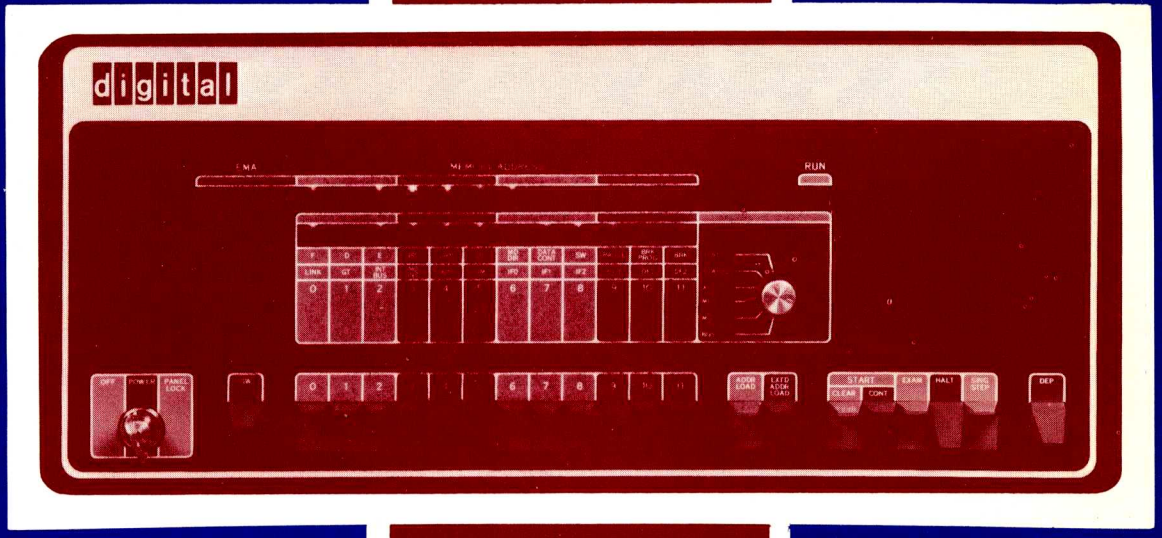
# SOFTWARE MANUAL

digital




INDUSTRIAL 14 FAMILY

14

OF CONTROLLERS

DEC-14-ISUMA-B-D

# INDUSTRIAL 14
# SOFTWARE MANUAL

digital equipment corporation • maynard, massachusetts

1st Edition, July 1974



Printed in U.S.A.

## CONTENTS

CONTENTS (Cont)

## CONTENTS (Cont)

## ILLUSTRATIONS

## ILLUSTRATIONS (Cont)

## ILLUSTRATIONS (Cont)



## TABLES

## TABLES (Cont)

# CHAPTER 1
# INTRODUCTION TO INDUSTRIAL 14 PROGRAMMING

Anyone who has designed machine controls has actually functioned as a programmer. Although relays are used rather than a computer, the designer must specify a series of events (in a certain sequence) to control machine operations. In a general sense, this specification of a sequence of events is equivalent to performing a programming task in conjunction with a computer.

## PROGRAMMING THE INDUSTRIAL 14

The *Industrial 14 System* has been specifically designed for controlling machines and processes. The programmer specifies only the control function to be performed in a particular application; this function can be specified in relay symbology and entered via the VT14 Programming Terminal or, as described in this manual, specified in equations or instructions and entered via a digital computer. In either instance, the control engineer is specifying the content of the Industrial 14's memory rather than directing the actions of an electrician in wiring a relay panel. Once the control function is specified, the Industrial 14 controls a machine (or a series of machines) in the same manner as relays in the same application.

Distinct advantages are evident in programming machine control with an Industrial 14 compared with "programming" with relays. For example, logic errors are easier to find and to correct; system changes are accommodated painlessly. Programming the Industrial 14 does not require previous computer experience although experience in machine control is necessary.

Industrial 14 programs provide relationships between input devices (limit switches, pushbuttons, selector switches, etc.) and output devices (solenoids, motor contactors, indicator lights, etc.) and devices associated with internal functions (counters, retentive memories, timers and shift registers). These relationships, or control functions, can be expressed as Boolean equations, circuit diagrams or machine language instructions, which, when solved for particular input values, specify the state (ON or OFF) of an output or internal function.

## INPUT/OUTPUT AND INTERNAL FUNCTION ORGANIZATION

Machine inputs and outputs must be assigned to the Industrial 14 input (I) and output (O) converters before an Industrial 14 control program can be written; internal functions must be assigned within the Industrial 14. These assignments permit the Industrial 14 instructions to test the state of specific inputs, outputs, and internal functions.

Technical descriptions of the I/O converters are contained in the *Industrial 14 Systems Manual* (DEC-14-HSMAA-A-D). A maximum of 32 input panels are available, each panel containing 16 input converters, allowing a maximum of 512 Industrial 14 inputs. Sixteen possible output panels are available; each panel containing 16 output converters, for a total of 256 available outputs.

A total of 256 internal functions are available within the Industrial 14 Controller. These functions retain the current state (ON or OFF) even if primary power to the Industrial 14 fails.

### Industrial 14 Inputs

The inputs to the Industrial 14 System are limit switches, selector switches, pushbutton switches, etc. Each input can have only one of two values at any one time, ON or OFF. These machine inputs are wired to an input converter. An input is ON if 115 Vac is present at the specific input terminal. The input is OFF to show the absence of 115 Vac. For programming purposes, all Industrial 14 inputs are denoted by numbers 0–777, designated in octal (base 8) representation. Specifically, the number 15 designates a machine input (e.g., a limit switch) which is wired to terminal 14 on the first input panel. Program inputs are numbered octally; 8s and 9s never appear in the number system.

The normal procedure for assigning input numbers is to complete an *Input Assignment Sheet,* as shown in Figure 1-1. This sheet is completed for each input panel used in

the Industrial 14 system. The Input Assignment Sheet for one panel records the following:

a. Machine input number (0–777)

b. LS201 (Limit Switch 201). (This will be a familiar symbol to the user.)

c. The function of this input (activated when slide 2 is in the full return position).

d. Normal condition of the contacts wired to the Industrial 14 input converter (normally open or normally closed).

**Industrial 14 Outputs**

Output devices controlled by the Industrial 14 System are solenoids, motor contactors, small motors, indicators, signal devices, etc. These outputs are placed ON or OFF by the Industrial 14, depending on the current state of inputs or outputs. For programming purposes, these 256 Industrial 14 outputs are designated by octal numbers 1000–1377 inclusive.

The procedure for assigning output numbers is similar to that used for assigning input numbers. A sample *Output Assignment Sheet* is shown in Figure 1-2. This sheet should be completed for each output panel in the Industrial 14 System. The machine output device (solenoid, motor contactor, etc.) wired to each output converter terminal is recorded along with the following information:

a. I/O Number (1000–1377)

b. Symbol (SOLA for Solenoid A)

c. Function (clamps part at beginning of cycle)

INPUT ASSIGNMENT SHEET

| I/O Number | Symbol | Function | Normal Condition |
|---|---|---|---|
| 0 | LS201 | ACTIVATED WHEN SLIDE 2 AT FULL RETURN POSITION | n/o |
| 1 | LS202 | ACTIVATED WHEN SLIDE 2 AT FULL DEPTH | n/o |
| | | | |

INPUT ASSIGNMENT SHEET

| I/O Number | Symbol | Function | Normal Condition |
|---|---|---|---|
| 20 | PB5 | CYCLE START PUSH BUTTON | n/o |
| 21 | PB12 | START MOTORS PUSH BUTTON | n/o |
| | | | |

Figure 1-1 Input Assignment Sheets

OUTPUT ASSIGNMENT SHEET

| I/O Number | Symbol | Function |
|---|---|---|
| 1120 | SOLA | CLAMPS PART AT BEGINNING OF CYCLE |
| 1121 | SOLB | RETURNS HEAD, ACTIVATED BY LS12 |
| 1122 | SOLF | UNCLAMPS PART WHEN LS4 IS TRIPPED |
| | | |

Figure 1-2 Output Assignment Sheet

**Internal Functions**

A similar assignment procedure is followed for internal functions as described in Chapter 3.

## BOOLEAN REPRESENTATIONS OF MACHINE CONTROL

Programming the Industrial 14 System requires familiarity with simple Boolean equations for control functions. These equations comprise operators and variables; variables of Industrial 14 control equations are inputs, outputs, and internal functions. Each variable has only two states: ON and OFF. The operators in these equations are as follows:

The asterisk(*) denotes the logical AND function
The plus sign (+) denotes the logical OR function
The slant sign (/) denotes the logical NOT function.

For example, the equation

1010 = 23+21*1007

is read "output 1010 is set ON when input 23 is ON, or when both input 21 and output 1007 are ON". This equation instructs the Industrial 14 to test input 23, and if it is ON, to set output 1010 ON; if input 23 is OFF, to test input 21 and output 1007 and if these are both ON, to set output 1010 ON. If neither set of conditions is satisfied, output 1010 is set OFF. This equation could be represented by the ladder diagram shown in Figure 1-3 and the following circuit:

SOLF = 2LS+3LS*7CR

SOLF corresponds to output 1010; 2LS corresponds to input 23; 7CR corresponds to output 1007; and 3LS corresponds to input 21. It is important to remember that input and output numbers are determined by the terminals to which these devices are wired.

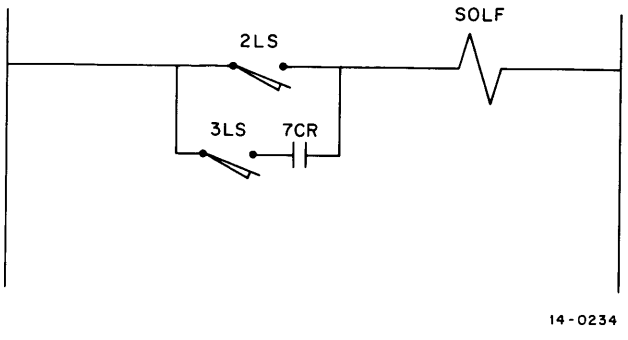



Figure 1-3 Ladder Diagram

**Equations Containing NOT**

The equation operator / (NOT) is often used in control functions.

For example, the equation

1027 = 5* / 30

could represent the control function "solenoid A is energized if limit switch 5 is tripped and pushbutton 12 is not activated". This function is equivalent to the following equation and the ladder diagram in Figure 1-4.

SOLA = 5LS* / 12PB

where SOLA corresponds to output 1027; 5LS corresponds to input 5; and 12PB corresponds to input 30.

**NOTE**

**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.**

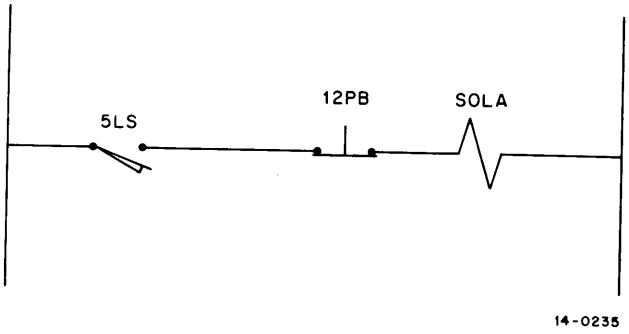



Figure 1-4 Ladder Diagram Illustrating NOT Function

The NOT sign in the Industrial 14 System relates to the absence of 115 Vac at an input or output; it does not specify physical switch positions. Normally open or normally closed contacts can be wired to the Industrial 14, depending on the user's choice. This is possible because the Industrial 14 can test an input for the ON state or the OFF state; therefore, only one set of contacts need be wired to the Industrial 14.

The choice of a normal condition for input contacts must be made before writing the control equations for the Industrial 14 and the normal condition of the contacts should be noted on the Input Assignment Sheet. When

normally open contacts are used, the equation 1005 = 15 specifies that output 1005 is set ON when machine input 15 is activated (e.g., a limit switch is tripped) and 115 Vac is applied. It also specifies that output 1005 is 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 Industrial 14, the equation 1005 =/15 specifies that output 1005 is set OFF when input 15 is activated and is set ON when input 15 is not activated.

Because normally closed contacts result in equations which appear contrary to common sense, it is suggested that all normally open contacts be used in the Industrial 14 system. When considerations such as safety dictate that a normally closed contact must be used, simply reverse the logic; therefore, 10 would be represented as /10 and /10 would be 10.

**Evaluating Equations**

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

1001 = 1*2+3

Does this equation represent 1001 = (1*2) +3 or 1001 = 1* (2+3)?

It is obvious from the two ladder diagrams in Figure 1-5 that the two interpretations are not equivalent.

To eliminate possible ambiguities when evaluating equations, variables grouped by the AND (*) operator are *always* combined before variables grouped by the OR (+) operator (when the order of combination is not clearly indicated by parentheses). In the preceding example, (1*2) +3 (the second ladder diagram) is the correct interpretation of the expression 1*2+3.

The following examples show the use of parentheses:

27+/17*20 = 27+(/17*20)
11*23+31*14 = (11*23) + (31*14)
(12+15) */21+22 = (12+15)*/21+22
21+22*23+24*25 = 21+ (22*23) + (24*25)

Parentheses can 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 or O converter.

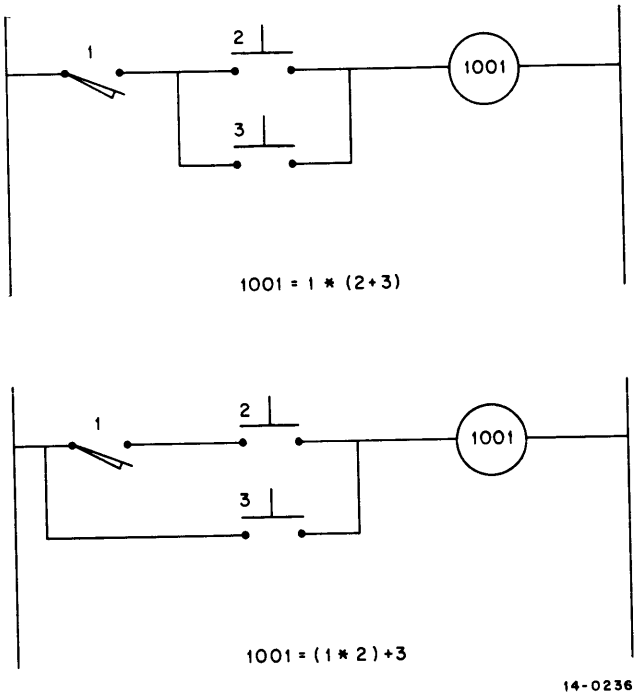



Figure 1-5 Evaluating Equations

**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:

1021 = /(121*26)

The states of inputs 121 and 26 are tested and the resultant state of the quantity in parentheses is inverted by the NOT operator to determine the value of output 1021. The following truth table summarizes the function:

| 121 | 26 | 121*26 | 1021=/(121*26) |
|---|---|---|---|
| ON | ON | ON | OFF |
| ON | OFF | OFF | ON |
| OFF | ON | OFF | ON |
| OFF | OFF | OFF | ON |

**NOTE**

**Output 1021 is ON if either 121 or 26 is OFF. The following equation expresses this relationship:**

**1021 = /121+/26**

Thus, the following substitution can always be made in control equations:

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

A similar substitution is possible for the following equation:

1022 = /(107+12)

The following truth table determines the state of output 1022 for all states of inputs.

| 107 | 12 | (107+12) | 1022=/(107+12) |
|---|---|---|---|
| ON | ON | ON | OFF |
| ON | OFF | ON | OFF |
| OFF | ON | ON | OFF |
| OFF | OFF | OFF | ON |

Restating this relationship, output 1022 is ON only if input 107 and input 12 are both OFF. Thus, the equation can be rewritten as

1022 = /107* /12

and the following substitution is always possible:

/(A+B) can always be replaced by /A*/B

**Equation Simplifications**

Several methods exist to simplify equations before they are programmed for the Industrial 14. Equations need not be in the simplest form; in many instances memory locations can be conserved and executing time for the program can be lessened.

Consider the following equation:

1005 = 1+(1*2) + (3*5)

The expression 1*2 adds no meaning to the equation. If 1 is ON, output 1005 will be set ON, regardless of the state of 2; if 1 is OFF, the state of 2 can in no way affect 1005. Thus, the equation can be simplified to:

1005 = 1+(3*5)

and the following rule is established:

A+(A*B) = A

The following equation can also be simplified:

1006 = (1*2+3) *4*/3

Because of the last element of the equation (/3), output 1006 can never turn ON when 3 is ON. Thus, the last element within the parentheses (+3) is meaningless (3 cannot be ON and OFF at the same time). The simplified equation is

1006 = 1*2*4*/3

and the following rules are established:

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

## INDUSTRIAL 14 PROGRAM DESCRIPTION

After recording the input, output and internal assignments and their functions, the programmer codes the program for all control functions. In general, an Industrial 14 control program consists of separate instruction groups. Each instruction group solves a Boolean equation by testing Industrial 14 inputs, outputs and internal functions; at its conclusion an output is set either ON or OFF. The final instruction group in the program ends with an unconditional jump to the start of the first instruction group; therefore, the Industrial 14 program is a closed loop.

For example, if a machine control requires 20 outputs, 20 instruction groups must appear in the control program. Figure 1-6 illustrates the execution of an Industrial 14 program.

The instruction groups are often separated by several no operation instructions (NOPs). The Industrial 14 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 important. If there was no instruction to return control to the beginning of the program, the Industrial 14 would stop executing instructions when it reached the instruction at the end of its memory and there would be no way for the program to automatically start again.

Each instruction group is *independent* of all other instruction groups. For example, in the same program, instruction groups can be controlling one device while other instruction groups are controlling a second device. The sharing of an

Industrial 14 System between many devices can be extended until any of the following limits are reached:

Inputs (512)
Outputs (256)
Internal functions (256)
Memory locations (4000 or 8000)

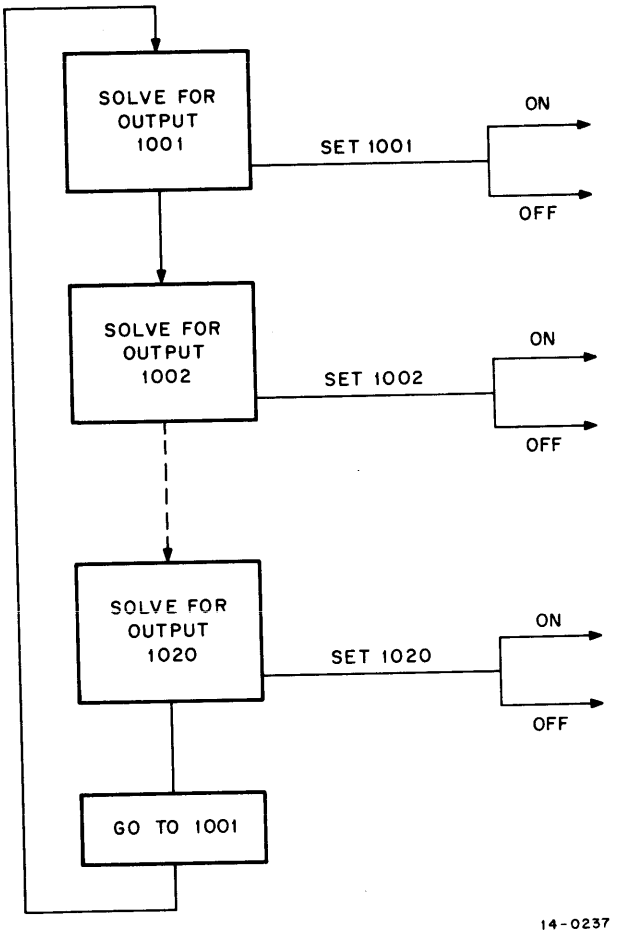



Figure 1-6 Industrial 14 Program Execution Cycle

## INTRODUCTION TO THE INDUSTRIAL 14 SOFTWARE

Two PDP-8 based software kits are available for the Industrial 14: one includes programs that operate on a PDP-8 paper tape system, the other includes programs that operate on an OS/8 (Operating System on the PDP-8) System. The *Industrial 14 Paper Tape Software Kit* (option number QLR08-AB) contains binary tapes for the following programs:

**BOOL-143** – a compiler which translates special Boolean equations into Industrial 14 machine instructions. This program is discussed in Chapter 4.

**SET-143** – an optional Symbolic Equation Translator (preprocessor) for BOOL-143. This program translates 6-character alphanumeric symbols into associated pre-defined Industrial 14 I/O numbers (0–1777) for processing by BOOL-143. Therefore, all Boolean equations may be written with symbols that enable the user to describe physical input and output components and internal function types (i.e., SRTPB1, CLPSOL, and DWLTMR). This program is discussed in Chapter 5.

**PAL-143** – the assembler for the Program Assembly Language of the Industrial 14. An assembly language is a set of instructions in mnemonic or symbolic form (letters) which have direct counterparts in the machine code of the Industrial 14. In other words, the PAL-143 assembler translates the symbolic representations into Industrial 14 machine code. This program is discussed in Chapter 6.

**ODP-143** – a debugging program for Industrial 14 machine language instructions stored in the Industrial 14's memory. The user may examine and change instructions when the Industrial 14 program is halted, or examine and set input and output states during program execution. This program is discussed in Chapter 7.

The *Industrial 14 OS/8 Software Kit* (option number QLR01) contains binary paper tapes of OS/8 versions of BOOL-143 (BOOL-143/OS8), SET-143 (SET-143/OS8), PAL-143 (PAL-143/OS8), and ODP-143 (ODP-143/OS8). These programs are discussed in Chapters 4, 5, 6 and 7 respectively. Loading and operation of these OS/8 versions are discussed in Chapter 8.

Before the system software can be understood, the reader must gain an understanding of the basic Industrial 14 instruction set and its operation. The instructions used to solve equations are described in Chapter 2; other Industrial 14 and PDP-8 instructions used by an external computer to monitor an Industrial 14 are described in Chapter 9. Chapter 8 describes operating procedures for the PDP-8 loaders and introduces a Text Editor program which is useful for Industrial 14 programming.

The control engineer who is using the computer simply to program the Industrial 14 will find BOOL-143 to be a practical approach. He can write Industrial 14 programs in equation form and have BOOL-143 generate the machine code instructions. These machine code instructions are debugged with ODP-143. BOOL-143 also enables the user to perform transition checking, (one form of computer-monitoring the Industrial 14).

PAL-143 can be used if greater flexibility in programming the Industrial 14 is required. PAL-143 provides the user with greater flexibility in the approach to monitoring and more efficient use of memory. ODP-143 is also used to debug this translated program.

In general, BOOL-143 and PAL-143 are mutually exclusive; the user chooses one or the other, although this is not absolutely necessary. If the required Industrial 14 program is a combination of straightforward equations and more complicated instruction sequences, the user can compile part of the program with BOOL-143 and assemble the remainder with PAL-143. The two partial programs are merged by ODP-143. Care must be exercised to ensure that the same memory locations are not used by both the BOOL-143 compiled program and the PAL-143 assembled program.

ODP-143 is a powerful tool for debugging and changing Industrial 14 machine language programs. The majority of Industrial 14 users will find ODP-143 helpful, regardless of whether BOOL-143 or PAL-143 is chosen to translate the program.

# CHAPTER 2
# INDUSTRIAL 14
# BASIC INSTRUCTIONS

Industrial 14 basic instructions specify the state (ON or OFF) of outputs as a consequence of the changing states of both inputs and outputs. They are the primary operating instructions of the Industrial 14 System.

An Industrial 14 program can be written as either equations for BOOL-143 or as instructions for PAL-143; however, the program's final form always uses the basic instructions. Industrial 14 instructions used for computer monitoring are discussed in Chapter 9.

## BASIC INSTRUCTION CLASSES

Three classes of basic instructions are used in Industrial 14 programs:

*Test Instructions* – sample the current state (ON or OFF) of an input or output.

*Set Instructions* – set outputs ON or OFF. The state of inputs is changed by the controlled equipment, not by the Industrial 14.

*Jump Instructions* – execute instructions out of sequence, i.e., "branch" the Industrial 14 program. Instructions are always executed in sequence unless a jump occurs.

## INDUSTRIAL 14 TEST FLAG

The TEST flag is the element of the Industrial 14 System which records the results of each test command. 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 later.) The condition of the TEST flag is sampled by jump instructions to cause outputs to be set ON or OFF. In this manner, basic instructions are used in conjunction 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 often used when describing the Industrial 14 processor hardware, in which the TEST flag is actually a solid-state flip-flop circuit.

## INDUSTRIAL 14 TEST INSTRUCTIONS

The Industrial 14 test instructions are:

Test an input, output, or an internal function for the ON state.

Test an input, output, or an internal function for the OFF state.

The symbolic and numeric forms of these instructions and their precise definitions are listed in Table 2-1.

**Table 2-1**
**Industrial 14 Test Instructions**

| Instruction | | Definition |
|---|---|---|
| **Symbolic** | **Numeric** | |
| TF IO | 4000 + IO | Test an input, output, or internal function for the OFF state. If input, output or internal function is OFF and the TEST flag is currently OFF, the TEST flag is set ON; otherwise the flag remains unchanged. |
| TN IO | 6000 + IO | Test an input, output or internal function for the ON state. If input, output or external function is ON and the TEST flag is currently OFF, the TEST flag is set ON; otherwise the flag remains unchanged. |

NOTE: IO represents the numeric values of inputs (0–777), outputs (1000–1377) and internal functions (1400–1777).

The definitions are explained by defining the terms "true test" and "false test" in the following example:

| Current State | Result of Test | |
|---|---|---|
| | Test For ON | Test For OFF |
| ON | True | False |
| OFF | False | True |

Using these definitions, the action of the test instructions can be summarized by the following statement: 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. The TEST flag is *not* cleared by a false test, as shown in the following example:

| Current State | TEST Flag State After Test | |
|---|---|---|
| | True Test | False Test |
| ON | ON | ON |
| OFF | ON | OFF |

## INDUSTRIAL 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 definitions are listed in Table 2-2. The SN and SF instructions reference outputs (1000–1377) and internal functions (1400–1777). Output designations are determined by the position of the output converter within the output panel and output rack, and by the selection of the output cable connector. Internal function designations have no real physical significance; they are discussed in Chapter 3.

**Table 2-2**
**Industrial 14 Set Output Instructions**

| Instruction | | Definition |
|---|---|---|
| Symbolic | Numeric | |
| SN O | 2000 + O | Set output or internal function O to the ON state. O remains ON until it is set OFF by an SF or CLR instruction. |
| SF O | 0000 + O | Set output or internal function O to the OFF state. O remains OFF until it is set ON by an SN instruction. |

NOTE: O represents numeric values of outputs (1000–1377) and internal functions (1400–1777).

## ADDRESSING MEMORY IN THE INDUSTRIAL 14

In any computer-like device, it must be possible to specify particular portions of a control program. For example, jump instructions cause the transfer of program control to a section of the program other than the next sequential program. Therefore, the jump instruction must be capable of specifying the section of the program (memory location) to which it is transferring control.

All instructions are stored in the Industrial 14 memory. The storage processor is not of concern at this point in the discussion, although it should be noted that the instructions are stored in numeric order (in a first, second, third location, etc., through to the last memory location). Each of these memory locations has an assigned number, or address. Thus, an Industrial 14 program consists of instructions (each has an address) which are stored in memory locations. This numbering system (addressing scheme) permits jump instructions to direct the Industrial 14 to "jump to memory location 305" for instance, (begin execution of the instructions starting in memory location 305).

Industrial 14 memory locations are not numbered in the decimal numbering system. They are numbered in the octal numbering system wherein the digits 8 and 9 never appear. Counting in octal is similar to counting in the familiar decimal numbering system except that any number which contains an 8 or 9 in any position is excluded. Thus, the number which follows 7 in octal is 10, and the number which follows 77 is 100.

Where confusion could occur, octal numbers are usually subscripted with an 8 ($10_8$). Thus, $10_8$ is not the same as the decimal number 10. Figure 2-1 illustrates the technique for counting in the octal numbering system.

### Memory Organization

The main memory of the Industrial 14 is subdivided into two fields. A field is 4096 (4K) locations* which are numbered $0_8 - 7777_8$. An Industrial 14/30 Controller can have one field (4K memory) or two fields (two 4K memories). An Industrial 14/35 Controller has two fields (one 8K memory). The lower or first field is designated Field 0; the upper or second field is designated Field 1.

The Industrial 14 memory is further subdivided into pages; each page contains 256 locations. This further subdivision is used for addressing memory by conditional jump instructions. Sixteen pages are in a 4K system; 32 pages are in an 8K system. The 256 locations of a page are numbered $0_8 - 377_8$.

*$4096_{10} = 7777_8$

**Absolute Address and Page Addresses**

Each location in the Industrial 14 memory has an "absolute" address and a "relative" or "page", address. Figure 2-2 shows the relationship between the numbering of locations through all of memory (absolute addresses) and the numbering of locations within each page of memory (relative or page addresses). For example, the location with absolute address 4377 has page address 377. Many locations have the same relative or page address, but each Industrial 14 location has a unique absolute address, within a memory field.

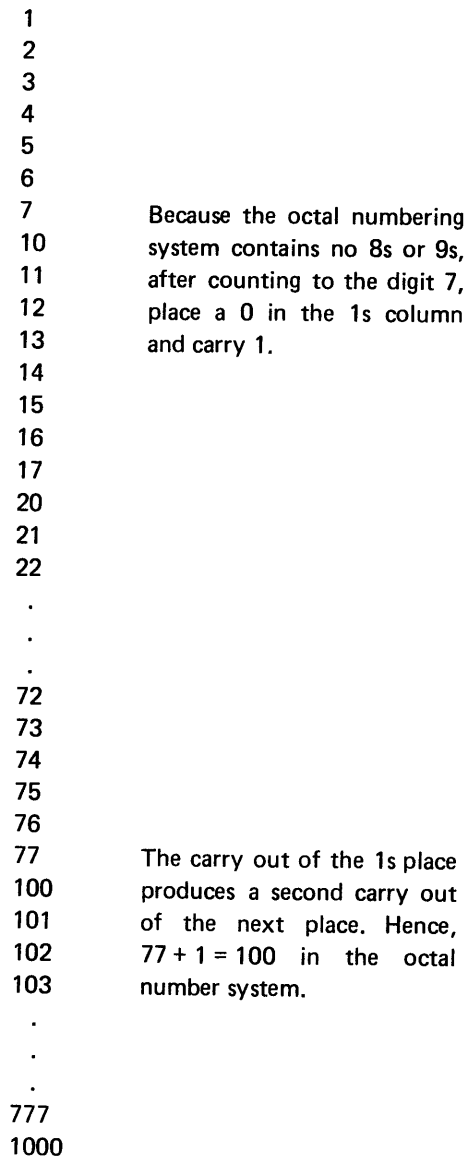



Figure 2-1 Octal Counting Technique

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

1 Disregard the memory field; if the absolute address is 1000 or greater, cross off the first (most significant) digit.

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

3 If the number resulting from step 2 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 1 Result | Page Address |
|---|---|---|
| 1234 | 234 | 234 |
| 3347 | 347 | 347 |
| 2652 | 652 | 252 |
| 7521 | 521 | 121 |
| 7400 | 400 | 0 |
| 5000 | 0 | 0 |

This procedure disregards the memory field. However, location 2652 in Field 0 and location 2652 in Field 1 are in different places in memory where different data is stored. An absolute address, therefore, should include a 0 or a 1 preceding the address to specify the memory field (for example 0 1234, 1 2075).

As will be discussed later, addresses beginning with 7400 and comprising the last page of Field 1 (or Field 0 in a 4K system) are used for storage of internal functions and are not available for programming.

**Industrial 14 Program Flow**

When first powered up, the Industrial 14 automatically begins program execution with the instruction stored in location 0 of Field 0. It then proceeds to location 1, then 2, etc. The result of testing operations often causes the program to jump over groups of instructions; however, the program continually proceeds toward the end of memory. When the end of the first memory field is encountered, the Industrial 14 automatically passes on to the second field. When the program cycle has been completed, control is passed back to the beginning of memory and the cycle begins again.

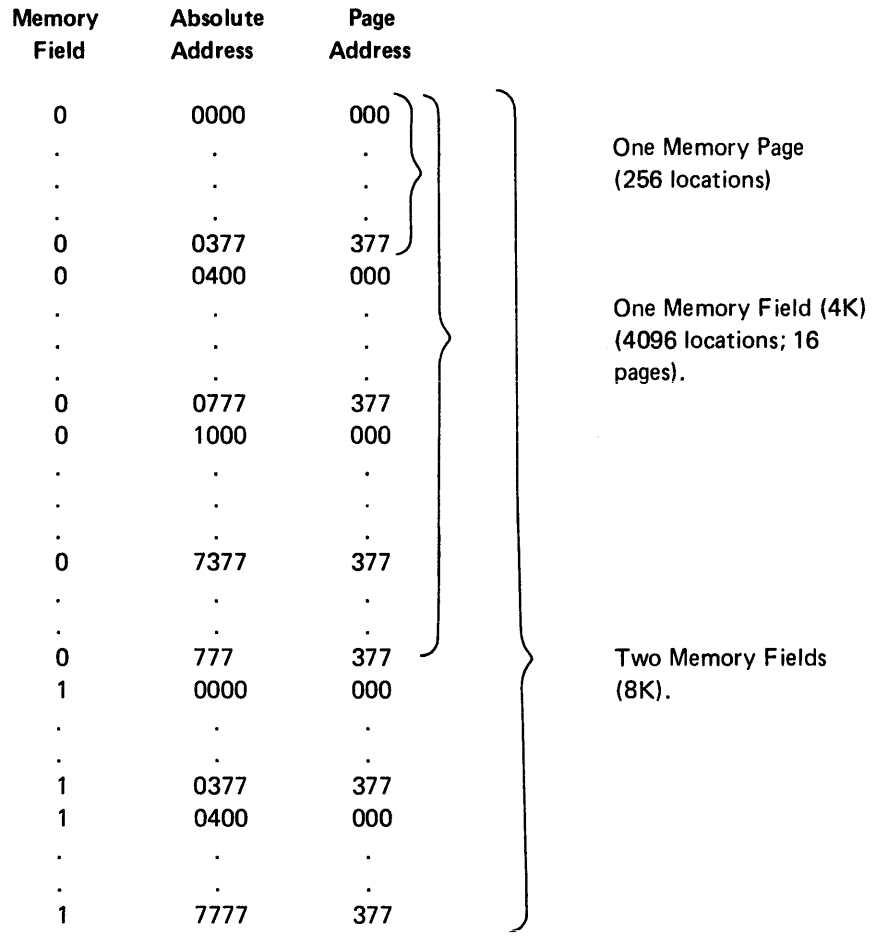



Figure 2-2 Industrial 14 Memory Organization

## INDUSTRIAL 14 JUMP INSTRUCTIONS

Jump instructions cause departures from the sequential execution of instructions. These departures can be conditional (dependent upon the result of I/O tests) or unconditional.

### Conditional Jump Instructions

The Industrial 14 has two conditional jump instructions. The state of the TEST flag determines if the jump will occur. The symbolic and numeric forms of the conditional jump instructions, with definitions, are listed in Table 2-3. Conditional jump instructions cause an SN (set output ON) instruction to occur for one state of the TEST flag and an SF (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 2-3**
**Industrial 14 Conditional Jump Instructions**

| Instruction | | Definition |
|---|---|---|
| Symbolic | Numeric | |
| JFF NNN | 2000 + 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. |
| JFN NNN | 2400 + 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. |

NOTE: NNN represents the relative or page address to which control can be transferred. Legal values are $0_8$ to $377_8$.

The instruction JFN 300 illustrates the use of a conditional jump instruction. This instruction causes a jump to relative location 300 on the current page if the TEST flag is ON. Remember that several absolute addresses have the same relative or page address; a conditional jump instruction, therefore, has different meanings when it is stored on different pages of Industrial 14 memory. Refer to the examples in the following paragraphs to properly understand the memory locations to be addressed. The following examples are the first and last addresses contained on each page of each field.

| | | | |
|---|---|---|---|
| 0000 | 2000 | 4000 | 6000 |
| 0377 | 2377 | 4377 | 6377 |
| 0400 | 2400 | 4400 | 6400 |
| 0777 | 2777 | 4777 | 6777 |
| 1000 | 3000 | 5000 | 7000 |
| 1377 | 3377 | 5377 | 7377 |
| 1400 | 3400 | 5400 | 7400 |
| 1777 | 3777 | 5777 | 7777 |

If the instruction address and the location to be addressed are on the same page, the conditional jump instruction is valid. The following are examples of valid and invalid conditional jump instructions:

| | |
|---|---|
| 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: | Invalid (off-page reference) |
| Location of Instruction: | 4251 |
| Location to be Addressed: | 5300 |
| Correct Instruction: | Invalid (off-page reference) |
| Location of Instruction: | 5631 |
| Location to be Addressed: | 5237 |
| Correct Instruction: | Invalid (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: | Invalid (off-page reference) |

**NOTE**

**The foregoing examples which were shown to be invalid cannot be performed with conditional jump instructions; modifications in the program are required. The program can be changed to use unconditional jump instructions to pass from memory page to memory page. Unconditional jump instructions are described in the following section.**

**Unconditional Jump Instructions**

The Industrial 14 has two unconditional jump instructions. The symbolic form, numeric form and definition of each instruction is listed in Table 2-4.

**Table 2-4**
**Industrial 14 Unconditional Jump Instructions**

| Instruction | | Definition |
|---|---|---|
| **Symbolic** | **Numeric** | |
| JMP NNNN | 0024 NNNN | Jump to location NNNN unconditionally. Execution of the Industrial 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 location which contains the JMP instruction. |
| SKP | 0010 | Skip the following memory location unconditionally. This instruction causes the Industrial 14 to disregard the instruction stored in the next sequential location. Sequential execution continues with the instruction stored in the second location following the SKP instruction. |
| CLRPC | 0004 | Clear the program counter. Causes an unconditional jump to location 0, Field 0. |

NOTE: NNNN represents an absolute address within one field of the Industrial 14 memory. No page restriction exists for the JMP instruction.

As stated in Table 2-4, the JMP is a two-location instruction. The first part of the instruction directs the Industrial 14 to jump to the address given in the second part of the instruction. The first instruction executed after the jump is the one stored in the location in the second part of the instruction.

The JMP instruction can reference any Industrial 14 memory location within its memory field 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 Industrial 14 memory location. Its use is convenient because the address of the location to which control is passed need not be specified. The SKP instruction has no need to reference memory. The SKP instruction skips only one Industrial 14 location; therefore, it can be used to skip any one-location instruction but cannot be used to skip a two-location instruction (such as a JMP).

The CLRPC instruction is used at the end of the Industrial 14 program to complete the cycle by transferring control back to the beginning of memory. It is a single-location instruction that unconditionally causes the instruction in location 0 of Field 0 to be executed next. This occurs regardless of where in memory the CLRPC instruction is located.

**Subroutine Jump Instructions**

Jump instructions that enable subroutines to be used in a program are included in the Industrial 14 basic instruction set. A subroutine is simply a set of instructions to be executed more than once on each pass through the Industrial 14 memory. Instead of repeating the instruction set for each time it is to be executed in the program, the

instructions are written only once, in the form of subroutine. Repeated execution of the same instructions is permitted by an instruction to jump to the start of the subroutine instruction. Another instruction causes a return to the main program after all instructions in the subroutine have been executed.

Table 2-5 lists the jump instructions used for subroutines. The JMS instruction is similar to the JMP instruction; both are two-location instructions which can address any Industrial 14 memory location within its memory field. The JMS differs from the JMP in that it enables the 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 2-5**
**Industrial 14 Subroutine Jump Instructions**

| Instruction | | |
|---|---|---|
| **Symbolic** | **Numeric** | **Definition** |
| JMS<br>NNNN | 0124<br>NNNN | Jump to the subroutine beginning in location NNNN. The Industrial 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 can directly address all 4K of Industrial 14 memory. |
| JMR | 0054 | Jump return to the location following the second part of the JMS instruction. The JMR must always be paired with a JMS instruction. |

NOTE: NNNN (0–7777) represents the absolute address of the start of a subroutine. No page restriction exists for the JMS instruction.

The Industrial 14 permits only one level of subroutines because only one return address can be stored at one time. If a subroutine jumps to a second subroutine, the first return address is lost, and return to the original program is impossible.

Industrial 14 subroutines are written in the following manner:

| Main Program | Subroutine |
|---|---|
| . | 1200 TN 27 |
| . | 1201 TN 30 |
| . | . |
| TN 5 | . |
| TF 7 | . |
| JFF 250 | JMR |
| JMS | |
| 1200 | |
| JFN 250 | |
| TN 12 | |
| . | |
| . | |
| . | |

In this example, the JMS 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. Subroutine transfers must always be within a single memory field.

### CHANGE FIELD INSTRUCTIONS

Industrial 14 Controllers equipped with 8K of memory must have the capability of passing from one memory field to the other. Normally, this is accomplished by permitting the program counter to simply overflow from memory Field 0 into memory Field 1. The normal return to Field 0 is via a CLRPC instruction. However, requirements exist for passing between fields by other methods; these requirements are served by the CIF instructions in Table 2-6.

The CIF instructions take effect only when a JMP, JMS, JMR, JFF, or JFN instruction is executed. Thus, normal useage has a CIF followed by a JMP NNNN, to transfer to location NNNN in the other memory field.

**Table 2-6**
**Change Field Instructions**

| Instruction | | Definition |
|---|---|---|
| Symbolic | Numeric | |
| CIF 0 | 0020 | Change to executing instructions in Field 0 on the next JMP, JMS, JMR, JFF, or JFN instruction. |
| CIF 1 | 0030 | Change to executing instructions in Field 1 on the next JMP, JMS, JMR, JFF, or JFN instruction. |

For example, to transfer control to 1:0100 (location 100 in Field 1 from Field 0).

| Location | Content |
|---|---|
| . | . |
| . | . |
| . | . |
| 5200 | CIF1 |
| 5201 | JMP |
| 5202 | 100 |
| . | . |
| . | . |
| . | . |
| 1:0100 | (Contains next instruction to be executed). |

## SAMPLE INDUSTRIAL 14 EQUATION AND ANALYSIS

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

$$1001 = 123+54*1003$$

This equation could represent the control function, "Solenoid A (1001) is energized when pushbutton 43 (123) is ON, or when both limit switch 12 (54) is activated and Solenoid C (1003) is energized."

The equation can be solved by the Industrial 14 with the instructions in Table 2-7. Note that the user must record the location of each instruction and refer to each input and output by its I/O number.

To demonstrate that the instructions in Table 2-7 actually solve the equation for all possible values of the variables, three cases are analyzed. 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 1003 OFF; Result: 1001 OFF.

| Location | Content | Program Execution |
|---|---|---|
| 300 | TF 123 | Input 123 is tested for the OFF state. Because it is OFF, the TEST flag is set ON, and program execution proceeds to the instruction contained in location 301. |

(continued on next page)

**Table 2-7**
**Sample Industrial 14 Program**

| Location | Symbolic Content | Comment |
|---|---|---|
| 300 | TF 123 | If input 123 is OFF, the flag is set ON. |
| 301 | JFF 305 | If the flag is OFF, jump to 305 and set |
| 302 | TF 54 | 1001 ON. Otherwise, if either input 54 |
| 303 | TF 1003 | is OFF, or if output 1003 is OFF, the |
| 304 | JFN 307 | flag is set ON. If the flag is ON, jump |
| 305 | SN 1001 | to 307 and set 1001 OFF. Otherwise set |
| 306 | SKP | output 1001 ON and skip the set output |
| 307 | SF 1001 | OFF instruction. |

| Location | Content | Program Execution |
|---|---|---|
| 301 | JFF 305 | If the TEST flag is OFF at this point, the program proceeds to location 305. The flag is ON, however, and program execution proceeds to the instruction contained in location 302. |
| 302 | TF 54 | The program tests input 54 for the OFF state. Because input 54 is OFF, the TEST flag is now set ON. Program execution proceeds to the instruction contained in location 303. |
| 303 | TF 1003 | The program tests output 1003 for the OFF state. Because output 1003 is OFF, the TEST flag would be set ON if it had not been set ON by the previous instruction. Program execution proceeds to the instruction contained in location 304. |
| 304 | JFN 307 | If the TEST flag is OFF at this point, the program proceeds to location 305 to set output 1001 ON. The TEST flag is ON, however, and program execution proceeds to the instruction contained in location 307. (The TEST flag is set OFF by the JFN before proceeding). |
| 307 | SF 1001 | Output 1001 is set OFF (as it should be). Program execution proceeds to the instruction contained in location 310. |

**NOTE**

**The instructions in locations 305 and 306 are not executed.**

**CASE 2** – Input 123 ON, input 54 ON, and output 1003 OFF; Result 1001 ON.

| Location | Content | Program Execution |
|---|---|---|
| 300 | TF 123 | Input 123 is tested for the OFF state. Because input 123 is ON, the TEST flag remains OFF and program execution proceeds to the instruction contained in location 301. |
| 301 | JFF 305 | Because the TEST flag is OFF at this point, program execution proceeds to the instruction contained in location 305. (The TEST flag is set OFF by the JFF before proceeding). |
| 305 | SN 1001 | Output 1001 is set ON (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. |

**NOTE**

**The instruction in locations 302, 303, 304, and 307 are not executed.**

**CASE 3** – Input 123 OFF, input 54 ON, and output 1003 ON; Result: 1001 ON.

| Location | Content | Program Execution |
|---|---|---|
| 300 | TF 123 | Input 123 is tested for the OFF state and because it is OFF, the TEST flag is set ON and program execution proceeds to location 301. |

| Location | Content | Program Execution |
|---|---|---|
| 301 | JFF 305 | Because the TEST flag is ON at this point, program execution does not proceed to location 305, but rather proceeds to the instruction contained in location 302. |
| 302 | TF 54 | Input 54 is tested for the OFF state. Because input 54 is ON, the TEST flag remains OFF and program execution proceeds to the instruction in location 303. |
| 303 | TF 1003 | Output 1003 is tested for the OFF state. Because output 1003 is ON, the TEST flag again remains OFF and program execution proceeds to the instruction contained in location 304. |
| 304 | JFN 307 | Because the TEST flag is OFF at this point, program execution proceeds to the instruction contained in location 305. |
| 305 | SN 1001 | Output 1001 is set ON (as it should be). Program execution then 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. |

**NOTE**

**The instruction in location 307 is not executed.**

Industrial 14 programs can be written as in Table 2-7, or they can be written in formats which are more convenient to work with. Techniques for writing Industrial 14 programs are fully described in Chapters 4, 5 and 6. However, an understanding of the material described in this chapter is necessary to perform the debugging procedure required for all BOOL-143 and PAL-143 developed Industrial 14 programs.

