3. Match the symbols to their correct definitions. ***Note:*** *not all 9 definitions are used.*

a. >	_____	1. less than 1
b. <	_____	2. less than
c. =	_____	3. less than or equal to
d. ≥	_____	4. greater than
e. ≠	_____	5. greater than or equal to
f. <>	_____	6. equal to
g. >=	_____	7. not equal to
h. <=	_____	8. not equal to 1
i. ≤	_____	9. greater than 1

4. Define the term *data compare*.
5. Write a program that compares the accumulated value of T4:0 to a constant of 250. The instruction is to be true when the accumulated value of T4:0 is greater than 250.
6. Define the term *mask*.
7. Give an example of how an Allen-Bradley LIM instruction is used.

CHAPTER

15 Math Functions

Objectives

After completing this chapter, you should have the knowledge to:

- List the four standard math functions available with most PLCs.
- Discuss the math functions and give examples of how they are used.
- Program the square root function.

USING MATH FUNCTIONS

The four basic math, or arithmetic, functions are addition (+), subtraction (–), multiplication (×), and division (÷), and can be used with constant values or values stored in a storage register, holding register, input/output registers, or any other accessible word locations.

A typical application of an arithmetic, or math function may be a chemical batch plant where a given mix of two chemicals (A and B) is to have a 2:1 ratio. By using analog input devices (discussed in Chapter 2), the weight of the first chemical, A, can be converted to a binary equivalent and stored. For a 2:1 mix, only one half as much of chemical B (by weight) is needed. The binary value of the weight that is stored can be divided by two to determine the amount of chemical B that is needed for a proper 2:1 mix.

ALLEN-BRADLEY PLC-5, SLC 500, AND MICROLOGIX MATH INSTRUCTIONS

Figure 15–1 illustrates a divide (DIV) instruction showing how the value of chemical A (Source A-N7:10) can be divided by 2 (Source B), and the result placed into Destination N7:3. A data comparison is then made to the data stored in integer file N7:3 to limit the amount of chemical B to one-half the amount of chemical A.

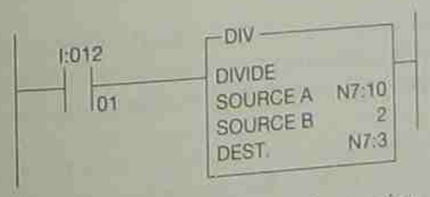


Figure 15–1 Divide (DIV) Instruction

When I:012/01 is closed, the value in Source A is divided by 2, and the result is stored in Destination word N7:3 on each processor scan. Once the total weight of chemical A has been determined and the amount divided by 2, the result (which is now stored in N7:3) is programmed with a DATA COMPARE instruction to control the feed of chemical B.

Figure 15–2 shows how an NEQ (not equal to) instruction is programmed to control the amount of chemical B that is to be mixed with chemical A.

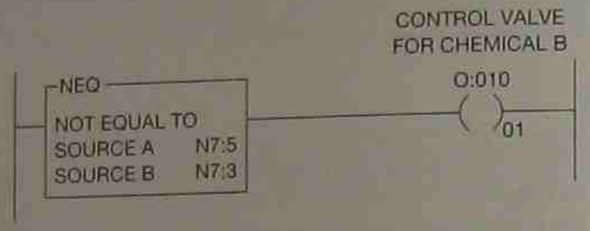


Figure 15–2 Data Compare Instruction NEQ (not equal) to Control Amount of Chemical B

As shown in the figure, output O:010/01, the control valve for chemical B, is true, or *ON*, as long as the value in source A is not equal to Source B. Source A, N7:5, is the word that stores the weight of chemical B. When the weight of chemical B is equal to one-half the weight of chemical A (the value stored in N7:3, Source B), the instruction goes false and the valve to chemical B is closed.

The math function might also be used with timer and/or counter values, accumulated or preset, to change a given process machine operation under varying conditions.

The DIV instruction divides the value in source A by the value of Source B and stores the answer in the destination word. Figure 15–3 shows the DIV instruction.

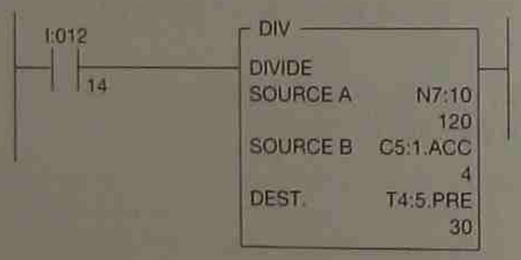


Figure 15–3 Divide (DIV) Instruction Used to Enter Preset Value in a Timer

When input device I:012/14 is true, the value of Source A, address N7:10 (120), is divided by the value of source B, 4, which is the accumulated value of counter 1 in counter file 5. The results of the division, 30, become the preset value of timer 5 in timer file 4.

Figure 15–4 shows the ADD instruction. This instruction adds the value in Source A to the value in Source B and places the results (answer) into the destination address. In the example, Source A is shown as N7:1 which indicates that this is word 1 of File 7, the integer file.

Source B is word 2 of the integer file (file 7), and the Destination is shown as word 10 of File 7. When Input address I:012/10 is true, the value in Source A (20) is added to the value in Source B (40) and the results (60) are placed in the Destination word.

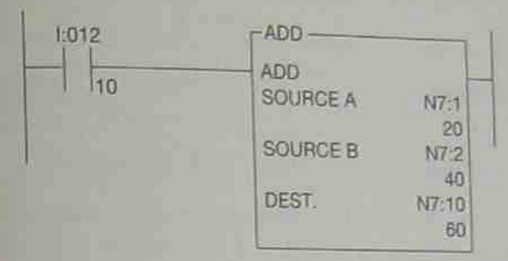


Figure 15–4 Addition (ADD) Instruction

Source A and B can be the value of a word address or can be a constant. If the results of the addition are larger than +32,767 or –32,768, an overflow bit (bit 1) in word 0 of the status file, File S, will be set to 1. Additional status bits for word 0 are: bit 0 which is set if a carry is generated; bit 2 which is set if the results of the math are 0 after the instruction has been implemented; or bit 3 indicates a negative value after the math instruction has been executed. Figure 15–5 shows the arithmetic status bits.

STATUS BIT	CONTROLLER ACTION
S:0/0 Carry (C)	set to 1 if a carry is generated
S:0/1 Overflow (V)	set to 1 when the results of the math will not fit the destination address
S:0/2 Zero (Z)	set to 1 if the math operation results in an answer of 0
S:0/3 Sign (S)	set to 1 when the results of the math instruction are negative (less than 0)

Figure 15–5 Arithmetic Status Bits

With the MicroLogix and the SLC 5/02, 5/03, and 5/04 processors, addition and subtraction using 32-bit words is possible. This allows for addition and subtraction beyond the normal limits of +32,767 and –32,768. For instructions on how to add or subtract 32-bit words, refer to the programming manual for the processor being used.

Figure 15–6 shows the subtract (SUB) instruction. With this instruction, the value in Source B is subtracted from the value in Source A and the results are placed in the destination file (DEST).

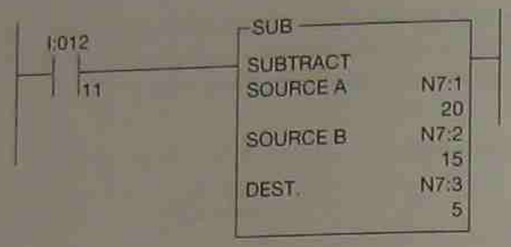


Figure 15–6 Subtract (SUB) Instruction

When input device I:012/11 goes true, the value in Source B, address N7:2 (15), is subtracted from the value in Source A, address N7:1 (20), and the result, 5, is stored in destination address N7:3 (5).

The multiply (MUL) instruction used for the PLC-5, SLC 500, and MicroLogix is shown in Figure 15–7.

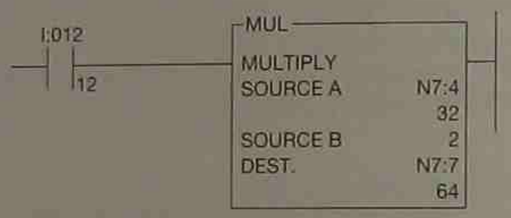


Figure 15–7 Multiply (MUL) Instruction

Like previous instructions, the MUL instruction multiplies the value in Source A times the value in Source B and place the answer into the destination address. In this example, when address I:012/12 goes true, the value stored in Source A, address N7:4 (32), is multiplied by a constant value of 2 and the answer, 64, is placed into destination word N7:7. As with previous math instructions, Source A and B can be values (constants) or addresses that contain values.

For the Allen-Bradley PLC-5 and the SLC 500 5/03, 5/04, and 5/05 processors, a more powerful math instruction called the compute (CPT) is available.

The CPT instruction is an output instruction that performs the operations defined in the expression and then writes the results to the destination address. Figure 15–8 shows the CPT instruction format.

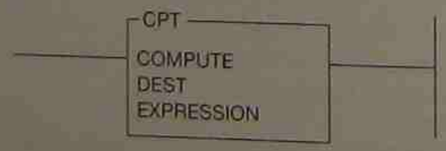


Figure 15–8 PLC-5 Compute (CPT) Instruction Format

The typical math functions that can be performed with the CPT instruction are add (+), subtract (–), multiply (*), divide (|), negate (–), square root (SQR), and exponential (**). The table in Figure 15–9 shows the functions and an example of each operation. The complexity of the math performed will vary with the PLC model.

Note: *Different symbols are used for some operations due to limited symbols on the keyboard.*

OPERATOR	DESCRIPTION	EXAMPLE
+	ADD	2 + 3 + 7
–	SUBTRACT	12 – 5
*	MULTIPLY	6 * (5 * 2)
|	DIVIDE	24 | 4
–	NEGATE (NEGATIVE)	–N7:0
SQR	SQUARE ROOT	SQR N7:1
**	EXPONENTIAL	10 ** 3 OR 10^3

Figure 15–9 Compute (CPT) Operators (Symbols)

Figure 15–10 shows how a CPT instruction is typically programmed. In some PLC-5 models, but not all, any combination of operators may be used with various addresses and/or constants. The expression can be up to 80 characters in length.

Figure 15–10 Programmed PLC-5 Compute (CPT) Instruction

When the instruction first appears on the screen during programming, the destination must be specified first. The destination address in the example is T4:1.ACC. Then the expression or math formula is entered. The expression states that the value in N7:0 is to be added to the value found in N10:1. The sum is then multiplied by 4.5. The result of the computation is then sent to the designated destination, in this case, the timer file, file 4, timer 1, accumulated value (T4:1.ACC).

Additional math functions available for the PLC-5, SLC 500, and MicroLogix are the Clear (CLR) and Square Root (SQR) instructions.

The clear instruction (CLR), as the name implies, clears the value in a word to zero. Figure 15–11 shows the CLR instruction.

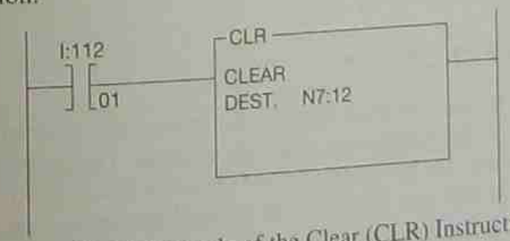


Figure 15–11 Example of the Clear (CLR) Instruction

When the input address is true, the value in the destination word is set to zero. That means that all 16 bits of the word 12, in integer file 7, are cleared or set to zero.

The Square Root instruction (SQR) is shown in Figure 15–12.

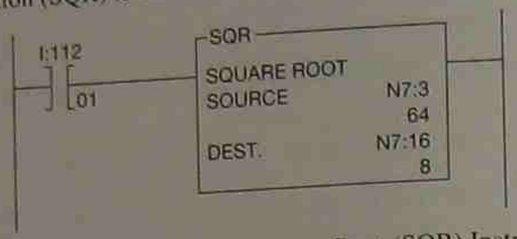


Figure 15–12 Example of the Square Root (SQR) Instruction

When the logic that precedes the SQR instruction is true, the square root of the Source value is transferred into the destination word. In the example in Figure 15–12, the square root of the source address, N7:3 (64), which is 8, is placed into destination word 16 of the N7 file.

Some PLC-5 processors and the SLC-5/02, 03, 04, and 05 processors also can perform the trigonometric functions sine (SIN), cosine (COS), tangent (TAN), arc sine (ASN), arc cosine (ARS), and arc tangent (ATN). These PLCs can also raise a number to a power by using the X to the Power of Y (XPY) instruction. This instruction is shown in Figure 15–13.

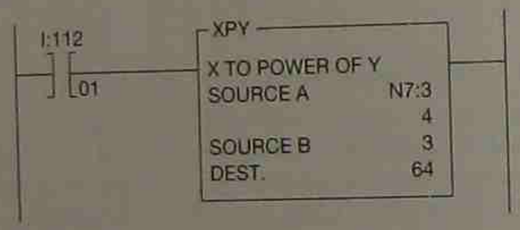


Figure 15–13 Example of the XPY Instruction

If we wanted to raise the value of a word by a power of 3, which is the same as saying X^3 or $X \cdot X \cdot X$ we could use the XPY instruction. If the value Source A, word N7:3, is 4, and this value is to be raised by a power of 3, Source B, the destination would hold the result of raising 4 by a power of 3, which is 64. Raising 4 by a power of 3 could be written as 4^3 or could be written as $4 \cdot 4 \cdot 4$. In either case, the answer is 64 ($4 \cdot 4 \cdot 4 = 64$).

COMBINING MATH FUNCTIONS

An example of using more than one math instruction can be illustrated using the Pythagorean theorem which is used to find the hypotenuse of a right triangle. The theorem states that $C^2 = A^2 + B^2$. This formula, or theorem, can be transposed and rewritten $C = \sqrt{a^2 + b^2}$. Figure 15–14 shows a right triangle. Side A is 3" and side B is 4". Side C, the hypotenuse, has no measurement and is the unknown the theorem will solve.

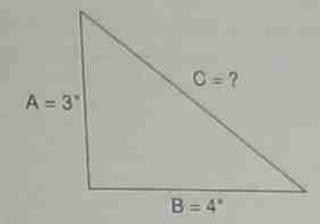


Figure 15–14 Solving the Hypotenuse of a Right Triangle Using the Pythagorean Theorem

Figure 15–15a shows how two XPY instructions, an ADD instruction, and an SQR instruction are used to find the dimension for side C of the right triangle shown in Figure 15–14. When the rung is true, the value in Source A (which is 3 from side A of the right triangle) of the first XPY instruction is increased by a factor of 2 (3^2) and stores the value of 9 in word 3 of file N7. The next XPY instruction takes the dimension of side B, which is 4, and factors it by 2 (4^2), then stores the answer (16) in word 4 of file N7. The ADD instruction now takes the values in source A and B and adds them together to get the answer of 25, as shown in word 5 of file N7. The last instruction, SQR, finds the square root of the value that is stored in word 5 of file N7. The value is 25, and the square root of 25 is 5. And, as we all remember from our high-school geometry class, if side A is equal to 3, and side B is equal to 4, then side C must be equal to 5. This is often referred to as a 3, 4, 5 triangle.

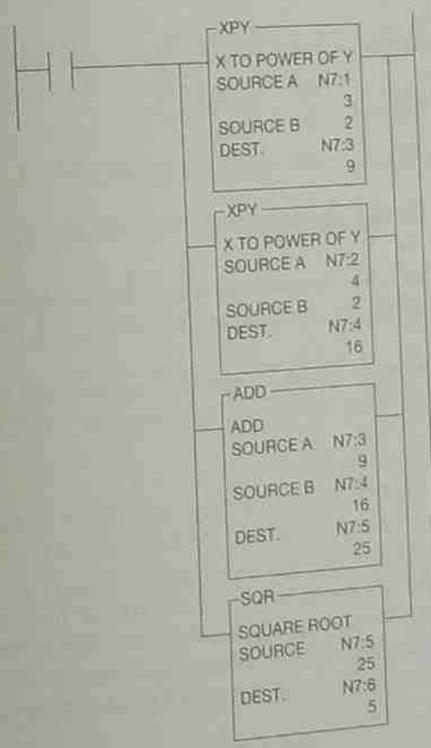


Figure 15–15a Combining Math Instructions

We could have solved the problem using a compute (CPT) instruction as shown in Figure 15–15b. With the CPT instruction, a formula, or expression is written that will solve the problem and the solution is then displayed in the destination word (N7:6).

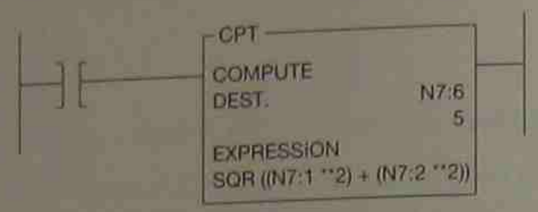


Figure 15–15b Using the CPT Instruction to Solve the Hypotenuse of a Right Triangle

GOULD 984 ARITHMETIC (MATH) INSTRUCTION

When using Modsoft® software for programming, pressing the Calculations (CALCS) designated software label key will provide the ADD, SUB, MUL, and DIV block functions. Figure 15–16 shows the format for the 984 ADD block function.

Figure 15–16 Format for Gould 984 Add Block Function

The top node is the first value (value 1), and can be a decimal number ranging from 1–999 in 16-bit CPUs or 1–9999 in 24-bit CPUs, an input register (30XXX), or a holding register (4XXXX).

The middle node is the second value (value 2), and can be a decimal number ranging from 1–999 in 16-bit CPUs or 1–9999 in 24-bit CPUs, an input register (30XXX), or a holding register (4XXXX).

The bottom node contains the symbol for the math function that is to be performed (ADD) and the address of the holding register where the results of the arithmetic will be stored (4XXXX).

The input line (Control In) controls the block. If the Control In line is true, the math function is performed on each scan of the processor, and the results are stored in the designated holding register. The overflow line, top right, passes power, or is true, if the results of the math operation is a number higher than 9999. Figure 15–13 shows how the ADD function is programmed.

In Figure 15–17, when 10001 is closed, value 1 (50) is added to value 2 (100), and the result (150) is stored in register 40160.

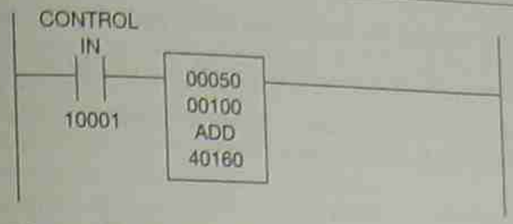


Figure 15–17 Add Function Block for Gould 984

The SUB function performs subtraction of values 1 and 2, and stores the result in a holding register. The function block holds the same information as the ADD block. The difference is the output side of the instruction. The block compares the values and identifies whether:

value 1 is > (greater than) value 2
value 1 is = (equal to) value 2
or if value 2 is > (greater than) value 1; this is the same as saying value 1 is < (less than) value 2.

Figure 15–18a shows a SUB function block, and Figure 15–18b shows how the block is programmed.

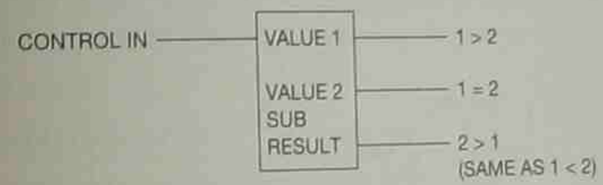


Figure 15–18a Subtract Function Block for Gould 984

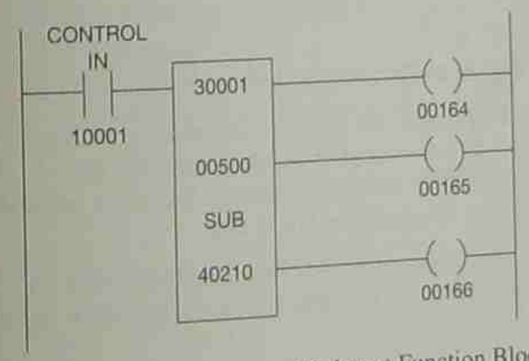


Figure 15–18b Programmed Subtract Function Block

When input device 10001 in the Control In line is true, value 2 (500) is subtracted from value 1 (input register 30001), and the result is stored in holding register 40210 on each scan of the processor. Any time input 10001 is set to 1, or *ON*, the function block makes the following comparisons:

If the value in register 30001 is greater than (>) than 500, output 00164 is *ON*.

If the value in register 3001 is equal to (=) 500, output 00165 is set to 1, or turned *ON*.

Any time the value in register 30001 is less than (<) 500, output 00166 is turned *ON*.

The MUL (multiply) function multiplies value 1 by value 2 and stores the result in a holding register. The MUL function block is shown in Figure 15–19.

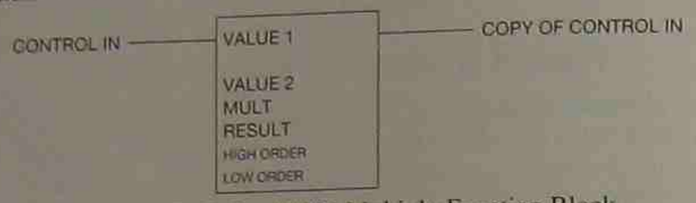


Figure 15–19 Gould 984 Multiply Function Block

In the MUL instruction, the bottom node contains the MUL symbol and a holding register (4XXXX). The results of the multiplication are stored in two consecutive registers. The higher order digits are stored in the register specified in the bottom node, and the lower order digits are stored in the next sequential register. If 40001 is specified in the bottom node, register 40002 is used to store the lower order digits. Figure 15–20 shows how a MUL function block is programmed.

Note: *When two words are used to store the results of a math instruction, it is referred to as* double precision arithmetic.

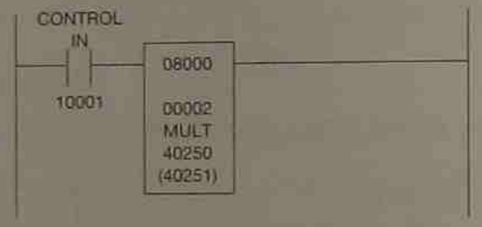


Figure 15–20 Gould 984 Multiply Function Block

When input device 10001 is closed, the block is activated and value 1 (8000) is multiplied by value 2 (2). The results (16,000) are stored in two sequential registers. In this example, 40250 contains the higher order digits (0001) and 40251 contains the lower order digits (6000). The output line (top right) is true any time the control in line is true.

The last of the arithmetic functions is the DIV, or division block. The DIV function divides value 1 by value 2 and stores the result and remainder in two consecutive holding registers (4XXXX). Figure 15–21 shows the DIV block.

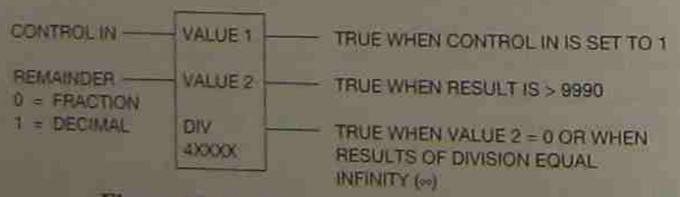


Figure 15–21 Division Block for Gould 984

The top node is value 1 and may be one of the following:

a. a decimal number ranging from 1–999 in a 16-bit CPU and 1–9999 in a 24-bit CPU
b. two sequential input registers. The first register holds the higher order numbers, and the second register holds the low order digits
c. two sequential holding registers. The first register holds the higher order numbers, while the second register holds the low order digits.

The middle node is value 2 and can be:

a. a decimal number ranging from 1–999 in a 16-bit CPU and 1–9999 in a 24-bit CPU
b. an input register (30XXX) or a holding register (4XXXX).

The bottom node contains the symbol DIV and the designated holding register. The results of the division are stored in the designated holding register, while the remainder is stored in the next sequential register.

The inputs function as follows:

Top: Control In line. When the control in line is true, the processor performs the required division on each scan;

Middle: When this line is true, the remainder is expressed in decimal form. When this line is false, the remainder is expressed as a fraction.

The outputs are as follows:

Top: True when control in line is true;

Middle: True when the result of the division is > than 9999;

Bottom: True when value 2 = 0 (zero) or if the result of the division would be infinity (∞).

Figure 15–22 shows how the DIV block is programmed.

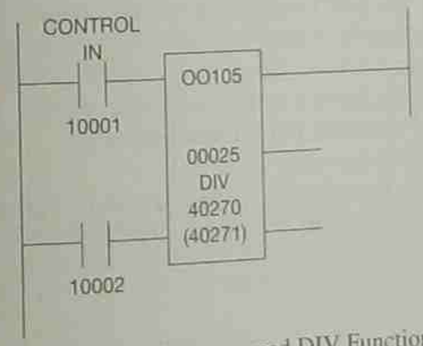


Figure 15–22 Programmed DIV Function

When 10001 is closed, value 1 (105) is divided by value 2 (25) and the results, 4.2 (4, with a remainder of 2) are stored in two sequential registers. Register 40270 contains the whole number and register 40271 contains the remainder. If 10002 is open, the remainder is fractional (5/25 in this example). If 10002 is closed, the remainder is expressed as a decimal (0.2 in this example). Register 40271 displays 20000. Although no decimal is shown, the decimal is implied because register 40271 holds the remainder.

The limits and limitations of the different arithmetic functions vary from manufacturer to manufacturer, but the concepts are basically the same. Values or contents of storage registers, data table words, or constants are combined arithmetically, and the results are stored in registers or data table words. Although the actual arithmetic function is performed using binary 2s complement, the values may be stored and/or displayed in binary, BCD, hexadecimal, or octal numbering systems, depending on the PLC. The only way to learn how to apply the arithmetic functions on a given PLC is to read the manufacturer's literature and *work* with *that* PLC.

Chapter Summary

As with most other features, arithmetic functions and formats vary with each manufacturer. Arithmetic functions of add, subtract, multiply, and divide can be combined with data manipulation instructions (data transfer and data compare) to provide expanded control and information from process or driven equipment. Memory words such as holding, storage, and data can be used with the arithmetic functions as well as words and constants, or just constants.

Review Questions

1. List the four math functions that can be performed by most PLCs.
2. T F Data manipulation instructions can be combined with arithmetic (math) instructions.
3. Define the term *double precision math*.
4. Give an example of how a math function is used in a PLC program.
5. Using Allen-Bradley format, describe the following status bits:
   S:0/0
   S:0/2
   S:0/3
6. When using the Allen-Bradley CPT instruction, what operator is used for:
   a. Multiply _____
   b. Divide _____
   C. Exponential _____
7. What is the function of the clear (CLR) instruction?
8. What does the XPY instruction do?
9. The Gould 984 uses two words to store the product of the multiply (MULT) instruction. Using two words to store math results is referred to as _____.

CHAPTER

16 Word and File Moves

Objectives

After completing this chapter, you should have the knowledge to:

- Describe the function of a synchronous shift register.
- Explain the function of *word-to-file*, *file-to-word*, and *file-to-file* instructions.
- Explain the difference between an *asynchronous shift register (FIFO)* and a *word-to-file* move.

Before word and file moves are discussed, the electrician and technician should understand the definition of both words and files.

Words, or registers as they are often referred to, are locations in memory that can be used to store different kinds of information. Typically, a word or register can store the status of inputs and outputs, hold numerical values used for math functions, other numerical data used for timers and counters, etc. Most words consist of 16 bits, although on older PLCs, an 8-bit word is sometimes used.

A **file** is a group of consecutive memory words used to store information. Words 1 through 5 would make up a consecutive 5-word file. Words 1, 2, 3, 6, and 7 are not used as a 5-word file because the numbers are not consecutive. A file is also referred to as a table by some PLC manufacturers.

WORDS

Information stored in a word can be shifted within the word, or from one word to another. Information stored in a word may also be moved into a file, or the information stored in a file can be transferred into a word. All of these different possibilities are discussed later in this chapter.

SYNCHRONOUS SHIFT REGISTER

When information is shifted—one bit at a time—within a word, or from one word to another, it is called a synchronous shift register. The bits may be shifted forward (left) or reverse (right).

Note: *The synchronous shift register may also be referred to as a serial shift register or bit shift.*

Figure 16–1 shows a 16–bit word used as a forward synchronous shift register.

