Chapter 9
Programming Ladder Logic

This chapter identifies and describes the elements of ladder logic programming. It includes the following topics:

Network elements

Network scanning

User logic reference numbers

Programming functions

Programmer Tape/disc

Features

Programming software is available on tape and disk media for creating, debugging, and editing user logic programs. Major features of programmer software are:

The user can create and modify networks: enter, delete, modify, and move logic elements.

The user can display the application program using ladder logic networks, I/O and register element listings on reference screens.

The Screen displays are on line, showing power flow and dynamic states of program logic elements.

Search and Trace commands support rapid location within a program of multiple logic elements referenced to a common element.

Functions

Standard logic elements

Configured Loadable functions

Configured Extended Memory functions

Configured ASCII functions

Reference Display/Write

Search

Trace/Retrace

Enable/Disable

Force ON/OFF

Simple ASCII Message Editor

Operations

➤ **CAUTION Always make sure that your 3.7v lithium battery is installed and that the polarity is correct. The BATTERY LOW indicator will be ON if the battery power level is low or if the polarity is wrong.**

Battery Backup

Lithium batteries will support the largest memory (32K), in the worst case environment, with the worst case memory bit pattern, for at least nine months without power.

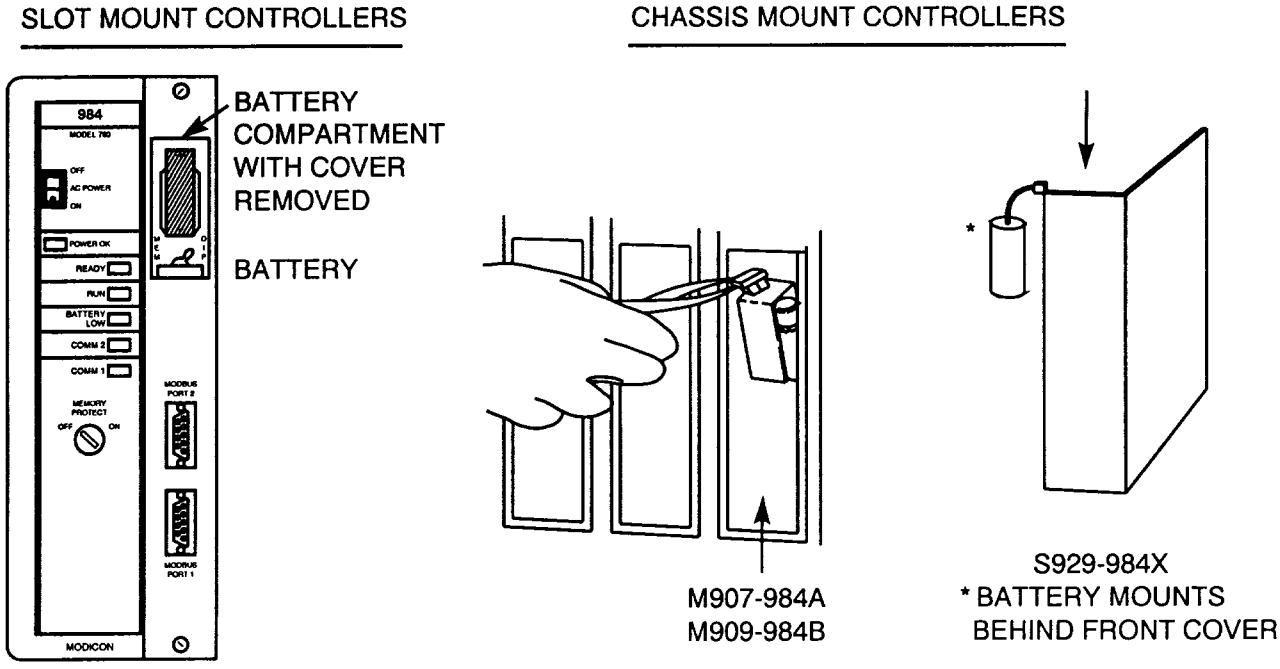
When the battery low light comes on, the batteries have sufficient life remaining to power the CMOS memory for 14 days.

Battery status is available for programmed use in bit 12 of the 1st word used by the Get Status block. Shelf life of the battery is five years. See the figure below for battery replacement procedure.

Nominal voltage for a 3.7v lithium battery = 3v.

Figure 9-1 Battery Check/Replacement

After the controller has been configured, traffic copped, and programmed, only remove or replace the batteries when the controller is ON. Removing the battery(s) when power is OFF may result in the controllers programming being lost.



Overview

P190 must be attached to the 984
984 configured to present application
Program tape loaded in the P19O
Programming is in ladder logic
Any one of the 70 element positions is called a node
Power cannot flow from right to left
Coils are automatically flushed right
Scan begins at network one and solves each column from top to bottom moving left to right
Information is available in order of solve

984 Set Up

User programs for the 984 PC are listed and described in Chapter 7. The following procedure must be followed to enter a program:

Step 1 The 984 PC must be configured to the user's specifications from a hardware and software perspective

Step 2 Insert the 984 Programmer Tape into the P190 tape drive (front upper right corner).

Step 3 Press the red INIT and INIT LOCK keys on the P190 panel simultaneously. When the tape is loaded and rewound, the following softkeys are displayed.

Step 4 Enter a unit number (controller address) into the AR. The unit number can be within the range of 1 to 247.

Step 5 Press the ATTACH software label key. The RESET LEVEL display screen will appear. See Reset Level Display, opposite page.

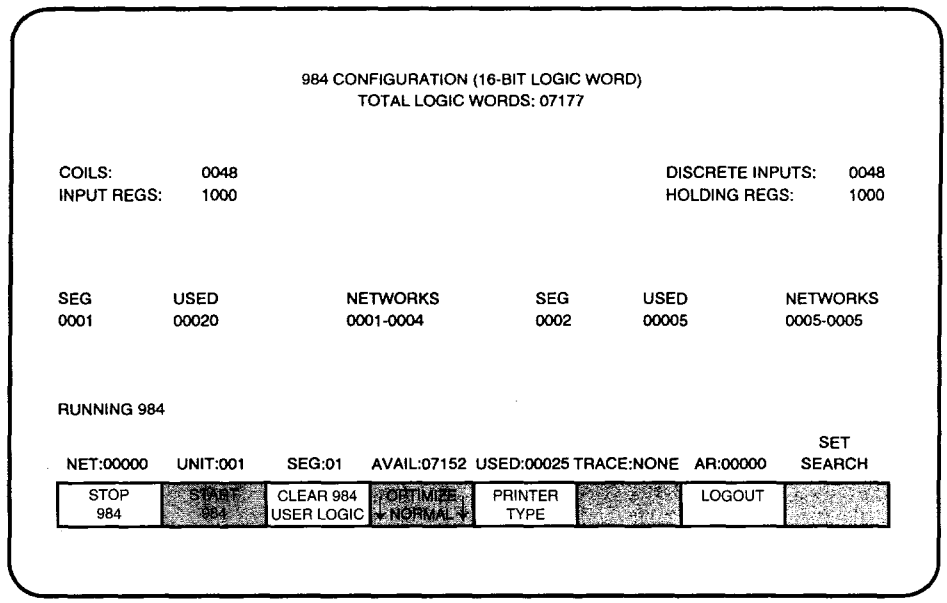
Step 6 Stop the 984 by pressing the STOP 984 software label key.

Step 7 Press CONFIRM software label key

Step 8 Press the EXIT key followed by the START NEXT key to begin entering a program.

Reset Level Display The figure below shows the information screen that is displayed whenever the 984 is attached to the P190. Since the 984 must have been configured in order to attach, element block size and number of drops has already been specified. A minium of 1 segment per drop is required. If any programming has been done, network to segment and memory word to segment allocation is shown. If the 984 has been configured for ASCII, "total message words" and "highest message used" will also be displayed.

Figure 9-2 Reset Level Display Screen



The software label keys are used to change the 984 PC's status and can be reached at any time while programming by pressing the SHIFT and RESET/EXIT keys simultaneously.

STOP 984 - Stops controller from solving logic. Requires configuration.

START 984 - Runs internal diagnostics, then starts controller solving logic. Requires configuration.

CLEAR 984 User Logic - Clears user logic from 984's memory. Controller must be stopped and requires confirmation.

Reference Numbers

Each element in a network is assigned a five-digit reference number to identify it. The first digit identifies the type of reference.

OXXXX: Coil or Discrete Output

A discrete (ON/OFF) signal that is controlled by logic.

Can be used to drive an output device through an output module.

Can be used internally to drive one or more contacts in user logic.

1XXXX: Discrete Input

Discrete (ON/OFF) state is controlled by an input module.

Used to drive contacts in user logic.

Can be used repeatedly in user logic.

3XXXX: Input Register

Stores a numerical value as received from an input module from an external source (analog A/D signal, thumbwheel, event counter).

Can store a binary value or a binary coded decimal (BCD) value.

4XXXX: Output Register or Holding Register

Stores a numerical value to send to an output module, for an external source (analog D/A signal, setpoint value, message display).

