numa·logic

HPPC-1500/1700
Systems Manual Supplement

REFER TO THE INSIDE REAR COVER FOR A QUICK-LOCATOR LISTING OF THE ADVANCED PROGRAMMABLE FUNCTIONS.

STANDARD FUNCTIONS QUICK-LOCATOR LIST

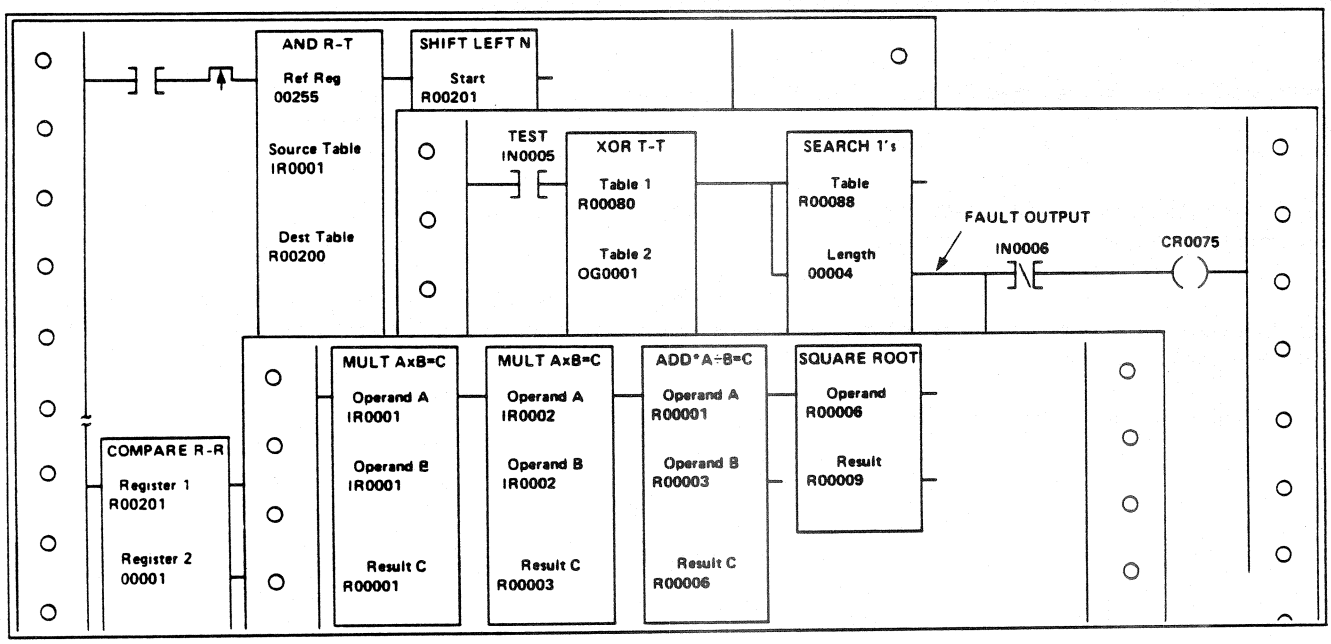
Type	Display No.	Title	Par. No.	Page No.
Coils	001	Bit Set	①	6-16
	002	Bit Clear	①	6-16
	003	Bit Follow	①	6-15
	004	Skip Coil	7-2	7-4
	005	MCR Coil	7-3	7-7
	011	CR Coil	①	6-13
	012	Latch Coil	①	6-18
Signal Conditioners	007	Transitional	7-4	7-10
Labels	021	MCR Label	7-3	7-7
	022	Skip Label	7-2	7-4
Timers	028	Time On 1.0	7-5	7-13
	029	Time On 0.1	7-5	7-13
	030	Time On .01	7-5	7-13
	034	One Shot 1.0	7-5	7-14
	035	One Shot 0.1	7-5	7-14
	036	One Shot .01	7-5	7-14
	037	One Shot SCN	7-5	7-14
Counters	038	Up Counter	7-6	7-18
	039	Down Counter	7-6	7-18
Math Functions	068	Add A + B=C	7-7	7-23
	070	Sub A - B=C	7-7	7-23
	072	Mult A x B=C	7-8	7-27
	073	Div A ÷ B=C	7-9	7-30
Shift Registers	040	Shift Left 1	7-10	7-33
	041	Shift Right 1	7-10	7-33
Conversions	045	BIN-BCD R	7-11	7-39
	046	BIN-BCD*R	7-12	7-42
	047	BCD-BIN R	7-13	7-45
	048	Analog In	7-14	7-48
	049	Analog Out	7-15	7-51
Move Functions	050	Move R-R	7-16	7-53
	096	Drum Control	7-17	7-56
Logic Functions	051	Zero Table	7-18	7-62
Table Operations	024	Diagnostic	7-19	7-64
Comparison	053	Compare R-R	7-20	7-67
I/O Functions	054	Update I/O	7-21	7-69

① Described throughout Paragraph 6-7.

numa·logic

HPPC-1500/1700
Programmable Controller

Systems Manual Supplement



Westinghouse Electric Corporation
Industry Electronics Division
1521 Avis Drive
Madison Heights, MI 48071

January, 1986

WARNING

THIS EQUIPMENT HAS NOT BEEN TESTED TO SHOW COMPLIANCE WITH NEW FCC RULES (47 CFR, PART 15) DESIGNED TO LIMIT INTERFERENCE TO RADIO AND TV RECEPTION. OPERATION OF THIS EQUIPMENT IN A RESIDENTIAL AREA IS LIKELY TO CAUSE UNACCEPTABLE INTERFERENCE TO RADIO COMMUNICATION, REQUIRING THE OPERATOR TO TAKE WHATEVER STEPS ARE NECESSARY TO CORRECT THE INTERFERENCE.

Since the equipment explained in this manual has a variety of uses, the user and those responsible for applying this equipment must satisfy themselves as to the acceptability of each application and use of the equipment. Under no circumstances will Westinghouse Electric Corporation be responsible or liable for any damage, including indirect or consequential losses resulting from the use, misuse, or application of this equipment.

The text, illustrations, charts, and examples included in this manual are intended solely to explain the use and application of the **Numa-Logic** PC-1500/-1700 Programmable Controllers. Due to the many variables associated with specific uses or applications, Westinghouse Electric Corporation **cannot** assume responsibility or liability for actual use based upon the data provided in this manual.

No patent liability is assumed by Westinghouse Electric Corporation with respect to the use of circuits, information, equipment, or software described in this manual.

This manual is printed in the U.S.A. and is subject to change without notice.

THUMB TAB INDEX

Table of Contents

List of Figures

List of Tables

Introduction to the Manual

INTRODUCTION

The purpose of a separate HPPC-1500/-1700 Systems Manual Supplement is to consolidate programmable function information in an easy-to-use volume designed especially for programmers. At the same time, the material in the Supplement is an inherent part of the basic Systems Manual. Thus paragraph, figure, and table numbering in both manuals are directly related.

At times cross references in one manual may refer to paragraphs or figures in the other. Since there is no duplication in the numbering system, this should cause no confusion.

The Sections contained in this manual are:

- 6 – General Programming Concepts
- 7 – Standard Functions
- 8 – Advanced Functions
- 10 – Programming Considerations

AVAILABLE DOCUMENTATION

Westinghouse provides a copy of the HPPC-1500/-1700 Systems Manual Supplement (NLAM-B807), as standard, with each HPPC-1500/-1700 mainframe chassis. In addition, a copy of the HPPC-1500/-1700 Systems Manual (NLAM-B821) is also provided.

When the optional Software Programming Package (NLSW-1581/-1581H) is ordered, a copy of the HPPC-1500/-1700 APL Programming Manual (NLAM-B806) is included as standard. This document explains the use of the Advanced Program Loader.

These 3 manuals are the basic user documentation set required to plan, install, start up, program, and troubleshoot an HPPC-1500/-1700 system.

Additional copies of these manuals are optionally available from Westinghouse. Contact your local Sales Representative. Specify the manual's Catalog Number to avoid confusion.

Section 6

General Programming Concepts

6-1. GENERAL DESCRIPTION

Programming the Numa-Logic HPPC-1500/-1700 programmable controller system is simply a way to specify a sequence of operations the application is to perform. (The processor, through programming, digitally duplicates the operation of a relay control system. The program consists of a set of programmable functions which specify the relay circuits and devices to be duplicated and, also, the operational sequence to be performed.)

In an application, these programmed functions together form a control program, which is called a ladder diagram. (These ladder diagrams are similar to the ladder diagrams used in relay control systems.)

The ladder diagram clearly shows the types of contacts used in each of the program's control circuits. It also shows how these user-selected contacts are configured to cause an assigned device to operate in the desired manner. A properly constructed ladder diagram simplifies the entry of circuits into the processor and makes the function of these circuits easier to understand.

This Section describes how to construct a ladder diagram containing coils and contacts. A description of each programmable function is given in Sections 7 and 8.

6-2. LADDER DIAGRAMS

The HPPC-1500/-1700 system uses ladder diagrams to document a control program. The diagrams differ from conventional relay control system ladder diagrams in 4 major ways:

1. Circuits are redrawn, when necessary, to ensure compatibility with the scanning process used by the programmable controller.

2. Circuits are redrawn to make contact connections easier to recognize.

3. Circuits are redrawn, when necessary, to fit within the allowed contact display area or "field."

4. Each contact and coil is assigned a special "label." (Label is a technical term explained later in this Section.)

In these ladder diagrams, input devices (e.g., pushbuttons, limit switches, pressure switches, etc.) are shown as relay contacts. Figure 6-1 illustrates this basic

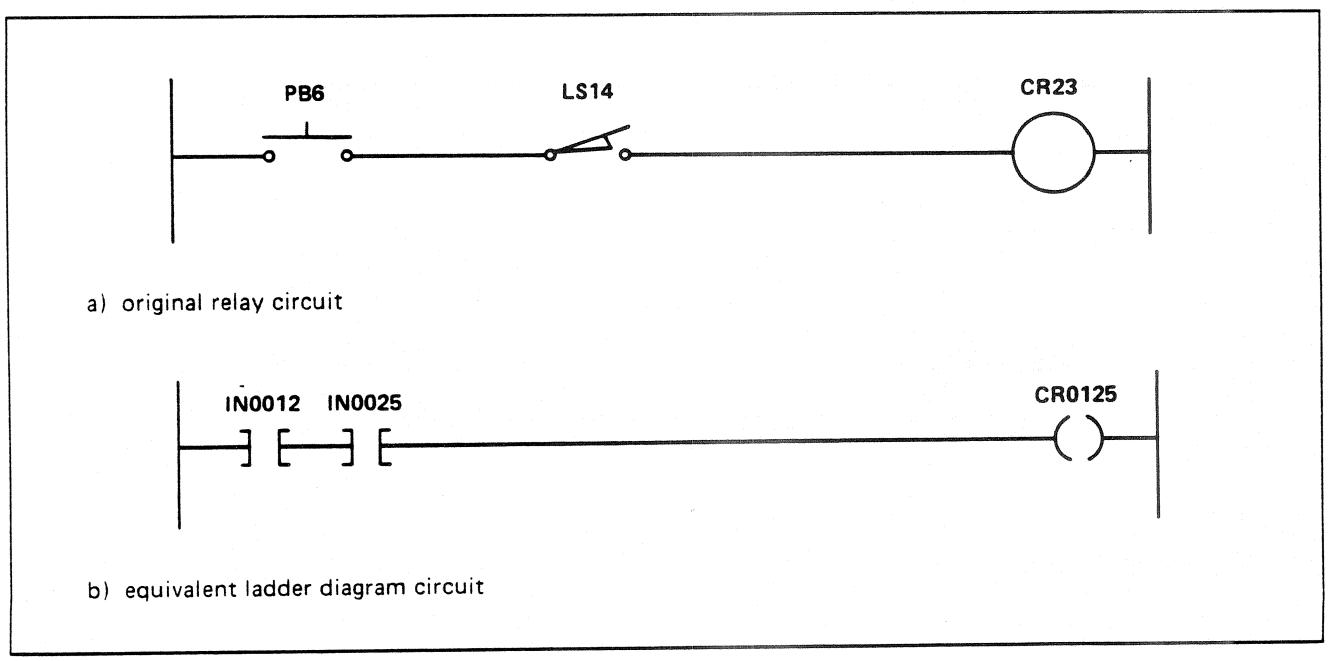


Figure 6-1. Input Devices Shown as Contacts

difference from conventional relay circuits. As shown, PB6 is replaced by IN0012 (input No. 12) and LS14 is replaced by IN0025 (input No. 25).

Each contact, coil or special function in the ladder diagram is also shown with a "label" (e.g., IN0025 or CR0125).

A label consists of a "type" designation (IN, CR, etc.) and a "reference number" (0001, 0025). (Labels are discussed in detail in Paragraphs 6-5 and 6-6.)

As shown in Figure 6-1, the IN part of the label identifies the type of element. In this example, the IN contacts are controlled by input circuits. As previously described, real-world input devices (e.g., pushbuttons and switches) are connected to converter circuits on input modules. The input circuits convert AC/DC voltage levels to the low DC levels required by the processor.

The reference numbers such as 0012 indicate exactly which input circuit address controls each of the input contacts.

In the ladder diagram, output devices (e.g., motor starters, solenoids, lights, etc.) are shown as coils. (See Figure 6-2.) Conventional control relay coils are also shown as coils. Therefore, there are really two different kinds of coils that can be shown in the ladder diagram. One kind of coil, called an output coil, controls internal contacts **and** is also associated with an output converter circuit that controls a real-world output device. The second kind of coil, called a logic coil, controls only internal contacts. (It does **not** directly control a real-world output device.) These logic coils are the equivalent of a conventional control relay.

Another difference between the ladder diagram and the relay ladder diagram is that many special functions (e.g., timers, counters, comparisons, addition, etc.) are shown as simple blocks. Each special function has at least 1 (and up to 6) contact circuits used to control its operation. For example, Figure 6-3 shows a timer. The timer accumulates when **both** the timing circuit **and** the enable circuit are conducting.

The special function names are contained within the special function block, as shown in the timer of Figure 6-3.

Special functions may respond to additional information supplied during the programming process. For example, Figure 6-3 shows a timer with a preset value of 180 seconds. This timer runs up to 180 seconds; it then energizes the coil. This operating data need not be a constant, but can be a variable contained in a register. It may be stored in some memory location by another function or by some real-world input. Thumbwheels, for example, could be used to manually set the preset value for a timer.

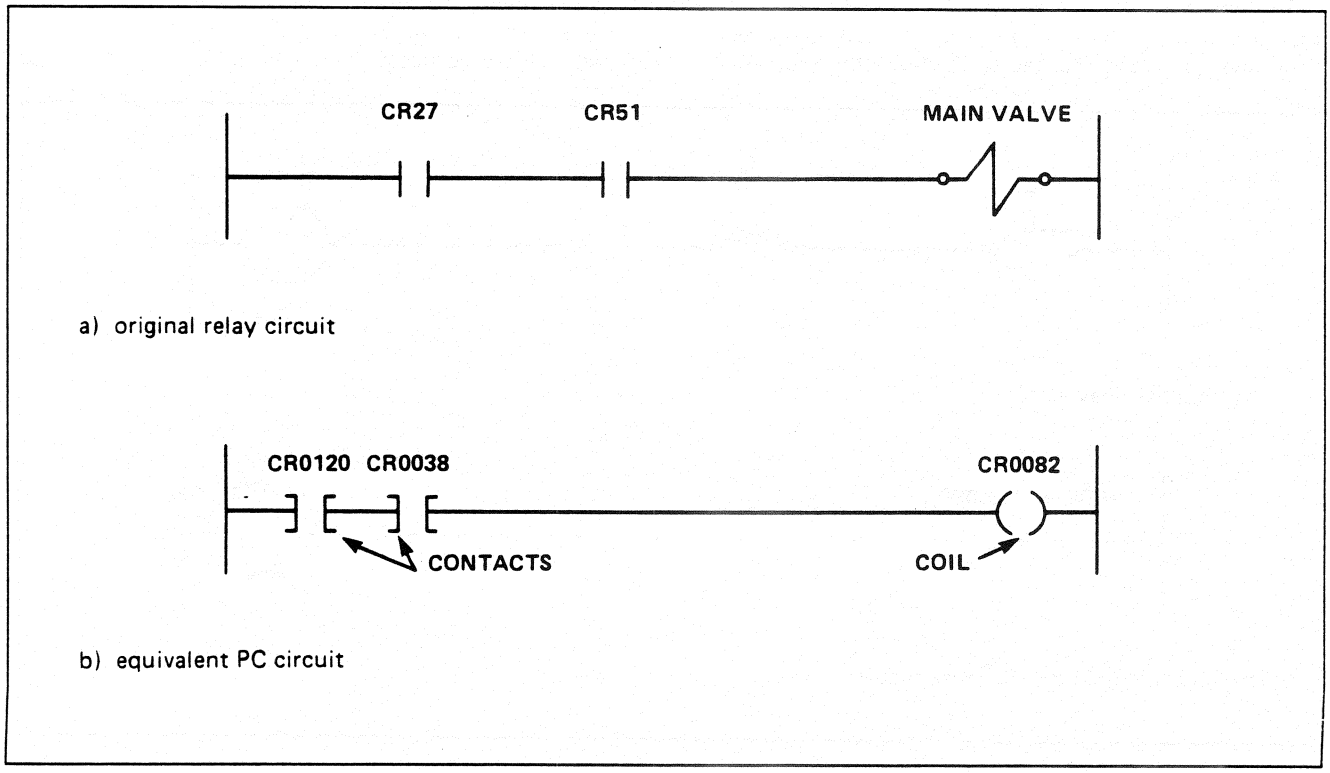


Figure 6-2. Outputs Shown as Coils

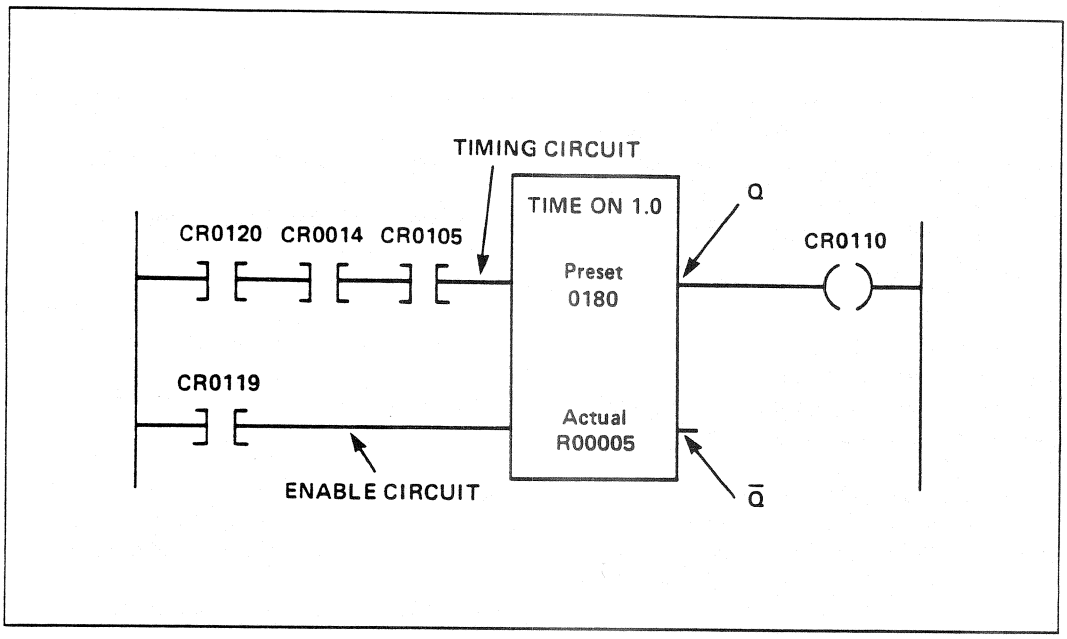


Figure 6-3. Special Function Block With Multiple Contact Circuits

The example also illustrates that special functions use numerical data. They accept the numerical data from registers, and, also, they supply similar numerical data to registers. Additionally, they function with discrete inputs and outputs (contact circuits and coils). When numerical data is required by a special function, the storage location of that data (input register, output register, or holding register) must be specified. For example, in Figure 6-3, the actual value of the timer is stored in holding register R00005.

Each of the register types contains 16 bits or individual storage locations where:

- Input registers contain the information from an Input Register Module. Each input register number or address corresponds to 16 bits of data available from the Input Register Module.
- Output registers store the value sent to an Output Register Module. Each output register number or address corresponds to 16 bits of data contained on the Output Register Module.
- Holding registers are used as internal storage locations assigned by the programmer. Each of the bits can be set and cleared from a separate coil and examined or modified by special functions.

A register may be associated with more than one special function. A given function may store numerical data in a register which is also used to supply numerical data to another special function. For example, consider an output register whose value may operate a 7-segment, 4-digit BCD display. This register can also provide numerical input data to another special function.

Registers, then, can perform a variety of functions, depending upon the application. They must be considered in the scheme used for designating the elements of the ladder diagram.

6-3. PROGRAMMING CONSIDERATIONS

The following considerations should be kept in mind when developing a ladder diagram:

- Processor scanning
- Conduction within ladder circuits

These are discussed separately in the following subparagraphs.

6-3-1. PROCESSOR SCANNING

To determine which outputs must be activated, the processor repeatedly scans the circuits programmed into it.

The processor scans the programmed circuits, starting with the first circuit programmed at the beginning of the ladder diagram. It does this to determine which, if any, circuits are conducting. The condition of each circuit (conducting or nonconducting) depends upon the states of any associated inputs and upon the states of the contacts that are controlled by other programmed coils. As the processor sequentially scans the programmed

circuits, the coil states are updated, one by one.

When a coil is updated during a scan, subsequent references to the updated coil's contacts reflect the updated status. In Figure 6-4, for example, coil CR0062 is controlled by the status established for IN0018 and CR0053 during the current scan, and by the status established for CR0101 during the **previous** scan (since CR0101 has not yet been scanned when CR0062 is being scanned).

As the circuits are scanned, the processor stores the newly determined coil states in a portion of memory. These memory states are used to update the on or off states of any associated real-world outputs.

To produce the desired system operation, circuits should be drawn and programmed in the order in which they should conduct. This order should be evaluated in light of the following discussion concerning conduction within ladder circuits.

6-3-2. CONDUCTION WITHIN LADDER CIRCUITS

When the processor is scanning the programmed circuits, a conduction path through contacts is considered to occur **only** from left to right. A conduction path for branches may also occur vertically (either up or down), as well as from left to right.

For example, consider the circuit shown in Figure 6-5. There are only three possible paths to energize coil G: A to B to C; A to D to E; and F to E. The path F to D to B does not energize coil G because it would require conduction from right to left through contact D.

6-4. CONSTRUCTION GUIDELINES

There are five rules which should be observed when drawing the circuits in the ladder diagram:

Rule 1: Contacts must be part of horizontal branches, **not** part of the vertical branches.

Rule 2: The conduction path flows from left to right; never from right to left.

Rule 3: Contacts connected to the right of a vertical branch should be connected to the topmost available junctions in the branch.

Rule 4: The contact area can normally accommodate a maximum of 10 series contacts in any given path with a maximum of 7 parallel rows.

Rule 5: Special function blocks (e.g., timers, counters, math functions, etc.) reduce the contact area normally available.

6-4-1. RULE 1

Always draw contacts as part of horizontal branches, not as vertical branches.

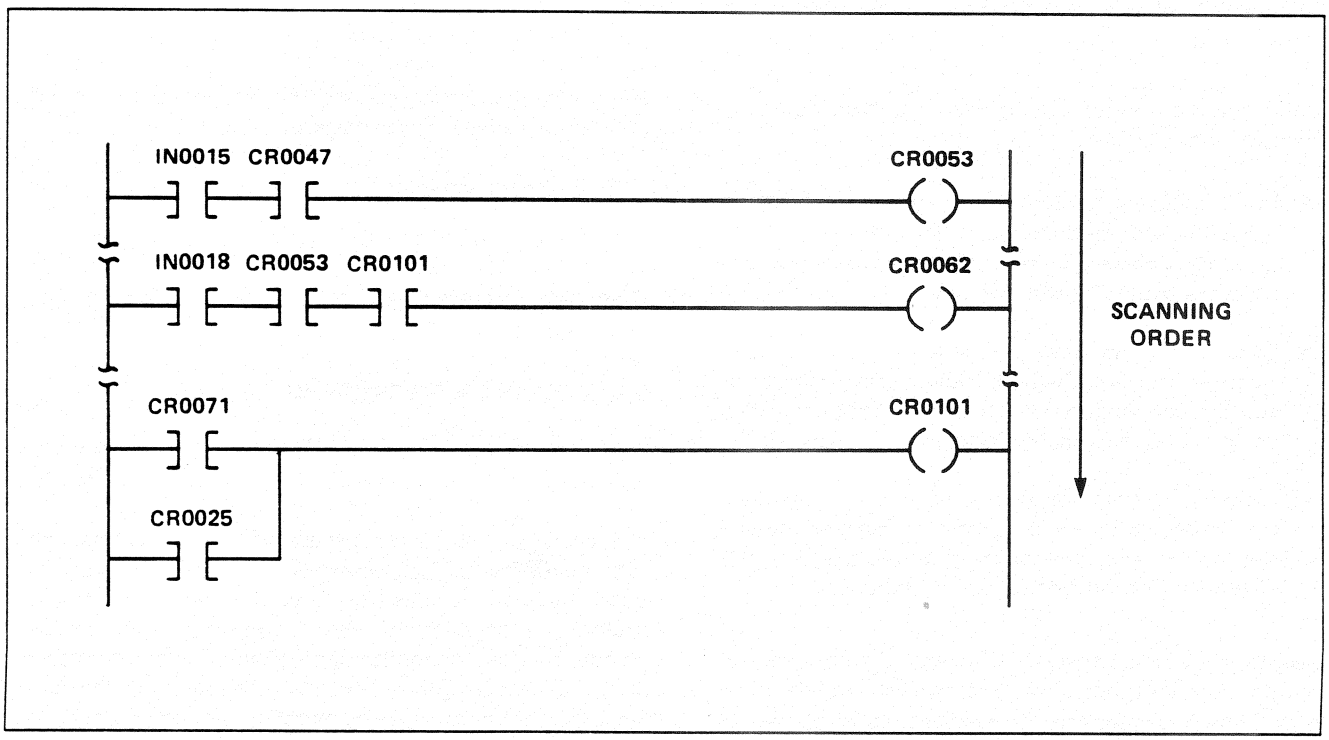


Figure 6-4. Processor Scanning

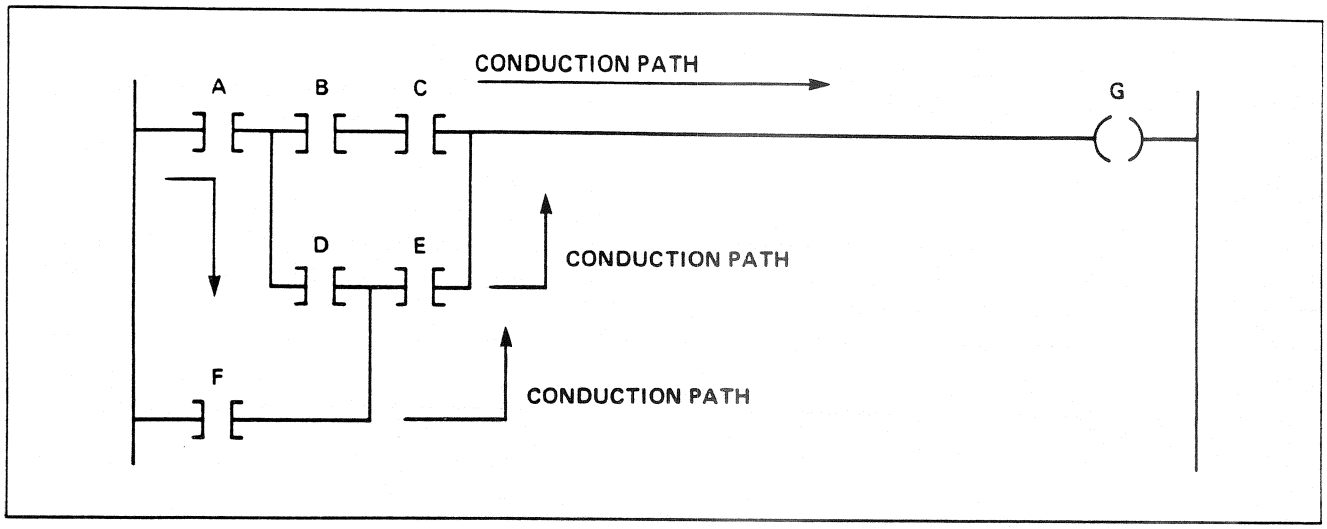


Figure 6-5. Circuit Conduction

To illustrate Rule 1, consider Figure 6-6. In the top circuit, contact C is drawn vertically, making it difficult to correctly identify the conduction path(s) associated with each contact.

In the bottom circuit, the conduction path associated with each contact is easier to identify. The direction of conduction through contact C is from left to right, as discussed previously. Note that in this circuit, path D to

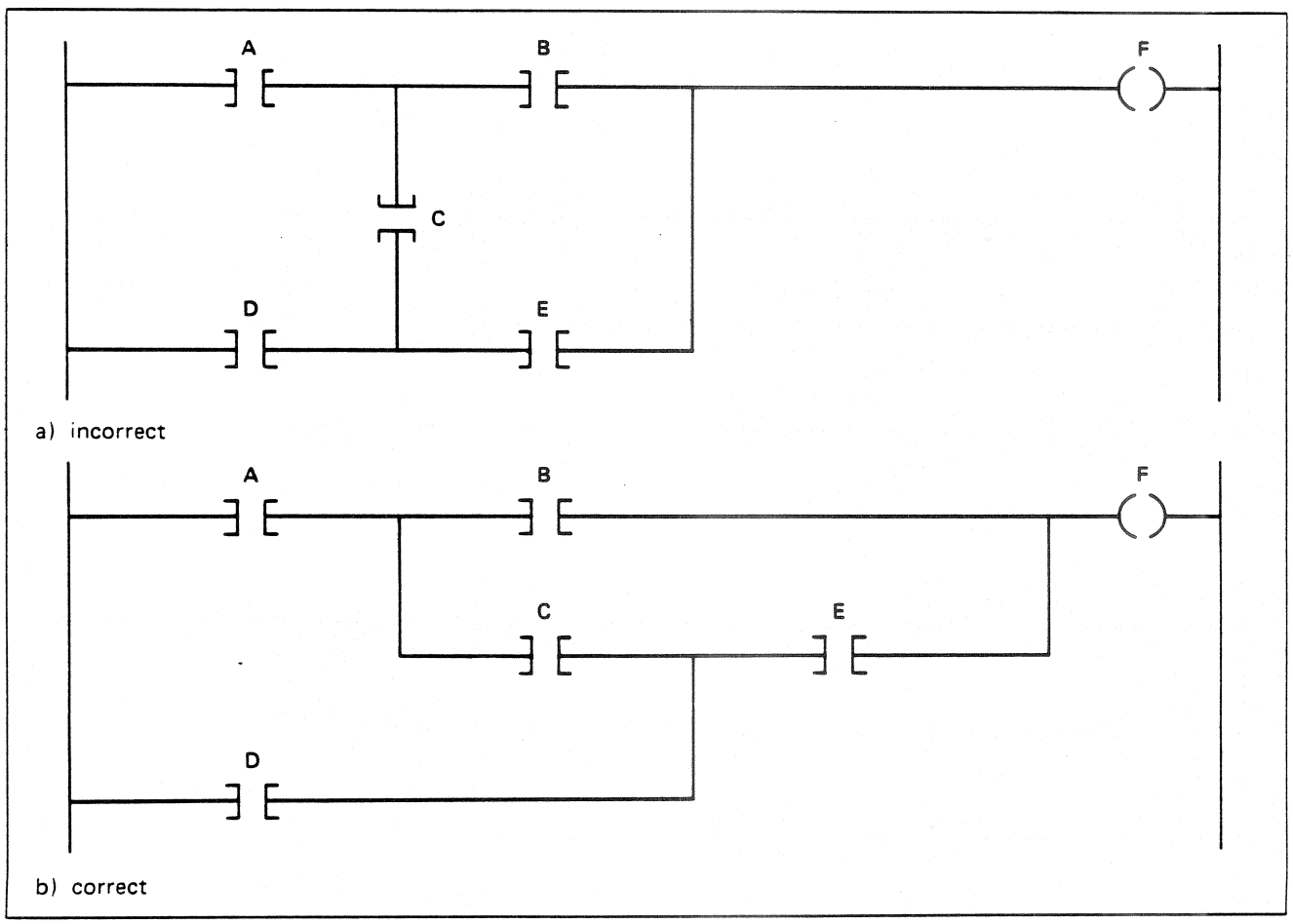


Figure 6-6. Rule 1: Contacts Must Occur in Horizontal Branches

C to B does not energize coil F. Figure 6-7 shows the same circuit with path D to C to B included to energize coil F

6-4-2. RULE 2

The conduction path flows from left to right, not from right to left. Circuits with improper conduction flow cannot be inserted in memory with the Loader.

To illustrate Rule 2, consider Figure 6-8. In the circuit at the top of the figure, the branch connection from the junction of contacts B and C to contacts H and I shows conduction from right to left as one possible path that may be followed.

In the circuit at the bottom of the Figure, the contact paths are easier to identify. It is also possible to insert this circuit into memory.

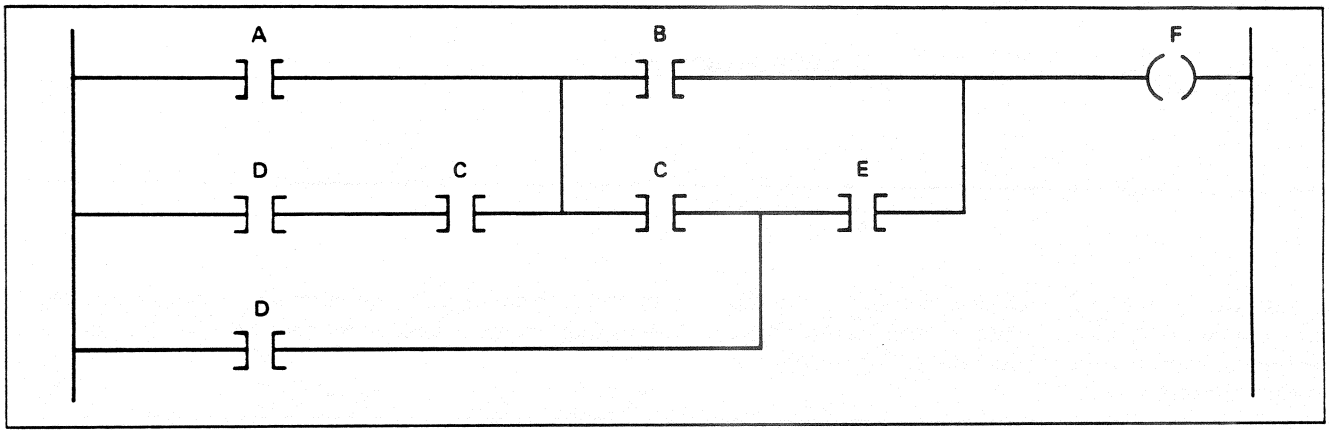


Figure 6-7. Addition to Two Contacts Adds Path D to C to B

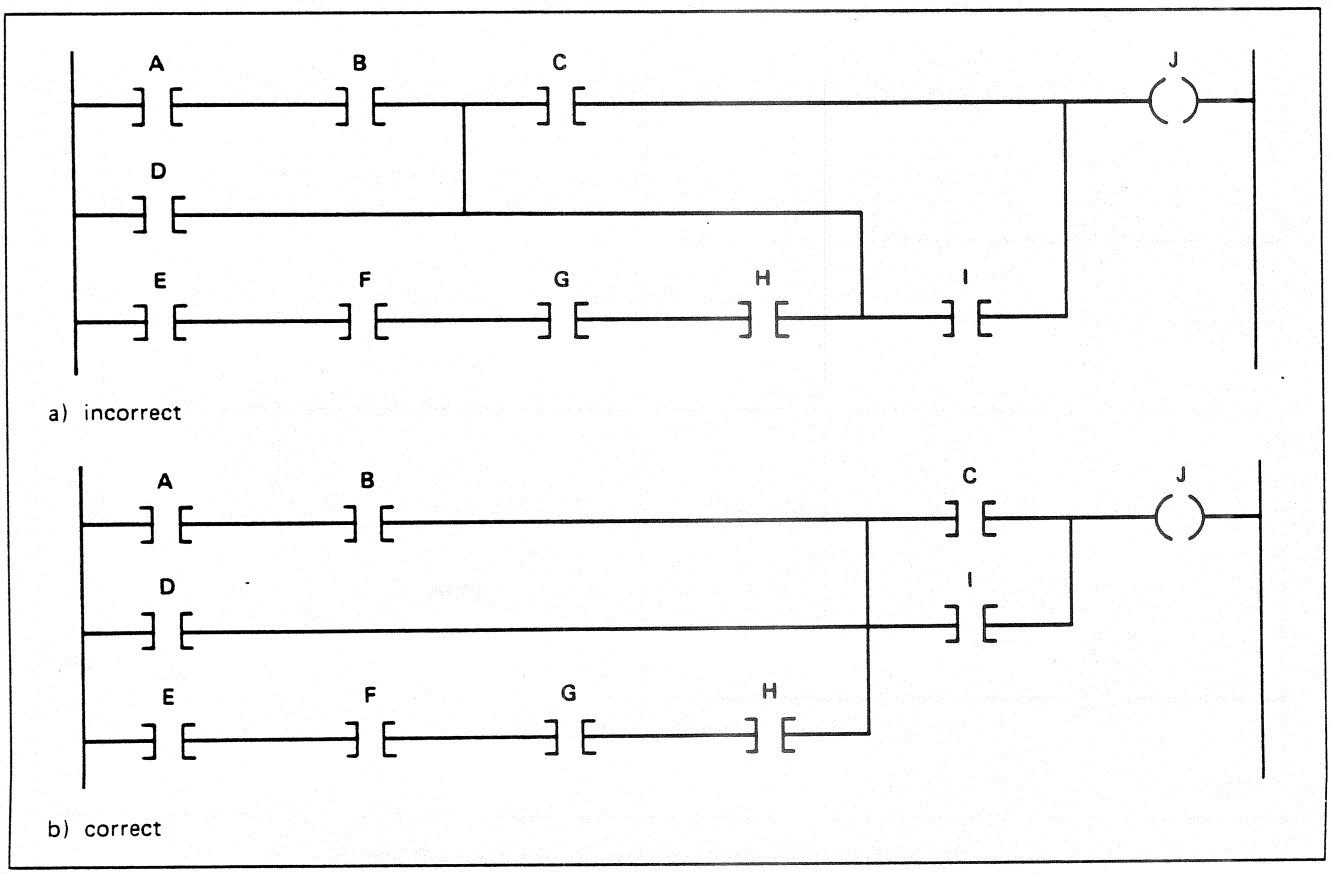


Figure 6-8. Rule 2: Branches Should Run Vertically

6-4-3. RULE 3

Contacts connected to the right of a vertical branch should be connected to the topmost available junctions in the branch.

To illustrate Rule 3, consider Figure 6-9. In the circuit at the top, contacts D and E are drawn at the bottom of the branch. The Advanced Program Loader automatically moves contacts to the topmost available junction.

In the bottom circuit, contacts D and E are drawn at the top of the branch, thus making it easier to identify the conduction paths associated with contacts B and C. Note that this rule also implies that a coil should appear in the topmost row of any circuit.

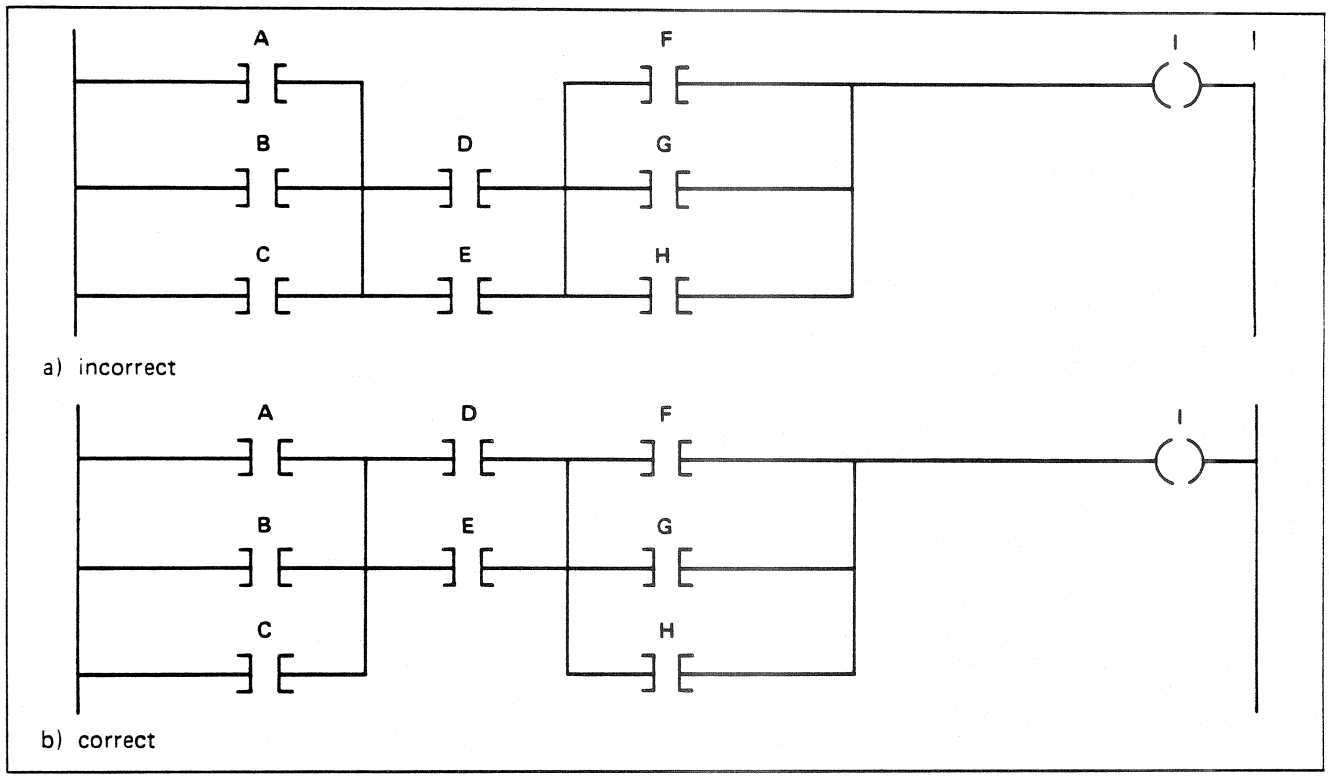


Figure 6-9. Rule 3: Contacts Should Be Connected to the Topmost Available Junctions

6-4-4. RULE 4

The contact area can normally accommodate a maximum of 10 series contacts in any given path with a maximum of 7 parallel rows.

Rule 4 describes the limits placed on the number of series contacts and parallel paths, or rows, of contacts. (See Figure 6-10.) A maximum of 10 series contacts can appear on each horizontal path. A maximum of 7 parallel rows of contacts can be programmed to a single coil. This is referred to as a 10x7 multiple contact area.

Figure 6-11 shows an example of an illegal series path containing 11 contacts. The Advanced Program Loader simply will not accept 11 series contacts or 8 parallel rows of contacts.

6-4-5. RULE 5

Special function blocks (e.g., timers, counters, math functions, etc.) reduce the contact area normally available.

Some special functions, because of the space they occupy on the display screen, reduce the space available

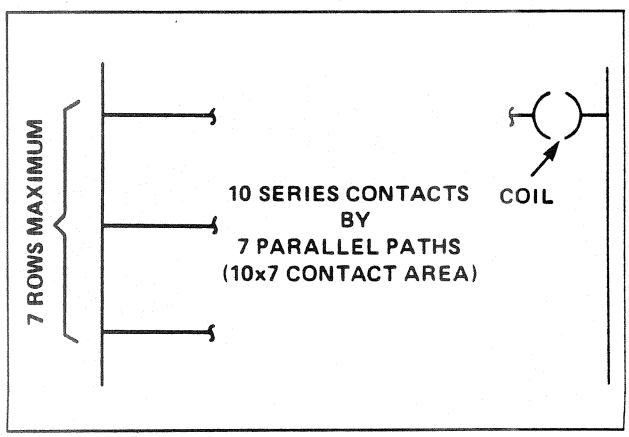


Figure 6-10. Rule 4: 10x7 Contact Area for Coils

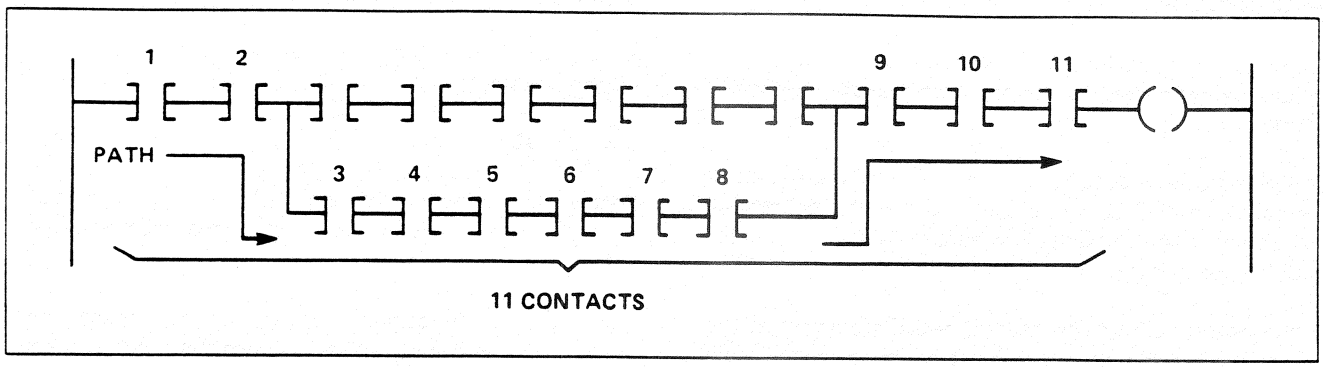


Figure 6-11. Illegal Path Example

in the normal 10x7 contact area. (See Figure 6-12.) Rule 5 states that special functions occupy a space equivalent to 2 contacts, horizontally. Also, the special function block requires the space of from 3 to 7 parallel paths, vertically.

Figure 6-12 represents the required space and indicates that there is an 8x7 contact area remaining. Another way of looking at this is to say that the special function occupies at least a "2x3 contact area."

The HPPC-1500/-1700 system allows special function blocks to be located in **any** position except the coil position at the extreme right. The overall remaining contact area—at the right, left, or either side—remains 8x7, assuming only one block is used. (See Figure 6-12.)

Keep in mind, however, that not all special function blocks require the same reserved space. Table 6-1 lists the reserved space requirements.

TABLE 6-1. SPECIAL FUNCTION BLOCK SPACE REQUIREMENTS

Function ①	Reserved Area ②
Timer	2x3
Counter	2x3
Math	2x4
Shift Register	2x3
Move	2x3
Drum Control	2x5
Comparisons	2x4 or 2x5
I/O	2x3

① Coils are not counted as part of the 10x7 or 8x7 contact area.

② These numbers represent the equivalent of a "contact area" that is used by the specific special function.

The purpose of Rules 1, 2 and 3 is to make programming easier. If violated, they will not inhibit the proper operation of the controller. Rules 4 and 5, however, **must be observed** for proper operation of the controller. Any lines that violate Rules 4 and 5 are illegal and cannot be entered. For example, Figure 6-11 illustrates a legal path, but because of the number of contacts, it would not be acceptable. In other words, any line that complies with Rules 4 and 5 is a valid line, even though the line may be drawn such that Rules 1, 2 and 3 are violated.

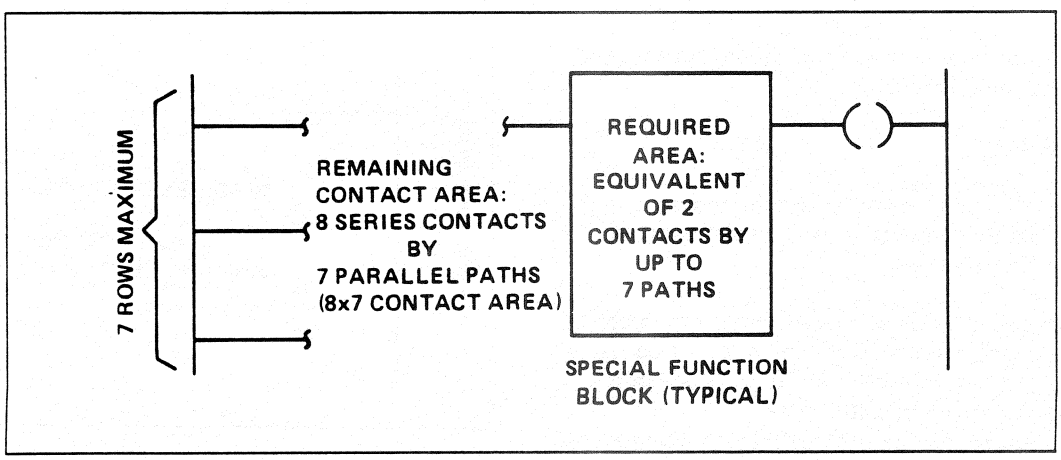


Figure 6-12. Rule 5: Special Function Blocks Reduce the Normal Contact Area

6-5. CONTACT LABELING SCHEME

A correctly documented ladder diagram has all contacts and coils labeled. That is, each is identified by type and given a unique reference number. (See Figure 6-13(a).)

There are 3 basic contact types available:

- IN contact, or input contact
- CR contact, or control relay contact
- BP contact, or bit pick contact

The IN contact examines a discrete input associated with a field device such as a limit switch or pushbutton. (See Figure 6-13(b).) The reference number specifies a particular I/O Rack, input module location in the Rack, and a particular input circuit. The input circuits are assigned reference numbers by the programmer in a range from 0001 thru 8144, depending on the configuration for the application.

The CR contact examines the status of a CR coil. (See Figure 6-13(b).) The reference number specifies the coil, output or logic, that controls the status of the CR circuit. For example, in Figure 6-4, contact CR0101 is controlled by coil CR0101. The reference numbers of the CR contacts fall within a range from 0001 thru 8144, or less—as assigned to the related coil by the system configuration.

The BP contacts (bit pick) examine a single bit in a specified register. (See Figure 6-13(c).) A register is an area of memory that stores data which is later made available to the processor. The BP contacts examine bit states in any of the following types of storage areas:

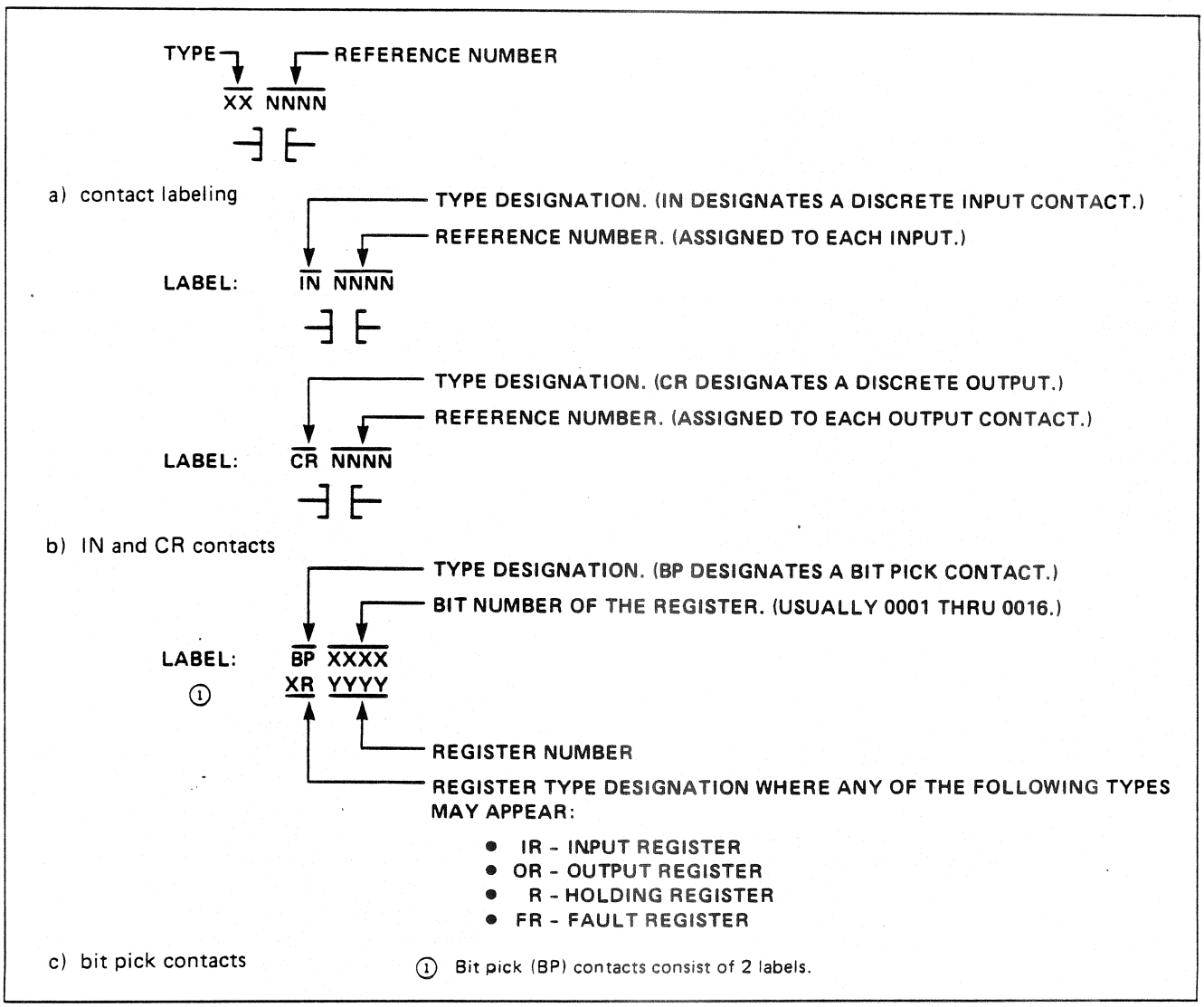


Figure 6-13. Labeling Scheme

- IR, or input register
- OR, or output register
- R, or holding register
- FR, or fault register

The IR, or input register, can accept data from a Register Input Module or an analog-to-digital input-type module. The data originates in the real world as 16-bit binary or BCD form.

The OR, or output register, outputs data from the ladder diagram to the Register Output Module or a digital-to-analog output-type module. The data is 16 bits in binary or BCD form.

The R, or holding register, is a storage area in memory that can be used by the programmer to temporarily or permanently store data for use by the processor.

The FR, or fault registers, contain system status information concerning the processor, software, or ladder program-dependent condition. There are 4 System Fault Registers and 6 IOP/SIM Fault Registers. Each contains 16 bits. (A listing is contained in Section 12.)

This information is summarized in Table 6-2.

TABLE 6-2. CONTACT LABEL SUMMARY DESCRIPTION

Contact Type	Description	Reference Number Range
IN0020	IN designates a discrete input circuit associated with real-world devices such as limit switches, pushbuttons or contacts from proximity switches. (This example examines the status of discrete input 20.)	0001 to 8144 corresponding to the I/O addresses of the actual input circuits, or as configured for the application.
CR0030	CR designates a discrete output circuit associated with real-world devices such as solenoids, motor starters or indicators or an internal logic coil. (This example examines the status of discrete output 30.)	0001 to 8144 corresponding to the actual addresses of the output circuits, or as configured for the application.
BP0002 R00010	BP designates a "bit pick," or the examination of a single bit in a holding register. (This example examines the status of bit 2 of holding register 10.)	BP. Generally, numbers here range from 0001 thru 0016 to correspond with the bits contained in a single reference number location. However, in some applications or functions it is desirable to cascade over into subsequent registers which may be acting as a group to store related data. In these cases the numbers may range from 0001 thru 9999. For example, BP0020/R00011 examines bit 4 (20 = 16 + 4) of holding register 12.
	R designates a holding register which is an internal storage area designated by the programmer. (This example examines bit 2 of holding register 10.)	R. Holding register should be assigned consecutive numbers starting with 0001, where each register contains 16 individual bits, or storage locations. If the programmer assigned data to R00125, 125 words of memory would be automatically reserved for holding registers, whether or not any data was contained in the first 124. Although never used, a maximum of 65,535 holding registers could be assigned.

(Cont'd.)

TABLE 6-2. CONTACT LABEL SUMMARY DESCRIPTION (Cont'd.)

Contact Type	Description	Reference Number Range
BP0005 IR0012	BP - See BP above	BP - See BP above
	IR designates an input register corresponding to an address of a real-world Register Input Module. (This example examines bit 5 of input register 12.)	IR. Register Input Modules are assigned numbers from 0001 to 0509 corresponding to their actual locations in the I/O subsystem, or as configured for the application.
BP0008 OR0009	BP - See BP above	BP - See BP above
	OR designates an output register corresponding to an address of a real-world Register Output Module. (This example examines bit 8 of output register 9.)	OR. The Register Output Modules are assigned consecutive numbers from 0001 to 0509, corresponding to their actual assignments in the I/O subsystem, or as configured for the application.
BP0002 FR0001	BP - See BP above	BP - See BP above
	FR designates a fault register. Certain bits in the fault registers are assigned to specific faults by the HPPC hardware. Other bits can be assigned faults by the programmer. Section 15 contains a complete listing of the fault registers, bit by bit.	10 fault registers are standard with the HPPC. Each contains 16 bits, although not all are currently used.

6-6. CONTACT SELECTION

Until now the description of the contacts has been centered on their labeling. For simplicity all of the examples have shown normally open (NO) contacts with a label. The process of entering each contact in a circuit actually involves selecting either a normally open (NO) or normally closed (NC) contact symbol, and only then specifying the label for each.

A summary description of the contacts is given in Table 6-3.

It is the programmer's responsibility to decide whether NO or NC contacts are to be used for each element and in each path of the ladder diagram. The type of input field device, the logic of the rung, and the anticipated output coil state all determine which contact symbol is to be used.

6-6-1. NORMALLY OPEN (NO) CONTACTS

Normally open IN contacts are open when the specified input circuit is OFF (activating voltage not applied). They are closed when the specified input circuit is ON (activating voltage applied).

Normally open CR contacts are open when the specified coil is de-energized. They are closed when the specified coil is energized.

Normally open BP contacts are open when the specified bit is logic 0. They are closed when the specified bit is a logic 1.

6-6.2. NORMALLY CLOSED (NC) CONTACTS

Normally closed IN contacts are closed when the specified field input circuit is OFF. They are open when the specified field circuit is ON.

Normally closed CR contacts are closed when the specified coil is de-energized. They are open when the specified coil is energized.

Normally closed BP contacts are closed when the specified bit is a logic 0. They are open when the specified bit is a logic 1.

6-7. COIL DESCRIPTION

Coils represent the output of a ladder diagram rung and

TABLE 6-3. CONTACT SUMMARY DESCRIPTION

Contact Type	-] [- Normally Open Contact	-]\[- Normally Closed Contact
IN	Open when the specified input circuit is OFF; closed when the specified input circuit is ON.	Closed when the specified input circuit is OFF; open when the specified input circuit is ON.
CR	Open when the specified coil is de-energized; closed when the specified coil is energized.	Closed when the specified coil is de-energized; open when the specified coil is energized.
BP	Open when the specified bit is a logic 0; closed when the specified bit is a logic 1.	Closed when the specified bit is a logic 0; open when the specified bit is a logic 1.
FR	Open when the specified fault is not active; closed when the specified fault occurs.	Closed when the specified fault is not active; open when the specified fault occurs.

are controlled by the logic of the rung. They always appear on the right most side of the display screen.

There are 6 types:

- CR coils, which are used to control real-world outputs or logic coils.
- BF (bit follow) coils, which are used to control a single bit either in a holding register or in an output register.
- BS/BC (bit set/bit clear) coils, which are used to control a single bit either in a holding register or in an output register.
- Latch coil, which is used to control either a real-world output or a logic coil.
- MCR (master control relay) coil, which allows the coils in a selected portion of the ladder program to de-energize.
- Skip coil, which allows the execution of selected portions of the ladder program to be skipped during the program scan.
- Jump coil, which allows looping back within the ladder diagram.

The CR, BF, BS/BC and Latch coils are described in Paragraphs 6-7-1 thru 6-7-4. The MCR and Skip coils are described in Section 7, and the Jump coil is described in Section 8.

The CR, BF, BS/BC and Latch coils noted above have the following common characteristics:

- Enable circuit
- Contact control
- Output **or** logic coils

Each of these characteristics is briefly discussed here.

All coils are energized or de-energized under the direction of an **enable circuit.** (See Figure 6-14.) This circuit provides a path of conduction. Each consists of up to 10 horizontal contacts located on up to 7 vertical paths. (This is the 10x7 contact area, explained in Paragraph 6-4.)

In addition to the contacts, the enable circuit may also include special function blocks which also energize or de-energize coils. More than one special function may be located on an enable circuit. (Sections 7 and 8 list each of these programmable functions and describe their uses.)

Coils may exercise **contact control.** All CR contacts in the ladder diagram with the **same** reference number as the coil are controlled by that coil. For example, the state of control relay contact CR0500 is controlled by the state of coil CR0500. (See Figure 6-15.)

Thus, when the coil is:

- **Energized,** normally open contacts with the same reference number are closed, or conducting. Normally closed contacts are open.
- **De-energized,** normally open contacts with the same reference number are open, or not conducting. Normally closed contacts are closed.

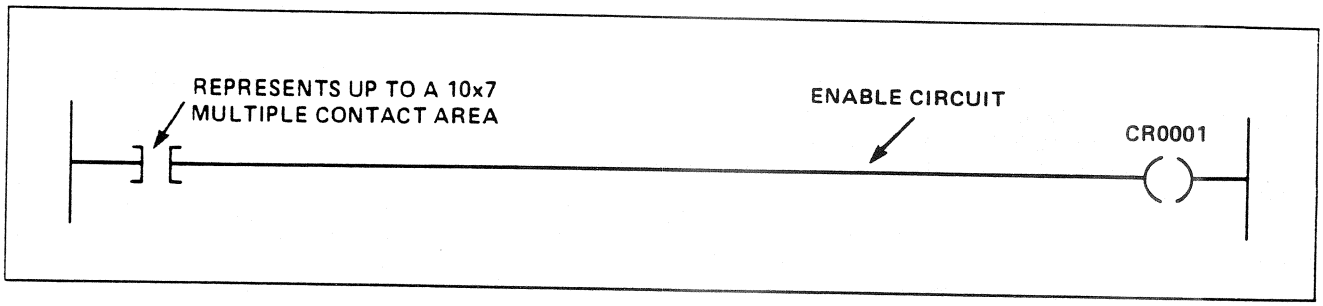


Figure 6-14. Enable Circuit

All coils are either **output or logic coils.** Specifically:

- Output coils control real-world output circuits linked to field devices. (These may be either discrete or register outputs.)
- Logic coils may at times be used to store data within the processor so that it is available for use in the ladder diagram. (They **do not** correspond to any real-world circuits.) However, the Bit Pick coil, explained in later Paragraphs, function may be easier to use.

Any coil, whether a logic or output type, must be configured for the application even through the logic coils are not assigned to a SIM.

6-7-1. CR COILS

The CR coil is used by the programmer to energize and de-energize either output or logic type coils. Output coils and logic coils may be assigned reference numbers from 0001 thru 8144, depending on the configuration of the application.

An example of a CR coil is shown in Figure 6-16. Study the Figure and note the coil's on/off state follows the conduction path, as shown in the timing chart.

6-7-1-1. SPECIFICATIONS

The CR coil is always located on the right-hand side of the display. With respect to labeling, the CR defines the type. Programmers assign the reference number in a range from 0001 thru 8144 for each unique coil, as defined by the configuration for the application.

See Paragraph 6-8 for other programming considerations.

6-7-1-2. APPLICATIONS

Three basic applications of the CR coil are discussed here. These are:

- Start/stop circuit
- Dummy coil
- Oscillator circuit

Each is described separately here.

A **start/stop circuit** application is shown in Figure 6-17. It can be assumed that its function is to allow operator-controlled, START/STOP pushbuttons to input signals into the HPPC-1500/-1700 in order to turn on/off a conveyor motor.

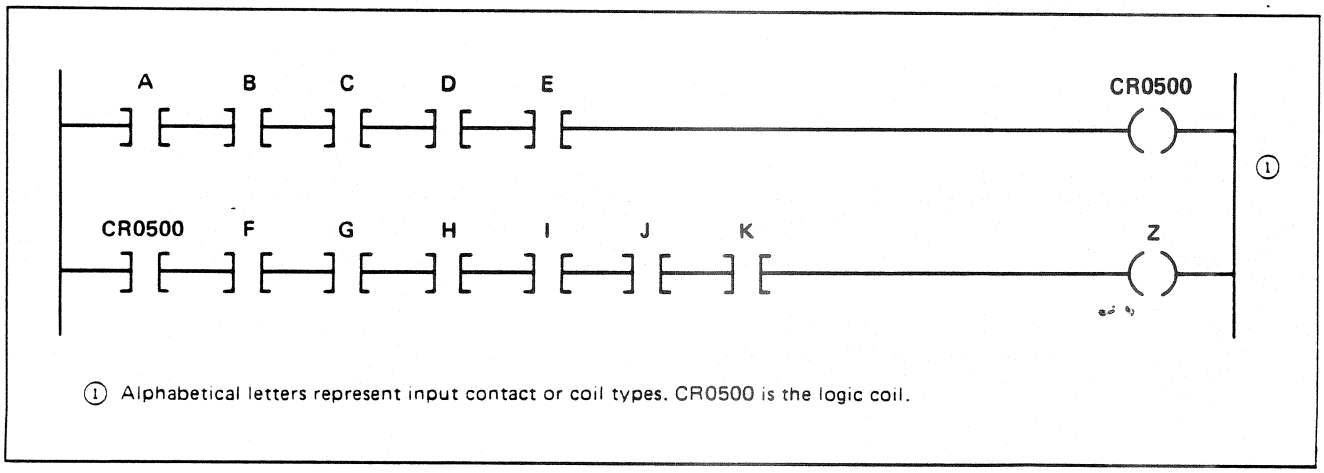


Figure 6-15. Contact Controlling a Coil

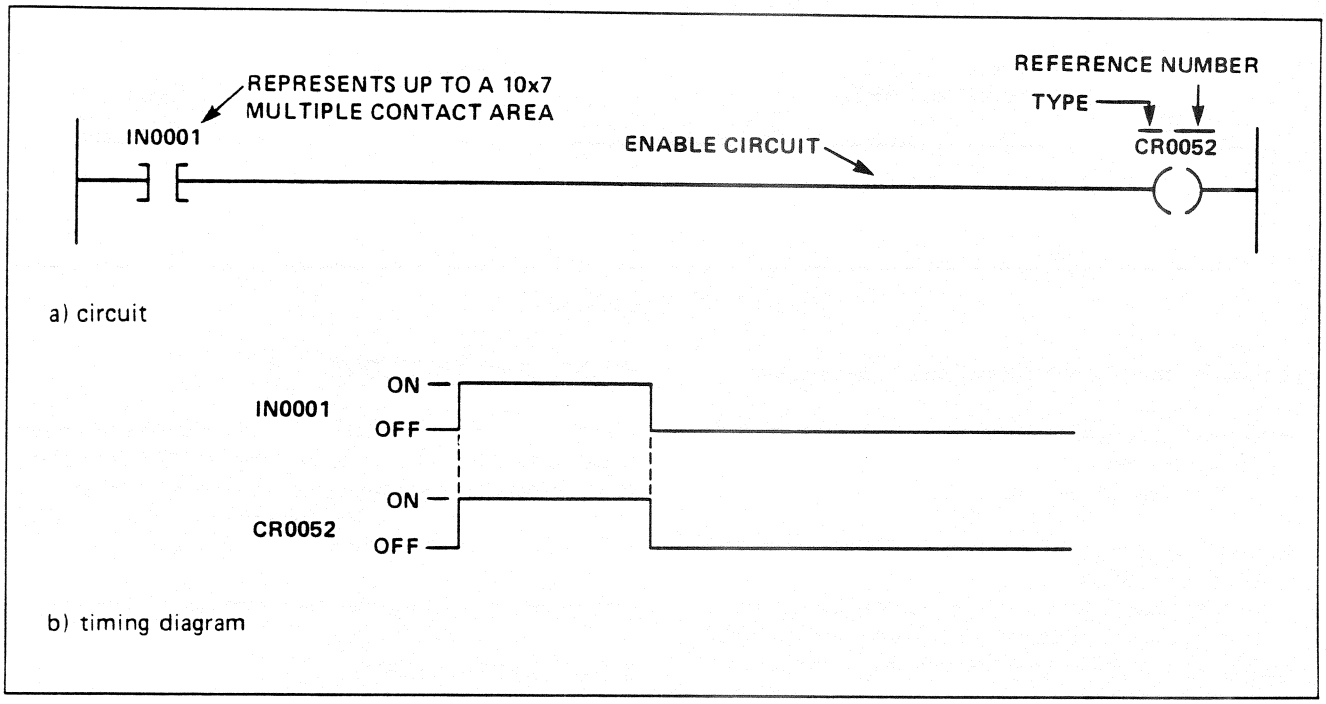


Figure 6-16. CR Coil

Here the discrete inputs START and STOP are assigned the reference numbers IN0001 and IN0002, respectively. The discrete output to the motor is assigned CR0001. Pressing START energizes CR0001 through normally closed contact IN0002.

But energizing the coil also causes normally open contact CR0001 on a parallel path to close. The result is to "hold in" output CR0001. Thus the normally open CR contact is a "seal-in" or "holding" circuit.

Only when STOP is pressed will the circuit open and de-energize output coil CR0001. This action causes CR contact 0001 to open, breaking the seal-in circuit.

A **dummy coil** application is shown in Figure 6-18. Some programming techniques require that specific contacts remain in a constant, predictable state for use in specific applications. To achieve this, the dummy coil's reference number is duplicated in 2 contacts programmed in series with the coil. (See the Figure. The dummy coil does not control any output.)

As shown in the Figure, coil CR0503 is always de-energized. The way that the dummy rung is constructed allows CR contact 0503 to be always open when examined by a NO contact. It is always closed when examined by a NC contact. In this way the constant state of the CR contact is assured.

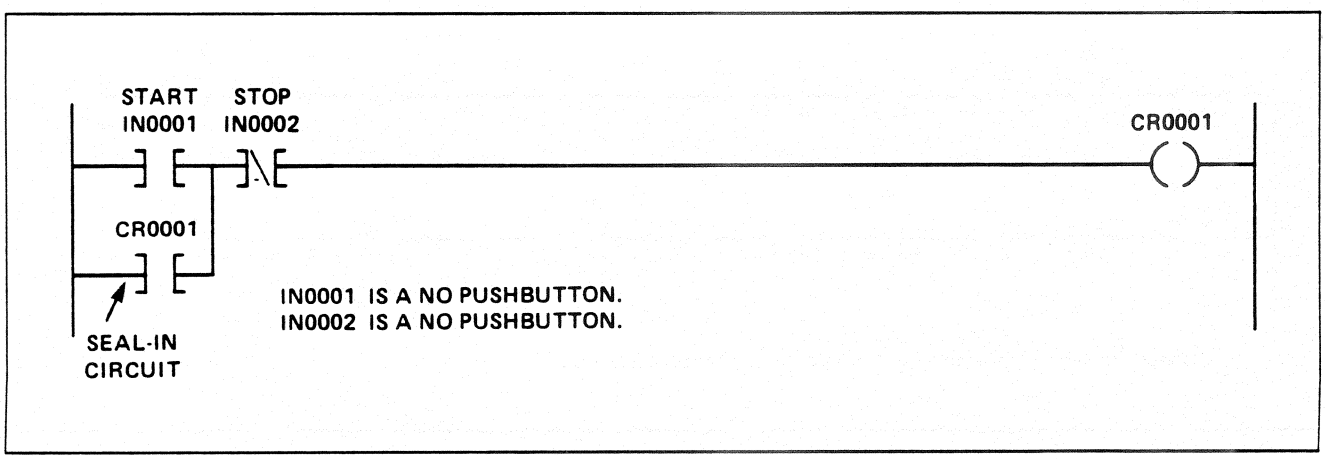


Figure 6-17. Start/Stop Circuit

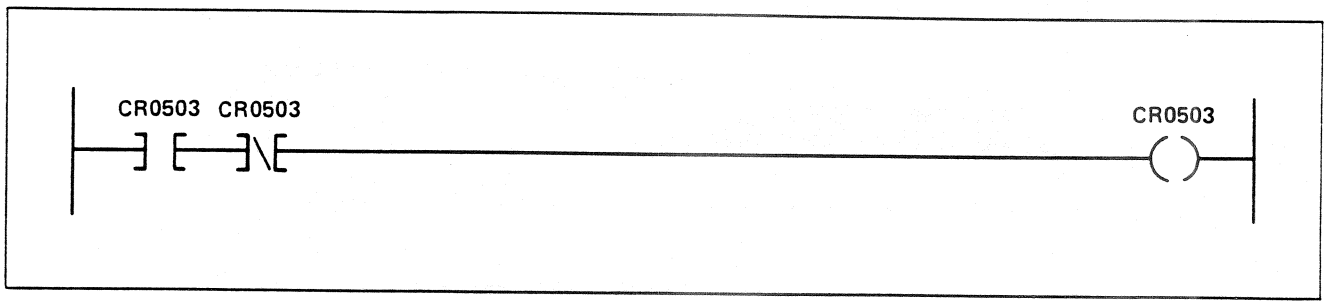


Figure 6-18. Dummy Circuit

An **oscillator circuit** is shown in Figure 6-19. There are times when it is valuable to have a CR contact turning on or off in sync with every other program scan. The contact can then be examined for an oscillating on/off state.

To achieve this, construct a circuit with an NC CR contact and a coil. Assign identical reference numbers to a contact and coil. When the processor begins scanning, CR contact 0502 is examined as conducting. Coil CR0502 is energized. On the next scan, the NC contact will not be conducting because the coil is energized. This logic de-energizes the coil. In this way the coil is energized every other scan, and the CR contact can be examined to clock the oscillation.

6-7-2. BIT FOLLOW COIL

The Bit Follow coil is used to control a single bit in an output register or in a holding register according to the enable circuit. Here "control" means that the bit is energized, or set, to a logic 1. Alternately, it means the bit is de-energized, or cleared, to a logic 0.

When the Bit Follow coil controls a bit in an output register, it is referred to as a BF/OR coil. (In this case real-world output circuit on a Register Output Module is being controlled.) When it controls a bit in a holding register, it is referred to as a BF/R coil. (In this case a storage bit in memory is being controlled.)

An example of the use of a Bit Follow coil is shown in Figure 6-20. Note that the coil—or the bit state—simply follows the conduction path of the enable circuit. When the enable circuit is conducting, the coil is enabled. The bit is set to logic 1. When the circuit is not conducting, the bit is cleared to logic 0.

6-7-2-1. SPECIFICATIONS

The BF coil is located on the right-hand corner of the display screen. The following programming specifications apply to this coil. (See Figure 6-21.)

Type. BF defines the Bit Follow coil.

Reference Number. The reference number defines exactly which bit is being controlled. The range used is normally 0001 thru 0016.

The HPPC-1500/-1700 processor also allows numbers from 0017 thru 9999 to be used. In this case the bit being controlled would be located in a register higher in number than the one designated. For example BF0026 is actually bit 10 of the next higher register (26 - 16 = 10). However, this feature is somewhat cumbersome, and is used only in special applications.

Register Type. The Bit Follow coil may be used with 2 types of registers: output register (OR) or holding register (R).

Register Reference Number. If a BF/OR coil is used, the range of reference numbers for the output register is from 0001 thru 0509, depending on the configuration for the application. These correspond directly to reference numbers assigned to Register Output Numbers. (These reference numbers represent 16 available bits.)

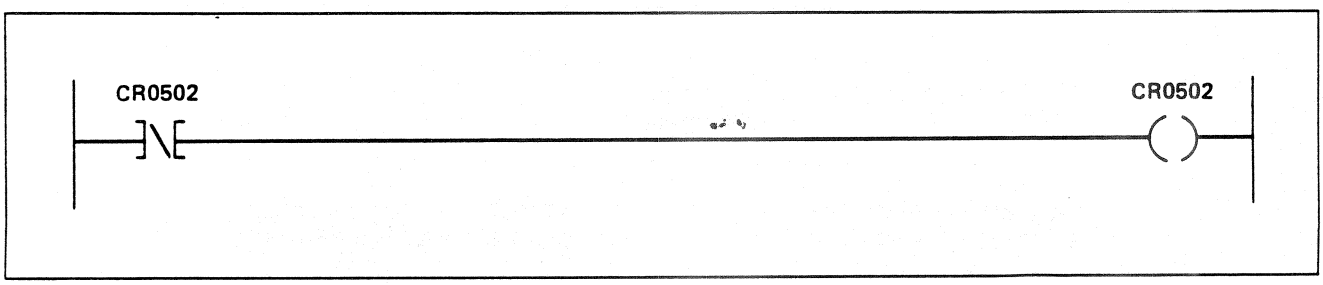


Figure 6-19. Oscillator Circuit

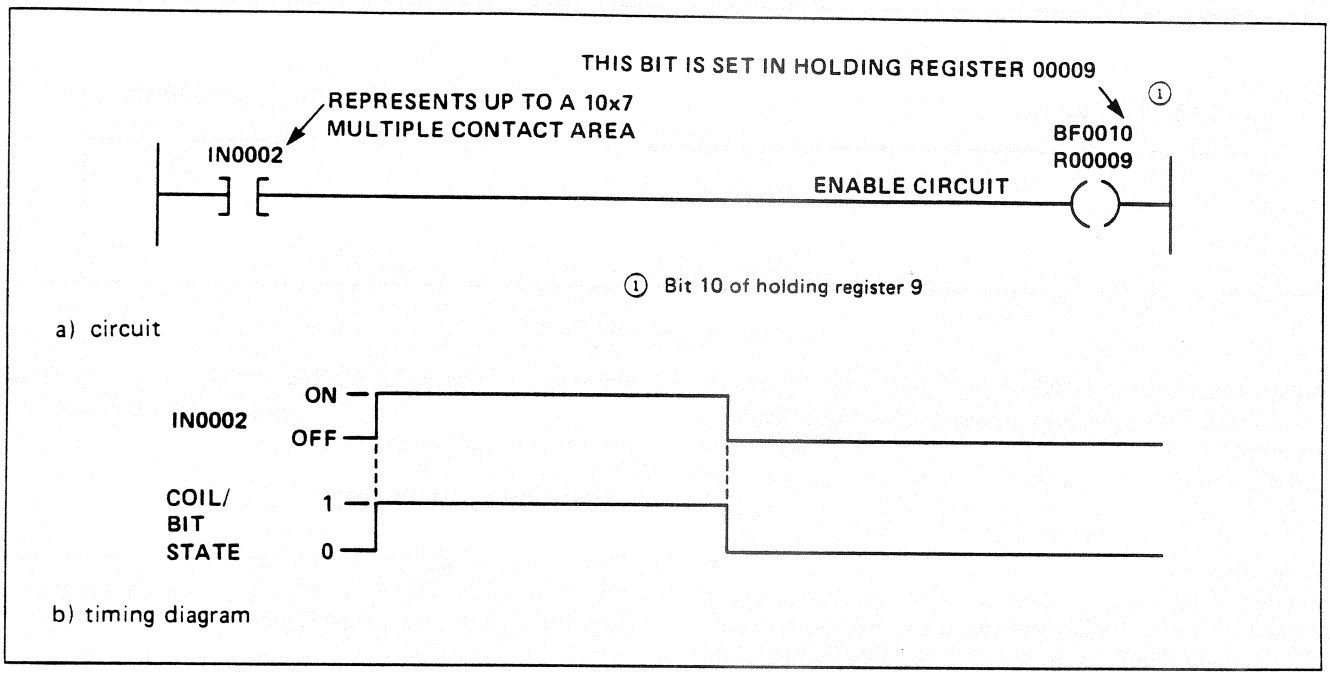


Figure 6-20. Bit Follow Coil

If a BF/R coil is used, the range of reference numbers is 00001 thru 65535, although it is doubtful anywhere near this number would actually be used. These reference numbers correspond directly to holding registers assigned by the programmer for storage.

Note

Assign reference numbers only to holding registers actually needed. Begin at 00001 and progress consecutively to use memory most efficiently. The processor automatically reserves registers (words) in memory from 00001 to the highest holding register used (HHRU), whether or not all are used. Memory could be inefficiently used if random numbers are chosen, although a block of approximately 10 registers could be reserved for subsequent program modification.

See Paragraph 6-8, Coil Considerations, for other programming considerations.

6-7-2-2. APPLICATIONS

The BF/R coil provides internal logic coils which may be used for program status indication, control of other enabling circuits or coils, or signals to be used on data highway messages.

6-7-3. BIT SET/BIT CLEAR

The Bit Set and Bit Clear coils, together, act as a pair to control the same bit in an output register or holding register according to the individual enable circuits. The Bit Set coil places the bit in a logic 1 state. The Bit Clear coil places the bit in a logic 0 state.

When the Bit Set or Bit Clear coils control a bit in an output register, they are referred to as a BS/OR and BC/OR coils, respectively. (In this case a real-world

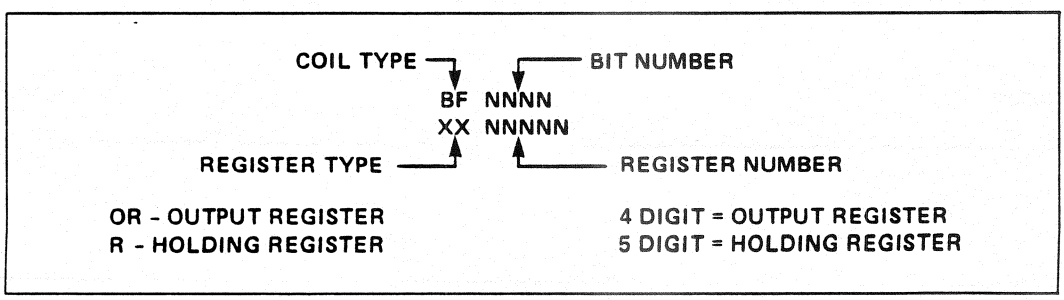


Figure 6-21. BF Label Description

output circuit on a Register Output Module is being controlled.)

When these coils control a bit in a holding register, they are referred to as BS/R and BC/R coils, respectively. (In this case a storage bit in memory is being controlled.)

An example of these coils is shown in Figure 6-22. Both coils are given the **same** reference number in the **same** holding register. During any given ladder diagram scan, the last coil enabled controls the bit. In a case where both enable circuits are conducting at one time—as shown in Figure 6-22— the bit would be cleared to logic 0 since the Bit Clear coil is the most recent one enabled. An exception to this statement is during the period of the ladder diagram scan between the Bit Set coil and the Bit Clear coil.

6-7-3-1. SPECIFICATIONS

The Bit Set and Bit Clear coils are located on the right-hand portion of the display screen. The following programming specifications apply to these coils. (See Figure 6-23.)

Type. BS defines the Bit Set coil. BC defines the Bit Clear coil.

Reference Number. The reference number defines exactly which bit is being controlled. The range used is normally 0001 thru 0016.

The HPPC-1500/-1700 processor allows numbers from 0017 thru 9999 to be used in case the bit being controlled would be located in a register higher in number than the one designated. For example BS0026 is actually bit 10 of the next higher register (26 - 16 = 10). However, this feature is somewhat cumbersome, and is used only in special applications.

Register Type. The Bit Set and Bit Clear coils may be used with 2 types of registers: output registers (OR) and holding registers (R).

Register Reference Number. If a BS/OR or BC/OR coil is used, reference numbers for the output registers range from 0001 thru 0509, depending on the configuration. These correspond directly to reference numbers assigned to Register Output Modules. (**Each** of these reference

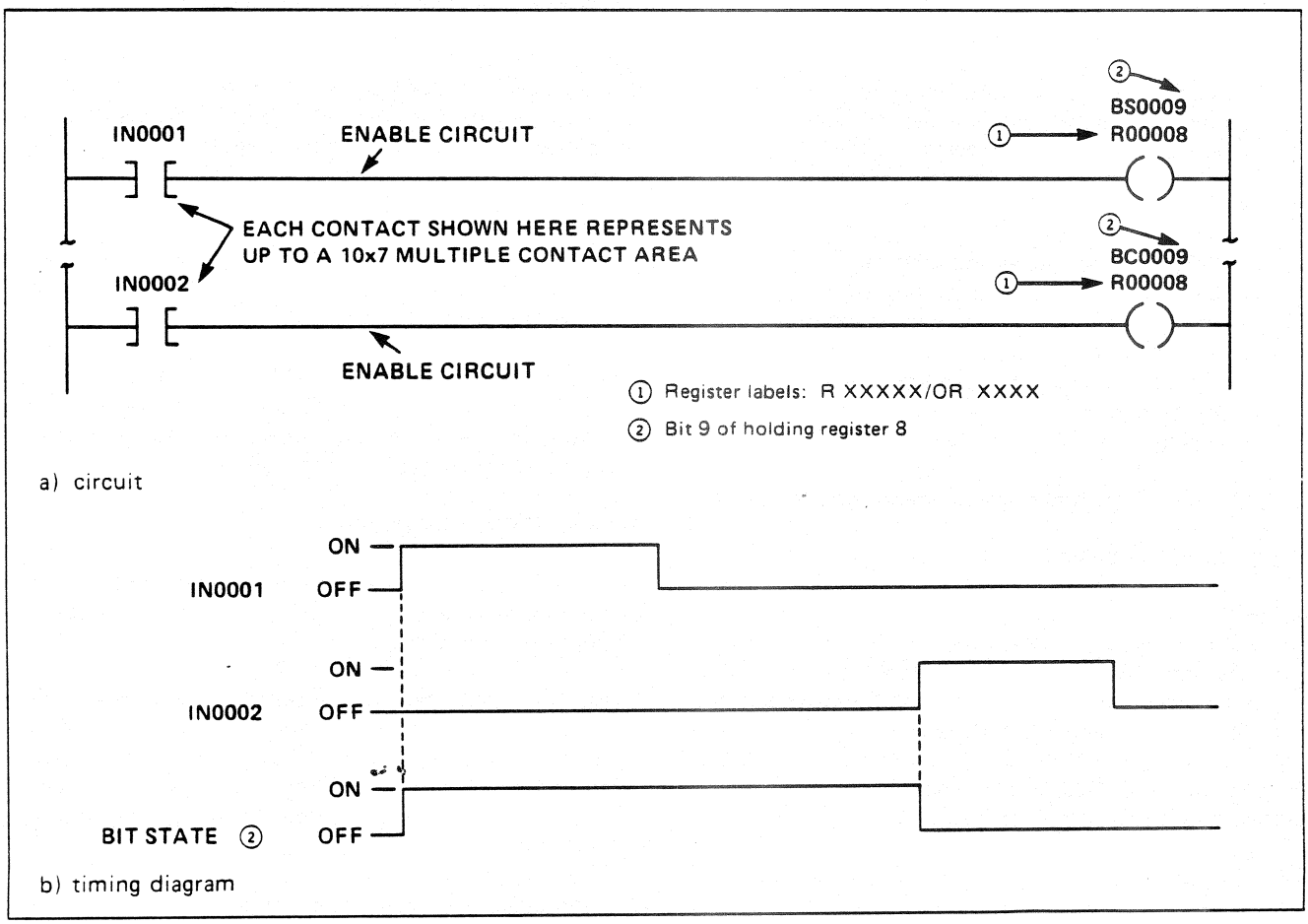


Figure 6-22. Bit Set/Bit Clear Coil

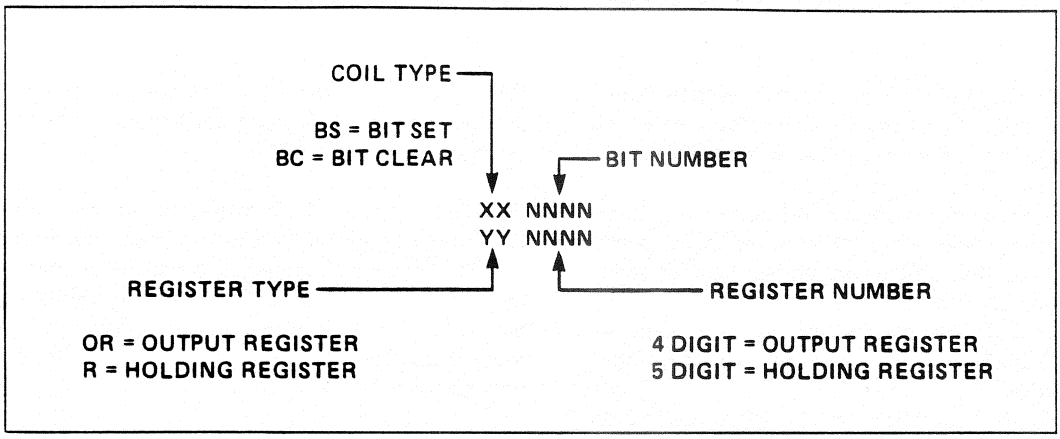


Figure 6-23. BS/BC Label Description

numbers represents 16 available bits.)

If a BS/R or BC/R coil is used, the range of reference numbers is 00001 thru 65535, although it is doubtful anywhere near this many would actually be used. These reference numbers correspond directly to holding registers assigned by the programmer for storage.

Note

Assign reference numbers only to holding registers actually needed. Begin at 00001 and progress consecutively to use memory most efficiently. The processor automatically reserves registers (words) in memory from 00001 to the highest holding register used (HHRU), whether or not all are used. Memory could be inefficiently used if random numbers are chosen, although a block of approximately 10 registers could be reserved for subsequent programming modifications.

See Paragraph 6-8 for other programming considerations.

6-7-3-2. APPLICATION

The advantage the BS and BC coils have over the Bit Follow coil is that at times they make programming easier. As indicated in Figure 6-20, the Bit Follow coil permits a single enable circuit to control only one state of the bit. The BS and BC coils **each** require an enable circuit to fully control the bit. The advantages are that the bit's state is fixed until a specific enable circuit changes it, and each enable circuit will be less complex.

6-7-4. LATCH

The Latch coil functions much like a CR coil, but it has the additional advantage of being able to resume its last state after AC power to the processor is removed and restored, assuming conditions on the enable circuit do not change. (See Figure 6-24.)

(The CR coil, when used to control real-world outputs, is de-energized during a power loss. The effect is to allow the processor to resume control in an orderly fashion **before** any field devices resume operation.)

All statements concerning the Latch coil here assume that a battery backup is supporting the ladder program. If the battery backup fails, the states of the Latch coils will be lost along with the ladder diagram program.

All coils associated with the Latch coil are either:

- Output coils, which control output circuits linked to field devices. (These may be either discrete or register inputs.)
- Logic coils, which may at times be used to store data within the processor so that it is available for use in the ladder diagram. (They **do not** correspond to any real-world circuits.)

6-7-4-1. SPECIFICATIONS

The Latch coil must be located in the right-hand column of the display screen. The following programming specifications apply to this coil.

Type. There is no type in the same sense as there is for the coils discussed thus far. A normal CR coil is first programmed, as described in Paragraph 6-7-1. Then, when the Latch function is selected on the Advanced Program Panel, the letter L appears within the output coil symbol.

Reference Number. Only the reference number of the CR coil need be entered. This may range from 0001 thru 8144, depending on the application's configuration.

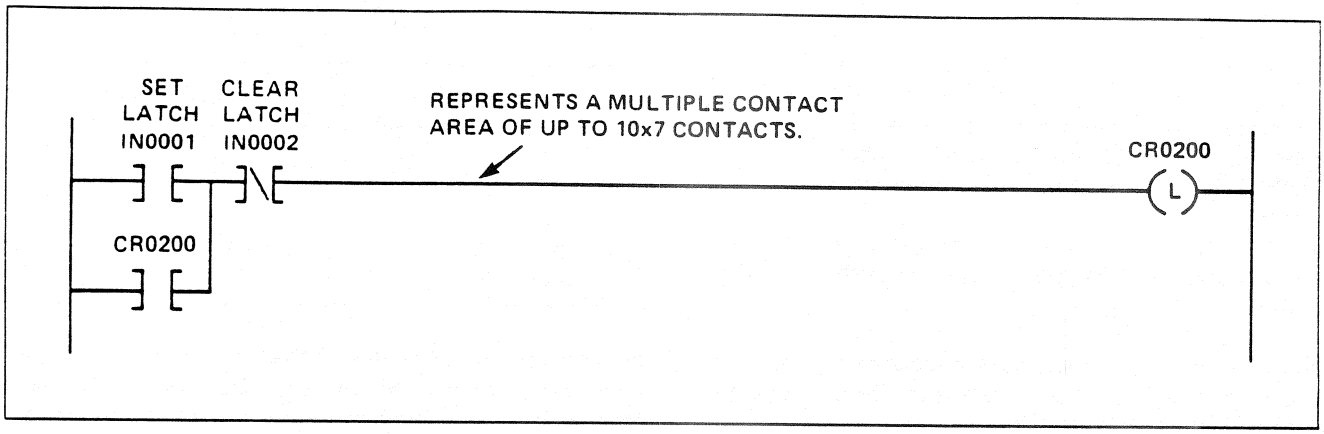


Figure 6-24. Latch Circuit (Typical)

WARNING

USE OF THE LATCH COIL CAN RESULT IN UNEXPECTED MACHINE MOVEMENTS OR PROCESS CONTROL OPERATIONS WHEN AC POWER IS FIRST APPLIED OR THE KEYSWITCH POSITION IS CHANGED. IT IS THE PROGRAMMER'S RESPONSIBILITY TO ASSURE HAZARDOUS CONDITIONS ARE NOT CREATED AT THESE TIMES. FAILURE TO COMPLY CAN RESULT IN SERIOUS OR FATAL INJURY AND/OR MACHINE DAMAGE.

6-7-4-2. APPLICATION

The Latch coil can be used as the equivalent of a retentive relay. (See Figure 6-25.) For example, assume motor starter MS27 is operating normally, but AC line power is temporarily lost. When the power is restored, Latch coil CR0001, which retained its energized state, automatically allows the controlled motor to restart. There is no need to manually depress the START pushbutton.

6-8. COIL CONSIDERATIONS

The considerations listed here apply to all coil types. As a programmer chooses among the available coil types to write the ladder diagram, he must keep in mind 3 general areas of consideration. These are:

- Retentive/nonretentive characteristics (Par. 6-8-1)
- SIM shutdown characteristics (Par. 6-8-2)
- Multiple-coil capabilities (Par. 6-8-3)

6-8-1. RETENTIVE/NONRETENTIVE CHARACTERISTICS

Being **retentive** or **nonretentive** is a characteristic common to all coils. For simplicity of explanation, no mention has been made of this until now, but each coil discussed must now be considered with respect to the characteristic.

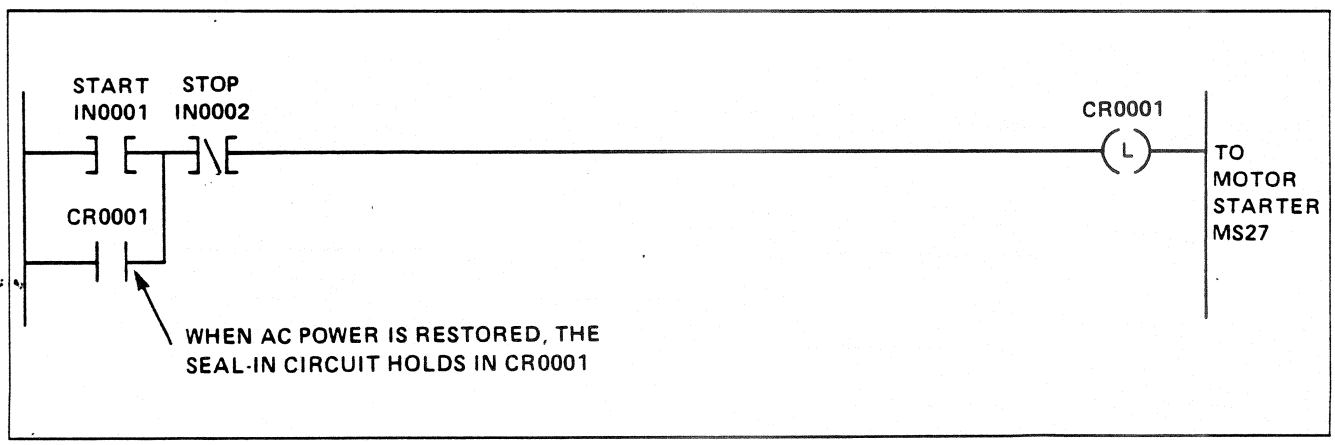


Figure 6-25. Latch Application

A retentive coil is able to retain its last state even during a loss of AC line power or a keyswitch transition.

A nonretentive coil does not retain its last state during a power loss, or when the keyswitch is changed to the STOP:PROGRAM position. A nonretentive output coil de-energizes, and a nonretentive logic coil is cleared to 0. Even if a seal-in circuit has been programmed, the nonretentive coil is de-energized/cleared, as is shown in Figure 6-26.

Note that, in general, logic coils associated with the bits of a holding register (R) are retentive. Output registers are nonretentive. (See Table 6-4.)

TABLE 6-4. RETENTIVE/NONRETENTIVE COILS

Retentive	Nonretentive
CR "logic" coil	CR "output" coil
BF/R coil	BF/OR coil
BS/R coil	BS/OR coil
BC/R coil	BC/OR coil
Latch coil	

6-8-2. SIM SHUTDOWN CHARACTERISTICS

Unlike many previous Westinghouse programmable controllers, the HPPC-1500/-1700 processor does not **directly** control the states of the field outputs during a power or communications loss. The Serial Interface Module (SIM) interfaces the processor with the I/O modules. Each SIM supports an ALL OUTPUTS OFF/LAST VALID STATE switch, which can be set according to the needs of the individual I/O drop. The SIM introduces a complicating factor in predicting what states field devices will be in since more "loss conditions" are

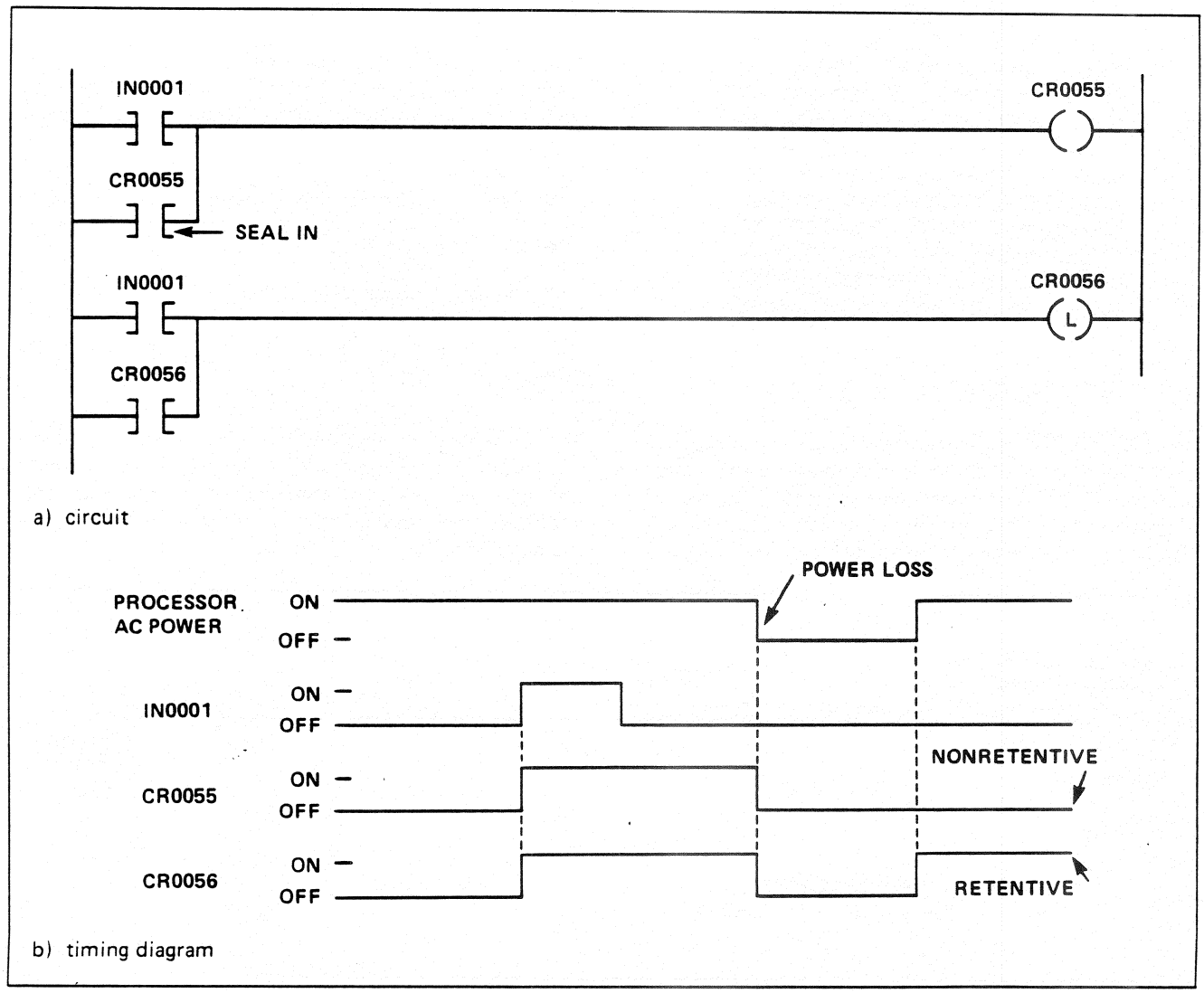


Figure 6-26. Nonretentive Coil Example

possible. There are 4 basic situations which can occur. These are:

1. AC power is lost only at the SIM, not the processor. If power is lost only at the SIM, if it drops beneath the required operating level, or if certain internal operating faults are detected, all field outputs are turned off. (The LAST VALID STATE switch has no effect.) The processor must be reset by performing a Retest Function from the Front Access Panel or Loader.

2. AC power is lost at the processor, not the SIM. If power is lost only at the processor, this unit's shutdown routine includes a transmission of a shutdown signal to the SIM. All outputs are turned off. Once power is restored, the first transmission of data to the SIM controls the field outputs states—which may be different from those the SIM fixed when the loss occurred.

3. Communication is lost between the SIM and the processor, but AC power is constant. If communication is lost between the processor and the SIM, but both units retain AC power, a communication loss is declared. In this case the outputs are controlled by the setting of the LAST VALID STATE/ALL OUTPUTS OFF rocker switch contained on each SIM. Normally, the I/O Processor Module is the master of the SIM-to-I/O Processor communication link. When the SIM does not receive controlling commands, the unit responds by clearing or holding outputs as directed by the LAST VALID STATE/ALL OUTPUTS OFF switch. (The communication loss can occur because of a processor fault, or for any problem causing the transmission of data to be interrupted.)

For operation to resume, an operator must manually initiate a retest on the Advanced Program Loader.

4. The processor's keyswitch is in the STOP:PROGRAM position. If the position is changed from any of the run positions to the STOP:PROGRAM position during normal operation, all outputs are turned off regardless of the setting of the LAST VALID STATE switch on the SIM.

6-8-3. MULTIPLE COILS ON CIRCUIT

One of the powerful programming features of the HPPC-1500/-1700 is that it allows two or more coils to be contained in a single interconnected network. See Figure 6-27(a) where output coils are connected in parallel.

Figure 6-27(b) shows that 2 or more coils can be connected to different contacts in an interconnected network.

These 2 basic configurations can be combined in a larger, interconnected network, as shown in Figure 6-28. Study the Figure and note that:

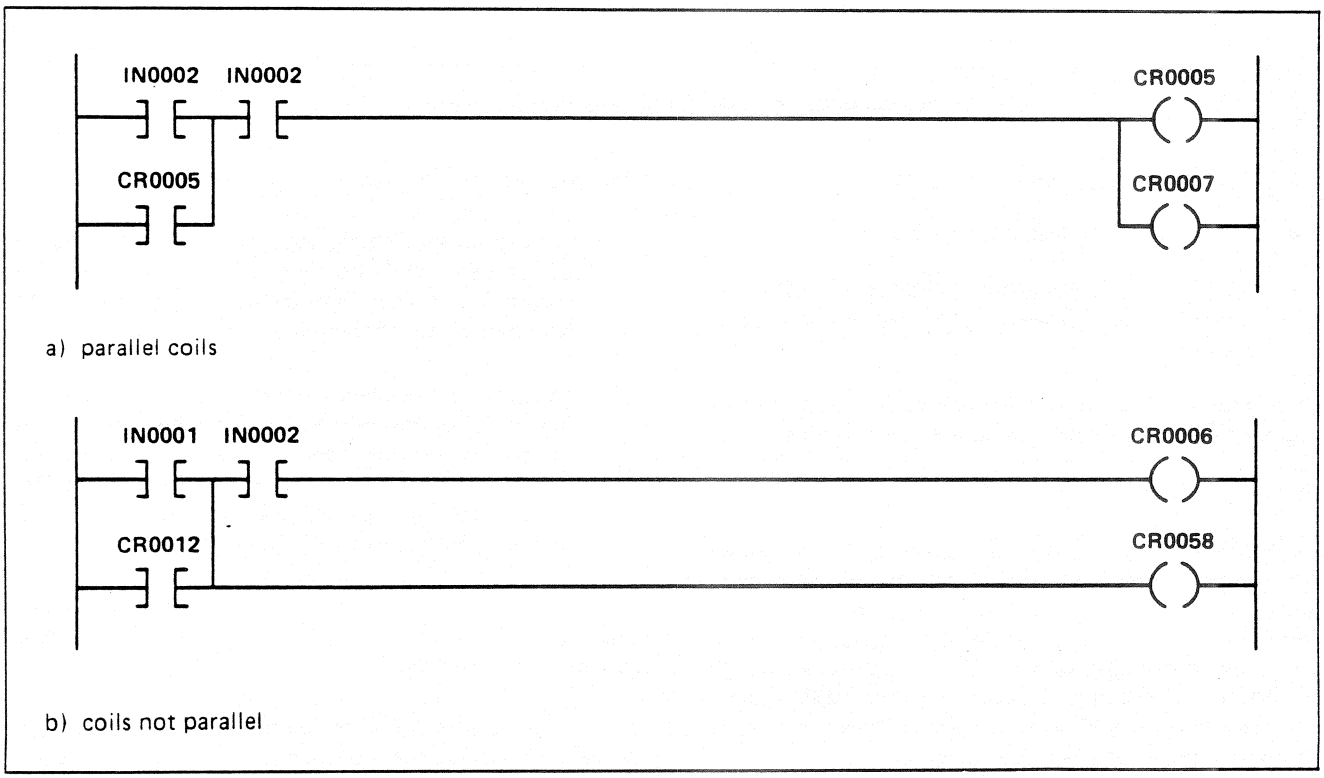


Figure 6-27. Multiple Coils

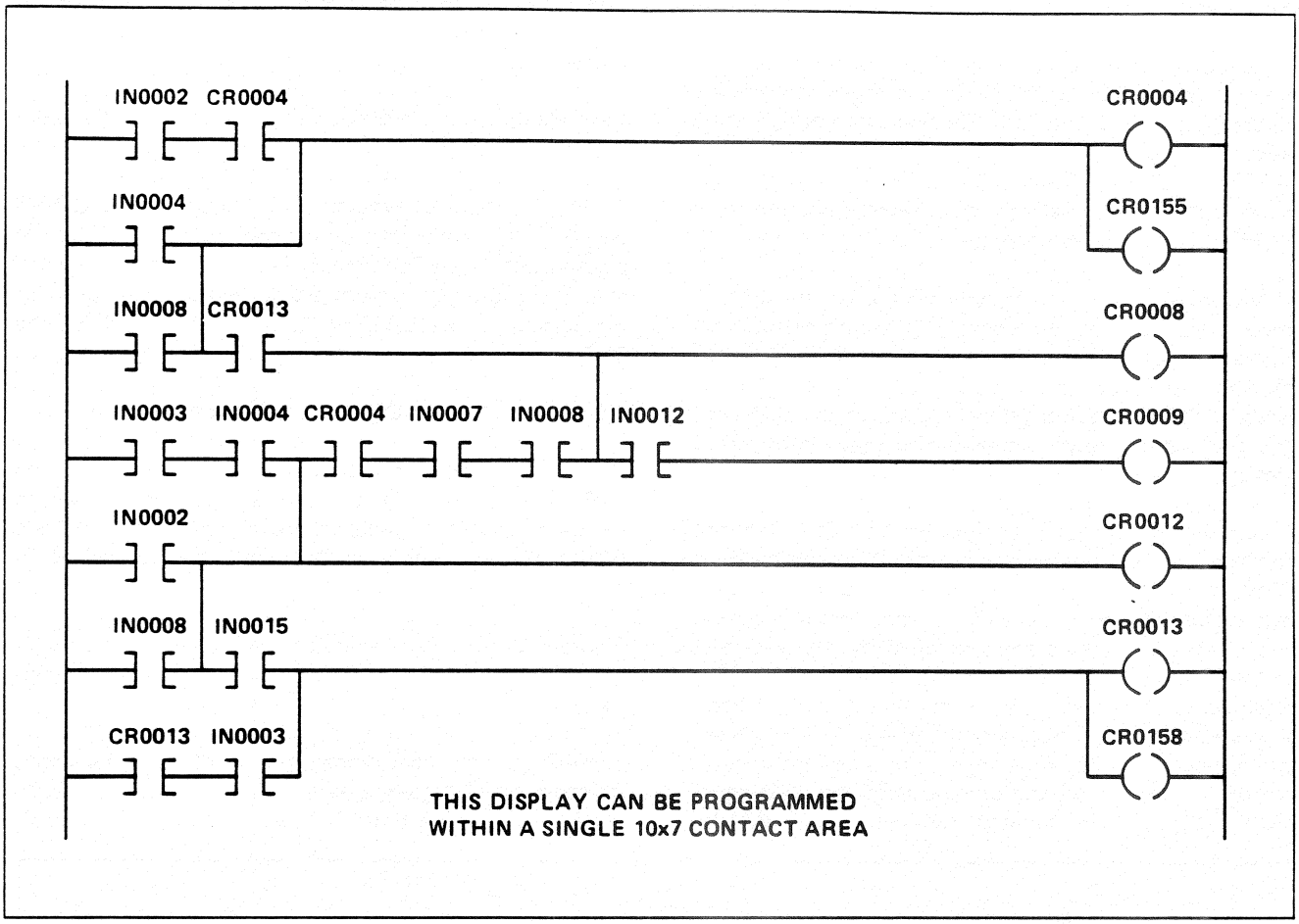


Figure 6-28. Extended Multiple Coil Example

- CR0004 and CR0155 are in parallel
- CR0013 and CR0158 are in parallel
- The other coils (CR0008, CR0009, CR0012) are connected to different contacts of the same network.