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
A	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	REGISTER 100

INITIAL CONDITION OF REGISTER

B	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	REGISTER 100

CONDITION REGISTER AFTER SHIFT FORWARD (LEFT)

Figure 16–1 Forward 16-Bit Synchronous Shift Register

Figure 16–1 (a) shows the bit status of register word 100 prior to the forward shift, while 16–1 (b) shows how the register looks after the bits have been shifted one place to the left, or forward.

Notice that when the register is shifted, the information (1 or 0) in bit 16 is shifted out, and is lost. If the register is continually shifted with a zero (0) in bit location 1, all of the 1s are shifted left (forward) until only 0s remain (Figure 16–2).

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	0
1	1	0	0	1	1	1	0	0	1	1	1	0	0	0	0
1	0	0	1	1	1	0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0
1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16–2 Register With All 1s Shifted Out

In a forward shift register, bit 1 is used to enter data (1 or 0). The data is then shifted forward, one bit at a time. Figure 16–3 shows a 2-word forward shift register. In this case data is entered at bit 1 and shifted one bit at a time to the left. With a 2-word shift register, the information (1 or 0) in bit 16 of word 1 is not shifted out and lost, but is shifted into bit 1 of the second word of the shift register.

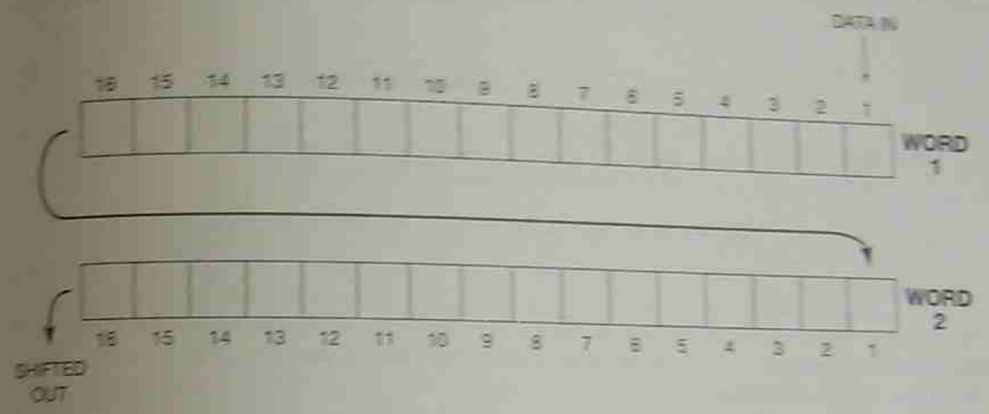


Figure 16–3 Two-Word Forward Shift Register

ALLEN-BRADLEY PLC-5, SLC 500, AND MICROLOGIX BIT SHIFT INSTRUCTIONS

Allen-Bradley has synchronous shift register instructions. They are Bit Shift Left (BSL) and Bit Shift Right (BSR). Figure 16–4 shows a BSL instruction.

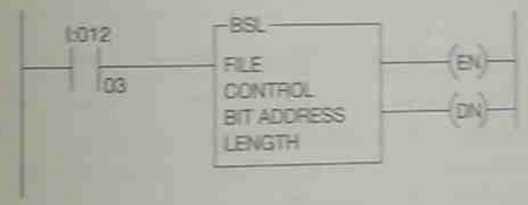


Figure 16–4 Bit Shift Left (BSL) Instruction

The BSL instruction shifts all bits to the left with each false-to-true transition. The file information that is entered is the address of the word or words that are to be shifted. This address must be a binary file address and the address must be preceeded with the file indicator symbol (#). The file indicator symbol indicates that the address is a user-defined file. The file is referred to as a bit array. The length of the file—bit array—must be entered when the instruction is being programmed. The bit array does not need to be in full 16-bit sections. A bit array length of 20 would use two memory words. All 16 bits of the first word would be used, but only 4 bits of the second word would be used. Figure 16–5 shows a two-word 20-bit array. Note than any unused bits in the file are invalidated addresses and cannot be used for any other programming. Figure 16–6 shows the allowable bit-array length for each type of Allen-Bradley PLC.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
—	—	—	I	N	V	A	L	I	D	—	—	19	18	17	16	WORD 2

Figure 16–5 Two-Word 20-Bit Array

ALLEN-BRADLEY PLC	MAXIMUM BIT ARRAY
PLC-5	15,999
SLC 500	2,048
MICROLOGIX 1000	1,680

Figure 16–6 Allowable Length of the Bit Array

The Control shown on the instruction in Figure 16–4 is the element that stores the status of the BSL instruction and the size of the bit array. Figure 16–7 shows the three word *control* element that is used for the BSL instruction.

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
WORD 0	EN		DN		ER	UL										
WORD 1	—	SIZE OF BIT ARRAY (NUMBER OF BITS)														—
WORD 2	R	E	S	E	R	V	E	D	—	—	—	—	—	—	—	—

Figure 16–7 BSL Three-Word Control Element

Word 0 of the control element holds the status bits for the instruction.

Unload Bit UL (bit 10) stores the status of the last bit in the array that is shifted out, or exits, from the array on each false-to-true transition.

Error Bit ER (bit 11) is set to 1 when an error in programming the BSL instruction is detected. This bit would be set if a negative number is entered for the length or position of the array.

Done Bit DN (bit 13) is set to 1 each time the bit array is shifted to the left.

Enable Bit EN (bit 15) is set to 1 on each false-to-true transition and indicates that the instruction is enabled.

When the array is shifted and the rung condition goes false, the enable, done, and error bits are reset to 0.

The Bit Address portion of the BSL instruction is the address of the bit whose information will be shifted into the bit array. The information will be entered into the lowest bit position of the array.

The Length portion of the instruction determines the length of the bit array.

Figure 16–8a shows a one-word (B3:0), 16-bit array, with the input bit address of I:012/00. The status of bit 00 of input word 012 will be inserted into the first bit of the array which is bit 00 when the bit array is shifted left. The information in bit 15 (position 16 of the array) will be shifted out and into the unload bit (UL) of the control element. Figure 16–8b shows the 16-bit array after the BSL instruction has made a false-to-true transition and the information has been shifted left.

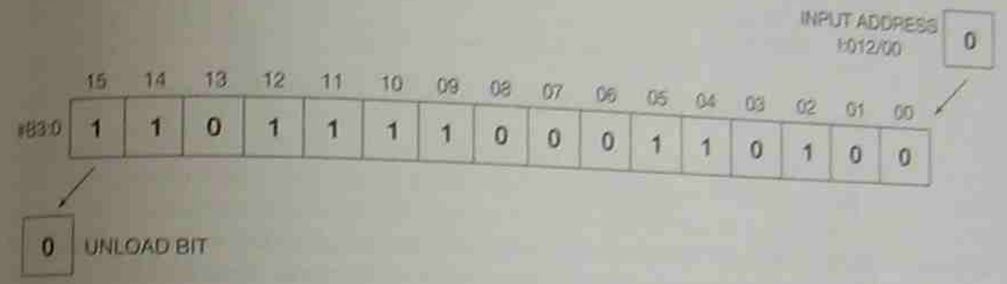


Figure 16–8a Bit-Shift Array Prior to BSL Execution

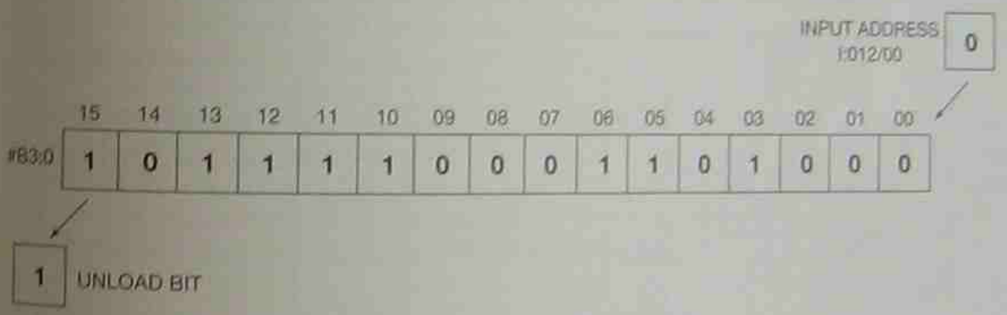


Figure 16–8b Bit-Shift Array after BSL Execution

An example of a practical application for a shift register is the overhead parts conveyor in Figure 16–9 which is used to transport parts into a paint booth for painting. If a part is on the hook as it enters the paint booth, limit switch 1 (LS-1) is activated. Limit switch 2 is activated each time a hook on the conveyor passes, even if no part is present.

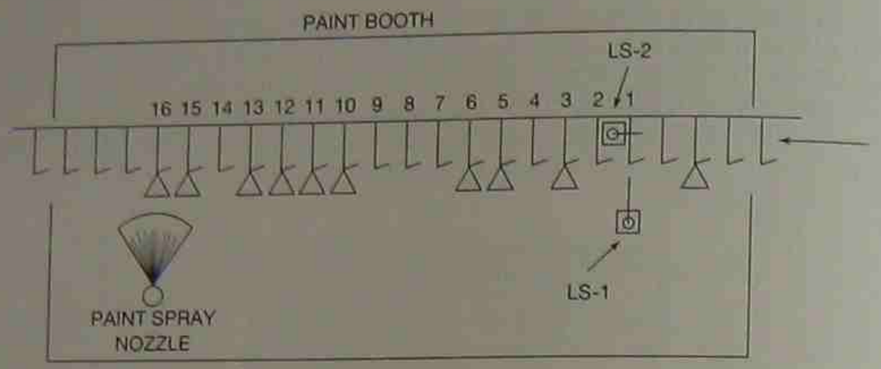


Figure 16–9 Applying a Forward Shift Register

Both limit switches (LS-1 and LS-2) are wired to an input module of a PLC, and the solenoid that operates the paint spray nozzle is wired to an output module, as shown in Figure 16–10.

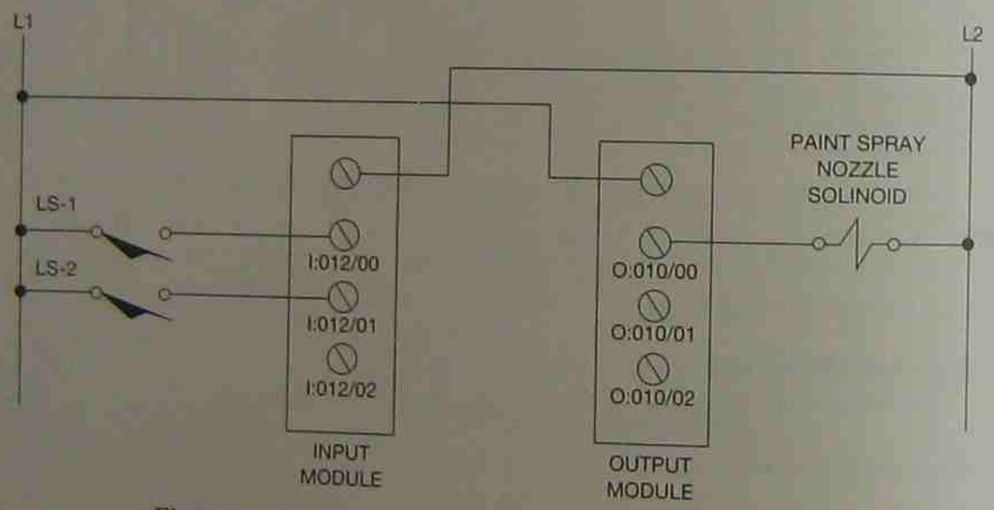


Figure 16–10 Input and Output Devices Wired to I/O Modules

The input addresses of the limit switches and the output address of the paint spray nozzle is now programmed with a forward shift register (Figure 16–11).

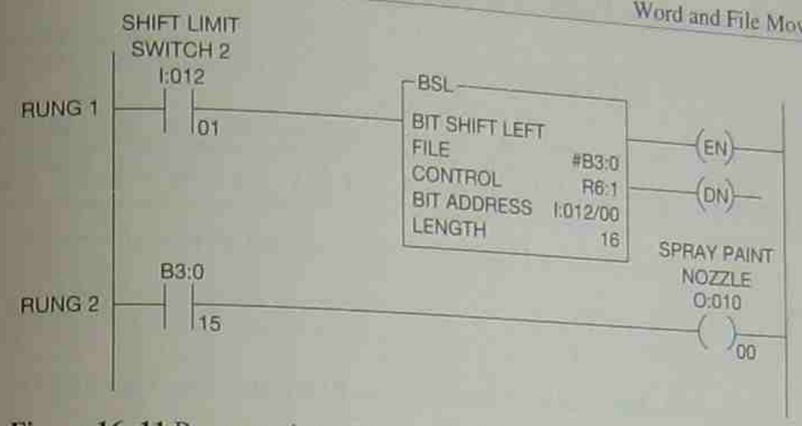


Figure 16–11 Programming a Forward Shift Register Using BSL Instruction

When a part activates LS-1 (address I:012/00), a 1 is placed in bit 00 of input word 012. As the part moves toward the spray nozzle, LS-2 is activated by the hook which closes LS-2. The closing of the LS-2 contacts (address I:012/01) gives a false-to-true transition for the BSL instruction and the instruction loads the status (1 or 0) of bit 00 of input word 012 into bit 00 of the bit array. The other information in the bit array is shifted left. The information in bit 15 of the array, the last bit of the 16-bit array, is shifted out and into the UL bit of the control element. As the conveyor continues to run, a 1 or 0 is entered into the bit 00 of input word 012, depending on whether a part is present or not. The data is then shifted as LS-2 is activated and deactivated by the moving hooks which causes a succession of false-to-true transitions of the BSL instruction. As the data shifts to the left, the paint spray nozzle solenoid (O:010/00) in Rung 2 is activated each time a 1 is shifted into bit 15 of the bit array. Bit 15, which is the 16th bit of the array, is equivalent to location 16 in the spray paint booth.

A reverse shift register is programmed using a Bit Shift Right (BSR) instruction. This instruction is the same as the BSL instruction except that instead of entering data at the lowest numbered bit in the bit array and unloading data from the highest numbered bit, this instruction enters (loads) data into the highest numbered bit and unloads at the lowest numbered bit. Figure 16–12a and 16–12b shows the load and unload order for a BSR instruction.

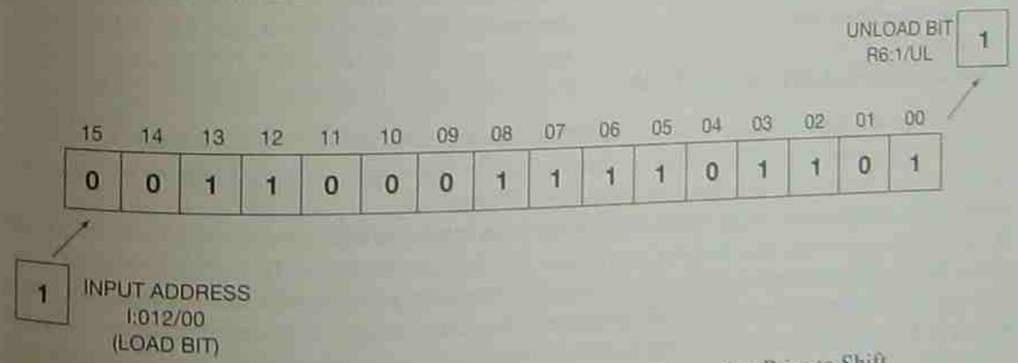


Figure 16–12a Load and Unload Bits for BSR Instruction Prior to Shift

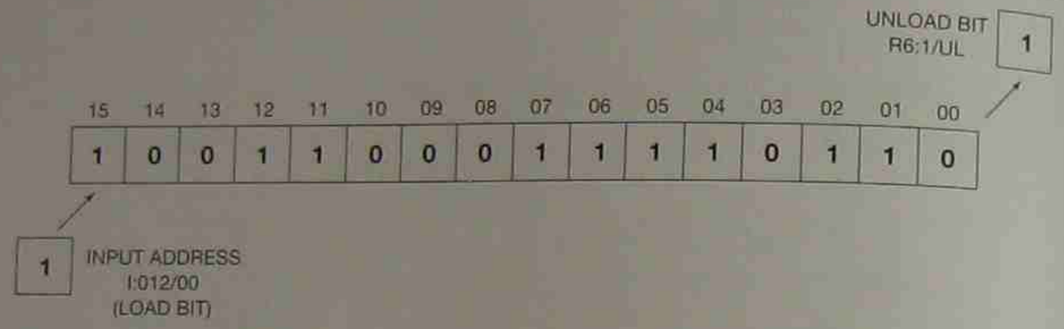


Figure 16–12b Load and Unload Bits for BSR Instruction After Shift

Similar to other functions, shift register formats vary from manufacturer to manufacturer, but the basic function of the synchronous shift register is the same.

FILE MOVES

As indicated earlier, a file, or table, is a group of *consecutive* words used to store or hold information. A file can consist of just a few words or can be several hundred words in length, depending on the PLC program. Figure 16–13 shows a five-word file using consecutive memory words 50 through 54.

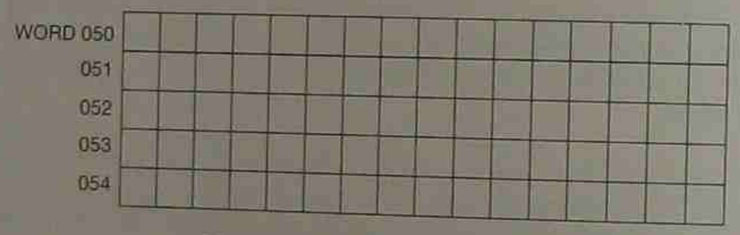


Figure 16–13 Five-Word File

Information (data) may be transferred into or out of a file by using data transfer instructions. The three most common data transfer instructions are word-to-file, file-to-word, and file-to-file.

WORD-TO-FILE INSTRUCTION

The word-to-file instruction is used to transfer data from a word into a file. For example, word 110 stores the temperature of the die for a plastic injection molding machine. A thermocouple is attached to the heated die and then connected to a thermocouple input module. Depending on the module, the temperature of the die is then stored in an input word in either binary or BCD format. By using a word-to-file data transfer instruction, the data (temperature) in word 110 can be transferred into a file. Once the word-to-file instruction has been programmed, the information stored in word 110 is transferred into a file each time the instruction is implemented. Figure 16–14a shows a 5-word file prior to a word-to-file instruction being implemented, and Figure 16–14b shows the file after the data transfer instruction is implemented.

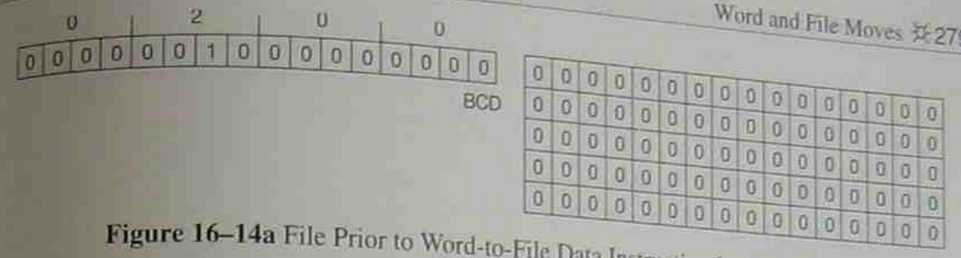


Figure 16–14a File Prior to Word-to-File Data Instruction Implementation

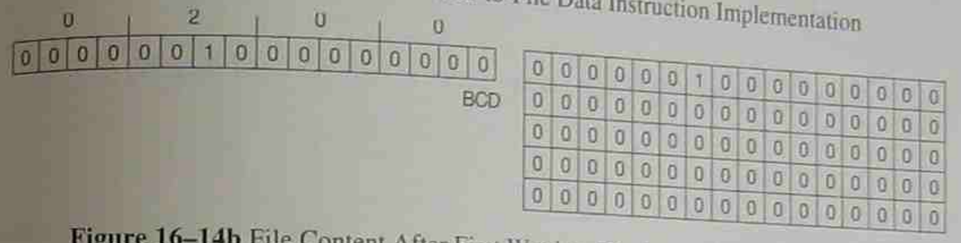


Figure 16–14b File Content After First Word-to-File Instruction is Implemented

The next time the word-to-file instruction is implemented (indexed), the current value in word 110 is transferred to the file. All information currently in the file is moved down to make room for the new data. Figure 16–15 illustrates this point.

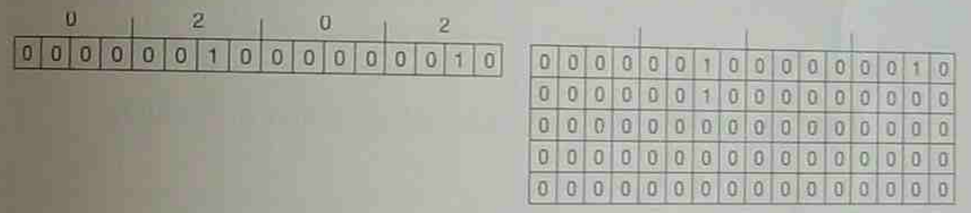


Figure 16–15 File After Second Word-to-File Instruction is Implemented

By using a timer, the word-to-file instruction could be implemented every 15 minutes. By increasing the size of the file, a record of the die temperature for an 8-hour shift can be stored, the data (temperature) from the file could be printed out, and the temperature of the die compared to quality-control records. The application of this instruction, like all other instructions, is limited only by imagination.

FILE-TO-WORD INSTRUCTION

The file-to-word instruction transfers data from a file into a word.

Using the previous example, the temperature of the injection molding machine die can be transferred to an output word (011) that controls a LED display. By incrementing or indexing the file-to-word instruction, the temperature of the die in 15 minute intervals is displayed. Figures 16–16a and 16–16b illustrate how a file-to-word instruction functions.

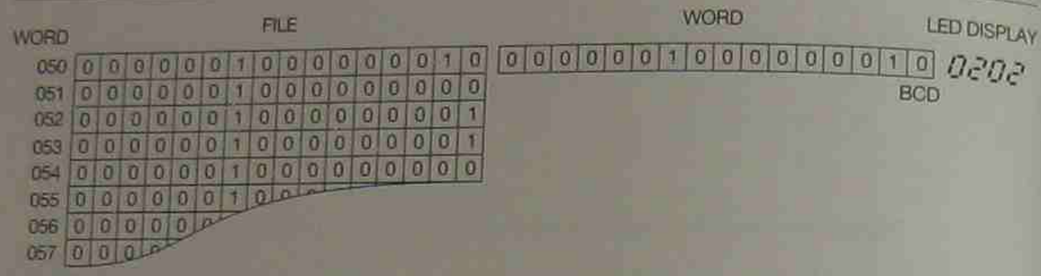


Figure 16–16a File-to-Word to Instruction at First Word of File (050)

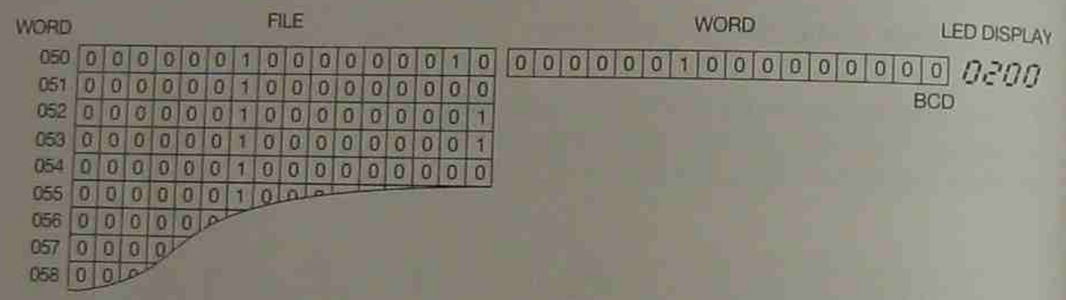


Figure 16–16B File-to-Word to Instruction at Second Word in File (051)

FILE-TO-FILE INSTRUCTION

This instruction moves data from one file to another. The data from the source file may be moved to the destination file one word at a time, or the entire contents of the file can be moved in one move, depending on the PLC.

An example of using the file-to-file move might be a chemical batch plant where different amounts and types of chemicals are mixed for a variety of products. The different mix ratios (recipes) are stored in different files, and could be transferred to a file that controls machine and/or plant operation for a given product.

Figure 16–17 shows how a file-to-file move works.

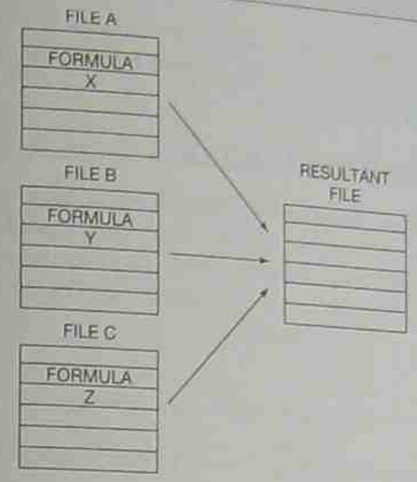


Figure 16–17 Data in File A, B, or C can be Transferred into the Resultant File When the File-to-File Move is Executed

ALLEN-BRADLEY PLC-5, SLC 500, AND MICROLOGIX FILE COPY INSTRUCTION

Allen-Bradley uses a file copy (COP) instruction for making file-to-file moves. File copy is an output instruction, and is programmed as shown in Figure 16–18.

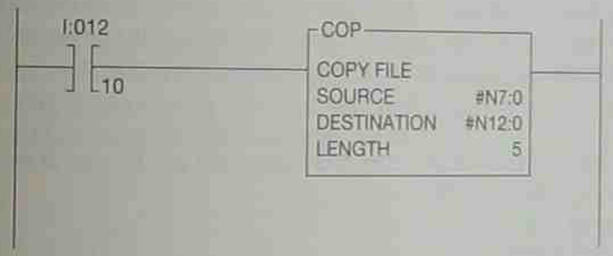


Figure 16–18 PLC-5 File Copy (COP) Instruction

The address of the first word in the source file is specified as well as the first address of the destination file. As both of these files are to be user-defined, the file numbers must start with the file indicator symbol (#). The length of the file is then specified. In this example, the source file starts with N7:0, the destination file is N12:0, and the file length is 5 words. When the input device is closed, the COP instruction is enabled, and the instruction copies data from the 5-word file starting at N7:0 into the 5 words of the destination file starting at N12:0.

When using the COP instruction, it is important to note that if the destination is a file of words, such as an integer file, the programmer must specify the file length in words. If, however, the source file is a timer or counter address, the table length is specified for the number of timers and/or counters, not words. Remember that each timer and/or counter instruction requires three

words. The destination file automatically adjusts in length when the source file is a timer or counter address. For example, if the source file contains 5 timer and/or counter addresses, the destination table is automatically set to 15 words.

GOULD DATA TRANSFER INSTRUCTIONS

The Gould 984 PLC uses a DX (data transfer) function for making word-to-file, file-to-word, and file-to-file moves. Gould uses the term *register* rather than word, and *table* rather than file. Therefore, their DX instruction makes register-to-table, table-to-register, and table-to-table moves. Figure 16–19 shows the format for the Gould 984 data transfer instruction.

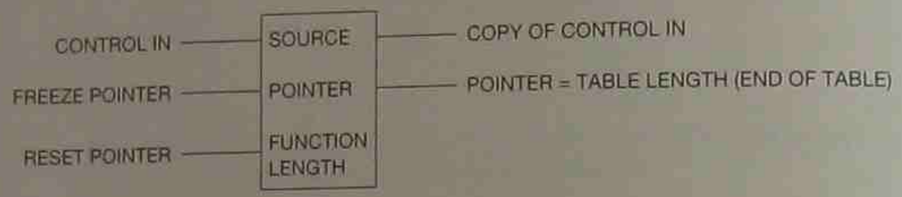


Figure 16–19 Gould 984 Data Transfer (DX) Instruction

The Control In line is used to activate the instruction. The middle line on the left is the Freeze Pointer line, and the bottom line on the left is the Reset Pointer line. The top line to the right is the Copy of Control In line; this line is true anytime that the Control In line is true. This line can be used for cascading instructions. The middle line on the right side is the Pointer line, and keeps tract of the number of data moves. This line is true when the data moves are equal to the data table length.