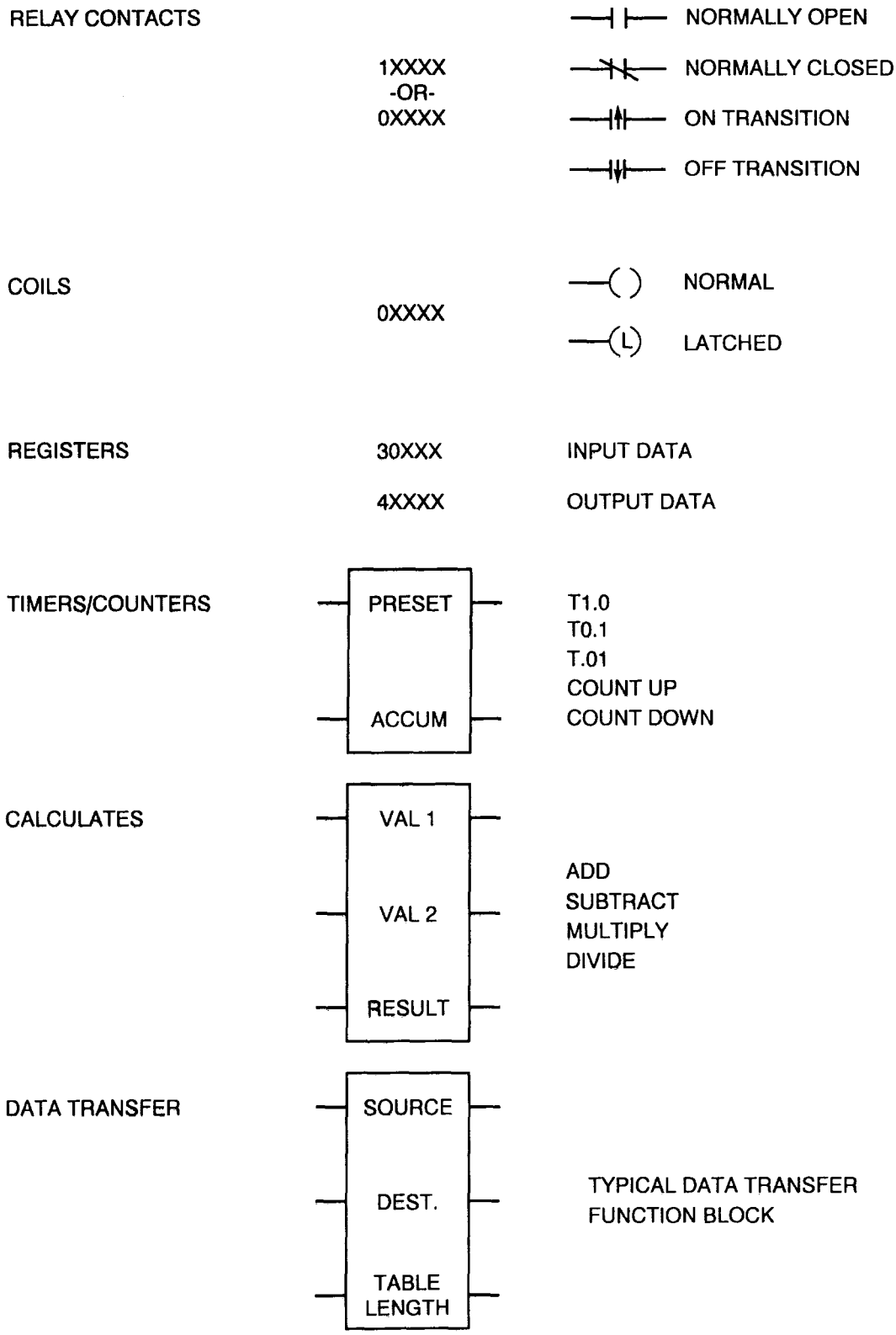
Can store a binary value or a binary coded decimal (BCD) value. for out-put or internal use.

6XXXX: Extended Memory Register

Stores a binary value in the Extended Memory area of the controller.

Accessed by Extended Memory instructions programmed in user logic.

Figure 9-3 Ladder Logic Symbols



Network Structure

A network is a set of interconnected logic elements that make up the user's application program. Each network has a maximum height of 7 nodes. Each network has a maximum width of 10 nodes, containing all program elements except coils. The user logic can occupy the whole network area or just a portion of it. Counters, Timers, the Status block, and the Skip block consume two nodes. Calculate blocks and DX function blocks consume three nodes.

Figure 9-4 Programmable Control Network Structure

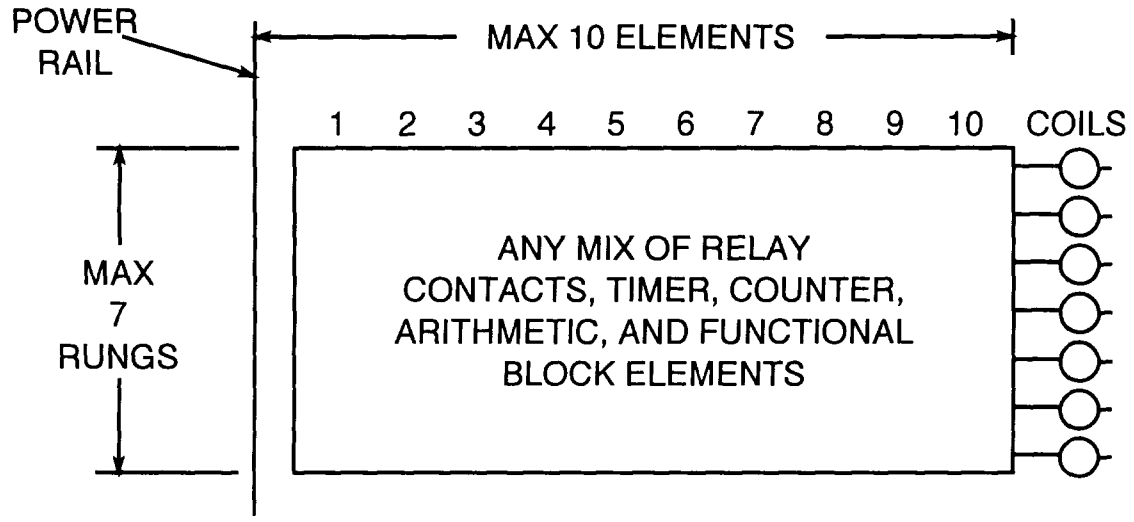
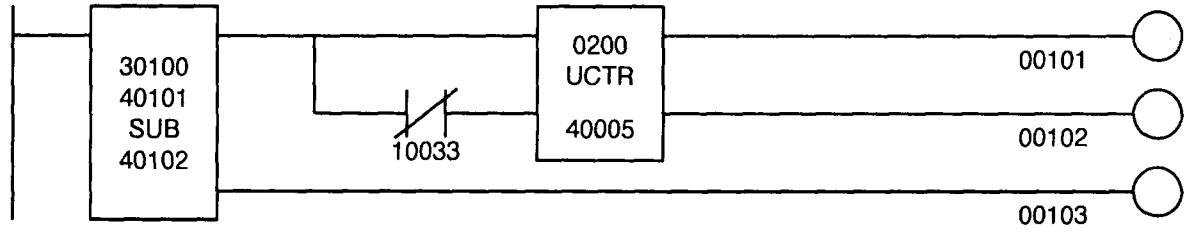


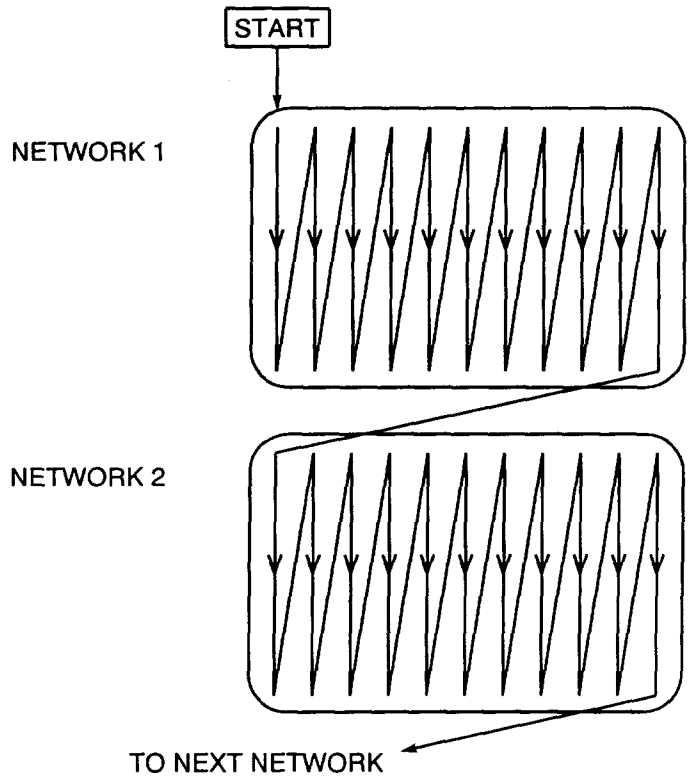
Figure 9-5 Sample Ladder Logic Elements in a Network



Logic Solving Sequence

The user logic is solved column by column from left to right within each network. The 984 solves one element at a time, from top to bottom, left to right. If the 984 is stopped, or loses power in the middle of a network, the logic solution starts over from the first network when power is restored to the controller. See figure below.

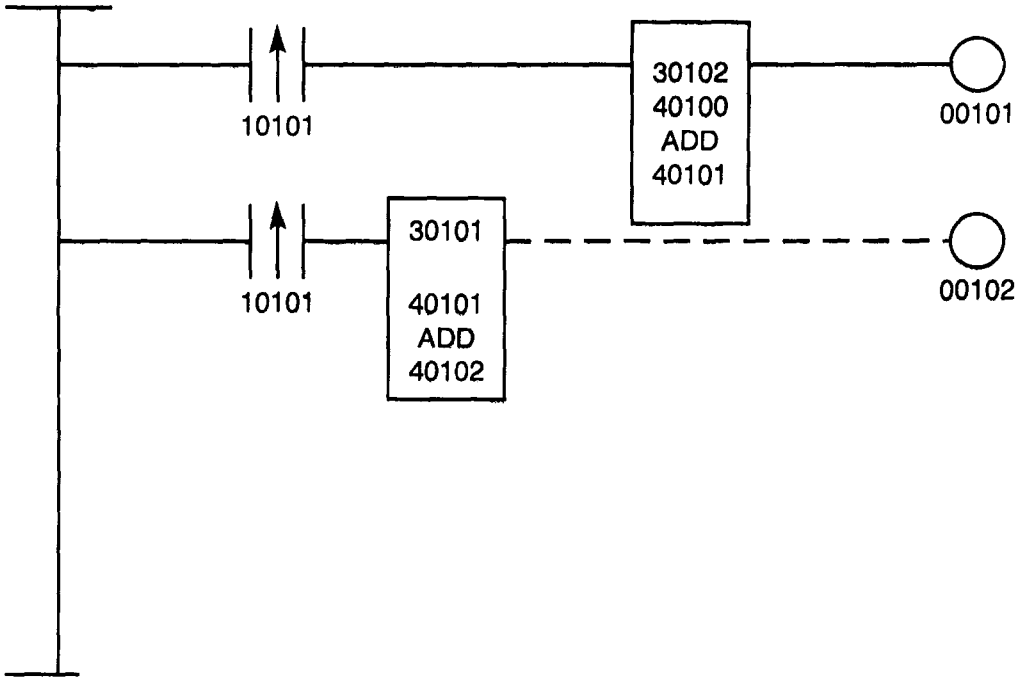
Figure 9-6 Network Solve Sequence



Element Positional Relationships

During the scan in which input 101 is first noted as being on, two adds take place. However, 40102 will receive its total before 40101. Also, coil 00102 will be solved before coil 00101 since it is actually solved in the column immediately following 40102. This is indicated by the dotted line connection from ADD Block 40102 to Coil 00102 in the figure below.

