damage if allowed to enter the PDP-14 system. This damage is not covered in the warranty. Therefore, DO ALL DRILLING FIRST, CLEAN UP the area, and THEN mount the parts of the PDP-14 system. Protect all system components from contamination.

ELECTRICAL GROUNDS

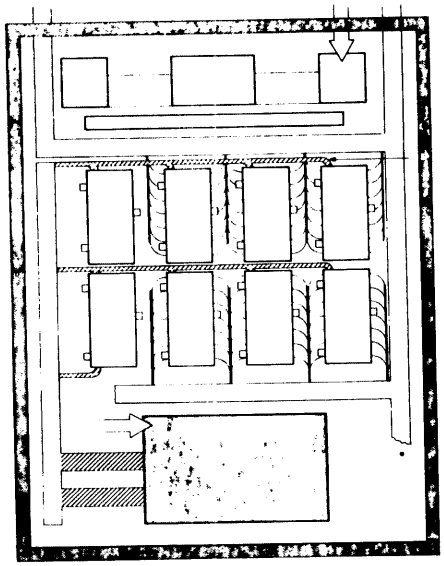
For proper electrical shielding, the NEMA 12 enclosure must be bonded to the normal ac grounds system, which must be low impedance to true earth ground. This grounding normally occurs when the ac metal conduit is installed. Ensure that insulating paint is removed or that paint-cutting fittings are used to establish good grounding to the chassis. Install a bonding strap between the rear mounting panel and the enclosure frame.

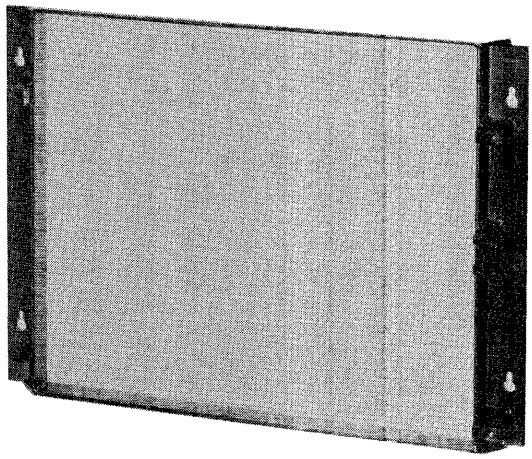
Each unit of the PDP-14 system must be kept at the same relative ground potential, by establishing metal-to-metal contact through the mounting panel. Ensure that the hardware which attaches the various units penetrates the panel paint to make firm contact with the metal surface. Remove small spots of paint or use lock-washers which will penetrate the paint under bolt pressure.

SYSTEM POWER AND OTHER DEVICES

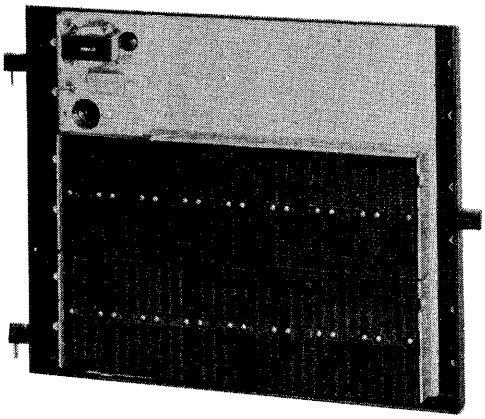
The ac power chain and the other devices shown in Figure 11-1 (top center) are optional for each installation. In this example, 440 Vac enters the enclosure at upper right and is controlled by a fused circuit breaker assembly. The voltage is reduced to 115 Vac by a control transformer, and filtered to remove line transients (hash), if necessary. This power is available at upper left to power both the CU and I and O box ac circuits. Box ac power requirements are discussed in Chapter 3. They will depend on the number of ac circuits activated and their load. The control transformer could be as small as 250 VA for a very small system, and above 50 KVA for a very large one. 115 Vac may, of course, be provided from outside the enclosure. If a computer is to be attached to the same circuit, be sure to add its power requirements. (The PDP-8/L computer and teletype require 500 VA.)

CONTROL UNIT MOUNTING

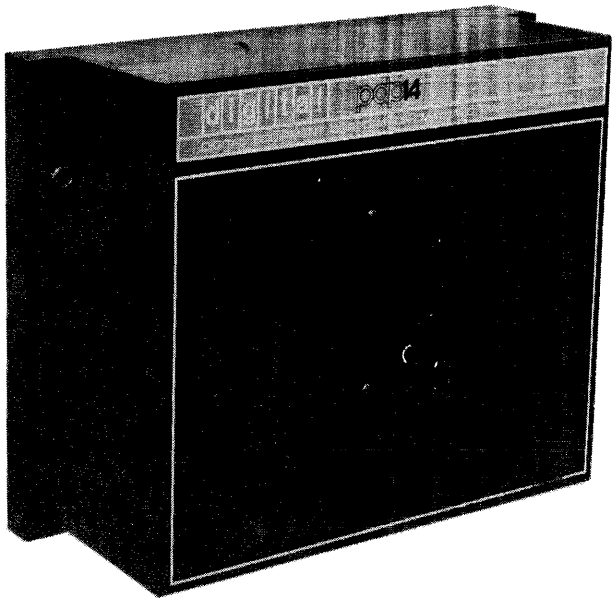




BASE FRAME



MAIN FRAME



SHIELD COVER

Figure 11-2 Control Unit Housing Components

The other devices indicated in the figure may be electromechanical timers, counters, or if necessary, small contactors. Large-current switching devices should always be mounted outside the enclosure.

The CU is installed in the lowest (assumed coolest) part of the enclosure. Thus it is as far as possible from interfering hash which could enter the enclosure through the AC wiring at the top.

The CU, although shown as one mechanical unit, consists of three separate articles, as shown in Figure 11-2.

a. A heavy metal base frame is attached to the mounting panel of the NEMA 12 using 1/4-inch dia. mounting bolts. The keyhole cutouts are placed over the bolts and the base frame drops in place; the bolts are then tightened. The base frame should be mounted flush to the NEMA mounting panel, so that the wiring normally within the frame is protected from falling debris.

b. The base frame provides wiring protection and a hinged mounting support for the CU mainframe, with open hinge pins, which drop into the base frame from above. This mainframe contains a wired panel of connector sockets accepting all CU circuits and cable modules. All permanent cabling enters the mainframe at the left, or hinge, side. Strain reliefs are provided to form a small clearance loop in each cable. The mainframe swings open to access its rear wiring, and is normally held closed by a locking pin at the right side.

c. The entire mainframe assembly is covered by a SHIELD COVER of sheet metal. This cover provides isolation from electrical noise and physical protection for the CU modules. It attaches to two locating pins at the sides of the mainframe, and is released by pressing the pushbuttons of both handles at the same time while pulling straight outward.

The mainframe is completely covered except for an aperture for control cables and AC power on the left side, vent openings, and a maintenance access port in the front. A small metal panel is screwed in place over this port when it is not used. The CU should always be operated with its shield cover in place except during maintenance.

CU MOUNTING DIMENSIONS

Figure 11-3 shows both mounting dimensions and clearances required for the CU. The unit requires minimum mounting clearances of:

a. Two inches above, to allow the mainframe hinge pins to enter their mating hinge supports on the frame and drop in place, and for cooling air to leave the top vent.

b. Six inches to the left, for cables to leave the mainframe and be routed toward the I and O boxes and to the ac power source (including two inches for a cable duct).

c. About two feet forward (and slightly to the left), to allow the mainframe to swing out for servicing. This clearance need only be available when the NEMA-12 enclosure door is open for servicing.

d. Six inches below, for cooling air to enter, and for wire terminations on the bottom of the NEMA 12, if used.

e. One inch to the right, for cooling air.

These clearances may be increased to make installation and servicing even easier. Mount the CU within 10 degrees of vertical.

CU MOUNTING SEQUENCE

All mounting holes should be drilled before installation begins. The CU is shipped as one package, and must be dismantled for installation. First, 1/4-inch mounting bolts are prepared for the NEMA 12 on centers

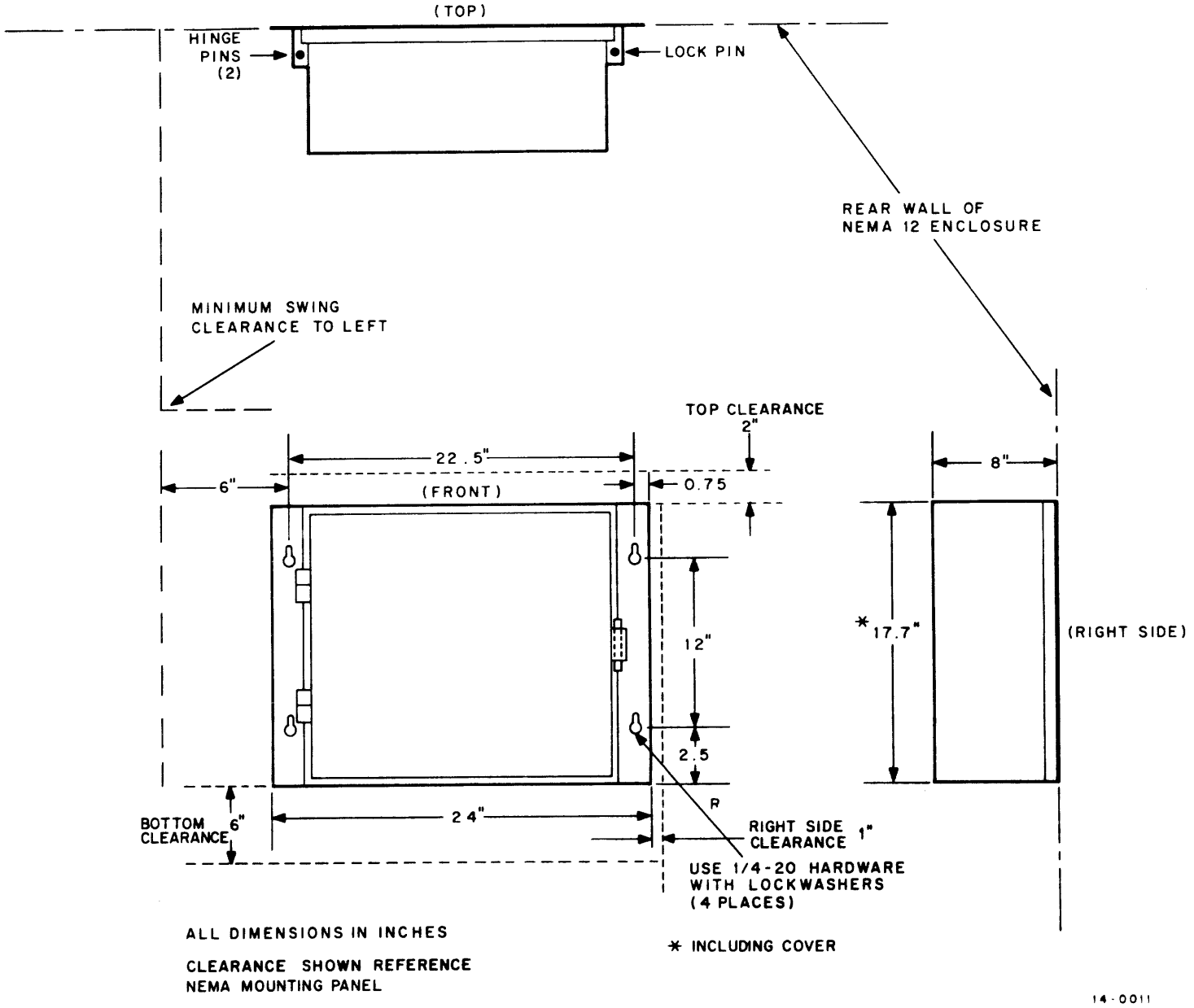


Figure 11-3 Control Unit Mounting Dimensions

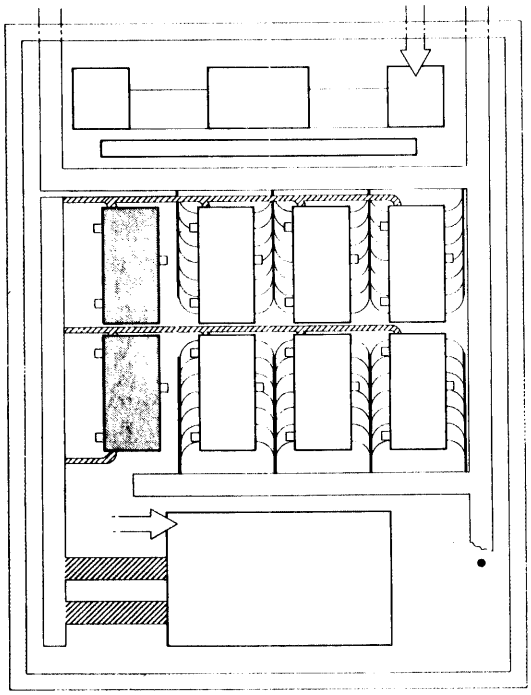
shown in Figure 11-3. The CU is unpacked, and the shield cover released by depressing the handle push-buttons and lifting the cover away. The mainframe and base frame may be left together, but for handling ease, temporarily detach the mainframe and place it aside.

CAUTION

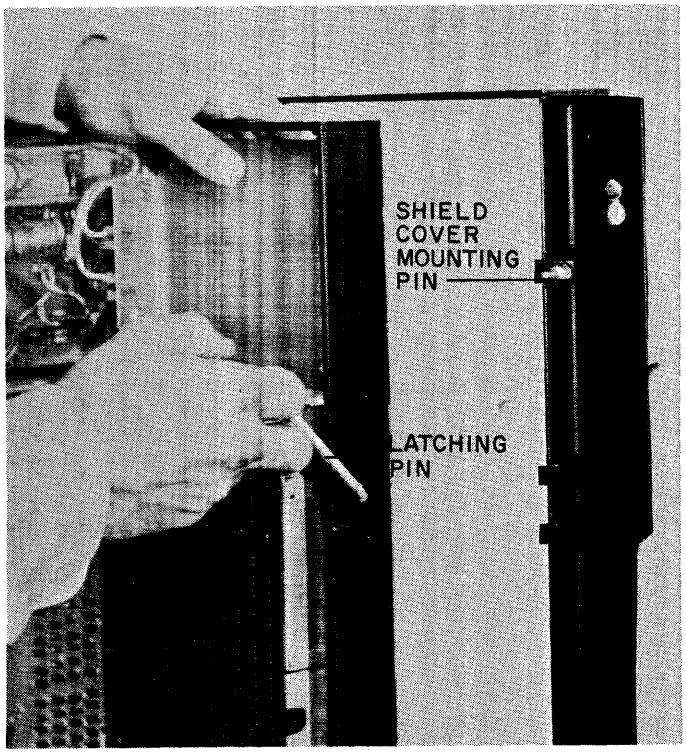
Be careful not to bend wiring pins which extend from the back of the mainframe.

The base frame is then bolted to the rear mounting panel of the NEMA 12 enclosure. Be sure to use locking hardware, to prevent vibration from loosening the installation. Slip the mainframe in place on its open hinge pins (Figure 11-4), and latch in place with its pin. Replace the shield cover. When all other mechanical installations in the NEMA 12 are completed, the cover will be removed again for cable insertions. This completes installation of the CU.

AUXILIARY BOX MOUNTING



The A and S boxes are mounted above the CU so that their control cables will have the shortest run to the mainframe sockets. This usually places them in the furthest left column of all I, O, A, and S box mounting, as shown above. A and S box clearance is the same as for I and O boxes. Note that each A-box control cable enters at the top left of the box, while each S box may require two control cables, at top and bottom left.



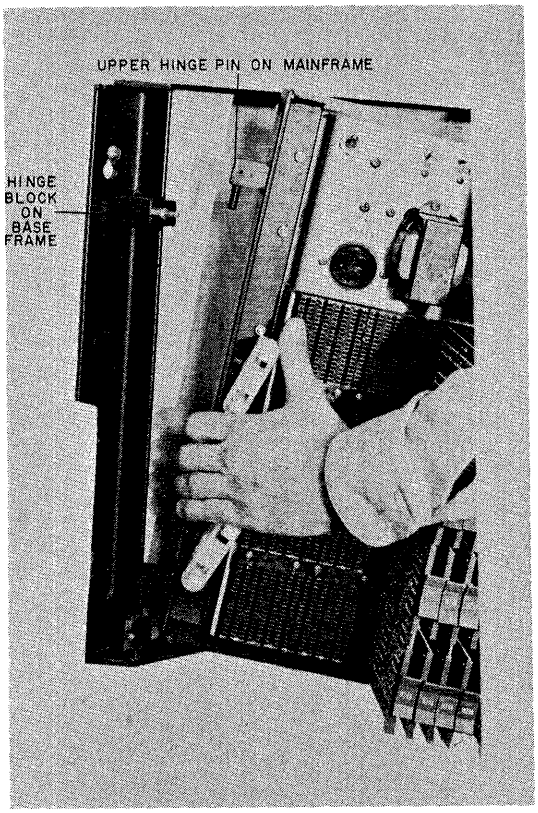


Figure 11-4 Control Unit Mainframe Mounting

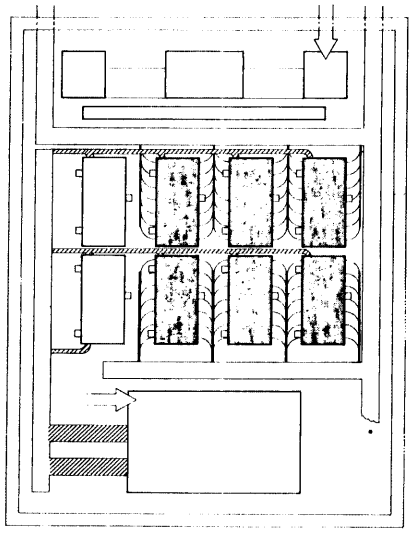


Figure C

I AND O BOX MOUNTING

I-and-O boxes are mounted in even rows and columns above the CU, as shown above. Clearance is allowed above and below the columns for cooling air, and between each box and on the sides for ac signal wiring. Space must be provided above and below each box for control cables which pass from right to left along the box rows. Each I-or O-box has one control cable, which enters the box from the top left.