Within the function block, the top node is the data source and can be one of the following:
- The first of 16 logic coils (OXXXX)
- The first of 16 inputs (1XXXX)
- An input register (30XXX)
- A holding register (4XXXX)

The middle node must be a holding register (4XXXX). This register is the pointer to the destination table which must be consecutive with this word. If the pointer register is 40300, the data table must start with register 40301.

The bottom node contains the symbol that indicates the type of instruction it is, and a number to indicate the length of the table. The length of the data table can range from 1–255 16-bit words on some models, and 1–999 on other 984 models.

Figure 16–20 illustrates how the DX instruction works and is programmed. The bottom node contains the symbol R→T, and indicates that the DX instruction is programmed to perform a register-to-table move.

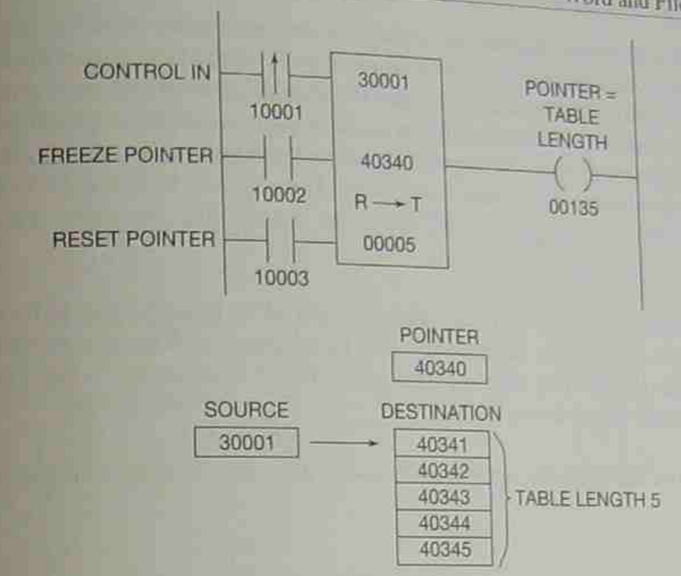


Figure 16–20 Data Transfer (DX) Instruction for Register-to-Table Moves

The contact in the Control In line is called a positive transition contact. When this instruction is programmed, the contact is only true for one scan when the input device makes an *OFF* to *ON* transition. This is similar to the Allen-Bradley PLC-5 one shot instruction. On the next *OFF* to *ON* transition, the instruction is again true, but only for one scan. Figure 16–21a shows the logic for the positive transitional contact. There is also a negative transition contact that is true for one scan on each *ON* to *OFF* transition of the input device. Figure 16–21b shows the negative transitional contact and the logic based on the state of the input.

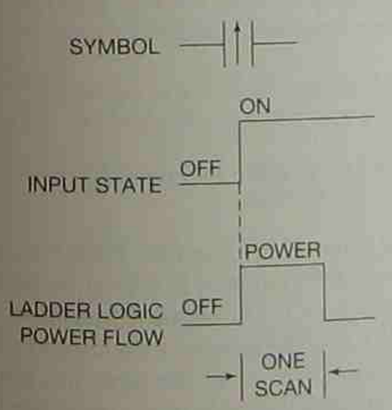


Figure 16–21a Gould Positive Transitional Contact Logic

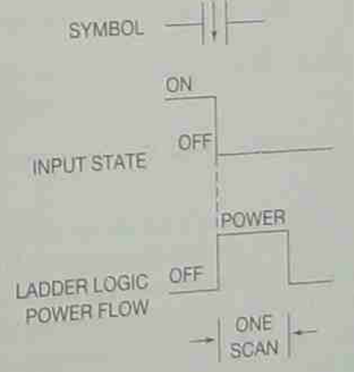


Figure 16–21b Gould Negative Transitional Contact Logic

With the first transition of input device 10001, the data in register 30001 is copied into the first word of the table (40341), and the pointer in pointer register 40340 is incremented to 1. On the second transition of 10001, the data in register 30001 is transferred into table word 40342 and the pointer in register 40340 is incremented to 2. With each successive transition of input device 10001, the data in register 30001 is sequentially moved into table words 40343, 40344, and 40345. Each transition also causes the pointer to increment from 2, to 3, to 4, and to 5. When the pointer value is equal to the table length (5 = 5), the center pointer line goes true, and output 00135 is turned *ON*.

It is possible to freeze the pointer at any location in the table. If the pointer is frozen at 3, the data from register 30001 is placed in table word 40343 (third word) with each transition of input device 10001. The pointer is frozen by closing input device 10002. The pointer is reset to 0 by activating the Reset Pointer line (input device 10003).

The 984 DX instruction also performs a table-to-register move (file-to-word) as shown in Figure 16–22.

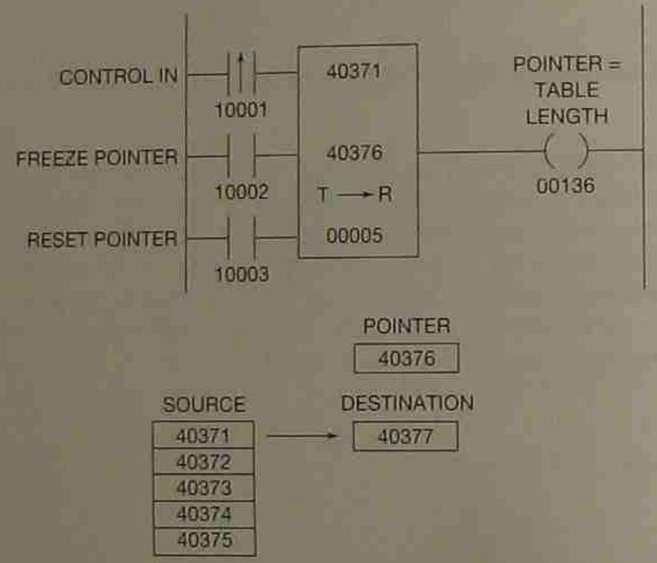


Figure 16–22 984 DX Instruction Programmed for a Table-to-Register Move

On each transition of input device 10001, data is moved sequentially from the table into the register (40377). On the first transition, the data in table word 40371 (source) is copied, or moved, into register 40377 (destination), and the pointer is incremented to 1. On the next transition, the data from table word 40372 is transferred into register 40377 and the pointer incremented to 2. This process continues with each transition of 10001 until the last word in the table has been transferred into register word 40377. At this time, the pointer equals the table size, and output 00136 is turned *ON*. The pointer can be frozen at any point, controlling the data that is transferred into register 40377.

The DX instruction makes table-to-table moves in two ways. One table-to-table instruction moves one word at a time from the source table to the destination table. A second instruction transfers data from all words in the source table to the destination table in one move. Gould calls this latter instruction a block move. Figure 16–23 shows how the table-to-table move (T→T) transfers, or copies, the bit status of one word in the source table to the corresponding word in the destination file.

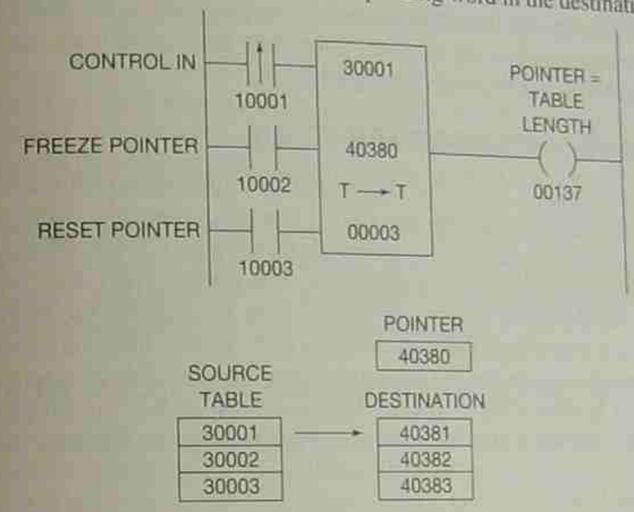


Figure 16–23 Table-to-Table Instruction for Gould 984

With the first transition of input device 10001, the instruction copies the bit status of 30001 into 40381. The pointer (40380) is incremented to 1. With each transition of 10001, data is transferred from subsequent source table words to the corresponding destination word. Although the table only consists of three words in the example, the table length could be up to 999 words, depending on the model of processor. As in previous DX instructions, if input device 10002 is turned ON, the pointer is frozen and the table-to-table move stops at whatever position it is when the pointer is frozen. To reinitiate the T→T function, the pointer is reset by closing and then opening input 10003. Once the pointer is reset, transitions of 1001 again cause data to be transferred from the source table to the destination table.

The Gould block move (BLKM) instruction moves all the data in one file to another file in one move. Figure 16–24 shows how the block move is programmed.

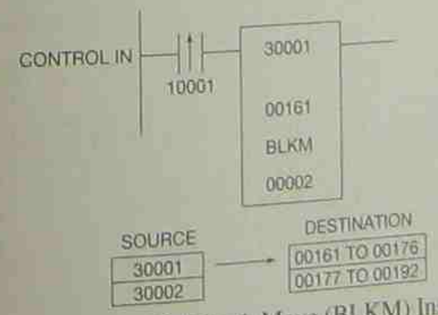


Figure 16–24 Gould 984 Block Move (BLKM) Instruction

The top node is the source table and displays the first word of the source table (30001). The center node is the destination table and displays the first word in the destination table. In this example, the destination table is a table of outputs (16 bits, 00161–00176). On each transition of input 10001, the contents of the source file is transferred into the destination file. This instruction does not have a pointer register because the data is not transferred a word at a time, but is done completely in one move. Although this example uses an output address for the destination table, the table could have consisted of holding registers. Several BLKM instructions can be used to transfer various formulas (data) as was illustrated in Figure 16–17.

ASYNCHRONOUS SHIFT REGISTER (FIFO)

The asynchronous shift register, instead of shifting bits of information within a word, or words, like the synchronous shift register, shifts the data from a complete word into a file, or stack. Although this appears to be just another name for a word-to-file instruction, it is not. There are similarities between the two, but there is also one major difference. In a word-to-file move, the information from the word is shifted into the top of the file and moved down through the file with each implementation, or indexing, of the instruction. The asynchronous shift register, however, allows the information transferred from a word to go to the last *unused* word of the file. This difference is why the asynchronous shift register is often referred to as a FIFO stack (first in–first out). Figure 16–25 compares the asynchronous shift register to the word-to-file instruction to demonstrate the difference.

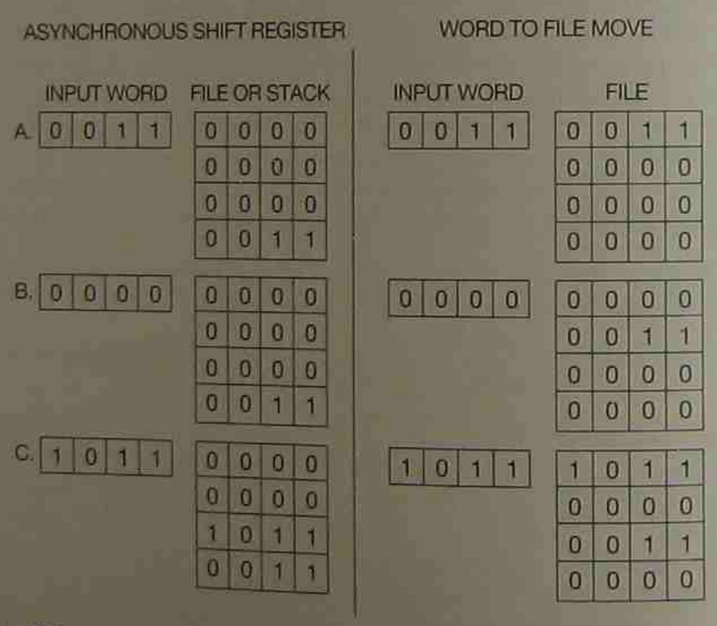


Figure 16–25 Comparison of Asynchronous Shift Register (FIFO) to Word-to-File Move

When data is transferred at position (a), the asynchronous shift register places the data in the last (bottom) unused word of the file or stack. In the word-to-file move, however, data is transferred into the first (top) word of the file.

At the next implementation at position (b), no data is present in the input word, and no change takes place in the FIFO stack, but the previous data is moved *down* one word in the word-to-file instruction.

When the instructions are indexed again at position (c), the data of the input word is transferred to the next available word at the bottom of the FIFO stack, whereas the data is entered at the top of the file and all previous data is shifted *down* in the word-to-file instruction.

The biggest difference between the two instructions is the ability of the word-to-file move to accept and store all zeros, whereas the asynchronous shift register (FIFO) ignores input data of all zeros. The FIFO stack is well suited when the programmer is only interested in data, and not concerned with periods where no data (all zeros) is transferred.

ALLEN-BRADLEY PLC-5, SLC 500, AND MICROLOGIX FIFO INSTRUCTION

Allen-Bradley uses a pair of output instructions to store and retrieve data in a prescribed order, or FIFO (First In–First Out). Words are loaded into a file and unloaded in the same order in which they were loaded. Figure 16–26 shows the First In–First Out Load (FFL) and First In–First Out Unload (FFU) Instructions.

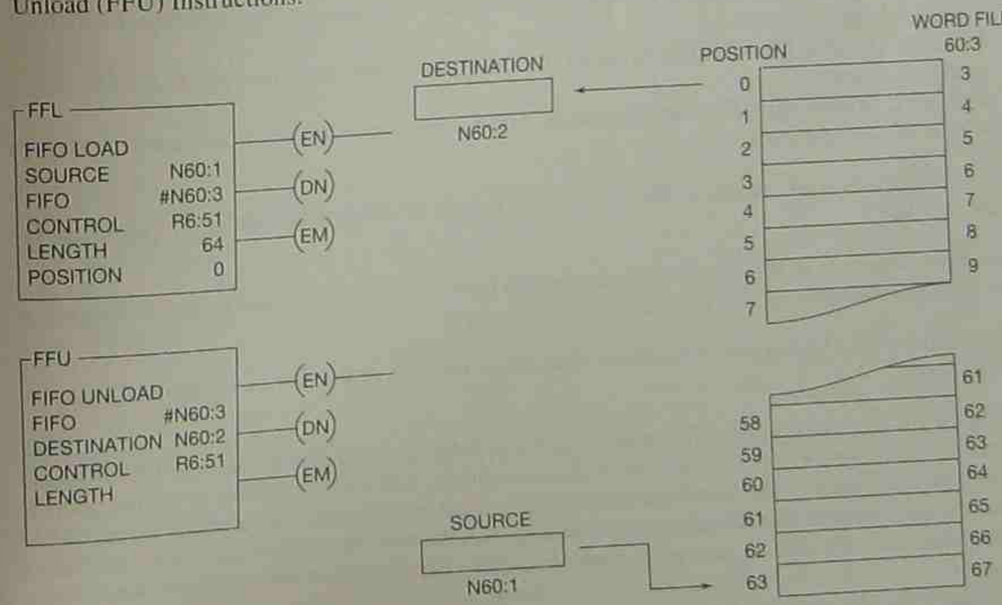


Figure 16–26 FIFO Load and Unload Instructions

The FIFO Load instruction loads the information stored in one word into a file, or stack. The FIFO Unload instruction retrieves information stored in the file, or stack, and places it into a destination word. The instruction components are as follows:

The Source is the address that stores the "data" that will be loaded into the file, or stack.

FIFO is the address of the first word in the file, or stack.

Control is the address of the control structure (48 bits—three 16–bit words) in the control area (R) of memory. The control structure stores the instruction's status bits, stack length, and next available position (pointer) in the stack as shown in Figure 16–27.

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
WORD 0	EN	EU	DN	EM												
WORD 1					STACK (FILE) LENGTH											
WORD 2					POSITION NUMBER											

Figure 16–27 FFL-Three Word Control Element

Empty Bit—EM (bit 12) is set to 1 when the stack, or file, is empty.
Done Bit—DN (bit 13) is set to 1 when the stack is full. This prevents any additional data from being loaded into the stack.
FFU/LFU Enable Bit—EU (bit 14) is set to 1 on a false-to-true transition and is reset to 0 on a true-to-false transition.
FFL/LFL Enable Bit—EN (bit 15) is set to 1 on a false-to-true transition and is reset to 0 on a true-to-false transition.

The Length portion of the instruction defines the length, or number of words, that make up the stack, or file.

The Position portion of the instruction indicates the position in the file, or stack, in which information from the source word will be entered.

Destination (FIFO Unload) is the address that stores the data that exits, or is removed from the file, or stack.

In Figure 16–26, note that the FIFO address is #N60:3 for both the FIFO Load and FIFO Unload instructions. Address #N60:3 is the start of the 64-word file and address N60:67 is the last address for the file. Position 0 is the first word of the file, or stack and Position 63 would be the last word of the 64-word file. With each false-to-true transition of the FFL instruction, data from the Source is loaded into the file (FIFO) and the position indicator will advance, or increment, by one.

When the rung that contains the FFL instruction goes true, the processor sets the EN bit (bit 15) *ON*, and loads the source data (N60:1) into the next available position in the stack. The processor

loads data from the source into the stack with each false-to-true transition. When the stack is full, the processor sets the DN bit (bit 13) to 1. The program should be programmed so that when the stack is full, no additional data can be loaded from the source.

When the rung that contains the FFU instruction goes from false to true, the processor sets the EU (enable unload bit 14) to 1 and unloads data from the first element of the stack into the destination word N60:2. As the data is shifted out, the processor shifts the remaining data in the stack *up* one position toward the first word. The processor continues to unload the stack each time the rung goes from false to true until it empties the FIFO stack.

LAST-IN FIRST-OUT (LIFO) INSTRUCTIONS

The Last-In First-Out Load (LFL) instruction loads words into a file. Then, using a LFU (LIFO Unload) instruction, retrieves them in inverse order. In other words, the instruction removes the last word that was entered into the file when the rung that has the LFU instruction makes a false-to-true transition. Figure 16–28 shows a LFL and a LFU instruction and illustrates how the words are entered and retrieved.

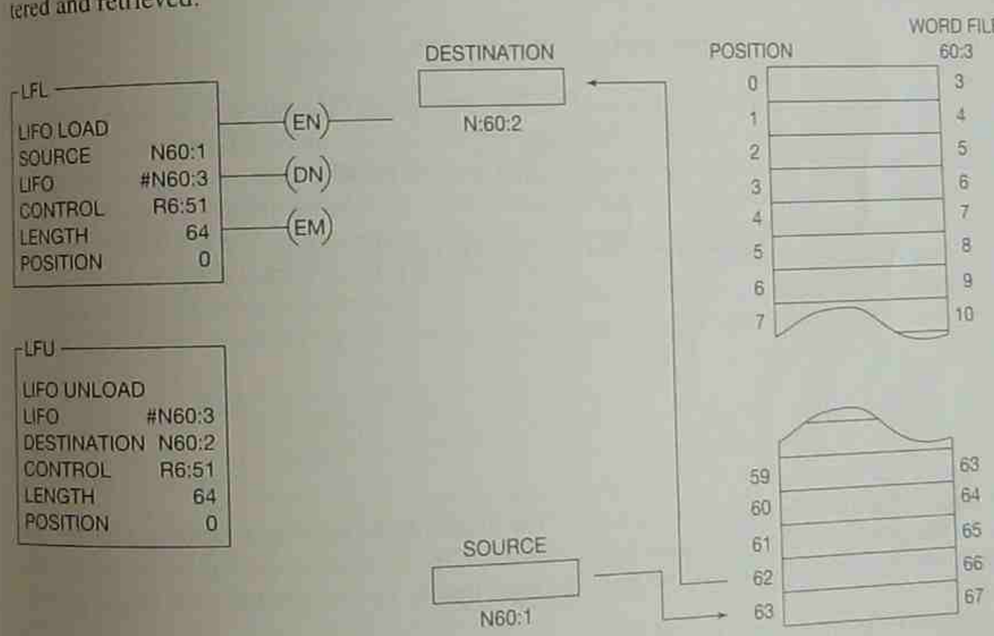


Figure 16–28 LIFO Load and Unload Instructions

When the rung condition makes a false-to-true transition, the LFL EN (enable bit) will be set to 1, LFL instruction will load the contents (data) from the source word N60:1 into the stack, and the position value will be incremented by 1. The LFL instruction will continue to load information into the file, or stack, on each false-to-true transition until the stack is full (64 words). When the stack is full, the processor will set the DN bit (bit 13) to 1. This prevents any additional information from being loaded into the file.

When the LFU rung makes a false-to-true transition, the LFU EU (unload bit) is set to 1 and the LFU instruction will unload the last word that was loaded into the file and place it into the destination word (N60:2). The file words will continue to be unloaded from the file each time the rung makes a false-to-true transition. Each time a word is removed from the file, or stack, the position indicator will decrement (decrease in value) by 1.

GOULD FIRST IN FUNCTION BLOCK

The Gould 984 also uses two instructions for loading and unloading a first in–first out file, stack, table, or queue. Queue means to line up, or wait in line, and is used by Gould in its program manual. Figure 16–29 shows the Gould 984 first in (FIN) instruction.

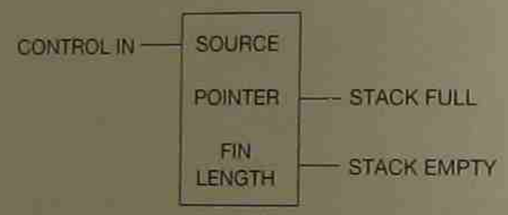


Figure 16–29 984 First In (FIN) Function Block

The top node is the address of the data source. The middle node *must* be a holding register, and acts as the pointer to the destination table, or stack. The bottom node contains the symbol (mnemonic) for First In (FIN), and also holds the stack length. The maximum stack length is 100.

The Control In line moves data from the Source into the destination stack on each scan that the instruction is true. A transitional contact is used if single operation is desired. The transitional contact data is only transferred on the first scan on each *OFF* to *ON* transition.

The middle line on the right is the Stack Full line, and is true when the stack is *full*. The bottom line is the Stack Empty line, and is only true when the stack is *empty*. Once data is moved into the stack, this line is set to 0.

Figure 16–30 shows how the FIN instruction operates.

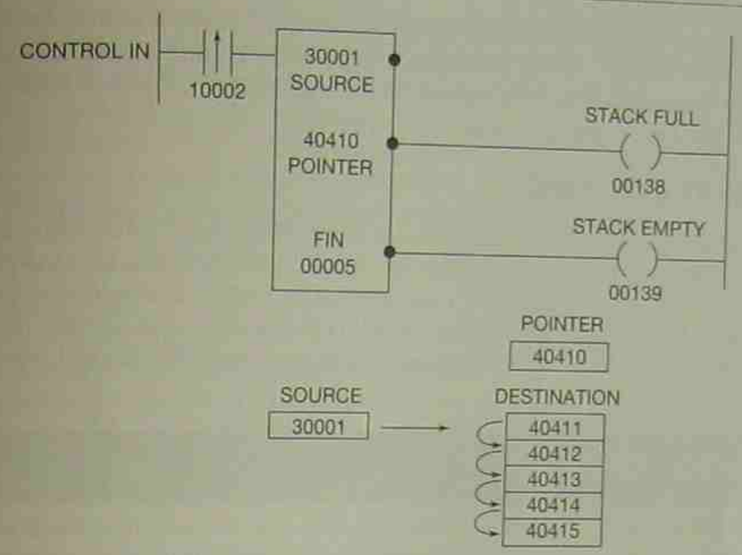


Figure 16–30 Gould FIN (First In) Operation

When the positive transitional contact makes a false-to-true transition, the data in the source word (30001) is transferred into word 40411 in the stack and the pointer is incremented to 1. On the next false-to-true transition, the data in word 40411 is shifted down to word 40412, and the data in word 30001 is copied into word 40411. On each subsequent transition of input device 10001, data from word 30001 is transferred into word 40411, and all previous data is shifted down. The transfer continues on each transition until the stack is full. The first out (FOUT) function is used to unload the stack. As the name implies, the data that was first entered into the stack, or queue, is retrieved first. Figure 16–31 shows how the FOUT instruction operates.

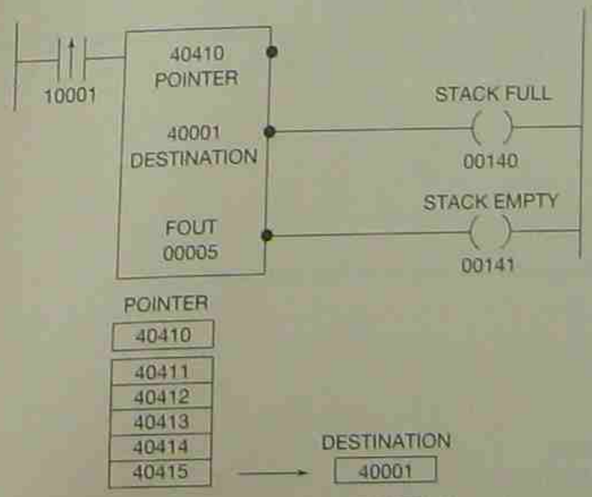


Figure 16–31 984 First Out (FOUT) Operation

On the first transition of input device 10002, the data in word 40415 is copied into the destination register 40001. Now that the stack is no longer full, the stack full line goes false, and output 00140 is set to 0. The processor continues to unload the stack on each transition of 10002 until the stack is empty. When the stack is empty, output 00141 is set to 1.

LAST IN–FIRST OUT (LIFO)

The last in–first out (LIFO) stack is the opposite of the FIFO stack. Information is placed in the stack and then removed in inverse order (the last data in is the first data out). Allen-Bradley uses LIFO load (LFL) and LIFO unload (LFU) instructions for loading and unloading LIFO stacks, or files. The instructions are programmed like the FIFO instructions, and will not be discussed further. The various PLC manufacturers' literature can be consulted if additional information and/or clarification is needed.

Chapter Summary

Although the keystrokes and instructions vary with each PLC manufacturer, the principles of word and file moves are the same. The synchronous shift register shifts bits of information left or right (forward and reverse) within a word or words, while file moves transfer data from words to files, files to words, or files to files. The asynchronous shift register is referred to as FIFO, or first in–first out, as data transfers or falls to the bottom of the stack and uses the last unused word. The data is retrieved in the order it enters the stack (first in–first out). The file can also be programmed to retrieve the data that was last in to be the first data out. This convention is referred to as a LIFO stack.

Review Questions

1. Define the term *word* as used in this chapter.
2. Define the term *file* as used in this chapter.
3. T F The synchronous shift register shifts data in a forward direction only.
4. In a 1-word shift register, the data is entered at bit:
   a. 1
   b. 2
   c. 4
   d. 8
   e. 16
   f. none of the above
5. In a 1-word shift register, the data is shifted out at bit:
   a. 1
   b. 2
   c. 4
   d. 8
   e. 16
   f. none of the above
6. Define the term *FIFO*.
7. Define the term *LIFO*.

8. Briefly describe the difference between *synchronous* and *asynchronous shift registers*.
9. When using the PLC-5 FFL instruction, which bit is set to 1 when the stack is full?
10. When using the PLC-5 FFU instruction, which bit is set to 1 when the stack is empty?
11. List two other names that could be used for a file.
12. When using a Gould 984 PLC, what is the maximum length of a FIFO stack?
13. What is the purpose of the PLC-5 *COP* instruction?
14. Which of the following group of words could *not* be a file?
    a. 50, 51 52
    b. 50, 51, 52, 53
    c. 100, 101, 102, 103
    d. 100, 101, 102, 103, 105
15. Briefly describe the function of a word-to-file move.
16. Briefly describe the function of a file-to-word move.
17. Briefly describe the function of a file-to-file move.
18. Which instruction is also known as a FIFO?
    a. synchronous shift register
    b. word-to-file move
    c. file-to-word move
    d. asynchronous shift register
    e. file-to-file move
19. T F When data is transferred into a file using a word-to-file move, the data is entered at the last unused word of the file.
20. The control element for a BSL instruction requires how many words? What information does each word hold?
21. User defined file numbers must start with what file indicator symbol when programming an Allen-Bradley PLC-5, SLC 500, or MicroLogix PLC?

CHAPTER

17 Sequencers

Objectives

After completing this chapter, you should have the knowledge to:

- Describe what a sequencer instruction does.
- Understand the basics of sequencer operation.
- Define the term *mask*.