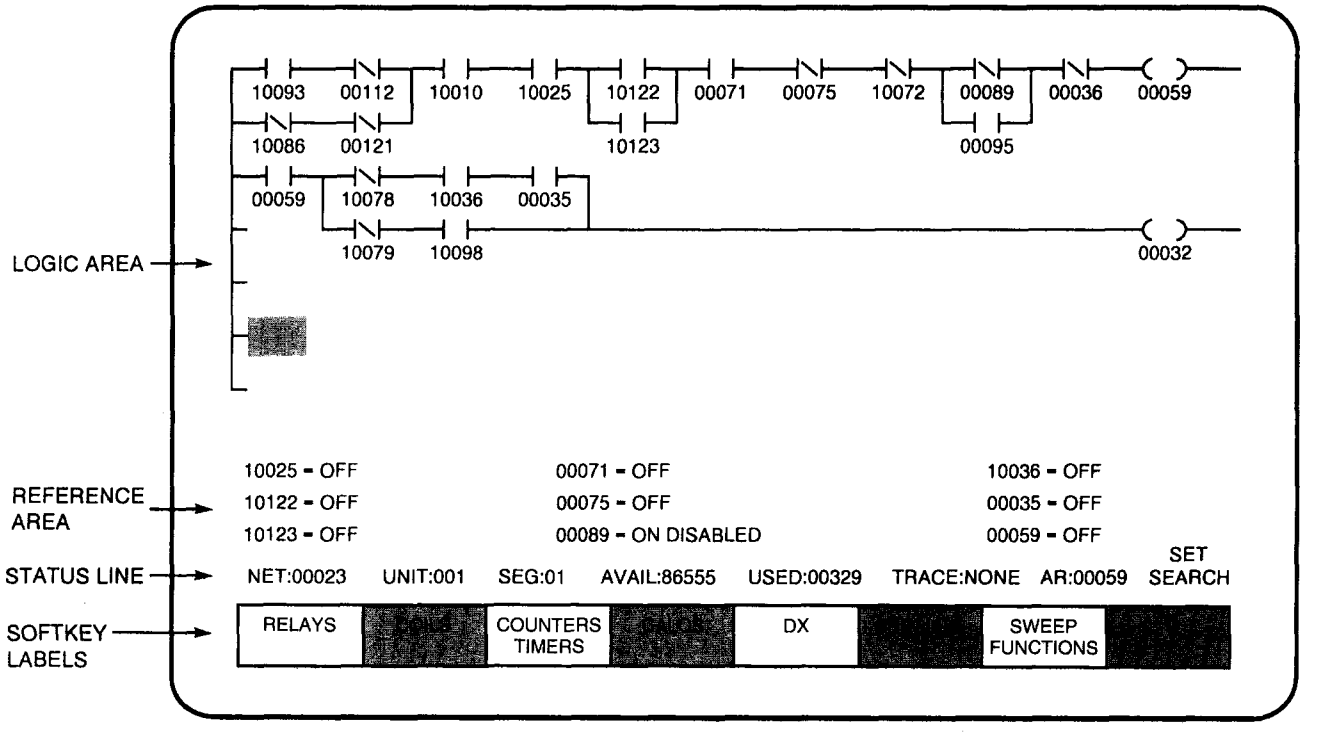
Figure 9-7 Coil Solve Sequence



Program Tape Screens

Logic Screen The figure below shows the screen that appears in the exit level of the program tape after a segment and a network are selected.

Figure 9-8 Logic Screen Example



Logic Area - Displays one network at a time

Reference Area - Displays the status of up to nine discretes or registers

Status line - Displays unit parameters including the assembly register (AR)

Error line - Just above status line (see P190 error messages in appendix)

Softkey Labels - Define the function of the unlabeled keys located just below the display of the P190.

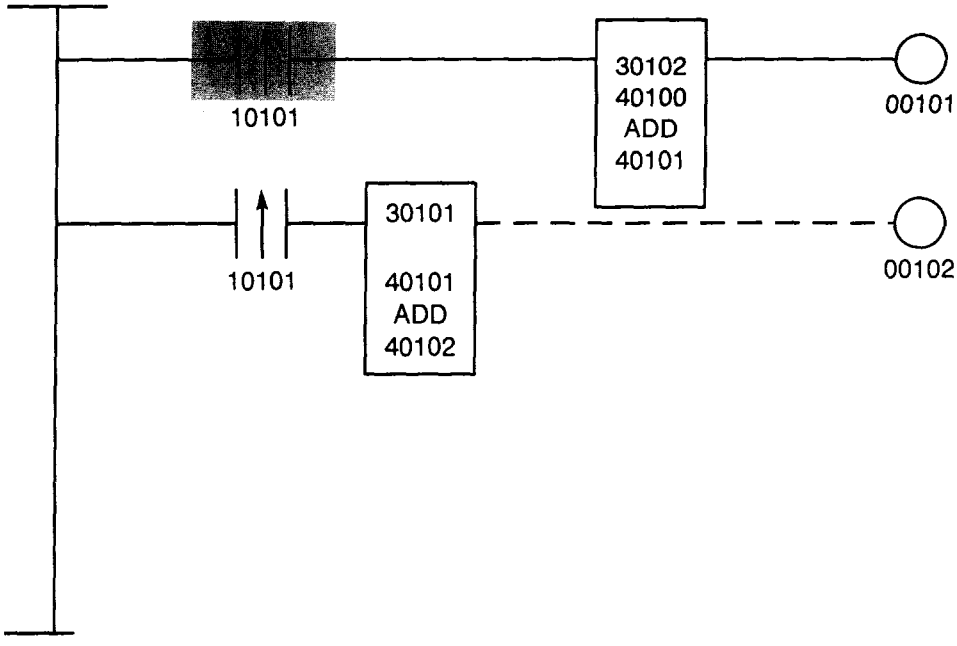
Entering A Program A power rail appears on the left of the P190 screen, the cursor is located at the top left node, and the following software labels are displayed:

RELAYS	COILS	COUNTERS TIMERS	CALCS	DX	SPECIALS	SWEEP FUNCTIONS	

Each software label key, when pressed, brings up another set of software labels and may add to, or alter, the program. To return to the original set of software labels, shown above, from any level of software labels except the PLC status level, press the CHG NODE (Change Node) key on the P190 panel. See the last page of this chapter for keyboard Menu Trees.

Cursor The cursor appears on the screen as a white rectangle. In the figure below, it overlays contacts 10101. Programming action takes place at the node to which the operator has moved the cursor.

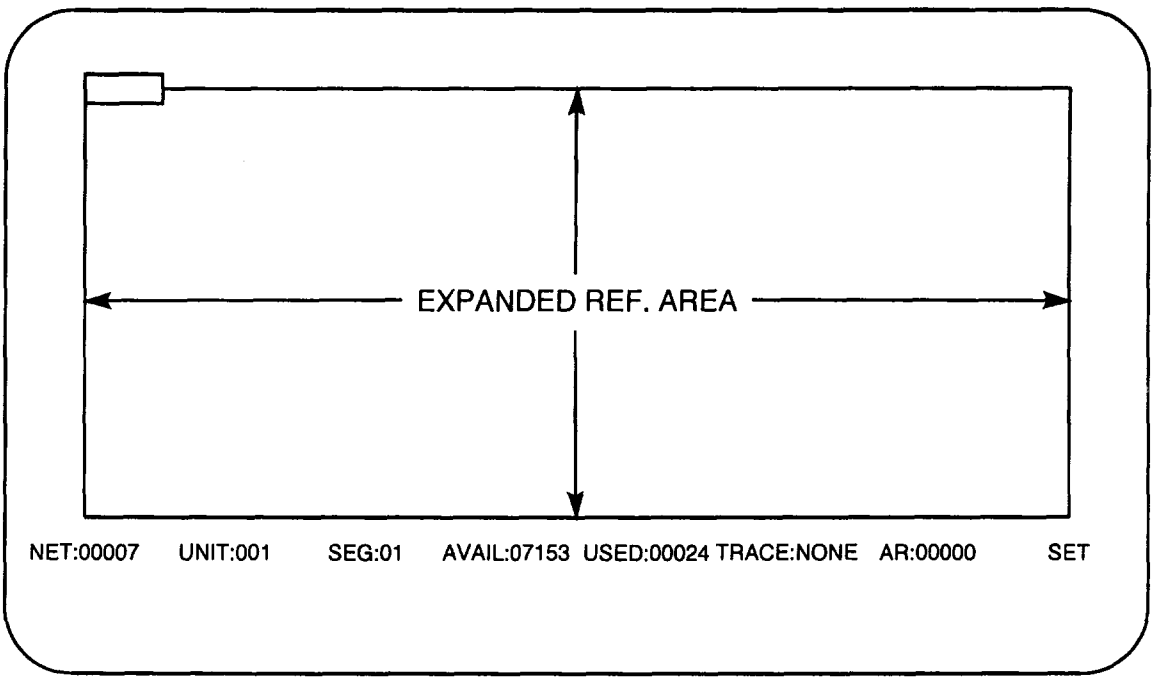
Figure 9-9 Cursor Positioning



Alternate or Reference Screen

The previous figures show the screen that is displayed whenever you are viewing the logic screen, press the P190 CHANGE SCREEN key, and then select the software label key function called "Full Reference".