I, O, A AND S BOX MOUNTING DIMENSIONS

Figure 11-5 shows all mounting dimensions for a single I or O box. Optional boxes indicated at top and at left show how columns and rows of boxes may be extended to any required length. Mount all boxes within 10 degrees of vertical, to promote proper convection cooling.

Approximately one inch must be left between box rows to allow cables to pass across the tops of the boxes. If boxes share vertical mounting centers, as shown, about 7/8-inch usable space remains between two boxes.

Boxes may be mounted in vertical columns and horizontal rows up to the limits of the enclosure. All control cables should run horizontally across the box rows to the left, and then down to the CU. Allow space along the left side of the mounting for this cable drop, along the right for AC wire ducts, and at least two inches above and below the box group for free air flow.

I, O, A, AND S BOX MOUNTING SEQUENCE

First determine the type and quantity of all boxes to be mounted, according to the demands of the system. Allow space for more boxes if the possibility exists that the system may be expanded later. Lay out the NEMA mounting panel and drill all holes for 1/4-inch mounting bolts on centers shown in Figure 11-5. Then bolt all boxes in place, using locking hardware. This completes the box installation. Control cables will be installed in a later step.

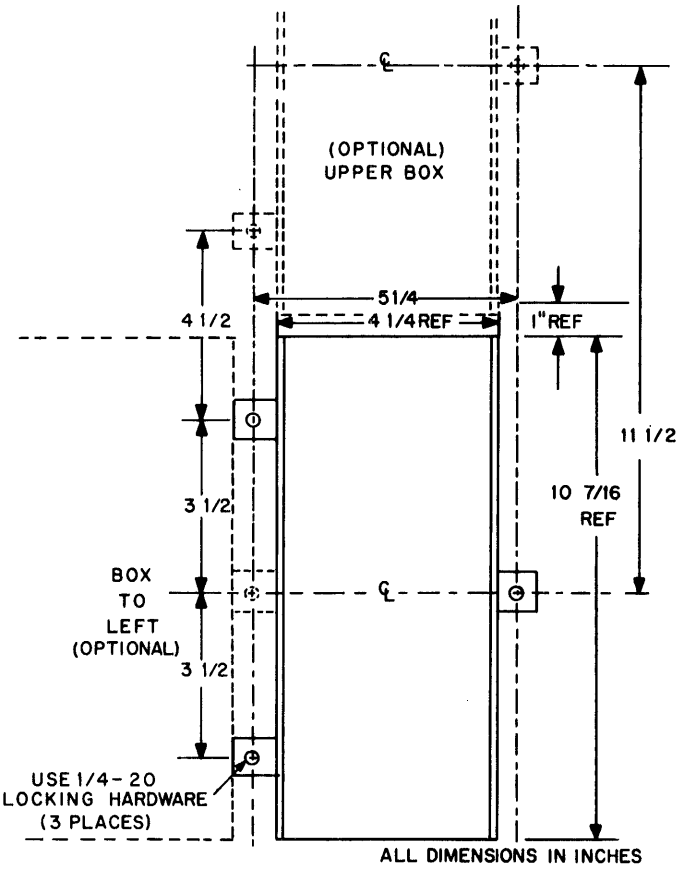


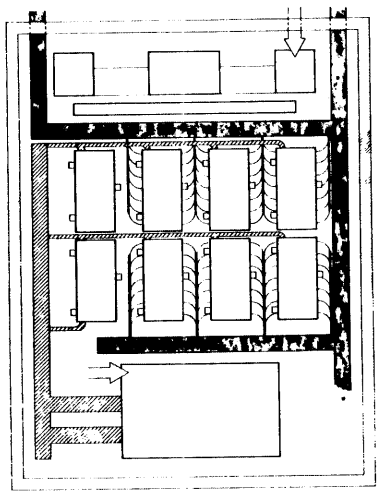
Figure 11-5 I or O Box Mounting Dimensions

WIRING AND CABLE INSTALLATION

This section shows how to install both control cables and AC wiring. The same procedure is used for initial work and for later additions to the system.

CAUTION

Insertion or removal of any PDP-14 system wiring must always be done with all power off. This includes both control cables and AC leads, as well as maintenance and computer interface connections. Failure to observe this caution could result in permanent damage to the circuits which is not covered by the warranty.



AC Wiring

The sample installation above shows the position of wiring ducts. Note that ac wiring duct is placed above and below the I-and O-box group, and single power leads are then run vertically between the box columns to their individual terminals. More ducting may be placed between box rows if space is provided. The control cables, in contrast, run horizontally along the box rows to the left, and then down a separate duct (shown crosshatched) until they exit, in two bundles, to the CU. This cable dressing minimizes transfer of noise from the ac wiring to the control cables, and should be followed as closely as possible. The ac power supply chain at top center in the drawing has already been discussed. It supplies both the CU and the box ac circuits with filtered 115 Vac.

To power the CU, run a flexible service cable (two No. 14 to 16 AWG leads) from the ac terminations at top left of the CU. Remove the CU cover, and swing the mainframe out fully. Figure 11-6 shows the back panel exposed. Loosen the two Phillips screws, which hold the protective cover, one full turn and slide the cover off. A power connector strip with two screw terminals is now exposed. Run the cable through the grommet in the left side of the mainframe, strip the ends, and install them on the connector strip. Leave a small service loop in the cable. Reinstall the protective plate and close and lock the mainframe.

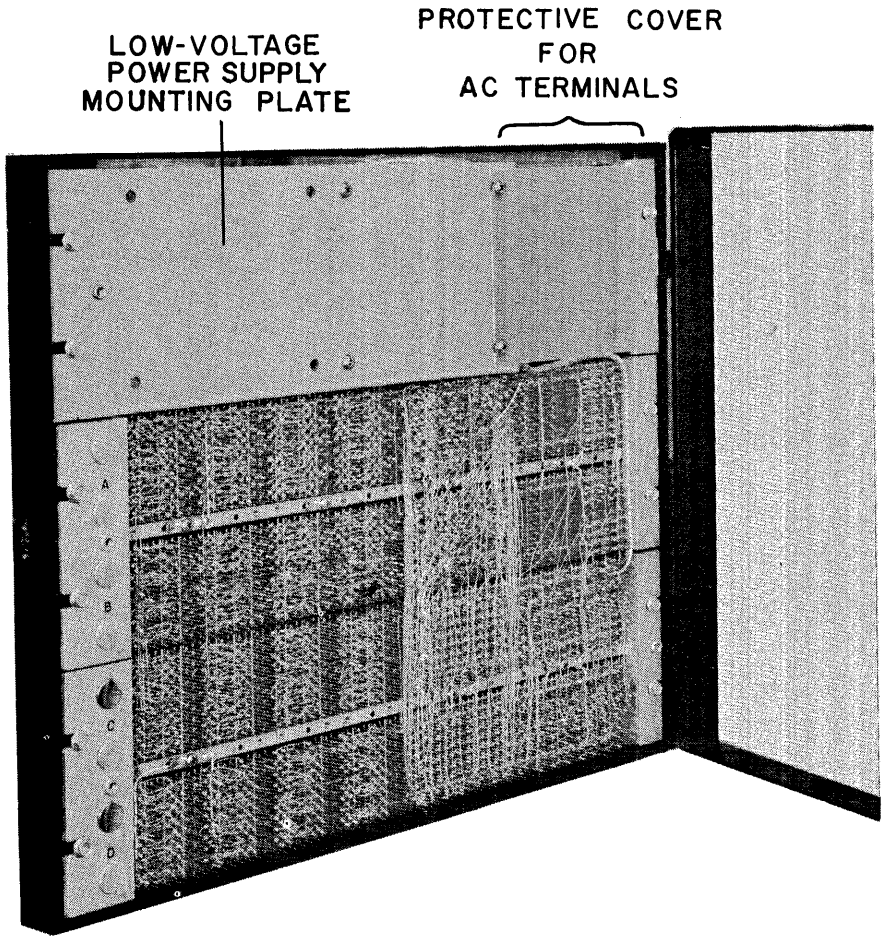


Figure 11-6 Mainframe Back Panel

NOTE

An ac convenience outlet is provided on the CU mainframe. This outlet is directly across the incoming ac, and is neither fused nor grounded through the CU. This outlet is ordinarily used for electronic diagnostic equipment or a computer, and external fusing should be provided to protect such equipment.

All box AC wiring is done with single conductor number 14 AWG copper with standard insulation. Initial checkout of the I and O boxes in the system requires that no ac wiring be installed on their terminals. This means that control cables are temporarily installed, as described in the next section, then computer test programs are run. The control cables are disconnected, the AC wiring connected, and the control cables then permanently reconnected. The wiring is installed behind the space reserved for control cables. This means it may extend from the mounting panel forward no more than three inches as it leaves the duct. Once the individual leads leave the duct and clear the control cables, they may be run forward to meet the screw terminals along each box front. Each terminal accepts one or two leads stripped about 1/4-inch and inserted under its integral clamping plate.

Figure 11-7 shows terminal connections to each I box. The entire box front is covered with a transparent plastic cover which protects from electrical shock and forms a wiring duct. The cover is removed for wiring by unscrewing four thumbscrews and pulling straight out. Each of the four K578 input modules has a 9-terminal barrier strip mounted on its front face.

One terminal must be returned to 115 Vac neutral, as marked. Ordinarily, it is easiest to loop all four neutral terminals on a single box and then return only one lead to the ac supply neutral. Each of the other eight terminals on the K578 is for an ac input circuit. Switched 115 Vac is to be provided to each terminal from whatever input sensors are used on the controlled device. A neon lamp corresponding to each terminal glows only when AC is being provided by the input sensor to the terminal.

Input circuits are identified by a alphanumeric combination. The letter refers to the specific I box from a possible eight (letters A through H), and the number to the box terminal (1 through 32). For example, if Figure 11-7 showed I box "D", then input D15 would be connected to the second from the lowest terminal on the left side. A list of the input circuits and sensors is produced in the programming operations discussed in Chapters 4 through 10. This list should be provided for wiring operations on "Input Assignment Sheets", one for each I box. Each I box should be assigned a letter according to programming instructions and should be marked with a small tag. All are then wired according to the assignment sheets. Leave protective covers off so that the control cables may be later inserted.

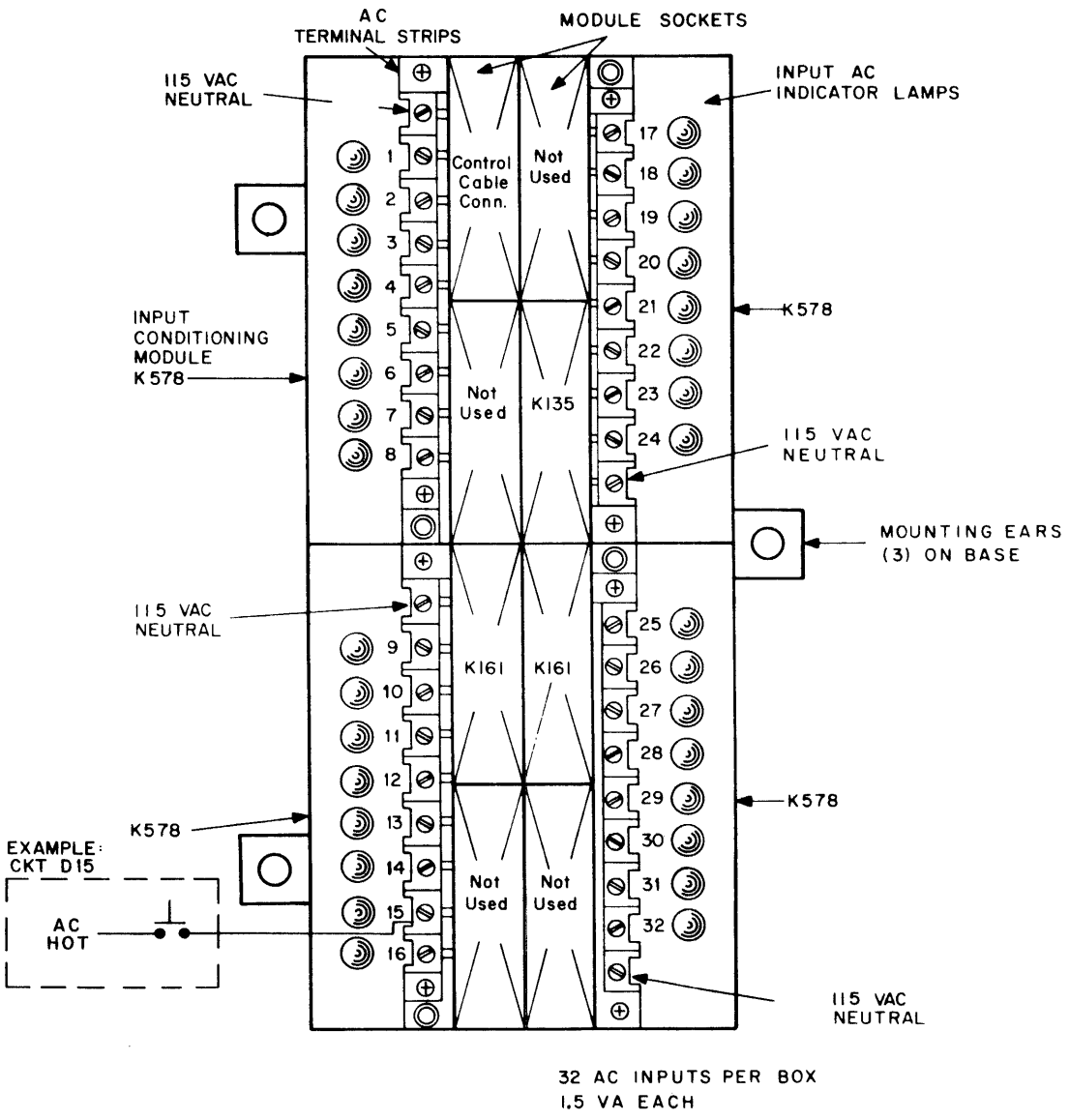


Figure 11-7 Input Box Terminal Connections

Output circuits are connected in the same general way as inputs. Figure 11-8 shows the output terminals for one O box. Each output switching circuit is electrically independent from every other, so that if necessary, it may be supplied from a separate source of 115 Vac. Ordinarily, you will supply an entire box from one power source, and loop the 115 Vac hot lead to each circuit, as shown in Figure 11-7.

As in the I box, each terminal strip has a single connection for ac neutral. But in the I box, this neutral was a common return for all circuits in the module. In this case, it is simply a common for the neon indicator lamps. The four neutral terminals in each O box should be looped together and tied to AC neutral (this is not shown in the drawing).

The four switched output terminals on each K614 module alternate with the looped ac hot leads, and are identified with a circuit number, as shown in Figure 11-7. Output circuit assignments should be provided on "Output Assignment Sheets", one for each box. An O box should be given a letter at this time (A thru S, excluding I, O, and Q), and marked with a tag. Every O-box terminal can now be defined by an alpha-numeric combination. For example, if Figure 11-8 showed O-box "N", then the switched terminal on the upper right would be circuit N9. All O boxes should now be wired with the circuits specified by the Output Assignment Sheets. Leave the protective plastic covers off the boxes so that control cables can be installed later. These steps complete all ac wiring for the PDP-14 control system.

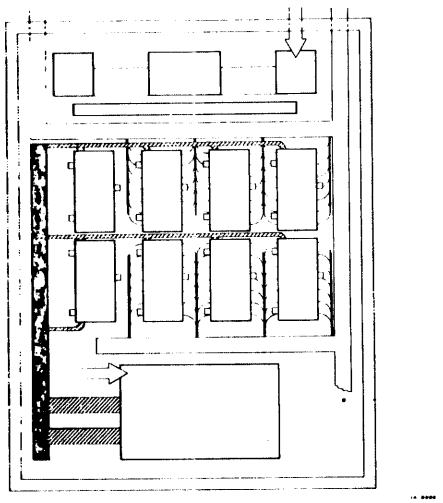


Figure E

Control Cable Installation

Control cables carry the CU command signals as well as low voltage power to the I, O and auxiliary boxes. They are flat, multiconductor wire packages called "ribbon" cables. Figure 11-9 shows the connection arrangement. Note that each cable extends from the left-hand side of the CU to one I or O box.