The sequencer instruction transfers information from memory words into output words. Sequencer instructions are typically used to control automatic assembly machines that have consistent and repeatable operations.

A programmed **sequencer** replaces the mechanical drum sequencer that was used in the past. On the mechanical sequencer, when the drum cylinder rotated, contacts opened and closed mechanically to control output devices. Figure 17–1 shows a mechanical drum cylinder with pegs placed at varying horizontal positions for step 1 of the sequence. When the cylinder rotated, contacts that aligned with the pegs closed, and contacts where no pegs existed remained open. In this example, the presence of a peg should be thought of as a 1, or *ON*, and the absence of a peg as a 0, or *OFF*.

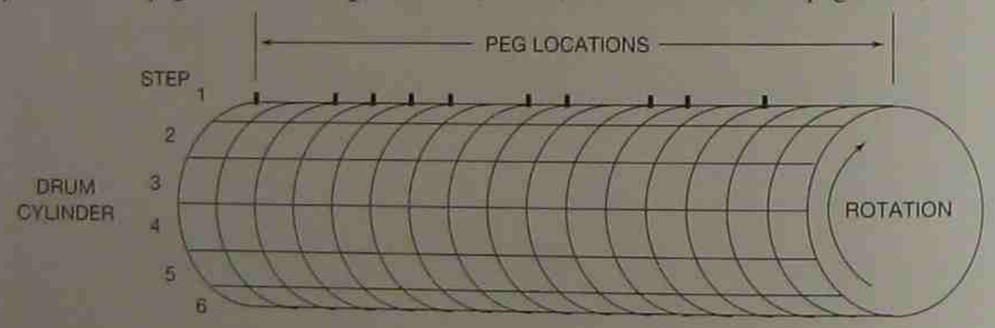


Figure 17–1 Drum Cylinder
(Courtesy of Allen-Bradley)

To program a sequencer, binary information is entered into a series of consecutive memory words. These consecutive memory words are referred to as a file. Information from the words in the file is transferred sequentially to the output word to control the outputs.

If the first six steps on the drum cylinder in Figure 17–1 are removed and flattened out, they appear as illustrated in Figure 17–2.

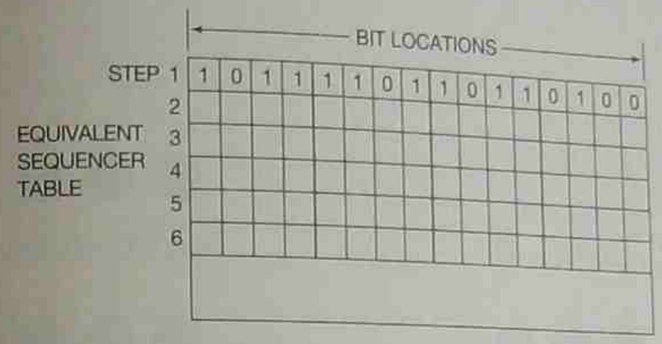


Figure 17–2 Sequencer Table

For step 1, each horizontal location where a peg is was located is now represented by a 1 (*ON*), and the positions where there were no pegs are represented by a 0 (*OFF*).

The six steps could also be viewed as a 6-word file with each 16-bit word representing a sequencer step. By entering different binary information (1s and 0s) into each word of the file, the file replaces the rotating drum cylinder.

To illustrate how this works, 16 lamps are used for outputs (as shown in Figure 17–3).

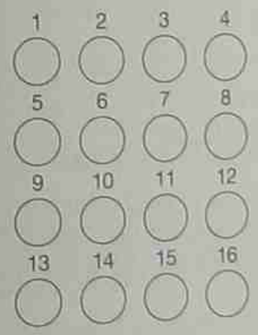


Figure 17–3 Output Lamps

Each lamp represents one bit address (1 through 16) of output word 25.

Assume, for the sake of discussion, that the operator wants to light the lamps in the 4-step sequence shown in Figures 17–4a, 17–4b, 17–4c, and 17–4d.

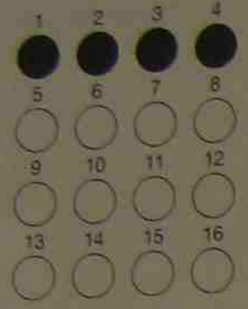


Figure 17–4a Sequence 1

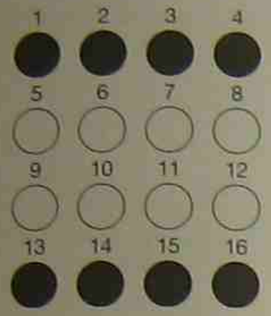


Figure 17–4b Sequence 2

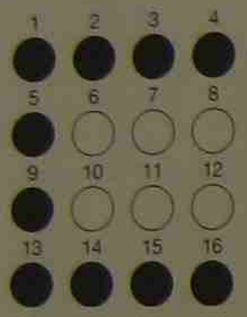


Figure 17–4c Sequence 3

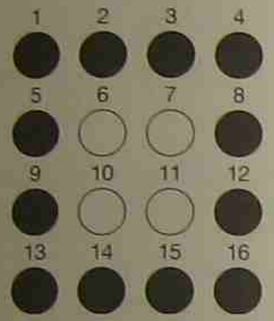


Figure 17–4d Sequence 4

The bit addresses of the lamps that are to be lit in each step of the sequence are written down to assist in entering data into a word file.

Step 1 1, 2, 3, 4
Step 2 1, 2, 3, 4—13, 14, 15, 16
Step 3 1, 2, 3, 4—5, 9—13, 14, 15, 16
Step 4 1, 2, 3, 4—5, 8, 9, 12—13, 14, 15, 16

The next step is to define a word file to store the binary data required for each step of the sequencer. Words 30, 31, 32, and 33 are used for the 4-word file. By using the programmer, binary information (1s and 0s) are entered into each word of the file to reflect the desired lamp sequence (Figure 17–5).

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
WORD 25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OUTPUT
WORD 30	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	STEP #1
WORD 31	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	#2
WORD 32	1	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	#3
WORD 33	1	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	#4

Figure 17–5 Binary Information for each Sequencer Step

Note: *Some PLCs allow the data to be entered using BCD which speeds the entry process. To use this feature, the required binary information for each sequencer step is converted to BCD. The information is then entered using a programming device into the word file with four key strokes for each word, rather than 16.*

Once the sequencer has been programmed and the data entered into the word file, the sequencer is ready to control the lamps. When the sequencer is activated and advanced to step 1, the binary information in word 30 (Figure 17–5) is transferred into word 25, and the lamps light in the pattern shown in Figure 17–4a. Advancing the sequencer to step 2 transfers the data from word 31 into word 25 for the light sequence shown in Figure 17–4b. Step 3 transfers the data from file word 32 into word 25, and step 4 transfers information from word 33 into word 25. When the last step is reached, the sequencer can be reset and sequenced again.

Depending on the PLC, sequencers can be programmed from a few steps up to hundreds of steps, and can control one output word or several.

MASKS

When a sequencer operates on an entire output word, there may be outputs associated with the word that the operator does not want controlled by the sequencer. To prevent the sequencer from controlling certain bits of an output word, a **mask** word is used. Figure 17–6 shows how a mask word works.

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
WORD 025	1	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	OUTPUT
WORD 20	1	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	MASK
WORD 30	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	FILE
WORD 31	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	
WORD 32	1	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	
WORD 33	1	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	
WORD 34	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Figure 17–6 Using a Mask Word

The mask word is a means of selectively screening out data from the sequencer word file to the output word. For each bit of output word 025 that the operator wants the sequencer to control, the corresponding bit of mask word 20 must be set to 1.

In Figures 17-4a, 17–4b, 17–4c, and 17–4d, bits 6, 7, 10, and 11 are not used. By *not* setting bits 6, 7, 10, and 11 of the mask word to 1, these bits can be used independently of the sequencer.

In Figure 17–6, a fifth step is added to the sequencer. File word 34 and bits 6, 7, 10, and 11 are set to 1. With bits 6, 7, 10, and 11 of mask word 20 set to 0, the data in file word 34 is screened out and prevented from being transferred into output word 025.

The sequencer works much like the file-to-word move discussed in Chapter 16. For programmable controllers that don't have a dedicated sequencer instruction, a file-to-word move instruction can be used.

ALLEN-BRADLEY PLC-5, SLC-500, AND MICROLOGIX 1000 SEQUENCER INSTRUCTION

The Allen-Bradley Sequencer Output Instruction (SQO) creates a sequencer file of information that is used to control various output devices. When the rung that contains the SQO instruction makes a false-to-true transition, the instruction increments to the next word in the sequencer file and loads the information into the destination word. Figure 17–7 shows an SQO instruction.

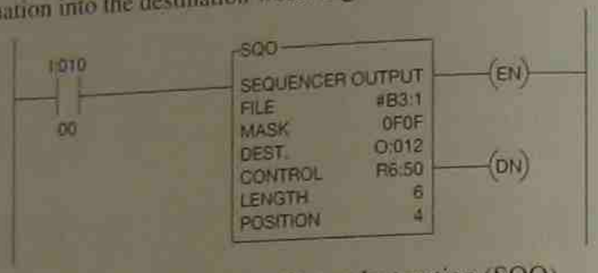


Figure 17–7 Sequencer Output Instruction (SQO)

The File portion of the instruction is the address of the sequencer file. You must use the file indicator symbol (#) for this address. In this illustration, the file has been addressed #B3:1. This address indicates that a file has been created in the Bit File portion of the processor memory starting with word 1.

The Mask, as explained earlier, is a filter through which all data from the sequencer file must pass before being placed into the destination, or output word. Allen-Bradley uses a hexadecimal number that represents the bit pattern that is desired by the mask for screening information, or data, from the sequencer file. A 1 must be placed in the mask bit location for information to be passed through.

The Destination is the address of the output word that is to be controlled by the sequencer instruction. In Figure 17–7, the destination address is O:012, or word 12 of the output image table.

The Control is the three-word element that stores the status bit for the sequencer, as well as the length of the sequencer file and the position of the sequencer. Figure 17–8 shows the three-word element for the SQO instruction.

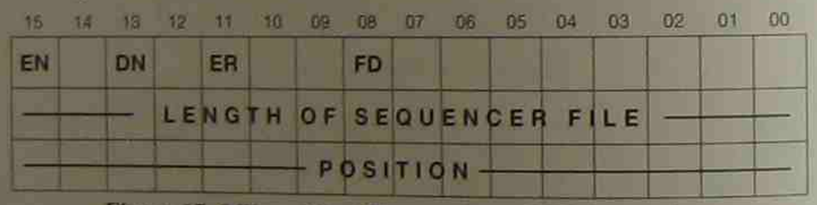


Figure 17–8 Three-Word Element for an SQO Instruction

The status bits shown in word 1 include:

Error Bit ER (bit 11) is set to 1 if the processor detects a negative *position* value or a negative or zero *length* value.

Done Bit DN (bit 13) is set to 1 when the last word of the sequencer file has been transferred into the *destination* word.

Enable Bit EN (bit 15) is set to 1, or is true, when the SQO instruction is enabled.

Found Bit (Bit 08)-SQC instruction only. When the status of all nonmasked bits in the source address match the reference word bits, bit 08 is set to 1.

The Length value in the instruction is the number of steps in the sequencer file starting with Step 1. Step 0 is the startup, or default value of the sequencer on startup. On the next false-to-true transition, after the last step of the sequencer has been loaded into the destination word, the sequencer will be reset to Step 1. The maximum length of a sequencer file is as follows:

PLC-5	1 - 1000
SLC 500	1 - 255
MicroLogix 1000	1 - 104

The Position indicates the step, or word, where the sequencer is currently positioned. The position number will increment with each false-to-true transition of the instruction.

Figure 17–9 shows a programmed SQO instruction with the sequencer file, mask word, destination word, and the status of the external output devices when the sequencer instruction is on Step 4.

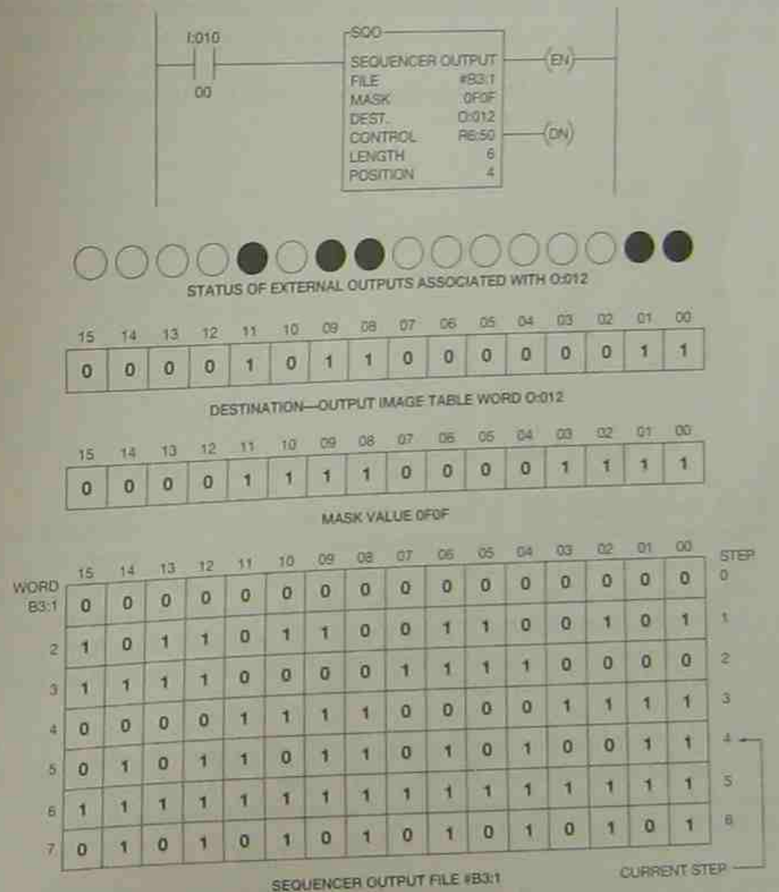


Figure 17–9 Sequencer (SQO) Instruction

As programmed in Figure 17–9, the sequencer instruction transfers data from the sequencer file each time that input device I:010/00 closes. In Figure 17–9, the sequencer is shown on Step 4. Even though the data in word B3:5 (Step 4) has a 1 set for bits 0, 1, 4, 6, 8, 9, 11, 12, and 14, the external output devices associated with output word O:012 only shows that outputs 0, 1, 8, and 11 are *ON*. The reason outputs 4, 6, 12, and 14 are not ON is because these bits are masked.

When a sequencer instruction has a mask address, data will transfer only from the sequencer file to the bits of the destination word that have a 1 in the mask address. The mask address in Figure 17–7 was hexadecimal number 0F0F. This address is the binary equivalent of 0000 1111 0000 1111.

Another Allen-Bradley sequencer instruction is the Sequencer Compare Instruction (SQC). This instruction compares all of the unmasked bit of the source word to the current step of the sequencer file. If all of the bits *match*, bit 08 of the control word is set to 1. Bit 8 is the Found Bit (FD). This instruction can be used to compare the status of a machine/equipment input devices with what is normal operation. This is a great way to do machine diagnostics. Figure 17–10 shows a SQC instruction.

Note: *The PLC-5 family of processors calls this instruction SQI for Sequencer Input. While the initials are different, the operation is the same.*

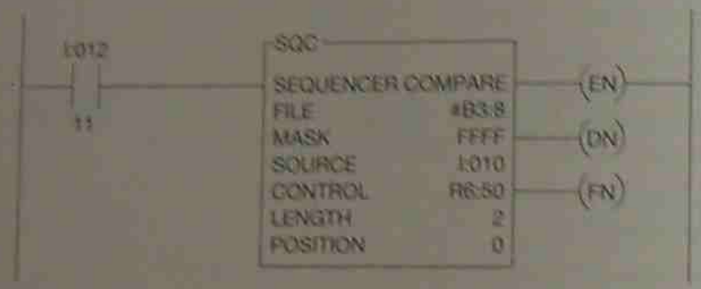


Figure 17–10 Sequencer Compare Instruction (SQC)

As programmed, the sequencer File is Bit File 3, starting with word 8 (B3:8). Because this is a user defined file, the address is preceded with the # symbol. The Mask has been set to all 1s by entering the hexadecimal number FFFF. The Source is listed as I:010, or word 10 of the input image table. This is the word that hold the status of the input devices for the process equipment. The Control, is a three-word element in the R (control) file that holds the status bits, the length of the sequencer file, and the current position (step) of the sequencer. The Length of the file is shown as 2, and current Position (step) is shown as 0.

The status bits for this instruction are the same as the SQO instruction previously discussed (Figure 17–8), with the addition of the Found Bit (FD) (bit 8). This bit will be set to 1 whenever the status of all of the unmasked bits in the source word match the status of the reference word in the sequencer file.

As an example, let us say that when a given production machine is operating normally and ready to produce Product A, the status of the input devices on the machine (*ON* or *OFF*) should be indicated by the data stored in Step 1 of the sequencer file. When input device I:012/11 is closed, making a false-to-true transition, the sequencer will increment and move from the startup position of 0 to Step 1. The bit status of input word I:010 will be compared to the status of the 16-bit word at Step 1 of the sequencer file. Step 1 is word B3:9.

As shown in Figure 17–11, if the bit status of I:010 matches the bit status of B3:9, the processor sets bit 08 (FD) of the Control word to 1. If bit 08 of control word R6:50 was programmed to an output device, like indicator lamp (O:014/00) shown in Figure 17–11, the output device would turn *ON* when bit 08 was set to 1. This light indicates that the input devices of the process machine are operating correctly and the machinery is ready to start producing Product A. If the data of the two words does not match, bit 08 will not be set to 1. The processor will continue to compare the status of input word I:010 with the data stored in Step 1 of the sequencer on each program scan. When the input devices are set (*ON* or *OFF*) correctly and match sequencer Step 1, bit 08 will be set to 1, and output indicator lamp O:014/00 will turn *ON*.

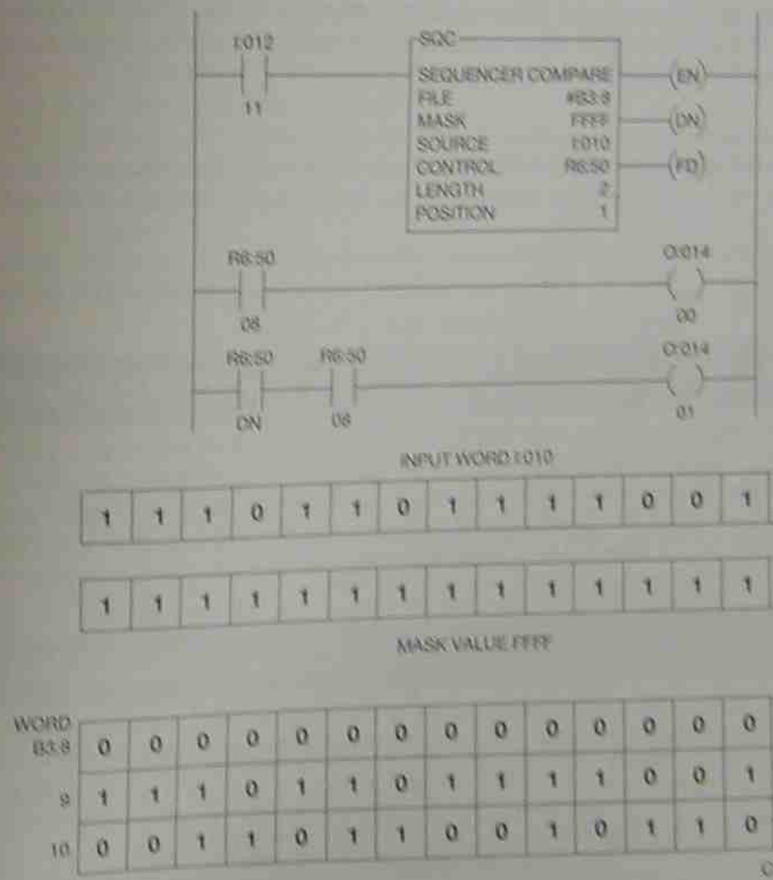


Figure 17–11 SQC Instruction with File Compare

Step 2 of the sequencer could be loaded with the bit status of the machine when it is correctly set up to produce Product B. By opening and closing input device I:012/11, the sequencer is activated and moves to Step 2. The bit status of Step 2 mirrors the status of the input devices on the machine when it is ready to produce Product B. If output light O:014/01 comes *ON*, the operator knows the input devices are operating correctly and he or she can start the process of producing Product B. As programmed, output O:014/01 can only be turned *ON* when the DN bit (bit 13) of the sequencer is set to 1, indicating that the sequencer is on the last step and the FD bit (bit 8) has been set to 1. The program could have been written by entering FD instead of 08, and 13 instead of DN. Either way, output light O:014/01 can only come *ON* when the sequencer is on the last step, Step 2, and the data in sequencer Step 2 matches the status of input word I:010. If the light fails to come *ON*, the operator knows that one or more of the input devices are not operating correctly. Using the Video Display on the programming device, it could be determined which device(s) are not set properly.

This instruction can be used as a powerful diagnostic tool to determine correct machine operation. This instruction can be used to compare input words as well as output words that represent input and output devices used for the manufacturing process.

Another sequencer instruction is the Sequencer Load (SQL) instruction. This instruction allows data to be loaded into a sequencer file from a source address on each false-to-true transition. Figure 17–12 shows the Sequencer Load instruction, the Source word, and the five-word sequencer file.

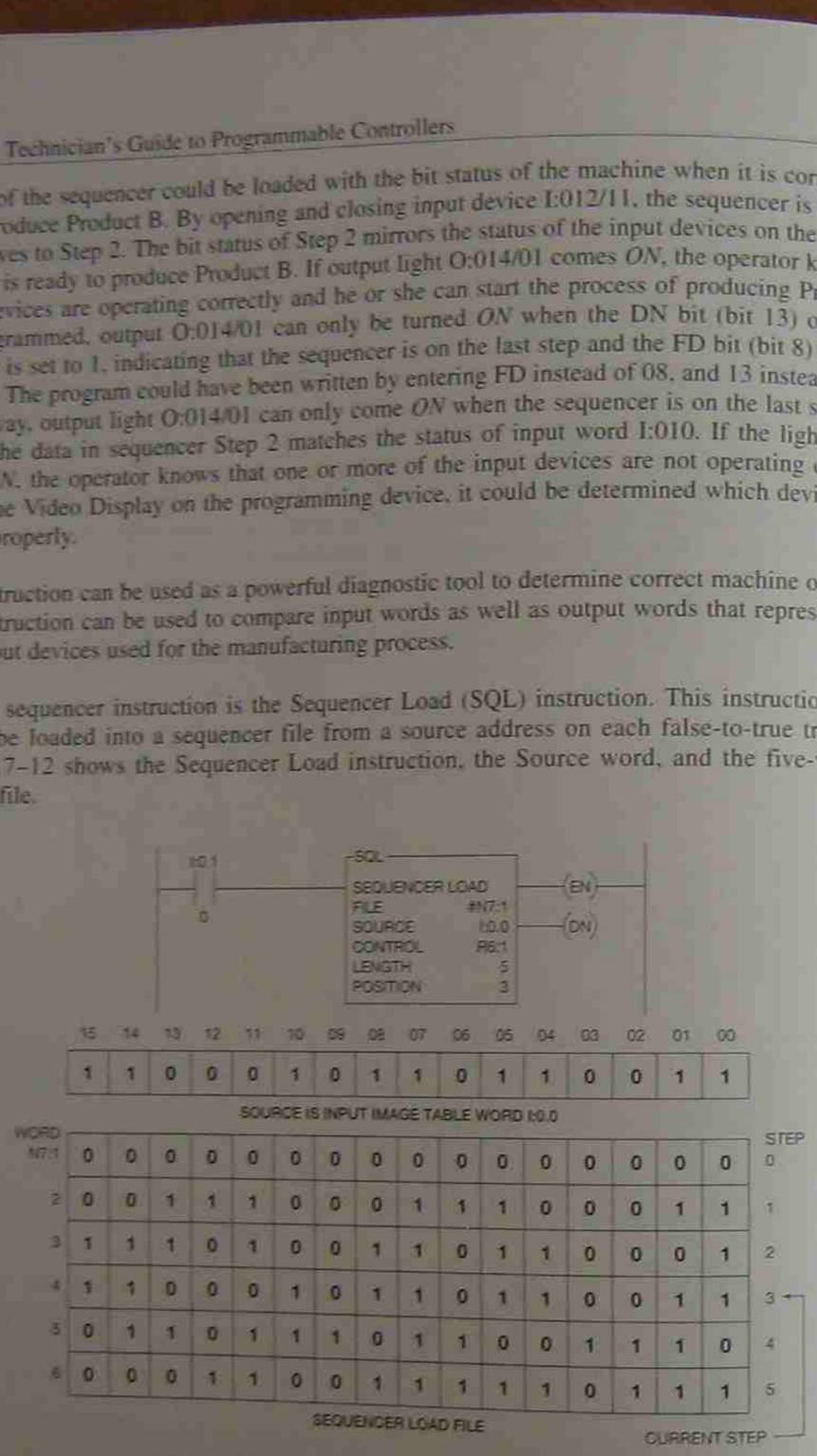


Figure 17–12 Sequencer Load (SQL) Instruction

As with the other sequencer instructions, the file used for the SQL instruction is a user-defined file and must be preceded by the # symbol. The source is shown as I:0.0. This address tells us that this instruction has been programmed using either an SLC 500 or a MicroLogix PLC with fixed I/O. The source word address is Input Word 0 in slot 0.

The input that controls the SQL instruction has an address of I:0.1/0, indicating that this is word 1, slot 0, bit 0. If you refer back to Figure 5–8, you will see that bit 0 of word 1 of the input image table is input device number 16.

On each false-to-true transition of input device I:0.1/0, the current status (1 or 0) of each input device represented by input image word 0 will be transferred into the sequencer file. Figure 17–12 shows the SQL instruction in Step 3. The information shown in the Source word I:0.0 has been written into Step 3 of the sequencer file. On each false-to-true transition, the instruction will be incremented by 1 and the current status of input devices in input image table word 0.0 will be written into the file N7:1. When the instruction has reached Step 5, the DN bit (bit 13) will be set to 1. On the next false-to-true transition of the instruction, the Sequencer Load instruction will recycle to Step 1 and the status of word 0.0 will be loaded into the file, overwriting previous information that had been loaded into Step 1 of the file.

In this SQL example, an input word was used for the source. The source can also be a file address or a constant (-32,768 to 32,767).

The SQL instruction is like a word-to-file move instruction and could be used to store numeric data from RTDs, thermocouples, and the like.

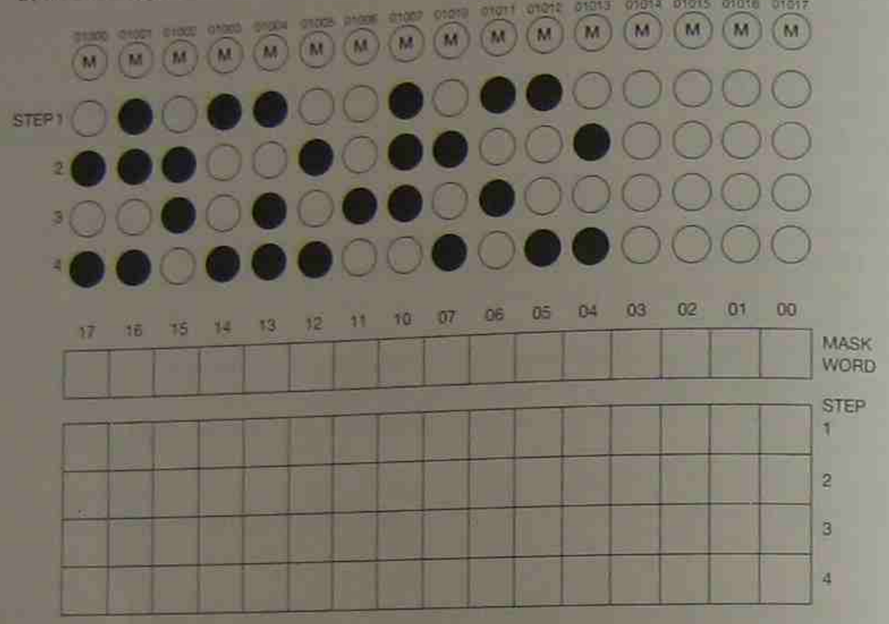
Chapter Summary

Although sequencers, like other data manipulation and arithmetic instructions, are programmed differently with each PLC, the concepts are the same. Data is entered into a word file for each sequencer step, and, as the sequencer advances through the steps, binary information is transferred sequentially from the word file to the output word(s). Output word bits can be masked so they can operate independently of the sequencer.