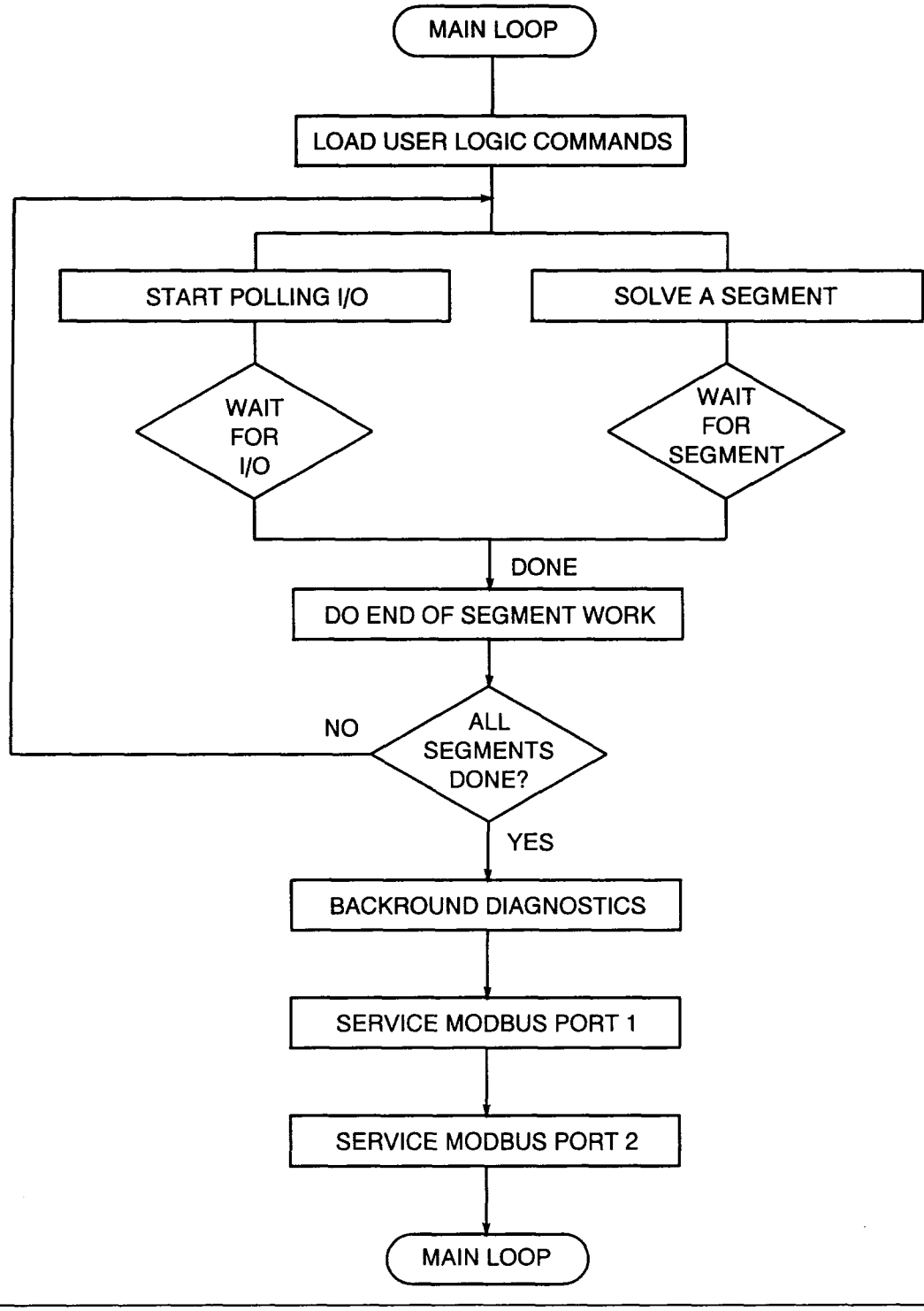
Figure 9-10 Alternate/Reference Screen Example



The reference screen is identical to the logic screen except that the logic area has now replaced by an expanded reference area. From this screen you can select and view the status of up to 51 different references.

Main Loop Sequence

Figure 9-11 Main Loop Sequence Flow Chart



Logic Solution - I/O Polling is a system the 984 uses to provide one logic segment for each drop in the system configuration. Each logic segment may contain as many networks as desired subject to memory constraints, I/O servicing, and a maximum scan time of 200 msecs.

The 984 uses 2 processors; one solves user logic, while the other reads inputs and drives outputs. When the 984 is started, the run light is on and the solution of the user logic and I/0 polling begins. The 984 solves user logic starting with network 1 and proceeds through all networks sequentially, until the last network is solved. Maximum allowable scan times range from .75 - 5ms/k.

The process of reading all inputs, solving all logic, and driving outputs is termed a "scan". When the controller is running it will repeatedly scan until it is stopped.

A scan starts with:

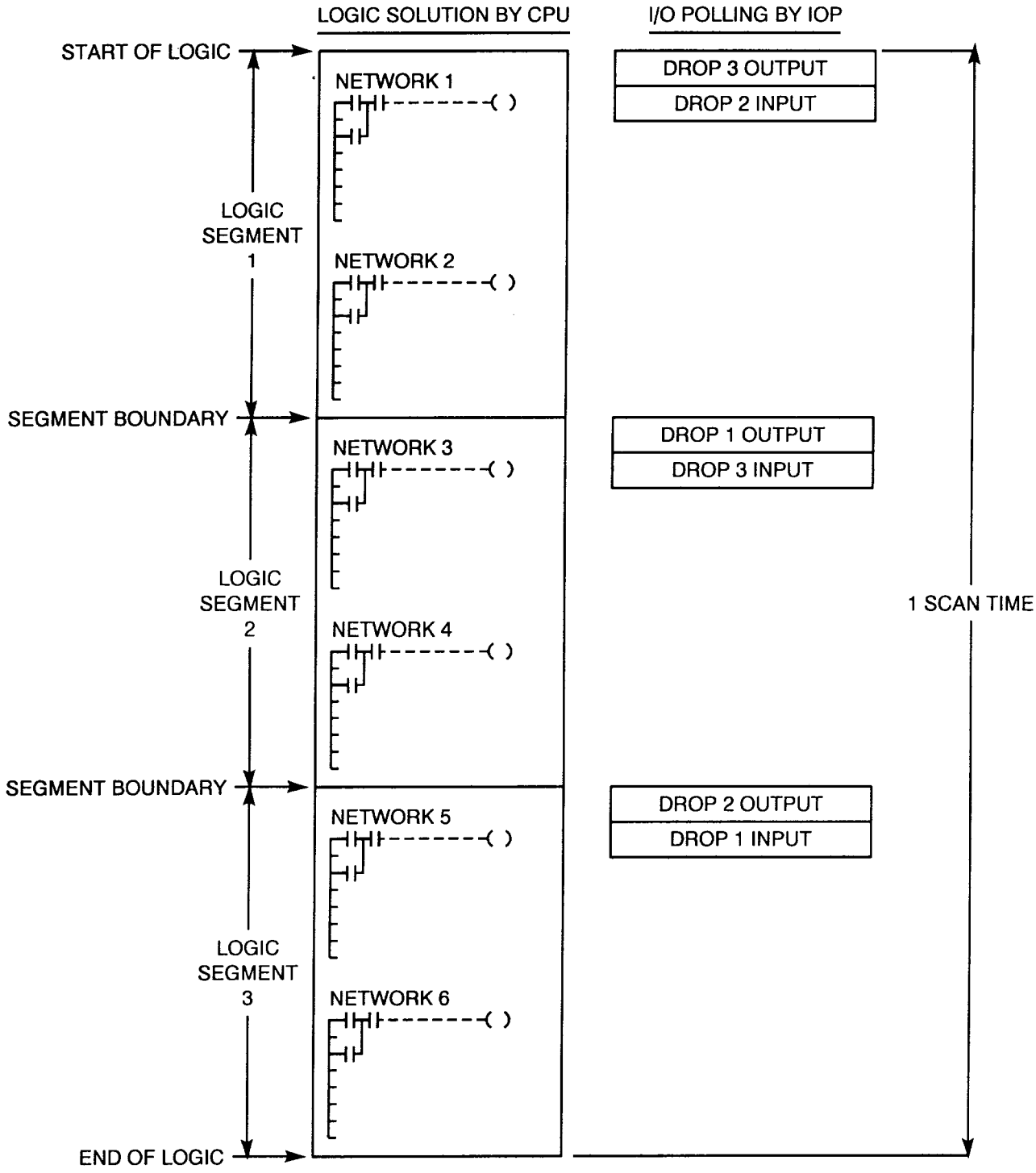
1. The solution of network #1.

2. Driving the outputs for the drop.

Solution of logic within a network starts with the top left node and solves one element at a time, top to bottom, left to right.

Coiis are automatically displayed in the 11th column, however, they are solved based on what node they were programmed in (where the dotted line starts).