A maximum of 7 coils, one in each parallel path, can be programmed into a single interconnected network.

6-9. LADDER DOCUMENTATION

The HPPC-1500/-1700 programmable controller allows user-written contact/coil identification and text comments to be added to the application program ladder diagram. This information is useful for troubleshooting, editing, and personnel training. Entries are made through the Advanced Program Loader's keyboard. This data may then be displayed on the Loader's screen or printed out on hardcopies of the ladder diagram.

(The entry method is detailed in the APL Programming Manual, Catalog No. NLAM-B806.)

Documentation is divided into 2 forms:

- **Special labels,** which appear directly above each CR or Latch coil and IN or CR contact label. (See Figure 6-29.) These special labels range from 1 to 6 alphanumeric characters.
- **Text,** which appears directly below the contact, special function and coil circuits. The text consists of messages the user decides are useful to maintenance, operator, or applications personnel. The text may consist of up to a maximum of 3 lines, each containing up to 80 characters.

The characters used for the documentation can be any legal ASCII characters. The text areas may be in upper and lower case letters.

The documentation is entered into the memory of the processor through the Advanced Program Loader. Each special label and text word requires a predetermined area of the documentation memory. See APL Programming Manual, Section 10, I/O Labels for more information.

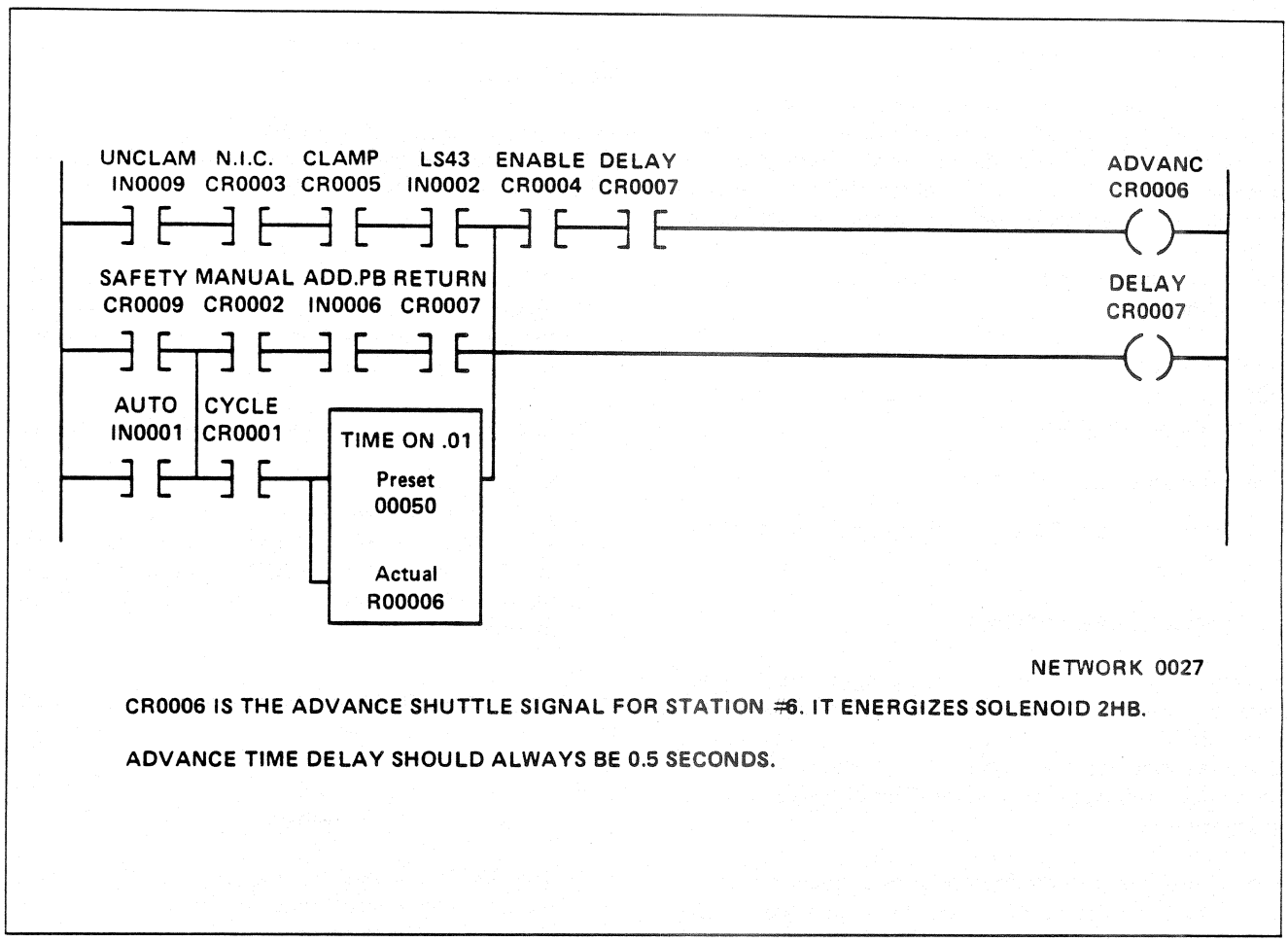


Figure 6-29. Ladder Documentation

6-10. NETWORKS

The Advanced Program Loader assigns "network numbers" to each ladder diagram segment which appears on the screen **and** has been entered into the processor's memory as a unit. (This statement assumes on-line operation.) A network number appears at the lower right-hand corner of the display. (See Figure 6-29.)

Any of the following program segments are given a network number by the Loader:

- A simple circuit containing at least a coil or special function.
- A single, interconnected group of contacts, special functions, and coils.
- Two or more groups of contacts, special functions, and coils. (For example, 7 totally unconnected circuits.)
- Any combination of separate or interconnected contacts, special functions, coils, and user-written documentation comments.

Network numbers appear on hardcopy printouts of the ladder diagram. These may then be used by the programmer as references to quickly display specific program segments. (The Loader must be in the Search mode, and NETWORK (F5) must be selected.)

In general, Westinghouse recommends that each separate, interconnected circuit be assigned as a unique network, even if it is very simple. A full explanation of this topic is given in the APL Programming Manual.

More detailed information on the subject of networks is given in Paragraph 10-2.

6-11. NUMBERING SYSTEMS

The descriptions of the programmable functions included in the following Sections use a variety of numbering systems to express the concepts related to the processor's operations. These include decimal, binary, BCD (binary coded digits), and 2's complement forms of binary. As much as possible, the specific numbering system is indicated near the stated digit(s) for reader convenience.

Numbering systems as applied to the HPPC-1500/-1700 and described here can also be divided into the following general 2 areas for discussion:

- Math operations (6-11-1)
- Front Access Panel or APL Loader monitoring (6-11-2)

6-11-1. MATH OPERATIONS

Many of the standard and advanced programmable functions perform math operations by using binary constants, or "values," contained in registers. These binary values represent numeric data. The processor considers the values associated with math operations to be true binary; that is:

- If positive, the values consist of a binary representation consisting of 1s and 0s, which are binary digits.
- If negative, the values are represented by the corresponding 2's complement of the binary numbers.

The processor recognizes the values associated with math operations as being negative when the most significant bit of a register is a logic 1.

6-11-2. FRONT ACCESS PANEL OR APL LOADER MONITORING

The Front Access Panel and APL Loader allow the user to enter, load, and view data contained in registers in a number of formats, as noted here:

- The decimal equivalent of a single-precision (16-bit) register, or a double-precision (32-bit) register can be displayed. Note: The conversion to the decimal display from binary is automatic and transparent to the user.
- The hexadecimal equivalent of a 16-bit register, where each 4 consecutive bits are converted to a hexadecimal value (from 0 to F) and then displayed. The conversion results in a range of values from 0000 to FFFF. The hexadecimal format can also be used to display BCD values since BCD and hexadecimal values in the range of 0 to 9 are identical.

In addition the APL Loader also allows the user to observe data contained in registers in two other formats:

- The binary data contained in any register can be displayed directly in binary.
- The ASCII equivalent characters of a 16-bit binary register, where each group of 8 bits is displayed as a single ASCII character.

The relationship among the various formats is shown in the examples contained in Table 6-5. Observe the Table and note the following items:

1. The ASCII result of seven 0s (as shown for the eight MSDs of R00003) is the absence of a character, not

TABLE 6-5. FORMAT EXAMPLES

Register	Binary Value		Decimal	Hex	ASCII
	MSD	LSD			
R00003	0000 0000	0100 0110	00070	0046	F
R00004	0111 0101	1000 1000	30088	7588	u^e
R00007	1000 1010	0111 0111	-30089	8A77	`ew

0—as might be assumed. This is technically called a "null."

2. Whenever a logic 1 state is contained in bit 16 of the register, the number is assumed in 2's complement form.

Table 6-5 does not include an example of a double-precision number. A double-precision value is handled by the processor as a 32-bit register in which 2 consecutively numbered registers are linked together. In the HPPC the first, or lower numbered, register contains the most significant bits, while the second consecutively numbered register contains the least significant bits. (See Figure 6-30.)

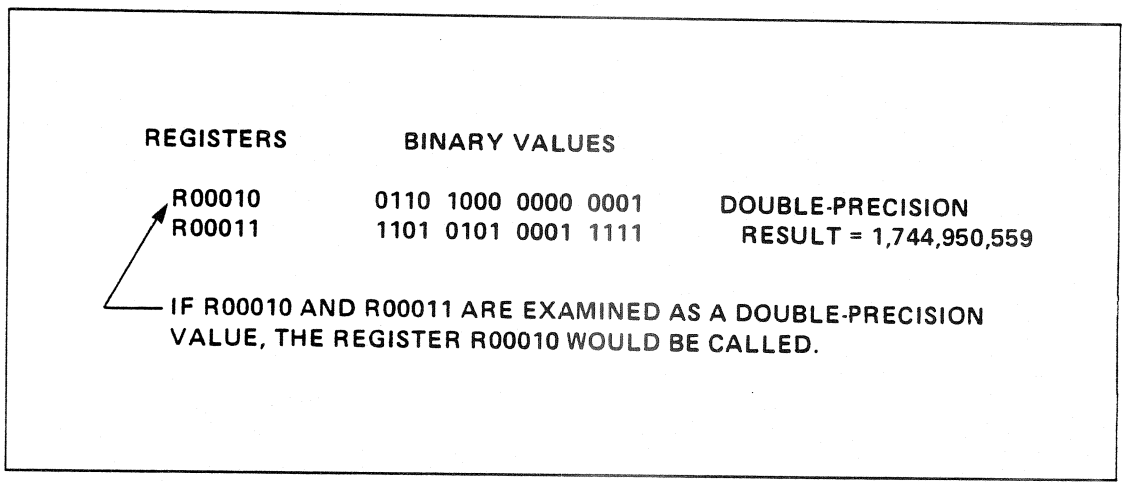


Figure 6-30. Double-Precision Example

Section 7

Standard Functions

7-1. INTRODUCTION

The HPPC-1500/-1700 programmable controller offers the user more than 80 software functions which can be used in the construction of a ladder diagram application program. It is the purpose of this Section to explain the functions that make up the "standard" programmable functions. Section 8 explains the "advanced" programmable functions.

7-1-1. STANDARD vs. ADVANCED

Every HPPC-1500/-1700 is supplied with the standard set, which is made up of approximately 36 programmable functions. Optionally, users may order the advanced set, which is made up of the remaining programmable functions. These functions are resident on PROMs located on the Control Store Board (NLCS-1510 or -1520). See Paragraph 3-6 for more details.

One simple means of identifying which programmable functions are supported by a specific HPPC-1500/-1700 mainframe chassis is to consult the nameplate. A -1 suffix on the Catalog No. indicates a standard set; a -2 suffix indicates an advanced set. Alternately, the PROMs on the Control Store Board can be counted. The standard set requires 7; the advanced set requires 14.

7-1-2. PROGRAMMABLE FUNCTION DISPLAY

When the HPPC-1500/-1700 is in the Program mode, a display of the complete list of both standard and advanced programmable functions can be activated by pressing the F4/SF key. (There are two displays due to the extensiveness of the list, and so it may be necessary to press the F9/page key.) The functions are organized according to general groupings such as coils, timers, math functions, etc. (See Figure 7-1.)

No distinction is made between the standard and advanced set on the display. However, if an attempt is made to program an advanced function in an HPPC-1500/-1700 system that contains only the standard function, the message UNSUPPORTED SF is flashed across the bottom on the Loader's display screen. (SF here means "special function;" that is to say, "advanced.")

7-1-3. NUMBERING SYSTEM

There is a 3-digit number in front of each item on the programmable function menu display. This number serves as an ordering principle. It is also used, by means of operator keyboard entries, to place an individual function into the circuit being constructed by the programmer. (Complete entry techniques are given in the APL Programming Manual.) See Figure 7-1.

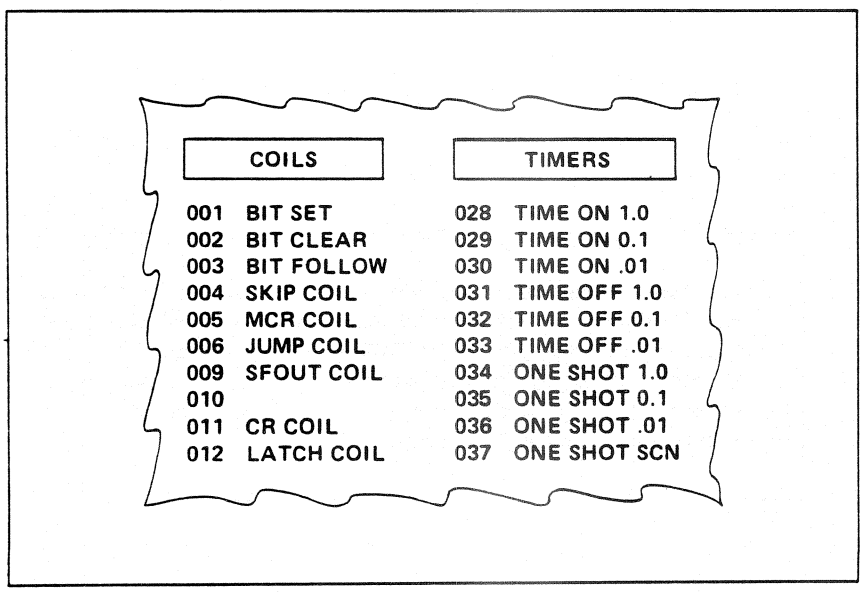


Figure 7-1. Programmable Function Display Menu (Partial)

This manual uses these 3-digit numbers in the basic format of Sections 7 and 8. They appear at the top and bottom left or right corners of each page to aid the user to locate a specific programmable function. Table 7-1 contains a list of the standard functions, along with the display screen's 3-digit number. With this information, the locating of a specific programmable function explanation is an easy matter.

Note

In cases where two, or more, programmable functions can be discussed together, the related 3-digit numbers all appear, in sequential order, on the corners of the page.

7-1-4. FUNCTIONS ALREADY DISCUSSED

Five of the standard programmable functions have already been explained in Section 6, General Programming Concepts. These are:

- 001 Bit Set
- 002 Bit Clear
- 003 Bit Follow
- 011 CR Coil
- 012 Latch Coil

They were included in the previous Section in order to explain the basic concept of the ladder diagram circuit. The functions included here begin with 004 Skip Coil.

7-1-5. ORDERING OF FIGURES, TABLES

For simplicity and convenience, subsequent figures and tables in these Sections are numbered according to 3 alphabetical letters that represent the function's name. A numerical suffix indicates the specific figure or table within the discussion of the function.

7-1-6. GENERAL FUNCTION GROUPINGS

Many of the standard and advanced programmable functions are placed in one of the following groupings:

- Coils
- Signal conditioners
- Labels
- Timers
- Counters
- Math functions
- Shift registers
- Conversions
- Move functions
- Logic functions
- Table operations
- Compare functions
- I/O functions

Note

For instructional purposes, certain aspects of the Advanced Program Loader displays have been exaggerated in the typical examples which follow. This is especially true with respect to the spacing between groups of data and some circuits. The content of the displays, however, reflects the actual CRT messages.

TABLE 7-1. STANDARD FUNCTIONS

Type	Display No.	Title	Par. No.	Page No.
Coils	001	Bit Set	①	6-16
	002	Bit Clear	①	6-16
	003	Bit Follow	①	6-15
	004	Skip Coil	7-2	7-4
	005	MCR Coil	7-3	7-7
	011	CR Coil	①	6-13
	012	Latch Coil	①	6-18
Signal Conditioners	007	Transitional	7-4	7-10
Labels	021	MCR Label	7-3	7-7
	022	Skip Label	7-2	7-4
Timers	028	Time On 1.0	7-5	7-13
	029	Time On 0.1	7-5	7-13
	030	Time On .01	7-5	7-13
	034	One Shot 1.0	7-5	7-14
	035	One Shot 0.1	7-5	7-14
	036	One Shot .01	7-5	7-14
	037	One Shot SCN	7-5	7-14
Counters	038	Up Counter	7-6	7-18
	039	Down Counter	7-6	7-18
Math Functions	068	Add A + B=C	7-7	7-23
	070	Sub A - B=C	7-7	7-23
	072	Mult A x B=C	7-8	7-27
	073	Div A ÷ B=C	7-9	7-30
Shift Registers	040	Shift Left 1	7-10	7-33
	041	Shift Right 1	7-10	7-33
Conversions	045	BIN-BCD R	7-11	7-39
	046	BIN-BCD*R	7-12	7-42
	047	BCD-BIN R	7-13	7-45
	048	Analog In	7-14	7-48
	049	Analog Out	7-15	7-51
Move Functions	050	Move R-R	7-16	7-53
	096	Drum Control	7-17	7-56
Logic Functions	051	Zero Table	7-18	7-62
Table Operations	024	Diagnostic	7-19	7-64
Comparison	053	Compare R-R	7-20	7-67
I/O Functions	054	Update I/O	7-21	7-69

① Described throughout Paragraph 6-7.

7-2. SKIP COIL (004), SKIP LABEL (022)

The Skip Coil and Skip Label programmable functions allow the execution of a portion of the ladder diagram to be selectively skipped. The portion skipped is determined by the location of the Skip Coil and Skip Label in the ladder diagram. (See Figure SKP-1.) Observe the Figure and note the following 4 points:

1. The same reference number is assigned to **both** the Skip Coil and the Skip Label functions. However, each distinct pair of Skip functions should be assigned a completely different reference number.

2. When the enable circuit is nonconducting, the Skip Coil is nonconducting, and the ladder diagram is executed normally.

3. When the enable circuit conducts, the Skip Coil conducts. At that time the portion of the ladder diagram between the Skip Coil and Skip Label is not executed. The coils are held in their last states. All standard and advanced functions such as timers or counters are also held in their last states.

4. The Skip Label function is programmed in the upper left portion of a network having no other contacts.

7-2-1. SPECIFICATIONS

The programming specifications for the Skip Coil and Skip Label functions are listed here.

SKIP COIL

The Skip Coil is de-energized when the enable circuit is nonconducting. During this time normal scanning and execution of the ladder diagram program occur. When the Skip Coil is energized, the portion of the ladder diagram located between the Skip Coil and Skip Label functions is not executed by the processor. This results in the following 3 points:

1. All coils are held in their last states—**except** coils being forced.

2. All standard and advanced functions such as timers and counters are held in their last states.

3. Register data is held in its last state, unless the register is also affected by a coil or function that is located in the ladder diagram outside the Skip Functions.

The Skip Coil must be programmed in the upper right corner of the network.

Note

The Skip Coil should be programmed in a network containing no other coils. Other coils programmed in the same network are **not** controlled by the Skip Coil function.

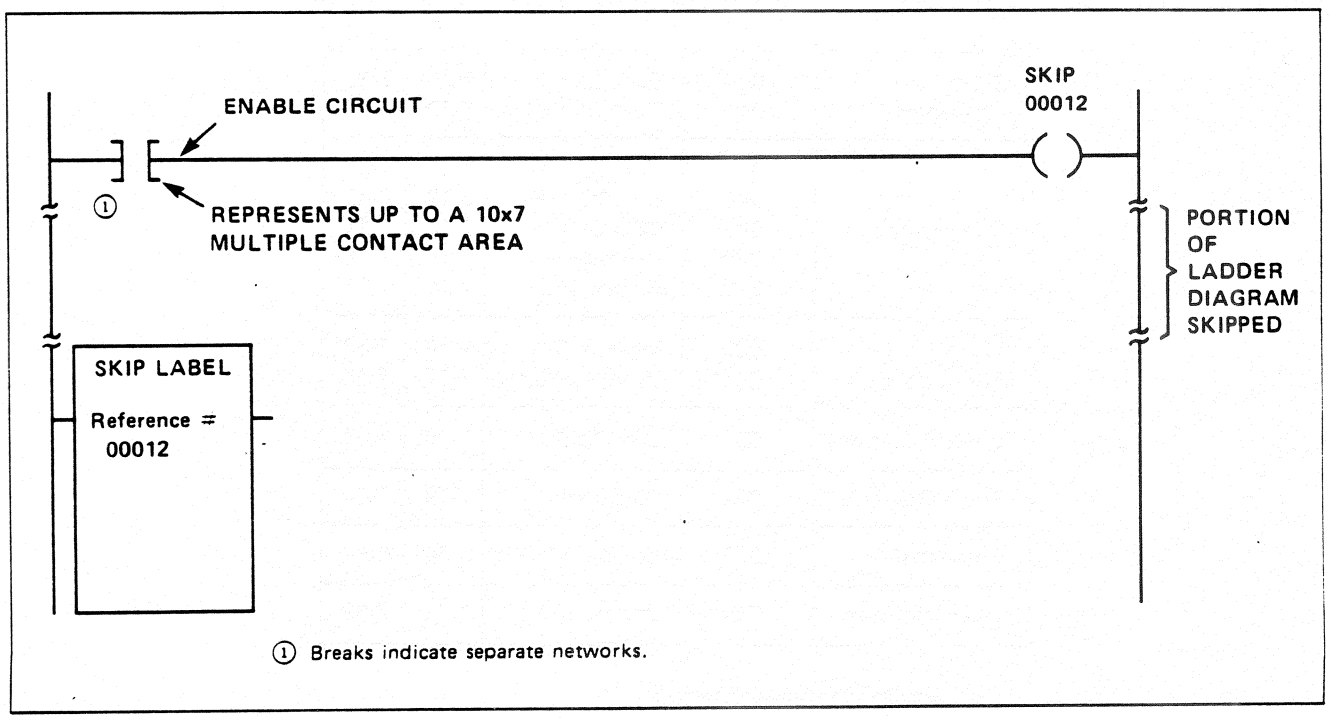


Figure SKP-1. Skip Function Circuit (Typical)

SKIP LABEL

The Skip Label function block must be located in the top left portion of a network without any other contacts or coils in the circuit. (Keep in mind that Figure SKP-1 assumes many ladder diagram circuits that are not shown.) One or more Skip Coils located at different positions in the ladder diagram can use the same reference number with a single Skip Label.

ENABLE CIRCUIT

The enable circuit for the Skip Coil consists of up to 10 horizontal contact spaces by 7 parallel paths. The rules which apply to the programming of contacts listed in Section 6 apply here, also.

REFERENCE NUMBER

The reference number assigned to both the Skip Coil and Skip Label pair must be the same. The reference number is specified by the programmer in the range from 0001 to 9999. Each distinct Skip Coil and Skip Label pair normally uses a completely different reference number, except where 2 or more Skip Coils reference the same Skip Label. An individual Skip Coil function must be located first in the ladder diagram sequence before a Skip Label function.

NESTED SKIP FUNCTIONS

It is permissible to have nested and/or overlapping Skip functions programmed together in the ladder diagram. However overlapping "skips" make the ladder diagram program more difficult to understand.

7-2-2. APPLICATIONS

The Skip function can be used when a series of outputs must be updated at a predefined interval such as every 5 or 10 seconds. Figure SKP-2 shows an example where a number of coils are updated after a 10-second interval. When the timing circuit to the timers is enabled for 10 seconds, the coil associated with the timer is enabled for 1 scan before the BP/R contact resets the timer. During that single scan, the Skip Coil de-energizes, thereby updating the coils. After that single scan, the $\overline{\text{reset}}$ circuit resets the timer. Each 10 seconds the cycle repeats, as long as IN0001 remains conducting.

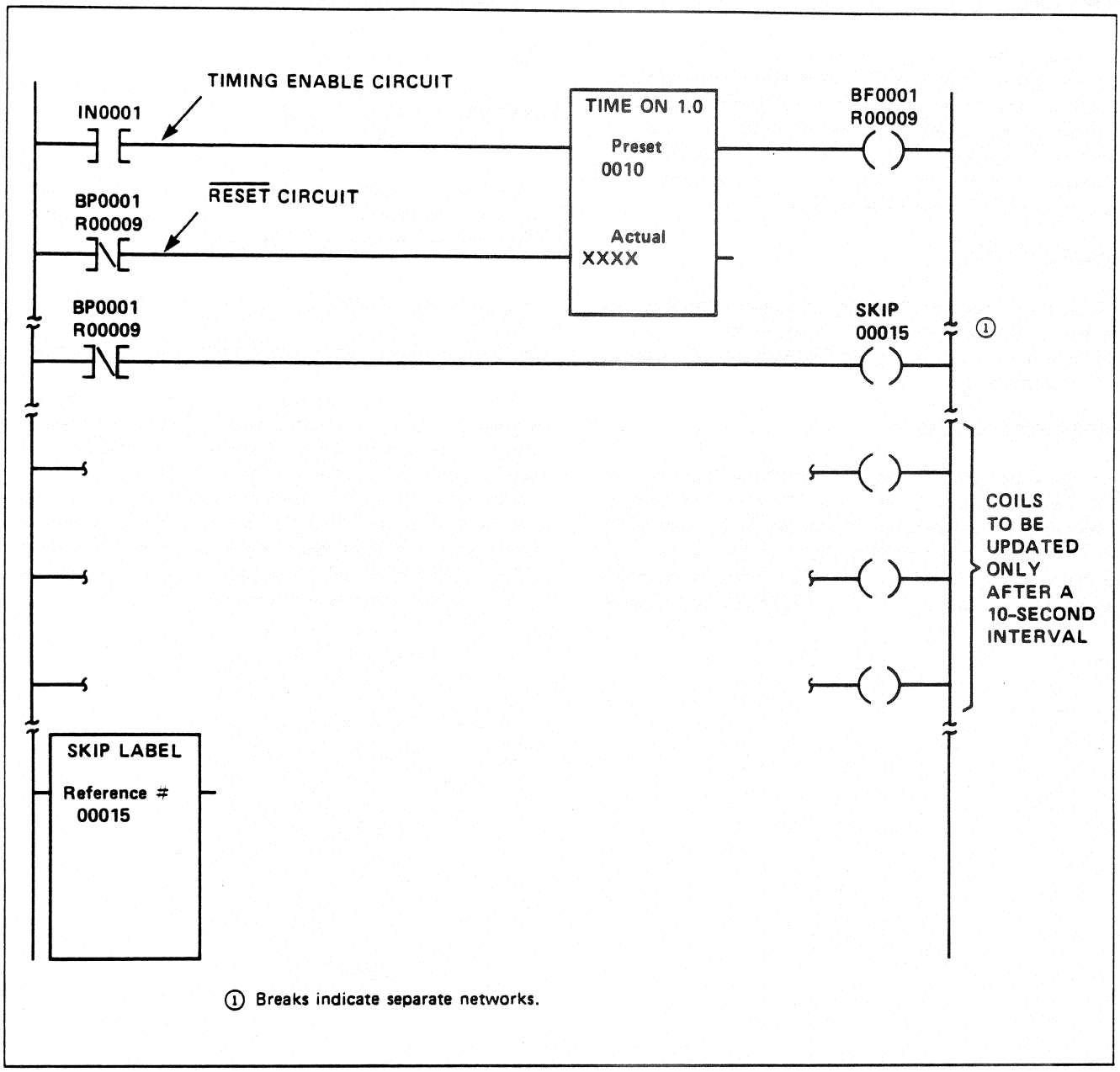


Figure SKP-2. Skip Application Example

7-3. MCR COIL (005), MCR LABEL (021)

The MCR Coil and MCR Label programmable functions allow selective disabling of a portion of the ladder diagram. The portion is determined by the reference number specified with the functions. (See Figure MCR-1.) Observe the Figure and note the following 4 points:

1. The same reference number is assigned to both the MCR Coil and MCR Label functions. However each distinct pair of MCR functions should be assigned a completely different reference number.

2. When the enable circuit conducts, the MCR Coil conducts, and the ladder diagram is executed normally.

3. When the enable circuit to the MCR Coil does not conduct, the portion of the ladder diagram between the MCR Coil and MCR Label is disabled. The coils are de-energized, and the standard and advanced functions stop operating.

4. The MCR Label must be located in the upper left portion of a network containing no other contacts or coils.

7-3-1. SPECIFICATIONS

The following programming specifications apply to the MCR functions:

MCR COIL

The MCR Coil energizes when the enable circuit conducts, thus allowing normal scanning and execution of the ladder diagram. When the MCR coil de-energizes, the portion of the ladder diagram between the MCR Coil and MCR Label is disabled, as listed in the following 3 points:

1. All coils are de-energized—except the coils being forced.

2. All standard and advanced functions such as timers and counters are held in their last states.

3. Register data remains at its last value, unless the register is also affected by a coil or function that is located in the ladder diagram outside the MCR function.

Note

The MCR Coil should be programmed in a network that does not contain other coils. Other coils programmed in the **same** network will **not** be controlled by the MCR Coil function.

MCR LABEL

The MCR Label function block must be located in the

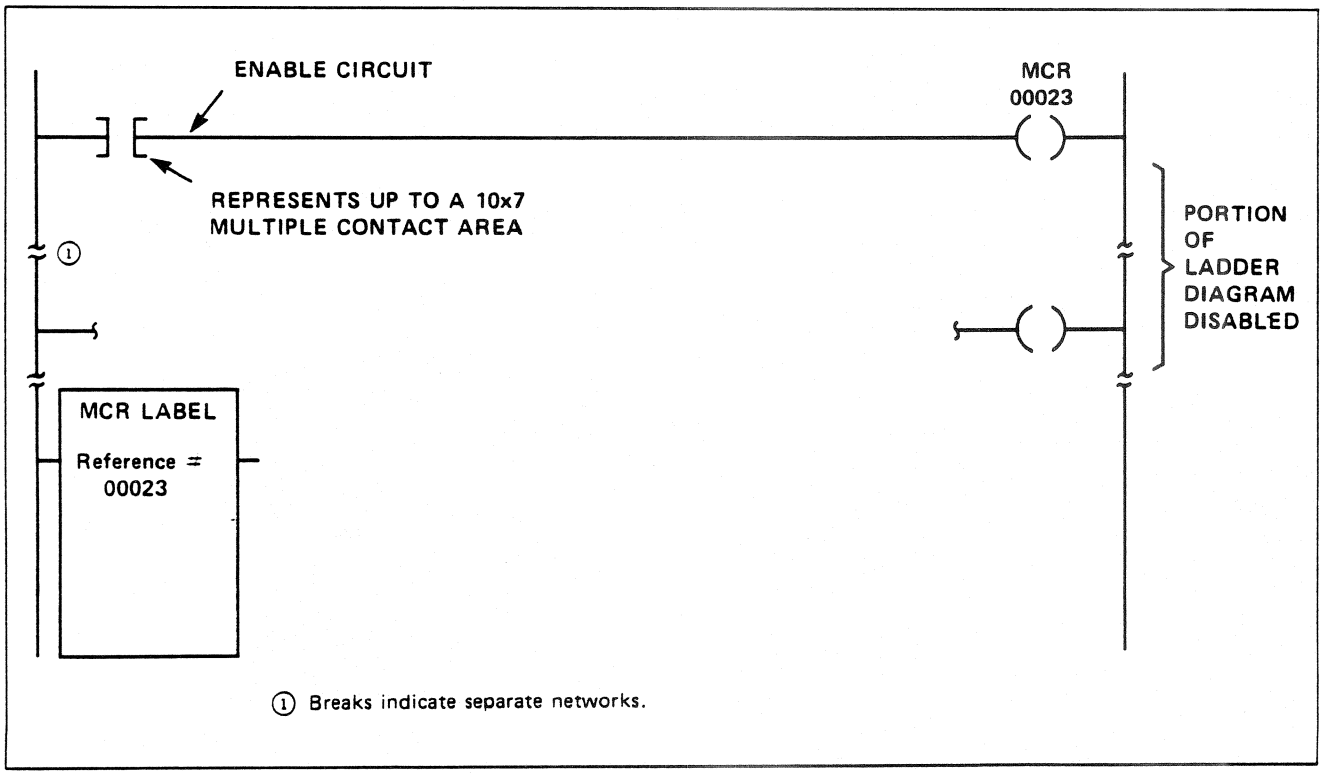


Figure MCR-1. MCR Function Circuit (Typical)

top left portion of a network without any other contacts or coils in the circuit. One or more MCR Coils located at different positions in the ladder diagram can use the same reference number with a single Skip Label.

ENABLE CIRCUIT

The enable circuit for the MCR Label consists of up to 10 horizontal contact spaces by 7 parallel paths. The rules which apply to the programming of contacts located in Section 6 apply here, also.

REFERENCE NUMBER

The reference number assigned to both the MCR Coil and MCR Label pair should be the same. The reference number specified by the programmer ranges from 0001 to 9999. Each distinct MCR Coil and MCR Label pair normally uses a completely different reference number, except where 2 or more MCR Coils reference the same Skip Label. An individual MCR Coil function must be located in the ladder diagram sequence before its associated MCR Label function.

NESTED MCR FUNCTIONS

It is permissible to have nested and/or overlapping MCR functions programmed. However, overlapping MCRs make the ladder diagram program more difficult to understand.

7-3-2. APPLICATIONS

When the incoming AC power is enabled, the processor automatically resets and begins execution of the ladder diagram. However, in many applications it is not desirable to automatically reset after a power interruption. Figure MCR-2 shows the MCR function used to prevent the processor restart until a manual signal is given.

All output coils including CR0001 shown in Figure MCR-2 are de-energized when a power failure occurs. The MCR Coil is false until IN0001, a manual push-button, is depressed sometime after AC power is restored. When IN0001 is depressed, coil CR0001 is energized, thereby enabling the MCR Coil and allowing the ladder diagram to be executed normally.

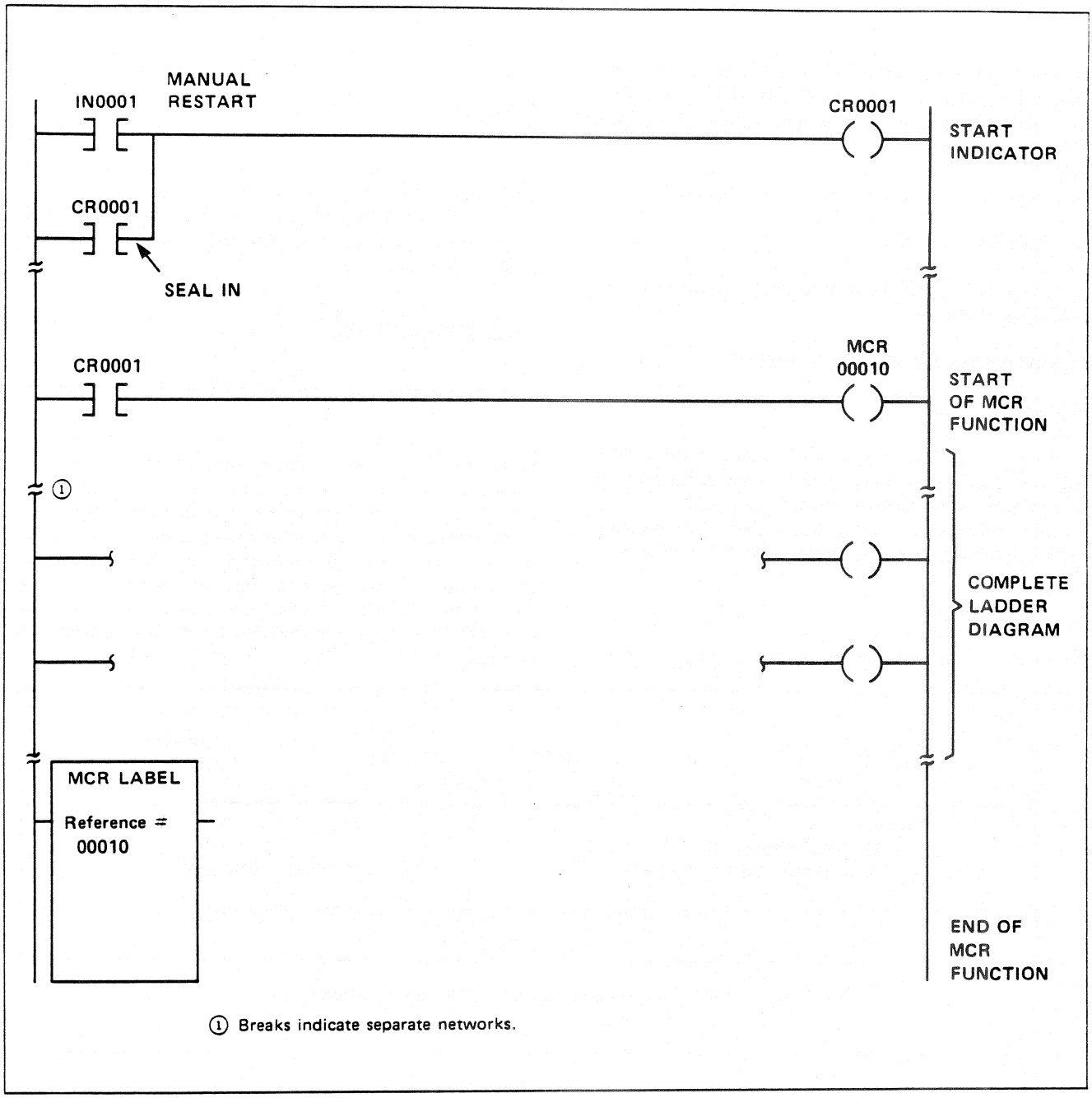


Figure MCR-2. Manual Restart Circuit Example

7-4. TRANSITIONAL (007)

The Transitional programmable function monitors the conduction status of contacts located to the left of the Transitional function on the ladder diagram. (See Figure TRN-1.) When the input to the Transitional function first conducts, the output from the Transitional function conducts for a single scan of the ladder diagram.

7-4-1. SPECIFICATIONS

The following programming specifications apply to the Transitional function:

TRANSITIONAL FUNCTION ELEMENT

The Transitional function requires the same space on the ladder diagram as an ordinary contact. It can be located anywhere in the 10x7 contact area, although **the Transitional function must always be positioned to the right of any contacts in the circuit.** (See Figure TRN-1.) (The rules for positioning of contacts of the ladder diagram listed in Section 6 apply to positioning the Transitional function.)

ENABLE CIRCUIT

The enable circuit, or input to the Transitional function, can consist of up to 9 horizontal contacts located in up to 7 parallel paths.

OUTPUT

The output of the Transitional function conducts for one ladder diagram scan only. This is when the input—or "enable circuit"—first conducts; that is, goes from an off to an on conducting state. See the timing diagrams of Figure TRN-2.

7-4-2. APPLICATION

The Transitional function can be used to allow an Up Counter function (038) to increment on the leading or trailing edge of a contact closure. (See Figure TRN-3.) When the Inverting function (008)—described in Par. 8-2—is placed in series with the Transitional function, the Actual Value Register of the Counter function is incremented by one whenever the contacts of the enable circuit change from a conducting to a nonconducting state. Without the Inverting function in the enable circuit of the Counter function, it counts the ladder diagram program scans whenever the timing enable circuit is conducting.

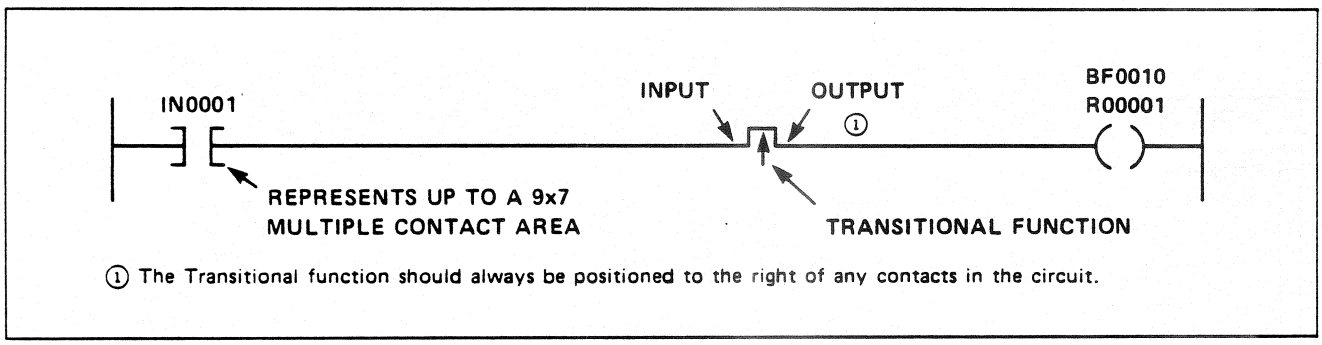


Figure TRN-1. Transitional Function Circuit (Typical)

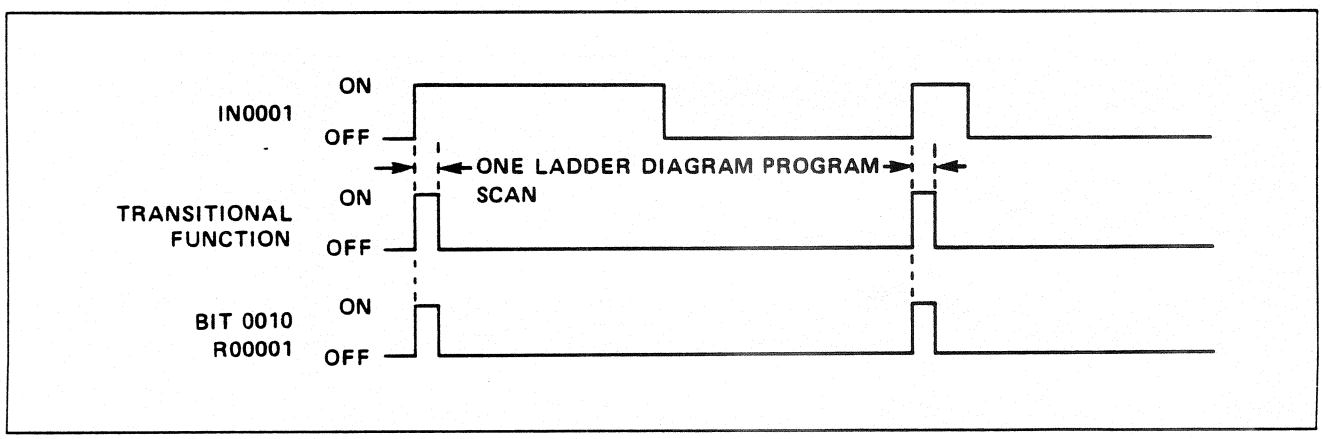


Figure TRN-2. Transitional Function Timing Diagram

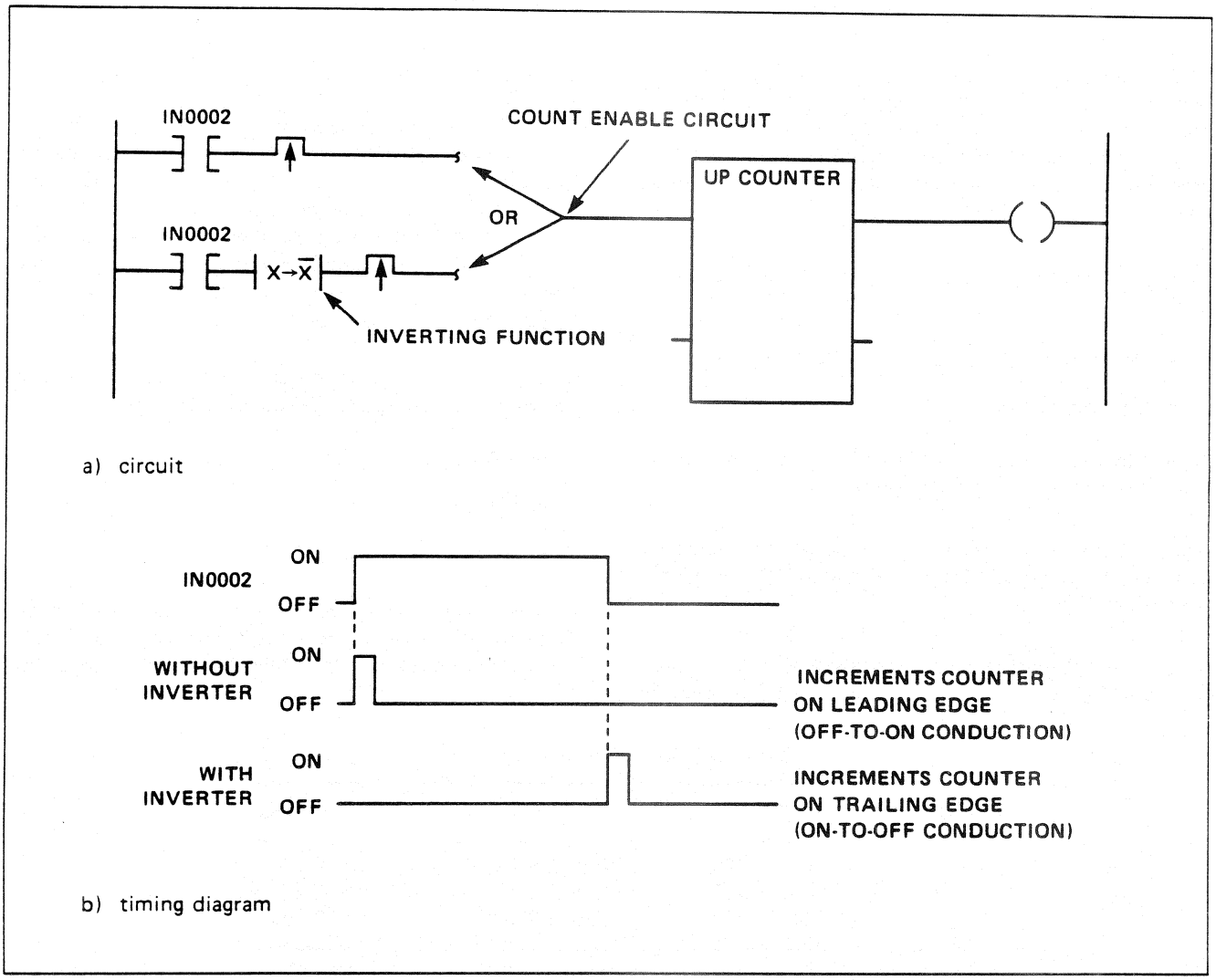


Figure TRN-3. Transitional Function Application Example

7-5. TIMERS (028-030, 034-037)

The Timer programmable functions are programmed to operate like any of the common type of electro-mechanical timers. The timer functions can be used either independently or in conjunction with other programmable functions in order to develop complex timing chains.

Timers are organized into 2 groups, as shown in Table TMR-1:

- On-delay timers
- One-shot timers

Note, however, that the off-delay timers are part of the advanced programmable function set.

Many of the characteristics of timers are the same regardless of the group. Observe Figure TMR-1 and note the following 5 points:

1. The timing enable circuit consists of one or more contacts located to the upper left of the timer function block on the ladder diagram. The timing circuit can be considered as in "input" to the timer. When the timing enable circuit conducts, the timing is enabled.

2. The $\overline{\text{reset}}$ circuit, used only with the Time On function, consists of one or more contacts located on the ladder diagram to the lower left of the Time On function. This circuit can be considered as the input to the timer that clears the value contained in the Actual Register. When the $\overline{\text{reset}}$ circuit conducts, the timer is functional. When it does not conduct, the timer is reset. (The actual value is cleared to 0.)

3. The Actual Value Register accumulates the time in one of the following user-selected increments:

- 1.0 second
- 0.1 second
- 0.01 second

4. The Preset Value Register determines the exact on-delay duration. When the actual value equals the preset value, the Q output of the timer is affected, as described next.

TABLE TMR-1. TIMER GROUPS

Groups	Display No.	Type	Timing Range
On-delay timers	028 029 030	Time On 1.0 Time On 0.1 Time On .01	1 to 32,767 sec. 0.1 to 3,276.7 sec. 0.01 to 327.67 sec.
One-shot timers	034 035 036 037	One Shot 1.0 One Shot 0.1 One Shot .01 One Shot SCN	1 to 32,767 sec. 0.1 to 3,276.7 sec. 0.01 to 327.67 sec. 1 to 32,767 scans

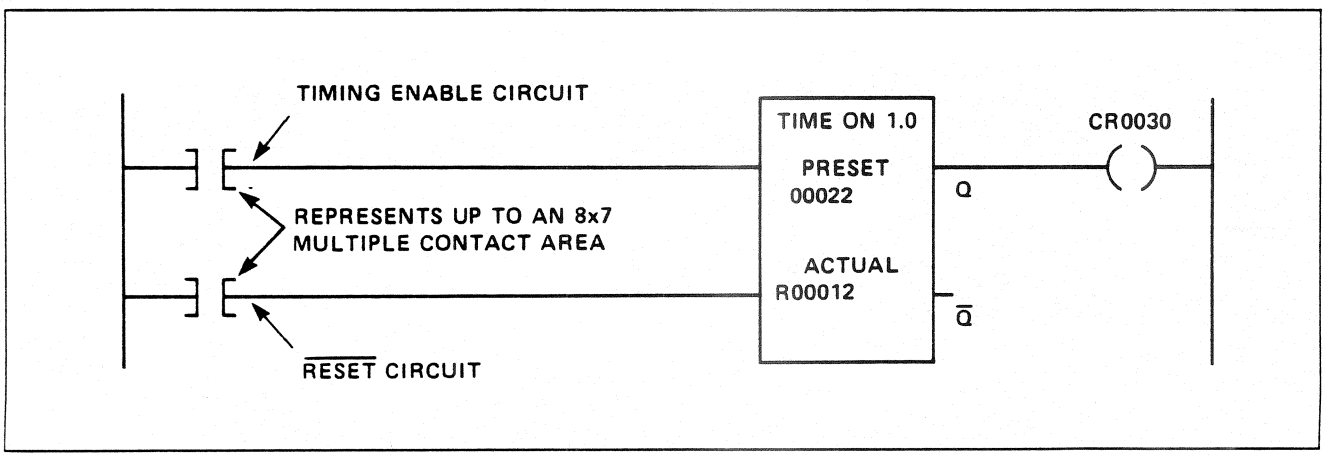


Figure TMR-1. Timer Circuit (Typical)

5. The Q and $\overline{Q}$ connections of the timer can be considered the outputs from the timer. They are always in opposite states. Either or both of these outputs can be used to provide a conducting path to other functions or to coils. The Q output changes state (from conducting to nonconducting) in response to the preset and actual values, as described for each timer function.

The Time On functions and the One Shot function are described separately in following subparagraphs.

7-5-1. TIME ON FUNCTIONS

The Time On function provides, as the name implies, an on-delay timing function. The 3 different Time On functions operate identically except that the time increments and their associated timing ranges vary, as shown in Table TMR-1. An example of a typical Time On function circuit is shown in Figure TMR-2. Observe the Figure and note the following 3 points:

1. The value of the Actual Register increments (times) whenever the timing enable circuit conducts and the reset circuit conducts.

2. The value of the Actual Register is reset to zero whenever a $\overline{\text{reset}}$ circuit is nonconducting—regardless of the state of the timing enable circuit. See actual value at

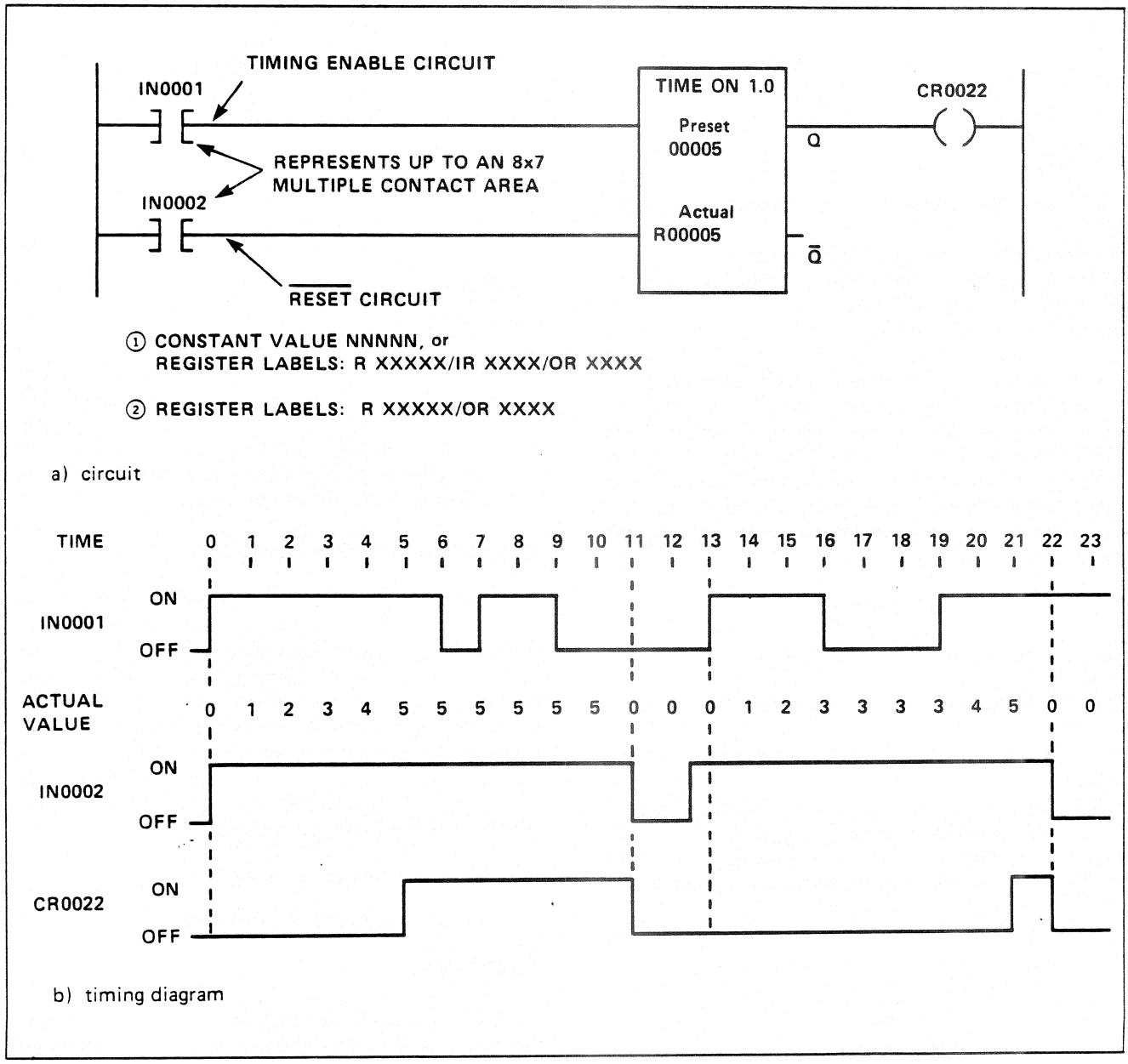


Figure TMR-2. Time On Function Circuit (Typical)

times 11 and 22 of Figure TMR-2. These are zero.

3. The Q output conducts when the Actual Value equals or exceeds the Preset Value. See Actual Value times 5 thru 11 and 21 and 22 of Figure TMR-2. In this example the Q output is used to control coil CR0022.

Table TMR-2 contains a truth table further describing the timing enable and reset circuit functions.

7-5-2. ONE SHOT FUNCTIONS

The One Shot functions provide a pulsed output when the timing enable circuit first conducts. The 4 different One Shot functions operate identically except for the following user-selected time bases:

- 1.0 second
- 0.1 second
- 0.01 second
- Count the number of ladder diagram scans (SCN)

Selections for the various time bases are shown in Table TMR-1. An example of a typical One Shot function circuit is shown in Figure TMR-3. Observe the Figure and note the following 2 points:

1. The value of the Actual Register is zeroed, and the Q output conducts immediately when the timing enable circuit first **conducts**. In this case, it is when switch IN0001 switches from off to on. (See times 0, 8, 10 and 16 of Figure TMR-3.) The one-shot timer is therefore an edge-triggered function. The rising edge, or the false-to-true transition, of the timing enable circuit initiates the timing or scan counting.

2. The Q output conducts when:

- The timing enable circuit first conducts, and
- The actual value is not yet equal to the preset value

(See times 0 thru 4, 8 thru 14, and 16 thru 20 of Figure TMR-3.) When the Actual Value equals the Preset Value, the Q output stops conducting. In this case the Q output controls coil CR0017 which is disabled at times 4, 14 and 20 when the Preset Value equals the Actual Value.

The truth table for the timing enable circuit of the One Shot function is shown in Table TMR-3.

Note

The Actual Value of the timers is retentive through an AC power loss as is the time duration of the output of the One Shot. In preparation for AC power restoration, special precautions must be taken to prevent the One Shot from completing a pulse initiated before AC power was removed.

TABLE TMR-2. TIME ON TRUTH TABLE

Timing Circuit ①	$\overline{\text{Reset}}$ Circuit ①	Result
0	0	Actual value held to 0
1	0	Actual value held to 0
0	1	Actual value is held in its last state
1	1	Actual value accumulates

① 0 = nonconducting; 1 = conducting.

7-5-3. SPECIFICATIONS

The following programming specifications apply to the timing function:

TIMER FUNCTION BLOCK

The timer function blocks require 2 horizontal contact spaces by 3 parallel paths on the ladder diagram. The timer functions can be located anywhere in the contact area of the ladder diagram. Note: Use of the One Shot Timer function requires the use of a history bit which is automatically assigned by the processor.

TIMING ENABLE/$\overline{\text{RESET}}$ CIRCUITS

The timing and $\overline{\text{reset}}$ circuits are installed on the ladder diagram to the left of the timing function. The contact area available for both circuits is 8 horizontal contacts located in up to 7 parallel paths. This assumes the timing function is placed on the far right portion of the ladder, even though the timing functions can be positioned anywhere in the contact area of the ladder diagram. The rules for programming contacts listed in Section 6 apply here, also. Timing diagrams for the timing enable and reset circuits are listed for each timer group as follows:

- Time On functions in Table TMR-2
- One Shot functions in Table TMR-3

PRESET VALUE

The Preset Value defines the time or number of ladder diagram scans defined by the timing function. The Preset Value is selected by the programmer as a constant in the range of from 1 to 32,767, or it may be specified as the

value contained in:

- A holding register (R)
- An output register (OR)
- An input register (IR)

The letters R, OR and IR precede the ladder diagram display of the Preset Value on the ladder diagram to indicate the type of Preset Value chosen. If a constant is selected, the constant is displayed on the ladder diagram with no letter preceding it.

ACTUAL VALUE

The Actual Value keeps track of the timing or scan counting taking place. The storage register for the Actual Value is specified by the programmer as either a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the ladder diagram display of the Actual Value Register. The contents of the Actual

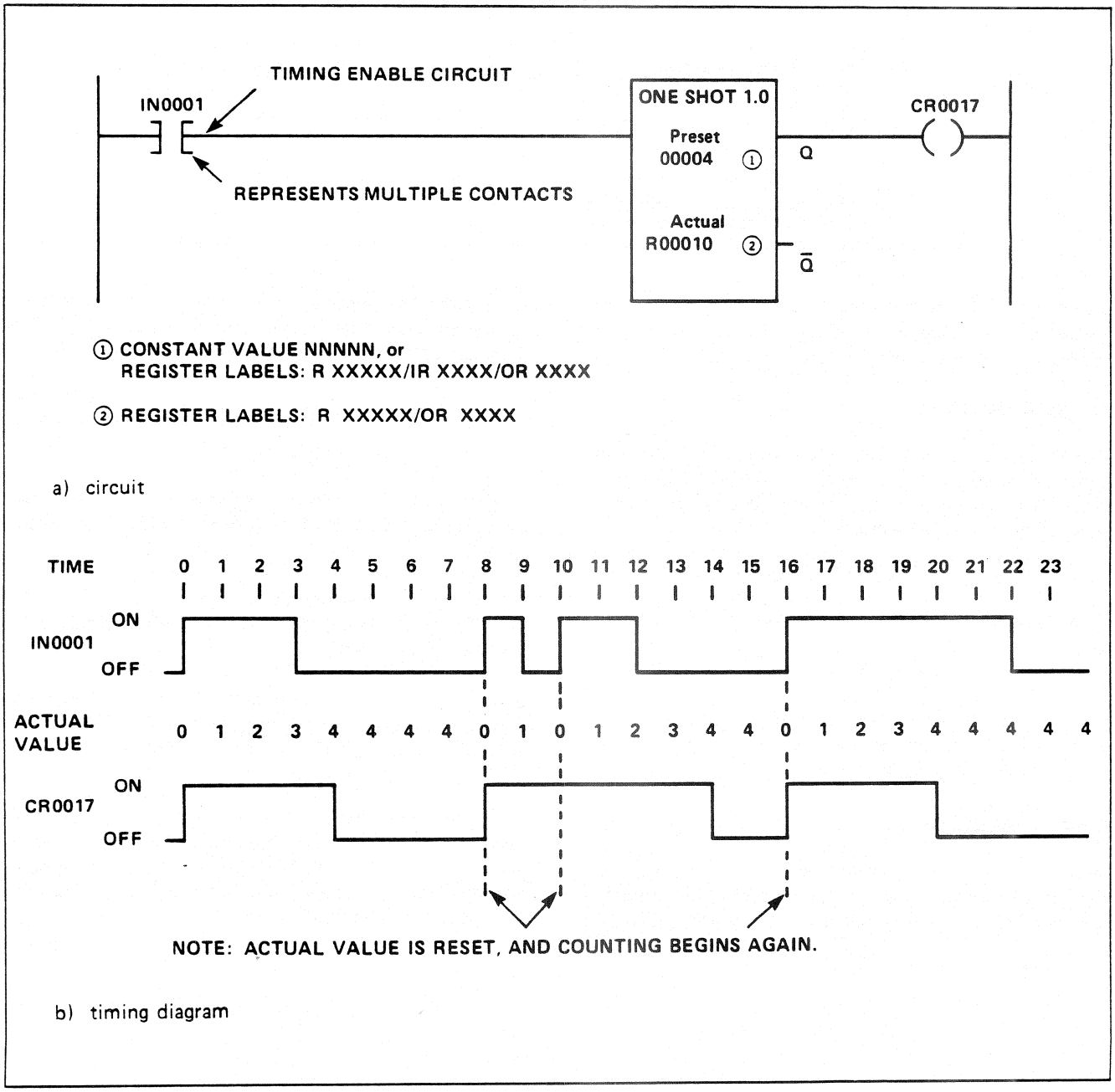


Figure TMR-3. One Shot Function Circuit (Typical)

TABLE TMR-3. ONE SHOT FUNCTION TRUTH TABLE

Enable Timing Circuit	Result
①	• The Q output conducts • Clears the Actual Value Register to 0 • Initiates the timing functions, incrementing the Actual Value • When the Actual Value equals the Preset Value, the Q output switches to a nonconducting state. ③
②	Has no effect on the One Shot function except to prepare for next transition.

① Indicates an off-to-on transition of the timing enable circuit conducting path.
② Indicates an on-to-off transition of the timing enable circuit conducting path.
③ When the Q output is nonconducting, if the Preset Value is increased, or if the Actual Value is decreased, the Q output conducts.

Value Register are retained when AC power to the processor is removed, or when the processor keyswitch is changed to various positions.

Q AND $\bar{Q}$ OUTPUTS

The Q and $\bar{Q}$ can be considered the outputs of the timer function block. These outputs are **always** opposite in state. See the appropriate timing diagrams for the conduction paths of the various timer functions:

- Time On function in Table TMR-2
- One Shot function in Table TMR-3

Note

The Q and/or $\bar{Q}$ outputs of any of the timer functions can be left unconnected in the network.

7-5-4. APPLICATIONS

Assume an application where 2 motors each control one-half of a conveyor. The motors are turned on and off as described here. The first half of the conveyor, called part A, receives the material. It is turned on for 45 seconds before the second half, called part B, is started. Part A is then turned off 1 minute before part B is turned off.

Figure TMR-4 shows how timer functions can be used to program the motor sequencing, described above. Observe the Figure and note the following 3 points:

1. Part A of the conveyor is controlled by discrete output CR0021. Part B is controlled by discrete output CR0022.

2. The Time On function with a preset of 45 delays CR0022 from being energized for 45 seconds after CR0021 is energized.

3. The Time On function with a preset of 100 holds CR0022 energized for 1 minute after CR0021 is de-energized, thus providing a time off function.

Note that this "artificial" time off is necessary only because a standard programmable function set is assumed here. If the HPPC-1500/-1700 system supports an advanced set, one of the Time Off functions (031 thru 033) could be used.

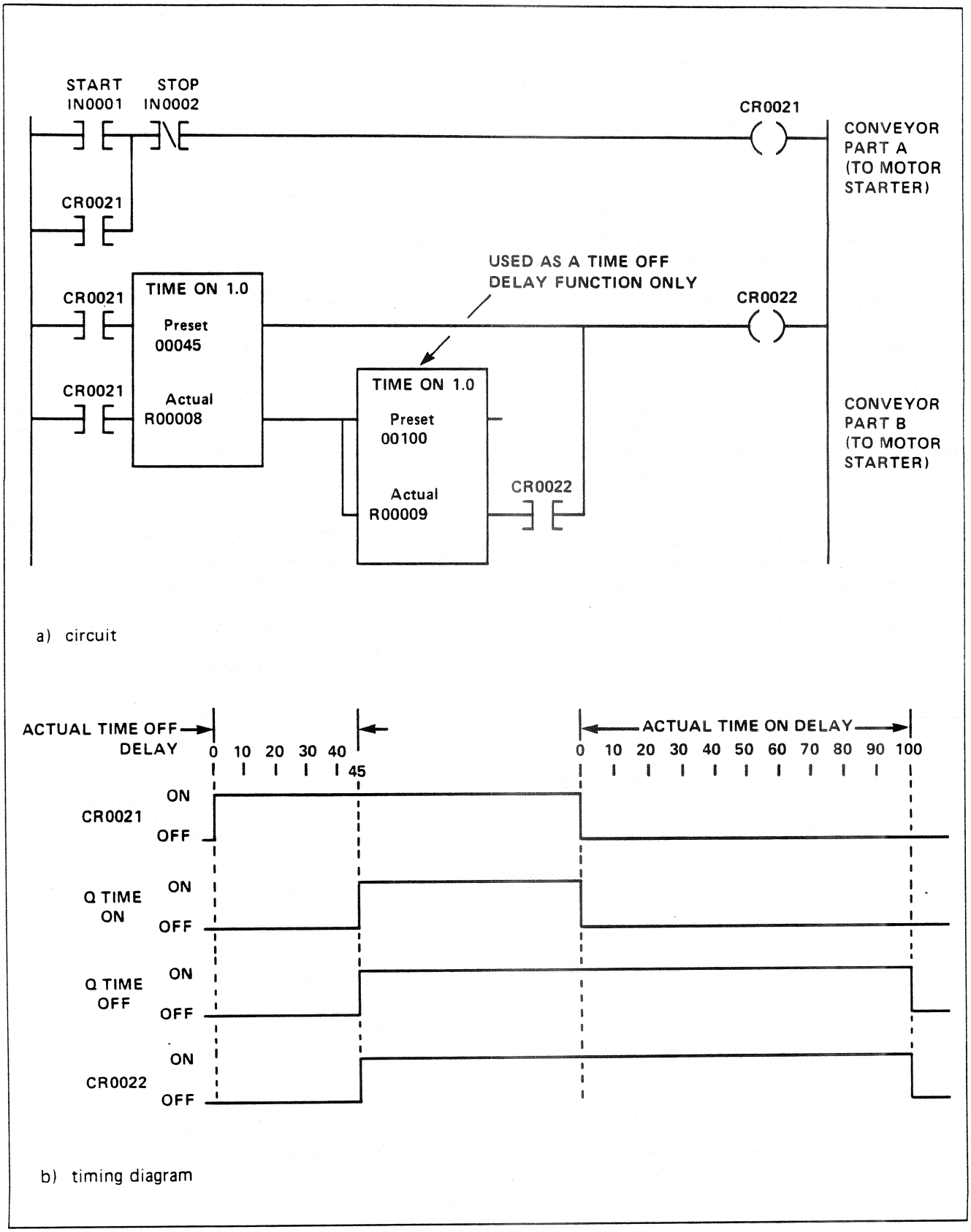


Figure TMR-4. Timer Application

7-6. UP COUNTER (038), DOWN COUNTER (039)

The Up Counter and Down Counter programmable functions keep track of individual increments or decrements. The Up Counter function is described in Paragraph 7-6-1; the Down Counter in Paragraph 7-6-2.

7-6-1. UP COUNTER FUNCTION

The Up Counter function counts up from zero to a user-selected number, which can be a maximum of 32,767. See Figure CNT-1 and note the following:

1. The Preset Value defines when the Q output conducts. When the Actual Value equals or exceeds the Preset Value, the counter's Q output conducts. The Preset Value can be entered as a constant or as a register.

2. The Actual Value Register stores the accumulated count of the counter. When the value of the Actual Register equals or exceeds the value of the Preset, the Q output of the Up Counter function conducts.

3. The count enable circuit conducts, thereby incrementing the value in the Actual Register by one for each scan of the ladder diagram. If a false-to-true transition is desired to enable the count (the more normal case), install a Transitional function (007) in the counting circuit as shown in Figure CNT-1.

4. The $\overline{\text{reset}}$ circuit clears the value in the Actual Register to 0 and holds it there as long as the circuit is nonconducting.

5. The Q and $\overline{Q}$ connections of the Up Counter function are considered the outputs of the counter. They are always in opposite states. When the value of the Actual Register equals the Preset Value, the Q output of the counter conducts. Table CNT-1 contains a truth table for the counting and reset circuits. Either or both of these outputs can be used to provide a conduction path to other functions or coils.

7-6-2. DOWN COUNTER FUNCTION

The Down Counter function counts down to zero from a user-selected Preset Value as high as 32,767. See Figure CNT-2 and note that the characteristics of the count enable circuit, Actual Register, and Preset Register are identical to those of the Up Counter function described in Paragraph 7-6-1. Other characteristics of the Down Counter function are:

1. The Preset Value is loaded into the Actual Register as long as the $\overline{\text{preset}}$ circuit is nonconducting. When the $\overline{\text{preset}}$ circuit conducts, the count enable circuit controls the counter, as listed in Table CNT-2.

2. The Q and $\overline{Q}$ connections of the Down Counter function are considered the outputs of the counter. They are always in opposite states. When the value of the Actual

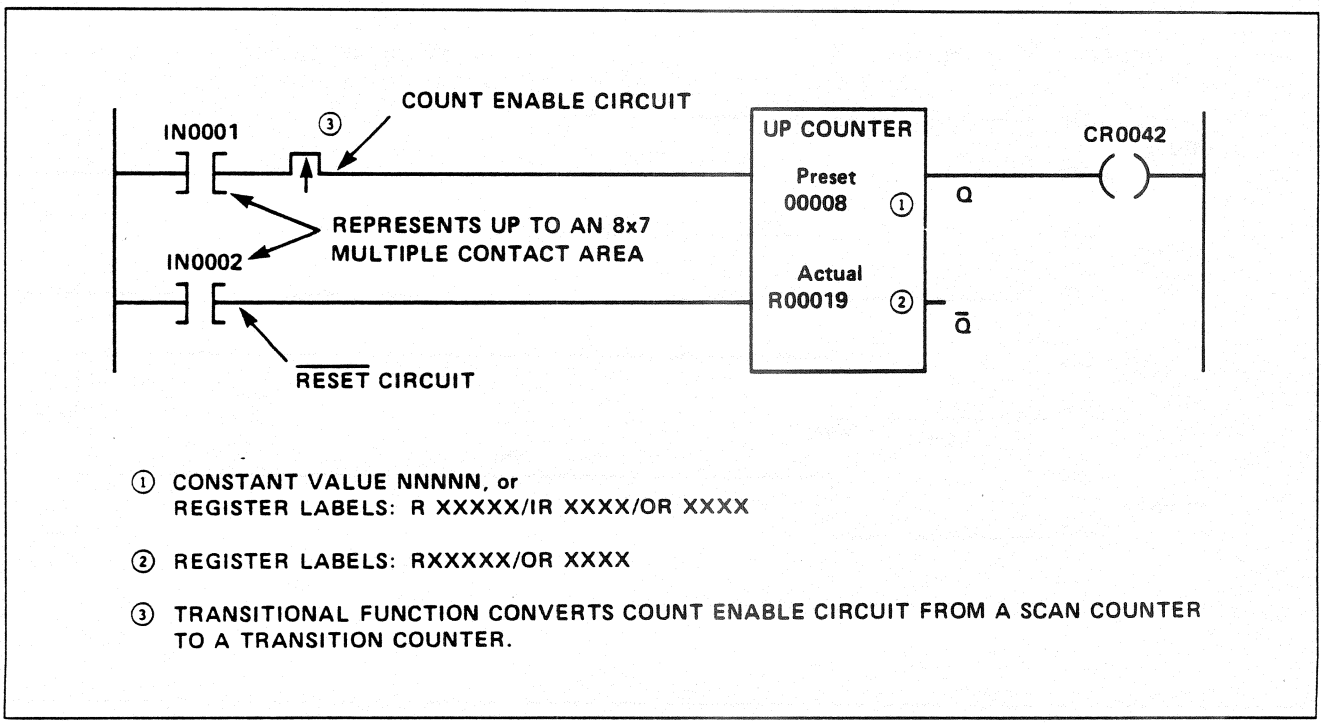


Figure CNT-1. Up Counter Circuit (Typical)

TABLE CNT-1. UP COUNTER TRUTH TABLE

Count Enable Circuit ①	$\overline{\text{Reset}}$ Circuit ①	Result
0	0	• Actual Value held at zero.
1	0	• Q output is nonconducting
0	1	Actual Value is held in its last state. If Actual equals or exceeds Preset Value, the Q output conducts.
1	1	Actual Value increments by 1 **each scan** of the ladder diagram. When the Actual equals or exceeds Preset Value, the Q output conducts.
① 0 = nonconducting; 1 = conducting.		

Register is equal to or less than zero, the Q output conducts. Either or both of these outputs can be used to provide a conduction path to other functions or coils.

7-6-3. SPECIFICATIONS

The following programming specifications apply to the counting functions:

COUNTER FUNCTION BLOCK

The Counter function blocks require 2 horizontal contact spaces by 3 parallel rows on the ladder diagram. The Counter functions can be located anywhere in the contact area of the ladder diagram.

COUNT ENABLE CIRCUIT

When the count enable circuit conducts, a count will be:

- Added to the Actual Value (Up Counter function)
- Subtracted from the Actual Value (Down Counter function)

The incrementing (adding) or decrementing (subtracting)

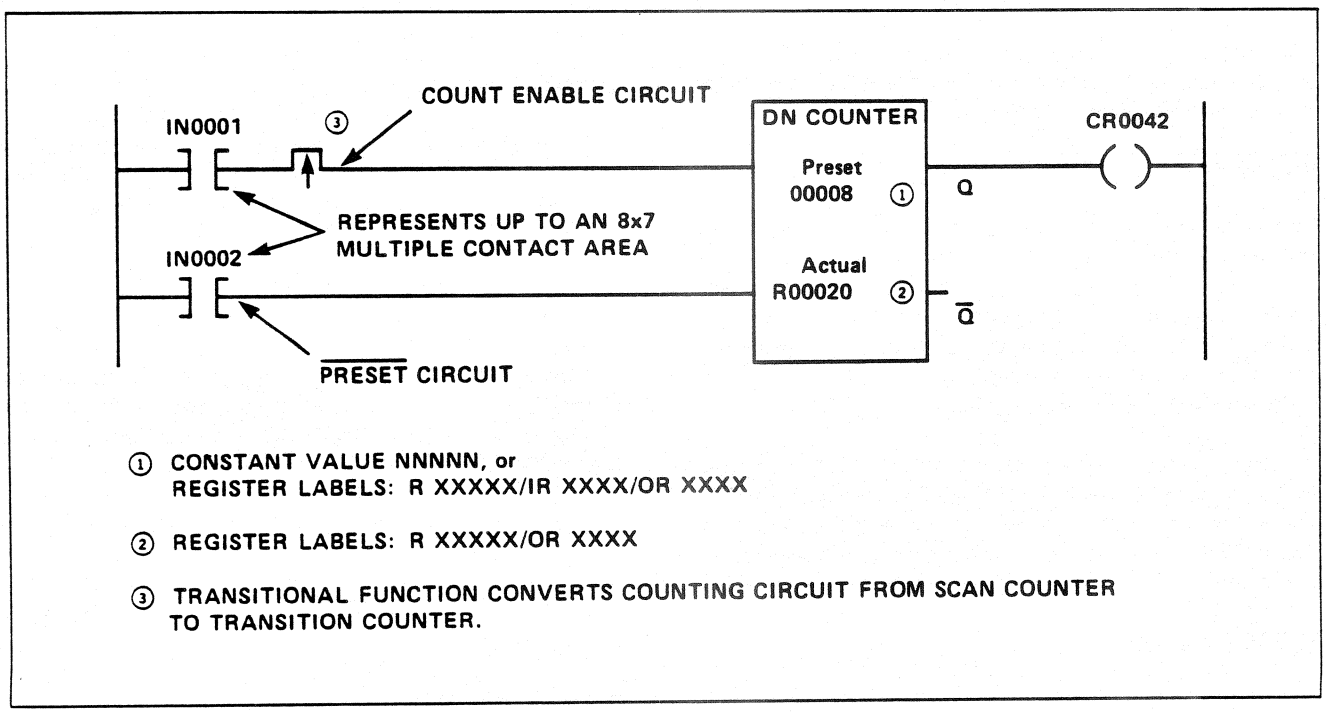


Figure CNT-2. Down Counter Circuit (Typical)

TABLE CNT-2. DOWN COUNTER TRUTH TABLE

Count Enable Circuit ①	$\overline{\text{Preset}}$ Circuit ①	Result
0	0	• Preset Value loaded into the Actual Value register. • The Q output does not conduct.
1	0	
0	1	Actual Value is held in its last state. If Actual is equal to or less than zero, the Q output conducts.
1	1	Actual Value is decremented by 1 count each scan of the ladder diagram. When the Actual Value is equal to or less than 0, the Q output conducts.
① 0 = nonconducting; 1 = conducting.		

of the Actual Value occurs for each scan of the ladder diagram. If the Transitional element is contained in series with the count enable circuit, a count occurs once for each nonconducting-to-conducting transition of the count enable circuit.

The count enable circuit, together with the $\overline{\text{reset}}$ (Up Counter) or $\overline{\text{preset}}$ (Down Counter) circuit, is located on the ladder diagram to the left of the timing function. The contact area available for both circuits consists of up to 8 horizontal contact spaces located in up to 7 parallel paths. This assumes the counting function is positioned on the far right portion of the ladder. The rules for programming contacts listed in Section 6 apply to the placement of contacts connected to the count enable and $\overline{\text{reset}}$/$\overline{\text{preset}}$ circuits.

$\overline{\text{RESET}}$/$\overline{\text{PRESET}}$ CIRCUIT

The $\overline{\text{reset}}$ and $\overline{\text{preset}}$ circuits function as listed in the following 2 points:

1. The $\overline{\text{reset}}$ circuit, associated with the Up Counter function, clears the Actual Value Register to zero when the $\overline{\text{reset}}$ circuit is nonconducting. Table CNT-1 describes how the $\overline{\text{reset}}$ circuit affects the Q output.

2. The $\overline{\text{preset}}$ circuit, associated with the Down Counter function, presets the Actual Value Register with the Preset Value when the $\overline{\text{preset}}$ circuit is nonconducting. Table CNT-2 describes how the $\overline{\text{preset}}$ circuit affects the Q output.

PRESET VALUE

The Preset Values define the number of counts which occurs before the Q output conducts. The Preset Value is a positive binary value ranging from decimal 1 to 32,767. The Preset is assigned as a constant value by the programmer, or it is specified as the value contained in:

- Holding register (R)
- Input register (IR)
- Output register (OR)
- Constant value from 1 to 32,767

The letters R, IR, or OR precede the function block display of the Preset Value. If a constant is selected, no letter precedes the value on the display.

ACTUAL VALUE

The Actual Value Register stores the current number of the accumulated counts. The storage register for the Actual Value is specified by the programmer as a:

- Holding register (R), or
- Output register (OR)

The letters R or OR precede the function block display of the Actual Value Register. It can range from +32,767 to -32,768 in binary. Overflows or underflows will not occur; rather the:

- Up Counter function stops incrementing at +32,767
- Down Counter function stops decrementing at -32,768

Q AND $\overline{\text{Q}}$ OUTPUTS

The Q and $\overline{\text{Q}}$ can be considered the outputs of the

counter. These outputs are always in opposite states. They conduct to energize coils or enable other functions. Refer to the following timing charts for a description of the Q and $\overline{Q}$ operation:

- Up Counter: see Table CNT-1
- Down Counter: see Table CNT-2

Note

The Q and $\overline{Q}$ outputs of the Counter function can be left unconnected in the network.

7-6-4. APPLICATIONS

The Up Counter and Down Counter functions can be combined to form an up/down counter. The counters are linked together simply by programming the same Actual Register label in both counters. Figure CNT-3 shows a ladder diagram of a typical up/down counter used to keep track of products entering and leaving a production line. A mechanical diagram of the line is shown in Figure CNT-4.

When IN0001 is closed, R00002 increments by one and continues to increment every time IN0001 is opened and closed. If IN0003 is opened and closed, the contents of R00002 are decremented by one and continue to decrement each time IN0003 is closed until 0000 is reached. When the RUN/RESET switch, IN0002, is closed, Actual Value Register R00002 is cleared to 0. When the switch is open, the counters are functional.

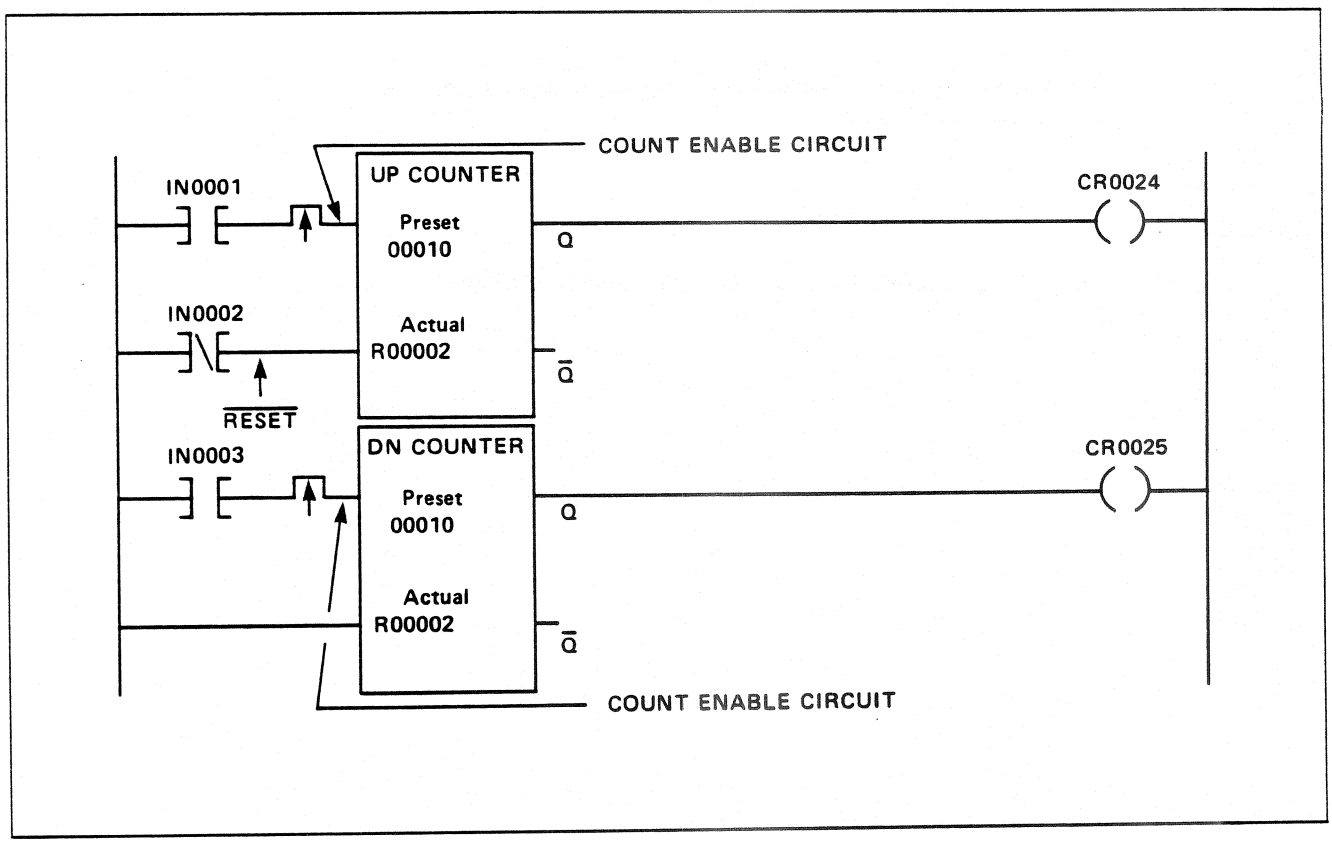


Figure CNT-3. Up/Down Counter Circuit (Typical)

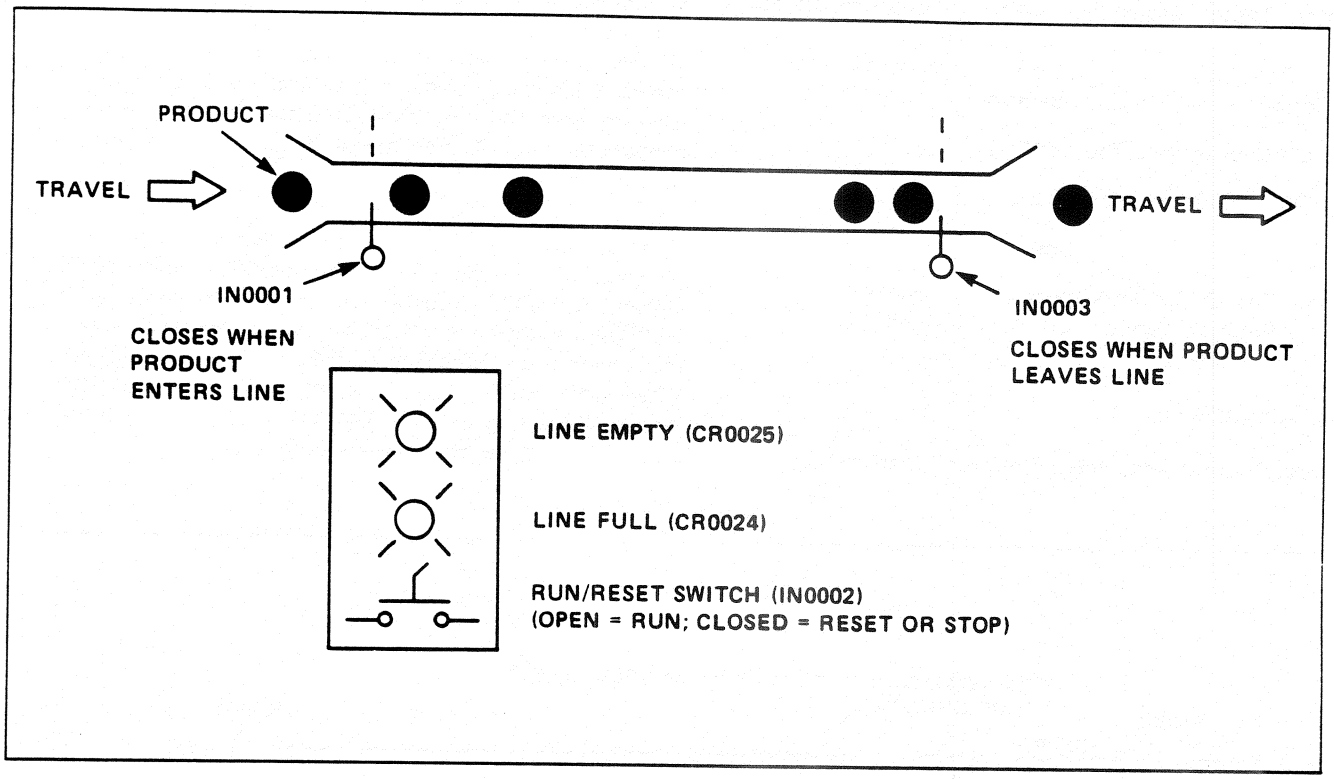


Figure CNT-4. Up/Down Counter Application Example

7-7. ADD (068), SUBTRACT (070)

The Add and Subtract programmable functions are used to add and subtract binary numbers up to 16 digits in length (from -32,768 to +32,767, decimal). The numbers can be positive or negative. Figure A/S-1 shows a typical ladder diagram containing the Add and Subtract functions. When the add enable circuit conducts, Operands A and B are added or subtracted. The result is stored in the Result C Register. When the add enable circuit is conducting, the Add's output conducts, and vice versa. The overflow output conducts whenever:

- The enable circuit conducts, and
- The result of the addition or subtraction is greater than 32,767 or less than -32,768.

Table A/S-1 contains a truth table for Add and Subtract functions.

7-7-1. SPECIFICATIONS

The following programming specifications apply to the Add and Subtract functions:

ADD/SUBTRACT FUNCTION BLOCK

The Add and Subtract function blocks each requires 2 horizontal contact spaces by 4 parallel rows on the ladder diagram. The Add and Subtract functions can be located anywhere in the contact area of the ladder diagram.

ADD ENABLE CIRCUIT

When the enable circuit conducts, the addition or subtraction operation occurs during each scan of the ladder diagram. The enable circuit is located on the ladder diagram to the left of the Add or Subtract function. The contact area available is 8 horizontal contact spaces located in up to 7 parallel paths. This assumes the function is positioned on the far right portion of the ladder. The rules for programming contacts listed in Section 6 apply to the contacts of the enable circuit.

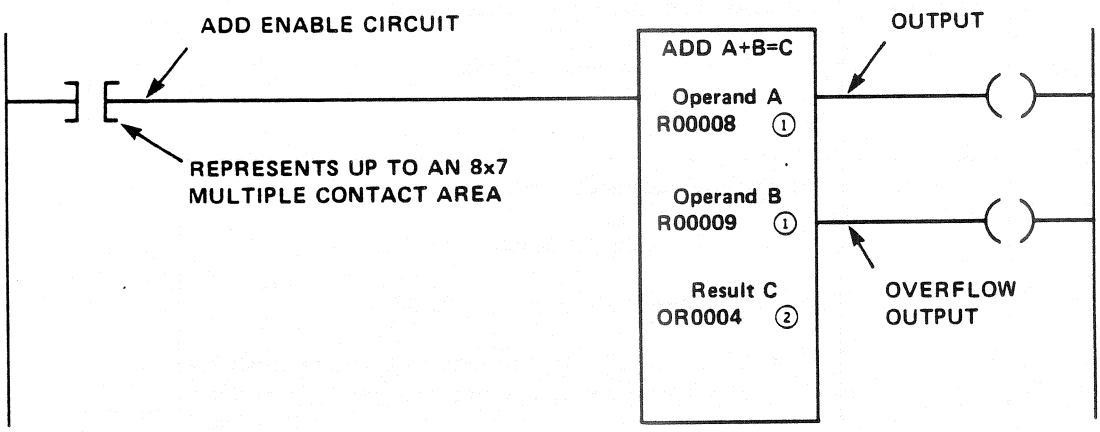
OPERAND A

The Operand A is either:

- One of 2 addends used in the Add function
- The minuend in the Subtract function

The operand value consists of a binary number (ranging from +32,767 to -32,768, decimal). The operand is assigned a constant value by the programmer, or it is specified as the value contained in any of the following:

- Holding register (R)



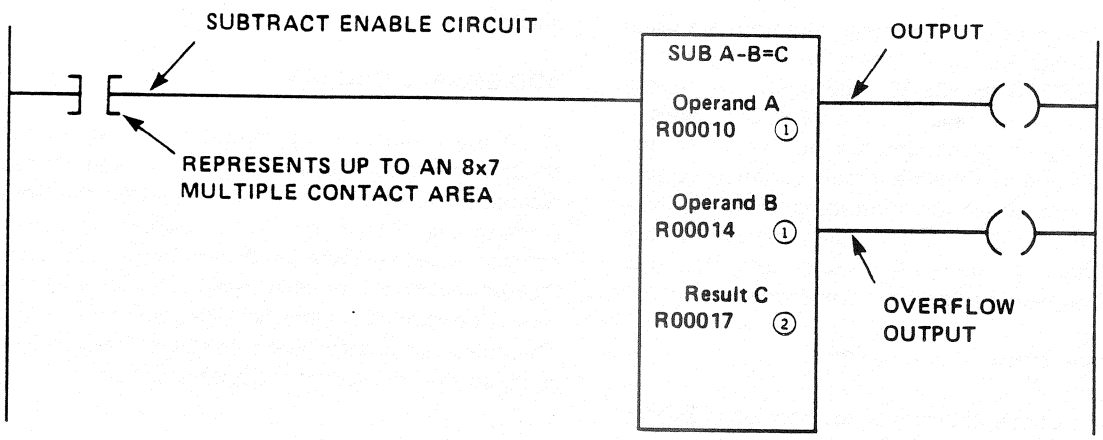
① CONSTANT VALUE NNNNN, or
REGISTER LABELS: R XXXXX/IR XXXX/OR XXXX

② REGISTER LABELS: R XXXXX/OR XXXX

a) Add function

(Cont'd.)

Figure A/S-1. Add and Subtract Circuits (Typical)



① CONSTANT VALUE NNNNN, or
REGISTER LABELS: R XXXXX/IR XXXX/OR XXXX

② REGISTER LABELS: R XXXXX/OR XXXX

b) Subtract function

Figure A/S-1. (Cont'd.)

TABLE A/S-1. ADD/SUBSTRACT TRUTH TABLE

Enable Input ①	Result
0	No addition or subtraction occurs and: • All outputs are nonconducting • Register contents remain unchanged

1	Addition or subtraction takes place and:
	• The add or subtract output is conducting
	• When the result is larger than 32,767:
	  • The value 32,768 is subtracted from the number, and the difference placed in the Result C register.
	  • The overflow output is conductive
	• When the result is smaller than -32,768:
	  • The value 32,768 is added to the number, and the sum placed in the Result C register.
	  • The overflow output is conductive

① 0 = nonconducting; 1 = conducting.

- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the ladder diagram display of the Operand Register. When a constant is selected, no letter precedes the display of the value.

OPERAND B

The Operand B is either:

- One of 2 addends used in the Add function
- The subtrahend used in the Subtract function

Operand B consists of a binary number (ranging from +32,767 to -32,768, decimal), assigned identically to Operand A. (See Operand A, above.)

RESULT C

The Result C value is the result of the addition or subtraction. The programmer assigns one of the following register types to contain the Result C:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the ladder diagram display of the Result C Register.

If an overflow condition occurs, the following operations take place automatically:

- Result = 32,768 or larger. The value 32,768 is subtracted from the number, and the difference placed in Result C. The overflow output conducts as long as the addition or subtraction is enabled.
- Result = -32,769 or smaller. The value 32,768 is added to the number, and the sum placed in Result C. The overflow output conducts as long as the addition or subtraction is enabled.

ADD/SUBTRACT OUTPUT

The add or subtract output simply repeats or follows the conducting or nonconducting state of the enable circuit.

OVERFLOW OUTPUT

The overflow output conducts whenever:

- The enable circuit conducts, and
- The result of the addition or subtraction is above 32,767 or below -32,768

Note

The outputs of the Add/Subtract function can be left unconnected in the network.

7-7-2. APPLICATIONS

When establishing deviation alarms for a process, it may be advantageous to allow different alarm ranges around the set point rather than the same deadband on each side of the set point. The Add and Subtract functions can establish the range of operation around the set point.

Figure A/S-2 shows a circuit for one such method where the set point is established by a thumbwheel switch. The upper and lower limits are independently established. If the upper and lower limits were always equal and opposite, a single register could be used as Operand B for both the Add and Subtract functions to save a word of memory.

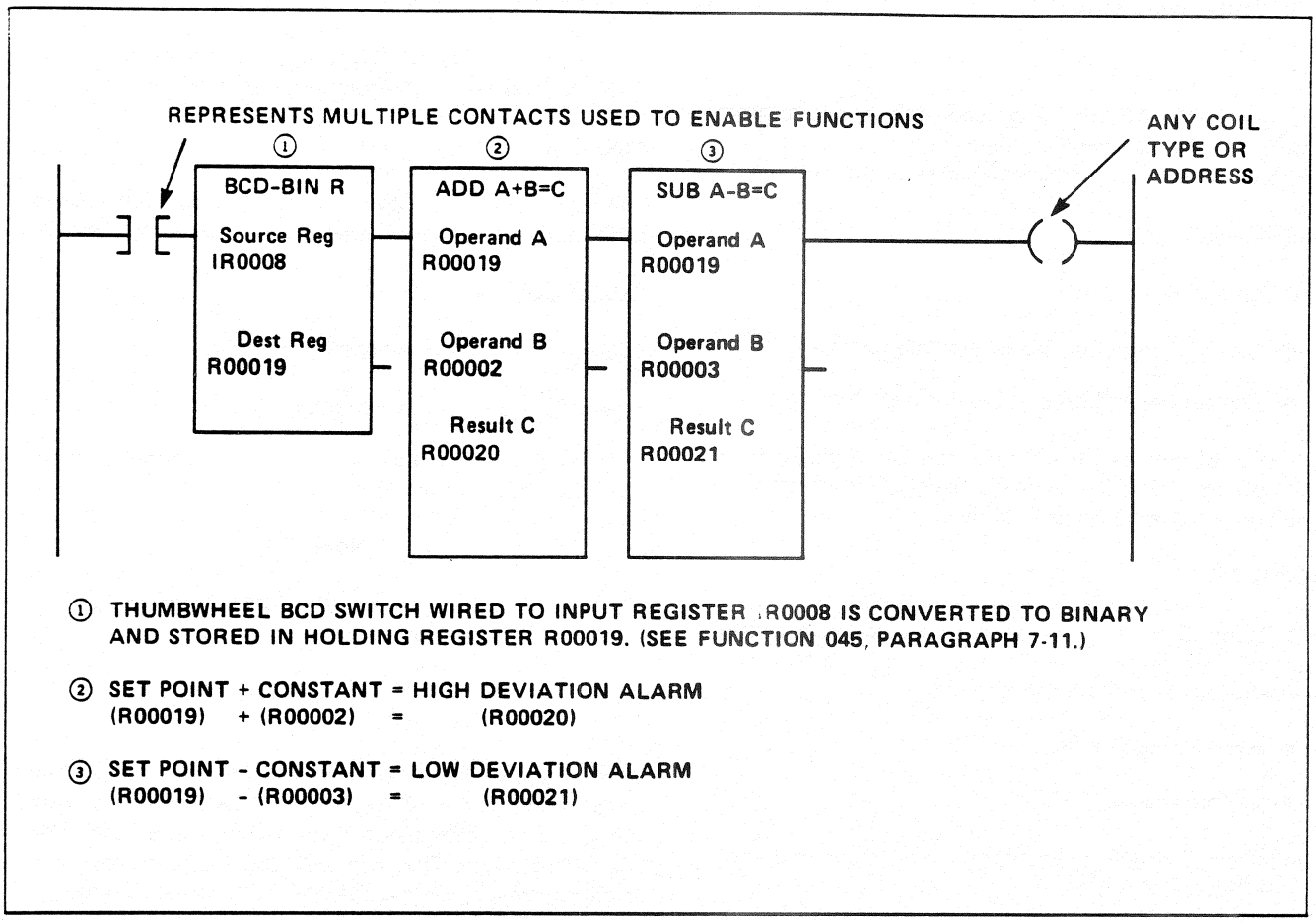


Figure A/S-2. Establishing Alarm Ranges

7-8. MULTIPLY (072)

The Multiply programmable function allows the multiplication of 2 binary numbers (Operand A and B). The product of the multiplication is contained in a double-precision register, that is, 2 consecutive registers referred to as Result C. Figure MULT-1 shows a typical use of the Multiply function. Observe the Figure and note the following:

1. The multiply enable circuit initiates the multiplication of the binary values contained in Operands A and B. When the multiply enable circuit conducts, the multiplication takes place, and the product is stored in the Result C Register.

2. The output of the Multiply function block simply repeats, or follows, the conducting or nonconducting state of the multiply enable circuit.

3. Operands A and B are the multiplicand and the multiplier, respectively. The binary values of the operands range from +32,767 to -32,768 decimal.

4. The Result C Registers contain the double-precision binary product of the multiplication. The registers consist of a consecutively numbered pair, where the higher register number contains the least significant digits. These registers are assigned by the programmer as a holding register or output register.

7-8-1. SPECIFICATIONS

The following programming specifications apply to the Multiply function.

MULTIPLY FUNCTION BLOCK

The Multiply function block requires 2 horizontal by 4 parallel contact paths of the ladder diagram. The Multiply function block can be positioned anywhere in the 10 by 7 contact area of the ladder diagram.

OPERANDS A AND B

The binary values of Operand A, the multiplicand, and Operand B, the multiplier, must be in the range of +32,767 to -32,768 decimal. Negative values must be in 2's complement form. The rules for multiplying algebraically apply as shown in Table MULT-1. If the Result C is negative, the binary number will be in 2's complement form.

The Operand A and B values are specified by the programmer as a constant, or as the value contained in one of the following specified registers:

- Holding register (R)
- Input register (IR)
- Output register (OR)

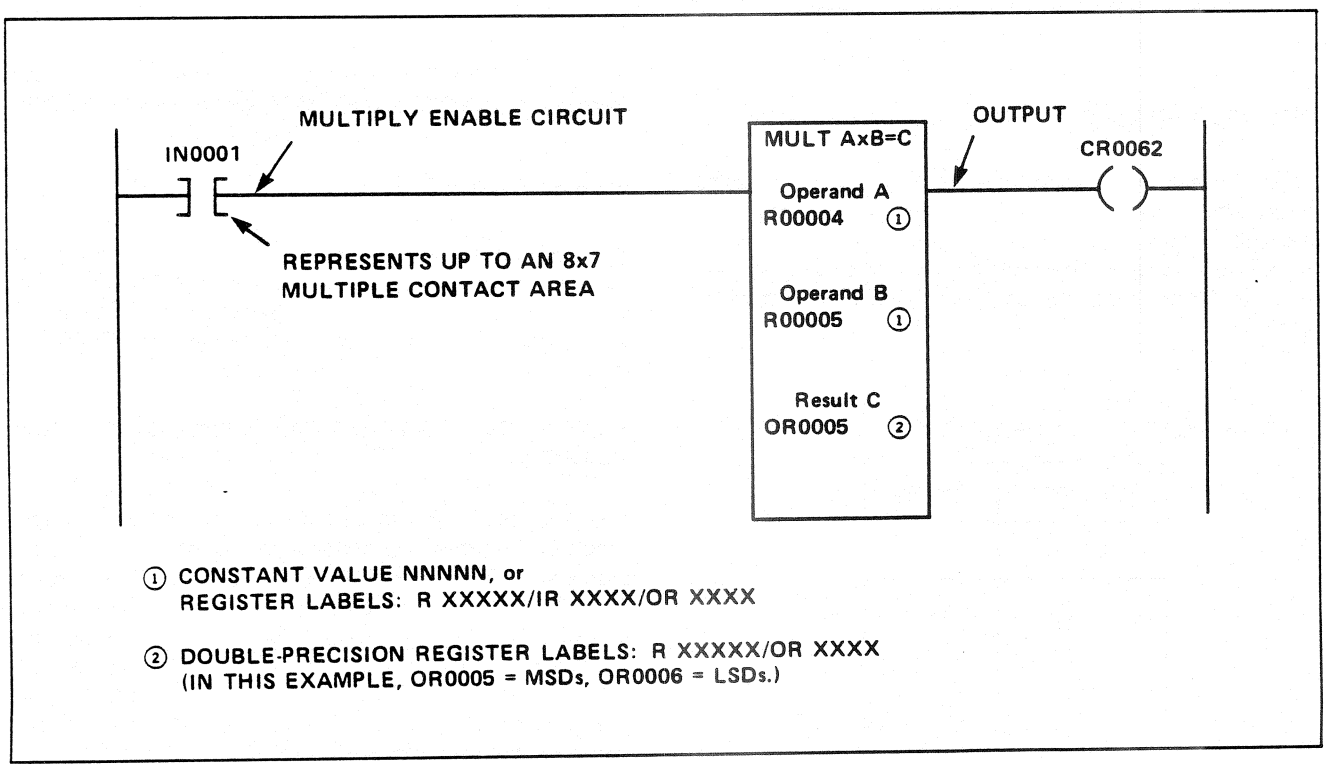


Figure MULT-1. Multiply Circuit (Typical)

The letters R, IR or OR precede the register number beneath the Operands as shown in Figure MULT-1. If a constant value is selected, the actual constant is displayed without any letters preceding it.

RESULT C

The Result C Registers contain the double-precision product of the Multiply function. The binary value is assigned to 2 consecutively numbered registers. The registers are specified as either a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the display. The register number actually programmed and displayed contains the most significant digits.

The value contained in the Result C Registers is positive or negative, depending on the signs of the Operands, as listed in Table MULT-1.

Note

Care must be exercised when using the double-precision result. It must be converted to a single-precision register for use where a single-precision register is required. See Paragraph 7-9-2-2.