Review Questions

1. Briefly describe a *sequencer*.
2. A series of consecutive words are referred to as:
   a. deck
   b. group
   c. file
   d. chain
3. What is the purpose of a *mask word* in a sequencer?
4. What device is commonly replaced by a sequencer instruction?

5. Set up the file in Figure 17–A so the sequencer will operate the motors as shown in steps 1, 2, 3, and 4. Program the circuit so motors 01015, 01016, and 01017 cannot be energized.



6. What is the maximum length of a sequencer file when using the SLC 500?
7. Which Allen-Bradley sequencer instruction is like a *file-to-word* move?
8. Which Allen-Bradley sequencer instruction is like a *word-to-file* move?
9. The maximum length of an SLC 500 sequencer file is _____.
10. What condition would cause bit 08 (FD) of an SQC instruction control word to be set to 1?
11. What would be the mask word bit status if the mask was set at FFFF?
12. What would be one use for an SQC instruction?

CHAPTER

18 Programming With Boolean

Objectives

After completing this chapter, you should have the knowledge to:

- Define the Boolean term *and*.
- Define the Boolean term *or*.
- Define the Boolean term *not*.
- Interpret truth tables.
- Use Boolean functions to write simple programs.

BOOLEAN ALGEBRA

Don't let the term Boolean or algebra scare you off at this point. Once the electrician or technician understands some Boolean concepts, he or she will find that programming in Boolean is not only fun, but also fast.

Note: *The Boolean concepts covered in this chapter are basic and oversimplified. Full coverage of Boolean algebra can be found in most electronic textbooks that deal with logic gates and logic circuits.*

Boolean algebra dates back to 1854 when mathematician George Boole developed his mathematical system of logic. In a logic problem there are only two states: *true* or *false*.

Boolean algebra is a unique system that differs from regular high school and/or college algebra. In the Boolean system there are only two digits: 0 and 1. Every number must be a one (1) or a zero (0), and there are no fractional or decimal numbers, no 2s, 3s, etc.

Relay ladder diagrams can be thought of as logic problems because relay contacts can have only two states: closed (true, or 1), or open (false, or 0).

To better understand the concept, consider the circuit in Figure 18–1.

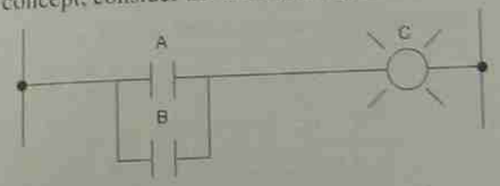


Figure 18–1 Basic OR Relationship

With contacts A and B wired in parallel, closing either contact A *or* contact B turns on lamp C. This type of circuit arrangement (when either A *or* B can turn on C) is referred to as **OR logic**.

The **Boolean equation** for an or statement would be written A + B = C. The plus sign (+) indicates OR logic. According to the formula, if A *or* B is true, then C is true. A truth table for the equation shown in Figure 18–2 further illustrates and clarifies the Boolean concept.

CONTACT	CONTACT	LAMP	
A	B	C	A + B = C
0	0	0	0 + 0 = 0
0	1	1	0 + 1 = 1
1	0	1	1 + 0 = 1
1	1	1	1 + 1 = 1

Figure 18–2 OR Truth Table

Note: *A 0 represents an open, or* OFF, *condition, whereas a 1 represents a closed, or* ON, *condition.*

The results of adding 1 + 1 is 1, not 2 as is normally the case. Remember, in Boolean there are only two digits: 0 and 1.

A logic statement for the series contacts shown in Figure 18–3 shows that if contacts A *and* B are closed, C lights.

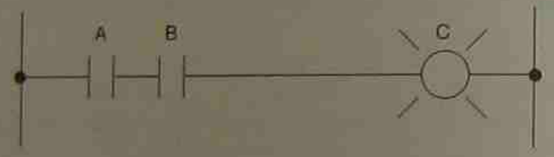


Figure 18–3 Basic AND Relationship

This circuit configuration (when both A *and* B must be closed to turn on C) is referred to as **AND logic**. The Boolean formula for an AND statement would be written A × B = C. The multiplication sign (×) indicates AND logic.

Note: *The AND function can also be expressed as A • B = C or AB = C.*

The formula states that if A *and* B are true, then C must be true. A truth table for this equation is illustrated in Figure 18–4.

CONTACT	CONTACT	LAMP	
A	B	C	A × B = C
0	0	0	0 × 0 = 0
0	1	0	0 × 1 = 0
1	0	0	1 × 0 = 0
1	1	1	1 × 1 = 1

Figure 18–4 AND Truth Table

Look at the simple STOP/START circuit illustrated in Figure 18–5.

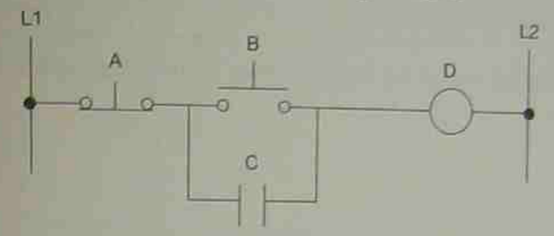


Figure 18–5 Standard STOP/START Circuit

This circuit contains both AND logic and OR logic. The logic is, if A is closed, *and* either B *or* C is closed, D lights. The Boolean formula is A × (B + C) = D. Figure 18–6 shows the truth table and equations.

CONTACT	CONTACT	CONTACT	LOAD	
A	B	C	D	A × (B + C) = D
0	0	0	0	0 × (0 + 0) = 0
0	0	1	0	0 × (0 + 1) = 0
0	1	0	0	0 × (1 + 0) = 0
0	1	1	0	0 × (1 + 1) = 0
1	0	0	0	1 × (0 + 0) = 0
1	0	1	1	1 × (0 + 1) = 1
1	1	0	1	1 × (1 + 0 = 1
1	1	1	1	1 × (1 + 1) = 1

Figure 18–6 Truth Table for Standard STOP/START Circuit Logic

Another Boolean concept is the **NOT function**. The NOT function is like the EXAMINE OFF instruction discussed in Chapter 8. The NOT function acts like a set of N.C. contacts. Consider the circuit in Figure 18–7.

Figure 18–7 Basic NOT Relationship

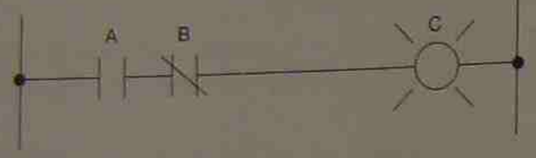
As long as A remains closed (*not* open), lamp B is *ON*. Another approach is to think of the NOT function as being true when the device or address that it represents is *not ON*, or is set to 0. The Boolean formula for a NOT function is $\bar{A} = B$. The truth table for a NOT function is shown in Figure 18–8.

CONTACT	LAMP	
$\bar{A}$	B	$\bar{A}$ = B
0	1	O = 1
1	0	1 = 0

Figure 18–8 NOT Truth Table and Boolean Formula

With the NOT function, when A is 0, the logic is true, so lamp B will be *ON*. When A is set to 1, the logic is false, and lamp B will be *OFF*.

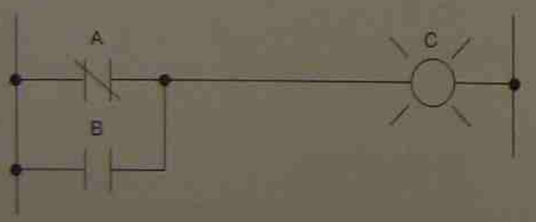
Combining the NOT function with the AND function is illustrated in the circuit and truth table shown in Figure 18–9.



CONTACT	CONTACT	LAMP	
A	$\bar{B}$	C	A x $\bar{B}$ = C
0	0	0	0 x 0 = 0
0	1	0	0 x 1 = 0
1	0	1	1 x 0 = 1
1	1	0	1 x 1 = 0

Figure 18–9 Circuit and Truth Table for AND and NOT Functions

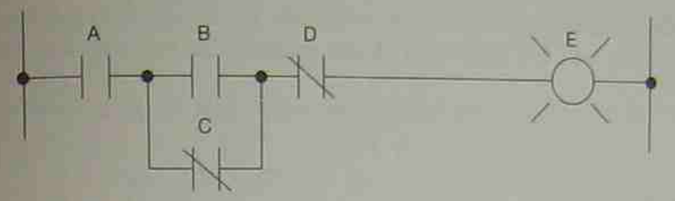
Combining the NOT function with the OR function is illustrated in the circuit and truth table shown in Figure 18–10.



CONTACT	CONTACT	LAMP	
$\bar{A}$	B	C	$\bar{A}$ + B = C
0	0	1	0 + 0 = 0
0	1	1	0 + 1 = 1
1	0	0	1 + 0 = 0
1	1	1	1 + 1 = 1

Figure 18–10 Circuit and Truth Table for NOT and OR Functions

Combining the AND, OR, and NOT functions are illustrated in the circuit and truth table shown in Figure 18–11.



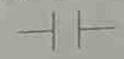
CONTACT	CONTACT	CONTACT	CONTACT	LAMP	
A	B	$\bar{C}$	$\bar{D}$	E	A x (B + $\bar{C}$) x $\bar{D}$ = E
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	1	
1	1	0	1	0	
1	1	1	0	1	
1	1	1	1	0	

Figure 18–11 Circuit and Truth Table Combining AND, OR, and NOT Functions

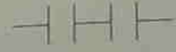
PROGRAMMING IN BOOLEAN

The techniques used to program a PLC in Boolean are very similar, no matter which manufacturers' PLC is being used. Although the keys used to enter the program may vary from manufacturer to manufacturer, the concepts are the same. Typical keys used are:

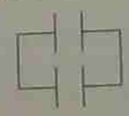
1. STR (Store)
   This instruction begins a rung or branch with a normally open contact.



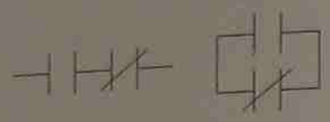
2. AND (AND function)
   This instruction will AND a normally open contact with another contact.



3. OR (OR function)
   This instruction will OR (place in parallel) a normally open contact with another contact.



4. NOT (NOT function)
   This instruction will add a normally closed contact. Can be used with an AND or an OR instruction.



5. OUT (Output)
   This instruction adds the output.

6. ENTER (ENTER instruction)
   This instruction enters instructions and addresses into program memory.

7. NEXT ADRS (Next address)
   This instruction is used to start the next series of instructions and addresses.

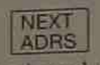


Figure 18–12 shows a simple line of ladder logic and the keystrokes required for programming.

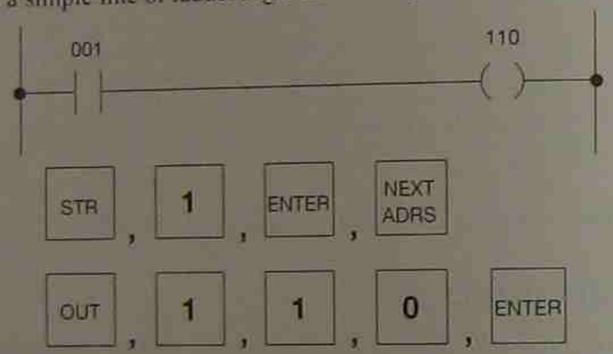


Figure 18–12 Programming Simple Line of Ladder Logic

Programming is started by pushing the STR (Store) key followed by up to a 3-digit address.

Note: *With most small PLCs, it is not necessary to enter zeros that preceed numbers in addresses. The address 001 requires that only a 1 be pushed. Similarly, the address 050 requires that only a 5 and a 0 be pushed.*

The STR key followed by the number 1 enters a N.O. contact addressed 001.

Pushing the ENTER and NEXT ADRS (next address) keys loads the N.O. contact 001 and readies the processor for the next instruction. Pressing the OUT (output) key and then the 3-digit number 110 gives us the output address 110. Pressing the ENTER key loads the rung of logic.

Note: *Because the OUTPUT was the last element in the program, only the ENTER key was needed to load the information. If an additional rung (or rungs) was to be programmed, then both the ENTER and NEXT ADRS keys would be used.*

Once the programming is complete, the processor is placed in the *monitor* mode (*run* mode) to verify the program.

Figure 18–13 shows two contacts in series (AND function) and the keystrokes required for programming.

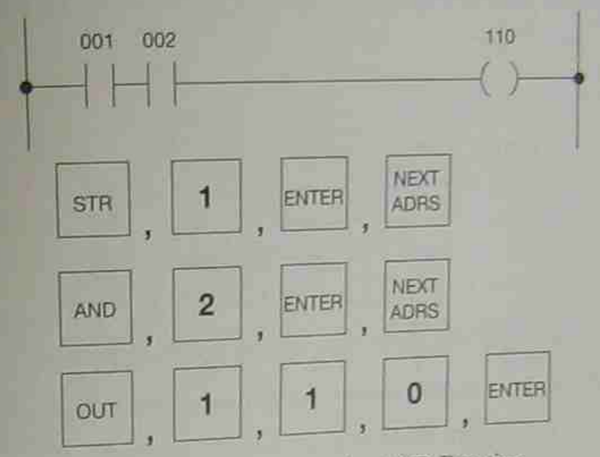


Figure 18–13 Programming AND Function

The keystrokes: STR, 1, gives us an N.O. contact 001. The keystrokes: AND, 2, puts an additional N.O. contact (002) in series with N.O. contact 001. The keystrokes OUT, 1, 1, 0, provides output 110 to complete the line of logic.

To program normally closed (N.C.) contacts (or a NOT function) the STR key is followed by the NOT key (shown in Figure 18–14).

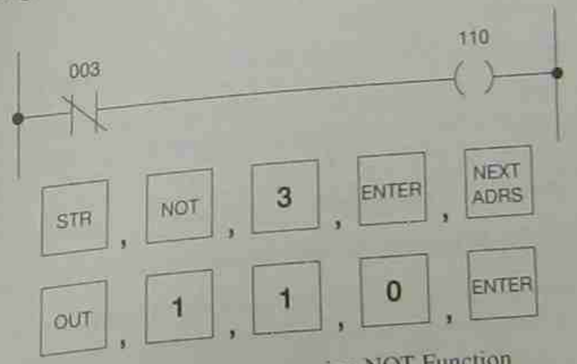


Figure 18–14 Programming NOT Function

Figure 18–15 shows a circuit with an N.O. contact 001 in parallel (or ORed with a N.C. or NOT contact 003) controlling output 110. The keystrokes used to program the circuit are also shown.

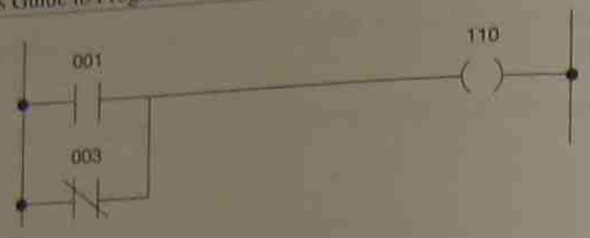


Figure 18–15 Programming OR Function

For more complex circuits where AND and OR functions are combined, the PLC processor typically uses a temporary file, or stack system, to store information as the circuit is being developed. Each time the STR key is pushed, the information that follows is put into the temporary file or stack. Information enters the stack from the bottom and pushes previous information up. Figure 18–16 illustrates this concept.

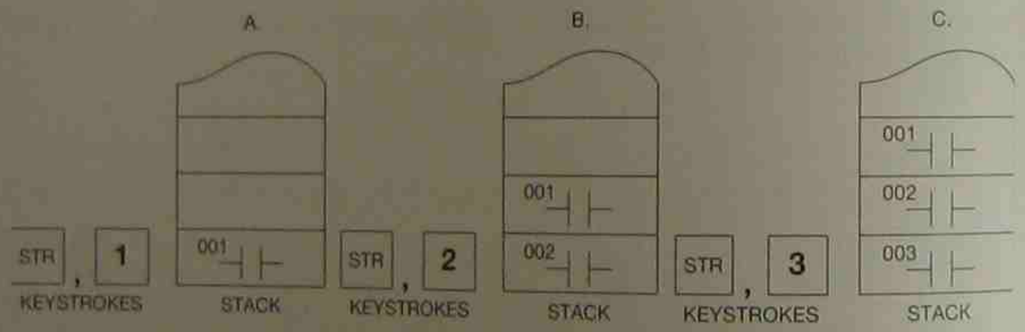


Figure 18–16 Entering Information into Temporary File (Stack)

Note: *Because it is understood that the ENTER and NEXT ADDRESS keys* must *be used during each step of program development, these keystrokes will no longer be shown.*

When the first STR (STR, 1) was entered at (a), N.O. contacts 001 were placed on the bottom of the stack. When the second STR (STR, 2) is entered at (b), N.O. contacts 002 are inserted at the bottom of the stack, and contacts 001 move up one place. At position (c), the third STR command is entered and N.O. contact 003 enters the bottom of the stack. The previous entered contacts

again move up. At any point in the program, the contacts can be retrieved in inverse order (last in–first out), or LIFO.

The keystrokes AND, STR take the last contact entered (003) and AND it with the next contact up in the stack (002). Figure 18–17 shows what the stack looks like after the AND, STR commands are entered.

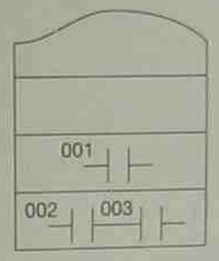


Figure 18–17 LIFO Stack after AND, STR Command

If the keystrokes OR, STR had been entered, the contacts 003 and 002 would have been ORed as shown in Figure 18–18.

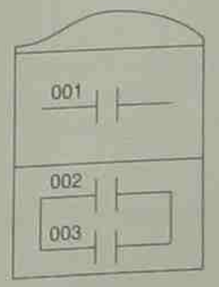


Figure 18–18 LIFO Stack after OR, STR Command

With the contacts in the order shown in Figure 18–18, an AND, STR, series of keystrokes would AND the ORed contacts 002 and 003 with contacts 001, as shown in Figure 18–19.

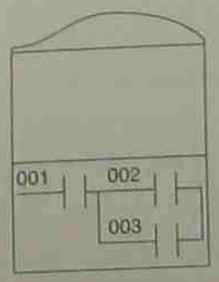


Figure 18–19 LIFO Stack after AND, STR Command

Different STR commands yield different results on the original stack in Figure 18–16. Figures 18–20a, 18–20b, and 18–20c show some alternative keystrokes, and the results of each.

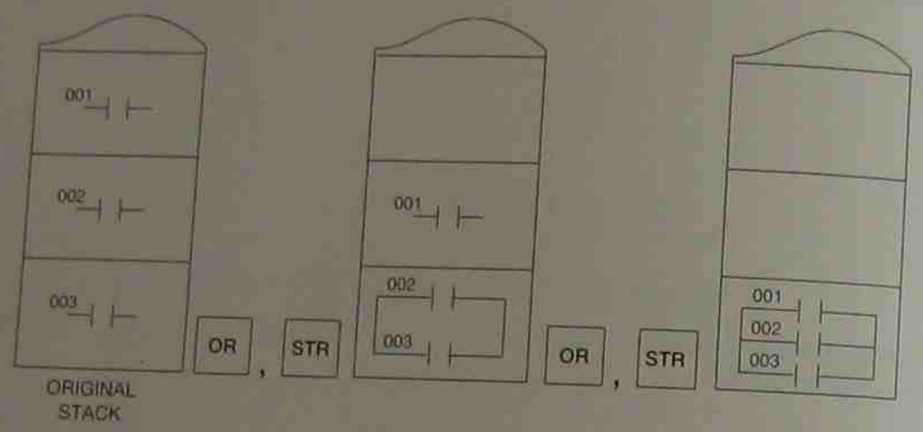


Figure 18–20a Stack after Two Consecutive OR, STR Commands

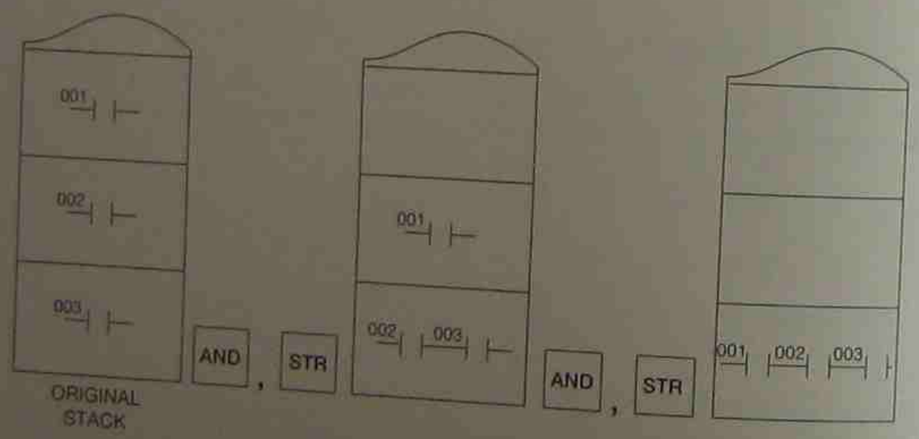


Figure 18–20b Stack after Two Consecutive AND, STR Commands

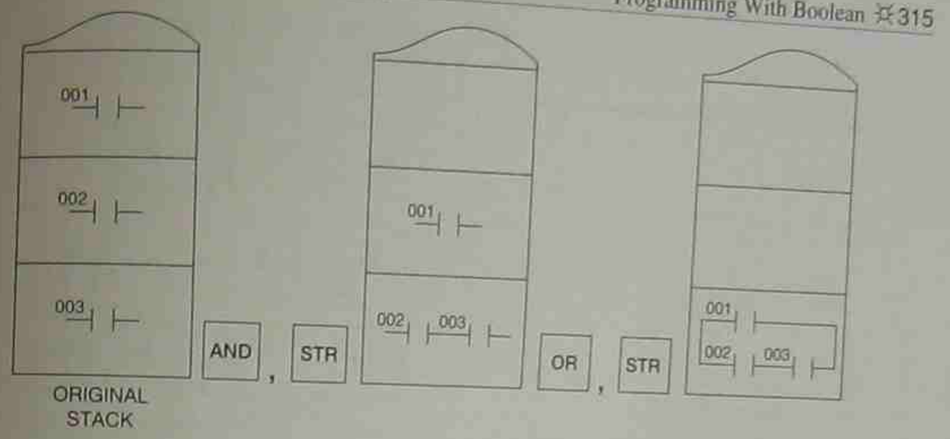


Figure 18–20c Original LIFO Stack after AND, STR and OR, STR Commands

The circuits shown in figures 18–20a, 18–20b, and 18–20c could have been programmed without first developing a stack of three contacts. This method was used to demonstrate a concept, and was used for illustration only.

The keystrokes used to program the three parallel contact shown in Figure 18–20a are:
STR, 1
OR, 2
OR, 3

To program the contacts in series as shown in Figure 18–20b, the following keystrokes are used:
STR, 1
AND, 2
AND, 3

The following keystrokes are used to program the parallel series combination circuit shown in Figure 18–20c:
STR, 2
AND, 3
OR, 1

To complete the circuits shown in these figures, the OUTPUT key is pressed and a 3-digit address entered. Although not all small PLCs allow for parallel outputs, most do. Figure 18–21 is an example of parallel outputs and the keystrokes required for programming.

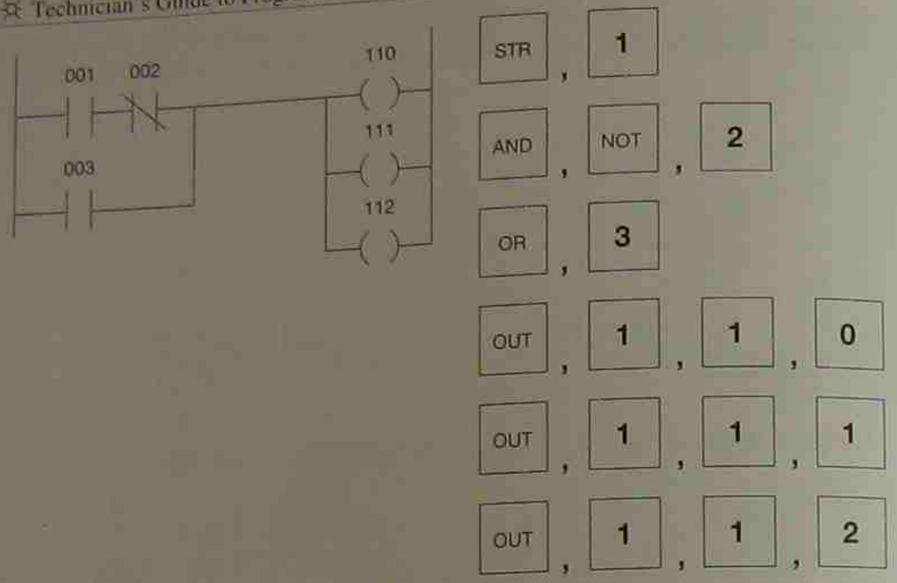


Figure 18–21 Programming Parallel Outputs

Note: *Remember that ENTER and NEXT ADDRESS follow each step of programming, and are not shown for the sake of simplicity.*

As stated earlier in this chapter, programming in Boolean is fun and fast. Notice that no BRANCH START or BRANCH END commands are needed, which speeds programming time.

To further illustrate the programming techniques and the LIFO (last in–first out) stack concept, Figure 18–22 shows a series parallel combination circuit with the keystrokes and the temporary stack results.

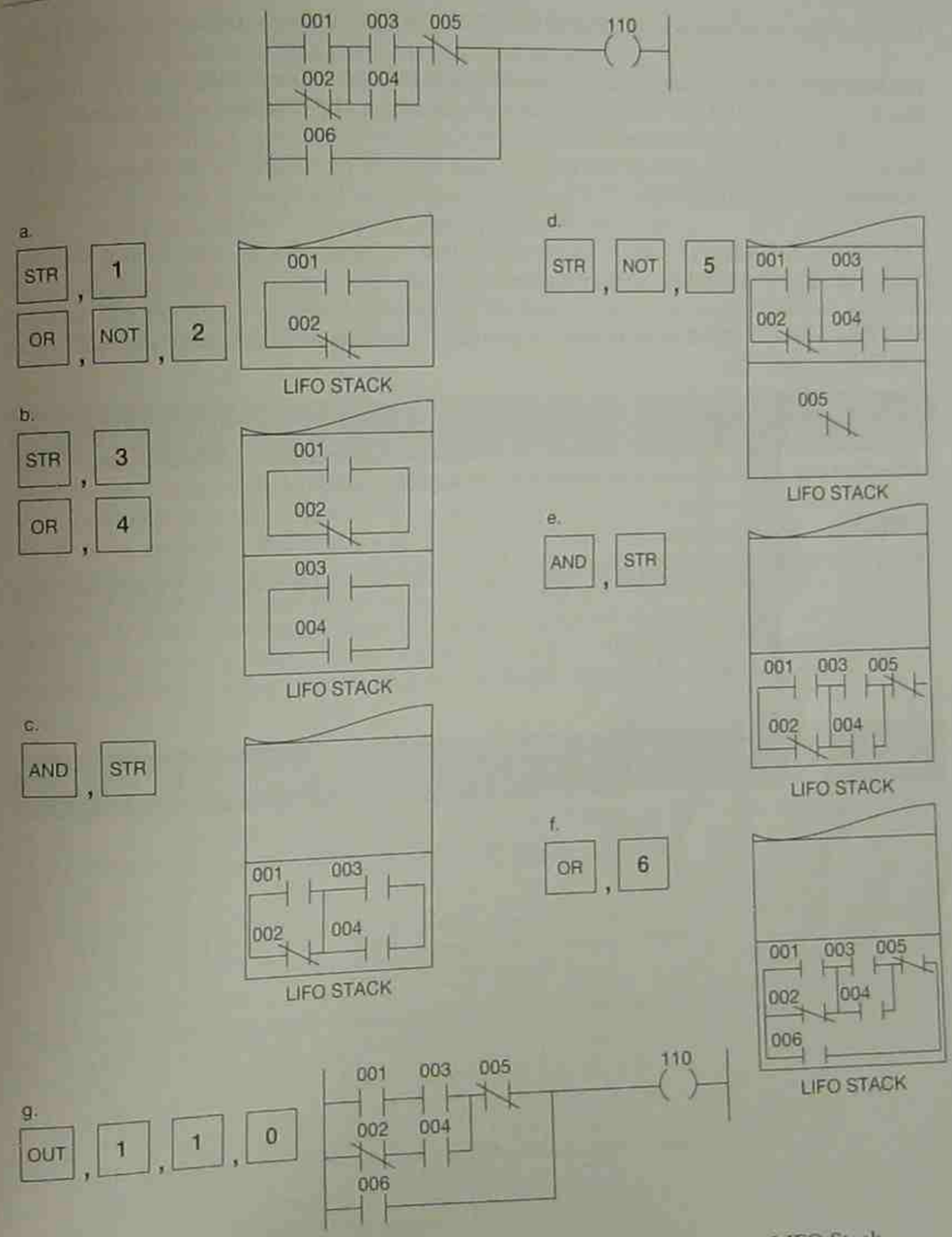


Figure 18–22 Programming a Combination Circuit and the Resulting LIFO Stack

The keystrokes in (a) connect contacts 001 and 002 in parallel, and place them in the LIFO stack.