Each control cable is a length of ribbon cable terminated at each end by a cable connector module which acts as a flat plug for the sockets in the CU and box. The cable assembly (Figure 11-10) is DEC part

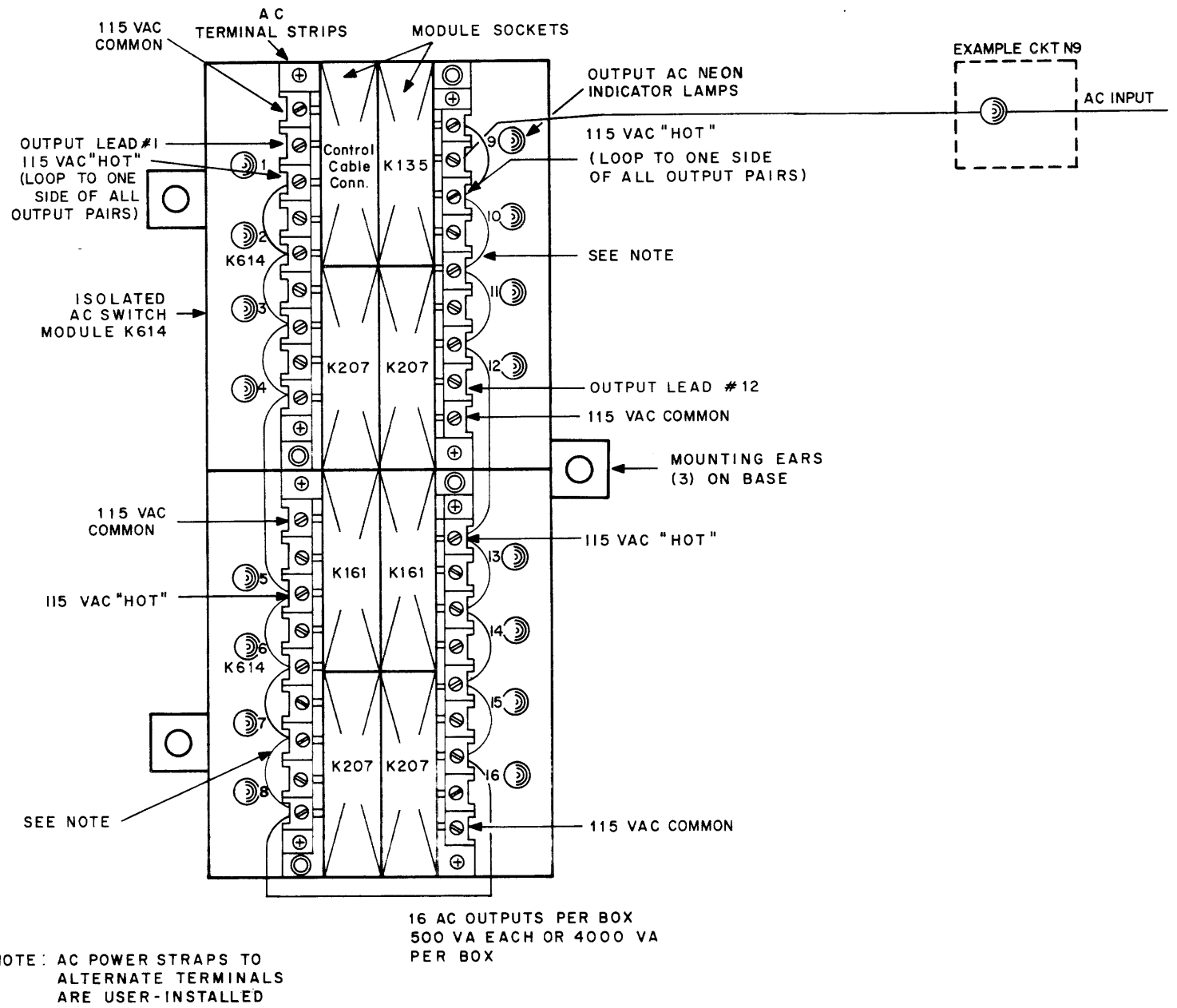


Figure 11-8 Output Box Terminal Connections

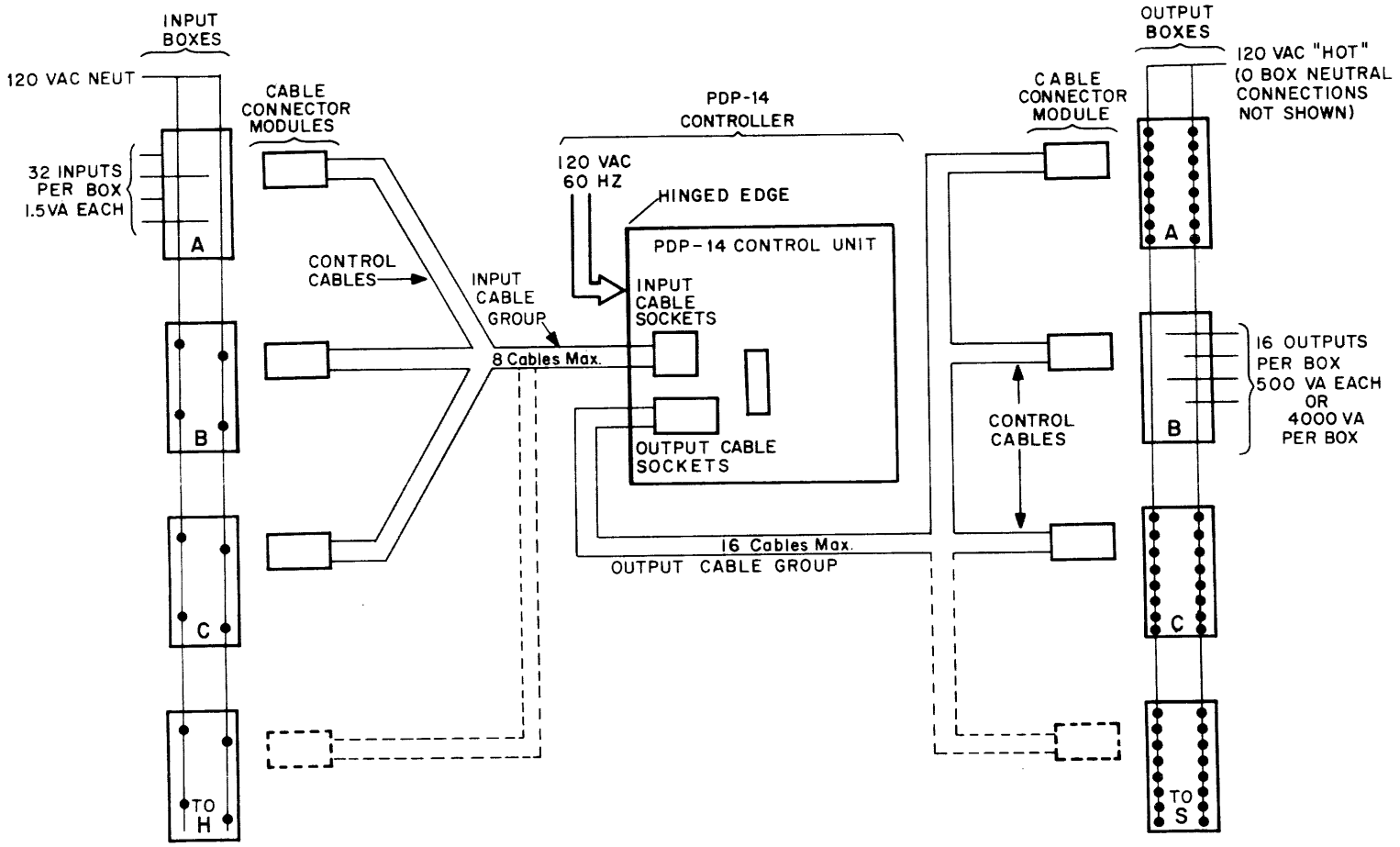


Figure 11-9 PDP-14 System Control Cable Arrangement

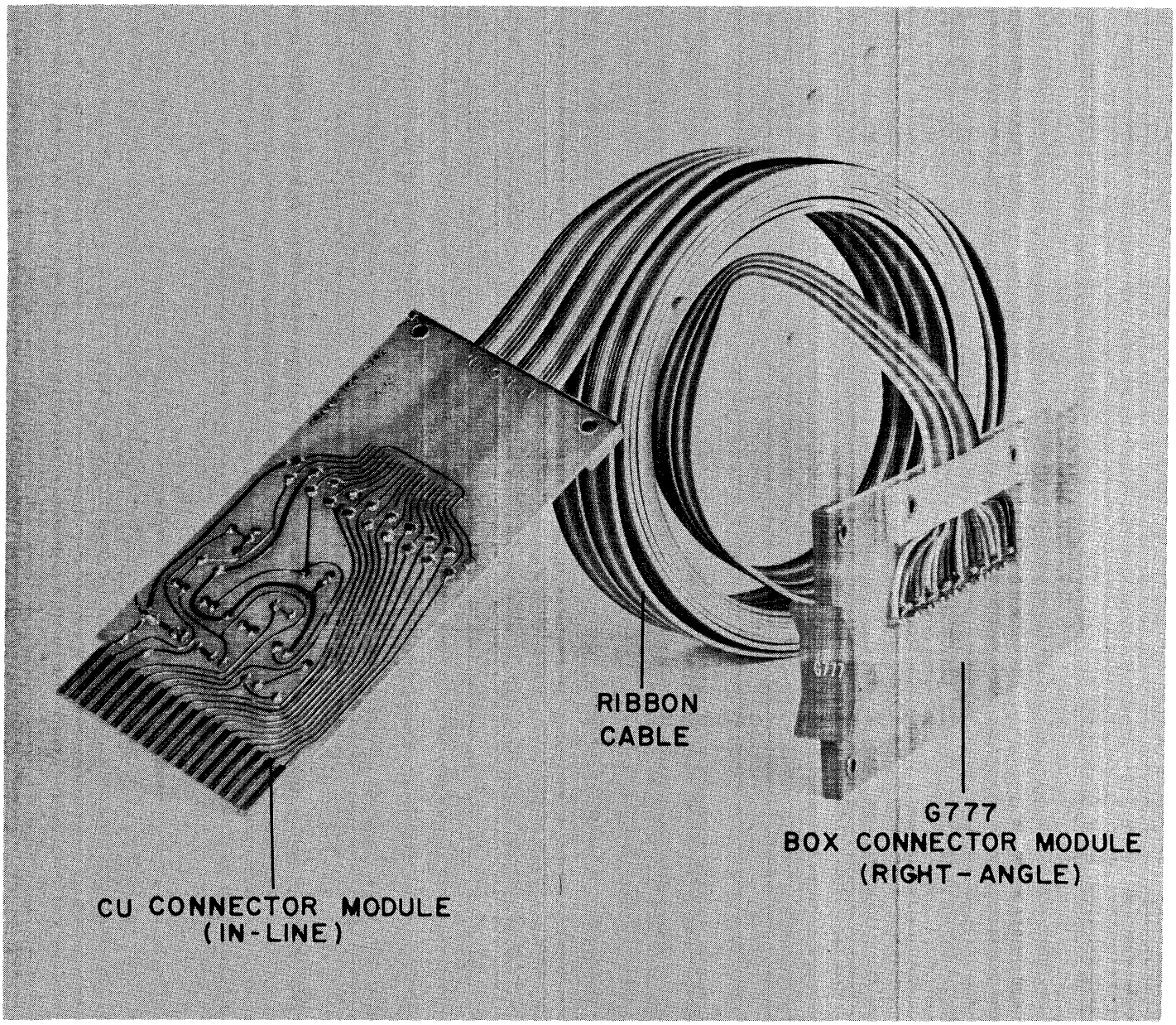


Figure 11-10 DEC BC-14A Control Cable Assembly

BC14A-XX* available and is supplied in standard lengths of 03, 05, 10, 15, and 25 feet. If no length is specified, the 10-foot length is supplied. The box connector module is at right angles to the cable and has a green handle marked "G777"; the CU module is in-line with the ribbon wire.

Each box is supplied with proper control cabling whose length should be stipulated on the box order. The cable assembly ordered should be long enough to extend from the particular box to the lower left-hand side of the CU, with about one foot of slack. This length of course depends on the individual mounting plan. One such cable is used for a half S box, and two for a full S box.

Box Cable Connections

Begin connecting control cables to the boxes closest to the CU (usually the A and S boxes). Starting with the box furthest left in the lowest row, insert the right-angle cable connector in the top left module slot in the box so that the cable comes out the top, as shown in Figure 11-11. (The full S box has an additional slot at bottom left; this cable comes out the bottom of the box.) The module is inserted in the slot by pushing carefully but firmly straight back until the handle is in line with the other modules in the box. Run the ribbon cable to the left over the top of the box and into cable duct at the left side of the NEMA enclosure. In a later step, the cable will be run down the duct and into the CU. Now mark the cable connector for the CU with the box type and letter, using a grease pencil or a small tag; for example, "A BOX B". Continue to the next box along the row to the right, install in the box and run the cable directly over the first box and into the duct. When the bottom row of boxes is completed, go on to the next upper row, until all cables are installed.

NOTE

Although the insulation on control cables is highly abrasion-resistant, do not run cables over sharp edges.

Always be sure to use the proper length cable to reach the CU, and mark the box identifier on the CU cable connector. All protective plastic box covers should now be reinstalled. Line up the finger screws with the box standoffs, as shown in Figure 11-12, and hand-tighten all four screws uniformly.

CU CABLE CONNECTIONS

The control cables connected to the boxes in the previous step are now in the duct, with the cable connectors marked. These cables can now be plugged one at a time into the CU mainframe. Remove the CU shield cover to expose the mainframe. Figure 11-13 shows the cable slots along the left side of the unit and their box allocations. The I-box slots are lettered A through H, and the O box slots A through S, to correspond with each box. This lettering does not appear on the CU. Since A and S boxes use O-box cable slots, you will need a list formed in the programming operation to indicate the proper slots for these boxes. A

*The "XX" is the length in feet.

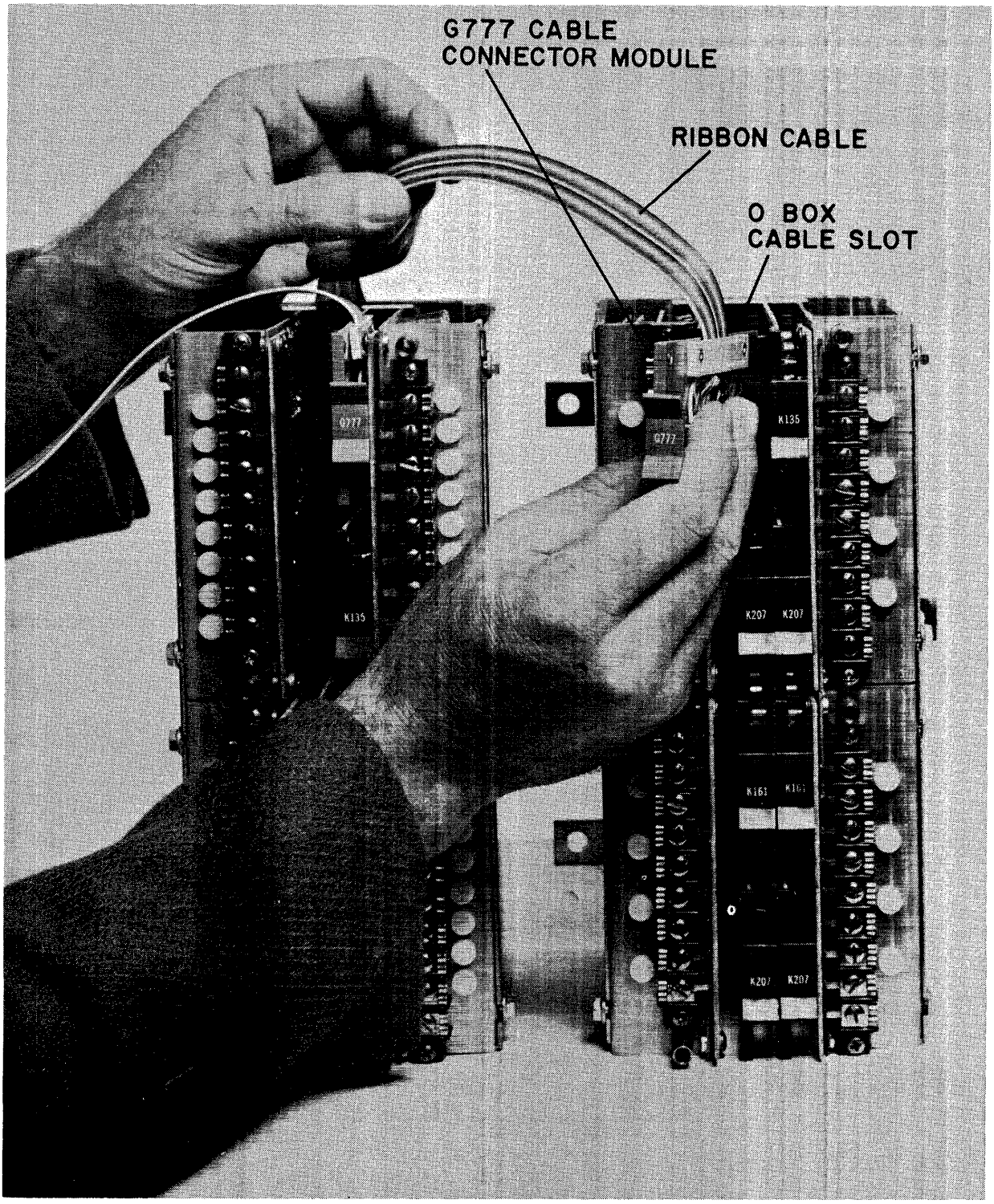


Figure 11-11 Box Control Cable Insertion (O Box Shown)