## PROGRAM EXAMPLES

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

An explanation of the control function.

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

A Boolean representation of the control function.

A wiring diagram for the Industrial 14 input and output terminals.

The Industrial 14 machine instructions for the control function.

In each example, the TEST flag is assumed to be initially OFF.

*Boolean OR Control Function* – A simple control circuit in which a solenoid is energized by either of two limit switches is present in Example 1. To solve this problem with an Industrial 14, test for either input's ON state. Because a positive test from either input sets 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.

**NOTE**

**Normally-open contacts of each switch are wired to the Industrial 14. Thus, testing for ON means testing for an activated switch. Testing for OFF means testing for a nonactivated 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).

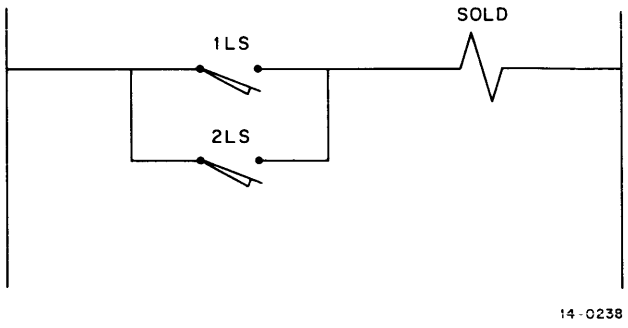



Figure 2-3 Ladder Diagram for Boolean OR Function

**Boolean Equivalent**

| Input | Output |
|---|---|
| 1LS (n/o) 11<br>2 LS (n/o) 12 | SOLD 1007 |

**Equation:** 1007 = 11 + 12 or
SOLD = 1LS + 2LS

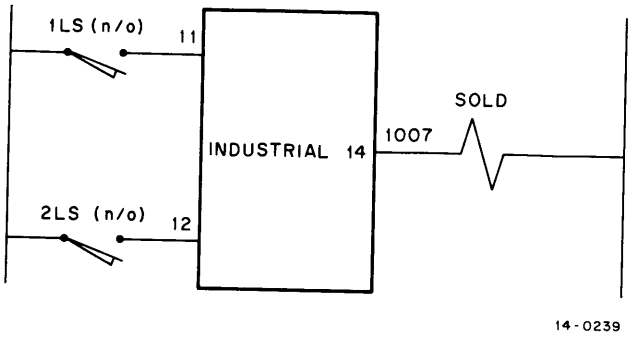



Figure 2-4 Typical Industrial 14 Wiring Diagram for Example 1

**Industrial 14 Program**

| Location | Content | Comments |
|---|---|---|
| 0 | TF 11 | Test input 11 for OFF. If it is ON, set output 1007 ON. If input 11 is OFF, test input 12 for OFF. If input 12 is ON, set output 1007 ON, if not, set output 1007 OFF. |
| 1 | JFF 4 | |
| 2 | TF 12 | |
| 3 | JFN 6 | |
| 4 | SN 1007 | |
| 5 | SKP | |
| 6 | SF 1007 | |

*Boolean and Control Function* – Example 2 illustrates a control function where two inputs are in series to drive an output. In other words, both Limit Switch 3 and Push-button 1 must be activated for Solenoid A to be energized. Normally open contacts are wired to the Industrial 14 input.

The Industrial 14 program checks for the condition which is not wanted. If either input is OFF, Solenoid A must not be energized. Thus, the program tests both inputs, setting the TEST flag if either is OFF. If the TEST flag is ON, the program jumps 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).

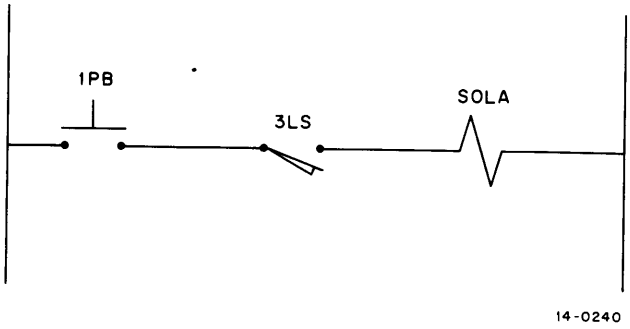



Figure 2-5 Ladder Diagram for Boolean AND Function

**Boolean Equivalent**

| Input | Output |
|---|---|
| 1PB (n/o) 1<br>3LS (n/o) 13 | SOLA 1004 |

**Equation:** 1004 = 1 * 13 or
SOLA = 1PB * 3LS

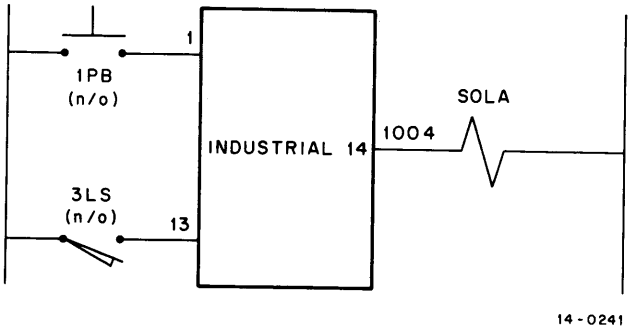



Figure 2-6 Typical Industrial 14 Wiring Diagram for Example 2

**Industrial 14 Program**

| Location | Content | Comment |
|---|---|---|
| 7 | TF 1 | Test inputs 1 and 13. If either is |
| 10 | TF 13 | OFF, output 1004 will be set OFF; |
| 11 | JFN 14 | otherwise 1004 will be set ON. |
| 12 | SN 1004 | |
| 13 | SKP | |
| 14 | SF 1004 | |

*Control Functions with Normally Open (NO) and Normally Closed (NC) Contacts* – Example 3 is a control function which (in ladder diagram form) has both normally open and normally closed contacts. The Industrial 14 does not require two types of contacts for such control functions. All normally open contacts can be used, as illustrated in Example 3. Normally open contacts for both Pushbutton 7 and Limit Switch 5 are wired to the Industrial 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 Industrial 14 program could also be written to test normally closed contacts. The program merely tests for the opposite state (ON or OFF) of the input. Thus, if normally closed contacts are used in the wiring, all TFs become TNs for the input and all TNs become TFs. An ON input from normally closed contacts specifies that the switch is not activated; an OFF input specifies that 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, Solenoid C is energized (otherwise Solenoid C is de-energized).

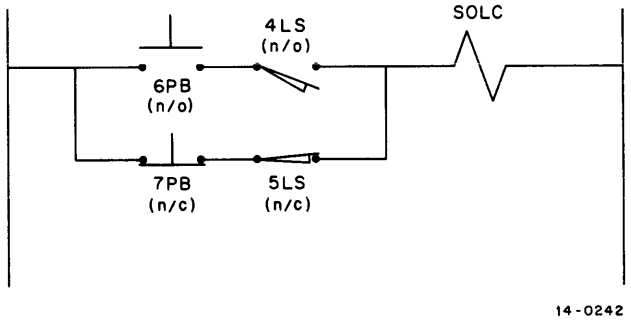



Figure 2-7 Ladder Diagram with NO and NC Contacts

**Boolean Equivalent**

| Input | Output |
|---|---|
| 6PB (n/o) 6 | SOLC 1006 |
| 7PB (n/c) 7 | |
| 4LS (n/o) 14 | |
| 5LS (n/c) 15 | |

**Equation:** 1006 = (6 * 14) + (7 * 15) or
SOLC = (6PB*4LS) + (7PB*5LS)

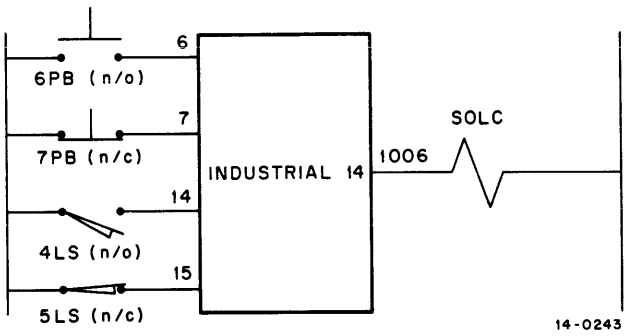



Figure 2-8a Typical Industrial 14 Wiring Diagram for Example 3 Using Normally Closed Contacts

Inputs and outputs are often sensed many times throughout an Industrial 14 program. To avoid confusion, these inputs and outputs should be wired normally open.†

**Boolean Equivalent**

| Input | Output |
|---|---|
| 6PB (n/o) 6 | SOLC 1006 |
| 7PB (n/o) 7 | |
| 4LS (n/o) 14 | |
| 5LS (n/o) 15 | |

**Equation:** 1006 = (6*14) + (/7*/15)
SOLC = (6PB*4LS) + (/7PB * /5LS)

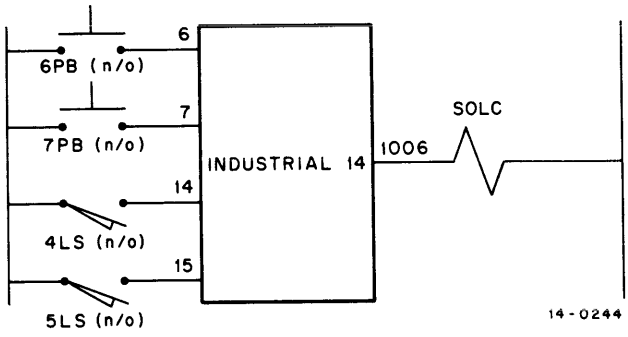



Figure 2-8b Typical Industrial 14 Wiring Diagram of Normally Open Input for Example 3

†Only inputs sensing emergency or critical conditions should be wired normally closed.

**Industrial 14 Program**

| Location | Content | Comment |
|---|---|---|
| 15 | TF 6 | If either input 6 or 14 is OFF, the |
| 16 | TF 14 | other leg must be tested. If both are |
| 17 | JFF 23 | ON, the output should be set ON. |
| 20 | TN 7 | If either input 7 or 15 is ON and |
| 21 | TN 15 | the other leg fails, set output OFF. |
| 22 | JFN 24 | Set output OFF if both legs fail. |
| 23 | SN 1006 | |
| 24 | SKP | |
| 25 | SF 1006 | Set output ON if either leg solves. |

*Latching Control Function* – Example 4 is a motor contactor latching function. Pushbutton 2 (2PB) is the start button, and Pushbutton 3 (3PB) is the stop button for controlling a motor contactor. Normally open contacts are assumed for all inputs; however, the normally closed contact for Pushbutton 3 could be used by replacing TF 3 by TN 3 in the Industrial 14 program.

The program uses a test output instruction, TF 1010, to latch the motor contact on. The output then remains ON as long as it is currently ON and Pushbutton 3 is not depressed. An Industrial 14 input for the motor contacts is not required because outputs can be tested as inputs.

**Example 4 – Control Function**
The motor (1M) is started if Pushbutton 2 is pressed, and continues to operate until Pushbutton 3 is pressed.

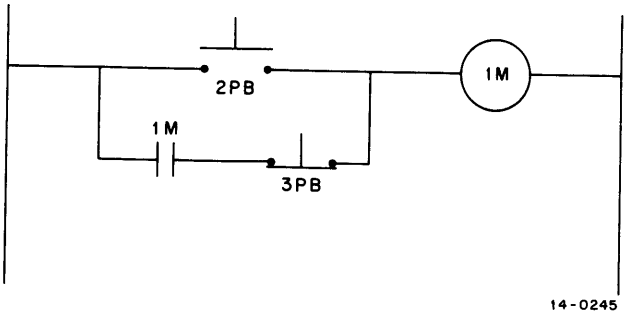



Figure 2-9 Ladder Diagram for Latched Control Function

**Boolean Equivalent**

| Input | Output |
|---|---|
| 2PB (n/o) 2 | 1M 1010 |
| 3 PB (n/o) 3 | |

**Equation:** 1M = 2PB + (3PB*1M) or 1010 = 2 + (/3*1010)

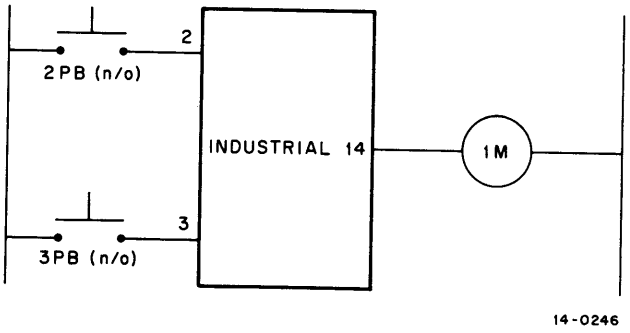



Figure 2-10 Typical Industrial 14 Wiring Diagram for Example 4

| Location | Contents | Comment |
|---|---|---|
| 26 | TF2 | If input 2 is OFF, test input 3; |
| 27 | JFF 33 | otherwise, set output 1010 ON. If |
| 30 | TN 3 | input 3 is ON or output 1010 is |
| 31 | TF 1010 | OFF, set output 1010 OFF; other- |
| 32 | JFN 35 | wise, set the output ON. |
| 33 | SN 1010 | |
| 34 | SKP | |
| 35 | SF 1010 | |

*Relay Controlled Function* – Example 5 uses a control relay to energize a solenoid. All inputs to the Industrial 14 are normally open contacts. (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 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 if either Pushbutton 5 is ON or Control Relay 2 is already ON, then Control Relay 2 is set ON. If Control Relay 2 is ON, Solenoid B is energized.

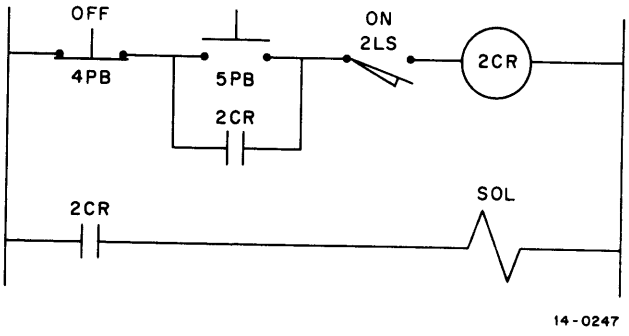



Figure 2-11 Ladder Diagram for Relay Controlled Function

**Boolean Equivalent**

| Inputs | Outputs |
|---|---|
| 4PB (n/o) 4 | 2 CR 1277 |
| 5PB (n/o) 5 | SOLB 1005 |
| 2LS (n/o) 12 | |

**Equation:** 1377 = /4* (5 + 1377) * 12
1005 = 1377 or
2CR = /4PB*(5PB + 2CR) *2LS
SOLB = 2CR

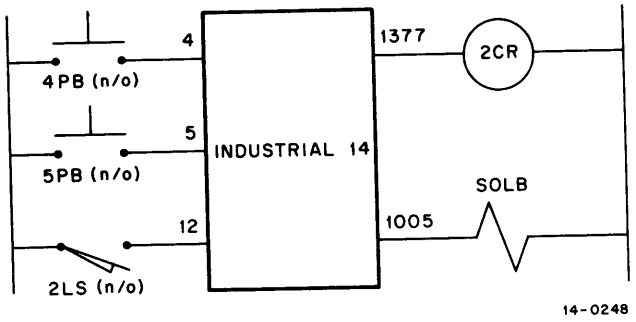



Figure 2-12 Typical Industrial 14 Wiring Diagram for Example 5

**Industrial 14 Program**

| Location | Contents | Comment |
|---|---|---|
| 36 | TN 4 | If input 4 is ON, or input 12 is |
| 37 | TF 12 | OFF, or input 5 is OFF, test output |
| 40 | TF 5 | 1001; otherwise, set output 1001 |
| 41 | JFF 44 | ON. If output 1001 is OFF set |
| 42 | TF 1001 | output 1001 OFF; otherwise, set |
| 43 | JFN 46 | output 1001 ON. |
| 44 | SN 1001 | |
| 45 | SKP | |
| 46 | SF 1001 | |
| 47 | TF 1001 | |
| 50 | JFN 53 | |
| 51 | SN 1005 | If output 1001 is OFF, set output |
| 52 | SKP | 1005 OFF; otherwise set 1005 ON. |
| 53 | SF 1005 | |

Example 5 illustrates an important Industrial 14 concept; although the original equation was 2 CR = 4PB*(5PB+2CR)*2LS, the program was written for the equation 2CR = 4PB*2LS* (5PB+2CR). Programming is more efficient when single variables, connected by the same operator, (AND or OR) are grouped together.

*Simplifications* – The control program for Example 5 could be greatly simplified by noting that Control Relay 2 is an unnecessary output. The control relay is needed in the ladder diagram to latch the circuit ON. In the Industrial 14, however, the output to energize the solenoid may be tested itself as an input. Thus, as shown in Example 5, Simplified (following), output 1001 can be eliminated and the control relay can be removed. In this case, the test of output 1005 is equivalent to testing output 1001.

**Example 5, Simplified – Control Function**
If Pushbutton 4 is not ON and Limit Switch 2 is actuated, and either Pushbutton 5 is ON or Solenoid B is ON already, then Solenoid B is engaged.

**Boolean Equivalent**

| Inputs | Outputs |
|---|---|
| 4PB (n/o) 4 | SOLB 1005 |
| 5PB (n/o) 5 | |
| 2LS (n/o) 12 | |

**Equation:** 1005 = /4 * (5 + 1005) * 12
SOLB = /PB* (5PB + SOLB) *2LS

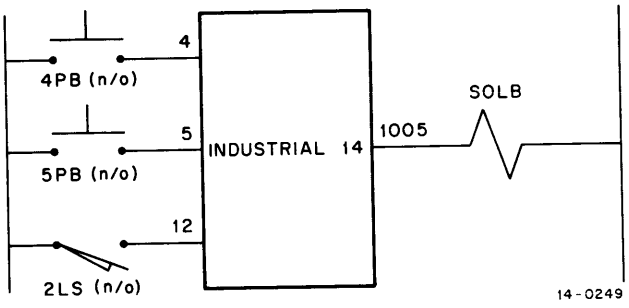



Figure 2-13 Typical Industrial 14 Wiring Diagram for Example 5, Simplified

**Industrial 14 Program**

| Location | Contents | Comments |
|---|---|---|
| 36 | TN 4 | If input 4 is ON and input 12 is |
| 37 | TF 12 | OFF, test output 1005; otherwise, |
| 40 | TF 5 | set output 1005 ON. If output |
| 41 | JFF 44 | 1005 is OFF, set output 1005 OFF; |
| 42 | TF 1005 | otherwise, set output 1005 ON. |
| 43 | JFN 46 | Simply set output 1005 ON or OFF |
| 44 | SN 1005 | directly, without using intermediate |
| 45 | SKP | control relay. |
| 46 | SF 1005 | |

# CHAPTER 3
# INTERNAL FUNCTIONS

This chapter describes Industrial 14 internal functions, I/O groupings, and typical applications. These typical applications are discussed both in terms of circuit diagrams and Boolean equation logic. Except for the shift register application, Industrial 14 machine language is not given for these applications because the instruction format is similar to that of output circuits. Timing diagrams are included with all timer applications to illustrate the sequence of operations.

## INTERNAL I/O GROUPINGS

The 256 internal functions which use I/O numbers 1400–1777 are subdivided as to type: retentive memories, shift registers, timers, counters, and up/down counters. This group can be easily adjusted to provide a "best fit" for each application. The normal groupings and I/O number allocations for internal I/O functions are shown in Figure 3-1. General assignments are as follows:

| I/O Numbers | Use |
|---|---|
| 1400–1577 | Retentive memories or shift registers |
| 1600–1757 | In pairs for timers or event counters |
| 1760–1777 | In groups of 4 for up/down counters |

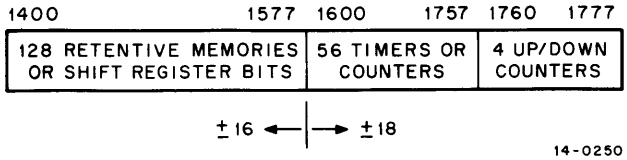



Figure 3-1 Internal I/O Groupings

The I/O numbers from 1400–1577 may be used for either retentive memories or shift registers, at random. When timers are required, the lower I/O numbers between 1600–1757 must be used, beginning at 1600, and consecutively up through the last timer. Counters must use I/O numbers greater than the last timer, and up through 1757.

A typical assignment of internal functions using 1600 as the I/O partition is shown in Table 3-1. The following are of particular importance:

a. Each retentive memory and shift register bit requires one I/O number.

b. All timers and counters require two I/O numbers.

c. The bits of a shift register must be consecutive; however, shift registers and retentive memories may be intermixed.

d. Timers must begin at the I/O partition (in this instance, 1600) and must be in consecutive order using I/O numbers below any counter circuit. (For this reason, several spare timers are provided.)

e. Four permanent up/down counters are provided in the Industrial 14.

## ADJUSTING THE INTERNAL I/O GROUPING

While the grouping of internal functions, shown in Figure 3-1, and Table 3-1, is suitable for a great number of applications, it may be necessary to modify this grouping in certain instances. This can be accomplished by adjusting switches within the Industrial 14 Control Unit and the necessary procedure is described in the Industrial 14 Systems Manual. This modification provides additional (or fewer) retentive memories and shift register bits, by sacrificing (or obtaining additional) timers and counters. This corresponds to relocating the starting point of the timer circuits from 1600 to some other I/O partititon. However, because two I/O numbers are required for a timer or counter, and only one is required for a retentive memory, the gain (or loss) of retentive memories is always double the loss (or gain) of timers and counters.

The internal I/O groupings that can be achieved are shown in Table 3-2. Quantities of timers and counters are gained or lost, in multiples of 8, by sacrificing or increasing retentive memories and shift register bits, in multiples of 16.

**Table 3-1**
**Typical Assignment of Internal I/O Numbers**

| I/O No. | Use | Comment |
|---|---|---|
| 1400 | Retentive memory | |
| 1401 | Retentive memory | |
| 1402 | | |
| 1403 | Shift register | |
| 1404 | | Random retentive memories and shift registers |
| . | | |
| . | | |
| . | | |
| 1450 | Retentive memory | |
| 1451 | Retentive memory | |
| 1452 | Retentive memory | |
| . | | |
| . | | |
| . | | |
| 1576 | Shift register | |
| 1577 | | Adjustable I/O partition |
| 1600 | | |
| 1601 | Timer | |
| 1602 | | |
| 1603 | Timer | Timers used *consecutively* |
| . | | |
| . | | |
| 1670 | | |
| 1671 | Timer – spare | |
| 1672 | | |
| . | Timer – spare | Several timers left as spares |
| . | | |
| . | | |
| 1676 | | |
| 1677 | Timer – spare | |
| 1700 | | |
| 1701 | Counter | |
| 1702 | | |
| 1703 | Counter | |
| . | | Event counters |
| . | | |
| . | | |
| 1757 | Counter | |
| 1760 | | |
| | 4 up/down counters | up/down counters |
| 1777 | | |

Table 3-2
Possible Internal Function Groupings

| Quantity | | | |
|---|---|---|---|
| Retentive Memories or Shift Register Bits | Timers or Counters | Up/Down Counters | I/O Partition (First Timer Circuit) |
| 0 | 120 | 4 | 1400 |
| 16 | 112 | 4 | 1420 |
| 32 | 104 | 4 | 1440 |
| 48 | 96 | 4 | 1460 |
| 64 | 88 | 4 | 1500 |
| 80 | 80 | 4 | 1520 |
| 96 | 72 | 4 | 5140 |
| 112 | 64 | 4 | 1560 |
| 128* | 56 | 4 | 1600 |
| 144 | 48 | 4 | 1620 |
| 160 | 40 | 4 | 1640 |
| 176 | 32 | 4 | 1660 |
| 192 | 24 | 4 | 1700 |
| 208 | 16 | 4 | 1720 |
| 224 | 8 | 4 | 1740 |
| 240 | 0 | 4 | 1760 |

*Normal grouping as shipped by *Digital Equipment Corporation.*

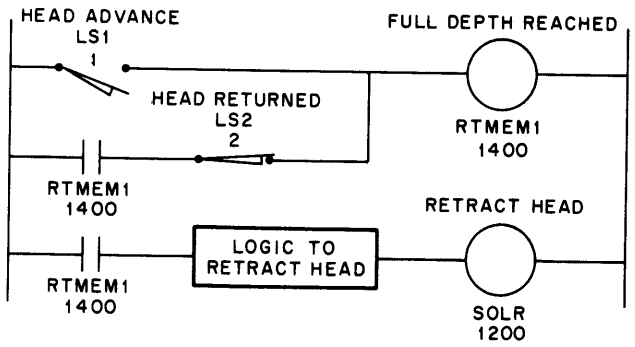
## RETENTIVE MEMORIES

Retentive memory circuits replace latching relays for recording operations or status when power failures occur. All outputs (I/O numbers between 1000–1377 inclusive) are cleared when the Industrial 14 loses power. Retentive memory outputs having I/O numbers between 1400–1577 inclusive, however, are not cleared and retain the previous state when the Industrial 14 is powered up again.

The retentive memory circuit in Figure 3-2 records the full-depth state of a drill-head or probe; the second circuit retracts the head or probe upon reaching the full-depth position.

**NOTE**

**A retentive memory may be programmed to clear in the event of a power shutdown. Refer to "Non-Retentive Internal Functions" presented later in this chapter.**




| INPUTS | OUTPUTS | INTERNAL FUNCTION |
|---|---|---|
| LS1 1 | SOLR 1200 | RTMEM1 1400 |
| LS2 2 | | |

1400 = 1 + ( 1400 * /2 )
1200 = 1400 * ( LOGIC TO RETRACT HEAD )

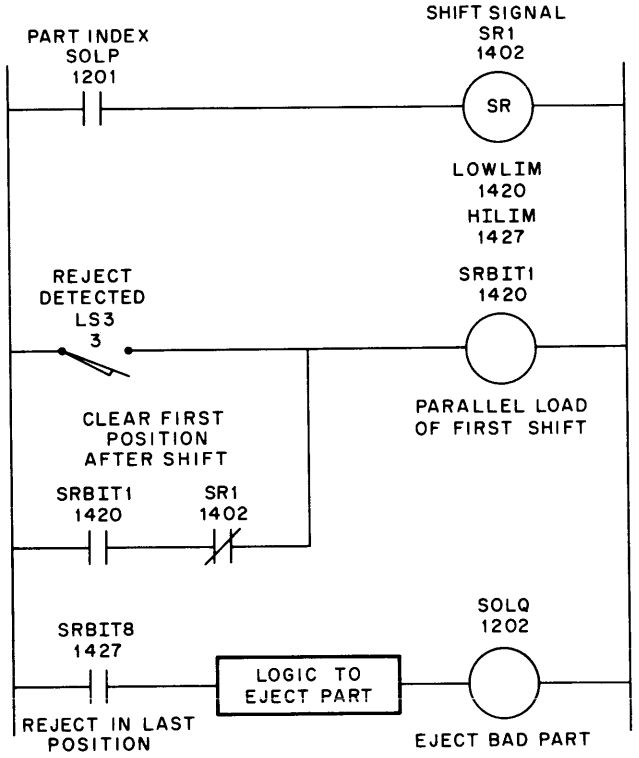
14-0251

Figure 3-2 Retentive Memory Circuit and Boolean Equations to Record Full-Depth Reached

## SHIFT REGISTERS

The Industrial 14 allows any consecutive group of external outputs (I/O numbers 1000–1377) or internal functions (normally I/O numbers 1400–1577) to be used as a shift register. When external outputs are used, the shift register is cleared whenever the Industrial 14 is shut down. When internal functions are used, the shift register holds its state during an Industrial 14 shutdown. (Internal I/O numbers may also be used for a non-retentive shift register. Refer to "Non-Retentive Internal Functions".)

The shift circuit transfers the data from each location to the next higher location (higher in terms of I/O numbers). The shift occurs when the coil of the shift circuit is first energized. The shift circuit remains energized so long as its circuit conditions are satisfied; the next shift occurs when the coil has been de-energized, then re-energized.




| INPUTS | OUTPUTS | INTERNAL FUNCTIONS |
|---|---|---|
| LS3 3 | SOLP 1201<br>SOLQ 1202 | SR1 1402<br>SRBIT1 1420<br>SRBIT8 1427 |

14-0252

1402 = 1201
1420 = 3+(1420*/1402)
1202 = 1427* (Logic to eject part)

Figure 3-3 Shift Register Circuit and Boolean Equations to Track a Reject Part

The shift register bits can be formed either from a series of retentive memories or from unused outputs; however, in either instance they must form a consecutive group. Because the shift register is comprised of individual I/O numbers, a circuit may be designed to parallel-load into any position of the shift register.

Shift registers are useful for numerous applications. One example is in identifying parts in a manufacturing transfer operation – either for acceptance testing or for performing an additional operation. As each part is transferred to a new station, a corresponding register bit is shifted. Figure 3-3 shows 3 circuits that keep track of parts moving through 8 stations and finally reject the part that is bad. The first circuit is the shift circuit; the second loads the reject status into the first bit in the shift register, and the third rejects the bad part based on the state of the last bit in the shift register.

The Industrial 14 machine language instruction format for the shift signal circuit differs from the format of other output or internal function circuits. Prior to setting an output ON when a shift circuit solves, Industrial 14 instruction (TN) tests the shift signal output to determine if an OFF-to-ON transition has occurred. If so, the MOVBIT instruction transfers the contents of the next-to-last shift register location to the last location. The second MOVBIT instruction transfers the contents of the next preceding shift register location to the next-to-last location. Thus, for a total of N shift registers, a total of N-1 MOVBIT instructions should always be executed. The following is the Industrial 14 machine language for the shift signal circuit just described.

```
TF 1201
JFN .+3
JFF .+4        Instead of setting 1402 ON, the
SKP            program jumps to test 1402 for
SF 1402        ON. This checks to determine if an
JFF NEXT       OFF-to-ON transition has occurred
TN 1402        and then sets 1402 ON.
SN 1402
JFN NEXT       If OFF-to-ON transition occurred,
MOVBIT         execute MOVBIT instructions.
1426
1427
MOVBIT
1425
1426
...
...
```

(continued on next page)

```
        . . .
        . . .
        MOVBIT
        1420
        1421
NEXT,
```

The circuit for 1420 in this example illustrates the general form for a shift register parallel-load circuit (Figure 3-4). If the application had called for a cancel signal to clear the shift register bit, it would have appeared in series with the output.

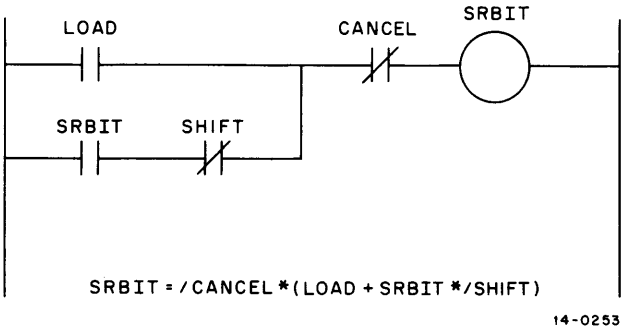



Figure 3-4 General SR Parallel-Load Circuit Ladder Diagram

The amount of controller memory required for a shift register includes the memory locations needed for storing the shift circuit plus the locations needed for moving each bit of the shift register (Figure 3-5). The total amount of memory for a shift register is therefore limited to 256 locations. The smaller the shift circuit, the more bits the shift register can include; larger shift registers possess commensurately fewer bits. The smallest shift circuit allows approximately 81 shift register bits, the largest shift circuit allows approximately 56 shift register bits. If the combination of the shift circuit and the instructions for moving each bit is too great, this condition is detected when the equation of instructions are read in BOOL-143 or PAL-143. If this occurs, simply break the shift register into 2 separate segments.

For example, 2 shift register circuits must be used to shift 120 bits. One circuit might shift the first 40 consecutive bits. The other senses the output coil of the first circuit and then shifts the last 81 consecutive bits (a total of 120 bits plus 1 bit to link the 2 shift registers). Figure 3-6 is a ladder diagram of this circuit.

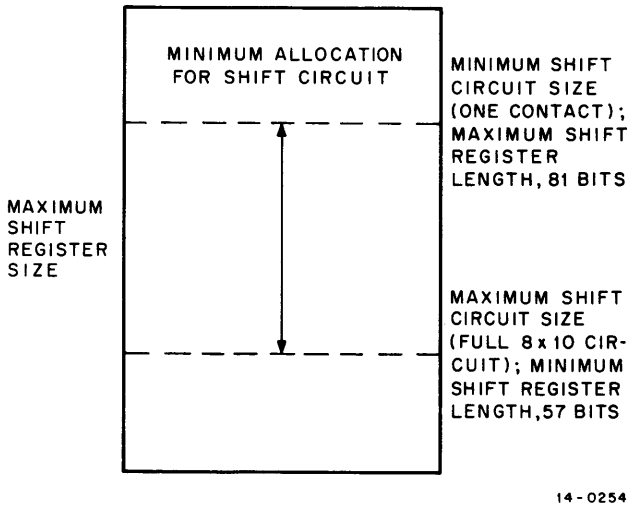



Figure 3-5 Industrial 14 Memory Allocation for Shift Registers

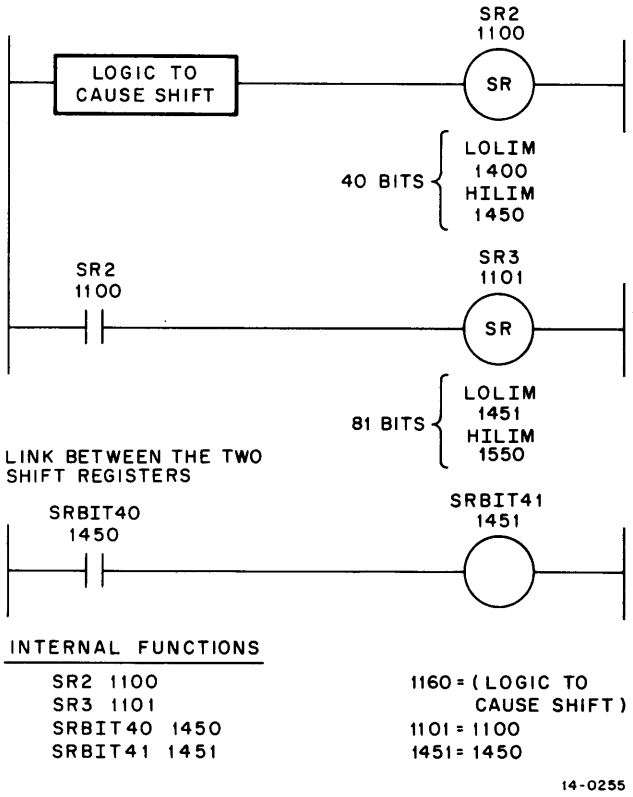



Figure 3-6 Ladder Diagrams and Boolean Equations to Link Two Shift Registers

## TIMERS

Industrial 14 timer circuits each require two I/O numbers; these are usually in the range 1600–1757 (Internal I/O Groupings, presented in this chapter). The timer circuit is programmed using the first, even numbered, I/O number. When this i/O number (for instance, 1600) is energized, the timer starts; when it is de-energized, the timer clears, regardless of whether or not the full time interval has elapsed.

The contact for the even I/O number (1600) senses the instantaneous state of the timer. The contact for the odd I/O number (1601) senses the delayed timed-out state. The states of these 2 contacts are shown in Figure 3-7. Notice the results when the timer circuit is de-energized prior to completing the full delay.

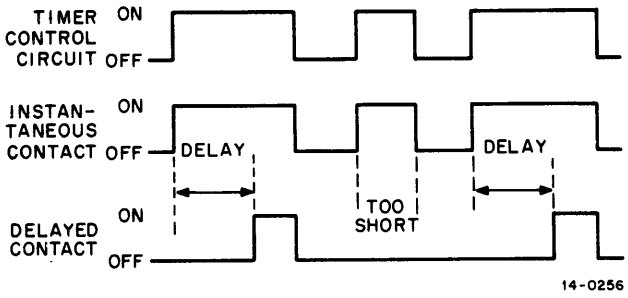



Figure 3-7 Timing Circuit Operation Timing

A timer can be used in conjunction with external outputs to delay the turn-on of power devices, such as solenoids or motor starters. Figure 3-8a shows timing in such a timer circuit. Notice that an input condition starts the timer and the timed-out state turns on output 1000. Figure 3-8b is a ladder diagram of the circuit.

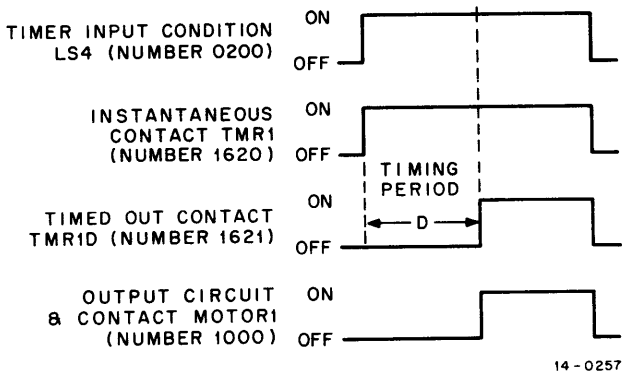



Figure 3-8a Turn-On Delay Circuit Timing

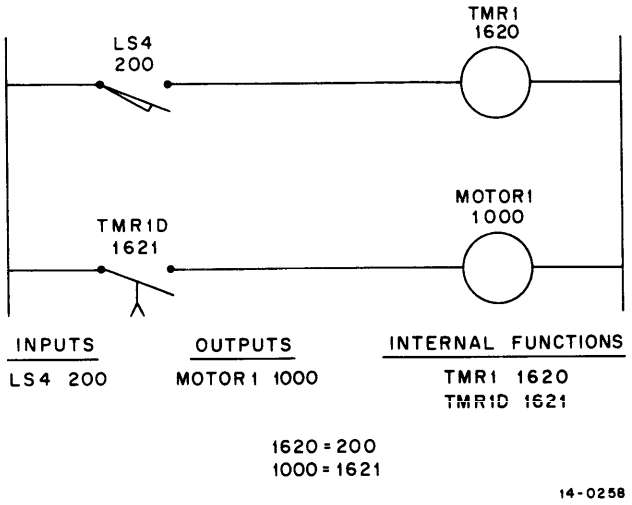



Figure 3-8b On-Delay Timer Operation Ladder Diagram

Occasionally, it is useful to sense the instantaneous contact (even number) of a timer in a lock-up (latch) circuit to turn on an external output at a given time after an input condition is sensed. An example of this is a package on a conveyor that passes a photoswitch and moves on to an elevator. A time delay must exist between the photoswitch detecting the package and the elevator starting to move. The instantaneous contact maintains the timer coil energized even though the input is no longer present. Figures 3-9a and 3-9b are timing and ladder diagrams respectively, for this example.

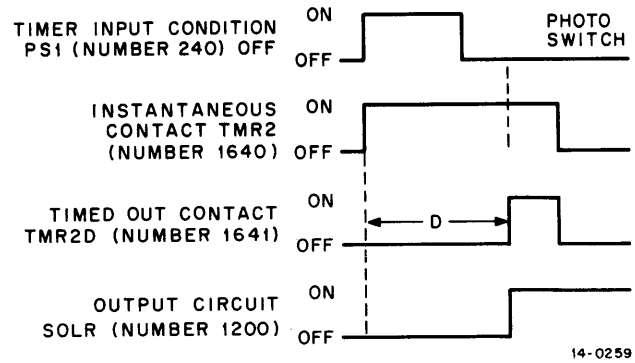



Figure 3-9a Timer Lockup Circuit Timing

Timers are also used in applications to extend the ON time of an output (Figures 3-10a and 3-10b) or to set an output ON for a given time after de-energizing an input (Figures 3-11a and 3-11b).

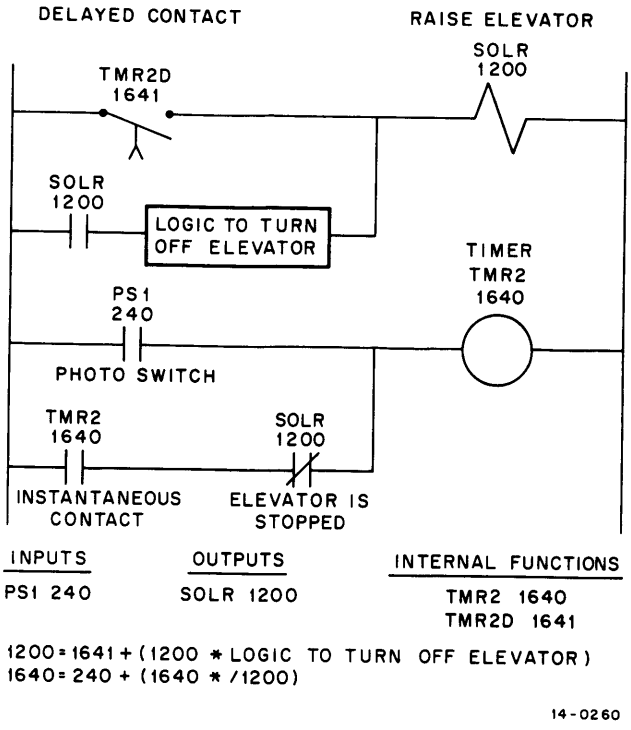



Figure 3-9b Timer Lockup Circuit Ladder Diagram and Boolean Equations

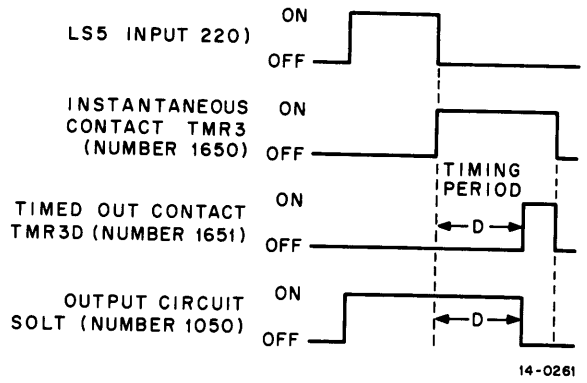



Figure 3-10a OFF Delay Timer Circuit Timing

Another timer application is in energizing an output for a fixed interval following de-energization of an input. The timing and ladder diagrams for this application are shown in Figures 3-11a and 3-11b respectively.

## EVENT COUNTERS

Event counters (up-count only) share I/O numbers 1600–1757 with timer circuits; however, each counter requires a higher I/O number than the highest timer I/O number. Two control circuits with consecutive numbers are required for each event counter: the even numbered circuit increments the counter when set ON and the odd numbered circuit clears the counter when energized. Increments that occur while the clear circuit is ON have no effect on the counter.

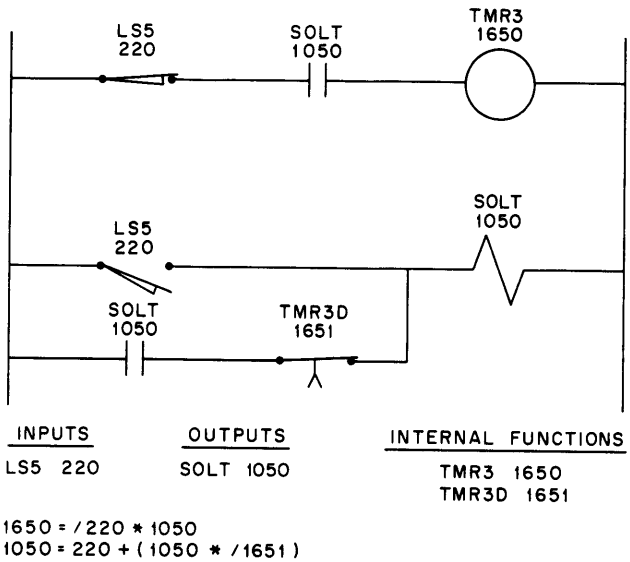



Figure 3-10b OFF Delay Timer Circuit Ladder Diagram and Boolean Equations

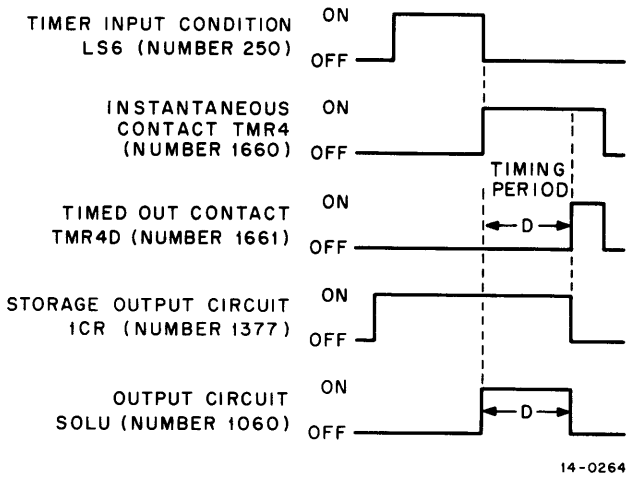



Figure 3-11a Circuit Timing for Energizing an Output for a Fixed Interval Following De-energization of an Input

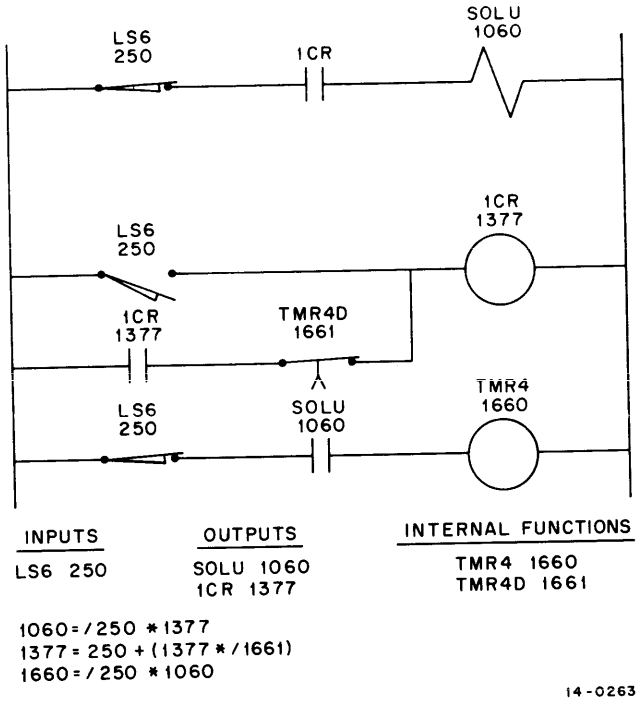



Figure 3-11b Circuit and Boolean Equations to Energize an Output for a Fixed Interval After De-energization of an Input

The odd-numbered contact of the up-counter senses whether the count equals or exceeds the preset count. This sensed contact can be used in parallel or in series with other contacts within control circuits.