The keystrokes in (b) connect contacts 003 and 004 in parallel, and enter this data into the stack. This moves contacts 001 and 002 up to make room for contacts 003 and 004 on the bottom.

The next step (c) brings contacts 001 and 002 back down to the bottom of the stack, and ANDs them with contacts 003 and 004.

The STR, NOT, 5, keystrokes at (d) moves the ANDed contacts up in the stack, and a N.C. NOT contact (005) moves into the bottom of the stack.

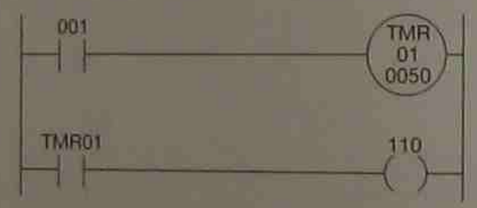
The AND, STR command at (e) recalls the contacts to the bottom of the stack, and ANDs them with NOT contact 005.

The next step (f) recalls the contacts from the LIFO stack, and ORs them with contact 006.

In the final step, at (g), the output is added with the keystrokes OUT, 110.

TIMERS

Figure 18–23 shows a simple on delay timer circuit and the keystrokes necessary for programming.



STEP 1	STR	1	ENTER	NEXT ADRS		
2	TMR	1	ENTER	NEXT ADRS		
3	5	0	ENTER	NEXT ADRS		
4	STR	TMR	1	ENTER	NEXT ADRS	
5	OUT	1	1	0	ENTER	

Figure 18–23 Programming Simple ON DELAY Timer

Step 1. The contact to control the timer is entered (N.O. 001).
Step 2. A timer is entered and identified as timer number 1.
Step 3. The preset for timer 1 is entered. The time base is 0.1 seconds. Timer 1 in this illustration is 5 seconds (50 × 0.1 seconds = 5 seconds).
Step 4. A set of timer contacts is entered that will close five seconds after timer 1 (TMR 1) is energized.
Step 5. The circuit is completed with output 110. Output 110 is controlled by timer contacts, and is delayed by five seconds after input device 001 closes.

The examples and discussion in this chapter provide only an introduction to programming in Boolean, but provides a basic understanding of the concept to enable an electrician or technician to read the user manuals for any small PLC, and to get started programming.

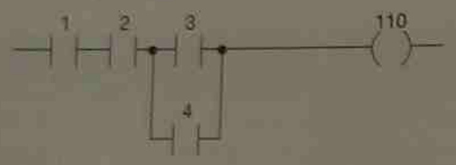
Chapter Summary

Programmable controllers can be grouped by I/O size. Generally, any PLC with up to 128 I/O is categorized as a small PC. The small PLCs are small physically, but offer many of the features of larger, more expensive, PLCs. Many of the small PLCs are programmed using limited Boolean statements rather than RELAY LADDER LOGIC. Dedicated programmer costs are high (some cost more than the processor, power supply and I/O section combined). Because cost is usually a factor in the small PLC market, Boolean programming is used to help create a system with minimal programmer costs. Typical Boolean functions are AND, OR, and NOT. Programming a small PLC that uses Boolean logic is fast and easy once the basics are understood.

Review Questions

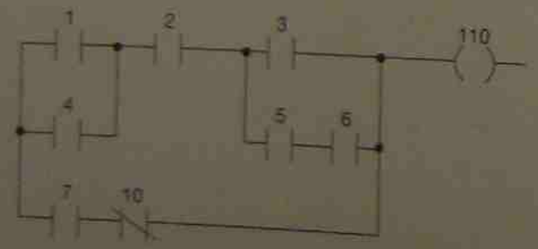
1. T F Boolean algebra was developed in the 1960s when PLCs were first being used.
2. In the Boolean system, which of the following digits are used?
   a. 1, 2, 3, and 4
   b. 1, 3, and 5
   c. 0, 1, 2, and 4
   d. 0, 1, and 2
   e. none of the above
3. What Boolean relationship do contacts have that are connected in parallel?
   a. AND
   b. OR
   c. NOT
   d. IF
4. In a Boolean equation, a plus sign (+) indicates which type of logic?
   a. AND
   b. OR
   c. NOT
   d. IF

5. A multiplication sign (×) indicates what type of Boolean logic?
   a. AND
   b. OR
   c. NOT
   d. IF
6. The NOT function in a circuit acts like a set of:
   a. N.O. contacts
   b. N.C. contacts
   c. timed contacts
   d. intermittent contacts
7. Using the programming format covered in this chapter, list the keystrokes necessary to program the circuits shown below.

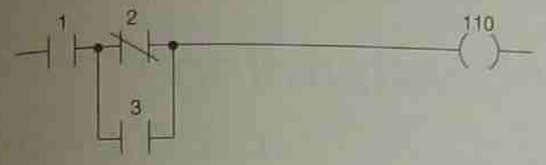


8. List the keystrokes necessary to program the following circuit.

9. What keystrokes are required to enter the following program?



10. Complete a truth table for the circuit shown below.



CONTACT 1	CONTACT 2	CONTACT 3	OUTPUT 110

11. Complete a truth table for the circuit shown in Question 8.

CONTACT 1	CONTACT 2	CONTACT 3	CONTACT 4	OUTPUT 110

12. Define the term *LIFO*.

CHAPTER

19 Understanding Basic MS-DOS® Commands

Objectives

After completing this chapter, you should have the knowledge to:

- Format a new disk.
- Create files and directories.
- Copy files and directories.
- Rename files and directories.
- Use the *tree* command.

While most programs used with today's PLCs are Windows® based, understanding basic computer commands is still helpful.

Microsoft Corporation, in cooperation with IBM®, developed software known as **MS-DOS®**, which is an acronym for **M**icro**S**oft **D**isk **O**perating **S**ystem. The number that follows the word DOS is the version number. For example, DOS 1.0 (version 1) was introduced in 1981, and since that time, there have been many revisions to the original program as the software is expanded and enhanced, each subsequent version will be assigned a new number. When the software is upgraded, the software engineers try to maintain **upward compatibility**. That means that the features of the older version are supported, and can be used along with the new features of the revised version. This allows files created in an older version of DOS to be used in a newer version. The examples and commands used in this chapter are for DOS 6.0.

STARTING THE COMPUTER

DOS software is normally stored on the hard drive of the computer, but when using a computer without a hard drive, the DOS program is stored on a diskette. To use the software, the computer must first be **booted**. This term indicates that the computer has been turned on, and the software loaded into memory. When the computer is first turned on, it is said to be cold started. Similar to a PLC, the first thing the computer does is run a self-diagnostic test (referred to as a power-on self-test or POST). After the initial self-test is run and passed, the computer reads the DOS software from either the diskette or the hard drive. Once the program is read, the system prompt is displayed. The prompt is for the default drive. On most computers *with* a hard drive, the default (or normal) drive is drive C. Therefore, the prompt that appears is C, followed by a colon (:). On a

computer *without* a hard drive, the default drive is usually drive A. Changes from the default drive to another drive are made by typing the letter of the desired drive, followed by a colon. For example, if the default drive is drive C, and a change to drive A is needed, at the prompt, the letter A is typed followed by a colon(:).

EXAMPLE:

Original prompt C:
Type A:, and hit the RETURN, or ENTER key

The prompt is changed to A:. Drive A is now the primary drive and programs can be loaded or run from that drive.

FORMATTING A DISK

Once the computer is booted (turned on and operational) and the DOS program loaded, commands can be entered. One of the first commands that should be learned is how to format a disk. Although it is possible to buy formatted disks, the electrician or technician may have to format the disk prior to using it. The formats for IBM compatible computers and Apple® computers, for example, are quite different. Therefore, the discussion here is only for IBM compatible computers running DOS. There are basically two sizes of floppy disks: 5¼" and 3½". Both are available in double sided, double density (DS,DD) and double sided, high density (DS, HD). The storage capacity is different for each size and density, and are as follows:

5¼" DS,DD 360K
5¼" DS,HD 1.2 M

3½" DS,DD 720K
3½" DS,HD 1.44M

The 3½" DS,HD disk is the most common disk used today because the most common drive is high density.

Note: *Storage is rated in K bytes and M bytes. K stands for* kilo *and equals 1,000. M stands for* mega *and equals 1,000,000.*

To format a new 3½" DS,HD disk, place the disk in the appropriate drive (in this example, drive A is the 3½" high density drive. At the C: prompt, type FORMAT A: and hit the ENTER key. The DOS software shows the following prompt:

Insert new diskette for drive A:
and press ENTER when ready. . .

Once the new 3½" DS,HD disk is inserted, and the RETURN or ENTER key is pressed, the next prompt reads:

Checking existing disk format.
Saving UNFORMAT information.
Verifying 1.44M

Note: *The "verifying 1.44M" message indicates that the software is checking the disk to make sure it is 1.44 megabytes. To format a 3½" DS,DD disk with 720K of storage in a high density drive, the following command line must be entered at the prompt:*

FORMAT A:/F:720

Formatting of the disk now begins, and the percent of the format that is completed is displayed on the screen. Once the disk format is complete, the prompt indicates "Format Complete." The next prompt asks if a volume label is to be assigned. The volume label is used to help identify the diskette. The label appears as an entry before the listing of files and helps to verify which diskette is being accessed. The volume label can be up to 11 characters in length. These characters can include letters, numbers, and spaces only: *no* special characters or punctuation can be used. If a volume label is not desired, press the ENTER key to indicate "None." The screen then lists the bytes of total disk space *used*, and the total disk space *available* for use. The prompt then asks if another disk is to be formatted (Y/N)?. If a second disk is to be formatted, remove the previously formatted disk and insert a new one, and press the Y key to initiate the formatting procedure.

CREATING FILES AND DIRECTORIES

Once a disk has been formatted, it can be used to store new information (PLC programs, files, etc.) or to store existing information as a backup to the hard drive. Before a program can be stored, it must be given a name so it can be located at a later time. The name given to the program is its file name. The file name may also have an extension to help identify the contents of the file. The original file name cannot be more than eight characters in length, and the extension cannot be more than three characters. Therefore, the file with its extension cannot exceed eleven characters (the period that separates the file name and the extension is *not* counted as a character). If a program is written for a machine that mixes different chemicals to make a lawn fertilizer, the file could be named *lawn*. If the mix uses 20 percent nitrogen, an extension could be added to the file name to further identify the program. An extension is added by placing a period (.) *after* the original file name and then adding the extension. The new file name with the extension in this example would be *lawn.20%*. MS-DOS has rules for naming files. The file name may contain letters, numbers, and some special characters, but for the sake of simplicity and ease in remembering file names, it is normal to use only letters and numbers.

As a general rule, use letters A through Z and the numbers 0 through 9. If desired, the following special characters could be used:

tilde	(~)	ampersand	(&)
exclamation point	(!)	parentheses	( )
at sign	(@)	underscore	(_)
number sign	(#)	hyphen	(-)
dollar sign	($)	braces	({ })
percent sign	(%)	apostrophe	(')
caret	(^)	grave accent	(`)

None of the other special characters can be used.

File names and extensions cannot contain spaces, commas, backslashes (\), or periods (except when the period is used for the extension to the original file name).

File names and extensions cannot use the following names that have been reserved for use by DOS: AUX, CLOCK$, COM1 through COM4, CON, NUL, and PRN.

Some program software requires that a file name be entered prior to the start of the programming process, whereas other software does not require that a file name be assigned until the program is complete or is to be stored or saved. PLC programs usually do not ask for an extension because the program automatically assigns an extension when the file is saved. However, once a number of files have been created, it becomes necessary to manage them so that any given file can be retrieved on demand. The disk (hard drive or diskette) where the files are stored should be thought of as a file cabinet. If the files are simply thrown into the cabinet, it is difficult to find them at a later date. Directories are used to organize the way in which files are stored, and are similar to the drawers in a file cabinet.

If 20 or 30 different files (programs) for various mixes of lawn fertilizer are created, it is easier to organize them if a directory, or drawer in the file cabinet, is named LAWN. The files containing the various *lawn* mixes are then placed into this directory (see Figure 19–1).

Figure 19–1 Directory with Files

Directories, like file names, can also have extensions, although the same restrictions apply to naming directories as those that are applied to naming files. If large numbers of files are under one directory, it may be necessary to create subdirectories to eliminate confusion and speed the retrieval process. For example, within the Lawn directory, subdirectories named NITROGEN and POTASH could be created. Each subdirectory would then contain the appropriate files.

A block diagram of a directory with subdirectories and files is shown in Figure 19–2.

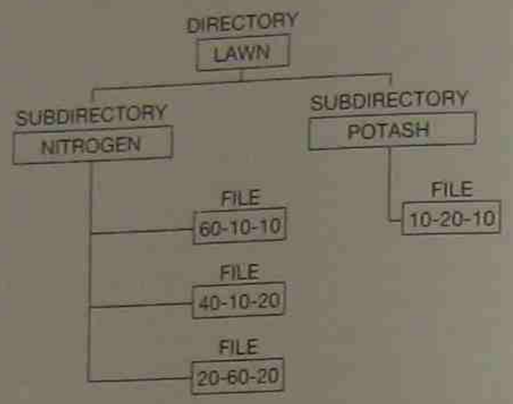


Figure 19–2 Directory with Subdirectories and Files

The directory that contains subdirectories is sometimes called the **parent** directory, and a subdirectory is sometimes called a **child** directory. The subdirectory can also be thought of as a file folder that holds files. The directory is the *drawer* in the file cabinet, and the subdirectory is the *file folder* in the *drawer*. When naming subdirectories, the same rules used for file names and directories must be followed. Subdirectories can also have 3-character extensions.

To retrieve a file, it is necessary to specify the path so the computer knows the location of the file. To retrieve file 20-60-20 from subdirectory NITROGEN in directory LAWN on drive C, the following path is entered at the prompt: C:

C:\LAWN\NITROGEN\20-60-20

Note: *The backslash (\) is used to specify the path from directory to subdirectory to file. Although the examples all appear in capital letters, it is not necessary to enter commands in capital letters.*

When additional programs are written, they are given file names as well as paths so they are stored in the appropriate location. For example, if a program is written for a new potash fertilizer mix that is 20-20-10, the following drive path series are entered to place file 20-20-10 into the subdirectory POTASH in the directory LAWN:

C:\LAWN\POTASH\20-20-10

This instructs the computer to save file 20-20-10 on drive C in directory LAWN and subdirectory POTASH. If the file is to be stored on a diskette in drive A, the following drive path is entered at the prompt C:

A:\LAWN\POTASH\20-20-10

If only the file name was entered, it would have been saved in the default, or current directory, of the specified drive. Although the file would have been saved—thrown in the file cabinet—it might be difficult later to remember where it is. Specifying the directory where a file is to be stored makes retrieval easier and faster.

DIRECTORY (DIR) COMMAND

DOS keeps track of all the directories, subdirectories, and files that are created. To obtain a list of the directories on the hard drive or on a diskette, the DIR command is used. To look at the directories stored on the hard drive, at the prompt C: type DIR, and hit the ENTER (or RETURN) key. The directory of drive C is displayed. If a large number of directories are present on the drive, they scroll rapidly from the start to the end of the directory listings. Only the last screen full of directories is displayed. To view all the directories a screen full at a time, an additional command is used with the DIR command to pause, or page through directories. To page, or go through the directories one screen full at a time, type a slash (/) and the letter P after the DIR command. The new command at the prompt is:

C: DIR/P

Press the ENTER key, and only one page, or screen, of the directory is displayed. A new prompt appears, stating "Press any key to continue." This allows the operator to view the listing one page (screen full) at a time.

To look at the directories on drive A, type:

C: DIR A:

When the ENTER key is pressed, the directory of drive A is displayed on the screen. If the directory on drive A is too long and fills more than one screen, the page command can be entered after the A:. The command line is:

DIR A:/P

It is also possible to pause the screen by pressing the control (CTRL) key and then the S key. This causes the directory screen to stop scrolling, so the directory listing can be read. Press any key to continue scrolling the listing. A third way to stop the screen from scrolling is to use the PAUSE/BREAK key. This key is normally found in the upper right-hand section of the normal keyboard. Pressing this key pauses the scrolling of the screen. To start the screen scrolling again, press any key to continue. Figure 19–3 shows a typical directory listing.

```
Volume in drive C has no label
Volume Serial Number is 2A4C-11DF
Directory of C:\

COMMAND    COM          52925     03-10-00     6:00a
                <DIR>              08-16-00    10:22a
DOS             <DIR>              08-16-00    10:23a
WINDOWS         <DIR>              08-16-00    10:25a
MOUSE           <DIR>              08-16-00    10:25a
PRODPACK        <DIR>              09-16-00    10:26a
TEMP            <DIR>              12-20-00     3:00p
DATA                        189    03-29-00     5:23p
CHKLIST    MS               668    04-08-00     2:39p
SCANV      BAT             7866    04-03-00     5:00p
SYSBACK    BAT             9349    03-10-00     6:00a
WINA20     386                     08-16-00    10:29a
DOS_DISK        <DIR>              08-16-00    10:30a
WIN_DISK        <DIR>              09-27-00     3:27p
NC                            37    03-29-00     5:22p
AUTOEXEC   BAK               232    01-25-00     5:05p
CONFIG     SYS                     09-27-00     3:50p
PF              <DIR>              09-27-00     4:35p
WP51            <DIR>              09-27-00     4:10p
123
Press any key to continue. . .
```

Figure 19–3 Directory Listings

Directories are indicated by <DIR>. The other listings are files. Notice that there is a date and time after each listing. This indicates the last date and time changes were made to the directory and/or files.

When the directory is too long to be displayed on one screen, it is possible to reformat the directory using the wide format display option. The wide format display lists the directories in five columns without dates and times. In many cases, this technique allows all of the directories and files to be displayed. To display the directory listings in wide format as shown in Figure 19–4, at the prompt type:

DIR/W

```
Volume Serial Number is 2A4C-11DF
Directory of C:\
COMMAND.COM    [DOS]          [WINDOWS]      [MOUSE]        [PRODPACK]
[TEMP]         [DATA]         CHKLIST.MS     SCANV.BAT      SYSBACK.BAT
MIDA20.J86     [DOS_DISK]     [WIN_DISK]     [NC]           AUTOEXEC.BAK
CONFIG.SYS     [PF]           [WP51]         [123]          [CLIN]
DIR            [LMODSOFT]     CONFIG.01      GDCP.PAR       [UTIL]
[SPSS]         [COXCO]        SPSS.LOG       SPSS.LIS       SCRATCH.PAD
[BOOF]         [BAT]          AUTOEXEC.BAT   [RENTAL]       [RES.PRJ]
[MISC]         [ADV.ELE]      WASTE          [QAPLUS]       [WINFAX]
[COM]          [TELIX]        [DOWNLOAD]     COMB1.PRG      [LAWN]
[LETTERS]      [IPDS]         AP.BAT         TREEINFO.NCD   [CTC]
               50 File(s)     123040 bytes
                                             171151360 bytes free
```

Figure 19–4 Wide Display Option

INTERRUPT COMMAND (BREAK COMMAND)

The interrupt command, or break command, is used to interrupt, or stop, the existing command and return to the prompt. The command consists of the CTRL key and the C key. For example, if only a portion of the directory is to be viewed, press the CTRL key, and while holding it down press the C key. This ends the directory listing and returns to the prompt. This combination of keys always interrupts whatever command is entered and returns to the prompt. If the computer has been configured for BREAK, pressing the CTRL key and the PAUSE/BREAK key also interrupts the current command and returns to the prompt. These commands are also called break commands.

Clear Screen Command

After entering several commands, the screen becomes crowded. To clear the screen, the clear screen command (CLS) is used. At the prompt type:

CLS, AND PRESS THE ENTER KEY

The screen is cleared of all previous commands and displays, and only the prompt appears at the top left of the screen.

Copy Command

The copy command is used to copy files from one disk to another, from one disk to the hard drive, or from the hard drive to a diskette. To copy file 10-20-10 from a diskette in drive A to a diskette in drive B, the following command line is typed at the C: prompt:

COPY A:10-20-10 B:

When the ENTER key is pressed, file 10-20-10 is copied from the diskette in drive A to a diskette in drive B. In this example, drive A is the source drive and drive B is the target drive. If no source drive is given, DOS assumes that the file is on the default drive. In the previous example, drive C was the default drive.

If a file named TIMERS from the hard drive (drive C) is to be copied to a diskette in drive A, the following command line is used at the C: prompt:

COPY TIMERS A:

When the ENTER key is pressed, the file TIMERS is copied from drive C to a diskette in drive A. A source drive is not designated because the file is on the default drive, and DOS assumes the source drive to be the same as the default drive.

To save time in copying all the files on one disk to another, the copy command is followed by and asterisk (*), a period (.), and another asterisk (*). To copy all the files on a diskette in drive A onto a diskette in drive B, the following command line is entered at the C: prompt:

COPY A:*.* B:

All the files on the diskette in drive A are copied onto the disk in drive B. Because no file names were specified, the file names from diskette A are used for the file names on diskette B.

The partial directory shown in Figure 19–3 can be used to further illustrate the copy command. To copy the COMMAND.COM file from drive C to a diskette in drive A, the following copy command is used at the C: prompt.

COPY COMMAND.COM A:

When the ENTER key is pressed, file COMMAND.COM on drive C is copied onto the diskette in drive A. In this example, the C drive is the source, and the A drive is the destination.

To copy the file in this example, but change the name of the file to COMM.2 on drive A, the following copy command is entered at the C: prompt.

COPY COMMAND.COM A:COMM.2

When the ENTER key is pressed, the COMMAND.COM file is copied from drive C to drive A, and the name is changed to COMM.2.

To copy a file that is in a directory and/or subdirectory, the drive path must be specified. In Figure 19–2, the mix for a certain fertilizer (10-20-10) was stored in directory LAWN and subdirectory POTASH. To copy file 10-20-10 onto a diskette in drive A, the following copy command is typed at the C: prompt.

COPY LAWN\POTASH\10-20-10 A:

The drive path specified tells the computer to copy file 10-20-10 from directory LAWN and subdirectory POTASH from C drive to A drive. The copy command cannot be successfully executed unless the full drive path is specified.

MAKING DIRECTORIES COMMAND

The make directory command (MD) is used to make a directory on a diskette or on the hard drive. At the prompt, type MD, and then the name of the directory. If a directory named PLC-5 is to be created, the following command is entered at the C: prompt:

MD PLC-5

When the ENTER key is pressed, a directory named PLC-5 is created on drive C. To verify that the directory was created, use the DIR command. At the C: prompt type:

DIR PLC-5

When the ENTER key is pressed, the directory named PLC-5 is displayed. If the directory was not created, the screen reads *file not found*.

To make a subdirectory of an existing directory, both the name of the directory and the subdirectory need to be entered. To create subdirectory NITROGEN under directory LAWN, type the following command line at the C: prompt:

MD LAWN\NITROGEN

When the ENTER key is pressed, a LAWN directory with a subdirectory NITROGEN is created. Notice that the directory name must be entered first, followed by the subdirectory name (names).

CHANGE DIRECTORIES COMMAND

To change from one directory to another, the change directory command (CD) is used. To move from the LAWN directory to the PLC-5 directory, type the following command at the C: prompt:

CD\PLC-5

When the ENTER key is pressed, the software changes from the LAWN directory to the PLC-5 directory.

RENAMING FILES COMMAND

The renaming files command (REN) is used when it is necessary to rename an existing file. This is done to further clarify the contents of the file, or to avoid duplication. If a new file is given the same name as an existing file, DOS destroys the previous file and creates a new one with the same name. To rename file 10-20-10 to FERT.A, the following command line is typed at the C: prompt:

REN 10-20-10 FERT.A

This command renames, or changes, the file from 10-20-10 to FERT.A.

DELETE FILES COMMAND

When it is necessary to delete or erase old files to make room for new files, or to simply delete files that are no longer needed, the delete command (DEL) is used. To delete the FERT.A file that was renamed in the previous example, the following command is entered at the C: prompt:

DEL FERT.A

When the ENTER key is pressed, the FERT.A file is deleted.

REMOVE DIRECTORY COMMAND

To remove an existing directory, the remove directory command (RD) is used. To remove the PLC-5 directory created earlier, type the following command line at the C: prompt:

RD PLC-5

Directories cannot be removed or deleted until all the subdirectories and files have been deleted.

DIRECTORY TREE COMMAND

It is not uncommon to forget what directories, subdirectories, or files are on a given disk or hard drive. The **tree command** is used to provide a list of all directories and subdirectories, and to display their organization. The tree is similar to a family tree in structure. To list the directories and their subdirectories, type the following command at the C: prompt:

TREE

When the ENTER key is pressed, the computer displays all directories and subdirectories. Figure 19–5 shows the results of the tree command.

```
C:\ > TREE
Directory PATH Listing
Volume Serial Number is 2A4C-11DF
C:
├─DOS
├─WINDOWS
│  ├─SYSTEM
│  └─LETTERS
└─LAWN
   ├─NITROGEN
   └─POTASH
```

Figure 19–5 Directory and Subdirectory Tree

To list the directories, subdirectories, and files, a /F is added after the tree command and is typed at the C: prompt:

TREE /F

This command produces the tree shown in Figure 19–6.

```
C:\ > TREE/F
Directory PATH Listing
Volume Serial Number is 2A4C-11DF
C:
└─── LAWN
     ├─ NITROGEN
     │       60-10-10
     │       40-10-20
     │       20-60-20
     └─ POTASH
             10-20-10
```

Figure 19–6 Tree with Directories, Subdirectories, and Files

The tree command usually produces more information than can be shown on one screen, and causes the screen to scroll. To stop the scrolling, use the PAUSE/BREAK key.

To view only one directory and its subdirectories, a different command line is used. To view the LAWN directory and its subdirectories, type the following command line at the C: prompt:

TREE LAWN

When the ENTER key is pressed, the tree for the LAWN directory is displayed as shown in Figure 19–7.

```
C:\ > TREE LAWN
Directory PATH Listing
Volume Serial Number is 2A4C-11DF
C: LAWN
   ├─ NITROGEN
   └─ POTASH
```

Figure 19–7 Tree for LAWN Directory

To look at the directory, its subdirectories, and its files, type the following command line at the C: prompt:

TREE LAWN/F

The results of this command are shown in Figure 19–8.

```
C:\ > TREE LAWN/F
Directory PATH Listing
Volume Serial Number is 2A4C-11DF
C: LAWN
├─── NITROGEN
│       60-10-10
│       40-10-20
│       20-60-20
└─── POTASH
        10-20-10
```

Figure 19–8 Tree Command for Directory, Subdirectory, and Files

To print a copy of the tree, the print command (PRN) is used in conjunction with a greater than (>) symbol. To print a copy of the tree on drive C, type the following command line at the C: prompt:

TREE > PRN

It is not the intention of this chapter to provide the reader with all of the possible DOS commands, but instead to introduce him or her to the basic commands that are needed to get started using DOS and PLC software packages. Additional information about DOS can be obtained from the many books and software programs that teach the concepts and application of DOS commands. These software programs are referred to as tutorials, and can be bought from any retailer that carries computers and software packages.