MULTIPLY ENABLE CIRCUIT

When the enable circuit conducts, multiplication takes place. When the enable circuit is nonconducting, multiplication ceases, and the data in the Result C Registers remains unchanged. The multiply enable circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths.

TABLE MULT-1. ALGEBRAIC MULTIPLICATION

Operands:		Result:
A	B	C
+	+	+
-	+	-
+	-	-
-	-	+

OUTPUT

The output of the Multiply function repeats, or follows, the conducting or nonconducting state of the enable circuit.

Note

The output of the Multiply function can be left unconnected in the network.

7-8-2. APPLICATIONS

The following applications apply to the Multiply function:

- Times 10 application (7-8-2-1)
- Squaring (7-8-2-2)
- Single-to-double precision conversion (7-8-2-3)

7-8-2-1. TIMES 10

A times 10 application is shown in Figure MULT-2 where:

- Operand 1 is the value to be multiplied
- Operand 2 contains the multiplier constant 10

When IN0001 is closed, the value contained in holding register R00011 is multiplied by 10. The result is placed in registers R00020 and R00031, where register R00021 contains the least significant digits.

7-8-2-2. SQUARING

An application which uses the Multiply function to square binary values is shown in Figure MULT-3. The same register is used for both Operand A and Operand B to multiply the contents of holding register R00005 by itself.

7-8-2-3. SINGLE-TO-DOUBLE PRECISION CONVERSION

The Multiply function can be used to convert a single-precision number represented in 16 bits to a double-precision number represented by 32 bits. The signs of the single- and double-precision numbers are always identified by their most significant bits. To perform the conversion, simply program the single-precision value as one of the Operands of the Multiply function. Program the second Operand with a constant of 1. The Result C contains the double-precision result.

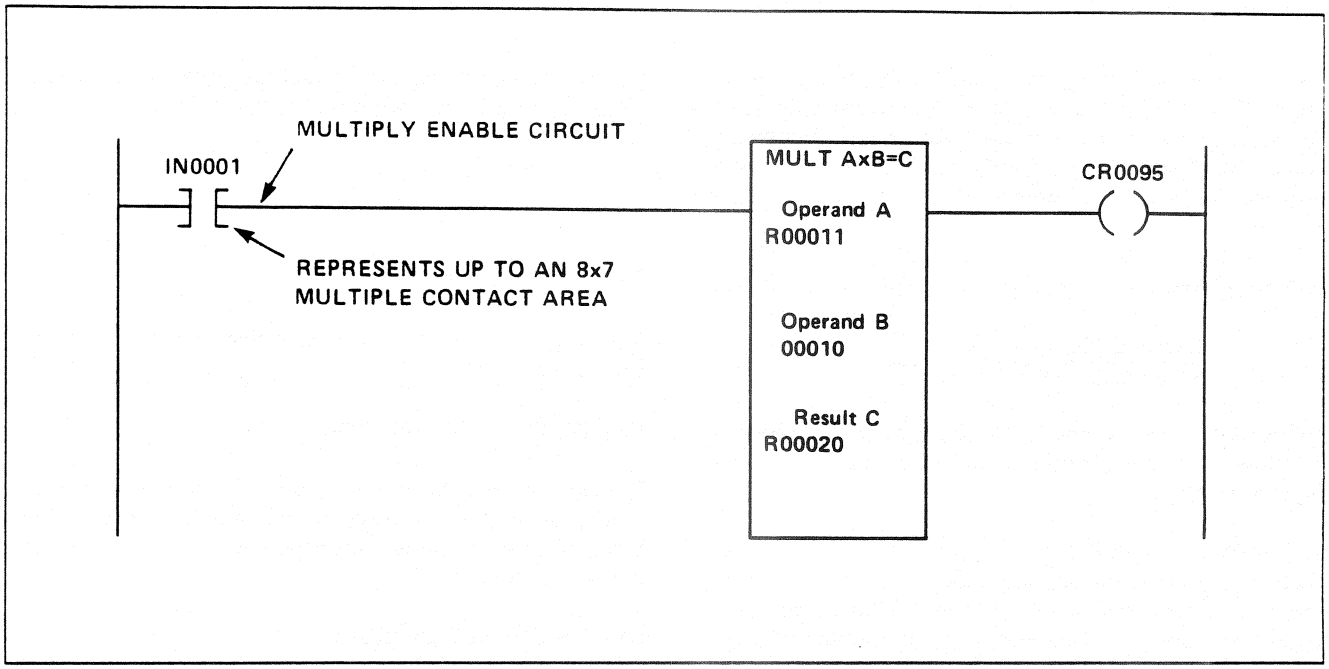


Figure MULT-2. Times 10 Application

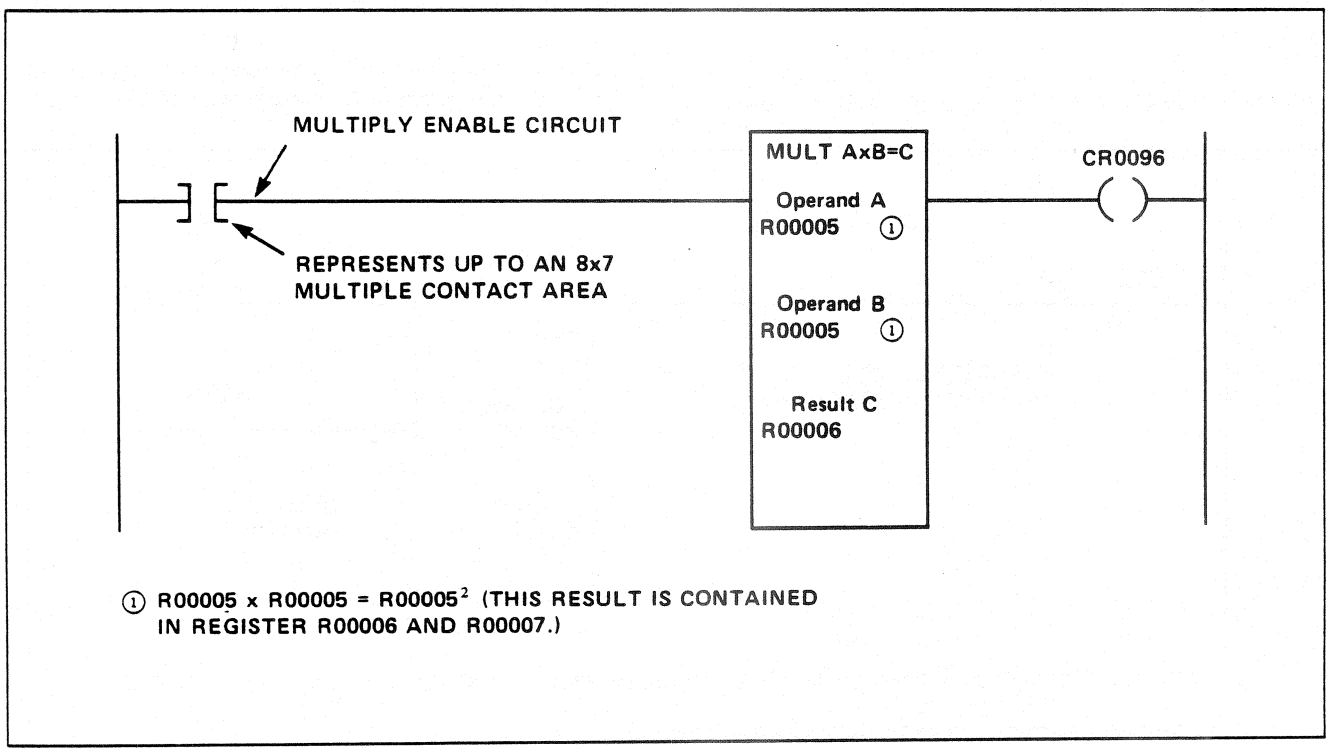


Figure MULT-3. Number Squaring

7-9. DIVIDE (073)

The Divide programmable function allows the division of 2 binary numbers (Operands A and B) and the storage of the result in a Result C Register. Figure DIV-1 shows a typical use of the Divide function. Observe the Figure and note the following 4 points:

1. The divide enable circuit initiates the division of the dividend (Operand A) by the divisor (Operand B) where:

$$\frac{\text{Dividend (Operand A)}}{\text{Divisor (Operand B)}} = \text{Result C}$$

The output of the Divide function block simply repeats, or follows the conducting or nonconducting state of the divide enable circuit.

2. Operands A and B are the dividend and divisor, respectively, as shown above. Operand B is a single-precision binary value which can be specified as a register or constant. Operand A is a double-precision binary value contained in a pair of consecutively numbered registers.

3. The Result C Register contains the binary result of the division in a single-precision register with the remainder contained in the next consecutively numbered register.

Either a holding register (R) type or output register (OR) type can be selected.

4. The overflow output conducts if the divide enable circuit conducts, and the result of the division is:

- Greater than 32,767, or
- Less than -32,767

7-9-1. SPECIFICATIONS

The following programming specifications apply to the Divide function:

DIVIDE FUNCTION BLOCK

The Divide function block requires 2 horizontal contact spaces by 4 parallel rows on the ladder diagram. The block can be positioned anywhere in the contact area of the ladder diagram.

DIVIDE ENABLE CIRCUIT

When the divide enable circuit conducts, the division process takes place. When the enable circuit is nonconducting, division ceases, and the result of the last division is held in the Result C Register. The divide enable circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths positioned to the left of the Division function block. The rules for programming listed in Section 6 apply to the contacts of this circuit.

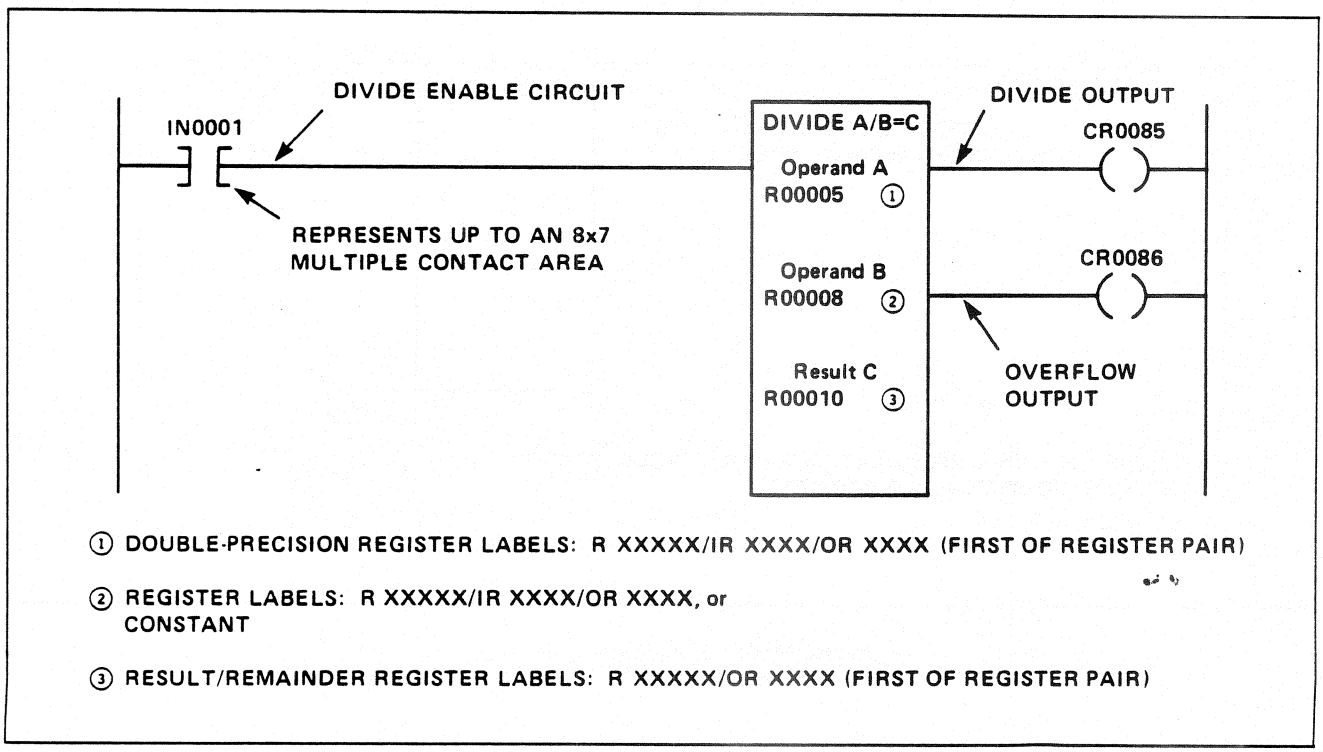


Figure DIV-1. Divide Circuit (Typical)

DIVIDE OUTPUT

The output of the Division function block simply repeats, or follows, the conducting or nonconducting state of the enable circuit.

Note

The divide and overflow outputs from the divide function can be left unconnected in the network.

OPERAND A

The Operand A Registers are the double-precision dividend of the Division function. The registers occupy 2 consecutively numbered register locations, where the register number programmed and displayed in the block contains the most significant digits. Operand A is specified by the programmer as a pair of:

- Holding registers (R)
- Input registers (IR)
- Output registers (OR)

The letters R, IR or OR precede the register number on the ladder diagram. The binary value of the registers can range from -2,147,483,648 to +2,147,483,647. Negative numbers must be in 2's complement form.

OPERAND B

Operand B is the divisor. It is specified by the programmer as a constant, ranging from -32,768 to +32,767, or as the value contained in one of the following:

- Holding register (R)
- Input register (IR)
- Output register (OR)

The letters R, IR or OR precede the register number on the ladder diagram. If a constant was selected, the value is displayed without any letter preceding it. The value of Operand B can range from +32,767 to -32,768. Negative numbers must be in 2's complement form.

RESULT C

The Result C Registers store the result of the division function. The registers consist of a single-precision register containing the result and the next consecutively numbered register containing the remainder.

The registers are specified as one of the following types:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the display.

The rules for dividing algebraically apply as shown in Table DIV-1.

TABLE DIV-1. ALGEBRAIC DIVISION

Operands:		Result:
A	B	C
+	+	+
-	+	-
+	-	-
-	-	+

OVERFLOW OUTPUT

The overflow output conducts when the enable input is conducting and the result of the division is either:

- Above 32,767
- Below -32,767

Whenever the overflow output is conducting, the value of the Result C Register remains unchanged. Also, if a division by zero is attempted:

- The content of the Result C Register remains in its last state, and
- The overflow output conducts

7-9-2. APPLICATION

The Division function can be applied to the following:

- Use with other math functions (7-9-2-1)
- Double-to-single precision conversions (7-9-2-2)

7-9-2-1. COMBINED MATH FUNCTIONS

The Division function can be combined in a series string with other math functions, allowing efficient mathematical computations to be performed within the ladder diagram. One example of computations is the conversion

of centigrade temperatures to Fahrenheit according to the following formula:

$$F^{\circ} = (C^{\circ} \times 9 \div 5) + 32$$

Figure DIV-2 performs the math function. Observe the Figure and note the following 5 points:

1. The binary data representing degrees centigrade is available at input register IR0011.

2. The Multiply function multiplies the centigrade data times 9 and stores the result in R00020.

3. The Divide function divides the product of the multiplication function by 5.

4. The Add function adds the constant 32 to the result of the Divide function to complete the conversion. The converted number is contained in holding register R00025.

5. Of course contact IN0001 or any series contacts need not be programmed in the series of math functions.

7-9-2-2. DOUBLE-TO-SINGLE PRECISION CONVERSION

The Division function can be used to convert a double-precision binary number (represented by 32 bits) into a single-precision binary number (represented by 16 bits). To perform this application, simply program Operand A with the double-precision value to be converted. Then load a constant of 1 in Operand B. The Result C register assigned contains the single-precision number. The overflow output of the Division function can be used to indicate that the value being converted is too large—or small if negative—to fit in a single-precision register.

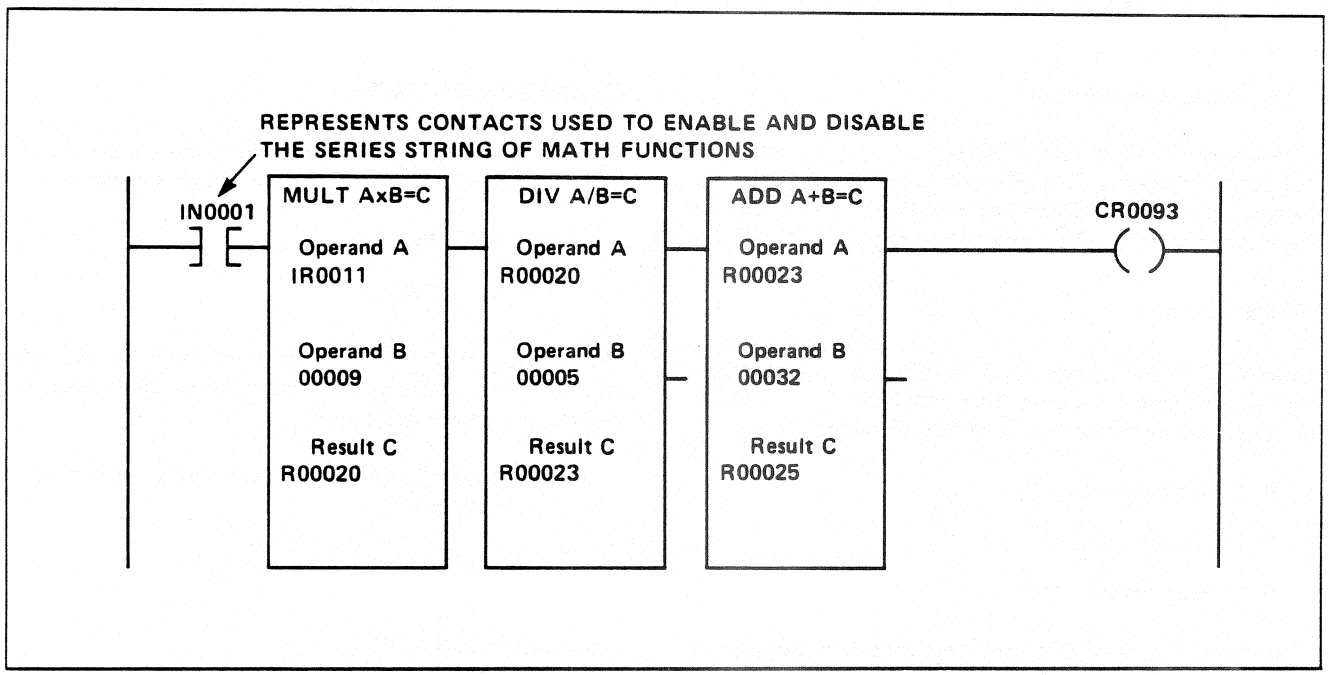


Figure DIV-2. Temperature Conversion Example

7-10. SHIFT REGISTERS (040, 041)

The Shift Left and Shift Right programmable functions shift the binary data contained in one or more selected registers one place to the left or right, respectively. "Left" is defined as moving data in the form of individual bits from the LSD position in a register toward the MSD position. "Right" is defined as moving data from the MSD position in a register toward the LSD position.

Observe the typical Shift Right function shown in Figure SR-1 and SR-2. Note the following 5 points:

1. The reference number of the beginning register in the "string of registers" appears in the function block. Referred to as the Start Register, it can be specified as either a holding or output register.

2. The number of consecutively numbered registers in the string (that is, the length) is a constant. The Length of the shift register shown in Figure SR-1 is 2. (Holding registers R00005 and R00006 have been assigned in the Figure.)

3. When the shift enable circuit conducts, the data contained in the string of registers selected is shifted one place to the right, as shown in Figure SR-2. The data is shifted one place, or "bit position," for each scan of the ladder diagram program. The Transitional element in the shift enable circuit of Figure SR-1 causes the shift register to be shifted only once—when the enable circuit first conducts, **not** during each scan.

4. The serial input circuit provides the input to bit 16 of the second consecutively numbered register, which is holding register R00006 in the example of Figure SR-1. The input is 1 if the serial input circuit is conducting, or 0 if the serial input circuit is nonconducting.

The input state (0 or 1) of the serial input circuit is input to the shift register when the shift enable circuit is conducting. Figure SR-1 shows a Transitional element used in the shift enable circuit which allows the register to shift 1 place to the right once when the shift enable circuit first conducts, not during each scan of the ladder diagram.

5. The serial output simply monitors the "output" of the register from bit 1 of register R00005 in Figure SR-2. The serial output is conducting and nonconducting, depending on the state of bit 1 when shifted out of the lowest register in the string, where 1 equals conducting, and 0 equals nonconducting. When the next new data is shifted from bit 1 of the lowest numbered register, the output data is lost.

A typical Shift Left function is shown in Figure SL-1 and SL-2. Observe the Figure and note the following:

1. The data shifts in the opposite direction from the Shift Right function. The serial input affects bit 1 of the first, or lower numbered, register. The serial output is derived from bit 16 of the highest numbered register in the register string.

2. The shift enable and serial shift input circuits operate identically in both the Shift Right and Shift Left functions.

An example of the operation of the Shift Right function is shown in Figure SR-3. A single-word holding register

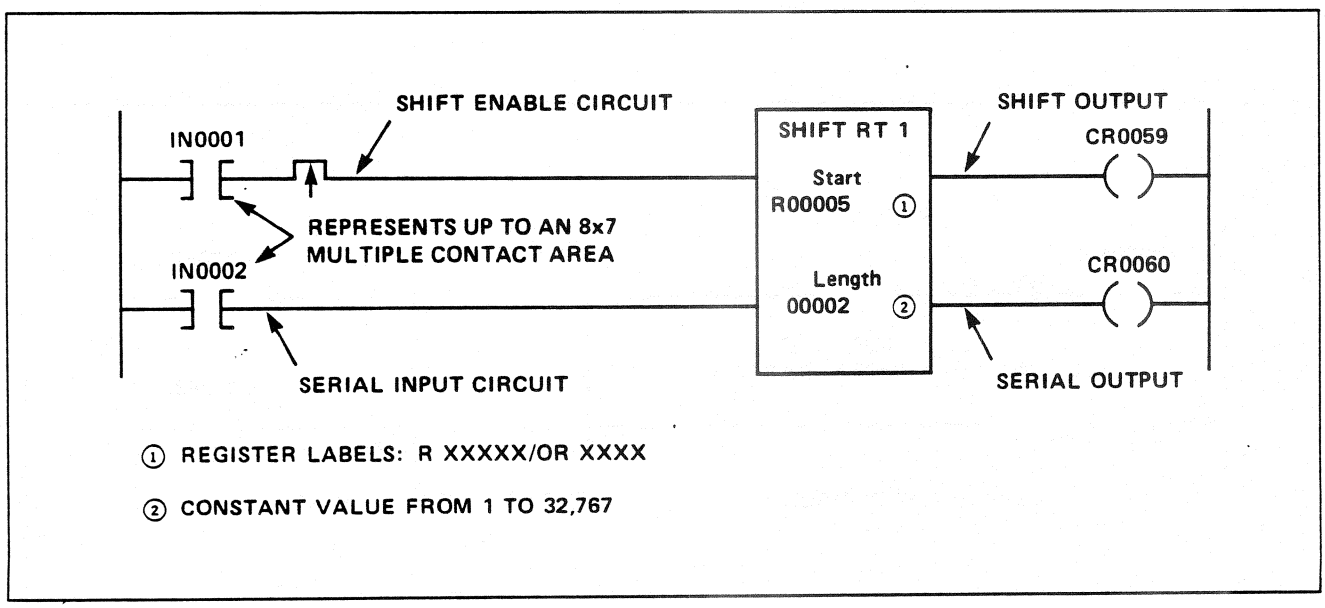


Figure SR-1. Shift Right 1 Circuit (Typical)

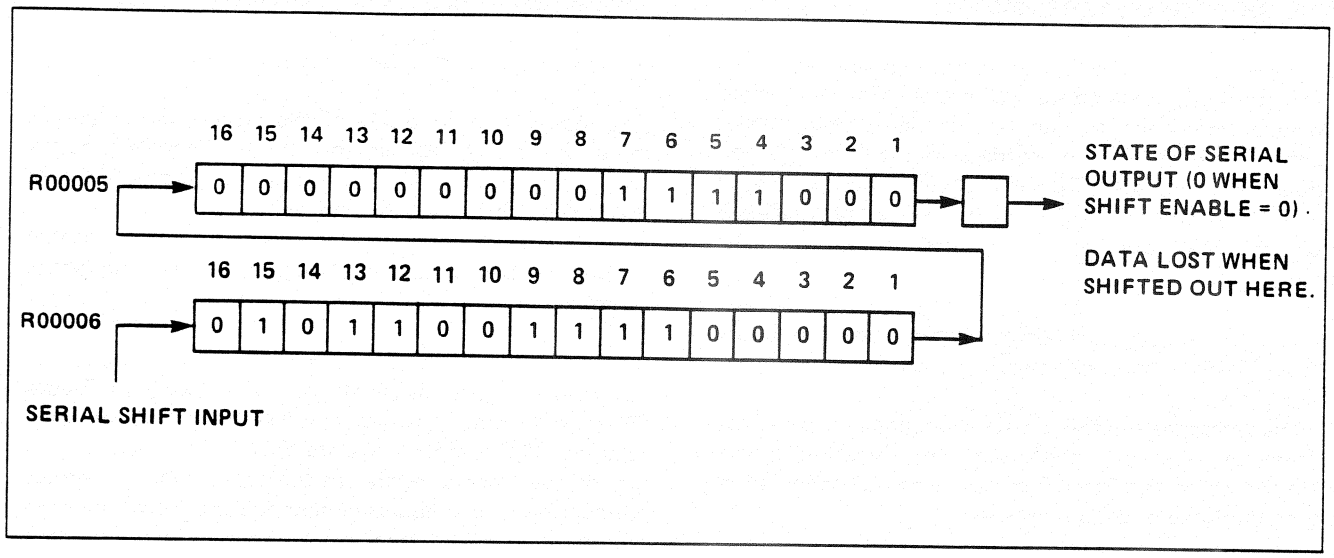


Figure SR-2. Shift Right Operation

(R00007) is shown for simplicity. The effects of 2 right shifts on the data contained in the register are also shown. Figure SL-3 shows the effects 2 left shifts have on holding register R00010.

7-10-1. SPECIFICATIONS

The programming specifications for the Shift Right and Shift Left functions are noted here.

SHIFT RIGHT/SHIFT LEFT FUNCTION BLOCK

The Shift Right and Shift Left function blocks each requires 2 horizontal contact spaces by 3 parallel rows on the display. The function blocks can be located anywhere in the contact area of the ladder diagram.

START REGISTER

The Start Register defines the first of the consecutively numbered registers used to make up the shift register string. This Register is specified by the programmer as a holding register (R) or output register (OR). The letters R or OR precede the register number on the ladder diagram display.

LENGTH

The Length of the register defines the number of holding

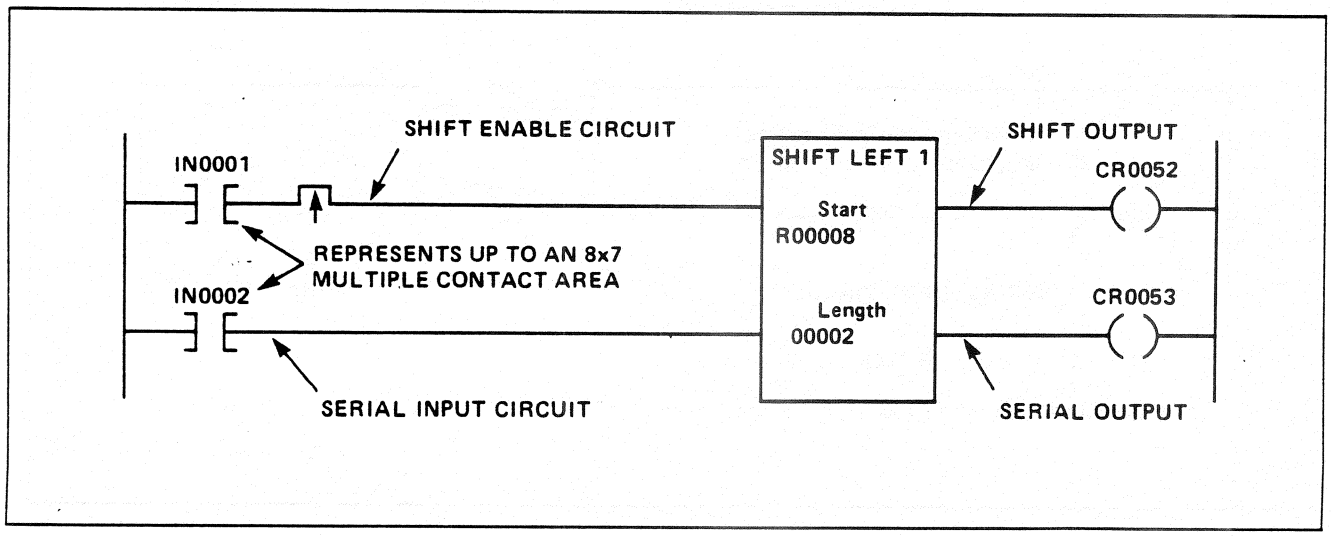


Figure SL-1. Shift Left Circuit (Typical)

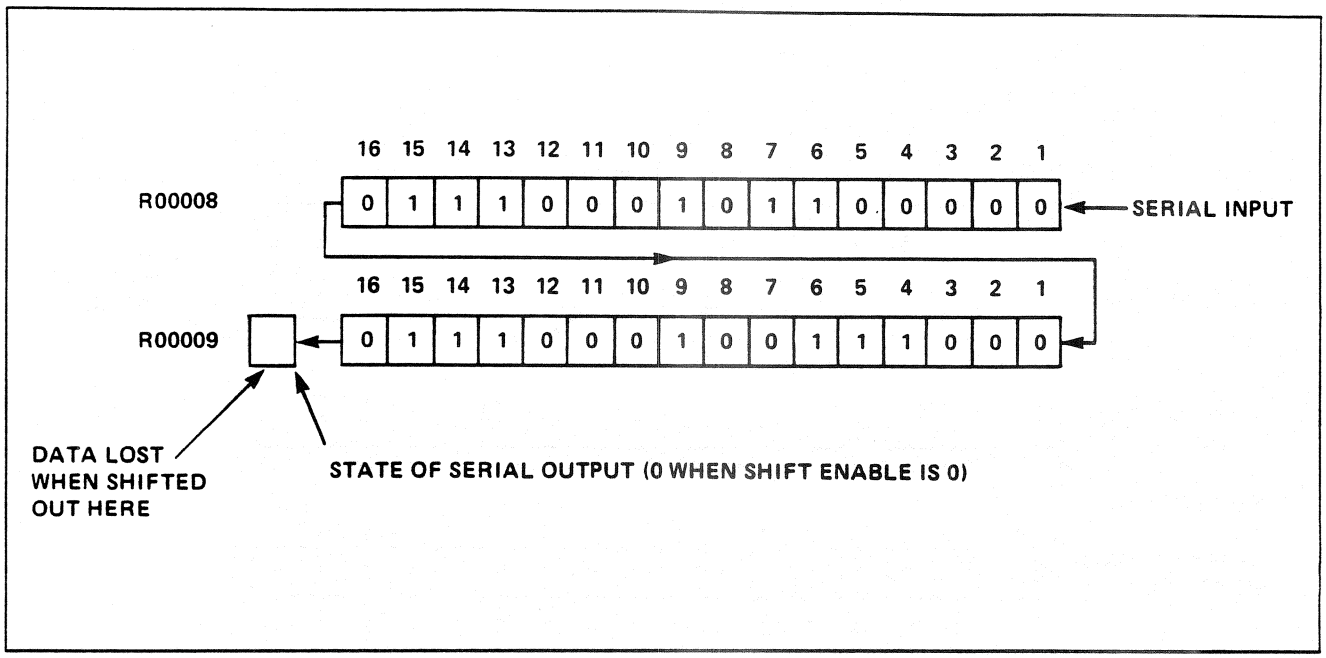


Figure SL-2. Shift Left Operation

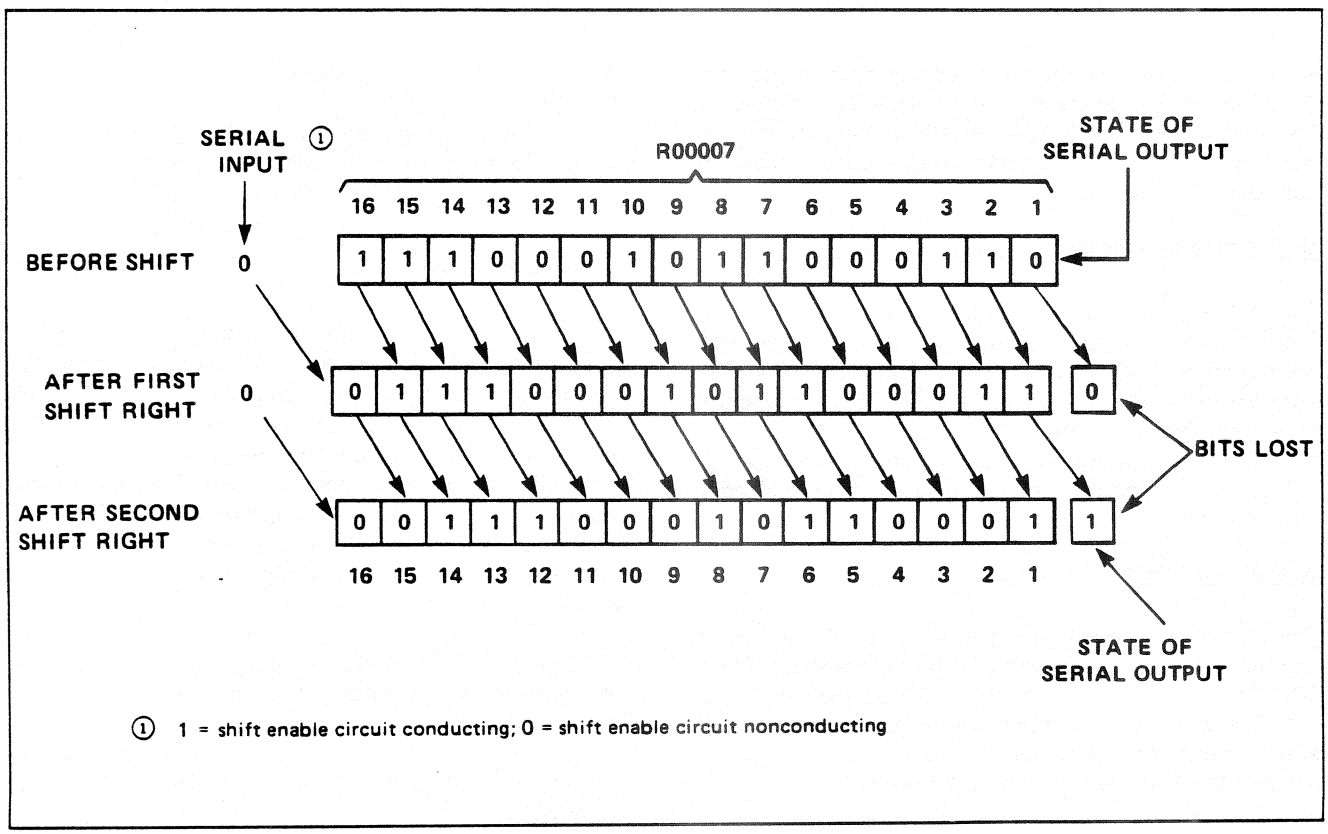


Figure SR-3. Shift Right Operation Example

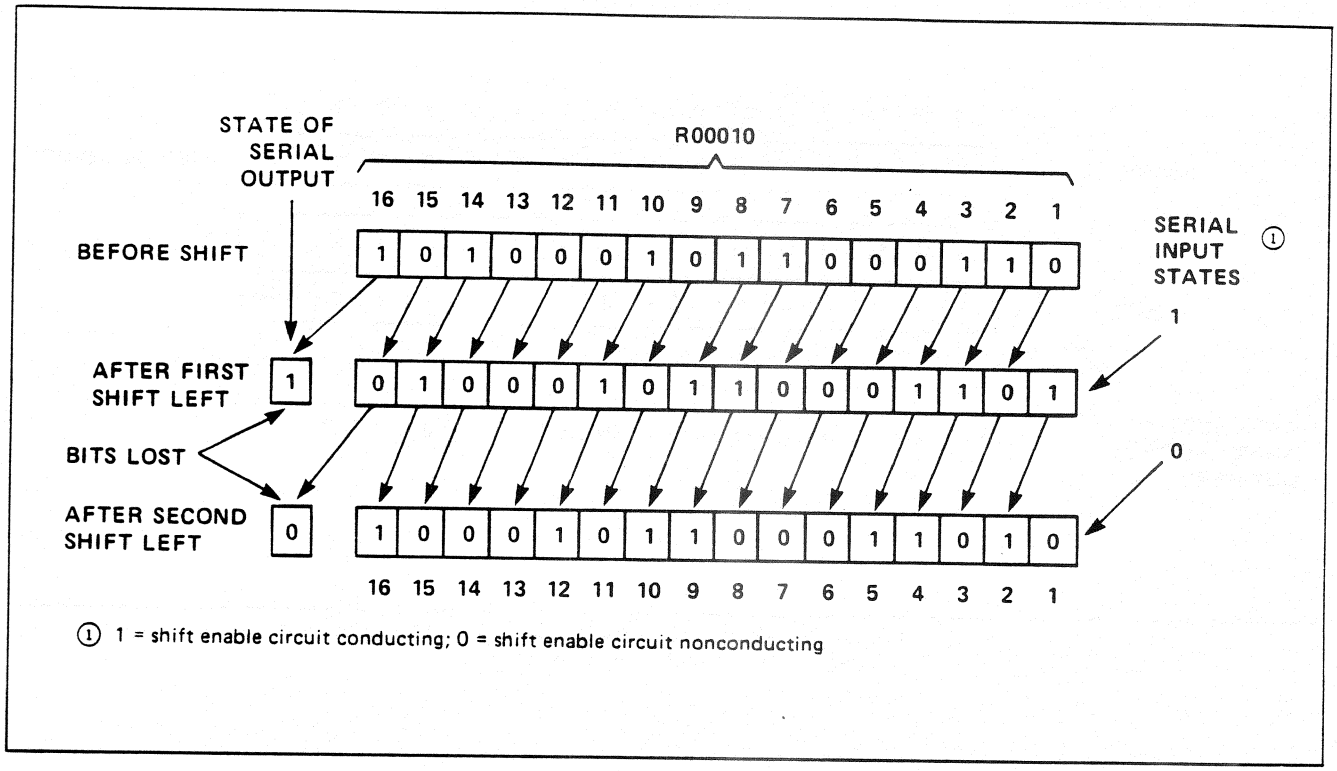


Figure SL-3. Shift Left Operation Example

or output registers defined in the string of shift registers. The number is a constant from 1 to 32,767 specified by the programmer. Since each register requires a word in memory, only the number of registers actually required should be programmed.

SHIFT ENABLE/SERIAL INPUT CIRCUITS

The shift enable and serial input circuits consist of up to 8 horizontal contacts located in up to 7 parallel paths positioned to the left of the Shift function block. The rules for programming contacts listed in Section 6 apply to the contacts of these circuits. Table SR-1 contains a truth table describing the characteristics of these input circuits. Note: The shift input can be left unconnected if not required.

SERIAL OUTPUT/SHIFT OUTPUT

The serial output circuit assumes the state of the current data (0 or 1) being shifted from the shift register. Table SR-1 shows how the input circuits affect the serial output. The data from the function block is present on the serial output for 1 scan only. It is the user's responsibility to latch or store this data, if needed.

The shift output simply follows, or repeats, the conducting or nonconducting state of the shift enable circuit.

Note

The outputs of the Shift Left and Shift Right functions can be left unconnected in the network.

7-10-2. APPLICATIONS

Shift Registers are useful for storing data in many process-type applications such as conveyors used for handling parts. The conveyors can be used for operations including painting, sand blasting, heat treating, etc. Figure SL-4 shows a typical process application where a continuously moving conveyor with equally spaced hooks, or stations, is used to transport parts through a production process. Each hook, or station, may or may not contain a part. Observe the Figure and note the following 2 points:

1. The step input (IN0001) closes as each hook or station passes by the input to the conveyor.

2. The "read" input (IN0002) consists of a photoelectric cell used to sense if a part is present (1) or not present (0).

The data is collected in the Shift Register and utilized as shown in Figure SL-4. Observe the Figure and note the

TABLE SR-1. SHIFT RIGHT/LEFT TRUTH TABLE

Shift ① Enable Input	Serial ① Input	Results:		
		Shift Output	Serial Output	Registers
0	0 or 1	0	Nonconducting	Remains in last state.
1	0	1	②	During each scan of the ladder diagram, data is shifted 1 position (right or left), and a 0 is loaded into the serial input.
1	1	1	②	During each scan of the ladder diagram, data is shifted 1 position (right or left), and a 1 is loaded into the serial input.

① 0 = circuit nonconducting; 1 = circuit conducting

② Current data (0 or 1) from the serial output of the shift register

following 3 points:

1. Holding registers R00005 and R00006 are used as a 2-word Shift Left register.

2. If a part is input to the line when the step input first conducts, a 1 is input to bit 1 of holding register R00005. If a part is not input to the line when the step input first conducts, a 0 is input to bit 1 of holding register R00005.

3. The various bits in the Shift Register can be continuously monitored to energize and de-energize outputs as required by the application.

Figure SL-4 contains two typical examples where bits in the Shift Register are examined. These are:

- Example 1, where CR0050 is energized whenever bit 3 of holding register R00005 is in a 1 state (the third position in the Shift Register = 1).
- Example 2, where CR0053 is energized whenever bit 5 of holding register R00005 is in a 1 state.

In this way Shift Register functions can be used to store data and control a machine or process.

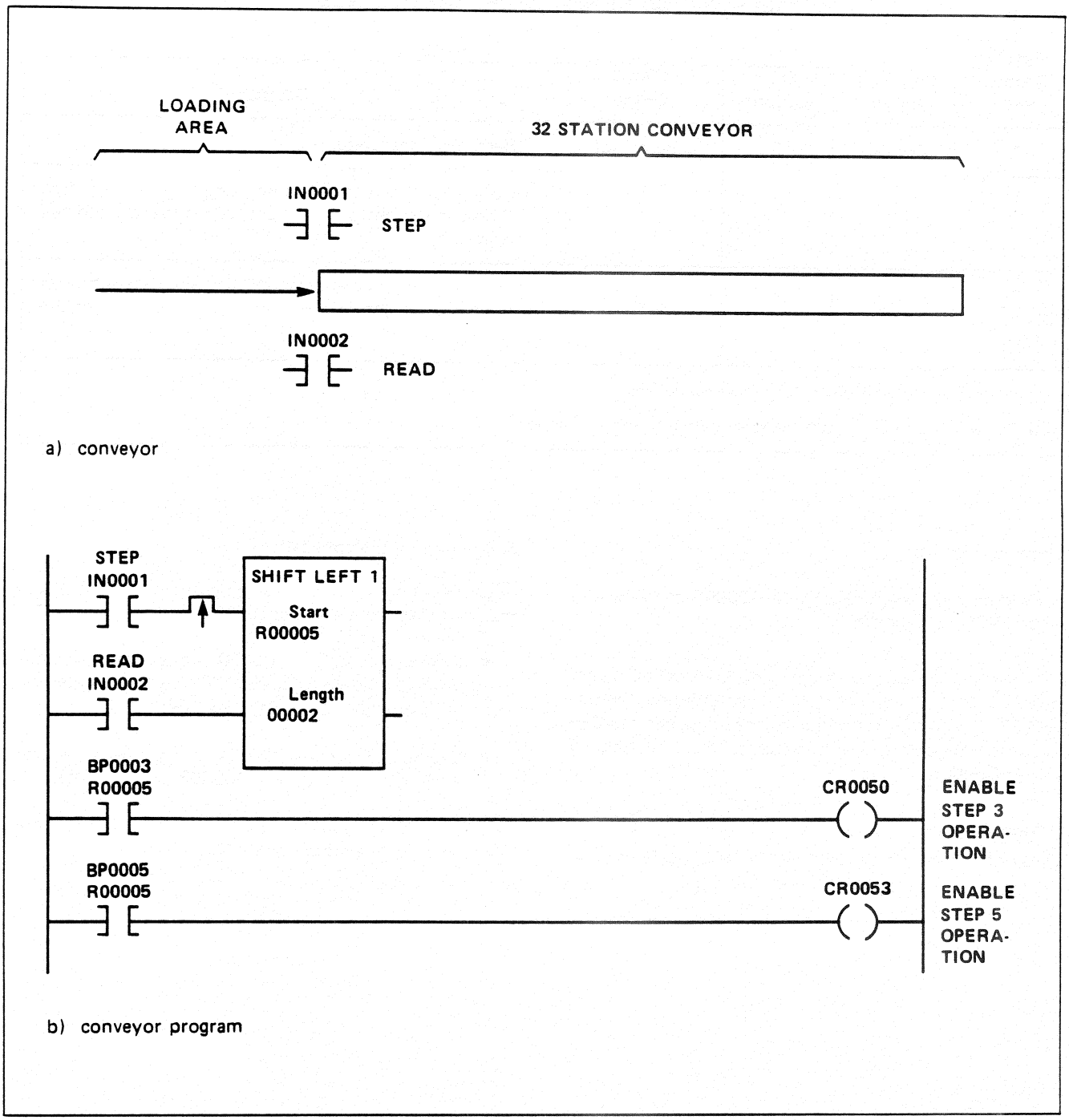


Figure SL-4. Shift Register Application

7-11. BINARY-TO-BCD REGISTER (045)

The single-precision Binary-to-BCD (binary coded decimal) Register programmable function converts the data contained in a single selected register from binary to BCD. Since a 16-bit binary number can require as many as 20 bits when converted to BCD, the result of the conversion is held in a consecutively numbered pair of registers. A typical example of the Binary-to-BCD Register function is shown in Figures B/BCD R-1 and R-2. Observe the Figures and note the following 4 points:

1. The conversion of the data contained in the Source Register takes place whenever the convert enable circuit is conducting.

2. The result of the conversion is held in 2 consecutively numbered registers. (These are holding registers R00007 and R00008 in this example.)

3. The convert output of the Binary-to-BCD Register function simply repeats, or follows, the conducting or nonconducting state of the convert enable circuit.

4. The sign output of the function block conducts when the Source Register contains a negative number (in 2's complement form) and when the convert enable circuit is conducting. In this example it is used to enable bit 5 of holding register R00007.

Note

The Binary-to-BCD Register function converts a single-precision (16-bit) register into a 5-digit BCD result contained in 2 consecutively addressed registers. Do not confuse it with the Double-Precision Binary-to-BCD Register function (046), which is described in Paragraph 7-12. The 046 programmable function converts a double-precision (32-bit) number into a 10-digit BCD result contained in 3 registers. Refer to Table 7-1, and note that the function menu uses an asterisk (*) after the double-precision function.

7-11-1. SPECIFICATIONS

The programming specifications for the Binary-to-BCD Register conversion function are listed here:

BINARY-TO-BCD REGISTER FUNCTION BLOCK

The Binary-to-BCD Register function block requires 2 horizontal contact spaces by 3 parallel rows on the ladder diagram. The Binary-to-BCD Register function block can be located anywhere in the contact area of the ladder diagram.

CONVERT ENABLE CIRCUIT

The convert enable circuit conducts, thereby enabling the binary-to-BCD conversion of data. When the convert enable is nonconducting, conversion ceases; all outputs are nonconducting; and the last data is held in the Destination Register.

The convert enable circuit is located on the ladder diagram to the left of the function block. The circuit

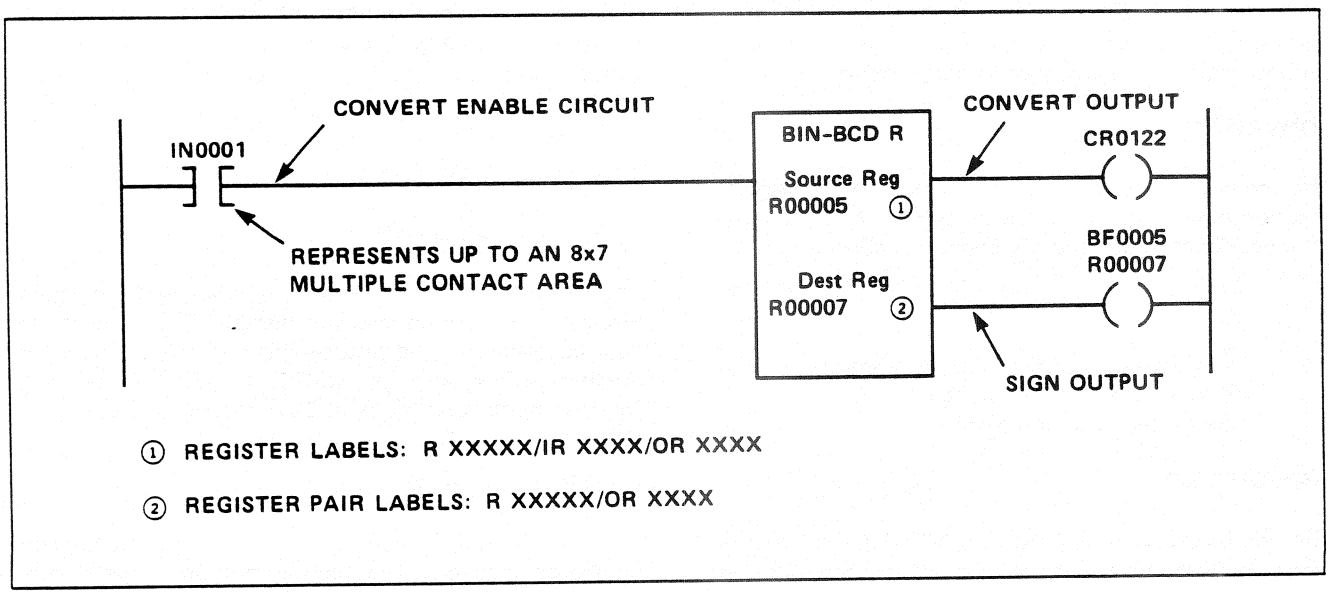


Figure B/BCD R-1. Binary-to-BCD Register Function Circuit (Typical)

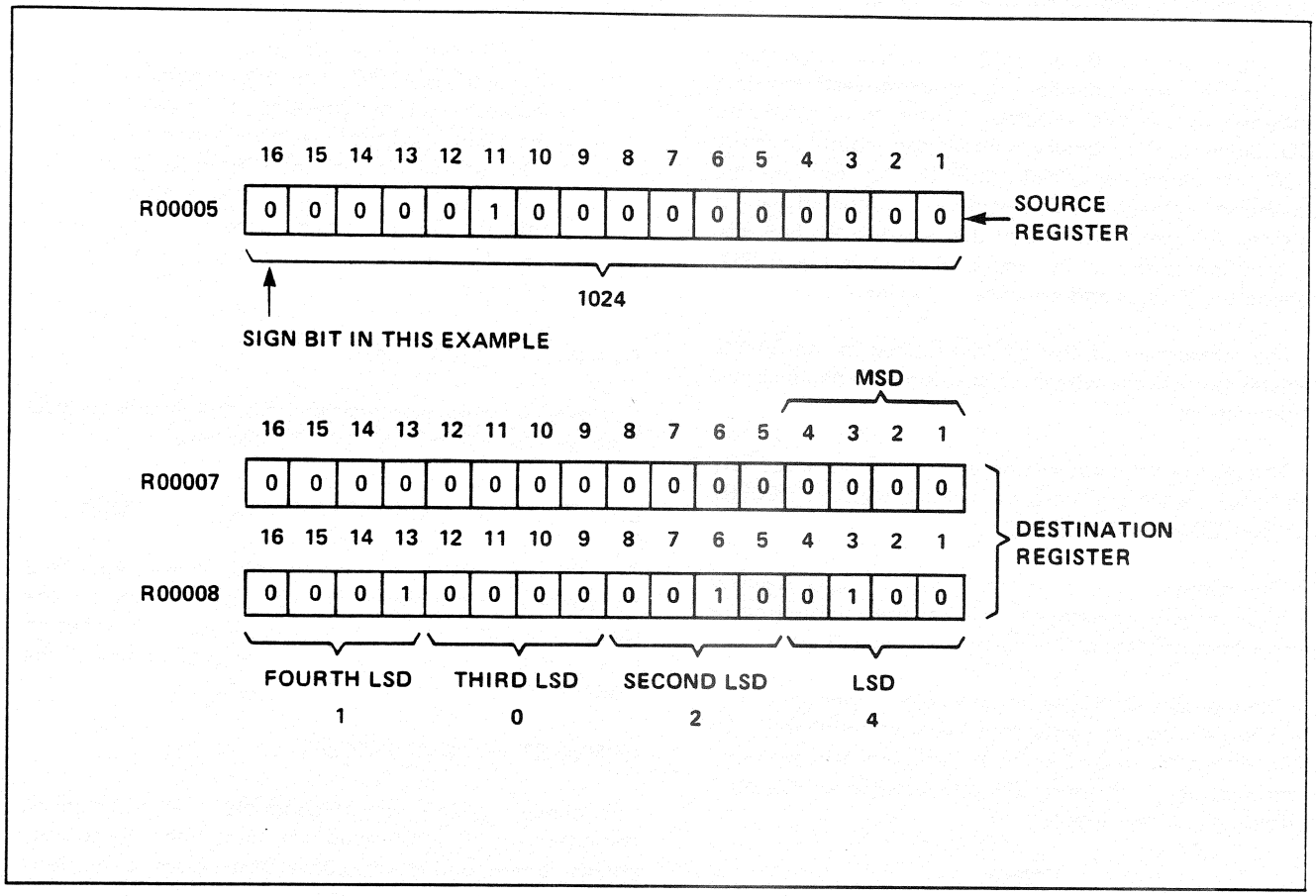


Figure B/BCD R-2. Binary-to-BCD Register Operation

consists of up to 8 horizontal contacts located in up to 7 parallel paths. This assumes the Binary-to-BCD Register function block is positioned on the far right portion of the display. The rules for programming contacts listed in Section 6 apply to the convert enable circuit.

CONVERT OUTPUT

The convert output of the Binary-to-BCD Register function block simply repeats or follows the conducting or nonconducting state of the convert enable circuit.

Note

The convert and sign outputs from the Binary-to-BCD function can be left unconnected in the network.

SIGN OUTPUT

The sign output of the Binary-to-BCD Register function block conducts whenever both the convert enable circuit of the function conducts, and the Source Register contains a negative number (bit 16 is in a 1 state).

SOURCE REGISTER

The Source Register defines the location of data to be converted. The Source Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR or IR precede the ladder diagram display of the Source Register number The value of the data contained in the Source Register can be any valid number in the range of +32,767 to -32,768, decimal. Negative numbers must be in 2's complement form.

DESTINATION REGISTER

The Destination Register consists of 2 consecutively numbered registers. The register number actually programmed and displayed in the Binary-to-BCD function block is the lower number of the pair. This Register

contains the most significant BCD digit. (See Figure B/BCD R-2.) The Destination Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the ladder diagram display of the Destination Register.

7-11-2. APPLICATIONS

A typical application of the Binary-to-BCD Register function is shown in Figure B/BCD R-3. It converts binary data from a holding register and stores the result in an output register. An Output Register Module, in turn, is wired to a 5-digit latching-type BCD display. In addition to the 5 BCD digits, the following outputs are also sent to the display:

- The sign bit from bit 5 of output register (R00007 in this case) is used to control the sign character of the display.
- The "convert enable" output (bit 6 of output register R00007 here) is used to latch the data in the BCD display. Data is accepted and can be changed by the BCD display only when this line is HI (1). When this line is LOW (0), the last state of the Destination Register is held in the display.

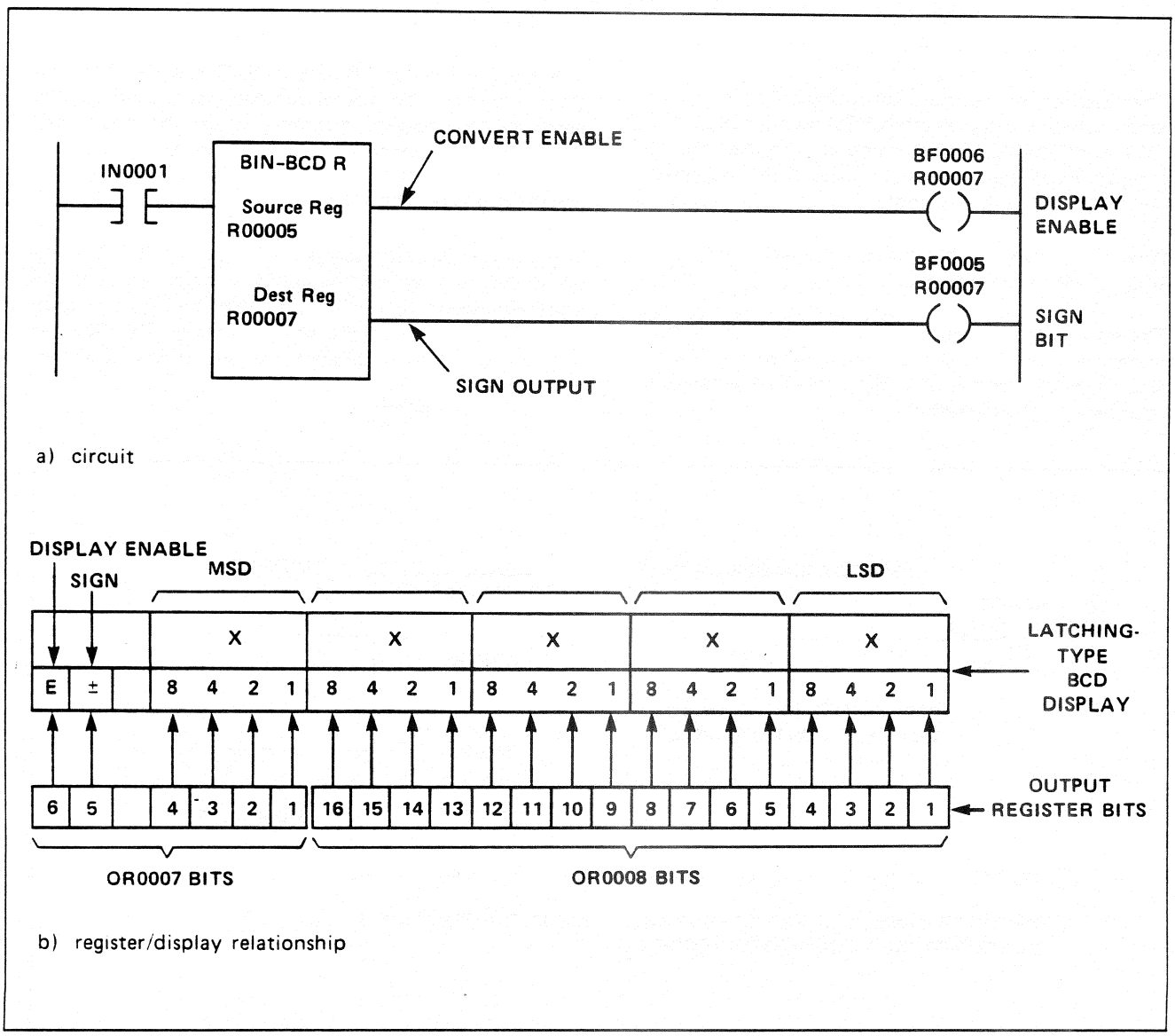


Figure B/BCD R-3. Binary-to-BCD Application

7-12. DOUBLE-PRECISION BINARY-TO-BCD REGISTER (046)

The Double-Precision Binary-to-BCD (binary coded decimal) Register programmable function converts a double-precision binary number (either positive or 2's complement form) to a BCD type number. The converted BCD number is placed in 3 consecutively numbered registers. See Figures B/BCD DP-1 and DP-2, and note the following 5 points:

1. The Source Register contains the binary number to be converted. It is located in 2 consecutively numbered registers where the most significant digits are contained in the register with the lower number (R00010 in this example).

2. The least significant digits of the Source Register are located in the second consecutively numbered register.

3. The Destination Register, consisting of three 16-bit registers, contains the converted BCD number (R00020, R00021 and R00022 in this example). The least significant digits are contained in the register with the highest number (R00022 in this example).

4. When the convert enable circuit conducts, the binary-to-BCD conversion takes place.

5. The sign output of the function block conducts when the binary number is negative—that is, when the Source Register is in 2's complement form. In this example the sign output energizes CR0048.

Note

The Double-Precision Binary-to-BCD Register function (046) converts a double-precision (32-bit) number. Do not confuse it with the Binary-to-BCD Register function (045) which converts a single-precision (16-bit) number. Refer to Table 7-1, and note that the function menu uses an asterisk (*) after the double-precision function.

7-12-1. SPECIFICATIONS

The programming specifications for the Double-Precision Binary-to-BCD function are listed here:

FUNCTION BLOCK

The Double-Precision Binary-to-BCD Register function block requires 2 horizontal contact spaces by 3 parallel paths on the ladder diagram. It can be positioned anywhere in the contact area of the display.

SOURCE REGISTER

The Source Register consists of 2 consecutively numbered registers, where the lower number contains the most significant digits. The data must be binary if positive, or in 2's complement form if negative. This Register is specified by the programmer as one of the following:

- Holding register (R)

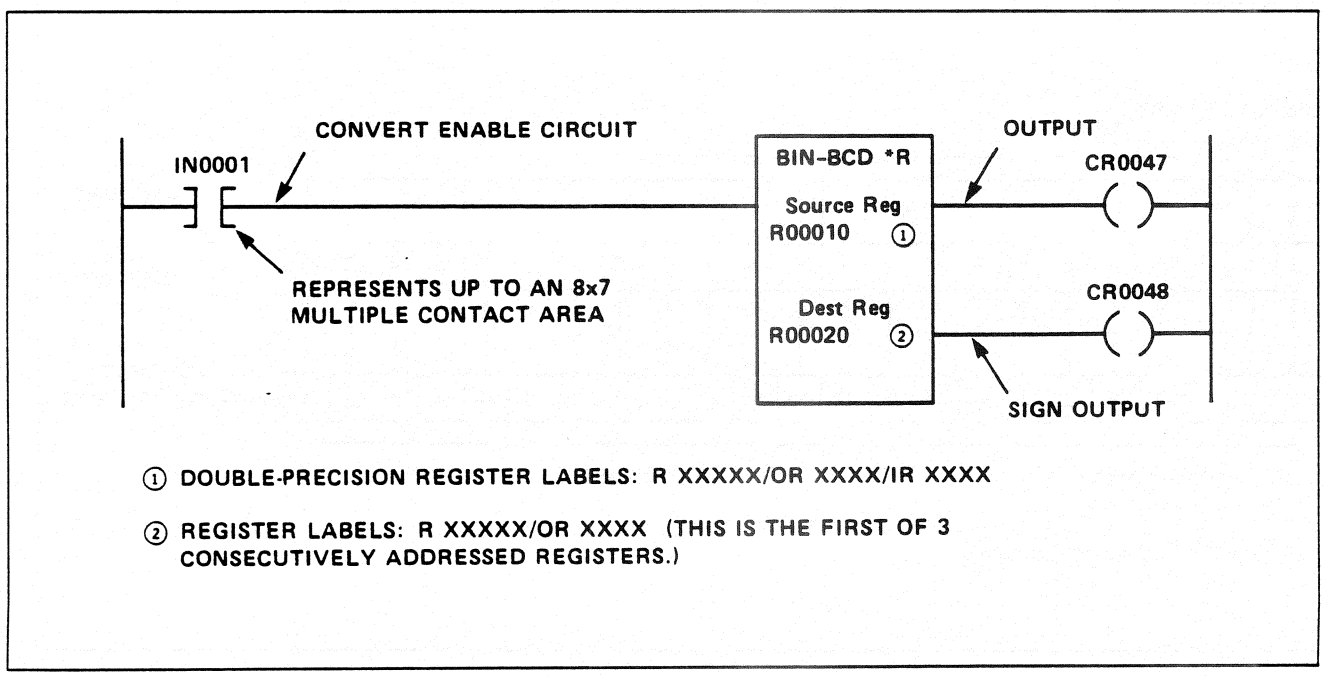


Figure B/BCD DP-1. Double-Precision Binary-to-BCD Circuit (Typical)

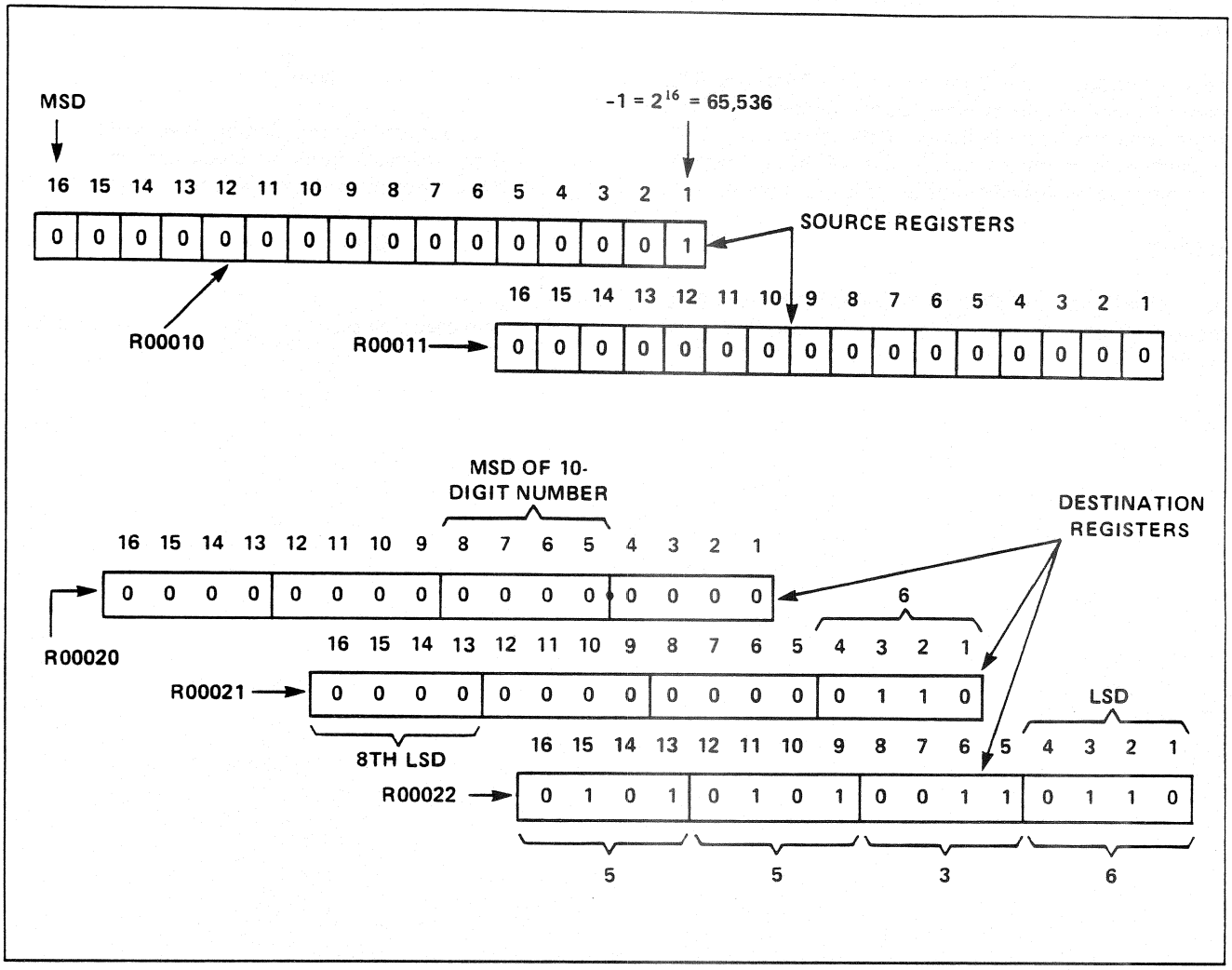


Figure B/BCD DP-2. Double-Precision Binary-to-BCD Operation

- Output register (OR)
- Input register (IR)

The letters R, OR or IR precede the register number on the display. The value of the data contained in the Source Register can be any valid number in the range of from $-(2^{31})$ to $+(2^{31}-1)$ or -2,147,483,648 to +2,147,483,647.

DESTINATION REGISTER

The Destination Register, consisting of three consecutively numbered registers, is used to store the BCD results of the conversion. The lowest numbered register of the 3 contains the most significant BCD digits. The converted BCD result is always BCD regardless of whether or not the Source Register is positive or negative.

This Register is specified by the programmer as one of the following:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the display.

CONVERT ENABLE CIRCUIT

The convert enable circuit conducts to enable the conversion process. When the circuit is nonconducting, conversion ceases, and the Destination Register is held in its last state. The enable circuit consists of up to 8 horizontal contact spaces by 7 parallel paths on the display.

SIGN OUTPUT

The sign output of the Double-Precision Binary-to-BCD function block conducts when both the convert enable circuit conducts and the Source Register contains a 2's complemented number—that is, bit 16 of the register containing the most significant digits is in a 1 state.

OUTPUT

The output of the Double-Precision Binary-to-BCD function simply repeats or follows the conducting or nonconducting state of the convert enable circuit.

Note

The outputs of the Double-Precision Binary-to-BCD function block can be left unconnected in the network.

7-12-2. APPLICATIONS

See Paragraph 7-11-2 for a typical application where a similar single precision binary word is converted to BCD.

7-13. BCD-TO-BINARY REGISTER (047)

The single-precision BCD-to-Binary Register programmable function converts the BCD data contained in a single register to binary format. The converted result is held in a separate register referred to as the Destination Register. A typical example of the BCD-to-Binary Register function is shown in Figures BCD/B R-1 and R-2. Observe the Figures and note the following 4 items:

1. The conversion of the value of the Source Register (IR0012 in this example) takes place whenever the convert enable circuit is conducting.

2. The result of the conversion is held in the Destination Register (holding register R00005 in this example).

3. The convert output of the BCD-to-Binary Register function simply repeats or follows the conducting or nonconducting state of the convert enable circuit.

4. The error output of the function block conducts when the number contained in the Source Register is not a legal BCD value and the convert enable circuit is conducting. This occurs when the binary value of any BCD digit is over 9 (1001). For example, a value of 11 (1011) in the least significant digit of the Source Register would enable the error output since a legal BCD digit range is from 0 to 9.

7-13-1. SPECIFICATIONS

The programming specifications for the BCD-to-Binary Register function are listed here:

FUNCTION BLOCK

This block requires 2 horizontal contact spaces by 3 parallel rows on the ladder diagram. The function block can be located anywhere in the contact area of the ladder diagram.

CONVERT ENABLE CIRCUIT

When the convert enable circuit conducts the BCD data contained in the Source Register is converted to binary data and placed in the Destination Register. The convert enable circuit is located on the ladder diagram to the left of the function block

The circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths. This assumes the BCD-to-Binary Register function block is positioned on the far right portion of the display. The rules for programming contacts listed in Section 6 apply to the convert enable circuit.

SOURCE REGISTER

The Source Register defines the location of data to be converted. The value of this Register can range from 0 to 9,999. The Source Register can be programmed as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR or IR precede the ladder diagram

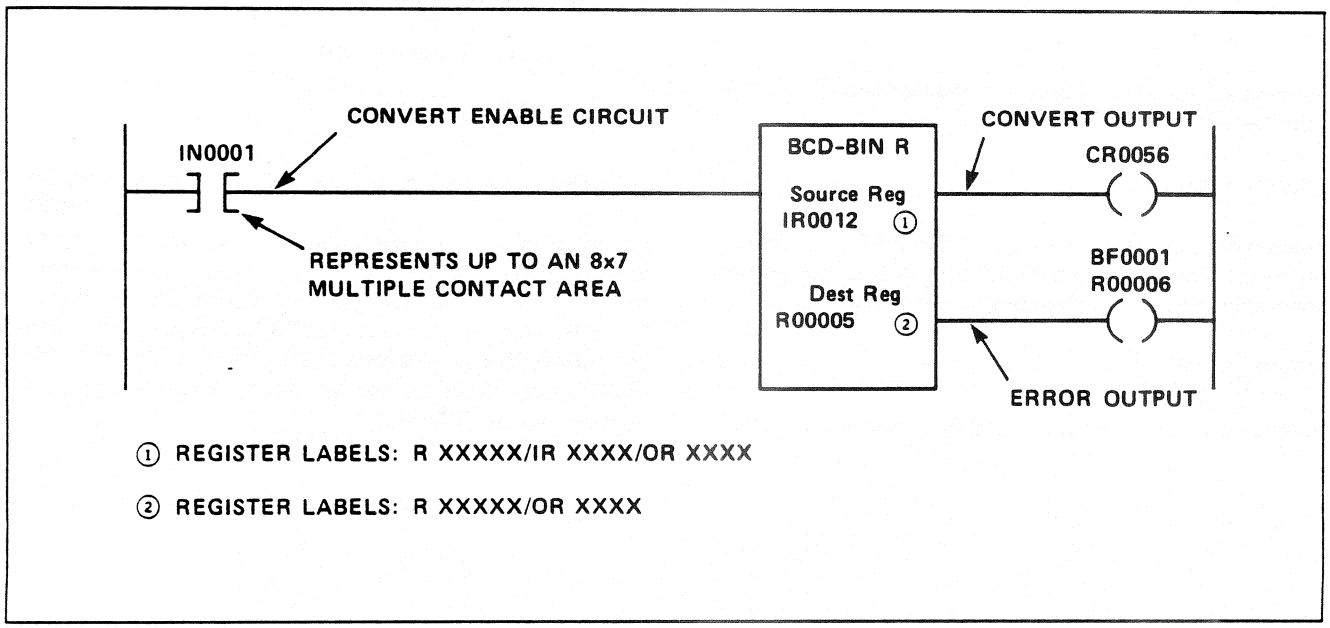


Figure BCD/B R-1. BCD-to-Binary Register Function Circuit (Typical)

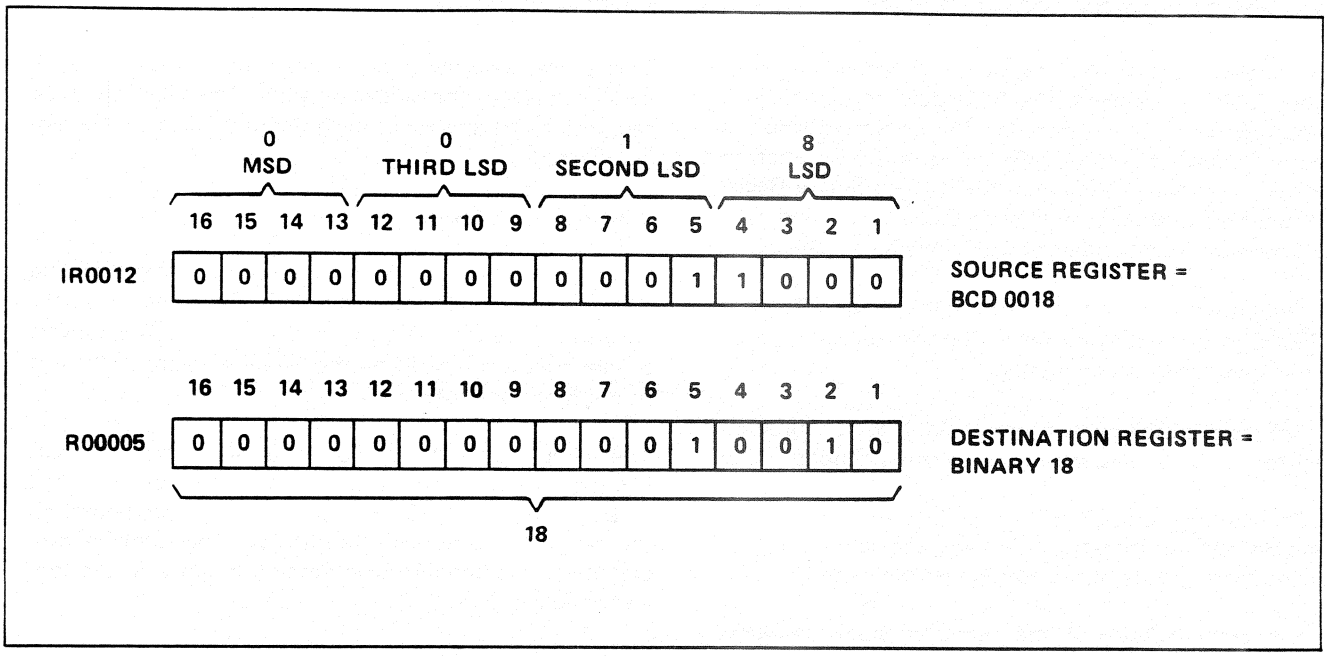


Figure BCD/B R-2. BCD-to-Binary Register Operation

display of the Source Register.

DESTINATION REGISTER

The Destination Register holds the binary value of the converted BCD data, as shown in Figure BCD/B R-2. The Destination Register can be programmed as a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the ladder diagram display of the Destination Register.

CONVERT OUTPUT

The convert output of the BCD-to-Binary Register function block simply repeats or follows the conducting or nonconducting state of the convert enable circuit.

ERROR OUTPUT

The error output of the BCD-to-Binary Register function block conducts whenever both:

- The convert enable circuit conducts, and
- Any BCD digit of the Source Register exceeds 9 (1001).

Note

The outputs of the BCD-to-Binary function block can be left unconnected in the network.

7-13-2. APPLICATIONS

A typical application of the BCD-to-Binary Register function is shown in Figure BCD/B R-3. It converts BCD information from a 4-digit thumbwheel switch wired to input register IR0008 into binary data. The data from the thumbwheel switch is only converted when the ENTER pushbutton wired to IN0012 is pressed. When the pushbutton is depressed, the information from the thumbwheel is converted to binary and entered into holding register R00022.

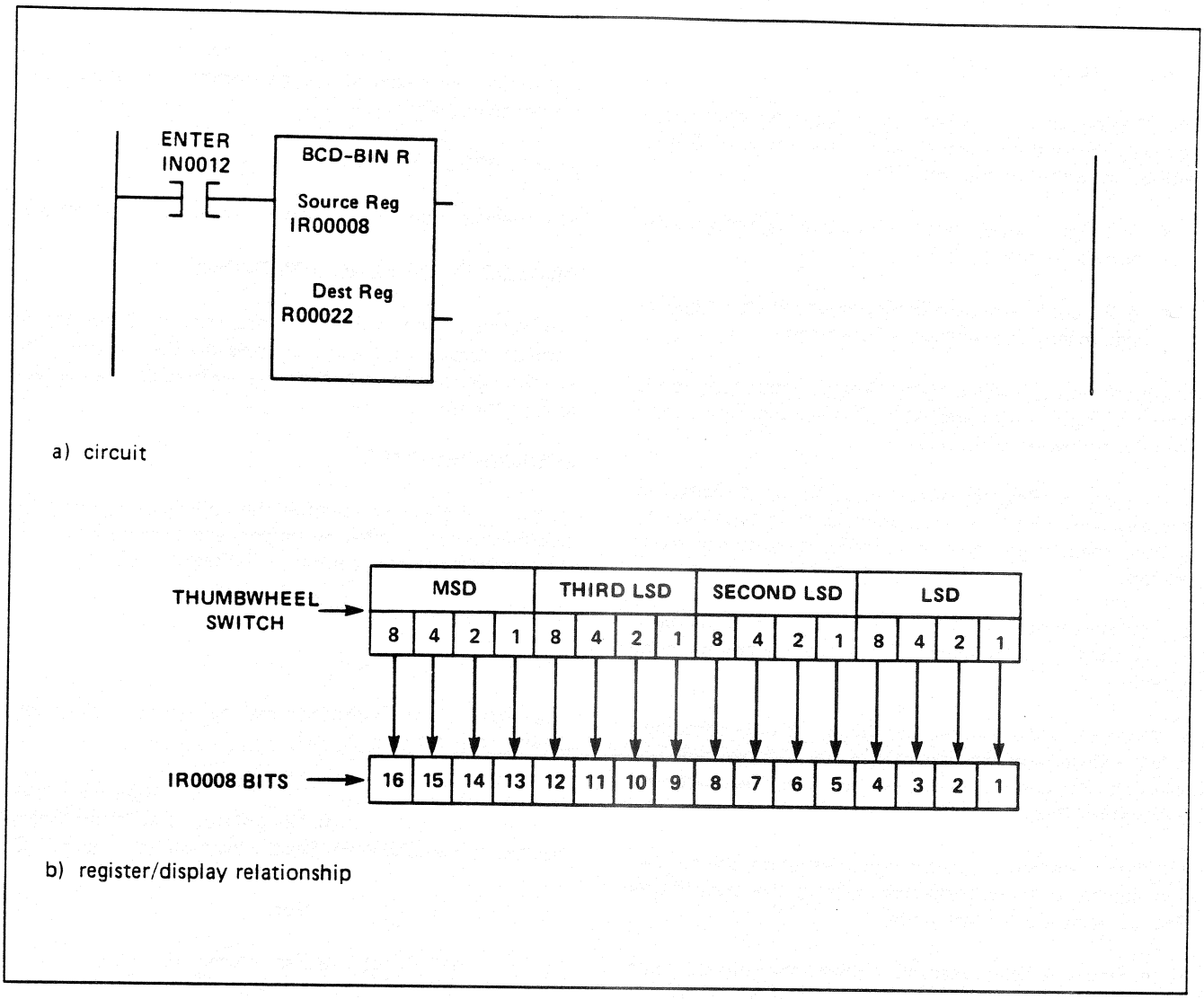


Figure BCD/B R-3. BCD-to-Binary Register Function Application

7-14. ANALOG INPUT (048)

The Analog Input programmable function converts a sign-magnitude type number within the range of 0 thru -4095 into either of the following:

- A binary value, when a positive sign-magnitude number is converted
- A 2's complemented binary value when a negative sign-magnitude number is converted

A typical circuit containing the Analog Input function is shown in Figures AI-1 and AI-2. Observe the Figures and note the following 6 points:

1. The Source Register, specified as an input register in this example, contains the sign-magnitude type data to be converted. It can be either a holding register or input register.

2. The Destination Register, specified as a holding register, is used to contain the converted 2's complement type data.

3. When the enable circuit conducts, the data contained in bits 1 thru 12 of the Source Register is converted. The resulting 2's complement number is loaded into the Destination Register.

4. When the enable circuit is nonconducting, the conversion ceases, and the converted data in the Destination Register is held at its last value.

5. In Figure AI-2(a), example 1 shows the conversion of a positive sign-magnitude number into a binary value.

6. In Figure AI-2(b), example 2 shows the conversion of a negative sign-magnitude type number into a 2's complemented binary value.

7-14-1. SPECIFICATIONS

The specifications for the Analog Input function follow.

ANALOG INPUT FUNCTION BLOCK

The Analog Input function block requires 2 horizontal contact spaces by 3 parallel rows on the display. The function block can be allocated anywhere on the contact area of the display screen.

SOURCE REGISTER

The Source Register specifies the register containing the sign-magnitude number to be converted into a binary or 2's complement type format. This Register is a:

- Holding register (R)
- Input register (IR)

The letters R or IR precede the register number on the display.

Bit 16 of this Register contains the sign (0 = plus; 1 = negative). Bits 1 thru 12 contain the absolute binary number to be converted. See the examples of Figure AI-2.

Note

The Front Access Panel or APL Loader cannot directly display a

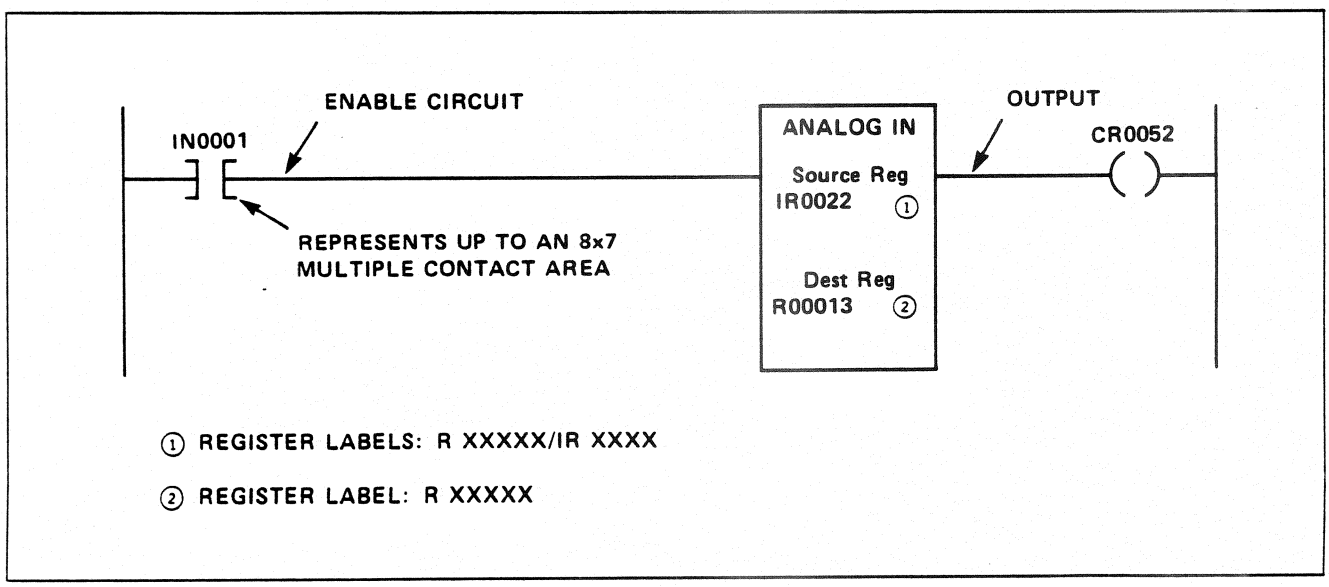


Figure AI-1. Analog In Function Circuit (Typical)

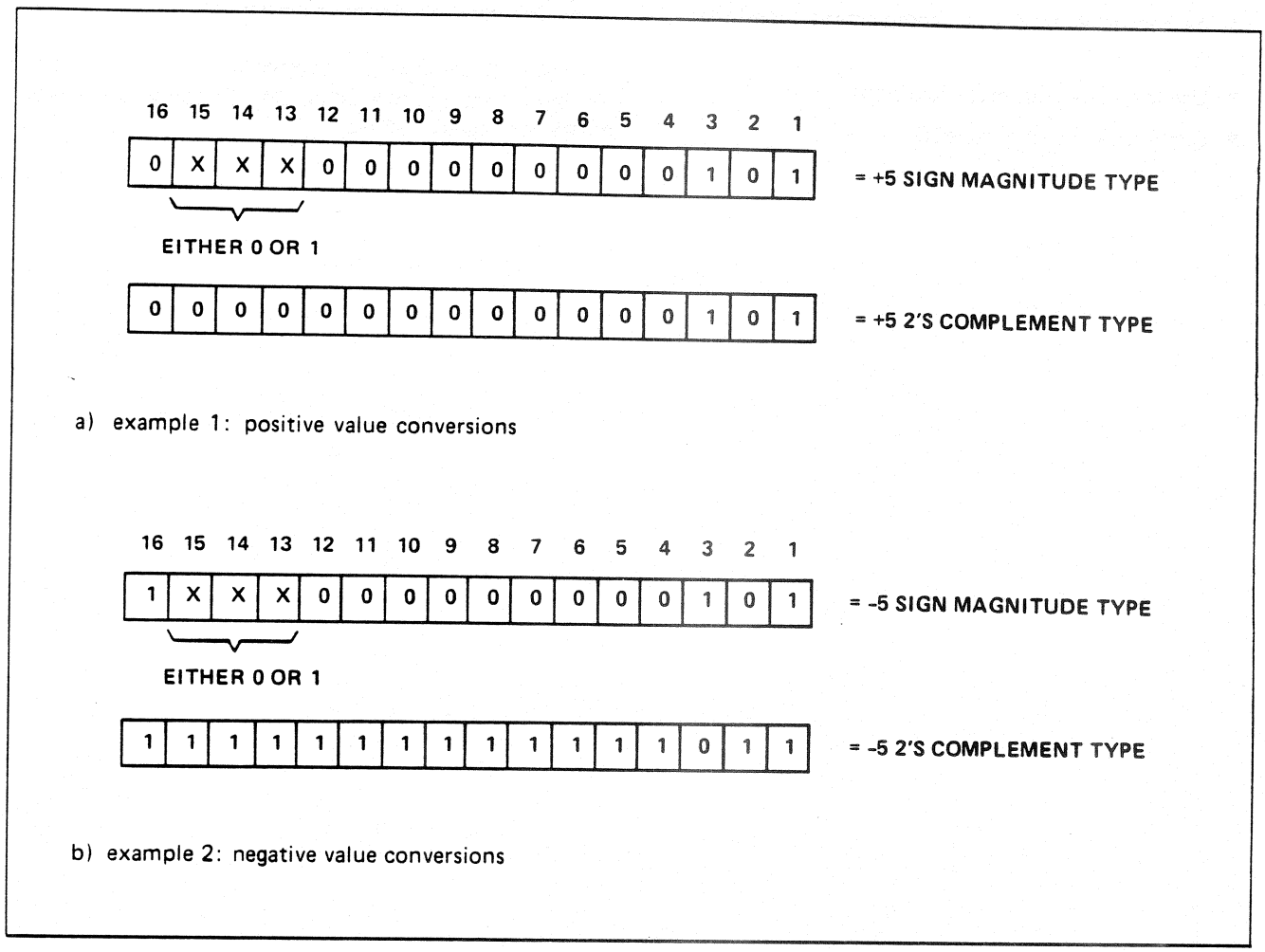


Figure AI-2. Analog In Function Operation

sign-magnitude value, although positive values are displayed correctly when the decimal mode is selected.

DESTINATION REGISTER

The Destination Register holds the binary or converted 2's complement type data. It is specified as a:

- Holding register (R)

ENABLE CIRCUIT

When the enable circuit conducts, the sign-magnitude type data is converted during each scan of the ladder diagram. When the enable circuit is nonconducting, no conversion takes place, and the data contained in the Destination Register is held at its last value.

The enable circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths placed to the left of the Analog Input function block on the ladder diagram.

OUTPUT

The output from the Analog In function simply follows or repeats the conducting or nonconducting state of the enable circuit.

Note

The output of the Analog Input function can be left unconnected in the network.

7-14-2. APPLICATIONS

The Analog Input function is used anytime it is necessary to convert a sign-magnitude type number into a 2's complement type binary number. Analog Input Modules, for example, supply a sign-magnitude number to the processor. The input from the Analog Input Module

must be converted to the 2's complement type binary number in order to:

- Use the value with math functions
- Convert the number to BCD
- Compare the number to other values

The math, conversion, and comparison type functions all use binary for positive values and 2's complement type binary for negative values.

7-15. ANALOG OUTPUT (049)

The Analog Output programmable function converts a binary or 2's complement type binary number into a sign-magnitude type number. The Analog Output function assumes that the data to be converted is:

- Pure binary within the range of +1 thru +4095 if the number is positive.
- 2's complement binary within the range of -1 thru -4096 if the number is negative.

The data to be converted is contained in bits 1 thru 12 of the Source Register while bit 16 is used as the sign bit.

A typical circuit containing the Analog Output function is shown in Figures AO-1 and AO-2. Observe the Figures and note the following 6 points:

1. The Source Register specifies the holding register containing the binary or 2's complement type binary number to be converted.

2. The Destination Register holds the converted value. It can be assigned to an output or holding register.

3. When the enable circuit conducts, the data contained in bits 1 thru 12 of the Source Register is converted. The resulting sign-magnitude data is loaded into bits 1 thru 12 of the Destination Register with bit 16 containing the sign information (0 = plus; 1 = minus).

4. When the enable circuit is nonconducting, the conversion ceases, and the converted data in the Destination Register is held at its last value.

5. Figure AO-2(a), shows the conversion of a positive binary number to a sign-magnitude number.

6. Figure AO-2(b), shows the conversion of a 2's complement number to a sign-magnitude number.

7-15-1. SPECIFICATIONS

The specifications for the Analog Output function are listed here.

ANALOG OUTPUT FUNCTION BLOCK

The Analog Output function block requires 2 horizontal contact spaces by 3 parallel rows on the display. The function block can be located anywhere on the contact area of the display screen.

SOURCE REGISTER

The Source Register specifies the holding register (R) which contains the binary of 2's complement type binary numbers. The value of this Register can range from -4096 thru +4095. Bits 13 thru 15 cannot be used to contain data, although when the value of the Source Register is:

- Positive, bits 13 thru 15 must be 0 to be in the range of from 0 thru 4095
- Negative, bits 13 thru 15 must be 1 to be in the range of from 0 thru -4096

See the conversion examples of Figure AO-2.

DESTINATION REGISTER

The Destination Register holds the converted

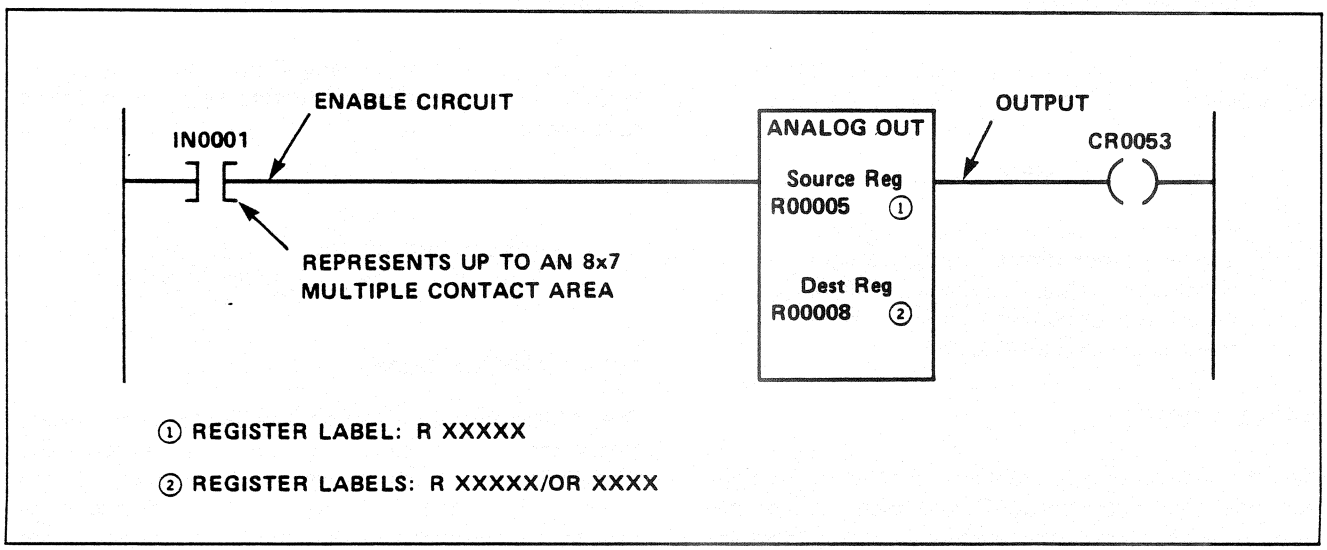


Figure AO-1. Analog Out Function Circuit (Typical)

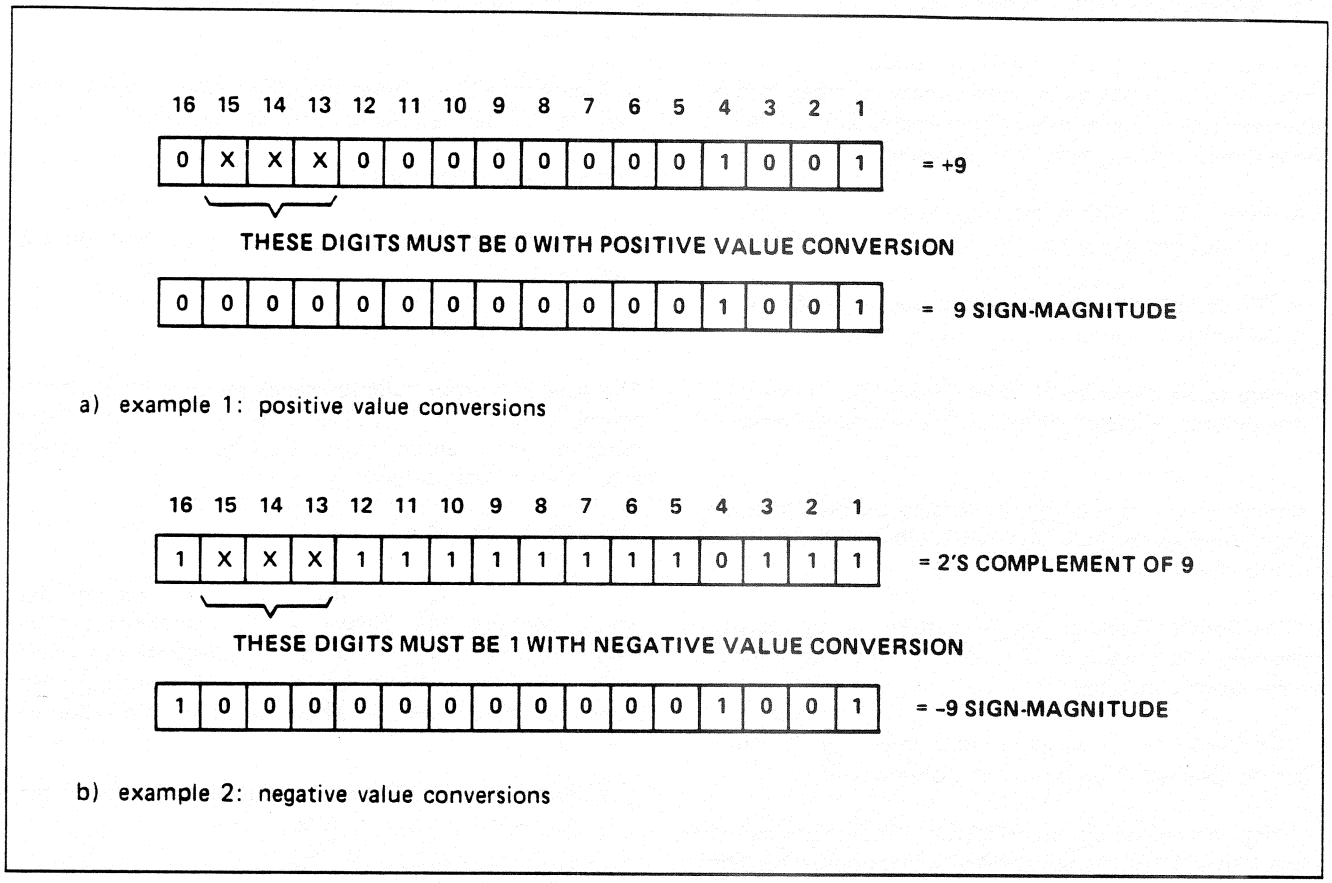


Figure AO-2. Analog Out Function Operation

sign-magnitude data. It is specified as a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the display screen.

Note

The Front Access Panel or APL Loader cannot directly display a sign-magnitude value, although positive values are displayed correctly when the decimal mode is selected.

ENABLE CIRCUIT

When the enable circuit conducts, the binary to sign-magnitude conversion occurs during each scan of the ladder diagram. When the enable circuit is nonconducting, no conversion takes place, and the Destination Register is held at its last value.

The enable circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths. The enable circuit is positioned to the left of the Analog Output function.

OUTPUT

The output from the Analog Output function simply follows or repeats the conducting or nonconducting state of the enable circuit.

Note

The output of the Analog Output function can be left unconnected in the network.

7-15-2. APPLICATIONS

The Analog Output function can be used to convert binary and 2's complement type binary data used in the math, comparison, and conversion type functions into sign-magnitude type data. The converted sign-magnitude data in the Destination Register of the Analog Output function is compatible with the Analog Output Module or any other device which accepts sign-magnitude data.

7-16. MOVE REGISTER-TO-REGISTER (050)

The Move Register-to-Register programmable function allows the data in a single register to be transferred, or "moved," and, in effect, duplicated in an entirely different register. A typical circuit containing the Move Register-to-Register function is shown in Figure MV-1. Observe the Figure and note the following 4 points:

1. The Source Register defines the register to be transferred. The data in this register does not change as the move occurs.

2. The Destination Register is the register into which the data is transferred.

3. When the move enable circuit conducts, the transfer occurs. Any previous data contained in the Destination Register is lost.

4. When the move enable circuit is nonconducting, no data is transferred. The data in the Destination Register is held at its last value.

7-16-1. SPECIFICATIONS

The programming specifications for the Move Register-to-Register function are listed here.

MOVE REGISTER-TO-REGISTER FUNCTION BLOCK

The Move Register-to-Register function block requires 2 horizontal contact spaces by 3 parallel rows on the display. The function block can be located anywhere in the contact area of the display screen.

SOURCE REGISTER

The Source Register defines the source of the register to be moved. This Register may be specified by the programmer as any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)
- Constant, from +32,767 to -32,768

The letters R, OR, IR, OG or IG precede the register number on the display. If a constant is selected, no letter precedes it on the display.

DESTINATION REGISTER

The Destination Register defines where the data is to be moved. It can be specified as any of the following:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR or OG precede the register or group

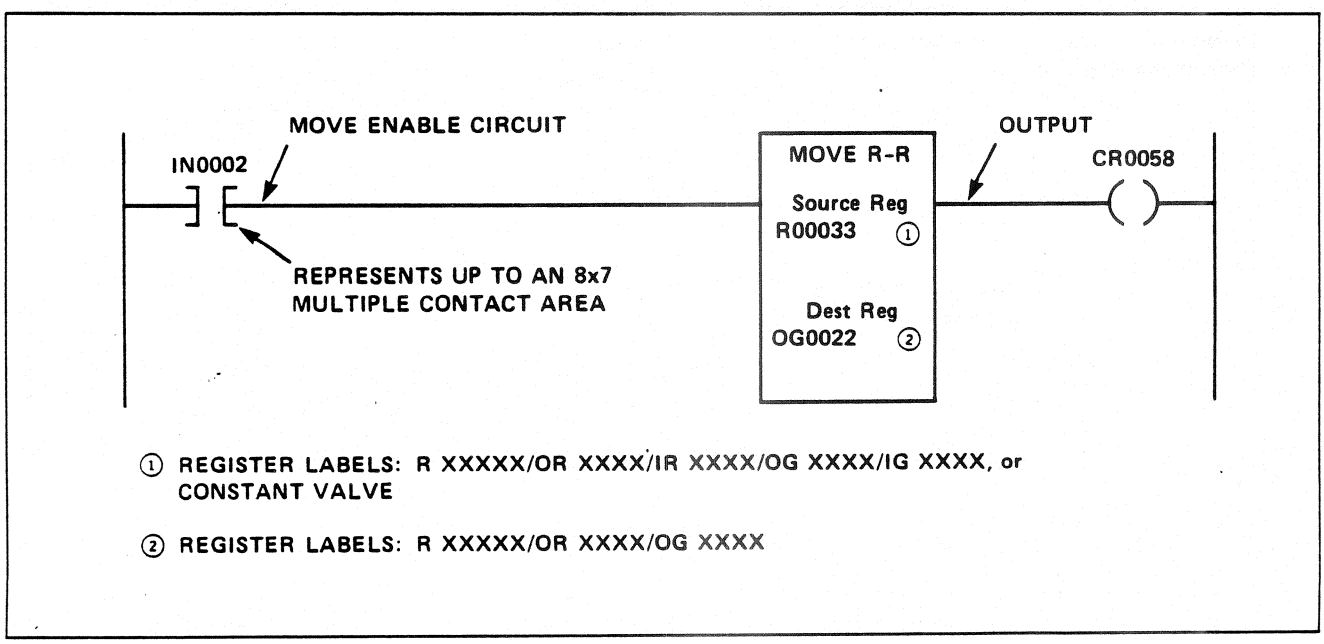


Figure MV-1. Move Register-to-Register Function Circuit (Typical)

number on the display.

WARNING

USING THE MOVE FUNCTION TO CONTROL AN OUTPUT GROUP (OG) OVERRIDES FORCE CONDITIONS IN EFFECT ON THE OUTPUTS PROGRAMMED. IF, FOR EXAMPLE, CR0006 WERE FORCED OFF PRIOR TO THE MOVE, IT COULD BE FORCED ON AFTER. FAILURE TO OBSERVE THIS WARNING CAN RESULT IN EQUIPMENT DAMAGE AND/OR PERSONNEL INJURY FROM UNEXPECTED ENERGIZING OR DE-ENERGIZING OF OUTPUTS.

MOVE ENABLE CIRCUIT

When the enable circuit conducts, the data is transferred from the Source to the Destination Register once during each scan of the ladder diagram. When the enable circuit is nonconducting, no transferring of data occurs, and the data in the Destination Register is held at its last value.

The move enable circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths. The move enable circuit is located to the left of the function block on the ladder diagram.

OUTPUT

The output of the Move Register-to-Register function simply follows or repeats the conducting or nonconducting state of the enable circuit.

Note

The output of the Move Register-to-Register function block can be left unconnected in the network.

7-16-2. APPLICATIONS

The Move Register-to-Register function allows the transfer of 16 bits of data, including data associated with the discrete inputs and outputs, to a holding or output register. This ability to transfer words of data can simplify programming as shown in Figure MV-2. Observe the Figure and note the following 3 points:

1. Input IN0027 monitors the status of an external emergency stop string.

2. In this application when an emergency condition occurs, it is necessary to force the outputs of the machine or process to a predetermined condition.

3. The Move Register-to-Register functions are used to preload the values:

- 4090 into output group OG0001
- -896 into output group OG0002

The normally closed contacts from output CR0042 must also be programmed in series with each of the rungs controlling the discrete output bits contained in output groups OG0001 and OG0002. Alternately, MCR Coil function (005) could be used to enable and disable the normal program and the E-stop rungs of the program. This forces the output coils CR0001 thru CR0032 to the energized (1) or de-energized (0) states shown in Figure MV-2.

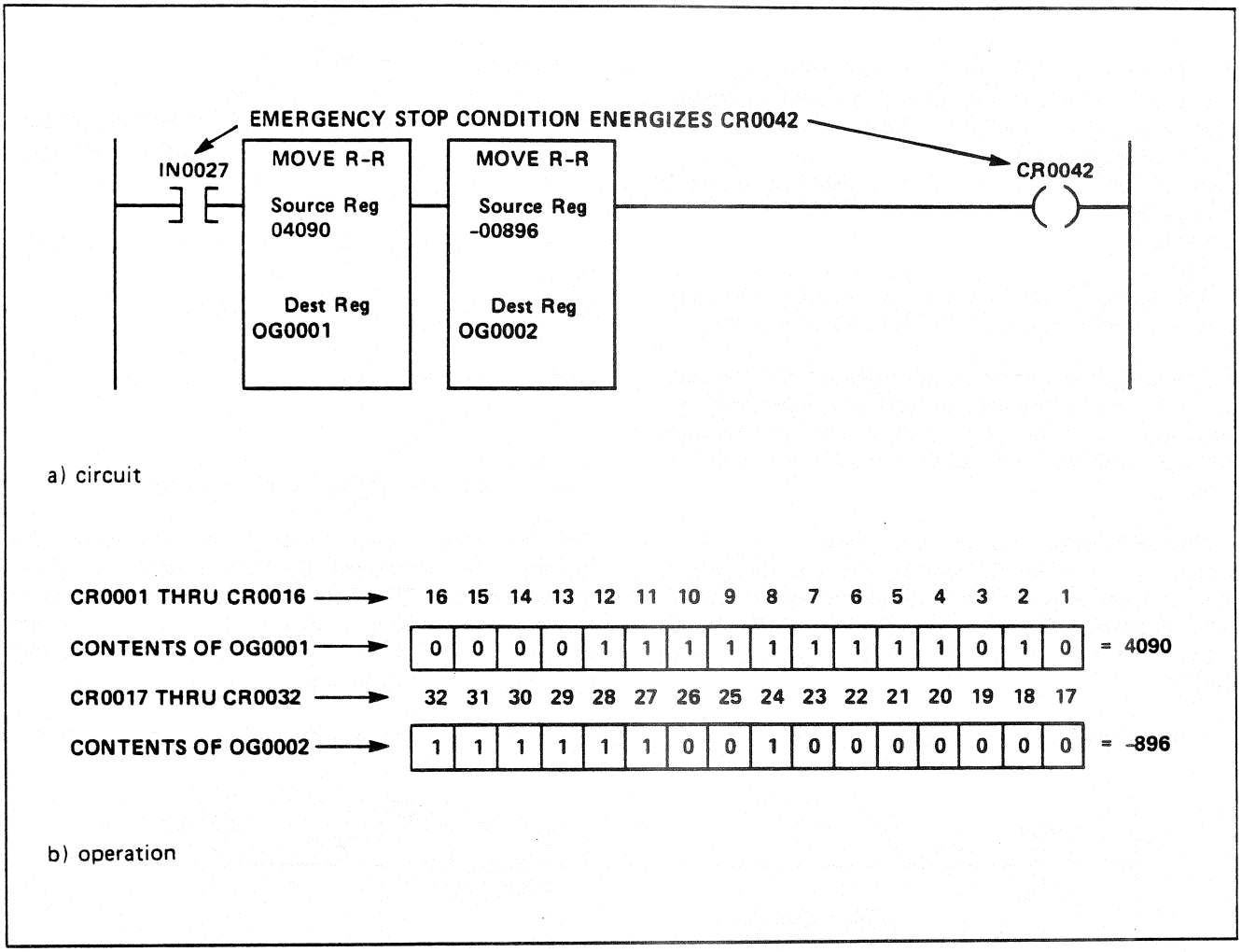


Figure MV-2. Move Register-to-Register Application

7-17. DRUM CONTROLLER (096)

The Drum Controller programmable function allows "stepping," or sequencing, through a group of registers referred to as a Source Table. The various states of each of the 16 bits of each register in the Source Table are made available to a Destination Register, as shown in Figures DM-1 and DM-2. Observe the Figures and note the following 4 points:

1. The Source Table lists the first register to be transferred by the Drum Controller, R00010 in this case.

2. The Length is a constant value representing the number of registers in the Source Table. In this example, 3 is specified as the Length, which means that registers R00010, R00011, and R00012 make up the Source Table.