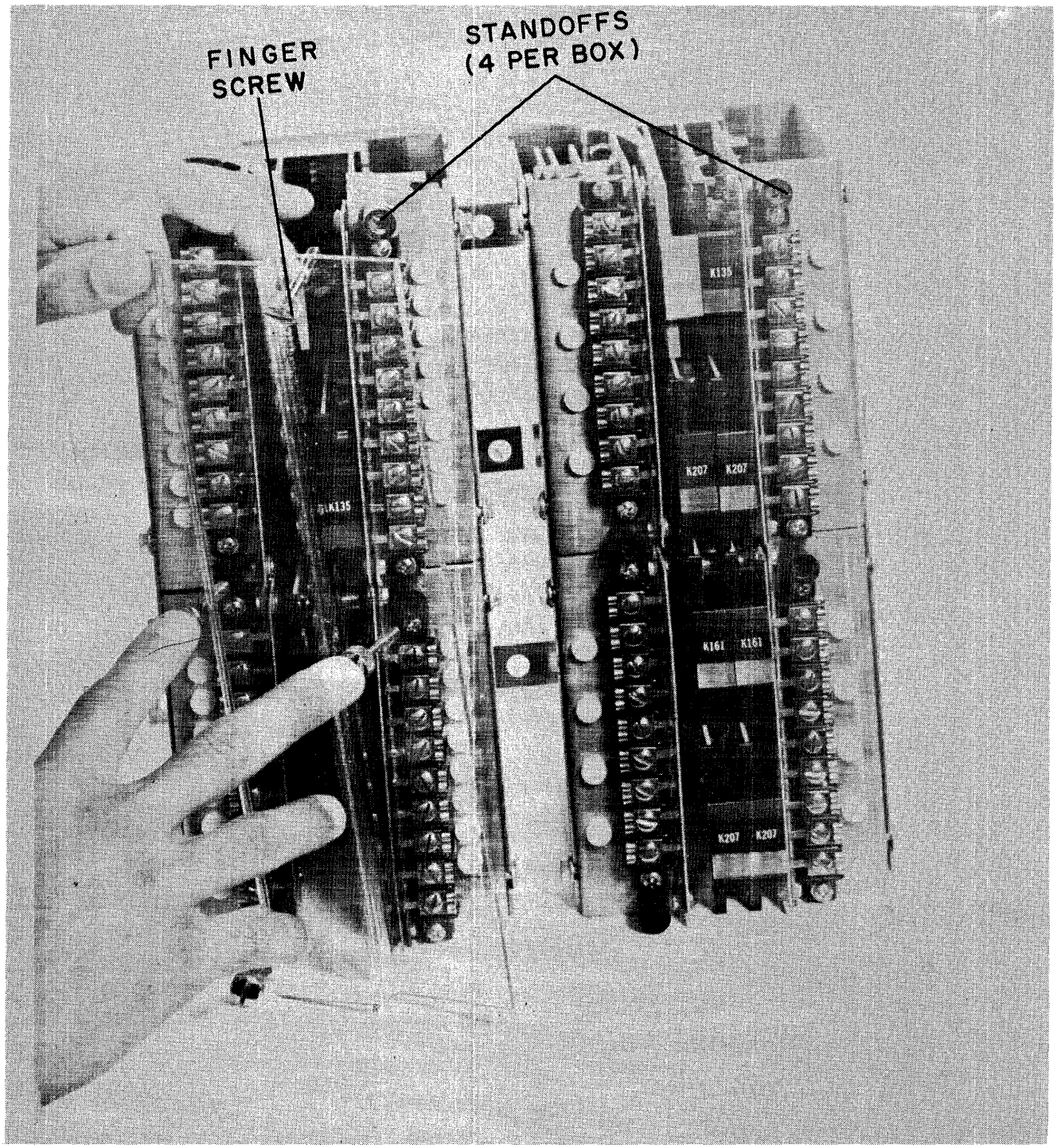


Figure 11-12 Installing Box Protective Cover

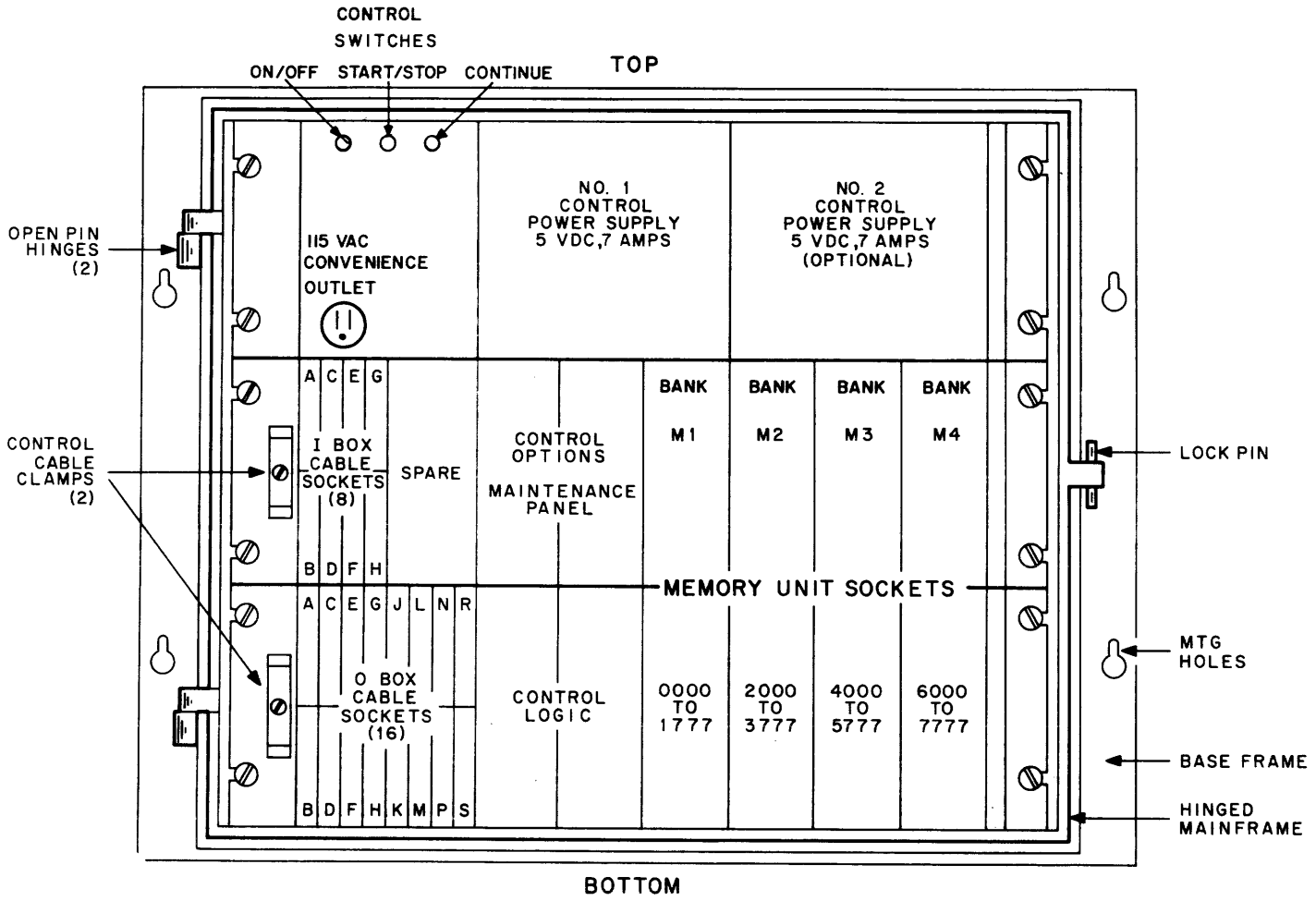


Figure 11-13 CU Cable Socket Allocations

cable is inserted in the CU in the same manner as in the boxes with the side of the module to which the cable wires are soldered to the right. Two key notches are cut in each connector module along the plug edges, and the shortest key notch should be at the top.

Figure 11-14 shows the connector for I-box "A" being inserted into the CU slot.

Install cables in this order:

1. All I-box cables
2. All O-box cables
3. All S-box cables (in 0 box slots)
4. All A-box cables (in 0 box slots).

Finally, the cables are formed into two flat bundles and run through the U-shaped cable clamps, as shown in Figure 11-15, and into the duct. Leave a small service loop between the mainframe and the duct so that the cables can move without binding when the frame is swung open. Place all slack cable in the duct, to complete the control cable installation. Figure 11-16 shows cables for I-box "A" and O-box "A" properly installed. If a computer interface is also to be installed, leave the CU shield cover off and proceed to the next section; if not, reinstall the cover.

CU POWER SUPPLY, COMPUTER INTERFACE, AND ROM INSTALLATION

Figure 11-17 shows the placement of equipment on the CU mainframe. It will be used in the following installation discussions.

CU POWER SUPPLY INSTALLATION

The CU power supplies, shown at the top of Figure 11-17, provide highly regulated dc to operate the circuits in the mainframe and also I-and O-boxes. Each of the two power supplies shown is rated at 5 Vdc at 7 Amperes. Number 1 power supply is shipped installed in the CU, and is sufficient to power a moderate sized control system. For increased numbers of I and O boxes, more control current is required, and must be provided by the optional Number 2 power supply. The following table of power consumption will allow you to calculate when a second power supply should be installed:

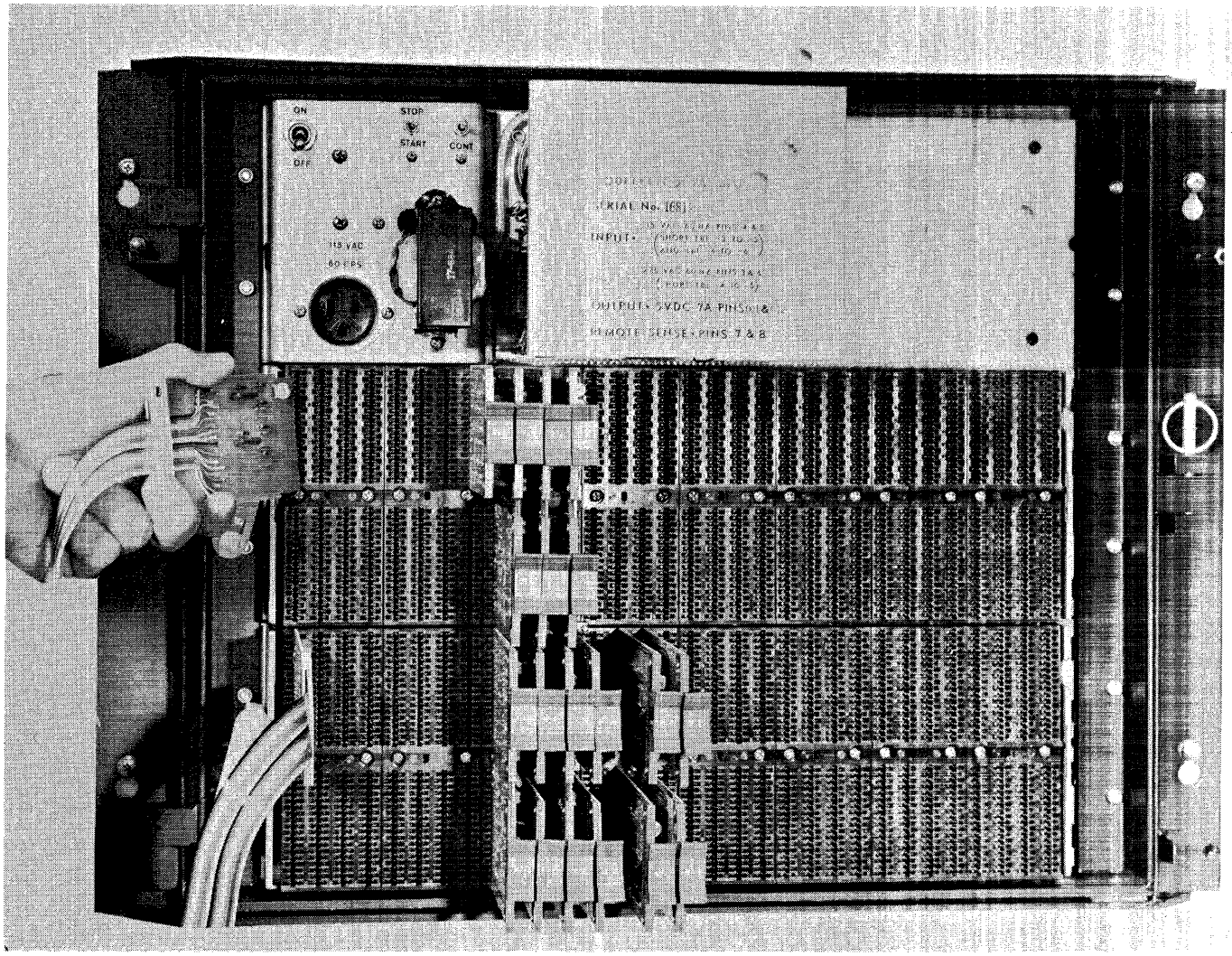


Figure 11-14 CU Control Cable Insertion

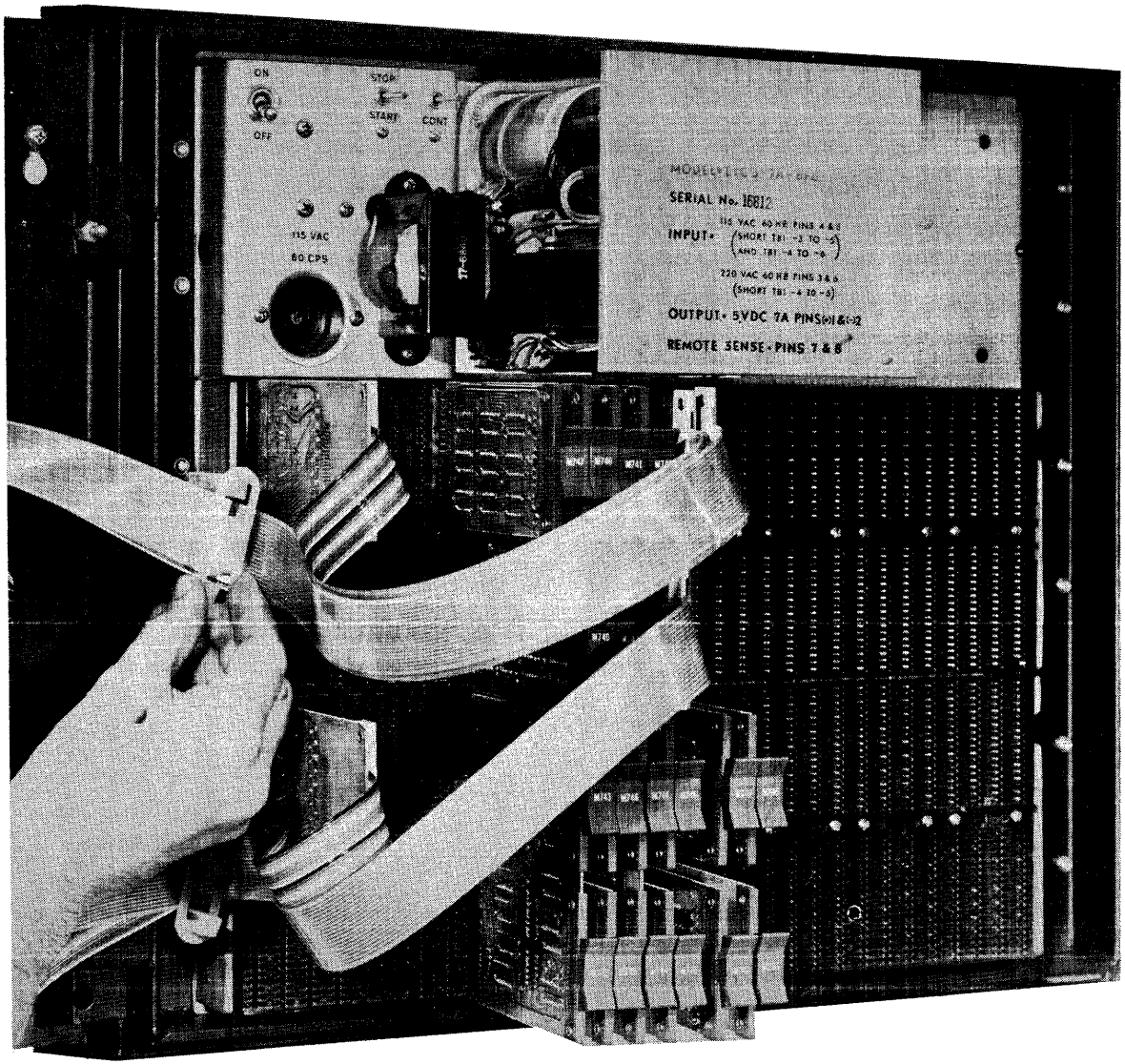


Figure 11-15 CU Control Cable Clamp Use

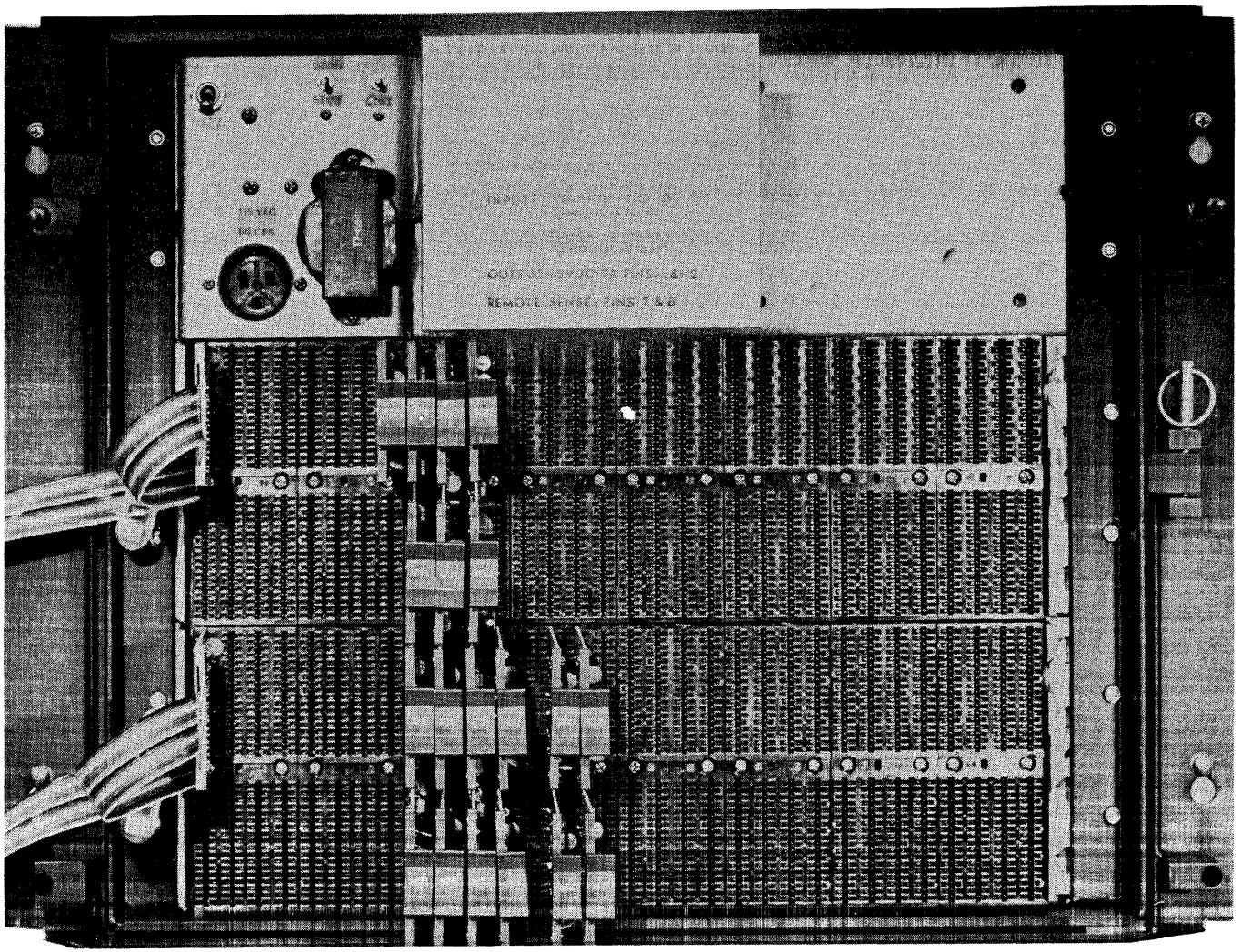


Figure 11-16 CU Control Cables Installed

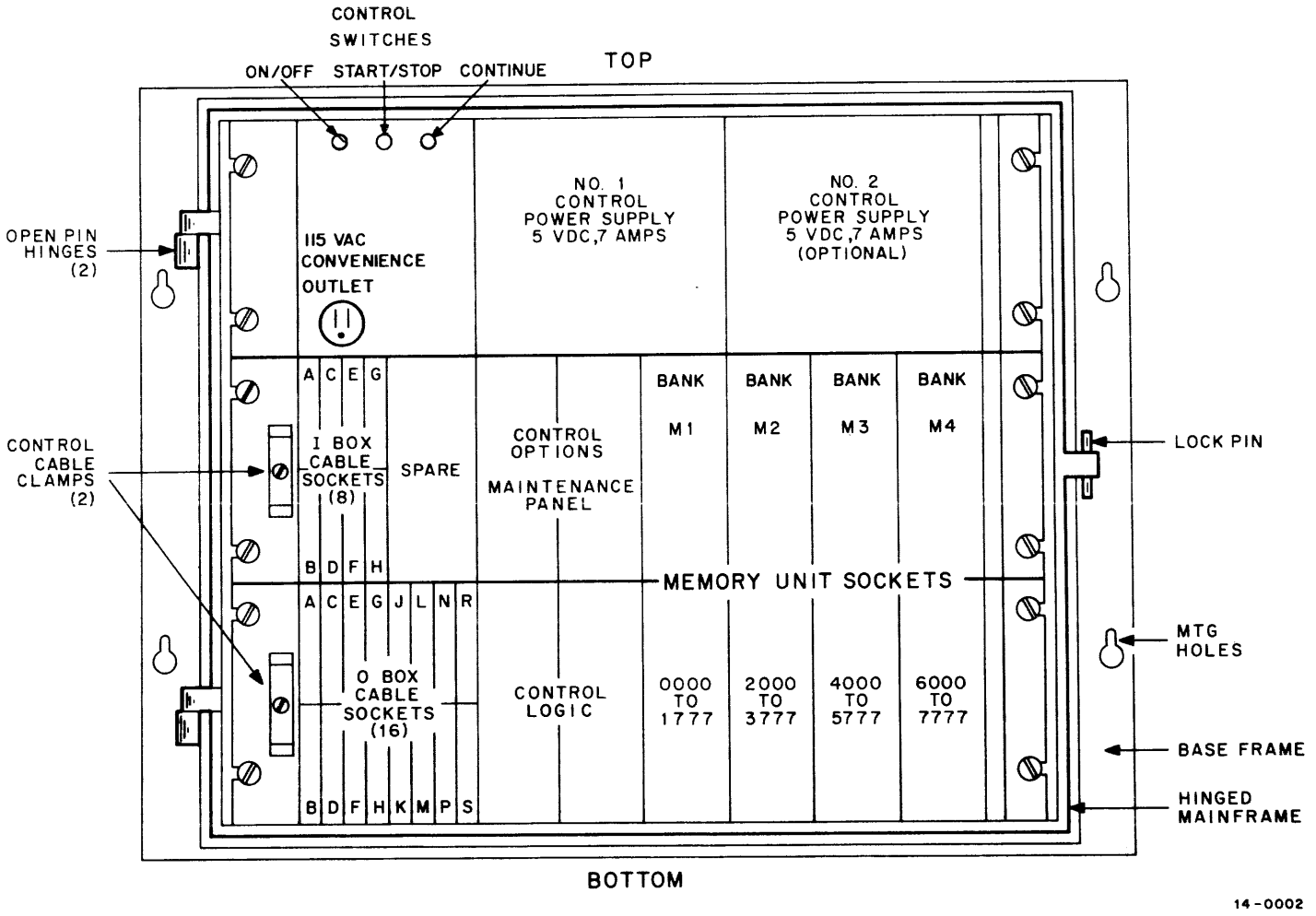


Figure 11-17 CU Mainframe Allocations

PDP-14 SYSTEM POWER CONSUMPTION

Unit	DC Amps @ 5V
14 Control Unit	2.50
Memory 1K	0.50
I Box	0.05
O Box	0.60 max, 0.50 avg
A Box	0.60
Half S Box	0.40
Full S Box	0.80
Computer Interface	0.40

Here are three examples of calculations for systems of various sizes:

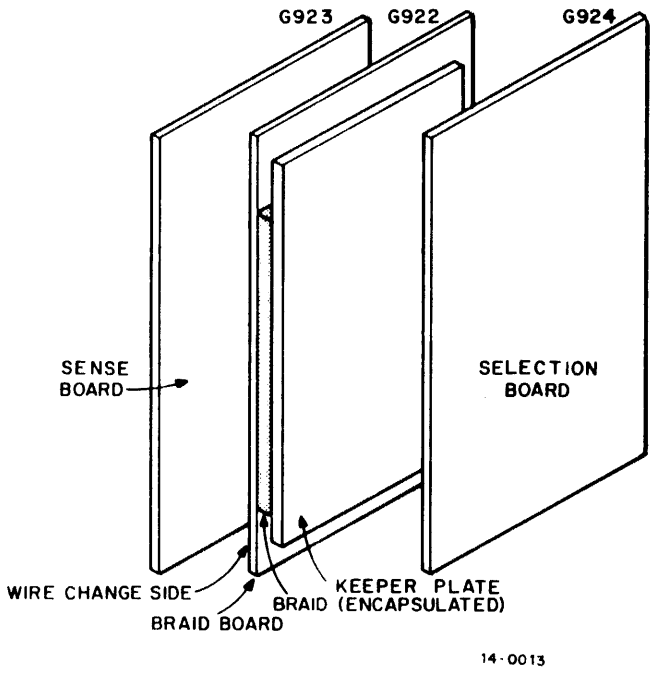
Examples:	DC Amps @ 5V
Basic Machine	
14 CONTROL	2.50
1K MEMORY	0.50
1 - I BOX	0.05
1 - O BOX	0.60 (max)
	3.65 amps
Largest system with one power supply:	
14 CONTROL	2.50
2K MEMORY	1.00
2 - I BOXES	0.10
4 - O BOXES	2.40 (max)
1 - A BOX	0.60
COMPUTER INTERFACE	0.40
	7.00 amps
Largest possible system:	
14 CONTROL	2.50
4K MEMORY	2.00
8 - I BOXES	0.40
16 - O BOXES	8.00 (avg)
COMPUTER INTERFACE	0.40
	13.30 amps

Note that a large system should not be expected to have all outputs activated at once, and therefore will not use full control current for each box. This means that an average O-box current value may be used for calculations.

Installation of Number 2 power supply is a simple bolt-in process. First, disconnect all AC power. Swing open the mainframe and attach the power supply in the space shown with the four bolts provided. Attach the supply connecting leads to the screw terminals provided on the right side of Number 1 power supply. (Follow the connection diagram packed with the power supply.) Close and latch the mainframe.

ROM INSTALLATION

The Read-Only Memory (ROM) stores all the instructions for operating the control system. The CU has space for one to four ROM banks DEC part no. MR-14, as shown in Figure 11-16.



The G923 Sense Board is attached directly to the G922 Braid Board, and they form one plug-in assembly. The G922 module contains the braid, or actual memory storage unit, and is different for each bank. Each bank also requires a G924 Selection Board, which is included in the MR-14 package. When more than one MR-14 is used, each Sense-Braid assembly is marked with an "M" or bank number as shown in Figure 11-17. This number appears on a sticker attached to the keeper plate. When a G922 module is provided, be sure to immediately determine its bank number so that it is installed properly.

Memory Bank	Instructions Stored
M1	0-1777
M2	2000-1777
M3	4000-5777
M4	6000-7777

To install a ROM bank, first obtain all Sense-Braid assemblies (G923) and G924 boards. The CU mainframe contains four slots for each ROM bank. Beginning with Bank 1, install the G923 assembly in the sockets furthest to the left, as shown in Figure 11-18. The G923 will slide into the first slot, the G922 will contact the second slot, and the third slot remains unused. The G924 board is next installed in the fourth slot, as shown in Figure 11-19. Although the boards are shown angled in the photos to show placement, do not angle them for insertion or removal. Instead, place the module squarely in the slots and make sure all sockets accept the board ends. Then press the module firmly straight back until it is seated completely in the sockets. A side-to-side rocking motion may aid insertion. The G923 double assembly will require considerable pressure, so be sure not to allow an individual board end to bend under the full pressure of insertion. Continue the installation for Banks 2 through 4, if they are used. This completes the ROM installation.