Figure 3-12 is an example of an up-counter to shut down a station after three successive bad parts are detected. Notice that the normally closed contact (1701) senses whether the number of parts is less than 3 and stops the station operation if the count is 3 or more.

## UP/DOWN COUNTERS

The Industrial 14's selection of internal functions always includes 4 up/down counters. These counters are permanently assigned the I/O numbers listed in Table 3-3. Up/Down Counters are programmed using 3 I/O numbers; the first to increment, the second to clear, and the third to decrement the counter. The fourth I/O number is not available for use.

Contacts for the first I/O number (1760, 1764, 1770, and 1774) are used in other control circuits to indicate that the preset value has been reached. Contacts for the second I/O number (1761, 1765, 1771, and 1775) are used to test for a zero counter value.

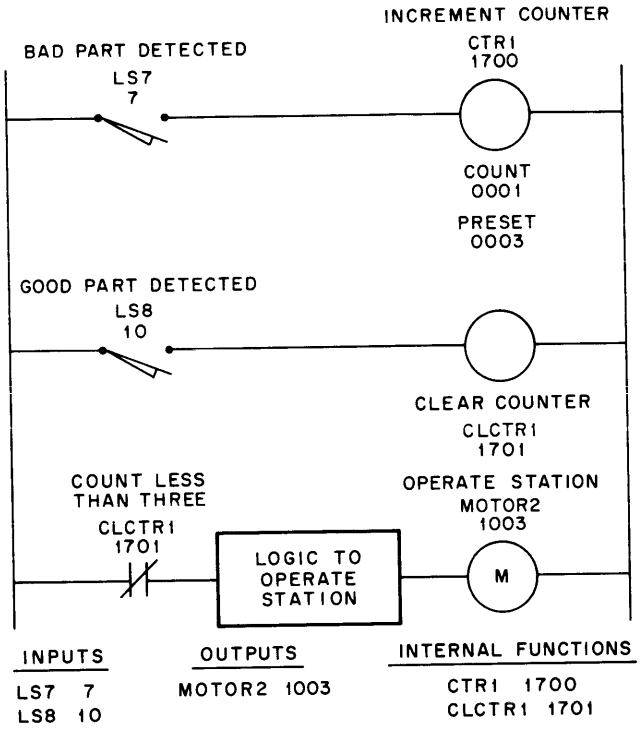



Figure 3-12 Up Counter Circuit and Boolean Equations to Shut Down a Station on Three Successive Bad Parts

A typical up/down counter application is in limiting parts on a conveyor. In the following example (Figure 3-13), the contact (1760) senses whether or not six parts are in a particular conveyor zone. If six parts are on the conveyor, a gate is activated to prevent additional parts from entering.

## CASCADING A TIMER AND COUNTER

A timer interval greater than 999 seconds can be obtained by cascading a timer with a counter. The preset value of each can be determined with the following equation:

TOTAL ELAPSED TIME = (TIMER PRESET VALUE) X (COUNTER PRESET VALUE)

For example, if a 1-hour (3600 second) timer is desired, preset the timer to 60 seconds and the counter to 60. This example is illustrated in Figure 3-14.

Table 3-3
Up/Down Counters

| Counter | I/O Number | Function | |
|---|---|---|---|
| | | As An Output | As A Contact |
| 1 | 1760 | Count up | = Preset |
| | 1761 | Clear | =0 |
| | 1762 | Count down | N/A |
| | 1763 | Do not use | N/A |
| 2 | 1764 | Count up | = Preset |
| | 1765 | Clear | =0 |
| | 1766 | Count down | N/A |
| | 1767 | Do not use | N/A |
| 3 | 1770 | Count up | = Preset |
| | 1771 | Clear | =0 |
| | 1772 | Count down | N/A |
| | 1773 | Do not use | N/A |
| 4 | 1774 | Count up | = Preset |
| | 1775 | Clear | =0 |
| | 1776 | Count down | N/A |
| | 1777 | Not used in Counter | Initialize |

## NON-RETENTIVE INTERNAL FUNCTIONS

Although all external output circuits turn OFF if the Industrial 14 loses power, all internal functions normally retain the current state or value during and after a power failure. This means, for example, that a timer which is interrupted part way through its interval by a shut-down of the Industrial 14 will continue (not reset) when power returns. However, the Industrial 14 has an I/O number (1777) which acts as a system initialize signal to reset timing or to initiate some special action when the Industrial 14 is first powered up. This contact designated "not initialize" (INITIALIZE) is OFF following a power shutdown for the first pass through Industrial 14 Controller memory and ON thereafter. If a pneumatic type timer is desired, i.e., one that resets with a power failure, the normally open, not initialize (1777) contact should be programmed in series with the control circuit (Figure 3-15).

To make a shift register non-retentive, a circuit must be entered for each bit (Figure 3-16).

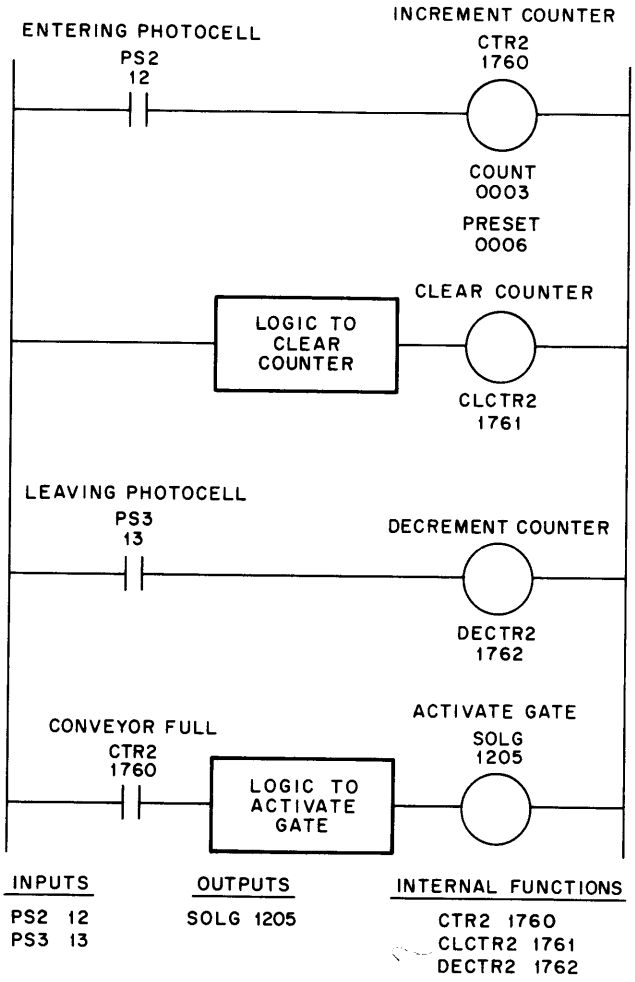



Figure 3-13 Up/Down Counter Circuit and Boolean Equation Parts on a Conveyor to a Maximum of Six

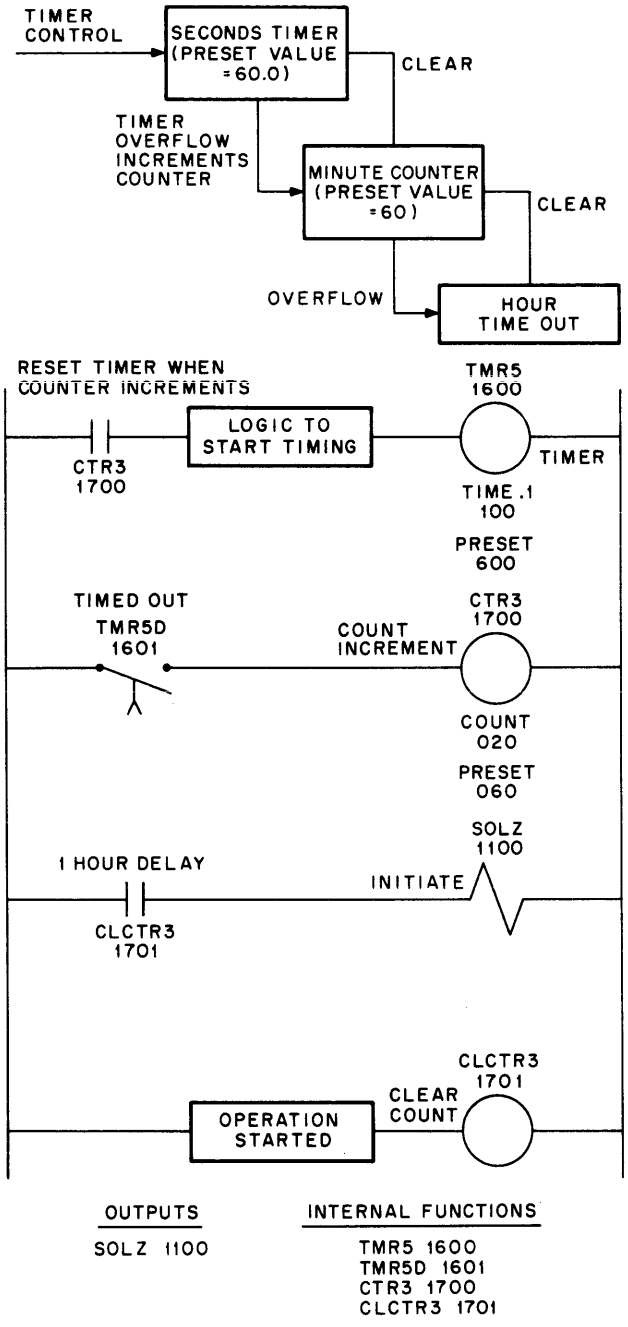



Figure 3-14 Circuit and Boolean Equations to Cascade a 60 Second Timer With a 60 Minute Counter

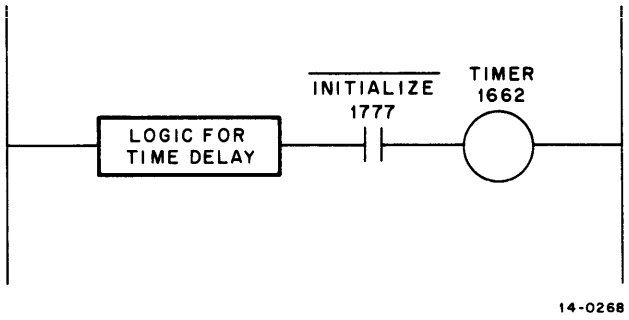



Figure 3-15 Clear Timer if Power is Lost Ladder Diagram

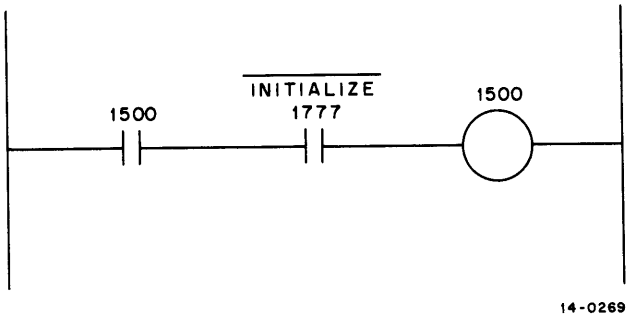



Figure 3-16 Circuit to Clear One Bit of a Shift Register on Power Failure

# CHAPTER 4
# BOOL-143 CONTROL EQUATION TRANSLATOR

Control equations, written with symbols similar to Boolean notation are translated using *BOOL-143,* the *Control Equation Translator,* into the Industrial 14 machine code instructions presented in Chapter 2. A series of these equations constitutes an Industrial 14 program. The program of control equations is prepared using the Editor and a "source program paper tape" is generated. Use of the Editor enables the source program to be corrected and easily revised during the stages of program development. BOOL-143 reads the source program paper tape and translates (compiles) the equations into an Industrial 14 machine-code program containing the instructions in Chapter 2.

The programming procedure used with BOOL-143 is shown in Figure 4-1. The importance of correcting errors in the source program with the Editor cannot be overemphasized. The user should maintain a current, correct version of the program, in equation form, throughout the life of the system. Failure to do this can result in many lost hours if changes must be made to the system and an accurate representation of the program is not available.

### VT14 COMPATIBILITY

The output tape from BOOL-143 can be read into the Industrial 14 and debugged using the VT14 Programming Terminal (Figure 4-1). However, the format of the Industrial 14 program prepared by BOOL-143 must be compatible with VT14 format. A yes (y) response to BOOL-143 query, VT14-NY?, enables BOOL-143 to flag (E69) certain VT14 incompatible commands and generates one NOP instruction between each Boolean equation. These incompatible commands are listed under Error Number 69 in Table 4-3 (Errors Detected and Diagnosed by BOOL-143). BOOL-143 will not flag Boolean equations with circuit diagrams too wide or too high for display on the VT14 screen. Examples of these two types of equations are explained later in this Chapter. Circuits with too many nested variable groups should be avoided.

### BOOL-143 STATEMENT

The body of BOOL-143 source programs is a series of control equations such as the following:

```
1010 = (1 + 2)* 3
```

Each character in the control equation has a special meaning to BOOL-143. For example, the nonprinting carriage return character ( ↵ ) signals BOOL-143 that the complete source statement has been read. The acceptable equation characters and the use of each are listed in Table 4-1; this table includes all legal BOOL-143 characters. Functional descriptions of these characters are presented later in this chapter.

The characters SPACE, TAB, LINE FEED, and FORM FEED may be included in BOOL-143 programs to format the source program. These characters are ignored during processing by BOOL-143. Blank leader/trailer (null) tape is also ignored by BOOL-143. All characters other than these formatting characters and the characters listed in Table 4-1 are illegal and cause errors if included in the statement portion of the BOOL-143 source program. All illegal nonprinting characters are replaced by a question mark (?) in the BOOL-143 listing.

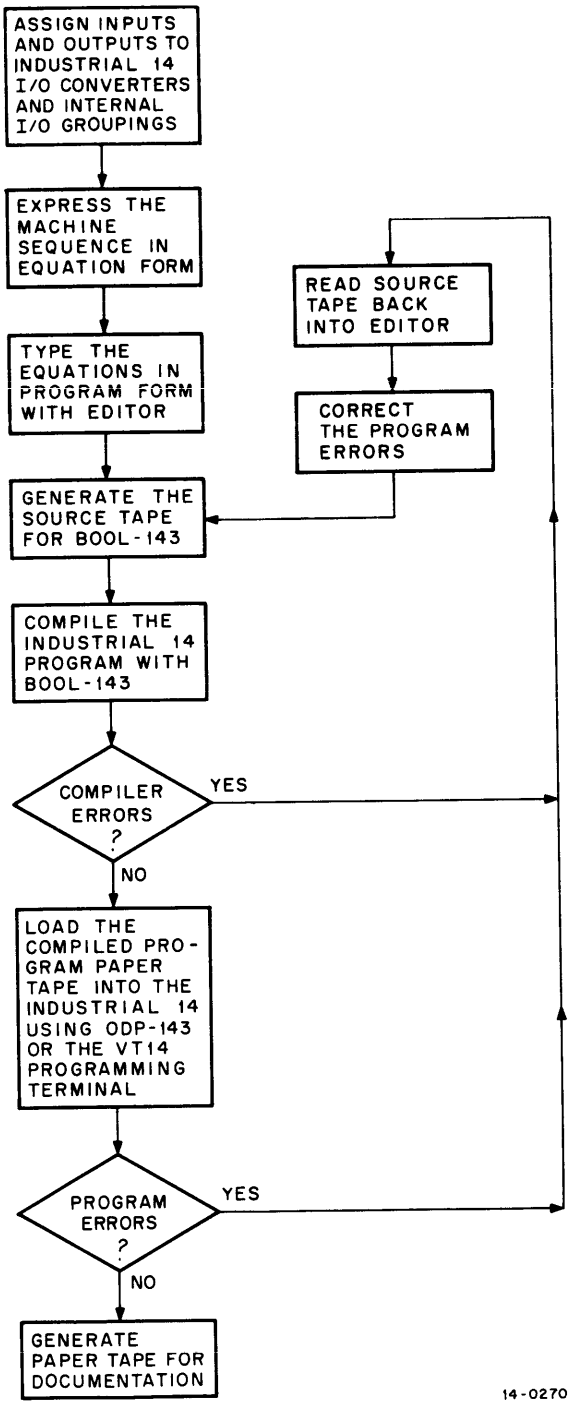



Figure 4-1 BOOL-143 Programming Procedure

**Input, Output, and Internal Function Specification**

Particular input and output numbers are determined by the selection of the input (or output) converters. These are octal numbers between 0–777, and 1000–1377. Internal functions stored in the Industrial 14's memory are designated by octal numbers 1400–1777. The state of an output or an internal function is determined by the state of inputs, outputs, and internal functions. For example, a single output number (1017 for instance) is the left-hand member of an equation, while the right-hand member of the equation is a combination of inputs, outputs, and internal functions. These inputs, outputs, or internal functions are combined as variables of an equation using the operators described in the next paragraph.

To be compatible with old BOOL-14, users can also express inputs as X0–X777, outputs as Y0–Y377 or Y1000–Y1377, and internal functions as Y400–Y777 or Y1400–Y1777.

**Operators**

The BOOL-143 operators are * (Boolean AND), + (Boolean OR, inclusive), and / (Boolean NOT). These operators combine input, output, and internal function values to determine the value of an output or internal function.

Thus, the equation:

1017 = 1017* /4+23

means that output 1017 is set ON if the following conditions are satisfied:

a. Output 1017 is already ON and input 4 is not ON, or

b. Input 23 is ON; otherwise, 1017 is set OFF.

**NOTE**

**The equation specifies the ON and OFF conditions for an output. Specifically, the output is set ON if the conditions are satisifed (true); otherwise it is set OFF (the conditions are false). The fact that an output is set OFF when the conditions are not satisfied is not always expressed explicitly, but is always implied in a BOOL-143 equation.**

The NOT operator (/) can negate the sense of single variables or variable groups. For example the equation:

1021 = 12 + /31

means that output 1021 is set ON if:

a. Input 12 is ON, or

b. Input 31 is OFF; otherwise output 1021 is set OFF.

Table 4-1
BOOL-143 Equation Characters

| Character | Meaning/Use |
|---|---|
| 0–777 or X0–X777 | Input numbers |
| 1000–1377 or Y1000–Y1377 or Y0–Y377 | Output numbers |
| 1400–1777 or Y1400–Y1777 or Y400–Y777 | Internal function numbers |
| = | 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 |

**Variable Groups**

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

```
1012 = (12 + 13) * [27 + 1015].
```

Square brackets and parentheses are treated identically by BOOL-143 and are available for the user's convenience in distinguishing between variable groups; however, BOOL-143 requires that these symbols be used in pairs. Any grouping opened with a square bracket must also be closed with a square bracket. For example:

```
1012 = (12 + 13] * [27 + 1015)
```

will generate an error because each grouping is closed with the wrong bracket type.

The equations:

```
1012 = (12 + 13) * [27 + 1015]
and
1013 = (22 + 1017 * [42 + 3] ) * 15
```

are in proper form.

**Statement Continuation**

Extremely long equations which do not fit on one line in the source program can be broken into two or more parts using the dollar sign ($) as the statement continuation character. Each incomplete line is terminated with a $. The equation is part of a source program paper tape generated with the Editor, in which the user has typed a carriage return after the $ on the source tape. Whenever a $ precedes a carriage return, BOOL-143 does not treat the carriage return as an end of statement indicator. It simply provides a new input line and accepts additional characters until it encounters a carriage return not preceded by a $.

For Example:

```
1021 = / (1 + 2 + 3) * [ (4 + [5*6] )* /7]$
+1012* (27 + 5 + /21)
```

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

**Comments**

Program comments are text lines included in the BOOL-143 source program to clarify the function of an equation or to document the program in general. Comments are merely typed by BOOL-143 as they appear in the source program. The comments have no affect on the machine code generated for an equation.

BOOL-143 comments must be preceded by a semicolon (;). When a semicolon is encountered, all characters to the right are treated as a comment. Any Teletype character can be included in the comment except the $. This character can be used to write comments on more than one line. Any characters to the right of the $ are ignored by BOOL-143.

For Example:

```
1015 = 1 + 2* (3 +/4)      ;SOLA = PB1 + PB2* (LS13$
                            +/LS14)
;LS14 TRIPPED WHEN SLIDE IS FULL FORWARD.
```

In this example the first comment is broken into two lines using the $.

**NOTE**

**BOOL-143 can handle approximately 160 input characters (equation and comment) in one statement. If this maximum is exceeded, BOOL-143 types:**

**BUFFER OVERFLOW**

**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-143 equations:

a. 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 perform this function.

b. R-functions solve for an output or internal function state. R-functions are identical to normal output equations except that provisions exist for solving for the output more than once on each pass through the program.

A BOOL-143 program can contain 64 subroutines (specifically, R- and Z-functions) for each 4K field of the Industrial 14 program.

**Intermediate Results**

In many instances, the storage of an intermediate result is required within Industrial 14 programs. 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. The following are three possible approaches for controlling such functions:

*Complete coding* – Control equations are written, within which, the variables that determine the state of the control relay are substituted for the control relay. Thus, the equation for each solenoid separately tests the control relay inputs to determine the state of the solenoid and 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 Industrial 14 output is set by the Z-function.) The solenoid equations can then contain Z-functions as variables. Only one set of Industrial 14 instructions is needed in solving 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 unwired output or internal function ON or OFF. These outputs can then be used as variables in other equations.

**NOTE**

**Before proceeding, review Subroutine Jump Instructions in Chapter 2.**

**Z-Functions**

Z-functions are repetitive variables which are solved whenever they are needed in normal output equations. They do not set an output to record the result of the testing; instead, the result is recorded by setting the Industrial 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-143:

```
1001 = 1* (21 + 13*/42 + 1017 * 11)
1002 = 2* (21 + 13*/42 + 1017* 11)
1003 = 3 * (21 + 13* /42 + 1017* 11)
1004 = 4* (21 + 13* /42 + 1017 * 11)
```

This set of equations would cause BOOL-143 to generate Industrial 14 machine codes to test each variable for each equation. Many of the instructions would be the same for all four outputs; specifically, those instructions which test the state of the variables within parentheses.

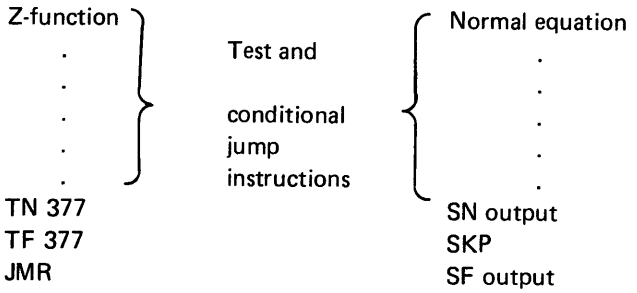
The same machine control function can be represented by the following group of equations with a considerable saving in the number of memory locations required for the program.

```
Z1 = 21 + 13* /42 + 1017* 11
1001 = 1*Z1
1002 = 2*Z1
1003 = 3*Z1
1004 = 4*Z1
```

The new form is analogous 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 can be any decimal number between 0—2047. The Z-function must be defined only once per memory field. It can be referenced as many times as necessary from that same field.

Given this group of equations, BOOL-143 generates the series of instructions to establish Z1 as a subroutine. The Z1 instruction tests the inputs as a normal output equation would; however, no actual output is set. The subroutine sets the TEST flag ON if the equation is true; otherwise, it sets the TEST flag OFF. BOOL-143 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 manner (compared to the termination of a normal equation):




If the result of the Z-function is not true, BOOL-143 generates a JFF or JFN instruction that jumps past the TN 377 and TF 377 to the JMR, thereby returning with the TEST flag OFF. If the result of the Z-function is true, the TN 377 and TF 377 instructions are executed. Because input 377 must be in one of the two states, one of the tests must be true and the TEST flag is therefore set before the JMR is executed.

The instructions for solving equations which contain Z-functions differ greatly from the original equations. In place of the instructions which test variables in the Z-function, BOOL-143 generates a JMS (jump to subroutine) to the beginning location of the subroutine that solves the Z-function. The subroutine returns to the main program with the result shown by the TEST flag. The main program then tests the result with a JFF or JFN instruction; thus, the JMS instructions serve 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 stored in the TEST flag.

The Industrial 14 accommodates only one level of subroutines; therefore, a subroutine cannot contain a JMS instruction. In terms of BOOL-143, this means that a Z-function cannot 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 TN or TF instruction in the subroutine, as described later in this chapter.

**R-Functions**

R-functions (repetitive equations) are subroutines which 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 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. Defining an R-function causes BOOL-143 to generate the equation as a subroutine. Whenever the output state is to be determined, BOOL-143 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 area in memory, but are executed as many times as necessary.

An R-function is defined as having the following form:

```
Rn: m = 1* ....
```

where n is any decimal number between 0—2047 and m specifies a specific output or internal function to set by the subroutine.

The subroutine is referenced by writing the R-function designation. For example, the following statement defines an R-function to set output 1053:

R32: 1053 = 32* (41 + 12* 1013) + 21

BOOL-143 generates the machine code instructions for solving this equation for 1053 in the same form as for a normal output equation.

After the set output instructions, BOOL-143 compiles a JMR instruction as follows:

.
.
.
SN 1053
SKP
SF 1053
JMR

Whenever the R-function is solved in the main program, the user writes the following statement in the source program.

R32

BOOL-143 then compiles a JMS instruction to the subroutine which solves for 1053.

In the following examples, all R- and Z-functions must be defined at the beginning of the source program (not following an output equation). Any R- or Z-function defined following an output equation (1001 = 4 for instance) generates an error. Each R- or Z-function must possess a unique decimal-identifying number.

**SUBROUTINE EXAMPLES**

| | |
|---|---|
| R13:1006 = 2+4 | ;Defines the subroutine |
| . | |
| . | |
| . | |
| R13 | References the subroutine, equivalent to writing 1006 = 2 + 4 |
| Z8 = 2+4*1007 | |
| Z9 = 31+4+12 | |
| Z8 = 21*1007 | Error; the Z8 function has already been defined in this field. |
| 1001 = 21+R8 | Error; the R-functions cannot be variables in equations. (Only Z-functions may.) |
| R21:1010 = Z1*17 | Error; an R-function cannot contain a Z-function as a variable. (A subroutine cannot jump to another subroutine.) |
| Z1 = 1+31*1021 | |
| Z2 = 3*(21*31)+Z1 | Error; the equation for a Z-function cannot contain another Z-function. (A subroutine cannot jump to another subroutine) |
| 1012 = 12*(13+15) | |
| R14:1012 = 15+1013 | Error; definition of all R- and Z-functions must precede any output equations. |

Storage outputs are designated as unused output numbers or retentive memory outputs of the Industrial 14 and are programmed exactly as other outputs except that these outputs do not drive solenoids, motor starters, etc. Once a storage output is set, however, it can be tested with the Industrial 14 TN and TF instructions. A retentive memory will hold its state if power to the controller is lost; an unused output will clear on a power loss.

By using storage outputs to record the test result, the instructions required for solving the result are executed only once for each pass through the program. The storage output itself can then be tested whenever it is needed. However, subroutines must be executed at each instance that the result is required in an output equation.

Storage outputs can 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 cannot jump to a second subroutine.

A storage output must be used in place of a subroutine whenever the control function itself is an element of the equation, (a latched function).

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

1333 = 23*21 + 1017

where output 1333 corresponds to a non-existent output converter. The result is tested by including the output as a variable in an equation.

For example,

1016 = 1333 + (3 + 5 + 7).

BOOL-143 then generates a TN 1333 instruction to solve this equation for 1016.

## CONTROL STATEMENTS

Control statements are directions to BOOL-143 which either affect the machine code instructions generated during compilation or affect the binary output tape generated by BOOL-143. Specifically, control statements affect the allocation of memory locations for BOOL-143 generated Industrial 14 program instructions and mark the end of source program tapes.

### Shift Circuits .SR mmmm, nnnn.

The .SR control statement designates that the following equation is a shift circuit with consecutive shift register bits from mmmm to nnnn.

The shift circuit and shift register bits may consist of external outputs (I/O numbers 1000–1377) or internal functions (normally, I/O numbers 1400–1577). A control statement to establish a shift register is of the form:

```
.SR 1501, 1517
1460 = 200*/370 + 1460
```

The amount of memory required to store a shift register consists of the memory locations to store the shift equation, plus the locations to move each bit of the shift register. The smaller the shift equation, the longer the shift register can be. The smallest shift equation allows approximately 81 shift register bits; the largest shift equation allows approximately 56 shift register bits.

### Field 1 .FLD

BOOL-143 can store the program in either Field 0 or Field 1. The .FLD control statement notifies BOOL-143 that the remainder of the program is to be stored in Field 1, and that all R- and Z-functions defined in Field 0 must be redefined in Field 1, if referenced. If the Industrial 14 program overflows Field 0, BOOL-143 automatically provides a .FLD statement and stores the remainder of the program in Field 1.

### Partition .PRTN = nnnn

The partition control statement enables the user to increase (or decrease) the number of retentive memories and shift register bits by decreasing (or increasing) the number of timers and event counters available. The Industrial 14 partition, set at 1600 when the unit is shipped, defines the first I/O number at which timers and counters may be designated. This partition may be changed by adjusting switches within the Industrial 14 Control unit. This adjustment must be specified to BOOL-143 by entering a .PRTN = nnnn (where nnnn is an octal number ranging from 1400–1760, in multiples of 20 I/O numbers). This partition statement must be entered at the beginning of the Industrial 14 program and should not be changed anywhere within the program. An example of a partition statement is:

```
PRTN 1520
```

where 1520 indicates the I/O number at which timers or counters may be designated. Table 3-2 indicates that there are 80 retentive memories or shift register bits and 80 timers or counters available.

### Timer Preset .SEC mmm and .TSEC mmm

Timers can be preset in seconds or tenths of seconds. The timer preset, .SEC mmm or .TSEC mmm, control statement should be entered on the line preceding its associated timer equation. The preset value is a decimal number from 1–999 seconds or 0.1–99.9 seconds. I/O numbers from 1600–1757 must be preset as timers, beginning at 1600, and at every consecutive even I/O number, up through the last timer; otherwise, these I/O numbers are assumed to be up-counters. Two examples of timer presets follow:

| Timer Preset | Explanation |
|---|---|
| .SEC 360<br>1600 = 200*1600*/1000 | Timer 1600 is preset to 360 seconds |
| .TSEC 305<br>1602 = 1060*/250 | Timer 1602 is preset to 30.5 seconds |

### Counter Preset .CNTR mmm

The event and up/down counter preset statement .CNTR mmm must be entered on the line preceding its associated counter equation. The counter preset value is a decimal number from 1–999. All counter equations must use a higher I/O number than any timer and only the even I/O number of a counter equation should be preset. An example of a counter preset follows:

| Counter Preset | Explanation |
|---|---|
| .CNTR 25<br>1760 = 6*1020 | Up/down counter 1760 is preset to 25 |

**End of Program – .END or .ENDN**

Each BOOL-143 source program must be terminated with an END or ENDN statement to indicate the end of compilation. The .END statement completes the BOOL-143 program by compiling a CLRPC instruction.

The .ENDN (end-no-jump) statement may be used to terminate an incomplete program where no CLRPC instruction is needed. This statement marks the end of the source program, but generates no machine code instruction. Examples of .END and .ENDN statements follow:

| Statement | Instruction Generated |
|---|---|
| .END 50 | Error—no value allowed<br>CLRPC Generated |
| .END | CLRPC |
| .ENDN | Nongenerated |
| .ENDN 50 | Error—no value allowed |

**End of Tape – .EOT**

The BOOL-143 source tape can 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-143 encounters an .EOT statement it types "EOT" and halts the computer system, allowing the user to load the paper tape reader with the next tape. Pressing the CONT switch permits the compilation to proceed.

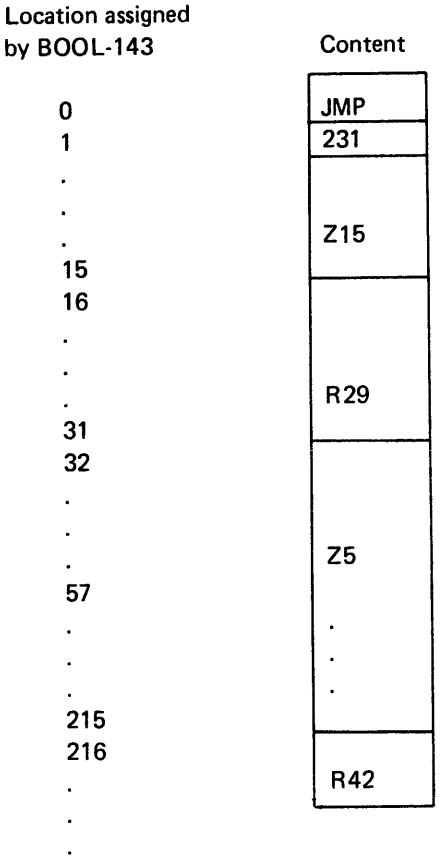
For example, if the program is on three tapes, the end of the tapes is as follows:

| | |
|---|---|
| .EOT ↵ | Tape 1 is terminated by an .EOT statement. |
| .EOT ↵ | Tape 2 is also terminated by an .EOT statement. |
| .END ↵ | Tape 3 is terminated by an .END or an ENDN statement. |

**Memory Allocation**

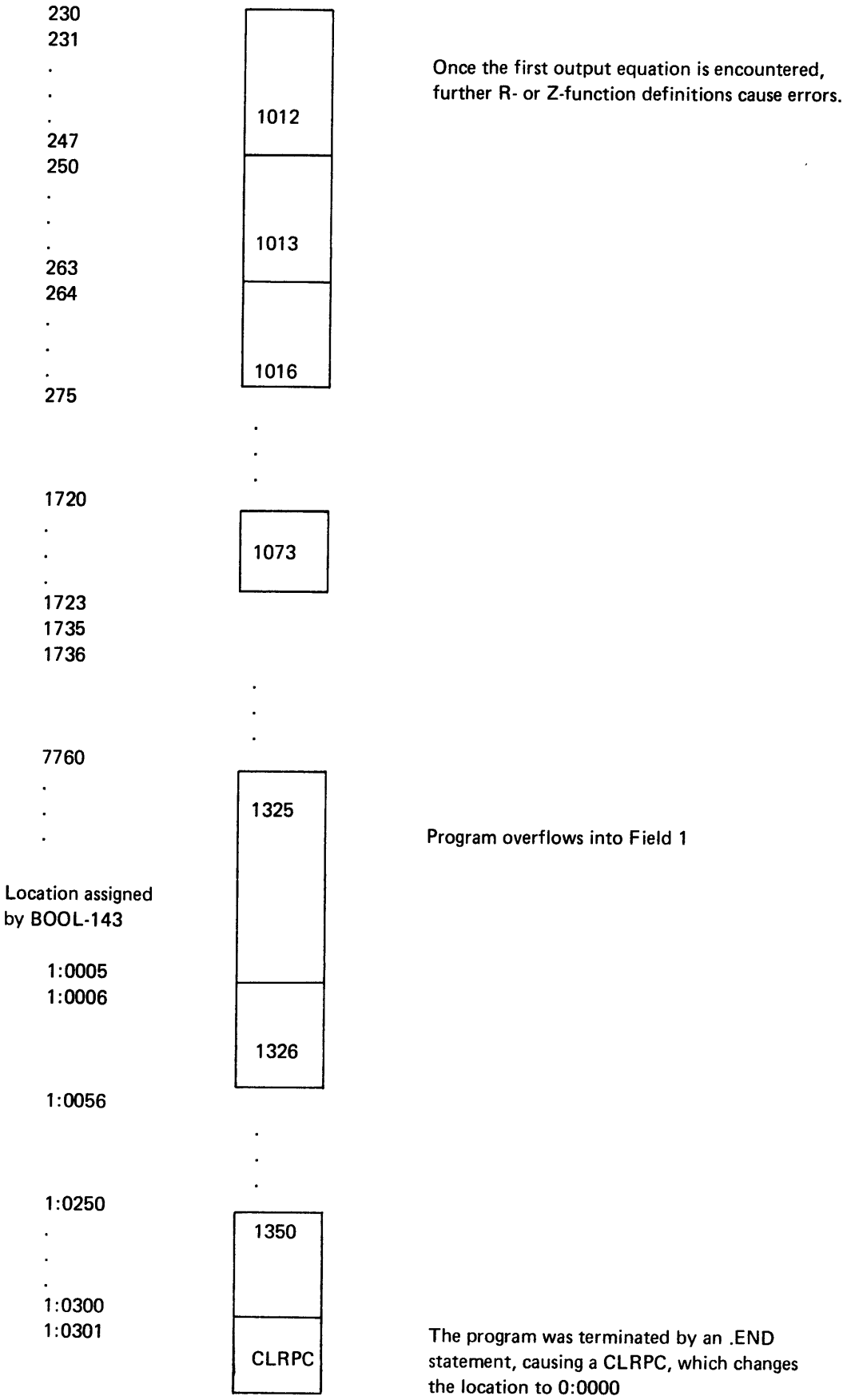
BOOL-143 compiles all subroutines (R- and Z-functions) as the first elements of the program. This allows BOOL-143 to generate JMS instructions to these subroutine locations from the main program. BOOL-143 places a JMP to the start of the main program which follows the subroutines immediately before the subroutines.

An example of memory assignment follows:




The JMP instruction goes to the first instruction of the first output equation (in this case 1012).

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


230
231
247
250
263
264
275
1012
1013
1016
1720
1723
1735
1736
1073
7760
1325
Location assigned
by BOOL-143
1:0005
1:0006
1326
1:0056
1:0250
1:0300
1:0301
1350
CLRPC
Once the first output equation is encountered, further R- or Z-function definitions cause errors.
Program overflows into Field 1
The program was terminated by an .END statement, causing a CLRPC, which changes the location to 0:0000

BOOL-143 assigns 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 space for changes with ODP-143. The .LOC, .FIXS and .VARS control statements, described later, enable 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-143 checks the number of remaining locations on the current page. If insufficient Industrial 14 memory locations exist for the machine code instructions to solve the equation, BOOL-143 compiles NOPs in the remaining locations of the current memory page and starts the equation in the first location of the next page; the NOPs are not typed as part of the listing.

The user must remember that Industrial 14 program execution starts in location 0. The content of the location 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-143 generates machine code instructions for the locations beginning with the value specified by .LOC. For example:

```
.LOC 50
1001 = 1 + 2
```

yields the compiled instructions:

```
0050    4001    TF      0001
0051    2054    JFF     0054
0052    4002    TF      0002
0053    2456    JFN     0056
0054    3001    SN      1001
0055    0010    SKP
0056    1001    SF      1001
```

NOTE

The instructions begin in location 50. If no .LOC statement were given, a .LOC 0 is assumed and the first instruction is placed in location 0.

More than one .LOC statement is permitted in a BOOL-143 program. The value of .LOC can be any octal number between 0–7777.

**Spacing Between Equations – .FIXS or .VARS**

BOOL-143 is directed to compile a specified number of NOP (no operation) instructions after equations by the .FIXS (fix spacing) and .VARS (vary spacing) control statement. The NOP instructions allow for program changes in ODP-143 during debugging. For example, the statement

```
.FIXS 5
```

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

The .VARS control statement affects only the equation which immediately follows it and overrides, for that one equation, any .FIXS statement currently in effect. The .FIXS statement is reinstated after each .VARS 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 0–2047. The .VARS statement can 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 NOPs after the equation |
|---|---|
| Z1 = (27 * 100) * 13 + 4 | none |
| .FIXS 5 | |
| R12: 1001 = 23 * 14 + 0 | 5 |
| .VARS 2 | |
| 1001 = 23 * Z1 | 2 |
| R12 | 5 |
| .FIXS 10 | |
| 1002 = 27 * 3 + 30 *2 | 10 |

The NOP instructions are not part of the compiler listing.

**MONITORING PROVISIONS IN BOOL-143**

Using control statements, BOOL-143 can be directed to automatically supply monitoring instructions if the operation of the Industrial 14 System is to be monitored by an external computer. The instructions used and the techniques involved are described in Chapter 9. The procedure involved in BOOL-143 to generate these instructions follows.

BOOL-143 can be directed to add instructions to monitor the transition state of an output. Specifically, instructions can be added to monitor OFF to ON transitions, ON to OFF transitions, or both, for any Industrial 14 output, internal function, or R-function.

The user chooses the RDBIT or RDMEM instruction for monitoring. The RDBIT instruction outputs a specific 12-bit word to the monitoring computer; the word is determined by the Industrial 14 (Chapter 9). RDMEM is a two-location instruction which outputs a 12-bit word specified by the user.

**Monitor Transitions to ON – .MN**

The .MN control statement causes BOOL-143 to add instructions to the first following output equation to monitor transitions from OFF to ON. These instructions load a 12-bit word into the Industrial 14 output register only when the output transitions from OFF to ON; no 12-bit word is loaded when the output remains the same, or when the output transitions from ON to OFF.

Example:

```
.MN
1005 = /2 * (3 + 1005)
(the instructions generated include a RDBIT 1005)
```

Only the equation immediately following the .MN control statement contains the sequence of monitoring instructions. The instructions generated are described in Chapter 9, Monitoring the Industrial 14.

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

Example:

```
.MN 4123
1005 = /2 * (3 + 1005)
(instructions generated include a RDMEM 4123)
```

If the user supplies less than four digits following the control statement, BOOL-143 assumes leading zeros.

**Monitor Transitions to OFF – .MF**

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

Example:

```
.MF
1006 = 3 * (4 * 5 + /10 * 11)
(the instructions generated include a RDBIT 1006)
```

The .MF control statement shown has effect only for the first following equation (1006).

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

Example:

```
.MF 103
1006 = 3 * (4 * 5 + /10 * 11)
(the instructions generated include a RDMEM 0103)
```

**Monitor All Transitions – .MFN**

The .MFN control statement is used to monitor all transitions in output state (specifically both ON to OFF and OFF to ON). Instructions are added which load a 12-bit word into the Industrial 14 output register on all transition states of the output. The output register is not loaded if the output remains ON or OFF.

Example:

```
.MFN
1007 = 21 + 15 * 1005
(the instructions generated include two RDBIT 1007
instructions)
```

If the output is to be monitored with RDMEM instructions, the user must supply two octal numbers following the .MFN statement.

Example:

```
.MFN 2007, 1007
(the instructions generated include:
RDMEM 2007 – Executed on transition to OFF
RDMEM 1007 – Executed on transition to ON)
```

**NOTE**

**The first number is used in transitions to OFF (.MFN); the second number is used in transitions to ON (.MFN).**

**Monitoring Input Transitions**

As previously stated, BOOL-143 allows monitoring of output states. If the user desires to monitor transition states

of an input, he must assign a storage output to record the previous state of the input. He can then monitor the transition states of the storage output to provide monitoring information concerning the input.

Example:

```
.MF
1040 = 1
(the instructions generated include a RDBIT 1040)
```

In this example, the transition states of output 1040 are used to monitor the transition state of input 1. Table 4-2 provides a summary of control statements and their meanings.

**Table 4-2**
**Summary of Control Statements**

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

**Table 4-2 (Cont)**
**Summary of Control Statements**

| Command | Control Operation |
|---|---|
| .ENDN | End of program – do not compile any CLRPC instruction. |
| .SR 1560, 1565 | The Boolean equation on the next line is a shift circuit whose consecutive shift register bits are from I/O number 1560–1565. |
| .FLD | The following Boolean equations will be stored in Field 1. All R- and Z-functions that will continue to be referenced should be redefined before any output or internal function equations are translated. |
| .PRTN 1660 | The Partition designating the starting address of timers and counters has been adjusted to I/O number 1660. |
| .SEC 120 | Preset the timer equation referenced on the next line to 120 seconds. |
| .TSEC 104 | Preset the timer equation referenced on the next line to 10.4 seconds. |
| .CNTR 160 | Preset the counter (up or up/down) equation referenced on the next line to 160 counts. |

## ERROR MESSAGES

Table 4-3 contains a list of all errors that can be detected and diagnosed by BOOL-143. The error numbers are typed under the associated equation in the form "Ennnn" where nnnn is the error number. All errors stop compilation of the erroneous equation; BOOL-143 then proceeds to the next equation.

No machine code instructions are generated for erroneous equations, unless they are first corrected.