3. The Destination Register is the register used as the output of the Drum Controller function. (See Figure DM-2.) It contains the contents of the current Source Table Register. The register, specified as OG0001 in this example, contains the contents of the following holding registers:

- R00011 during step 2
- R00012 during step 3

4. The Pointer contains a number representing the current step of the Drum Controller function. When reset, the Pointer is in a 1 state.

Three input circuits control the Drum Controller function. These are:

- Step enable
- Reset circuit
- Enable transfer

Each is discussed in the following subparagraphs.

The step enable circuit, when conducting, causes the Pointer to be incremented by 1 during each scan of the ladder program. The Transitional function contained in the step enable circuit of Figure DM-1 causes the Drum Controller function to step only once when the step enable circuit first conducts.

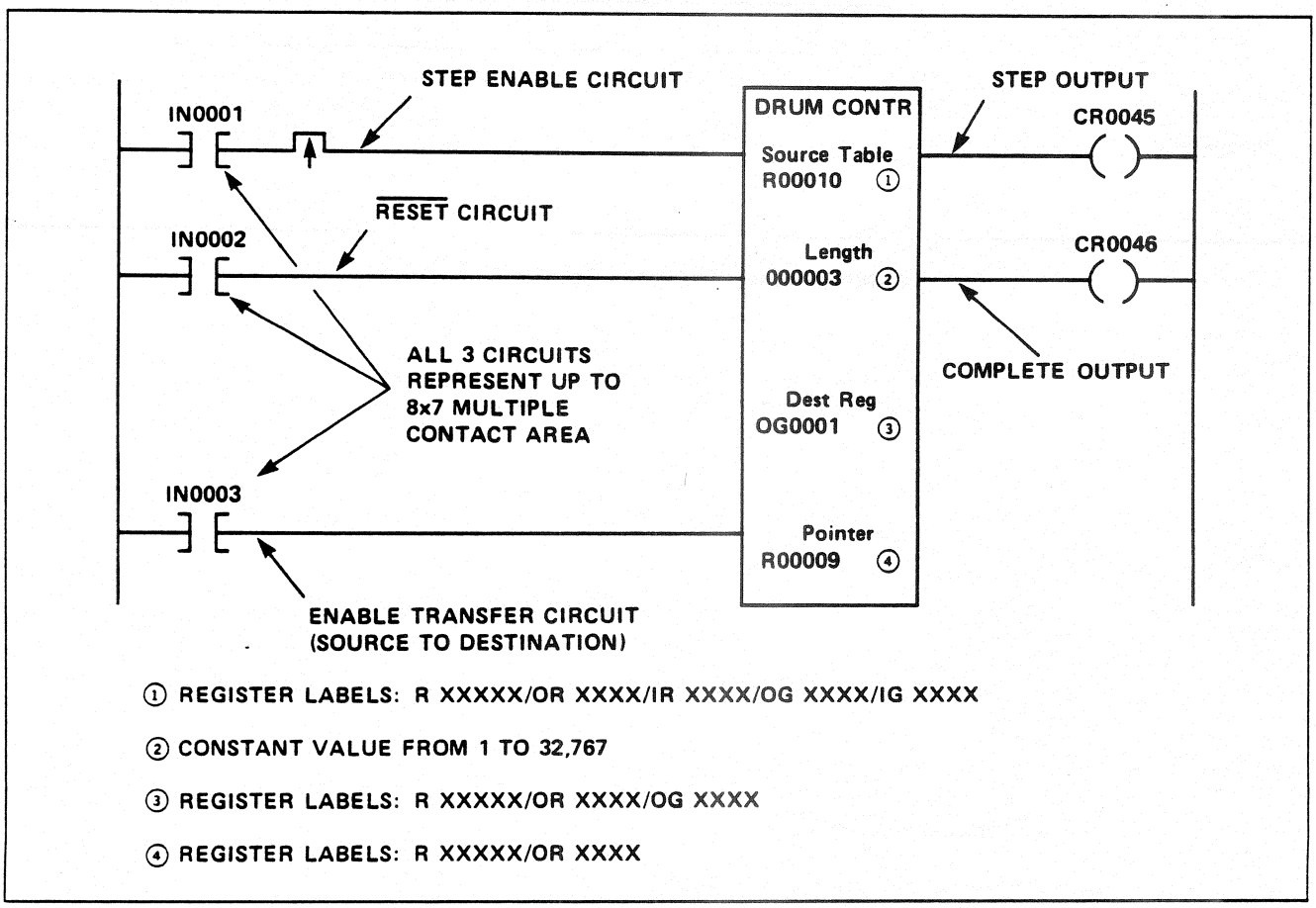


Figure DM-1. Drum Controller Function Circuit (Typical)

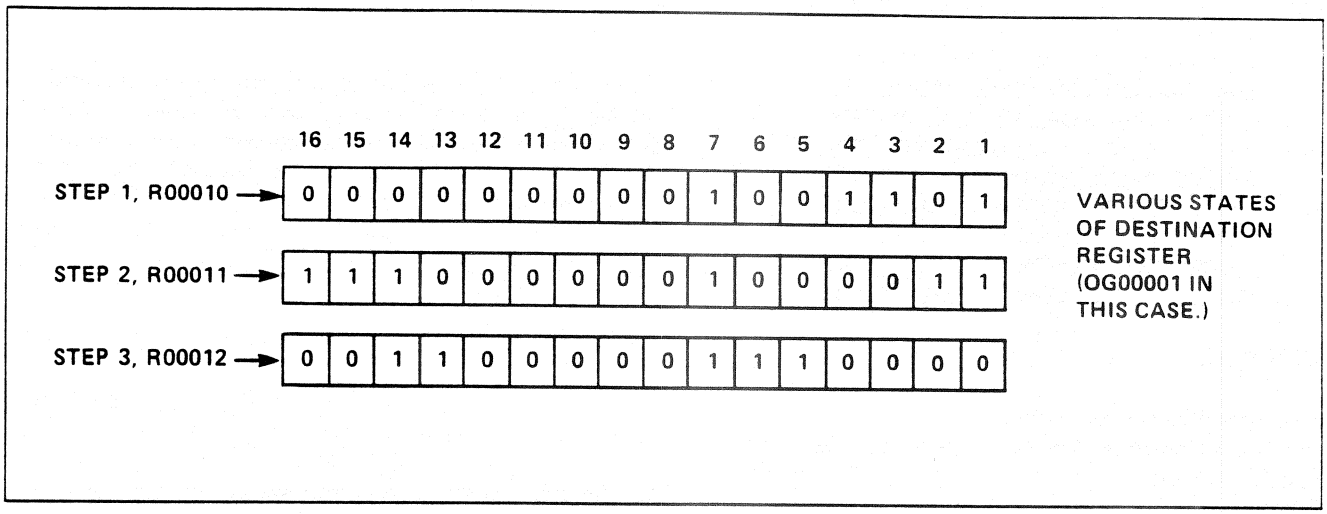


Figure DM-2. Drum Controller Function Operation

The reset circuit resets and holds the Pointer Register to 1 whenever the circuit is nonconducting.

The enable transfer circuit, when conducting, allows data from the Source Table to be transferred to the Destination Register. When the circuit is nonconducting, the last data in the Destination Register is held there.

7-17-1. SPECIFICATIONS

The programming specifications for the Drum Controller function are listed here.

DRUM CONTROLLER FUNCTION BLOCK

The Drum Controller function block requires 2 horizontal contact spaces by 5 parallel paths on the display. The function can be positioned anywhere on the 10x7 contact area of the display.

LENGTH

The Length, or number of steps associated with the Drum Controller function, is a constant value from 1 thru 32,767.

SOURCE TABLE

The Source Table consists of a group of registers used to contain the data which is output to the Destination Register. During each step of the controller's sequence, one register from the Source Table is output to the Destination Register. The Source Table, programmed as part of the Drum Controller function and displayed in the function block, is the first register in the table. The Source Table is selected from any of the following registers or groups:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG or IG precede the display of the Source Table register or group number.

DESTINATION REGISTER

The Destination Register defines the register or group type and number into which the data from the Source Table is transferred. The Destination Register contains 16 unique bits of data from a different register contained in the Source Table during each step of the Drum Controller function. The Destination Register is selected as follows:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR or OG precede the display of the Destination Register or group number.

WARNING

USING THE DESTINATION REGISTER TO CONTROL AN OUTPUT GROUP (OG) OVERRIDES FORCE CONDITIONS IN EFFECT ON THE

OUTPUTS PROGRAMMED. IF, FOR EXAMPLE, CR0006 WERE FORCED OFF PRIOR TO THE MOVE, IT COULD BE FORCED ON AFTER THE MOVE. FAILURE TO OBSERVE THIS WARNING CAN RESULT IN EQUIPMENT DAMAGE AND/OR PERSONAL INJURY FROM UNEXPECTED ENERGIZING OR DE-ENERGIZING OF OUTPUTS.

POINTER REGISTER

The Pointer Register contains the current step number of the Drum Controller function. The Pointer Register is selected as follows:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the display of the Pointer Register number.

INPUT CIRCUITS

Three input, or enabling, circuits control the operation of the Drum Controller function, as listed in Table DM-1. These input circuits can consist of up to 8 horizontal contacts by 7 parallel paths located to the left of the Drum Controller function block on the ladder diagram. The rules for programming contacts listed in Section 6 apply to these circuits.

OUTPUTS

The Drum controller function block has 2 outputs, as listed in the following 2 points:

1. The step output simply follows or repeats the conducting or nonconducting state of the step enable circuit.

2. The complete output conducts when the value of the Pointer Register equals the programmed Source Table's Length.

TABLE DM-1. DRUM CONTROLLER TRUTH TABLE

Input Circuit ①			Pointer Register	Destination Register
Step Enable	$\overline{\text{Reset}}$	Enable Transfer		
1 or 0	0	0	Reset to 1	Last data held.
1 or 0	0	1	Reset to 1	First register of Source Table is moved to Destination Register.
0	1	0	Held at last state	Last data held.
0	1	1	Held at last state	New data transferred from the register in the Source Table as directed by the Pointer Register.
1	1	0	Incremented by 1 ② ③	Last data held.
1	1	1	Incremented by 1 ② ③	New data transferred from the register in the Source Table as directed by the Pointer Register.

① 1 = conducting; 0 = nonconducting.

② Incremented by 1 during each scan of the ladder program. If a Transitional function (007) is programmed in the step enable circuit, the Pointer is incremented when the circuit first conducts.

③ When the value of the Pointer Register equals the length, the next ladder program scan (or transition if the Transition function is contained in the step enable circuit), the Pointer is reset to 1.

Note

The outputs from the Drum Controller function can be left unconnected in the network.

7-17-2. APPLICATIONS

In its simplest form the Drum Controller function simultaneously drives 16 discrete outputs through as many steps as desired. There are 32,767 steps possible, although this many will probably never be utilized. Two variations on the basic Drum Controller function follow:

- Bidirectional Drum Controller (Par. 7-17-2-1)
- 32-bit Drum Controller (Par. 7-17-2-2)

7-17-2-1. BIDIRECTIONAL DRUM CONTROLLER

One method of producing a bidirectional Drum Controller function is shown in Figure DM-3. Observe the Figure and note the following 2 points:

1. The step enable circuit IN0001 controls the normal step function while IN0004 controls the step down function by decrementing holding register R00004. Note this is the same holding register used for the Pointer.

2. If the Pointer is decremented from 1 to 0, the output of the Down Counter function (039) monitors this condition and preloads the Pointer with the highest step number, 5 in this example. In this way the Drum Controller function would step from 5, 4, 3, 2, 1, and back to 5. In effect it "rolls over."

7-17-2-2. 32-BIT DRUM CONTROLLER

If more than 16 bits are needed to be controlled using the same step conditions, etc., 2 Drum Controller functions can be connected together, as shown in Figure DM-3. Observe the Figure and note the following 3 points:

1. Both Drum Controller functions use the same Pointer Register R00019.

2. The step input to the second Drum Controller is connected to a contact which never conducts. (Bit 16 of the Pointer is always positive in this example.)

3. The $\overline{\text{reset}}$ circuit of the second Drum Controller function uses a straight-thru branch since the second function will be reset whenever the first Drum Controller function is reset.

4. The enable transfer circuit, input IN0003 in this example, is the same for both Drum Controller.

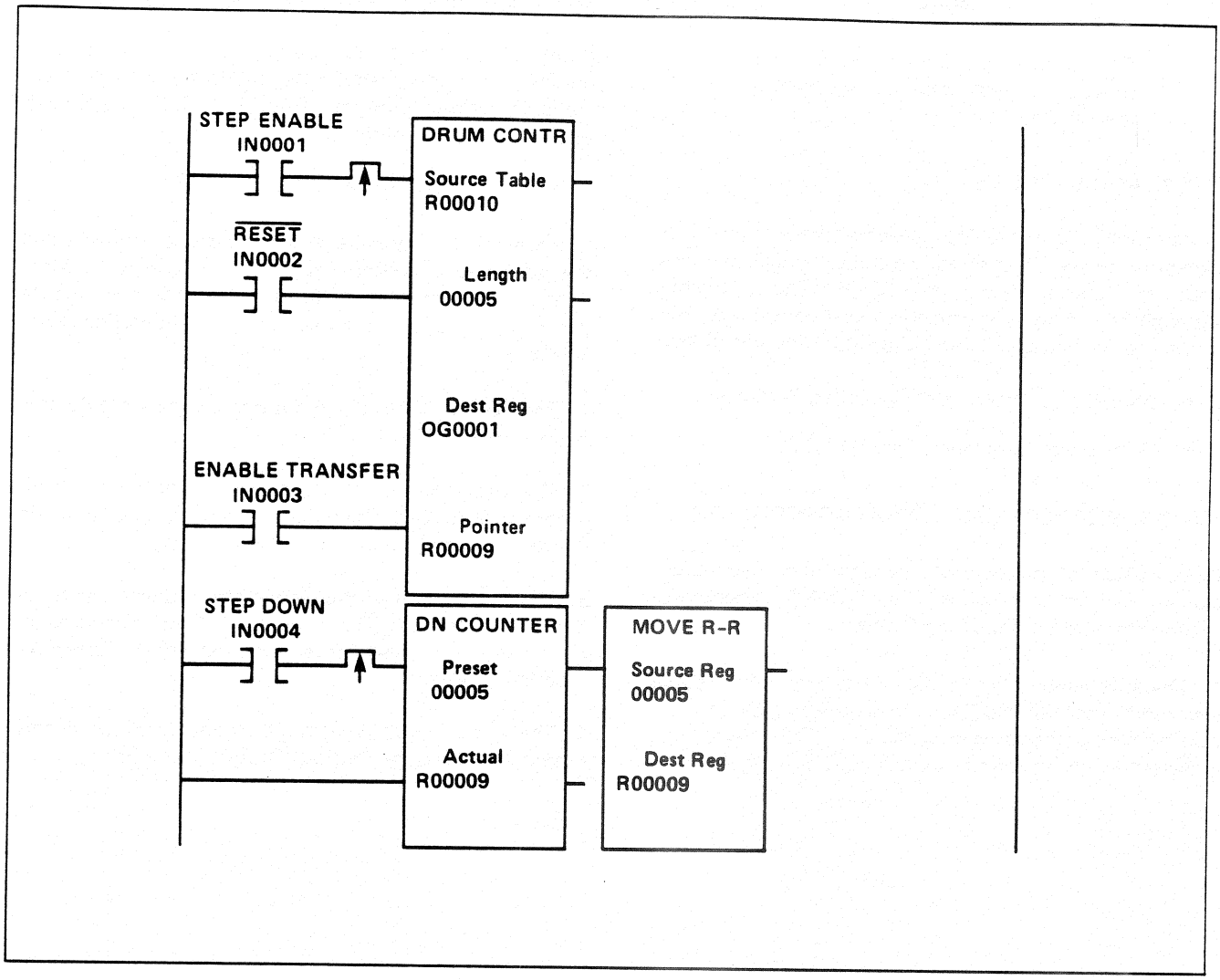


Figure DM-3. Bidirectional Drum Controller Example

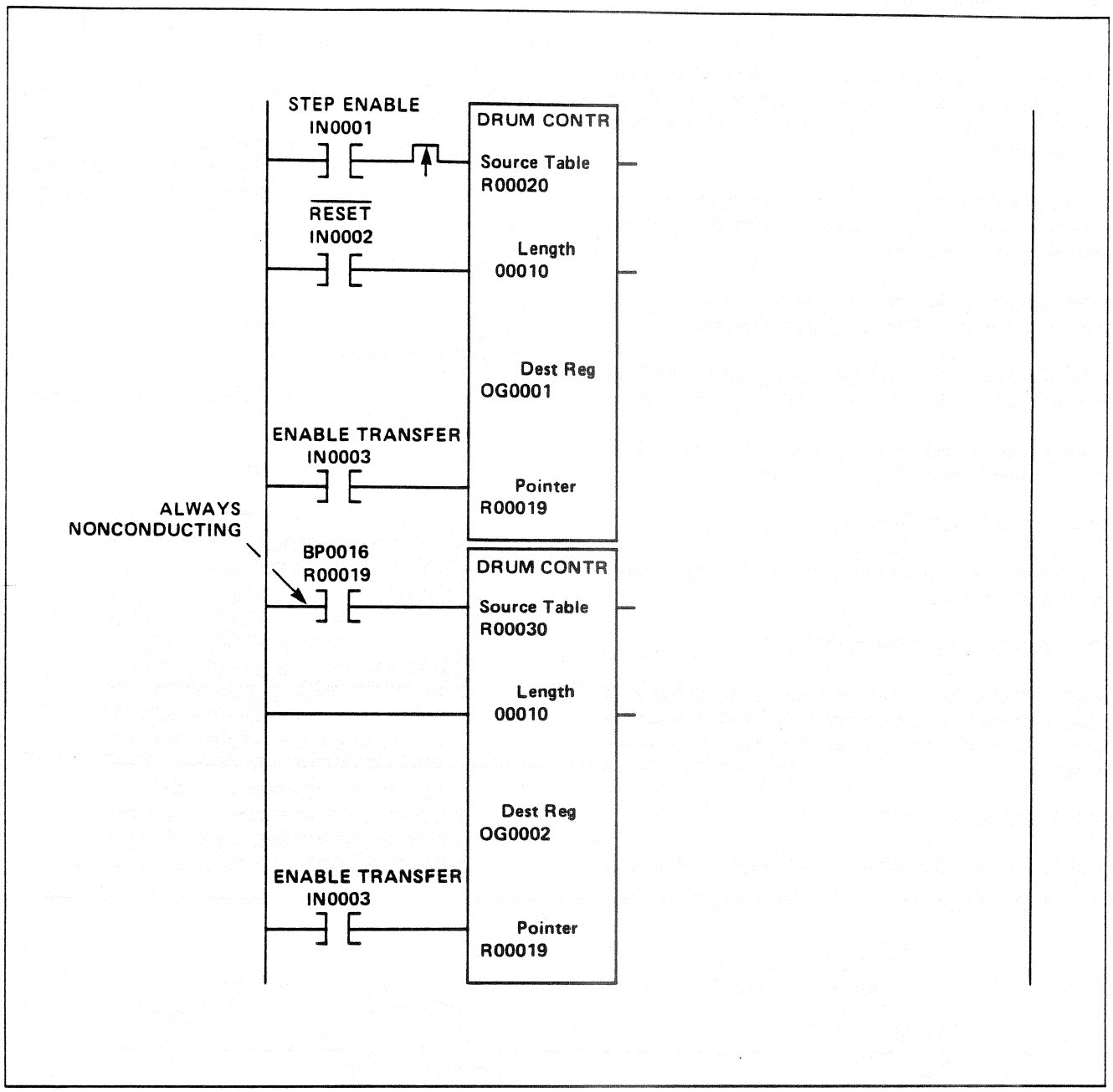


Figure DM-4. 32-Bit Drum Controller Example

7-18. ZERO TABLE (051)

The Zero Table programmable function resets to zero all bits contained in a specified Table, or block, of consecutively numbered registers. See the typical Zero Table function circuit contained in Figure ZT-1 and observe the following 4 points:

1. The Table Register lists the first register in the table to be reset. It can be programmed as a holding register, output register, or output group.

2. The Length consists of a constant value used to define the number of registers (length) of the table.

3. The enable circuit, when conducting, causes the Zero Table function to reset all registers in the table to zero.

4. The enable output simply follows the conducting or nonconducting state of the enable circuit.

7-18-1. SPECIFICATIONS

The programming specifications of the Zero Table function are contained here:

ZERO TABLE FUNCTION BLOCK

The Zero Table function block requires 2 horizontal contact spaces by 3 parallel paths. The function can be located anywhere within the 10x7 contact area of the display.

ENABLE CIRCUIT

The enable circuit of the Zero Table function consists of up to 8 horizontal contacts located in up to 7 parallel paths. The rules for programming contacts listed in Section 6 apply to the enable circuit.

TABLE

The Table, programmed as part of the Zero Table function, defines the first register in the table to be reset. The Table is specified by the programmer as any of the following:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR or OG precede the register or group number on the display.

WARNING

USING THE ZERO TABLE FUNCTION TO CONTROL AN OUTPUT GROUP (OG) OVERRIDES FORCE CONDITIONS IN EFFECT ON THE OUTPUTS PROGRAMMED. IF, FOR EXAMPLE, CR0006 WERE FORCED ON PRIOR TO THE EXECUTION OF THE ZERO TABLE FUNCTION, IT WOULD BE FORCED OFF AFTER THE EXECUTION. FAILURE TO OBSERVE THIS WARNING CAN RESULT IN EQUIPMENT DAMAGE AND/OR PERSONNEL INJURY FROM UNEXPECTED ENERGIZING OR DE-ENERGIZING OF OUTPUTS.

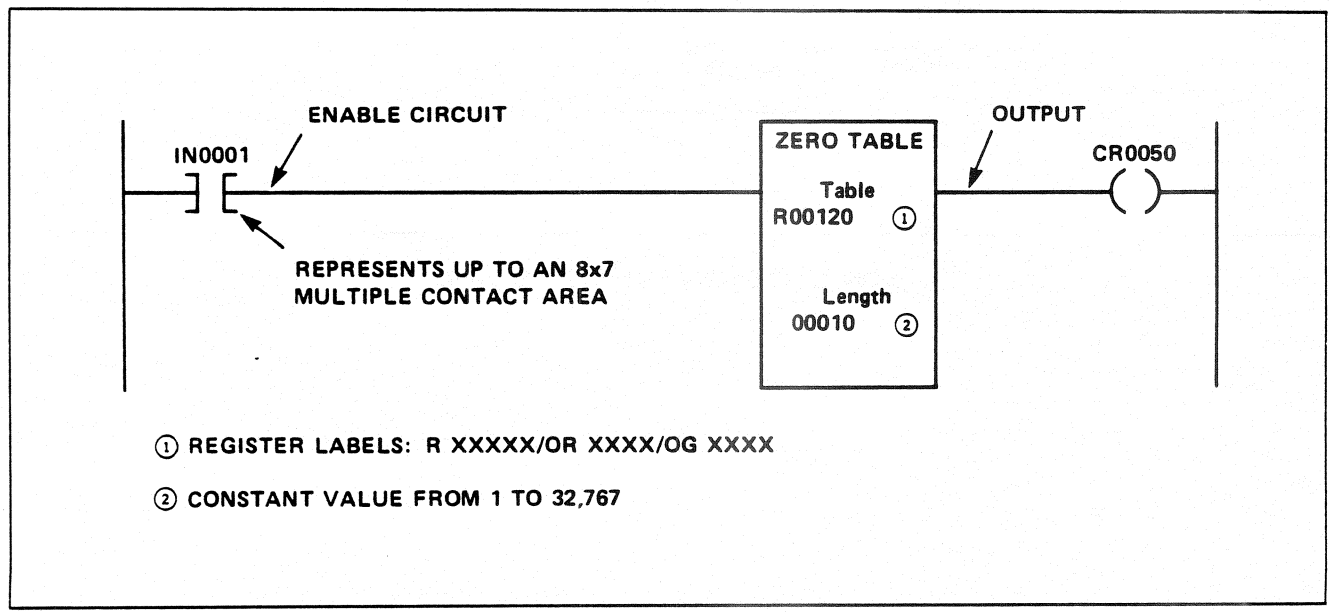


Figure ZT-1. Zero Table Circuit (Typical)

LENGTH

The Length value is a constant programmed in the range of from 1 to 32,767. It defines the number of registers in the Table to be zeroed.

OUTPUT

The output from the Zero Table function block simply follows or repeats the conducting or nonconducting state of the enable circuit.

Note

The output from the Zero Table function can be left unconnected in the network.

7-18-2. APPLICATIONS

The Zero Table function is useful whenever it is desirable to clear to zero a table of registers.

7-19. DIAGNOSTIC (024)

The Diagnostic programmable function searches through the nonfatal fault registers of the processor. If a fault is found, a related bit number, referred to as a specific fault bit, is stored in the Pointer Register, and the found output of the function block conducts. The nonfatal faults are arbitrarily assigned a number by the Diagnostic function. This fault bit number as shown in Table DIAG-1 is stored in the Pointer Register in the event of a fault. See Figure DIAG-1 and observe the following 3 points:

1. When the search enable circuit conducts, the Pointer increments through the nonfatal faults until a fault is detected. If no faults are detected, the Pointer stops at the end of the fault matrix, and the complete output conducts. If a fault is detected, the Pointer contains the fault bit number, and the found output conducts.

2. If a fault is detected, the fault bit number is held in the Pointer Register until the next scan. During each scan, the next higher fault is found and held. The search stops at the end of the fault bits, and a zero is contained in the Pointer Register.

7-19-1. SPECIFICATIONS

The specifications for the Diagnostic function are listed here.

DIAGNOSTIC FUNCTION BLOCK

The Diagnostic function block requires 2 horizontal contact spaces by 4 parallel paths on the display. It can be located anywhere in the 10x7 contact area.

SEARCH ENABLE CIRCUIT

The search enable and $\overline{\text{reset}}$ circuits conduct, initiating the search for a nonfatal fault. (See Table DIAG-1.) During each scan that the search enable circuit conducts, the nonfatal faults are searched until either a fault is detected or no faults are detected. If a fault is detected, the Pointer Register will hold the fault bit number for 1 scan.

POINTER REGISTER

The Pointer Register holds the fault bit number of the last nonfatal fault detected. If no faults are detected during the last scan, the Pointer Register contains a zero. This Register is specified by the programmer as either a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the ladder diagram display.

$\overline{\text{RESET}}$ CIRCUIT

The $\overline{\text{reset}}$ circuit conducts, allowing normal operation of the Diagnostic function. When the $\overline{\text{reset}}$ circuit is nonconducting, the Pointer is held at 0, and the Diagnostic function is disabled.

FOUND OUTPUT

The found output conducts when:

- The search enable circuit conducts, and

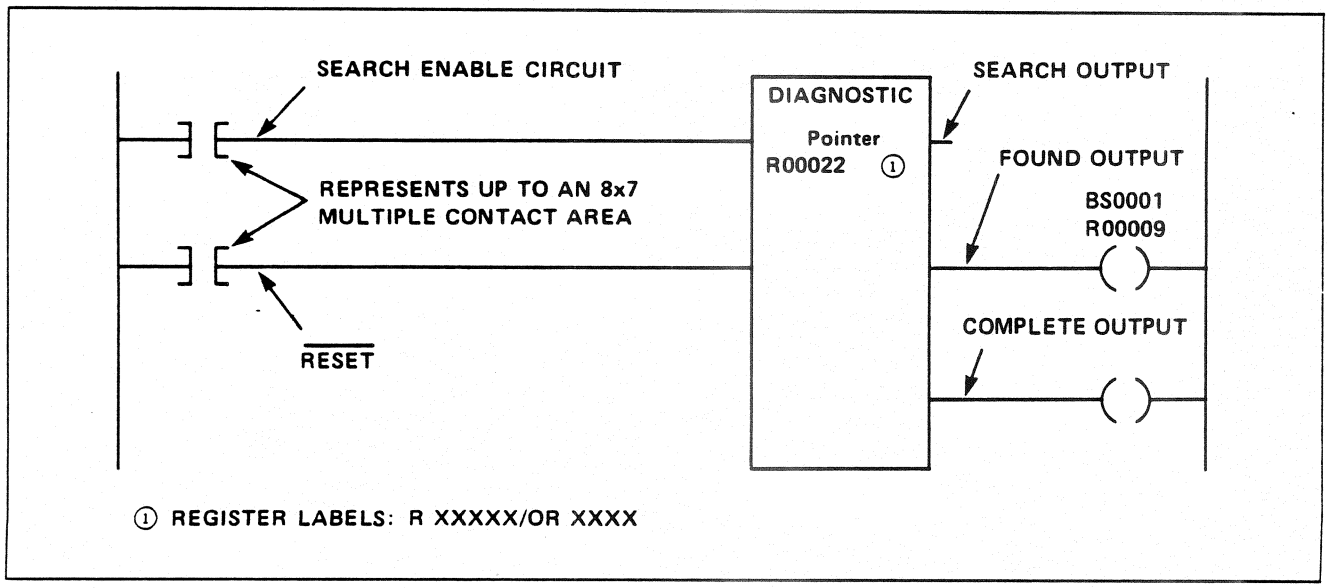


Figure DIAG-1. Diagnostic Function Circuit (Typical)

- A nonfatal fault was detected during the last scan of the Diagnostic function.

COMPLETE OUTPUT

The complete output conducts when the search enable and reset circuits are conducting, and the nonfatal fault registers do not contain any faults.

SEARCH OUTPUT

The search output follows or repeats the conducting or nonconducting state of the search enable circuit.

7-19-2. APPLICATION

The Diagnostic function can be used to activate or illuminate an external alarm, annunciator, etc. when a nonfatal fault occurs. In Figure DIAG-1, for example, the found output energizes a BS/R coil (001) when a nonfatal fault occurs. This coil could be wired to an alarm to signal personnel. Also, the contacts from this coil in this example can also be used in the ladder diagram to initiate special measures such as energizing or de-energizing outputs, or resetting timers or counters. A reset BC/R coil (002; not shown) would be necessary to clear the bit after corrective action has been taken.

TABLE DIAG-1. NONFATAL FAULT BIT NUMBERS

Fault Bit Number	Fault Description	Fault Bit Number	Fault Description	Fault Bit Number	Fault Description
33 thru 45	Smart Modules, Redundancy				
46	Parity				
47	Battery Low				
48	FAP Checksum				
49 thru 64	Reserved				
65	IOP1 SIM 1	81	IOP1 Link A SIM 1	97	IOP1 Link B SIM 1
66	IOP1 SIM 2	82	IOP1 Link A SIM 2	98	IOP1 Link B SIM 2
67	IOP1 SIM 3	83	IOP1 Link A SIM 3	99	IOP1 Link B SIM 3
68	IOP1 SIM 4	84	IOP1 Link A SIM 4	100	IOP1 Link B SIM 4
69	IOP1 SIM 5	85	IOP1 Link A SIM 5	101	IOP1 Link B SIM 5
70	IOP1 SIM 6	86	IOP1 Link A SIM 6	102	IOP1 Link B SIM 6
71	IOP1 SIM 7	87	IOP1 Link A SIM 7	103	IOP1 Link B SIM 7
72	IOP1 SIM 8	88	IOP1 Link A SIM 8	104	IOP1 Link B SIM 8
73	IOP1 SIM 9	89	IOP1 Link A SIM 9	105	IOP1 Link B SIM 9
74	IOP1 SIM 10	90	IOP1 Link A SIM 10	106	IOP1 Link B SIM 10
75	IOP1 SIM 11	91	IOP1 Link A SIM 11	107	IOP1 Link B SIM 11
76	IOP1 SIM 12	92	IOP1 Link A SIM 12	108	IOP1 Link B SIM 12
77	IOP1 SIM 13	93	IOP1 Link A SIM 13	109	IOP1 Link B SIM 13
78	IOP1 SIM 14	94	IOP1 Link A SIM 14	110	IOP1 Link B SIM 14
79	IOP1 SIM 15	95	IOP1 Link A SIM 15	111	IOP1 Link B SIM 15
80	IOP1 SIM 16	96	IOP1 Link A SIM 16	112	IOP1 Link B SIM 16
113	IOP2 SIM 1	129	IOP2 Link A SIM 1	145	IOP2 Link B SIM 1
114	IOP2 SIM 2	130	IOP2 Link A SIM 2	146	IOP2 Link B SIM 2
115	IOP2 SIM 3	131	IOP2 Link A SIM 3	147	IOP2 Link B SIM 3
116	IOP2 SIM 4	132	IOP2 Link A SIM 4	148	IOP2 Link B SIM 4
117	IOP2 SIM 5	133	IOP2 Link A SIM 5	149	IOP2 Link B SIM 5
118	IOP2 SIM 6	134	IOP2 Link A SIM 6	150	IOP2 Link B SIM 6
119	IOP2 SIM 7	135	IOP2 Link A SIM 7	151	IOP2 Link B SIM 7
120	IOP2 SIM 8	136	IOP2 Link A SIM 8	152	IOP2 Link B SIM 8
121	IOP2 SIM 9	137	IOP2 Link A SIM 9	153	IOP2 Link B SIM 9
122	IOP2 SIM 10	138	IOP2 Link A SIM 10	154	IOP2 Link B SIM 10
123	IOP2 SIM 11	139	IOP2 Link A SIM 11	155	IOP2 Link B SIM 11
124	IOP2 SIM 12	140	IOP2 Link A SIM 12	156	IOP2 Link B SIM 12
125	IOP2 SIM 13	141	IOP2 Link A SIM 13	157	IOP2 Link B SIM 13
126	IOP2 SIM 14	142	IOP2 Link A SIM 14	158	IOP2 Link B SIM 14
127	IOP2 SIM 15	143	IOP2 Link A SIM 15	159	IOP2 Link B SIM 15
128	IOP2 SIM 16	144	IOP2 Link A SIM 16	160	IOP2 Link B SIM 16

7-20. COMPARE R-R (053)

The Compare Register-to-Register programmable function compares the values of 2 registers, or groups, and, depending on the results of the comparisons, enables or disables outputs. Refer to the typical circuit shown in Figure CRR-1 and note the following 3 points:

1. The comparison is active only when the compare enable circuit is conducting.

2. Each of the 3 outputs are conducting when the values contained in registers 1 and 2 are as follows:

- Register 1 = register 2
- Register 1 > register 2
- Register 1 < register 2

3. All 3 outputs are nonconducting when the complete enable circuit is nonconducting.

7-20-1. SPECIFICATIONS

The programming specifications for the Compare Register-to-Register function are listed here.

COMPARE R-R FUNCTION BLOCK

The Compare Register-to-Register function block requires 2 horizontal contact spaces by 4 parallel paths on the ladder diagram. The function block can be located anywhere in the contact area of the ladder diagram.

REGISTER 1, 2

Register 1 and Register 2 can be programmed as a constant value in the range of +32,767 to -32,768. Negative values must be in 2's complement form in order for the comparison to be valid. Register 1 and Register 2 also may be specified by the programmer as any of the following registers or groups:

- Holding registers (R)
- Output registers (OR)
- Input registers (IR)
- Output groups (OG)
- Input groups (IG)

The letters R, OR, IR, OG or IG precede the register or group numbers on the display.

COMPARE ENABLE CIRCUIT

When the compare enable circuit conducts, the Compare Register-to-Register function compares the contents of Register 1 and Register 2 and enables the appropriate output. When it is nonconducting, all outputs from the Compare Register-to-Register function are nonconducting. The enable circuit consists of up to 8 horizontal contact spaces by 7 parallel paths located to the left of the function block on the ladder diagram.

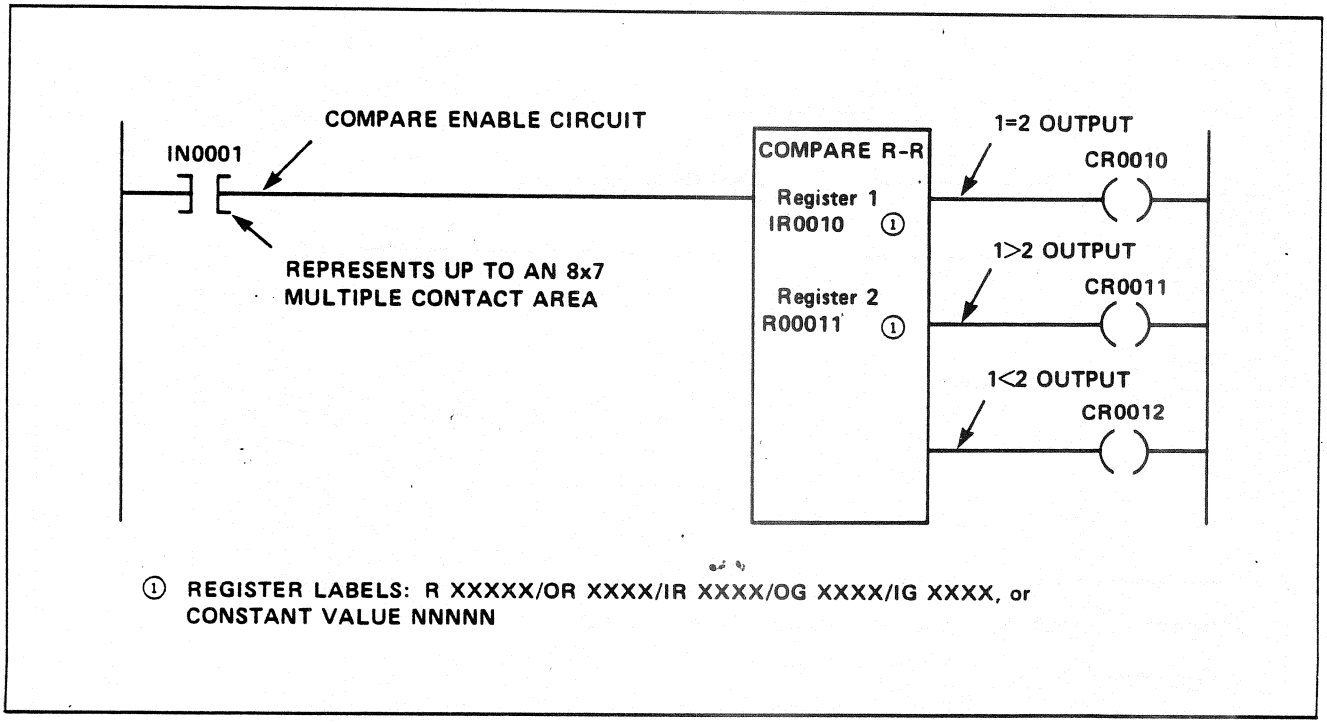


Figure CRR-1. Compare Register-to-Register Circuit (Typical)

OUTPUTS

When the compare enable circuit conducts, the three outputs from the Compare Register-to-Register function block conduct as follows:

- Register 1 = Register 2; top output conducts
- Register 1 is greater than Register 2; middle output conducts
- Register 1 is less than Register 2; bottom output conducts

Note

The outputs from the Compare Register-to-Register function can be left unconnected in the network.

7-20-2. APPLICATIONS

The Compare Register-to-Register function is useful whenever values contained in registers need to be compared. An example of the comparison is shown in Figure CRR-2, where the input from an Analog Input Module is compared to a high and low limit value. Observe the Figure and note the following 4 points:

1. The Analog Input Module is addressed as IR0020. The Analog In function (048) converts the sign-magnitude data from the Analog Input Module to binary data. The converted data is contained in register R00150.

2. Constants 240 and 2500 are the low and high limits, respectively. The low limit energizes CR0050 and the high limit energizes CR0051.

3. Output coil CR0050 is enabled if the value of the Analog Input Module falls below 240. Output CR0051 is enabled if the value of the Analog Input Module is larger than or equal to 2500.

4. The limit values contained in this example were constants. These values can be obtained from a holding or input register which is selectable.

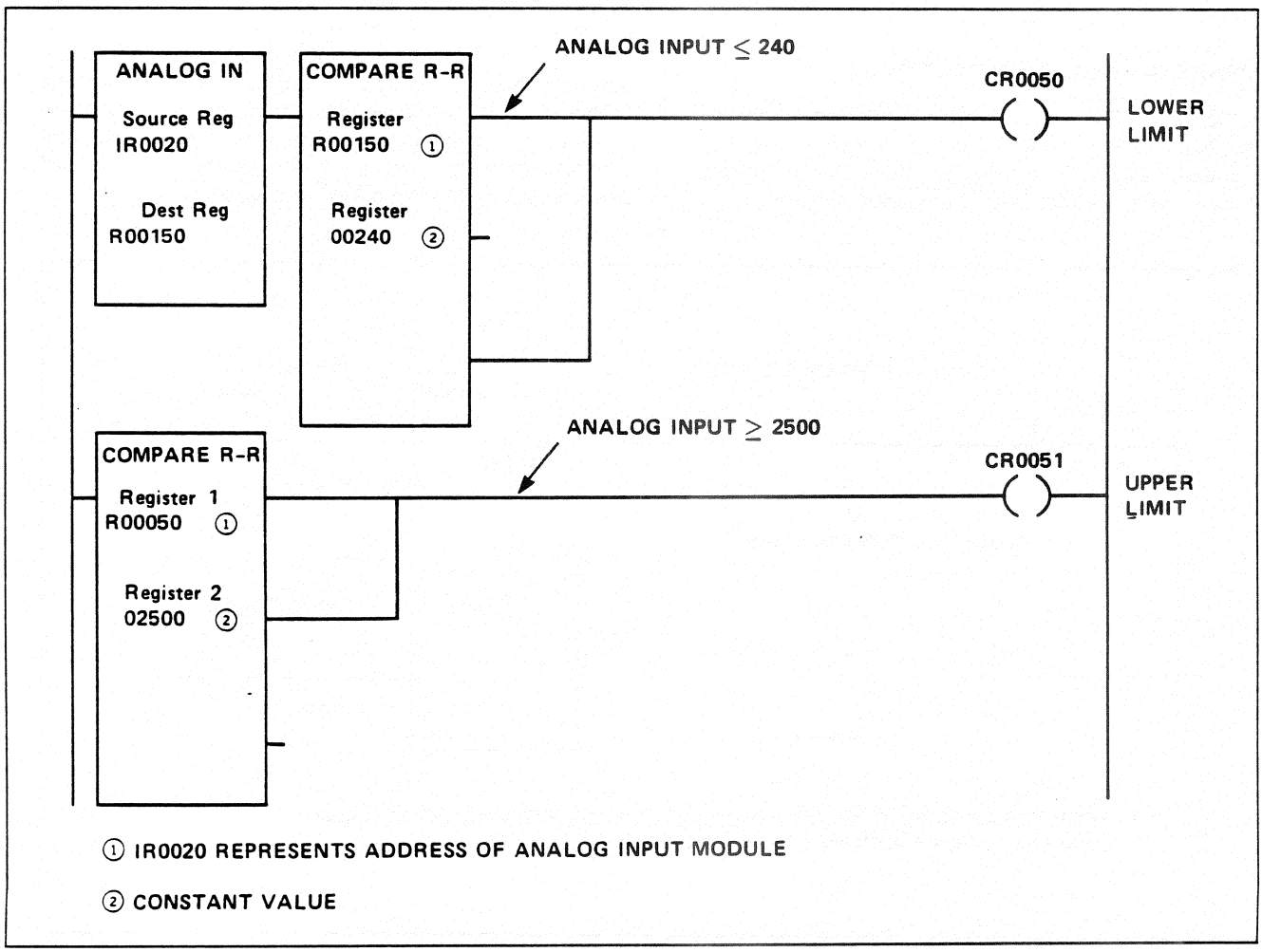


Figure CRR-2. Compare Register-to-Register Limit Application Example

7-21. UPDATE I/O (054)

The Update I/O programmable function performs an "I/O service routine" on a specified group or block of I/O registers or groups. This routine stops the normal processor scanning of the ladder diagram and immediately "performs an update," that is:

- The current status (0 or 1) of the specified inputs are read, or
- The outputs specified are energized or de-energized according to their current ladder program status.

Figure UIO-1 shows a typical circuit containing the Update I/O function. Observe the Figure and note the following 2 points:

1. The I/O Start defines the first register or group of inputs or outputs to be updated.

2. When the enable update circuit conducts, the specified inputs or outputs are updated.

7-21-1. SPECIFICATIONS

The programming specifications for the Update I/O function are listed here.

UPDATE I/O FUNCTION BLOCK

The Update I/O function block requires 2 horizontal contact spaces by 3 parallel paths on the ladder diagram. The function can be located anywhere in the contact area of the ladder diagram.

CAUTION

The Update I/O function should be used sparingly. Each I/O update requires approximately 2.5 msec. plus 0.2 msec. times the number of registers updated. This time is added to the ladder diagram scan time. Since a ladder diagram scan of 25K words requires approximately 20 msec., it can be seen that very few I/O updates are needed to significantly increase the overall scan time.

I/O START

The I/O Start is the first register or group in the block to be updated. It is specified by the programmer as any of the following:

- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters OR, IR, OG or IG precede the number of the register or group on the display.

LENGTH

The Length is a constant, specified by the programmer. It defines the number of consecutively numbered registers or groups to be updated. The value of the Length must be within the I/O groups and register numbers

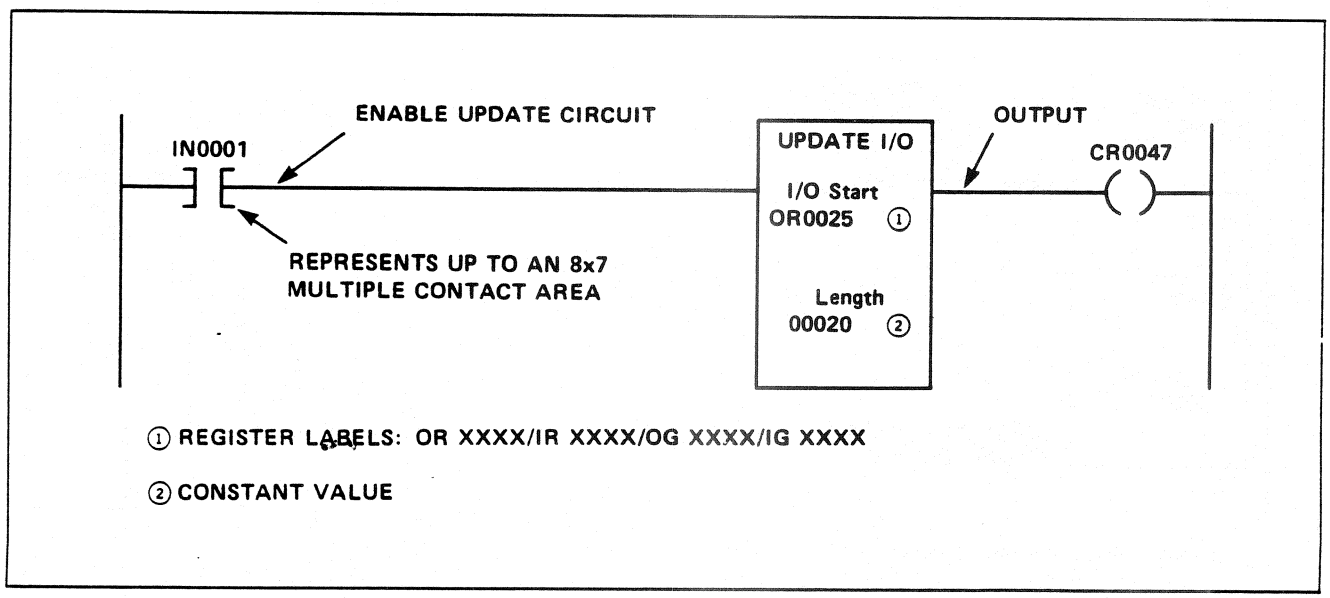


Figure UIO-1. Update I/O Circuit (Typical)

configured for the system.

Note

All of the I/O numbers referenced by the I/O Start and Length values must be contained within 1 SIM to prevent a processor fault occurring when the function is entered into the memory.

ENABLE UPDATE CIRCUIT

When the enable update circuit conducts, the Update I/O function occurs. The contact area for the enable update circuit consists of up to 8 horizontal contact spaces located in 7 parallel paths located to the left of the function block. This assumes the function is positioned on the far right portion of the display. The rules for programming contacts listed in Section 6 apply to the contacts of the enable update circuit.

OUTPUT

The output of the Update I/O function simply follows or repeats the conducting or nonconducting state of the enable update circuit.

Note

The output of the Update I/O function can be left unconnected in the network.

7-21-2. APPLICATIONS

Before actually programming the Update I/O function, 2 considerations must be understood and kept in mind. These are:

- Program scan time (7-21-2-1)
- SIM Allocation Overlaps

7-21-2-1. PROGRAM SCAN TIME

The Update I/O function is used to service the inputs and outputs at a more rapid rate. Figure UIO-2 shows a typical application where the states of one or more inputs must be examined and an output turned on or off accordingly at a worst-case rate of 25 milliseconds. Observe the Figure and note the following points:

1. During operation without the Update I/O function, if the ladder diagram scan time is 25 milliseconds, the worst-case time from when input IN0001 or IN0002 changes state and CR0010 changes state would be 75 milliseconds as described here:

- Monitor input status change 25 msec.
- Change ladder state of output 25 msec.

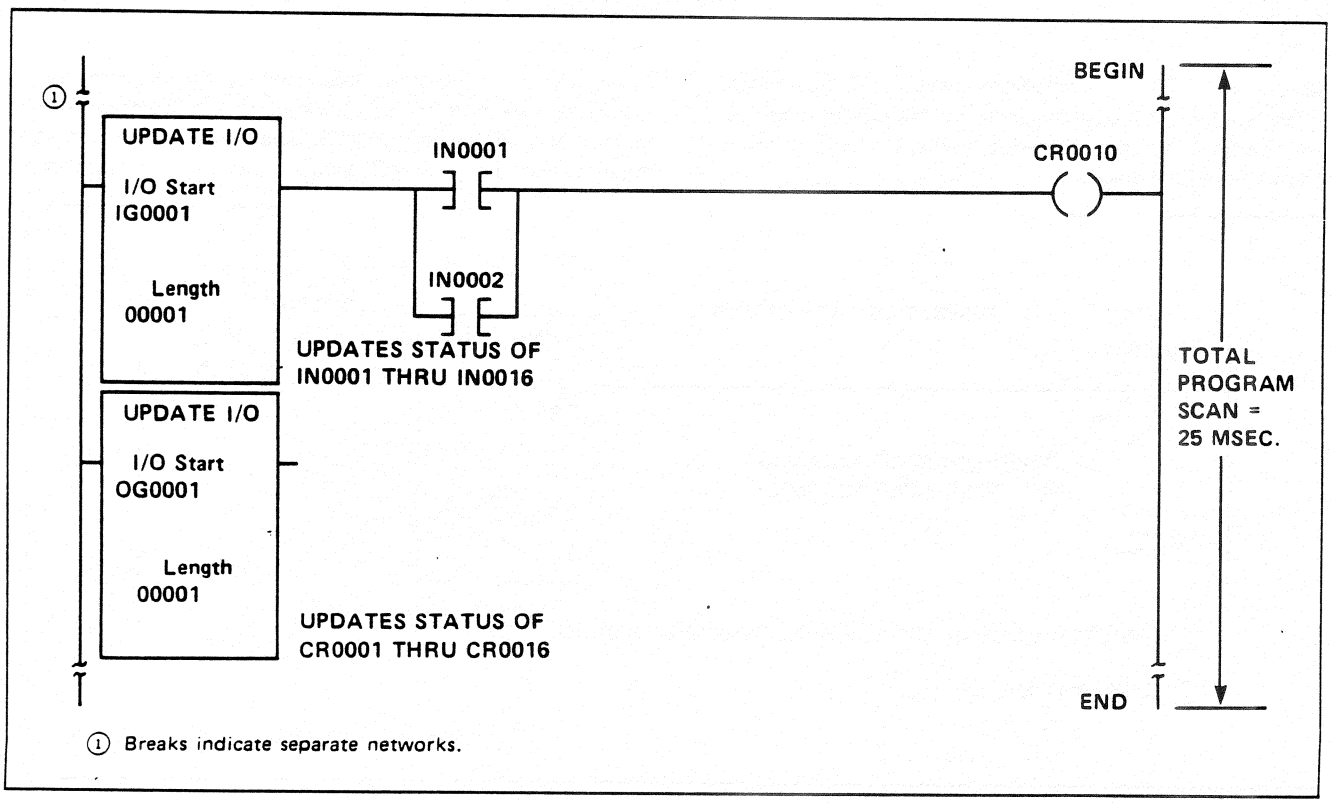


Figure UIO-2. Update I/O Example

- Change output status at I/O Module 25 msec.

2. If the ladder scan time is 25 milliseconds long, and if the Update I/O functions are programmed as shown, output CR0010 is updated every 25 msec. (This assumes the normal I/O update time of the SIM is 25 msec., or less. See Section 10, Programming Considerations, for details.)

7-21-2.2. SIM ALLOCATION OVERLAPS

When planning the use of the Update I/O function, care must be taken during the allocation of specific registers or IG/OG groups with specific SIM configurations, as detailed here. (Refer to the HPPC-1500/-1700 APL Programming Manual, NLAM-8806, Paragraph 7-8-4 concerning this allocation process.)

As noted in Paragraph 7-21-1, above, the function block's I/O Start and Length values must be configured within a single SIM.

Beyond this, there is a further restraint which must be observed when two different SIMs are allocated "overlapping" register or IG/OG group numbers.

As background for understanding the overlapping problem, recall that each SIM is capable of handling a maximum of 32 input and/or output groups along with a maximum of 16 input groups and/or output groups.

Overlapping of these types of numbers between 2 SIMs could occur when **fewer** than the maximum number of registers or IG/OG groups are used for one SIM, and the remaining for a second SIM. For example:

- SIM No. 1 is allocated IG 001 thru 006, and SIM No. 2 is allocated IG 007 thru 016. (See Figure 7-12 in the APL Programming Manual.)

- SIM No. 1 is also allocated input registers 001 thru 014, and SIM No. 2 is allocated input registers 015 thru 030.

Overlapping, then, is the crossing over of available registers and/or IG/OG groups from one SIM to another. This practice is acceptable when programming all other standard or advanced programmable functions. It is **not** always acceptable, however, when the Update I/O function is used.

If overlapping were used with the registers and/or I/O groups used with the Update I/O function, at least 2 results would occur:

- SIM No. 1, which is capable of reading the maximum number of registers and IG/OG groups, would receive the update request for **all** possible registers and/or groups, not just the allocated ones.

- SIM No. 2 would receive no updating to or from these overlapped registers and/or groups allocated to it.

The Advanced Program Loader assigns each Update I/O function to an individual SIM during the initial entry of a network. At this time the comparisons, shown in Figure UIO-3 are performed. When this comparison is applied to overlapping SIM addresses, the Update I/O function will not be assigned to the intended SIM.

By way of example, IG 001 thru 006 are allocated to SIM No. 1. IG 007 thru 016 are allocated to SIM No. 2. Perform the comparisons listed in Figure UIO-3. When an Update I/O function referencing in IG 007 thru 016 is programmed, the specific Update I/O function is automatically assigned to SIM No. 1, as indicated in the Figure.

The most straightforward solution to the situation is to program the Update I/O function within registers and groups that do not overlap.

However, it may be absolutely necessary to configure with one SIM allocated IG 001 thru 006; a second SIM allocated IG 007 thru 016; and the Update I/O function reading data from within IG 007 thru 016. To solve the problem, allocate IG 007 thru 016 to SIM No. 1, and IG 001 thru 106 to SIM No. 2. Since the first SIM "test" meets the requirements of the Update I/O function, as shown in the Figure, the function is assigned to the intended SIM.

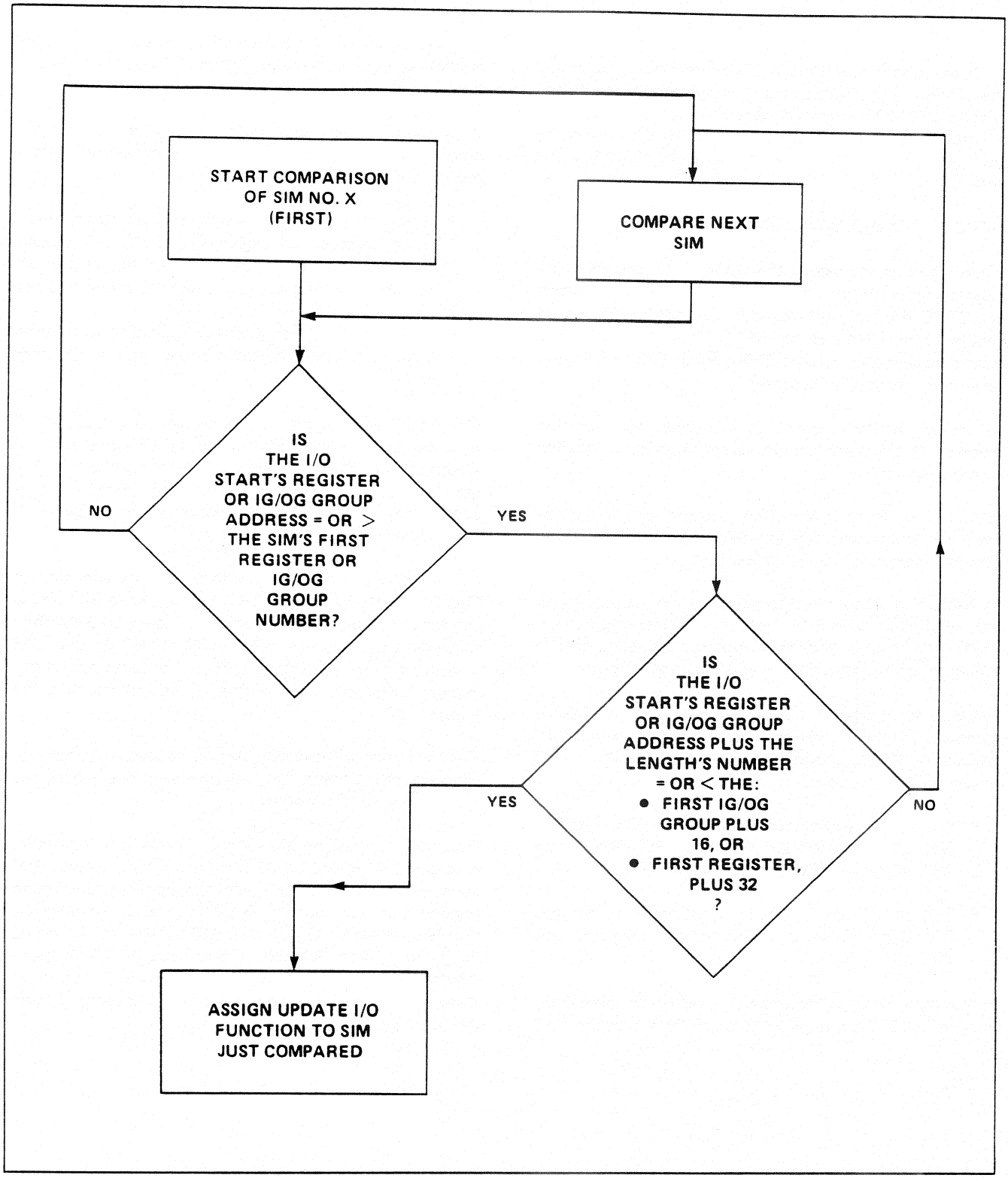


Figure UIO-3. Update I/O-to-SIM Assignment

Section 8

Advanced Functions

8-1. INTRODUCTION

The HPPC-1500/-1700 programmable controller optionally offers the user approximately 43 advanced software functions which extend the range of control that can be built into the ladder diagram application program. It is the purpose of this Section to explain the functions that make up the "advanced" programmable functions. (Section 7 explains the "standard" programmable functions.)

Before reading further, be sure to review Paragraphs 7-1-1 thru 7-1-6. These explain important concepts necessary for an understanding of the discussions which follow in this Section.

Both the standard and advanced functions reside on the Control Store Board (NLCS-1510 or -1520, respectively). This Board is one of 3 that make up the Logic Processor Module Assembly. By changing the Board, it is possible to update an HPPC system from standard to advanced programmable capability.

Table 8-1 contains a list of the advanced functions, along with the display screen 3-digit number associated with each. The Table may be used as a quick locator for the contents of the Section.

8

TABLE 8-1. ADVANCED FUNCTIONS

Type	APL No.	Designation	Par. No.	Page No.
Coils	006	Jump Coil	8-2	8-3
Signal Conditioners	008	Inverting	8-3	8-5
Labels	023	Jump Label	8-2	8-3
Timers	031	Time Off 1.0	8-4	8-6
	032	Time Off 0.1	8-4	8-6
	033	Time Off .01	8-4	8-6
Math Functions	042	Square Root	8-5	8-8
	043	Negate	8-6	8-11
	044	Negate *	8-6	8-11
	069	Add * A + B=C	8-7	8-13
	071	Sub * A - B=C	8-7	8-13
	093	Average	8-8	8-17
Shift Registers	089	Shift Left N	8-9	8-18
	090	Shift Right N	8-9	8-19
Conversions	074	BIN-BCD T	8-10	8-25
	075	BCD-BIN T	8-11	8-27
	118	Scale	8-12	8-29
Move Functions	076	Block Move	8-13	8-32
	077	Byte Move	8-14	8-36
	094	Move R-T	8-15	8-40
	095	Move T-R	8-16	8-43
	119	N-Bit Move	8-17	8-47

(Cont'd.)

TABLE 8-1. ADVANCED FUNCTIONS (Cont'd.)

Type	APL No.	Designation	Par. No.	Page No.
Logic Functions	078	Complement	8-18	8-52
	079	Search 1's	8-19	8-55
	097	AND R-T	8-20	8-59
	098	OR R-T	8-20	8-59
	099	XOR R-T	8-20	8-59
	100	AND T-T	8-21	8-62
	101	OR T-T	8-21	8-62
	102	XOR T-T	8-21	8-62
Table Operations	081	Bit Operate	8-22	8-65
	106	Search =	8-23	8-69
	107	Search ⟩ =	8-23	8-69
	108	Ascend Sort	8-24	8-73
Stack Operations	103	First In	8-25	8-79
	104	First Out	8-25	8-79
	105	Last Out	8-25	8-80
Scan Alteration	052	Lock Scan	8-26	8-89
	128	UDSF	8-27	8-92
Comparison	067	Limit Test R	8-28	8-99
	091	Compare T-T	8-29	8-101
	092	Compare R-T	8-30	8-105
	113	Limit Test T	8-31	8-109

Note

For instructional purposes, certain aspects of the Advanced Program Loader displays have been exaggerated in the typical examples which follow. This is especially true with respect to the spacing between groups of data and some circuits. The content of the displays, however, reflects the actual CRT messages.

8-2. JUMP COIL (006), JUMP LABEL (023)

The Jump Coil and Jump Label programmable functions operate as a pair to modify the normal sequential-type execution of the ladder diagram. When the circuit associated with the Jump Coil conducts, the execution is altered. The next function to be executed is the Jump Label function, which sequentially occurs **before** the Jump Coil enable circuit. (See Figure JMP-1.) The Jump Coil, when conducting, initiates a "jump back" to the Jump Label function programmed earlier in the ladder diagram. The "reference number" here is a single number, from 1 to 32, assigned to **both** the Jump Label and Jump Coil. Each Jump Label and Jump Coil "pair" should be assigned a reference number different from any other pair.

8-2-1. SPECIFICATIONS

The programming specifications for the Jump Label and Jump Coil functions are listed here.

JUMP LABEL

The Jump Label function (023) should be programmed in the ladder diagram sequence **before** the associated Jump Coil. The Jump Label function must be located by itself in the upper left portion of the network.

JUMP COIL

The Jump Coil function (006) is programmed in the coil position of the network. The associated enable circuit consists of up to 10 horizontal by 7 vertical paths of contacts.

Note

There are no contacts associated with the Jump Label. The reference number only references the Jump Coil. Also, the coil should be programmed in a network with no other coils.

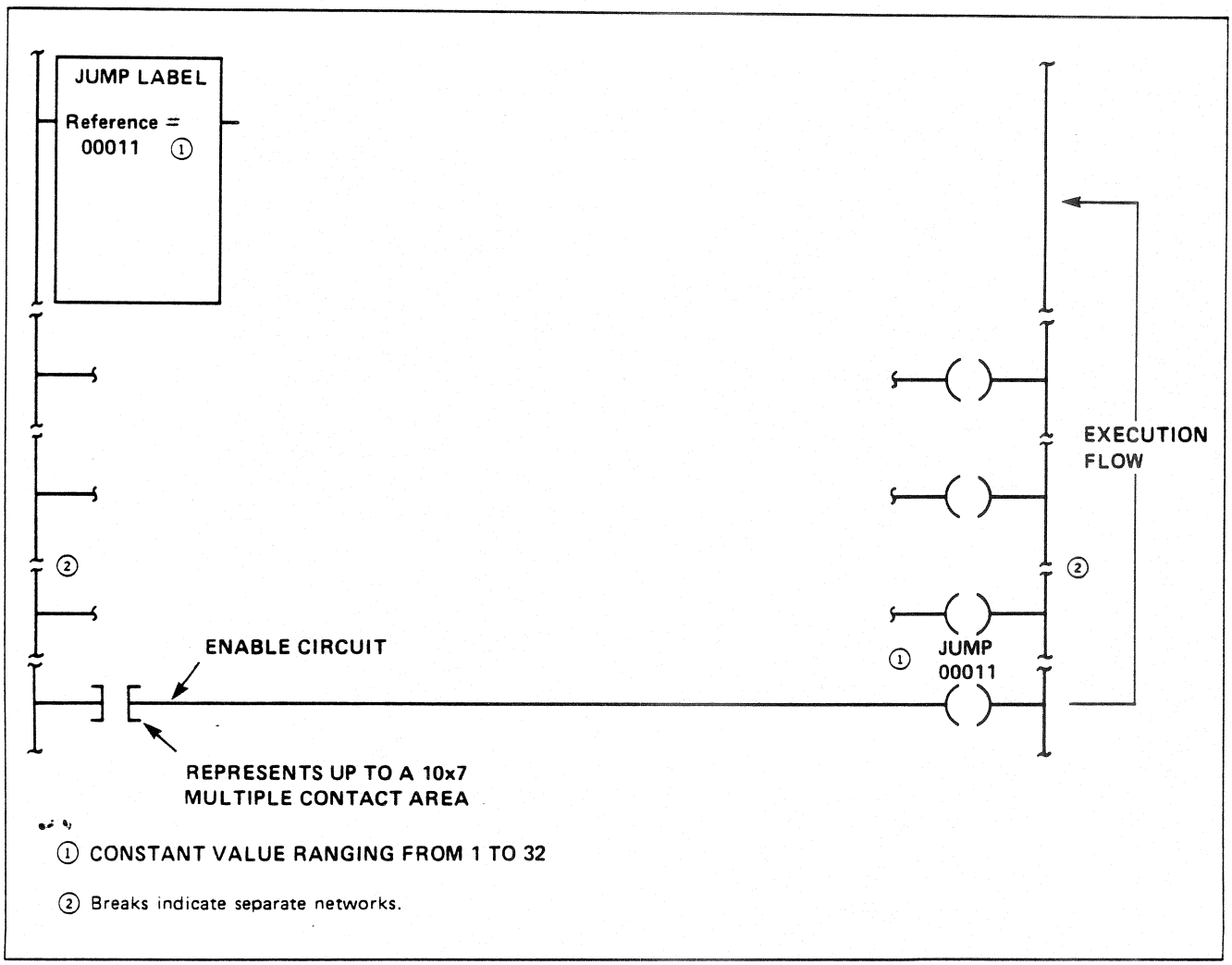


Figure JMP-1. Jump Function Circuit (Typical)

REFERENCE =

The same Reference Number (=) must be programmed with each Jump Coil and Jump Label function pair. However, each distinct Jump Coil and Jump Label pair must be assigned a different Reference Number. This Number is a constant from 1 to 32. Jump Coil and Jump Label pairs can be nested, assuming each distinct pair has a different Reference Number.

CAUTION

Provision must be made in the ladder diagram to prevent a continuous "jumping back" or execution of the Jump Coil function. Failure to do so results in a fatal watchdog timeout fault. Specifically, bit 3 of Fault Register 2, the LP hardware watchdog timer, will detect a fault under these circumstances.

8-2-2. APPLICATION

The Jump Coil and Jump Label function pair are used whenever it is desirable to execute a portion of the ladder diagram more than once during a single ladder diagram scan. One example of the use of the Jump function pair is shown in Paragraph 8-16-2.

8-3. INVERTING FUNCTION (008)

The Inverting programmable function inverts the conducting or nonconducting status of the circuit path in which it is programmed. (See Figure INV-1.) The Inverting function can be located anywhere in the contact area of the ladder diagram.

8-3-1. SPECIFICATIONS

The programming specifications for the Inverting function are listed here.

INPUT CIRCUIT

The input circuit to the Inverting function can consist of up to 9 horizontal contacts located in up to 7 parallel paths. The Inverting function requires the same space on the ladder diagram as a contact, and can be located anywhere in the contact area of the ladder diagram.

OUTPUT

The Inverting function inverts the conducting state of the input. When the contacts to the left of the Inverting function are conducting, the output will not conduct, and vice versa.

8-3-2. APPLICATION

The Inverting function is used to invert the conducting state of the input. Figure INV-2 shows a typical example where two coils in opposite conduction states are desired.

An example where the Inverting function is used to allow a counter to count on the trailing edge of the count enable circuit is shown in Figure TRN-2 and described in Paragraph 7-4-2.

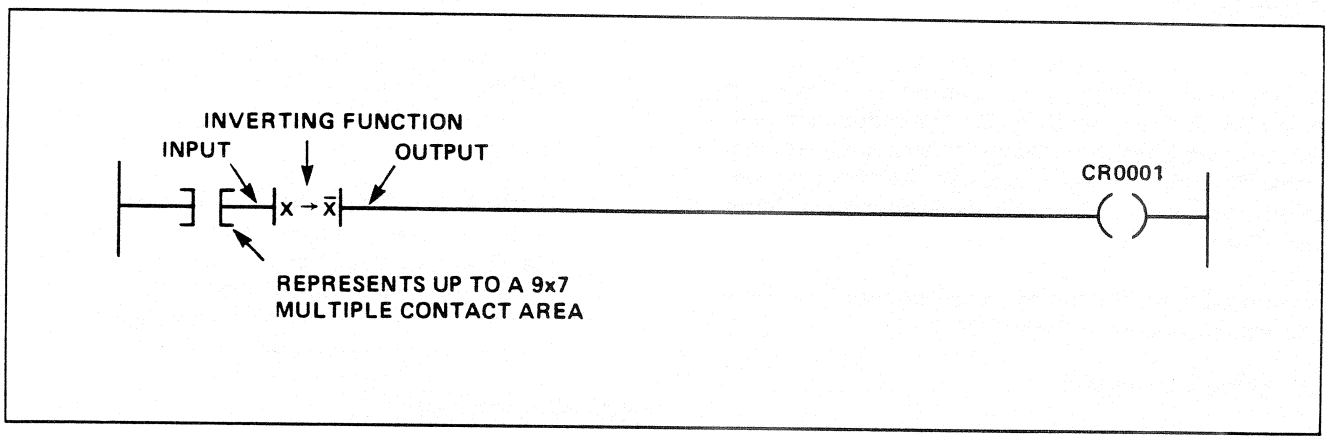


Figure INV-1. Inverting Function in Circuit (Typical)

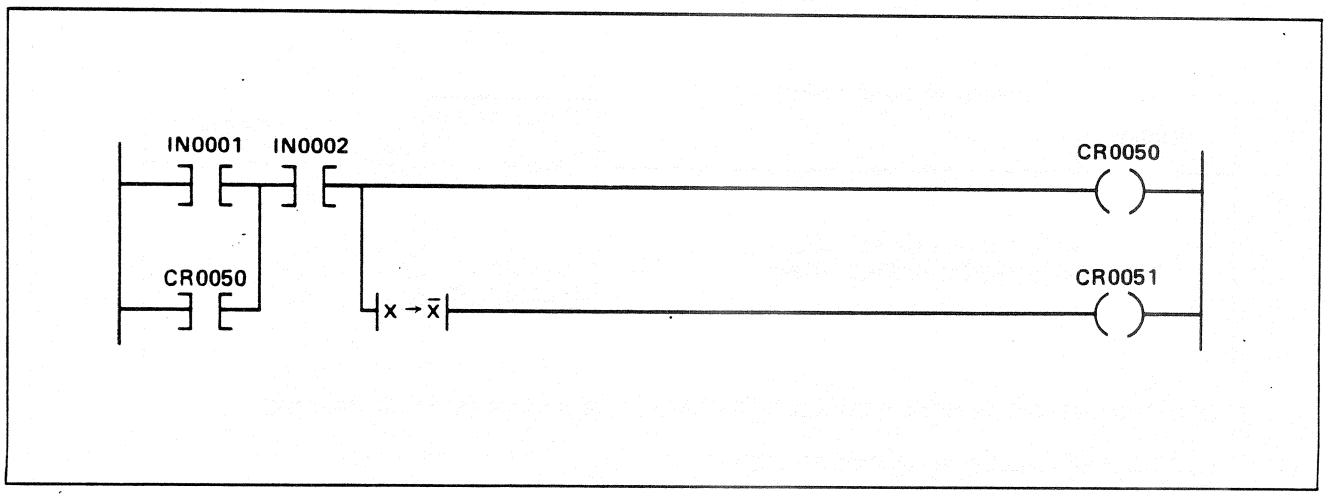


Figure INV-2. Inverting Function Example

8-4. TIME OFF FUNCTIONS (031-033)

The Time Off programmable function provides, as the name implies, an off-delay timing function. The 3 different Time Off functions operate identically except that the time increments and their associated timing ranges vary, as shown in Table TMR-1. An example of a typical Time Off function circuit is shown in Figure TMR-1. Observe the Figure and note the following 3 points.

1. The value of the Actual Register begins to increment, or time, when the path of the timing enable circuit first **stops** conducting, as shown in Figure TMR-2. (See times 1 and 9, 13 and 16.)

2. The value of the Actual Register is reset when the conducting path of the timing enable circuit first conducts (goes from off to on). (See times 0, 8, 11 and 15 in Figure TMR-2.)

3. The Q output conducts when the timing enable circuit first conducts (off-to-on transition), as shown in Figure TMR-2. (See times 0 and 8.) The conduction path of Q continues after the timing enable circuit goes false, thereby initiating the timing function. At this point conduction stops when the Actual Value equals the Preset Value. (See times 7 and 22 on Figure TMR-2.)

The truth table for the timing enable circuit of the Time Off function is shown in Table TMR-2.

8-4-1. SPECIFICATIONS

The programming specifications for the Time Off function are identical to those listed in Paragraph 7-5-3 with the following differences:

1. The Time Off function does not have a reset circuit.

TABLE TMR-1. OFF-DELAY TIMER RANGES

Display No.	Type	Timing Range
31	Time Off 1.0	1 to 32,767.0 seconds
32	Time Off 0.1	0.1 to 3,276.7 seconds
33	Time Off .01	0.01 to 327.67 seconds

2. The Actual Register of the Time Off function is reset to zero whenever the timing enable circuit conducts.

3. Table TMR-2, located here, is to be used to determine the effects of the timing enable circuit.

Note

The Actual Register value of the Time Off function is retentive—even if power is removed and restored, or if the keyswitch is put in the STOP: PROGRAM mode.

Note

If the Actual Register is incrementing while the ladder diagram scan stops, the Q output then stops conducting. When scanning resumes, the output conducts, unless contacts in the timing enable circuit have changed states.

8-4-2. APPLICATIONS

The Time Off function provides a direct equivalent of a hardwired off-delay timer. Paragraph 7-5-4 describes an application which provides a time-off delay using the On Delay function.

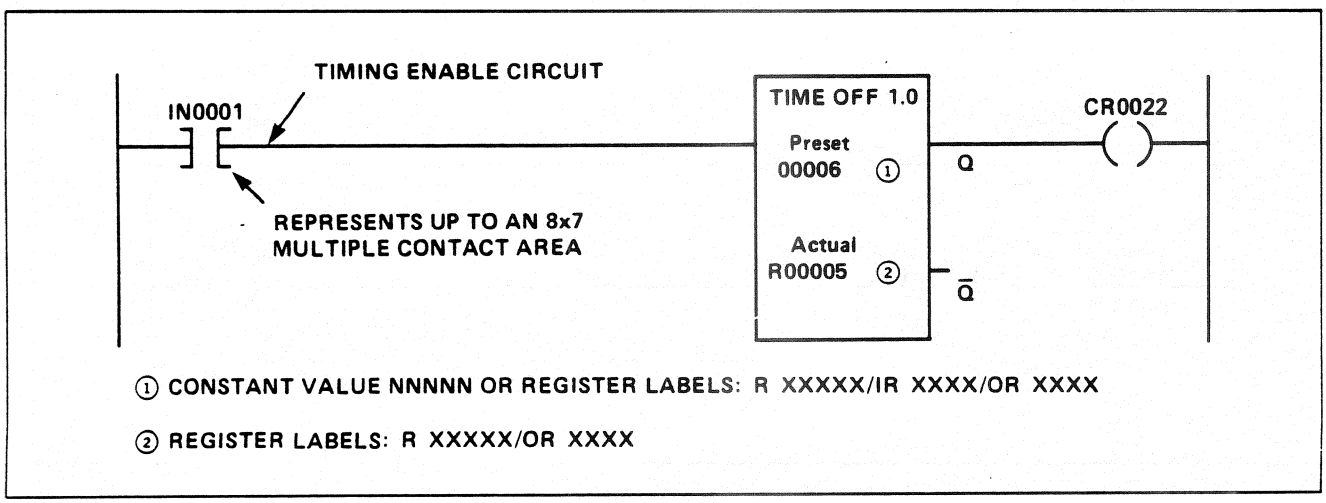


Figure TMR-1. Time Off Circuit (Typical)

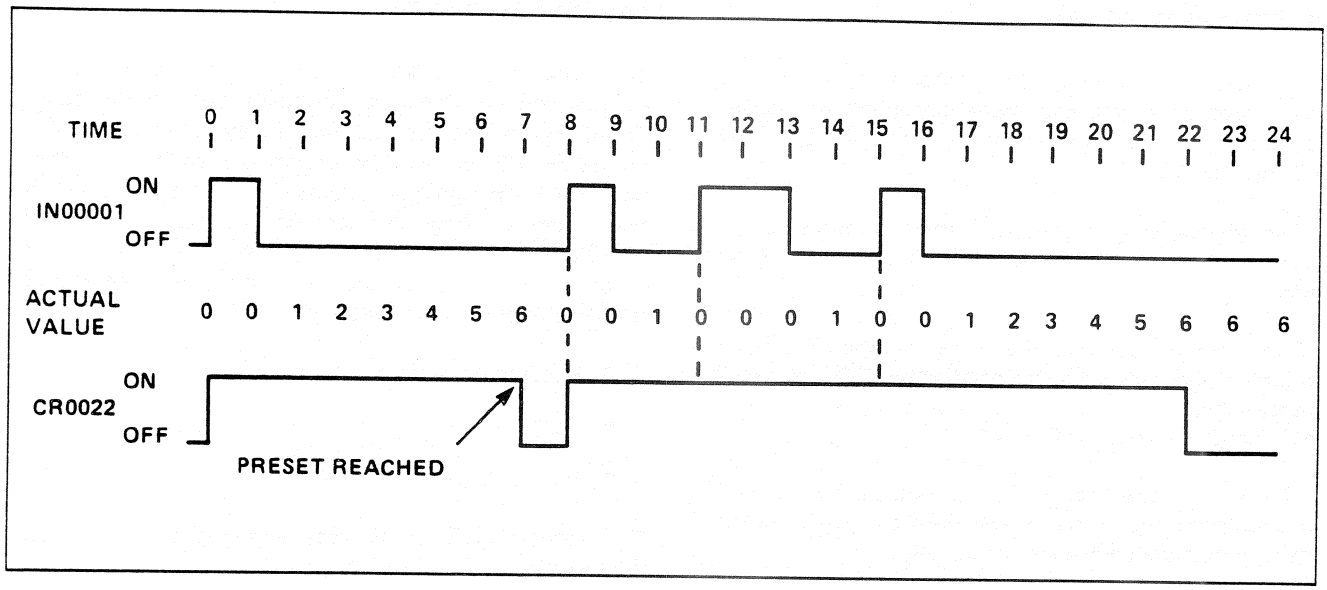


Figure TMR-2. Timing Diagram

TABLE TMR-2. TIME OFF TRUTH TABLE

Timing Enable Circuit	Result
①	• Reset the Actual Register to 0 • The Q output conducts
②	• Initiates timing; allows the Actual Value to increment • When Actual Value equals Preset Value, Q output becomes nonconducting

① Indicates an off-to-on transition of the conducting status of the timing enable circuit.

② Indicates an on-to-off transition of the conducting status of the timing enable circuit.

8-5. SQUARE ROOT (042)

The Square Root programmable function calculates the square root of a whole number contained in double-precision form in a pair of registers. See Figure SQ-1 and note the following 4 points:

1. The enable circuit conducts, thereby allowing the square root operation to occur during each scan of the ladder diagram.

2. The Operand displays the lower numbered register containing the most significant digits (MSD) of the 2 consecutively numbered registers upon which the square root function is to be performed.

3. The square root result is rounded to the nearest whole number and stored in the Result Register during each scan that the enable circuit conducts.

4. The error output conducts should an attempt be made to find the square root of a value which is negative or greater than $32,767^2$.

8-5-1. SPECIFICATIONS

The programming specifications for the Square Root function are listed here.

FUNCTION BLOCK

The Square Root function block requires 2 horizontal contact spaces by 3 parallel paths on the display. The function can be located anywhere in the contact area.

OPERAND REGISTER

The Operand Register references the positive whole number value from which the square root is to be found. The Operand Register displayed in the function block is the lower of 2 consecutively numbered registers and contains the most significant digits. The value contained in the Operand Register can be up to $32,767^2$ or 1,073,676,289. If the number in the Operand is larger or negative, the error output conducts. The Operand is specified as any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the register number on the display.

RESULT REGISTER

The Result Register contains the square root of the number contained in the Operand. The Result Register is rounded to the nearest whole number. The Result is specified as a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the display.

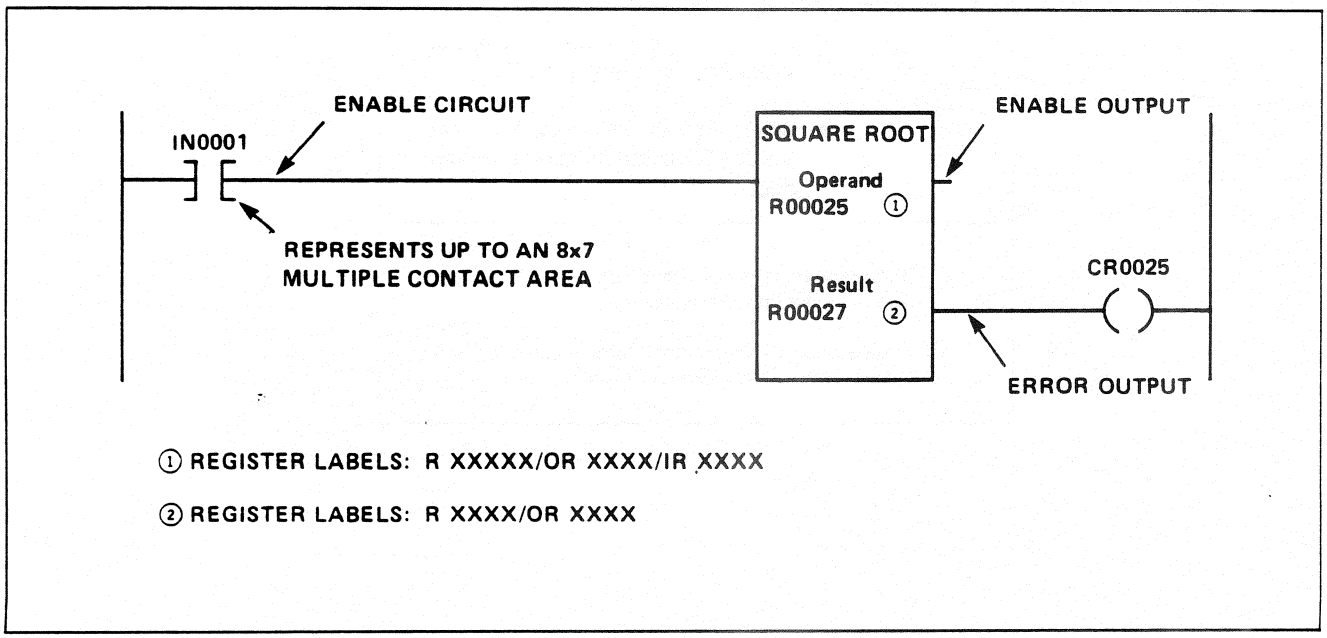


Figure SQ-1. Square Root Function Circuit (Typical)

ENABLE CIRCUIT

When the enable circuit conducts, the square root calculation on the value contained in the Operand Register is performed during each scan of the ladder diagram. The square root value is stored in the Result Register. The enable circuit consists of up to 8 horizontal contacts by 7 parallel paths on the display. The rules for programming contacts listed in Section 6 apply here.

ENABLE OUTPUT

The enable output simply follows or repeats the conducting or nonconducting state of the enable circuit.

ERROR OUTPUT

The error output conducts when the enable circuit conducts, and the value of the Operand Register is either:

- A negative number, or
- Larger than 1,073,676,289

When the Operand Register contains a number that is too large, the error output conducts, and the Result Register holds the last valid calculation data.

When the Operand Register contains a negative value, the error output conducts, and the Result Register contains the equivalent positive square root of the Operand Register's value.

Note

The outputs of the Square Root function block can be left unconnected in the network.

8-5-2. APPLICATIONS

The analog outputs of a kilowatt/kilovolt-ampere (kw/kVAR) reactive transducer are converted to digital data by Analog Input Modules. Square roots are required to calculate a kVA value which is used to derive the power factor (PF) from these inputs. Figure SQ-2 shows the calculations required to derive that power factor. Figure SQ-3 contains the ladder diagram used to provide the power factor calculations.

Observe the Figure and note the following points:

1. The calculations are performed continuously, although contacts could be located in series with the functions.

2. The resulting power factor used would be a fraction. The kilowatts are multiplied by 1000, so the answer is a whole number in the range of 1 thru 999.

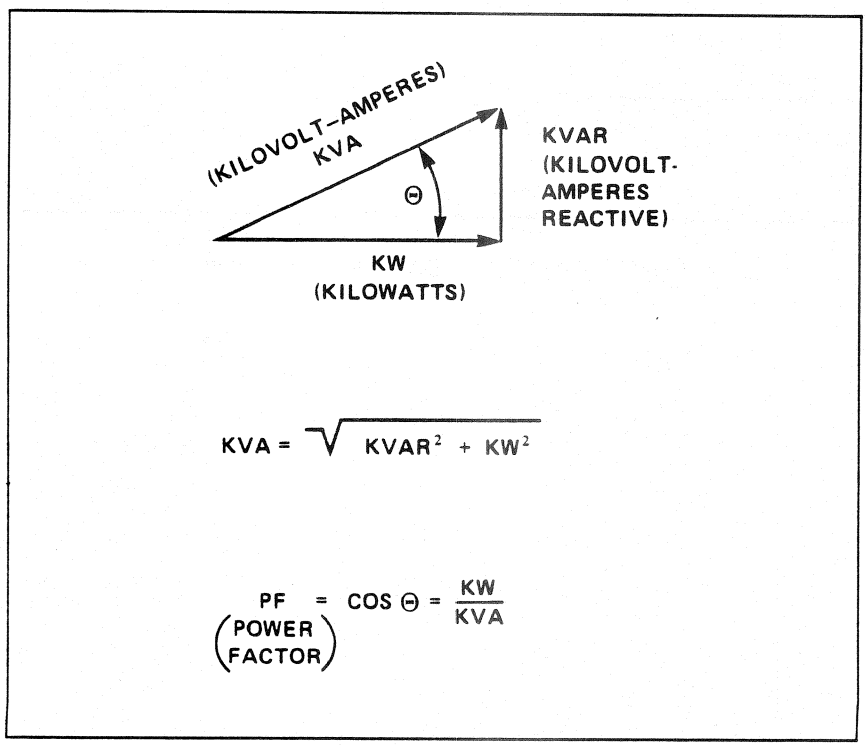


Figure SQ-2. Power Factor Calculation

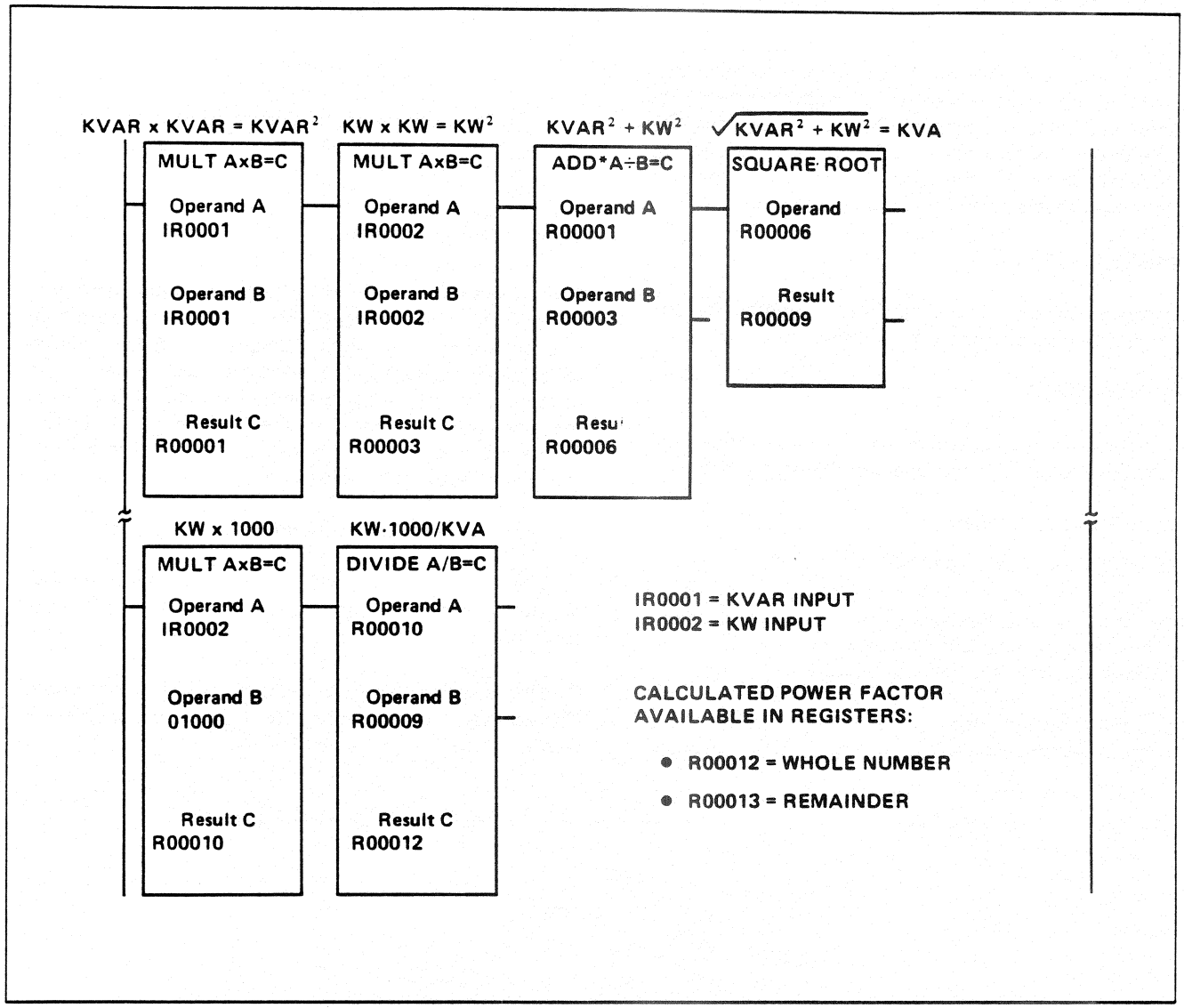


Figure SQ-3. Power Factor Ladder Diagram

8-6. NEGATE FUNCTIONS (043, 044)

The single-precision Negate (043) and double-precision Negate (044) programmable functions perform a 2's complement conversion on the binary data contained in a specified register. The data to be converted may be in true binary (positive) or 2's complement (negative) form. (See Figure NG-1.) The Operand Register specifies the register containing the value to be converted. During each ladder diagram scan, when the enable circuit conducts, the value contained in the Operand Register is converted and stored in the Result Register.

8-6-1. SPECIFICATIONS

The programming specifications for the Negate functions are listed here.

FUNCTION BLOCK

The Negate function block requires 2 horizontal contact spaces by 3 parallel paths on the display. The function block can be located anywhere in the contact area.

OPERAND REGISTER

The Operand Register references the value to be converted. The Operand is a single register for the single-precision Negate function. The double-precision Negate function provides 2 consecutively numbered registers for the Operand where:

- The lower number register displayed on the ladder diagram contains the most significant digits (MSD)
- The higher, or second, numbered register contains the least significant digits (LSD)

The Operand Register of the single-precision Negate function ranges from -32,768 to +32,767, while the double-precision Operand Register ranges from -2,147,483,648 to +2,147,483,647.

The Operand is specified by the programmer as a:

- Holding register (R)

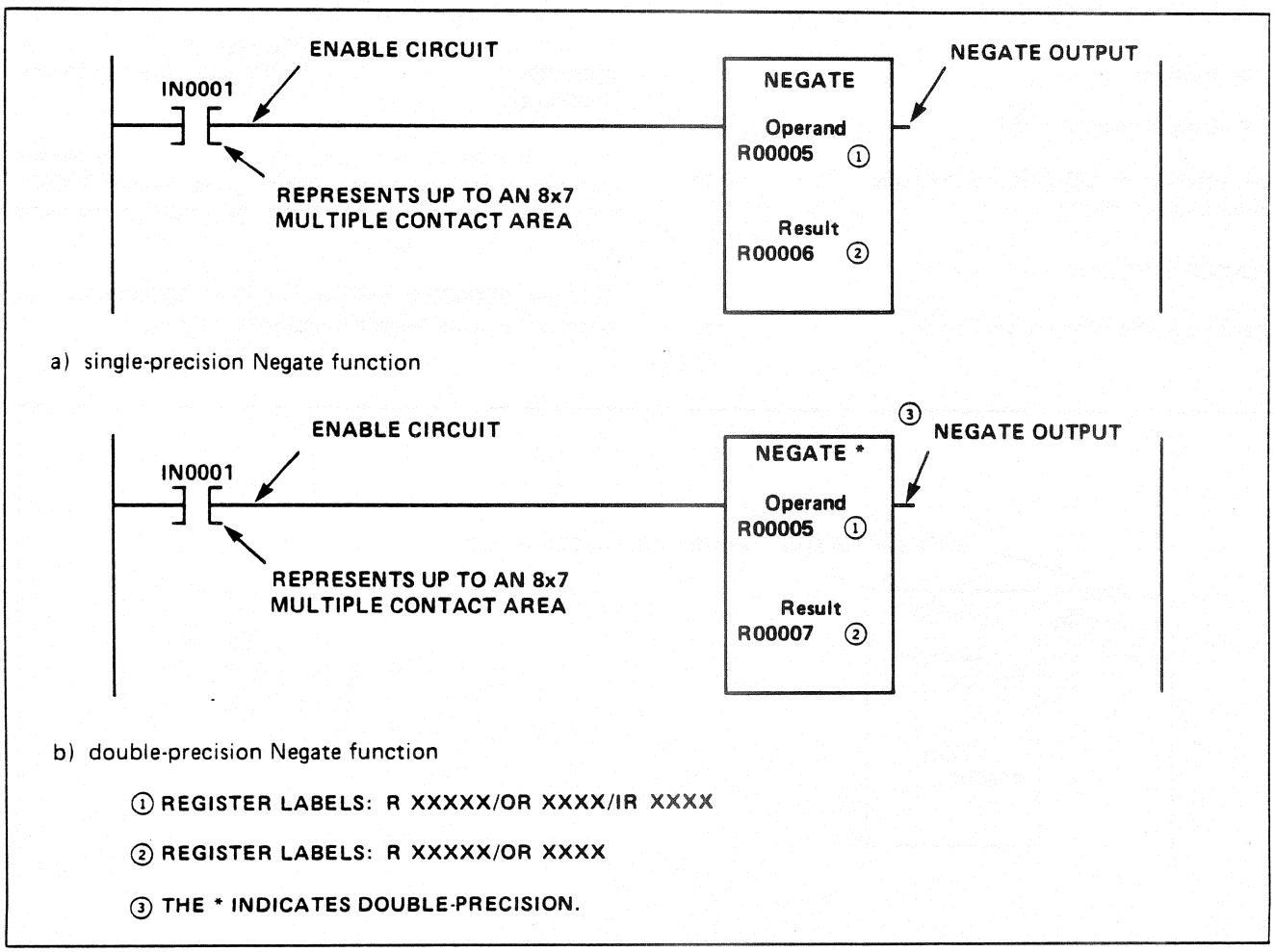


Figure NG-1. Negate Function Circuits (Typical)

- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the register number on the ladder diagram display.

RESULT REGISTER

The Result Register contains the 2's complemented value of the Operand Register. The conversion takes place during each scan if the enable circuit is conducting. The Result Register consists of a single-precision register for the single-precision Negate function. The double-precision Negate function's Result Register consists of 2 consecutively numbered registers, where the:

- Lower numbered register contains the most significant digits
- Higher numbered register contains the least significant digits

The register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the ladder diagram display.

ENABLE CIRCUIT

The 2's complement conversion takes place during each scan when the enable circuit conducts. A positive binary value is converted to a negative value, and a negative value is converted to a positive value. The enable circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths. The rules for programming contacts listed in Section 6 apply to these circuits.

NEGATE OUTPUT

The output of the Negate function block simply follows or repeats the conducting or nonconducting state of the enable circuit.

Note

The output of the Negate function can be left unconnected in the network.

8-6-2. APPLICATIONS

The Negate function can be used to convert a binary or 2's complement type value to an absolute value for use by other functions in the ladder diagram. Figure NG-2 shows a single-precision Negate function used to convert the value contained in holding register R00009 to an absolute value. Observe the Figure and note the following points:

1. When R00009 is negative (bit 16 = 1), the Negate function is enabled, thus converting the value in R00009 to a positive value and storing the result in the same holding register, R00009.

2. When R00009 is positive, bit 16 of R00009 does not conduct, and the Negate function is inactive.

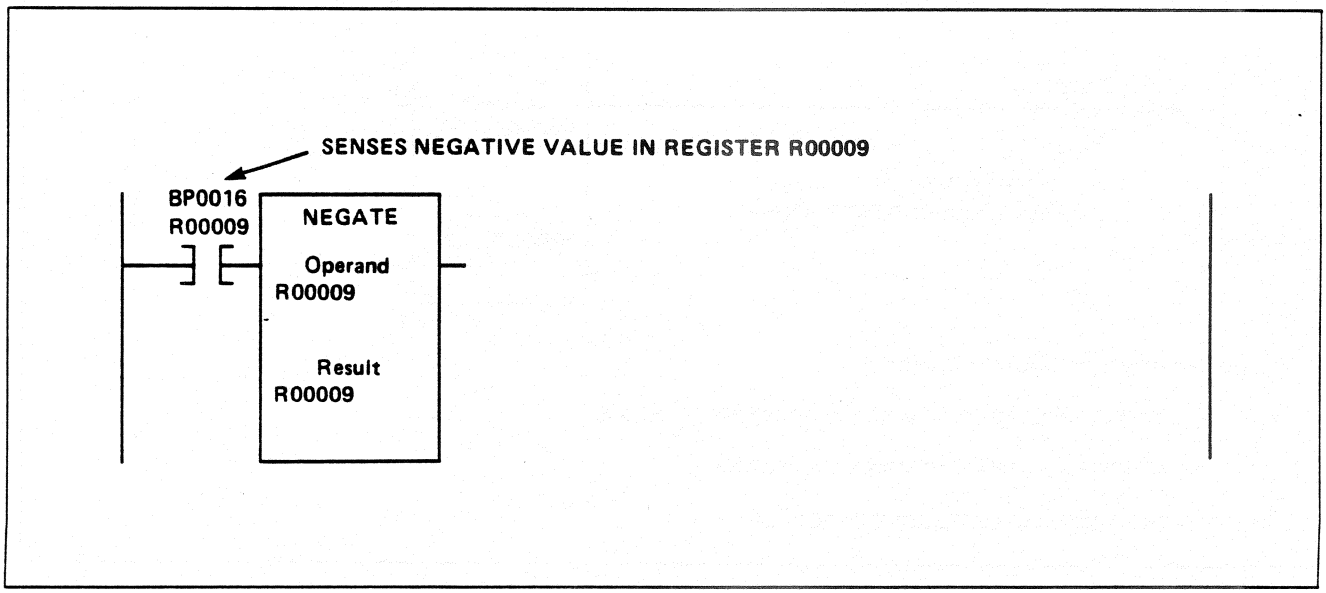


Figure NG-2. Negate Function Application

8-7. DOUBLE PRECISION ADD (069), SUBTRACT (071)

The Double Precision Add and the Double Precision Subtract programmable functions add or subtract registers containing 32 binary digits, or bits. (See Figure DAS-1.)

The Double Precision Add and Subtract functions operate and are programmed identically to the Single Precision Add (068) and Subtract (070) functions with the following 3 exceptions:

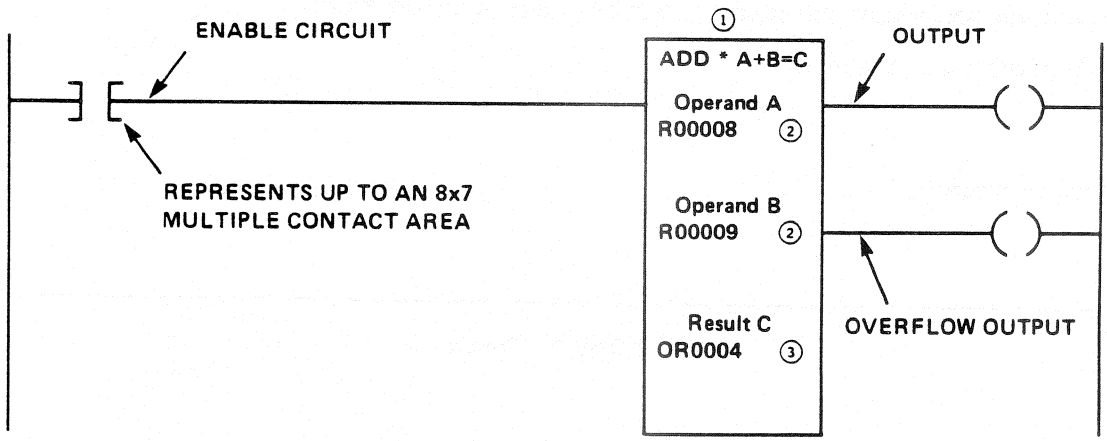
1. The Double Precision function blocks display an asterisk directly after the add or subtract name on the display. (See Figure DAS-1.) This indicates double precision.

2. Operands A and B and the Result C Register are consecutively numbered, double-precision registers, where the:

- Lower numbered register contains the most significant digits
- Higher numbered register contains the least significant digits

3. The overflow output conducts when the Result C Register contains a value above $(2^{31} - 1)$ or below $-(2^{31})$. Note: $2^{31} = 2,147,483,648$.

See Paragraph 7-7-1 for programming specifications and Paragraph 7-7-2 for applications.



① THE * INDICATES DOUBLE-PRECISION FUNCTIONS

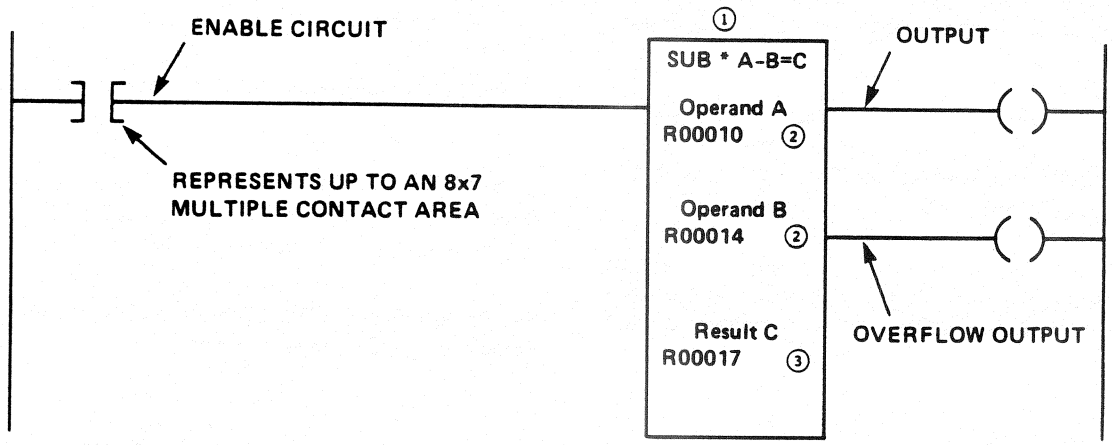
② CONSTANT VALUE (NNNN) OR REGISTER LABEL (R, IR, OR XXXX)

③ REGISTER LABEL (R, OR YYYY)

a) Add function

(Cont'd.)

Figure DAS-1. Double Precision Add and Subtract Functions Circuits (Typical)



① THE * INDICATES DOUBLE-PRECISION FUNCTIONS

② CONSTANT VALUE (NNNN) OR REGISTER LABEL (R, IR, OR XXXX)

③ REGISTER LABEL (R, OR YYYY)

b) Subtract function

Figure DAS-1. (Cont'd.)

8-8. AVERAGE FUNCTION (093)

The Average programmable function calculates the average value of a table of binary values. It then stores the result in the Average Register. The registers involved are single-precision types. See Figure AV-1 and note the following 4 points:

1. The Table Register defines the first register of the table to be averaged.

2. The Length is a constant value which specifies the number of registers in the table.

3. The Average Register contains the result of the calculations rounded to the nearest whole number. Calculations are performed when the enable circuit conducts.

4. The Scratchpad specifies the first register of a 3-register set used when over 100 registers are contained in the Table. To limit the scan time required by the Average function, a maximum of 100 registers are averaged during a single ladder diagram scan.

8-8-1. SPECIFICATIONS

The programming specifications for the Average function are listed here.

FUNCTION BLOCK

The Average function block requires 2 horizontal contact spaces by 5 parallel paths on the display. The function block can be located anywhere in the contact area of the ladder diagram.