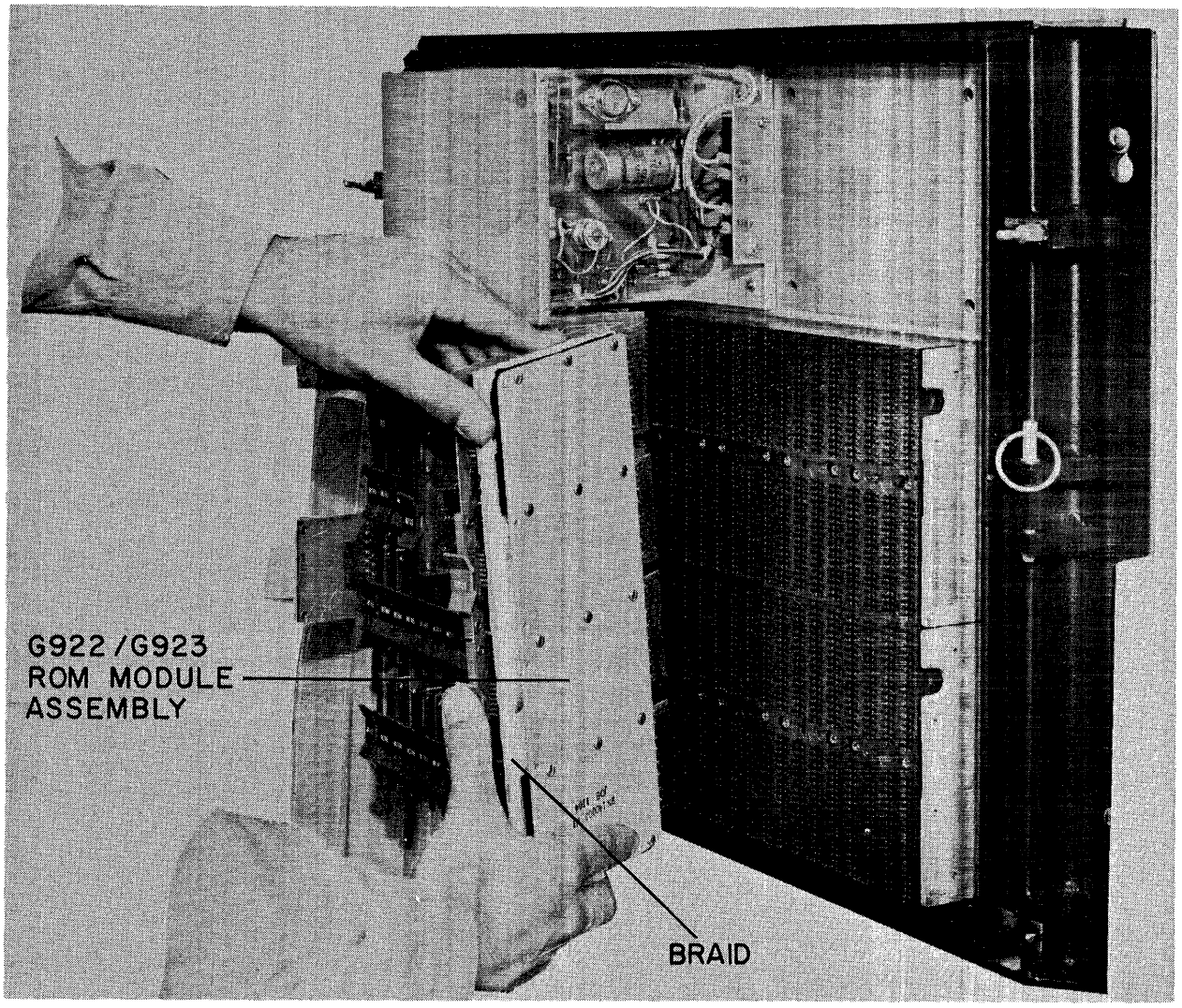


Figure 11-18 Installing G922/G923 ROM Modules Assembly

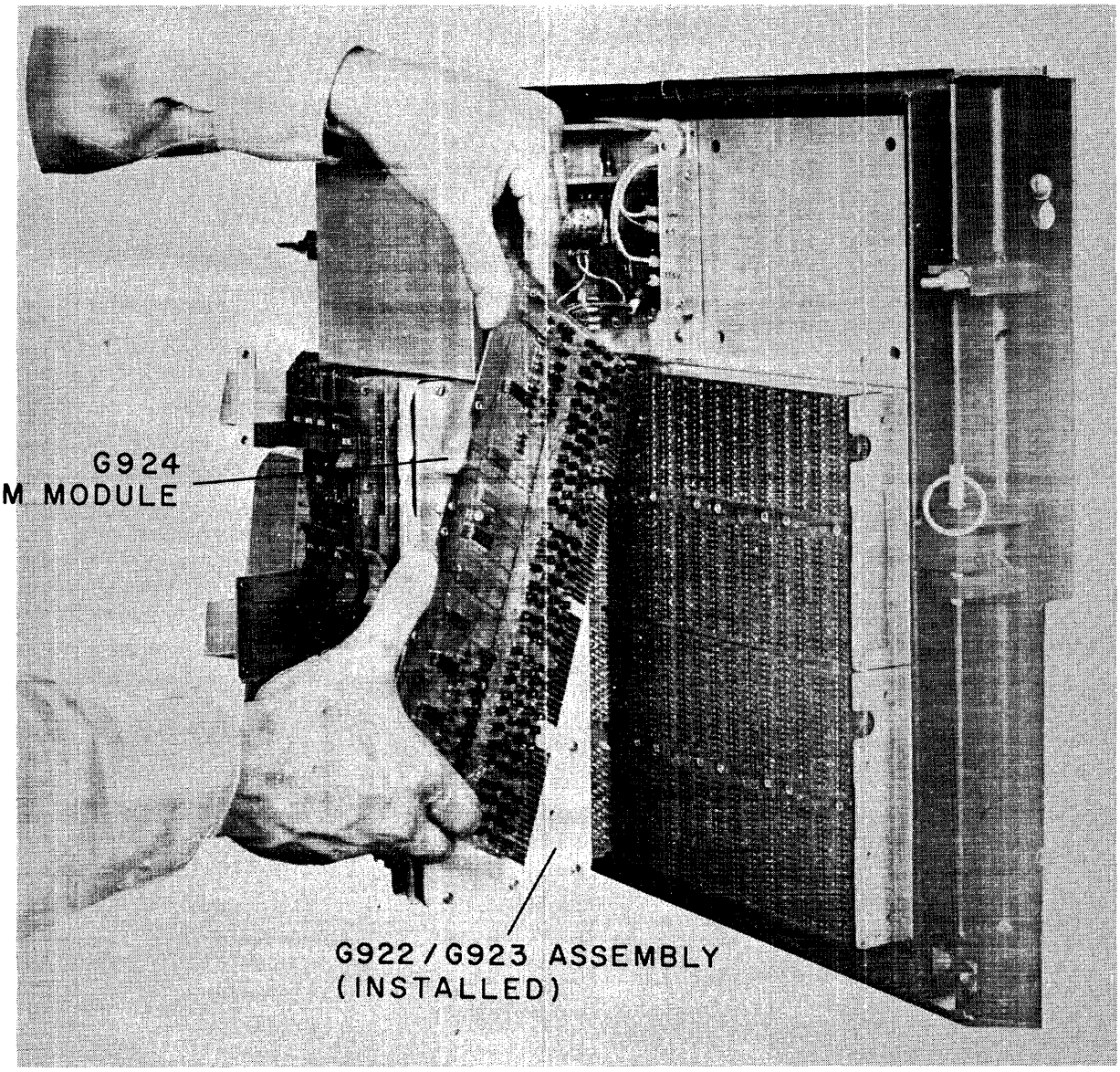


Figure 11-19 Installing G924/ROM Module

COMPUTER INTERFACE INSTALLATION

The computer interface for the PDP-8/I or 8/L consists of these modules:

1 - M1Ø6
1 - M745
4 - M746

and these control cables:

for the PDP-8I, 3 - BCØ8C-15
for the PDP-8L
or PDP-12, 3 - BCØ8A-15

The control cables and modules are installed in the CU mainframe in the same manner described for other similar units. For the proper module and cable socket positions, refer to the "CU Socket Allocations" chart supplied with each CU. Ordinarily, the computer interface is the last to be installed in the mainframe, since other modules are shipped installed. The computer interface is shipped installed if initially ordered with the CU. Figure 11-20 shows a completed computer interface installation. These cables are slightly different from those used for control cables. This flat cable is called "flexprint".

CAUTION

Any computer or other electronic equipment connected by control cables to the PDP-14 system must share the same AC neutral and metallic ground potential. A difference in neutrals, such as caused by use of two AC power sources, could cause large current flows in control cables and permanently damage electronic equipment. The convenience outlet in the CU mainframe provides external equipment the same AC power as the system.

The M921 Device Selector jumper module allows each PDP-14 to respond to a unique device code transmitted by the computer. The Device Selector shipped will respond to codes 16 and 17, since these have been specifically assigned. If more than one PDP-14 system is to be connected to one computer, an expander is required, and is available as DEC DB-14. Each PDP-14 in the expanded system must be assigned a different device code for identification. The Device Selector in each interface must be rewired for the proper code. A PDP-14 Applications Engineer can explain the wiring changes and provide a list of suggested devices codes for use in a particular computer system.

INITIAL CHECKOUT

Before applying power to any new PDP-14 system, an orderly check should be made of the complete installation. Be sure that:

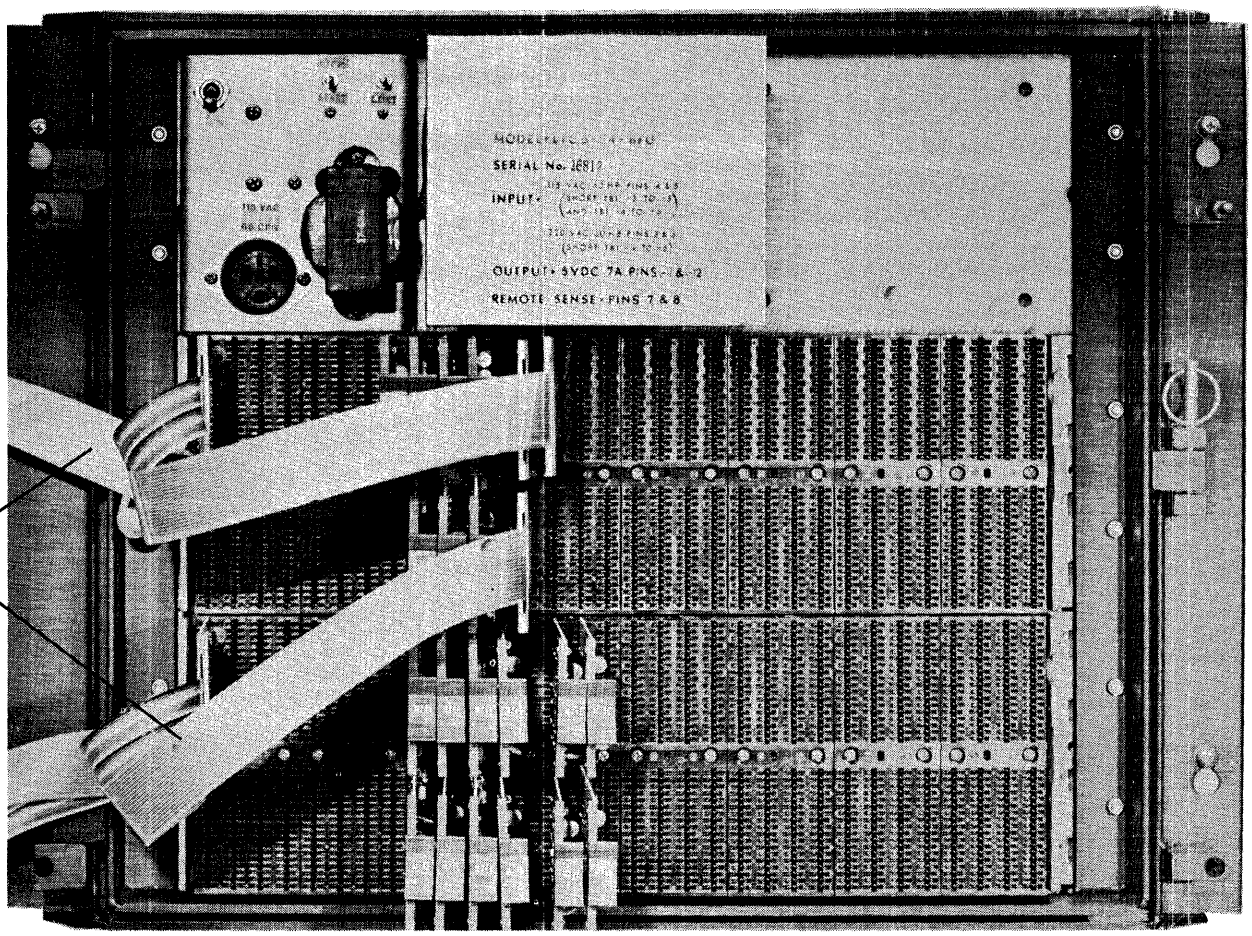


Figure 11-20 Computer Interface Installations

a. No metal or paint chips or other foreign matter has fallen into equipment or is left in the NEMA enclosure

b. Proper grounding has been maintained between all units

c. Output circuits are not shorted and loads will be properly distributed among K614 modules

d. Control cables are completely seated in their proper slots, and all other modules are firmly in their sockets

e. An optional low-voltage power supply has been installed if needed

f. Only 115 Vac, 60 Hz will be supplied to the PDP-14 system when power is energized

g. People around the installation know that the NEMA enclosure is to be kept closed at all times. A lock is suggested

h. If ROM banks are available for this installation, they are correctly in place.

The first powered operation of the PDP-14 system should be to check out electrical operation of all its components. This initial test is done with the PDP-8/I or 8/L computer, using these programs:

a. TEST-14, which checks the CU, and I, O, and S boxes; -(I and O box AC wiring must be disconnected for them to be tested with this program);

b. VER-14, which checks the ROM contents and operation; and

c. A BOX-14, which checks all A boxes.

The computer is connected to the system through its interface, and the test programs are run according to the instructions provided with them. The result of this testing will be printed on the Teletype printer connected to the computer, including faults, if any. If module changes are considered necessary, refer to Chapter 12 for further maintenance information.

INITIAL STARTUP

The CU automatically starts the PDP-14 system whenever ac power is applied. The control system then operates constantly, waiting for commands from the control device. If power is interrupted, the control sequence simply starts from the beginning whenever power is reapplied.

The CU control panel is shown in Figure 11-21. The large toggle switch to the left is the power switch, which interrupts ac to the low-voltage power supplies. This switch is normally left in the ON position, the NEMA door is closed, and power is applied to the entire control system through the circuit breaker interlocked to the door. The CU will then automatically start its control sequence. If the toggle switch is turned OFF, the control operation stops and all outputs are also turned off. The smaller switches on the right of the control panel are marked START/STOP and CONTINUE. The START/STOP switch turns the control sequence on and off, but without turning any outputs off.

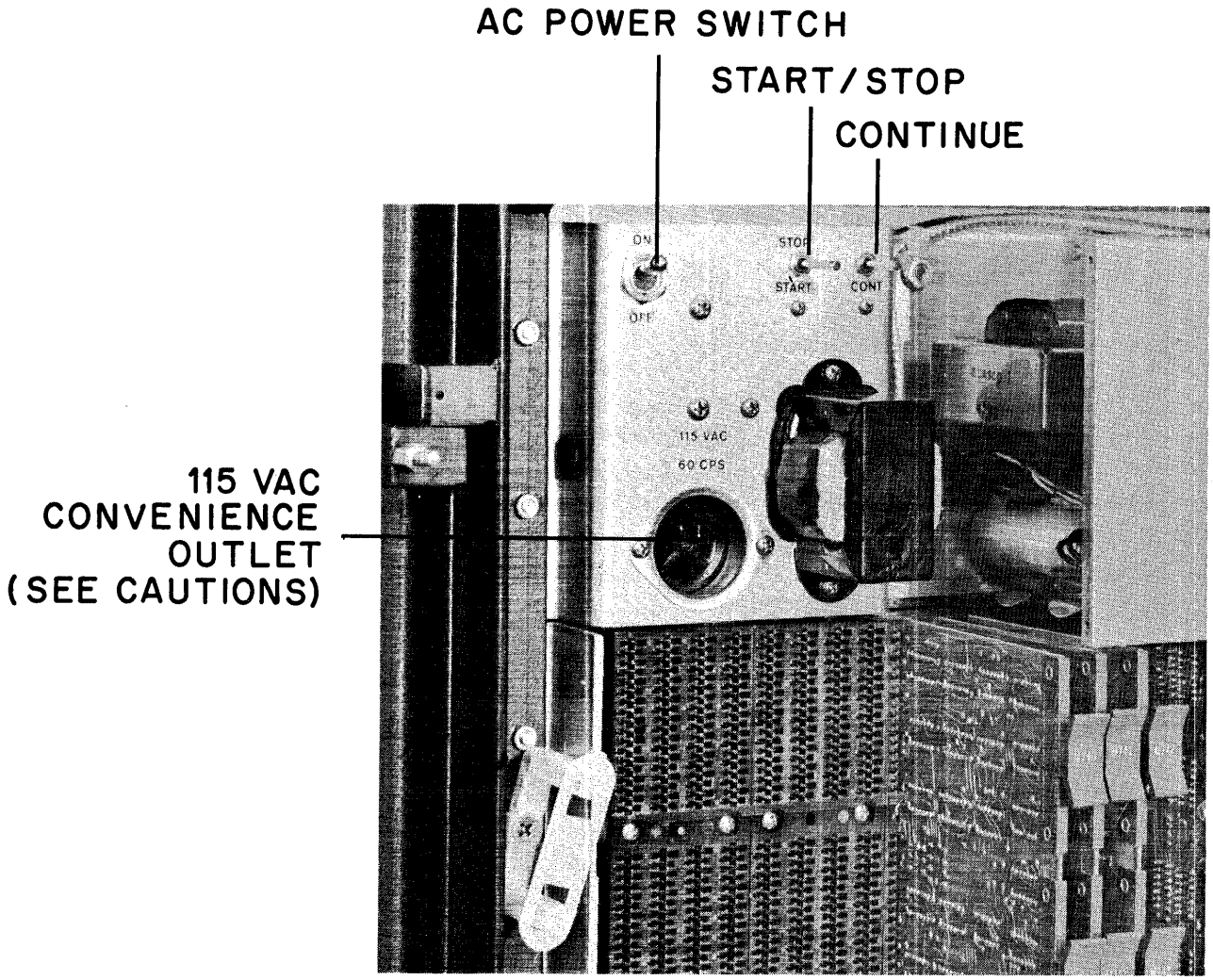


Figure 11-21 CU Control Panel

CAUTION

The STOP position of the START/STOP switch does not stop activity of the controlled device; it merely suspends control at an indefinite place in the sequence. Do not use it as an emergency stop switch.

One special use of the START/STOP switch is when you wish to activate the CU power supplies without starting the control operation. For example, for initial checkout using the computer programs, the CU must be on to activate its interface, but control operations should not be started. Figure 11-22 shows the use of the switches. Begin with the power switch OFF but ac supplied to the CU. Then, while holding the START/STOP switch in its STOP position, turn the power switch ON. The START/STOP switch may then be released. The low-voltage power supplies are now powered, but the control sequence is interrupted. Remember that the control operation will start if power is momentarily interrupted. Further use of the START/STOP and CONTINUE switches is reserved for programming and diagnostic purposes.

The PDP-14 system installation should now be complete and operating. This chapter has been able to deal only with a generalized installation scheme, since so many variations are possible. You may wish to design your system and then consult with a PDP-14 Applications Engineer before making all plans final.