Table 4-3
Errors Detected and Diagnosed by BOOL-143

| Error Number | Cause |
|---|---|
| 1 | A required number is missing after a Z-function, R-function, or control statement.<br><br>Examples:<br>Z = 3 + 1001<br>1017 = 5 * Z<br>R: 1012 = 27 * 1<br>.FIXS |
| 2 | A non-octal digit is encountered after an .END, .LOC. |
| 3 | Illegal number error caused by: 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 must be an octal number or a y.<br><br>Examples:<br>B = 1<br>/Y14 = 27 * 1003 |
| 5 | The equal sign (=) is either missing or is not in the proper place within an equation, Z-function, or R-function.<br><br>Examples:<br>100<br>Y1 1 * 2<br>10<br>15/ = 3 |
| 6 | A variable grouping is not closed with the character type with which it was opened.<br><br>Examples:<br>[1 + 2)<br>[3 + (5 * 6] + 1007) |

Table 4-3 (Cont)
Errors Detected and Diagnosed by BOOL-143

| Error Number | Cause |
|---|---|
| 7 | The character following a negation sign (/) is not valid.<br><br>Examples:<br>1023 = /B1 * 2 |
| 8 | The character following a left parenthesis or left bracket is not valid.<br><br>Example:<br>1010 = (R5 * 2) |
| 9 | The character following an AND (*) or OR (+) sign is not valid.<br><br>Examples:<br>1011 = 1 + *<br>1012 = 10 * R1 |
| 10 | The character following an equal sign (=) is not valid.<br><br>Example:<br>1021 = R1 |
| 11 | The character following a number is not valid.<br><br>Example:<br>1032 = 1/ * 2 |
| 12 | The character following a right parenthesis or right bracket is not valid.<br><br>Example:<br>1007 = 12 * (5 + 7) / * 2 |
| 13 | A Z-function is contained as a variable in an R-function or Z-function.<br><br>Examples:<br>Z29 = 3 + Z50<br>R10: 73 = 1 * Z12 |

Table 4-3 (Cont)
Errors Detected and Diagnosed by BOOL-143

| Error Number | Cause |
|---|---|
| 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.<br><br>Example:<br>1037 = 1 * (2 * 24 + 25] * 4) |
| 16 | An equation contains fewer right parentheses and brackets than left parentheses and brackets.<br><br>Example:<br>1042 = 12 * 2 * (2 * [23 + 17] * 14 |
| 17 | A control statement is not recognized.<br><br>Examples:<br>.MZN<br>.MNF<br>.SAZ |
| 18 | A control statement contains an illegal character.<br><br>Examples:<br>.M#N<br>.M.N |
| 19 | No control statement follows the period (.). |
| 20 | At least one .LOC statement must precede all equations, R-functions and Z-functions.<br><br>Examples:<br>Z12 = 1 * 2<br>.LOC 50 |
| 21 | A subroutine definition is encountered after at least one equation has been compiled.<br><br>Examples:<br>1012 = 3 * 15<br>R12: 1005 = 27 * /51 |

Table 4-3 (Cont)
Errors Detected and Diagnosed by Bool-143

| Error Number | Cause |
|---|---|
| 22 | The statement defines an R-function or Z-function which has been defined previously.<br><br>Examples:<br>Z129 = 25 + 3 * 12<br>Z129 = 1003 * 21 + 5 * 12 |
| 23 | The source program contains too many R- and Z-functions per memory field. The maximum is 64 R-functions and Z-functions per 4K memory field. |
| 24 | The character following a colon (:) is not valid.<br><br>Examples:<br>R15: Z5 = 27 * 12<br>R29: R5 = 121 + 1013 * 122 |
| 48 | Shift register (containing shift equation plus locations to move each bit of the shift register) is too large. |
| 49 | Partition Address is bad: (Address is greater than 1760 or less than 1400, or I/O address is not an octal multiple of 20 between I/O number range (1400–1760).<br><br>Example:<br>.PRTN 1630<br>.PRTN 1360 |
| 50 | Preset of counters (CNTR) or timers (SEC or TSEC) is greater than 999.<br><br>Example:<br>.TSEC 1200<br>1630 = 1 + (2 *1200) |
| 51 | Preset counter or timer at an address below designated partition address.<br><br>Example:<br>PRTN 1640<br>.<br>.<br>.<br>.<br>.CNTR 60<br>1600 = 40 + 41 * (1200 + 1201). |

**Table 4-3 (Cont)**
**Errors Detected and Diagnosed by BOOL-143**

| Error Number | Cause |
|---|---|
| 52 | Preset timer at an up/down counter address or at initialize address 1777.<br><br>Example:<br>.SEC 60<br>1760 = 100 + 101 * 1001 |
| 53 | Shift register limits bad. (Upper I/O address limit is lower than lower I/O address limit).<br><br>Example:<br>.SR 1457, 1451<br>1400 = 200 + /370 + 1400 |
| 54 | Wrong control statements preceding shift circuit equation. (6 .MF, .MN, .MFN) |
| 69 | All Z- and R-function equations and references, monitoring transition commands, and statements controlling spacing between equations (e.g., .FIXS and .VARS) are incompatible with the VT14.<br><br>Examples:<br>.MF<br>.MN<br>.MFN<br>Z4=1+2<br>.VARS5<br>.FIXS10<br>1003=Z2*X10<br>R5:Y2=1+3+5<br>R3 |

## OPTIMUM FORM FOR BOOL-143 EQUATIONS

Although equations can be written in many forms, the optimum form for BOOL-143 equations is achieved by observing the following rules. Rules 1 and 2 affect the number of memory locations used to solve the equation. Rules 3–6 affect the amount of time required to execute the program. Rules 7 and 8 must be followed if Boolean equations are to be VT14 compatible. These rules are not flagged by responding positively to the "VT14 YN?" query.

Adherence to these rules is not mandatory for proper Industrial 14 operation. The rules serve only to increase the efficiency and speed of program execution in the Industrial 14 and provide compatibility with the VT14 programmable controller.

**Rule 1** – All single variables (inputs or outputs) combined with the same operator, should be grouped.

Examples:
1001 = 1 + 2 + (3 * 4 * 5) + 1012 + 15

should be rewritten:

1001 = 1 + 2 + 1012 + 15 + (3 * 4 * 5)
and
1012 = 3 * 1007 * (4 + 27 + 21) * 1005 * 32

should be rewritten:

1012 = 3 * 1007 * 1005 * 32 * (4 + 27 + 21).

**Rule 2** – BOOL-143 generates a test instruction for every variable occurrence on the right-hand side of the equation; therefore, equations should be factored to eliminate repeated variable occurrences.

Examples:
1015 = 3 * 1005 + 6 * 1005 + 12 * 1005 + 1017 + 1005

should be rewritten:

1015 = 1005 * (3 + 6 + 12 + 1017)

**Rule 3** – Where a single variable is ANDed with an expression, the single variable is placed before the expression.

Example:
1006 = (31 + 4 * 52) * 27

should be rewritten:

1006 = 27 * (31 + 4 * 52)

**Rule 4** – Variables and expressions should be grouped in order of complexity, with single variables at the beginning of equations and the large and more complex expressions at the end.

Example:

1031 = [ (5 * 7 + 3) * 21 + (1052 * 1012) +
(13 * 1017 * / 1005) ] * 22

should be rewritten:

1031 = 22*[ (1052*1012) + (13*1017 */1005) +21*
(3+5*7) ]

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

Example:
1005 = (12 * 4 + /27) + (21 * [3 + /51] ) + (1007 * /23)

true 30% of time | true 20% of time | true 50% of time

should be rewritten:

1005 = (1007 * /23) + (/27 + 12 * 4) + (21 * [3 + /51] )

**Rule 6** — Subroutines should be grouped near the end of equations or expressions. Subroutines take longer to test than a single variable.

Example:
1011 = (Z3 * /21) + ( [Z7 + 4] * 3 * 5) + 1 * 2

should be rewritten:

1011 = 1 * 2 + (/21 * Z3) + (3 * 5 * [4 + Z7] )

**Rule 7** — Boolean equations whose circuit diagrams are too wide when displayed on the VT14 screen must be divided into two equations. The first equation drives a storage output and the second equation senses the storage output and drives the original output.
Example:
1001 = 1*2*3*4*5*6*7*10*11*(12+13) * (14+15)

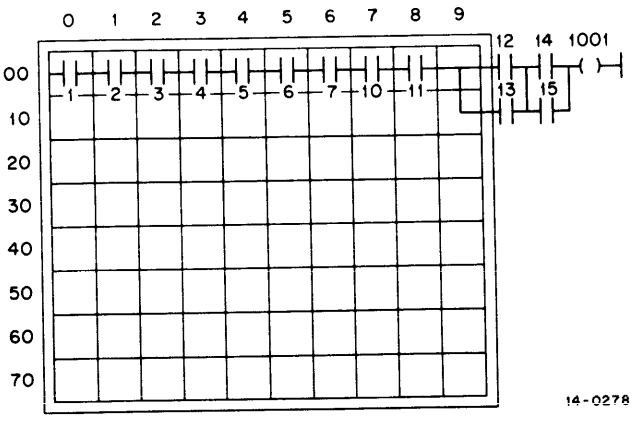



Figure 4-2 VT14 Screen (Circuit Diagram Too Wide for Display)

Solution: 1376 = 1*2*3*4*5*6*7*10*11
1001 = 1376* (12+13) * (14+15)

**Rule 8** — Boolean equations with circuit diagrams too high for display on the VT14 screen must be divided into two equations; again the first equation drives a storage output and the second equation senses the storage output and drives the original output.

Example:
1002 = 20* (21+22+23+24+25+26+27+30+31)

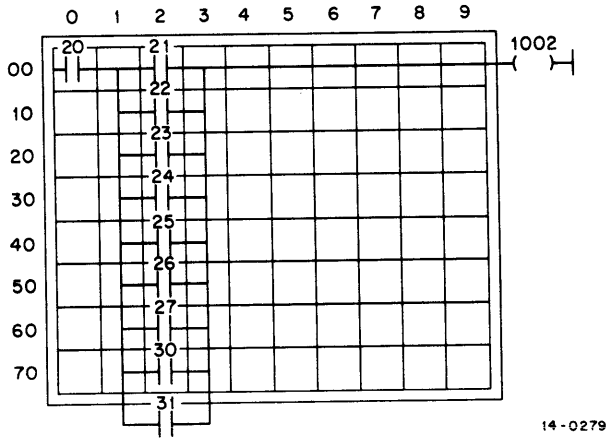



Figure 4-3 VT14 Screen (Circuit Diagram Too High for Display)

Solution: 1375 = 21+22+23+24+25+26+27+30
1002 = 20* (1375 + 31)

If the foregoing rules sometimes appear contradictory, remember that they are presented only as guidelines to assist in writing equations in the form which permits the Industrial 14 to determine as quickly as possible whether an output is to be set ON or OFF. The rules attempt to limit the amount of testing the Industrial 14 must perform before it can make the ON/OFF decision.

## SAMPLE BOOL-143 PROGRAM

A BOOL-143 program representing a simple control circuit follows. Comments are used to indicate the physical input or output which is represented. The control statements .LOC, .FIXS, .VARS, and .END are illustrated. Z-functions and R-functions are included with the regular output and internal equations. The output listing from BOOL-143 follows the program. BOOL-143 output is generated in 8-1/2 X 11 page format. Each page has a heading and a dashed line to indicate where the pages should be cut apart.

BOOL-143/8 Output Listing

```
;BOOL14/8 SAMPLE PROGRAM
;COMMENTS TO IDENTIFY THE PROGRAM WOULD APPEAR HERE.
;CONTROL STATEMENTS WHICH AFFECT THE WHOLE PROGRAM FOLLOW.
.PRTN 1600
.LOC250
.FIXS5
;SUBROUTINES PRCEED THE MAIN PROGRAM.
;2CR=/6PB*/7PB*/1PS*/1FS*/5CR
Z2=/13*/14*/6*/7*/1011
;SOLE=3CR=(8PB*CRH+CRA)*7CR*8CR*9CR*/3LS*/4LS*/5LS
R1:1004=(15*1007+1006)*1012*1013*/1014*/2*/3*/4
;SOLF=4CR=/7CR*(CRA*5CR+/9CR+9PB*CRH)
R2:1005=/1012*(1006*1011+1014+16*1007)
;NORMAL EQUATIONS FOLLOW.
;2LT=CRA=RESET*/CRH*(AUTO+CRA)
1006=10*1007*(11+1006)
;CALL TO REPETITIVE EQUATIONS.
.VARS0
R1
R2
;NORMAL EQUATIONS CONTINUE BELOW.
;3LT=CRH=RESET*(MAN+CRH)
1007=10*(12+1007)
;SOLA=SOLC=CR1=/6CR*(6PB*7PB*2CR*/1LS*/2LS*5LS*6LS*7CR+1CR)
1000=/1003*(13*14*Z2*/0*/1*4*5*1012+1000)
1002=0
;4LT=/6PB*/7PB*(1PS+1FS)
1010=/13*14*(6+7)
;5LT=5CR=3LS*/4LS+5CR*8CR
1011=2*/3+1011*1013
;SOLB=SOLD=6CR=7CR*(CRA*/5LS*/6LS*(6CR+5CR*8CR)+PB10*CRH)
1002=1012*(1006*/4*/5*(1002+1011*1013)+17*1007)
;6LT=7CR=4LS
1012=3
;7LT=8CR=1LS*2LS
1013=0*1
;8LT=9CR=/11PB*(9CR+12PB)*(7CR+8CR)
1014=/20*(1014+21)*(1012+1013)
.SEC 60
;TMR1 =1FS*2LT*4LT*/6LS*(2LS+3LS+4LS)
1600=7*1006*1010*/5*(1+2+3)
;SOLG=TMROT1
1016=1601
.SR 1401,1403
;SHFT1=(SOLA+SOLB+SOLF)*1FS
1400=(1000+1002+1014)*7
;CALL TO REPETITIVE EQUATIONS.
.VARS0
R1
R2
.END
```

```
SRC-LH?L
BIN-NLH?L
LST-NLH?L
VT14-NY?N

TURN PUNCH ON

BOCL-143/8   V0  PAGE 0001

;BCOL14/8 SAMPLE PROGRAM
;COMMENTS TO IDENTIFY THE PROGRAM WOULD APPEAR HERE.
;CONTROL STATEMENTS WHICH AFFECT THE WHOLE PROGRAM FOLLOW.
.PRTN 1600
.LOC250
.FIXS5
;SUBROUTINES PRCEED THE MAIN PROGRAM.
;2CR=/6PB*/7PB*/1PS*/1FS*/5CR
Z2=/13*/14*/6*/7*/1011

    0252 6013 TN       0013
    0253 6014 TN       0014
    0254 6006 TN       0006
    0255 6007 TN       0007
    0256 7011 TN       1011
    0257 2662 JFN      0262
    0260 4377 TF       0377
    0261 6377 TN       0377
    0262 0054 JMR

;SCLE=3CR=(8PB*CRH+CRA)*7CR*8CR*9CR*/3LS*/4LS*/5LS
R1;1004*(15*1007+1006)*1012*1013*/1014*/2*/3*/4

    0270 4015 TF       0015
    0271 5007 TF       1007
    0272 2275 JFF      0275
    0273 5006 TF       1006
    0274 2706 JFN      0306
    0275 5012 TF       1012
    0276 5013 TF       1013
    0277 7014 TN       1014
    0300 6002 TN       0002
    0301 6003 TN       0003
    0302 6004 TN       0004
    0303 2706 JFN      0306
    0304 3004 SN       1004
    0305 0010 SKP
    0306 1004 SF       1004
    0307 0054 JMR
```

```
;SCLF=4CR=/7CR*(CRA*5CR+/9CR+9PB*CRH)
R2!1005*/1012*(1006*1011+1014+16*1007)

    0315 7012 TN       1012
    0316 2731 JFN      0331
    0317 5006 TF       1006
    0320 5011 TF       1011
    0321 2327 JFF      0327
    0322 5014 TF       1014
    0323 2327 JFF      0327
    0324 4016 TF       0016
    0325 5007 TF       1007
    0326 2731 JFN      0331
    0327 3005 SN       1005
    0330 0010 SKP
    0331 1005 SF       1005
    0332 0054 JMR

;NORMAL EQUATIONS FOLLOW.
;2LT=CRA=RESET*/CRH*(AUTO+CRA)
1006=10*1007*(11+1006)
```

---



```
    0250 0024 JMP
    0251 0340 0340

    0340 4010 TF       0010
    0341 5007 TF       1007
    0342 2751 JFN      0351
    0343 4011 TF       0011
    0344 2347 JFF      0347
    0345 5006 TF       1006
    0346 2751 JFN      0351
    0347 3006 SN       1006
    0350 0010 SKP
    0351 1006 SF       1006

;CALL TO REPETITIVE EQUATIONS.
.VARS0
R1

    0357 0124 JMS
    0360 0270 0270

R2

    0361 0124 JMS
    0362 0315 0315
```

```
;NORMAL EQUATIONS CONTINUE BELOW,
;3LT=CRH=RESET*(MAN+CRH)
1007=10*(12+1007)

    0400 4010 TF       0010
    0401 2410 JFN      0010
    0402 4012 TF       0012
    0403 2006 JFF      0006
    0404 5007 TF       1007
    0405 2410 JFN      0010
    0406 3007 SN       1007
    0407 0010 SKP
    0410 1007 SF       1007

;SCLA=SOLC=CR1=/6CR*(6PB*7PB*2CR*/1LS*/2LS*5LS*6LS*7CR+1CR)
1000=/1003*(13*14*22*/0*/1*4*5*1012+1000)

    0416 7003 TN       1003
    0417 2440 JFN      0040
    0420 4013 TF       0013
    0421 4014 TF       0014
    0422 2434 JFN      0034
    0423 0124 JMS
    0424 2252 0252
    0425 2034 JFF      0034
    0426 6000 TN       0000
    0427 6001 TN       0001
    0430 4004 TF       0004
    0431 4005 TF       0005
    0432 5012 TF       1012
    0433 2036 JFF      0036

--------------------------------------------------------------------

BOCL-143/8   V0  PAGE 0003

    0434 5000 TF       1000
    0435 2440 JFN      0040
    0436 3000 SN       1000
    0437 0010 SKP
    0440 1000 SF       1000

1002=0

    0446 4000 TF       0000
    0447 2452 JFN      0052
    0450 3002 SN       1002
    0451 0010 SKP
    0452 1002 SF       1002
```

```
;4LT=/6PB*/7PB*(1PS+1FS)
1010=/13*14*(6+7)

     0460 6013 TN       0013
     0461 4014 TF       0014
     0462 2471 JFN      0071
     0463 4006 TF       0006
     0464 2067 JFF      0067
     0465 4007 TF       0007
     0466 2471 JFN      0071
     0467 3010 SN       1010
     0470 0010 SKP
     0471 1010 SF       1010

;5LT=5CR=3LS*/4LS+5CR*8CR
1011=2*/3+1011*1013

     0477 4002 TF       0002
     0500 6003 TN       0003
     0501 2105 JFF      0105
     0502 5011 TF       1011
     0503 5013 TF       1013
     0504 2507 JFN      0107
     0505 3011 SN       1011
     0506 0010 SKP
     0507 1011 SF       1011

;SCLB=SOLD=6CR=7CR*(CRA*/5LS*/6LS*(6CR+5CR*8CR)+PB10*CRH)
1002=1012*(1006*/4*/5*(1002+1011*1013)+17*1007)

     0515 5012 TF       1012
     0516 2535 JFN      0135
     0517 5006 TF       1006
     0520 6004 TN       0004
     0521 6005 TN       0005
     0522 2530 JFN      0130
     0523 5002 TF       1002
     0524 2133 JFF      0133
     0525 5011 TF       1011
     0526 5013 TF       1013
     0527 2133 JFF      0133
     0530 4017 TF       0017
     0531 5007 TF       1007
     0532 2535 JFN      0135
     0533 3002 SN       1002
     0534 0010 SKP
     0535 1022 SF       1002
```

---

```
;6LT=7CR=4LS
1012=3

     0543 4003 TF       0003
     0544 2547 JFN      0147
     0545 3012 SN       1012
     0546 0010 SKP
     0547 1012 SF       1012

;7LT=8CR=1LS*2LS
1013=0*1

     0555 4000 TF       0000
     0556 4001 TF       0001
     0557 2562 JFN      0162
     0560 3013 SN       1013
     0561 0010 SKP
     0562 1013 SF       1013

;8LT=9CR=/11PB*(9CR+12PB)*(7CR+8CR)
1014=/20*(1014+21)*(1012+1013)

     0570 6020 TN       0020
     0571 2604 JFN      0204
     0572 5014 TF       1014
     0573 2176 JFF      0176
     0574 4021 TF       0021
     0575 2604 JFN      0204
     0576 5012 TF       1012
     0577 2202 JFF      0202
     0600 5013 TF       1013
     0601 2604 JFN      0204
     0602 3014 SN       1014
     0603 0010 SKP
     0604 1014 SF       1014

;SEC 60
;TMR1 =1FS*2LT*4LT*/6LS*(2LS+3LS+4LS)
1600=7*1006*1010*/5*(1+2+3)

     0612 4007 TF       0007
     0613 5006 TF       1006
     0614 5010 TF       1010
     0615 6005 TN       0005
     0616 2627 JFN      0227
     0617 4001 TF       0001
     0620 2225 JFF      0225
     0621 4002 TF       0002
     0622 2225 JFF      0225
     0623 4003 TF       0003
     0624 2627 JFN      0227
     0625 3600 SN       1600
     0626 0010 SKP
     0627 1600 SF       1600
```

```
;SCLG=TMROT1
1016=1601

    0635 5601 TF       1601
    0636 2641 JFN      0241
    0637 3016 SN       1016
    0640 0010 SKP

-------------------------------------------------------------

BOOL-143/8   V0  PAGE 0005

    0641 1016 SF       1016

;SR 1401,1403
;SHFT1=(SOLA+SOLB+SOLF)*1FS
1400=(1000+1002+1014)*7

    0647 5000 TF       1000
    0650 2255 JFF      0255
    0651 5002 TF       1002
    0652 2255 JFF      0255
    0653 5014 TF       1014
    0654 2661 JFN      0261
    0655 4007 TF       0007
    0656 2661 JFN      0261
    0657 2263 JFF      0263
    0660 0010 SKP
    0661 1400 SF       1400
    0662 2274 JFF      0274
    0663 7400 TN       1400
    0664 3400 SN       1400
    0665 2674 JFN      0274
    0666 0133 MOVBIT
    0667 1402 1402
    0670 1403 1403
    0671 0133 MOVBIT
    0672 1401 1401
    0673 1402 1402
    0674 0000 NOP
```

```
;CALL TO REPETITIVE EQUATIONS.
;VARS0
R1

    0702 0124 JMS
    0703 0270 0270

R2

    0704 0124 JMS
    0705 0315 0315

;END

    0713 0004 CLRPC

ERROR LINES: 0000

LAST LOCATION:  0713
```

# CHAPTER 5
# SET-143
# SYMBOLIC EQUATION TRANSLATOR

Industrial 14 programming, using BOOL-143, requires that the control equations be written in terms of input numbers 0 = 777 (X0-X777 optional), output numbers 1000–1377 (Y0–Y377 or Y1000–Y1377 optional) and internal function numbers 1400–1777 (Y400–Y777 or Y1400–Y1777 optional). Input numbers are obtained by assigning inputs (pushbuttons, limit switches, etc) to input converters and selecting the corresponding input numbers from a chart. A similar procedure is followed for assigning output numbers. Internal Function* numbers are assigned according to function (retentive memory, shift register, timer and counter). The user then translates control expressions in terms of input and output physical components and internal function types.

Because some find this translation cumbersome, and because the translation is subject to human error, an optional preprocessor for BOOL-143 is available. This preprocessor, designated as *SET-143* (for *Symbolic Equation Translator),* accepts symbolic equations in terms meaningful to BOOL-143 and the Industrial 14 System. The translation is performed according to a user-supplied symbol table, which provides the I/O number equivalents of the equation elements.

SET-143 is completely optional. The user may choose to work directly with BOOL-143, performing any necessary translations. SET-143 uses the computer to perform and document the translation from meaningful user symbols to the I/O numbers required by BOOL-143 and the Industrial 14. The SET-143 output listing is a useful system document, especially from a maintenance point of view. The input tape to SET-143 is prepared with the PDP-8 Editor program; this input tape can be saved for editing when future developments change the control equations. The output tape from SET-143 can also be corrected with the PDP-8 Editor.

*Chapter 3 contains details concerning Internal Function I/O assignment.

**Hardware Requirements**

SET-143 translates an Industrial 14 program of any length using an 8K PDP-8 family computer. SET-143 utilizes low-speed Teletype or high-speed paper tape input and output. No provision exists in SET-143 for keyboard input.

**Symbols**

Symbols in a SET-143 equation must be unique and are limited to 6 characters. Each character must be either a letter (A–Z) or a number (0–9); no other characters are allowed within a symbol. A symbol can begin with either a number or a letter. Some sample symbols follow; many others are possible.

| | | | |
|---|---|---|---|
| LS12 | BCKLS | NEXT | ST2MAN |
| SPINMT | FUNDEP | CHECK | MIXRUN |
| 15PB | AUTOPB | RUNLT | LIQFUL |
| SOLA | 157TMR | ST1ON | 15STOP |

The following are examples of illegal symbols:

| | |
|---|---|
| MANUALPB | More than 6 characters |
| STA#1 | Illegal character within the symbol |

**Symbol Definition**

SET-143 uses symbol definitions to associate symbolic names with the I/O number (X, Y optional), R, and Z values required by BOOL-143. The user can assign any legal symbol to an I/O number in the range 0–1777 octal. The following are typical assignments:

| | |
|---|---|
| LS12 = 10 | NEXT = 0 |
| SPINMT = 57 | RUNLT = 1200 |
| FULDEP = 123 | MIXRUN = 1023 |
| AUTOPB = 50 | LIQFUL = 12 |

The symbol must be followed by an equal sign, and the equal sign must be followed by an I/O number (X or Y optional), R or Z. Spaces are allowed within the statement and have no affect on the processing.

SET-143 does not permit the user to assign a symbol to an I/O number value that contains an 8 or 9, or which is greater than 1777. In addition, SET-143 does not permit the same symbol to be defined for two different values (for example, AUTO = 1 and AUTO = 321). In these instances, SET-143 generates error messages.

A symbol can be defined as equal to either a BOOL-143 R- or Z-function. The value of the R- or Z-function can be any decimal number from 0–511.

**Symbol Table**

The SET-143 symbol table contains definitions of user's symbols in terms of I/O numbers 0–1777, X, Y, Z or R numbers. These symbol definitions allow equations to be written in a format other than the I/O number format required by BOOL-143. The program may contain several symbol tables in which symbols are defined in terms of I/O numbers, Z-values, or R-values. All SET-143 symbol tables must be preceded by a number sign (#) as the first character of a line. The # sign must also appear at the end of the symbol table. Other characters on the line containing the number signs are ignored and may be used for comments.

A sample symbol table is in Figure 5-1.

The symbol table should be organized as shown in Figure 5-1 with inputs, outputs, R-functions, and Z-functions separated. This separation simplifies checkout and limits the possibility of duplicating assignments. The symbol table is especially useful when the inputs and outputs are listed by numerically increasing input and output values. The wiring of inputs and outputs to the Industrial 14 input and output converters can then be checked against this list. Comments, preceded by a semicolon, can also be used to further identify the input or output assignments.

The symbol table must be separated from the rest of the program with number signs (#) both preceding and following the symbol table. In particular, BOOL-143 control statements (LOC, VARS, etc.) or symbolic equations should not appear within the two number signs.

The symbol table permits the definition of 1088 input, output, internal function, Z- and R-functions combined. If the program contains more than 1088 symbols (an extremely unlikely eventuality) the program must be translated by SET-143 in two sections. Each part of the program must contain the definition for each symbol used in that section.

```
#START OF SYMBOL TABLE
;                  TABLE OF OUTPUT VALUES
SS1=                  1000              ;START SPINDLE MOTOR 1
SS2=                  1001              ;START SPINDLE MOTOR 2
LCL=                  1002              ;LEFT CLAMP
LUNCL=                1003              ;LEFT UNCLAMPED
RCL=                  1004              ;RIGHT CLAMP
RUNCL=                1005              ;RIGHT UNCLAMPED
;                   TABLE OF INPUT VALUES
RESET=                0                 ;AUTO/MAN RESET PUSHBUTTON
AUTOPB=               1                 ;AUTOMATIC MODE PUSHBUTTON
MANPB=                2                 ;MANUAL MODE PUSHBUTTON
CLPTL=                3                 ;CLAMPED PART LEFT LIMIT SWITCH
CLPTR=                4                 ;CLAMPED PART RIGHT LIMIT SWITCH
HDFWD=                5                 ;HEAD FORWARD LIMIT SWITCH
HDRET=                6                 ;HEAD RETURNED LIMIT SWITCH
;                  TABLE OF Z FUNCTIONS
CLMPIN=               Z1                ;CLAMPS IN PLACE INDICATION
HDRT=                 Z2                ;ALL HEADS RETRACTED INDICATION
;                  TABLE OF R FUNCTIONS
EMRET=                R1                ;EMERGENCY RETURN SHUTDOWN CHECK
FULDEP=               R2                ;FULL DEPTH ACHIEVED
#END OF SYMBOL TABLE
```

Figure 5-1 Input Symbol Table to SET-143

SET-143 copies the symbol table onto the output tape and causes the symbol assignments to become a permanent part of the BOOL-143 listing. SET-143 precedes each line that contains a symbol assignment with a line containing ;$. These two characters inform BOOL-143 that the following line is a comment. (The ; specifies "start of comment"; the $ specifies "continued on next line").

**Symbolic Equations**

Once the symbol table has been supplied to SET-143, the user can provide symbolic equations to be tranlated. Three examples of symbolic equations follow:

```
AUTOLT = AUTOPB + AUTOLT*/RESTPB
STMTR1 = SLD1RT * TRNSDN + STNTR1 */
SLD1FD
VLV1OP = (MIXDN* /OPLS1 + VLV1OP) *
MIXRUN
```

The symbols used in the equations should be selected in a manner meaningful to the control system designer and easily understood by those who use and maintain the system. The symbolic equations become part of the BOOL-143 source program just as the symbolic assignments. That is, SET-143 precedes each symbolic equation by a ;$, thereby specifying that the equation is a comment.

A symbolic equation can contain a maximum of 250 characters; therefore, a lengthy equation can be written on up to 3-1/2 lines. Any intermediate line is ended by a $, and the last line ends with the normal carriage return. If an equation exceeds 250 characters, SET-143 types the following message:

```
; ??LINE TRUNCATED
```

and translates only those symbols contained in the first 250 characters. This message is typed before the offending line. SET-143 processes each symbolic equation according to the following procedure:

1. Output the complete symbolic equation as a comment.
2. Read each symbol of the equation.
3. Search the symbol table for the I/O number (X or Y optional) R- or Z-value of each symbol.
4. Output the translated value for each symbol in a BOOL-143 equation format.
5. All internal equation characters such as =, *, +, /, (,), [, and ] are not changed whenever they are found in the symbolic equation.
6. Any symbols that are part of the symbolic equation and are not found in the symbol table are not changed when they are inserted in the I/O numbers equation. This permits the user to intermix an I/O number, R- or Z-value with an equation to be translated.

**Handling of BOOL-143 R and Z Functions**

R- and Z-functions can be included in the program to be translated by SET-143. The values of the functions must be in the range 0–511 decimal (R0–R511, Z0–Z511). A symbol can be assigned to the R- or Z-function in the same manner as an I/O number value, or the R- or Z-value can be used alone. For example, suppose that MTR1 is to be an R-function. One approach is as follows:

```
#
MTRI     = 1012
AUTO     = 31
CLAMP    = 1057
FULDEP   = 107
RMTRI    R5; REPEAT MOTOR 1
#
RMTRI:   MRTI = AUTO * CLAMP * FULDEP
  .
  .
  .
;START OF MAIN PROGRAM
RMTR1; CALL TO MRT1 SUBROUTINE
```

Alternatively, the R-function need not be assigned a symbolic value and the arbitrary R-number can be used. If this approach is chosen, the RMTR1 = R5 entry is deleted from the symbol table, and R5 is used throughout the program in place of RMTR1.

The same procedure is used for Z-functions. A Z-function is a group of Industrial 14 variables (inputs, outputs, or internal functions) that are tested and the result recorded by setting the TEST flag, rather than setting an output. A symbolic value can be assigned to the Z-function, such as AHDRET (all heads returned) or AVALCL (all values closed). The Z-function subroutine could test a group of limit switches or, perhaps, pressure sensors. The Z-function could also be identified with the arbitrary Z-number, which does not require translation by SET-143.

**Comments**

Normal BOOL-143 comments can be inserted in the source program to SET-143 by simply preceding each comment with a semicolon. SET-143 outputs any existing comments unchanged.

**NOTE**

**Everything on the SET-143 source tape, except BOOL-143 control statements, becomes a comment on the output tape from SET-143. All symbol assignments and symbolic equations become comments to BOOL-143.**

**BOOL-143 Control Statements**

SET-143 processes BOOL-143 control statements by simply transferring them, unaltered, onto the BOOL-143 source tape. SET-143 recognizes a control statement by the period that precedes each statement (VARS5, .LOC50, etc.).

The .END .ENDN, and .EOT statements affect the operation of SET-143 in addition to being output on the BOOL-143 source tape. The .END and .ENDN statements halt SET-143 and the translation process. The .EOT statement acts as a normal end-of-tape statement to allow a new tape to be loaded. The translation process continues after an .EOT when the PDP-8 CONT switch is pressed.

**Error Messages**

SET-143 errors are detected only during symbol definitions. The error message is typed on the line immediately after the offending definition. Table 5-1 lists the errors that can be detected.

Whenever an error is detected, the definition that caused the error is ignored and SET-143 continues with the processing. The error messages are typed as comments on the output listing and/or tape in the following format:

```
; ??Enn
```

where nn is one of the error codes listed in Table 5-1.

The following message is typed if the program exceeds 1088 symbols:

```
SYMBOL TABLE OVERFLOW
```

SET-143 halts in the unlikely instance that this condition arises. The program should then be segmented with the BOOL-143 control statement .EOT to permit its translation by SET-143 in two parts. The symbols required for each program part must be defined in that section.

**Table 5-1**
**SET-143 Error Messages**

| Error Number | Cause |
|---|---|
| 01 | First character of a user symbol is not an alphanumeric character.<br><br>Examples:<br>/LS12=1<br>=23 |
| 02 | The user symbol contains more than 6 alphanumeric characters.<br><br>Examples:<br>STA1MTR = 1053<br>CLVALVE = 1102 |
| 03 | The user symbol is not followed by an equal sign.<br><br>HDRET - 27<br>CONVOK; 1053 |
| 04 | The equal sign is followed by an illegal character (not an X, Y, R, Z or I/O number).<br><br>Examples:<br>MIXRUN = ;1057<br>12TIMR = P27 |
| 05 | A non-octal number (8 or 9) is contained in an I/O number. A non-octal digit (8 or 9) follows an X, Y; or a non-numeric character follows an X, Y, Z or R. |
| 06 | An illegal value is assigned to an I/O number X, Y, Z or R (I/O number X or Y greater than 1777, R or Z greater than 511).<br><br>Examples:<br>TMR23 = 2407<br>RS OLB = R2000 |
| 07 | An illegal character other than ; or ) or space follows the variable number.<br><br>Examples:<br>OPVLV3 = 1050 OPEN VALVE<br>BACKLS = 27:BACK LIMIT SWITCH |

Table 5-1 (Cont)
SET-143 Error Messages

| Error Number | Cause |
|---|---|
| 08 | A user symbol is defined equal to two different values.<br><br>Examples:<br>MTR53 = 1005<br>.<br>.<br>.<br>MTR53 = R20 |
| 09 | A previously defined I/O number (X, Y), Z or R-number has been used again.<br><br>Examples:<br>RUNLT = 1036<br>.<br>.<br>.<br>SOLB = 1036 |

**Error Listing**

At the end of every SET-143 symbol table (after the second #) SET-143 types the following message:

; **SYMBOL ERRORS: n

where n is the number of errors in the preceding table of symbolic definitions. The number n is the decimal error total and is greater than or equal to zero.

When the high-speed punch is used as the output device, the symbol error message allows the translation to be stopped by pressing the PDP-8 STOP switch if any errors are detected in the program. Thus, the program can be corrected before the equations are translated.

**Sorted Symbol Table**

An additional listing of the symbol table is typed after SET-143 has completed the translation process (after an .END statement). This symbol table is an alphabetic sort of all defined symbols and associated I/O number, R- and Z-values.

This optional symbol list is only generated on the output device (Teletype or high-speed punch). The symbol list is not in the form of BOOL-143 comments and is not read by BOOL-143 because the symbol list follows an .END statement. However, the symbol list is a permanent part of the output tape and can be listed off-line at any time.

If the original symbol table is written in increasing I/O, R- or Z-numbers, two approaches are provided for identifying a symbol. To determine which output is 1057, for example, refer to the original symbol table. To determine which output light is associated with motor starter STMTR1, refer to the alphabetically-organized symbol table supplied by SET-143.

Any defined symbols not referenced in an equation are followed by a question mark in the alphabetized symbol list. For example, the statement

STOPPB 27?

signifies that the symbol STOPPB was defined, but was never used in a symbolic equation.

**Sample SET-143 Program**

Figures 5-2 through 5-5 show a sample program and its translation by SET-143. Figure 5-2 is the program as read into SET-143. Figure 5-3 is the Translated Symbol Table (optional). Figure 5-4 is the translated control equations program which will be read into BOOL-143. Figure 5-5 is the optional symbol table listing which is arranged alphabetically.

```
   # START OF SYMBOL TABLE
   ;TABLE OF INPUTS
   MSTSTP= 0       ;MASTER STOP PUSHBUTTON
   MSTRST= 1       ;MASTER START PUSHBUTTON
   1MCONT= 2       ;CONTACT ON MOTOR 1
   2MCONT= 3       ;CONTACT ON MOTOR 2
   SPNSTP= 4       ;SPINDLE STOP PUSHBUTTON
   SPNST=  5       ;SPINDLE START PUSHBUTTON
   LEVENG= 6       ;LEVER ENGAGED FOR SPINDLE DIRECTION
   CLUENG= 7       ;CLUTCH ENGAGED PRESSURE SWITCH
   DECSPD= 10      ;DECREASE SPINDLE SPEED
   INCSPD= 11      ;INCREASE SPINDLE SPEED
   LUBPRS= 12      ;PRESSURE SWITCH ACTIVATED BY ADEQUATE LUBE PRESSURE
   LUBLEV= 13      ;LEVEL SWITCH ACTIVATED BY ADEQUATE LUBE SUPPLY

   ;TABLE OF OUTPUTS
   SPNCLU= 1000       ;SOLENOID 1 TO ENGAGE SPINDLE CLUTCH
   FEEDCL= 1001       ;SOLENOID 2 TO ENGAGE FEED CLUTCH
   SPDINC= 1002       ;SOLENOID 3 TO INCREASE SPINDLE SPEED
   SPDDEC= 1003       ;SOLENOID 4 TO DECREASE SPINDLE SPEED
   LUBSOL= 1004       ;SOLENOID 5 TO PROVIDE SPINDLE LUBRICATION
   LBFAIL= 1005       ;LUBE FAILURE LIGHT
   TFDCLU= 1020      ;FEED CLUTCH TIME DELAY
   TLUBIN= 1021      ;LUBRICATION INTERVAL TIME DELAY
   TLUBFL= 1022      ;LUBRICATION FAILTRE @NDICATION DELAY
   # END OF SYMBOL TABLE


;CONTROL EQUATIONS FOLLOW.
.LOC200
.FIXS5
SPNCLU=/MSTSTP*(MSTRST+1MCONT*2MCONT)*/SPNSTP*LEVENG*[SPNST+SPNCLU]
TFDCLU=CLUENG
.MFN
FEEDCL=TFDCLU
SPDINC=SPNCLU*INCSPD*/DECSPD
SPDDEC=SPNCLU*DECSPD*/INCSPD
LUBSOL=TLUBIN
.MN
LBFAIL=TLUBFL
TLUBIN=LUBPRS
TLUBFL=LUBLEV*TLUBIN
.END200
```

Figure 5-2 SET-143 Input Program

```
;$
# START OF SYMBOL TABLE
;TABLE OF INPUTS
;$
MSTSTP= 0       ;MASTER STOP PUSHBUTTON
;$
MSTRST= 1       ;MASTER START PUSHBUTTON
;$
1MCONT= 2       ;CONTACT ON MOTOR 1
;$
2MCONT= 3       ;CONTACT ON MOTOR 2
;$
SPNSTP= 4       ;SPINDLE STOP PUSHBUTTON
;$
SPNST=  5       ;SPINDLE START PUSHBUTTON
;$
LEVENG= 6       ;LEVER ENGAGED FOR SPINDLE DIRECTION
;$
CLUENG= 7       ;CLUTCH ENGAGED PRESSURE SWITCH
;$
DECSPD= 10      ;DECREASE SPINDLE SPEED
;$
INCSPD= 11      ;INCREASE SPINDLE SPEED
;$
LUBPRS= 12      ;PRESSURE SWITCH ACTIVATED BY ADEQUATE LUBE PRESSURE
;$
LUBLEV= 13      ;LEVEL SWITCH ACTIVATED BY ADEQUATE LUBE SUPPLY

;TABLE OF OUTPUTS
;$
SPNCLU= 1000       ;SOLENOID 1 TO ENGAGE SPINDLE CLUTCH
;$
FEEDCL= 1001       ;SOLENOID 2 TO ENGAGE FEED CLUTCH
;$
SPDINC= 1002       ;SOLENOID 3 TO INCREASE SPINDLE SPEED
;$
SPDDEC= 1003       ;SOLENOID 4 TO DECREASE SPINDLE SPEED
;$
LUBSOL= 0004       ;SOLENOID 5 TO PROVIDE SPINDLE LUBRICATION
;$
LBFAIL= 1005       ;LUBE FAILURE LIGHT
;$
TFDCLU= 1020      ;FEED CLUTCH TIME DELAY
;$
TLUBIN= 1021      ;LUBRICATION INTERVAL TIME DELAY
;$
TLUBFL= 1022      ;LUBRICATION FAILURE INDICATION DELAY
;$
# END OF SYMBOL TABLE
```

Figure 5-3 SET-143 Translated Symbol Table (Optional)

```
;CONTROL EQUATIONS FOLLOW.
.LOC200
.FIXS5
;$
SPNCLU=/MSTSTP*(MSTRST+1MCONT*2MCONT)*/SPNSTP*LEVENG*[SPNST+SPNCLU]
1000=/0*(1+2*3)*/4*6*[5+1000]

;$
TFDCLU=CLUENG
1020=7

.MFN
;$
FEEDCL=TFDCLU
1001=1020

;$
SPDINC=SPNCLU*INCSPD*/DECSPD
1002=1000*11*/10

;$
SPDDEC=SPNCLU*DECSPD*/INCSPD
1003=1000*10*/11

;$
LUBSOL=TLUBIN
1004=1021

.MN
;$
LBFAIL=TLUBFL
1005=1022

;$
TLUBIN=LUBPRS
1021=12

;$
TLUBFL=LUBLEV*TLUBIN
1022=13*1021
.END 200
```

Figure 5-4 SET-143 Translated Program

```
1MCONT   2
2MCONT   3
CLUENG   7
DECSPD   10
FEEDCL   1001
INCSPD   11
LBFAIL   1005
LEVENG   6
LUBLEV   13
LUBPRS   12
LUBSOL   1004
MSTRST   1
MSTSTP   Z0
SPDDEC   1003
SPDINC   1002
SPNCLU   1000
SPNST    5
SPNSTP   4
TFDCLU   1020
TLUBFL   1022
TLUBIN   1021
```

Figure 5-5 Sorted Symbol Table (Optional)

# CHAPTER 6
# PAL-143
# SYMBOLIC PROGRAM ASSEMBLER

The *PAL-143 Symbolic Program Assembler* translates programs into binary machine code for the Industrial 14. Industrial 14 programs can be written with the Editor and "assembled" (translated into machine code) by PAL-143, enabling the user to correct program errors with the Editor and to reassemble the program with PAL-143. This capability simplifies the debugging procedure because sections of the source program can be moved, or new lines can be inserted, without retyping the remainder of the program. Figure 6-1 shows the procedure for developing programs using PAL-143.