TABLE REGISTER

The Table Register defines the first register in the table

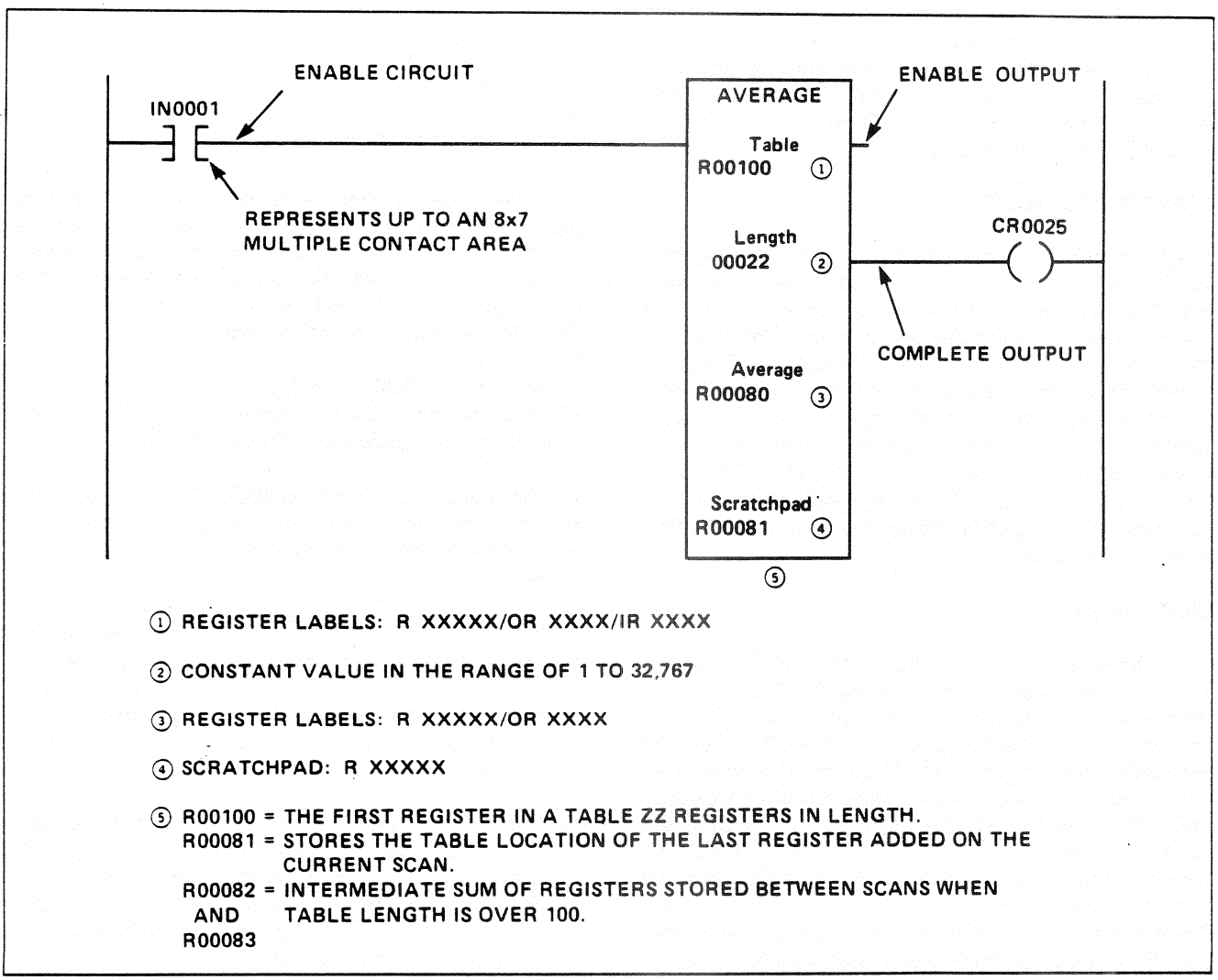


Figure AV-1. Average Function Circuit (Typical)

to be averaged. It is specified by the programmer as one of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the register number on the display.

LENGTH

The Length is specified as a constant value defining the number of registers in the table to be averaged. The Length is in the range of from 1 to 32,767.

AVERAGE REGISTER

The Average Register contains the calculated average value of the table rounded to the nearest whole number. It is a single-precision register specified by the programmer as either a holding register (R) or output register (OR). The letters R or OR precede the register number on the ladder diagram.

SCRATCHPAD REGISTER

The Scratchpad Register consists of 3 holding registers used when the table length specified exceeds 100 registers. The calculation of the average value is limited to 100 registers per ladder diagram scan in order to reduce the scan time necessary for long tables. The register number programmed and displayed stores the table location of the last register added. The next 2 consecutively numbered registers store the current sum of the registers between scans. The use of the Scratchpad Register is handled automatically by the Average function. It is only necessary to specify the register number of the first Scratchpad Register.

ENABLE CIRCUIT

When the enable circuit conducts, the average value calculation takes place. When the enable circuit is nonconducting, no calculations take place, and the first register in the Scratchpad is cleared to 0. When the Scratchpad contains more than 100 registers, the enable circuit must be energized properly to enable the calculations. The circuit must be energized until the complete output turns on for 1 or more consecutive scans than the apparent calculations needed. The conduction of the complete output indicates that the data in the Average Register is valid. For example, when averaging 400 registers, 5 consecutive scans would be needed. If after 2 scans the enable circuit becomes nonconducting, the Scratchpad is reset to 0. The enable circuit consists of up to 8 horizontal contacts by 7 parallel paths located to the left of the Average function block. The rules for programming contacts listed in Section 6 apply to these circuits.

ENABLE OUTPUT

The enable output simply repeats or follows the conducting or nonconducting state of the enable circuit.

COMPLETE OUTPUT

The complete output conducts during each scan when the enable circuit conducts and the average calculation for the whole Table has been completed.

Note

The outputs of the Average function block can be left unconnected in the network.

8-8-2. APPLICATIONS

An inspection station automatically measures the diameter of production parts. After every 10 measurements a calculation of the average dimension of the last 10 parts is performed. Figure AV-2 shows one method of collecting, storing, and calculating the average value. Observe the Figure and note the following 4 points:

1. An input, IN0001, enables the reading of data from IR0001, the input register for the Shift Left N function. (This function is discussed in Paragraph 8-9.)

2. The Shift Left N function (089) shifts the data contained in the table 16 places and stores the new data from IR0001. The table stores the data from 100 measurements.

3. The Up Counter function (038) keeps track of how many measurements are made. After each 10 measurements the Average function is enabled when the Q output of the counter conducts. After averaging is completed, the counter will be reset by output bit 1 of holding register 1.

4. The average measurement averages the first 10 registers in the table beginning with holding register R00100. The result of the averaging is stored in OR0001. This result can be compared to predetermined values to automatically set an alarm. The register can also be output to a display or printer for monitoring purposes.

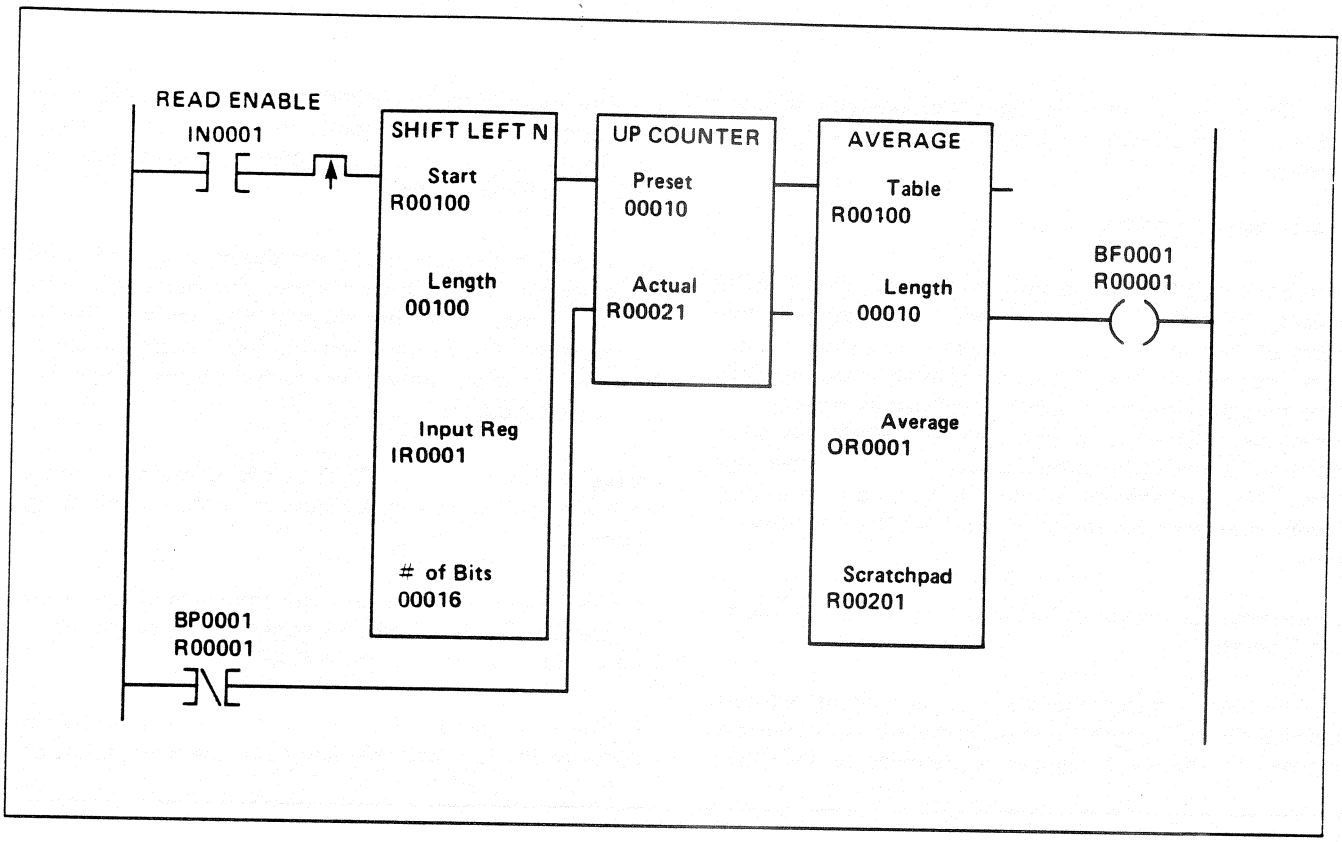


Figure AV-2. Average Application

8-9. SHIFT LEFT N (089), SHIFT RIGHT N (090)

The Shift Left N and Shift Right N programmable functions are described in Paragraphs 8-9-1 and 8-9-2, respectively.

8-9-1. SHIFT LEFT N

The Shift Left N function shifts to the left, from 1 to 16 places, the data in a "shift register." This data is in the form of individual bits. The function also transfers the same number of bits (1 to 16) from an Input Register into that shift register. "Left" is defined as moving data from the LSD position in a register toward the MSD position. If the shift causes a cascading into another register, "left" is defined as moving from bit 16 of the lower numbered register into bit 1 of the next higher one.

See the typical circuit in Figure S-1 and note the following 3 points:

1. The shift register consists of a number of registers defined by the Length. In this example it is 2. The first register in the shift register is defined by the Start Register, here R00010.

2. The Input Register (R00008 in this example) is the source of the data to be transferred into the shift register. In this example 6 bits, specified by the # of Bits, are to be transferred to the shift register.

3. The shift is N bits to the left during each scan if the shift enable circuit is conducting. The number of bits shifted is determined by the # of Bits Register. In this example the Transitional function (007) is used to allow the shift to occur only once—each time the enable circuit first conducts.

A typical example of the Shift Left N functions is shown in Figure S-2. Observe the Figure and note the following 3 points:

1. The example shown uses the following values from Figure S-1: 6 bits shifted; a 2-register shift register starting at R00010; Input Register R00008.

2. The data contained in the shift register is shifted 6 places to the left, and the data from the lower 6 bits of

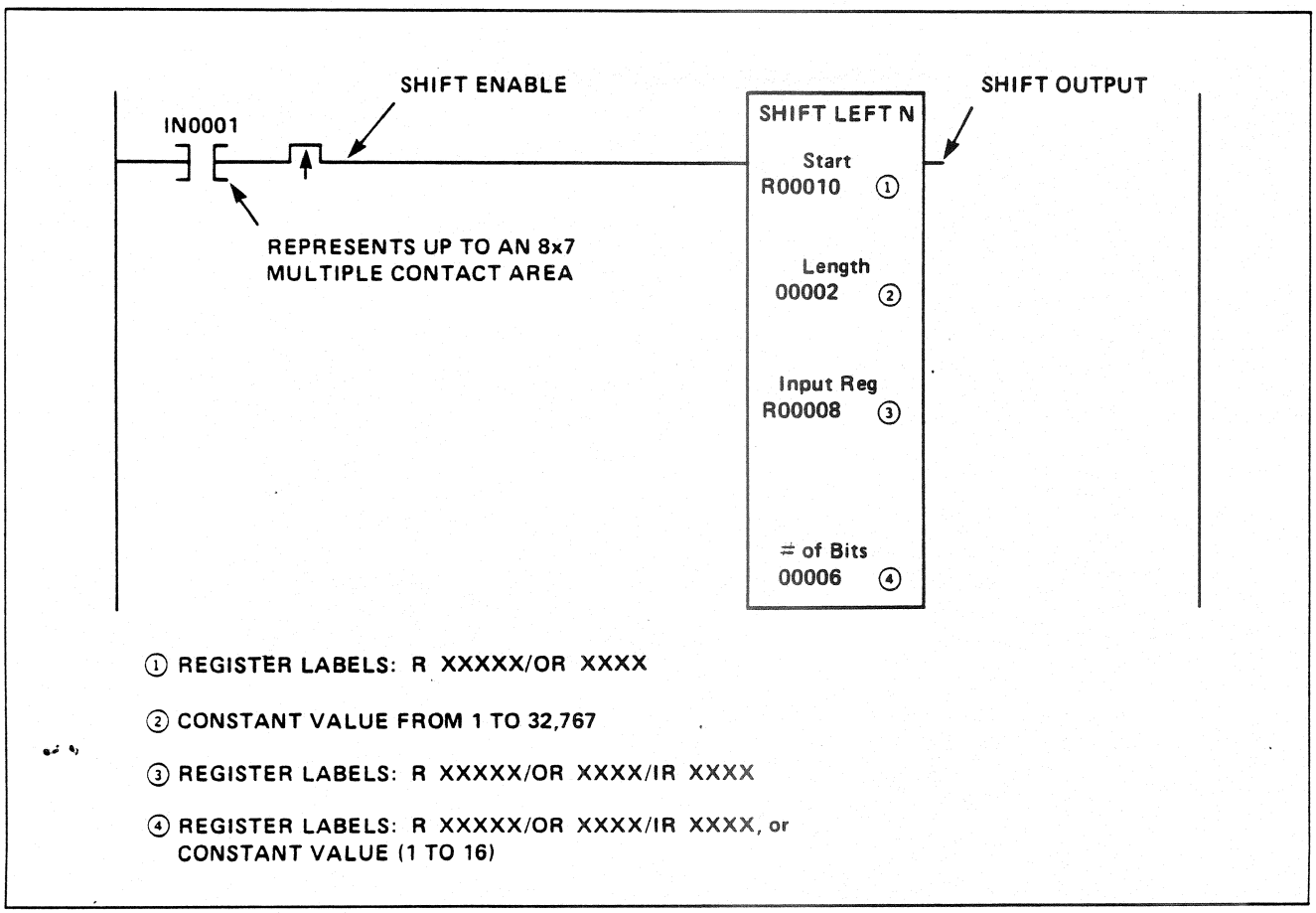


Figure S-1. Shift Left N Circuit (Typical)

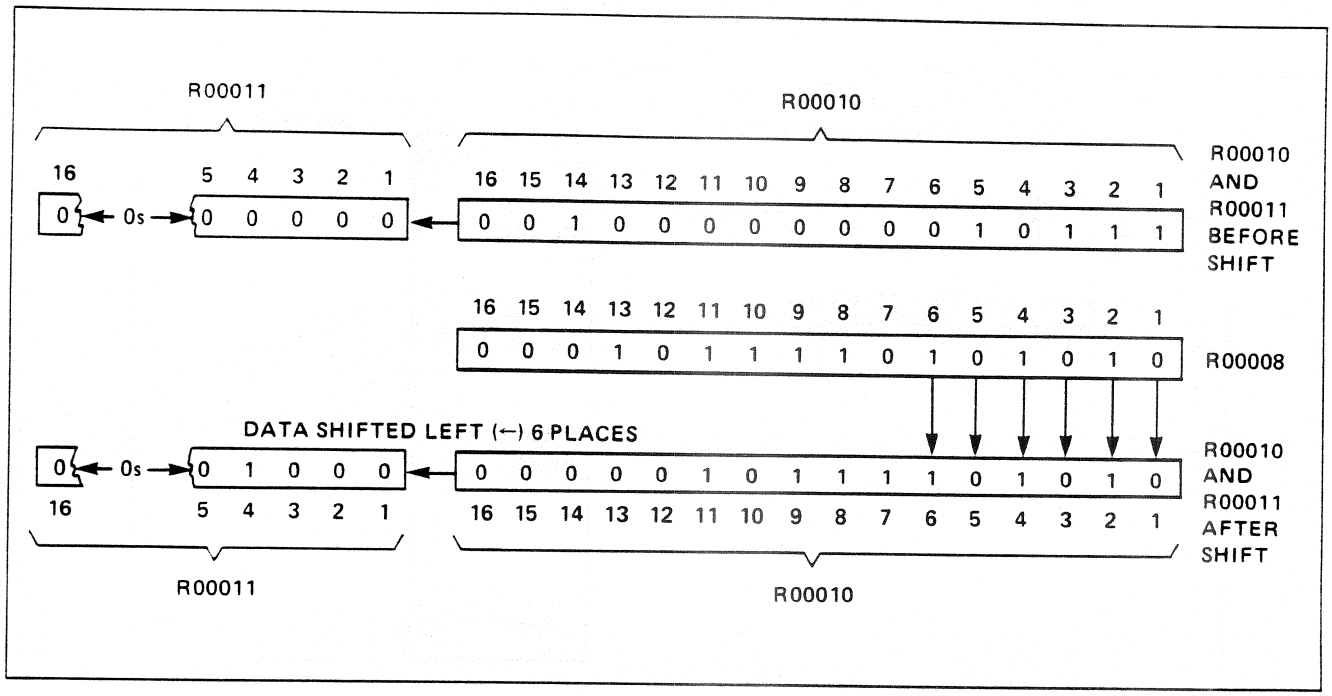


Figure S-2. Shift Left N Example

the Input Register is transferred to the lower 6 bits of the shift register during each scan when the enable circuit conducts.

3. Each scan, when the enable circuit conducts, data from the 6 most significant bits of the shift register is also shifted out of the shift register.

8-9-2. SHIFT RIGHT N

The Shift Right N function shifts to the right the data contained in a group of registers, referred to as a shift register. "Right" is defined as moving data from the MSD position in a word toward the LSD. If the shift causes a cascading into another register, "right" is defined as moving from bit 1 of the higher numbered register into bit 16 of the next lower one.

See the typical circuit in Figure S-3 and note the following 4 points:

1. The Output Register (R00005 in this example) is the register into which data is transferred from the shift register.

2. The Start Register defines the first register in the shift register. The Length defines the number of registers in the shift register.

3. The = of Bits defines how many bits are to be transferred from the shift register to the Output Register. It also specifies the number of places data in the shift register is shifted to the right.

4. The shift occurs during each scan if the shift enable circuit is conducting. In this example the Transitional function (007) is used to allow the shift only once each time the enable circuit first conducts.

Refer to Figure S-4 for a typical example and note the following 4 points:

1. The example shown uses the following values from Figure S-3: 4 bits shifted; a 2-register shift register starting at R00008; and an Output Register R00005.

2. The data from the lower 4 bits is transferred from the lower 4 bits of the shift register to the lower 4 bits of the Output Register (R00005).

3. After the data is transferred to the Output Register, the data contained in the shift register is shifted 4 places to the right.

4. The data which was contained in the lower 4 bits of the shift register is no longer contained in the shift register—only the Output Register.

8-9-3. SPECIFICATIONS

The programming specifications for the Shift Left N and Shift Right N functions are listed here:

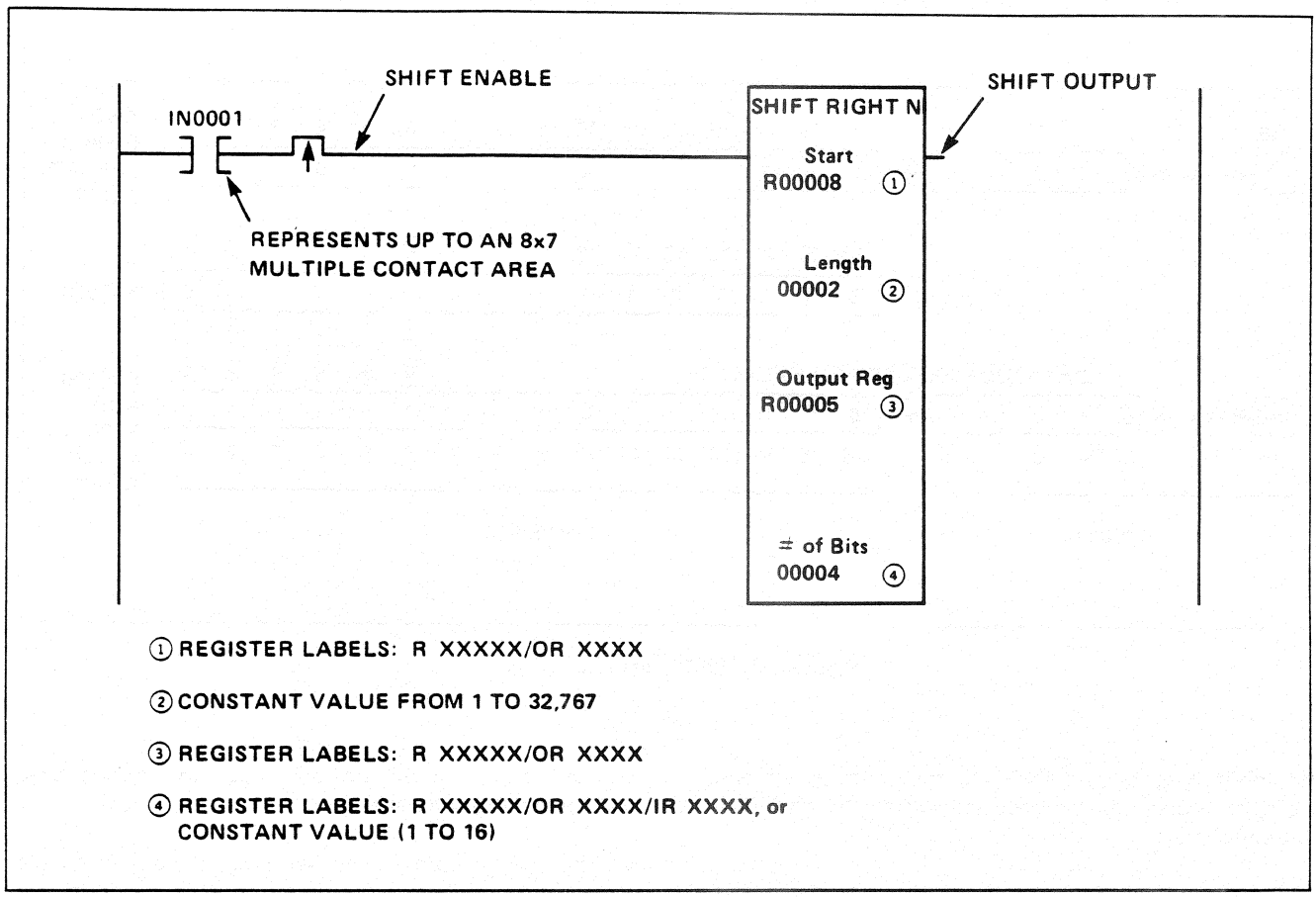


Figure S-3. Shift Right N Circuit (Typical)

FUNCTION BLOCK

The Shift Left N and Shift Right N function blocks each requires 2 horizontal contact spaces by 5 parallel paths on the display. The function blocks can be located anywhere in the contact area of the ladder diagram.

START REGISTER

The Start Register defines the lowest numbered register in the shift register. The register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the display.

LENGTH

The Length defines the number of registers in the shift register. It is a constant value in the range of from 1 to 32,767, although the number of registers in the table will normally be much lower.

INPUT REGISTER

The Input Register, associated with the Shift Left N function, defines the source from which data is to be shifted. It is specified by the programmer as one of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the register number on the loader diagram.

OUTPUT REGISTER

The Output Register, associated with the Shift Right N function, defines the destination of the data to be shifted from the shift register. The Register is specified

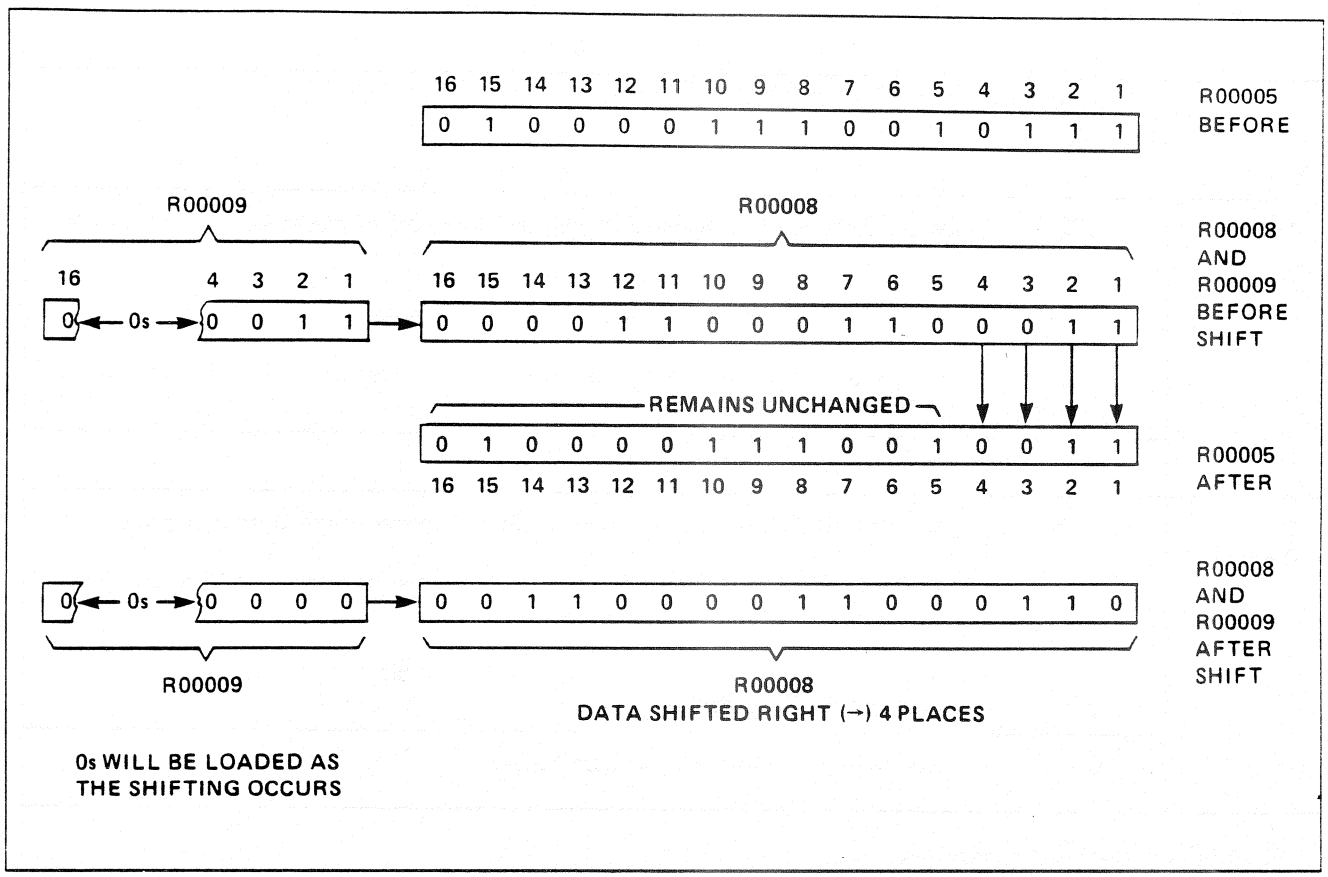


Figure S-4. Shift Right N Example

by the programmer as one of the following:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the display.

= OF BITS

The = of Bits defines the number of bits shifted during each scan of the ladder diagram, and the number of bits, starting at 1, to be transferred between the shift register and the:

- Input register in the Shift Left N function
- Output register in the Shift Right N function

The number must be in the range from 1 to 16. It is contained in one of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Constant value

The letters R, OR, or IR precede the register number on the display. If a constant is selected, no letter precedes it on the display.

SHIFT ENABLE CIRCUIT

The enable circuit enables the Shift function to occur, as listed in Table S-1. The shift enable circuit can consist of up to 8 horizontal contacts by 7 parallel paths located to the left of the function block. The rules for programming contacts listed in Section 6 apply to this enable circuit.

SHIFT OUTPUT

The shift output simply follows or repeats the conducting or nonconducting state of the enable circuit.

Note

The outputs of the Shift Left N and

TABLE S-1. SHIFT FUNCTIONS

Function	Shift Enable ① State	Description ②
Shift Left N	1 ③	• Data shifted left in the shift register N number of places (bit 1 to bit 2 to bit 3, etc.) • N bits from the highest register of the shift register are lost. • N bits of data transferred from the Input Register to the Start Register of the Table.
	0	No shifting or transferring of data occurs.
Shift Right N	1 ③	• N bits of data transferred from the Start Register to the Output Register. • Data shifted right in the shift register N number of places (bit 16 to bit 15, bit 14, etc.) • N zeroes are loaded into the most significant bits of the most significant register of the shift register.
	0	No shifting or transferring of data occurs.

① 0 = Nonconducting, 1 = conducting

② N = The number of bits programmed from 1 to 16.

③ Shifting occurs during each scan of the ladder diagram unless the Transitional element is used in the enable circuit.

Shift Right N function blocks can be left unconnected in the network.

8-9-4. APPLICATIONS

The Shift Left N and Shift Right N functions can be used in conjunction with conveyor systems. One type of conveyor application is shown in Figure S-5 where up to 20 different products or parts can be input into a single conveyor. As each part is entered onto Station 1, its BCD part code, ranging from 1 to 20, is read. The part is stored in a shift register in the order in which it was read. When the correct part enters the correct station, a solenoid connected to a diverter gate removes the part from the conveyor. Figure S-6 contains a ladder diagram used to control the conveyor hardware shown in Figure S-5. Observe the Figure and note the following 3 points:

1. The source of the BCD product code associated with each part is input register IR0001. The And Register-to-Table function (097) performs an AND function between IR0001 and constant 255 (255 = binary 0000000011111111). This "masks" bits 9 thru 16, effectively clearing any information which might be contained in these bits. Bits 1 thru 8 of IR0001 are loaded unchanged into holding register R00200.

2. The Shift Left N function receives the data from holding register R00200 and shifts it one word (16 bits) to the left each time the read new part signal occurs. Thus register R00200 holds the contents of Station No. 1, R00201 holds the contents of Station No. 2, etc.

3. A separate Compare R-R function (053) monitors each of the Stations, R00201 thru R00220. If the BCD code equals the part code, the associated diverter will be enabled. Note: The BCD codes contained in Register 2 of each Compare R-R function are converted to binary for comparison to the shift register data contained in Register 1 of each Compare R-R function.

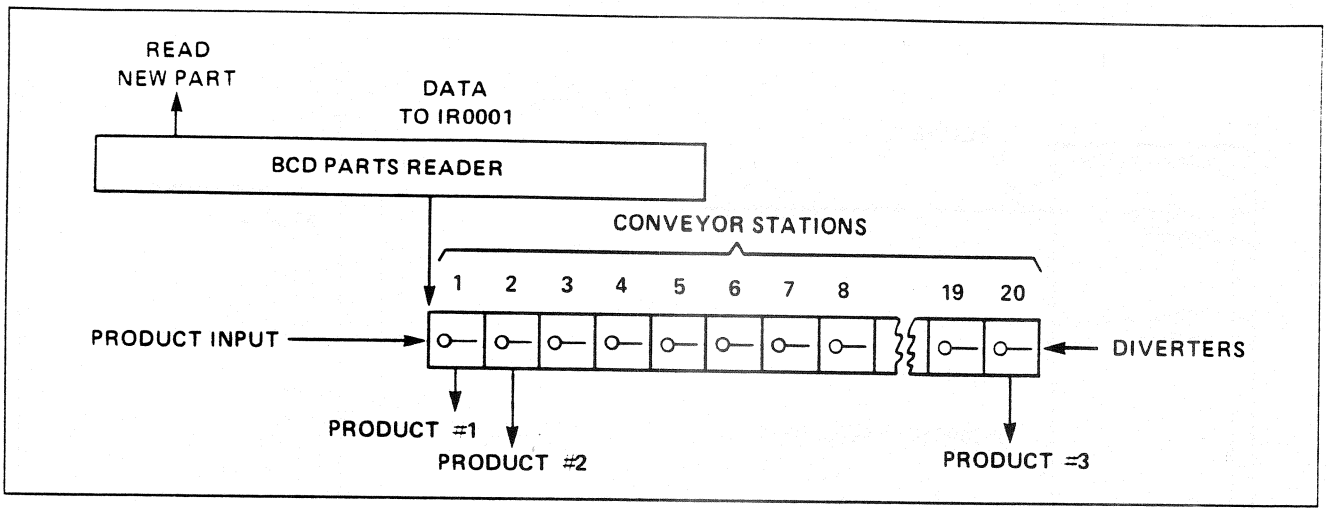


Figure S-5. Conveyor Application Mechanical Layout

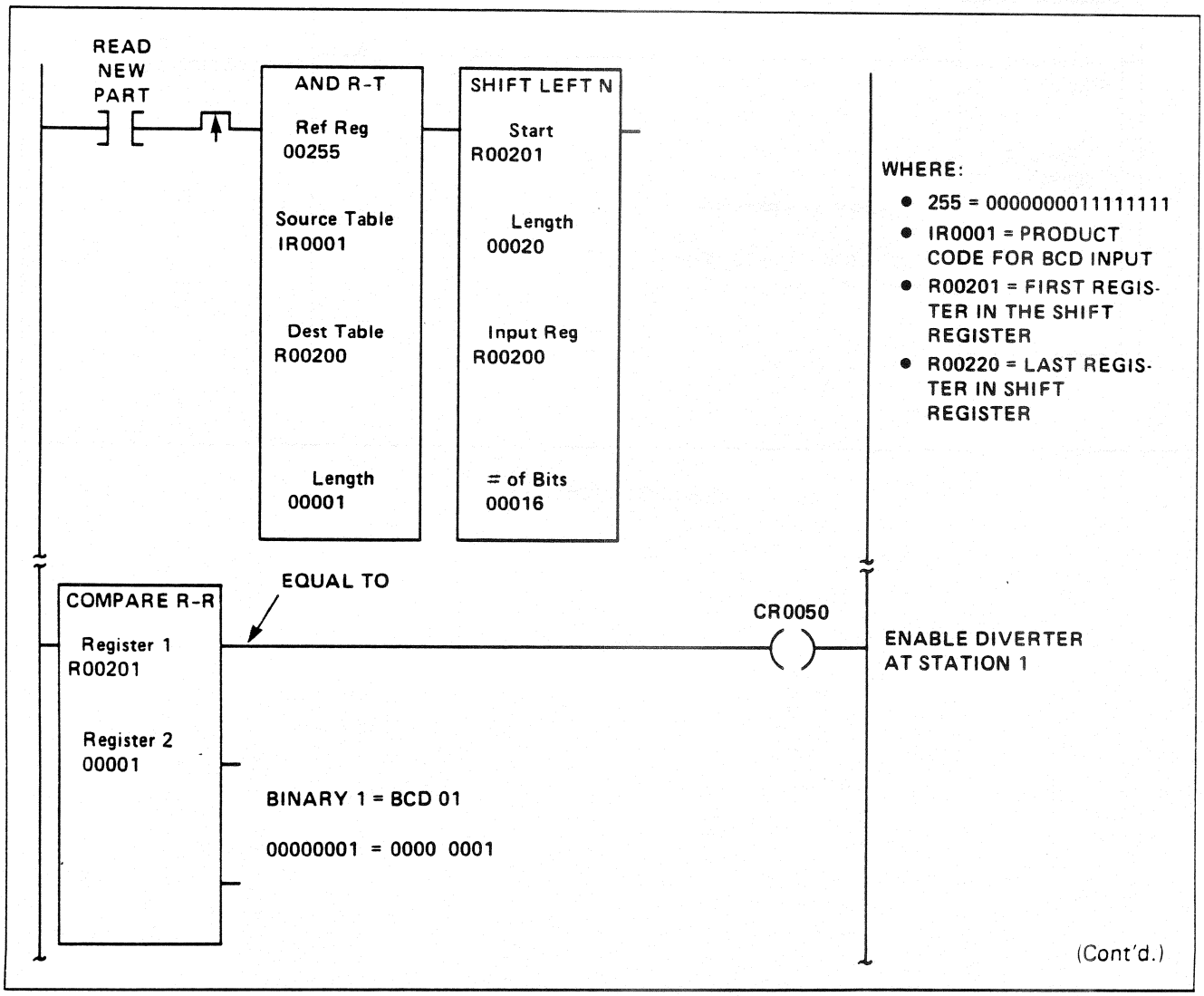


Figure S-6. Conveyor Application

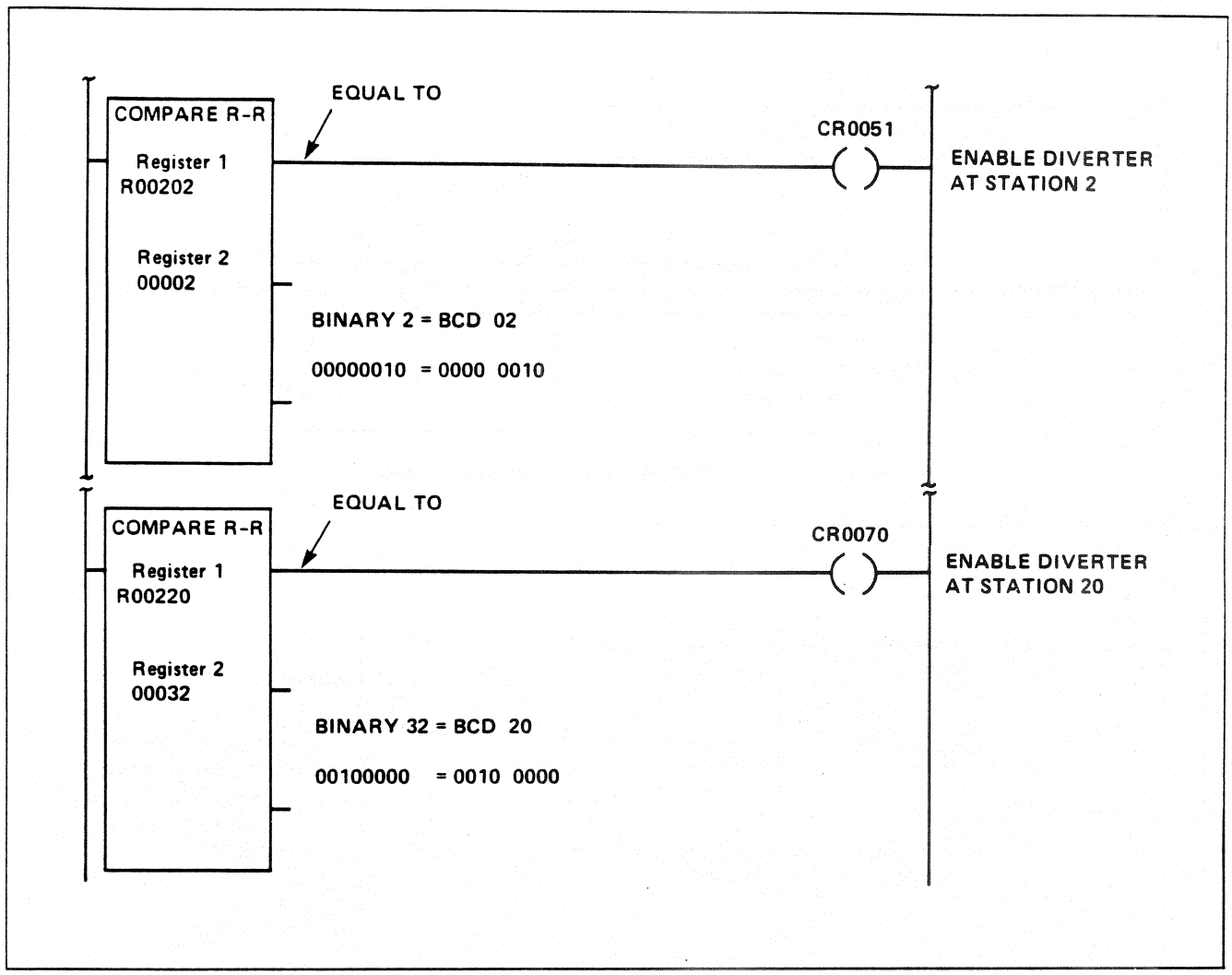


Figure S-6. (Cont'd.)

8-10. BINARY-TO-BCD TABLE (074)

The Binary-to-BCD Table programmable function converts a table of registers, referred to as the Source Table, from binary data into binary coded decimal (BCD) format. It then stores the result in a table of registers referred to as the Destination Table. Observe Figure BD-1 and note the following 5 points:

1. The conversion from binary data to BCD data takes place during each scan if the enable circuit is conducting.

2. The Source Table Register specifies the first register in the Source Table. In the example of Figure BD-1, it is holding register R00100.

3. The Destination Table Register specifies the first register in the Destination Table used to hold the converted BCD values. In this example it is holding register R00103.

4. The Length defines how many registers are contained in both the Source and Destination Tables.

5. The overrange output conducts if any register in the Source Table contains a binary value above 9,999. (This is the maximum BCD number which can be contained in 16 bits.)

8-10-1. SPECIFICATIONS

The programming specifications for the Binary-to-BCD Table function are listed here.

FUNCTION BLOCK

The Binary-to-BCD Table function block requires 2 horizontal contact spaces by 4 parallel paths on the display. The function block can be programmed anywhere in the contact area of the ladder diagram.

SOURCE TABLE REGISTER

The Source Table Register specifies the first register in the Source Table. It is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the register number on the ladder diagram display. There are 3 types of values in the Source Table:

- Values in the range of from 0 to 9,999, which are converted to BCD and held in the Destination Register.
- Negative values, which are converted to positive values and stored in the Destination Register. The negative values must be in 2's complement form to be converted correctly. (See R00101 and R00104 in Figure BD-2.)
- Values above 9,999, which cannot be contained in

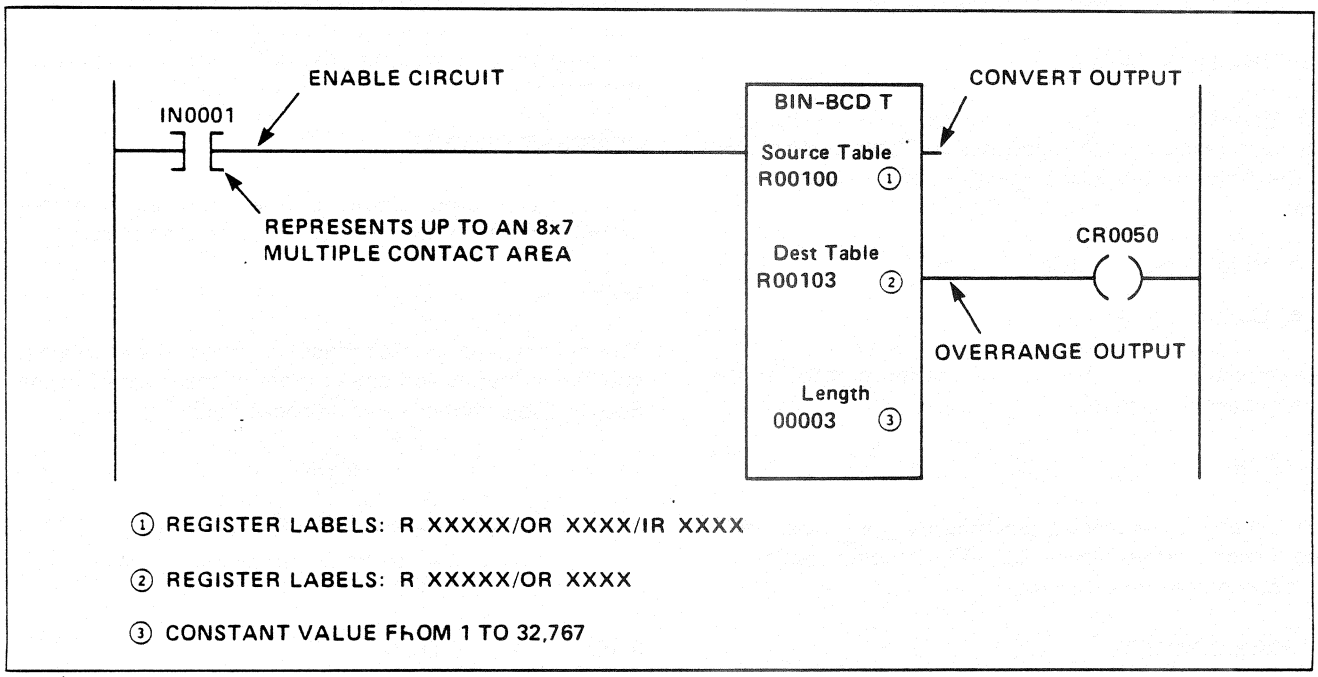


Figure BD-1. Binary-to-BCD Table Circuit (Typical)

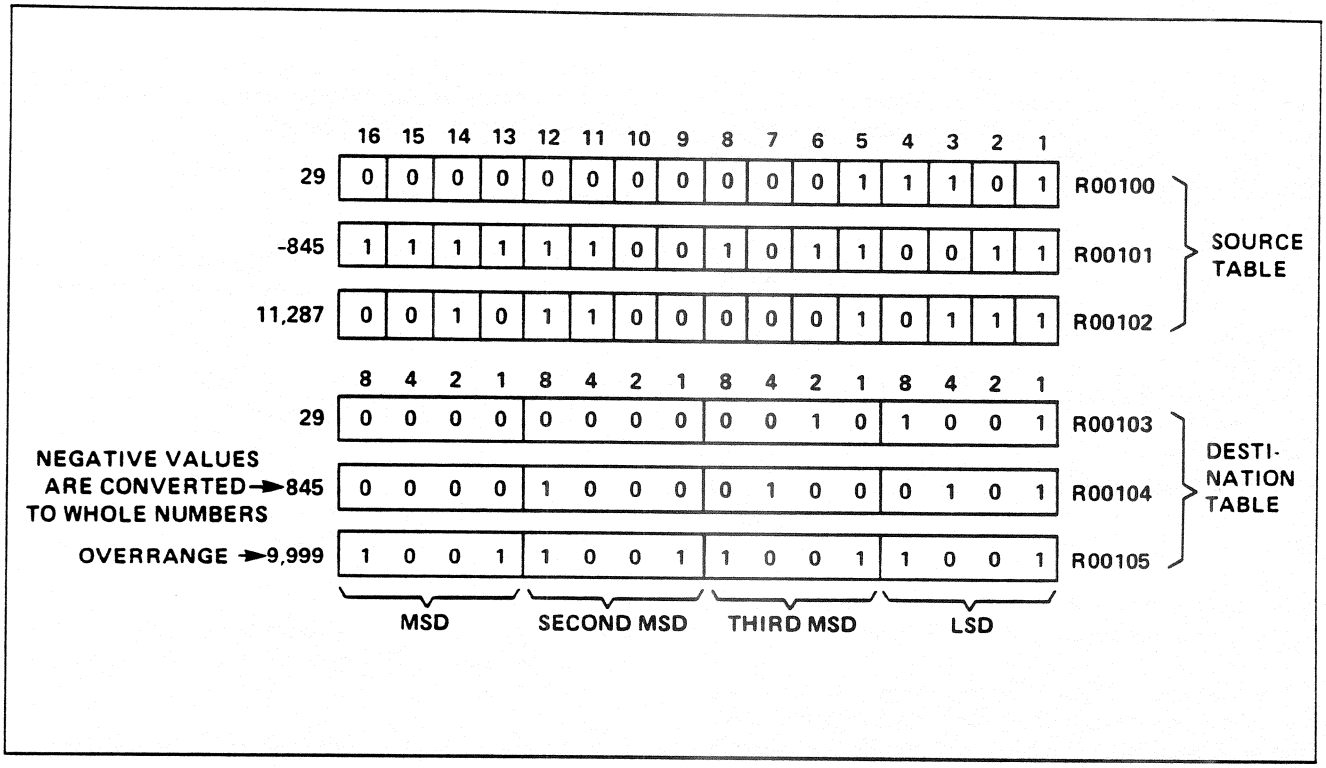


Figure BD-2. Binary-to-BCD Example

4 digits of BCD. The overrange output conducts, and the value 9,999 is held in the register. (See R00102 and R00105 in Figure BD-2.)

DESTINATION TABLE

The Destination Table stores the converted BCD data. The first register in the Destination Table is displayed in the function block and specified by the programmer as a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the ladder diagram display.

LENGTH

The Length is a constant specified by the programmer. The Length ranges up to 32,767, although tables are normally much smaller in size.

ENABLE CIRCUIT

When the enable circuit conducts, the data in the Source Table is converted from binary to BCD and stored in the Destination Table. The conversion occurs during each scan if the enable circuit is conducting. The enable circuit consists of up to 8 horizontal contacts by 7 parallel paths located to the left of the function block. The rules for programming contacts listed in Section 6 apply to this circuit.

CONVERT OUTPUT

The convert output simply follows or repeats the conducting or nonconducting state of the enable circuit.

OVERRANGE OUTPUT

The overrange output conducts whenever the enable circuit is conducting and one or more of the registers in the Source Table contains a value above 9,999.

Note

Outputs from the Binary-to-BCD function block can be left unconnected in the network.

8-10-2. APPLICATION

See Paragraph 7-11-2, Applications, for a Typical Binary-to-BCD application.

8-11. BCD-TO-BINARY TABLE (075)

The BCD-to-Binary Table programmable function converts a table of registers, referred to as a Source Table, from BCD data into binary format. It also stores the result in a table of registers referred to as the Destination Table. Observe Figure BDB-1 and note the following 5 points:

1. Conversion takes place if the enable circuit is conducting.

2. The Source Table Register defines the first register in the Source Table.

3. The Destination Table Register defines the first register in the Destination Table.

4. The Length defines how many registers are contained in both the Source and Destination Tables.

5. The overrange output conducts when the enable circuit is conducting and an illegal BCD value is contained in the Source Table. (Such a value is a digit "above 9," 1001, binary.)

A typical example is shown in Figure BDB-2.

8-11-1. SPECIFICATIONS

The programming specifications for the BCD-to-Binary Table function are listed here.

FUNCTION BLOCK

The BCD-to-Binary Table function block requires 2 horizontal contact spaces by 4 parallel paths on the display. The function block can be programmed anywhere in the contact area of the ladder diagram.

SOURCE TABLE REGISTER

The Source Table Register specifies the first register in the Source Table. It is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the register number on the ladder diagram display. Any BCD digit in the Source Table which is not a legal BCD value (0000 thru 1001) enables the overrange output.

DESTINATION TABLE REGISTER

The Destination Table Register specifies the first register in the Destination Table. It is specified by the programmer as a:

- Holding register (R)
- Output register (OR)

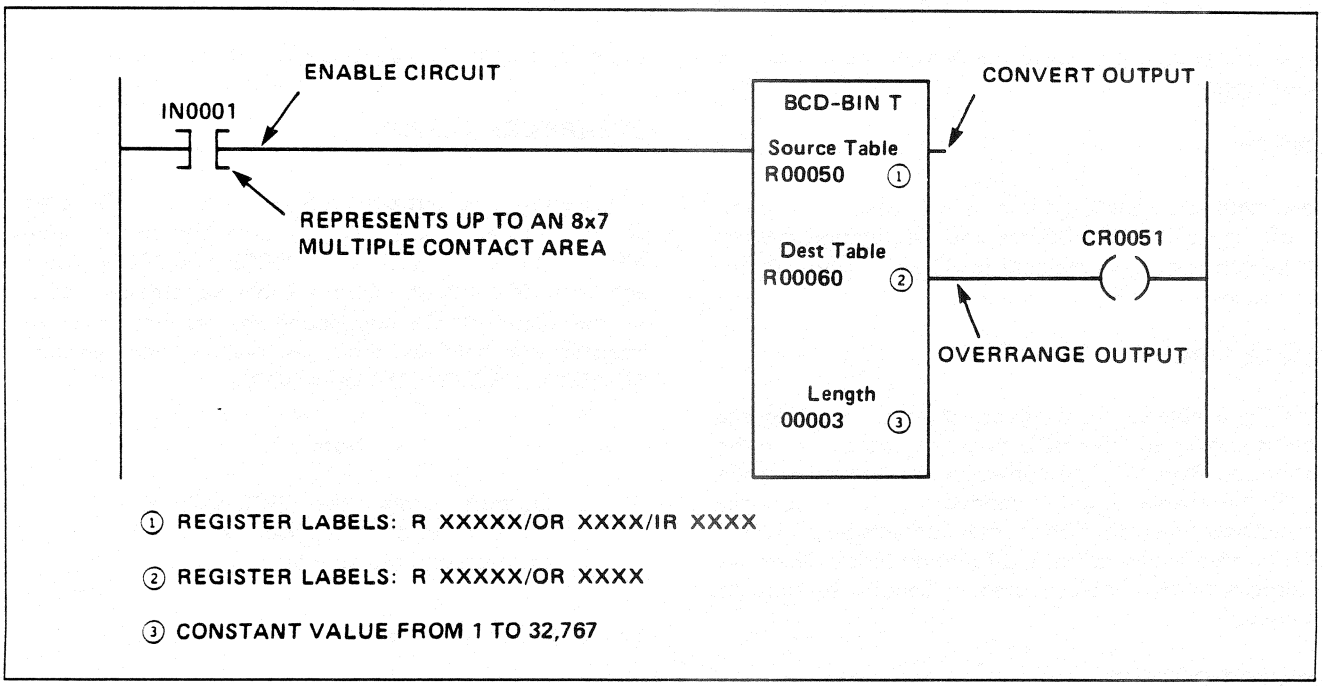


Figure BDB-1. BCD-to-Binary Table Circuit (Typical)

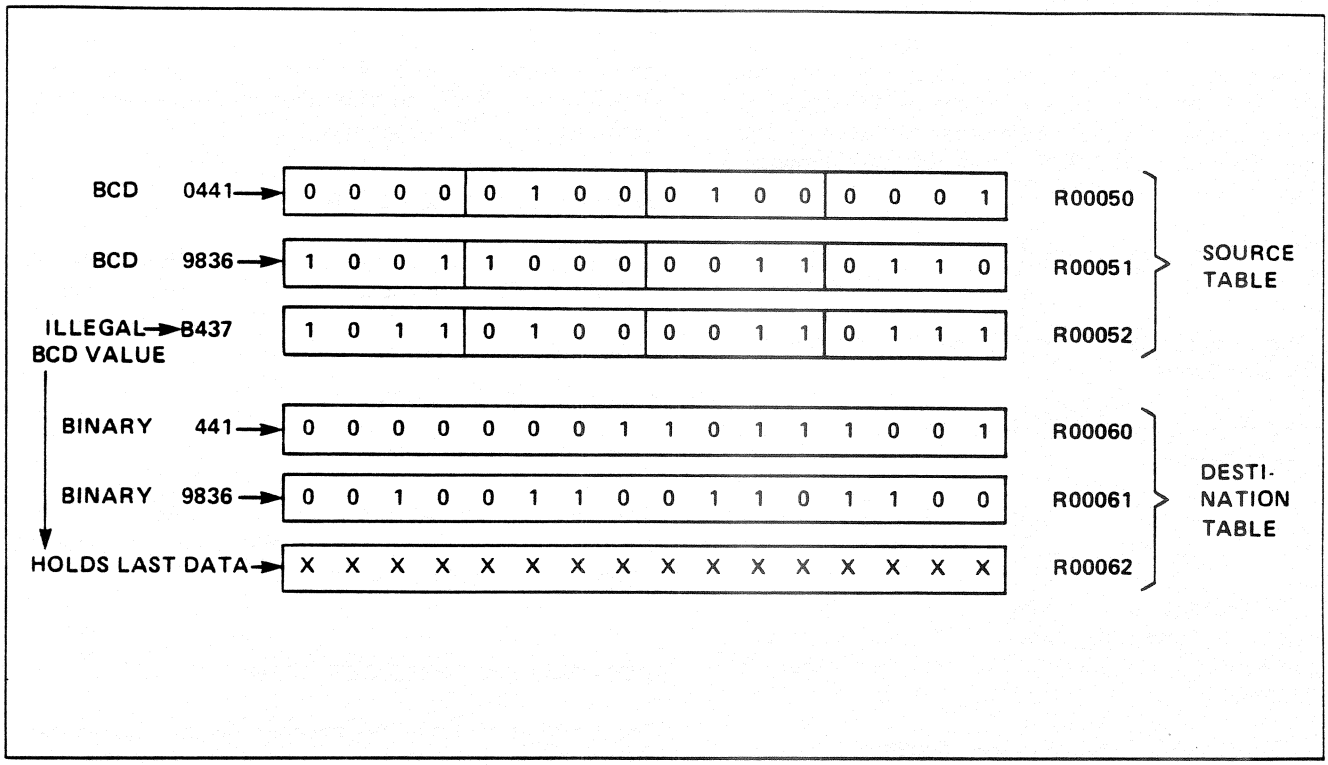


Figure BDB-2. BCD-to-Binary Table Example

The letters R or OR precede the register number on the ladder diagram display.

LENGTH

The Length is a constant value used to specify the number of registers in the Source and Destination Tables. The Length is programmed up to 32,767, although normally tables will be much smaller.

ENABLE CIRCUIT

When the enable circuit conducts, the data in the Source Table is converted from BCD to binary and stored in the Destination Register. The conversion occurs during each scan if the enable circuit is conducting. The enable circuit consists of up to 8 horizontal contacts by 7 parallel paths located to the left of the function block. The rules for programming contacts listed in Section 6 apply to this circuit.

CONVERT OUTPUT

The convert output simply follows or repeats the conducting or nonconducting state of the enable circuit.

OVERRANGE OUTPUT

The overrange output conducts whenever any BCD digit of any register in the Source Table contains a value which is not a legal BCD value (0000 thru 1001). When one or more digits of a Source Table Register are illegal, all the digits of the corresponding Destination Table Register are held at their last values. See registers R00052 and R00062 of Figure BDB-2.

Note

Outputs from the BCD-to-Binary Table function block can be left unconnected in the network.

8-11-2. APPLICATIONS

See Paragraph 8-21-2, Applications, for a typical BCD-to-Binary Table application.

8-12. SCALE (118)

The Scale programmable function converts an input range of values—referred to as a "scale"—into a second range of values, or scale. The second scale may be output for use in the ladder diagram. Figure SC-1 shows a typical circuit containing the Scale function. Observe the Figure and note the following 5 points:

1. The Data 1 Register is considered the input to the Scale function, while the Data 2 Register contains the output of the Scale function. In effect, these registers specify the source and destination of data.

2. The values of the High Limit 1 and Low Limit 1 Registers, or constants, define the high and low values input to the Data 1 Register. The High Limit 2 and Low Limit 2 Registers, or constants, define the high and low values of the Data 2 Register output. These values determine the scaling or conversion which takes place. The actual scaling, or conversion, is duplicated by the following formulas:

$$(\text{Data 1 Register Value} - \text{Low Limit 1}) \times \text{Scaling Factor} + \text{Low Limit 2} = \text{Data 2 Register Value}$$

Where: Scaling Factor is equal to:

$$\frac{\text{High Limit 2} - \text{Low Limit 2}}{\text{High Limit 1} - \text{Low Limit 1}}$$

3. The scaling operation, or conversion, from one scale of values to a second scale of values occurs during each scan if the enable circuit is conducting.

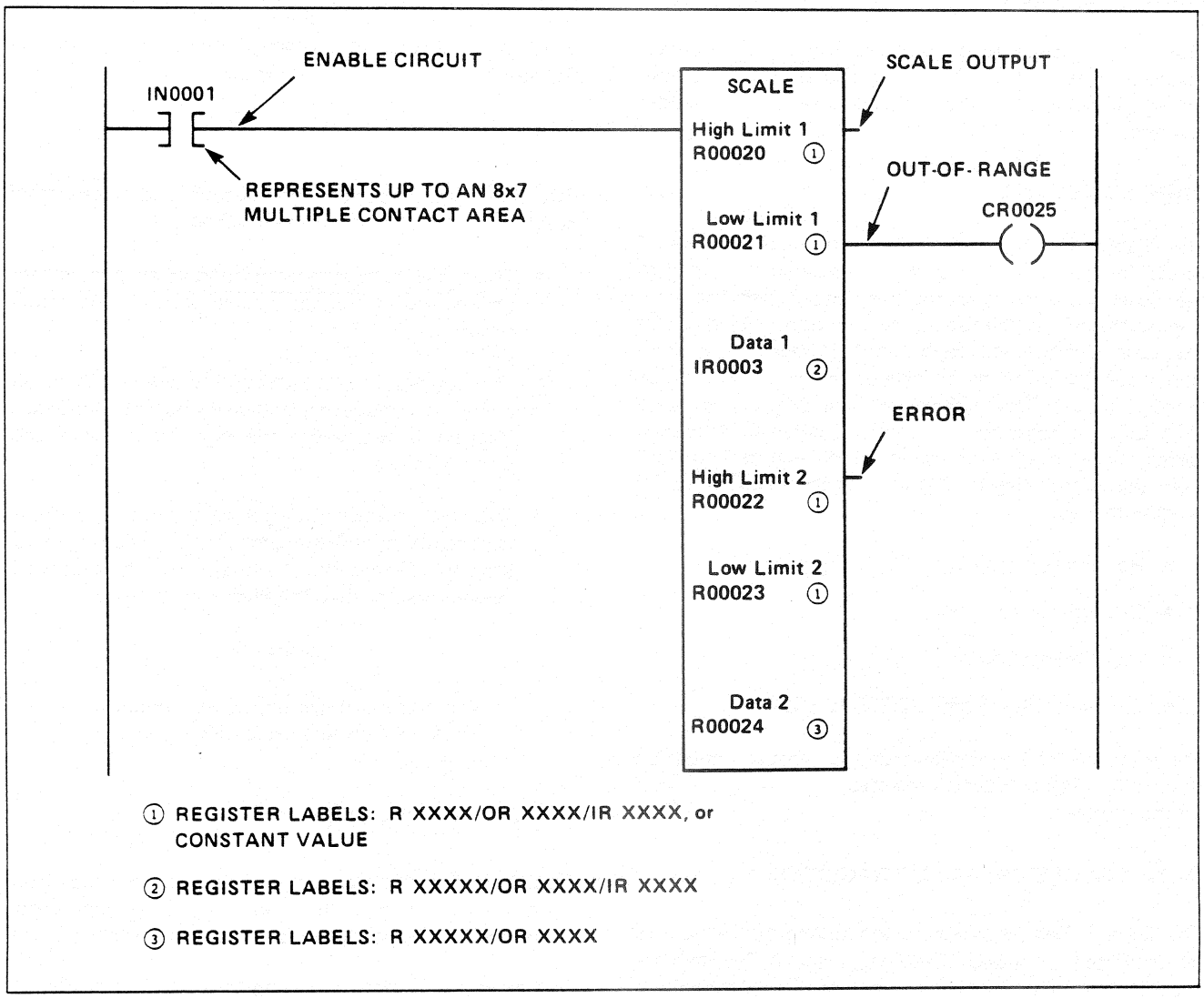


Figure SC-1. Scale Function Circuit (Typical)

4. The out-of-range output conducts whenever the contents of the Data 1 Register are not within the High and Low Limit 1 ranges.

5. The error output conducts when the enable circuit is conducting and the:

- Low Limit 1 value is greater than the High Limit 1 value, or
- Low Limit 2 value is greater than the High Limit 2 value

8-12-1. SPECIFICATIONS

The programming specifications for the Scale function are listed here.

FUNCTION BLOCK

The Scale function block requires 2 horizontal contact spaces by 6 parallel paths on the display. It can be positioned anywhere in the contact area of the ladder diagram.

HIGH, LOW LIMITS

The High and Low Limits specify the range, or scale, of values for the Data 1 Register, considered the input to the Scale function, and for the Data 2 Register, considered the scaled output of the function. The values of the Low Limit 1 and High Limit 1 Registers, or constants, specify the range or scale of values expected to be "input" to the Data 1 Register. The Low Limit 2 and High Limit 2 Registers, or constants, determine the converted range or scale of values which will be "output" from the Data 2 Register. The limits are specified by the programmer as:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Constant value from -32,768 to +32,767

The letters R, OR, or IR precede the register number on the display. If a constant is selected, no letter precedes it on the display.

DATA 1 REGISTER, DATA 2 REGISTER

The Data 1 Register specifies the source of the data to be converted, or scaled. The Data 1 and 2 Registers are specified by the programmer as any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR) (Data 1 Register, not Data 2 Register)

The letters R, OR, or IR precede the register number on the display.

The Data 2 Register specifies the location of the converted, or scaled, data output from the Scale function. The Data 2 Register is specified by the programmer as a holding register (R) or output register (OR). The letters R or OR precede the register number on the display.

ENABLE CIRCUIT

The enable circuit of the Scale function conducts, allowing the scaling to take place. This circuit can consist of up to 8 horizontal contacts by 7 parallel paths located to the left of the Scale function block on the ladder diagram. The rules for programming contacts listed in Section 6 apply to these circuits.

OUTPUTS

The scale output, out-of-range output, and error output conduct as noted in the following 3 points:

- The scale output simply follows or repeats the conducting or nonconducting state of the enable circuit.
- The out-of-range output conducts when the enable circuit is conducting and the value of the Data 1 Register is **not** within the High Limit 1 and Low Limit 1 values.
- The error output conducts when the enable circuit is conducting and the Low Limit 1 value is greater than the High Limit 1 value, or the Low Limit 2 value is greater than the High Limit 2 value.

Note

The outputs from the Scale function block can be left unconnected in the network.

8-12-2. APPLICATIONS

The Scale function can be used to convert values from various types of input modules to common engineering units such as pounds, gallons, or revolutions per minute.

For example, a certain Analog-to-Digital Input Module receives voltages in the range of 0 to 8 volts in response to a spindle speed of 0 to 10,000 rpm. The Scale

function shown in Figure SC-2 converts value (in a range from 0 to 3,276), received from the Input Module address IR0001, to a value ranging from 0 to 10,000. Thus the data is converted and can be monitored directly in rpm units..

See the Analog In function, Paragraph 7-14, for sign-magnitude conversion.

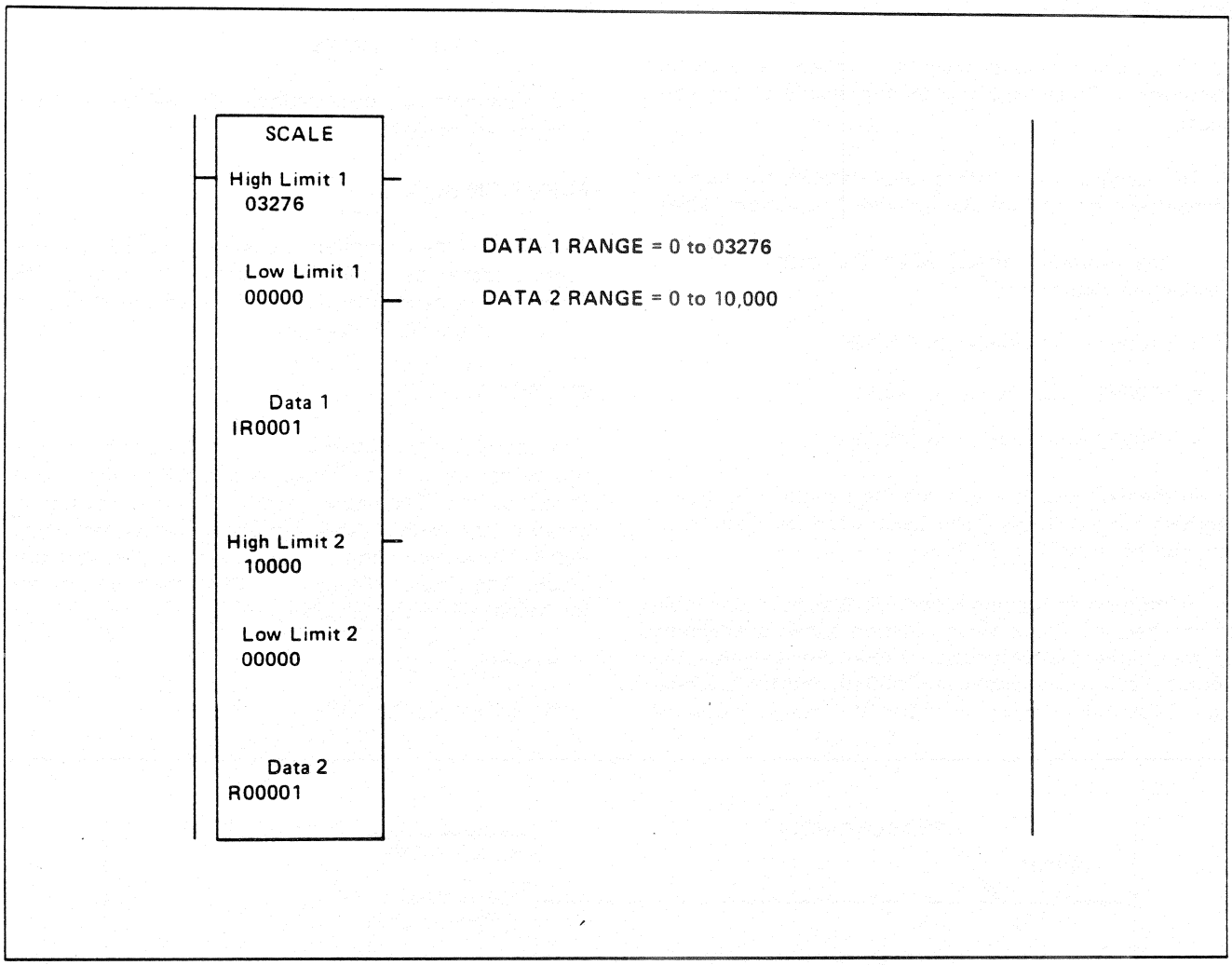


Figure SC-2. Scale Function Application

8-13. BLOCK MOVE (076)

The Block Move programmable function provides for the movement of data from one table of registers to another table of registers. Observe Figure BLM-1 and note the following 4 points:

1. The transfer of data from the Source Table to the Destination Table occurs when the enable circuit conducts.

2. The Length is a constant value defining the number of registers in both the Source and Destination Tables.

3. In the example shown, when the enable circuit is conducting, data from:

- IR0001 is transferred to R00080
- IR0002 is transferred to R00081
- IR0003 is transferred to R00083

4. In the example shown, since the Length is 3, only 3 registers are transferred. The Length can be specified as any number from 1 to 32,767.

5. In addition to moving blocks of data from one table to another, the Block Move function allows overlapping of the Source and Destination Tables. For example, the Source Table could begin at R00010, and the Destination Table could begin at R00012. This selection of registers shifts data 2 registers up when the enable circuit conducts. Also the Source Register could be larger than the Destination Register, in which case the data would be shifted in the opposite direction of the example just described.

8-13-1. SPECIFICATIONS

The programming specifications for the Block Move function are listed here.

FUNCTION BLOCK

The Block Move function block requires 2 horizontal contact spaces by 4 parallel paths on the display. The function can be positioned anywhere on the 10x7 contact area of the ladder diagram.

SOURCE TABLE

The Source Table consists of a specified number of registers or groups used to obtain data which is then transferred to the Destination Table. The first register or group in the Source Table is programmed as part of the Block Move function and displayed in the function block. The Source Table is selected from any of the following registers or groups:

- Holding register (R)
- Output register (OR)

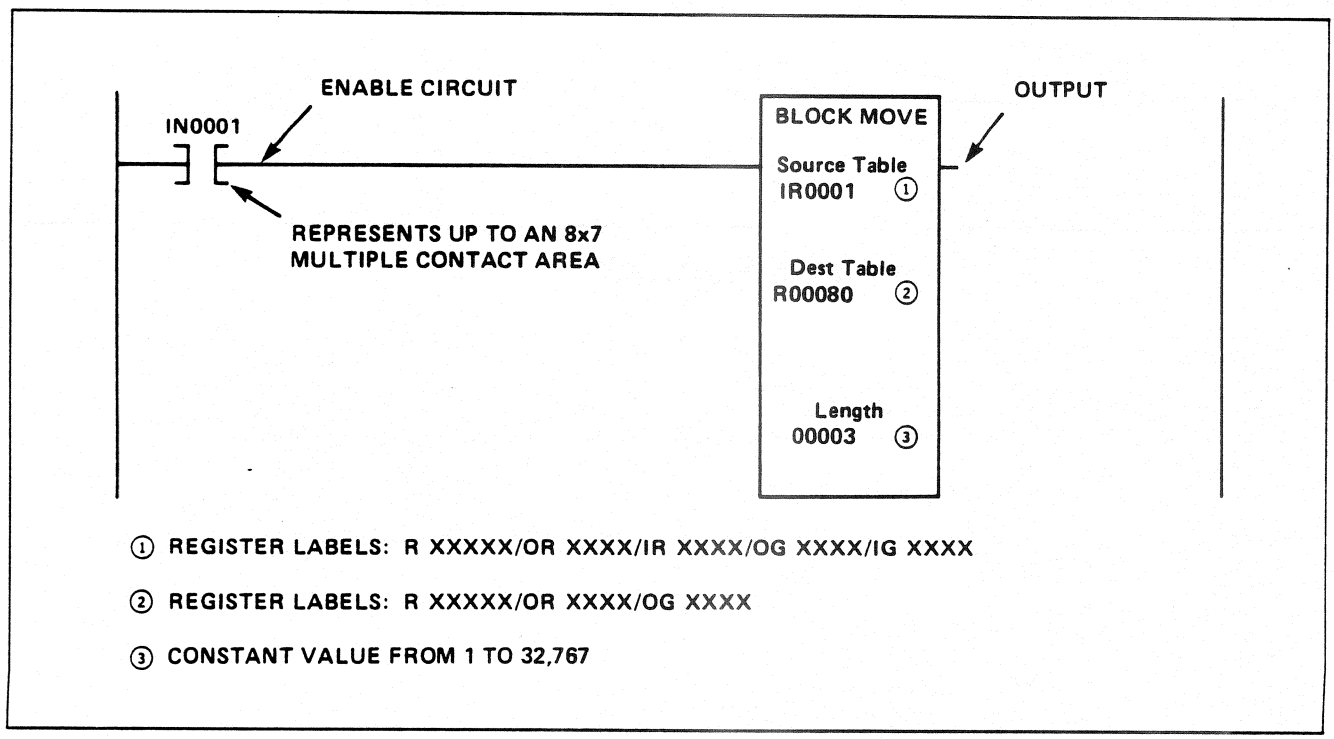


Figure BLM-1. Move Block Function Circuit (Typical)

- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the display of the Source Table register or group number.

DESTINATION TABLE

The Destination Table consists of a number of registers or groups used to store data transferred from the Source Table. The first register or group in the Destination Table is programmed as part of the Block Move function and displayed in the function block. The Destination Table is selected from any of the following registers or group:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR, or OG precede the register or group number in the display of the function block.

WARNING

THE USE OF AN OUTPUT GROUP FOR THE DESTINATION REGISTER OF THE BLOCK MOVE FUNCTION OVERRIDES EXISTING FORCE CONDITIONS ON THE OUTPUTS ASSOCIATED WITH THE OUTPUT GROUP. THIS CAN RESULT IN UNEXPECTED MACHINE MOVEMENTS OR PROCESS OPERATIONS, WHICH MAY CAUSE PERSONNEL INJURY AND/OR EQUIPMENT DAMAGE.

LENGTH

The Length is a constant value used to define the number of registers in the Source and Destination Tables. The Length is specified as any value from 1 to 32,767.

ENABLE CIRCUIT

The enable circuit, when conducting, allows the data from the Source Table to be transferred to the Destination Table during each scan of the ladder diagram. This circuit can consist of up to 8 horizontal contacts by 7 parallel paths located to the left of the Block Move function block on the ladder diagram. The rules for programming contacts listed in Section 6 apply to these circuits.

Note

When the enable circuit first conducts and overlapped Tables (as described in item 5 of the Block Move description) are used, the function determines which direction the data is to be shifted in the Tables.

OUTPUT

The output from the Block Move function block simply follows or repeats the conducting or nonconducting state of the enable circuit.

Note

The output from the Block Move function block can be left unconnected in the network.

8-13-2. APPLICATIONS

The Block Move function facilitates the movement of relatively large blocks of data from one location of processor's memory to another.

In the example shown in Figure BLM-2, 4 different tables, representing the energized or de-energized state of outputs for 4 different modes, are compared to an "actual state" table. If the outputs are not correct, deviations are placed in an error table. Figure BLM-3 shows the ladder diagram used to implement this application. Observe the Figure and note the following 4 points:

1. Inputs IN0001 thru IN0004 enable one of the 4 Block Move functions at any given time. The Destination Table of each Block Move function transfers data from a different set of 4 holding registers to R00080 thru R00084.

2. The XOR Table-to-Table function (102) compares the data from the active Block Move function to the actual outputs OG0001 thru OG0004.

3. The Search 1's function (079) enables output CR0075 if any of the outputs does not match the data obtained from the Block Move function currently supplying data to R00080. The Pointer Register R00089 contains the bit number of the comparison which did not match. This bit number could be transferred to another register for storage, or to an LED display for maintenance purposes.

4. Output CR0075 is enabled and latched if the comparison does not match. IN0006 is used to reset output CR0075.

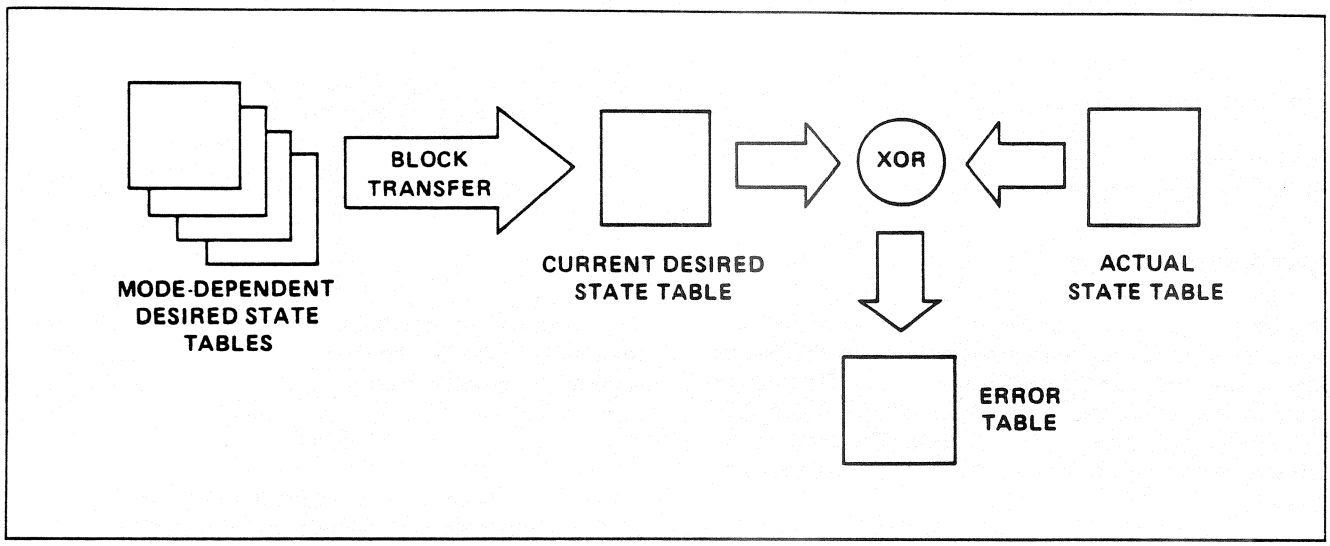


Figure BLM-2. Monitoring Outputs (Block Diagram)

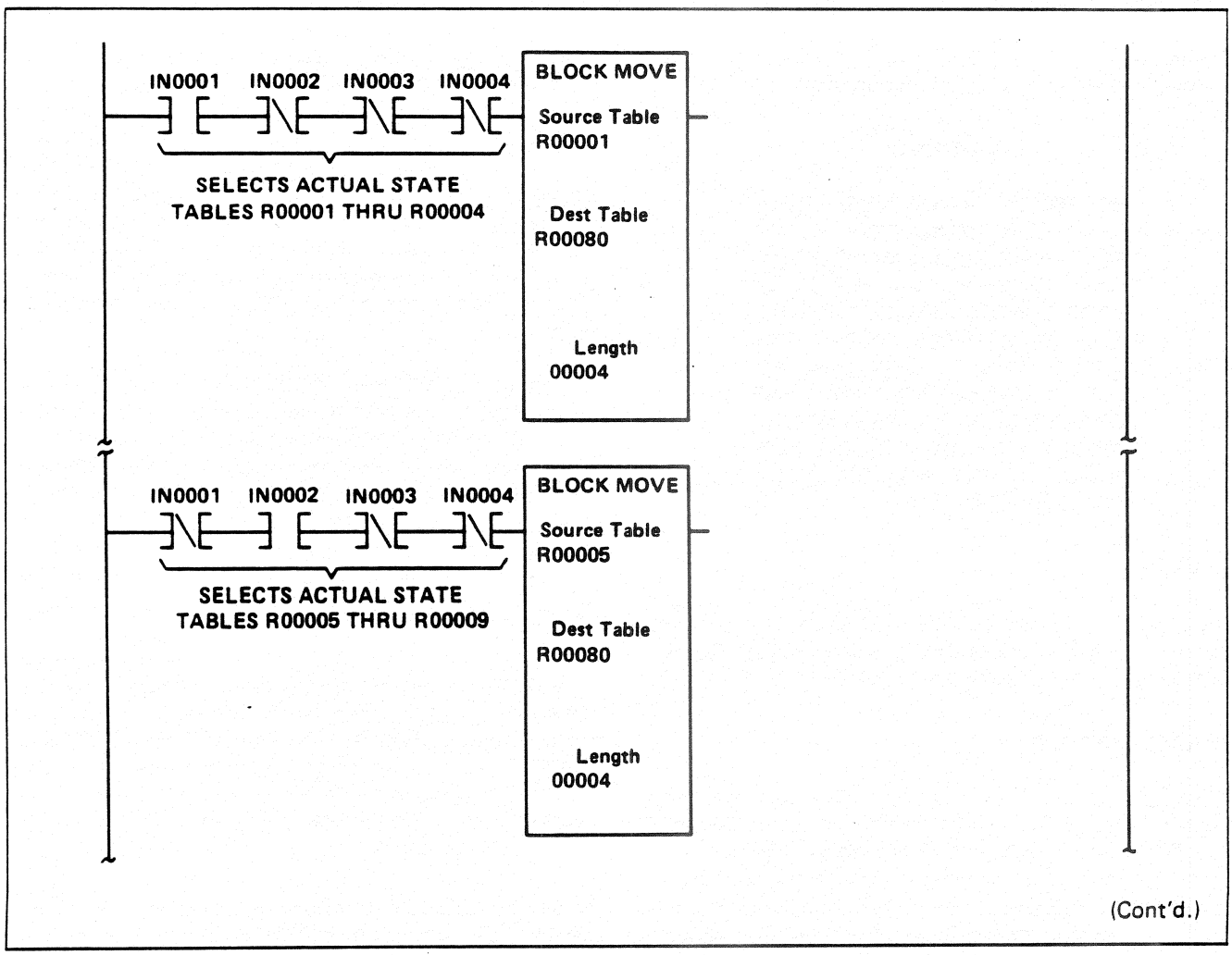


Figure BLM-3. Monitoring Outputs Application

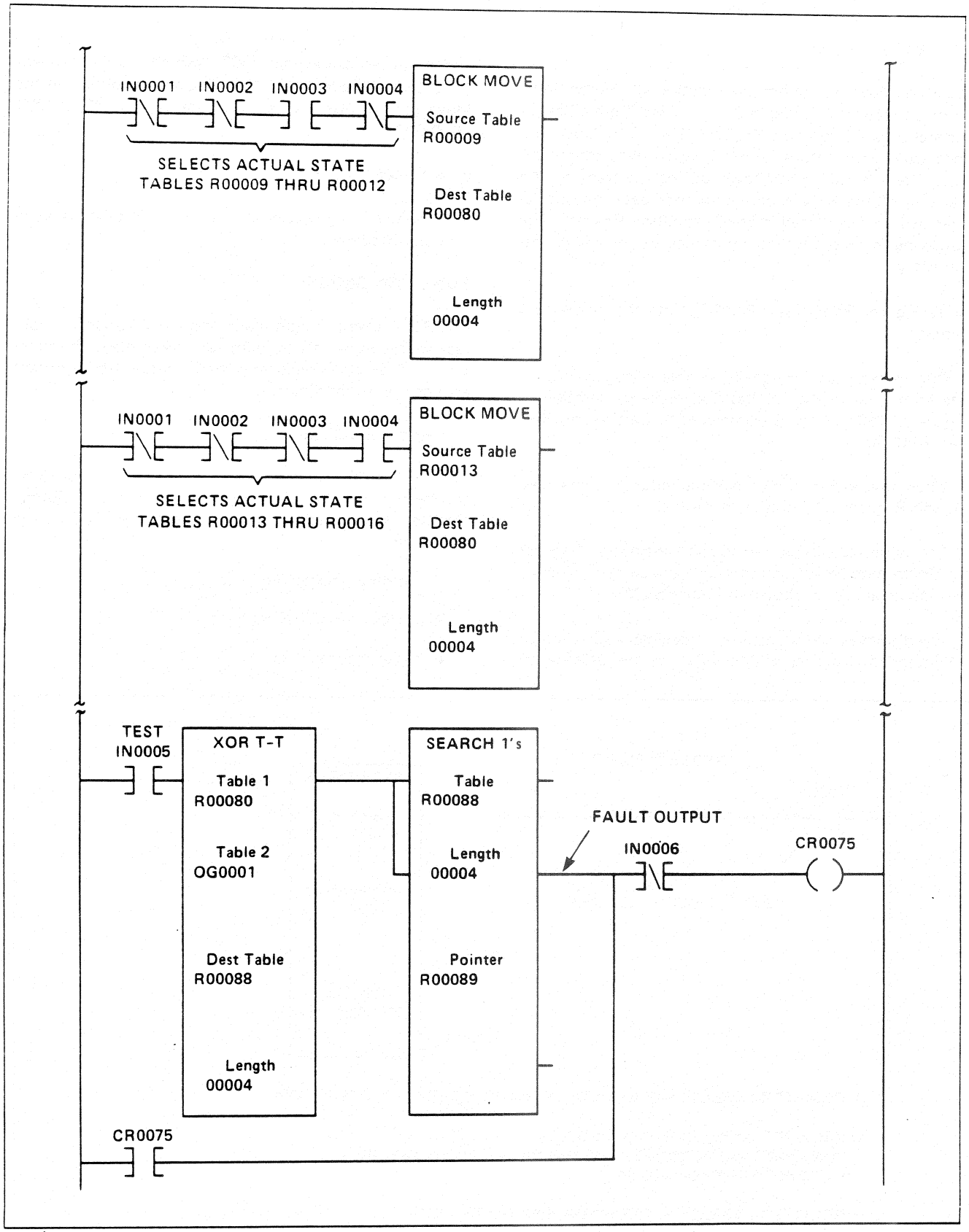


Figure BLM-3. (Cont'd.)

8-14. BYTE MOVE (077)

The Byte Move programmable function allows the transfer of bit states contained in a single word from a Source Register to a Destination Register. The 16 bits within a word are divided into 4 "bytes." (Each byte contains 4 bits.) The function also allows, during the transfer, the changing of the byte ordering so that data may be reorganized as it is placed in the Destination Register. The reordering is accomplished by means of the Move Pattern entry.

Study Figures BYM-1 and BYM-2. Note the following 4 points:

1. During each scan, when the enable circuit conducts, data contained in the 4 bytes that make up the Source Register, IR0011 here, is transferred to Destination Register, R00080.

2. When the enable circuit is nonconducting, the Destination Register is held at its last value.

3. The value actually stored in the Destination Register is a binary number which represents the current states of the individual bits, as shown in Figure BYM-2.

4. The displayed Move Pattern is a decimal equivalent of the intended ordering of the bytes in the Destination Register. For Figure BYM-2, the Move Pattern is entered and displayed as decimal 12577, which is the equivalent of BCD or hexadecimal 3121. The actual content of the Move Pattern Register is, of course, binary: 0011 0001 0010 0001.

8-14-1. SPECIFICATIONS

The programming specifications for the Byte Move function are listed here.

FUNCTION BLOCK

The Byte Move function block requires 2 horizontal contact spaces by 4 parallel paths on the display. The function can be positioned anywhere in the 10x7 contact area of the ladder diagram.

SOURCE REGISTER

The Source Register contains the 4 bytes of data to be transferred to the Destination Register. The Source Register is specified by the programmer as one of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)

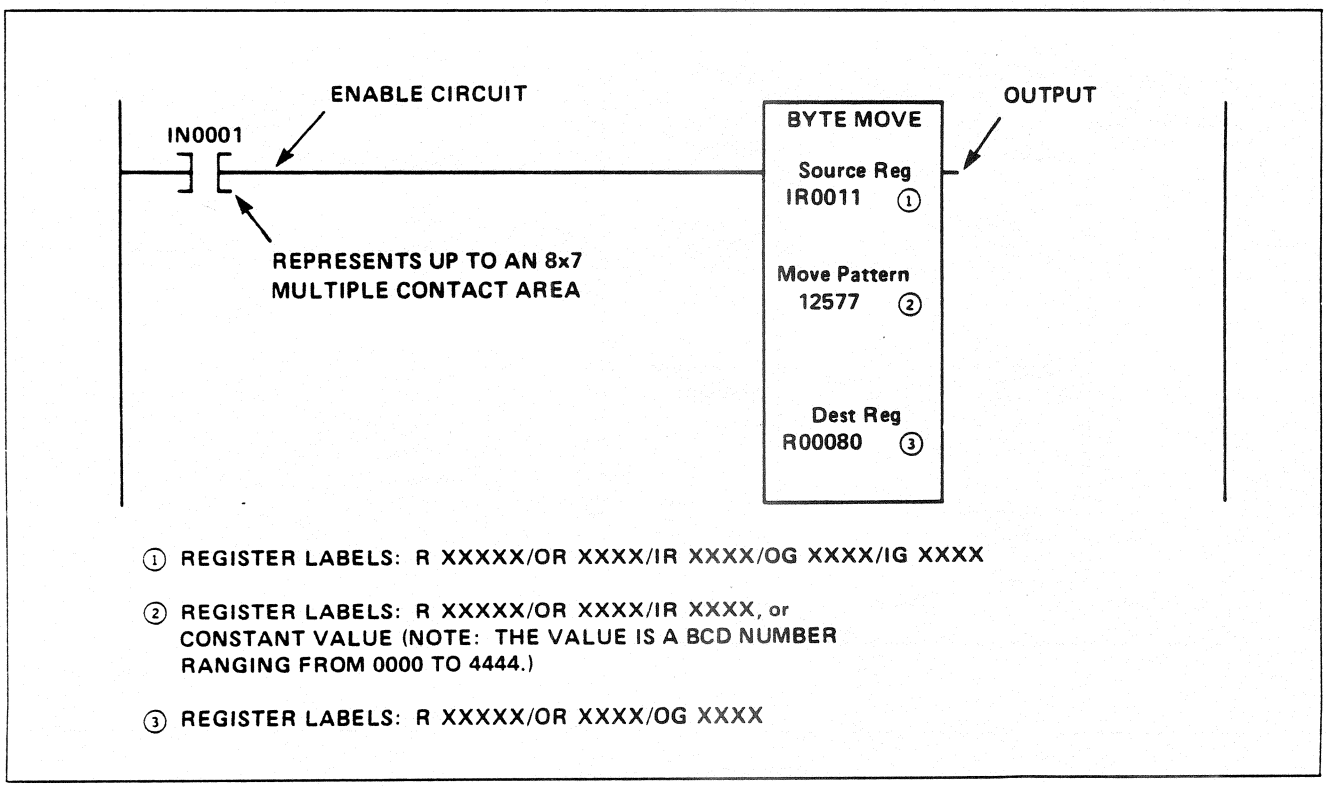


Figure BYM-1. Byte Move Function Circuit (Typical)

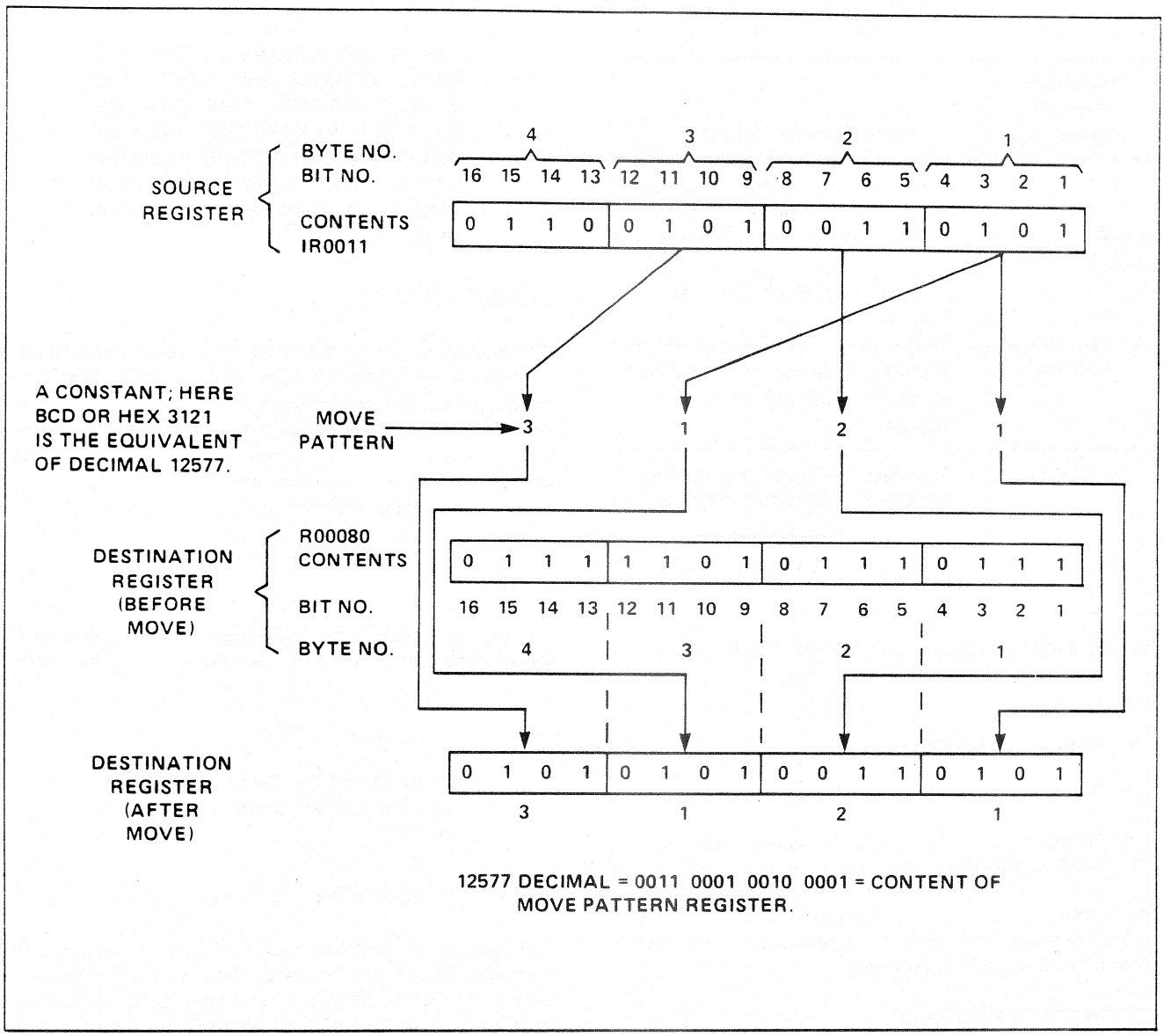


Figure BYM-2. Byte Move Example

- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the display of the Source Table register or group number.

MOVE PATTERN

The Move Pattern defines the reordering that takes place as the 4 bytes are transferred from the Source Register to the Destination Register. The Move Pattern is entered by the programmer and displayed as a decimal number. The Move Pattern represents the reordering of the Source Register's byte number ordering (4321).

When the reordering of all 4 bytes is desired, the range of available BCD numbers is 1111 thru 4444. If it is not desired to change a value during the transfer, the digit 0 is used in one or more places. (Data in the corresponding byte of the Destination Register is held at its last value.) Should a digit above 4 be used, values in the Destination Register's corresponding byte are also held in their last states. As shown in Figure BYM-2, if identical data is required in 2 or more bytes of the Destination Register, a digit may be reused as, for example, in the Move Pattern (3121).

Table BYM-1 relates the 4 byte positions that make up the BCD Move Pattern number.

TABLE BYM-1. MOVE PATTERN BYTE TRANSFER

Destination Register Position ①	Byte Transfer Initiated
MSD	Transfers byte number specified to MSD of Destination Register.
Second MSD	Transfers byte number specified to second MSD of Destination Register.
Third MSD	Transfers byte number specified to third MSD of Destination Register.
LSD	Transfers byte number specified to LSD of Destination Register.
① MSD = most significant digit; LSD = least significant digit.	

The Move Pattern is specified as any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Constant value, being a BCD value in the range of 0000 to 4444

The letters R, OR, or IR precede the display of the register number. If a constant is selected, no letter precedes the display of the constant.

DESTINATION REGISTER

The Destination Register specifies the register location used to receive the data transferred from the Source Register. It is specified by the programmer as one of the following:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR, or OG precede the register or group number on the display.

WARNING

THE USE OF AN OUTPUT GROUP FOR THE DESTINATION REGISTER OF THE BYTE MOVE FUNCTION OVERRIDES EXISTING FORCE CONDITIONS ON THE OUTPUTS ASSOCIATED WITH THE OUTPUT GROUP. THIS CAN RESULT IN UNEXPECTED MACHINE MOVEMENTS OR PROCESS OPERATIONS, WHICH MAY CAUSE PERSONNEL INJURY AND/OR EQUIPMENT DAMAGE.

ENABLE CIRCUIT

The enable circuit of the Move Byte function conducts, thereby enabling the transfer of the bytes. When the enable circuit is nonconducting, the data in the Destination Register is held at its last value since no transfer occurs. The enable circuit consists of up to 8 horizontal contact spaces by 7 parallel paths on the display. The rules for programming contacts listed in Section 6 apply to these circuits.

OUTPUT

The output of the Byte Move function simply follows or repeats the conducting or nonconducting state of the enable circuit.

Note

The output of the Byte Move function block can be left unconnected in the network.

8-14-2. APPLICATIONS

The Byte Move function is useful anytime it is desirable to manipulate bytes of data. One such application is shown in Figure BYM-3 where data is being received from input register IR0005, which is the Source Register. Four bytes of data are contained in it, and it is intended to place 2 bytes in one Destination Register and 2 bytes in another. The Byte Move function separates the bytes and stores them in individual holding registers, here R00090 and R00091. Observe Figure BYM-3 and note the following 2 points:

1. The Move Pattern for the upper Byte Move function is a constant 33 decimal, or 0021 BCD. This entry moves the data in IR0005, bits 1 thru 8, to R00090, bits 1 thru 8. (The entry does not initiate a reordering.) At this time the data in IR0005, bits 9 thru 16, is **not** transferred since 00 was part of the entry that corresponds to their positions.

2. The Move Pattern for the lower Byte Move function is a constant 67 decimal, or 0043 BCD. This entry moves the data in IR0005, bits 9 thru 16, to R00091, bits 1 thru 8.

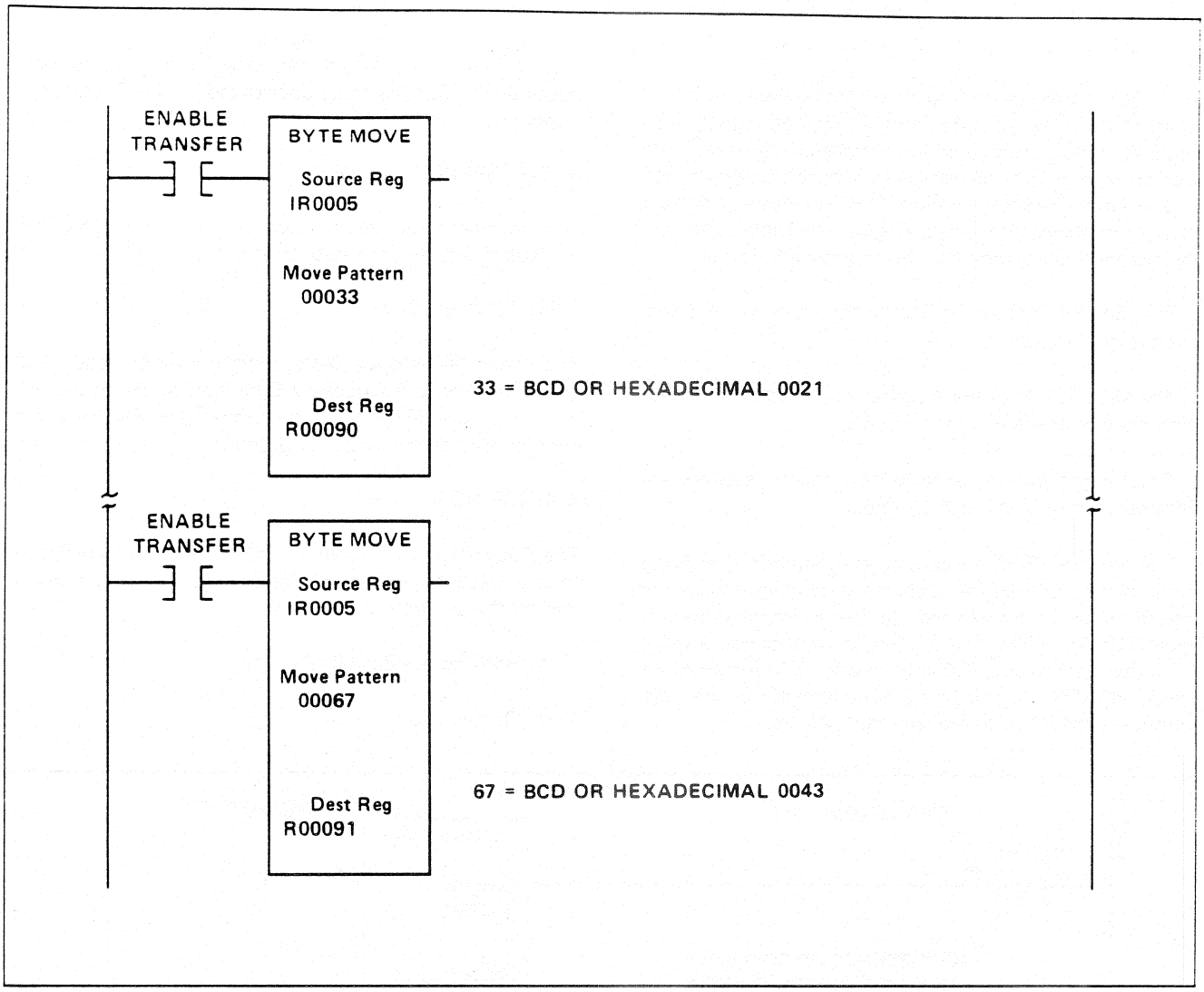


Figure BYM-3. Byte Move Application

8-15. MOVE REGISTER-TO-TABLE (094)

The Move Register-to-Table programmable function allows the transfer of data from a specified register into a single register contained in a table of registers. The location into which the data is transferred is determined by a Pointer Register. Figure MRT-1 shows a typical circuit containing the Move Register-to-Table function. Observe the Figure and note the following 5 points:

1. The Source Register contains the data to be transferred into the Table.

2. The Destination Table Register specifies the first register or group contained in the Table.

3. The Length defines exactly how many registers are contained in the Destination Table.

4. The Pointer Register contains a number specifying where in the Destination Table the data from the Source Register is to be transferred. In the example shown in Figure MRT-1, if the Pointer Register contained a value of 1, the data from IR0001 would be transferred to register R00120. If the Pointer contained a 2, the data would be transferred to register R00121, etc.

5. Data is transferred during each scan if the enable circuit is conducting. When the enable circuit is nonconducting, the data remains unchanged in the Destination Table.

8-15-1. SPECIFICATIONS

The programming specifications for the Move Register-to-Table function are listed here.

FUNCTION BLOCK

The Move Register-to-Table function block requires 2 horizontal contact spaces by 5 parallel paths on the display. The function block can be located anywhere in the contact area of the ladder diagram.

SOURCE REGISTER

The Source Register contains the data to be transferred to the Destination Table. The Source Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)

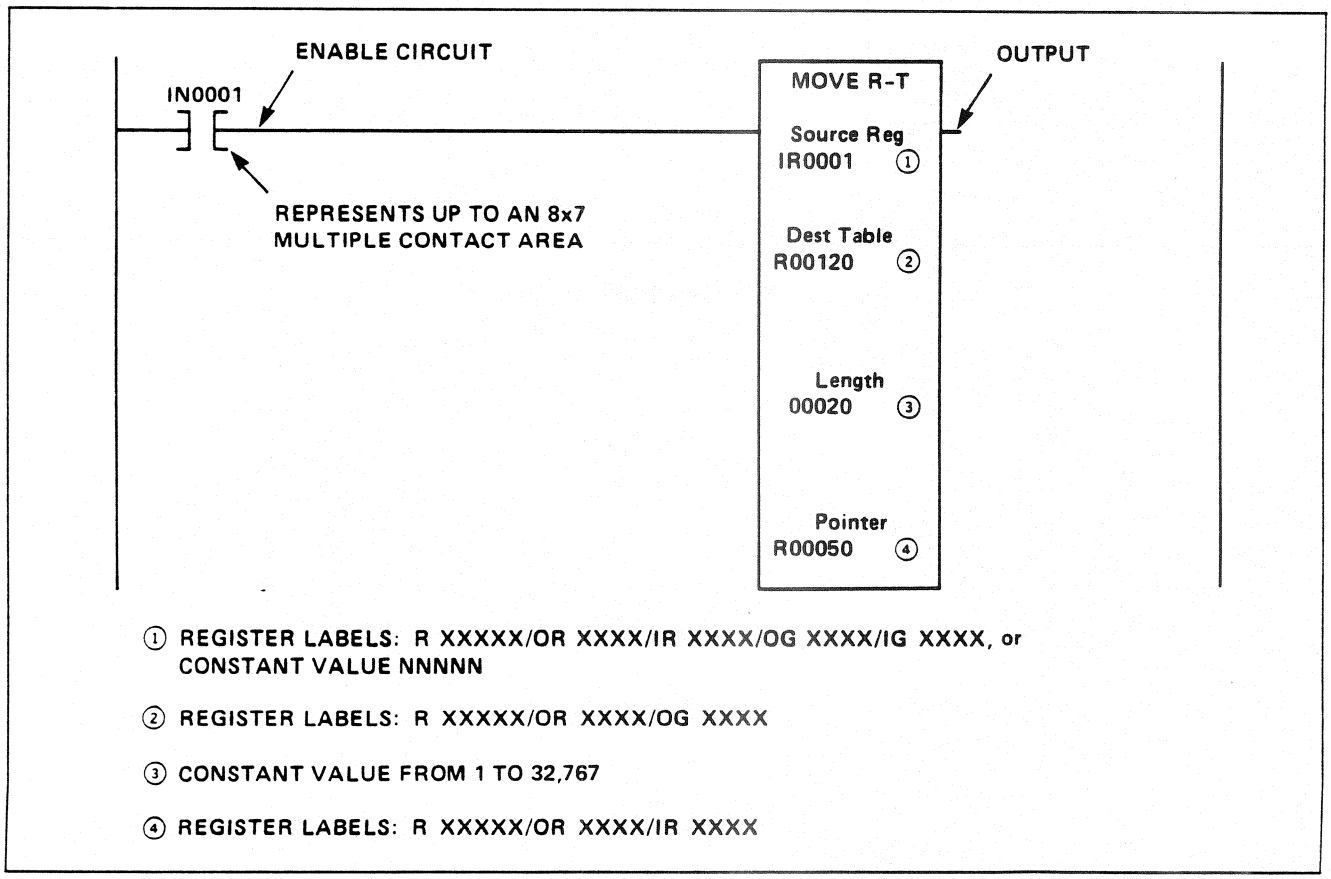


Figure MRT-1. Move Register-to-Table Circuit (Typical)

- Input register (IR)
- Output group (OG)
- Input group (IG)
- Constant value in the range of from -32,768 to +32,767

The letters R, OR, IR, OG, or IG precede the display of the register or group number. If a constant is selected, no letter precedes the display of the constant value.