Figure 9-12 Logic/Network I/O Polling Example



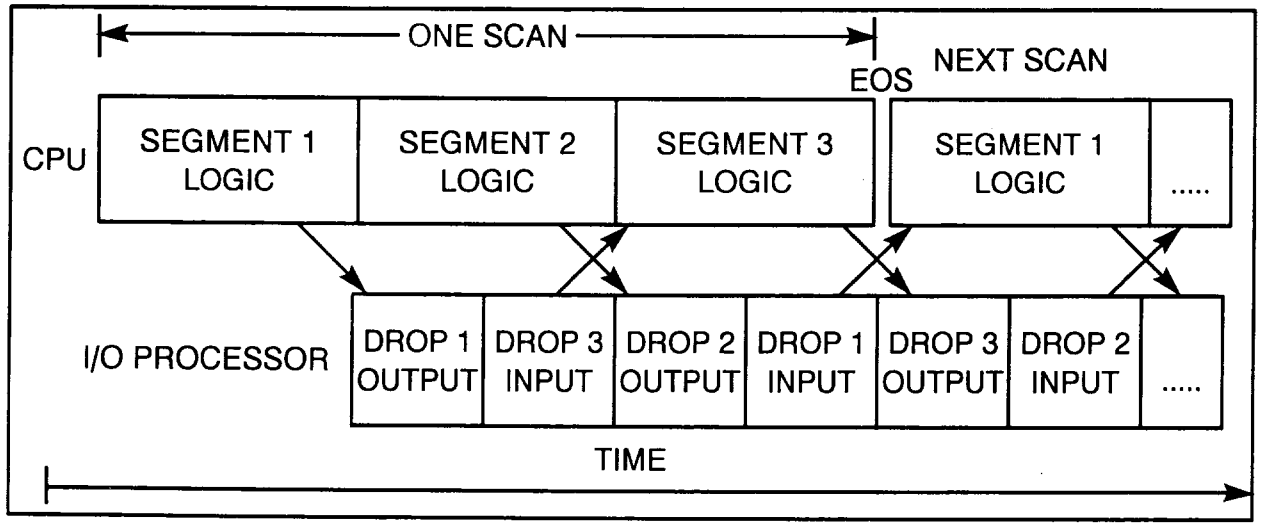
Segmentation A ladder logic program is a collection of segments. As a rule, the number of segments equals the number of I/0 drops being driven by the controller; although in many cases there may be more. A segment is made up of a group of networks. There is no prescribed limit on the number of networks in a segment - the size is limited only by the amount of User Memory available and by the maximum amount of time available for the CPU to scan the logic (200 ms). It is generally advisable to try to distribute your networks evenly through the segments to prevent the program from becoming logic bound or I/O bound.

You can modify the order in which logic is solved with the Segment Scheduler, an editor available with your panel software that allows you to adjust the Order of Solve table in System Memory. You may also create an unscheduled segment that contains one or more ladder logic subroutines, which can be called from the scheduled segments via the JSR funclion.

While the controller's CPU solves each logic segment consecutively (network by network), the I/0 processor simultaneously handles all input signals from and output signals to the I/0 modules in the drop.

The default condition for the order-of-solve table is for each segment in logic to be solved consecutively, one time per scan. See figure below.

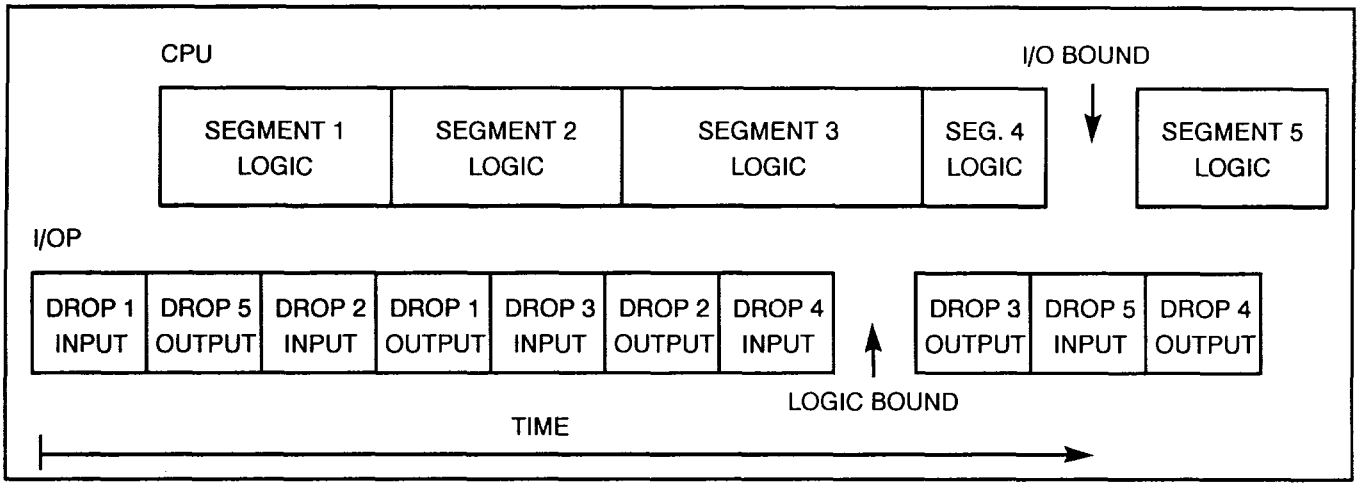
Figure 9-13 Segment Order of Solve



Logic Bound and I/O Bound Programs Scan time can be unnecessarily extended if logic networks are unevenly distributed within the segments.

In the figure below, logic segment 3 contains more networks than the other segments, and logic segment 4 contains fewer networks than the other segments. By evenly distributing the networks among the logic segments, you can prevent your program from becoming logic bound (too much logic relative to I/O activity) or I/O bound (too little logic relative to I/O activity).

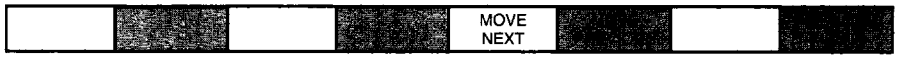
Figure 9-14 Logic Bound and I/O Bound Example



You can use the Move Segment capabilities in the configurator editor to move segment boundries forward or backward to adjust the CPU/IOP balance.

Move Segment The MOVE SEGMENT functions allow whole or partial segments to be moved into adjacent segments. When this software label key is pressed, the following software labels are displayed on the screen:

Figure 9-15 Move Segment Softkey Labels



Move Next Enter a network number into the AR (e.g., 28) and press this software label key. All the networks, starting at the number entered and continuing to the end of the segment, are moved into the next segment.

➤ **NOTE** When moving a portion of a segment to the next segment, remember that only the networks from a specific network to the end of the segment can be moved. A group of networks in the middle of a segment or from the beginning to middle of a segment cannot be moved using this software label key.

Move Previous Enter a network number into the AR (e.g., 35) and press this software label key. All the networks, from the beginning of the segment to the network number entered minus one (e.g., 34), are moved to the previous segment.

➤ **NOTE** When moving a portion of a segment to the previous segment, remember that only the networks from the beginning of a segment to a specific network in the segment can be moved. A group of networks in the middle of a segment or from the middle to the end of a segment cannot be moved using this software label key. Also, the last network in a segment cannot be moved.

Increasing the Number of Segments The following instructions allow you to increase the numbers of segments in a 984 without disturbing user logic. Be certain to have 2 copies of your user logic on tape. If you make an error following these instructions, you will have to remove line and battery power from the 984 and "cold start" it per the instructions given at the beginning of this chapter.

Step 1 Load the utility tape into the P190 and simultaneously press the 2 red buttons, "INIT" and "INIT LOCK"

Step 2 Enter the 984 unit I.D. # into the assembly register

Step 3 Press "ATTACH"

Step 4 Press "EXIT"

Step 5 Press "CONTROLLER OPERATIONS"

Step 6 Press "STOP"

Step 7 Press "PROCEED"

Step 8 Press "PREVIOUS MENU"

Step 9 Press "EXAMINE MEMORY"

Step 10 Type 6A

Step 11 Press "GET"
The number displayed to the right of the = sign is the current number of segments displayed in hex.

Step 12 Enter the desired number of segments into the assembly register.
Remember: 1 segment = 2 Channels (S901) or 1 Drop (S908)

Step 13 Press "ENTER"

Step 14 Press "↓ "

Step 15 Type 6B

Step 16 Press "GET"
The number displayed to the right of the = sign is the "POINTER" to another address.

Step 17 Press "↓ "
You now must find an address derived from the number (pointer) displayed in Step 16. For example, if the current number of segments equal 3, Step 16 would have resulted in the following display:

0006B = B78A Prefix

Note the prefix in the pointer table below ("B" in our example) and find the address to the right of it ("F3XXX" in this example)

Pointer		Address
8XXX		FOXXX
9XXX		F1XXX
AXXX		F2XXX
BXXX	←	F3XXX
CXXX		F4XXX
DXXX		F5XXX
EXXX		F6XXX
FXXX		F7XXX

Step 18 Enter this address into the assembly register and fill in the 3 "X's" with the last 3 digits displayed in Step 16 (in this example "78A"), with a resultant entry = F378A.

Step 19 Press "GET" to see address F378A.

Step 20 Press " ↓ "

Step 21 Enter the number displayed to the right of the "=" sign in Step 19 into the assembly register

Step 22 Press "GET"
The number displayed at this address will be 2800 from the factory.

Step 23 An "END OF SEGMENT NODE" must be added for each segment to be added. Type a number which is one greated than the one displayed (e.g., 2801).

Step 24 Press "ENTER"
Steps 23 and 24 add one "END OF SEGMENT" node if only one segment is to be added, skip to Step 28. To add another "END OF SEGMENT" node go to Step 25.

Step 25 Press "GET NEXT"

Step 26 Repeat Step 23

Step 27 Press "ENTER"
Repeat Steps 25 thru 27 to add the required number of segments.

Step 28 Press "EXIT"

Step 29 Press "CONTROLLER OPERATIONS"

Step 30 Press "START"

Step 31 Press "PROCEED"
The 984 should not start and an error message should show stop state 0004.

Step 32 Press "PREVIOUS MENU"

Step 33 Press "EXAMINE MEMORY"

Step 34 Type "5D"

Step 35 Press "GET"

Step 36 Enter the number displayed to the right of the "=" sign into the assembly register.

Step 37 Press "GET" ("0001" should be displayed)

Step 38 Press the "GET NEXT" key until there is an address with the contents "FFFF", following a three digit number ("0XXX", where X equals the segment number).

Into this address, key the hexadecimal NUMBER (see table) of the next segment to be added, press "ENTER".

Segment	Hex Value	Segment	Hex Value	Segment	Hex Value
2	0002	7	0007	12	000C
3	0003	8	0008	13	000D
4	0004	9	0009	14	000E
5	0005	10	000A	15	000F
6	0006	11	000B	16	0010

Step 39 To add additional segments, press "GET NEXT", key in the hexadecimal segment number, and press "ENTER".

Step 40 When all new segment numbers have been entered, press "GET NEXT" to access the next address.

Step 41 To mark the end of segment numbers, key in the hexadecimal number FFFF and press "ENTER".

Step 42 Make a new copy of your Program using the TAPE LOADER TAPE (Do not use one of the original user logic tapes, in the event an error has been made).