## PAL-143 GENERAL FEATURES

Chapter 2 of this manual describes the Industrial 14 machine code instructions. Test instructions refer to specific input, output, and internal function numbers; jump instructions reference locations by an address number (specifically, an absolute address for a JMP or JMS and a relative or page address for a JFF or JFN). Consider the following Industrial 14 program segment:

```
      .
      .
      .
227   TN 15
230   JFN 233
231   SN 1012
232   SKP
233   SF 1012
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 TN, SN, and SF instructions refer to inputs or outputs by specific number.

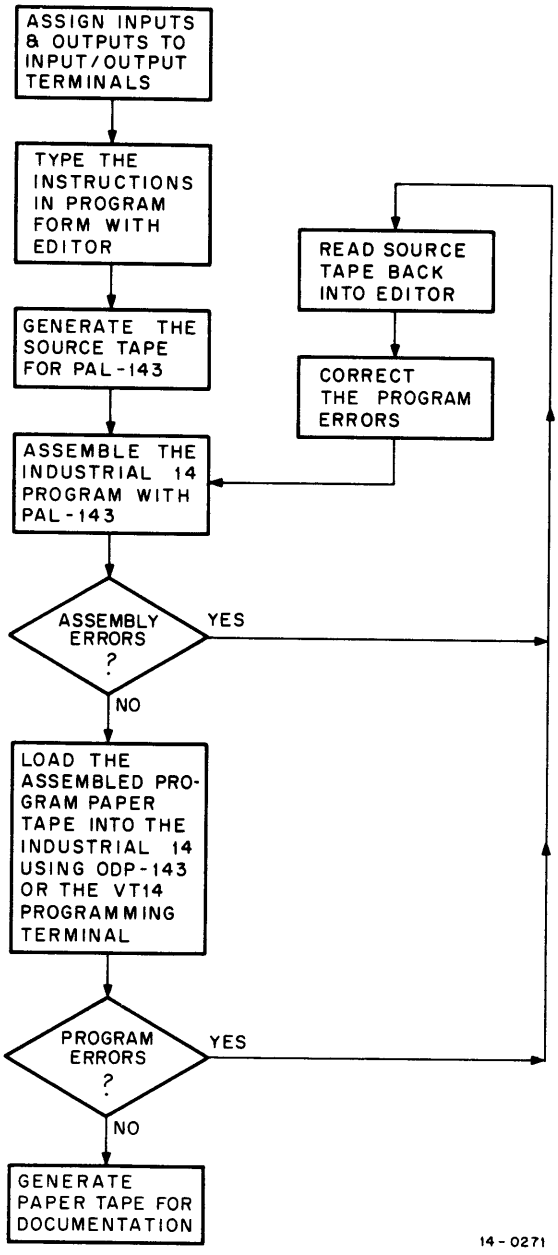



Figure 6-1 Procedure for Developing Programs Using PAL-143

Changing Industrial 14 programs written with numeric addresses and references is cumbersome. For instance, in the preceding program example, assume that a TN 1027 instruction is required following the TN 15 instruction stored in location 227. To correct the program with ODP-143, each instruction is moved to the next sequential location by typing it into that location, thereby enabling TN 1027 to be inserted at location 230. However, moving each instruction to the next successive location does not solve the problem.

The result of moving the instructions is as follows:

```
        .
        .
        .
227  TN 15
230  TN 1027
231  JFN 233
232  SN 1012
233  SKP
234  SF 1012
235  JMP
236  437
        .
        .
        .
```

Inspection shows that by relocating the program segment, the sense of the JFN 233 instruction has changed. Originally the jump caused a transfer to the SF 1012 instruction; now it causes transfer to the SKP instruction. The JFN instruction must be changed to a JFN 234 to preserve the original sense of the program. Dangers are involved in simply relocating programs with ODP-143. These dangers can be avoided by using PAL-143 and reassembling, as described in the paragraphs that follow.

**Location Assignment**

Industrial 14 programs are stored in consecutive memory locations; therefore, the need to specify the address of each instruction is unnecessary. The locations can be assigned by PAL-143 using an origin statement for the sequence of instructions. The statement is LOC NNNN where NNNN is an absolute Industrial 14 address. This statement instructs PAL-143 to assign the instructions which follow to consecutive Industrial 14 locations, beginning with location NNNN. For example:

```
LOC 227
TN 15
TN 1027
JFN 234
SN 1012
SKP
SF 1012
JMP
437
```

PAL-143 recognizes LOC 227 as an origin statement and assigns the instructions which follow to sequential locations, beginning with location 227. Hence, TN 15 is stored in 227; TN 1027 is stored in 230; JFN 234 is stored in 231; etc. The need to write location numbers is eliminated by using PAL-143.

**Symbolic Addresses**

The LOC statement alone does not eliminate problems that arise when Industrial 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 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 by PAL-143 during the assembly of the program.

The instruction JFN 234, for example, can be written as JFN TAG. The location to be jumped to is assigned the address label TAG, followed by a comma, and precedes the reference instruction. Consider the following:

```
        .
        .
        .
JFN TAG
        .
        .
        .
TAG, SF 1012
        .
        .
        .
```

The JFN instruction *references* the symbolic address TAG while the statement TAG, SF 1012 *defines* the symbolic address.

PAL-143 assigns an address value to each instruction of the program using a Location Counter (LC). The LOC statement sets the LC to a specific value; for instance, LOC 200 sets the LC to 200. PAL-143 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 Industrial 14 memory. Whenever a symbolic address is encountered, (recognized because it is followed by a comma) it is assigned the value of the LC. If the LC is equal to 234 when the line TAG, SF 1012 is encountered in the source program, the SF 1012 is stored in Industrial 14 memory location 234, and TAG is assigned the value of 234. The instruction JFN TAG becomes JFN 234 when assembled by PAL-143.

**NOTE**

**Programs in this form can be readily changed to eliminate bugs. The program can 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-143 correctly assigns 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 results if a PAL-143 program contains instructions which reference symbolic addresses that are not defined within the same Industrial 14 memory page. Only the JMP or JMS instruction can 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 follow. For example, SETY10N is a valid PAL-143 symbol because it contains 7 characters; however, only the first six are recognized by PAL-143. Thus, if the symbols SETY10N and SETY10FF are both used as symbolic addresses in a PAL-143 program, a multiple definition error would occur. PAL-143 recognizes only the first six characters, and cannot distinguish between two identical symbols 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 because they differ in the first six characters.

*Rule 1* — The first character of a symbol must be a letter.

*Rule 2* — The successive characters in the symbol can be letters or numbers only; no special characters such as *, +, -, /, and $ are permitted within symbols.

*Rule 3* — A symbol must be defined at only one point in a program. Symbols can be referenced as many times as necessary.

*Rule 4* — Only the first six characters of the symbol are meaningful to PAL-143 although more characters may be used.

*Rule 5* — No spaces or tabs are allowed within the symbol.

## ASSIGNMENT STATEMENTS

PAL-143 assignment statements equate a six-character symbol with a numeric value. Symbols used in assignment statements are subject to the rules just given. Assigning a value to a symbol, which is also used as a symbolic address, is illegal in PAL-143; however, the same symbol can be used in more than one assignment statement. The last assigned value is used whenever a symbol has been redefined in this manner.

The most common use of assignment statements in PAL-143 programs is to identify input and output assignments. It is easier to remember that "LS1 is limit switch 1" rather than "17 is limit switch 1". Since Industrial 14 programs must refer to inputs and outputs by the numbers which are determined by the I/O terminal assignments (17 for instance), the following PAL-143 assignment statement is used:

```
LS1 = 17
```

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

```
                LS1 = 17
                LS2 = 30
                SOLA = 1011
                SOLB = 1012
                PB1 = 10
                PB2 = 11

LOC 250
Y11,            TN LS1
                TF LS2
                JFF SOLAN
                TF PB1
                TN SOLB
                JFN SOLAF
SOLAN,          SN SOLA
                SKP
SOLAF,          SF SOLA
END
```

The END statement is necessary to mark the end of the program for PAL-143. Assignment statements need only be made at one point in the program and the symbols thus assigned can then be used throughout the program. (This example controls only one output.) Note that LS1 = 17 is used in place of input 17.

## PAL-143 PROGRAM CONVENTIONS

PAL-143 programs are free-form; spaces and tabs can be used to organize and format the program as desired. At least one space (or a tab) must be used between an instruction such as TN or SF and the reference symbol or operand (SL1 or SOLB for instance). Tabs and spaces cannot be used within a symbol or instruction; they can only be used between symbolic addresses, instructions, operands, and comments. Otherwise, these characters have no affect on the machine code generated by PAL-143 and are only used to format the source program. For example:

```
SOLA,    TF     LS1
```

generates the same machine code as

```
SOLA, TF LS1
```

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

### Comments

Program comments are lines of text included in the PAL-143 program to clarify the function of an instruction or a group of instructions. Comments do not affect the machine code generated by PAL-143; these are only reminders for later reference to the program listing. Comments can be added within the PAL-143 program by preceding them with a slash (/). Comments can either conclude lines of a program, or they can be on separate lines by themselves. PAL-143 instructions, or assignment statements, must not be included on a line started with a comment because all information would then be treated as part of the comment.

The following are examples of PAL-143 comments:

```
/THE FOLLOWING PROGRAM CONTROLS
/SOLENOID A

    LOC 200

Y1, TF LS1     /LS1 ACTIVATED WHEN SLIDE 2
               /IS FULL FORWARD

    TN PB1     /PB1 IS SHUT DOWN BUTTON

    .
    .
    .
```

### Multiple Location Instructions

Industrial 14 two-location instructions (JMP, JMS, TRM) are written in the PAL-143 source program with both parts of the instruction on one line. These instructions require two Industrial 14 memory locations when translated into machine code instructions. Other two-location instructions and some three-location instructions are introduced in Chapter 8. The following example illustrates the two-location translation process by PAL-143. The input to PAL-143 appears at the left; the translated output in machine code is at the right. The symbols OUTPT5 and NEXT are symbolic addresses which have been assigned the values 1323 and 1401, respectively, by the current location counter of PAL-143.

```
STAIN = 2002
LOC 200
TRM STAIN          200 TRM
                   201 2002
JMS OUTPT5         202 JMS
                   203 1323
JMP NEXT           204 JMP
  .                205 1401
  .                  .
  .                  .
OUTPT5, TN LS16      .
  .
  .
  .
NEXT, TN SOLB
  .
  .
  .
```

### Permanent Symbol Table

PAL-143 must "know" what Industrial 14 machine code to generate for each symbol of a source program before it can translate that program into a complete Industrial 14 machine code program which is acceptable to ODP-143. (Numbers are the only elements of PAL-143 source programs that require no definition; all symbols must be defined in terms of these numbers.) Definitions can be in the form of assignment statements (for instance, LS1=27) and symbolic addresses (values of which are defined by the location counter and LOC statements).

In addition to user-defined symbols, PAL-143 has a table of permanently-defined symbols and numeric equivalents. The permanent symbol table enables the user to write statements such as TN 15 without previously defining the symbol TN. This symbol and all others given in Table 6-1 are permanently defined within PAL-143. The list includes the basic Industrial 14 instructions, introduced in Chapter 2, as well as the extended instructions introduced in Chapter 9.

**Table 6-1**
**PAL-143 Permanent Symbol Table Industrial 14 Instructions**

| Symbolic | Octal | Meaning | Argument | Range of Argument |
|---|---|---|---|---|
| TF | 4000 | Test input, output or internal function for OFF | Input, output or internal function number | 0–1777 |
| TN | 6000 | Test input, output or internal function for ON | Input, output or internal function number | 0–1777 |
| SF | 0000 | Set output or internal function OFF | Output or internal function | 1000–1777 |
| SN | 2000 | Set output or internal function ON | Output or internal function | 1000–1777 |
| JFF | 2000 | Jump if test flag is OFF | Page address | 0–377 |
| JFN | 2400 | Jump if test flag is ON | Page address | 0–377 |
| JMP | 0024 | Jump unconditional | Absolute address | 0–7777 |
| JMS | 0124 | Jump to subroutine | Absolute address | 0–7777 |
| SKP | 0010 | Skip next location | None | – |
| JMR | 0054 | Jump return from subroutine | None | – |
| NOP | 0000 | No operation | None | – |
| CLR | 0170 | Clear all external outputs | None | – |
| LEM | 0040 | Leave external mode | None | – |
| EEM | 0060 | Enter external mode | None | – |
| CLRPC | 0004 | Clear 13-bit PC | None | – |
| RDPC | 0046 | Read Industrial 14 Program Counter | None | – |
| EOL | 0130 | Enable output loop | None | – |

Table 6-1 (Cont)
PAL-143 Permanent Symbol Table Industrial 14 Instructions

| Symbolic | Octal | Meaning | Argument | Range of Argument |
|---|---|---|---|---|
| DOL | 0140 | Disable output loop | None | – |
| EOM | 0150 | Enable output multiplexer | None | – |
| DOM | 0160 | Disable output multiplexer | None | – |
| CIF0 | 0020 | Change to instruction field 0 | None | – |
| CIF1 | 0030 | Change to instruction field 1 | None | – |
| CDF0 | 0600 | Change to data field 0 | None | – |
| CDF1 | 0700 | Change to data field 1 | None | – |
| CLRWD | 0003 | Clear I/O word | I/O word | 0–1777 |
| SETWD | 0013 | Set I/O word to 7777 | I/O word | 0–1777 |
| LDMEM | 0022 | Load into memory | Absolute address | 0–7777 |
| LDWD | 0023 | Load I/O word with absolute address | Absolute address, I/O word | 0–1777 |
| RDMEM | 0026 | Read memory to output register | Absolute address | 0–7777 |
| MOVWD | 0033 | Move I/O word from I/O address to I/O address | I/O word, I/O word | 0–1777 |
| RDWD | 0036 | Read I/O word to output register | I/O word | 0–1777 |
| CLRBIT | 0103 | Clear I/O bit | I/O word | 0–PA* |
| SETBIT | 0113 | Set I/O bit | I/O word | 0–PA* |
| LDBIT | 0123 | Load I/O bit with absolute address | Absolute address, I/O word | 0–PA* |
| MOVBIT | 0133 | Move I/O bit from I/O address to I/O address | I/O word, I/O word | 0–PA* |
| RDBIT | 0136 | Read I/O bit to output register | I/O word | 0–1777 |

*PA = Assigned Partition Address

Table 6-1 (Cont)
PAL-143 Permanent Symbol Table Industrial 14 Instructions

| Symbolic | Octal | Meaning | Argument | Range of Argument |
|---|---|---|---|---|
| **The following instructions are compatible with first generation PDP-14:** | | | | |
| TXF | 4000 | Test input for OFF | Input number | 0–777 |
| TXN | 6000 | Test input for ON | Input number | 0–777 |
| TYF | 5000 | Test output for OFF | Output number (9 bit) | 0–777 |
| TYN | 7000 | Set output for ON | Output number (9 bit) | 0–777 |
| SYF | 1000 | Set output OFF | Output number (9 bit) | 0–777 |
| SYN | 3000 | Set output ON | Output number (9 bit) | 0–777 |
| TRM | 0026 | Transfer memory to output register | Value to be transferred | 0–7777 |

Each of these symbols can be used in PAL-143 source programs without definition by the user. These symbols must not be used in PAL-143 source programs as symbolic addresses or in assignment statements.

The following are pseudo-instructions.

| Symbolic | Meaning |
|---|---|
| END | End of source program |
| EOT | End of tape in a segmented program |
| FLD 1 | Memory Field 1 |
| LOC | Set location counter |
| PAGEJ | Page jump to start of next page |
| PRE | Preset timers and counter<br>CNTR – Counter<br>SEC – Timer with 1 sec increments<br>TSEC – Timer with .1 sec increments |

**User Symbol Table**

The PAL-143 symbol table can accommodate 792 symbols. Both symbolic addresses and assigned symbol values are entered in the user symbol table. Thus, the number of symbols defined as addresses, added to the number of symbols with assigned values, may not exceed 792. When this symbol capacity is exceeded, the following message is typed:

*SMB OFLW

Recovery requires the user to segment his program and to assemble it in parts.

**Special Characters and Operators**

Several characters of the Teletype keyboard have special meaning to PAL-143. An example is the slash (/) which denotes the start of a comment. Another example is the carriage return, which terminates a statement line. Table 6-2 contains all legal characters in the operation portion of a PAL-143 statement (namely, the statement exclusive of any comment field). If any other character appears in the operation portion of a PAL-143 statement, the "I" error (illegal character) is generated.

**Table 6-2**
**PAL-143 Special Characters**

| Character | | Use in PAL-143 |
|---|---|---|
| + | (plus) | Combines symbols or values by 2's complement addition |
| - | (minus) | Combines symbols or values by 2's 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) | When encountered in the program, is assigned a value equal to the present value of the PAL-143 location counter |

The following characters are ignored in the PAL-143 source program, but they do not generate errors:

Line feed
Form feed
Rubout
Leader/trailer (null)

*Addition and Subtraction* – The + and - characters can 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 the location labeled START (location 102 and the TN PB3 instruction).

```
LOC 100
START, TN PB 1
JFN OUT
TN PB3
   .
   .
   .
JMP START +2
   .
   .
   .
```

The minus sign can also be used; for example, JFF SOLAN-2 causes a JFF to the second location before the location labeled SOLAN. The location counter of PAL-143 assigns a value to each label; PAL-143 then computes the values of all symbolic addresses which contain operators and symbols.

*Logical OR* – The logical OR character (!) can be used in PAL-143 programs to combine values of symbols or octal numbers. The result is the bit-by-bit inclusive OR of the two values. Table 6-3 lists the resultant value when any two octal digits are combined with the ! character.

**Table 6-3**
**Inclusive OR For Octal Numbers**

| Octal Digit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| | Resultant Value | | | | | | | |
| 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 containing two or more digits are combined, each digit in the result is obtained by combining the corresponding digits in the original numbers using Table 6-3.

1234 : 2460 = 3674

The operator can be used in PAL-143 in the following manner:

RDMEM 15 : 27 translates to RDMEM 0037

The values of symbols can also be combined with the colon (:) character. For example:

ON = 1
SOLA = 1500
RDMEM SOLA : ON translates to RDMEM 1501

*Period* — The period (.) is a special symbolic address used in PAL-143. Its value changes for each instruction and is always equal to the present value of the PAL-143 location counter. Jump instruction operands can contain the period (often read "dot") and the +, −, and : operators. For example, JFF .+12 (read, JFF dot plus 12) specifies JFF to the twelfth octal (tenth decimal) location after the JFF instruction. The period is most useful with small (less than 10) octal numbers; large jumps are best performed in PAL-143 with true symbolic addresses.

JFF .+1 is a variation of this instruction, which causes the next instruction to be executed for both states of the TEST flag. Because the TEST flag is cleared in both instances, the instruction JFF .+1 can be used to unconditionally clear the TEST flag.

The period can also be used with two-location instructions such as JMP or JMS. However, the value of the period is the value of the location counter when the second half of the instruction is assigned to its location; JMP .+2, therefore, causes a transfer to the third location which follows the JMP. Assuming that no two-location instruction follows the JMP, the jump is to the second instruction in the sequence. For example:

| | | |
|---|---|---|
| LOC 200 | translates to | 203 JMP |
| JMP .+2 | | 201 203 (201 + 2) |
| TN 1 | | 202 TN 1 |
| TN 2 | | 203 TN 2 |

**Pseudo-Instructions**

Pseudo-instructions are included in the PAL-143 program listing, stored on the source paper tape, to direct the assembly process. Pseudo-instructions are not translated into Industrial 14 machine code. However, they do affect translation of other instructions into machine code. PAL-143 has six pseudo-instructions, including the LOC and END statements, mentioned earlier in this chapter.

*END* — The END pseudo-instruction marks the end of a source program and directs PAL-143 to terminate the current assembly pass. The complete assembly process requires either two or three passes (depending on the assignment of I/O devices). During the assembly process, the source tape is read by PAL-143 and translated into Industrial 14 machine code. The END statement terminates the current pass and causes PAL-143 to halt while the source tape is reloaded to begin the next assembly pass.

Because the END statement halts the reading of a paper tape during assembly, any information present on the source tape after the END statement is not read by PAL-143. However, the comments are a permanent part of the source program and can be read with the Editor. The assembly process is faster if lengthy comments and program documentation are written after the END statement.

*EOT* — The pseudo-instruction EOT (end of tape) enables a long PAL-143 program to be segmented and recorded on more than one source paper tape. The EOT is typed as the last statement on all source tapes except the last tape; the last tape must be concluded with the END pseudo-instruction. At each instance that the EOT statement is read during an assembly pass, PAL-143 stops and types EOT. The next source tape is then loaded and assembly continues. This procedure is repeated for each source tape in the complete program.

*LOC* — The LOC pseudo-instruction directs PAL-143 to set its location counter to the operand value of the LOC. As many LOC statements as desired can be used; however, if no LOC statement is present, PAL-143 assumes a LOC 0000 statement at the beginning of the program.

The LOC statement can 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 is lost. For example,

```
LOC 50
        TN LS1
        TN LS2
        .
        .
        .
LOC 50
        SN SOLA
        .
        .
        .
```

results in the assembly of two instructions for the same location. The first instruction (TN LS1) is lost from the assembly; Industrial 14 location 50 will contain the assembled instruction for SN SOLA.

The LOC statement can have an expression or a symbolic operand. All terms of the expression must be previously defined to enable PAL-143 to set the location counter to the proper value. For example:

```
        .
        .
        .
SOLAF, SF SOLA

/LEAVE 8 LOCATIONS BEFORE STARTING
/NEXT PROGRAM SEGMENT

LOC SOLAF+10

        .
        .
        .
```

The value of SOLAF is assigned at assembly time and the LOC statement sets the location counter 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-143 to advance the location counter to the first location of the next Industrial 14 memory page. The statement is evaluated by PAL-143 as: "The current value of the location counter (.) OR'ed with 377 (octal) plus 1." If the location counter is 1312, for example, the result of the OR is 1377 and the LOC statement sets the location counter to 1400 ($1377_8 + 1_8$).

*PAGEJ* – The PAGEJ (page jump) pseudo-instruction causes PAL-143 to include in the assembled program whatever instruction is necessary to transfer control to the first location of the next Industrial 14 memory page. The actual code generated depends on 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 is not assembled as the first or last instruction in the page, PAL-143 supplies a JMP instruction. The operand of the JMP instruction is 1 plus the OR of the current location counter value with 377. The JMP instruction always jumps to the first location of the next memory page.

The PAGEJ pseudo-instruction also acts as a LOC .: 377+1 in instances where a JMP instruction was assembled. Thus, PAGEJ supplies the necessary Industrial 14 instruction to transfer control to the beginning of the next memory page and also sets the PAL-143 location counter to that location.

*PRE* – (Preset) pseudo-instruction causes PAL-143 to preset a timer or counter (up or up/down) at a particular I/O address to an *octal* preset value.

PRE (I/O NUMBER) (FUNCTION TYPE):
(PRESET VALUE)

Where I/O Number =
1400–1776 (even number only)

Function Type =
CNTR – up or up/down counter
SEC – Timer using 1 sec increment.
TSEC – Timer using .1 sec increment.

Preset value, in octal =
0–1777.

The PRE instruction does not generate machine language but is punched on the binary output tape to preset the internal function when the tape is used to load the Industrial 14 memory.

*FLD 1* – (Field 1) pseudo-instruction causes PAL-143 to store the Industrial 14 instructions that follow in Field 1, starting with location 0000.

## PAL-143 INTERROGATION

The input of PAL-143 consists of an ASCII source tape of the Industrial 14 instructions generated from the Editor. The input device is requested by the user in response to the PAL-143 message:

*SRC – The Industrial 14 program should be read-in with what device? The output of PAL-143 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 two PAL-143 messages:

*BIN – Is binary output required; if so, on what device?

*LST – Is an assembly listing required; if so, on what device?

The responses to the three messages are:

L – The input or output is requested and directed to the low-speed device.

H – The input or output is requested and directed to the high-speed paper tape punch.

RETURN – The output is not requested.

### Binary Output

The PAL-143 binary output is a paper tape. This can be used as input to ODP-143 where the program can be debugged. A punched code at the end of the paper tape (checksum) is used by ODP-143 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 Industrial 14 machine code, and error codes. PAL-143 formats the assembly listing into pages of 66 lines (approximately 11 in.) and types the page number (in octal) at the top of each output page.

The assembly listing contains four fields:

*Error Field* – contains letters which identify all assembly errors detected in the lines of the source program.

*Location field* – the four-digit octal address of the Industrial 14 location which contains the assembled instruction. Only assembled instructions have entries in the field.

*Code field* – contains the numeric value of the code for assembled instructions, assignment statements, or the LOC pseudo-instruction.

*Source statement* – contains the line of user program as read from the source tape. If the source statement exceeds 54 characters, one output line is too short and PAL-143 will break the statement into two or more output lines, preceding the second and later lines with an asterisk.

At the end of the program listing PAL-143 types:

*ERR LINES* – denotes the number of lines (in decimal) in the source program which contain at least one error.

*MEM BRK* – specifies 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-143 symbol table is an alphanumeric listing of all user-defined symbols in the source program (namely, 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 the value +0. Thus, if the statement PB1=PB0 + 1 is used and no statement defines PB0, the symbol table contains 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 line containing the source statement in 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 typed only once.

PAL-143 stops processing the source statement when it encounters an S (syntax) error, (Table 6-4). Thus, the source statement can contain undetected errors if they occur after the cause of the syntax error. These errors can be detected after the cause of the syntax error is removed and the statement is reassembled.

Table 6-4 lists all errors recognized by PAL-143. The table includes examples of source statements which cause the particular error and the action taken by PAL-143 when the error occurs.

Table 6-4
**Errors Recognized by PAL-143**

| 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.<br><br>Example:<br>At location 376<br>A LEAVE, JFN.+3 | The machine code is generated for JFF 0 or JFN 0; therefore, the code for the example is JFN 0. |
| D | *Double Defined Symbol Error* – an instruction references a symbol which has been defined more than once in the source program.<br><br>Example:<br>A, TN SOLA<br>A, JFN OUT<br>D JMP A | PAL-143 uses the first definition. The JMP A in the example would jump to the TN SOLA instruction. |
| I | *Illegal Character Error* – the statement contains a character which is not acceptable to PAL-143.<br><br>Example:<br>I A, JMP A#BC | PAL-143 ignores the illegal character. Thus the example is interpreted:<br><br>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.<br><br>Example:<br>L 50=LS1<br>L 47, JMP RESTART | The label or assignment is ignored by PAL-143. |

**Table 6-4 (Cont)**
**Errors Recognized by PAL-143**

| Error Code | Meaning | Action Taken |
|---|---|---|
| M | *Multiple Definition Error* – a label is defined more than once in the source program.<br><br>Example:<br>M A, TN SOLB<br>M A, JFN SOLBN | The first definition is used and all others are ignored by PAL-143. In the examples, A is equal to the location which contains TN SOLB. |
| O | *Operation Error* – the source program contains an illegal operation e.g., two instructions in the same statement.<br><br>Example:<br>O START, JMP TEST JMP | PAL-143 ignores the operation and all subsequent information. In the example, the machine code is generated as if the statement were:<br><br>START, JMP TEST |
| P | *Phase Error* – A label has a different Pass 1 and Pass 2 value. This error is normally caused by a symbolic LOC statement in which the operand is defined after it is used.<br><br>Example:<br>P LOC CDE<br>P XYZ, JMP.+2<br>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-143 permanent symbol (see Table 6-1) or a previously defined label.<br><br>Examples:<br>R LOC50<br>JMP=400<br>ABC, TN LS1<br>.<br>.<br>.<br>ABC=70 | The symbols retain the original values and the assignment statement is ignored. |

**Table 6-4 (Cont)**
**Errors Recognized by PAL-143**

| Error Code | Meaning | Action Taken |
|---|---|---|
| S | *Syntax Error* – the source statement is not meaningful to the assembler.<br><br>Example:<br>S ABC, JMP.+<br>S LS1 + LS2,<br>S LS1 =<br>S SOLA, TN+=3<br>S JFN SOLAN, | The statement is evaluated from left-to-right, up to the point where the error occurred.<br><br>Thus:<br><br>ABC, JMP.+<br><br>assigns ABC the current value of the location counter and generates the machine code for JMP.; The value of LS1 plus LS2 is stored in the next memory location (without any symbolic address). The statement LS1=does not generate any machine code. The location with label SOLA contains the machine code for TN 0. The JFN SOLAN, is assembled as JFN SOLAN. |
| T | *Truncation Error* – a single value exceeds 7777 octal, or the maximum argument value for the instruction (Table 6-1) if maximum is less than 7777.<br><br>Examples:<br>C = 3<br>T A=40543+C<br>ABC=375 + 100<br>T TN ABC | 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-143 substitutes the value 0 when the value is too large. Thus the value of:<br><br>ABC = 375 + 100<br>TN ABC<br><br>is the following:<br><br>ABC=475<br>TN 0 |
| U | *Undefined Symbol Error* – an instruction references an undefined symbol.<br><br>Example:<br>XYZ = 50<br>U JMP XZZ+3<br><br>where XZZ is not defined | The undefined portion of the statement is set equal to 0. Thus:<br><br>JMP XZZ + 3 becomes<br>JMP 0 + 3 or JMP 3 |
| V | *Illegal Value Error* – a value given in the source program cannot be evaluated by PAL-143.<br><br>Example:<br>V ABC=8402+5<br>LV 4LS, TN LS4 | The illegal digit is ignored, and the remainder of the expression is evaluated. Thus:<br><br>ABC = 402+5 = 407<br>However, in the second case, the whole label is ignored. |

**Error Listing**

The assembly listing typed during Pass 2 of PAL-143 includes error messages for all errored lines. However, PAL-143 also types a partial error listing of all lines containing errors detected during Pass 1. In general, this listing does not report all errors, because some errors are detected only during Pass 2. Furthermore, the listing may report undefined symbols which are properly defined during Pass 2. For example,

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

results in ABC being undefined at the end of Pass 1. ABC is therefore 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 an error is not generated for Pass 2.

Although the Pass 1 error listing may be incomplete, it does enable 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 are generated, the user can 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 causes Pass 1 errors to be recorded there also.

**Sample Output**

A sample assembly output including the assembly listing and symbol follows. 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)*/PB5
/
/
                SOLA=1005
                LS1=415
                LS2=16
                PB2=7
                PB3=10
                PB5=12


LOC 200
SOLA,   TF      LS1
        TF      PB2
        TN      P*B3
        JFF     CK5
        TF      SOLA
        TN      LS2
        JFF     SOLAN
5CHK,   TN      PB5
        JFN     SOLAF
SOLAN,  SN      SOLA            /TURN SOLA ON
        SKP
SOLAF,  SF      SOLA            TURN SOLA OFF
END
```

SOURCE PROGRAM

```
*BIN-L
*LST-L              COMMAND STRING
*SRC-L
R     0200  4415   SOLA,  TF      LS1
I     0202  6010           TN      P*B3
LV    0207  6012  5CHK,    TN      PB5
ERR LINES-0003  MEM BRK-0014

UNDEFINED SYMBOLS


CK5   -U
OFF   -U
TURN  -U


*PAS
```

PASS 1 ERRORS

PASS 1 SYMBOL TABLE

```
PAGE- 0001    - PAL-143/8  V0

                    /THE FOLLOWING PROGRAM SOLVES THE EQUATION
                   /SOLA=(LS1*PB2*/PB3+SOLA*/LS2)*/PB5
                   /
                   /
R                                        SOLA=1005
                0415                     LS1=415
                0016                     LS2=16
                0007                     PB2=7
                0010                     PB3=10
                0012                     PB5=12


        0200            LOC 200
        0200    4415     SOLA,  TF       LS1
        0201    4007            TF       PB2
I       0202    6010            TN       P*B3
U       0203    2000            JFF      CK5
        0204    4200            TF       SOLA
        0205    6016            TN       LS2
        0206    2211            JFF      SOLAN
LV      0207    6012    5CHK,   TN       PB5
        0210    2613            JFN      SOLAF
        0211    3200    SOLAN,  SN       SOLA          /TURN SOLA ON
        0212    0010            SKP
U       0213    1400    SOLAF,  SF       SOLA          TURN SOLA OFF
                        END
```

PASS 2 LISTING

ERROR CODES

```
ERR LINES-0005  MEM BRK-0014

SYMBOL TABLE


CK5    -U
LS1    -00415
LS2    -00016
OFF    -U
PB2    -00007
PB3    -00010
PB5    -00012
SOLA   - 0200
SOLAF  - 0213
SOLAN  - 0211
TURN   -U
```

PASS 2
SYMBOL TABLE
(ADDRESSES ARE 4-DIGIT NUMBERS;
ASSIGNED VALUES ARE 5-DIGIT NUMBERS).

```
*PAS
●●B●0

*PAS
```

Spurious characters are typed while binary is punched.

Pressing CONT will restart PAL-143/8.

# CHAPTER 7
# ODP-143
# ONLINE DEBUGGING PROGRAM

ODP-143 is a computer-based debugging program for the Industrial 14 system. It is used primarily to assist the Industrial 14 user in debugging programs compiled by BOOL-143 or assembled by PAL-143.

## MODES OF OPERATION

Two basic modes of ODP-143 operation exist: Program and Run. Program mode is used to manipulate and modify Industrial 14 programs. In this mode, Industrial 14 programs can be loaded, verified, listed, altered, and punched out on paper tape. The Run mode is used for interacting with the Industrial 14 program as it is controlling the equipment. In this mode, inputs and outputs can be disabled and forced ON or OFF to check field wiring and equipment operation. Interrogation of input, output and internal functions can be achieved in both modes.

## CONVENTIONS

The following conventions are used to describe the dialogue between the user and ODP-143.

a. All characters typed by ODP-143 are identified in this manual by an underline.

   Examples:

   NOP  Typed by ODP-143
   TN1  Typed by user

b. The nonprinting character RETURN is noted in this manual by a curved arrow ( ↵ ). No character appears on the teleprinter when this character is typed.

   Example:

   TN1↵  User concluded typing with a carriage return.

c. The nonprinting character LINE FEED is noted in this manual by a straight arrow pointing downward (↓).

   Example:

   TN1 ↓  User concludes typing with a line feed.

## ODP-143 COMMAND DESCRIPTION

ODP-143 has many commands to control program debugging. Some commands are recognized only in Program mode; others are recognized only in Run mode; still others are recognized in both modes. If the user types a command that is not recognized in the present mode of operation, ODP-143 types the error message M? (mode error). The command that caused the error is then ignored and the user may either change modes or type a new command.

All commands in ODP-143 are terminated by a carriage return. When the carriage return is typed, ODP-143 acts on the command. Any spaces typed by the user are ignored.

Three types of commands exist in ODP-143. If a command is not typed in its proper form, the error message S? (syntax error) is typed. The three ODP-143 command type forms are:

a. One or two letters that 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 given in form c. can also be typed in form b. Specific examples of these command forms are covered in the following pages. It is important that the reader understand the three forms, and that a command *must* be typed in the form expected by ODP-143.

Commands in this manual are presented as specific examples without presenting the general form. Commands that have limits (for instance, form c.) can be typed in form b. if only one entity is affected by the command.

For illustration purposes, the user can substitute any Industrial 14 program address, Industrial 14 input or output number, or other parameter for those given in this manual.

### SWITCH REGISTER 5

To halt any printing (for a List or Interrogate command) set Switch Register 5, on the front panel of the computer, to 1. To issue further commands, set SR 5 back to 0 to return to Program or Run mode. The following example demonstrates the use of this command.

| | |
|---|---|
| #LS 200–250 | List symbolically the contents of locations 200 through 250 |
| 0200 TN 011 | |
| 0201 JFF 207 | |
| 0202 TN 001 | |
| 0203 TN 002 | |
| 0204 JFN 207 | |
| 0205 SN 1001 | |
| 0206 SKP | |
| 0207 SF 1001 | |
| 0210 JMS | After location 210 is listed symbolically, the user sets Switch Register 5 = 1 on the computer console to omit the printing of locations 0211 through 0230. |
| # | He then sets SR5 = 0 and ODP-143 returns to Program Mode – Ready to accept further commands |

### RUBOUT KEY

If the user types a command incorrectly or wishes to have ODP-143 disregard a command for any reason, he can strike the Teletype RUBOUT key, provided that he has not already typed a carriage return to terminate the command. ODP-143 records the RUBOUT as the character ?

| | |
|---|---|
| #ZN? (Rubout) | The user intended to type ZM and therefore strikes the RUBOUT key at this point. ODP-143 ignores the command and types ?, the carriage return, and a number sign to signal readiness to accept a further command. The user then types ZM, followed by a carriage return. |
| #ZM | |
| DONE | |
| # | |

**NOTE**

**The RUBOUT key can only be used to ignore commands that have not been acted on by ODP-143 (specifically, those which have not been followed by a carriage return).**

### PROGRAM COMMANDS

The following set of ODP-143 commands enables the user to:

a. Specify interface used

b. Zero Industrial 14 memory

c. Read paper tape programs

d. Change memory field

e. Change these programs

f. List these changed programs

g. Interrogate input, output, and internal functions states

h. Start Industrial 14 execution ectuion

i. Punch and verify new paper tape records of the changed program.

Except for the Interrogation commands, these commands are always used in the Program mode.

Because Industrial 14 programs can be readily changed with ODP-143, the user must maintain an accurate record of the current state of his program. This record may be in the form of ODP-143 generated listings; however, it is strongly recommended that the user maintain a correct, current version of this program in its original "source" form (BOOL-143 or PAL-143 input). If major program changes are required at a later date, the user will need a correct and debugged version of his original program, in the format of the BOOL-143 compiler, or the PAL-143 assembler, so that he can make changes with the Editor. A correct, current program listing and paper tape record of each Industrial 14 program in BOOL-143 or PAL-143 source form is essential if changes are to be made.

**Interface Commands**

Installation of the Industrial 14 and the interface between the external computer and the Industrial 14 must be completed before using ODP-143. Upon start-up, ODP-143 will assume that an SO (Serial One) command has been issued and that it is communicating with the Industrial 14 via serial line interface through the utility port (output register 1). If ODP-143 is to use the parallel interface or the serial line interface via the monitoring port (output register 6) the user should enter the Interface Command PA (Parallel) or SS (Serial Six) respectively. The utility port and the monitoring port are discussed in Chapter 9.

#SS ↵
#

Specifies that the ODP-143 communicate with the Industrial 14 through the serial line interface via the monitoring port (output register 6).

If ODP-143 attempts to communicate with the Industrial 14 via a non-existent interface, a "14 HUNG" or "14 STOPPED" message will be printed on the Teletype.

**Zero Memory Command**

To zero all locations in Industrial 14 memory, type ZM (zero memory) followed by a carriage return. ODP-143 will zero all locations in the Industrial 14 4K or 8K memory. Before reading a program into the Industrial 14, a ZM command should be issued to avoid adding any irrelevant instructions to the program.

#ZM ↵
DONE

The user types ZM, followed by a carriage return. ODP-143 zeroes all locations in memory and types DONE, followed by a carriage return and a number sign (#) when it is finished.

**Read Commands**

*Low-Speed Reader* – The user types R, followed by a carriage return, to direct ODP-143 to read a paper tape from the low-speed reader unit of the Teletype console. The paper tape must be in the binary form obtained from the BOOL-143 compiler, the PAL-143 assembler or ODP-143.

The procedure for reading paper tapes with the low-speed reader is as follows:

1. Type R, followed by the RETURN key
2. Place paper tape in the Teletype's low-speed reader. The leader section of the tape must be positioned over the reading head
3. Switch the reader to START
4. The tape is read by ODP-143
5. When the tape stops, switch the reader to STOP
6. Remove the paper tape from the reader
7. Press the CONT switch on the computer console

#R ↵
OK
#

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, ODP-143 prints OK. The user switches the reader to OFF, removes the tape and depresses the PDP-8 or PDP-11 console switch marked CONT. ODP-143 then types the number sign (#) to signal readiness to accept further commands.

*High-Speed Reader* – The RH command is used to read paper tapes with the high-speed reader, provided that the PDP-8 or PDP-11 computer is equipped with this device. The binary paper tape (output from ODP-143, BOOL-143 or PAL-143) is first placed in the reader unit; the user then types RH, followed by a carriage return. ODP-143 reads the paper tape record into memory and types the period at the left margin to signal readiness for a new command. (The user need not press the CONT switch of the computer as in the R command).

```
#RH
OK
#
```

After placing the paper tape in the high-speed reader, the user types RH, followed by a carriage return. The tape is read and ODP-143 prints OK, carriage return, and then a number sign 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.

ODP-143 types the error message "Checksum Error" when a checksum error occurs while reading a paper tape with the R or 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-143 or PAL-143.

**Change Memory Field**

An F0 or F1 is typed previous to an Open or List command to enable the user to change or to list contents of locations in memory field 0 or 1, respectively. Upon startup, ODP-143 assumes access to memory field 0.

```
#F1
#
```

User can now change or list contents of locations in memory field by typing the appropriate command.

**Open Location Commands**

*Open a Location Symbolically* – A colon (:) typed following a legal Industrial 14 address causes the content of that location to be typed symbolically (with its instruction mnemonic). The user can then alter the content of that location by typing a legal Industrial 14 instruction, next to the content typed by ODP-143, before terminating the line with a carriage return. If it is not necessary to change the content of the location, the user simply types a carriage return after the instruction typed by ODP-143.

```
#10:NOP TN7
#
```

The user changes the content of location 10 from NOP to TN7: modification is terminated by a carriage return and ODP-143 waits for a new command.

Alternatively, a line feed can be used to terminate a line that is typed by ODP-143 and which may or may not have been modified. This causes 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 is also opened symbolically. This feature can be used to scan a series of Industrial 14 instructions by continually typing line feed.

```
#10:TN 007
0011:NOP TN10
#
```

The user opens location 10 but does not modify the content. Termination by a line feed causes the next sequential location to be opened for modification. The user terminates this second line with a carriage return and ODP-143 responds by typing a number sign to request a new command.

The following example shows the manner in which an Industrial 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 can be typed at any time to open the next location, as illustrated.

```
#60:NOP TN2 ↵
#↓
0061:NOP JFF65↓
0062:NOP TN3↓
0063:NOP TF10↓
0064:NOP JFN67↓
0065:NOP SN1010↓
0066:NOP SKP↓
0067:NOP SF1010 ↵
```

The user types a carriage return after modifying location 60; ODP-143 responds with a number sign. The user really wants to enter a group of instructions; he types a line feed and the next location (61) is opened. The user continues to type program instructions, terminating each line with a line feed.

*Open a Location Numerically* – A slash (/) following a legal Industrial 14 address causes the content of that location to be typed numerically. The user can alter the content, close the location unaltered, or close the location (altered or unaltered), and open the next sequential location. This command can also be used with the colon (:) to open locations in both symbolic and numeric form.

```
#65/3010↓
0066/0010↓
0067/2010 ↵
#
```

The user opens three sequential locations, numerically, by placing a line feed after each line except the last. No modifications are performed in this example although the user does have this option.

The following example illustrates both the symbolic and numeric opening of registers. The same register can be opened in both forms on the same line. No modifications are performed in the example although they could be made at any time.

```
#60/6002↓
0061/2065↓
0062/6003↓
0063/5010 TF1010↓
0064:JFN070↓
0065:SN1010/3010↓
0066/1110↓
0067/2010↓
0070/2010↓
0071/0010 ↵
#
```

The user opens 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.

**Open I/O Status Word**

A right hand caret (>) following a legal Industrial 14 I/O address causes ODP-143 to print the type of preset internal function and its I/O status word. This command is normally used to examine or alter timer and counter presets and counts. Opening a timer and counter even address I/O status word reveals the current time or count, in decimal. Opening a timer and counter odd address I/O status word reveals the timer or counter preset.

After opening the I/O status word, the user can alter the type of internal function (CNTR to TSEC) and/or preset or count, close the location (altered or unaltered), or open the next sequential location.

*Example 1*

To examine preset of Timer 1601, type:

```
#1601 > TSEC 0600
```

Open Timer 1601 odd status word. ODP-143 prints that the internal function is a 1/10 second timer (TSEC) and that its preset is 60 seconds.

*Example 2*

To examine current count and preset of Counter 1740, type:

```
#1740 > CNTR 0022↓
1741 > CNTR 0040
```

Open up Counter 1740 even I/O status word. ODP-143 prints that the internal function is a counter and that its current count = 22. By typing a line feed, ODP-143 prints again that the internal function is a CNTR and that its preset = 40.

*Example 3*

To alter a Counter 1660 with a preset of 20 to a timer with a preset of 40 seconds, type:

| | |
|---|---|
| #1660 > CNTR 0011 SEC 0<br>1661 > CNTR 0020 SEC 40 | Open Counter 1660 status word. ODP-143 prints "CNTR" and the current count "0011". The user types SEC, changing the counter to a timer and sets the current time to zero. After pressing the line feed key, ODP-143 prints "CNTR" and the counter preset "0020". The user then types SEC and timer 1660's preset (40 seconds). |

Further discussion concerning I/O status words is presented in Chapter 9.

**List Commands**

*List Symbolically* – 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 limits specified by the LS command. The user specifies the limits of the listing by following the LS with two numbers.

**NOTE**

**The carriage return ↵ must be typed after the LS command and after all other commands described in this chapter.**

| | |
|---|---|
| #LS200–230↵<br>0200:TN011<br>0201:JFF207<br>0202:TN001<br>0203:TN002<br>0204:JFF207<br>0205:SN1001<br>0206:SKP<br>0207:SF1001<br>0210:JFF215<br>0211:NOP<br>0212:NOP<br>0213:NOP<br>0214:NOP<br>0215:JMS<br>0216:1000<br>0217:JFF227<br>0220:JMS<br>0221:1012<br>0222:JFF227<br>0223:TN1001<br>0224:JFF227<br>0225:SN1005<br>0226:SKP<br>0227:SF1005<br>0230:JFF234 | The user requests that ODP-143 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.<br><br>The user lists the complete set of instructions by not altering the initial values of M and W (specifically M = 0000 and W=NOP). |

The user can modify the function of the LS command to list only certain program instructions, or he can 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 locations W and M are modified, the LS command types only certain locations within the limits specified, thereby searching for particular instructions. Table 7-1 lists the LS command useage.

**Table 7-1**
**Use of the LS Command**

| To List: | Change M to: | Change W to: |
|---|---|---|
| All SF instructions | 7000 | SF or 1000 |
| All SN instructions | 7000 | SN or 3000 |
| All JMP instructions | 7777 | JMP or 0024 |
| All JMS instructions | 7777 | JMS or 0124 |
| All JMR instructions | 7777 | JMR or 0056 |
| All JFF or JFN instructions | 7000 | JFF or 2000 |
| All JFF instructions | 7400 | JFF or 2000 |
| All JFN instructions | 7400 | JFN or 2400 |
| All TN or TF instructions | 4000 | TF or 4000 |
| All TF instructions | 6000 | TF or 4000 |
| All TN instructions | 6000 | TN or 6000 |
| Any particular instruction | 7777 | The instruction |
| All locations within limits | 0000 | NOP or 0000 |
| A particular I/O number | 1777 | The I/O number |

The user alters the locations M and W in the same manner that Industrial 14 program locations are altered. The colon or slash is used to open the location; after the location is changed, it is closed with a carriage return.

```
#M/0000 8000↵
#
#W:NOP SF↵
#
```

The mask is changed to 7000.

The word is changed to SF.

The following examples list different instructions within the same limits by changing the M and W locations.

```
#M/0000 7000↵
#W:SF 000 SN↵
#LS200–267↵
0205:SN1001
0225:SN1005
0244:SN1006
0261:SN1007
#
#M/7000 7777↵
#W:SN 000 JMP↵
#LS200–267↵
0267:JMP
#
#M/7777
#W:JMP JFF264↵
#LS200–267↵

#M/7777↵
#W:JFF 264 JFN264↵
#LS200–267↵
0256:JFN 264
#
```

The user changes M and W so that only the SN instructions are listed.

The user changes M and W to list only the JMP instructions within the limits of the command.