DESTINATION TABLE REGISTER

The Destination Table Register or group, specified by the programmer and displayed in the function block, is the first register or group number that makes up the table of registers. The Destination Table is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR, or OG precede the ladder diagram display of the Destination Table number.

WARNING

THE USE OF AN OUTPUT GROUP FOR THE DESTINATION REGISTER OF THE MOVE REGISTER-TO-TABLE FUNCTION OVERRIDES EXISTING FORCE CONDITIONS ON THE OUTPUTS ASSOCIATED WITH THE OUTPUT GROUP. THIS CAN RESULT IN UNEXPECTED MACHINE MOVEMENTS OR PROCESS OPERATIONS, WHICH MAY CAUSE PERSONNEL INJURY AND/OR EQUIPMENT DAMAGE.

LENGTH

The Length defines the number of registers or groups in the Destination Table. The Length is specified by the programmer as a constant value in the range from 1 to 32,767.

POINTER REGISTER

The Pointer Register directs the relocation of the data being transferred. The value of the Pointer Register ranges from 1 to 32,767, where: a 1 directs the transfer to the first location of the Destination Table; a 2 directs the transfer to the second location of the Destination Table; etc. The value of the Pointer Register is controlled (incremented and/or decremented) external to the Register-to-Table function. If the value of the Pointer Register is 0, any negative number, or a number larger than the Table Length, no data is transferred. Also any data currently in the specified register is held at its last value.

The Pointer Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the display of the Pointer Register number.

ENABLE CIRCUIT

The transfer of data from the Source Register to the Destination Table occurs during each scan if the enable circuit is conducting. When the enable circuit is nonconducting, no data is transferred. The enable circuit is located on the ladder diagram to the left of the function block. The circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths. (This assumes the function block is positioned on the far right of the display.) The rules for programming contacts listed in Section 6 apply to the enable circuit.

OUTPUT

The output of the Move Register-to-Table function simply repeats or follows the conducting or nonconducting state of the enable circuit.

Note

The output of the Move Register-to-Table function block can be left unconnected in the network.

8-15-2. APPLICATIONS

The Move Register-to-Table function can be used to load "step data" associated with the various steps of the Drum Controller function (096). The step data is subsequently output from the Source Table of the Drum Controller function during the normal operation of the Drum Controller function. Figure MRT-2 shows one method of loading a table for a Drum Controller function. Observe the Figure and note the following 4 points:

1. The table of values used by the Drum Controller

function is contained in holding registers R00100 thru R00119.

2. The register in the table is loaded with data from IR0001, the Source Register of the Move Register-to-Table function, when pushbutton IN0001 is closed.

3. Input register IR0002, the Pointer Register for the Move Register-to-Table function, specifies which register in the table will be loaded with the data from IR0001. This number must be in the range from 1 to 20 to load holding registers R00100 thru R00119, respectively. (It is stored in binary format.)

4. Input IN0002 disables the Drum Controller function during the loading of the Source Table.

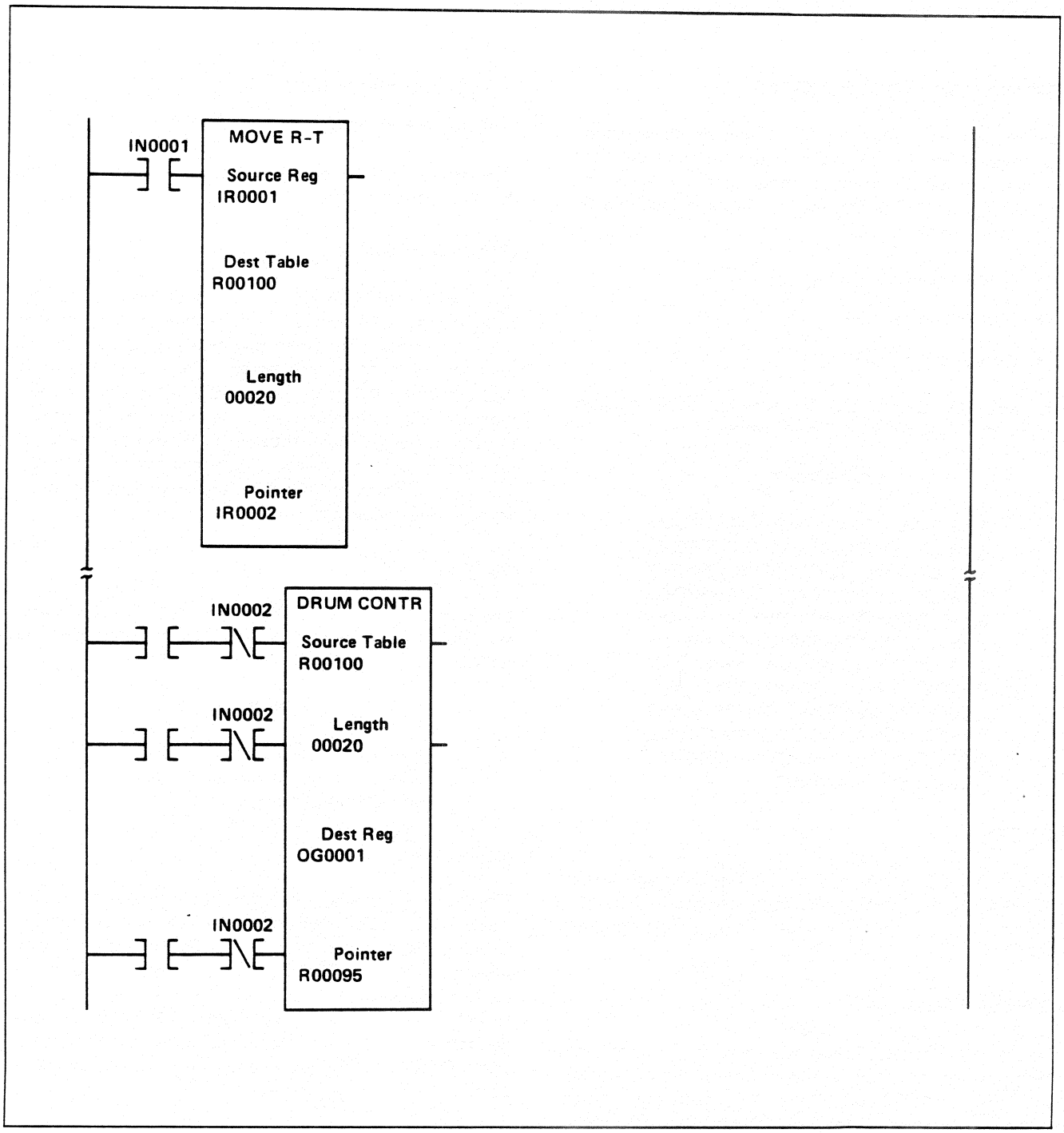


Figure MRT-2. Move Register-to-Table Application

8-16. MOVE TABLE-TO-REGISTER (095)

The Move Table-to-Register programmable function allows the transfer of data from a single register contained in a table of registers into another specified register. The location from which the data is transferred is determined by the Pointer Register. Figure MTR-1 shows a typical circuit containing the Move Table-to-Register function. Observe the Figure and note the following 3 points:

1. The Source Table contains the data to be transferred. The data transfer is directed by the Pointer Register. For example, if the Pointer Register contains a value of 2, the second register from the Source Table (R00081 in this example) is transferred to OR0005.

2. The Destination Register receives the data from a register in the Source Table during each scan if the enable circuit is conducting.

3. If the enable circuit is nonconducting, or if the value of the Pointer Register is 0, or is larger than the table Length, no data transfer occurs. (In Figure MTR-1, 21 would be larger than the specified Length.)

8-16-1. SPECIFICATIONS

The programming specifications for the Move Table-to-Register function are listed here.

FUNCTION BLOCK

The Move Table-to-Register function block requires 2 horizontal contact spaces by 5 parallel paths on the display. The function block can be located anywhere in the contact area of the ladder diagram.

SOURCE TABLE REGISTER

The Source Table Register contains, in a single register, the data to be transferred to the Destination Register as directed by the Pointer Register. The Source Table Register indicates the starting location of the entire table. This Register is displayed in the function block and specified by the programmer as a:

- Holding register (R)
- Output register (OR)

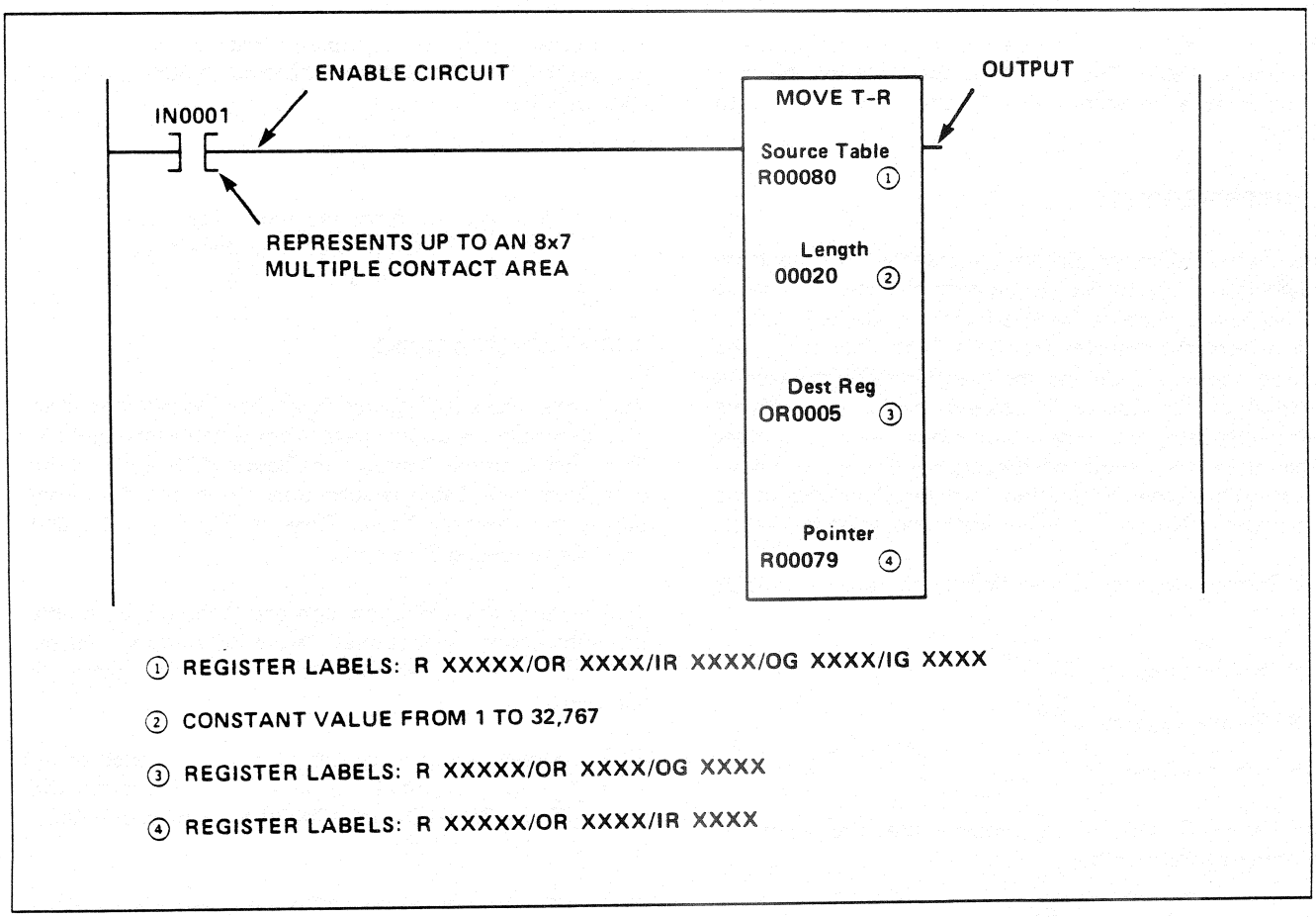


Figure MTR-1. Move Table-to-Register Function (Typical)

- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the register number on the display.

WARNING

THE USE OF AN OUTPUT GROUP FOR THE DESTINATION REGISTER OF THE MOVE TABLE-TO-REGISTER FUNCTION OVERRIDES EXISTING FORCE CONDITIONS ON THE OUTPUTS ASSOCIATED WITH THE OUTPUT GROUP. THIS CAN RESULT IN UNEXPECTED MACHINE MOVEMENTS OR PROCESS OPERATIONS, WHICH MAY CAUSE PERSONNEL INJURY AND/OR EQUIPMENT DAMAGE.

LENGTH

The Length defines the number of registers or groups in the Source Table. The Length is specified by the programmer as a constant value in the range from 1 to 32,767.

POINTER REGISTER

The Pointer Register directs the transfer of data from the Source Table to the Destination Register. The value of the Pointer Register ranges from 1 to 32,767, where: a 1 directs the transfer from the first location of the Source Table; a 2 directs the transfer from the second location of the Source Table, etc. If the value of the Pointer Register is 0, any negative number, or a number larger than the Length of the Source Table, no data is transferred to the Destination Register. Any data in the Destination Register remains unchanged at its last value.

The Pointer Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the display of the Pointer Register number.

DESTINATION REGISTER

The Destination Register is the register used to receive the data transferred from the Source Table. The Destination Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR, or OG precede the ladder diagram display of the Destination Register number.

ENABLE CIRCUIT

The transfer of data from the Source Table to the Destination Register occurs during each scan when the enable circuit is conducting. The enable circuit is located on the ladder diagram to the left of the function block. The circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths. (This assumes the function block is positioned on the far right of the display.) The rules for programming contacts listed in Section 6 apply to the enable circuit.

OUTPUT

The output of the Move Table-to-Register function simply repeats or follows the conducting or nonconducting state of the enable circuit.

Note

The output from the Move Table-to-Register function block can be left unconnected in the network.

8-16-2. APPLICATIONS

The Move Table-to-Register function is useful whenever it is desirable to obtain data from a table of registers. One such example is shown in Figure MTR-2 where the data from one Table is subtracted from the data contained in a second Table. Observe Figure MTR-2 and note the following 6 points:

1. The enable subtraction contacts conduct to enable the subtraction of 2 tables. Bit 1 of holding register R00001 is set and remains set until the subtraction is completed.

2. The Move Table-to-Register functions are enabled by bit 1 of holding register 1, causing the first register in the Source Table R00220 and R00230 to be placed in registers R00040 and R00041.

3. The Subtract function (070) subtracts the contents of R00041 from R00040 and places the result in R00042. The Move R-T function on the right places the

result in a Table beginning with register R00240.

4. The Up Counter function is incremented by 1. The actual value contained in R00019 now contains a 1, and, since the counter has not reached the preset value the Jump Coil causes the ladder diagram to loop, or jump, back to the Jump Label function (023).

5. The ladder diagram scan continues to loop, or jump, back until all 10 registers in the tables have been subtracted. During each loop, the Actual Register of the Up Counter function (038) is incremented by 1. Since the Actual Register of the counter (R00019) is the same register used as the Pointer Register in all the Move Table functions, all 10 registers from Table R00230 are subtracted from R00220 when the looping is complete. The Move Register-to-Table function loads the difference in the Table consisting of registers R00240 thru R00249.

6. When all the registers have been subtracted, the Q output of the Up Counter function conducts, enabling the BF/R coil and turning on bit 3 of holding register 1. This is examined in the enable circuit of the Jump Coil where the Jump Coil will be de-energized, stopping the looping action and allowing the scan of the ladder diagram to continue.

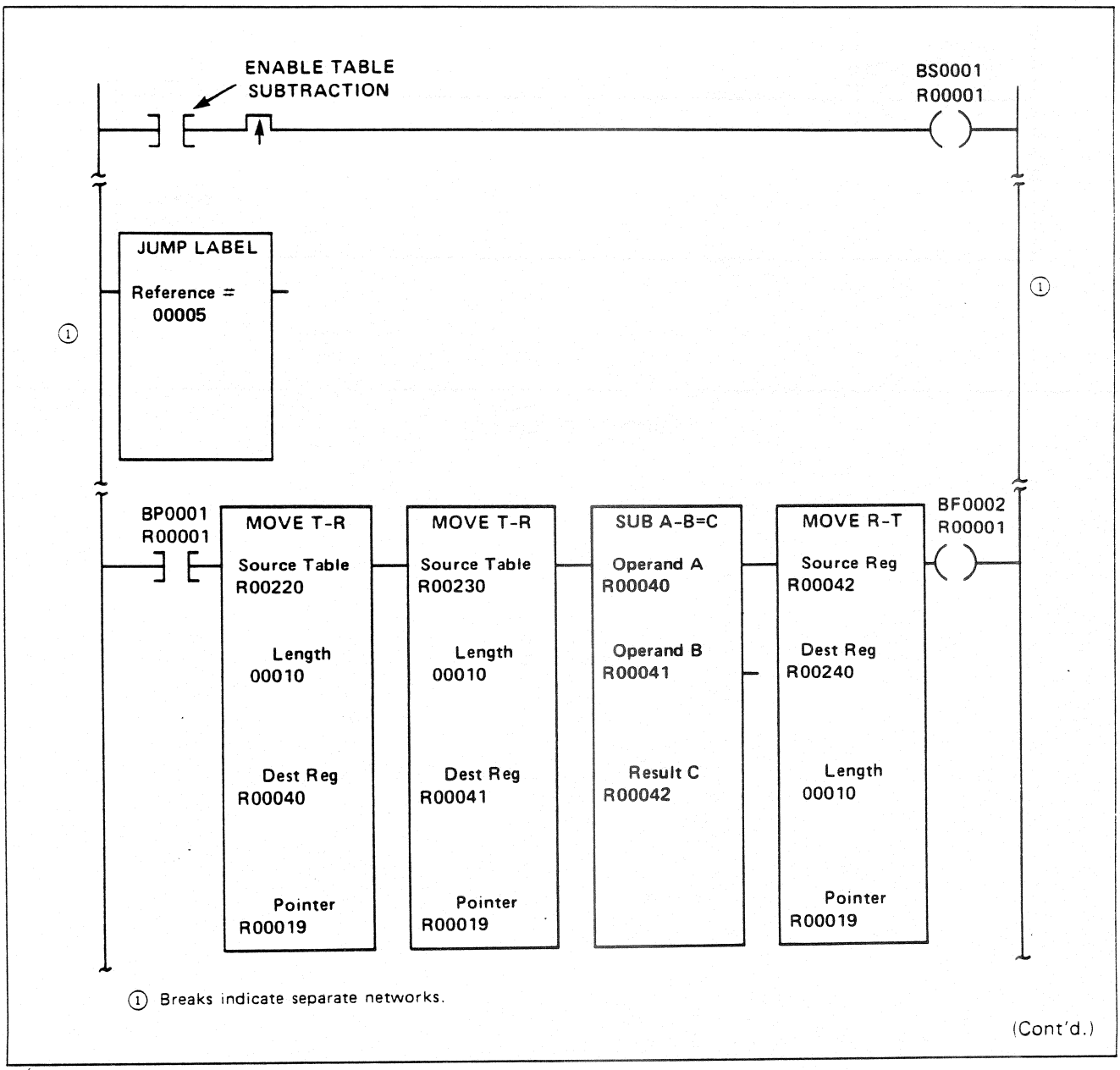


Figure MTR-2. Table-Subtraction Application

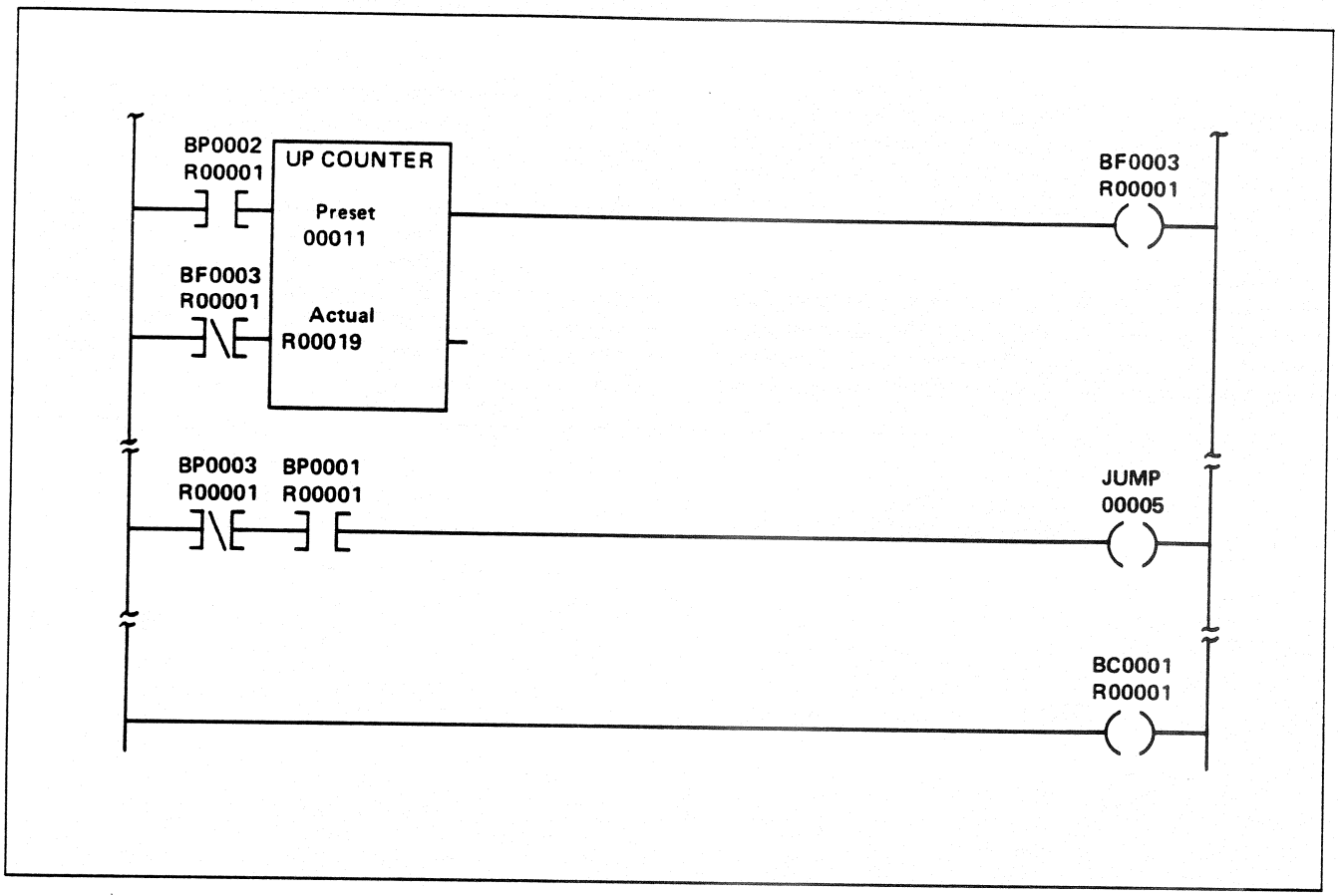


Figure MTR-2. (Cont'd.)

8-17. N BIT MOVE (119)

The N Bit Move programmable function allows the transfer of a specified number of bits from one table of registers to another table of registers. Figure NB-1 shows a typical circuit containing the N Bit Move function. Observe the Figure and note the following 4 points:

1. The first registers in both the Source and Destination Tables are displayed in the function block. The Length is a constant which defines the length or number of bits to be transferred between the Tables.

2. The specified number of bits is transferred from the Source Table to the Destination Table during each scan if the enable circuit conducts.

3. The Source Pointer and Destination Pointer Registers direct the transfer of data. The binary number contained in these Pointer Registers is the "bit number" of the first bit of the group of bits to be transferred between the Tables.

4. The total number of bits transferred is specified in the = of Bits entry. See Figure NB-2 and note the bit transfer in the example shown where bits 20 thru 27 of the Source Table are transmitted to bits 33 thru 40 of the Destination Table.

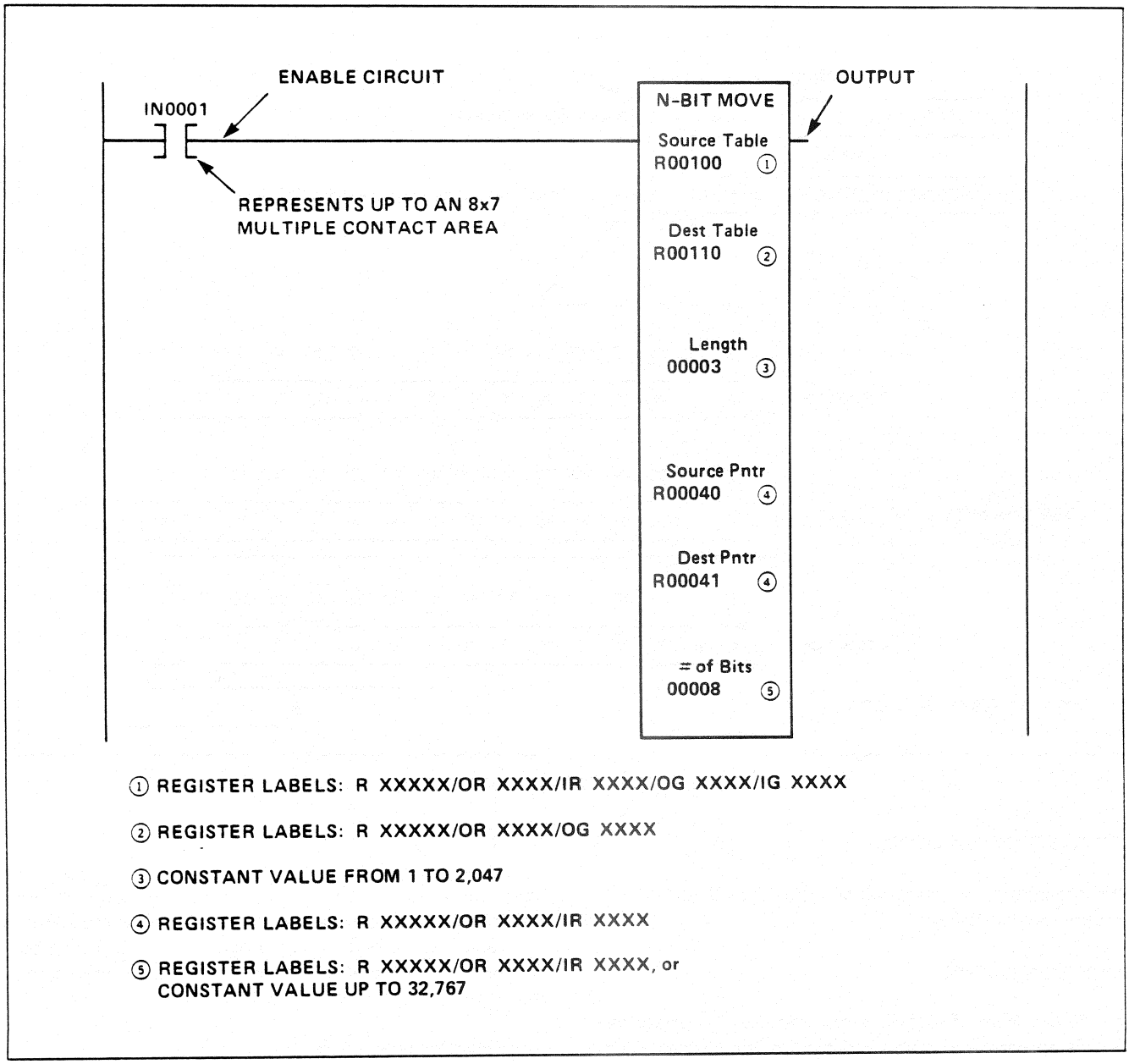


Figure NB-1. N Bit Move Function Circuit (Typical)

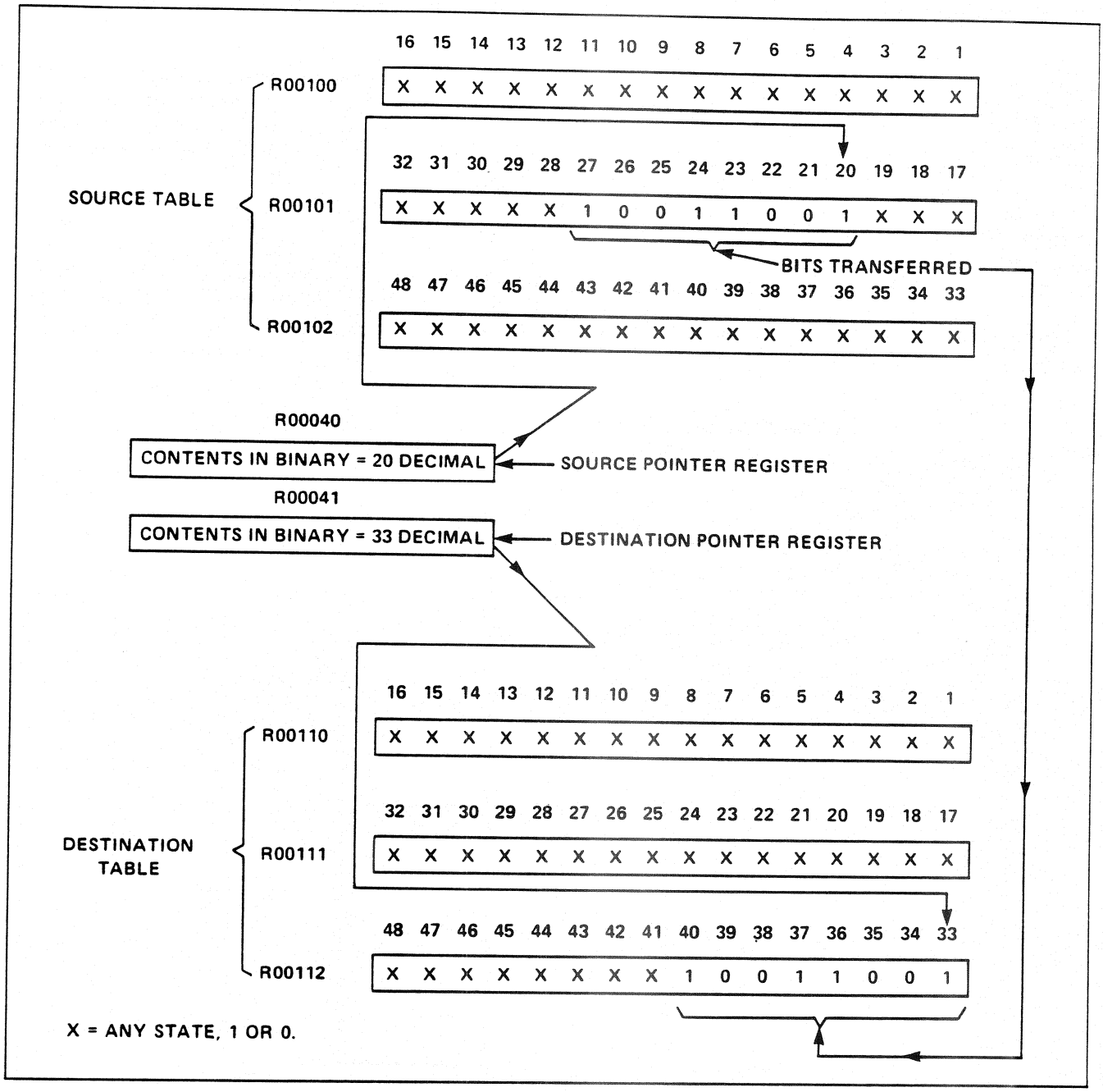


Figure NB-2. N Bit Move Example

8-17-1. SPECIFICATIONS

The programming specifications for the N Bit Move function are listed here.

FUNCTION BLOCK

The N Bit Move function block requires 2 horizontal contact spaces by 6 parallel paths on the display. The function can be positioned anywhere in the contact area of the ladder diagram.

SOURCE TABLE

The Source Table consists of a group of registers from which N bits of data are transferred. The first register in the Table is specified by the programmer and displayed in the function block. The Source Table is selected from any of the following registers or groups:

- Holding register (R)
- Output register (OR)

- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the display of the Source Table register or group number.

DESTINATION TABLE

The Destination Table consists of a group of registers into which N bits of data are transferred. The first register in the Table is specified by the programmer and displayed in the function block. The Destination Table is selected from any of the following register or group type:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR, or OG precede the display of the register or group number.

WARNING

THE USE OF AN OUTPUT GROUP FOR THE DESTINATION REGISTER OF THE N BIT MOVE FUNCTION OVERRIDES EXISTING FORCE CONDITIONS ON THE OUTPUTS ASSOCIATED WITH THE OUTPUT GROUP. THIS CAN RESULT IN UNEXPECTED MACHINE MOVEMENTS OR PROCESS OPERATIONS, WHICH MAY CAUSE PERSONNEL INJURY AND/OR EQUIPMENT DAMAGE.

LENGTH

The Length, entered by the programmer and displayed in the function block, is a constant value specifying the number of registers in the Source and Destination Tables. The values can range from 1 to 2,047. Note: A Table containing 2,048 registers would contain 32,768 bits, where the last bit is beyond the range (32,767) of the Pointer Register.

SOURCE/DESTINATION POINTERS

The Source Pointer and Destination Pointer Registers specify the first bit number of the bits to be transferred between the Source and Destination Tables, respectively. The value of the bit number contained in the Source and Destination Pointer Registers can contain any number in the range of from 0 to 32,767. The Pointer Register values and the number of bits to be transferred must not exceed the length of the table. For example, if the Pointer value of the N Bit Move function shown in Figure NB-1 were 41 or larger, the transfer would **not** take place. (The table contains only 48 bits: and 41 bits + 8 bits = 49 bits.)

The Pointer Register, specified by the programmer and displayed in the function block, is selected from any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the display of the Pointer Register number.

=OF BITS

The Number (=) of Bits specifies the number of bits, up to 16, to be transferred between the Source and Destination Tables. The bits are transferred whenever the enable circuit conducts. The Number of Bits is specified by the programmer as any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Constant value from 1 thru 16

The value of the Number of Bits register or constant is limited by the length of the Tables, as described in the Source and Destination Pointer listing above. The letters R, OR, or IR precede the register number on the display. If a constant is selected, no letter precedes the value of the display.

ENABLE CIRCUIT

The enable circuit conducts to allow the transfer of data from the Source Table to the Destination Table as directed by the Pointer Registers. The enable circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths. (This assumes the function block is positioned on the far right of the display.) The rules for programming contacts listed in Section 6 apply to the enable circuit.

OUTPUT

The output from the N Bit Move function block simply follows or repeats the conducting or nonconducting

state of the enable circuit.

Note

The output from the N Bit Move function block can be left unconnected in the network.

8-17-2. APPLICATIONS

The N Bit Move function provides the ability to easily transfer data between areas of memory. A simplified example of this data manipulation is contained here.

A transfer line moves parts between work stations. A 5-digit binary code, associated with each part, is used to store information pertinent to the part. The N-Bit Move application shown in Figure NB-3 provides the ability to monitor the 5-bit code. Observe the Figure and note the following 5 points:

1. The actual shift register function block is not shown in the Figure, although it is assumed to be contained in holding registers R00050 thru R00059. The 10 holding registers with a total of 160 bits store data for 32 stations (32 stations x 5 bits/station = 160 bits).

2. The N Bit Move function contains 10 holding registers; the Destination Pointer is loaded with a value of 145. Since the first register of the Destination Table is R00051, this will load whatever binary station code is accessed from the shift register into the lower 5 bits of holding register R00060. Note: R00060 is not part of the shift register, but is one register beyond the shift register.

3. The Source Pointer, R00048, is controlled by a thumbwheel switch located on an operator panel and wired to IR0006. The switch selects a station number ranging from 1 to 32.

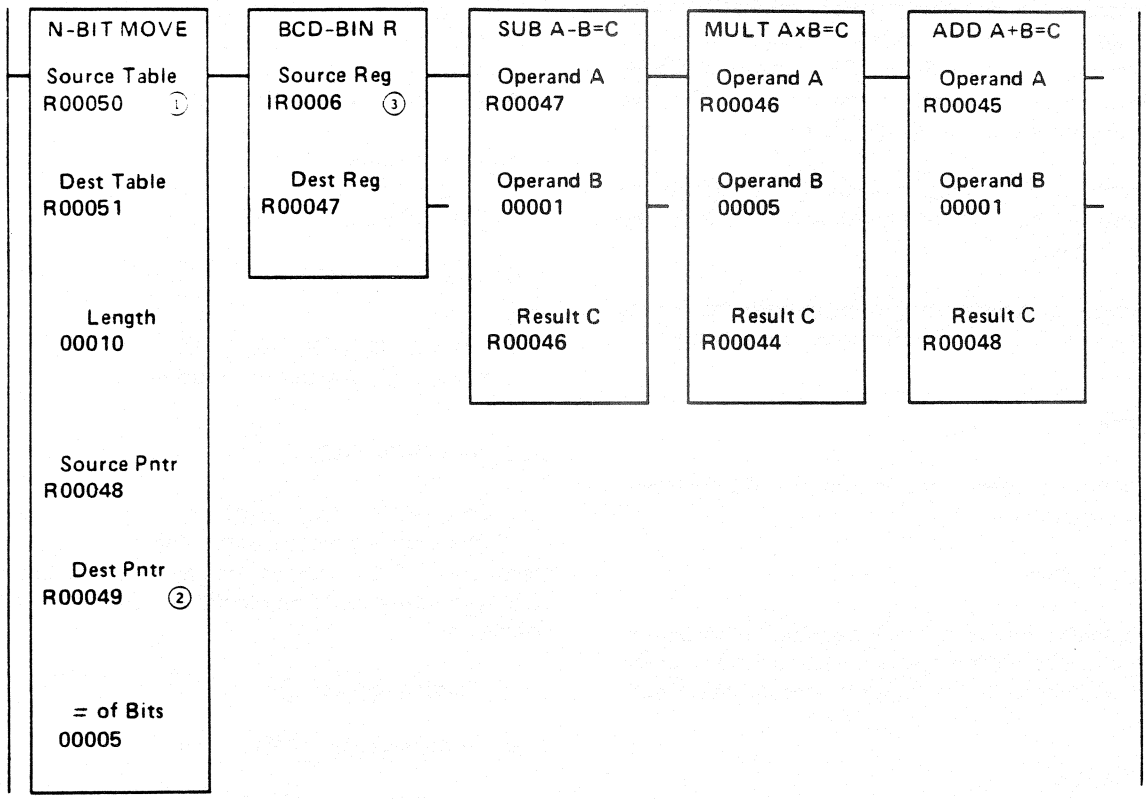
4. The group of instructions located in Figure NB-3 to the right of the N Bit Move function calculates the value used for the Source Pointer from the station number. Note: By inspection of Table NB-1 the station number can be obtained from the following formula:

$$\text{Station No.} = [(\text{station No.} - 1) \times 5] + 1$$

5. The station number is obtained from a BCD thumbwheel switch to IR0006. The BCD value obtained from IR0006 is converted to binary; subtracted from 1; multiplied by 5 and then added to 1 to obtain the correct binary code for use by the Source Pointer.

TABLE NB-1. SOURCE POINTER VALUE

Station No.	Source Pointer Value
1	1
2	6
3	11
32	156



① The functions which control the shift register contained in registers R00050 thru R00059 are not shown here.

② The Destination Pointer contains a constant value of 145.

③ IR0006 is wired to a BCD thumbwheel switch used to select station 1 to 32.

Figure NB-3. N Bit Move Application

8-18. COMPLEMENT (078)

The Complement programmable function allows the complementing and transferring of a table of registers to another location. During the transfer the state of each bit in the table is inverted. (All 0s are changed to 1s, and all 1s are changed to 0s.) Figure CP-1 shows a typical circuit containing the Complement function. Observe the Figure and note the following 2 points:

1. The enable circuit conducts, causing the data in the Source Table to be complemented and placed in the Destination Table.

2. The number of registers or groups in both the Source and Destination Tables are defined by the Length constant.

8-18-1. SPECIFICATIONS

The programming specifications for the Complement function are listed here.

FUNCTION BLOCK

The Complement function block requires 2 horizontal by 4 parallel paths on the display. The function can be positioned anywhere in the contact area of the ladder diagram.

SOURCE TABLE

The Source Table contains the registers or groups of data to be complemented. The first register or group in the Source Table is specified by the programmer and displayed in the function block. The Source Table is selected from any of the following registers or groups:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the display of the Source Table register or group number.

DESTINATION TABLE

The Destination Table contains the registers which receive the complemented data. The first register in the Destination Table is specified by the programmer and displayed in the function block. The Destination Table is selected as a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the display of the Destination Table register or group number.

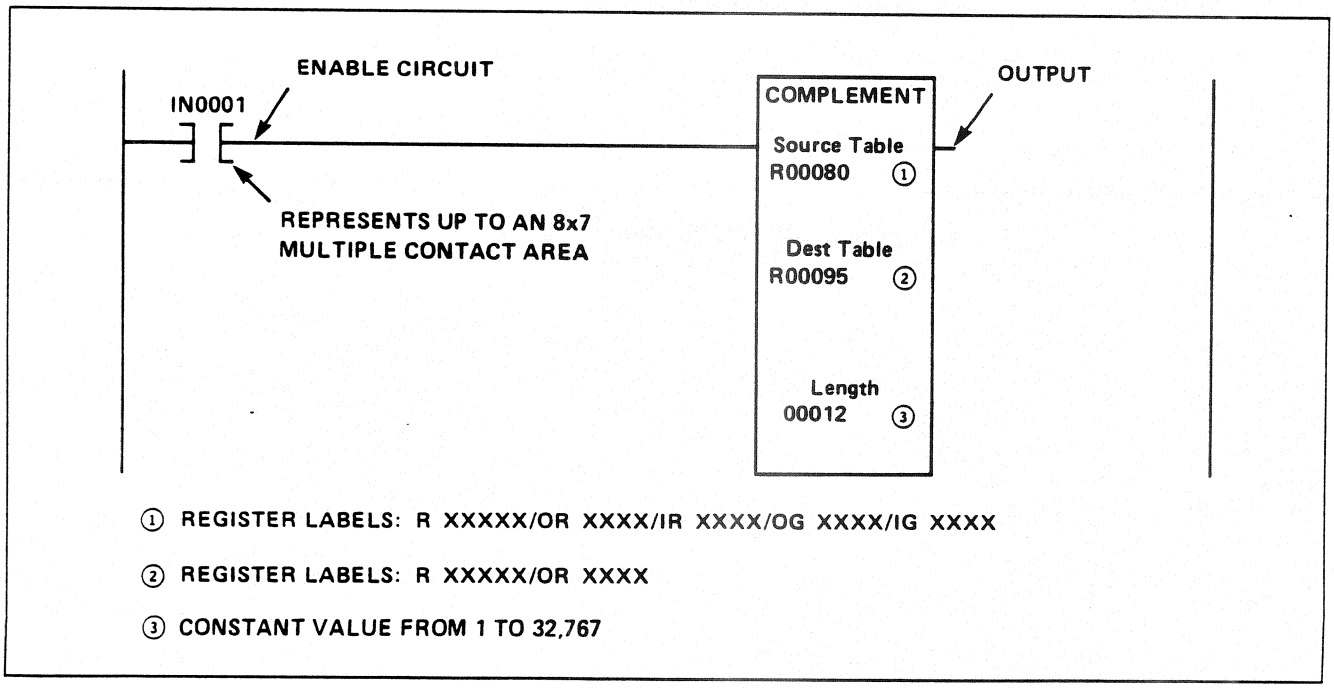


Figure CP-1. Complement Function Circuit (Typical)

LENGTH

The Length defines the number of registers in both the Source and Destination Tables. The Length is specified by the programmer in the range from 1 to 32,767.

ENABLE CIRCUIT

The enable circuit conducts, complementing the data contained in the Source Table and loading the complemented data into the Destination Table. The last data contained in the Destination Table is held when the enable circuit is nonconducting. The enable circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths. The rules for programming contacts listed in Section 6 apply to the enable circuit.

OUTPUT

The output of the Complement function block simply repeats or follows the conducting or nonconducting state of the enable circuit.

Note

The output of the Complement function block can be left unconnected in the network.

8-18-2. APPLICATION

In certain cases it is desired to check the status of bits in a table. Figure CP-2 shows a situation where outputs CR0001 thru CR0032 are examined. At the time of the examination, the outputs should be energized, and all bits in the table beginning with register R00005 should be in the 0, or off, state. If any bit is not off when examined, the output CR0025 should be energized. Observe Figure CP-2 and note the following 5 points:

1. When the contact in series with the Complement function and Search 1's function (079) conducts, the examination of the outputs CR0001 thru CR0032 takes place.

2. The Source Table of the Complement function is OG0001 and OG0002, which together contain CR0001 thru CR0032. This Table is inverted and stored in holding registers R00005 and R00006.

3. The found output of the Search 1's function conducts, energizing bit 1 of holding register R00001 if any output is energized.

4. Bit 1 of holding register 1 is examined to enable CR0025 when any output is found that is energized.

5. The reset contact in the example represents an external contact used to reset bit 6 of holding register R00001.

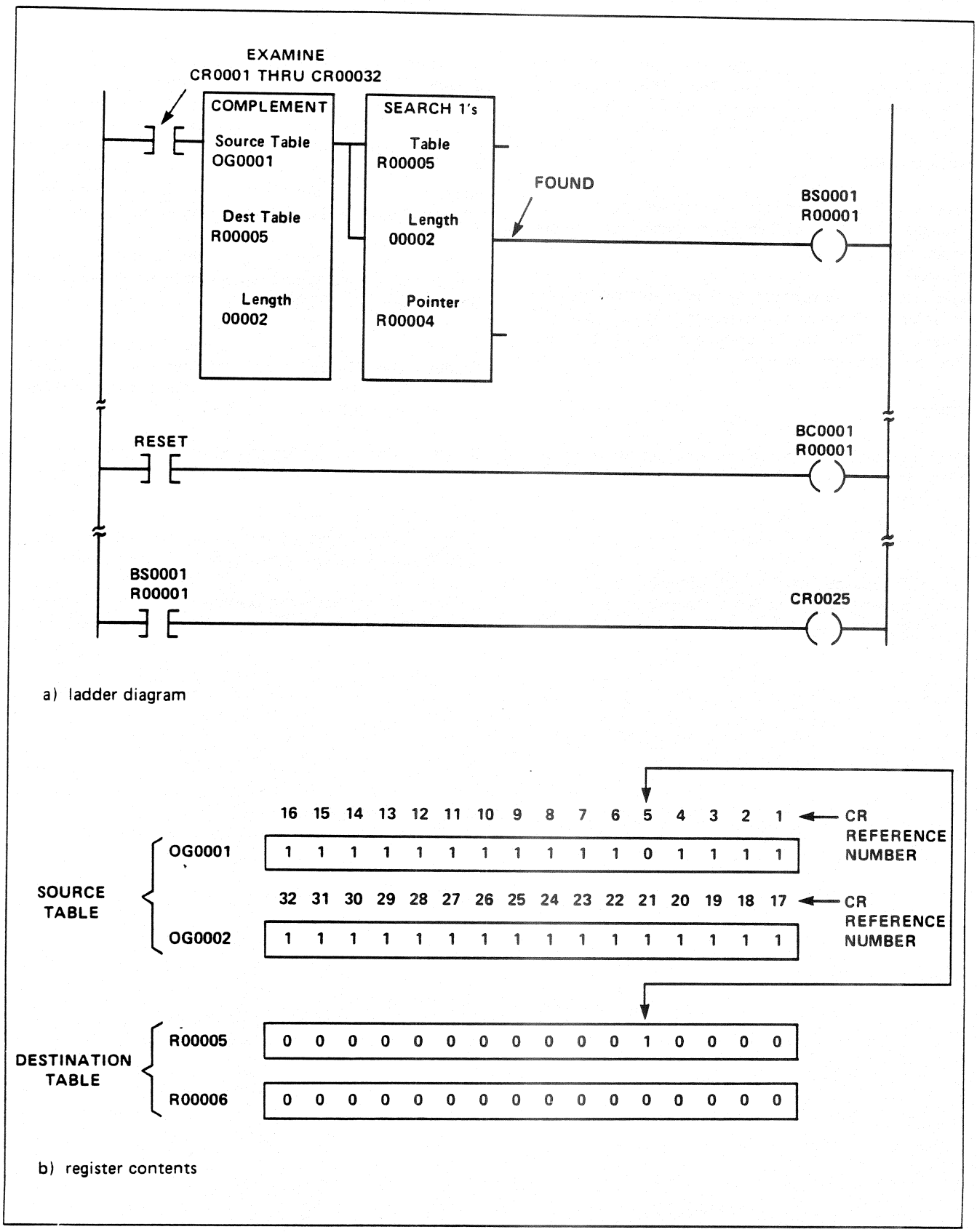


Figure CP-2. Complement/Search 1's Application

8-19. SEARCH 1's (079)

The Search 1's programmable function allows the searching for and locating of 1, or on, states of bits in a table of registers. Figure SO-1 shows a typical circuit containing the Search 1's function. Observe the Figure and note the following 5 points:

1. The Table lists the first register or group in the Table to be searched. The Length specifies the number of registers or groups in the Table.

2. The Pointer Register contains the bit number (in binary) of a bit in the Table currently found to be in a 1 state.

3. During each ladder diagram scan when the enable circuit conducts, the Search 1's function searches from the current Pointer Register position to the next **higher** bit number containing a 1, or until all bits of the table have been searched and found to be in a 0 state. (See Figure SO-2.)

4. The data contained in the Pointer Register is held between scans. When the reset circuit is nonconducting, the Pointer Register resets to zero.

5. The data contained in the Pointer Register can change during each scan if the enable circuit conducts. If the value of the Pointer Register between scans is desired, it is the user's responsibility to save the value.

8-19-1. SPECIFICATIONS

The programming specifications for the Search 1's function are listed here.

FUNCTION BLOCK

The Search 1's function block requires 4 horizontal contact spaces by 4 parallel paths on the display. The function can be positioned anywhere in the contact area of the ladder diagram.

TABLE

The Table contains the registers or groups to be searched. The Table register or group is actually the first register in the Table. This register or group is specified by the programmer and appears in the function block. The Table register or group is selected from any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the display of

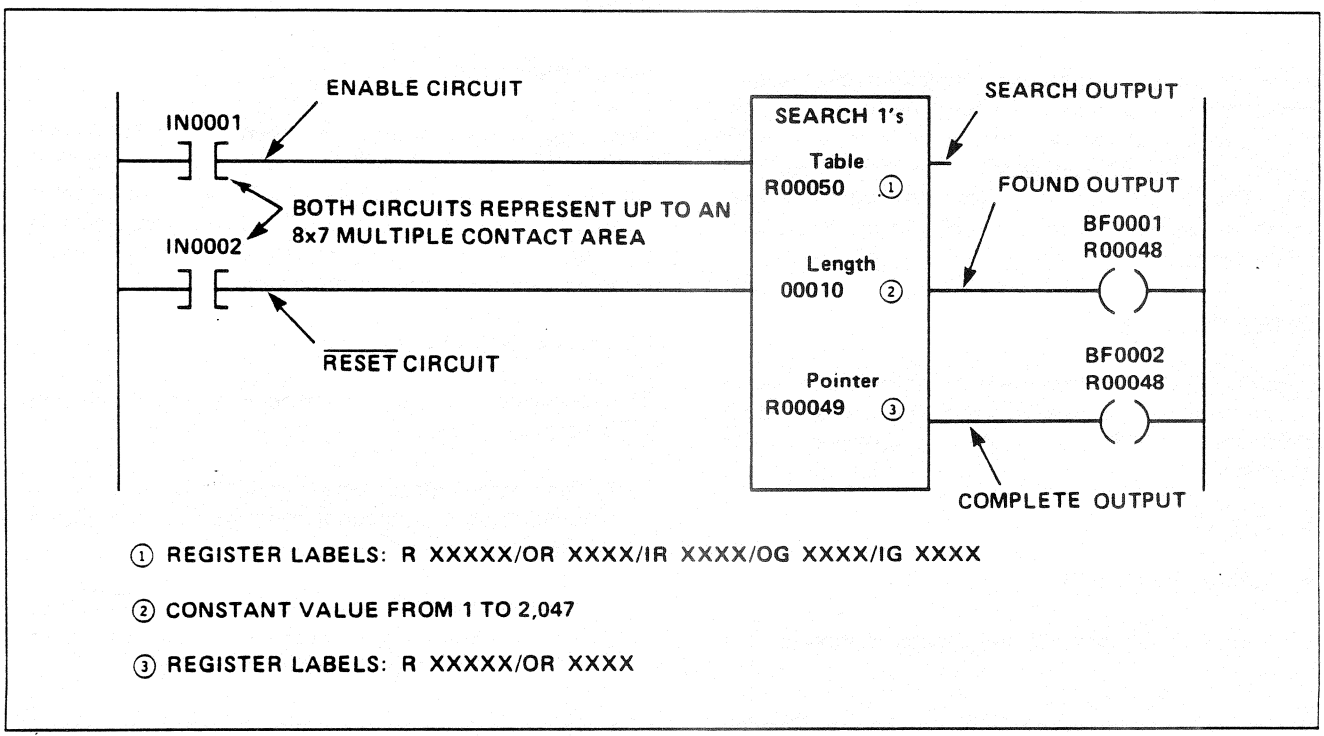


Figure SO-1. Search 1's Function Circuit (Typical)

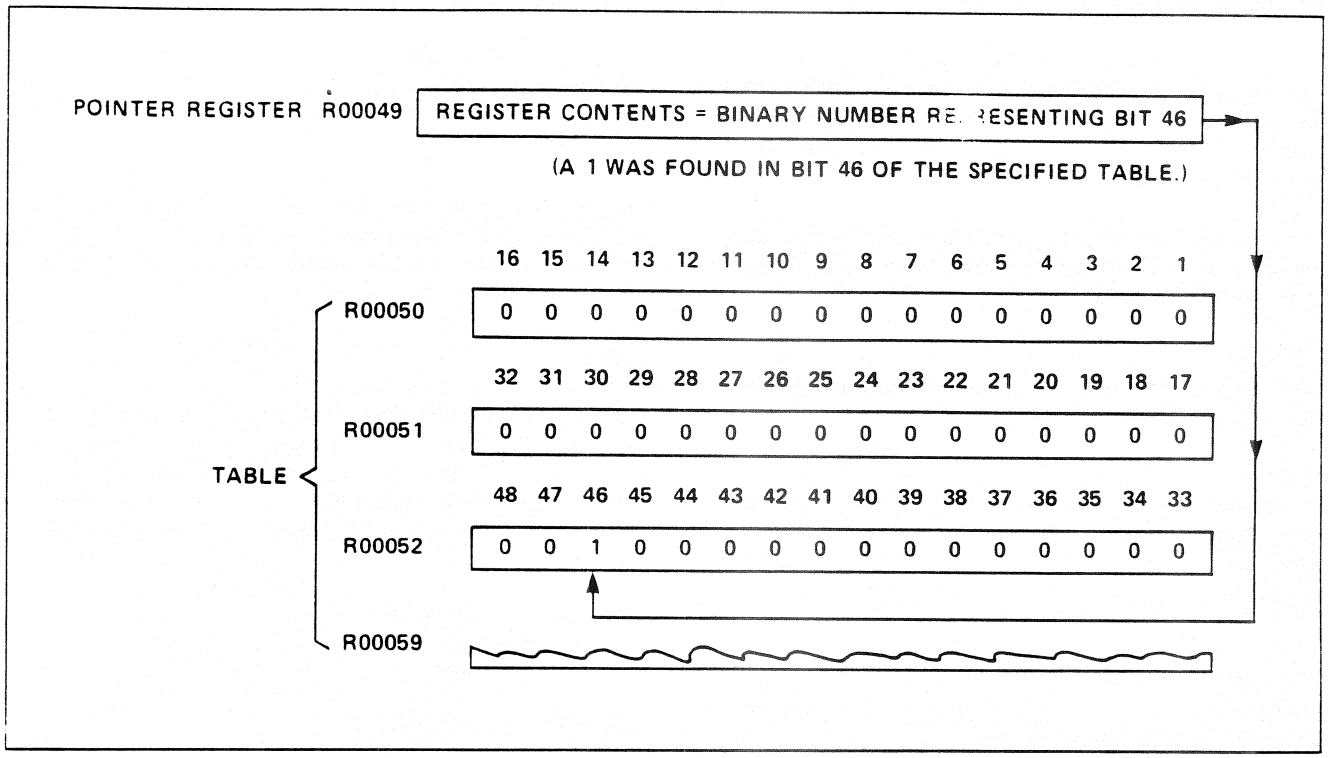


Figure SO-2. Search 1's Example

the Table register or group.

LENGTH

The Length is a constant value specified by the programmer and is displayed in the function block. It specifies the number of registers in the table that is to be searched. The Length can vary in the range from 1 to 2,047. (The Length can be programmed up to 32,767, although, when above 2,047, the bits in the Table cannot be accessed.)

POINTER REGISTER

The Pointer Register holds the bit number examined in the Table found to be in a 1 state during the last ladder diagram scan. The Pointer Register is specified by the programmer as:

- Holding Register (R)
- Output register (OR)

The letters R or OR precede the register number on the display.

Each time a 1 is found in the Table, the Pointer Register holds the bit number, and the ladder diagram scan continues. During the next scan the searching continues until another 1 is found, or until the Table is searched to the end. If no 1s are found in the Table, the Pointer Register is reset to 0.

INPUT CIRCUITS

Two input circuits control the operation of the Search 1's function, as listed in Table SO-1. These circuits can consist of up to 8 horizontal contacts by 7 parallel paths located to the left of the Search 1's function block on the ladder diagram. The rules for programming contacts listed in Section 6 apply to these circuits.

OUTPUT CIRCUITS

The following 3 outputs are associated with the Search 1's function:

- The search output simply follows or repeats the conducting or nonconducting state of the enable circuit.
- The found output conducts when the $\overline{\text{reset}}$ circuit conducts and the Pointer Register contains the bit number of a 1 state found in the Table.
- The complete output conducts when the enable circuit conducts and the search is completed. At this time the Pointer Register contains a zero.

TABLE SO-1. INPUT CIRCUIT OPERATIONS

Enable Circuit ①	$\overline{\text{Reset}}$ Circuit ①	Operation
0 or 1	0	No searching occurs. Pointer Register held at 0.
0	1	No searching occurs. Pointer Register held at its last value.
1	1	Searching takes place. After each scan, the Pointer Register contains one of the following: • Bit number of the next 1 found in Table • Zero if the complete output of the Search function is conducting.
① 1 = conducting, 0 = nonconducting		

Note

The outputs of the Search 1's function block can be left unconnected in the network.

8-19-2. APPLICATIONS

The Search 1's function can be used as part of a monitoring program where outputs are examined for predetermined states. (See Figure SO-3.) By performing an exclusive OR type comparison between actual outputs and a table containing the predetermined, or "desired-state," conditions, any deviations can be detected as a 1 condition stored in the Destination Register. Observe the ladder diagram shown in Figure SO-4 and note the following 4 points:

1. The Exclusive OR Table-to-Table function (101) compares the actual states of CR0001 thru CR0080 (OG0001 thru OG0005) to the information in holding registers R00070 thru R00074.

2. Whenever the bit-by-bit comparison of the data contained in the 2 tables is different, a 1 state is stored in the Destination Table, R00076 thru R00080.

3. The Search 1's function examines each bit contained in the table R00076 thru R00080. Each time a 1 is present, it enables the found output until the next scan.

4. The found output enables bit 1 of holding register R00048. Contacts from this bit (not shown in Figure SO-4) would be used to perform operations such as disabling the machine or process or storing the bit data from the Pointer Register (R00080) for use by the ladder diagram or subsequent maintenance operations.

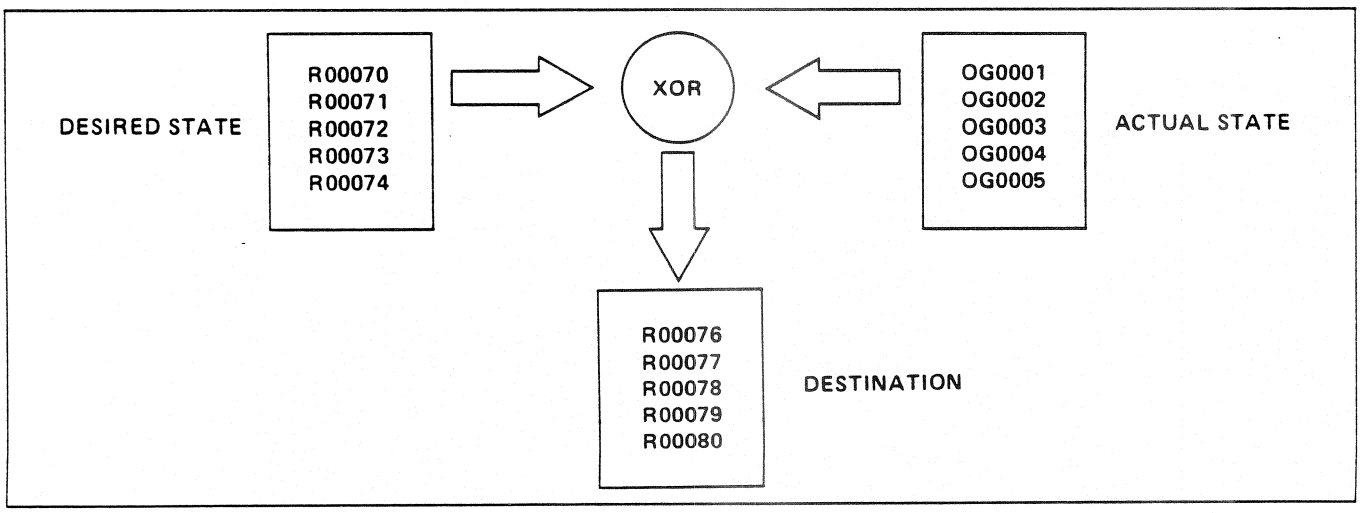


Figure SO-3. Examine Outputs Examination

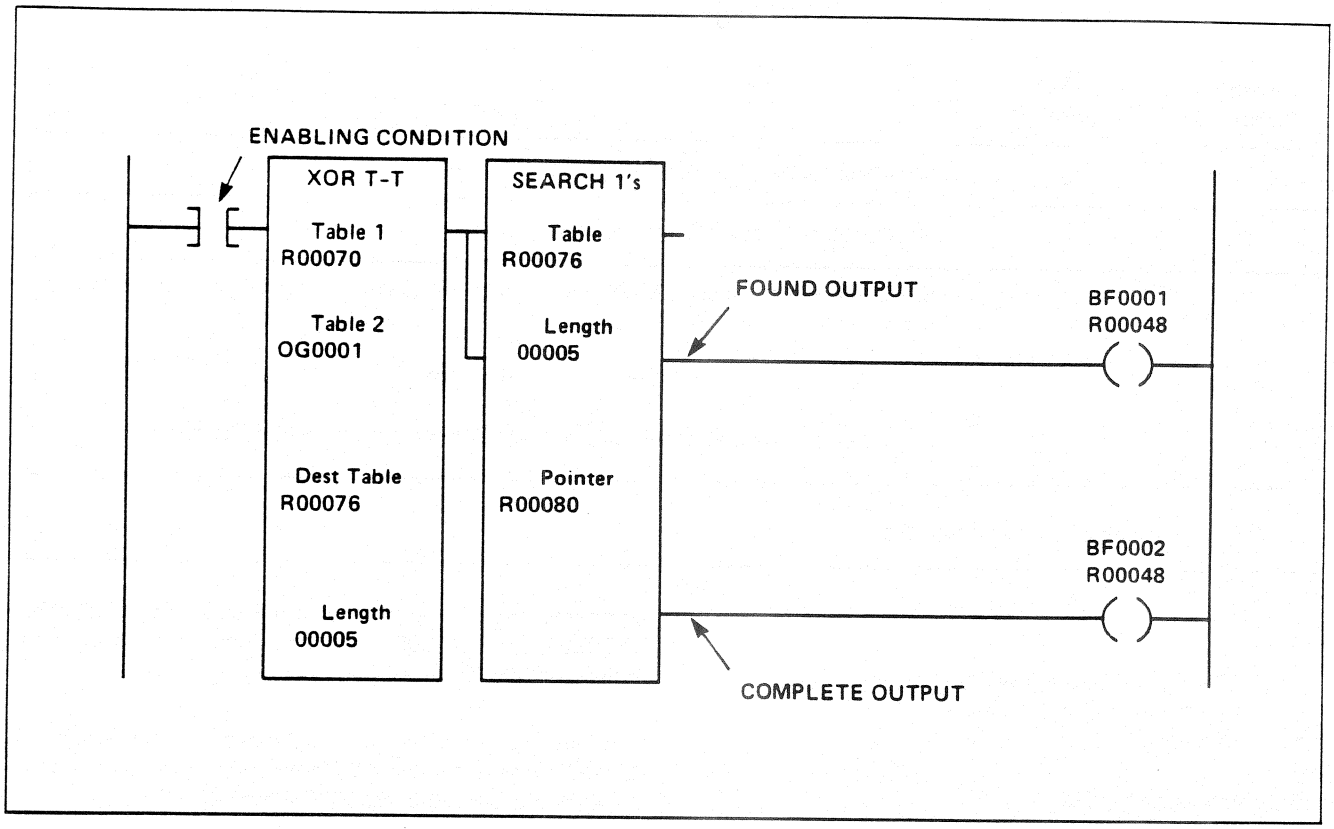


Figure SO-4. Examine Outputs Application

8-20. REGISTER-TO-TABLE LOGIC FUNCTIONS (097-099)

The Register-to-Table logic type programmable functions include the:

- AND Register-to-Table logic function (097)
- OR Register-to-Table logic function (098)
- XOR Register-to-Table logic function (099)

The 3 Register-to-Table logic type functions have many common characteristics. Observe Figure RT-1 and note the following 3 points:

1. A logic operation is performed between the Reference Register and the data contained in each register of the Source Table. Bit 1 of the Source Register is logically compared to bit 1 of the first register in the Reference Register; bit 2 is compared to bit 2; etc. The results of the logic operation are loaded into the Destination Table.

2. The Length defines the number of registers in **both** the Source Table and Destination Table.

3. The logic operation between the Reference Register and each register of the Source Table takes place during each scan if the enable circuit is conducting. Table RT-1 contains truth tables for each of the Register-to-Table functions.

Figure RT-2 shows the result of the AND, OR, and XOR Register-to-Table logic functions between Reference Register R00038 and Source Table Registers R00040 and R00041. The results are located in the Destination Table, registers R00060 and R00061.

8-20-1. SPECIFICATIONS

The specifications for the Register-to-Table logic type functions are listed here.

FUNCTION BLOCK

The function block for the Register-to-Table logic

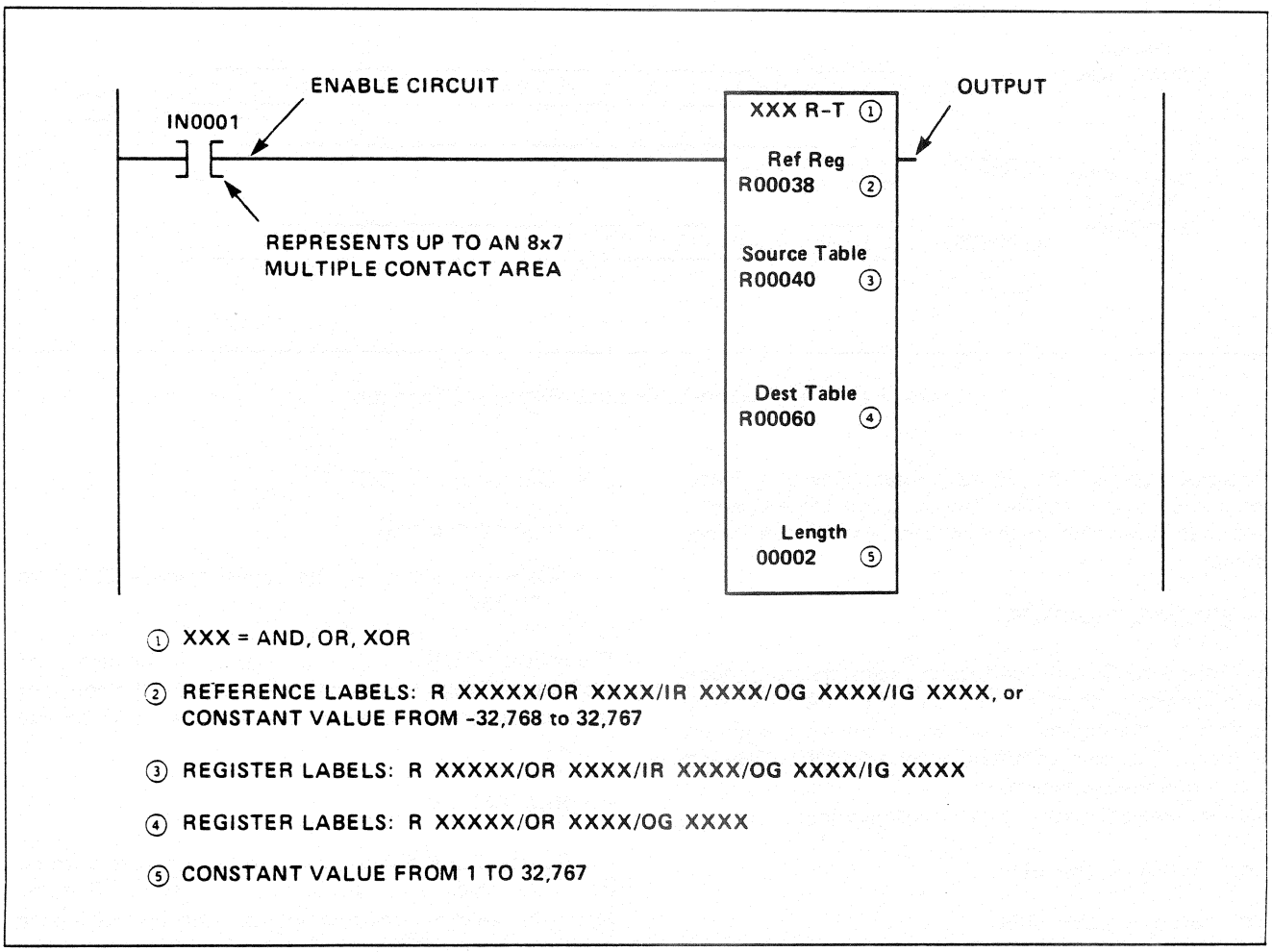


Figure RT-1. Register-to-Table Logic Functions Circuit (Typical)

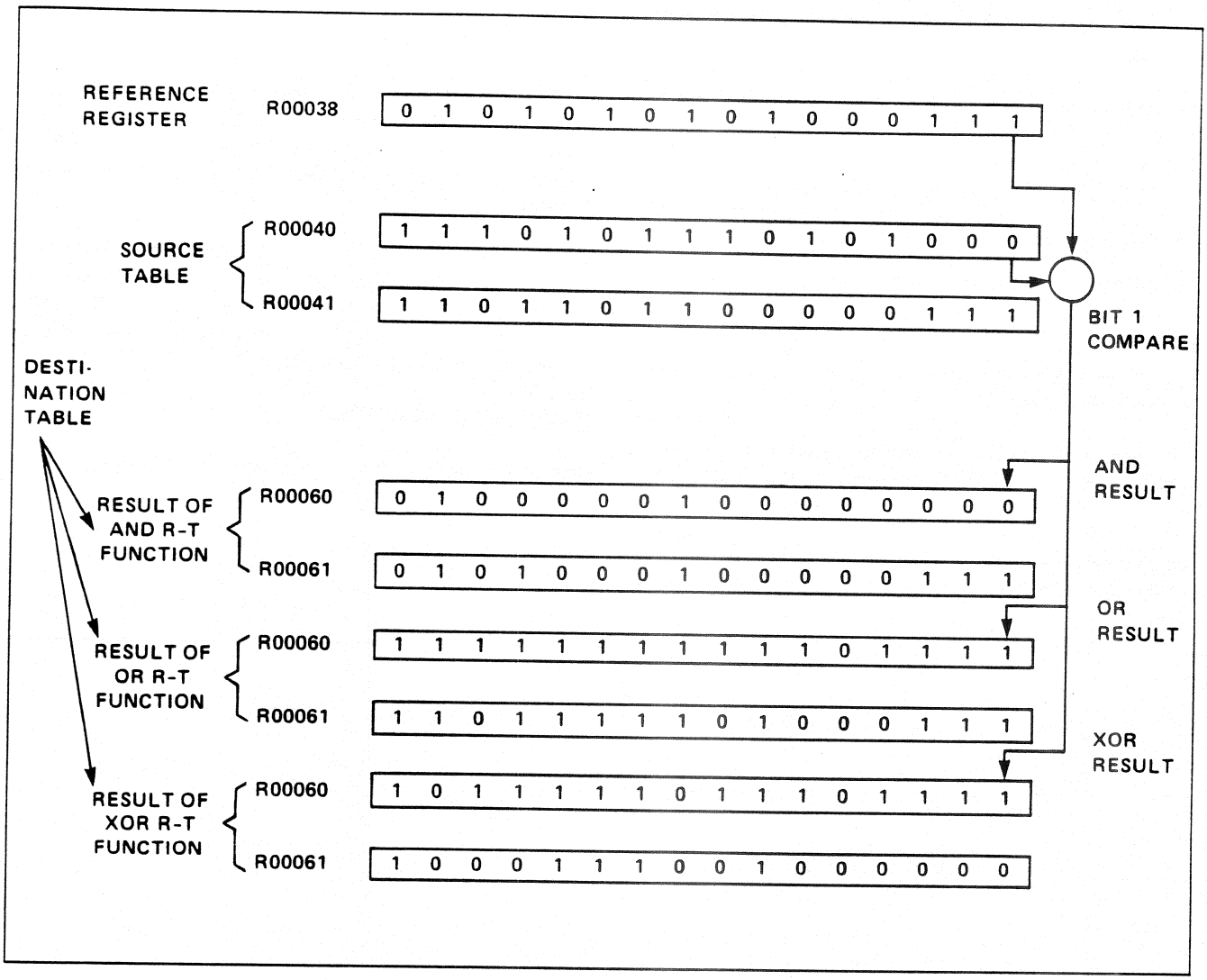


Figure RT-2. Register-to-Table Logic Functions Example

functions requires 2 horizontal contact spaces by 5 parallel paths on the display. The function blocks can be positioned anywhere in the contact area of the ladder diagram.

REFERENCE REGISTER

The Reference Register defines the register, group number, or constant which is to be logically compared to the registers in the Source Table when the enable circuit conducts. The logic comparison for the AND, OR, and XOR functions are listed in Table RT-1. The Reference Register is selected from any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)
- Constant value in the range from -32,768 to +32,767

The letters R, OR, IR, OG, or IG precede the display of the Source Table register or group number. If a constant value is selected, no letters precede the value on the display.

SOURCE TABLE

The Source Table consists of a group of registers which are to be logically compared to the Reference Register when the enable circuit conducts. (See the Reference Register, above.) The first register or group of the Source Table is displayed in the function block and specified by

TABLE RT-1. LOGIC TRUTH TABLE

	AND Function				OR Function				XOR Function			
Reference Register	0	1	0	1	0	1	0	1	0	1	0	1
Source Table	0	0	1	1	0	0	1	1	0	0	1	1
Destination Table (Result)	0	0	0	1	0	1	1	1	0	1	1	0

the programmer as any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the display of the Source Table register or group number.

DESTINATION TABLE

The Destination Table contains the results of the logical comparison between the Reference Register and the Source Table. The first register in the Destination Table is displayed in the function block and specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR, or OG precede the display of the Destination Register in the function block.

WARNING

THE USE OF AN OUTPUT GROUP FOR THE DESTINATION REGISTER OF THE REGISTER-TO-TABLE LOGIC FUNCTIONS OVERRIDES EXISTING FORCE CONDITIONS ON THE OUTPUTS ASSOCIATED WITH THE OUTPUT GROUP. THIS CAN RESULT IN UNEXPECTED MACHINE MOVEMENTS OR PROCESS OPERATIONS, WHICH MAY CAUSE PERSONNEL INJURY AND/OR EQUIPMENT DAMAGE.

LENGTH

The Length consists of a constant value in the range from 1 to 32,767 which specifies the length of the Source and Destination Tables.

ENABLE CIRCUIT

When the enable circuit conducts, the logical comparison between the Source Register and the Source Table occurs. The results are stored in the Destination Table. When the circuit is nonconducting, all data is held in its last state. The enable circuit consists of up to 8 horizontal contact spaces by 7 parallel paths on the display.

OUTPUT

The output of the Register-to-Table logic function block simply repeats or follows the conducting or nonconducting state of the enable circuit.

Note

The output of the Register-to-Table logic function block can be left unconnected in the network.

8-20-2. APPLICATION

The Register-to-Table logic type functions are used whenever an AND, OR, or XOR logic function between a register and table of values is desired. An example of the use of Register-to-Table logic functions is contained in Paragraph 8-9-2.

8-21. TABLE-TO-TABLE LOGIC FUNCTIONS (100-102)

The Table-to-Table logic type programmable functions include the:

- AND Table-to-Table logic function (100)
- OR Table-to-Table logic function (101)
- XOR Table-to-Table logic function (102)

The 3 Table-to-Table logic type functions have many common characteristics. Observe Figure TT-1 and note the following 3 points:

1. A logic operation is performed between Table 1 and Table 2, where: bit 1 of the first register in Table 1 is compared to bit 1 of the first register in Table 2; bit 2 is compared to bit 2; etc. The logical results of the operation are loaded into the Destination Table.

2. The Length defines the number of registers in Table 1, 2, and the Destination Table. (This number is always the same for all 3 Tables.)

3. The logic operation between Table 1 and Table 2 occurs during each scan when the enable circuit is conducting. Table TT-1 contains the truth table for each of the Table-to-Table logic type functions.

Figure TT-2 shows the results of the AND, OR, and XOR Table-to-Table logic functions. In the examples shown:

- Table 1 = IR0010 and IR0011
- Table 2 = R00050 and R00051
- Destination Table = R00052 and R00053

8-21-1. SPECIFICATIONS

The specifications for the Table-to-Table logic functions are listed here.

FUNCTION BLOCK

The function block for the Table-to-Table logic type functions requires 2 horizontal contact spaces by 5 parallel paths on the display. The function can be positioned anywhere in the contact area of the ladder diagram.

TABLES 1 AND 2

Table 1 and Table 2 define the register or group numbers

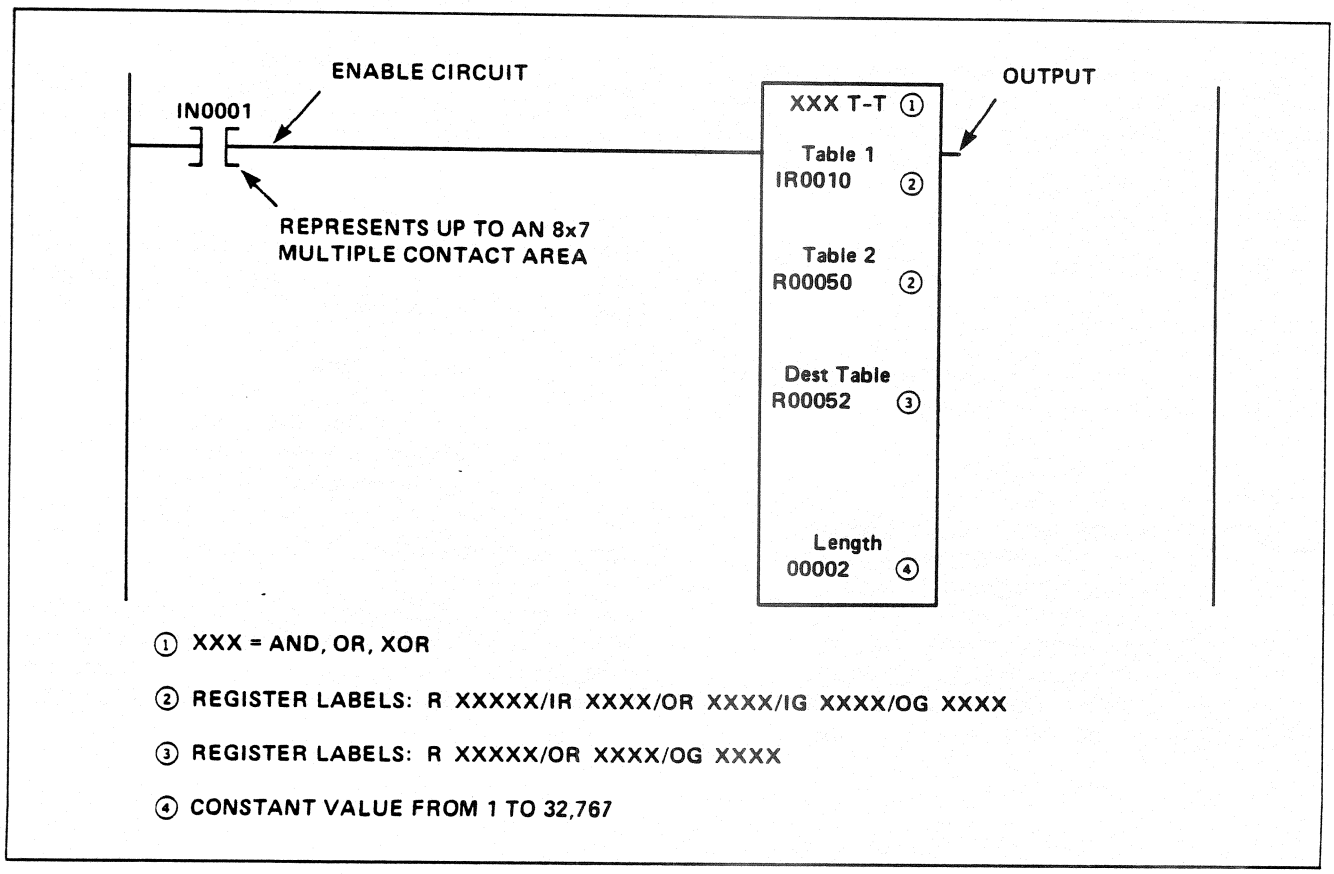


Figure TT-1. Table-to-Table Function Circuit (Typical)

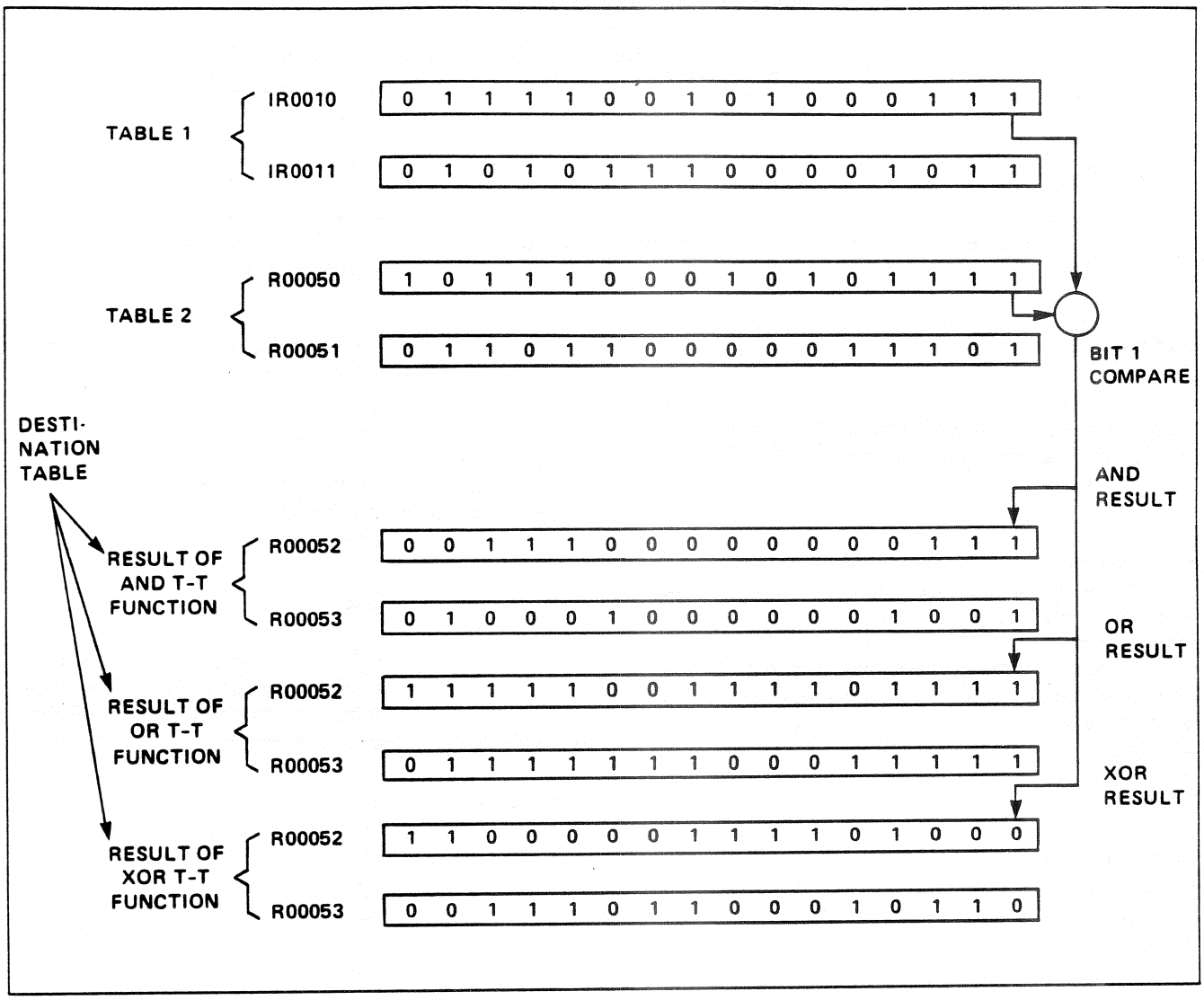


Figure TT-2. Table-to-Table Logic Function Examples

which are to be logically compared. Table TT-1 contains the logic truth tables for the AND, OR, and XOR Table-to-Table logic functions. The first register or group of the Tables is specified by the programmer and displayed in the function block. The Tables are selected from any of the following registers or groups:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the display of the Table register or group numbers.

DESTINATION TABLE

The Destination Table holds the result of the logical operation between Table 1 and Table 2. The first register of the Destination Table is specified by the programmer and displayed in the function block. The Table is selected from a:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR, or OG precede the display of the Destination Table group or register number.

TABLE TT-1. LOGIC TRUTH TABLES

	AND Function	OR Function	XOR Function
Table 1	0 1 0 1	0 1 0 1	0 1 0 1
Table 2	0 0 1 1	0 0 1 1	0 0 1 1
Destination Table (Result)	0 0 0 1	0 1 1 1	0 1 1 0

LENGTH

The Length is a constant value used to specify the number of registers in Table 1, Table 2, and the Destination Table. The value of the Length can range from 1 to 32,767.

ENABLE CIRCUIT

The enable circuit conducts, allowing the logical comparison between Table 1 and Table 2, and also storing the results in the Destination Table. When the circuit is nonconducting, all Table data is held in its last state. The enable circuit consists of up to 8 horizontal contact spaces by 7 parallel paths on the display. The rules for programming contacts listed in Section 6 apply to the enable circuit.

OUTPUT

The output of the Table-to-Table logic type function block simply repeats or follows the conducting or nonconducting state of the enable circuit.

Note

The output of the Table-to-Table logic type function block can be left unconnected in the network.

8-21-2. APPLICATIONS

The Table-to-Table logic type functions are used whenever an AND, OR, or XOR logic function between two tables of values is desired. An example of the use of the Table-to-Table logic type functions is contained in:

- Paragraph 8-13-2
- Paragraph 8-19-2

8-22. BIT OPERATE (081)

The Bit Operate programmable function allows the following operations to be performed on any bit contained in a table associated with the function:

- Set the bit to a 1 state
- Clear the bit to a 0 state
- Examine the status (0 or 1) of the bit

Figure BO-1 shows a typical circuit containing the Bit Operate function. Observe the Figure and note the following 3 points:

1. The Table entry specifies the first register in the Table. The Length defines how many registers are contained in the Table.

2. The Pointer Register contains the current bit number. This value is a binary number used to specify a bit in the Table. See Figure BO-2 where bit 20 is referenced. The bits in the Table simply begin with 2 and are numbered consecutively to the end of the Table. In this example, the bits are numbered to 48.

3. In the example shown in Figure BO-2, bit 20 is set to 1 if the enable circuit is conducting and the bit set circuit conducts. Bit 20 is cleared when the enable circuit is conducting, and the bit clear circuit conducts. Also, when the enable circuit conducts, the bit follow output conducts if bit 20 = 1, or does not conduct if bit 20 = 0.

8-22-1. SPECIFICATIONS

The specifications for the Bit Operate function are listed here.

FUNCTION BLOCK

The Bit Operate function block requires 2 horizontal contact spaces by 4 parallel paths on the display. The function block can be located anywhere in the contact area of the ladder diagram.

TABLE

The Table consists of a group of registers associated with the Bit Operate function. The first register in the Table is specified by the programmer and displayed in the function block. The Table is selected from any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Input group (IG)
- Output group (OG)

The letters R, OR, IR, IG, or OG precede the register number on the display.

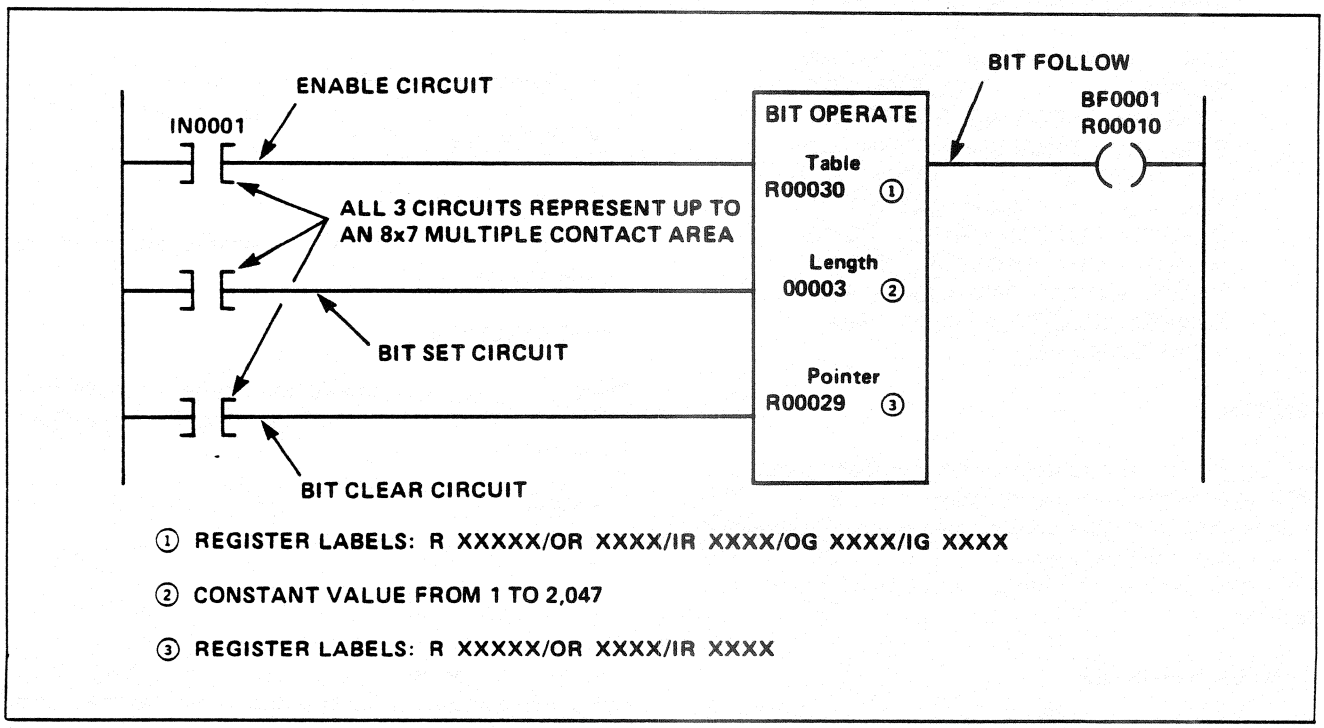


Figure BO-1. Bit Operate Function Circuit (Typical)

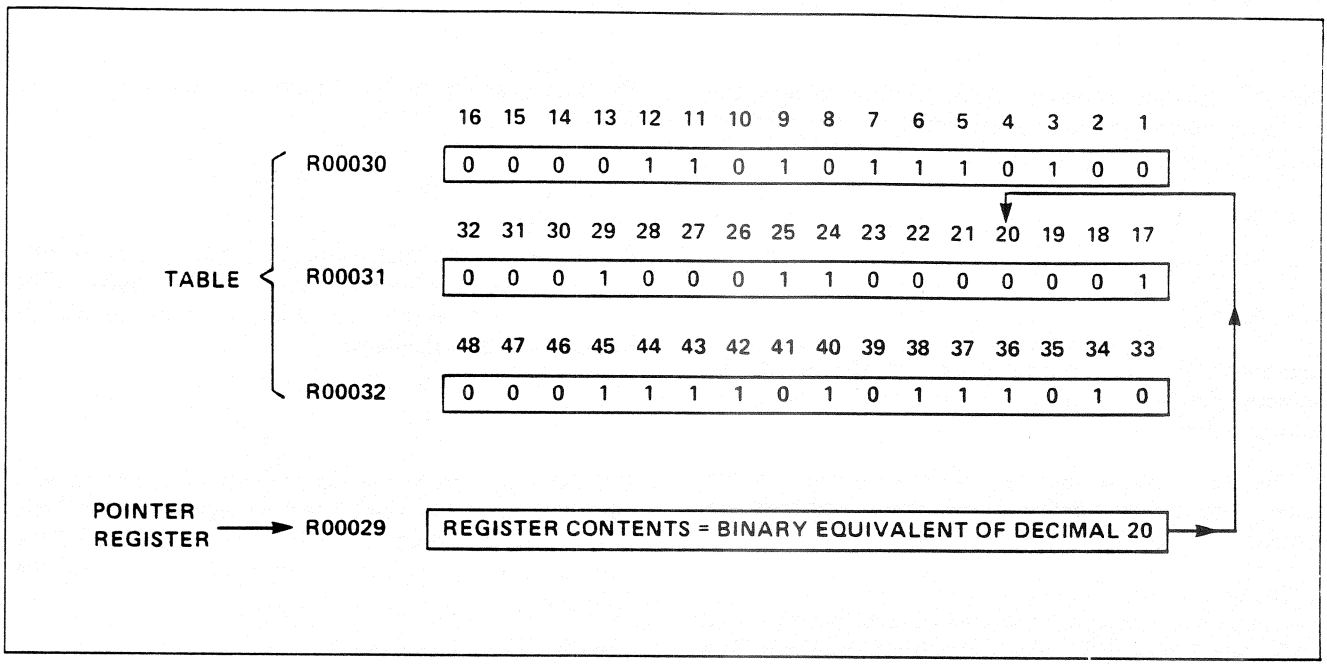


Figure BO-2. Bit Operate Function Example

WARNING

WHEN THE TABLE OF THE BIT OPERATE FUNCTION IS SPECIFIED AS AN INPUT GROUP (IG), THE SETTING OR CLEARING OF BITS IN THE TABLE OVERRIDES EXISTING FORCE CONDITIONS FOR THE REMAINDER OF THE SCAN. WHEN SPECIFIED AS AN OUTPUT GROUP (OG), THE SETTING OR CLEARING OVERRIDES EXISTING FORCE CONDITIONS ON THE OUTPUTS ASSOCIATED WITH THE OG. THIS CAN RESULT IN UNEXPECTED MACHINE MOVEMENTS OR PROCESS OPERATIONS, WHICH MAY CAUSE PERSONNEL INJURY AND/OR EQUIPMENT DAMAGE.

LENGTH

The Length is a constant value ranging from 1 to 2,047 which specifies the number of registers in the Table. This value is entered by the programmer. Note: A Table containing 2,048 registers would contain 32,768 bits. The last bit is beyond the range (32,767) of the Pointer Register.

POINTER REGISTER

The Pointer Register contains the current bit number which is to be operated on by the Bit Operate function. Bit numbers are in consecutive order from 1 to 32,767, as shown in the example of Figure BO-1. If the Pointer Register exceeds the number of bits in the Table defined by the Length, the Bit Operate function operates as if the enable circuit is nonconducting. See Table BO-1.

The Pointer Register is specified by the programmer as any of the following:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the display of the register number.

INPUT CIRCUITS

There are three input circuits that control the operation of the Bit Operate function as listed in Table BO-1. These are:

- Enable circuit
- Bit set circuit
- Bit clear circuit

All 3 circuits together can consist of up to 8 horizontal contact spaces by 7 parallel paths located to the left of the Bit Operate function block on the ladder diagram.

TABLE BO-1. INPUT CIRCUITS OPERATION

Enable Circuit ①	Bit Set Circuit	Bit Clear Circuit	Result ②
0	0 or 1	0 or 1	The bit referenced by the Pointer Register is held in its last state. The bit follow output is nonconducting.
1	0	0	The bit referenced by the Pointer Register is examined. If the bit is a: • 1, the bit follow output conducts • 0, the bit follow output is nonconducting
	0	1	The bit referenced by the Pointer Register is cleared to 0. The bit follow output is nonconducting.
	1	0	The bit referenced by the Pointer Register is set to a 1. The bit follow output conducts.
	1	1	The bit referenced by the Pointer Register is set to 0. The bit follow output is nonconducting.

① 0 = nonconducting, 1 = conducting

② If the bit number referenced by the Pointer Register exceeds the number of bits in the Table, the Bit Operate function operates as if the enable circuit is nonconducting, or 0.

The rules for programming contacts listed in Section 6 apply to these circuits.

OUTPUT

The bit follow output conducts when the enable circuit conducts, and the bit referenced by the Pointer Register is in a 1 state. The bit follow output is nonconducting at all other times.

Note

The output of the Bit Operate function block can be left unconnected in the network.

8-22-2. APPLICATIONS

The Bit Operate application can be used to allow monitoring of the status of selected input circuits. A BCD thumbwheel switch contained on an operator panel is used to select the desired input. If the input is energized, an output turns on. See the ladder diagram shown in Figure BO-3 and note the following 3 points:

1. The BCD input from the thumbwheel switch wired to input register IR0001 is used to select a bit number. This bit number references the desired bit from input groups IG0001 thru IG0020.

2. The BCD-to-Binary Register function (047) converts the BCD number from the thumbwheel into a binary number. This number is used as the Pointer Register for the Bit Operate function. Here a value of 1 examines CR0001, a value of 2 examines CR0002, etc. If the bit number examined contains a 1, the output CR0010 is energized. If the bit number examined contains a 0, the output is de-energized.

3. By selecting the desired bit number and depressing the enable pushbutton wired to IN0001, the state of CR0010 reflects the state of the bit examined in the input group.

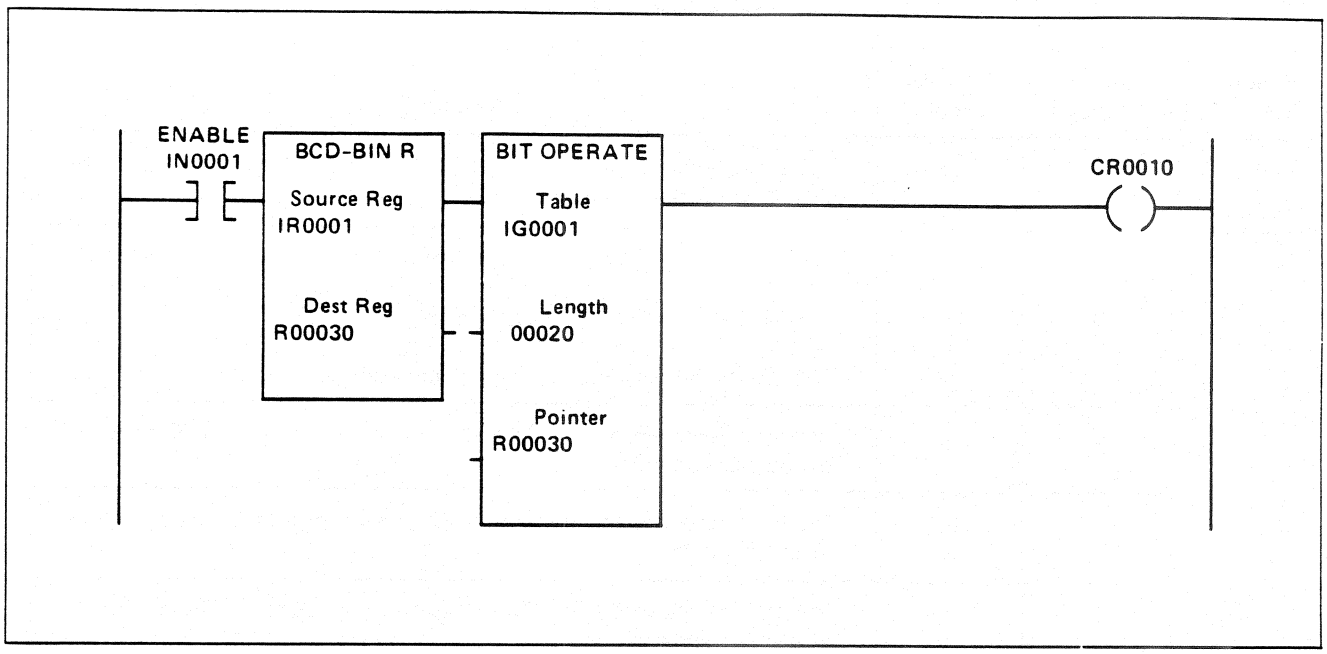


Figure BO-3. Bit Operate Application (Typical)

8-23. SEARCH = (106), SEARCH > = (107)

The Search type functions consist of 2 types.

The first type is the Search Equal To function (106). It is used to search through a table of registers and locate the register with contents equal to a specified value. The location of the register found in the Table will be contained in the Pointer Register associated with the search type function.

The second type is the Search Equal To or Greater Than function (107). It is used to search through a table of registers and locate the register with contents equal to or greater than a specified value. The location of the register found in the Table will be contained in the Pointer Register associated with the search type function.

Observe Figure SE-1 and note that the Search type functions have the following 4 characteristics in common:

1. The function searches through the Table either for the registers with values equal to or greater than, or for those simply equal to the value of the Reference Register. The functions are active when the:

- Enable circuit conducts, and the
- $\overline{\text{Reset}}$ circuit conducts

As long as the enable and reset circuits conduct, the searching occurs during each scan until a register in the Table with equal contents is found, or until the Table is searched completely.

2. When a search is successful, the found output conducts, and the Pointer Register contains the number of the register in the Table containing the value equal to or greater than the value of the Reference Register. When the $\overline{\text{reset}}$ circuit is nonconducting, the value of the Pointer Register is cleared to 0.

3. The Length, ranging from 1 to 32,767, specifies the number of registers in the Table to be searched.

4. The complete output conducts after the Search type function is executed, and the Table is searched to its end with no values equal to or greater than the value of the Reference Register found.

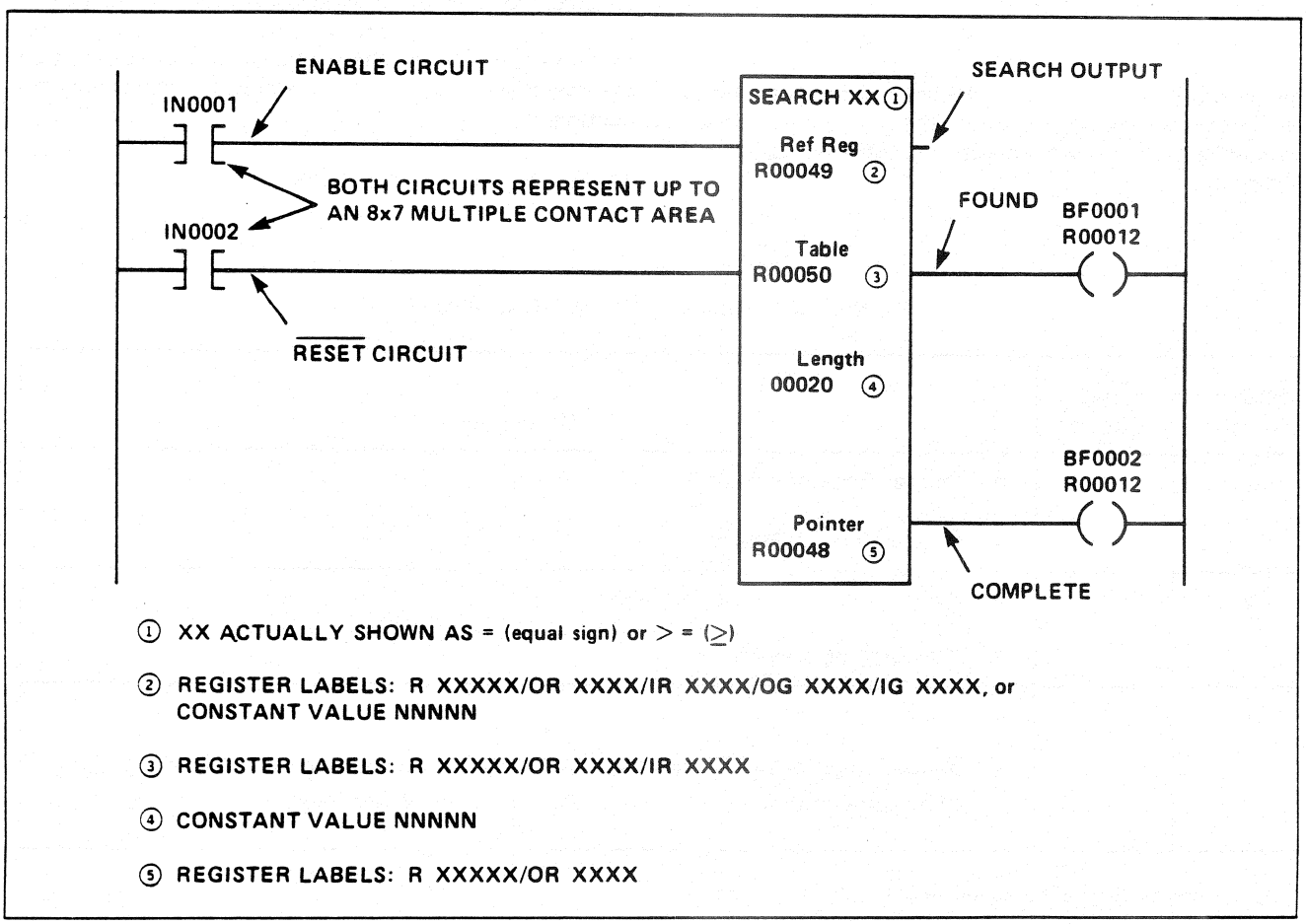


Figure SE-1. Search Type Functions Circuit (Typical)

8-23-1. SPECIFICATIONS

The programming specifications for the Search type functions are listed here.

FUNCTION BLOCKS

The Search type function blocks require 2 horizontal contact spaces by 5 parallel paths on the display. The function block can be positioned anywhere on the 10x7 contact area of the ladder diagram.

REFERENCE REGISTER

The Reference Register contains the value which is to be compared to the value of the registers in the Table. The Reference Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)
- Constant value from -32,768 to +32,767

The letters R, OR, IR, OG, or IG precede the display of the Reference Register group or register number. If a constant value is selected, no letters precede the value on the display.

TABLE

The Table contains the number of registers which are compared to the Reference Register. The first register in the Table, specified by the programmer and displayed in the function block, is selected as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the display of the Table register or group number.

LENGTH

The Length, or number of registers contained in the Table, is a constant value ranging from 1 to 32,767.

POINTER REGISTER

The Pointer Register contains the number of the register in the Table which contains the value equal to or greater than the value of the Reference Register. The input circuits control the Pointer Register as listed in Table SE-1. If the value of the Pointer Register is negative or greater than the Table length, the enable circuit has no effect on the operation, and the complete output remains nonconducting.