Step 43 Using the UTILITY TAPE (or IBM Ladder Lister Software) with the EXAMINE MEMORY function, place "006A" into the AR, then press "GET". The number displayed is the present number of segments.

Step 44 Enter the OLD (original) number of segments into the AR, Press "ENTER".

Step 45 Using the CONFIGURATOR TAPE, select "CONFIG". (for your specific controller) and change the number of segments and drops to the desired amount.

➤ **NOTE** If you have any loadable modules, you will have to reload them. Then Press "Write Config".

Step 46 Start the Controller, and verify that the RUN LIGHT comes on.

Step 47 Stop the Controller.

Step 48 Using the TAPE LOADER TAPE, select the "RELOCATE LOGIC" function, and load the tape you made in Step 42.

Step 49 Re-Traffic Cop the controller.

Step 50 Start the controller.

Background Diagnostics

Performs one of the following per scan:

Checksum executive proms (16 words per test).

Write/read memory test pattern (word each test to user logic and state RAM) 3 patterns each test.

Hardware Logic Solver (HLS). 1 network with relays, coil, vertical and horizontal shorts.

Loopback/user logic checksum (4 words, 3 patterns) Checksum user logic while waiting.

Checksum loadable software module area (16 words per test).

Checksum extended memory parity checker with extended memory.

Hints to Minimize Scan Time

Only configure the number of I/O drops that will be required. This will limit the number of program segments being solved.

Distribute user logic evenly among the number of program segments configured in the system. Since one drop of I/O is serviced while a segment of logic is being solved, there is a fair amount of logic that can be solved without an impact on scan time.

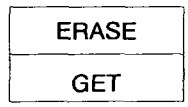
Don't solve complex logic every scan if, say, 5 times per second will do just as well.

Use the skip functions to bypass conditionally unnecessary logic. Remember that skip 0 is fastest of all. Skip 0 stops at the end of the segment.

Small systems are usually I/0 bound, medium and large systems are almost always logic bound. Focus your attention accordingly.

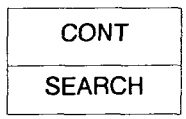
P190 Keyboard Functions

← ↑ ↓ →	Moves the cursor one position in the direction of the arrow. The cursor will wrap around the screen in all 4 directions.
ENTER	Copies the value in the assembly register to the cursor position.
NTWK DELETE NODE	Shifted: Deletes the network under the cursor. Unshifted: Deletes the node under the cursor.
START NEXT	Starts a new network immediately after the one currently displayed. The networks displaced are renumbered upwards by 1.
CHG SCREEN	Toggles the display between the "network/partial reference" screen and the "full reference" screen.
PRINT	Dumps the display currently on screen to a printer connected to port two.
CHG NODE	Displays programmer tape main program element menu.
RESET / EXIT	Shifted: Accesses reset level of tape. Unshifted: Accesses exit level of tape.

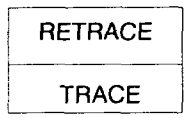


Shifted: Clears network or reference under the cursor from screen.
Unshifted: Displays network or reference specified in the assembly register (AR).

Shifted: Displays previous network if cursor in network area.
Displays previous reference if cursor in reference area.
Unshifted: Displays next network if cursor in network area.
Displays next reference if cursor in reference area.



Shifted: Continues to search for parameters entered in search area.
Unshifted: Searches for parameters entered in Search Area.

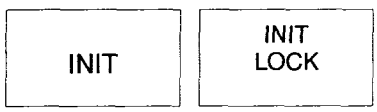


Shifted: Returns to network with original 0XXXX contact.
Unshifted: With cursor over 0XXXX contact, displays network which has coil which drives 0XXXX contact. With the cursor over any other reference in the network area, the reference is displayed in the reference area.

A typical numeric key--
Shifted: Numeric entries (A-F hexadecimal).
Unshifted: Numeric entries (0-9)

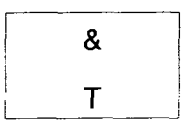
Erases the error message currently displayed.

Places zeroes in the assembly register (AR). Also clears error messages.



Working tapes are loaded into the P190 memory when these keys are pressed simultaneously.

Provides upper function of dual action keys. Shift must be pressed at the same time the function key is pressed.



Typical alphabetic key-- used to generate tape titles, ladder listing headers, and ASCII messages.

Discrete Traffic Cop Numbers

Each discrete module has 16 inputs or outputs, but only one address may be specified per slot -- the first of 16. See table below.

Table 9-1 First of 16 Reference Numbers for Coils and Outputs

1	1025	2049	3073	4097	5121	6145	7169
17	1041	2065	3089	4113	5137	6161	7185
33	1057	2081	3105	4129	5153	6177	7201
49	1073	2097	3121	4145	5169	6193	7217
65	1089	2113	3137	4161	5185	6209	7233
81	1105	2129	3153	4177	5201	6225	7249
97	1121	2145	3169	4193	5217	6241	7265
113	1137	2161	3185	4209	5233	6257	7281
129	1153	2177	3201	4225	5249	6273	7297
145	1169	2193	3217	4241	5265	6289	7313
161	1185	2209	3233	4257	5281	6305	7329
177	1201	2225	3249	4273	5297	6321	7345
193	1217	2241	3265	4289	5313	6337	7361
209	1233	2257	3281	4305	5329	6353	7377
225	1249	2273	3297	4321	5345	6369	7393
241	1265	2289	3313	4337	5361	6385	7409
257	1281	2305	3329	4353	5377	6401	7425
273	1297	2321	3345	4369	5393	6417	7441
289	1313	2337	3361	4385	5409	6433	7457
305	1329	2353	3377	4401	5425	6449	7473
321	1345	2369	3393	4417	5441	6465	7489
337	1361	2385	3409	4433	5457	6481	7505
353	1377	2401	3425	4449	5473	6497	7521
369	1393	2417	3441	4465	5489	6513	7537
385	1409	2433	3457	4481	5505	6529	7553
401	1425	2449	3473	4497	5521	6545	7569
417	1441	2465	3489	4513	5537	6561	7585
433	1457	2481	3505	4529	5553	6577	7601
449	1473	2497	3521	4545	5569	6593	7617
465	1489	2513	3537	4561	5585	6609	7633
481	1505	2529	3553	4577	5601	6625	7649
497	1521	2545	3569	4593	5617	6641	7665
513	1537	2561	3585	4609	5633	6657	7681
529	1553	2577	3601	4625	5649	6673	7697
545	1569	2593	3617	4641	5665	6689	7713
561	1585	2609	3633	4657	5681	6705	7729
577	1601	2625	3649	4673	5697	6721	7745
593	1617	2641	3665	4689	5713	6737	7761
609	1633	2657	3681	4705	5729	6753	7777
625	1649	2673	3697	4721	5745	6769	7793
641	1665	2689	3713	4737	5761	6785	7809
657	1681	2705	3729	4753	5777	6801	7825
673	1697	2721	3745	4769	5793	6817	7841
689	1713	2737	3761	4785	5809	6833	7857
705	1729	2753	3777	4801	5825	6849	7873
721	1745	2769	3793	4817	5841	6865	7889
737	1761	2785	3809	4833	5857	6881	7905
753	1777	2801	3825	4849	5873	6897	7921
769	1793	2817	3841	4865	5889	6913	7937
785	1809	2833	3857	4881	5905	6929	7953
801	1825	2849	3873	4897	5921	6945	7969
817	1841	2865	3889	4913	5937	6961	7985
833	1857	2881	3905	4929	5953	6977	8001
849	1873	2897	3921	4945	5969	6993	8017
865	1889	2913	3937	4961	5985	7009	8033
881	1905	2929	3953	4977	6001	7025	8049
897	1921	2945	3969	4993	6017	7041	8065
913	1937	2961	3985	5009	6033	7057	8081
929	1953	2977	4001	5025	6049	7073	8097
945	1969	2993	4017	5041	6065	7089	8113
961	1985	3009	4033	5057	6081	7105	8129
977	2001	3025	4049	5073	6097	7121	8145
993	2017	3041	4065	5089	6113	7137	8161
1009	2033	3057	4081	5105	6129	7153	8177

No number larger than specified under the "SET SIZE" menu during configuration may be used as an address for either registers or discretes.

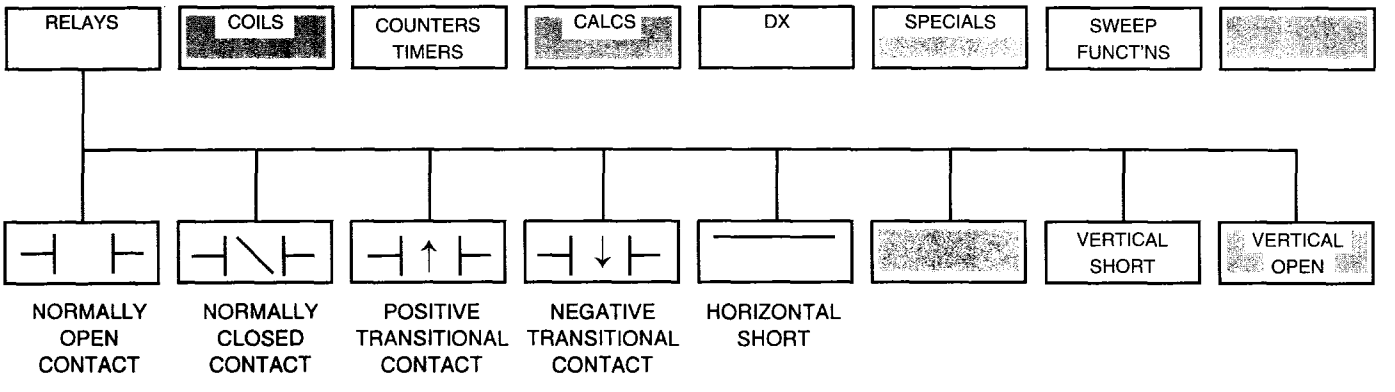
Relay Contacts

The relay contact is the basic programming element. it can be referenced to a logic coil (0XXXX) or a discrete input (1XXXX). There are 4 types of relay contacts:

normally open
normally closed
positive transitional
negative transitional

These contacts, and horizontal and vertical shorts are programmed under the relay portion of the main programming menu as shown below.