Chapter Summary

It is common practice to make backup copies of original software disks, to use the copies as working disks, and to make copies of files from the hard drive to a diskette to serve as a backup. It may also be necessary to copy files from one diskette to the hard drive, or to another diskette. Various DOS commands are used to rename files and directories, or to copy files and directories. When using directories and subdirectories, it is important to define the path so the computer will know where to find the file, subdirectory, or directory.

Review Questions

1. What does the acronym *DOS* stand for?
2. What does *upward compatibility* mean in the context of this chapter?
3. Define the term *booted*.
4. A subdirectory is sometimes called a _____.
5. The directory that contains subdirectories is sometimes called the _____.
6. What command is given at the prompt to look at the directories on the current drive?
7. What command is given to show the directories in the wide version?
8. What does the *break command* do?
9. What command is used to clear the screen?
10. Write the command line that is necessary to copy a file named DEMO from drive A to drive B. Assume the current drive is drive C.
11. Write the command line that is necessary to change the name of the file DEMO to TEST.
12. What is the function of the *tree command*?

CHAPTER

20 Start Up and Troubleshooting

Objectives

After completing this chapter, you should have the knowledge to:

- Understand start-up procedures listed in the manufacturers' literature.
- Explain how input devices are tested.
- Explain how to test output devices using a push button or other input device.
- Explain safety considerations when testing output devices.
- Describe how voltage readings are taken to check input and output modules.

START UP

Careful start-up procedures are necessary to prevent damage to the driven equipment and the programmable controller system, or, more importantly, injury to personnel.

Prior to beginning a system start-up procedure, it is important to check and verify that the system has been installed according to the manufacturers' specifications, and that the installation meets local, state, and national codes. Special attention should be given to system grounding.

Before applying power to the controller, complete the following steps:

Step 1. Verify that the incoming power matches the jumper selected voltage setting of the power supply. Figure 20–1 shows a typical AC power terminal strip with the jumper position indicated at the sides of the strip.

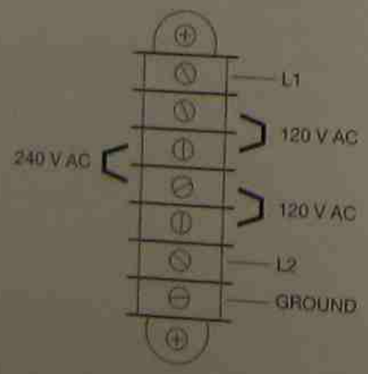


Figure 20–1 Power Supply Terminal Strip With Jumper Positions Indicated

Note: *For 120 V AC, the neutral conductor is connected to the L2 terminal, and the 240 V jumper is removed.*

Step 2. Verify that a hard-wired safety circuit or other redundant EMERGENCY STOP device (described in chapters 2 and 11) has been installed and is in the open position.

Step 3. Check all power and communication cables to ensure that connector pins are straight, and not bent or pulled out.

Step 4. Connect all cables making sure that connectors are fully inserted into their sockets. Secure connectors as applicable.

Step 5. Ensure that all modules are securely held in the I/O rack, and that field wiring arms (if applicable) are fully seated and locked.

Step 6. Place the PLC processor key switch to a safe position as indicated in Figure 20–2.

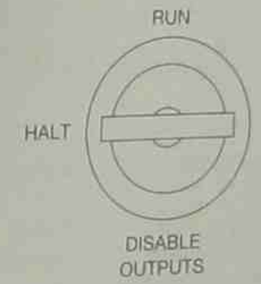


Figure 20–2 Key Switch in Halt Position

Step 7. Double check the setting(s) of all DIP switches.

Caution: Before proceeding further, make sure that the safety circuit or other EMERGENCY STOP devices are *OFF*, or open, and that power is removed from all **output** devices.

Apply power and observe processor indicator light(s) for proper indications.

When power is applied and the safety switch is closed, the power supply should provide the necessary DC voltage for the processor and I/O rack. If the proper voltage is present, the input indicator LEDs of the input modules will be functioning. Any input device that is closed, or *ON*, will have an illuminated LED (Figure 20–3).

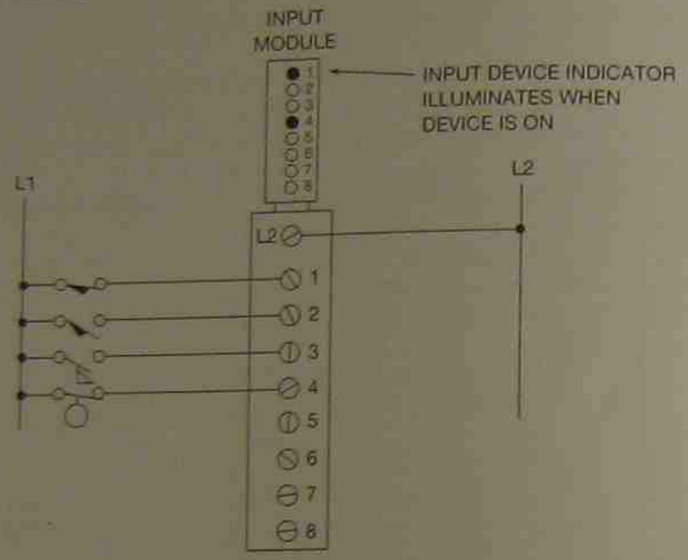


Figure 20–3 Input Module Indicators

TESTING INPUTS

Each input device can be manipulated to obtain open and closed contact conditions.

Caution: *Do not* activate the input devices mounted on equipment by hand, because unexpected machine motion could cause injury. Use a wooden stick or other nonconducting material to activate input devices mounted on equipment.

Each time an input device is closed, the corresponding LED on the input module should illuminate. Failure of a LED to illuminate indicates:

1. Improper input device operation.
2. Incomplete or incorrect wiring.
   a. Check to be sure that the input device is wired to the correct input module and proper terminal.
3. Loss of power to the input device.
4. Defective LED and/or input module.

To further check the system, a program can be developed and entered into the processor that uses each input device address. With the key switch in the test, or disable output position, the status of the input devices may now be monitored by using the VDT of a desktop or computer programmer, or by LED indicators on a hand-held programmer. On a VDT, the input contact becomes intensified or goes to reverse video, depending on the model PLC, when the instruction is true. An EXAMINE ON instruction intensifies or goes to reverse video when the input device was *ON*, or closed. An EXAMINE OFF instruction intensifies or shows reverse video when the input is *OFF*, or open. Figure 20–4 shows a Rung for testing input devices.

Figure 20–4 Rung for Testing Input Devices

Once all input devices have been tested and checked out as operational and properly terminated, the output devices can be tested.

TESTING OUTPUTS

Before testing output devices, it must be determined which devices can safely be activated and which devices should be disconnected from the power source. Figure 20–5 shows a motor starter with the motor disconnected for safety. In this configuration, the motor starter coil (the output device) is activated for checkout without energizing the motor. This prevents unwanted machine motion that might cause injury to personnel or damage the machine.

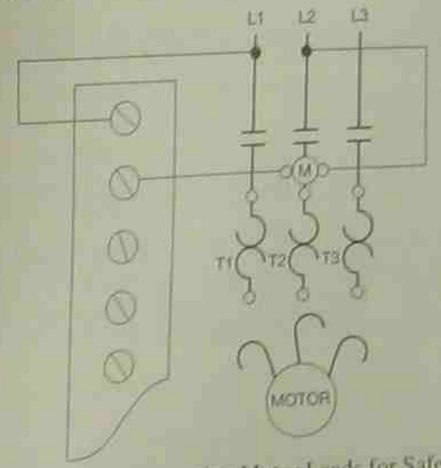


Figure 20–5 Disconnecting Motor Leads for Safety

For outputs that can be safely started, be sure equipment is in the start-up position, is properly lubricated, and is ready to run.

There are two methods used to test output devices. The first method uses a push button or other convenient input device that is part of the control panel. The push button is programmed to energize each output, one at a time.

The second method uses the FORCE function of the PLC to energize outputs, one at a time.

When using a pushbutton (or other input device), the address of the pushbutton is programmed in series with the output device to be tested (shown in Figure 20–6).

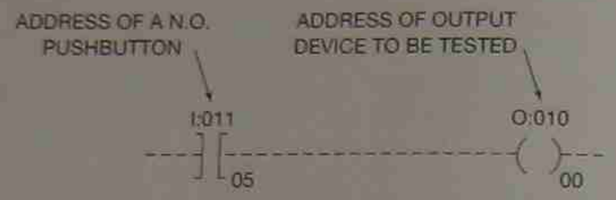


Figure 20–6 Rung for Testing Output Devices

Once the Rung has been programmed and entered into processor memory, the processor is placed in the *run* mode. Pressing the push button (I:011/05) illuminates the output indicator on the output module for address O:010/00. If there is an output connected, verify that the output device is energized. If the output indicator does not illuminate, using the VDT, verify that the input instruction I:011/05 is intensified or showing reverse video to indicate an *ON* condition. Double check the output address, and verify that the output instruction indicates an *ON* condition. If both instructions indicate an *ON* condition and the output address is correct, a defective module is likely the problem.

If the output module indicating LED is illuminated, but a connected output device does not energize, check the following:

1. *Wiring* to the output device.
2. *Operation* of the output device.
3. Proper *potential* to the output device.
4. Output device wired to correct output module and proper terminal.

A second method for testing output devices (for PLCs with the option) is the FORCE feature. The FORCE feature allows the user to turn an output device *ON* and *OFF* without using a push button or other contacts. Figure 20–7 shows the Allen-Bradley format for programming a Rung to test output address O:010/00 using the FORCE ON function.

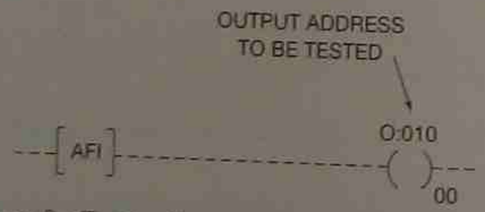


Figure 20–7 Rung for Testing Output Devices using FORCE ON Function

The Allen-Bradley ALWAYS FALSE instruction (AFI) is used to create an open condition in the Rung. This prevents the output from being unconditionally energized when the PLC is placed in the *run* mode.

Note: *For other PLC systems, an open instruction would be used to open the Rung.*

By moving the cursor to the output instruction, the FORCE ON function can be initiated, which should illuminate the output module indicating LED, and turn *ON* the output device, if one is connected. If the LED does not illuminate, and the output instruction is intensified, exchange the module. If the LED lights, but the output device does not energize, proceed as previously described.

FINAL SYSTEM CHECKOUT

After all input and output circuits have been tested and verified, the electrician or technician is ready for the final system checkout.

Reconnect any output loads (motors, solenoids, etc.) that were previously disconnected. In the case of motors, correct rotation needs to be established before the complete machine or process can be tested. Using a momentary push button that is part of the control panel (or using one installed specifically for this purpose), load a Rung of logic into the processor as previously discussed for testing outputs (Figure 20–6).

Caution: Because this part of the test causes machine motion, make sure the machine is operational and all personnel are in the clear. Station someone at the EMERGENCY STOP or disconnect location to deenergize the system, if necessary.

Close the push button and immediately release or open it. This momentary operation of the push button is called jogging or bumping, and allows the output (motor starter) to energize only momentarily. The motor starter is only energized long enough to determine the direction of rotation of the motor. If the rotation is wrong, reverse any two motor leads (assuming 3-phase power) and repeat the test for verification. Continue testing all output loads previously disconnected until all of them function correctly. Once all machine components are tested and correct rotation(s) are established, total machine operation testing can be accomplished.

For final system checkout, the following steps should apply:

Step 1. Place the processor in the *program* mode.
Step 2. Clear the memory of any previous rungs used for testing.
Step 3. Using a programming device, enter the program (ladder diagram) into memory.
Step 4. Place the processor in the *test* or *disable output* mode, depending on the PLC, and verify correctness of program.

Note: *In the* test, *or* disable output *mode, the outputs cannot be energized. All logic of the circuit is verified, input devices function, but no outputs come on.* This step must not be skipped if injury to personnel or damage to equipment is to be avoided.

Step 5. Once the circuit operation has been verified in the *test* or *disable output* mode, the processor can be placed in the *run* mode for final verification.
Step 6. Make changes to the program as required (timer settings, counter presets, and the like).
Step 7. Once the circuit is in final form, and the machine or process is running correctly, it is recommended that a copy of the program be made.

TROUBLESHOOTING

The key word to effective troubleshooting is *systematic*. To be a successful troubleshooter, the technician must use a *systematic* approach.

A systematic approach should consist of the following steps:
Step 1. Symptom recognition.
Step 2. Isolate the problem.
Step 3. Corrective action.

The electrician and/or technician should be aware of how the system normally functions if he or she expects to successfully troubleshoot the system. When prior knowledge of system operation is not possible, the next best source of information, if applicable, is the operator. Don't hesitate to ask the operator what the symptoms are and what he or she thinks the problem might be. If no operator is available, the next best source of information is the PLC system itself. Although the PLC can't talk, it can communicate in various ways to show what the problem is. There are status lights on the processor, power supply, and I/O rack that indicate proper operation, as well as status lights that alert the troubleshooter to the problem. The status lights of a typical processor with built-in power supply indicate:

1. DC POWER ON—If this LED is not lit, there is a fault in the DC power supply. Check the power supply fuse and/or incoming power.
2. MODE—Indicates which operating mode the processor is in (*run*, *halt*, *test*, *program*, etc.). The fault may simply be that the key switch is in the wrong position.
3. PROCESSOR FAULT—When this status light is on, it indicates a fault within the processor. This is a major fault, and requires changing the processor module.
4. MEMORY FAULT—This status light illuminates when a parity error exists in the transmission of data between the processor module and the memory module. Replace only one module at a time. If the first module does not correct the problem, reinstall the original module and then replace the second module. If replacing the second module doesn't clear the problem, replace both modules.
5. I/O FAULT—This light indicates a communication error between the processor and the I/O rack. Check that the communication cable(s) are fully inserted into their sockets. If available, connect a programming device with a VDT to the processor, and look for error codes and/or fault messages for further diagnostic assistance.

Note: *Refer to manufacturers' literature for explanation of error codes.*

6. STANDBY BATTERY LOW—When this LED is illuminated, the RAM backup batteries are low and need to be replaced. Although this is not a fault condition, failure to replace batteries results in losing the program when the system is shut down or a power failure occurs.

Status lights on the I/O modules also assist with troubleshooting problems that involve input and output devices.

If the operator confirms that the solenoid that activates a brake isn't working, the first step is to determine the address of the solenoid. Once the address is known, the programming device can be

used to ensure that the output circuit to the solenoid has been turned *ON*. Are all input devices closed that should be closed? Has the rest of the rung logic been completed? If the answers are yes, then it is necessary to determine the hardware location. (Some PLCs use the address to specify the hardware location, whereas others do not.)

Output modules have LED indicators that illuminate when each of the output circuits are turned *ON*. If the LED is lit for the location of the solenoid, it indicates that the problem is not with the output module, but with the circuit from the module to the solenoid, or with the solenoid itself.

A voltage check from L2 to the terminal as shown in Figure 20–8 verifies the conclusion that the module is working properly, but that the problem is either in the wiring to the solenoid, or in the solenoid itself. Further voltage checks will locate the problem.

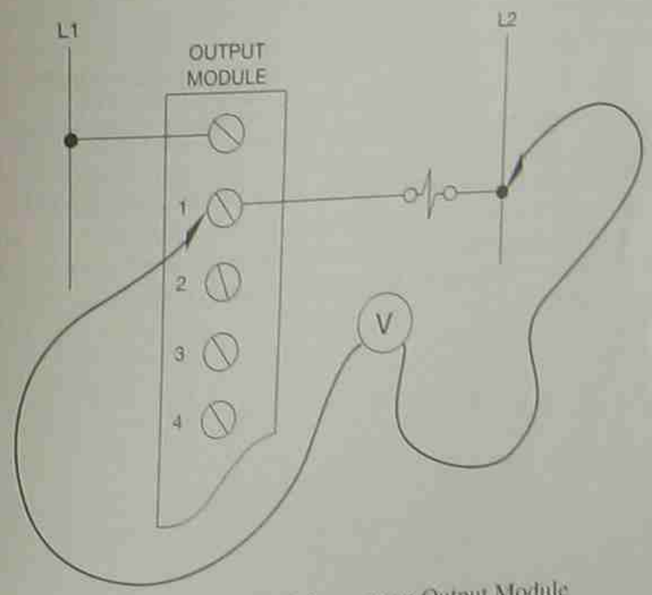


Figure 20–8 Testing Voltage on an Output Module

Note: *When testing AC output modules, the high internal resistance of most analog or digital meters act like a series voltage divider when measuring across an open load (Figure 20–9). The result is a reading of nearly full voltage even after the triac has been turned OFF. For accurate readings, a 10K ohm resistor can be placed in parallel with the meter leads as shown in Figure 20–10, or a solenoid-type tester (Wiggins) with low internal resistance can be used.*

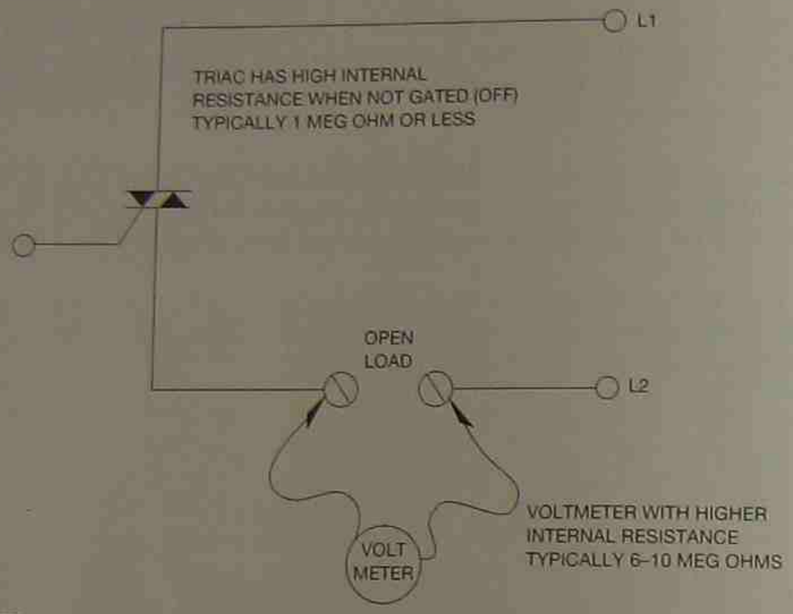


Figure 20–9 Series Voltage Divider Effect When Reading Across an Open Load

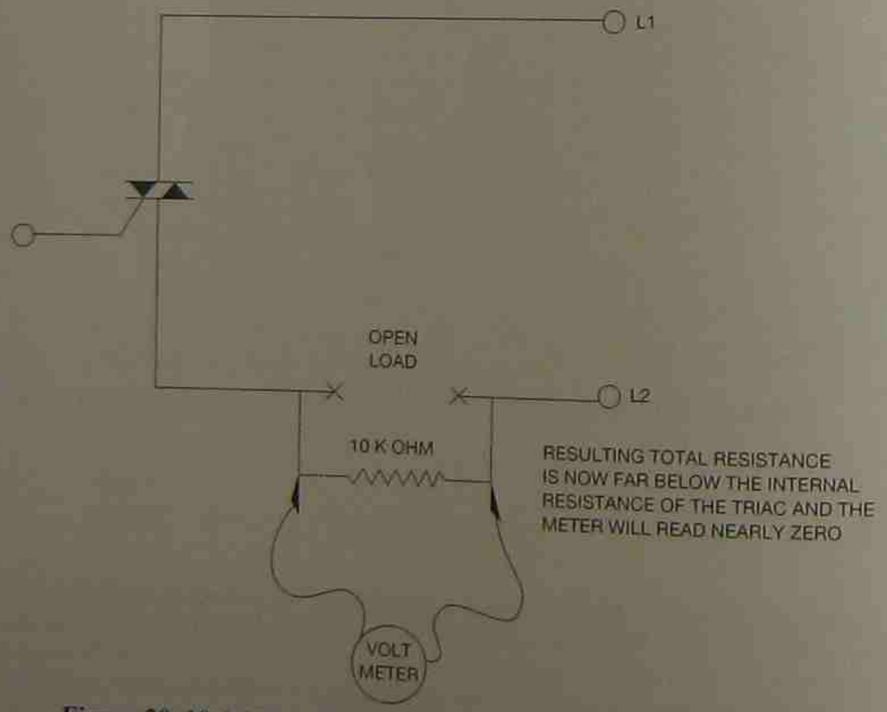


Figure 20–10 Adding a Resistor in Parallel with Meter to Reduce Voltage Divider Effect

If the indicator LED had not been lit, the first reaction might be to change the output module. Instead, look first for a blown-fuse indicator LED. A fuse might be all that's needed to make the solenoid operational. If, however, there was no blown fuse, replacing the output module should correct the problem.

Note: *Remove all power from the I/O rack before changing modules.*

Some PLC manufacturers offer deluxe output modules that have two indicator LEDs. One indicates that the logic from the processor has been received to turn on the output; the second LED comes on when the triac, or power transistor, has been turned *ON*. These two LEDs should come on simultaneously, unless the PLC is in the *test* mode. In the *test* mode, only the logic LED is lit because the output circuits are isolated, and kept from being turned *ON*.

Troubleshooting input modules follows the same basic procedure. If it was determined that the solenoid had not energized because limit switch 1 was not being shown closed on the programming device, the LS-1 address would need to be determined. From the address, determine the terminal on the input module that LS-1 is connected to.

An illuminated LED indicates that the limit switch is closed, but that the state of the switch (*ON*) is not being communicated to the processor. Exchanging the input module should make the system operational. However, had the indicator LED for LS-1 not been lit, there could be several other possible problems such as a bad limit switch, faulty wiring from LS-1 to the input module, or a bad input module. Closing the limit switch and taking a voltage check as shown in Figure 20–11 determines if the limit switch and associated wiring is operational.

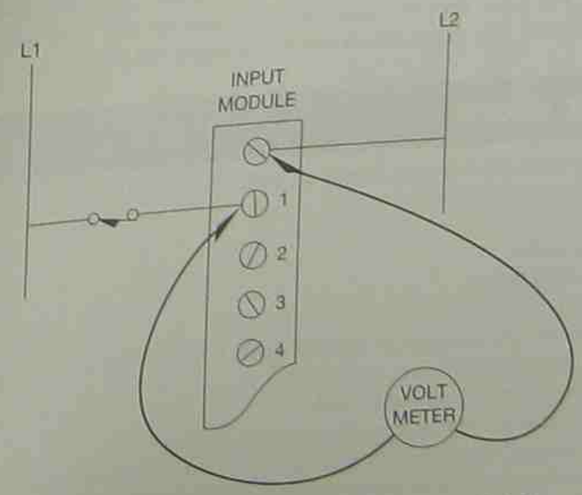


Figure 20–11 Testing Voltage on an Input Module

A voltage reading equal to the applied voltage indicates that the limit switch and wiring are operating, but that there is a faulty input module. No voltage reading indicates a problem with either the limit switch or its wiring. Further voltage checks will isolate the problem.

Similar to the deluxe output modules, there are also input modules that have two indicating LEDs. The first LED indicates that the input device has closed, and a voltage signal has been received by the input module; the second LED indicates that the status of the input device (*ON*) has been communicated to the processor.

Note: *Once the PLC program has been proven, or checked out, for complete and accurate operation, any future problems normally will be bad field wiring and/or bad field devices.*

Chapter Summary

Start-up procedures check each part of the PLC system for proper installation and operation. Safety must *always* be the overriding factor when testing or operating a PLC system. Care must be taken to prevent unexpected or incorrect machine motion if injury to personnel and/or damage to equipment is to be avoided.

Once the system start up is complete and the system is operational, problems or faults can occur. To successfully troubleshoot the system, a systematic approach must be used. This systematic approach includes recognizing the symptoms, isolating the problem, and taking corrective action. A variety of indicator and status lights, as well as error messages and fault codes, assist the electrician or technician in troubleshooting a given system.

The information covered in this chapter is intended to be general in nature and is not specific for any particular PLC. For more specific information on start-up and troubleshooting procedures, refer to the manufacturer's operating manual that accompanies the PLC.

Review Questions

1. A programmable controller system should be installed according to:
   a. manufacturer's specifications
   b. local electrical codes
   c. state electrical codes
   d. national electrical codes
   e. all of the above
2. Explain why a safety circuit or other EMERGENCY STOP devices are important.
3. Describe briefly how input devices are tested.
4. List two methods for testing output devices.
5. Why would output devices be disconnected before testing?
6. Draw an input module complete with input devices and power connected (L1-L2), and show how a voltage reading is taken to verify the operation of an input device.
7. On a deluxe input module, what do the two LED indicators represent?

8. On a deluxe output module, what do the two LED indicators, other than the blown-fuse indicator, represent?
9. Define the term *jogging*.
10. List the three steps of systematic troubleshooting.

GLOSSARY

ADDRESS: A location in processor memory.

ANALOG INPUT MODULE: A module that converts an analog input signal to a binary or BCD number for use by the processor.

ANALOG OUTPUT MODULE: A module that provides an output proportional to a binary or a BCD number provided to the module by the processor.

ANALOG SIGNAL: A continuous signal that depends directly on magnitude (voltage or current) to represent some condition. For example, a voltage might represent the speed of a motor (5 V corresponding to 200 rpm; 10 V corresponding to 400 rpm, etc.).

AND LOGIC: Logic that has a series relationship. Two devices that are in series would both have to be true to pass the logic. Device one AND two need to be true for the logic to pass.

ARITHMETIC CAPABILITY: The ability of a PLC to perform addition, subtraction, multiplication, division, and other math functions.

ASYNCHRONOUS SHIFT REGISTER: An instruction that shifts data one word at a time into a file or register.

ASCII: Acronym for **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. It is a seven- or eight-bit code for representing alphanumerics, punctuation marks, and certain special characters for control purposes.

BAUD: The seven or eight bits that make up a character. A character can be a letter, number, symbol, etc.

BAUD RATE: A unit of data transmission speed equal to the number of code elements (characters) per seconds. For example, 300 baud is thirty characters—letters, numbers, symbols—per second. 1200 baud is 120 characters per second.

BINARY CODED DECIMAL (BCD): One of several numbering systems used with PLCs. This unique numbering system uses four binary digits to represent each decimal digit from 0 to 9. Groups of four binary digits are grouped together to display decimal numbers. Twelve bits can represent a three-digit number. Sixteen bits are needed to represent a four-digit number.

BINARY: A numbering system that uses a base of two. There are two digits (1 and 0) in the binary system.

BIT: An acronym for **B**inary dig**IT**. A bit can be only one of two possible states: *ON* or *OFF*; high or low; logic 1 or logic 0; etc.

BOOLEAN ALGEBRA: Shorthand notation for expressing logic functions.

BOOLEAN EQUATION: Expression of relations between logic functions and/or elements.

BOOT: A term used in the computer world to indicate that a computer has been turned on, and the software has been loaded.

BRANCH: A parallel logic path within a user program rung.

BREAKDOWN VOLTAGE: The voltage at which a disruptive discharge takes place, either through or over the surface of insulation.

BUFFER: A temporary storage area where information is held while the printer or other device catches up with the transmission speed of the data.

BYTE: A sequence of binary digits usually operated upon as a unit (normally eight bits).

CASCADING: A programming technique that extends the ranges of timer and/or counter instructions beyond the maximum values that normally may be accumulated.

CASSETTE RECORDER: A peripheral device for transferring information between PLC memory and magnetic tape.

CENTRAL PROCESSING UNIT (CPU): Another term for processor.

CHARACTER: One symbol of a set of elementary symbols, such as a letter of the alphabet, a decimal numeral, a punctuation mark, etc.