The Pointer Register is specified as a:

TABLE SE-1. SEARCH FUNCTION INPUTS

Enable ① Circuit	$\overline{\text{Reset}}$ Circuit	Description
0 or 1	0	• Pointer Register held at 0 • No searching takes place
0	1	• Pointer Register held at last value • No searching takes place
1	1	• Searching takes place • Pointer Register equals either zero or the number of the register in the Table found to be equal to or greater than the value of the Reference Register.

① 0 = nonconducting, 1 = conducting

- Holding register (R)
- Output register (OR)

The letters R or OR precede the display of the Pointer Register number.

INPUT CIRCUITS

The enable circuit and $\overline{\text{reset}}$ circuit control the operation of the Search type functions, as listed in Table SE-1. These input circuits consist of up to 8 horizontal contacts by 7 parallel paths located on the ladder diagram to the left of the function block. The rules for programming contacts listed in Section 6 apply to these circuits.

OUTPUT CIRCUITS

The Search type functions each have 3 outputs, as listed in the following 3 points:

1. The search output simply follows or repeats the conducting or nonconducting state of the enable circuit.

2. The found output conducts when the Search type function searches and finds a value in the Table equal to or greater than the value of the Reference Register. This output continues to conduct until the next scan of the ladder diagram at which time the search resumes from the current location as contained in the Pointer Register. If, during the next scan, another value is found in the Table equal to or greater than the value of the Reference Register, the found output continues to conduct until the Table is completely scanned.

3. Whenever the search, during the last scan, did not find any registers in the Table equal to or greater than the Reference Register, the complete output conducts for one scan.

Note

Outputs from the Search type function blocks can be left unconnected in the network.

8-23-2. APPLICATIONS

The Search type functions allow data contained in registers to be searched for a range of values. One specific application is shown in Figure SE-2 where a table is used to store production data. Observe the Figure and note the following 4 points:

1. The Search Equal To function, when enabled, is used to search for the next available empty location in the table starting at register R00280. (The Reference Register contains the constant 0.)

2. The table is 200 registers long, starting at R00280.

3. When an empty location is found in the table, the Move R-T function (094) loads the location with the production data from IR0010.

4. The network shown in Figure SE-2 does not show the functions associated with monitoring or clearing the data contained in the table.

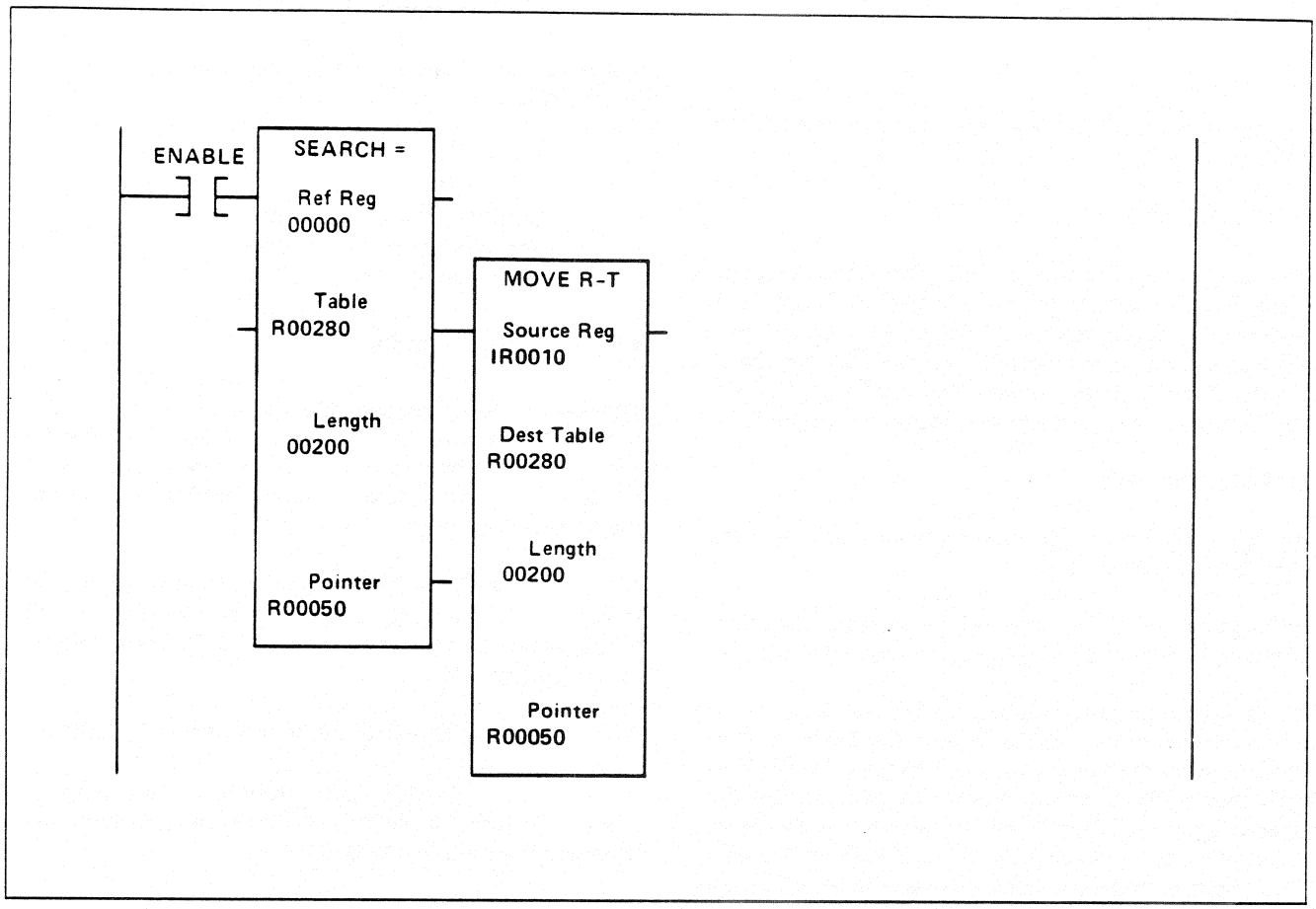


Figure SE-2. Search Application

8-24. ASCEND SORT (108)

The Ascend Sort function arranges data in a table of registers according to an ascending order of least to greatest. When the sort is completed, the lowest numbered register contains the smallest value, etc.

Although data is rearranged in a table of registers called Table 1, a second, "companion" Table is also required. Its function is to contain entirely different identifying data associated with the Table 1 values. Table 2 is the same length as Table 1, and each register in it is paired with a directly corresponding register in Table 1. (For example, the lowest numbered Table 2 register is paired with the lowest numbered register in Table 1.)

When an Ascend Sort function occurs, data in both Tables is rearranged, as determined by the values contained in the Table 1 registers. The results of a typical Ascend Sort operation are shown in Table AS-1.

A typical circuit containing the Ascend Sort function is shown in Figure AS-1. Observe the Figure and note the following 3 points:

1. The Ascend Sort operation occurs during each scan when the enable circuit is conducting.

2. The number of scans required to perform the sorting operation depends on the number of registers contained in the Tables and the amount of reordering required. The complete output conducts when the Ascend Sort function completes the sorting operation.

3. The complete output conducts during each scan when the Ascend Sort operation has been completed.

8-24-1. SPECIFICATIONS

The programming specifications for the Ascend Sort function are listed here.

FUNCTION BLOCK

The Ascend Sort function block requires 2 horizontal contact spaces by 5 parallel paths on the display. The function can be positioned anywhere on the 10x7 contact area of the ladder diagram.

TABLE 1, TABLE 2

Table 1 contains registers the values of which are to be sorted in ascending order, least to greatest. See the example shown in Table AS-1.

Table 2 contains the registers which are paired with the Table 1 registers. Their contents may be identifying data used in association with the directly corresponding

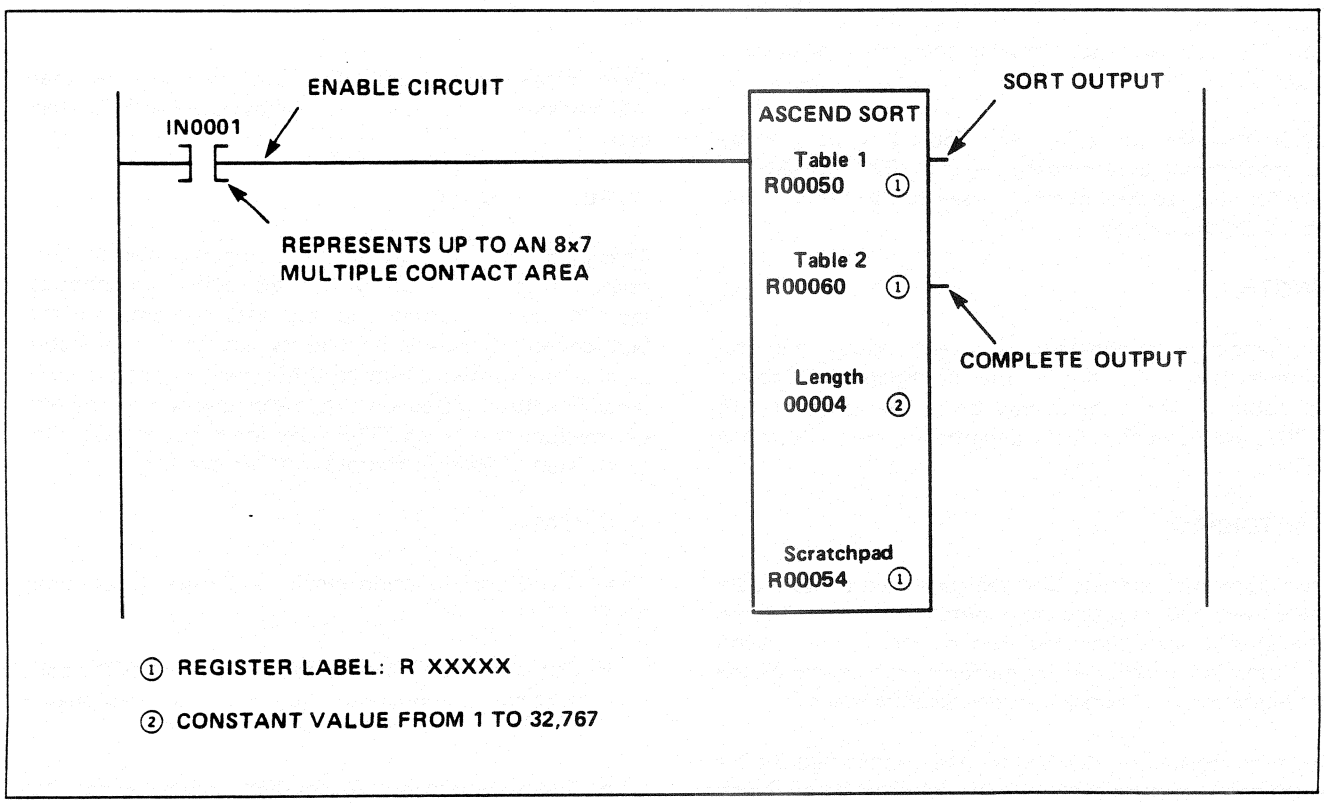


Figure AS-1. Ascend Sort Function Circuit (Typical)

TABLE AS-1. ASCEND SORT EXAMPLE

Tables Before Ascend Sort			
Table 1		Table 2	
Register	Contents ①	Register	Contents ①
R00050	Binary Value = 84	R00060	Binary Value = 10
R00051	Binary Value = 1021	R00061	Binary Value = 21
R00052	Binary Value = 5	R00062	Binary Value = 14
R00053	Binary Value = -20	R00063	Binary Value = 61
Tables After Ascend Sort			
Table 1		Table 2	
Register	Contents ①	Register	Contents ①
R00050	Binary Value = -20	R00060	Binary Value = 61
R00051	Binary Value = 5	R00061	Binary Value = 14
R00052	Binary Value = 84	R00062	Binary Value = 10
R00053	Binary Value = 1021	R00063	Binary Value = 21

① The contents of a register are actually binary digits, but a decimal equivalent is shown here for convenience.

values in Table 1. There is, however, no necessity for Table 2 to contain data, although there must be a Table 2 used.

The first register in Table 1 and Table 2 is specified by the programmer as a holding register (R). The letter R precedes the register number assigned to Table 1 and Table 2 on the display.

LENGTH

The Length is a constant value entered by the programmer. It specifies the number of registers in Table 1 and Table 2. The Length may be in the range of 1 to 32,767, although the Table Lengths are normally much smaller.

SCRATCHPAD

The Scratchpad consists of 4 registers used to store data when over 100 registers are contained in the Tables. (The overall scan time required by the entire Ascend Sort function is reduced by performing portions of the sort operation in successive ladder diagram scans.)

The first register of the Scratchpad is specified by the programmer as a holding register (R). The letter R precedes the register number assigned to the Scratchpad on the display.

Even if the Ascend Sort function involves fewer than 100 registers, 4 registers must be assigned as a Scratchpad.

ENABLE CIRCUIT

The enable circuit conducts, allowing the Ascend Sort operation to take place. When the circuit is nonconducting, no data is sorted, and any data contained in the Scratchpad registers is held in its last state. The enable circuit can consist of up to 8 horizontal contacts by 7 parallel paths located on the ladder diagram to the left of the function block. The rules for programming contacts listed in Section 6 apply to these circuits.

OUTPUTS

The Ascend Sort function block has 2 outputs, as noted here:

- The sort output simply follows or repeats the conducting or nonconducting state of the enable circuit.
- The complete output conducts when the sort has been completed. (See the Scratchpad description.)

Note

The outputs from the Ascend Sort function block can be left unconnected in the network.

8-24-2. APPLICATIONS

The Ascend Sort function is used to assemble or rearrange data into a desired order. One example of the rearrangement of data is shown in the following example where data is received from 3 identical machines. Observe Figure AS-2 and note the following 4 points:

1. From each machine data specifying the type of part being produced is entered along with the cycle start and cycle complete signals.

2. The received part code data is stored in a Table for subsequent use.

3. The time required to produce each part is stored in a second Table.

4. Although parts are produced at random, the Ascend Sort function actually sorts the parts according to the numerical value of the part codes.

The portion of the ladder diagram used to collect and sort data for machine No. 1 is shown in Figure AS-3. Observe the Figure and note the following 3 points:

1. A One-Shot Timer function (034) begins timing each time the cycle start input, IN0001, first conducts.

2. The Actual Register of the Up Counter function (038) is incremented each time the complete input conducts. This register is used as the Pointer Register for both of the Move Register-to-Table functions (094).

3. The Ascend Sort function is enabled when input IN0008 first conducts. The function performs an ascend sort on the part codes contained in Table R00100.

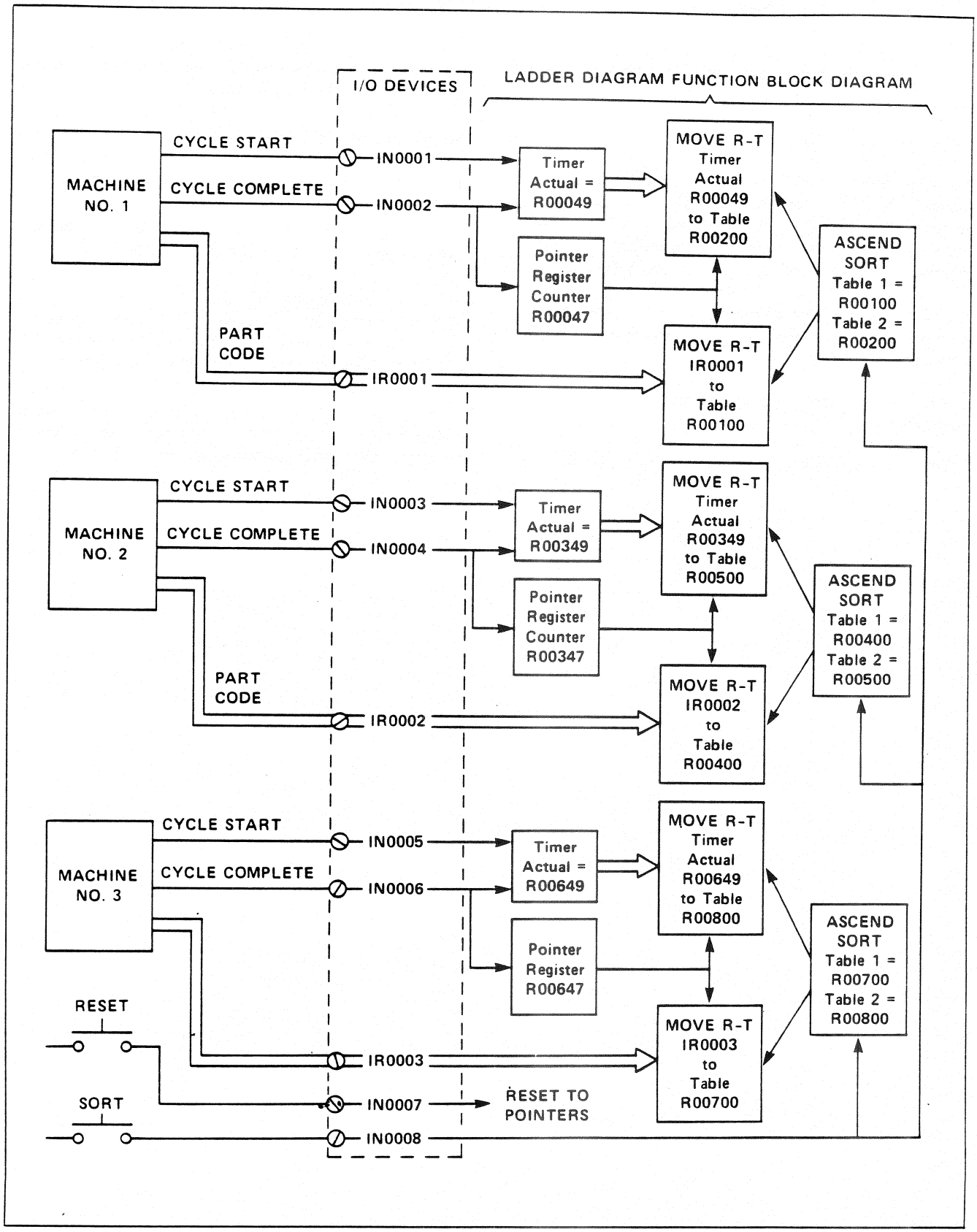


Figure AS-2. Ascend Sort Machine Application Overview

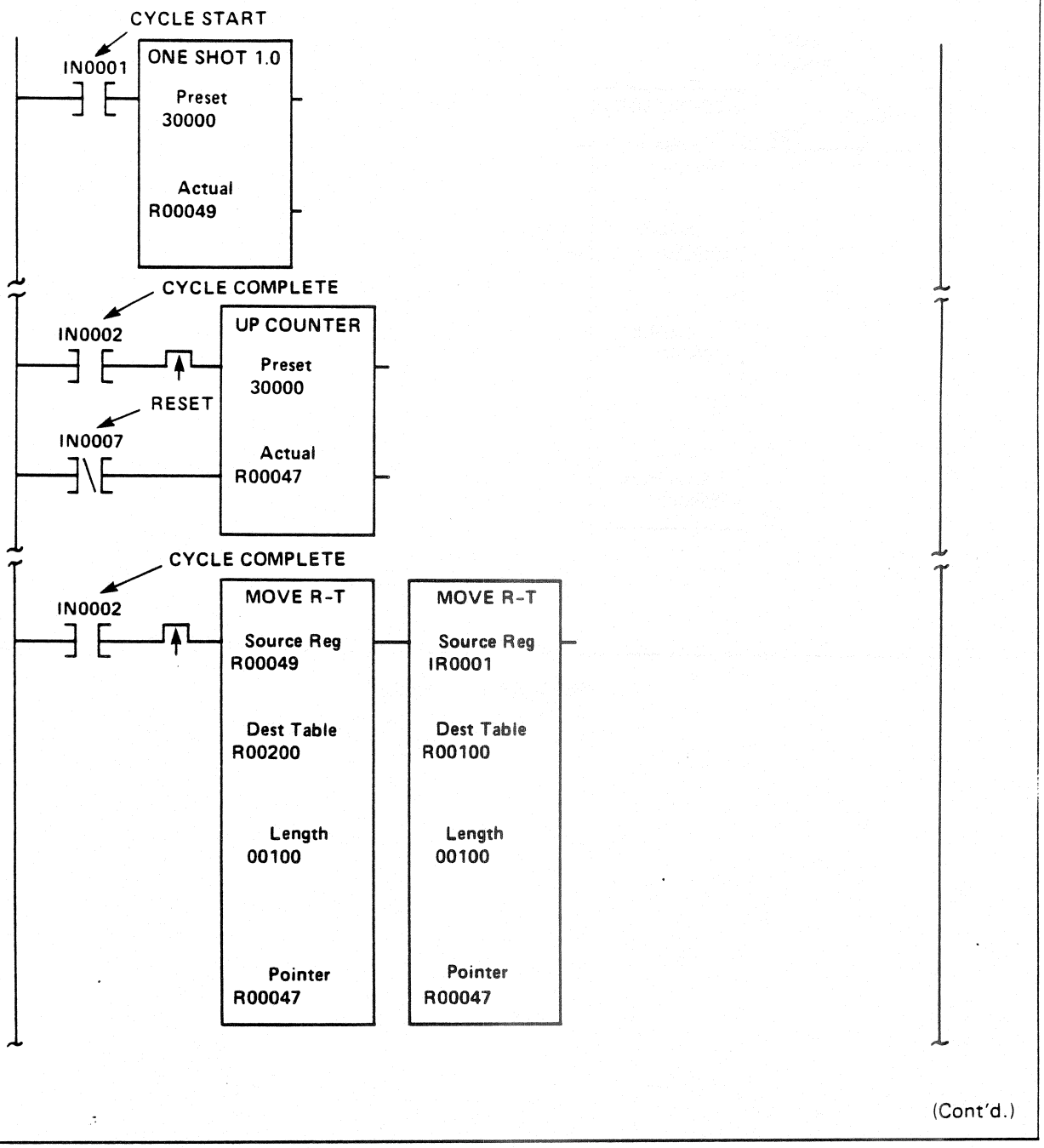


Figure AS-3. Ascend Sort Machine Application (Partial)

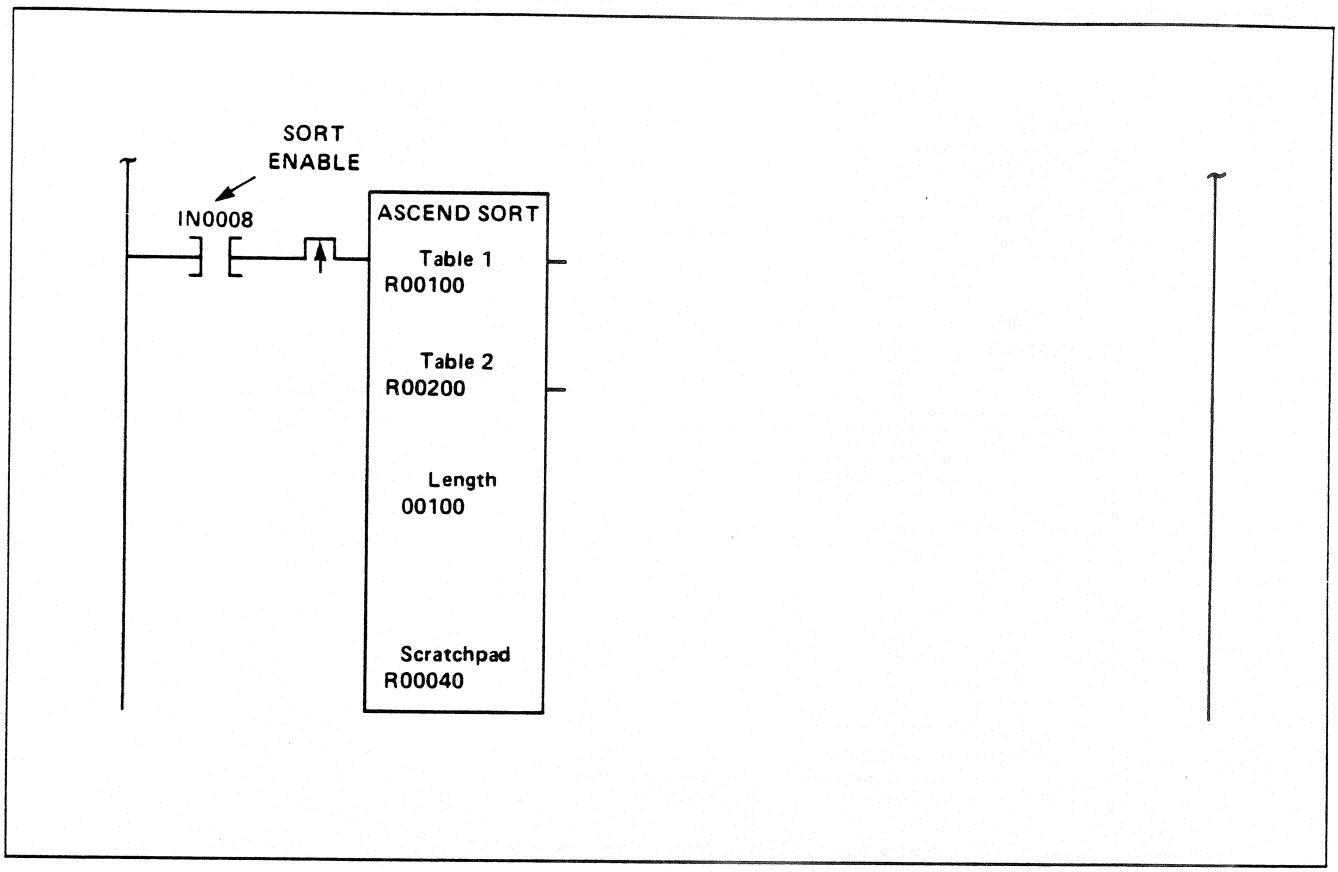


Figure AS-3. (Cont'd.)

8-25. FIRST IN (103), FIRST OUT (104), LAST OUT (105)

The First In Stack and First Out Fetch programmable functions are used together to form a FIFO stack operation, as described in Paragraph 8-25-1. The First In Stack and Last Out Fetch functions are used together to form a FILO stack operation as described in Paragraph 8-25-2.

8-25-1. FIFO STACK

The First In Stack and First Out Fetch functions are designed to be used together to store and retrieve data from a Table referred to as a Stack Table. The combination of the First In Stack and First Out Fetch functions is referred to as a FIFO stack operation. The first data entered into the FIFO stack is the first data retrieved.

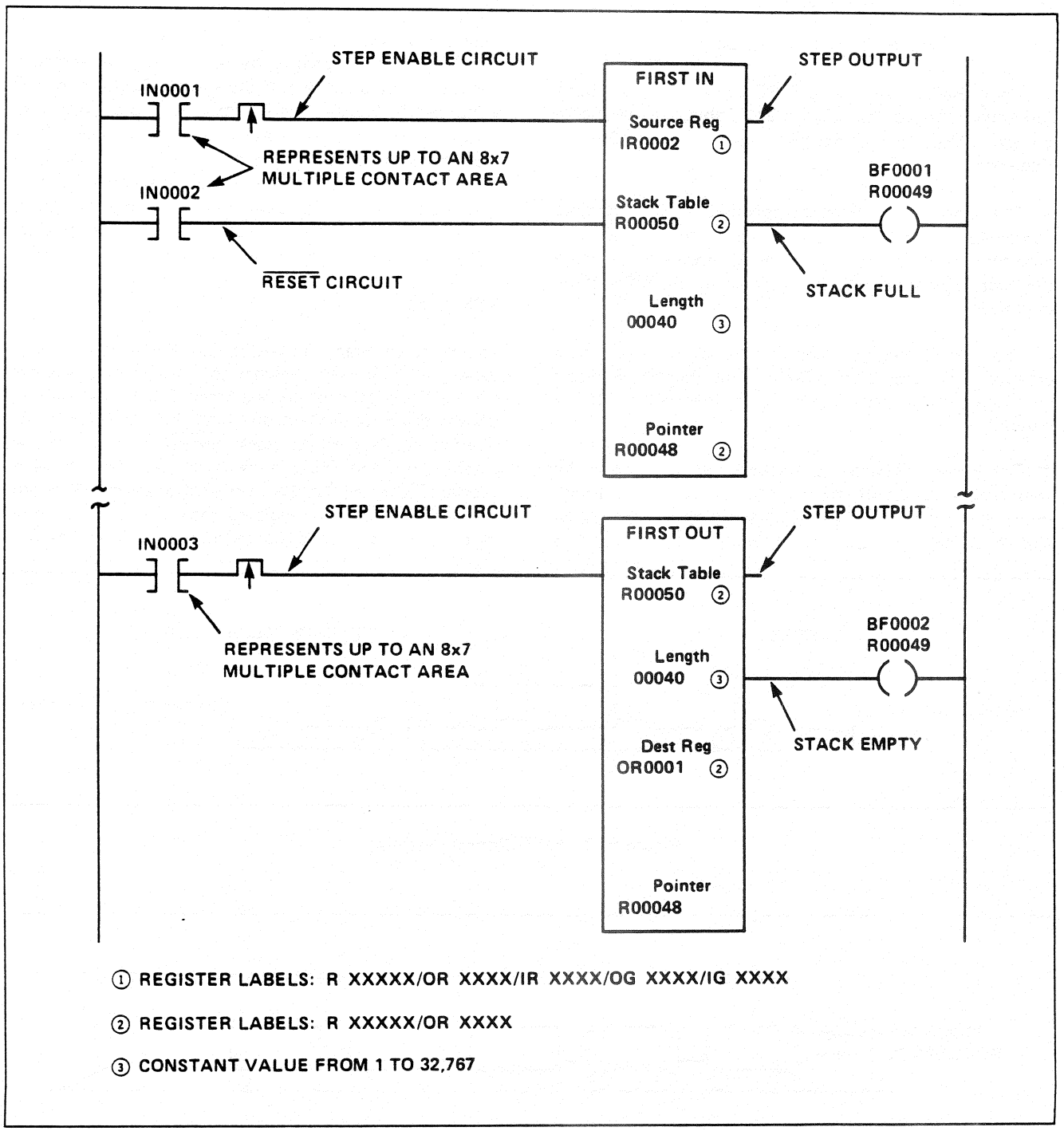


Figure FI-1. FIFO Stack Operation Circuit (Typical)

Figure FI-2 shows a railroad boxcar analogy in which boxcars are entered or stacked into a railroad siding. When the cars are unloaded from the opposite end of the siding, the operation is similar to the FIFO stack operation. In the FIFO stack operation, data is entered using the First In Stack function and removed using the First Out Fetch function.

A typical circuit containing the First In Stack and First Out Fetch functions is shown in Figure FI-1. Observe the Figure and note the following 5 points:

1. When used together as a FIFO stack operation, both functions refer to the same Stack Table with the same Length. Also, the functions use the same Pointer Registers.

2. The Destination Register specifies where the data contained in the Stack Table is to be transferred. The data sent to the Destination Register always comes from the first register of the Stack Table in the FIFO stack operation.

3. The Source Register specifies the source of the data entered into the Stack Table. The Pointer Register directs where in the Stack Table the data is to be entered.

4. The Pointer Register is incremented by 1 when the step enable circuit of the First In Stack function conducts. It is decremented by 1 when the step enable circuit of the First Out Fetch function conducts. The Transitional functions (007) are used in the step enable circuit of Figure FI-1 to prevent incrementing or decrementing during each scan.

5. The Pointer Register is cleared to 0 when the First In function is reset.

A summary of the FIFO stack operation follows. The Source Register enters data, starting at the first register in the Stack Table. Successive data is entered into successive registers of the Table. The data once entered remains in sequence with the Pointer Register and specifies the last data entered. If the Pointer Register contains a number 6, it indicates 6 registers in the Stack Table contain data. The next data entered would be entered into the seventh register in the Stack Table. Also, the next data to be fetched would always come from the first register in the Table. At the same time the Pointer Register is decremented, and all the data in the register is shifted down 1 register in the Stack Table.

8-25-2. FILO STACK

The First In Stack and Last Out Fetch functions are designed to be used together to store and retrieve data from a Table referred to as a Stack Table. The combination of the First In Stack and Last Out Fetch function is referred to as a FILO stack operation. The first data entered into the FILO stack is the last data to be retrieved. Figure FI-3 shows a boxcar analogy where boxcars are entered, or "stacked into," a railroad siding. When the cars are loaded and unloaded from the same

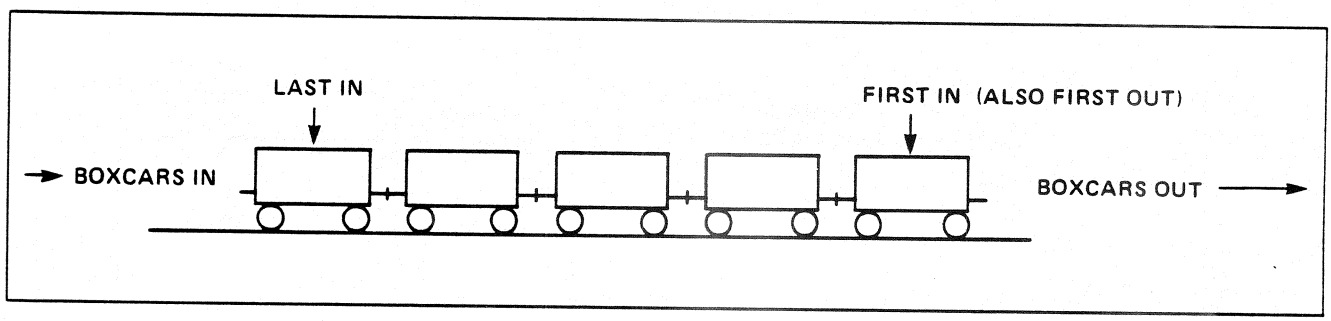


Figure FI-2. FIFO Boxcar Analogy

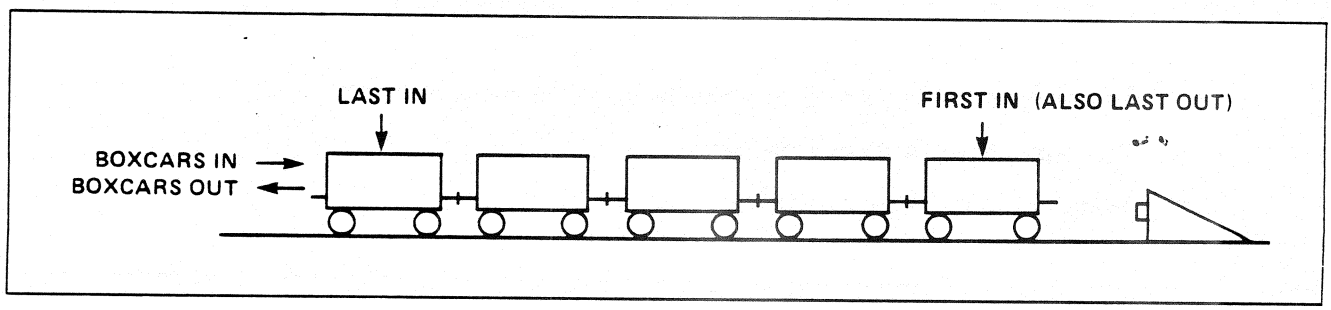


Figure FI-3. FILO Boxcar Analogy

end of the siding, the operation is similar to the FILO stack operation. In the FILO stack operation, data is entered using the First In Stack function, and data is removed using the Last Out Fetch function.

A typical circuit containing the First In Stack and Last Out Fetch functions is shown in Figure FI-4. Observe the Figure and note the following 4 points:

1. When used together as a FILO stack operation, both functions refer to the same Stack Table with the same Length. Also, the functions use the same Pointer Register.

2. The Destination Register specifies where the data contained in the Stack Table is to be transferred. In the FILO stack operation, the data sent to the Destination

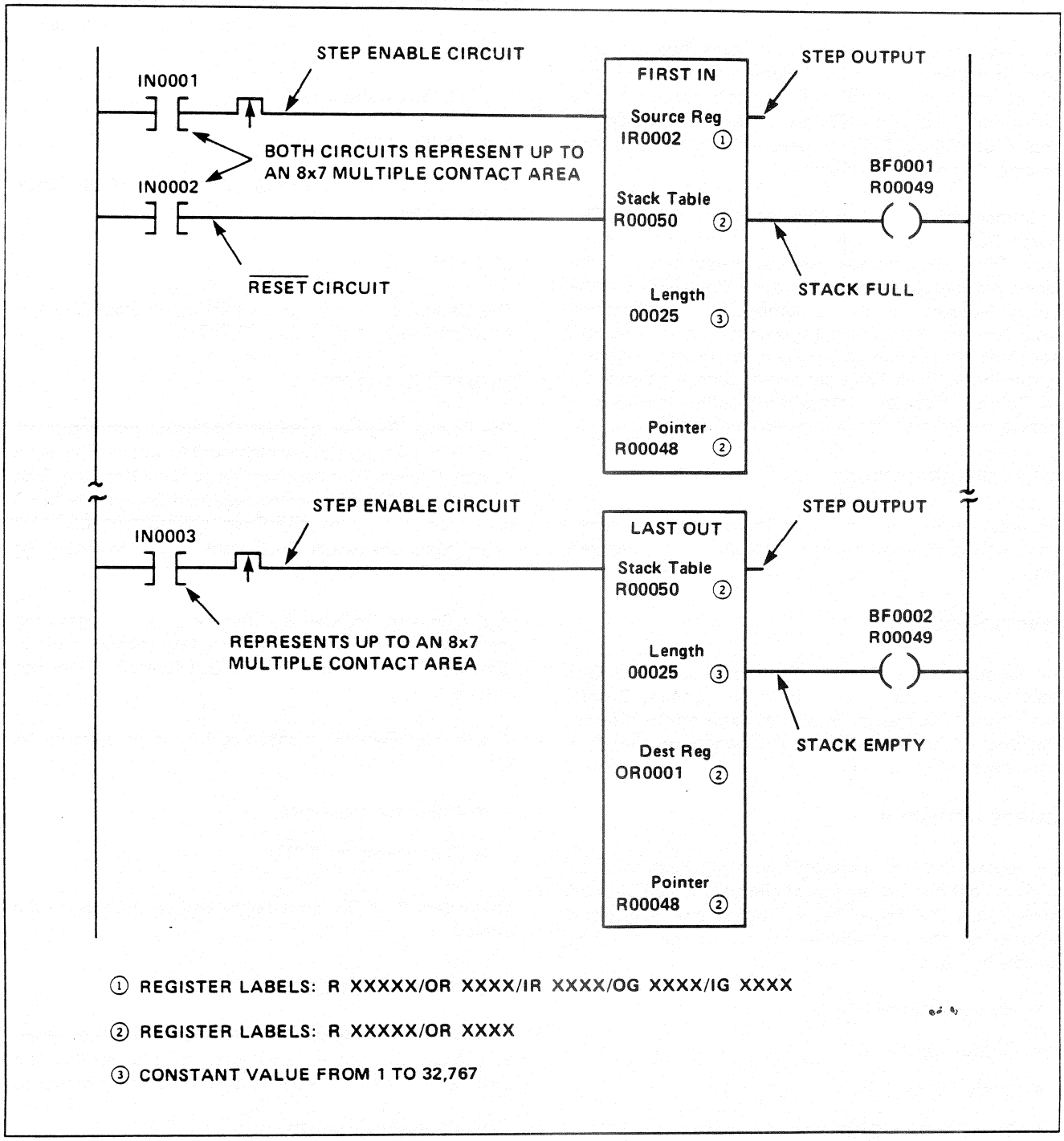


Figure FI-4. FILO Stack Operation

Register always comes from the highest register in the Stack Table containing data. This is the data loaded in first.

3. The Source Register specifies the source of the data entered into the Stack Table. The data is always entered into the Register of the Stack Table, as directed by the Pointer Register.

4. The Pointer Register is incremented by 1 when the step enable circuit of the First In Stack function conducts. It is decremented by 1 when the step enable circuit of the Last Out Fetch function conducts. The Transitional functions (007) are used in the step enable circuits of Figure FI-4 to prevent incrementing and decrementing during each scan.

A summary of the FILO stack operation follows. The Source Register enters data into the first register in the Stack Table. The Pointer Register keeps track of the highest register containing data in the Stack Table. If the Pointer Register contains a number 6, it is indicating that 6 registers in the Stack Table contain data. The next data to be fetched would always come from the highest register in the Stack Table containing data as directed by the Pointer Register. Immediately after the data is fetched, the Pointer Register is decremented by 1.

8-25-3. SPECIFICATIONS

The programming specifications for the First In Stack, First Out Fetch, and Last Out Fetch functions are listed here.

FUNCTION BLOCK

The function block for the First In Stack, First Out Fetch, and Last Out Fetch functions requires 2 horizontal contact spaces by 5 parallel paths on the display. The function can be positioned anywhere on the 10x7 contact area of the display.

SOURCE REGISTER

The Source Register, associated with the First In Stack function, defines the source of the data which is to be entered into the Stack Table. The Source Register is specified by the programmer as any of the following registers or groups:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the display of the Source Table register or group number.

STACK TABLE

The Stack Table consists of a group of registers used to store data for the FIFO stack and FILO stack operations. The Stack Table register, specified by the programmer and displayed in the function block, is the first register in the Stack Table. The Stack Table is selected as either a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the display of the Stack Table register.

LENGTH

The Length, or number of registers in the Stack Table, is a constant value from 1 thru 32,767.

POINTER REGISTER

The Pointer Register specifies the highest numbered register in the Stack Table which contains data. The same Pointer Register number must be specified for the First In Stack and First Out Fetch functions used in the FIFO stack operation. Also, the First In Stack and Last Out Fetch functions of the FILO stack operations must use the same Pointer Register.

If the Pointer Register is either negative or larger than the Length of the Stack Table, the step enable circuit of the First In, First Out, or Last Out functions does **not** initiate a step.

The Pointer Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)

The letters R or OR precede the register number on the display.

DESTINATION REGISTER

The Destination Register specifies the register into which data from the Stack Table is to be transferred. The Destination Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)

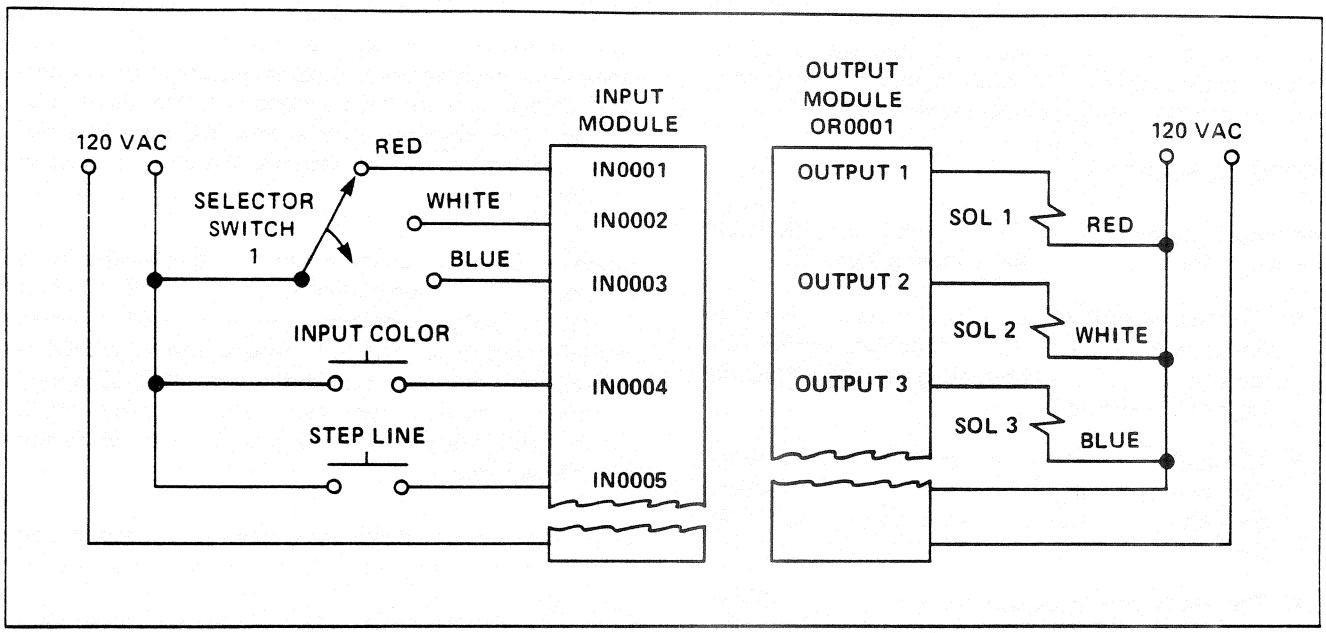


Figure FI-5. FIFO Application Wiring

The letters R or OR precede the register number on the display.

INPUT CIRCUITS, FIRST IN STACK FUNCTION

Input circuits control the First In Stack function as listed in the following 2 points:

1. When the step enable circuit conducts, it provides the following operations during each scan of the ladder diagram:

- Pointer Register is incremented by 1.
- Data is transferred from the Source Register to the location of the Stack Table as directed by the Pointer Register.

The Transitional function (007) can be used in the step enable circuit to allow the stepping to occur only once when the enable circuit first conducts. When the circuit is nonconducting, no data is transferred, and the Pointer Register remains unchanged.

2. The $\overline{\text{reset}}$ circuit of the First In Stack function conducts, allowing normal operation. When the $\overline{\text{reset}}$ circuit is nonconducting, the Pointer Register is reset and held at 0, and no transferring of data occurs.

These 2 input circuits can consist of up to 8 horizontal contacts by 7 parallel paths located to the left of the function block on the ladder diagram. The rules for programming contacts listed in Section 6 apply to these circuits.

INPUT CIRCUIT, LAST OUT FUNCTION

When the step enable circuit of the First Out Fetch function conducts, it provides the following operations during each scan of the ladder diagram:

- The data, as directed by the Pointer Register, is transferred from the Stack Table to the Destination Register.
- The Pointer Register is decremented by 1.

The Transitional function can be used in the step enable circuit to allow the stepping to occur only when the enable circuit first conducts.

This circuit can consist of up to 8 horizontal contacts by 7 parallel paths located to the left of the function block on the ladder diagram. The rules for programming contacts listed in Section 6 apply to these circuits.

INPUT CIRCUIT, FIRST OUT FUNCTION

When the step enable circuit of the First Out function conducts, it provides the following operations during each scan of the ladder diagram:

- The data from the first register of the Stack Table is transferred to the Destination Register.
- Data contained in each register of the Stack Table is shifted down in succession 1 register.
- The Pointer Register is decremented by 1 count.

This circuit can consist of up to 8 horizontal contacts by 7 parallel paths located to the left of the function block on the ladder diagram. The rules for programming contacts listed in Section 6 apply to these circuits.

OUTPUT CIRCUITS

The output circuits for the FIFO stack and FILO stack operations are described in the following 3 points:

- The step output of the First In Stack, First Out Fetch, and Last Out Fetch functions simply follows or repeats the conducting or nonconducting state of the enable circuits.
- The stack full output of the First In Stack function conducts when the enable circuit conducts and the Pointer Register is equal to or greater than the Length.
- The stack empty output of the First Out Fetch and Last Out Fetch functions conducts when the enable circuit conducts and the Pointer Register equals 0.

Note

The outputs from the stack operation type function blocks can be left unconnected in the network.

8-25-4. FIFO STACK APPLICATION

The FIFO stack operation can be used in situations requiring a controlled sequence to be performed along with variables which can be stored in registers. For example, a machine used to paint objects in several possible colors can be programmed for the day's production. The operator first enters the color choices. Later, the ladder diagram retrieves the color choices in the proper order.

Figure FI-5 shows a simplified wiring diagram for a painting operation controlled by a FIFO stack. The first 3 bits of IG0001 represent the red (IN0001), white (IN0002) and blue (IN0003) colors. Input IN0004 is used to enter data into the Stack Table. IN0005 is used to retrieve data from the Stack Table. Solenoids 1, 2, and 3 enable the colors to be sprayed: red, white and blue, respectively.

Observe the simplified ladder diagram for the application shown in Figure FI-6 and note the following 3 points:

1. The First In Stack function receives data from input group IG0001 each time IN0004 is cycled. The color input depends on the setting of the selector switch.

2. The First Out function outputs the data from the first register in the Stack Table each time switch IN0005 is pressed. The data from the first register in the Stack Table is transferred to output register OR0001.

3. The bits of output register OR0001 are examined to enable CR0001, CR0002, or CR0003, as shown at the bottom of Figure FI-6.

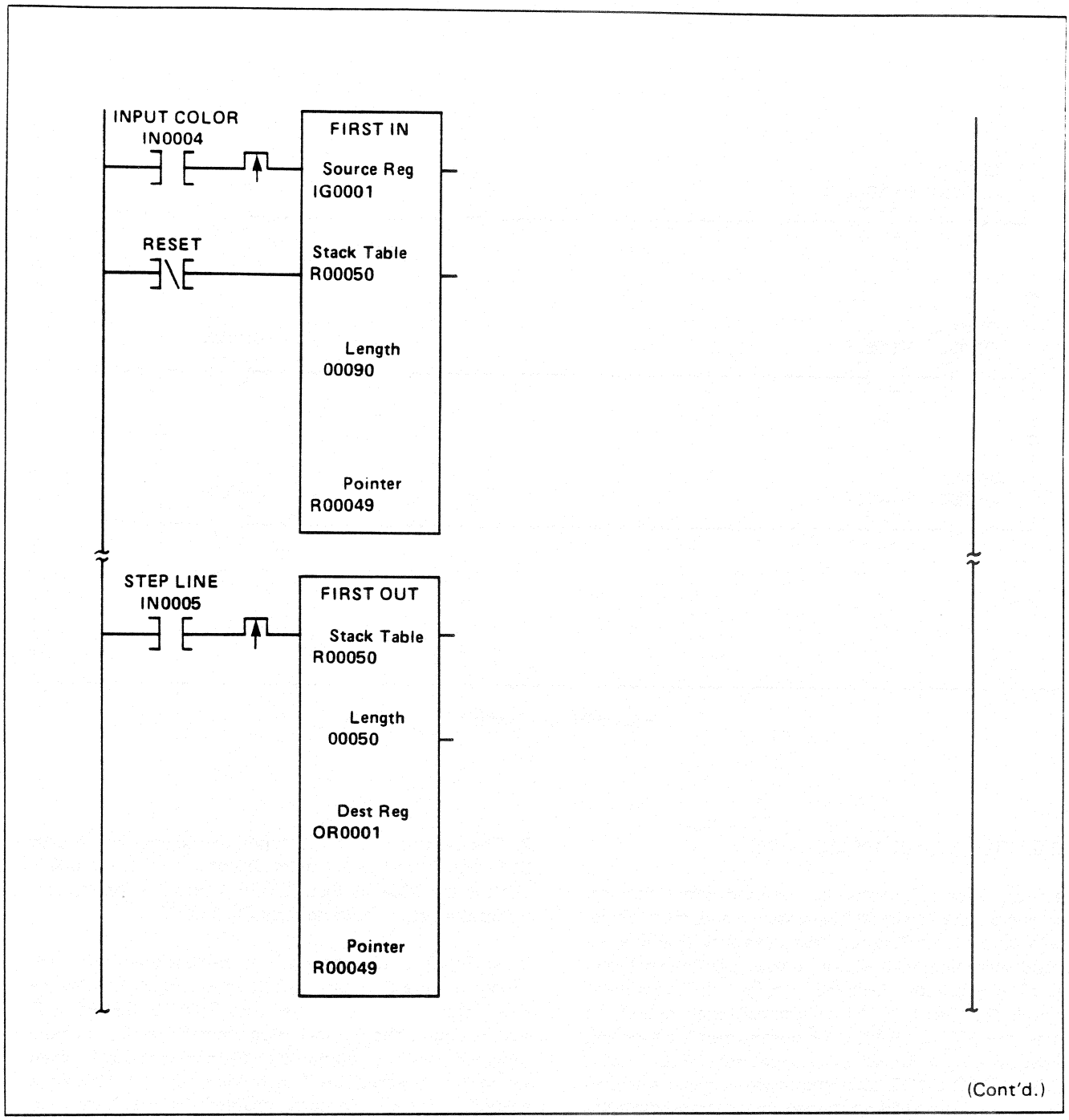


Figure FI-6. FIFO Application

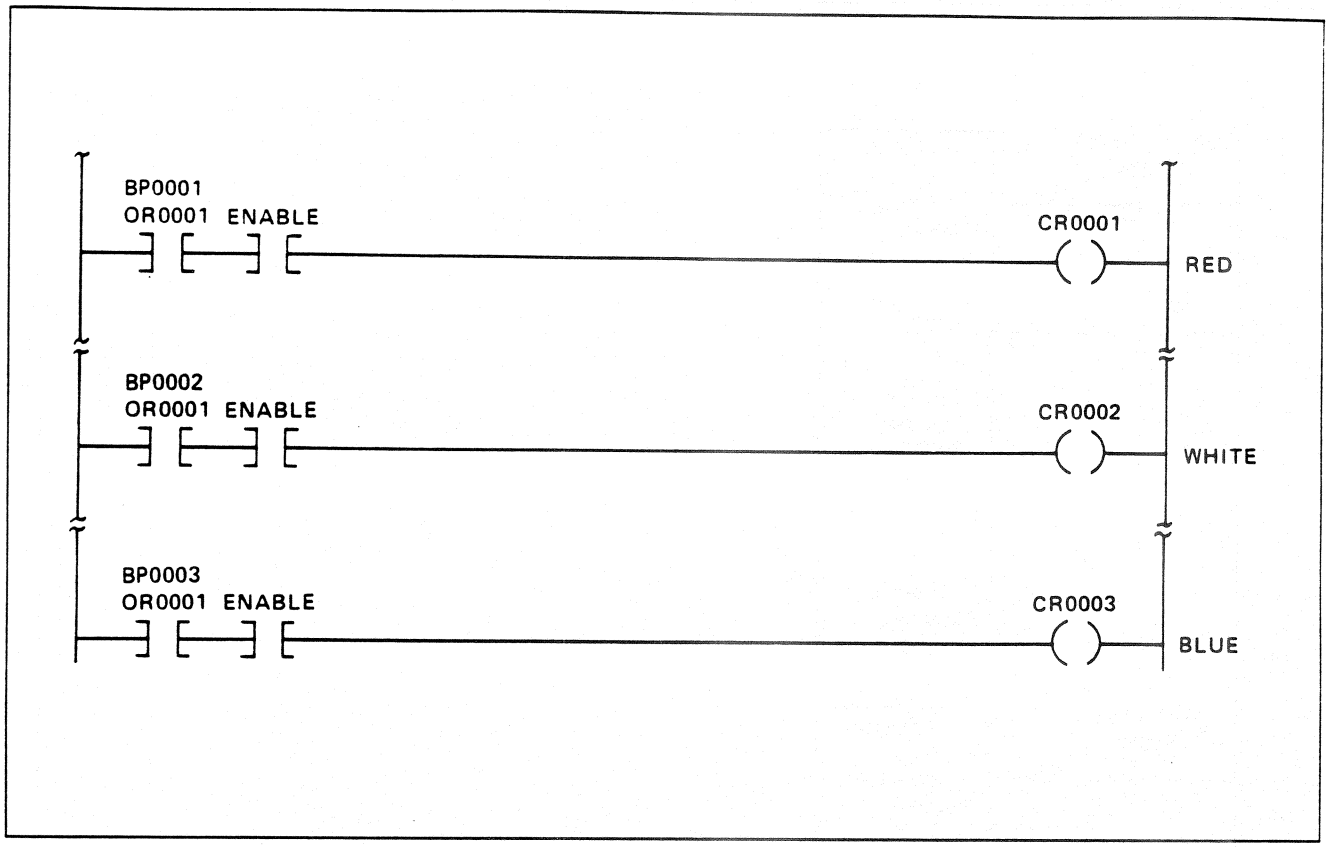


Figure FI-6. (Cont'd.)

8-25-5. FILO STACK APPLICATION

The FILO stack application can be used in any application where data is stored sequentially and must be retrieved on a first-in/last-out basis. One such example is a railroad switching yard where railroad cars are sorted in one of 5 separate dead-end sidings for subsequent retrieval. (See Figure FI-7.) As each car enters one of the sidings, the ENTER RAILCAR pushbutton is depressed, thereby loading a 4-digit BCD code into a FILO stack. The RAILCAR CODE BCD thumbwheel selects one of 5 FILO stacks (1 thru 5) contained in the ladder diagram. The SIDING SELECTOR and EXIT RAILCAR pushbuttons are used to retrieve the railcar code which is available at output register OR0001 and displayed on the operator panel.

Figure FI-8 shows a simplified ladder diagram used to control the operation. Observe the Figure and note the following 5 points:

1. The first Byte Move function (077) transfers the BCD bit pattern from the first 3 digits of IR0001. The BCD Move Pattern of 0321 is displayed as a binary 801.

2. The second Byte Move function transfers the fourth BCD digit from IR0001 by means of Move Pattern 0004. (This is the SIDING SELECTOR 1 thru 5.) The transfer places the data in holding register R00047.

3. A FILO stack operation is programmed for each siding where a Compare R-R function (053) enables the FILO stack operation when the SIDING SELECTOR switch equals the number programmed. In the example shown R00047 is compared to the constant 0001. When siding 1 is selected, bit 1 of holding register R00046 is enabled. This bit, in turn, enables the FILO stack to operate.

4. When the ENTER RAILCAR pushbutton is pressed, the data from the first 3 digits of IR0001, stored in R00048, is entered into the Stack Table.

5. When the EXIT RAILCAR pushbutton is pressed, the data from the Stack Table, as directed by the Pointer Register, is output to OR0001. Output register OR0001 is sent to the LED display contained on the operator panel. Also, OR0001 can be used in other networks of the ladder diagram.

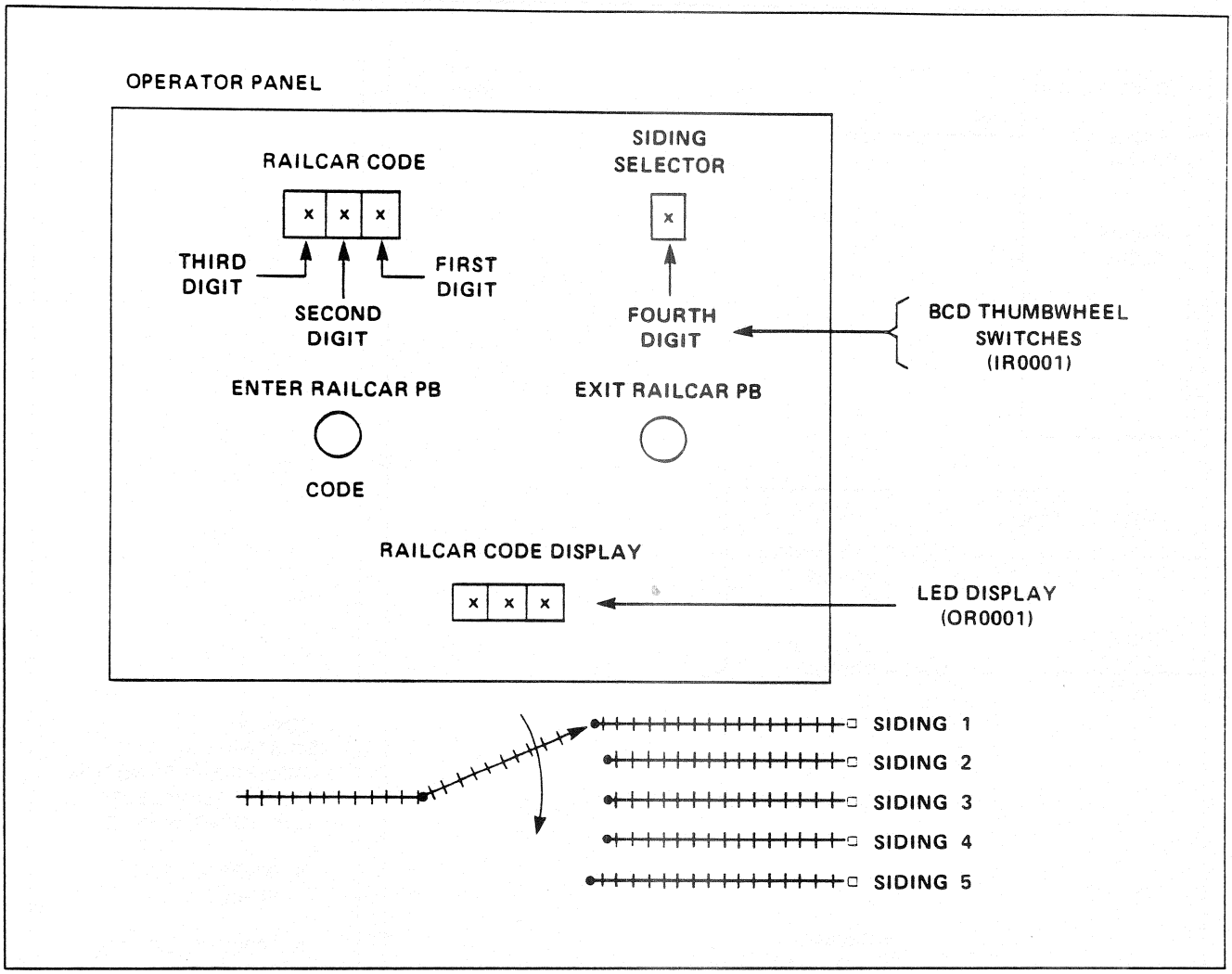


Figure FI-7. Railroad Siding Layout

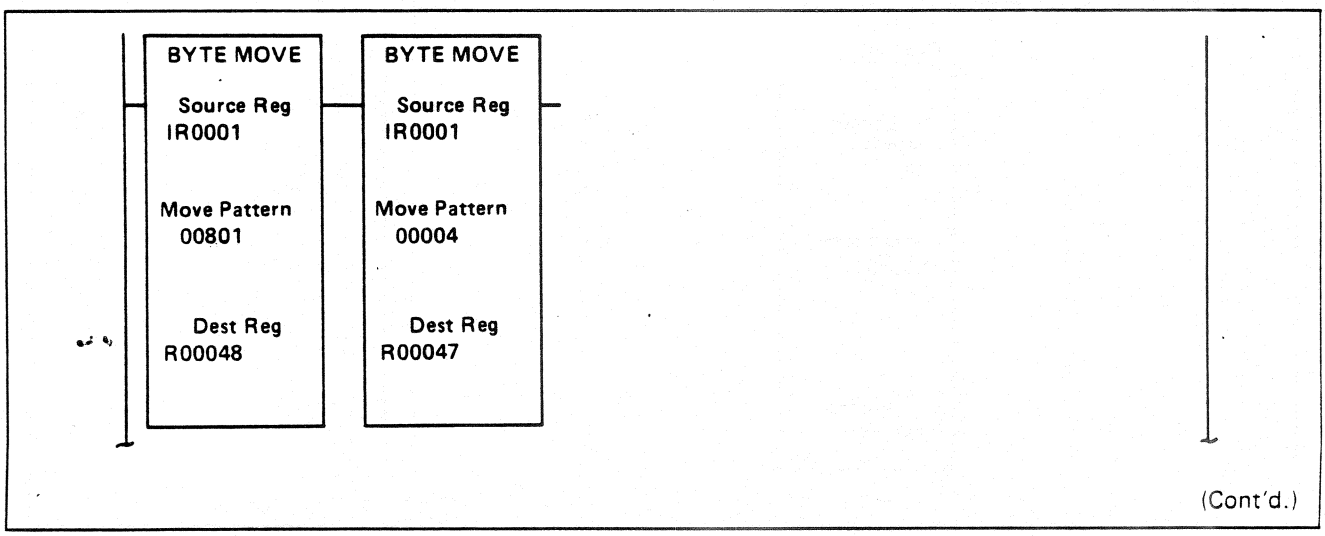


Figure FI-8. Railroad Siding Ladder Diagram

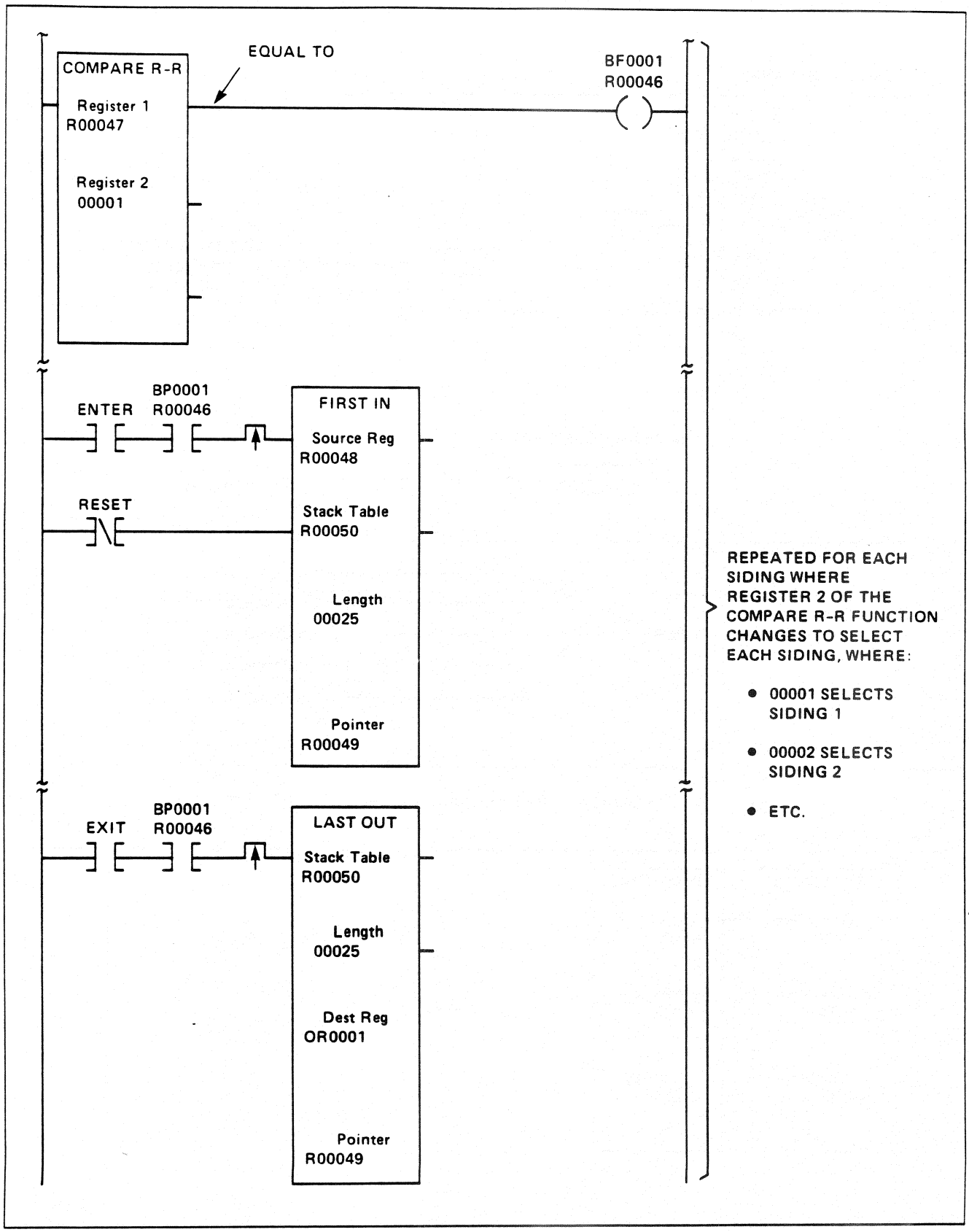


Figure FI-8. (Cont'd.)

8-26. LOCK SCAN (052)

The Lock Scan programmable function allows the ladder diagram scan time to be increased to a predetermined length of time. Also, if the time exceeds the predetermined limit, an "overtime" output from the function can be used to signal that the scan exceeded the predetermined time limit. Figure LS-1 shows a typical circuit containing the Lock Scan function. Observe the Figure and note the following 5 points:

1. The Preset Register defines the time, in milliseconds, of the desired overall program scan. In this example, the contents of R00025 contain, in binary form, a whole number equal to the desired milliseconds. As shown in Figure LS-1, the Preset displays the time in decimal format at the right of the Preset Register.

2. When the enable circuit conducts, the ladder diagram's scan is automatically increased to this time.

3. The Actual Register receives and stores the unextended scan time from the processor. This time is displayed in decimal to the right of the Actual display. In this example, 28 milliseconds would be required if the function was not operating.

4. The overtime output conducts whenever the actual time exceeds the preset time. It is connected to an annunciator from output CR0065 in this example.

5. The output of the Lock Scan function repeats the conducting or nonconducting state of the enable circuit. It is not used in this example.

8-26-1. SPECIFICATIONS

The programming specifications for the Lock Scan function are listed here.

FUNCTION BLOCK

The Lock Scan function block requires 2 horizontal contact spaces by 3 parallel rows on the ladder diagram. The function can be located anywhere in the contact area of the ladder diagram.

PRESET

The Preset value is entered by the programmer as a constant, or it is specified as a register or group as follows:

- Holding register (R)
- Input register (IR)
- Output register (OR)
- Constant value from 10 to 450

The letters R, IR, or OR precede the register number programmed. If a constant is entered, no letter precedes the constant value. This value cannot exceed the watchdog timeout of the processor as measured in milliseconds. If it does, it causes a watchdog timeout to occur.

This value appears to the right of the Preset Register or group display in decimal format or directly as a constant value. The value represents milliseconds.

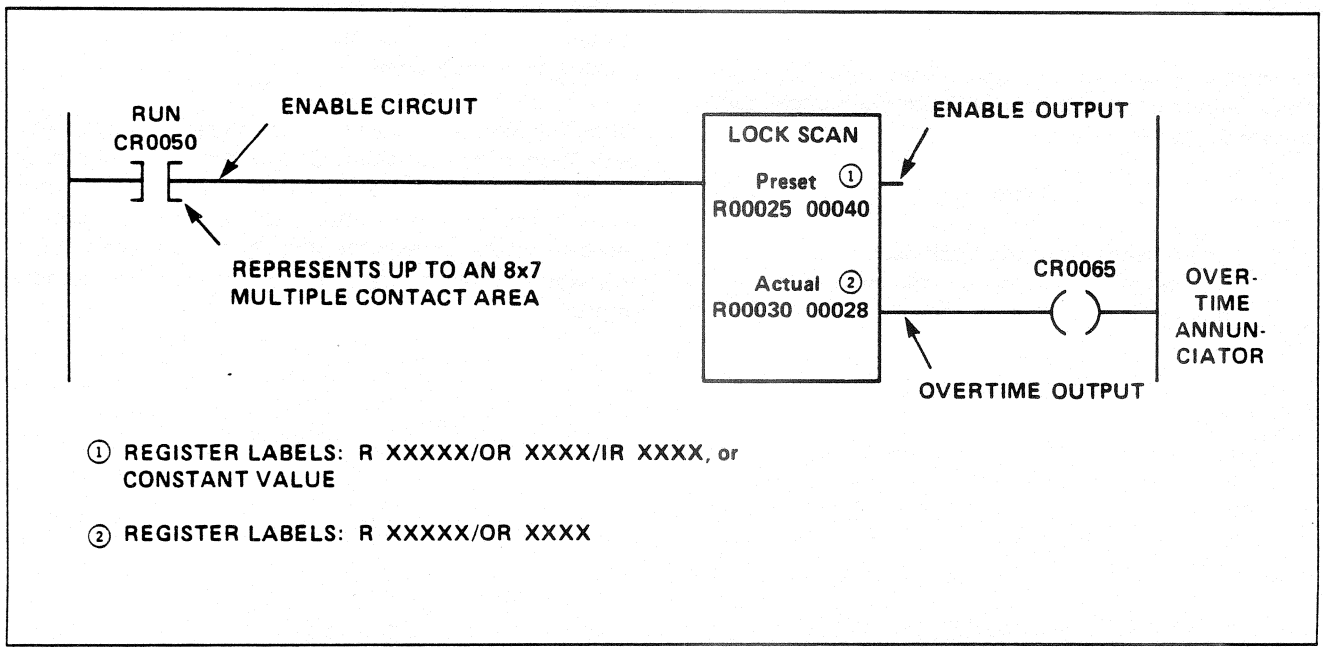


Figure LS-1. Lock Scan Circuit (Typical)

ACTUAL

The Actual value of the ladder diagram scan in milliseconds is displayed automatically to the right of the Register display in decimal format. This is the value that the ladder diagram scan time would have been if the Lock Scan function was not active. The Actual value is loaded by the processor and stored in a register or group that must be specified by the programmer as one of the following:

- Holding register (R)
- Output register (OR)
- Output group (OG)

The letters R, OR, or OG precede the register or group number on the display.

ENABLE CIRCUIT

When the enable circuit conducts, the ladder diagram scan is extended to the time in milliseconds contained in the preset register. The enable circuit consists of up to 8 horizontal contacts located in up to 7 parallel paths located to the left of the Lock Scan function block.

OVERTIME OUTPUT

The overtime output from the Lock Scan function block conducts whenever the enable circuit is conducting and the actual time exceeds the preset time. When the enable circuit is nonconducting, the overtime output is nonconducting.

OUTPUT

The enable output of the Lock Scan function block simply follows or repeats the conducting or nonconducting state of the enable circuit.

Note

The outputs from the Lock Scan function block can be left unconnected in the network.

8-26-2. APPLICATIONS

The following applications are shown for the Lock Scan function:

- Loop control (8-26-2-1)
- Worst-case scan (8-26-2-2)

8-26-2-1. LOOP CONTROL

It is often desirable to perform loop control at precisely known and predetermined intervals of time. Examples of loop control are:

- PID loops
- Feedback loops associated with velocity or positional movement

The Lock Scan function can be used in loop applications to allow the program to begin scanning at precisely defined intervals of time.

8-26-2-2. WORST-CASE SCAN

The Lock Scan function can be used with the Move Register-to-Register function (050) to determine the "worst-case" scan time of a ladder diagram. (See Figure LS-2.) Whenever the overtime output of the Lock Scan function occurs, the Actual time from register R00036 is transferred to the Preset Register, here R00035. After running through several typical machine or process cycles, the preset time of the Lock Scan function will contain the worst-case scan time of the machine or process being monitored.

Often the worst-case scan time can be reduced by careful evaluation of the program. One example of a reduction in scan time is by staggering the calculations within many programmable functions so that they are calculated over several scans rather than during a single scan. In this way the worst-case scan time can be considerably improved.

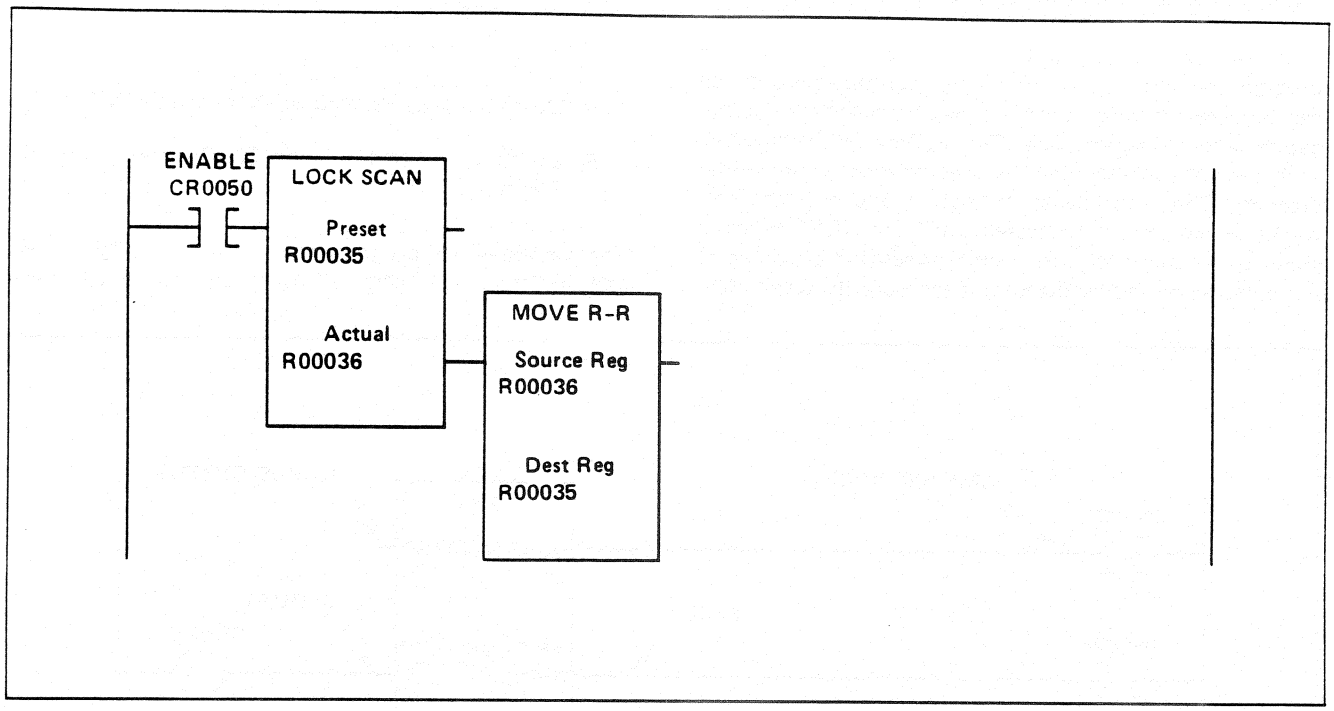


Figure LS-2. Worst Case Scan Circuit

8-27. USER DEFINED SPECIAL FUNCTION (128)

The User Defined Special Function (UDSF) provides the programmer the ability to write "subroutines" which can be accessed one or more times from the main application ladder diagram. Specifically, they can be accessed each time the UDSF block is programmed in the ladder diagram. The subroutines consist of one or more networks which are not accessed until the UDSF block is enabled. At this time, the normal execution sequence of the application ladder diagram is temporarily suspended.

In addition to accessing the same subroutine more than once, the UDSF block also provides the ability to change the following variables:

- Up to 5 inputs, labeled SFIN 1 thru SFIN 5
- Up to 7 registers, labeled SF Register 1 thru SF Register 7

The variables are programmed as part of UDSF block and, in effect, are "carried" to the subroutine whenever

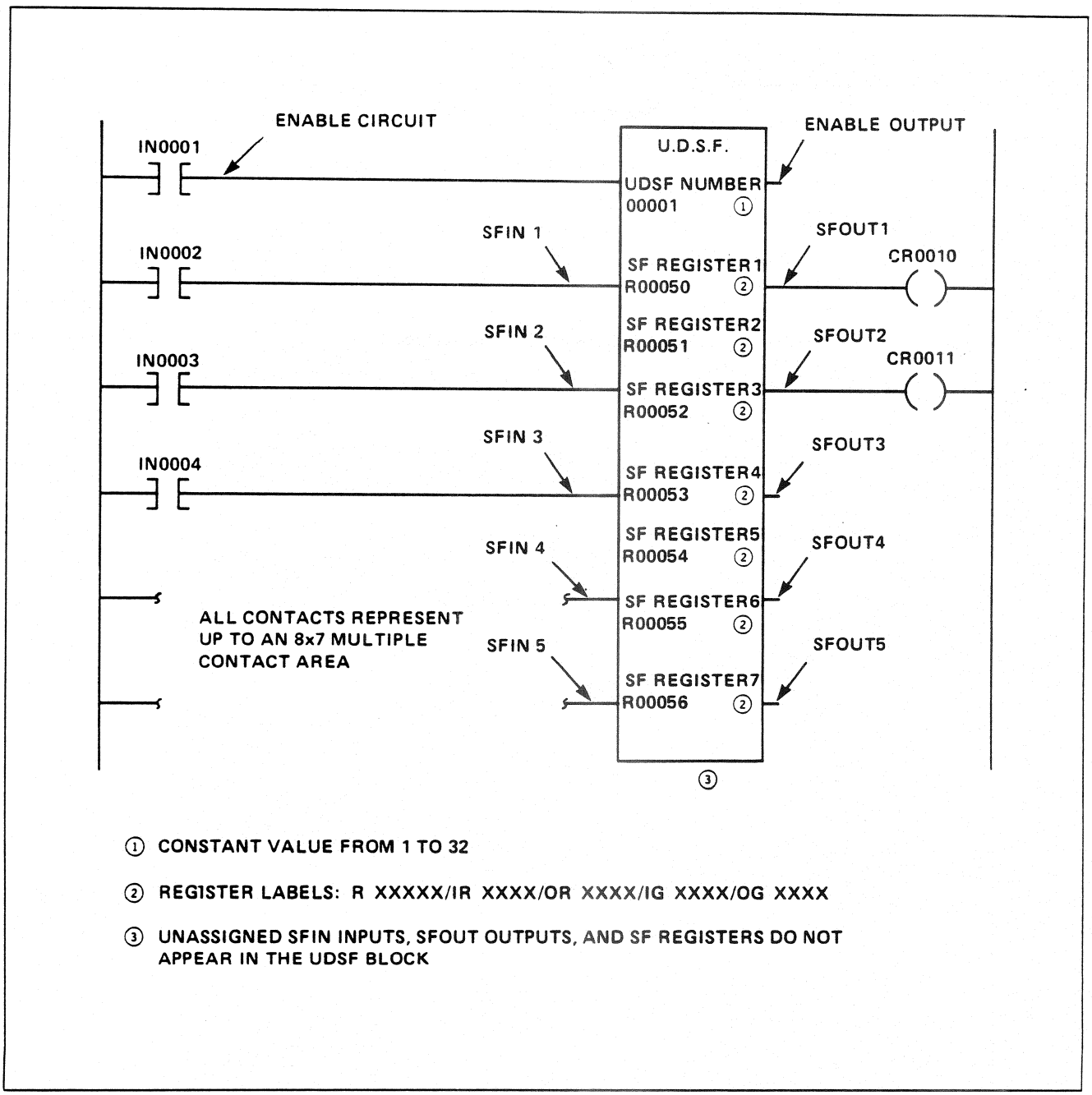


Figure UDSF-1. User Defined Special Function Circuit (Typical)

the UDSF block is enabled. Figure UDSF-1 shows a typical circuit containing a UDSF block. Observe the Figure and note the following 5 points:

1. The enable circuit conducts, allowing the UDSF block to alter the ladder diagram order of execution. The subroutine UDSF Number referenced by the UDSF block will be executed next.

2. The UDSF Number references the subroutine associated with it.

3. The circuits associated with each special function input SFIN 1 thru SFIN 5 define the inputs to the subroutine. These inputs can be different with each UDSF block used to reference the subroutine. Not all need be used.

4. The registers SF Register 1 thru SF Register 7 specify the source of the register data available for use by the subroutine referenced by the UDSF Number. Not all need be used.

5. The outputs from the UDSF block, SFOUT1 thru SFOUT5, are used to energize coils from the UDSF subroutine. Depending on the logic of the subroutine and the state of the SF inputs and SF Registers, the coils associated with the UDSF Number may or may not be energized.

The ladder diagram subroutines associated with the User Defined Special Functions are stored in a memory division distinct and separated from the main application ladder diagram. (See Figure 5-2.) Each of the subroutines contains 1 or more networks, which are entered in a manner similar to the main application ladder diagram. The first network of the application ladder diagram is always executed before any UDSF subroutines are executed.

A single UDSF subroutine can be "called," or referenced, from 2 or more distinct UDSF block functions within the application ladder diagram.

8-27-1. SPECIFICATIONS

The programming specifications for the User Defined Special Functions are listed here.

FUNCTION BLOCK

The UDSF block requires 2 contact spaces by from 3 to 7 parallel paths on the display. (The number of parallel paths is determined by the number of SFIN inputs, SFIN outputs, and SF Registers contained in the UDSF block.) The function block can be positioned anywhere on the 10x7 contact area of the display.

UDSF NUMBER

The UDSF Number is used in one or more UDSF blocks to enable the UDSF subroutine with the same number. Each UDSF subroutine must have a unique reference number. The UDSF Number is a constant value ranging from 1 to 32.

SF REGISTERS 1 THRU 7

SF Register 1 thru SF Register 7 specify the source of the data available for use with the subroutine. Different register data can be specified by each UDSF block. Not all registers need be used. Unused SF Registers are not displayed in the function block.

The SF Registers are specified as any of the following types:

- Holding register (R)
- Input register (IR)
- Output register (OR)
- Input group (IG)
- Output group (OG)

The letters R, IR, OR, IG, or OG precede the register or group numbers on the display.

ENABLE CIRCUIT, SF INPUTS

The enable circuit conducts to alter the ladder diagram execution. The special function inputs SFIN 1 thru SFIN 5 are solved, and the SF Register data is collected for use by the special function subroutine referenced. When the subroutine is completed, the appropriate SF outputs are energized, the SF Registers in the UDSF function block are updated, and the ladder diagram execution continues. Not all special function inputs need be used. Unused SF inputs are not displayed in the function block.

The enable circuit, along with the special function input circuits SFIN 1 thru SFIN 5, together can consist of up to 8 horizontal contacts by 7 parallel paths. The rules for programming contacts listed in Section 6 apply to these circuits.

OUTPUTS

The enable output and the special function outputs, SFOUT1 thru SFOUT5, can be used to enable the coils or other special functions as listed in the following 2 points:

- The enable output simply follows or repeats the

conducting or nonconducting state of the enable circuit.

- The conduction or nonconduction of the special function outputs SFOUT1 thru SFOUT5 is determined by the logic of the subroutine. When the subroutine execution is complete, the states of the outputs are determined. Unused SF outputs are not displayed in the function block.

Note

Outputs from the UDSF block can be left unconnected in the network.

8-27-2. UDSF ENTRY, MONITORING

Certain APL Loader and processor characteristics must be understood before the UDSF subroutines and main application ladder diagram can be loaded and run. In addition to the common items detailed in the APL Programming Manual (Catalog No. NLAM-B806), consider these 5 additional items:

1. UDSF subroutines can be entered into the processor only when the keyswitch is in the STOP:PROGRAM position.

2. The UDSF subroutine **must** be entered into memory **before** any application ladder diagram containing the related UDSF block is programmed. (However, an application ladder diagram could be entered first if it did not contain a related UDSF block. Later editing could add the block—after the subroutine entry.)

3. Once the UDSF subroutine ladder diagram is entered into memory, no editing (adding or removing new elements or register data) of the SFOUT, SFIN, or SF Registers is possible until all the related UDSF blocks in the application ladder diagram have been deleted. (The subroutines may then be edited, and then the related SF blocks can be entered again.)

4. Networks in the UDSF subroutines containing Transitional elements cannot be changed until all related UDSF blocks in the application ladder diagram have been deleted.

5. The APL Loader cannot monitor the status of the application ladder diagrams when it is displaying the networks within a UDSF subroutine. However, the UDSF block in the application diagram can be monitored to determine the status of the subroutine itself.

8-27-3. UDSF PROGRAM EXECUTION

The following sequence of events occurs when the UDSF block contained in the main application ladder diagram is encountered during the program scan. These items assume that the enable input is conducting.

1. The status of each user defined special function input (that is, SFINs) is, in effect, transferred to the UDSF subroutine.

2. The contents of all the special function (SF) registers are, in effect, transferred to the UDSF subroutine.

3. The application ladder diagram scan is interrupted, and the UDSF subroutine is executed.

4. When the UDSF subroutine is completed, the contents of the special function (SF) registers and the status of the SF coils are, in effect, transferred back to the UDSF block in the application ladder diagram at which time execution of the main ladder diagram resumes.

8-27-4. APPLICATIONS

The User Defined Special Function is useful whenever it is desired to reference a subroutine more than once from the ladder diagram. Two such examples are listed as follows:

- Hypotenuse calculations (8-27-4-1)
- Motor starter control application (8-27-4-2)

8-27-4-1. HYPOTENUSE CALCULATIONS

An example of a calculation for the hypotenuse of a right triangle is shown in Figure UDSF-2. Observe the Figure and note the following 4 points:

1. Two sides of the triangle are known. These values are loaded into special function registers SF Register 1 and SF Register 2. Note: The UDSF block uses registers IR0001 and IR0002.

2. Additional UDSF blocks could reference the same subroutine (number 1 in this example) and supply different SF Register types and/or numbers.

3. The result of the subroutine converted to BCD is contained in SF Registers 3 and 4. This data is transferred to output registers OR0001 and OR0002 by the UDSF block for use by the application ladder diagram when the subroutine execution is completed.

4. Completion of the subroutine automatically returns control of the ladder diagram scan to the UDSF block which initiated the subroutine.

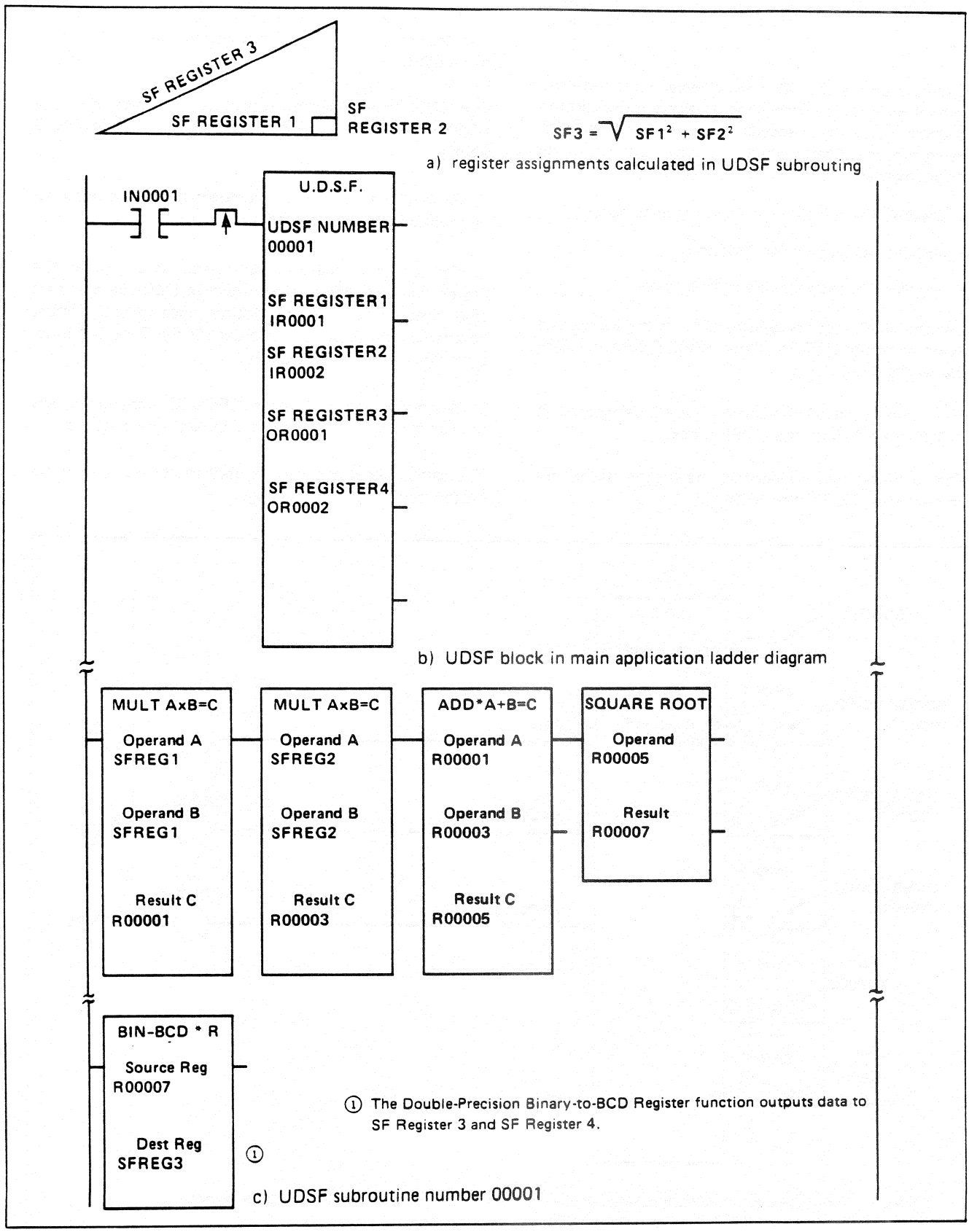


Figure UDSF-2. Right Triangle Calculation Application

8-27-4-2. MOTOR STARTER CONTROL APPLICATION

An example of the User Defined Special Function used to control outputs is noted here. In this application the programmer creates a special function for use with reversing motor starters. When a motor is running in one direction, execution of the subroutine:

- Disables the run forward or run reverse output
- Applies a DC brake for 4 seconds
- Enables the opposite-direction output

One of the UDSF blocks used to execute the subroutine is shown in Figure UDSF-3. Observe the Figure and note the following 3 points:

1. The inputs used by the subroutine are designated as SFIN 1 thru SFIN 5 by the UDSF block.

2. The outputs used to control the motor starter are designated at SFOUT1 thru SFOUT3.

3. The special function registers 2 thru 7 are not used in this application, although they are not prohibited from being used.

The UDSF motor control subroutine is shown in Figure UDSF-4. Observe the Figure and note the following 3 points:

1. Each of the outputs to the motor starter is controlled by separate group of contacts.

2. The brake is enabled for 4 seconds whenever the motor forward 1MF or motor reverse 1MR contacts first stop conducting. This subroutine, once initiated, must be enabled continuously in order for the Time Off function (031) to operate properly.

3. When the brake is enabled, SFOUT3 is energized, and the run forward and run reverse outputs are disabled.

The same subroutine can be used to control 2 or more motor starters, thereby saving:

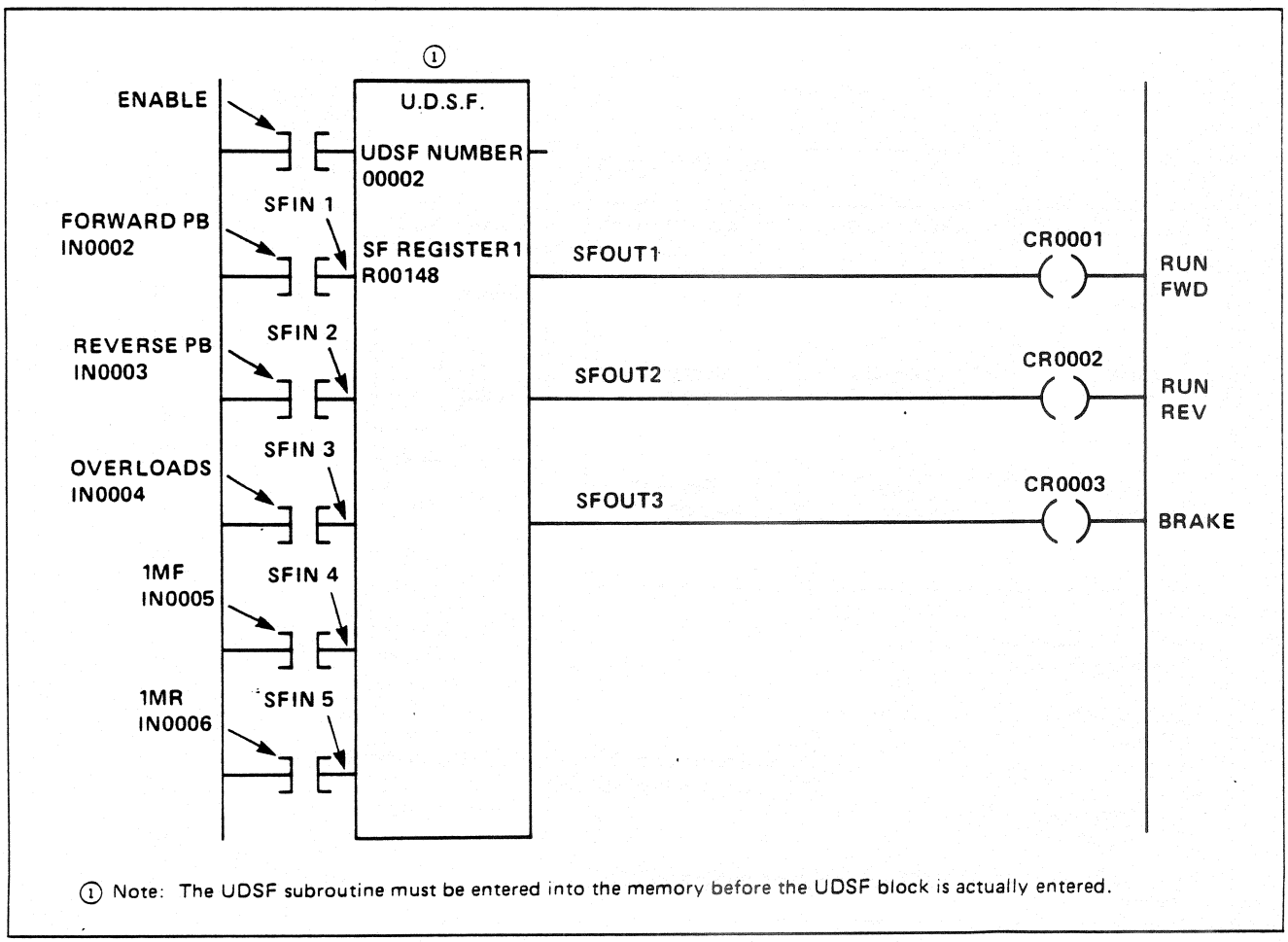


Figure UDSF-3. Main Ladder UDSF Block Reference Subroutine

- Memory requirements
- Program scan time when the subroutines are not enabled
- Programming time since the subroutine is entered once

8-27-5. UDSF PROGRAMMING CONSIDERATIONS

When programming the User Defined Special Function blocks and subroutines, observe the following 6 considerations:

1. Some functions must be executed once per scan in order to function properly. For example, a timer, when enabled, must be executed each scan, or it will not time properly. As a result when a Timer function is located in a UDSF subroutine, the subroutine must be continuously executed during each scan as long as the timer is to be enabled.

2. Any CR coils contained in a UDSF subroutine are not cleared during the normal AC powerup routines. These coils, if required, must be initialized or cleared from networks contained in the main application ladder diagram.

3. When programming the UDSF subroutine, contacts addressing SFOUT coils cannot be directly entered. Use a CR coil in parallel with the SFOUT coil, as shown with coil CR0500 in Figure UDSF-4.

4. If more than 5 inputs, 5 outputs, or 7 registers are needed by the UDSF subroutine, simply transfer the data to a specially designated area (scratchpad) in networks immediately before the UDSF block. For example, the same conduction path used to enable the UDSF block is used to enable a Block Move function (076) to transfer a block of registers for use by the UDSF subroutine.

5. The UDSF subroutines may **not** be nested. They may only be referenced by a UDSF block contained in the main ladder diagram.

6. The following functions are not allowed to be programmed in a UDSF subroutine:

- MCR coil
- Skip coil
- UDSF block (See item 5, above.)

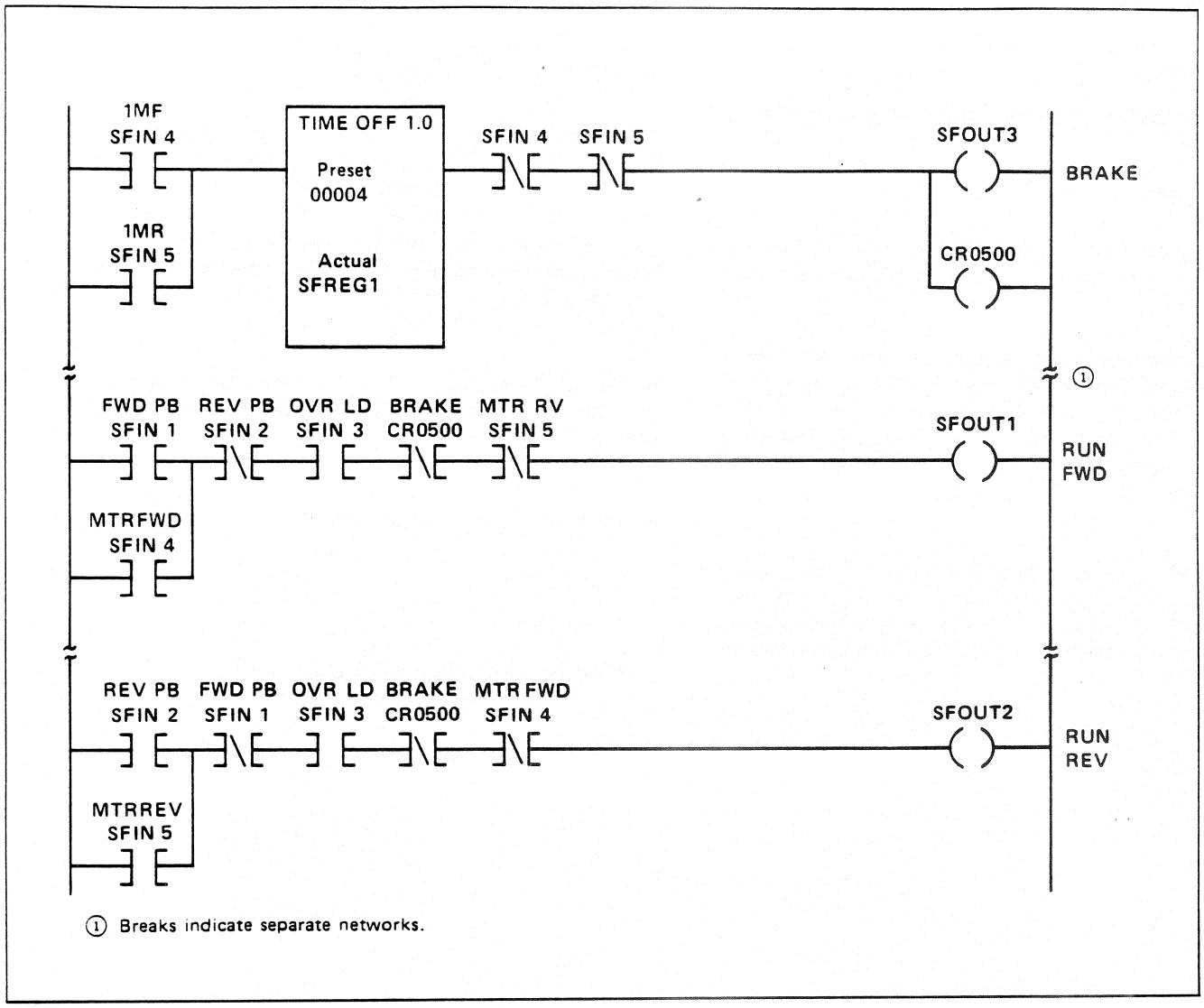


Figure UDSF-4. Motor Starter Subroutine

8-28. LIMIT TEST R (067)

The Limit Test Register programmable function determines if the value of a register is within specified limits. Figure LTR-1 shows a typical circuit containing the Limit Test Register function. Observe the Figure and note the following 3 points:

1. The values of the High and Low Limits are compared to the value of the Test Register whenever the enable circuit is conducting.

2. One of the following outputs conducts, depending on the value of the Test Register being checked as follows:

- Within limits: Test Register value is equal to or between Low and High Limit values
- Overrange: Test Register value is above High Limit value
- Underrange: Test Register value is below Low Limit value

3. All outputs are nonconducting when the enable circuit is nonconducting.

8-28-1. SPECIFICATIONS

The programming specifications for the Limit Test Register function are listed here.

FUNCTION BLOCK

The Limit Test Register function block requires 2 horizontal contact spaces by 4 parallel paths on the ladder diagram. The function block can be positioned anywhere in the contact area of the display screen.

TEST REGISTER

The Test Register contains the value to be compared to the High and Low Limits. The Test Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the register or group number on the display of the function block.

HIGH, LOW LIMITS

The values of the High and Low Limits define the upper and lower limits which are compared to the value in the Test Register. The comparison affects the outputs as listed in the Outputs paragraph. The High and Low Limits are specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Constant value from -32,768 to +32,767

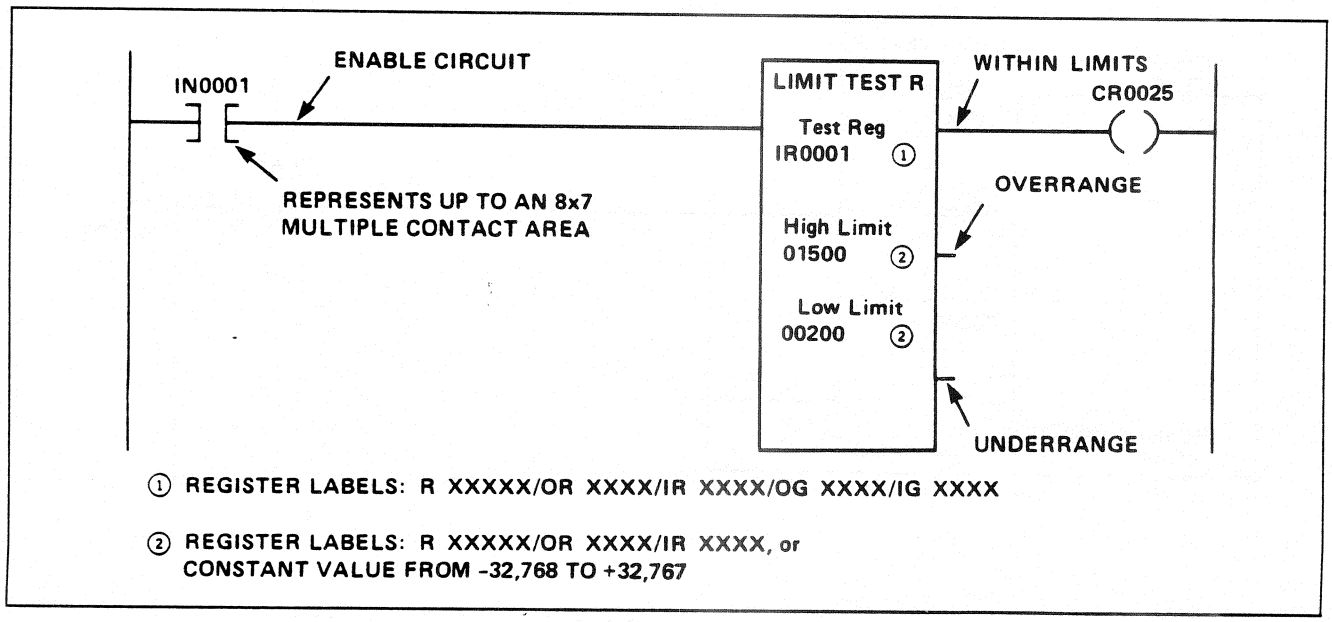


Figure LTR-1. Limit Test Register Circuit (Typical)

The letters R, OR, or IR precede the register number on the display. If a constant value is selected, no letters precede the constant value on the display.

ENABLE CIRCUIT

The enable circuit conducts, allowing the Limit Test Register function to compare the Test Register to the High and Low Limits. The results of the comparison affect the outputs of the function block as described in the Outputs paragraph, listed next. When the enable circuit is nonconducting, no comparisons occur, and all outputs are nonconducting.

The enable circuit can consist of up to 8 horizontal contacts located in up to 7 parallel paths. The rules for programming contacts listed in Section 6 apply to the enable circuit.

OUTPUTS

The outputs of the Limit Test Register function conduct when the enable circuit conducts and the examination of the Test Register is as listed in the following 3 points:

1. The within limits output conducts when the value of the Test Register or group is equal to or within the values of the High and Low Limits. (High Limit = or > Test Register = or > Low Limit.)

2. The overrange output conducts when the value of the Test Register under test is above the value of the High Limit. (Test Register > High Limit.)

3. The underrange output conducts when the value of the Test Register is below the value of the Low Limit. (Test Register < Low Limit.)

Note

Any or all of the outputs from the Limit Test Register function block can be left unconnected in the network.

8-28-2. APPLICATIONS

The Limit Test Register function is used whenever it is desired to monitor the value of a register or group to determine if it is within predetermined limits. One such example is shown in Figure LTR-2 where input IR0001 from a temperature sensor is compared to predetermined limits. Observe the Figure and note the following:

1. The input data received from IR0001 is a binary value where each count of 10 equals 1 degree Fahrenheit.

2. The output CR0015 is energized if the temperature of the Test Register exceeds 250°F or drops below 20°F.

3. Output CR0015 will be held conducting until a reset condition occurs again, allowing the Limit Test Register function to control the output.