Figure 9-16 Programmer Tape Relay Menu

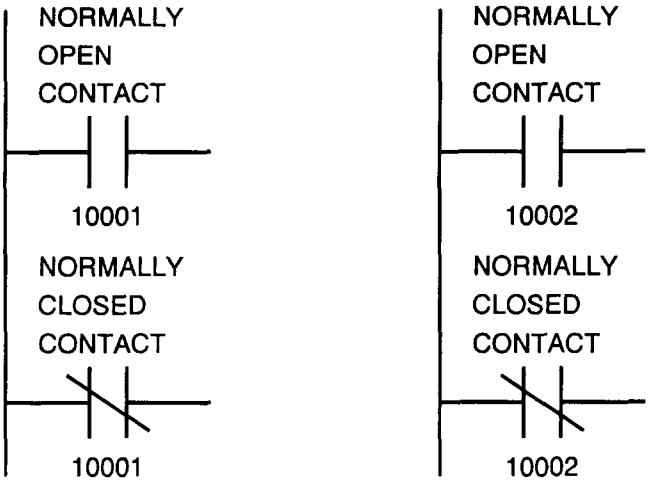


Normally Open Contact
It may be referenced to coils (0XXXX) or inputs (1XXXX).
It passes power when its referenced coil or input is ON.

Normally Closed Contact
It may be referenced to coils (0 XXXX) or inputs (1XXXX).
It passes power when its referenced coil or input is OFF.

Programming A relay contact is inserted into a program by positioning the cursor over the desired location, entering a 0XXXX or 1XXXX reference into the Assembly Register (AR) and pressing the appropriate software label key. To change the type of relay contact, position the cursor over the contact to be changed and press the desired software label key. To change the reference numbers below a contact, position the cursor over the contact, enter a new value into the AR, and press the ENTER key.

Figure 9-17 Relay Contacts Network Example



Network Description In the above circuit assume that 10001 and 10002 are momentary pushbuttons. 10001 is field wired normally open, and 10002 is field wired normally closed. The following table details network power flow for these two switches.

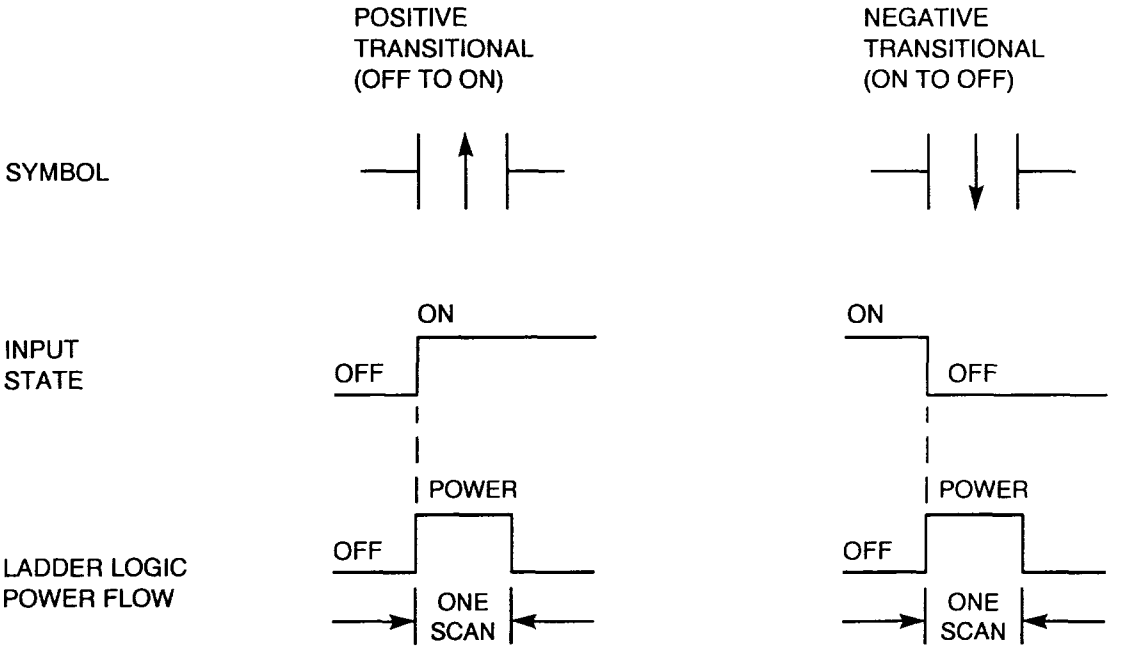
Table 9-2 Field Wiring Determination of Ladder Logic Power Flow

REFERENCE NUMBER	FIELD WIRED CONTACT	PROGRAMMED CONTACT	PUSHBUTTON RELEASED	PUSHBUTTON DEPRESSED
10001	NORMALLY OPEN	NORMALLY OPEN NORMALLY CLOSED	PASSES POWER	PASSES POWER
10002	NORMALLY CLOSED	NORMALLY OPEN NORMALLY CLOSED	PASSES POWER	PASSES POWER

Transitional Contacts

Transitional contacts, like "NO" and "NC" contacts, are controlled by either inputs (1XXXX) or coils (0XXXX) via the reference number system, however, transitional contacts pass power for only the one scan after which their controlling coil or input changes state. The figure below explains positive and negative transitional contact.

Figure 9-18 Transitional Contacts



Vertical and Horizontal Shorts

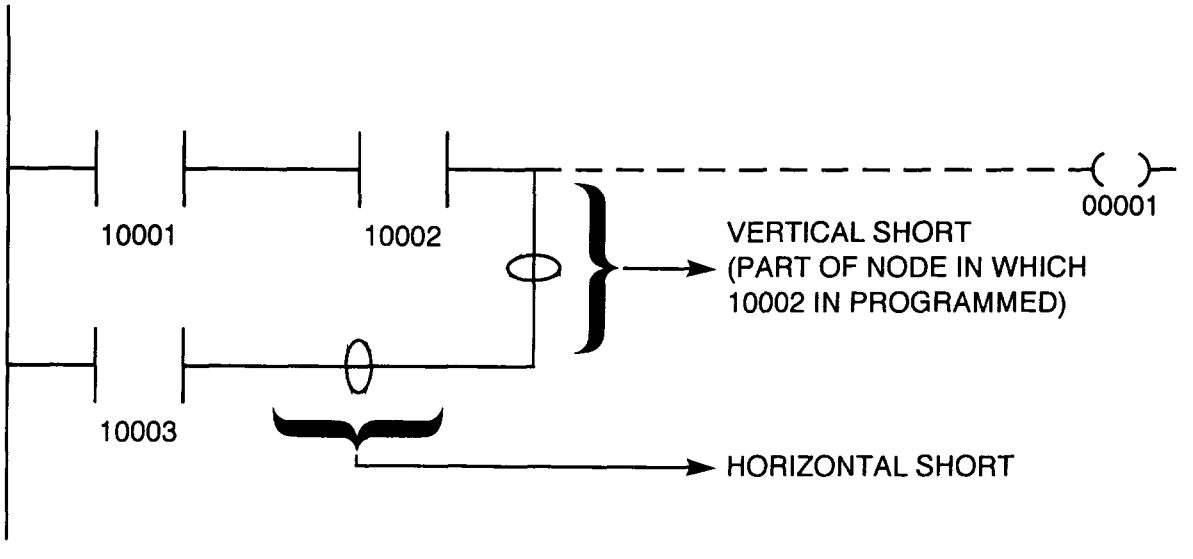
Vertical and horizontal shorts are simply straight line connections between contacts. Vertical shorts are used to connect contacts and function blocks one above the other in a network. Vertical shorts can also be used to connect inputs or outputs in a function block to create either/or conditions. When two contacts are connected by vertical shorts, a vertical short on each side, power is allowed to pass through if either, or both, contacts receive power.

Enter a vertical short into a program by positioning the cursor over the node to the left of and above the location desired for the short and press the VERTICAL SHORT software label key. A vertical short is cleared by pressing the VERTICAL OPEN software label key. A vertical short does not take any user memory.

Horizontal shorts are used in combination with vertical shorts to expand logic within a network without breaking the power flow. They can be used to create either/or conditions using basic relay contacts. For example, if one line of logic contains two relay contacts, and the line below it only contains one contact, a horizontal short is placed beside the single contact. See figure below. A vertical short is used to connect the horizontal short to the top logic line. Power passes through to energize the coil if the two top contacts are energized or if the bottom contact is energized.

Enter a horizontal short into a program by positioning the cursor over the node desired for the short and pressing the " — " (Horizontal Short) software label key. To clear a horizontal short, press the DELETE NODE software label key. A horizontal short uses two words of memory.

Figure 9-19 Horizontal/Vertical Short Network Example



Coils

A coil is used to activate logic within the user's program and/or to control an output circuit. Coils are either on or off as a result of power flow within the user ladder logic program. When a coil is powered on, it may pass power to a discrete output and/or cause relay contacts that it controls thru the reference number system to change state. It is represented by a 0XXXX reference number and either of two symbols.

A Normal Coil, -◯- is turned OFF if controller power is lost.

A Latched Coil, -Ⓛ- retains its previous state (ON or OFF) for the first scan after power is restored to the controller.

Coils are displayed in the far right (eleventh) column of a network. Each network can contain a maximum of seven coils.

Each 0XXXX reference can be used as a coil only once, but can be referenced to any number of relay contacts. Output coils are generally given the lower 0XXXX reference numbers and internal coils are given the higher 0XXXX reference numbers.

When a logic coil is inserted into a program. The cursor can be directly beside the last logic element in a row. When the coil software label key is pressed, dashed lines are inserted and the coil is placed in the eleventh column. The only reference allowed is a 0XXXX reference, unlike relay contacts which allow 0XXXX or 1XXXX references.