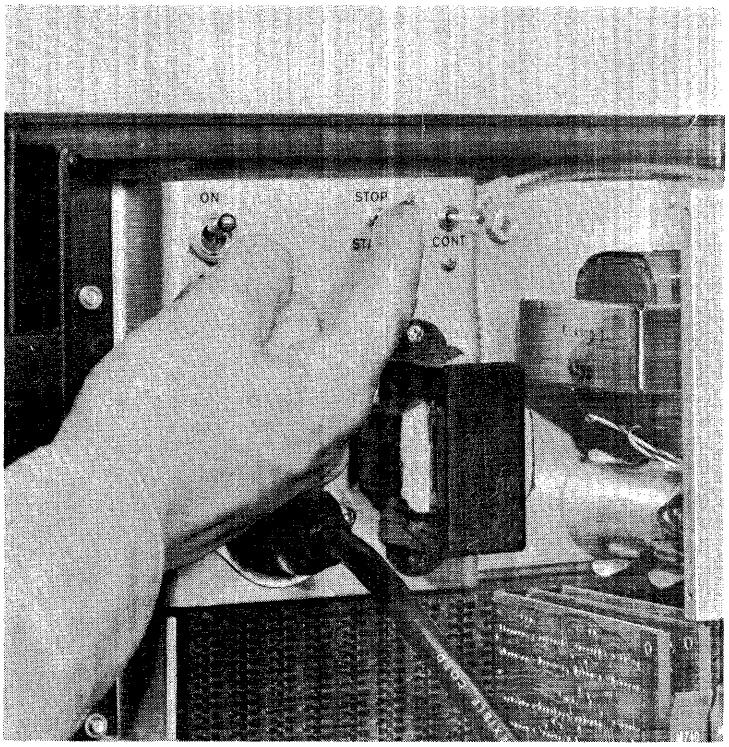


Figure 11-22 Special Startup Procedure

CHAPTER 12
MAINTENANCE CONCEPT

This chapter explains the general maintenance concepts applied when a device controlled by a PDP-14 fails to perform properly. Modifications to the ROM are described at the end of the chapter.

Only the general method of servicing PDP-14 contolled equipment is described herein, along with some of the diagnostic aids available. A detailed explanation of troubleshooting for the electrical serviceman and complete electronic circuit information for repair technicians, to service digital logic equipment, is located in the maintenance manual.

APPROACH

Every control system is unique; when failure occurs, it can be for countless reasons. Therefore, an orderly servicing scheme should be employed to reduce downtime. This section discusses some of the general approaches to servicing which may speed the job.

The first step in servicing equipment controlled by a PDP-14 system is to distinguish equipment failures in the controlled device from those in the control system. The equipment of the controlled device is always first to be suspected of malfunction. Equipment malfunction can be caused by mechanical problems, loss of power to circuits, stoppage of hydraulic or pneumatic actuators, and by other defective components. Input sensors, (such as limit switches) are especially prone to failure, since they are usually mechanical and subject to wear and corrosion.

If the controlled device is totally free of defect, the PDP-14 control system can then be checked. This testing is divided conveniently between the major elements of the system; the CU, the ROM, and the I- and O-box group.

Figure 12-1 explains the general sequence that testing should follow.

The maintenance procedure can be considered as an orderly sequence of the following three basic steps:

a. The mechanical and electrical operation of the controlled device is checked completely in order to isolate the controlled device faults from the control unit faults. The probability is high that electrical and mechanical sensors and actuators are responsible for the failure. Testing this equipment first will tend to reduce much needless search for defects in the control system.

The controlled device fault diagnosis is greatly aided by the indicator lamps in the I and O boxes and by the complete lists of machine control sequences formed in the PDP-14 programming operation.

b. If the controlled device is entirely free of defect, the CU and ROM are tested by automatic computer programs, which quickly pinpoint a circuit failure.

c. The Input and Output box system is last to be checked. Since all other equipment has now been tested, box error can be found by simple substitution of modular components.

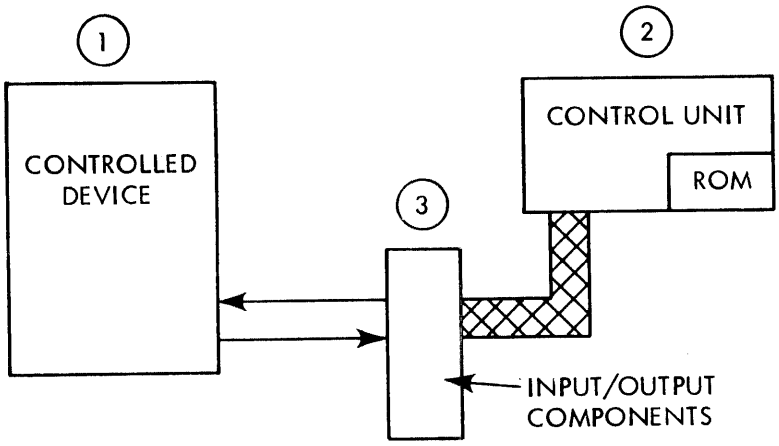


Figure 12-1 General Testing Sequence

The following discussion explains some of the general procedures followed for each of the three basic servicing sequences.

Determining controlled device malfunction

When a failure of a controlled device occurs, the servicing objectives are to quickly:

a. Define which operation did not occur,

b. Reduce the operation to a limited number of circuits to be tested,

c. Isolate the defective circuit,

d. Make the repair,

e. Re-check the operation for other errors.

Usually, trouble is apparent in a single output or small group of them. To switch one output on or off, a limited group of conditions is required. Certain inputs and other outputs (and possibly timers) must be on or off. This situation can be shown in the ladder diagram Figure 12-2.

Another program list would state that:

Y14 = SOL A
X5 = LS17
Y143 = 1 MOT
X37 = PB11

These program lists are an extremely effective diagnostic aid, since they show the control equations affecting every output circuit in the controlled device. The lists should always be kept convenient to the PDP-14 system for reference. They may be attached directly to the inside of the NEMA 12 door.

Usually, the problem of why a single output is not turning on or off, as required, is solved by checking the limited group of inputs and outputs specified by the equation which controls that output. The problem is manageable because it contains a limited number of items to be tested.

The state of each control equation element is determined by examining the neon lamps corresponding to each input or output circuit specified. These lamps are extremely convenient and accurate troubleshooting aids, since they indicate the actual AC power condition of all circuits.

Checking the CU and ROM

Diagnostic programs run on a PDP-8/I or PDP-8/L computer provide complete testing of all circuits in the CU. Similar programs for the ROM will verify that the memory system is working properly and that the contents of the ROM braid (the control instructions) are correct. Each diagnostic program is supplied as a paper tape. A complete explanation of the various tests performed and the meaning of their error messages is provided with the tape. These messages are printed on the computer Teletype and can then be correlated to specific circuit repair or module replacement. To operate such a program, a computer interface is first installed in the CU mainframe as described in Chapter 11. A computer is attached through control cables to the interface, the program is inserted in the computer, and the diagnostic is run. The total operation of the computer requires no more than several minutes, and checks every possible instruction in every circuit many times over. Because of this repetitive testing, the computer diagnostics effectively check the possibility of intermittents.

Checking the Input and Output Boxes

The I- and O-box components are last in the testing sequence because it should now be known that both the controlled device and the CU and ROM are operating properly. By elimination, the indication is that further malfunction must be in the I and O boxes or their cables. Since the failure of control has been isolated earlier to a single control equation or group of them, only a very few box modules are suspect. These modules are simply substituted, one at a time, until the defect is eliminated. Substitute modules should be stocked so that they may be available for quick replacement.

Summary of Maintenance Approach

The general maintenance approach to PDP-14 controlled equipment implies the following steps:

a. The controlled device is tested, and is assumed to be the cause of error unless demonstrated otherwise.

b. The CU and ROM are checked, and faulty circuit operation serviced by module replacement.

c. Finally, the I and O box system is serviced by simple module and cable substitution.

ROM MODIFICATIONS

Industrial control requirements can change. These changes are often limited and may be due to errors or ommissions in the system design. When the entire control operation is to be redefined, it is probably easiest to program for a completely new ROM braid and insert it into the CU. However, when lesser changes must be made, the ROM program can be modified by manually placing wires corresponding to the new program. These wires selectively replace the wires which are permanently sealed in an ROM braid assembly.

Since the manual ROM reprogramming process is relatively slow, it is recommended only when a maximum of 10 percent of the total instructions (approximately 120) must be changed. If necessary, all instructions can be manually changed.

The ROM change operation has two general stages:

a. First, the new program, including changes, is determined and the program is converted to a series of binary numbers.

b. Wires are then hand-positioned in the ROM assembly according to the pattern determined by the binary numbers.

Programming for ROM Changes

A specific program instruction is described by two numbers; it has a numbered position (a location or address) and also the number code which is the actual instruction (the contents of the location). An example would be a stack of parts boxes numbered consecutively from top to bottom, with a specific number of parts in each box.

Each box could be described by its number (giving its location or address in the stack) and by the quantity of parts inside (its contents). A program list is nothing more than all the locations in consecutive order, and their contents (the instruction commands themselves).

ROM changes which alter locations in the middle of a braid must occupy the same number of locations as the old program, or less. For example, if a TXN 26 instruction is an error, and should be a TXF 6, both instructions

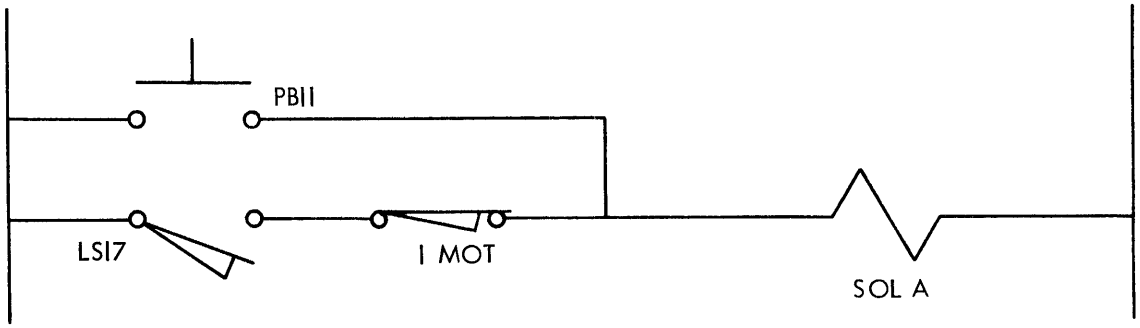


Figure 12-2 Ladder Diagram Showing Output Controlled and Inputs and Output Tested

In this example, SOL A is the output controlled, and PB11, LS17 and 1 MOT are the inputs and output tested. One "rung" of the ladder diagram is equivalent to one control equation in the PDP-14 system.

The PDP-14 control system provides a ready-made documentation of every operation which is to occur in the controlled device. Detailed program lists are produced when the ROM program is designed. Like the ladder diagram, these lists represent control equations in a symbolic form. For example, a control designer might begin his programming by thinking as follows:

"Motion of a table to the left requires that the table be at the right, and that a certain motor be on, or that a manual pushbutton be depressed".

He would translate this statement into terms of actual sensor and actuator devices:

SOL A = LS17 AND 1 MOT
OR
PB11

where, SOL A controls motion of the table to the left;
LS17 senses that the table is in position at the right;
1 MOT is the motor circuit supplying hydraulic power to the table (through SOL A); and
PB11 is the manual operator pushbutton

The PDP-14 program lists later would show all these inputs and outputs in a symbolic relationship, for example:
Y14 X5 (AND) Y143 (OR) X 37

use one memory location, and its contents are simply changed to correct the error. But if a two-location instruction, such as a JMP, or a new one-location instruction must be added to the previous program, room must be made for the added location required. In other words, a NOP or other unused instruction in a nearby location must be deleted. Conversely, a NOP can be inserted in a program change which deletes instructions and leaves an unused location.

Each of the four possible ROM banks in the Control unit has "1K" or 1024 locations, numbered (in octal) from 0 to 1777. The first three digits identify a ROM address wire (0 through 177), and the last digit gives a specific row of transformer cores (0 through 7). The ROM reads a specific address by selecting and pulsing one wire and selecting one transformer row. For example (see Figure 12-3):

This allows you to determine the position of an instruction physically in the ROM from its location number.

LOCATION	1346
is	
WIRE NO.	134
and	
ROW	6

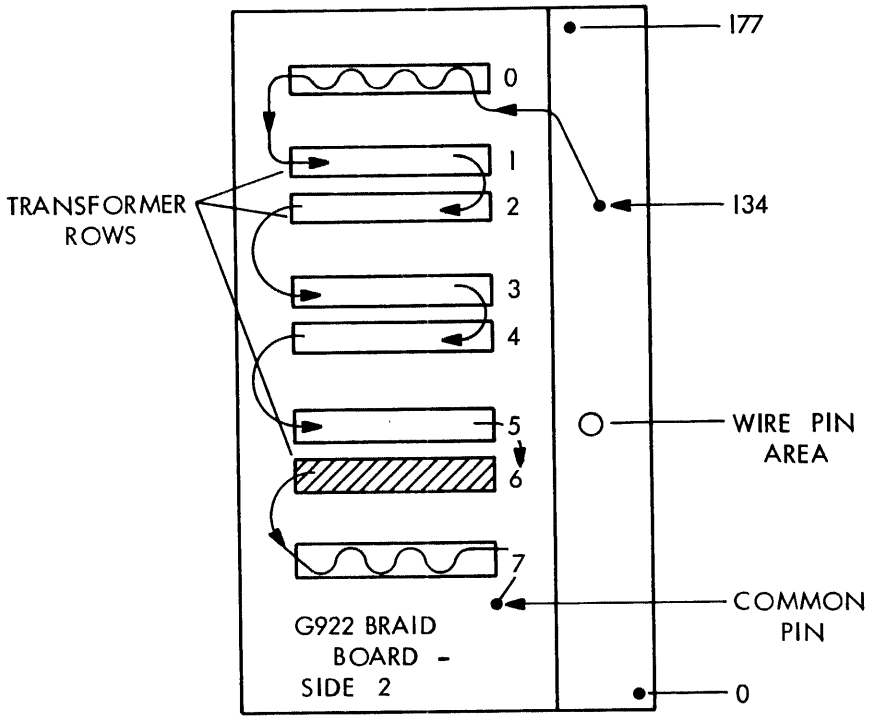


Figure 12-3 Wire Placement in ROM Transformer Rows

Each ROM wire runs through all eight transformer rows and makes available eight instructions each time it is pulsed. Only one transformer row is selected to be read at a time, and thus only one instruction. In the previous example, wire 134 is pulsed and the output from row 6 is used as the instruction.

As you can see, each time an instruction is changed by replacing a wire, all eight instructions must be replaced. Therefore, the program list should contain locations in the group (both those changed and others unchanged). Here is an example:

LOCATION			
WIRE NO.	ROW NO.	INSTRUCTION	OCTAL INSTRUCTION CODE
062	0	TXF 6	2006
	1	TXF 14	2014
	2	JFF 76	5076
	3	TXN 7	2407
	4	TXN 154	2554
	5	JFN 77	5477
	6	SYN 6	3406
	7	JMP	4224

This list should include all program changes.

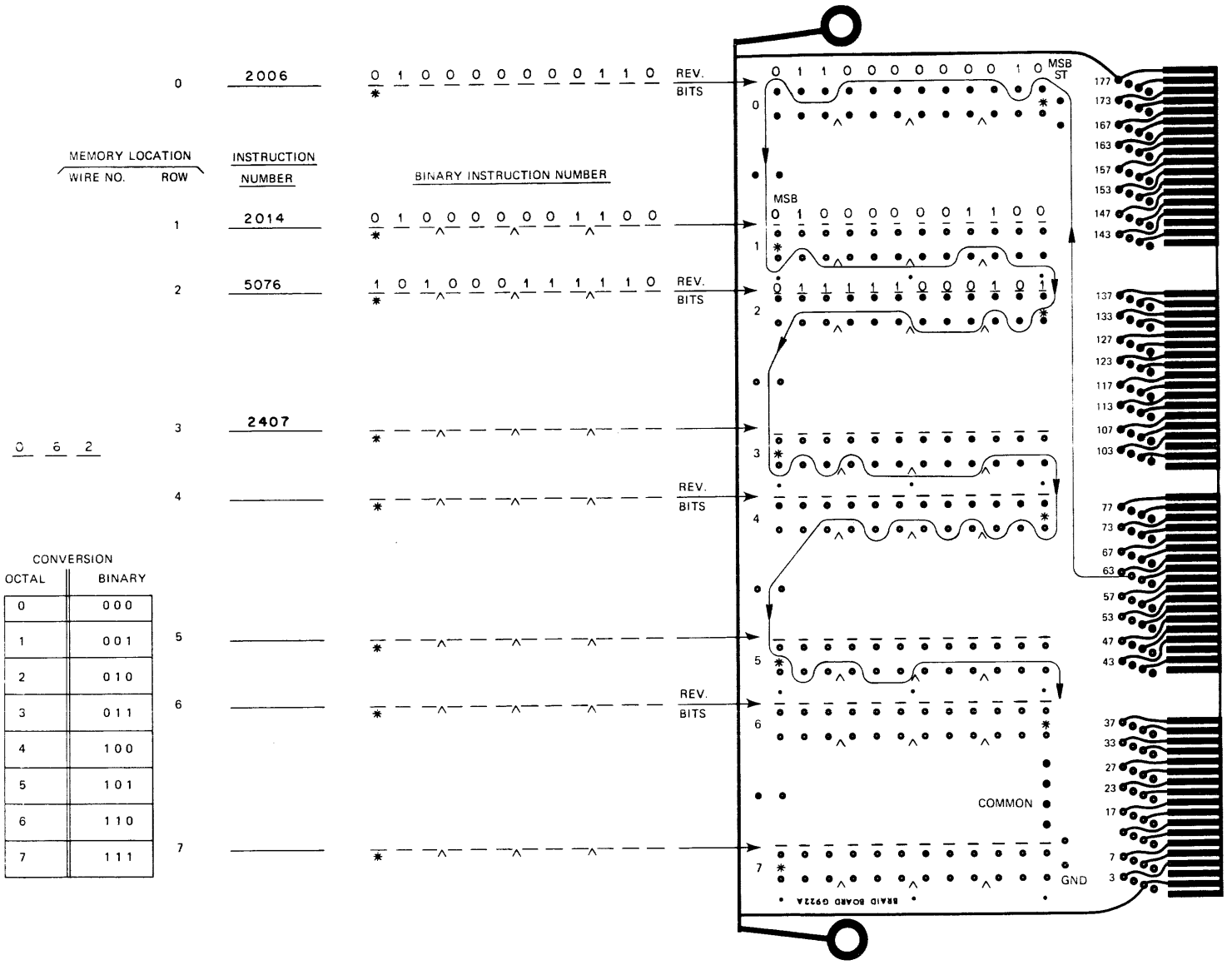
A two-location instruction uses two transformer rows. In the above case, the second location of the JMP (which specifies the JMP address) is in the next consecutive memory location, 063 and is therefore on the next wire. If this instruction were changed, two wires would be affected.