The user changes the content of M and W to have ODP-143 type all the JFF 264 instructions within the limits. No such instructions exist, however, and ODP-143 types the period to request a new command.

The user changes the content of W to list the JFN 264 instructions and ODP-143 types the single JFN 264 instruction contained within the specified limits.

*List Numerically* – If a section of the Industrial 14 program is to be listed in numeric form, the LN command is used. All locations within the limits specified will be typed. The LN command is subject to the M and W locations described for the LS command.

The following examples illustrate the use of the LN command.

```
#LN200–235↵
0200/6011
0201/2207
0202/6001
0203/6002
0204/2207
0205/3001
0206/0010
0207/1001
0210/2215
0212/0000
0213/0000
0214/0000
0215/0124
0216/1012
0217/2227
0220/0124
0221/1012
0222/5227
0223/7001
0224/2227
0225/3005
0226/0010
0227/1005
0230/2234
0231/0000
0232/0000
0233/0000
0234/0124
0235/1000
#
#M/0000 7777↵
#W:NOP JMS↵
#LN200–235↵
0215/0124
0220/0124
0234/0124
#
#M/7777↵
#W:JMS SKP↵
#LN200–235↵
0206/0010
0226/0010
#
```

The user requests that ODP-143 list the locations from 200 to 235, inclusive. All locations are typed (numerically) if the user has not altered the content of M and W. Otherwise, each instruction within the limits is 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.

The user requests that ODP-143 type numerically all JMS instructions within the limits specified.

The user requests that ODP-143 type all SKP instructions numerically.

**Interrogation Command**

To interrogate consecutive input, output and internal function states, type I, followed by the limits of inputs, outputs, or internal function states desired. ODP-143 will output the desired input, output, or internal function number and then its state, N or F. This command can be issued in both Program and Run modes.

```
#I 200↵

200 N
```

Interrogate the single input 200 and type its state. ODP-143 types the input (200) and its state (N).

```
#
```

A number sign (#) is typed when ODP-143 is ready for the next command.

```
#I 1200–1210 ↵
   1200 N
   1201 N
   1202 F
   1203 F
   1204 F
   1205 F
   1206 F
   1207 F
   1210 N
```

Interrogate outputs 1200 through 1210 and type out the respective states. ODP-143 types outputs 1200 through 1210 and the state of each.

```
#I 1450 ↵
   1450 F
#
```

Interrogate the single internal function 1450 and type its state. ODP-143 types the internal function (1450) and its state (F). A number sign is typed when ODP-143 is ready for the next command.

**I/O Interrogate Disables**

The Interrogate Disables command interrogates and lists all inputs or outputs that are currently disabled. This command is helpful when the user desires to reenable inputs and outputs before or during execution of the Industrial 14 program. The following example illustrates a typical response to this command.

```
#ID ↵
   10 D
   47 D
   50 D
   66 D
  150 D
  157 D
  305 D
  711 D
  721 D
  723 D
 1127 D
 1300 D
```

User types ID↵. ODP-143 prints each input or output I/O number and the letter D, indicating that the input or output is disabled.

**Start Command**

To enter Run mode and start Industrial 14 program execution, type S, followed by a carriage return. ODP-143 starts execution of the program in Industrial 14 memory at location 0 (places the Industrial 14 in internal mode.) To restart program execution following a halt of the program, the S command turns outputs ON that were ON when the program was halted.

```
#S ↵
@
```

The user types S↵. ODP-143 starts execution of the program in Industrial 14 memory and types an @, indicating that it has entered Run mode.

**Punch Commands**

*Low Speed Punch* – The user can generate a paper tape record of his Industrial 14 program on the low-speed punch by using the following ODP-143 procedure:

1. Type P, followed by a carriage return.
2. Turn the paper tape punch unit of the Teletype ON.
3. Press the CONT switch of the PDP-8 or PDP-11 computer console.
4. After the tape is punched, turn the punch unit OFF, and remove the punched tape.
5. Press the PDP-8 or PDP-11 CONT switch.

This procedure is illustrated as:

```
#P ↵
DONE
#
```

The user requests that ODP-143 punch a paper tape record of the program instructions in Industrial 14 memory. When the tape has been punched, ODP-143 types DONE, carriage return, and then a number sign (#) to request a new command.

*High Speed Punch* – To punch a paper tape record on the high-speed punch, type PH (punch high) followed by a carriage return. ODP-143 punches the paper tape without further operator action (it is not necessary to follow the operating procedure given for the P command).

| | |
|---|---|
| #PH ↵ | The user types PH followed by the limits to be punched. The command is terminated by a carriage return. ODP-143 punches the tape, and types DONE, carriage return and then a number sign (#) when it is ready for a new command. |
| DONE<br># | |

The high-speed punch button must be activated to punch the paper tape. It can be shut off after the tape is punched.

**Verify Commands**

Once the user has punched a paper tape record of his program, the following command can be used to validate that no punching errors have occurred.

| | |
|---|---|
| #V ↵<br>#OK | Verify the paper tape in the low-speed reader by comparing it with Industrial 14 memory. ODP-143 prints OK when the tape is verified. |
| #VH ↵<br>#OK | Verify the paper tape in the high-speed reader by comparing it with Industrial 14 memory. ODP-143 prints OK when the tape is verified. |

The first command (V) uses the low-speed reader unit of the Teletype console. When using the V command, the following procedure applies:

1. Type the command V, followed by a carriage return.
2. Place the paper tape in the low-speed reader.
3. Switch the low-speed reader to START.
4. When the tape is read, switch the low-speed reader to FREE, and remove the paper tape.
5. Press the PDP-8 or PDP-11 CONT switch.

When using the VH command, it is not necessary to follow the above procedure. Instead, simply place the paper tape in the reader and type VH, followed by a carriage return. The tape is then read and ODP-143 prints OK after error-free verification.

The tape Verify commands compare the tape being read with the Industrial 14 memory. If the content of the tape does not agree with the content of memory, ODP-143 types "Data Error".

Whenever this message occurs, a new paper tape should be punched and the old tape destroyed. If the error message "Checksum Error" is typed, the paper tape has a bad checksum punched at its end, and a new tape should be punched.

It is good practice to verify the paper tapes, generated during the debugging state, to be certain that they are accurate. Punching errors, which cause tape verification errors, do not occur often, but such errors can change the user's program with dangerous results.

**RUN MODE COMMANDS**

The following set of ODP-143 commands enables the user to:

a. Interrogate input, output, and internal function states

b. Enable and disable inputs and outputs

c. Force ON and OFF inputs or outputs

d. Halt Industrial 14 program execution.

Except for Interrogation commands, these commands may only be used in Run mode. The presence of Run mode is exemplified when an @ is typed on the Teletype.

**Interrogation Command**

To interrogate one, or a series of consecutive input, output and internal function states, type I, followed by the limits of the I/O number of inputs, outputs or internal functions desired. As in the Program mode, ODP-143 will output the desired input, output or internal function I/O number and then its state. Refer to Interrogation Command in Program mode for example.

**ID Interrogate Disables Command**

The Interrogate Disables Command interrogates and lists all inputs or outputs that are currently disabled during execution of the Industrial 14 program. This command is identical to the Interrogate Disables Command, available in Program mode. Refer to Interrogation Disables Command in Program mode for examples.

**Disable and Enable Commands**

ODP-143 can logically override the input or output state of devices wired to the Industrial 14 Controller; internal functions cannot be overridden in this fashion. The logical override is accomplished by first disabling the input or output and then forcing it to either the ON or OFF state. Figure 7-1 illustrates this feature.

For an input, the state, as tested by the controller, is that state of the input converter when the input is not disabled and is the forced state when the input is disabled. For an output, the state of the output converter is as set by the control program when the output is not disabled and is the forced state when the output is disabled.

To disable an input or output, type DI, followed by the desired input or output number (I/O numbers range from 0–1377). To enable a disabled input or output, type EN, followed by the desired input or output number.

| | |
|---|---|
| @DI 275 ↵ | Disable input 275 |
| @EN 1312 ↵ | Enable output 1312 |
| @ | |

Any combination of inputs and outputs may be disabled at the same time. The "disable" remains until it is removed.

**Force Commands**

After the input or output is disabled, it can be forced either ON or OFF. The current state of an input or output (found by issuing an interrogation command) is the forced state if it is disabled, or the true state if it is not disabled. This state is also used by the controller when processing Industrial 14 instructions.

Any output or internal function may be forced to its opposite state, whether disabled or not. However, if the output is currently enabled, the control program can return the output to its original state, almost immediately. This feature is particularly useful for clearing or setting internal functions such as timers, counters, retentive memories, shift registers, etc., which cannot be disabled. It is also helpful for turning ON an output that uses a sealing contact.

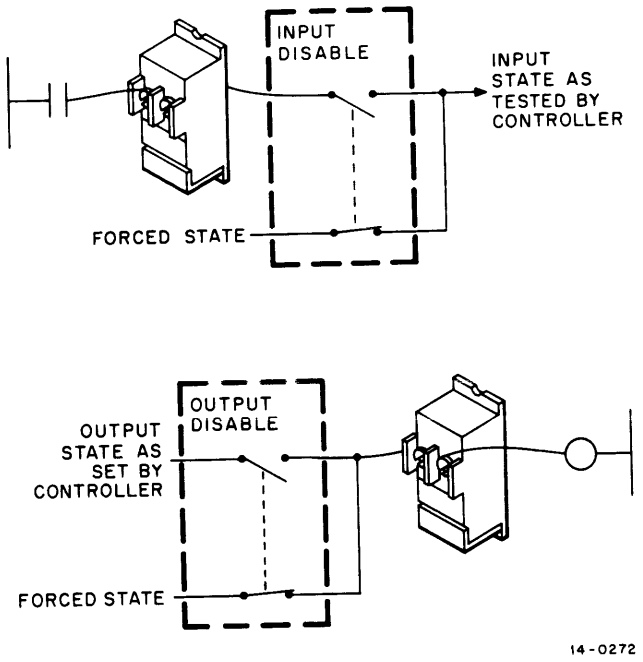



Figure 7-1 Disabling Inputs and Outputs

To force an input, output, or internal function ON, type FN, followed by the desired I/O number (I/O numbers range from 0–1777).

To force an input, output, or internal function OFF, type FF, followed by the desired I/O number.

| | |
|---|---|
| @FN 1612 ↵ | Assuming the timer and counter partition to be 1600, force timer or counter 1612 ON. |
| @FF 15 ↵ | Force input 15 OFF. |
| @ | |

**Halt Program Execution Command**

The execution of an Industrial 14 program can be halted by typing HP. ODP-143 turns all outputs electrically OFF but retains the logic state for interrogation; it then enters Program mode and places the Industrial 14 into External mode. Thus, all outputs should be forced OFF before halting the Industrial 14 program. Internal functions retain the current state. If the program is restarted again (by typing S ↵ ) all outputs are switched ON according to their

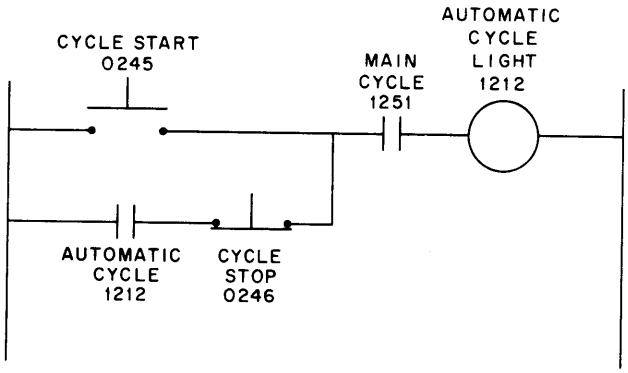
previously established states. If program execution is to start with all outputs OFF, simply power the Industrial 14 down, then back up.

**NOTE**
**ODP-143 must be in Run mode**

| | |
|---|---|
| @HP ↵ | Industrial 14 program execution stops |
| # | Enter Program mode |

**Using the Disable/Force Feature**

A practical example of the preceding discussion follows. Figure 7-2 shows a control circuit, a Boolean equation, and Industrial 14 machine language that turns ON the automatic cycle light for one station on a transfer line. To turn the automatic cycle light ON, without actually having either the MAIN CYCLE or CYCLE START pushbutton ON, first disable and then force ON the CYCLE START and MAIN CYCLE contacts.




```
1212=1251*(245+[1212*/246])

200 TF   1251      205 TN   246
201 JFN  211       206 JFN  211
202 TF   245       207 SN   1212
203 JFF  207       210 SKP
204 TF   1212      211 SF   1212
```

14-0273

Figure 7-2 Automatic Cycle Light Control Circuit, Boolean Equation and Industrial 14 Instructions

To validate that the automatic cycle circuit remains ON after the CYCLE START pushbutton is released, force the CYCLE START contact 245 OFF. To turn the automatic cycle light OFF, disable the CYCLE STOP contact 0246 and force it OFF. After the circuit is checked, the disabled inputs and outputs must be reenabled.

Other Disable and Force features follow:

a. During checkout, to prevent an output from energizing at all, disable it and force it OFF.

b. During checkout, to achieve a dry run when the logic requires that a part be in place, disable the part-in-place input and force it ON.

c. During checkout, when some inputs required are not yet wired into the controller, disable the inputs that should be ON, and force them accordingly.

d. When an input is suspected to be intermittent, disable it and force it to the state it should be in. If the problem disappears, the input is intermittent, and the switch and/or input converter should be checked.

I/O disables only remain in effect as long as the Industrial 14 is powered up. To clear all I/O disables, power down the controller with the power supply ON/OFF switch.

## ODP-143 SUMMARY

Tables 7-2 through 7-4 summarize the commands and error messages for ODP-143. Specific values for inputs, outputs and program numbers have been used to describe the commands and error messages. Any Industrial 14 location address, input number or output number may be substituted in place of those used for illustration.

**Table 7-2**
**Program Mode Commands**

| Command | Function |
|---|---|
| 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 Industrial 14 simulated memory, noting any discrepancies. |
| VH | Verify the paper tape in the high-speed reader by comparing it with the Industrial 14 simulated memory, noting any discrepancies. |
| 23: | Open location 23, symbolically, and allow the user to modify its contents. |

**Table 7-2 (Cont)**
**Program Mode Commands**

| Command | Function |
|---|---|
| 57/ | Open location 57, numerically, and allow the user to modify its contents. |
| 1620> | Open status word for I/O address 1620 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 Industrial 14 location in the same form (symbolically or numerically) and allow the user to modify its content. |
| LS0–30 | List the contents of locations 0–30, symbolically, (by typing the instructions in mnemonic form). |
| LN0–10 | List the contents of locations 0–10, numerically, (by typing the instructions in numeric octal form). |
| P | Punch a paper tape record of the contents of Industrial 14 memory on the low-speed punch. |
| PH | Punch a paper tape record of the contents of Industrial 14 memory 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 a mask location and allow modification. |
| W: | Open a work location and allow modification. |

**Table 7-2 (Cont)**
**Program Mode Commands**

| Command | Function |
|---|---|
| ZM | Zero Industrial 14 memory |
| I40–45 | Interrogate the current states of inputs 40–45 |
| ID | Interrogate and list all currently disabled inputs and outputs. |
| S | Start Industrial 14 program execution at address 0 (places Industrial 14 in "internal" mode). |
| PA | Specifies that ODP-143 will communicate with the 14 via the DA14-E parallel interface. |
| SO | Specifies that ODP-143 will communicate with the 14 through the serial line interface via the utility port (output register 1). |
| SS | Specifies that ODP-143 will communicate with the 14 through the serial line interface via the monitoring port (output register 6). |
| F1 | Change to memory field 1, in order to open or list contents of locations in that field. |

**Table 7-3**
**@ Run Mode Commands**

| Command | Function |
|---|---|
| I1200–1220 | Interrogate the current state of outputs 1200–1220. This command can also be issued in Program mode. |
| ID | Interrogate and list all currently disabled inputs and outputs. |
| D150 | Disable input 50. |
| EN1300 | Enable output 1300. |

**Table 7-3 (Cont)**
**@ Run Mode Commands**

| Command | Function |
|---|---|
| FN 1660 | Assuming timer and counter partition to be at address 1600, force timer or counter 1660 ON |
| FF1050 | Force output 1050 OFF |
| HP | Halt Industriai 14 program; ODP-143 enters Program mode |

**Table 7-4**
**ODP-143 Error Messages And Causes**

| Error Message | Cause |
|---|---|
| S? | *Syntax Error* – The user has violated a syntax rule for ODP-143 commands. Each ODP-143 command is in one of the following forms:<br>a. One or two letters with no following numbers (for instance, ZM)<br>b. One or two letters followed by a number (for instance, DI1234)<br>c. One or two letters, followed by two numbers, separated by a minus sign (for instance, I 1–20). The first number must be less than the second number.<br>When a command is not typed in its specified format, the S? error results. Commands of form c can be typed in form b without error (for instance, I5 is the same as I5-5). |
| N? | *Number error* – The user has included an illegal number in his program. The following conditions produce the code N?: |

**Table 7-4 (Cont)**
**ODP-143 Error Messages And Causes**

| Error Message | Cause |
|---|---|
| | a. A non-octal digit is used in a program (8 or 9).<br>b. A test or set instruction references an illegal input or output number (greater than 1777).<br>c. The address of a JFF or JFN is illegal (greater than 377).<br>d. A JMP or JMS instruction references a location which is not in the simulated memory.<br>e. The second part of a two-word instruction is typed on the same line as the first part (for instance, JMP 1567). |
| DATA ERROR | A data error has occurred on a tape read by ODP-143 with the V or VH command. The tape should be repunched. |
| NO FIELD 1 | An attempt has been made to read or verify an 8K Industrial 14 memory. |
| 14 STOPPED | ODP-143 has attempted to communicate with an Industrial 14 that is not running or is not interfaced to the proper output port. |
| 14 HUNG | ODP-143 has attempted to communicate with an Industrial 14 that is not interfaced to the proper output port. |
| CHECKSUM ERROR | A checksum error has occurred on a tape read by ODP-143 with the R, RH, V or VH command. The tape should be reread. (If R or RH, ZM should be executed first). |

Table 7-4 (Cont)
ODP-143 Error Messages And Causes

| Error Message | Cause |
|---|---|
| M? | Mode error – The user has typed a command which is illegal in the present mode of ODP-143. ODP-143 has two such modes: Program and Run. |
| C? | Command Error – The user has typed a command that ODP-143 does not recognize. |

# CHAPTER 8
# PDP-8 OPERATIONS

This chapter describes the use of PDP-8 utility programs developed for the Industrial 14 System (Loader and Editor). The operation of the PDP-8 hardware system (computer console and 33 ASR Teletype) is described in the *PDP-8/E, PDP-8/M, and PDP-8/F Small Computer Handbook.*

**PDP-8 SWITCH AND SWITCH REGISTER OPERATIONS**

The PDP-8 is operated by a series of switches on the computer console. For Industrial 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 specify when these switches are to be used. The switches are all spring-loaded and return to the normal position after release.

The Industrial 14 software will not run if the SING STEP or SING INST switches are set on the PDP-8 console. These switches must always remain OFF. (The SING INST switch does not exist on the PDP-8/L, 8/E, 8/F, or 8/M consoles.) In the PDP-8/I, the top of these two switches should be in the IN position, with the bottom OUT. In the PDP-8/L, 8/E, 8/F and 8/M, the SING STEP switch should be in the UP position.

The twelve switches of the PDP-8 Switch Register are arranged in four groups of three switches each. Adjacent groups of switches are of different colors to aid in identification.

Figure 8-1 shows every possible switch pattern for a 3-switch group. Eight patterns exist and these are numbered 0–7. The chart is essentially the octal-binary equivalents for the digits 0–7, with 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, 8/E, 8/F and 8/M switch is ON when it is in the UP position.**

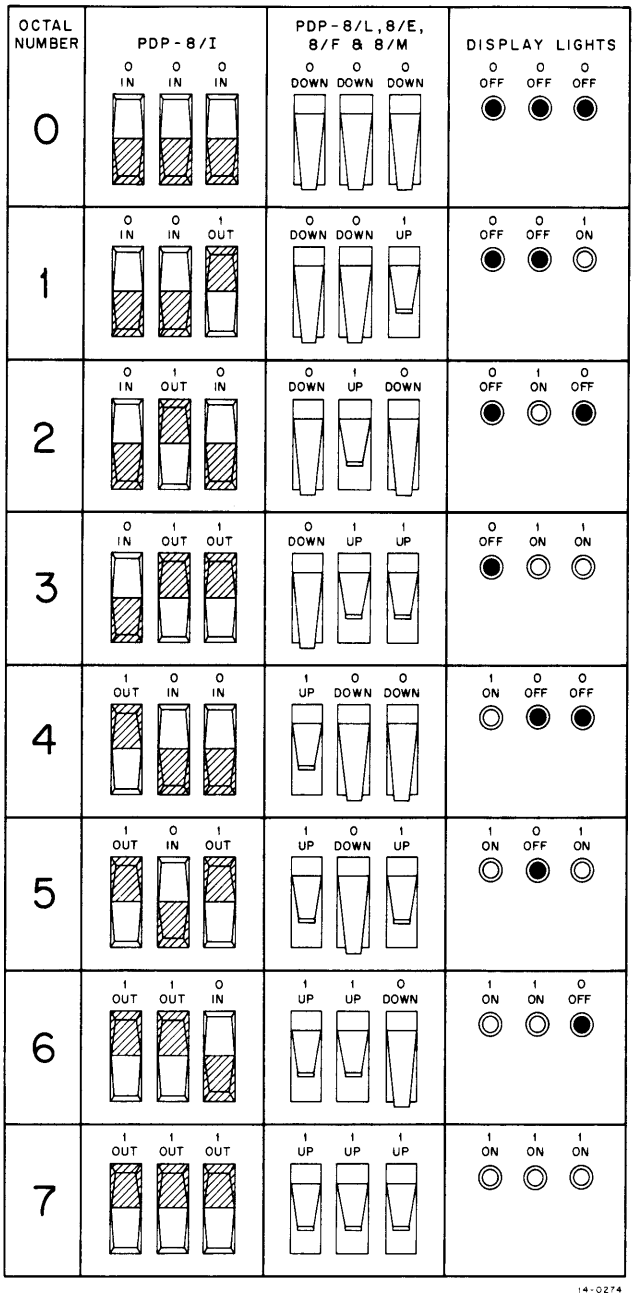



Figure 8-1 PDP-8/I, 8/L, 8/E, 8/F and 8/M Switch Settings for Octal Digits 0–7

The following example further illustrates the use of Figure 8-1. To specify the octal number 5 (corresponding to the binary code 101), the switches, as shown in Figure 8-2, are set as follows:

Switch 1 = ON
Switch 2 = OFF
Switch 3 = ON

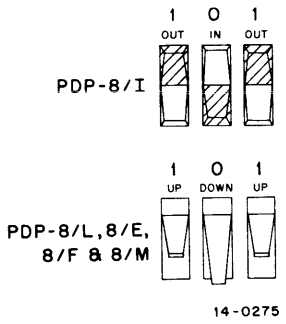



Figure 8-2 Switches Set to Octal Number 5

As previously stated, the Switch Register consists of 12 switches, arranged in four 3-switch groups. Each group of switches represents an octal digit. Thus, a four-digit octal number defines the settings for all 12 switches. Figures 8-3, 8-4 and 8-5 show the switch settings for three octal values.

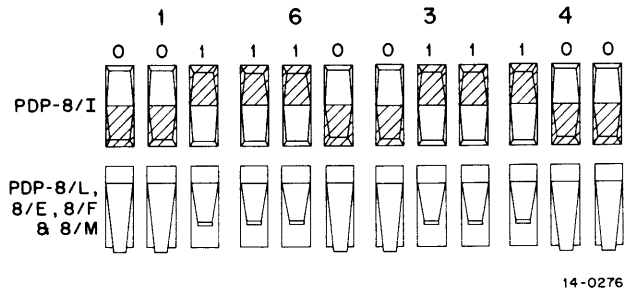



Figure 8-3 Switch Register Set to $1634_8$

## PDP-8 LOADER PROGRAM

Before any PDP-8 based program can be loaded into the PDP-8 memory, a loader program must be stored that will read the binary format paper tape on which these programs are stored. The loader program is then used to transfer BOOL-143/8, ODP-143/8 and all other system software into the PDP-8 memory.

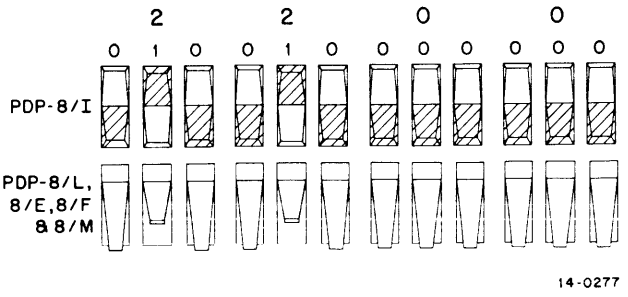



Figure 8-4 Switch Register Set to $2200_8$

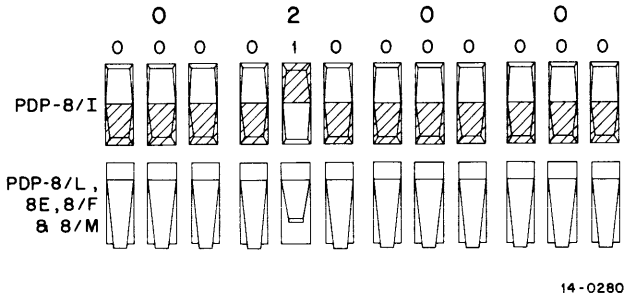



Figure 8-5 Switch Register Set to $0200_8$

### Loading the RIM Loader

To store the RIM loader in memory, proceed as follows.

Before the loader program is stored in PDP-8 memory, the Data Field and Instruction Field switches should be set to 0 (if the computer has 4K of memory) or to 1 (if the computer has 8K of memory). If a PDP-8/E, 8/F, or 8/M is used, the Data and Instruction Fields are set by Switch Register switches SR 6–11 and Extended Address Load Key (EXTD ADDR LOAD). First, set the Indicator Selector switch to STATUS, then set SR 6–8 for the desired Instruction Field. SR 9–11 should be used to indicate the desired Data Field. To specify both Instruction Field 1 and Data Field 1, set only bits 8 and 11 to 1; then press the EXTD ADDR LOAD key. If a PDP-8/L is used, the MEM PROT (memory protection) switch must be in the DOWN position (off) while the loader program is being stored in memory.

Set the Switch Register to either the low-speed or the high-speed RIM Loader settings, listed in the following table, depending on whether the system is equipped with a low-speed Teletype reader or a high-speed paper tape reader. After each setting, press the operation switch noted at the right of the setting.

| Switch Register Setting | | |
|---|---|---|
| Low Speed | High Speed | Operation Switch |
| 7756 | 7756 | LOAD ADD |
| 6032 | 6014 | DEP |
| 6031 | 6011 | DEP |
| 5357 | 5357 | DEP |
| 6036 | 6016 | DEP |
| 7106 | 7106 | DEP |
| 7006 | 7006 | DEP |
| 7510 | 5510 | DEP |
| 5357 | 5374 | DEP |
| 7006 | 7006 | DEP |
| 6031 | 6011 | DEP |
| 5367 | 5367 | DEP |
| 6034 | 6016 | DEP |
| 7420 | 7420 | DEP |
| 3776 | 3776 | DEP |
| 3376 | 3376 | DEP |
| 5356 | 5357 | DEP |

The first number listed is the initial address where the instructions are to be stored. The numbers that follow are instructions to be deposited in the PDP-8 memory.

When the instructions are loaded, validate that the program was loaded correctly by examining the stored instructions. When the EXAM switch is pressed, the lamps of the memory buffer console display light, indicating the stored instruction. The light patterns correspond to the following table, if the instructions have been correctly loaded.

| Switch Register Setting | Operation Switch | |
|---|---|---|
| 7756 | LOAD ADD | |
| | Memory Buffer Display | |
| Operation Switch | Low Speed | High Speed |
| EXAM | 6032 | 6014 |
| EXAM | 6031 | 6011 |
| EXAM | 5357 | 5357 |
| EXAM | 6036 | 6016 |
| EXAM | 7106 | 7106 |
| EXAM | 7006 | 7006 |
| EXAM | 7510 | 7510 |
| EXAM | 5357 | 5374 |
| EXAM | 7006 | 7006 |
| EXAM | 6031 | 6011 |
| EXAM | 5367 | 5367 |
| EXAM | 6034 | 6016 |
| EXAM | 7420 | 7420 |
| EXAM | 3776 | 3776 |
| EXAM | 3376 | 3376 |
| EXAM | 5356 | 5356 |

If the instructions have been loaded incorrectly, restart the switch-setting process. If the RIM instructions have been loaded correctly, the Binary Loader can be stored in memory using the RIM load program.

**Loading a Binary Tape**

All Industrial 14 System paper tape software contains the self-starting Binary Loader program at its beginning. The self-starting Binary Loader, however, can only be loaded into PDP-8/E, 8/F and 8/M computers. The regular Binary Loader Program (DEC-08-LBAA-LM) must be loaded into the PDP-8/I or 8/L computers.

*Procedure for loading the regular Binary Loader*

1. Place the Binary Loader paper tape (DEC-08-LBAA-LM) in the paper tape reader and turn the reader ON

2. Set the Switch Register to 7756 and press the LOAD ADDR and START switches on the computer console.

3. The paper tape containing the loader will then be read.

4. When the tape has been read, turn off the reader and remove the tape.

Now any system program can be stored in PDP-8/I or 8/L memory.

If a PDP-8/L is used for programming, the MEM PROT switch on the PDP-8/L console should be moved to the UP position once the loader is in memory. This prevents unwanted destruction of the loader program. This switch is not available on the PDP-8/I computer.

*Procedure for loading a system software paper tape into the PDP-8/L or 8/I memory*

1. Place the system tape in the paper tape reader and turn the reader ON. The tape must be positioned with the leader/trailer code between the Binary Loader and the Binary System Program over the reading head (Figure 8-6).

2. Set the PDP-8/L or 8/I Switch Register to 7777 and press the LOAD ADDR switch on the computer console.

3. If the high-speed reader is being used, set SR 0 = 0.

4. Press the START switch on the computer console.

5. If this procedure has been followed correctly, the tape is read into memory.

6. When the tape reaches the end, it will stop. If any lamp of the accumulator display on the PDP-8/I or 8/L console (not LINK display) remains lit, a reading error has occurred and the tape must be reread. If a lamp remains lit, a checksum error has occurred.

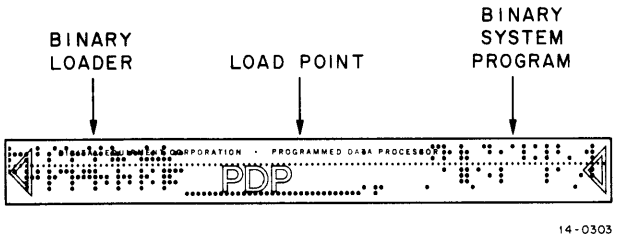



Figure 8-6 Paper Tape Load Point

*Procedure for loading self-starting binary tapes*

1. Place an Industrial 14 System tape in the reader and turn the reader ON.

2. Set Instruction Field and Data Field on the PDP-8/E, 8/F or 8/M console to the field where the RIM loader was deposited.

3. Set the Switch Register on the computer console to 7756 and press the LOAD ADDR switch on the PDP-8/E, 8/F, or 8/M console.

4. If the high-speed reader is being used, set SR 0 = 0.

5. Press the START switch on the computer console and the Industrial 14 System tape will read in.

6. When the Industrial 14 System tape has been read in, the system program will self-start, activating the printer.

7. If there is no response, check the PDP-8/E, 8/F or 8/M accumulator for a checksum error (lamps lit in the accumulator) and reload the Industrial 14 System program.

## PDP-8 EDITOR

Industrial 14 users of BOOL-143 and PAL-143 must prepare a tape record of their program. This paper tape is referred to as a *program source tape* or *source language tape.* BOOL-143 and PAL-143 translate the source tape into the machine language or binary tape used by the Industrial 14. Machine language is the form in which Industrial 14 programs are debugged with ODP-143.

### Composing Source Programs

The PDP-8 Editor program can be used to compose programs in the BOOL-143 and PAL-143 source language. The Editor enables the Industrial 14 user to type the program on the Teletype keyboard and to correct typing errors as they occur. Sections of the program can be moved or deleted by typing single Editor commands. Program source tapes are generated by the Editor and are used as input to BOOL-143 or PAL-143.

### Updating Source Programs

A second, and important function of the Editor is updating user programs. During the life of an Industrial 14 program, many changes will be made to the original version. Errors uncovered with ODP-143, changes in machine function, and additions of new inputs or outputs to the machine all require changes to the Industrial 14 program. These changes can be made with ODP-143; however, no permanent record of the altered program will exist. 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 Industrial 14 programs typed by the user, 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 decimal-numbered; each line is referenced by its line number. By referring to specific line numbers, the Editor commands can be used to change, list, and delete lines of text, and to insert lines of text before a specified line.

### Modes of Operation

Because the Editor accepts both commands and text from the Teletype keyboard, it must recognize whether the characters currently being typed are program text, being created or modified by the user, or commands, directing the Editor to perform text functions. The Editor makes this distinction by dividing its operation into two modes. In Command mode, all characters typed on the Teletype keyboard are interpreted as commands to the Editor to

perform some operation on the content of the text buffer, or to allow the user to operate on the text. In Text mode, all characters typed on the keyboard are entered into the text buffer. Text can replace, be inserted into, or be added to the current content of the text buffer.

The current mode of the Editor must always be noted. 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 enters the 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. In Text mode, all characters typed are entered in the text buffer. To exit from Text mode, press the CTRL key of the Teletype keyboard while striking the FORM key (CTRL/FORM). The Editor responds by ringing the Teletype bell and waits for a new command (Figure 8-7).

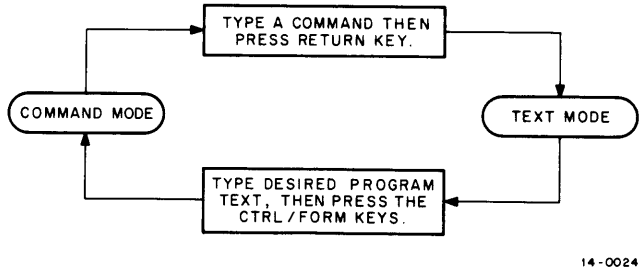



Figure 8-7 Transitions Between Editor Modes

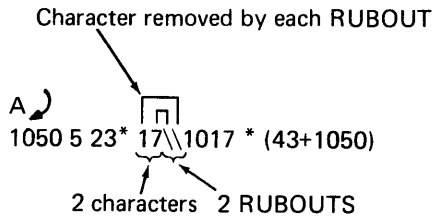
**Adding Text to the Buffer**

To compose Industrial 14 programs with the Editor, type A. The A command instructs the Editor to enter Text mode and to add all of the text typed on the keyboard to the text buffer. The new text is added at the end of any text currently in the buffer.

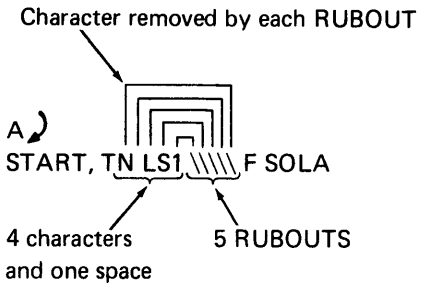
**Editing Keys**

Typing errors can be erased with the RUBOUT key. This key deletes (to the left) one character, or space, each time it is struck. The RUBOUT key can delete characters in only one line of text. The Editor types a backslash (\) each time the RUBOUT key is struck. The following examples illustrate the use of the RUBOUT key.

*Example 1*




*Example 2*




In the above examples, the user enters Text mode with the A command and begins typing a BOOL-143/8 program (Example 1) or a PAL-143/8 program (Example 2). An error is made, however, and the user deletes the incorrect information by repeatedly striking the RUBOUT key, erasing all characters back to the * in the BOOL-143 program and back to the T in the PAL-14 program. The correct characters are then typed and the line is terminated by typing RETURN. The correct versions of the above examples are as follows:

```
1050 = 23 * 1017 (43 + 1050) ↵
or
START, TF SOLA ↵
```

The backarrow (←) is also used to erase typed characters from the text buffer. It is typed by holding the SHIFT key down and striking the letter O key. This special character deletes the complete line of characters between the left margin and itself. The following example illustrates the use of the backarrow:

```
1005 = 2 +/(16 * 1003) ← ;THE FOLLOWING
                         ;EQUATION CONTROLS
                         ;SOLENOID A.↵
```

In the above example, a complete line of text is deleted, using the backarrow, and a comment is inserted in its place. The line of the text buffer reads as follows:

```
;THE FOLLOWING EQUATION CONTROLS
;SOLENOID A.
```

**Reading a Paper Tape**

Source tapes, previously generated by the Editor, can be read into the text buffer for modification or addition by using the command:

R↵

The R command (read) causes the paper tape to 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 rings the Teletype bell and enters Command mode.

The R command reads a paper tape from the high-speed reader when the last switch of the Switch Register (SR 11) is set ON. Otherwise, the tape is 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 following command.

L↵

The complete content of the text buffer is typed on the Teletype printer.

The L command can be modified to type only certain lines by specifying limits for the list. The limits are typed preceding the L, 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 decimal numbered, starting with line 1. After the Editor has typed the text, it enters Command mode.

**Inserting a New Line of Text**

Industrial 14 programs can 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, the command

15I↵

allows one or more new lines of text to be typed before line 15 of the current text buffer.

Editor remains in Text mode and accepts all characters and lines typed by the user into the text buffer. When all lines are inserted in the text buffer, type CTRL/FORM to return the Editor to Command mode.

The Editor "pushes down" text in the buffer when the I command is used. The new text is automatically numbered, thereby changing the previously assigned line numbers. Thus, if one line is inserted before line 15 (15I↵), line 15 becomes line 16 and the newly typed line becomes line 15.

**Deleting Lines from the Text Buffer**

Industrial 14 programs can also be modified by deleting lines with the D command. The D command is typed, preceded by the line or lines to be deleted. For example, the command

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.

When a line has been deleted, the remaining line numbers are adjusted immediately. For example, the two commands listed below actually delete lines 16 and 17 of the original text buffer:

16D↵
16D↵

The complete buffer is deleted by the K (kill) command. This command can 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 enables the user to replace a line, or a group of lines, by text typed on the keyboard. For example, the command

15, 17C↵

deletes lines 15 through 17 and then causes the Editor to enter Text mode. The text typed by the user is then inserted as lines 15, 16, 17, 18, etc. The user can type as many lines as needed. When the new text has been typed, strike CTRL/FORM to return to Command mode.

The C command can be used to replace one line with many lines, or many lines with only one; no equivalence is required between the total number of lines deleted and the total number of lines added in their place.

**NOTE**

**Line numbers are automatically adjusted for the new text when the user presses the CTRL/FORM key to return to Command mode.**

**Moving Lines of Program Text**

Occasionally, Industrial 14 programs must be completely reorganized, for instance, to better fit the page structure of the Industrial 14. This reorganization can easily be accomplished with the M (move) command. The M command has a special command format that specifies the line, or group of lines, to be moved and the line number on which the moved lines will be placed. For example, the command

15,27$30M ↵

moves lines 15 through 27 before line 30. The line numbers are automatically adjusted and the Editor returns to Command mode. In the example, line 30 remains line 30; line 28 becomes line 15; line 29 becomes line 16; and lines 15 through 27 become lines 17 through 29.

The M command can move one or more lines before the line number that is preceded by a $ in the command. The Editor automatically returns to Command mode after an M command is executed.

**Search Feature**

The search feature is a helpful feature, available with the Editor, which enables the user to modify a line of text without completely retyping the line. The line number of the line to be searched is typed, followed by S and a carriage return. The user then types the "search character". This character can be any keyboard character (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 occurence of the designated search character. When the Editor locates and types the search character, typing stops and any combination of the following operations can be typed by the user:

a. To change the entire line to the right of the search character, enter new text and terminate the line with the RETURN key.

b. Delete the entire line to the left of the search character using the backarrow (←).

c. Delete the entire line to the right of the search character using the RETURN key.

d. Delete one character (moving from right to left) for each RUBOUT typed.

e. Insert a CARRIAGE RETURN/LINE FEED, thereby dividing the line into two lines.

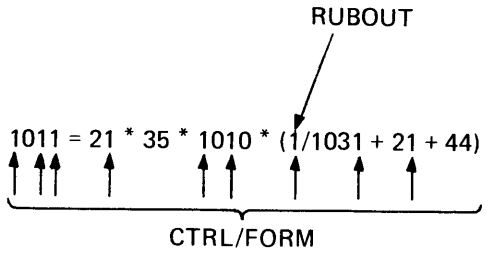
f. Press CTRL/FORM to search for the next occurrence of the search character.

For example, consider the following equation:

1011 = 21 * 35 * 1010 * (131 + 21 + 44)

Assume that 131 should be 1031.

In the example, the number can be selected as the search character. However, since there are six occurrences of number 1 prior to the one to be altered, the CTRL/FORM keys must be used to bypass the six preceding number 1s. Once the seventh 1 is reached, the numbers 1 and 0 are inserted. 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 was used after the letter was changed, the remainder of the line would be deleted.




The intelligent choice of a search character can ease the editing task when the search feature is used. For example, if the number 3 is selected as the search character, one CTRL/FORM is required to get to the second 3. Two RUBOUTS are then typed, thereby removing the 3 and 1. After 103 is typed, CTRL/FORM is used to repeat the remainder of the line.

**Special Characters**

In most cases, use of the Editor commands requires the user to type line numbers. These line numbers are computed by counting the number of preceding lines, or the following special characters can be used:

| | |
|---|---|
| .(period) | Has a value equal to the line number of the last edited line. Thus after a line is edited, it can 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) | Used with the . or / characters to represent line numbers. For example, .+5L means list the fifth line after the last edited line. |
| = (equals) | Used with the . or / characters to find the value of these characters. When the user types .=, the Editor types the decimal number of the current line. When the user types /=, the Editor types the decimal value of the last line in the buffer. |

**Punching a Source Tape**

When the Industrial 14 program has been prepared with the Editor, a paper tape is punched with the P command. This command can be used to punch the complete program (P↵ ) or a group of lines within the program (15,30P↵ ).

The following procedure generates a source tape using the low-speed punch (LSP) on the Teletype:

1. Type T and quickly press LSP ON. The leader/trailer tape will be punched.
2. Press 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 the CONT switch, located on the PDP-8 console. The program will now be punched.
6. Press LSP OFF.
7. Type F and RETURN keys. Turn LSP ON. A special character is typed to signal the end of the tape for the Editor.
8. Turn LSP OFF.
9. Type T and RETURN keys. Turn LSP ON. Leader/trailer tape will be generated.
10. Turn LSP OFF and remove the source tape.

The following procedure generates a source tape using the high-speed punch (HSP).

1. Set the next-to-last switch of the Switch Register (SR 10) ON.
2. Turn 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 can be segmented and punched in sections using the .EOT (BOOL-143/8) or EOT (PAL-143/8) control statements to conclude all but the final segment. Thus, the foregoing procedures can be used to punch all, or part of, an Industrial 14 program. This enables the user to edit and generate tape for programs that are not small enough to fit completely within the text buffer.

**Editor Summary Tables**

Tables 8-1 and 8-2 summarize the characters and symbols used with the Editor. Note that the characters have different meanings when used in Command or Text mode.

**Table 8-1**
**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 |

**Loading The Editor**

The Editor is supplied as a binary paper tape (DEC-08-ESAB-PB) and is loaded with the Binary loader program. It operates 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 ADDR and START switches are then pressed, in that order. The Editor prints a carriage return and waits for a typed command from the user. All further operation is controlled by commands.

The paper tape input and output are controlled by the last two switches of the Switch Register. If the high-speed reader/punch is 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 publication Introduction to Programming (Pages 5-11 through 5-42).**

**Table 8-2**
**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 |
| | 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; m is smaller than n; k is a decimal number.

The P command halts the Editor to enable 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.

## SET-143/8 OPERATIONS

### Loading and Starting SET-143

SET-143 is supplied as a self-starting binary paper tape and is loaded into Field 0 of the PDP-8 with the RIM loader. To load the tape, use either the paper tape reader on the 33 ASR Teletype or the high-speed reader.

If the high-speed reader is to be used for input, the source program paper tape should be loaded in the high-speed reader before answering the queries described in the following:

When started, SET-143 responds by typing three queries:

| | |
|---|---|
| *IN-LH? | What device will be used to supply the source tape? Will it be the low-speed paper tape reader (L) or the high-speed paper tape reader (H)? |
| *OUT-LH? | What device will be used to punch the output tape? Will it be the low-speed punch (L), the high speed-punch (H), or will there be no punched output (N)? |
| *SYM-Y, N? | Does the user want both the symbol table and the sorted symbol table output, yes (Y), or no (N)? |

When the third query has been answered, SET-143 types the message:

TURN ON PUNCH

and halts. Always validate that the input tape is loaded in the proper reader, and turn on the proper punch according to the query responses (L or H). When the PDP-8 CONT switch is pressed, SET-143 punches the leader tape and begins the translation.

If the low-speed output is used, a copy of the output tape is typed as the tape is being punched. Any error messages are typed and punched in the format of BOOL-143 comments. At the end of each symbol table input, the number of errors detected is also typed as a comment. If high-speed output is used, no record is typed during translation. A record can be listed off-line on the Teletype at any time. The only message typed on the Teletype is the number of errors detected after the symbol table input.

SET-143 halts on each .EOT statement to allow a new input tape to be loaded. Pressing CONT causes SET-143 to continue with the translation process, ignoring all blank tape (leader and trailer).

SET-143 stops processing on an .END or .ENDN statement and punches the trailer tape. At this time, a complete, alphabetized, symbol-assignment listing is generated if the user has answered the query SYM-Y, N? with a Y (yes).