CHILD: A term used to identify a subdirectory in a main directory.

CLOCK: A device (usually a pulse generator) that generates periodic signals for synchronization or timing.

CMOS: An acronym for **C**omplimentary **M**etal **O**xide **S**emiconductor. A family of very low power, high-speed integrated circuits.

CODE: A system of symbols (bits) for representing data (characters).

COMPARE FUNCTION: A program instruction that compares numerical values for "equal," "less than," "greater than," etc.

COMPATIBILITY: The ability of various specified units to replace one another, with little or no reduction in capability.

COMPUTER GRADE TAPE: A high quality magnetic digital recording tape which must be rated at 1600 FCI (flux changes per inch), or greater.

COMPUTER INTERFACE: A device designed for data communication between a central computer and another unit, such as a PLC processor.

CONTACT SYMBOLOGY DIAGRAM: Commonly referred to as a ladder diagram, it expresses the user-programmed logic of the controller in relay-equivalent symbols.

COUNTER: A device that can count up or down in response to transitions (*OFF* or *ON*) of an input signal and opens and/or closes contacts when a predetermined count is reached. Counters are internal to the processor and are not real-world devices.

CPU: An abbreviation for **C**entral **P**rocessing **U**nit. It is used interchangeably with processor.

CRT: The **C**athode **R**ay **T**ube, which is an electronic display tube similar to the familiar TV picture tube. It is more commonly called a **VDT**, or **V**ideo **D**isplay **T**erminal.

CURSOR: A means for indicating on a VDT screen the point at which data entry or editing is to occur.

DATA MANIPULATION: The process of altering and/or exchanging data between storage words.

DATA TRANSFER: The process of exchanging data between PLC memory words and/or areas.

DECREMENT: A term used with counters to indicate that the value of the counter has decreased. When a counter value goes from 4 to 3, it is said to have decremented by 1.

DEFAULT: The initial setting of a value, or the initial assignment of a file by the software. Default values may or may not be changed, depending on the software and the PLC manufacturer.

DEFAULT DRIVE: The drive that will be used if no other drive has been specified.

DIGITAL: The representation of numerical quantities by means of discrete numbers. It is possible to express in binary digital form all information stored, transferred, or processed by dual-state conditions, e.g., *ON/OFF*, open/closed, etc.

DIP Switch: Dual In-Line Package; multiple switches installed in a small device or package that are used to set PLC parameters.

DISCRETE INPUTS: An input that is either *ON* or *OFF*. Examples of discrete inputs are limit switches, push buttons, float switches, etc.

DISCRETE INPUT MODULE: A module that converts signals from real-world input devices to logic level signals for use by the processor.

DISCRETE OUTPUTS: An output that is either *ON* or *OFF*. Examples of discrete outputs are solenoids, motor starter coils, pilot lights, etc.

DISCRETE OUTPUT MODULE: A module that converts the logic levels of the processor to an output signal to control a real-world output.

DISKETTE: A magnetic medium for storing information that can later be read by the computer or PLC.

DOS®: Disk **O**perating **S**ystem. A registered trademark of the Microsoft Corporation. The full name is MS-DOS for **M**icro**S**oft **D**isk **O**perating **S**ystem.

DOUBLE PRECISION ARITHMETIC: Two words are used to store the results of arithmetic operations, thereby increasing the size of the value that can be stored. (See single precision arithmetic)

DUMP: A term used when information stored in memory is copied or recorded onto magnetic tape or disk.

DUPLEX: A means of two-way data communication. Also see **FULL DUPLEX** and **HALF DUPLEX**.

EAROM: A type of programmable memory that can be erased or altered electrically. The term stands for **E**lectrically **A**lterable **R**ead **O**nly **M**emory.

EEPROM or **E2PROM:** A memory chip that can be programmed using a standard programming device, and can be erased when the proper signal is applied to the erase pin. The initials stand for **E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory.

ELECTRICAL NOISE: Noise, or voltage spikes, that are generated whenever inductive loads such as relays, solenoids, motor starters, and motors are operated by "hard contacts" such as push buttons, selector switches, and relay contacts.

ELECTRICAL-OPTICAL ISOLATOR: A device that couples different voltage levels using a light source and detector in the same package. It is used to provide electrical isolation between line voltage input and output circuitry and the processor.

ELEMENT: A program instruction (N.O. contact, timer, counter, etc.) displayed on a VDT.

EMI: Electromagnetic Induction. The term used to describe electrical noise.

ENABLED: A term used to indicate that a function or operation has been activated.

EVEN PARITY: The condition that occurs when the sum of a string of binary digits, 1s and 0s, add up to an even number. Parity is used for error checking.

EXAMINE OFF: An EXAMINE OFF PLC instruction is a true precondition if its addressed bit is *OFF* (0). It is false if the bit is *ON*, or 1.

EXAMINE ON: An EXAMINE ON instruction is a true precondition if its addressed bit is *ON* (1). It is false if the bit is *OFF*, or 0.

FALSE: When relating to PLC instructions, an *OFF* state or condition.

FAULT: Any malfunction which interferes with normal operation.

FIFO: First In-First Out. A reference to the way that information is stored and removed from a file or register.

FILE: A group of words, usually consecutive, that are used to store information.

FORCE: A mode of operation or instruction that allows the operator (as opposed to the processor) to control the state of an input or output device.

FORCE OFF FUNCTION: A feature that allows the user to deenergize any input or output by means of the programmer, independent of the PLC program.

FORCE ON FUNCTION: A feature that allows the user to energize any input or output by means of the programmer, independent of the PLC program.

FORTRAN: An acronym for **FOR**mula **TRAN**slation, a scientific programming language.

FULL DUPLEX (FDX): A mode of communications in which data may be simultaneously transmitted and received by both ends (sender/receiver).

GROUND: A conducting connection, intentional or accidental, between an electric circuit or equipment chassis and the earth ground.

GROUND POTENTIAL: Zero voltage potential with respect to earth ground.

HALF DUPLEX (HDX): A mode of data transmission capable of communicating in two directions, but in only one direction at a time.

HARD CONTACTS: Any type of physical switch contacts. Contrasted with electronic switching devices, such as triacs and transistors.

HARD COPY: Any form of printed document such as a ladder diagram, program listing, data table configuration, etc.

HARDWARE: The mechanical, electrical, and electronic devices that comprise a programmable logic controller and its application.

HARDWARE KEY: A piece of hardware that is required for a program to run. It may be in the form of a plug or connector plugged into the printer port that allows the software to run. Without the hardware key installed, the software does not work, or limits the access to certain portions of the program.

HARD DRIVE: A storage system that consists of an inflexible (hard) disk, as opposed to a floppy disk, that is used to store files, directories, software programs, etc.

HARD-WIRED: Electrical devices interconnected through physical wiring.

HEAT SINK: Heat sinks work on convection and are used to dissipate the heat generated by electronic devices.

HEXADECIMAL: The numbering system that represents all possible statuses of four bits with sixteen unique digits (0–9 then A–F).

HIGH = TRUE: A signal type wherein the higher of two voltages indicates a logic state of 1, or *ON*. (See **LOW = TRUE**.)

HOLDING REGISTER: A register or file that holds a value or values for comparison or for use in a user program.

IEEE: An acronym for **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers.

IMAGE TABLE: An area in PLC memory dedicated to I/O data. Ones and zeros (1s and 0s) represent *ON* and *OFF* conditions, respectively. During every I/O scan, each input controls a bit in the *input image table*; and each output is controlled by a bit in the *output image table*.

INCREMENT: A term used with counters to indicate that the value has increased. When a counter has counted up from 3 to 4, it is said to have incremented by 1.

INPUT DEVICES: Devices such as limit switches, pressure switches, push buttons, etc., that supply data to a programmable logic controller. These real-world inputs are of two types: those with common returns, and those with individual returns (referred to as isolated inputs). Other inputs include analog devices and digital encoders.

INSTRUCTION: A command or order that causes a PLC to perform one certain prescribed operation.

INTERFACING: Interconnection of a PLC with its input and output devices and data terminals through various modules and cables. Interface modules convert PLC logic levels into external signal levels, and vice versa.

INTERPOSING RELAY: A relay that is added to a PLC circuit to handle current values larger than can be handled by one terminal of an output module.

INTERRUPTIBLE: Interruptible refers to a timer that can be interrupted, but still retain its accumulated time.

I/O: An abbreviation for **I**nput/**O**utput. For example, a group of input modules and output modules would be referred to as I/O modules.

I/O ELECTRICAL ISOLATION: Separation of the field-wiring circuits from the logic level circuits of the PLC. This is typically achieved using electrical-optical isolators mounted in the I/O module.

I/O MODULE: The printed circuit assembly that interfaces between the user devices and the PLC.

I/O RACK: A chassis which contains I/O modules.

I/O SCAN TIME: The time required for the PLC to monitor all inputs, read the user program, and control all outputs. The I/O scan repeats continuously.

I/O SECTION: Interfaces the different signals from real-world devices and sensors to signals the CPU can use.

JOGGING: The momentary operation of a motor to test rotation or to cause small movements of the driven equipment.

JUMPER: A short length of conductor used to make a connection between terminals, around a break in a circuit, or around a device.

KEYED: A term used with PLCs to indicate that a keying device has been installed to prevent I/O modules from being installed in the wrong slot.

KILO: A prefix used with units of measurement to designate quantities 1000 times as great (as in kilowatt). The exception to K having a value of 1000 is when referring to computer memory. Computer memory is counted using the binary numbering system and one kilo (or K) of memory is 1024, not 1000. (210 = 1024).

LADDER DIAGRAM: A complete control scheme normally drawn as a series of contacts and coils arranged between two vertical supply lines so that the horizontal lines of contacts appear similar to rungs of a ladder. A ladder diagram is normally the reference document used by the operator when entering the control program. (See **CONTACT SYMBOLOGY DIAGRAM**.)

LADDER DIAGRAM PROGRAMMING: A method of writing a user's PLC program in a format similar to a relay ladder diagram.

LANGUAGE: A set of symbols and rules for representing and communicating information (data) among people, or between people and machines.

LATCH: A device that continues to store the state of the input signal after the signal is removed. The input state is stored until the latch is reset.

LATCH INSTRUCTION: A PLC instruction which causes an output to stay *ON*, regardless of how briefly the instruction is enabled. (It can only be turned *OFF* by an UNLATCH INSTRUCTION in a separate Rung.)

LATCHING RELAY: A relay constructed so that it maintains a given position by mechanical means until released mechanically or electrically.

LEAST SIGNIFICANT DIGIT (LSD): The digit that represents the smallest value. In the number 102, the 2 is the least significant number, or digit.

LED: Acronym for **L**ight **E**miting **D**iode.

LIFO: Last In-First out. A reference to the way information is stored or removed from a file or register.

LIMIT SWITCH: A switch that is actuated by some part or motion of a machine or equipment to alter the electrical circuit associated with it.

LIQUID CRYSTAL DISPLAY (LCD): A reflective visual readout. Because its segments are displayed only by reflected light, it has extremely low power consumption, as contrasted with a LED display which emits light.

LOAD: 1) The power delivered to a machine or apparatus. 2) A device intentionally placed in a circuit or connected to a machine or apparatus to absorb power and convert it into the desired useful form. 3) To place data (e.g., a ladder diagram) into the processor's memory.

LOGIC LEVEL: The voltage magnitude associated with signal pulses representing ones and zeros (1s and 0s) in binary computation.

LOW = TRUE: A signal type wherein the lower of two voltages indicates a logic state of 1, or *ON*. (See **HIGH = TRUE**.)

MAGNETIC TAPE: Tape made of plastic and coated with magnetic material; used to store

information.

MALFUNCTION: Any incorrect functioning within electronic, electrical, or mechanical hardware. (See **FAULT**.)

MANIPULATION: The process of controlling bits or words within a program to obtain the required program outcomes.

MASK: Bits in a word that are used to prevent other bits in a different word from being used. If there is a 1 in the bit location of the mask, the corresponding bit in the output word is enabled and can be turned *ON* and *OFF*. If the bit is set to 0, the corresponding bit in the output word is disabled, and does not allow the output to be turned *ON*, even if the program called for the bit to be turned *ON*.

MASK WORD: A word used to mask or selectively screen out data or bits of a word of memory.

MCR: Master Control Relay is used to disconnect all power to a PLC system.

MECHANICAL DRUM CONTROLLER: A type of SEQUENCER which operates switches by means of pins or cams placed on a rotating drum. The switch sequence may be altered by changing the pin or cam pattern.

MEMORY: The section of the programmable logic controller that stores the user program and other data. The storage may be either temporary or semipermanent.

MEMORY PROTECT: The hardware capability to prevent a portion of the memory from being altered by an external device. This hardware feature can be under actual key and lock control, or may use passwords that are referred to as software keys.

MENU: A display on the computer or PLC monitor that offers options or gives the operator choices to select from.

MIDDLE DIGIT (MD): The middle digit of a three digit number.

MILLIAMPERE (mA): One thousandth of an ampere: 10-3 or 0.001 ampere.

MILLISECOND (ms): One thousandth of a second; 10-3 or 0.001 second.

MINI-PLC: A scaled-down version of a standard PLC with small I/O capability.

MNEMONIC: A shorthand notation used with PLC instructions such as OTE, the mnemonic for Output Energized, BST, for Branch Start, and so forth.

MODE: A selected method of operation (for example, *run*, *test*, or *program*.)

MODEM: Acronym for **MO**dulator/**DEM**odulator. A device used to transmit and receive data by frequency-shift-keying (FSK). It converts FSK tones into their digital equivalents, and vice versa.

MODULE: An interchangeable "plug-in" item containing electronic components which may be combined with other interchangeable items to form a complete unit.

MOST SIGNIFICANT DIGIT (MSD): The digit representing the greatest value. In the number 102, the 1 is the most significant number, or digit.

MOTOR CONTROLLER: A device, or group of devices, that serves to govern, in a predetermined manner, the electrical power delivered to a motor.

NAND LOGIC: NAND Logic is a combination of a NOT gate and an AND gate. By placing the invert, or NOT, symbol at the output of the AND gate the output can only be true when one or both of the inputs are false, or set to 0.

NEMA STANDARDS: Consensus standards for electrical equipment approved by the majority of the members of the National Electrical Manufacturers Association.

NESTING: A programming technique that has a "branch-within-a-branch." Depending on the PLC manufacturer, nesting may or may not be allowed.

NETWORK: A group of connected logic elements used to perform a specific function. A network can range from one element to a complete matrix of elements, plus coil(s) as desired by the user. The size and configuration of the matrix or rungs varies with PLC manufacturers.

NODE: A common connection point between two or more contacts or elements in a circuit.

NOISE: Extraneous signals; any disturbance which causes interference with the desired signal or operation.

NOISE SPIKE: Voltage or current surge produced in the industrial operating environment.

NONVOLATILE MEMORY: A memory that is designed to retain its information even though its power supply is turned off.

NOR LOGIC: NOR logic is the combination of a NOT gate and OR gate. The NOR gate will only be logically true when both inputs, NOT and OR are false, or set to 0.

NOT LOGIC: The NOT gate, often referred to as the inverter, will only have one input lead and one output lead. If the input is OFF, or set to 0, then the output will be ON, or set to 1. If the input is ON, or set to 1, then the output will be OFF, or set to 0.

OCTAL NUMBERING SYSTEM: A numbering system that uses a base eight. Only the digits 0 through 7 are used.

ODD PARITY: The condition that occurs when the sum of 1s and 0s in a binary word is an odd number. Parity is used for error checking.

OFF DELAY TIMER: 1) In relay panel application, a device in which the timing period is initiated upon deenergization of its coil. 2) In a PLC, an instruction which starts the delay whenever the timer rung goes false.

OFF-LINE PROGRAMMING: A method of programming that is done while the processor is not communicating with the outputs.

OFF DELAY TIMER: 1) In relay panel application, a device in which the timing period is initiated upon deenergization of its coil. 2) In a PLC, an instruction which starts the delay whenever the timer rung goes false.

OFF-LINE PROGRAMMING: A method of programming that is done while the processor is not communicating with the outputs.

ON DELAY TIMER: 1) In relay panel applications, a device in which the timing period is initiated upon energization of its coil. 2) In a PLC, an instruction which starts the delay whenever the timer rung goes true.

ON-LINE OPERATION: Operations where the programmable logic controller is directly controlling the machine or process.

ON-LINE PROGRAMMING: A method of programming by which rungs in the program may be inserted, changed, or deleted while the processor is running and controlling the process equipment.

OPERAND: 1) Either of the two numbers used in a basic computation to produce an answer. For example, in the computation 2 X 3 = 6, 2 and 3 are the operands. 2) Data required for the operation of a special function.

OPTICALLY COUPLED/OPTICAL ISOLATION: The use of a light emitting diode and a photo transistor to communicate a signal or state to the processor. Optical coupling is used in input and output modules to isolate the logic level signal for line voltage sources.

OR LOGIC: Logic that has a "parallel" relationship. When two devices are in parallel, if either device one OR device two is true, the logic passes.

OUTPUT CIRCUIT: An output module point, real-world device (e.g., motor starter, digital readout, solenoid, etc.), and its associated wiring. The output module's function is to convert processor signal levels to field voltage levels necessary to control the real-world devices.

OUTPUT DEVICES: Devices such as solenoids, motor starters, etc., that receive data (control) from the programmable logic controller.

OUTPUT SIGNAL: A signal provided by the processor to the real-world output devices that control their status (*ON* or *OFF*).

OVERLOAD: A load greater than that which a device is designed to handle.

PARALLEL COMMUNICATIONS: A type of communication or information transfer whereby a group of digits (bytes) are transmitted simultaneously. This is different from serial communications where the data bits are transmitted one at a time—sequentially—in a string.

PARENT: A name given to a directory that has subdirectories.

PARITY: A method of testing the accuracy of binary numbers used in recorded, transmitted, or received data.

PARITY BIT: An additional bit added to a binary word to make the sum of the number of 1s in a word always even or odd.

PARITY CHECK: A check that tests whether the number of 1s in an array of binary digits is odd or even.

PASSWORD: A word used to gain access to a program or process. A password serves the same function as a hardware key, except that it is not a piece of hardware, but rather a word that when entered at the keyboard, gains access for the user to the program.

PC: Abbreviation for Personal Computer (also used as an abbreviation for Programmable Controller). To avoid the confusion caused by using the same abbreviations for two different types of systems, the programmable logic controller is now most often referred to as a PLC.

PLC: See PROGRAMMABLE LOGIC CONTROLLER.

PILOT DEVICE: A device used in a circuit that performs a control function only. Pilot devices are limited to 10 amps of current carrying capacity, and are to be used in control circuits only. They are not designed to control the power and current required by the operating equipment.

POWER SUPPLY: Supplies the DC power for the CPU and for the I/O section. The voltage is typically +5 volts. The power supply can be internal with the processor, rack mounted, or externally mounted as a separate unit. A separate power supply is required if DC voltage is required for the actual input and/or output devices.

PRIORITY: Order of importance.

PROCESSOR/PROCESSOR UNIT: The part of the programmable logic controller that performs logic solving, program storage, and special functions within a PLC system. It scans all the inputs and outputs in a predetermined order. The processor monitors the status of the inputs and outputs in response to the user programmed instructions, and energizes or deenergizes outputs as a result of the logical comparisons made through these instructions.

PROGRAM: A sequence of instructions executed by the processor to control a machine or process.

PROGRAM PANEL (PROGRAMMER): A device for inserting, monitoring, and editing a program in a programmable logic controller.

PROGRAM SCAN TIME: The time required for the processor to execute all instructions in the program one time.

PROGRAMMABLE LOGIC CONTROLLER (PLC): A solid-state control system that has a user programmable memory for storage of instructions to implement specific functions such as I/O control logic, timing, counting, arithmetic, and data manipulation. A PLC consists of the processor, input/output interface, memory, and programming device that typically uses relay-equivalent symbols. The PLC is purposely designed as an industrial control system to perform functions equivalent to a relay panel or a wired solid-state logic control system.

PROGRAMMER: A device that is needed to enter, modify, and troubleshoot the PLC program, and check the condition of the processor. The programmer may be hand-held, dedicated desktop-type, or a personal computer.

PROM: Acronym for **P**rogrammable **R**ead **O**nly **M**emory. A type of read only memory that requires an electrical operation to generate the desired bit or word pattern. In use, bits or words can be accessed on demand, but cannot be changed.

PROTECTED MEMORY: Storage (memory) locations reserved for special purposes or use by the processor into which data cannot be entered directly by the user.

PROTOCOL: A defined means of establishing criteria for receiving and transmitting data through communication channels.

RACK: A PLC chassis that contains modules. Some PLC manufacturers, like Allen-Bradley, use the term rack to indicate a given number of I/O points rather than to identify a specific piece of hardware. In the Allen-Bradley scheme, a chassis could contain a number of racks. With most other manufacturers, however, rack and chassis are used interchangeably, and mean the hardware that holds the various modules, power supplies, etc.

RAM: Acronym for **R**andom **A**ccess **M**emory. Random access memory is a type of memory that can be read from (accessed) or written into by the user.

RANDOM ACCESS: See **RAM.**

RATED VOLTAGE: The maximum voltage at which an electrical component can operate for extended periods without damage or undue degradation.

READ/WRITE MEMORY: A memory into which data can be placed (*write* mode) or accessed (*read* mode). The *write* mode destroys previous data; the *read* mode does not alter stored data.

REGISTER: A word or group of words used to store numerical values.

REPORT: A display of data, or a printout, containing data and/or information that is useful to the user or operator. Reports can include operator messages, part records, production lists, etc. Reports are normally stored in a memory area separate from the user's program.

REPORT GENERATION: The printing or displaying of user-formatted application data by means of a programming device. Report generation can be initiated by means of either the user's program or a programming device keyboard.

RETENTIVE: To retain a value or time.

RETENTIVE OUTPUT: An output that remains in its last state (*ON* or *OFF*), depending on which of its two program Rungs (one containing a LATCH INSTRUCTION, the other an UNLATCH INSTRUCTION) was the last to be true. The retentive output remains in its last state when both Rungs are false. It also remains in its last state if power is removed from, then restored to, the PLC.

RETENTIVE TIMER: A PLC instruction which accumulates the amount of time, whether continuous or not, when the preconditions of its rung are true, and which controls one or more outputs after the total accumulated time is equal to the preset time. When the rung is false, the accumulated time is retained. Moreover, if the outputs have been energized, they remain *ON*. Additionally, the accumulated time and energized outputs are retained if power is removed from, then restored to, the PLC.

ROM: Acronym for **R**ead **O**nly **M**emory. A read only memory is a solid-state digital storage memory whose contents cannot be altered by the user.

RS-232C: An Electronic Industries Association (EIA) standard for data transfer and communication.

RTD: Resistance Temperature Detector, or temperature probes that can be connected to a special PLC input module to indicate temperature reading.

RUNG: A grouping of PLC instructions that control one output or storage bit. Some PLCs can have multiple outputs on the same rung. A rung is also referred to as a network.

SCAN: The time required to make one complete scan through memory and update the status of all inputs and outputs.

SCHEMATIC: A diagram of a circuit in which symbols illustrate circuit components.

SCR: An acronym for **S**ilicon **C**ontrolled **R**ectifier. The SCR is used to convert AC current to DC current.

SEQUENCER: A controller which operates an application through a fixed sequence of events.

SERIAL COMMUNICATION: A type of communication or information transfer within a programmable logic controller whereby the bits are handled sequentially rather than simultaneously as they are in parallel communications. Serial operation is slower than parallel operation for equivalent clock rate. However, only one channel is required for serial operation.

SHIELDING: The practice of confining the electrical field around a conductor to the primary insulation of the cable by putting a conducting layer around the cable insulation.

SIGNED BIT: A way to indicate that a number is negative or positive.

SINGLE PRECISION ARITHMETIC: Only one word is used to store the results of math or arithmetic operations. Single precision arithmetic is limited to a maximum value of 999. (See double precision arithmetic.)

SOFTWARE: The manufacturers' program that controls the operation of a programmable logic controller.

SOLID-STATE: Circuitry designed using only integrated circuits, transistor, diodes, etc.; no electromechanical devices such as relays are utilized. High reliability is obtained with solid-state logic.

SOLID-STATE DEVICES (SEMICONDUCTORS): Electronic components that control electron flow through solid materials (e.g., transistors, diodes, integrated circuits, etc.).

STATE: The logic condition, 1 or 0, in PLC memory, or at a circuit's input or output.

STORAGE: Synonymous with MEMORY.

STORAGE MEMORY: That part of the memory that stores the status of the input and output devices, numeric values for timers and counters, numeric values for arithmetic functions, status of internal relays, and information stored in holding and storage registers.

SURGE: A transient variation in the current and/or voltage at a point in the circuit.

SWITCHING: The action of turning a device *ON* and *OFF*.

SYMBOLIC NAME: A user designation for an application I/O device (e.g., S-1, LS-4, or SOL-7).

SYNCHRONOUS SHIFT REGISTER: An instruction that shifts information one bit at a time within a word or from one word to another.

SYSTEM PROMPT: The system prompt indicates the current drive for the computer. If the prompt is C:\, then the current drive is C drive.

THUMBWHEEL SWITCH: A rotating numeric switch used to input numeric information to a controller.

TIMER: In relay-panel hardware, an electromechanical device that can be wired and preset to control the operating interval of other devices. In a PLC, a timer is internal to the PROCESSOR, meaning that it does not exist in the real world, but can be controlled by a user-programmed instruction. A timer instruction has greater accuracy and timing range than a hardware timer.

TOGGLE SWITCH: A panel-mounted switch normally used for *ON* or *OFF* switching.

TRANSFORMER COUPLING: One method of isolating I/O devices from the controller.

TREE: A command that is used to display the organization of all the directories, subdirectories, and files on a given disk or hard drive.

TRIAC: A solid-state component capable of switching alternating current.

TRUE: As related to a PLC instruction, an enabling logic state or *ON* condition. (See FALSE.)

TRUTH TABLE: A matrix which shows all the possible states (*ON* or *OFF*) of a single input device or combination of input devices, and the corresponding state (*ON* or *OFF*) of the output device.

TTL: Abbreviation for **T**ransistor-**T**ransistor **L**ogic, a family of integrated circuit logic. (Usually 5 volts is high, or 1, and 0 volts is low, or 0.)

TUTORIALS: Text included in software that provides the user with helpful information on how the software works or how to use functions of the software. Typically can be accessed by using the Help option.

TWOS COMPLEMENT: A convention for binary representation of negative and positive decimal numbers.

UPDATING: A term used to indicate that the processor has scanned and checked the status of all input and output devices. After the status of the inputs and outputs are known, the data table is updated to reflect the current status.

UPWARD COMPATIBILITY: The ability of a new version of software to support previous editions of the same software. If version 6.6 of a particular software allows documents and files from previous versions (5.2 and 6.0) to be read, the software is said to have upward compatibility.

UNLATCH INSTRUCTION: A PLC instruction which causes an output to unlatch, or turn *OFF*, regardless of how briefly the instruction is enabled. (It can only be turned back *ON* by a LATCH INSTRUCTION in a separate rung.)

UNCONDITIONAL: A term applied to an output (or other instruction) that is always true.

USER MEMORY: The portion of memory that is set aside for the storage of the user program (i.e., ladder diagrams, program messages, etc.).

UV ERASABLE PROM: An erasable programmable read only memory which can be erased or cleared (set to 0) by exposure to intense ultraviolet light. After being cleared, it may be reprogrammed.

VALUE: 1) A number that represents a computed or assigned quantity. 2) A number contained in a register or file word.

VDT: Video Display Terminal, the TV-like screen that displays the PLC programs, files, and so on.

VOLATILE MEMORY: A memory that loses its information if the power is removed.

WATCHDOG TIMER: A timer that is used within the PLC processor to verify that the program scan has been completed correctly in the allotted amount of time. If there is a program error, the scan is not completed in the prescribed amount of time, and the watchdog timer times out and indicates that there is a problem with the circuit.

WORD: A grouping, or a number of bits, in a sequence that is treated as a unit.

WRITING OVER: A term used to indicate that information will be replaced (or the existing information will be written over) by new information.

XOR LOGIC: The XOR logic gate will only turn the output ON when either input A or B is ON–but *not* both ON.

INDEX