The next programming step is to convert the octal instruction code to a binary number. Refer to the left side of the Figure 12-4 "ROM Change Sheet". The example shows the octal instruction numbers for wire 062. Each octal digit is converted to three binary digits, as given in the conversion table (Figure 12-4). Convert the four octal digits, one at a time. For example, in location 622, the instruction 5076 is:

```
      5 =  101
      0 =      000
      7 =          111
      6 =              110
or,       101 000 111 110
```

For ease in handling, the binary number is written in four groups of three digits each, corresponding to the previous octal numbers.

The binary numbers can now be written directly on the braid board outline on the right of the sheet, as a reference when adding the new wire. The electrical layout of the ROM transformers reverses the binary

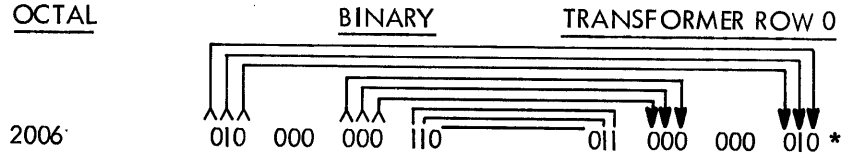


CONVERSION

OCTAL	BINARY
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Figure 12-4 ROM Change Sheet - Example

bits in each alternate row. The asterisk (*) indicates the beginning of the binary code. For example, in location 620, we have:



In location 621, the binary code is simply transferred to the braid board just as read. But in location 622 (and all even-numbered locations), the order of bits is reversed, end-to-end, as in the example. The "Rev Bits" remarks (Figure 12-4) remind you to make this reversal when transferring the number to the board drawing. Now complete the chart.

When all binary numbers are on the board drawing of Figure 12-4, the entire process should be rechecked. This completes the programming section of the ROM changes operation.

ROM Wire Additions

The clue to ROM wire additions is that a binary 1 (as written on the drawing) corresponds to a wire placed through the center of an individual transformer core, and a binary 0 to a wire around the outside of the core. It makes no difference to which side of the core the wire runs to form a 0. See Figure 12-5 for example.

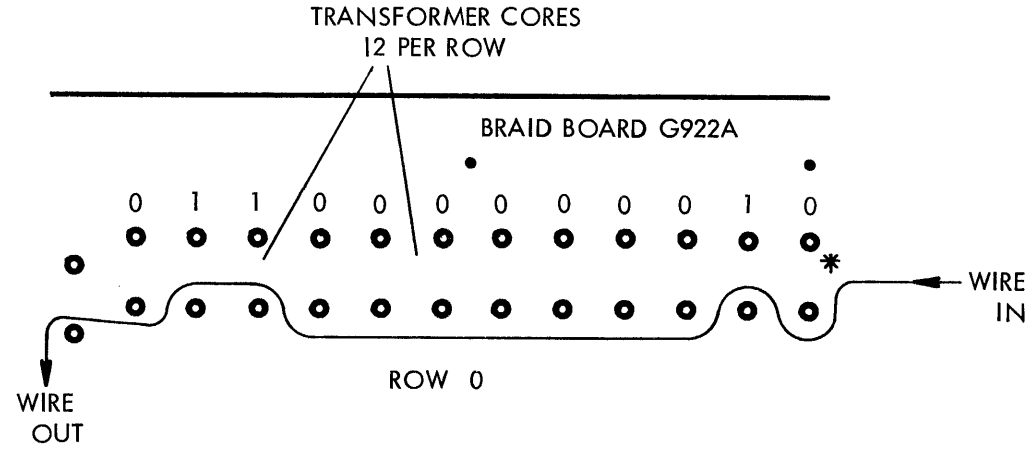


Figure 12-5 ROM Wire Placement Example For One Transformer ROW

Preparing the ROM Braid Board

Before removing a ROM bank from the CU, prepare an area which is free from dirt, metal particles and other contaminants. A clean, flat table should be provided, covered with soft, lint-free cloth or paper. Good lighting and an ac receptacle for a small soldering iron will also be required.

Locate the proper ROM bank in the CU by its "M" or bank number printed on a sticker on the G922 keeper plate.

The ROM consists of a double board assembly, G922 and G923, and a separate G924 Sense Board, as described in Chapter 11. Remove only the G922-923 assembly from the Control Unit. The boards are separated from each other to expose the wire changes side of the G922 "Braid Board".

WARNING

Transformer cores which will be exposed in this operation are fragile. They are formed of ferrite, or powdered iron, and can be chipped or broken if abused. Never force cores in any direction, and avoid resting assemblies on their core surfaces. Do not touch cores with any hard object, or with oil or other contaminants. DO NOT DROP ROM ASSEMBLIES.

Place the double assembly down with its aluminum keeper plate on the bottom. Remove the fifteen screws holding the two boards together and lift the G923 straight up and off the G922. Do not bend the transformer cores. Lay the G923 down in a protected place, with its cores facing up. Do not disturb the screws holding the aluminum keeper plate to the G922. The surface now exposed on the table is the "G922 Braid Board, Side 2". Place this assembly to the left of Figure 12-4, with its pins on the right side.

The example shows wire 062 installed. First, cut 5 feet of no. 36 enameled wire. Tin one end, and solder it to its proper pin, as shown in Figure 12-6. Use a small, clean, pencil-type iron of not more than 40 watts, and 60/40 rosin core solder.

In our Figure 12-6, pin 62 is not marked on the board, and is found by counting from the last marked pin (57) from right to left, bottom to top. Remember to skip 8 and 9 when counting octal numbers. The pins are: 57 - 60 - 61 - 62. Bring the wire to the ST (start) position in the top right row and weave it through the first row of cores (row 0) as shown. Lay the wire at the bottom of each plastic sleeve, and do not pull it tight. Continue through all eight rows, alternating entry of the wire onto the rows, until the wire is at the END position at bottom right. Then, clip the excess wire, tin the end, and solder it to any one of the four COMMON pins. This completes the wire addition.

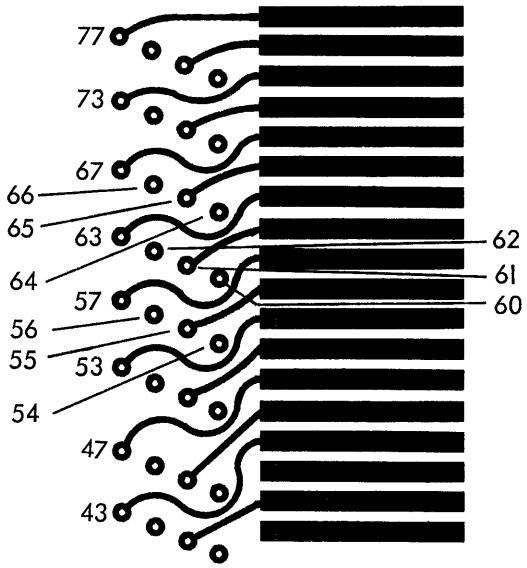


Figure 12-6 Pin Location Example - Pin 62

Removing the Old Braid Wire

Locate the wire pin for the wire just replaced. This pin extends through the braid board to the braid on the under side. One braid wire is connected to the pin at this point and should be clipped off with small wire cutters. Bend the wire back gently so that it cannot touch other wire pins, and insulate it with tape. Be very careful not to break the other fine braid wires.

Continue making wire changes until all new wires are installed and old ones clipped.

Replacing ROM Braid Board

The G923 board should now be slid gently in place on the G922 and screwed in place. The board package is shown on edge in Figure 12-6 to illustrate the arrangement of parts. The three screws in the center of the board are tightened first. Work toward the outside of the board, tightening by hand pressure only. Avoid exerting extreme pressure. The board assembly can now be installed in the Control Unit to complete the ROM modification operation.

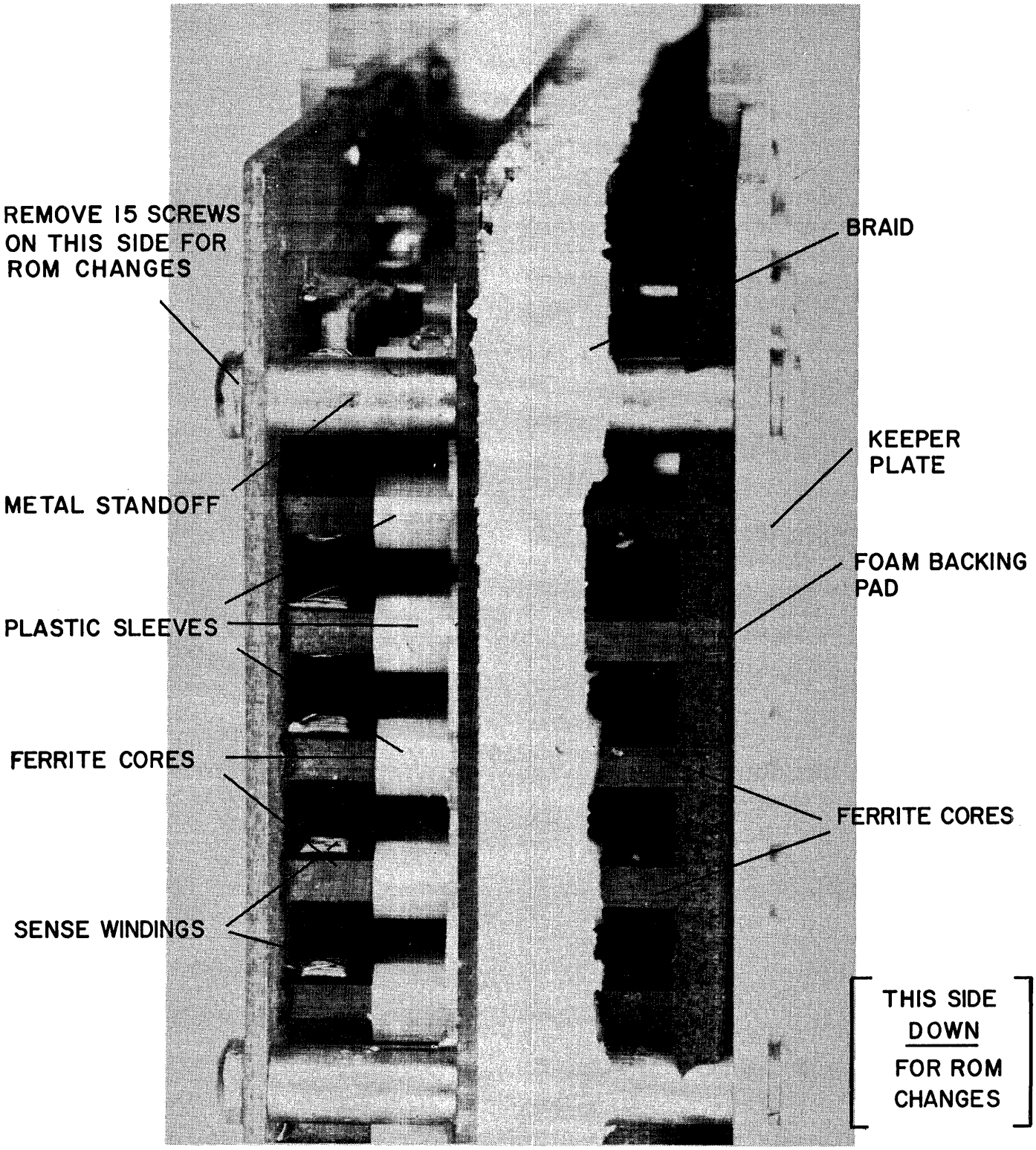


Figure 12-7 ROM Board Assembly - G922-G923 (Edge View)

INPUT NUMBERS

FOR EIGHT INPUT BOXES

Box	Box Terminal Number																															Box
↓	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	↓
A	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	A
B	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77	B
C	100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117	120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137	C
D	140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157	160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177	D
E	200	201	202	203	204	205	206	207	210	211	212	213	214	215	216	217	220	221	222	223	224	225	226	227	230	231	232	233	234	235	236	237	E
F	240	241	242	243	244	245	246	247	250	251	252	253	254	255	256	257	260	261	262	263	264	265	266	267	270	271	272	273	274	275	276	277	F
G	300	301	302	303	304	305	306	307	310	311	312	313	314	315	316	317	320	321	322	323	324	325	326	327	330	331	332	333	334	335	336	337	G
H	340	341	342	343	344	345	346	347	350	351	352	353	354	355	356	357	360	361	362	363	364	365	366	367	370	371	372	373	374	375	376	377	H
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
Box	Box Terminal Number																															Box

Note: No "8" or "9" digits appear in program numbers

OUTPUT NUMBERS

FOR SIXTEEN OUTPUT BOXES

Box	Box Terminal Number																Box
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	A
B	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	B
C	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57	C
D	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77	D
E	100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117	E
F	120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137	F
G	140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157	G
H	160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177	H
J	200	201	202	203	204	205	206	207	210	211	212	213	214	215	216	217	J
K	220	221	222	223	224	225	226	227	230	231	232	233	234	235	236	237	K
L	240	241	242	243	244	245	246	247	250	251	252	253	254	255	256	257	L
M	260	261	262	263	264	265	266	267	270	271	272	273	274	275	276	277	M
N	300	301	302	303	304	305	306	307	310	311	312	313	314	315	316	317	N
P	320	321	322	323	324	325	326	327	330	331	332	333	334	335	336	337	P
R	340	341	342	343	344	345	346	347	350	351	352	353	354	355	356	357	R
S	360	361	362	363	364	365	366	367	370	371	372	373	374	375	376	*	S
	1	2	3	4	5	6	7	8	†9	10	†11	12	†13	14	†15	16	
Box																	Box

Box Terminal Number

Notes: 1) No "8" or "9" Digits appear in program numbers
2) No boxes lettered "I", "O", or "Q" are used, to avoid confusion with numbers
* Special output-unused for O boxes.

PDP-14 INPUT ASSIGNMENT SHEET -P1

INPUT BOX _____ (A THROUGH H)

INPUT NUMBER	ASSIGNMENT	NORM COND.	PROGRAM NUMBER
1			X
2			X
3			X
4			X
5			X
6			X
7			X
8			X
9			X
10			X
11			X
12			X
13			X
14			X
15			X
16			X

PDP-14 INPUT ASSIGNMENT SHEET -P2

INPUT BOX _____ (A THROUGH H)

INPUT NUMBER	ASSIGNMENT	NORM COND.	PROGRAM NUMBER
17			X
18			X
19			X
20			X
21			X
22			X
23			X
24			X
25			X
26			X
27			X
28			X
29			X
30			X
31			X
32			X

PDP-14 OUTPUT ASSIGNMENT SHEET

OUTPUT BOX _____ (A THROUGH S EXCLUDING I, O & Q)

OUTPUT NUMBER	ASSIGNMENT	PROGRAM NUMBER
1		Y
2		Y
3		Y
4		Y
5		Y
6		Y
7		Y
8		Y
9		Y
10		Y
11		Y
12		Y
13		Y
14		Y
15		Y
16		Y

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback: your critical evaluation of this document. Please give specific page and line references when appropriate.

ERRORS NOTED IN THIS PUBLICATION: _____

_____

_____

_____

_____

_____

_____

SUGGESTIONS FOR IMPROVEMENT OF THIS PUBLICATION: _____

_____

_____

_____

_____

_____

_____

_____

_____

Name _____ Date _____

Organization _____

Please describe your position _____

Street _____

City _____ State _____ Zip Code _____

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Digital Equipment Corporation
PDP-14 Marketing Manager
146 Main Street
Maynard, Massachusetts 01754

Digital Equipment Corporation
Maynard, Massachusetts

digital

printed in U.S.A.