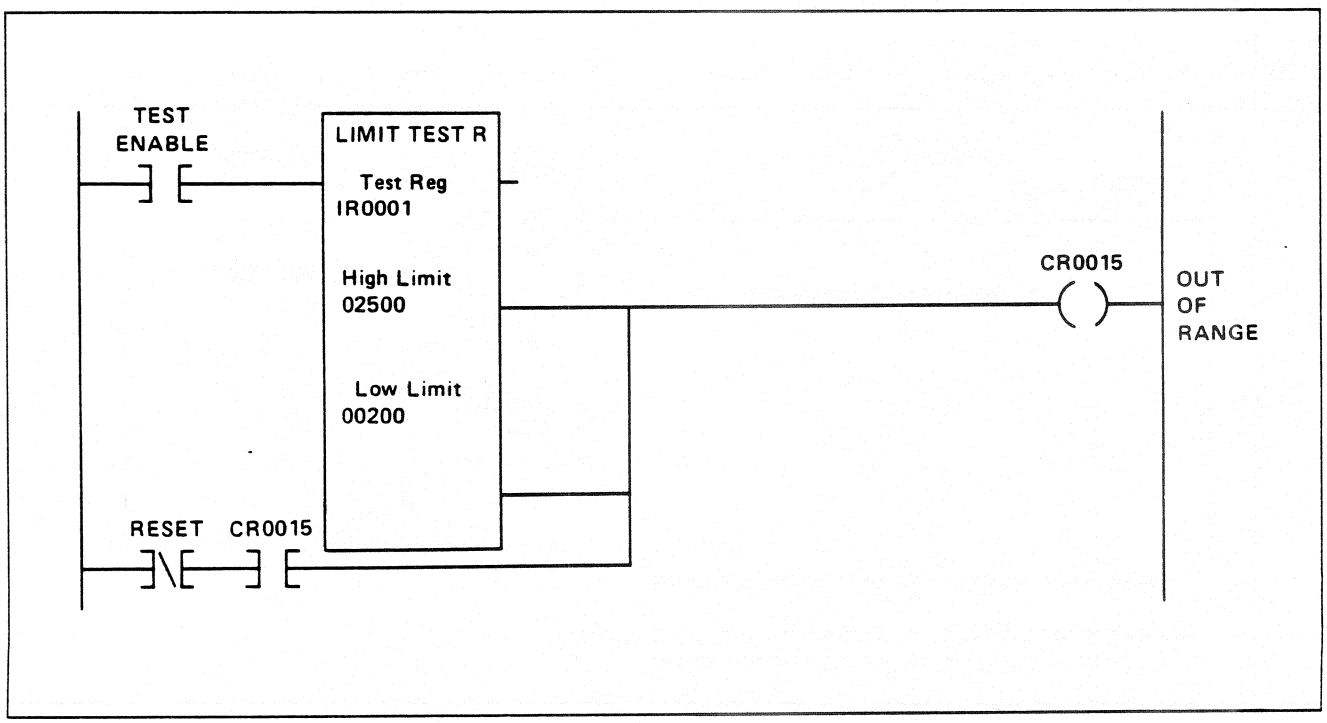


Figure LTR-2. Test Register Application

8-29. COMPARE TABLE-TO-TABLE (091)

The Compare Table-to-Table function performs a comparison between the values of 2 separate and specified registers which are contained in 2 separate and specified Tables. The results of the comparison allow one of the 3 outputs from the function block to conduct as listed in Table CTT-1.

Figure CTT-1 shows a typical circuit containing the Compare Table-to-Table function. Observe the Figure and note the following 3 points:

1. The enable circuit conducts, allowing the comparison to take place.

2. The Pointer Register determines which registers or groups in the Tables are to be compared. If the value of the Pointer Register equals:

- 1, the first register or group in each Table is compared
- 2, the second register or group in each Table is compared
- Etc.

3. The appropriate output conducts whenever the enable circuit conducts, and the values of the selected register or group in Table 1 are equal to, greater than, or less than the values in Table 2. (See Table CTT-1.)

An example of a typical comparison using the valu from the typical circuit of Figure CTT-1 is shown in Figure CTT-2. Observe Figure CTT-2 and note the following 2 points:

1. The value of the Pointer Register is 2. This compares registers R00051 with R00071, the second register in both Tables.

2. Since the value in R00051 is less than the value in R00071, the less than output conducts. (This assumes the enable circuit is conducting.)

8-29-1. SPECIFICATIONS

The programming specifications for the Compare Table-to-Table function are listed here.

FUNCTION BLOCK

The function block for the Compare Table-to-Table function requires 2 horizontal contact spaces by 5

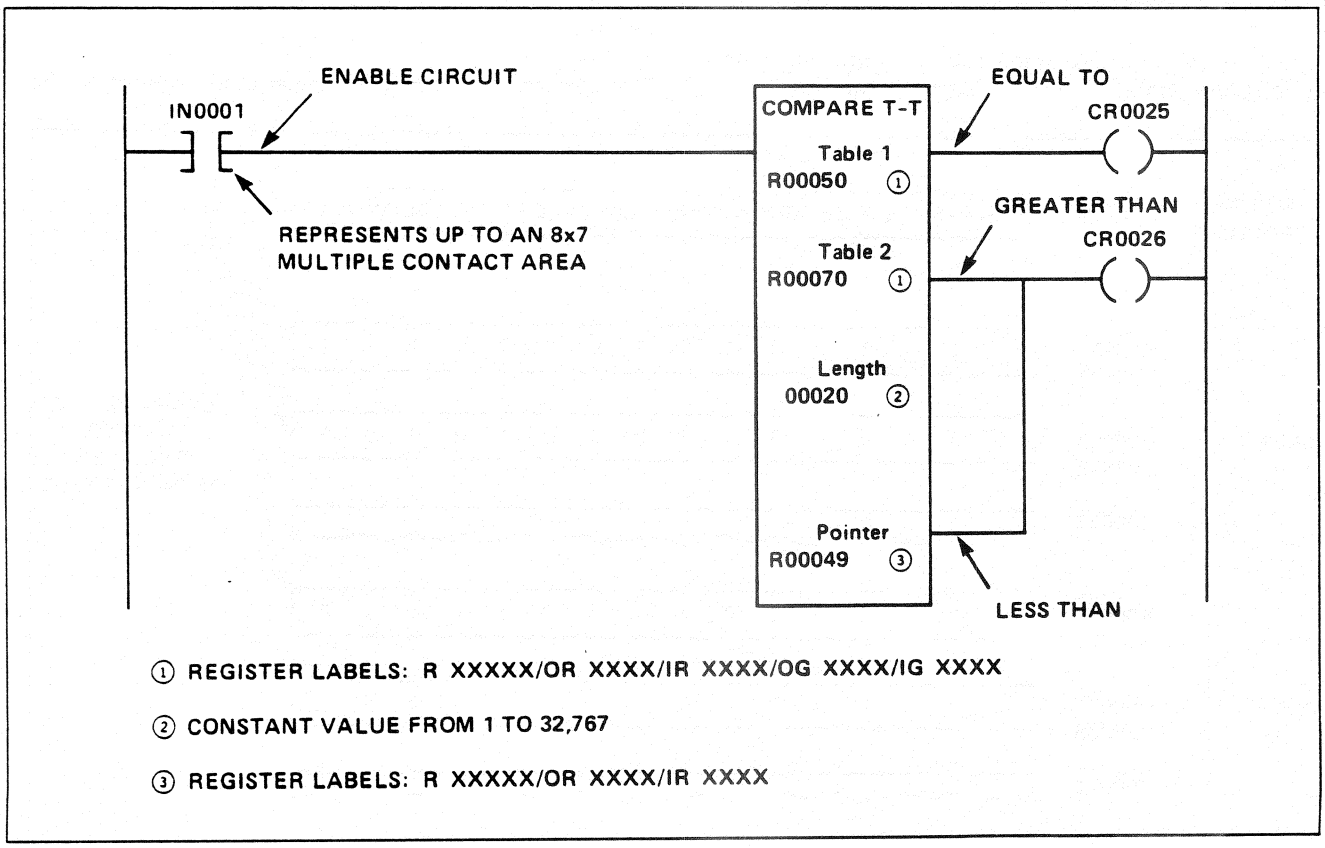


Figure CTT-1. Compare Table-to-Table Circuit (Typical)

TABLE CTT-1. OUTPUT CONDUCTION

Output	Conduction State ①
Equal to	Table 1 Register or group **equals** Table 2 Register or group.
Greater than	Table 1 Register or group is **greater than** Table 2 Register or group.
Less than	Table 1 Register or group is **less than** Table 2 Register or group.

① The registers or groups compared are directed by the value of the Pointer Register bit number. Example: When Pointer Register = 5, the fifth register or group in Tables 1 and 2 are compared.

parallel paths on the ladder diagram. The function block can be positioned anywhere in the contact area of the display screen.

TABLES 1, 2

Table 1 and Table 2 Registers contain the values to be compared as directed by the Pointer Register when the enable circuit conducts. The values compared can range from -32,768 to +32,767. Note: The function automatically assumes any negative number (when bit 16 = 1) is in 2's complement form.

The first register or group in the Tables is displayed in the function block and specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the register or group number on the display.

LENGTH

The Length is a constant value used to specify the

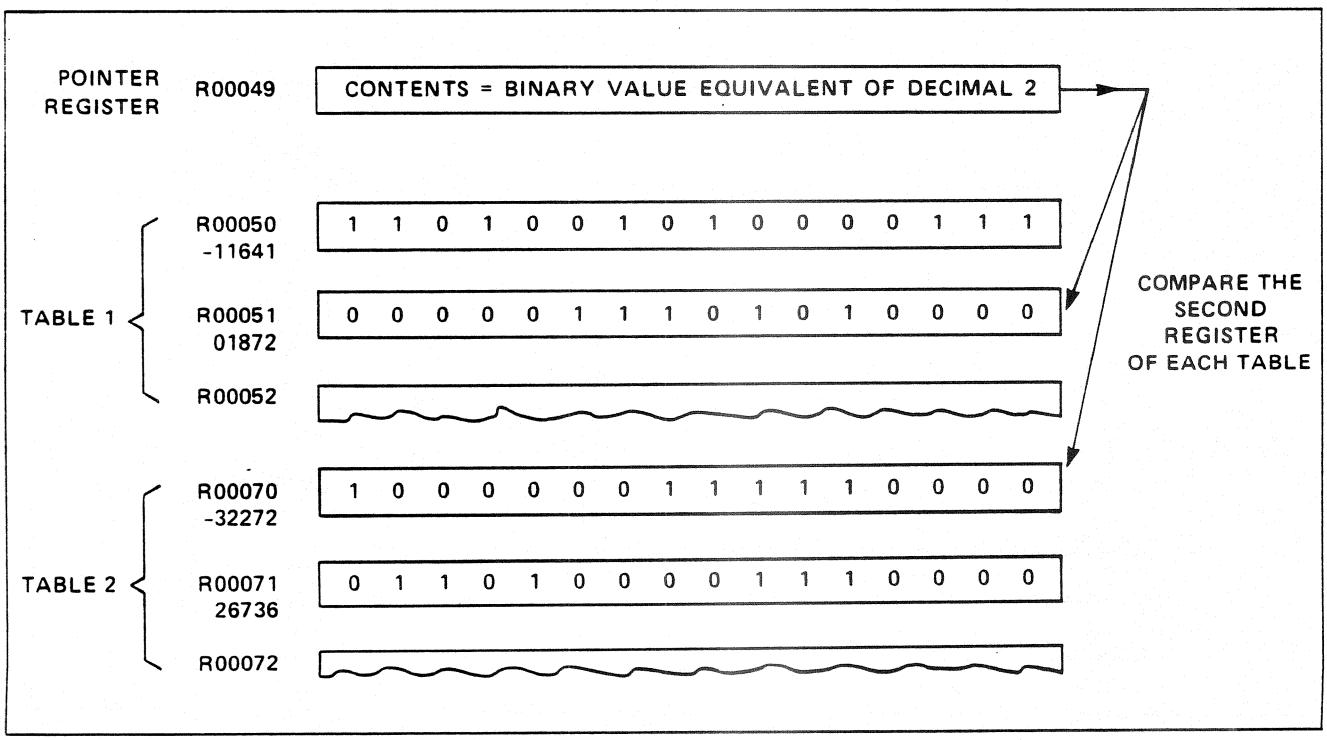


Figure CTT-2. Compare Table-to-Table Example

number of registers in Table 1 and Table 2. The constant value can range from 1 to 32,767.

POINTER

The Pointer Register specifies the registers or groups contained in Tables 1 and 2 whose contents are compared when the enable circuit conducts. The Pointer Register contains a binary value used to direct the comparison where a value of:

- 1 directs the comparison of the first register of both Tables
- 2 directs the comparison of the second register of both Tables
- Etc.

It is the user's responsibility to manipulate the value of the Pointer Register.

The Pointer Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the register number on the display.

ENABLE CIRCUIT

The enable circuit conducts, allowing the Compare Table-to-Table function to compare the registers or groups in Tables 1 and 2 as directed by the Pointer Register. When a comparison occurs, the outputs conduct as described in Table CTT-1. When the enable circuit is nonconducting, no comparisons occur and the outputs are nonconducting.

The enable circuit can consist of up to 8 horizontal contacts located in up to 7 parallel paths. The rules for programming contacts listed in Section 6 apply to the enable circuit.

OUTPUTS

The outputs of the Compare Table-to-Table function conduct when the enable circuit conducts, and the comparisons of the register or group in the Test Tables are as listed in the following 3 points:

- The equal to output conducts when the values of the registers or groups compared are equal.
- The greater than output conducts when the value of the register or group in Table 1 is greater than the value of the register or group in Table 2.
- The less than output conducts when the value of the register or group in Table 1 is less than the value of the register or group in Table 2.

Note

The outputs from the Compare Table-to-Table function block can be left unconnected in the network.

8-29-2. APPLICATIONS

The Compare Table-to-Table function is used whenever it is desirable to compare the values in 2 tables. One such example is shown in the ladder diagram of Figure CTT-3 where the values derived from 8 pressure measurements are compared to predetermined values. In this application each pressure must be equal to, or greater than, a predetermined value for the test to continue. Observe Figure CTT-3 and note the following 5 points:

1. The actual pressure readings are obtained from input registers IR0001 thru IR0008. In this example it is assumed that the values obtained are already in binary form.

2. The Pointer Register directs a different pressure measurement obtained from IR0001 thru IR0008 to be compared during each scan.

3. If any pressure monitored is not equal to, or greater than, its corresponding value contained in Table 2 (registers R00050 thru R00057), bit 1 of holding register R00048 is de-energized. The Pointer Register stops incrementing since the enable circuit of the Up Counter function (038) contains a BP/R contact examining bit 1 of R00048.

4. When a pressure is found to be deficient, a restart test contact must be closed to re-initiate the testing.

5. The Move Register-to-Register function (050) transfers the data from the Pointer Register to output register OR0001 when a pressure is **not** equal to or greater than the specified value. The output register, in turn, is connected to an LED display associated with an operator panel. This display supplies a visual indication of the pressure (number 1 thru 8) which is deficient.

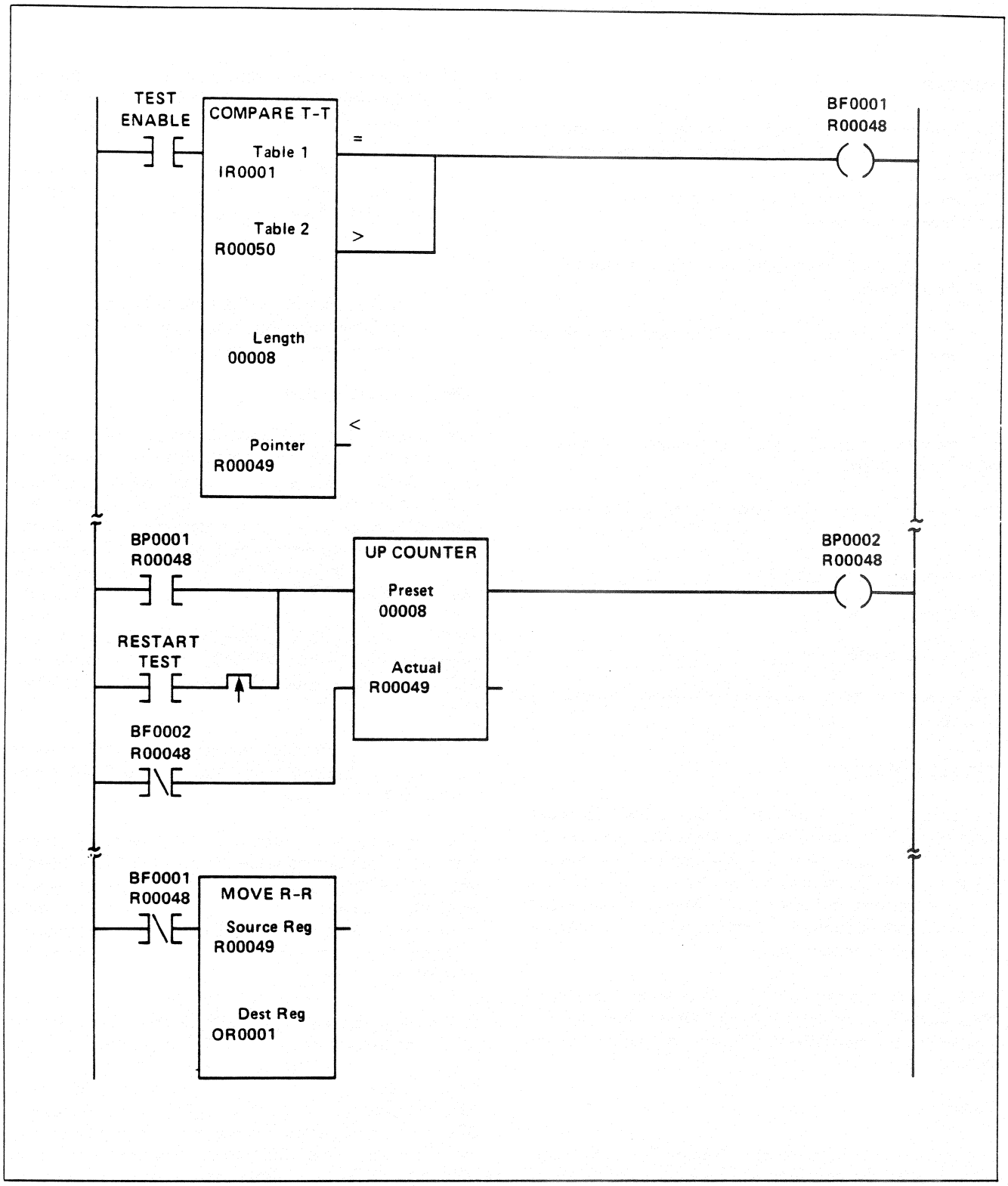


Figure CTT-3. Compare Table-to-Table Application

8-30. COMPARE TABLE-TO-REGISTER (092)

The Compare Table-to-Register programmable function performs a comparison between the value contained in the Register and the value contained in a single register of a Table. The results of the comparison allow one of the following 3 outputs to conduct:

- Equal to
- Greater than
- Less than

Figure CRT-1 shows a typical circuit containing the Compare Table-to-Register function. Observe the Figure and note the following 3 points:

1. The enable circuit conducts, allowing the comparison to take place.

2. The Pointer Register determines which register or group contained in the Table is to be compared to the Register specified in the function block.

3. The appropriate output conducts whenever the enable circuit conducts and the value of the selected register or group in Table 1 is equal to, greater than, or less than the value in the Register. (See Table CRT-1.)

An example of the operation of the Compare Table-to-Register function is shown in Figure CRT-2 using the registers shown in Figure CRT-1. In this example the registers R00029 and R00032 are compared. Since the value in the Table (R00032) is greater than the Register (R00029) the greater than output conducts.

8-30-1. SPECIFICATIONS

The programming specifications for the Compare Table-to-Register function are listed here.

FUNCTION BLOCK

The Compare Table-to-Register function block requires 2 horizontal contact spaces by 5 parallel paths on the ladder diagram. The function block can be positioned anywhere in the contact area of the display screen.

TABLE

The Table contains the registers or groups which are compared to the value of the Register. During each scan when the enable circuit is conducting, the Register is compared to another single register or group contained

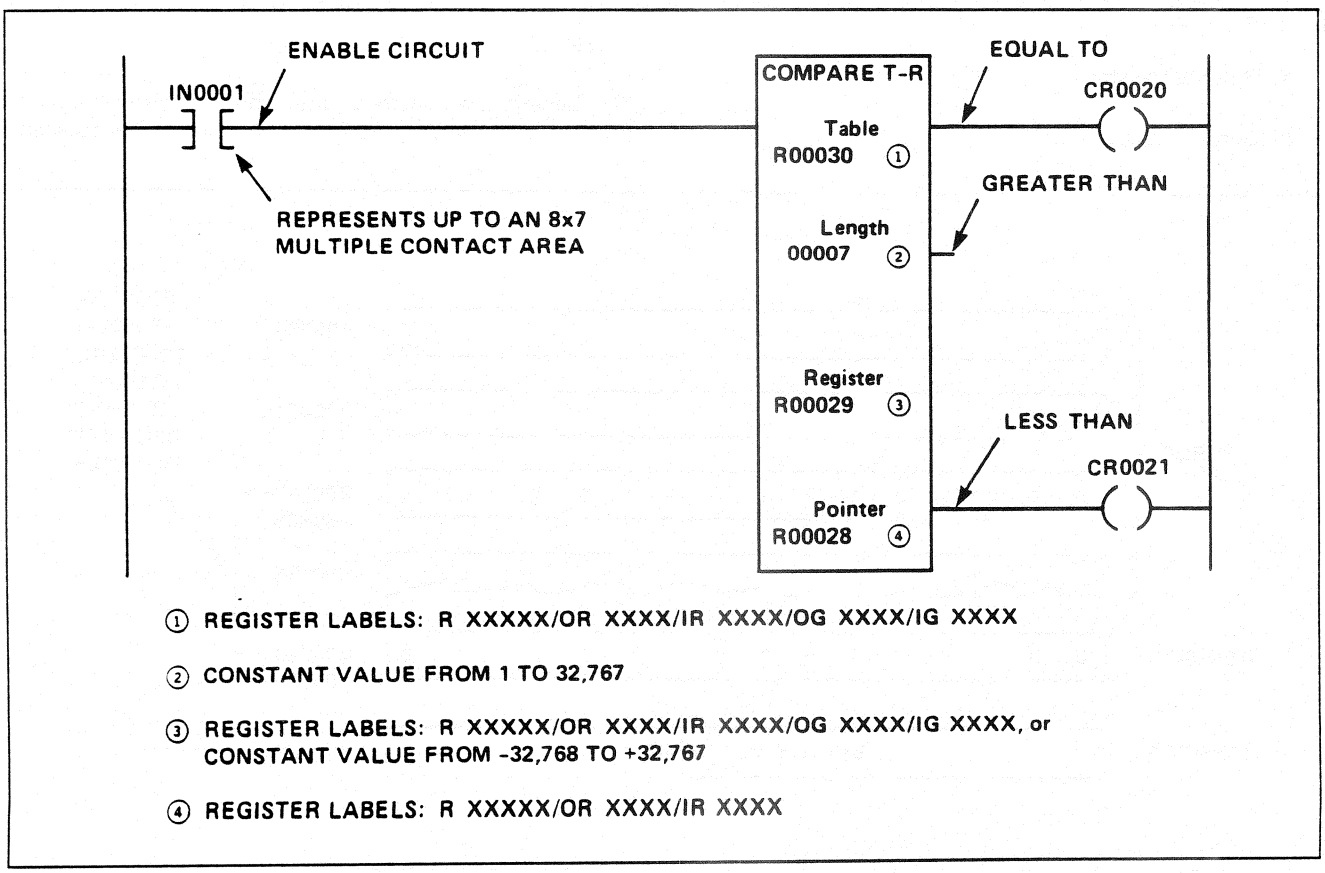


Figure CRT-1. Compare Table-to-Register Function Circuit (Typical)

TABLE CRT-1. OUTPUT CONDUCTION

Output	Conduction State ①
Equal to	Conducts when the value of the register or group in the Table is equal to the value in the Register.
Greater than	Conducts when the value of the register or group in the Table is greater than the value in the Register.
Less than	Conducts when the value of the register or group in the Table is less than the value in the Register.

① The Pointer Register specifies the register or group in the Table which is to be compared. For example, when the Pointer Register contains a binary number equal to 5 decimal, the fifth register in the Table is compared to the data contained in the Register.

in the Table as directed by the Pointer Register. (See Pointer Register described below.) The values compared can range from -32,768 to +32,767. The function automatically assumes any negative number (when bit 16 = 1) is in 2's complement form.

The first register or group in the Table is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the register or group number on the display.

LENGTH

The Length is a constant value which specifies the number of registers or groups in the Table. The constant

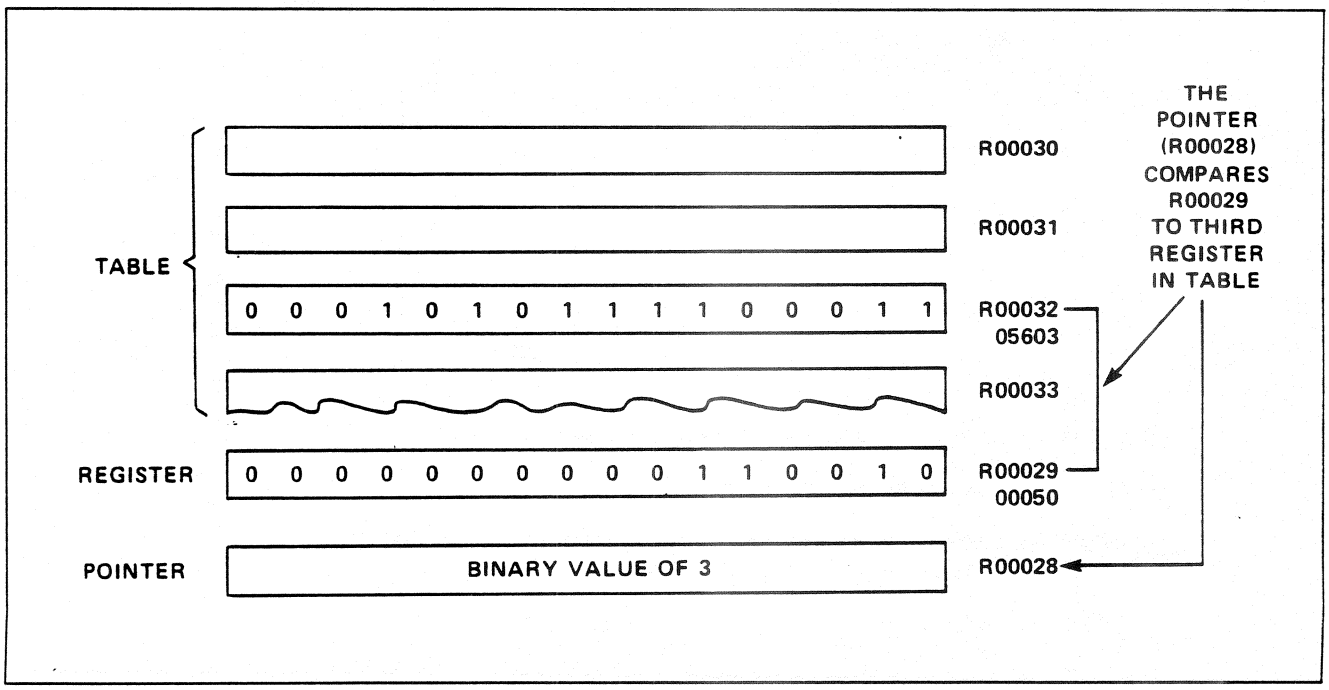


Figure CRT-2. Example Comparison

value ranges from 1 to 32,767.

REGISTER

The Register contains the value which is compared to a register or group contained in the Table as directed by the Pointer Register. See the Pointer Register description below. The register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)
- Constant from 1 to 32,767

The letters R, OR, IR, OG, or IG precede the register or group number on the display. If a constant value is selected, no letter precedes the value on the display.

The Pointer Register specifies the register or group in the Table to be compared to the Register when the enable circuit conducts. The Pointer Register contains a binary value used to direct the comparison where a value of:

- 1 directs the comparison of the first register or group of the Table to the Register
- 2 directs the comparison of the second register or group of the Table to the Register
- Etc.

If the Pointer Register is 0, a negative number, or a number larger than the Table Length, no comparison occurs, and all outputs will be nonconducting.

It is the user's responsibility to manipulate the value of the Pointer Register.

The Pointer Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the register number on the display.

ENABLE CIRCUIT

The enable circuit conducts, allowing the Compare Table-to-Register function to compare the value of the Register to the value of the register or group in the Table as specified by the Pointer Register. When a comparison occurs, the outputs conduct as described in Table CRT-1. When the enable circuit is nonconducting, no comparisons occur, and the outputs are nonconducting.

The enable circuit can consist of up to 8 horizontal contacts located in up to 7 parallel paths. The rules for programming contacts listed in Section 6 apply to the enable circuit.

OUTPUTS

The equal to, greater than, and less than outputs conduct as listed in Table CRT-1.

Note

The outputs from the Compare Register-to-Table function block can be left unconnected in the network.

8-30-2. APPLICATIONS

The Compare Table-to-Register function can be used whenever it is desired to compare values contained in a register with values contained in a separate Table. One such example is in a motor test stand application where an oil pressure measurement is taken at 8 different times in a test cycle. The pressures must be equal to or greater than a predetermined level for each of the 8 measurements. Figure CRT-3 shows the circuit portion used to provide the pressure monitoring. Observe the Figure and note the following 6 points:

1. The Table, consisting of registers R00060 thru R00067, contains the values used for comparison during each step of the test.

2. The pressure measurement data is received at input register IR0001. The pressure is represented by a binary value which is the equivalent of 100 counts equaling 1 psi.

3. The Pointer Register is controlled by the Up Counter function (038). Each time a step in the test is completed, the Actual Value of the counter is incremented by 1.

4. If the pressure during any test is low, CR0024 is energized. This output connects to a panel indicator and stops the testing in a portion of the ladder diagram not shown here.

5. The output of the Compare T-R function enables CR0024 when the pressure in the Table is greater than the test pressure. This indicates a low test pressure.

6. The test complete contacts reset the Up Counter function when the 8 steps of the test have been completed.

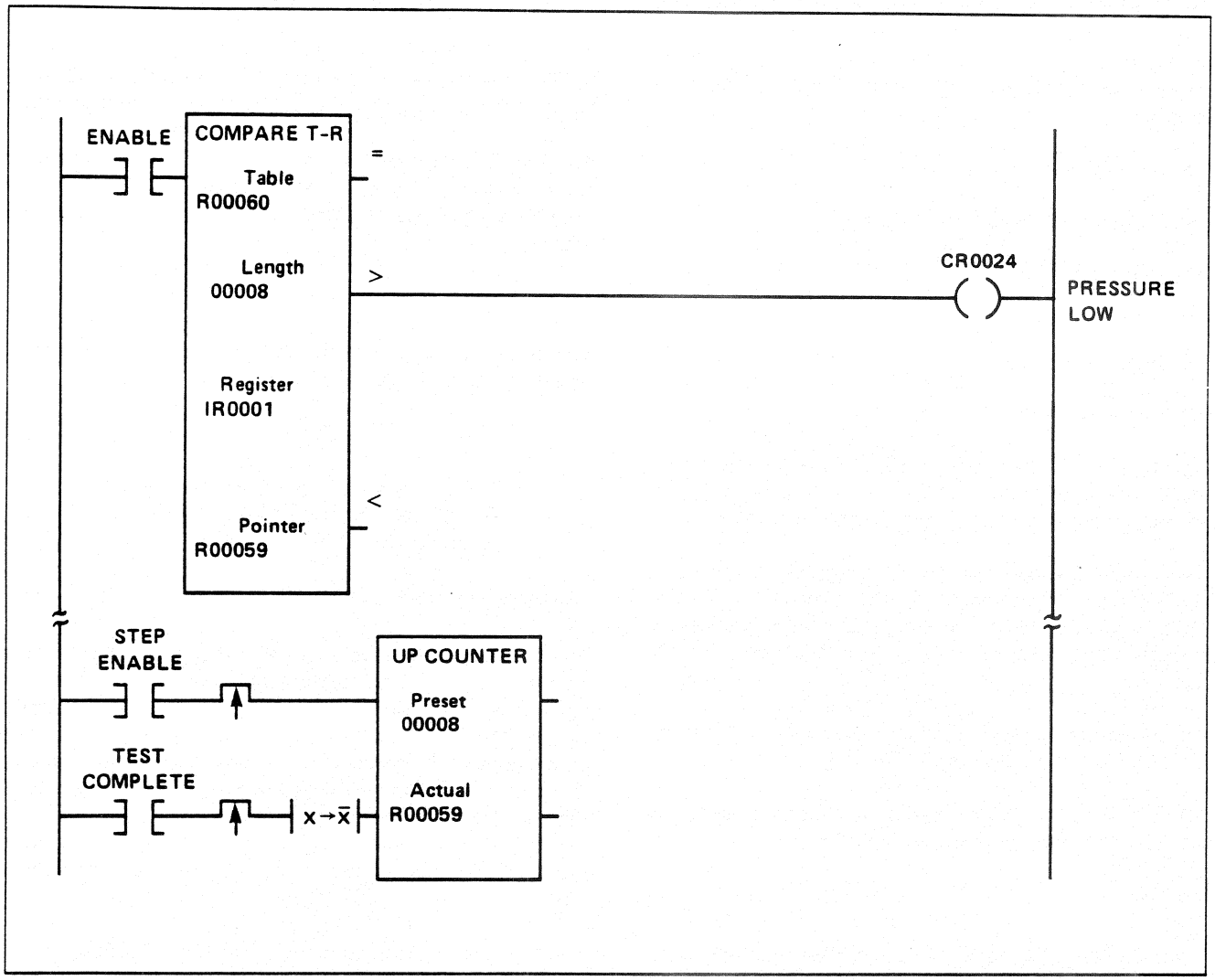


Figure CRT-3. Compare Table-to-Register Application

8-31. LIMIT TEST TABLE (113)

The Limit Test Table programmable function determines if the value of a register contained in a Table is within specified limits. Figure LTT-1 shows a typical circuit containing the Limit Test Table function. Observe the Figure and note the following 4 points:

1. The values in the High and Low Limits are compared with the value in the Test Table as directed by the Pointer Register whenever the enable circuit conducts.

2. It is the user's responsibility to manipulate the value of the Pointer Register.

3. One of the following three outputs from the function block conducts, depending on the status of the register being checked:

- Within limits
- Overrange
- Underrange

4. All outputs are nonconducting when the enable circuit is nonconducting.

An example showing the effects of a Pointer Register value of 2 is shown in Figure LTT-2. In this example the first register of the Table is R00050.

8-31-1. SPECIFICATIONS

The programming specifications for the Limit Test Table function are listed here.

FUNCTION BLOCK

The Limit Test Table function block requires 2 horizontal contact spaces by 5 parallel paths on the ladder diagram. The function block can be positioned anywhere in the contact area of the display screen.

TEST TABLE

The Test Table register or group lists the first register or group contained in the Test Table. See the Outputs

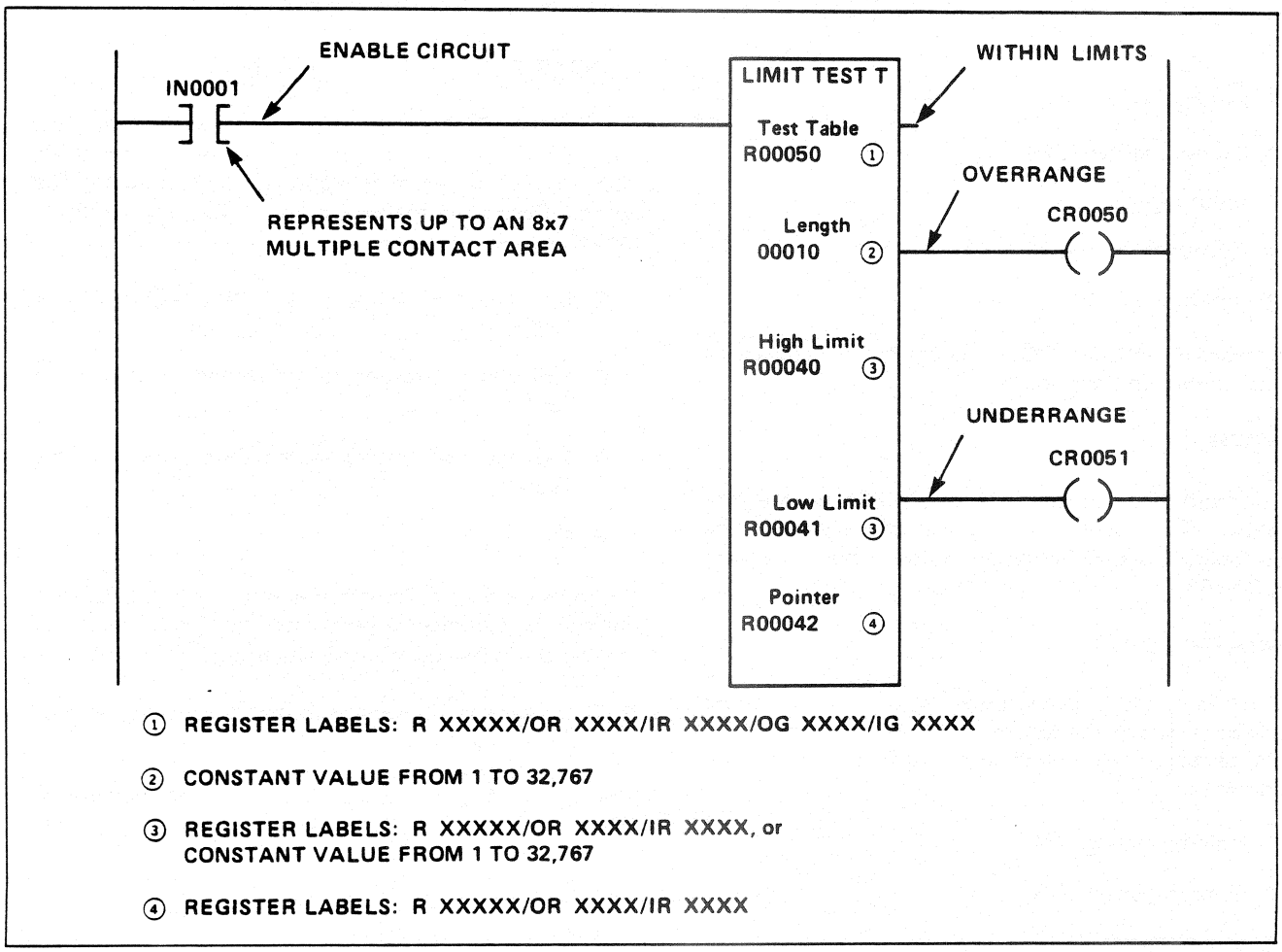


Figure LTT-1. Limit Test Table Circuit (Typical)

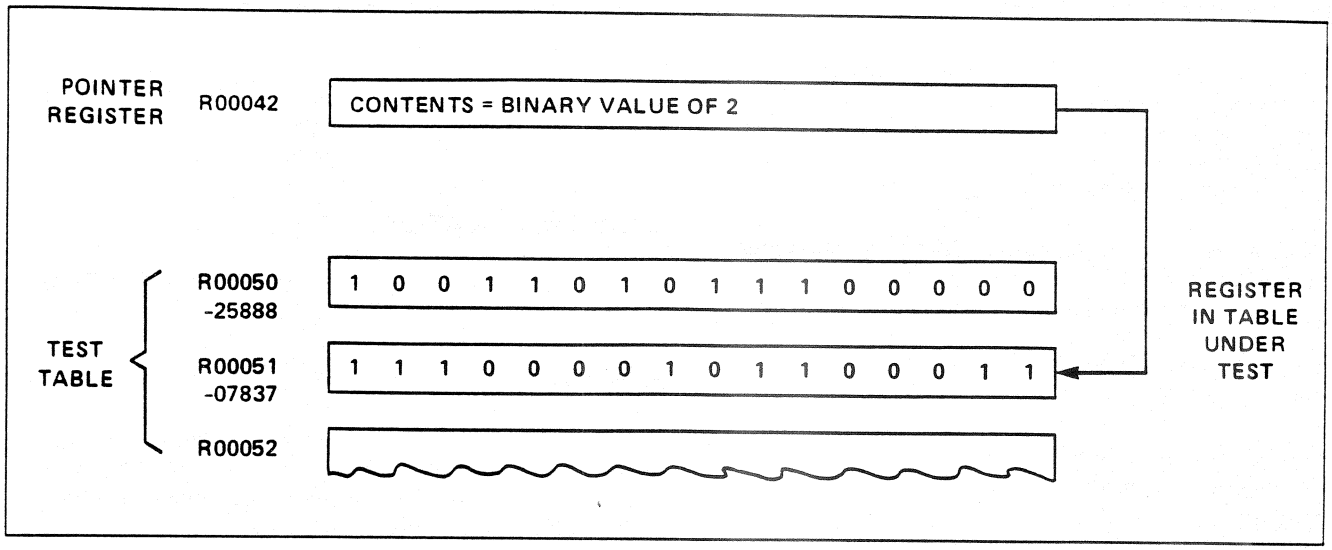


Figure LTT-2. Limit Test Table Example

description for a listing of how the values of the Test Table registers or groups affect the outputs. The Test Table register or group is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Output group (OG)
- Input group (IG)

The letters R, OR, IR, OG, or IG precede the register or group number on the display.

LENGTH

The Length is a constant value specified by the programmer which defines the number of registers in the Test Table. It is specified as any value in the range from 1 to 32,767.

HIGH/LOW LIMITS

The values of the High and Low Limits define the upper and lower limits of the register in the Test Table which is being tested. These limits are specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)
- Constant value from -32,768 to +32,767

The letters R, OR, or IR precede the register number on the display. If a constant value is selected, no letters precede the constant value on the display.

POINTER REGISTER

The Pointer Register specifies the register in the Test Table whose contents are to be tested during the ladder diagram scan when the enable circuit conducts. The Pointer Register contains a binary value used to direct the testing, where a value of:

- 1 directs the testing to the first register in the Table
- 2 directs the testing to the second register in the Table
- 3 directs the testing to the third register in the Table
- Etc.

If the value of the Pointer Register is zero, a negative number, or a number greater than the Table Length, no comparison is performed, and all outputs remain off.

It is the user's responsibility to manipulate the value of the Pointer Register.

The Pointer Register is specified by the programmer as a:

- Holding register (R)
- Output register (OR)
- Input register (IR)

The letters R, OR, or IR precede the register number on the display.

ENABLE CIRCUIT

The enable circuit conducts, allowing the Limit Test Table function to compare the register in the Test Table to the High and Low Limits as directed by the Pointer Register. When the enable circuit is nonconducting, no testing occurs. The enable circuit can consist of up to 8 horizontal contacts located in up to 7 parallel paths. The rules for programming contacts listed in Section 6 apply to the enable circuit.

OUTPUTS

One of the outputs of the Limit Test Table function conducts when the enable circuit conducts, and the examination of the register or group in the Test Table is as listed in Table LTT-1.

8-31-2. APPLICATIONS

The Limit Test Table function is used whenever it is desired to monitor a table of values to determine if they are within predetermined limits. One such example is shown in Figure LTT-3 where temperature readings from 5 sources, IR0001 thru IR0005, are monitored to determine if they are within the range of 60° to 110°F. Observe the Figure and note the following points:

1. The inputs from IR0001 thru IR0004 represent temperatures in degrees Fahrenheit. (Here one binary count equals one degree.)

2. The High and Low Limits are constant values of 110 and 60, respectively.

3. During normal operation the Up Counter function is incremented each time the ladder diagram scan takes place. The Actual Register of the Up Counter function (038) is used as the Pointer Register of the Limit Test Table function. After the actual value of the Up Counter function equals 5, the counter resets itself.

4. If one of the values in the Test Table is not within range, the temperature out-of-range output CR0025 energizes. Contacts from CR0025 disable further testing of the Test Table and incrementing of the Up Counter function until the temperature is brought within limits and the reset contact is closed.

TABLE LTT-1. OUTPUT CONDUCTION

Output	Conduction State
Within Limits	Test Table Register or group value is equal to or within the High Limit and Low Limit values.
Overrange	Test Table Register or group value is above the value of the High Limit.
Underrange	Test Table Register or group value is below the value of the Low Limit.

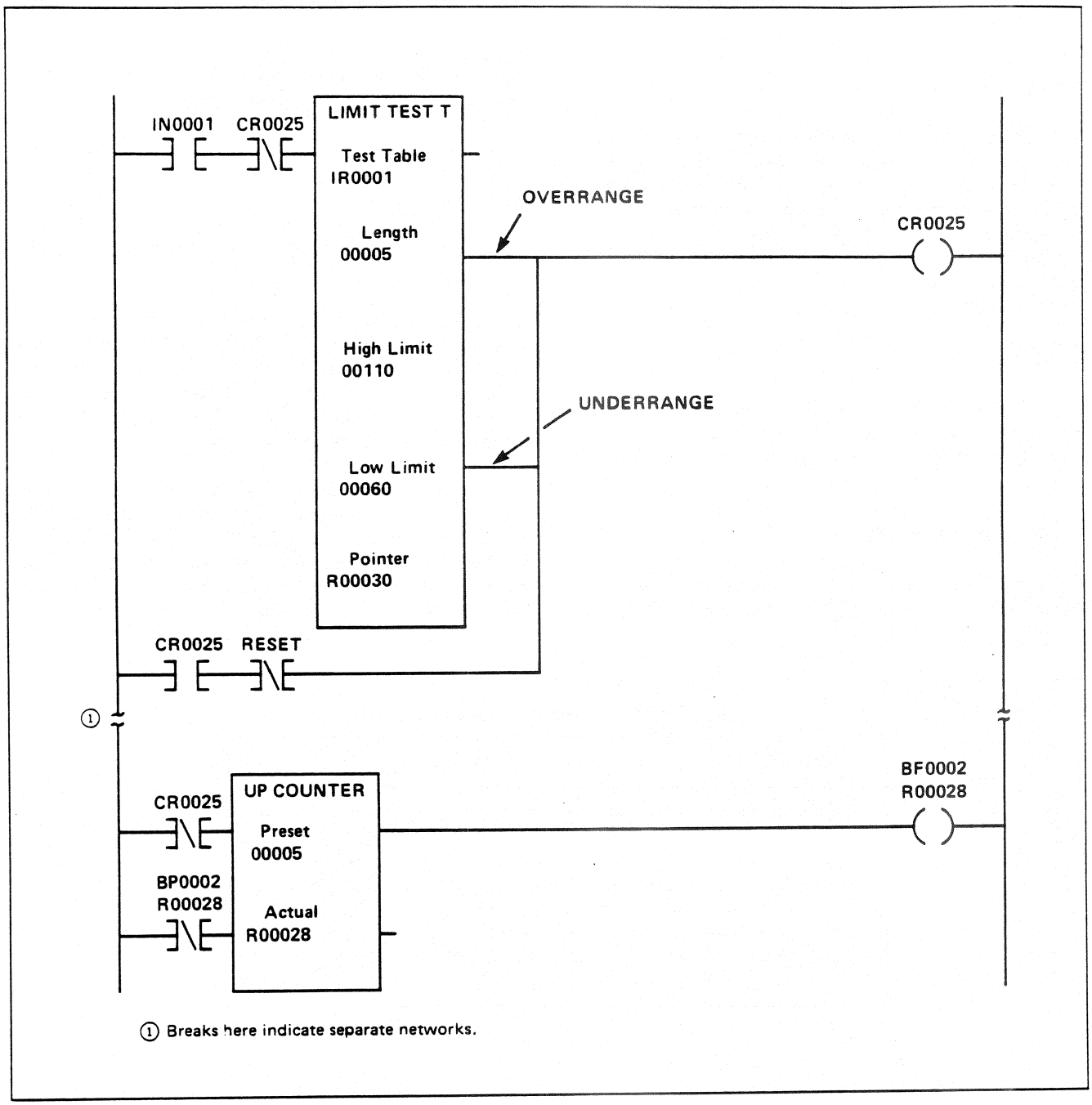


Figure LTT-3. Limit Test Table Application

Section 10

Programming Considerations

10-1. INTRODUCTION

The information contained in this Section extends the discussion of programmable functions into areas of special use and programming techniques. These items are frequently overlooked, but, if observed, they can result in increased system efficiency.

The following subjects are discussed:

- Coil utilization (10-2)
- Network reconstruction (10-3)
- Network considerations (10-4)
- Program execution time and memory requirements (10-5)
- Throughput considerations (10-6)

Note

This Section should be read and understood **before** attempting to program the HPPC-1500/-1700.

10-2. COIL UTILIZATION

Coils can be programmed in a variety of ways. It is important to note that an effective program is designed to increase overall efficiency and, at the same time, to decrease memory consumption. The programmer determines the manner in which techniques are applied in order to optimize both time and storage space. With practice, the programmer becomes adept at structuring programs for effectiveness. Some of the techniques used to achieve this objective are discussed in this Section. They are:

- Logic coils (10-2-1)
- Dummy coils (10-2-2)
- Program order (10-2-3)
- Multiple programs (10-2-4)

10-2-1. LOGIC COILS

In addition to other obvious functions, effective use of logic coils can help reduce program length. When an identical group of contacts is examined repeatedly, a single logic coil can be set once with the state of the entire contact group. Subsequently, only the single logic coil is examined in place of that entire contact group. A specific example of wasteful programming involves the same 4 series contacts being examined in 20 networks. This would require 80 contacts. However, by examining the 4 contacts once, and then storing the result of the contacts' states in a single logic coil, the total elements required would be reduced to 25. (This total is composed of the initial 4 contacts, 1 coil, and 20 contacts, thereby saving the programming of 55 contacts.)

In this and similar ways logic coils can be used to reduce the number of instructions contained in the ladder diagram.

10-2-2. DUMMY COILS

A dummy contact is one which will **always** remain either:

- Nonconducting
- Conducting

The use of a dummy contact requires the examination of a "dummy coil" which remains in a constant state, as explained in Paragraph 6-7-1-2 and Figure 6-18. (A dummy coil is one that is never energized and whose contacts never change state.)

It may appear, at first, that the easiest method of developing a dummy coil is simply not to program it, although an "unprogrammed coil" can be examined by the program.

However, in practice, there are two conditions that make this impractical. The first is that the dummy coil, for some reason, could be forced on. With no logic to cause it to later change state, it remains on even when the force is cleared. The second drawback is that if someone else uses the program, an undocumented dummy coil could be unknowingly programmed to serve another function, thereby creating confusion and potential danger.

The best method for programming a dummy coil, which is never energized, is illustrated in Figure 10-1(a). As shown, forcing coil CR0099 causes its contacts to change state. When the force is cleared, the coil automatically returns to the off state. In addition to

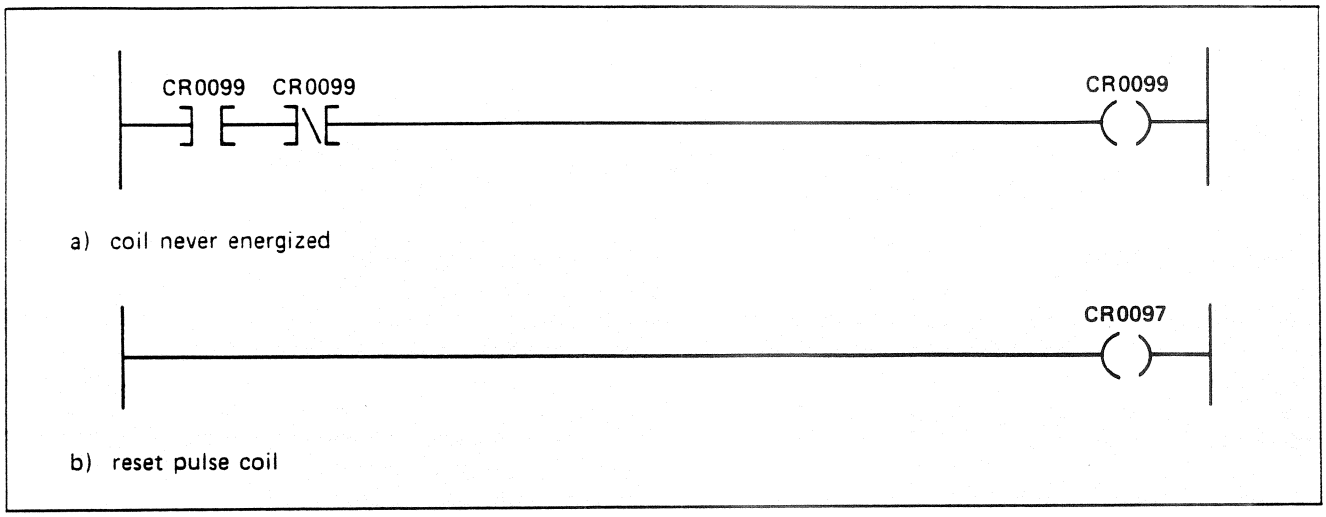


Figure 10-1. Dummy Coil Circuit

consistently functioning properly, another benefit of using this circuit is that the coil is documented on any printout or drawing, preventing erroneous future use.

A pulse used to generate the resetting of programmable functions such as timers, counters, or the Zero Table (051) function can be generated as shown in Figure 10-1(b). Locate the rung like the one shown in the Figure at the end of the main ladder program. The coil will be de-energized until the first scan is completed after AC power is first applied to the processor or the keyswitch is turned to any RUN position from the STOP: PROGRAM position. Contacts from the coil (CR0097 in this example) provide a nonconducting path during the first scan, resetting the desired functions. During subsequent scans, the same contacts conduct.

10-2-3. PROGRAM ORDER

In general the order of the program, including where a coil is placed in the order, is generally not critical. However there may be instances where coil placement demands careful consideration. It is important to avoid the situations shown in the following examples. Otherwise many hours of problem analysis could result.

10-2-3-1. PROGRAM ORDER EXAMPLE

The circuit shown in Figure 10-2 illustrates a situation

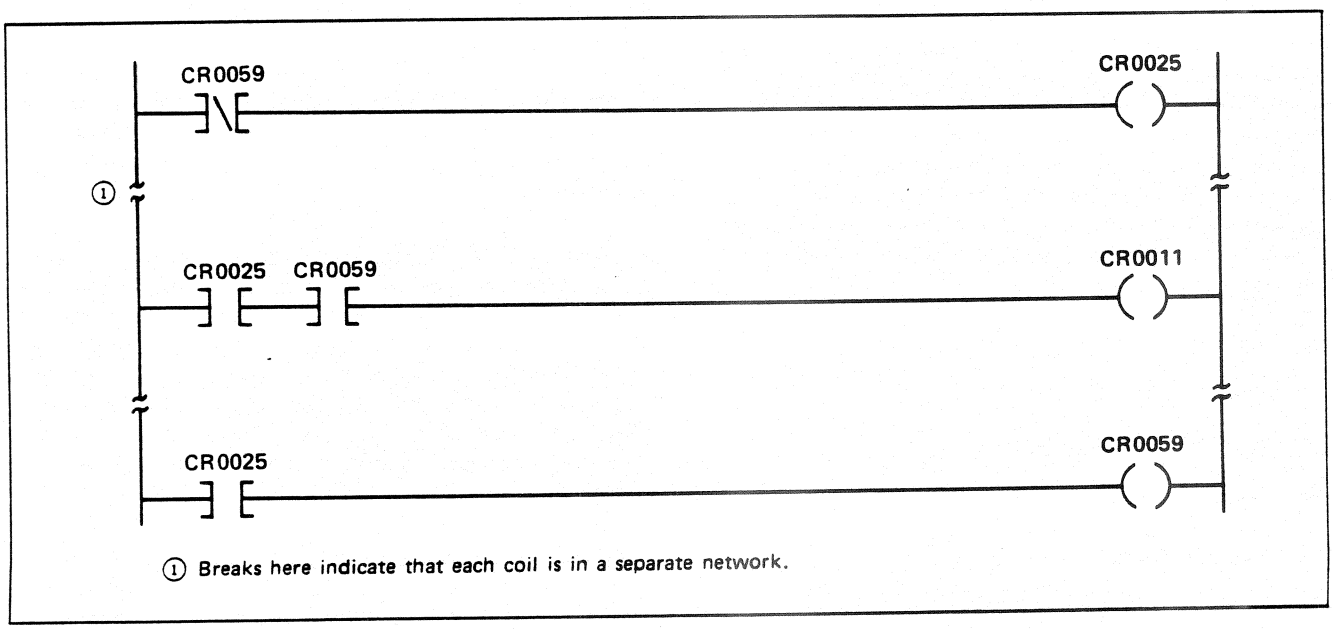


Figure 10-2. Improper Programming Order

where coil placement can be important. Since coil CR0059 is currently off, coil CR0025 energizes through the NC contacts of CR0059. Coil CR0011 does not energize, because the NO contact of CR0059 is open. Since coil CR0025 is now energized, coil CR0059 is energized. Scanning the circuit a second time finds the NC contacts of CR0059 now open, thus turning coil CR0025 off. CR0011 does not energize, because this time the NO contacts of CR0025 are open. Coil CR0059 turns off. As a result, CR0025 and CR0059 alternate on and off, and CR0011 is never on. The situation is demonstrated in Table 10-1.

TABLE 10-1. COIL STATES

Coil	Scan 1	Scan 2	Scan 3	Scan 4
CR0025	on	off	on	off
CR0011	off	off	off	off
CR0059	on	off	on	off

To change this situation in order to produce the desired result, all that is required is a change in the program order. Figure 10-3 illustrates what is required to correct the programming discussed above; the results of this program order are shown in Table 10-2.

10-2-4. MULTIPLE PROGRAMS

The HPPC-1500/-1700 offers such a large memory capacity that it is possible to load in more than one ladder diagram program in order to control more than one application or process. (The exact number depends upon the available hardware, required memory, and available memory.) It may also control a single machine or process in distinctly different ways.

TABLE 10-2. COIL STATES AFTER REORDERING

Coil	Scan 1	Scan 2	Scan 3	Scan 4
CR0025	on	off	on	off
CR0011	on	off	on	off
CR0059	on	off	on	off

For example, each control program can be manually or automatically initiated at separate times. (See Figure 10-4.) The entirely different ladder diagrams can control the same inputs and outputs, but the operational sequences are distinct.

As shown in Figure 10-4, the 3 programs are controlled by inputs IN0001, IN0002, and IN0003. Only one Skip coil is enabled at any given time.

The MCR Coil (005) and Skip Coil (004) functions can also be used to divide a lengthy program into portions where only one portion is executed during each program scan. By installing an Up Counter (038) function which increments during each program scan, the various bits of the Actual Register can be examined in place of the inputs IN0001, IN0002, and IN0003 shown in Figure 10-4. Dividing a program in this way reduces the overall scan time since the various elements in an inactive area of the program are only accessed, not executed.

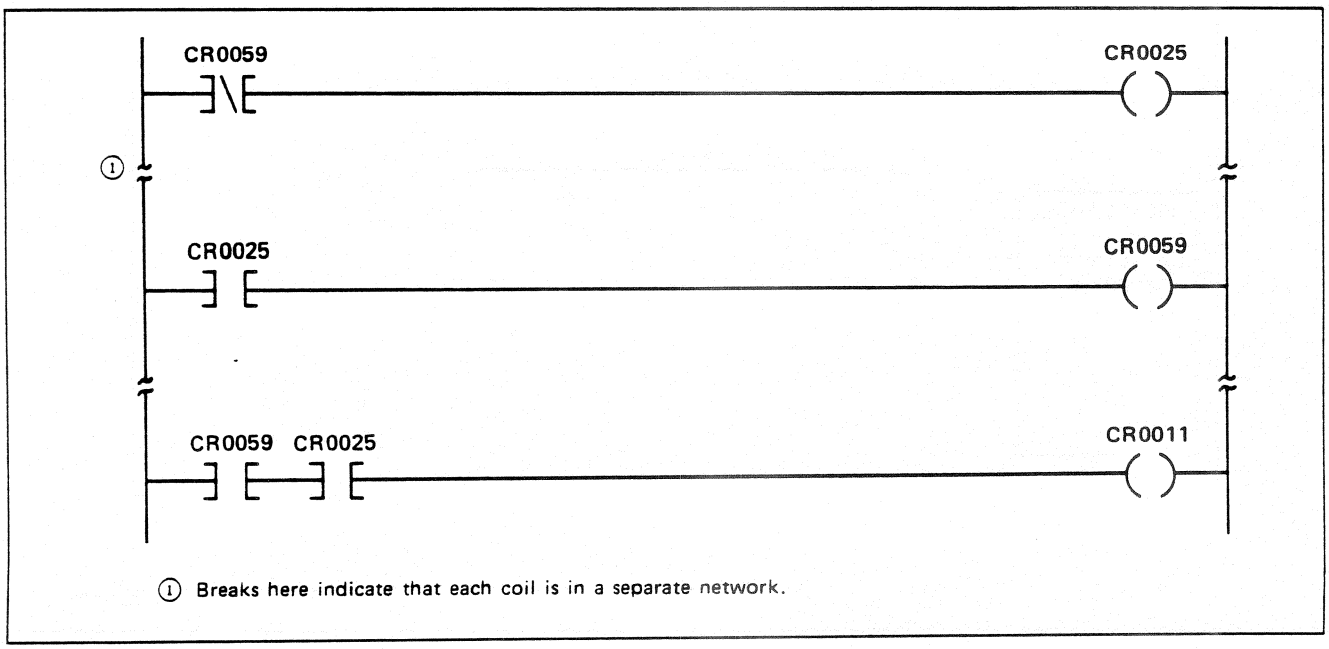


Figure 10-3. Correct Programming Order

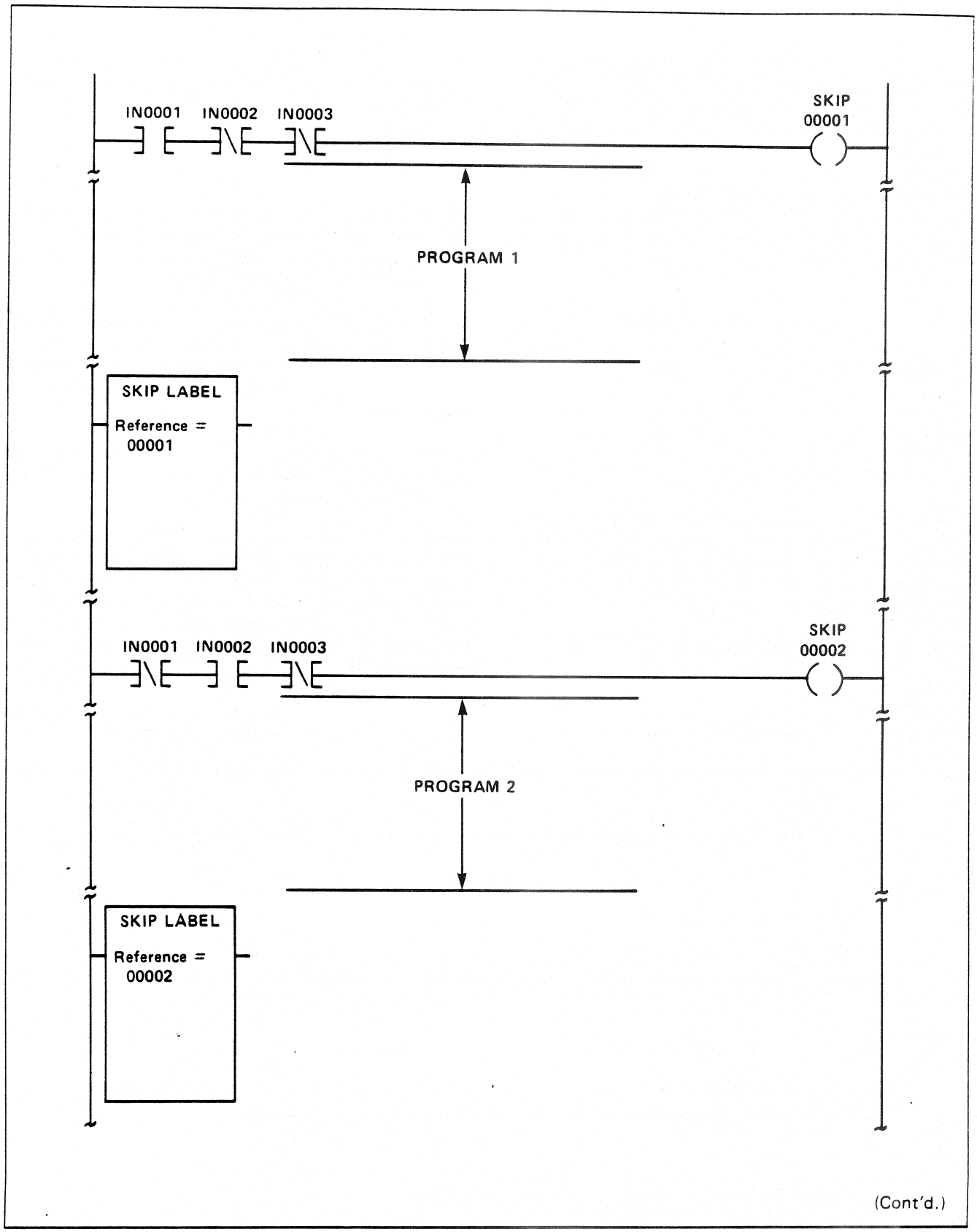


Figure 10-4. Multiple Programs

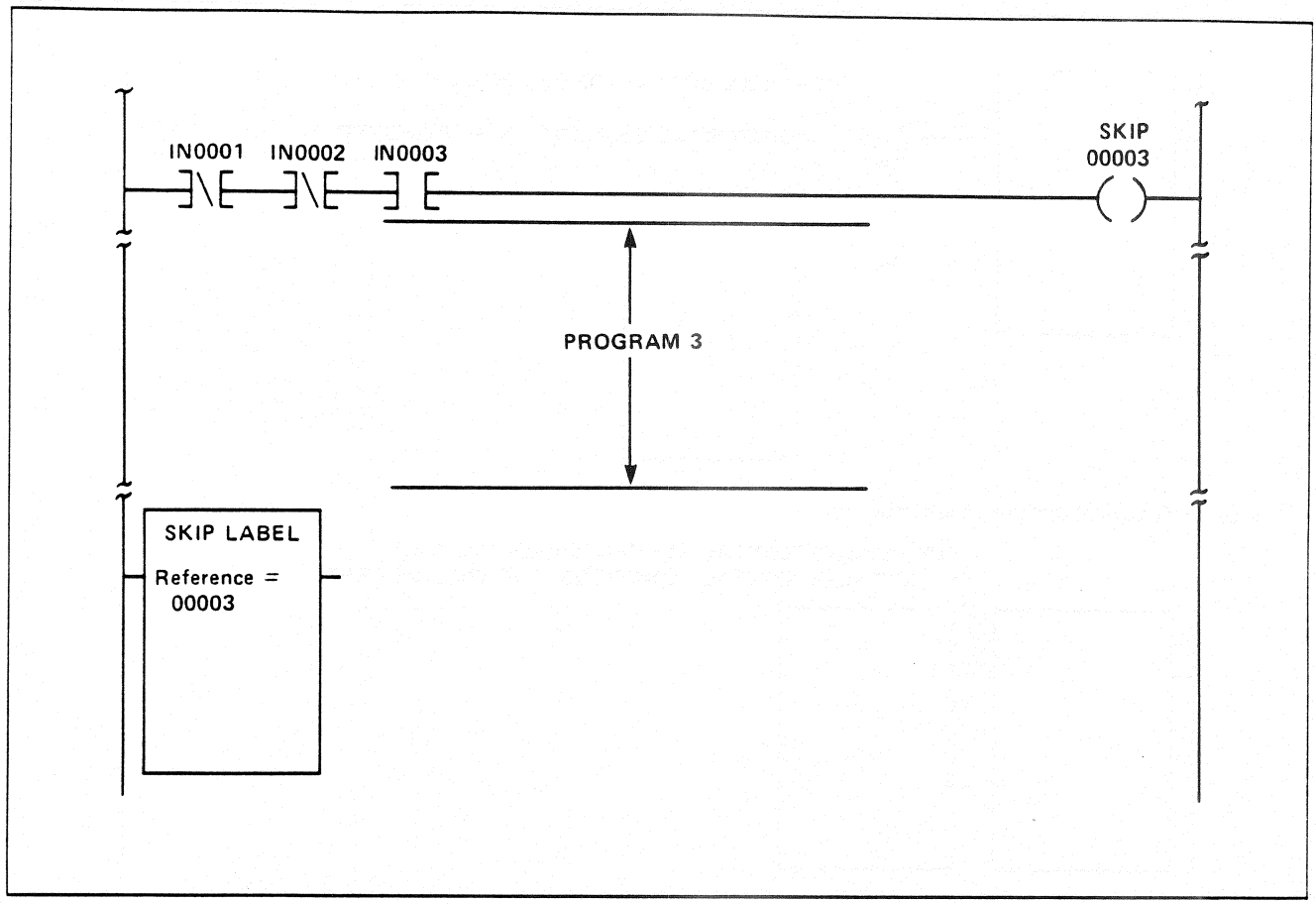


Figure 10-4. (Cont'd.)

Note

The MCR Coil (005) function de-energizes outputs contained between the MCR coil and associated label when the MCR coil is de-energized. The Skip (004) function allows the outputs contained between the Skip coil and associated label to hold their last states when the Skip coil is de-energized. When dividing a program, either the MCR and the Skip functions can be used as required by the machine or process.

Where critical outputs must be serviced more frequently, the networks containing these outputs can often be repeated in each program portion or placed outside the influence of the MCR or Skip coils.

Obviously, unless absolutely necessary, these steps should not be taken. Care must be exercised whenever dividing up a program in this way.

10-3. NETWORK RECONSTRUCTION

The Advanced Program Loader performs a "reconstruction," if possible, on each network as each is inserted into memory. This allows for more efficient use of memory by eliminating unnecessary horizontal elements and vertical branch elements. As the result, the spacing and/or positioning of the elements in the inserted network often is changed from the network actually constructed by the programmer. Also, when possible, any programmable function blocks or other elements contained in a network are moved to the left portion of that network.

An example of the horizontal elements and vertical branch element removal is shown in Figure 10-5. The reconstruction of Figure 10-5(a) results in the element positioning shown in Figure 10-5(b).

In summary, the reconstruction process performs the following 2 functions:

- Removes unnecessary vertical branch elements and horizontal elements

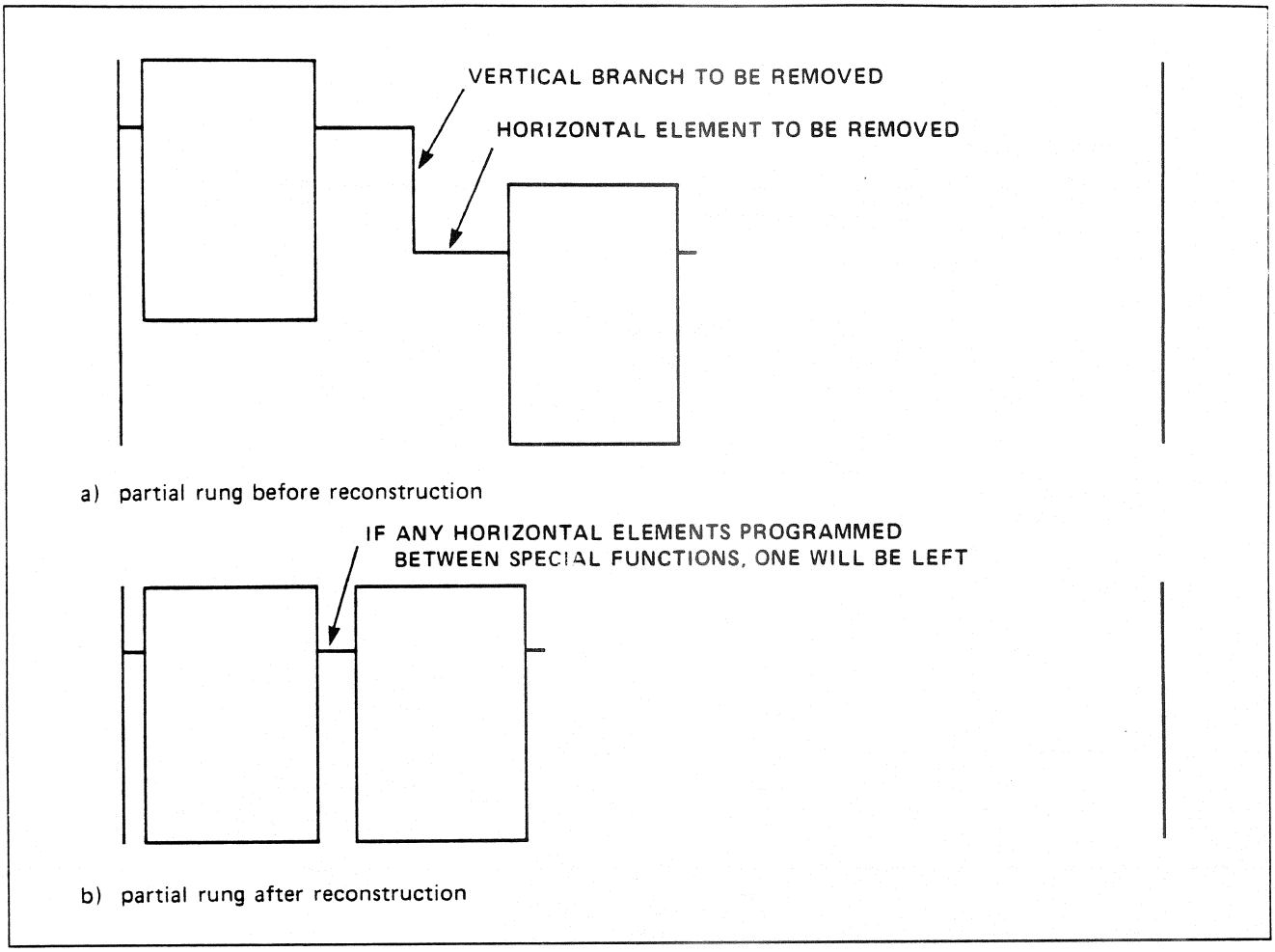


Figure 10-5. Reconstruction Example

- Moves elements to the left portion of the network

10-4. NETWORK CONSIDERATIONS

This Paragraph describes the following network considerations which should be kept in mind when programming:

- Network element execution (10-4-1)
- Special functions programming (10-4-2)

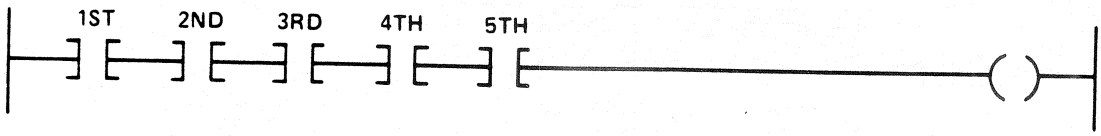
10-4-1. NETWORK ELEMENT EXECUTION

The elements contained in the ladder diagram networks are solved in a predictable manner: all contacts and programmable functions are solved or executed before the coil states are updated. The order of contact and programmable function execution illustrated in Figure 10-6 is described in the following 3 points:

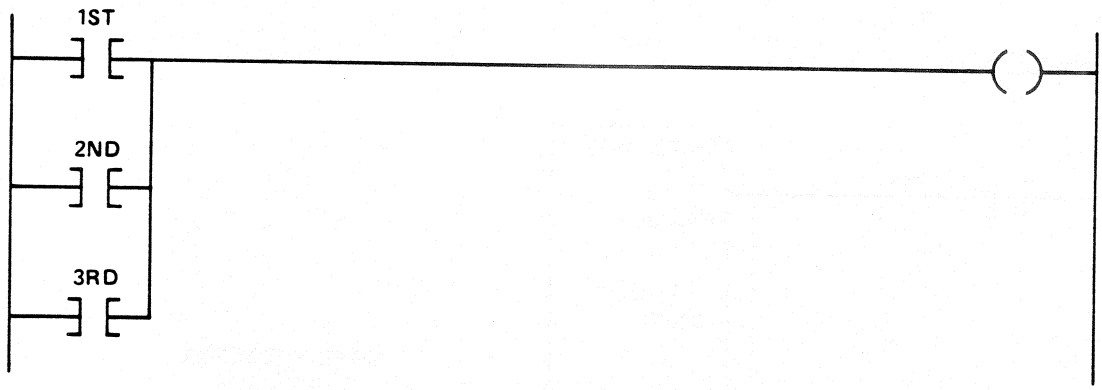
1. The contacts shown in each part of Figure 10-6 represent programming elements. These elements can consist of contacts or programmable functions.

2. Contacts or programmable functions are solved from left to right for series elements and from top to bottom for parallel elements. See Figures 10-6(a) and 10-6(b).

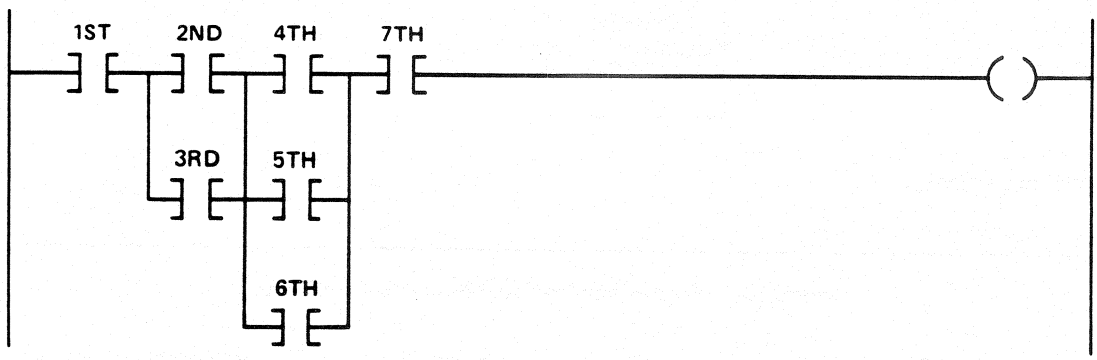
3. When series and parallel elements are combined, the order of execution is as shown in Figure 10-6(c).



a) series elements



b) parallel elements



c) combination series and parallel elements

Figure 10-6. Execution Example A

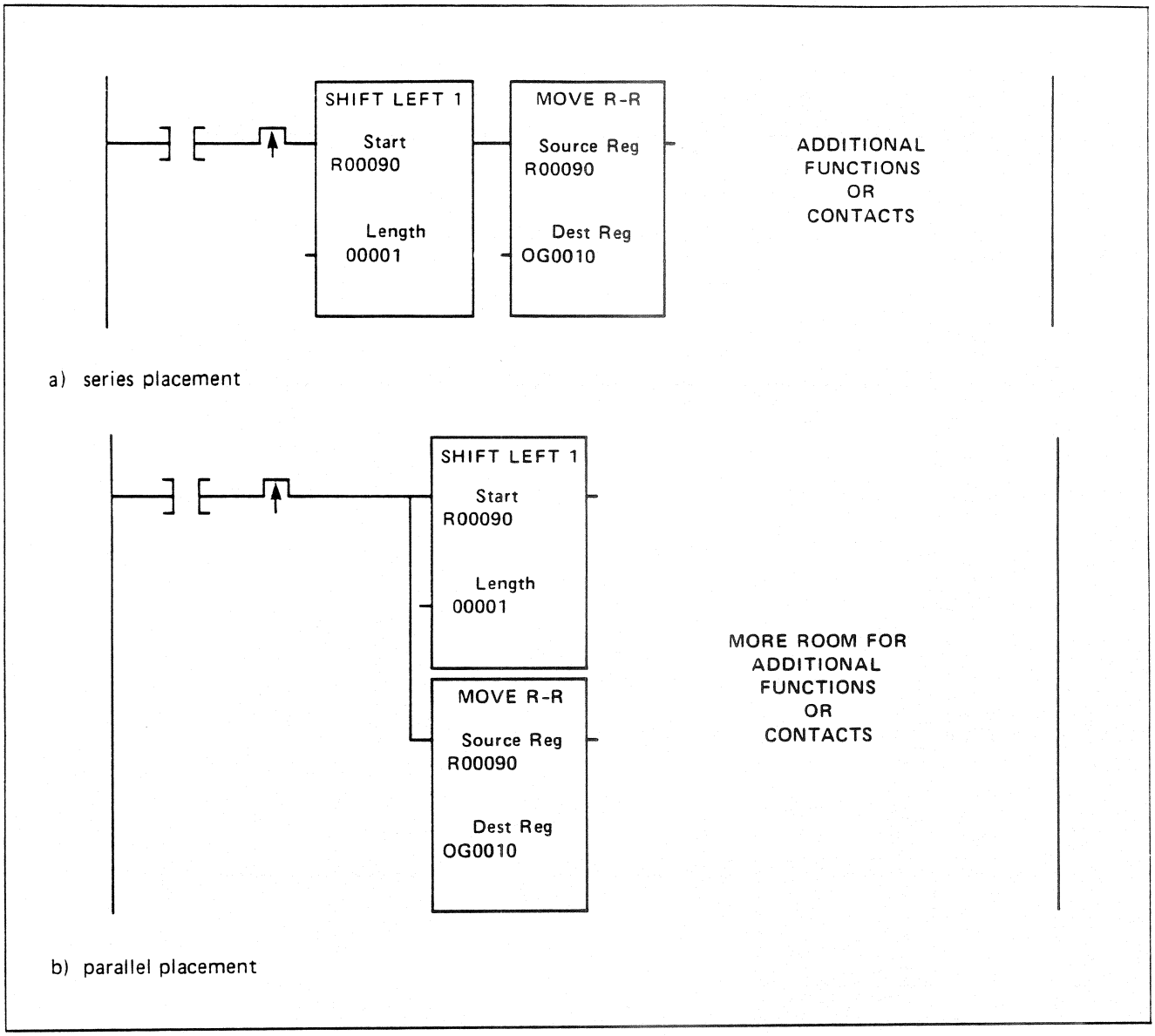


Figure 10-7. Reconfiguring Programmable Functions

The order of execution affects the operation of a rung, especially when several programmable functions are contained in the network. An example showing the relationship between the execution of programmable functions is shown in Figure 10-7. Observe the Figure and note the following 3 points:

1. The data contained in register R00090 is shifted left one place and then moved into OG0010 in Figure 10-7(a).

2. The same programmable functions shown in Figure 10-7(a) are arranged in parallel in Figure 10-7(b). This arrangement would allow more programmable functions to be programmed to the right, increasing the total number of possible programmable functions in the network.

3. If the 2 programmable functions shown in Figure 10-7(b) had been interchanged, the results from the execution of the network would be different. The data would be moved to output group OG0010 before being shifted, thereby resulting in the incorrect data being transferred.

A final example showing the order of execution of a network containing contacts and programmable functions is shown in Figure 10-8. Observe that the coils are updated from bottom to top only **after** all contacts and programmable functions are solved.

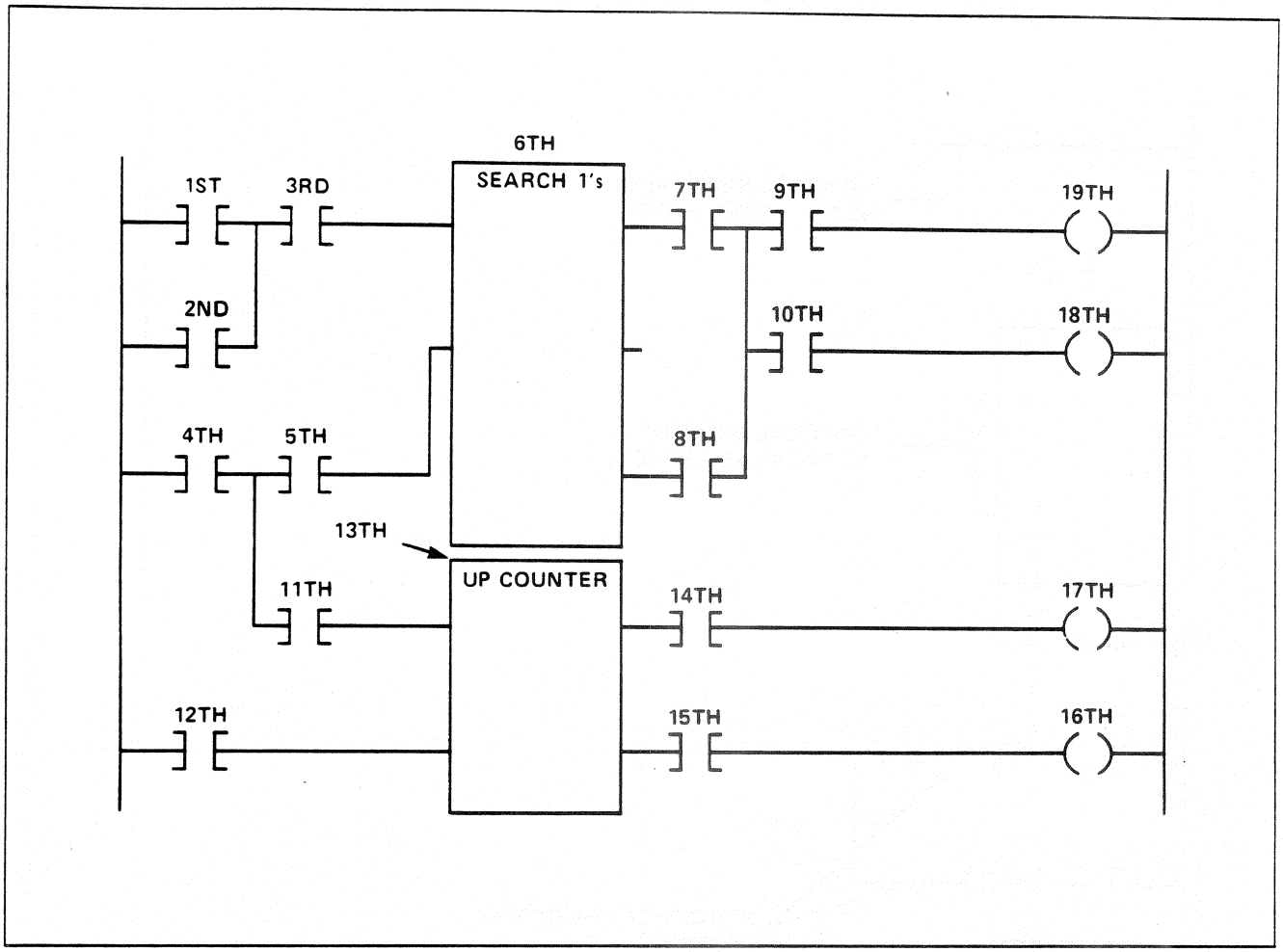


Figure 10-8. Execution Example B

10-4-2. PROGRAMMABLE FUNCTION PROGRAMMING

When programming basic or advanced functions, the following 3 considerations should be kept in mind:

1. The entry of vertical branch elements immediately to the right of the programmable functions is prohibited by the APL Loader. See Figure 10-9(a). Horizontal elements placed to the right of the programmable function before the vertical branch allow the vertical branch element to be programmed as shown in Figure 10-9(b).

2. At times, in order to fit additional programmable functions in a network, it may be necessary to add "dummy" contacts, as shown in Figure 10-10. The network shown in Figure 10-10(a) is prohibited from entry because the APL Loader positions the programmable functions by removing vertical branch elements. Note: When vertical branch elements are removed, the programmable functions shown in Figure 10-10(a) require more than 7 horizontal paths. The dummy contact, addressed such that the contact set never conducts, shown in Figure 10-10(b), allows the network to be inserted.

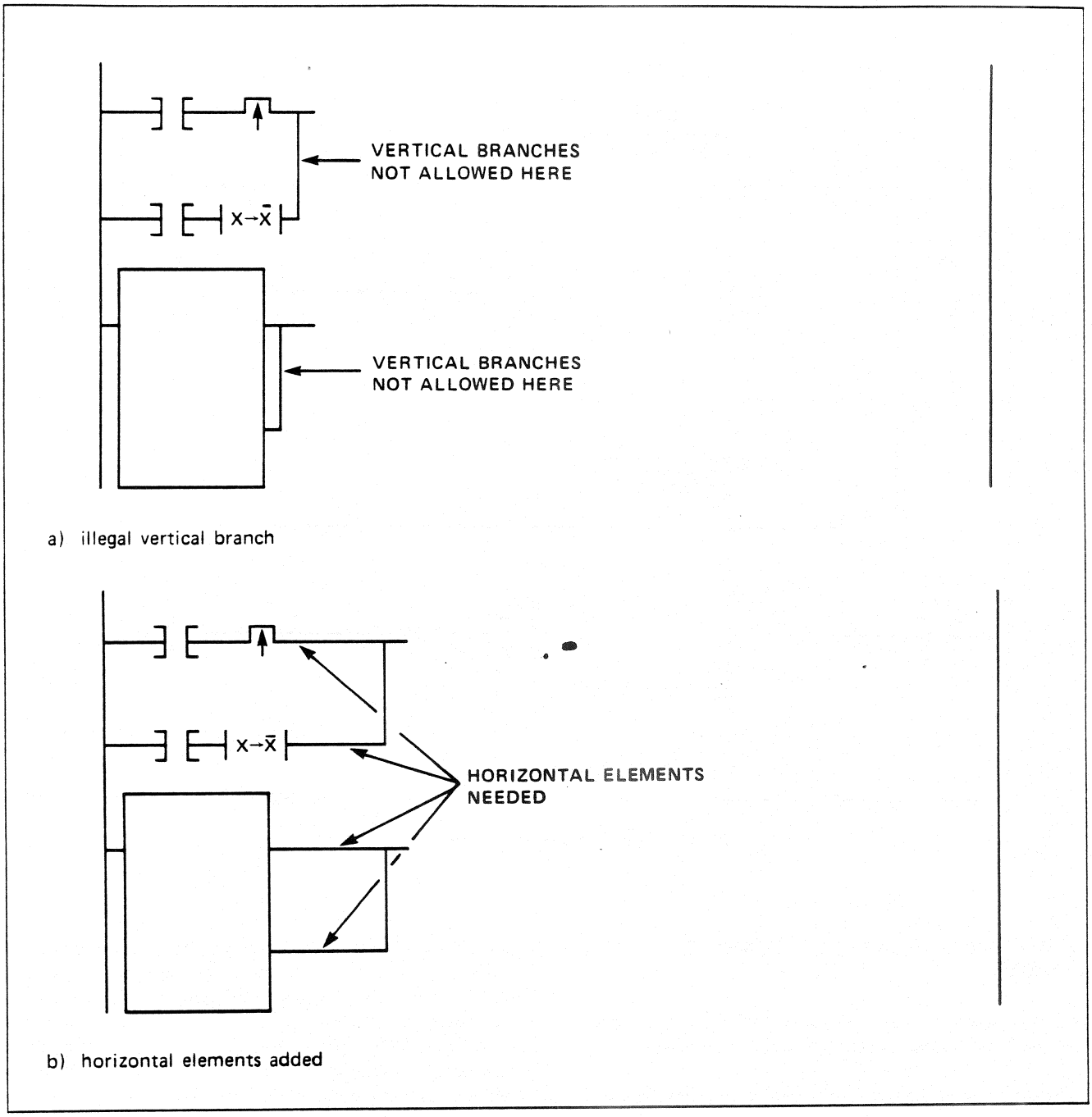
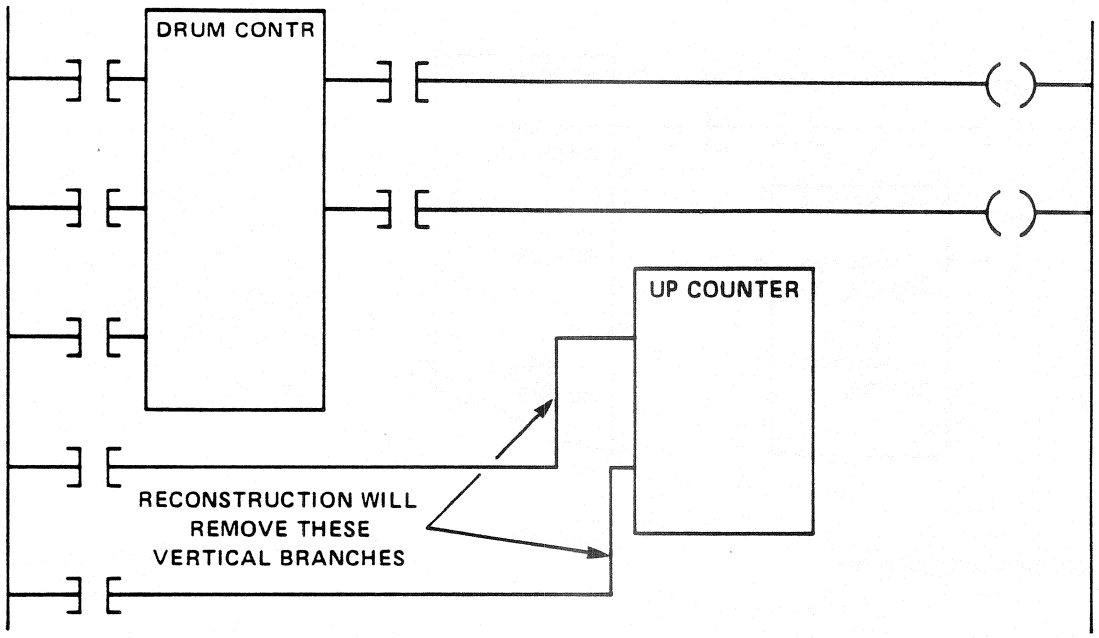
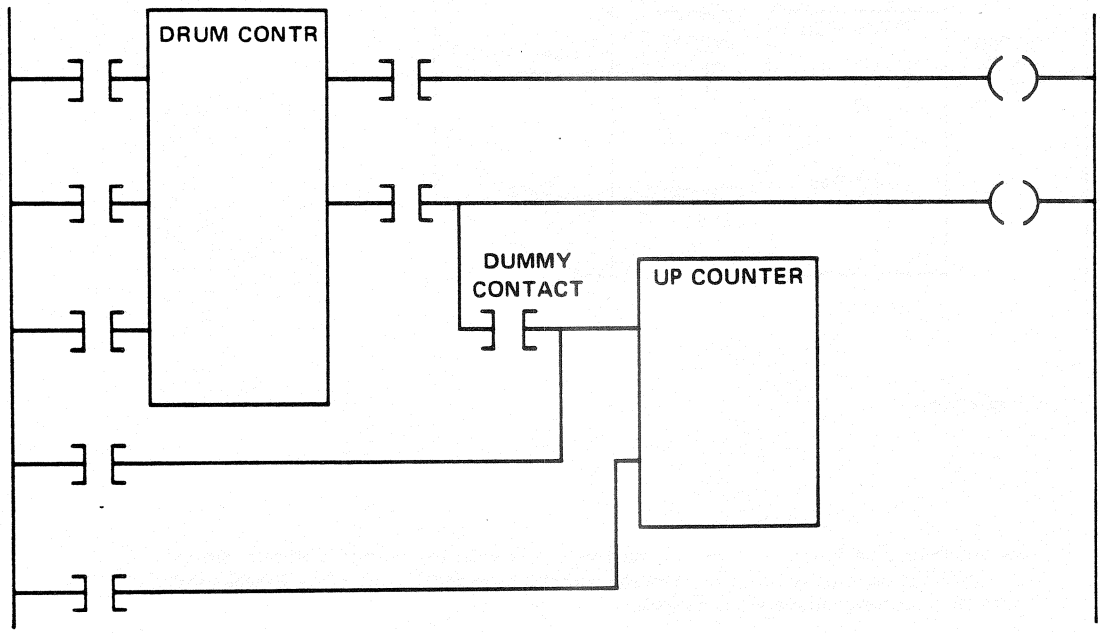


Figure 10-9. Vertical Branch Programming

3. As noted earlier, the APL Loader reconstructs the networks as they are inserted into the processor's memory. During this time, blocks without any outputs connected are repositioned such that the unconnected outputs could, if necessary, connect in a straight, horizontal line to another function block or coil. Figure 10-11(a) illustrates an example where the position of the programmable function block causes repositioning. The Move R-R (050) function block will automatically be moved vertically down from the position shown. In fact, this configuration will not be accepted into processor's memory since the Move R-R function is moved partially off the screen. Figure 10-10(b) shows how a dummy contact can be added to allow the Move R-R function to be contained in the network.

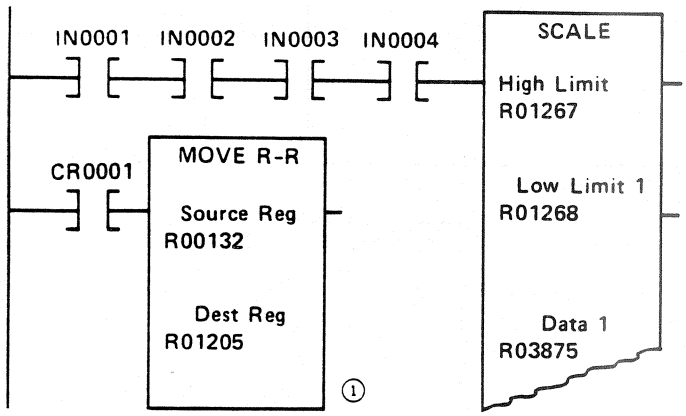


a) network cannot be inserted

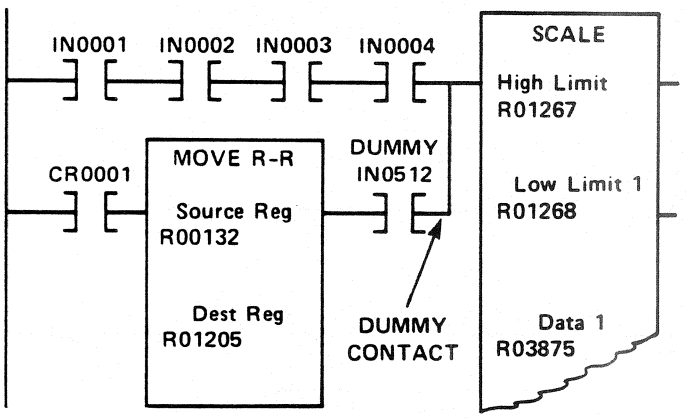


b) network with dummy contact

Figure 10-10. Dummy Contact Example 1



a) illegal rung configuration



b) legal configuration

① This programmable function block is moved vertically until the output is able to run in a straight line horizontally to a coil. With the configuration shown, the block exceeds the 7 vertical spaces available and, as a result, cannot be inserted.

Figure 10-11. Dummy Contact Example 2

10-5. PROGRAM EXECUTION TIME/MEMORY REQUIREMENTS

It may be desirable to estimate the amount of memory required for the ladder diagram even though the PC status function of the APL Loader displays this figure after the ladder diagram is inserted into memory. (See the APL Programming Manual (NLAM-B806) for details on monitoring this function. Also, in this manual, see Figure 5-3, callout 3.) To manually calculate the memory required, use Tables 10-3 and 10-4, Memory Requirements.

In addition, any documentation feature labels or comments associated with the ladder diagram require the following memory:

- Documentation labels over contacts and coils each requires 4-1/2 words, regardless of whether 1 through a maximum of 6 characters is used.
- Comments require 1 word per 2 characters or spaces plus 2-1/2 overhead words.

The programmer must allocate memory for documentation during the configuration process, as described in the APL Programming Manual (NLAM-B806). Also see Paragraph 5-4-2 and Figure 5-4, callout 2, in this manual.

10-6. THROUGHPUT CONSIDERATIONS

Throughput can be simply defined as the ability of the HPPC-1500/-1700 system's outputs to respond to changing input conditions within a given time, as measured in milliseconds. More specifically, throughput defines the time necessary for the HPPC-1500/-1700 system to accomplish all of the following:

- Sense an input state change (on-to-off or off-to-on)
- Communicate the change of state from the input module to the Logic Processor Module's I/O image RAM
- Modify, when the ladder diagram scan occurs, one or more outputs contained in that I/O image RAM by using the new input state data
- Communicate the change of state of one or more outputs from that I/O image RAM to the output module
- Respond to the new change of state by controlling the output module's terminal

TABLE 10-3. STANDARD FUNCTIONS' MEMORY REQUIREMENTS

Type	Function	Words
Coils	Bit Set	4.2
	Bit Clear	4.5
	Bit Follow	4.5
	Skip Coil	3
	MCR Coil	3
	CR Coil	1.5
	Latch Coil	1.5
Signal Conditioners	Transitional	3
Labels	MCR Label	1.5
	Skip Label	1.5
Timers	Time On 1.0	4.5
	Time On 0.1	
	Time On .01	
	One Shot 1.0	6
	One Shot 0.1	
	One Shot .01	
	One Shot SCN	
Counters	Up Counter	4.5
	Down Counter	4.5
Math Functions	Add A + B=C	6
	Sub A - B=C	6
	Mult A x B=C	7.5
	Div A ÷ B=C	7.5
Shift Registers	Shift Left 1	4.5
	Shift Right 1	4.5
Conversions	BIN-BCD R	6
	BIN-BCD*R	7.5
	BCD-BIN R	4.5
	Analog In	4.5
	Analog Out	4.5
Move Functions	Move R-R	4.5
	Drum Control	7.5
Logic Functions	Zero Table	4.5
Table Operations	Diagnostic	4.5
Comparison	Compare R-R	4.5
I/O Functions	Update I/O	6

TABLE 10-4. ADVANCED FUNCTIONS' MEMORY REQUIREMENTS

Type	Function	Words
Coils	Jump Coil	3
Signal Conditioners	Inverting	1.5
Labels	Jump Label	1.5
Timers	Time Off 1.0 Time Off 0.1 Time Off .01	4.5 4.5 4.5
Math Functions	Square Root Negate Negate * Add * A + B=C Sub * A - B=C Average	6 4.5 6 7.5 7.5 9
Shift Registers	Shift Left N Shift Right N	7.5 7.5
Conversions	BIN-BCD T BCD-BIN T Scale	6 6 10.5
Move Functions	Block Move Byte Move Move R-T Move T-R N-Bit Move	4.5 6 7.5 7.5 10.5

TABLE 10-4. ADVANCED FUNCTIONS' MEMORY REQUIREMENTS (Cont'd.)

Type	Function	Words
Logic Functions	Complement Search 1's AND R-T OR R-T XOR R-T AND T-T OR T-T XOR T-T	6 6 7.5
Table Operations	Bit Operate Search = Search > = Ascend Sort	6 7.5 7.5 10.5
Stack Operations	First In First Out Last Out	7.5 7.5 7.5
Scan, Alteration	Lock Scan U.D.S.F.	4.5 6 to 24
Comparison	Limit Test R Compare T-T Compare R-T Limit Test T	6 7.5 7.5 9

Throughput is dependent on the following 4 conditions:

1. Input circuit filtering delays, nominally 10 milliseconds for an AC input.

2. The time required to transfer data between the Logic Processor Module's I/O image RAM and the I/O modules. This is referred to as the I/O scan time.

3. The ladder diagram scan time, as described in Paragraph 10-4.

4. Output circuit delays; that is, the time required for the output module to respond to a commanded change, would be nominally 2 to 10 millisec. for an AC output.

Since the ladder diagram scan and I/O scan run concurrently and are initiated together, the longer scan time—either the ladder diagram scan or I/O scan—determines the scan time. Paragraph 5-5 discusses the program scan while Paragraph 5-6 contains a description of the I/O scan.

10-6-1. THROUGHPUT EXAMPLE

An example showing typical times for an HPPC-1500/-1700 system is listed in Table 10-5. Observe the Table and note the following 3 points:

1. The worst-case I/O and program scan time is 25 milliseconds in this example.

2. Three scans are required to change the state of the output after an input changes state.

3. The example assumes the AC input and AC output modules being used have a nominal 10 millisecond delay.

TABLE 10-5. THROUGHPUT EXAMPLE

Event	Time (in msec.)
I/O module receives signal:	10
I/O image RAM receives signal:	25
Ladder diagram changes state of output in I/O image RAM:	25
I/O Image RAM change of state sent to I/O Rack:	25
Output(s) change state:	10
Total throughput time (worst case):	95

10-6-2. REDUCING THROUGHPUT

System throughput can be reduced in either of the following 2 ways:

1. Reducing the I/O scan or ladder diagram scan, whichever is longer

2. Using the Update I/O (054) programmable function, as described in Paragraph 7-21

In many instances strategic placement of a limited number of Update I/O functions will satisfy throughput requirements. However, since each Update I/O (054) function requires approximately 3.0 milliseconds to execute, the use of this function is limited.

In cases where too many Update I/O functions would be required to provide the desired throughput, the alternative is to reduce either the program scan or I/O scan times, as described next.

10-6-3. REDUCING I/O SCAN

The I/O scan time can be reduced as listed in the following 2 points:

1. Use a second I/O Processor Module. Divide the SIMs to two equal groups and connect half of the SIMs to the second Module. In other words, distribute the I/O evenly between the 2 Modules. If done properly, the I/O scan time will be nearly cut in half.

Note

A fully configured SIM adds approximately 5 milliseconds to the I/O scan, while a SIM configured for one-half the I/O possible would require 3 milliseconds. When dividing SIMs for assignment to a second I/O Processor Module, the number of I/O configured for each SIM must be considered in order to evenly divide the time in half.

2. Configure only the I/O actually being used in each SIM. (Often unused I/O are assigned for future expansion. Although this may be a good engineering practice, it will result in increased scan times.)

10-6-4. REDUCING PROGRAM SCAN

The ladder diagram program scan can be reduced in a variety of ways; however, many of the various means increase the complexity of the ladder diagram. Increased ladder diagram complexity makes program debugging and subsequent maintenance operations more difficult and, for this reason, should be avoided whenever possible.

The ladder diagram program scan can be reduced in the following 3 ways:

1. Use of logic coils as described in Paragraph 10-2-1.

2. Use of the Skip Coil (004) and MCR Coil (005) functions to make portions of the ladder diagram inactive, as described in Paragraphs 7-2 and 7-3, respectively. Those portions of the ladder diagram contained within the MCR or Skip Coil and corresponding Label functions will still be accessed from memory, but they are **not** executed. This results in a reduction in the program scan. Paragraph 10-2-4 describes how to divide a program into parts to reduce program scan time.

3. Use of the advanced programming functions, when available, can often reduce overall program scan time. For example, consider the case described in Paragraph 8-31-2 where 5 different limits are tested using one Limit Test T (113) function. This same programming would require many more programmable function elements if performed only by means of the standard functions.

REFER TO INSIDE FRONT COVER FOR A QUICK-LOCATOR LISTING OF THE STANDARD PROGRAMMABLE FUNCTIONS.

ADVANCED FUNCTIONS QUICK-LOCATOR LIST

Type	APL No.	Designation	Par. No.	Page No.
Coils	006	Jump Coil	8-2	8-3
Signal Conditioners	008	Inverting	8-3	8-5
Labels	023	Jump Label	8-2	8-3
Timers	031	Time Off 1.0	8-4	8-6
	032	Time Off 0.1	8-4	8-6
	033	Time Off .01	8-4	8-6
Math Functions	042	Square Root	8-5	8-8
	043	Negate	8-6	8-11
	044	Negate *	8-6	8-11
	069	Add * A + B=C	8-7	8-13
	071	Sub * A - B=C	8-7	8-13
	093	Average	8-8	8-17
Shift Registers	089	Shift Left N	8-9	8-18
	090	Shift Right N	8-9	8-19
Conversions	074	BIN-BCD T	8-10	8-25
	075	BCD-BIN T	8-11	8-27
	118	Scale	8-12	8-29
Move Functions	076	Block Move	8-13	8-32
	077	Byte Move	8-14	8-36
	094	Move R-T	8-15	8-40
	095	Move T-R	8-16	8-43
	119	N-Bit Move	8-17	8-47
Logic Functions	078	Complement	8-18	8-52
	079	Search 1's	8-19	8-55
	097	AND R-T	8-20	8-59
	098	OR R-T	8-20	8-59
	099	XOR R-T	8-20	8-59
	100	AND T-T	8-21	8-62
	101	OR T-T	8-21	8-62
	102	XOR T-T	8-21	8-62
Table Operations	081	Bit Operate	8-22	8-65
	106	Search =	8-23	8-69
	107	Search ⟩ =	8-23	8-69
	108	Ascend Sort	8-24	8-73
Stack Operations	103	First In	8-25	8-79
	104	First Out	8-25	8-79
	105	Last Out	8-25	8-80
Scan Alteration	052	Lock Scan	8-26	8-89
	128	UDSF	8-27	8-92
Comparison	067	Limit Test R	8-28	8-99
	091	Compare T-T	8-29	8-101
	092	Compare R-T	8-30	8-105
	113	Limit Test T	8-31	8-109

Westinghouse

numa-logic

Catalog No. NLAM-B807
Style No. B-807/16-350

January, 1986

Westinghouse Electric Corporation
Industry Electronics Division
Madison Heights, Michigan 48071