### Using SET-143

Because SET-143 generates a symbolic tape, both the output tape and the input tape can be edited with the PDP-8 Editor. Program mistakes in BOOL-143 and ODP-143 can be corrected by using the Editor to change the input to SET-143 or to change the output from SET-143 (which is the input) to BOOL-143. When the SET-143 input tape is edited, only the symbolic equation and/or symbol table must be changed. When the output tape is edited, the symbolic equation and/or symbol table and the control equations must be changed.

SET-143 also facilitates major changes to the input or output terminals, which are wired to machine inputs and outputs. The symbol table can be modified and the tape translated to incorporate the new I/O numbers.

SET-143 also allows the user to store proven control approaches in an untranslated form. The control for standard units or processes can be recorded in a SET-143 input format of symbolic equations. When a particular application arises for the unit or process (as part of a larger system, for example), the symbols may be assigned an input and output designation, and the control equations for BOOL-143 can be generated.

## BOOL-143/8 OPERATIONS

### BOOL-143/8 Options

Options available to the user are requested through a series of four queries which are typed to start the BOOL-143/8 compilation process. If the user types a response other than a legal response to a query, BOOL-143/8 will retype the query.

Example:

| | |
|---|---|
| *SRC-LH? N ↵ | N is not a legal response to the SRC query, |
| *SRC-LH? | BOOL-143/8 will retype the query. |

If the user types more than one character in response to a query, BOOL-143/8 recognizes only the last character typed before the carriage return. Therefore, if a wrong response is typed, the user can correct it by typing the correct character, prior to the carriage return.

**Binary Output**

The binary paper tape output from BOOL-143/8, which is input to ODP-143/8, or to the VT14 Programming Terminal, is requested in response to the query:

*BIN-NLH?

N, L, and H are the possible user-supplied responses and have the following meanings:

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 three 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 can compile the program and obtain 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.

**Compiler Listing**

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

*LST-NLH?

where N, L, and H have the following meanings:

N No listing output is requested; BOOL-143/8 types only the lines which contain errors and the applicable error number.

L The complete listing of source statements, compiled machine code instructions and error messages is typed on the Teletype.

H The complete listing is punched by the high-speed paper tape punch. Errors are typed on the Teletype.

The user types one of the three responses, followed by a carriage return. If only an error listing is wanted, the user types N. BOOL-143/8 types only those lines of the source program which contain errors. For example, BOOL-143/8 might type:

1010 = 27 (1003 + 21) * 4 Note that an operator is missing after 27.

E011

The user could then correct this and any other error detected during compilation and generate an updated, correct source program tape with the Editor. When all errors have been corrected, the program is recompiled with BOOL-143/8 and a complete compiler listing should be requested.

The L response causes BOOL-143/8 to generate the complete compiler listing on the low-speed teleprinter. 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 Industrial 14 machine code instructions are listed below the equation.

The H response causes BOOL-143/8 to generate a listing on the high-speed punch. This allows off-line generation of several Teletype listings. If the listing is generated on the high-speed punch, errors are also typed on the Teletype.

The compiler listing contains the Industrial 14 generated machine code instructions following each equation. These instructions are typed in four columns: the first column is the octal absolute address in memory where the instruction is stored, the second column is the numeric form of the instruction and the third and fourth columns are the symbolic form of the instruction. For example:

```
•LOC 0

1001=1+(32*3+1012)

0000  4001  TF      0001
0001  2007  JFF     0007
0002  4032  TF      0032
0003  4003  TF      0003
0004  2007  JFF     0007
0005  5012  TF      1012
0006  2411  JFN     0011
0007  3001  SN      1001
0010  0010  SKP
0011  1001  SF      1001
```

```
1013=2*4+1013*(/4+/5)

0012 4002 TF      0002
0013 4004 TF      0004
0014 2023 JFF     0023
0015 5013 TF      1013
0016 2425 JFN     0025
0017 6004 TN      0004
0020 2023 JFF     0023
0021 6005 TN      0005
0022 2425 JFN     0025
0023 3013 SN      1013
0024 0017 SKP
0025 1013 SF      1013
```

At the end of the program listing, BOOL-143/8 types the following messages:

ERROR LINES: The decimal number of lines in the source program which contained errors.

PROGRAM BREAK: The highest absolute Industrial 14 memory address used by the compiled program.

**Source Program**

The source program can be supplied to BOOL-143/8 from one of two devices: the low-speed paper tape reader or the high-speed paper tape reader, if the PDP-8 System is so equipped. The source is specified in response to the query:

*SRC-LH?

where the L and H have the following meanings:

L The source program is in paper tape form and is to be read with the low-speed reader of the Teletype unit.

H The source program is in paper tape form and is to be read with the high-speed reader.

**VT14 Compatability**

The VT14 compatability option query enables BOOL-143/8 to detect BOOL commands whose machine language cannot be executed in the VT14 programming terminal. The query is:

VT14 - NY?

where the possible responses N and Y specify No and Yes respectively. With the VT14 compatability option, BOOL-143/8 will detect all R- and Z-function commands, monitor transition commands (.MN, .MF, and .MFN), and certain control statements (.FIXS and .VARS).

**Loading and Starting BOOL-143/8**

BOOL-143/8 is supplied as a self-starting binary tape and is loaded into Field 0 of the PDP-8 memory with the RIM loader. Either the paper tape reader on the 33 ASR Teletype or the high-speed reader can be used to load the tape.

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-143/8 responds by typing a sequence of queries, for example:

*SRC-LH? What is the source device from which the program will be supplied to BOOL-143/8? Will it be the low-speed reader of the Teletype (L), or the high-speed paper tape reader (H)?

*BIN-NLH? What device is to be used for the binary output tape? Is it no binary output requested (N); binary output on the low-speed paper tape punch (L); or binary output on the high-speed paper tape punch (H)?

*LST-NLH? What listing form is required? No listing is required (N). In this case, only error messages are typed. A low (L) speed listing is requested. A high (H) speed punched tape listing is requested.

*VT14-NY? Is VT14 compatability requested, (N) No or (Y) Yes?

When the VT14-NY? query has been answered, compilation begins. If the low-speed reader is used to read a source paper tape, the reader must be switched to START before compilation begins. If the high-speed reader is used, and it was not loaded prior to answering the final query, BOOL-143/8 must be stopped and restarted at 200, after the tape to be translated is loaded into the reader.

As BOOL-143/8 reads and compiles the source tape, it outputs a listing and a binary tape. If the listing and the binary output devices are the same, the binary output tape will be created first. As soon as the binary tape is punched, press the PDP-8 CONT switch to start the listing, with the first source tape in the reader.

BOOL-143/8 normally completes the compilation after reading the source paper tape once. However, if the program has been segmented with the .EOT statement, BOOL-143/8 halts after the incomplete tape and types EOT. The next tape should then be loaded and the PDP-8 CONT switch pressed.

When compilation begins BOOL-143/8 types 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 can simply press the PDP-8 CONT switch after the binary paper tape is punched. If no second pass is needed for a listing, BOOL-143/8 then begins the option queries by typing *SRC-LH?.

Figure 8-8 is a flow chart for BOOL-143/8 operation.

## PAL-143/8 OPERATIONS

### Loading and Starting PAL-143/8

PAL-143/8 is supplied as a self-starting binary tape and is loaded into the PDP-8 memory with the RIM loader program. Either the paper tape reader on the Teletype or the high-speed reader can be used to load the tape.

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, PAL-143/8 types the following queries to determine the devices to be used during assembly.

*BIN — Type the letter signifying the device on which the binary tape will be punched. The binary tape is the assembly output.

*LST — Type the letter signifying the device on which the assembly listing 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
*SRC-L
```

To generate an error listing only, request no binary, no listing, and specify only the source device.

If a mistake is made while typing a response to a query, the user may change the response *before* typing 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 can erase all previous query responses and restart the sequence of queries by typing the response X. For example:

```
*BIN-L
*LST-X
*BIN-          The query sequence has been restarted.
```

Two additional responses (N and E) are used to control the output or optional error listings. These character responses can be typed, along with the required response to any query. They can be erased by typing the X response and restarting the query sequence.

N — No separate error listing is desired on the Teletype. PAL-143/8 suppresses 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 the Pass 2 error listing to the Teletype is also 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.

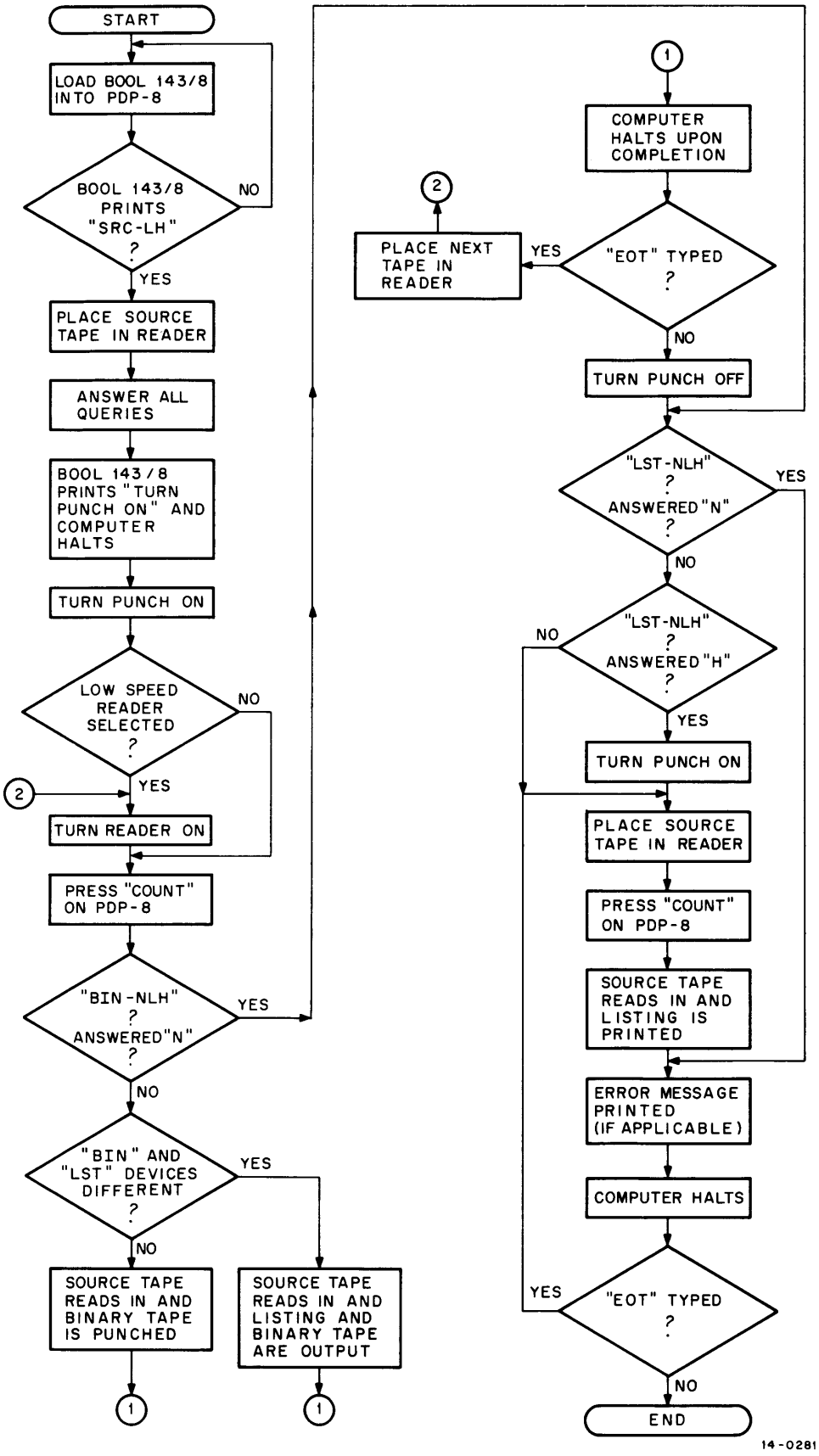



Figure 8-8 BOOL-143/8 Operational Flow Chart

For example, the following sequence of queries and responses results in binary output on the high-speed punch:

1. The assembly listing is punched on the high-speed punch
2. The source tape is read from the high-speed reader
3. (No Pass 1 or Pass 2 error listings are typed on the Teletype)
4. The Pass 1 error listing is punched on the high-speed punch.

*BIN-HN ↵
*LST-HE ↵
*SRC-H ↵

**Assembly Passes**

PAL-143/8 must read the program source tape possibly two or three times, depending on the selection of assembly devices. The first pass of PAL-143/8 defines symbols and checks for errors. The second pass types the listing and the symbol table. If separate devices are designated for the listing and the binary output, the binary output will also be punched on the second pass. If the same device (for instance, the Teletype) is used for the listing and the binary output, a third pass will be required to punch the binary tape and thereby complete the assembly process.

PAL-143/8 types the message:

*PAS

and halts when it has completed an assembly pass. If a further pass is required, 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 punches the binary tape output (either Pass 2 or Pass 3) the punch designed in the BIN query must be turned ON.

More than one program can be assembled without reloading PAL-143/8 by pressing CONT in response to the *PAS statement, typed after the binary output has been punched. PAL-143/8 responds by typing the initial queries (*BIN, *LST, and *SCR) and the assembly process may be restarted.

If a PAL-143/8 source program is contained on two or more source tapes, the assembler stops after reading all but the last source tape and types the following message:

*EOT (end of tape)

Whenever this message is typed, the user should load the next sequential tape and press the PDP-8 CONT switch.

**Error Halt**

Symbol table overflow is the only error which will halt the assembly. If this occurs, PAL-143/8 types the following message:

*SMB OFLW

Recovery is possible only if the user segments the program and reassembles it, in parts.

## ODP-143 OPERATIONS

ODP-143/8 is supplied as a self-starting binary tape and is loaded into Field 0 of an 8K PDP-8 memory with the toggled-in RIM loader. Either the paper tape reader on the model 33 ASR Teletype or a high-speed reader can be used to load the tape. Before loading and starting ODP-143/8, installation of the Industrial 14 and the PDP-8 to Industrial 14 serial interface must be completed. If this is not done, ODP-143/8 will print "INDUSTRIAL 14 HUNG".

To restart ODP-143/8, set the Switch Register to 0200, and press the START switch(es) on the PDP-8 console. Upon starting or restarting, ODP-143/8 will enter Program mode and place the Industrial 14 into External mode.

## USING OS/8 SYSTEM TO DEVELOP INDUSTRIAL 14 PROGRAMS

BOOL-143/8, SET-143/8, PAL-143/8 and ODP-143/8 can be used with the powerful OS/8 system to develop Industrial 14 programs more efficiently. This section describes briefly, the loading, accessing, and operation of BOOL-143/OS8, SET-143/OS8, PAL-143/OS8, and ODP-143/OS8 programs. It is assumed that the user has a thorough knowledge of the OS/8 system. If not, before continuing, it is recommended that the user read the *"OS/8 User's Guide"*, (DEC S8-OSRMA-A-D).

**Saving 143/OS8 Programs**

The BOOL-143/OS8, SET-143/OS8, PAL-143/OS8 and ODP-143/OS8 Programs are supplied on paper tape. These 143/OS8 programs should be loaded into core and then stored in Save Format on the system DECtape or disk unit

with the Keyboard Monitor before running them. When Saving each program on the system device, core limits (locations in memory), starting address, and the job status word must be specified. Table 8-3 lists core limits (locations in memory), the starting address, and job status word for each 143/OS8 program.

For example, to Save a paper tape of BOOL-143/OS8 on the system device type:

```
.SAVE SYS OS8BOL 2000–2777, 10000–16577;
102005 = 4002
```

To access and run the BOOL-143/OS8 Program, type:

```
.R OS8BOL
```

The OS/8 System enters the Command Decoder and prints an asterisk (*) when it is ready to accept a command string.

**Command Decoder Input Strings**

OS/8 Command Decoder expects the following input string for BOOL-143/OS8 and PAL-143/OS8:

file . BIN, file . LS < file . PA

Where file .BIN is the binary (assembled or compiled) output files.

file .LS is the output (assembled or compiled) listing.

file .PA is the input source file.

OS/8 Command Decoder expects the input string for SET-143/OS8 and ODP-143/OS8 to be:

file .BN < file . BN

Where the input and output files are both binary format.

To read the input file, type R ↵. To punch the output file, type P ↵.

**143/OS8 Program Differences**

The OS/8 Command Decoder eliminates all queries of BOOL-143/8, SET-143/8 and PAL-143/8 except:

VT14-NY? – for BOOL-143/8
and
SYM-YN? – for SET-143/8

These queries are printed after the Command Decoder input string has been issued.

Upon SET-143/OS8's translation, BOOL-143/OS8's compilation, or PAL-143/OS8's assembly of the Industrial 14 source program, each program will exit automatically to the Keyboard Monitor (printing a ↑C). To exit from the ODP-143/OS8 program to the Keyboard Monitor, or to halt any printing (for a List or Interrogate command) the user must press the CONTROL and C keys, simultaneously. To restart the ODP-143 program, type ST ↵.

**Table 8-3**
**143/OS8 Program Specifications**

| 143/OS8 Programs | 6 Character Filename | Core Limits | Starting Address | Status Word |
|---|---|---|---|---|
| BOOL-143/OS8<br>DEC14-IBOLB-A-PB | OS8BOL | 2000 – 2777<br>10000 – 16577 | 10200 | 4002 |
| SET-143/OS8<br>DEC14-ISETB-A-PB | OS8SET | 0–3777 | 00200 | 2000 |
| PAL-143/OS8<br>DEC14-IPALB-A-PB | OS8PAL | 0–5577<br>17000 – 17577 | 00200 | 2000 |
| ODP-143/OS8<br>DEC14-IODP-A-PB | OS8ODP | 0–5177 | 00200 | 2001 |

### 143/OS8 I/O Errors

The I/O error messages, shown in Table 8-4, may appear when reading, writing, opening or closing OS/8 files.

If one of these error messages occur:

a. Check the input and output device in use for proper mode of operation.

b. Validate that all necessary input or output devices were issued to the Command Decoder.

Once this error message is printed, control returns to the OS/8 Keyboard Monitor. To reenter the Command Decoder, type "ST" (Start).

**Table 8-4**
**143/OS8 I/O Error Messages**

| I/O Error Messages | Meaning |
|---|---|
| Open Error | 143/OS8 System cannot open an output file. |
| Read Error | 143/OS8 System cannot read an input character. |
| Write Error | 143/OS8 System cannot write an output character. |
| Close Error | 143/OS8 System cannot close an output file. |

# CHAPTER 9
# MONITORING

Although the Industrial 14 is primarily a stand-alone control system, its use in conjunction with a computer offers an added dimension to a control system. This chapter details the manner in which the Industrial 14 can communicate with a computer.

## INTRODUCTION

This chapter introduces the instructions which are executed within the Industrial 14 to cause it to react to computer input, and the instructions that cause it to send information to the computer. Later sections of the chapter describe the available communication interfaces. The instructions which are executed within the monitoring computer to pass the previously mentioned Industrial 14 instructions to the controller, and receive data from the controller, are also described.

## MONITORING FACILITIES

The Industrial 14 has two primary facilities for computer connection: a memory port and an output register (Figure 9-1).

The *memory port* can be equated to an opening in memory through which externally supplied instructions are passed. In actuality, the memory port is a three-word by 12-bit register which passes the externally supplied instruction to the control logic for execution. The memory port is permitted to interrupt the internal program and supply instructions on a cycle-stealing basis.

Associated with the memory port is the *external flag* which signals the status of the memory port. When a new instruction is supplied to the memory port, the flag is cleared. The memory port then waits for the controller to complete its current instruction; at this time, the externally-supplied instruction is executed by the controller. The external flag is set after the full one-, two- or three-word instruction is executed; at this time, the controller executes the next instruction of its internal program.

The *output register* holds data returning from the Industrial 14 controller to the computer. This data word is 12-bits in length and can be generated by either internal- or externally-supplied instructions. Associated with the output register is an output flag which sets whenever a new data word is loaded into the output register.

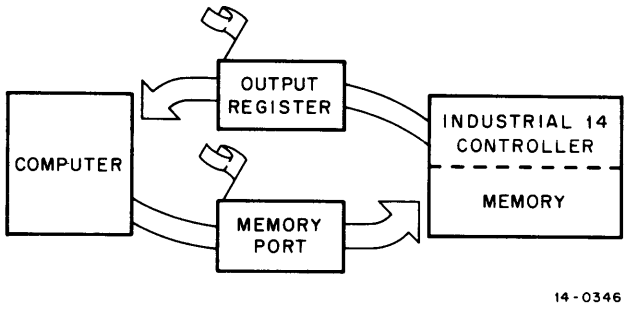



Figure 9-1 Industrial 14 Monitoring Facilities

## MONITORING APPROACHES

Four general approaches to monitoring are available with the Industrial 14; they can be used individually or collectively to fulfill particular application requirements.

*Status reporting* is achieved by placing instructions in the controller's memory which report changes in state of outputs to the computer. The controller is programmed (usually with BOOL-143) to recognize that a meaningful change has occurred; an appropriate data word is then loaded into the output register and transmitted to the monitoring computer.

*Polling* allows the computer to read the status of controller inputs and outputs at any time. The computer supplies instructions through the memory port which interrupt the Industrial 14 and cause it to test the required points, loading their status into the output register. A single point,

or a consecutive group of eight points, may be polled in a single transfer. Once the controller has loaded the output register, it returns to its internal program until it is again polled.

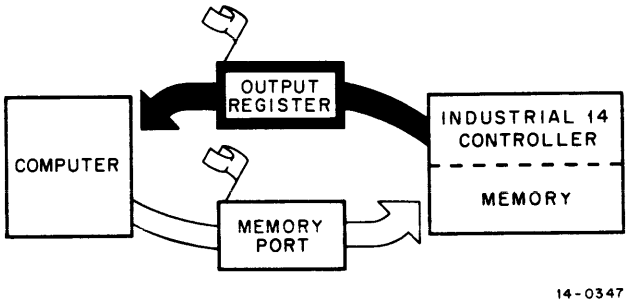



Figure 9-2 Status Reporting

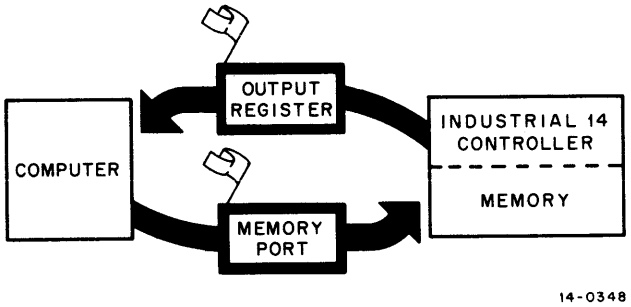



Figure 9-3 Polling

*Computer command* of an Industrial 14 Controller is achieved by interrupting the controller and supplying an instruction through the memory port which sets an I/O point ON or OFF. The I/O point, set by the computer, can be either an external or internal I/O number. The particular point chosen can be an unused input number, which the computer must disable before forcing it ON or OFF; it also could be an unused external output number, which could be reset by the controller to signal completion of the task. In each case, the Industrial 14 loses the computer command if power is lost. The use of a retentive memory allows retention of the computer command.

The controller reacts to this command simply by testing the I/O number in its internal program and reacting according to the computer-supplied input. The computer command does not cause a return transmission through the output register. The external flag is sampled to determine if the controller is ready to accept a new command.

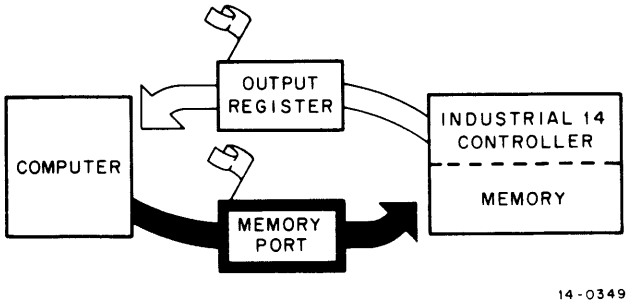



Figure 9-4 Computer Command

*Down line loading* is used to supply a partial or a complete program to the controller. It is achieved by interrupting the controller and locking it into *external mode,* in which only externally-supplied instructions are accepted. After turning the controller's outputs OFF, the computer loads the controller's memory as necessary, through the memory port. Once loaded, memory can be verified for proper content by supplying instructions to read back memory content through the output register. The computer then releases the controller from external mode and the Industrial 14 begins execution of the new control program.

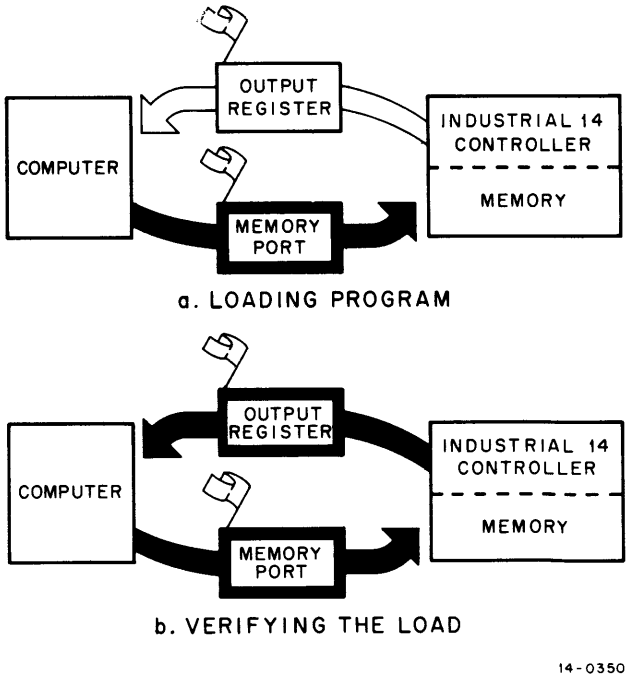



Figure 9-5 Down Line Loading

## DUAL INTERFACE PORTS

The Industrial 14 is capable of communication with two separate external devices through dual interface ports. The two ports supply instructions through a common memory port; priority logic prevents any confusion of externally-supplied instructions. Each port has an output register to hold data for the external device.

The *utility port* is a DC14-F 9600 baud, serial interface, built into the Industrial 14 and is primarily used for programming or maintenance operations. This is the low-priority interface; output register 1 is used for communicating with a VT14, PDP-8, PDP-11 or other device.

The *monitoring port* is available for dedicated computer connection and is utilized by separately-purchased options, described later in this chapter. It has the highest priority of access to the memory port and uses output register 6.

These priorities are for the infrequent occurrence when both ports want to supply an instruction to the Industrial 14 at the same time. The monitoring port is granted access to the memory port for its instruction, then the utility port is serviced. This added wait is only a few microseconds. Although the memory port is shared, each port has an external flag (Figure 9-6).

The output register designations (1 and 6) do not imply availability of any other output registers; the Industrial 14 is only capable of supporting two external interfaces at any one time.

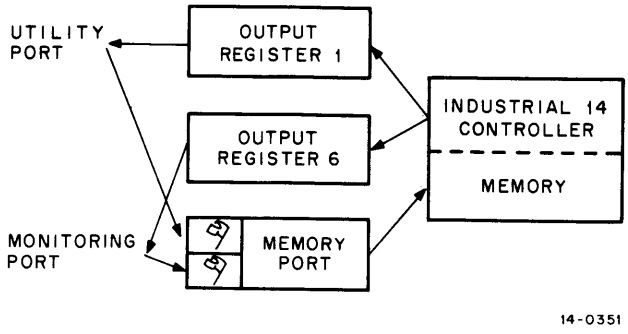



Figure 9-6 Industrial 14 Dual Interface Ports

## INTERNAL INSTRUCTIONS

In addition to the instructions introduced in Chapter 2 for normal control functions, the Industrial 14 has an additional repertoire of instructions, many of which are useful for monitoring and computer-connected operations. The following paragraphs introduce these instructions, which unlike the normal testing and setting instructions, often require two or three 12-bit memory words to express the complete operation.

In general, these instructions can be stored in the Industrial 14's memory for execution, or they can be supplied by an external computer, over any of the interface options described later in this chapter.

## MEMORY INSTRUCTIONS

A number of internal Industrial 14 instructions are available for reading, writing, and controlling memory operations; they are summarized in Table 9-1.

The first two instructions deal with the program counter (PC); this is a 13-bit register, which sequences through internal memory by pointing to the next location to be executed. The CLRPC instruction is used at the end of a program to cycle back to the beginning. The RDPC instructions allows the external computer to obtain the current PC value through the appropriate output register.

The next two instructions are used throughout normal Industrial 14 programs. The NOP is stored in every unprogrammed location. The SKP is used to bypass SF instructions when an SN has been executed.

The CIF instructions are used to pass from one memory field to the other; these take effect only on jump type instructions. Normally, these instructions are not needed because the program counter automatically starts at the beginning of the second memory field when it reaches the end of the first. (Similarly, the CLRPC instruction links back to field 0 location 0, regardless of where in memory it is executed.) Normal programming procedures are concerned with instruction fields.

**Table 9-1**
**Memory Control Instructions**

| Symbolic | Octal | Meaning |
|---|---|---|
| CLRPC | 0004 | Clear PC – forces the next instruction to be taken from location 0 in field 0. CLRPC also sets ON internal function 1777 (INITIALIZE). |
| RDPC | 0046<br>0041 | Read PC – reads the 12-bit value of the program counter into the output register, specified by the least-significant octal digit. |
| NOP | 0000 | No operation – used to occupy unprogrammed memory locations. |
| SKP | 0010 | Skip – increments the program counter, thereby causing the next memory location to be bypassed. |
| CIF0 | 0020 | Change to instruction field 0 – causes the Industrial 14 to begin executing instructions from field 0 after the next JMP, JMS, JMR, JFF or JFN instruction. |
| CIF1 | 0030 | Change to instruction field 1 – causes the Industrial 14 to begin executing instructions from field 1 after the next JMP, JMS, JMR, JFF, or JFN instruction. |
| EEM | 0060 | Enter External Mode – causes the Industrial 14 to suspend operations from its internal memory and to execute only externally-supplied instructions (e.g. to allow memory to be written.) |
| LEM | 0040 | Leave External Mode – causes the Industrial 14 to begin operation from internal memory at the location specified by the current value of the program counter. |
| LDMEM<br>Word | 0022<br>Word | Load Memory – loads the 12-bit word that follows into memory at the location specified by the program counter. (This instruction can only be executed when supplied externally and when the Industrial 14 is in external mode.) |
| RDMEM or TRM<br>AAAA | 0026 or<br>0021<br>AAAA | Read Memory – causes the content of memory location AAAA of the memory field, set with CDF instructions, to be read into the output register, specified by the last octal digit. |

**Table 9-1 (Cont)**
**Memory Control Instructions**

| Symbolic | Octal | Meaning |
|---|---|---|
| CDF0 | 0600 | Change to data field 0 – causes all memory reading by an external computer to be from the first 4K of memory. |
| CDF1 | 0700 | Change to data field 1 – causes all memory reading by an external computer to be done from the second 4K of memory. |

**Table 9-2**
**I/O Control Instructions**

| Symbolic | Octal | Meaning |
|---|---|---|
| CLR | 0170 | Clear all outputs – clears the I/O state memory for all external outputs (1000–1377). |
| DOM | 0160 | Disable Output Multiplexer – prevents the logical states of outputs, as set in I/O storage memory, from being supplied to the actual output converters. Input states continue to be updated. |
| EOM | 0150 | Enable Output Multiplexer – allows output states to be read to the output converters. |
| EOL | 130 | Enable Output Loop – for diagnostic purposes, this instruction logically connects each output to two inputs in an inverse manner. After an EOL, when output 1000 is on, inputs 0 and 400 will be off; 1001 turns off 1 and 401; etc. External input states have no effect on this operation. |
| DOL | 140 | Disable Output Loop – allows the input to reflect actual input conditions, rather than the inverse of the associated output. |

The CIF instructions are required to specify the memory field for memory writing operations. An EEM instruction locks the Industrial 14 into External Mode to allow writing of memory; a CIF instruction, followed by a JMP instruction, loads the program counter with the memory field and location to be written; a LDMEM transfers the new content of memory; and finally, an LEM releases the Industrial 14 to begin execution of the new program.

The last three instructions in Table 9-1 are used for reading memory. The CDF instructions specify the memory field to be read; the RDMEM instruction supplies the specific address to be read into either output register 1 or 6, as indicated in the instruction. Reading of Industrial 14 memory can be performed in Interrupt Mode without any effect on the internal operation of the controller.

Detailed procedures for reading and writing memory are given later in this chapter.

### I/O CONTROL INSTRUCTIONS

In addition to the normal testing and setting I/O instructions covered in Chapter 2, additional I/O related instructions listed in Table 9-2, are available that control the operation of the I/O multiplexer logic.

The CLR instruction is used to logically (electrically) turn off all external outputs. It is equivalent to issuing 256 SF instructions, one for each output number from 1000–1377.

The DOM instruction has the same external effect as CLR; however, it does not clear the logical state of the output, it merely halts the multiplexing of the output drivers such that they turn off electrically. The DOM/EOM instruction pair can therefore be used to switch all outputs off for some period of time and later return them to the original state.

The EOL instruction is only used diagnostically to check the I/O circuitry within the control unit. It inversely connects each output to two inputs: turning output 1000 ON turns OFF inputs 0 and 400; 1001 turns 1 and 401 OFF, etc.

### I/O MANIPULATION INSTRUCTIONS

Inputs, outputs and internal functions can be accessed by the TN, TF and SN, SF instructions, described in Chapter 2; however, additional access and manipulation of I/O states is possible in the Industrial 14, using the instructions listed in Table 9-3. These instructions make use of the Industrial 14's unique structure for storing I/O states.

External I/O states are stored in the manner represented in Figure 9-7. They can be read, loaded, moved, cleared and set individually with the bit-mode instructions in Table 9-3 by referencing the particular point. Similarly, they can be manipulated in words of eight points, using the WD-mode instructions by referencing any point within the group.

As noted in Chapter 7, I/O points can be disabled. This capacity is achieved through a second I/O storage memory which holds a disable state for each of the external I/O points. Writing a 1 for a point in the disable memory disables the point. These disable bits are accessed by adding 2000 to the respective I/O point. Figure 9-8 illustrates this disable memory. It can be accessed by the load, clear, read, set and move instructions in both bit and word mode.

Internal functions (I/O numbers beginning with 1400) are not stored in I/O storage memory; they are stored in the last 256 words of main memory. However, these functions are accessed directly by the I/O numbers, not by reading memory.

The functions before the I/O partition are accessed and manipulated identically to external I/O points (both word and bit mode instructions are available). However, these points do not have an associated I/O disable memory and cannot be disabled or forced. Although these points are stored in the Industrial 14 12-bit main program memory, the first four bits are zero and only the last eight bits are meaningful (Figure 9-9).

Beyond the I/O partition, the information stored for internal functions is considerably different (Figure 9-10). All 12 bits are used and two words are associated with each function. The first word (even address) holds the current value as a straight 10-bit binary value. The second word (odd address) holds the preset value also as a straight 10-bit binary value. The leading two bits of the even address are used to record the status of the function; the leading two bits of the odd address record the type of function.

The state of internal functions beyond the I/O partition, as stored in bits 0 and 1 of the even address, are established in response to SN and SF instructions. Timers increment during "clock ticks"; bit 0 of the even address is set ON during the clock tick, which causes the current value to equal the preset. Counters increment on the first SN instruction following an SF; bit 0 of the even address for counters will be set ON only during the increment which causes equivalence between the current value and the preset. Specifically, the preset-reached bits, and the states tested by the Industrial 14's control program, are not set by loading a number into the current value which equals or exceeds the preset.

**Table 9-3**
**I/O Manipulation Instructions**

| Symbolic | Octal | Meaning |
|---|---|---|
| CLRWD<br>AAAA | 0003<br>AAAA | Clear I/O Word – when referencing external I/O, I/O disables, or internal functions up to the I/O partition, all eight points with the same first three octal digits AAAX are set OFF. When referencing an I/O function beyond the partition, the complete 12-bit data word is cleared. |
| SETWD<br>AAAA | 0013<br>AAAA | Set I/O Word – when referencing external I/O, I/O disables, or internal functions up to the I/O partition, all eight points with the same first three octal digits AAAX are set ON. When referencing an I/O function beyond the I/O partition, the complete 12-bit word is set to 7777. |
| LDWD<br>DDDD<br>AAAA | 0023<br>DDDD<br>AAAA | Load I/O Word – when referencing external I/O, I/O disables, or internal functions up to the I/O partition, the eight points with the same first three octal digits AAAX are set according to the last eight bits of the data word XDDD. When referencing an I/O function beyond the partition, the complete 12-bit data word is loaded into the timer or counter as shown in Figure 9-10. |
| MOVWD<br>AAAA<br>BBBB | 0033<br>AAAA<br>BBBB | Move I/O Word – when referencing external I/O, I/O disables, or internal functions up to the I/O partition, the eight points with the same first three octal digits AAAX are parallel transferred to the points BBBX. When internal functions beyond the I/O partition are referenced, the complete 12-bit word is transferred. Move operations leave the source word unchanged. |
| RDWD<br>AAAA | 0036 or<br>0031<br>AAAA | Read I/O Word – transfers the I/O word referenced to the specified output register, in the format of Figure 9-9, for all points except internal functions beyond the I/O partition, which are read in the format given in Figure 9-10. |
| CLRBIT<br>AAAA | 0103<br>AAAA | Clear I/O Bit – sets off the single point specified by AAAA. This instruction is not defined for values of AAAA beyond the I/O partition (timers and counters). |

**Table 9-3 (Cont)**
**I/O Manipulation Instructions**

| Symbolic | Octal | Meaning |
|---|---|---|
| LDBIT<br>M000<br>AAAA | 0123<br>M000<br>AAAA | Load I/O Bit – loads I/O point AAAA with the most-significant bit of mask M. This instruction is not defined for values of AAAA beyond the I/O partition (timers and counters). |
| MOVBIT<br>AAAA<br>BBBB | 0133<br>AAAA<br>BBBB | Move I/O Bit – transfers the state of I/O point AAAA to I/O point BBBB. The source point is unaffected by the move. This instruction is not defined for values of AAAA and BBBB beyond the I/O partition (timers and counters). |
| RDBIT<br>TD or<br>AAAA | 0136 or<br>0131<br>AAAA | Read I/O Bit – loads the specified output register with the state of the I/O point AAAA in the format given in Figure 9-11. RDBIT is defined for all I/O points including internal functions beyond the partition. |
| SETBIT<br>AAAA | 0113<br>AAAA | Set I/O Bit – sets on the single point specified by AAAA. This instruction is not defined for values of AAAA beyond the I/O partition (timers and counters). |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Inputs |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | |
| 770 | 771 | 772 | 773 | 774 | 775 | 776 | 777 | |
| 1000 | 1001 | 1002 | 1003 | 1004 | 1005 | 1006 | 1007 | Outputs |
| 1360 | 1361 | 1362 | 1363 | 1364 | 1365 | 1366 | 1367 | |
| 1370 | 1371 | 1372 | 1373 | 1374 | 1375 | 1376 | 1377 | |

Figure 9-7 I/O Storage Memory Organization

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | |
| 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | |
| 2020 | 2021 | 2022 | 2023 | 2024 | 2025 | 2026 | 2027 | Input Disables |
| 2770 | 2771 | 2772 | 2773 | 2774 | 2775 | 2776 | 2777 | |
| 3000 | 3001 | 3002 | 3003 | 3004 | 3005 | 3006 | 3007 | |
| 3360 | 3361 | 3362 | 3363 | 3364 | 3365 | 3366 | 3367 | Output Disables |
| 3370 | 3371 | 3372 | 3373 | 3374 | 3375 | 3376 | 3377 | |

Figure 9-8 Disable Storage Memory Organization

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| X | X | X | X | 1400 | 1401 | 1402 | 1403 | 1404 | 1405 | 1406 | 1407 |
| X | X | X | X | 1410 | 1411 | 1412 | 1413 | 1414 | 1415 | 1416 | 1417 |
| X | X | X | X | 1420 | 1421 | 1422 | 1423 | 1424 | 1425 | 1426 | 1427 |
| X | X | X | X | 1430 | 1431 | 1432 | 1433 | 1434 | 1435 | 1436 | 1437 |
| X | X | X | X | 1440 | 1441 | 1442 | 1443 | 1444 | 1445 | 1446 | 1447 |
| X | X | X | X | 1560 | 1561 | 1562 | 1563 | 1564 | 1565 | 1566 | 1567 |
| X | X | X | X | 1570 | 1571 | 1572 | 1573 | 1574 | 1575 | 1576 | 1577 |

NOTE: I/O Partition is assumed to be 1600.

Figure 9-9 Organization of Bit-Oriented Internal Functions

Up/down counters use three I/O addresses but only two real data words. The "state" of the up/down counter is not directly stored in these words. Bits 0 and 1 store the enable bits for the bi-directional counter. The state of the preset equalling the current value is not stored in memory; it is determined only when tested by TD (RDBIT), TN and TF instructions.

In working with timers and counters, always keep in mind that timers must have I/O numbers below all counters. This means that the Industrial 14 will treat all functions beyond the I/O partition as counters after the first I/O function which has bit 0 of the odd address equal to 0.

**I/O POLLING**

Polling of I/O states by an external computer can be done either singly or in a group of eight. The RDBIT and RDWD instructions cause the specified output register to be loaded with the words given in Figure 9-11.

The RDBIT instruction is used for all I/O points. When RDBIT addresses a timer or a counter, it receives as the current state the same state as would be read by a TN instruction.

The RDWD instruction reads eight bits, in-parallel, for all external I/O (including I/O disables) and for internals up to

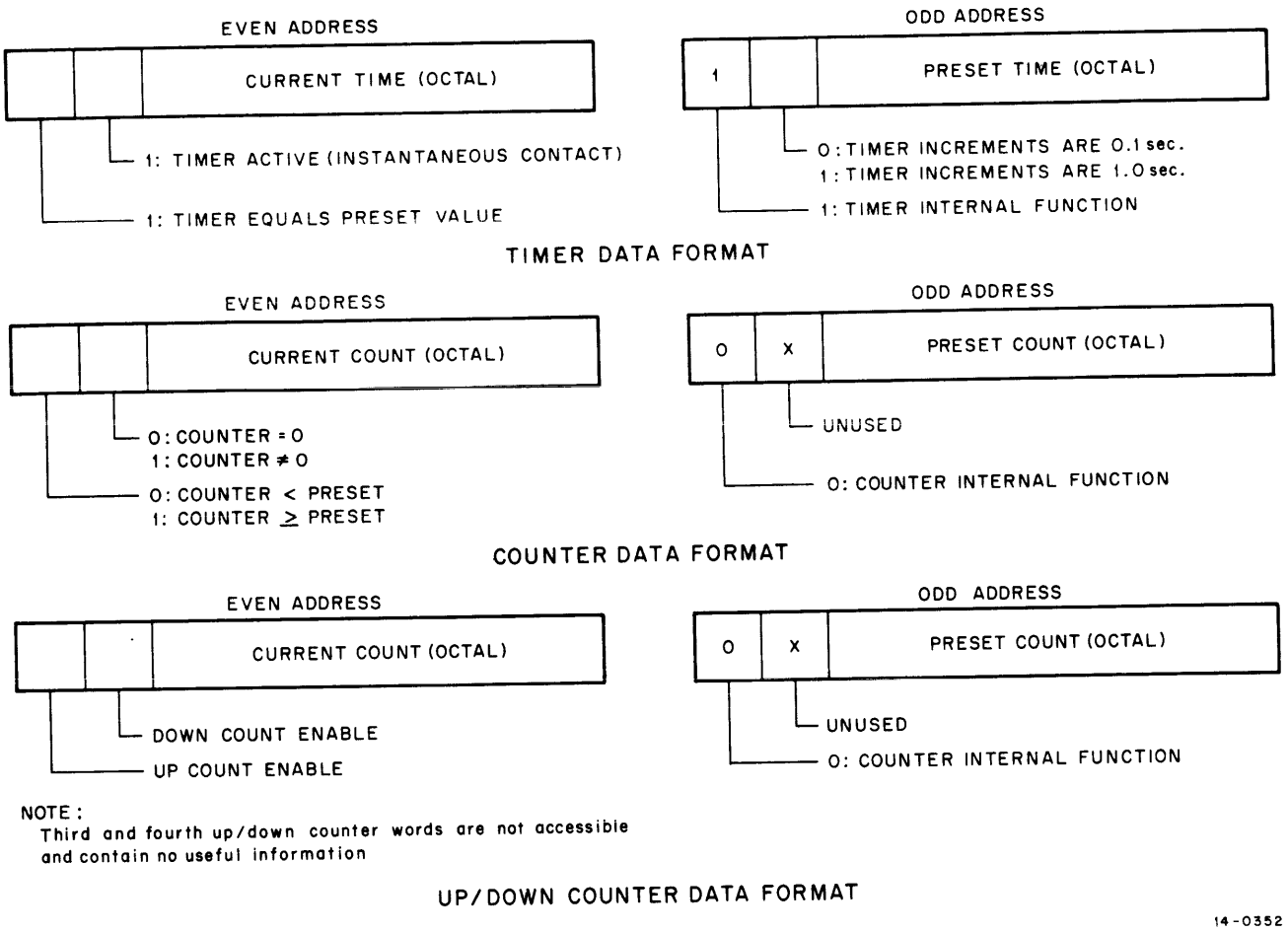



Figure 9-10 Internal Function Storage Format

the I/O partition. Beyond the I/O partition, RDWD accesses the full timer or counter data words as given in Figure 9-10. Notice that the timer or counter value is in octal, not binary-coded decimal, or any other form.

## PROCESSOR STATUS WORD

I/O address 7777 (*not* memory address 7777) is used to read the Industrial 14 processor status word. Figure 9-12 illustrates the bit format with an explanation of the internal function partition bits. The processor status word is only read with the RDWD instruction. It can never be altered by a move or load instruction.

## ERROR CONDITIONS

One requirement of any monitoring system is the ability to determine current status within the Industrial 14; this is particularly true if the controller is inoperative.

A loss of either ac power to the controller or dc power within the controller will stop the Industrial 14 and cause all external outputs and I/O disables (if any) to be cleared. Internal functions retain their state unless programmed to clear when power returns. In the event of power loss, no communication with the Industrial 14 is possible until power returns.

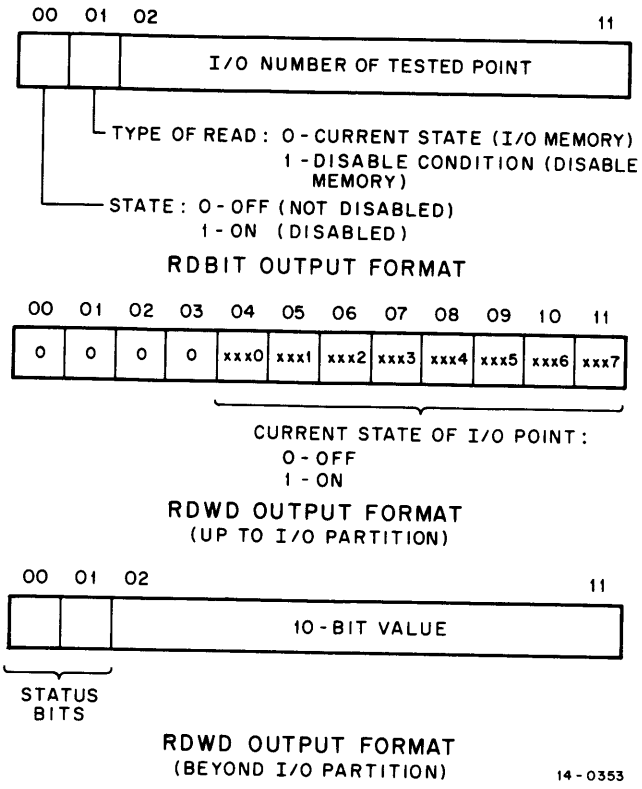



Figure 9-11 Read I/O Output Register Format

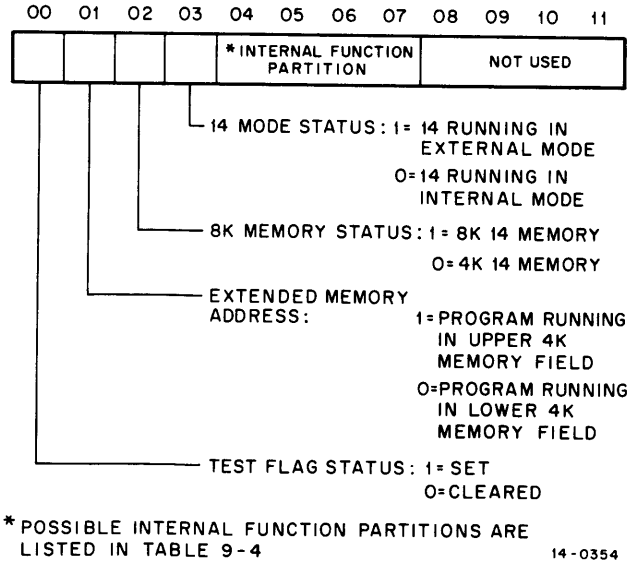



Figure 9-12 Processor Status Word Format

Detection of an error condition within the Industrial 14 will not prevent communication to the controller. Error traps cause the Industrial 14 to enter external mode, which can be read via the processor status word. Simultaneously, the output multiplexer is disabled, causing the output converters to be turned off, although their logic state at the time of the error trap is preserved in the I/O memory. Likewise, all internal functions and all I/O disables remain fixed from the time the error was detected. Therefore, the monitoring computer can access any of this status, if desired. Error traps will occur if the Industrial 14 has a loss of internal timing or has a short circuit on the memory data bus. A third error trap is caused by the Industrial 14 program counter referencing the first location of internal function storage (location 7400 in the last memory field installed in the controller). The computer can read the program counter to determine if this trap has occurred.

## INSTRUCTION EXAMPLES

A series of instruction examples follow to illustrate the use of some of the extended instructions of the Industrial 14.

**Table 9-4**
**Possible Internal Function Partitions**

| Bit 4 | Bit 5 | Bit 6 | Bit 7 | I/O Partition |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1400 |
| 0 | 0 | 0 | 1 | 1420 |
| 0 | 0 | 1 | 0 | 1440 |
| 0 | 0 | 1 | 1 | 1460 |
| 0 | 1 | 0 | 0 | 1500 |
| 0 | 1 | 0 | 1 | 1520 |
| 0 | 1 | 1 | 0 | 1540 |
| 0 | 1 | 1 | 1 | 1560 |
| 1 | 0 | 0 | 0 | 1600 |
| 1 | 0 | 0 | 1 | 1620 |
| 1 | 0 | 1 | 0 | 1640 |
| 1 | 0 | 1 | 1 | 1660 |
| 1 | 1 | 0 | 0 | 1700 |
| 1 | 1 | 0 | 1 | 1720 |
| 1 | 1 | 1 | 0 | 1740 |
| 1 | 1 | 1 | 1 | 1760 |

**Example 1:** **Clear consecutive retentive memory addresses 1400–1421 upon initial start-up.**

| Instructions | Explanation |
|---|---|
| TN 1777 | Perform the clear on start-up only. |
| JFN .+11 | |
| CLRWD 1400 | Clears retentive memories or shift registers 1400 – 1407. |
| CLRWD 1410 | Clears retentive memories or shift registers 1410 – 1417 |
| CLRBIT 1420 | Clears individual retentive memories 1420 and 1421 |
| CLRBIT 1421 | |
| JFN .+11 | |

**Example 2:** **Reset timer 1620 by clearing its instantaneous and delayed contacts and current value.**

| Instructions | Explanation |
|---|---|
| CLRWD 1620 | Totally clears the current timer word, leaving its preset and timer type unchanged. |

**Example 3:** **Disable and force on the consecutive outputs 1000 – 1012.**

| Instructions | Explanation |
|---|---|
| SETWD 3000 | Disable outputs 1000–1007 |
| SETWD 1000 | Force ON 1000–1007 |
| SETBIT 3010 | Disable 1010 |
| SETBIT 1010 | Force ON 1010 |
| SETBIT 3011 | Disable 1011 |
| SETBIT 1011 | Force ON 1011 |
| SETBIT 3012 | Disable 1012 |
| SETBIT 1012 | Force ON 1012 |

**Example 4:** **Clear counter 1700 and preset it to 60.**

| Instructions | Explanation |
|---|---|
| CLRWD 1700<br>LDWD 0074 1701 | Clear counter overflows, counter enable and current count bits; then preset counter to 60 (74 octal). |

**Example 5:** **Shift data in registers 1420, 1421, 1422, 1423 and 1424 when 1402 transitions to ON.**

| Instructions | Explanation |
|---|---|
| . . . | Test conditions for 1402 shift circuit |
| JFN .+3 | |
| SF 1402 | Because 1402 is being set off, bypass the shift |
| JFF NEXT | |
| TN 1402 | Check if 1402 is already ON; if not, set ON. |
| SN 1402 | |
| JFN NEXT | Is this an OFF-to-ON transition? If not, bypass shift. |
| MOVBIT 1423 1424 | Shift instructions if this is a transition to ON. |
| MOVBIT 1422 1423 | |
| MOVBIT 1421 1422 | |
| MOVBIT 1420 1421 | |
| . . . | |

## INTERFACE OPTIONS

Interfaces are required in both the Industrial 14 and within the computer to allow them to pass data in the form of the specific instructions just introduced. The available interfaces vary as to their capability and performance; Table 9-5 summarizes these capabilities.

More information describing these hardware options, including installation procedures, is contained in the *Industrial 14 Systems Manual.*

**Table 9-5**
**Industrial 14 Interface Characteristics**

| Industrial 14 Interface: | DA14-E (EL) | DC14-F | |
|---|---|---|---|
| Connecting to: | PDP-8 | PDP-8 | PDP-11 |
| Using Computer Interface: | (8/L External bus) KA8/E | KL8-JA | DL11-C |
| Transfer Type: | Parallel | Serial – 9.6K baud | |
| Maximum Distance: | 50′ | 50′ | |
| Number of 14s: | 1 | 1/KL8 | 1/DL11 |
| Error Detection: | None | Parity | |
| Interrupt Capability: | Yes | Individual | |
| Capable of Controller Initiated Transmissions: | Yes | Limited | |
| Time to Poll One I/O Point or Word (Transmission only): | ≈.02 ms | ≈6.6 ms | |
| Time to Set One I/O Point or Word (Transmission only): | ≈.02 ms | ≈4.1 ms | |

## GENERAL CHARACTERISTICS

All Industrial 14 interfaces use the memory port and an output register. For computer monitoring, the monitoring port and output register 6 are normally utilized by whichever interface option is selected.

Another characteristic is the manner in which instructions can be supplied to the Industrial 14. The most common technique for supplying instructions is in *Interrupt Mode.* This operation allows an external computer to interrupt the internal program of the controller, and to cause a single instruction of any length to be executed by the controller; the controller then returns, unaffected, to its internal program.

When it is necessary to prevent the controller from returning to its internal program, Interrupt Mode is used to load the EEM (Enter External Mode) instruction. Thereafter, all instructions are supplied externally until the LEM (Leave External Mode) instruction, is given to the Industrial 14 controller.

While in External Mode, instructions can be supplied in two ways. The first, and more common approach, is to continue to transfer instructions as if using Interrupt Mode. The second is to use the External Load Mode, which allows the program counter to increment as it would if the externally-supplied instructions were actually stored in internal memory. However, it is usually easier to prevent the program counter from operating, even when in External Mode, by always supplying instructions as in Interrupt Mode. This manual uses the Interrupt Mode for supplying all examples, forcing the controller into External Mode where necessary.

Another characteristic found in all interfaces is the ability to sample the external flag to determine that the controller has executed the instruction supplied to it by the computer. All interfaces also have the ability to sample the output flag to determine that data has been loaded which now should be read. The manner in which these flags are actually sampled by a specific interface varies; the techniques used include skip instructions, program interrupts and bit test operations.

The various logical combinations of flag states have different meanings (Table 9-6).

Table 9-6
Industrial 14 Interface Flag Meanings

| External Flag | Output Flag | Meaning |
|---|---|---|
| 1 | 1 | An externally-supplied instruction has loaded the output register with data to be read by the monitoring computer. In this case, the flags will be set simultaneously. |
| 1 | 0 | An externally-supplied instruction has been completed. This instruction did not load data into the output register. |
| 0 | 1 | An internal instruction within the Industrial 14 has loaded data to be read by the monitoring computer. |
| 0 | 0 | This combination is used in some interfaces to signal a transmission error. |

## DA14-E PARALLEL INTERFACE

The DA14-E (and DA14-EL) interface is available for connecting a single Industrial 14 to a PDP-8 family computer. Selection of the -E or -EL model is dependent upon the PDP-8 configuration. Refer to the *Industrial 14 Systems Manual* for details. The DA14-E passes data at extremely high speeds, but it is restricted to a maximum cable length of 50 feet. A further restriction is that only one Industrial 14 can be connected to a PDP-8 family computer via a DA14-E at one time.

The DA14-E instructions used by the PDP-8 to communicate with the Industrial 14 are listed in Table 9-7; the first two instructions are used to pass instructions to the Industrial 14. The GNI is used to pass instructions in Interrupt Mode, or in External Mode, without affecting the program counter within the Industrial 14. Each word of the instruction is passed to the Industrial 14 with a GNI instruction. When all words have been supplied to the interface, the internal program is interrupted at the conclusion of its present instruction and the externally-supplied instruction is loaded through the memory port for execution. When the complete instruction has been executed, the external flag is set and the controller returns to its internal memory for the next instruction.

The LDE instruction can only be used after the Industrial 14 has been forced into External Mode using a GNI instruction. Once this is accomplished, the LDE or the GNI can be used to supply all subsequent instructions. Using the LDE results in the program counter incrementing as it would if the instructions were supplied from internal memory. Using the GNI leaves the program counter unchanged unless it is directly affected by the instruction supplied by the GNI (a JMP instruction, for example).

The ROR instruction is used to read into the PDP-8's accumulator any data that has been loaded into the output register by either internal memory or by externally-supplied instructions. The ROR instruction is usually preceded in the program by a test of the output flag, using the SOF instruction to determine if data is present. The ROR clears the output flag when executed. Because the output register is not buffered, any subsequent loading of the register will destroy whatever had been loaded, therefore, it is important that the output register be read before another instruction, which causes a second data transfer, is executed.

The SEF and CEF instructions sample and clear the external flag, respectively; the SOF and COF instructions sample and clear the output flag, respectively. The external and output flags are both connected to the PDP-8 skip bus and the program interrupt bus. This means that if the program interrupt facility is enabled and either of these flags is set, the PDP-8 will receive a program interrupt. Refer to the *PDP-8 Introduction To Programming* for details.

The SCR is used to determine that the Industrial 14 is running before attempting to give it any command. A loss of Run in the Industrial 14 will cause an interrupt of the PDP-8 computer for a maximum of five milliseconds until the SCR instruction is executed.

Figure 9-13 is a sample program using the DA14-E to pass instructions to the Industrial 14. The routine is called by the following instruction sequence:

```
JMS XECUTE
#OF WORDS
WORD 1
WORD 2          (If needed)
WORD 3          (If needed)
```

The XECUTE routine passes all words of the instruction to the Industrial 14, then waits for the external flag before returning. If data is loaded into the output register, it is returned in the accumulator. If the output flag is not set, the routine returns with accumulator zero.

Table 9-7
DA14-E Interface Instructions

| Symbolic | Octal | Meaning |
|---|---|---|
| GNI | 6165 | Generate an Interrupt – clears the external flag and loads one word into the memory port from the PDP-8 accumulator. When all words of the instruction have been loaded individually with the GNI, the instruction is executed and the external flag is set. GNI has no effect on the internal Industrial 14 program except as a direct result of the instruction it is used to supply. |
| LDE | 6164 | Load and execute External – with the Industrial 14 in External Mode, LDE loads the memory port from the PDP-8 accumulator identically to the GNI. The LDE allows the program counter within the Industrial 14 to increment normally with each word loaded by the LDE. |
| ROR | 6176 | Read Output Register – clears the output flag, clears the PDP-8 accumulator, then loads the output register value into the accumulator. |
| SEF | 6161 | Skip on External Flag – skip the next PDP-8 memory location if the external flag is set within the Industrial 14. The external flag is set at the conclusion of each externally-supplied instruction. |
| CEF | 6172 | Clear External Flag – clear the external flag, to remove an interrupt request once it is detected by the PDP-8. |
| SOF | 6171 | Skip on Output Flag – skips the next PDP-8 memory location if the Industrial 14 output flag is set, indicating that new data has been loaded into the output register. |
| COF | 6167 | Clear Output Flag – clear the output flag to remove an interrupt request once it is detected by the PDP-8. It is unnecessary to use the COF instruction when an ROR is used to read the output register. |
| SCR | 6175 | Skip on Controller Running – skips the next PDP-8 memory location if the Industrial 14 is running. It is not possible to enter external mode or to have any effect on the Industrial 14 if it is not running. |

```
/DA14-E EXECUTE SUBROUTINE
XECUTE,   0
          CLA
          TAD  I XECUTE   /GET WORD COUNT
          CIA
          DCA    CNTR     /SET UP COUNTER
XECUTE1,  ISZ    XECUTE   /INCREMENT POINTER
          TAD  I XECUTE   /PICK UP NEXT WORD
          GNI             /SEND TO THE 14
          CLA
          ISZ    CNTR     /ALL WORDS SENT?
          JMP    XECUT1   /NO: GET NEXT
          ISZ    XECUTE   /YES: SET UP RETURN
          SEF             /WAIT FOR EXTERNAL FLAG
          JMP    .-1
          SOF              /OUTPUT FLAG SET?
          SKP              /NO: RETURN WITH AC=0
          ROR              /YES: READ OUTPUT REGISTER INTO AC
          JMP  I XECUTE   /RETURN
CNTR,     0
```

Figure 9-13 DA14-E Execute Subroutine

Program examples at the end of this chapter make use of this routine to poll I/O points, load memory, etc. This routine does not make use of the program interrupt facility, nor does it handle an Industrial 14-initiated transfer. If the monitoring computer expects to receive status reporting data from the Industrial 14, it is recommended that program interrupt techniques be used. The skip chain for handling an interrupt would include the SOF instruction to detect the presence of an Industrial 14-initiated transfer.

```
  .
  .
  .          Save program constants

SOF          /is Industrial 14 interrupting?
SKP          /No: Check other devices
JMP  DATA 14 /Yes: Service by reading output
  .          /register
  .          Service slower devices.
  .
```

It is recommended that the Industrial 14's output register be serviced as early in the skip chain as possible to protect against lost data.

### DC14-F SERIAL INTERFACE

The DC14-F is a standard EIA interface for the Industrial 14 that normally transmits data at 9600 baud. It can communicate to a PDP-8/E, M or F, using a KL8-JA Teletype interface, or to a PDP-11, using a DL11 interface. (Consult Digital Equipment Corporation for information on the use of the DC14-F at rates other than 9600 baud.)

Each word to be transmitted to the Industrial 14 is split into two separately-transmitted characters as shown in Figure 9-14. Two mode bits specify the form in which the instruction is to be executed: Interrupt Mode (no PC increment) or External Load Mode (PC increment). Setting both mode bits to 1 when using the DC14-F is equivalent to transferring the word with a GNI when using the DA14-E. Similarly, setting ON only Mode Bit 0 when using the DC14-F is equivalent to transferring the word with an LDE instruction when using the DA14-E. Normally, only Interrupt Mode (both mode bits = 1) is used, since it is undesirable to affect the program counter except by a direct JMP instruction.

When transmitted, an optional eighth bit can be added for parity, depending upon switch selection in the DC14-F interface and in the computer interface (KL8-JA or DL11-C). A start and a stop bit complete the number of bits in the character transmitted.

Notice that two-word instructions require the transmission of four characters and three-word instructions require six characters. Similarly, data received from the Industrial 14 comprises two seven-bit characters as shown in Figure 9-15 (plus parity, start, and stop bits). The six most-significant bits are received first, together with the equivalent of the

external flag. The six least-significant bits are received in the second character with the equivalent of the output flag. Parity is generated and can be checked by the receiving computer interface.

Several error checks are performed by the DC14-F interface; parity is checked for each character as it is received. If a parity error is detected, the DC14-F clears its buffers without causing the controller to execute the instructions, and returns two all-zero characters to the computer. This is equivalent to external flag = 0, output flag = 0, and data word = 0.

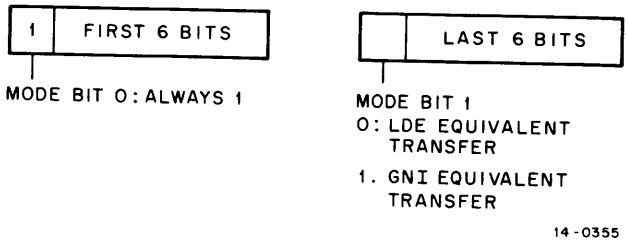



Figure 9-14 DC14-F Receiver Data Format

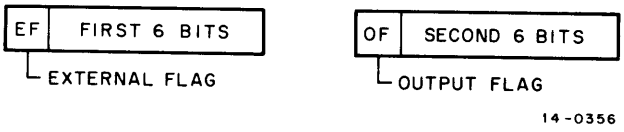



Figure 9-15 DC14-F Transmitter Data Format

A second check ensures proper synchronization between the controller and the computer. Because multiple characters (two, four or six) are required for a single interrupt of the Industrial 14, the hardware must protect against the execution of misconstructed instructions. This protection is provided by restricting the amount of time allowed between characters of an externally-supplied instruction. Once the first character of an instruction is sent to the Industrial 14, each successive character must be transmitted within one-half character time. At the normal transmission speed of 9600 baud, this means that no more than 500 microseconds can be tolerated between XMIT instructions until the complete instruction of up to six characters has been sent. If the Industrial 14 does not receive each character within the allotted time, it ignores all the characters and initializes itself. The computer can detect this loss of transfer as it will not receive the transmission of the external flag, which will occur if the complete instruction is received and executed.

The KL8-JA instructions, used to transfer data to the Industrial 14, are listed in Table 9-8. These instructions include a group for the transmitter to the Industrial 14 (octal codes 631X) and for the receiver from the Industrial 14 (octal codes 630X). The XMIT instruction is used to send a character to the Industrial 14 and the STDF checks to determine if the interface is ready to accept the next character. The SRCF is used to determine if data has returned from the Industrial 14 and the RDRD reads the received data into the PDP-8 accumulator.

The LDSE instruction allows the program to disconnect the KL8-JA interface from the interrupt bus.

This permits use of the program interrupt facility for other devices without causing interrupts for the Industrial 14. The LDSE also controls the reading of error status with each data word read by an RDRD instruction. The error status obtained when it is enabled is shown in Figure 9-16. For a further discussion of these errors, consult the *KL8-JA Asynchronous Data Terminal Control Manual* (DEC-8E-HRJC-D-KL8-JA).

Figure 9-17 is a sample program using the DC14-F and KL8-JA to pass instructions to the Industrial 14. Its function is identical to the similar routine presented in Figure 9-13 for the DA14-E. It splits all words of the instruction into two characters, transmits all characters, receives both characters of the return transmission and returns with the output register value (if any) in the PDP-8's accumulator. Like the DA14-E XECUTE routine, it is usable for all examples presented at the end of this chapter.

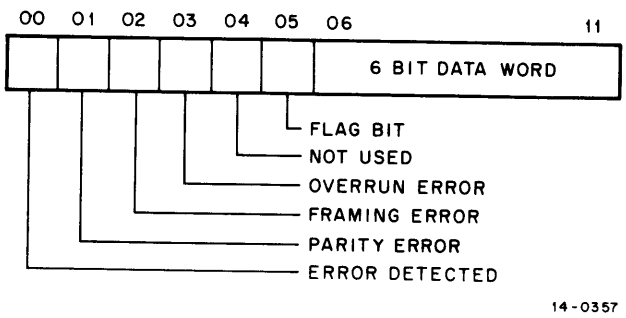



Figure 9-16 KL8-JA Receiver Word

**Table 9-8**
**KL8-JA Instructions for DC14-F Communication**

| Symbolic | Octal | Meaning |
|---|---|---|
| STTF | 6310 | Set Transmitter Flag – sets the transmit done flag without transmitting a character. |
| STDF | 6311 | Skip on Transmit Done Flag |
| CTDF | 6312 | Clear Transmit Done Flag |
| SKPI | 6315 | Skip on Interrupt – skips the next sequential PDP-8 memory location if either the transmit or receive flag is set, provided the interrupt enable is set. |
| XMIT | 6316 | Transmit – clears the transmit flag and transfers the character from the seven least-significant bits of the accumulator to the Industrial 14. |
| SRCF | 6301 | Skip on Receiver Flag |
| CRCF | 6302 | Clear Receiver Flag – also clears the PDP-8 accumulator. |
| RDCO | 6304 | Receive Data ORed – reads the receiver data inclusively ORing it with any data already in the PDP-8 accumulator. |
| LDSE | 6305 | Load Status Enable – set interrupt enable according to AC bit 11 and status word enable according to bit 10. (1) = enable, (0) = disable. |
| RDRD | 6306 | Receiver Data Read – read data from the Industrial 14 into the PDP-8 accumulator. Error status is also read provided it is enabled. |

The DC14-F is not recommended for status reporting (transition monitoring) from the Industrial 14 to the computer because it does not double buffer the output register. If several instructions in the Industrial 14's memory could all load the output register within one transmission time (approximately 1.1 ms), lost data would result. Therefore, initiated transfers for status reporting are possible over the DC14-F only if two such reports can never occur within the same 1.1 ms period.

## PROGRAM EXAMPLES

This section presents examples of monitoring functions with the Industrial 14. The examples make use of the "XECUTE" subroutines, introduced previously, and are therefore independent of the selected interface.

Figure 9-18 is an instruction sequence to read the state of a single I/O point. It can be used to read the state of any input, output, internal function or disable state using the proper value for IONUM (Table 9-9).

The program in Figure 9-19 will read eight points in parallel. Only the I/O functions appended by an asterisk may be read in parallel.

The program in Figure 9-20 will set an input, output, retentive memory, or I/O disable ON. A similar routine could be used for CLRBIT or LDBIT. This routine would only be used for the I/O numbers followed by an asterisk (Table 9-9).

The technique for loading, clearing or setting eight points in parallel is illustrated in Figure 9-21.

Loading and reading timers uses the technique shown in Figures 9-22 and 9-23.

```
/DC14-F/KL8-JA EXECUTE SUBROUTINE
XECUTE,  0
         CLA
         TAD   I   XECUTE     /GET WORD COUNT
         CIA
         DCA       CNTR       /SET UP COUNTER
XECUT1,  ISZ       XECUTE     /INCREMENT POINTER
         TAD   I   XECUTE     /PICK UP NEXT WORD
         DCA       SAVE       /STORE IT TEMPORARILY
         CLA CLL CMA RAL      /SET AC = -2
         DCA       CNTR2      /SET UP CHARACTER COUNTER
         TAD       SAVE       /GET WORD
         BSW                  /RTR; RTR, RTR, IN PDP-8/L
XECUT2,  AND       K77        /GET NEXT CHARACTER
         TAD       K100       /SET INTERRUPT MODE BITS
         XMIT                 /SEND CHARACTER
         STDF                 /WAIT FOR TRANSMITTER DONE
         JMP .-1
         CLA
         TAD       SAVE       /GET WORD
         ISZ       CNTR2      /SENT BOTH CHARACTERS?
         JMP       XECUT2     /NO: SEND NEXT
         CLA                  /YES
         ISZ       CNTR       /ALL WORDS SENT
         JMP       XECUT1     /NO: GET NEXT WORD
         ISZ       XECUTE     /YES: SET UP RETURN
         CLA CLL CMA RAL      /SET AC = -2
         DCA       CNTR2      /SET UP RECEIVED CHARACTER COUNTER
WAIT,    DCA       CNTR3      /SET UP OVERTIME COUNTER FOR 4K
         SRCF
         SKP
         JMP       XECUT3
         ISZ       CNTR3      /WAITED LONG ENOUGH?
         JMP       WAIT+1     /NO: CHECK FLAG AGAIN
         JMP       ERROR1     /YES: ERROR
```

(Continued on next page)

Figure 9-17 DC14-F Execute Subroutine

```
XECUT3,  CLA CLL CML RTL        /SET AC = 2
         LDSE                   /SET TO READ ERROR STATUS
         RDRD                   /READ CHARACTER AND STATUS
         SPA
         JMP        ERROR2      /COMMUNICATIONS ERROR
         ISZ        CNTR2       /NO ERRORS:  WHICH CHARACTER?
         JMP        XECUT4      /FIRST HALF
         AND        K177        /STRIP TO 7 BITS AND ADD CONSTANT
         TAD        K7700       /TO SET LINK = OUTPUT FLAG
         AND        K77         /STRIP TO 6-BIT DATA WORD
         TAD        OUTREG      /COMBINE WITH FIRST HALF
         JMP  I     XECUTE      /RETURN WITH VALUE IN AC
XECUT4,  CLL RTL                /FIRST WORD – MOVE LEFT
         RTL                    /TO PUT FLAG IN LINK AND WORD
         RTL                    /IN THE MOST SIG. BITS
         AND        K7700       /STRIP TO SIX BITS
         DCA        OUTREG      /SAVE FIRST HALF
         SZL
         JMP        WAIT
ERROR,   .                      /IF EXTERNAL FLAG NOT SET
         .                      /PROBABLY DUE TO PARITY ERROR
         .                      /AT RECEIVING DC14-F. SHOULD RETRANSMIT
                                /COMPLETE INSTRUCTION

ERROR1,  .                      /NO RETURN IN REASONABLE DELAY
         .                      /FAULTY INTERFACE OR STOPPED 14.
         .

ERROR2,  .                      /ERROR DETECTED ON INCOMING
         .                      /WORD FROM 14.
         .

CNTR,    0
SAVE,    0
CNTR2,   0
K77,     77
K100,    100
CNTR3,   0
K177,    177
K7700,   7700
OUTREG,  0

XMIT =   6316                   /KL8-JA TRANSMIT INSTRUCTION
STDF =   6311                   /KL8-JA SKIP ON TRANSMIT DONE
SRCF =   6301                   /KL8-JA SKIP ON RECEIVER FLAG
LDSE =   6305                   /KL8-JA LOAD STATUS DISABLE
RDRD =   6306                   /KL8-JA READ RECEIVED DATA
```

Figure 9-17 DC14-F Execute Subroutine (Continued)

```
POLL,    JMS      XECUTE    /CALL EXECUTE ROUTINE
         2
         RDBIT
POINT,   IONUM
         CIA                /CHECK FOR PROPER RETURNED WORD
         TAD POINT
         SNA
         JMP OFF            /I/O POINT IS OFF
         CLL RAL
         SNA
         JMP ON             /I/O POINT IS ON
         JMP ERROR          /I/O POINT IMPROPERLY READ
RDBIT =  0136
IONUM =                     /POINT TO BE TESTED
```

Figure 9-18 Polling a Single Point

**Table 9-9**
**I/O Number Ranges**

| Polled State | Range For IONUM |
|---|---|
| Input* | 0–777 |
| Output* | 1000–1377 |
| Retentive Memories* | 1400 – I/O Partition |
| Timers | |
| Instantaneous | Odd Address Beyond I/O Partition |
| Delayed | Even Address Beyond I/O Partition |
| Event Counters | |
| Preset reached | Odd Address Beyond I/O Partition |
| Up/Down Counters | |
| Equal preset | 1760, 1764, 1770, 1774 |
| Equal 0 | 1761, 1765, 1771, 1775 |
| External I/O Disables* | 2000–3377 |

*Valid for both BIT and WD operations

```
READ8,  JMS       XECUTE   /CALL EXECUTE ROUTINE
        2
        RDWD
        IONUM
        CLL RAL
        SPA SZL
        JMP ERROR          /YES: ERROR
        RTL
        SPA SZL            /EITHER OF NEXT TWO BITS SET?
        JMP ERROR          /YES: ERROR
        RTL                /MOVE WORD LEFT
        SZL                /I/O POINT 0 ON?
        JMS       ON0      /YES
        SPA                /I/O POINT 1 ON?
        JMS       ON1      /YES
        RTL                /MOVE WORD LEFT
        SZL                /I/O POINT 2 ON?
        JMS       ON2      /YES
        SPA                /I/O POINT 3 ON?
        JMS       ON3      /YES
        RTL                /MOVE WORD LEFT
        SZL                /I/O POINT 4 ON?
        JMS       ON4      /YES
        SPA                /I/O POINT 5 ON?
        JMS       ON5      /YES
        RTL                /MOVE WORD LEFT
        SZL                /I/O POINT 6 ON?
        JMS       ON6      /YES
        SPA                /I/O POINT 7 ON?
        JMS       ON7      /YES
          .
          .
          .

RDWD =  0036
IONUM =                    /ONE POINT IN GROUP OF 8 TO BE
                           /TESTED
```

Figure 9-19 Polling Eight Points in Parallel

```
FORCE,      JMS         XECUTE
            2                       /FOR CLRBIT, OR SETBIT:  3 FOR LDBIT
            SETBIT
            IONUM                   /ALSO SUPPLY A MASK FOR LDBIT
            SZA
            JMP         ERROR
              .
              .
              .
SET BIT =   0113                    /COULD USE CLRBIT OR LDBIT
IONUM =                             /POINT TO BE SET ON
```

Figure 9-20 Setting a Single I/O Bit

```
FORCE8,     JMS         XECUTE
            3                       /FOR LDWD; 2 FOR SET WD OR CLRWD
            LDWD
            MASK                    /FOR LDWD ONLY
            IONUM
            SZA
            JMP         ERROR
              .
              .
              .
LDWD =      0023                    /COULD USE SETWD OR CLRWD
MASK =                              /STATE OF POINTS TO BE SET
IONUM =                             /I/O POINTS TO BE LOADED
```

Figure 9-21 Loading Eight I/O Points in Parallel

```
TMRSET,   JMS       XECUTE
          3
          LDWD
          VALUE
          IONUM
          SZA
          JMP       ERROR
            .
            .
            .
LDWD =    0023
VALUE =                    /TIMER VALUE IN OCTAL WITH PROPER CONTROL BITS
IONUM =                    /ODD I/O NUMBER OF TIMER
```

Figure 9-22 Loading a Timer Preset

```
TMRRD,    JMS       XECUTE
          2
          RDWD
          IONUM
            .
            .
            .
RDWD =    0036
IONUM =                    /ODD ADDRESS TO READ PRESET
                           /EVEN ADDRESS TO READ CURRENT VALUE
```

Figure 9-23 Reading a Timer Preset

Figure 9-24 shows a technique for reading memory content from the Industrial 14. A simple looping routine could be added to successively compare memory content read from the Industrial 14 to a core image stored within the monitoring computer. Reading of memory does not require the controller to be in External Mode and can actually be done without affecting the control of the system by the Industrial 14.

Writing memory as seen in Figure 9-25 requires the Industrial 14 to be in External Mode. Note the use of the CLR instruction to turn all outputs off while memory is written. Simple modifications would load a whole section of memory rather than just a single location. Once loaded, memory should be verified, then the internal program is started by setting the program counter and causing the Industrial 14 to leave External Mode.

```
/READ 14 MEMORY ROUTINE
READM,      CLA
            JMS         XECUTE      /SET DATA FIELD TO EITHER
            1                       /FIELD ZERO OR ONE AS DESIRED
            CDFZ14
            SZA                     /NO OUTPUT REGISTER SHOULD HAVE
            JMP         ERROR       /BEEN RETURNED
            JMS         XECUTE      /READ MEMORY CONTENT AT
            2                       /SPECIFIED LOCATION
            RDMEM
            LOCADD                  /LOCATION TO BE READ
            DCA         MEM14       /MEMORY CONTENT AS IN AC
              .
              .
              .
CDFZ14 =    0600 or 0700            /EITHER CDF0 or CDF1 FOR THE 14
LDMEM =     0026
LOCADD =                            /ADDRESS OF LOCATION TO BE READ
```

Figure 9-24 Reading a Memory Location Within the Industrial 14

```
/WRITE 14 MEMORY ROUTINE
WRITEM,     CLA
            JMS         XECUTE      /FORCE 14 INTO EXTERNAL MODE
            1
            EEM
            SZA
            JMP         ERROR       /A RETURNED VALUE IS AN ERROR
            JMS         XECUTE      /CLEAR ALL 14 OUTPUTS
            1
            CLR
            SZA
            JMP         ERROR       /A RETURNED VALUE IS AN ERROR
            JMS         XECUTE      /SET INSTRUCTION FIELD FOR LOAD
            1
            CIFZ14
            SZA
```

(Continued on next page)

Figure 9-25 Writing a Memory Location Within the Industrial 14

```
        JMP     ERROR   /A RETURNED VALUE IS AN ERROR
        JMS     XECUTE  /SET PROGRAM COUNTER FOR
        2               /LOCATION TO BE WRITTEN
        JMP14
        LOCADD
        SZA
        JMP     ERROR   /A RETURNED VALUE IS AN ERROR
        JMS     XECUTE  /WRITE ONE WORD OF 14 MEMORY
        2
        LDMEM
        LOCVAL
        SZA
        JMP     ERROR   /A RETURNED VALUE IS AN ERROR
          .             MODIFY OTHER LOCATIONS AND VERIFY
          .             PROPER LOADING BY READING MEMORY
          .
        JMS     XECUTE  /SET PROGRAM COUNTER TO
        1               /STARTING LOCATION
        CLRPC
        SZA
        JMP     ERROR   /A RETURNED VALUE IS AN ERROR
        JMS     XECUTE  /RELEASE 14 FROM EXTERNAL MODE
        1
        LEM
        SZA
        JMP     ERROR   /A RETURNED VALUE IS AN ERROR
          .
          .
          .

EEM =       0060
CLR =       0170
CIFZ14 =    0020 or 0030    /EITHER CIF0 OR CIF1 FOR THE 14.
JMP 14 =    0024
LDMEM =     0022
LOCADD =                    /MEMORY ADDRESS TO BE LOADED
LOCVAL =                    /MEMORY VALUE TO BE STORED
CLRPC =     0004
LEM =       0040
```

Figure 9-25 (Continued) Writing a Memory Location Within the Industrial 14

# APPENDIX A
# INPUT ASSIGNMENT SHEET

## INPUT ASSIGNMENT SHEET

| I/O Number | Symbol | Function | Normal Condition |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# APPENDIX B
# OUTPUT ASSIGNMENT SHEET

## OUTPUT ASSIGNMENT SHEET

| I/O Number | Symbol | Function |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# APPENDIX C
# INTERNAL FUNCTION ASSIGNMENT SHEET

## INTERNAL FUNCTIONS ASSIGNMENT SHEET

| I/O Number | Symbol | Function | Preset or Limits | Type |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# APPENDIX D
# INDUSTRIAL 14 PROGRAM COMPATIBILITY WITH THE VT14

Industrial 14 programs prepared by PAL-143 and BOOL-143 and debugged by ODP-143 may not be compatible with the VT14 Programming Terminal and are therefore incapable of being changed.

The VT14 Programming Terminal recognizes only the instructions in Table D-1. Special attention must be given to the restrictions posed by these instructions within the Industrial 14 Program.

**NOTE**

**Industrial 14 monitoring, subroutine, and change field instructions are incompatible with the VT14.**

**JFN Restriction**

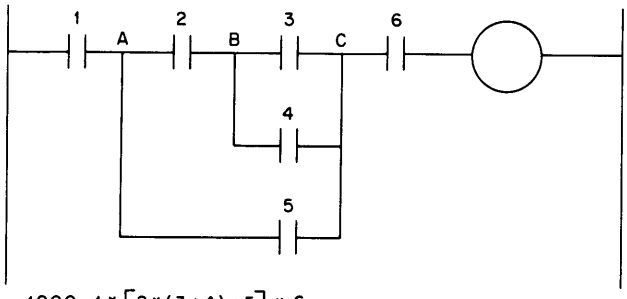
The JFN instruction is caused by the last or only entry into a branch (going from left to right) or the last instruction before the set output ON position in a circuit diagram; otherwise a JFF instruction should be used.

For example, the JFN instructions in locations 401, 403, and 411 (Figure D-1) are caused by the last entry into branches A, B, and C.

If a JFN instruction is inserted incorrectly, it is displayed as a dummy branch on the VT14 Programming Terminal. If an attempt is made to store a circuit with dummy branches, the VT14 will output a "FORMAT ERROR" message.

**CLRPC Restriction**

Binary tapes read by the VT14 have only one CLRPC instruction in memory; this is always stored in location 7377 of the highest memory field. To ensure this, the VT14 should be used to clear memory first, and then load the program tapes via the Teletype and VT14.




1000 = 1 * [2 * (3 + 4) + 5] * 6

```
400  TF   0001      410  TF   0005
401  JFN  16        411  JFN  16
402  TF   0002      412  TF   0006
403  JFN  10        413  JFN  0016
404  TF   0003      414  SN   1000
405  JFF  12        415  SKP
406  TF   0004      416  SF   1000
407  JFF  12
```

14-0282

Figure D-1 JFN Restriction Ladder Diagram, Boolean Equation and Equivalent Industrial 14 Instructions

If ODP-143 is used to load the Industrial 14 program, the user *must* validate that only one CLRPC instruction is in memory (location 7377 of the highest available memory field).

**NOP Restriction**

To omit sensing a contact (TF or TN) using ODP-143, a NOP instruction must not be inserted in its place. The VT14 interprets the NOP instruction as the end of a circuit. Thus, in Figure D-2, if input 2 is not needed, insert an instruction sensing one of the adjoining inputs (TF1 or TF3) instead of a NOP instruction.

**Table D-1**
**Industrial 14 Instructions Compatible With the VT14**

| Instruction | | Meaning | Restriction |
|---|---|---|---|
| **Symbolic** | **Numeric** | | |
| TF IO | 4000 + IO | Test an input, output, or internal function for the *OFF* state. | None |
| TN IO | 6000 + IO | Test an input, output, or internal function for the *ON* state. | None |
| SF | 0000 + 0 | Set output or internal function 0 OFF. | Instructions must always be programmed in the order:<br>SN<br>SKP<br>SF |
| SN | 2000 + 0 | Set output or internal function 0 ON | The SN instruction must always occupy a location whose address is divisible by 4 |
| JFF NNN | 2000 + NNN | Jump to location NNN if the test flag is OFF. | Only the last entry into a branch left-to-right causes a JFN instruction; otherwise, all other jump instructions should be JFF. |
| JFN NNN | 2400 + NNN | Jump to location NNN if the test flag is ON. | A JFN instruction must always precede an SN position. |
| SKP | 0010 | Skip the following memory location unconditionally. | None |
| CLRPC | 0004 | Clear the program counter. Causes an unconditional jump to Location 0 in Field 0. | This instruction must occupy only location 7377 in the highest memory field. |
| NOP | 0000 | No operation | This instruction signifies the end of a circuit to the VT14. |
| MOVBIT | 0133 | Move I/O bit from I/O address to I/O address. | This instruction should only be used in a shift circuit. Refer to the format described in the Movbit Restriction (following). |

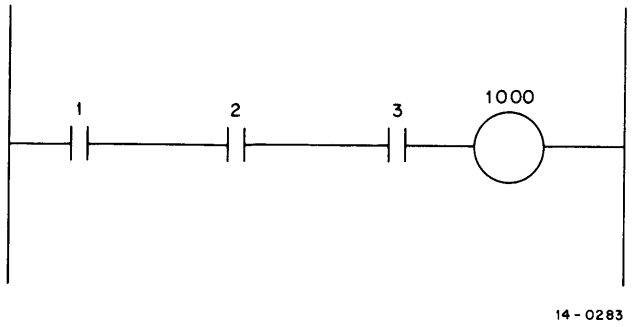



Figure D-2 Circuit Diagram

The original and corrected Industrial 14 instructions follow:

| | Original | | Corrected | |
|---|---|---|---|---|
| 200 | TF | 0001 | TF | 0001 |
| 201 | TF | 0002 | TF | 0001 |
| 202 | TF | 0003 | TF | 0003 |
| 203 | JFN | 006 | JFN | 006 |
| 204 | SN | 1000 | SN | 1000 |
| 205 | SKP | | SKP | |
| 206 | SF | 1000 | SF | 1000 |

**Movbit Restrictions**

Before changing the shift register limits with ODP-143, the user should become familiar with the Industrial 14 instruction format for a shifter circuit (Chapter 3). The following example illustrates the method for changing shift register limits while still maintaining VT14 compatibility.

To reduce the shift register limits to 1420 as the low limit, and 1423 as the high limit, movbit instructions and two conditional jump instructions must be altered.

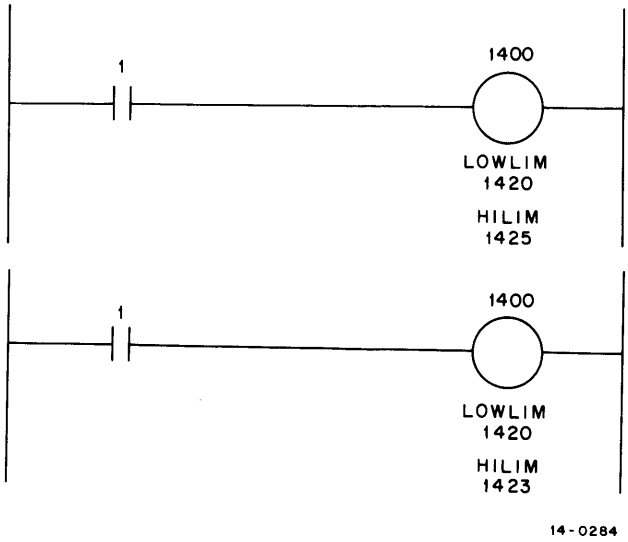



Figure D-3 Shift Register Circuits for VT14 Compatibility

The following are the original and changed Industrial 14 instructions:

| Original Industrial 14 Instructions | | | Changed Industrial 14 Instructions | | |
|---|---|---|---|---|---|
| 1000 | TF | 0001 | 1000 | TF | 0001 |
| 1001 | JFN | .+3 | 1001 | JFN | .+3 |
| 1002 | JFF | .+4 | 1002 | JFF | .+4 |
| 1003 | SKP | | 1003 | SKP | |
| 1004 | SF | 1400 | 1004 | SF | 1400 |
| 1005 | JFF | .+23 | 1005 | JFF | .+15 |
| 1006 | TN | 1400 | 1006 | TN | 1400 |
| 1007 | SN | 1400 | 1007 | SN | 1400 |
| 1010 | JFN | .+20 | 1010 | JFN | .+12 |
| 1011 | MOVBIT | | 1011 | MOVBIT | |
| 1012 | 1424 | | 1012 | 1422 | |
| 1013 | 1425 | | 1013 | 1423 | |
| 1014 | MOVBIT | | 1014 | MOVBIT | |
| 1015 | 1423 | | 1015 | 1421 | |
| 1016 | 1424 | | 1016 | 1422 | |
| 1017 | MOVBIT | | 1017 | MOVBIT | |
| 1020 | 1422 | | 1020 | 1420 | |
| 1021 | 1423 | | 1021 | 1421 | |
| 1022 | MOVBIT | | 1022 | NOP | |
| 1023 | 1421 | | | | |
| 1024 | 1422 | | | | |
| 1025 | MOVBIT | | | | |
| 1026 | 1420 | | | | |
| 1027 | 1421 | | | | |
| 1030 | NOP | | | | |

INDUSTRIAL 14 SOFTWARE MANUAL
DEC-14-ISUMA-B-D

# Reader's Comments

**Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.**

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? _____

What features are most useful? _____

What faults do you find with the manual? _____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

Would you please indicate any factual errors you have found. _____

Please describe your position. _____

Name _____ Organization _____

Street _____ Department _____

City _____ State _____ Zip or Country _____

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 Equipment Corporation
Technical Documentation Department
146 Main Street
Maynard, Massachusetts 01754