Disable/Enable Any logic coil or discrete input in the user's program can be disabled and forced ON or OFF. To disable a coil or discrete input, do the following:

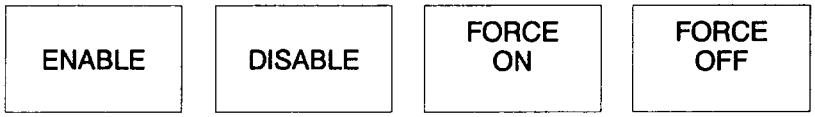
Step 1 Ensure that Memory Protect is OFF on both the P190 Programmer and the 984 PLC.

Step 2 Position the cursor at the bottom of the P190 screen.

Step 3 Enter the coil reference number or discrete input reference number into the Assembly Register (AR).

Step 4 Press the ERASE/GET key.

The reference number appears at the cursor position with its state, ON or OFF, and the following software labels appear on the screen:



Step 5 Move the cursor onto the coil in the network and press the DISABLE software label key. The coil or input is now DISABLED ON or DISABLED OFF, depending on its state when the DISABLE key was pressed.

Force ON/OFF After being disabled, a discrete may be "FORCED" ON or OFF. Press the FORCE ON or FORCE OFF software label keys to change the state of the coil or input. Press the ENABLE software label key to enable the coil or input. If the coil or input is enabled, it cannot be forced ON or forced OFF; it returns to the state it had when the ENABLE software label key was pressed and is logic controlled.

Disabling a coil or input causes the programmed logic to bypass that coil or input. The coil's state or the input's state is now controlled by the user through the FORCE ON and FORCE OFF software label keys. Memory Protect must be OFF.

➤ ➤ **WARNING There are exceptions to the logic bypassing a disabled coil. All the DX function blocks which allow a coil in the destination node override the coil even if it is disabled.**

Duplication of Programmed Output Reference

If the middle node entry for a data transfer function is an output reference number (0XXXX), it counts as the one time the referenced coils may be used.

This applies to:

Block Move (BLKM)

First Out (FOUT)

Complement (COMP)

Logical And (AND)

Logical Or (OR)

Logical Exclusive Or (XOR)

Bit Modify (MBIT)

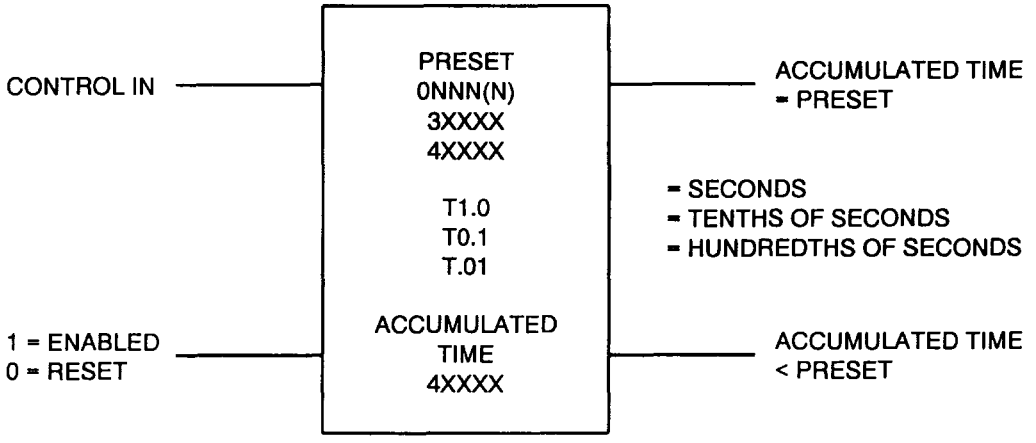
Bit Sense (SENS)

Bit Rotate (BROT)

Timers

This function is used to time an event or create a delay. Time can be measured in seconds (T1.0), tenths of seconds (T0.1), or hundredths of seconds (T.01).

Figure 9-20 Timer Function Block



Function Block The top node is the timer preset, and can be one of the following: A decimal number ranging from 1 to 999 in 16 bit controllers or 1 to 9999 in 24 bit controllers, an input register (3XXXX), or a holding register (4XXXX). The bottom node contains the symbol T1.0 or T0.1 or T.01 and a holding register (4XXXX) which stores the accumulaled time. This register value is clamped at the preset value.

Inputs
Top: Control in. Time accummulates when this input is powered and the bottom input is powered. Bottom: When powered, enables the timer, when not powered, timer is reset to zero.

Outputs
Top: Passes power when the register accumulating time equals the preset value. Bottom: Passes power when the register accumulating time does not equal the preset value.

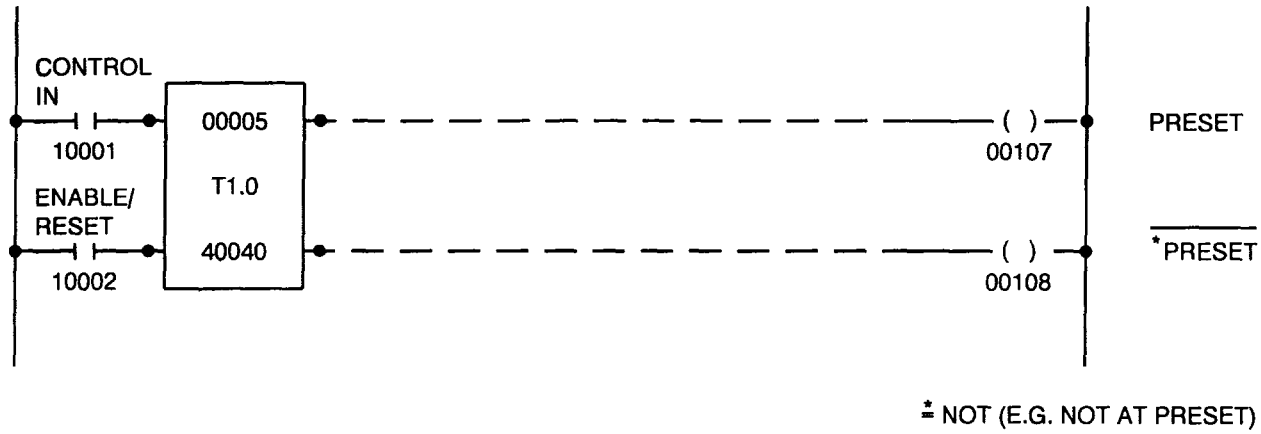
➤ **CAUTION Cascading timers with presets of one (1) causes these timers to time-out together. To avoid this problem, change presets to 10 and substitute a tenth of a second timer (T0.1) for a one second timer (T1.0) or a hundredth of a second timer (T.01) for a tenth of a second timer (T0.1).**

Network Description In the figure below, assume that 10002 is closed (timer enabled) and that the value contained in register 40040 equals zero. Since 40040 does not equal the preset of 5 entered in the top node, coil 00107 will be off, and coil 00108 will be on.

When 10001 is closed, 40040 will begin to accumulate counts at one second intervals until 40040 equals 5. At that point 00107 will be on, and 00108 will be off.

When 10002 is opened 40040 will be reset to zero, coil 00107 will be off, and 00108 will be on.

Figure 9-21 Timer Network Example

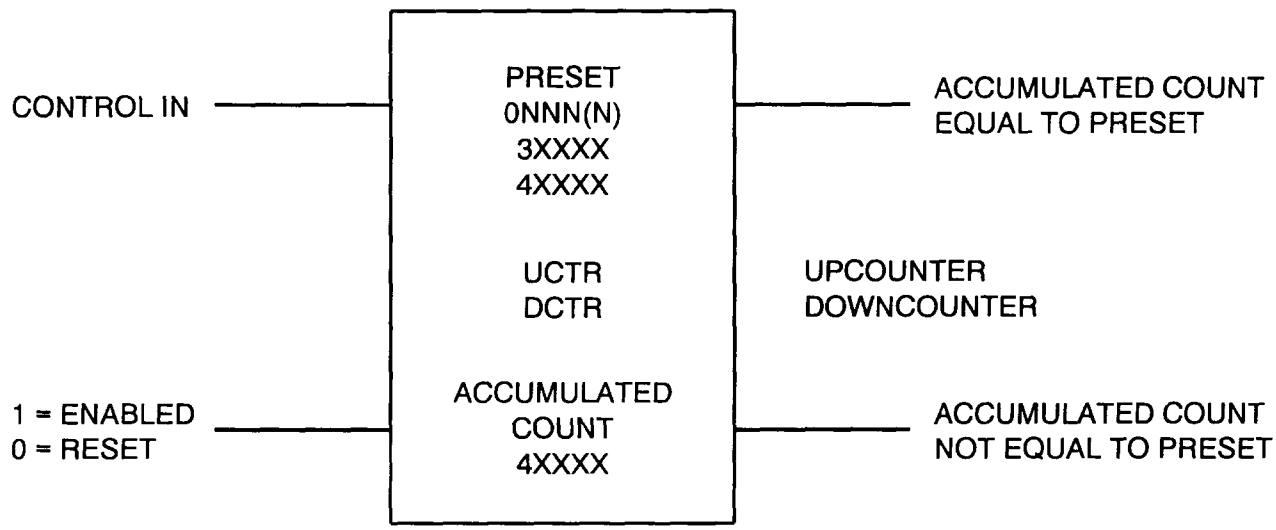


➤ **NOTE** If the accumulated time does not equal the preset, the bottom output will pass power even though no inputs to the block are present.

Counters

The up counter (UCTR) and down counter (DCTR) functions both count when the control input transitions from off to on. The up counter (UCTR) counts up from zero to a preset number, and the down counter (DCTR) counts down from the preset to zero.

Figure 9-22 Counter Function Block



Function Block The top node is the counter preset, and can be one of the following: a decimal number ranging from 1 to 999 in 16 bit controllers and 1 to 9999 in 24 bit controllers, an input register (3XXXX), or a holding register (4XXXX).

The bottom node contains the symbol UCTR or DCTR and a holding register (4XXXX) which stores the accumulated count. This register value is clamped at the preset value.

Inputs
Top: Control in
Bottom: When powered, enables the counter. When not powered, counter is reset to zero.

Outputs
Top: Passes power when the register accumulating counts equals the preset value. Preset value equals zero for DCTR.
Bottom: Passes power when the register accumulating count does not equal the preset value.

Figure 9-23 Up Counter Network Example

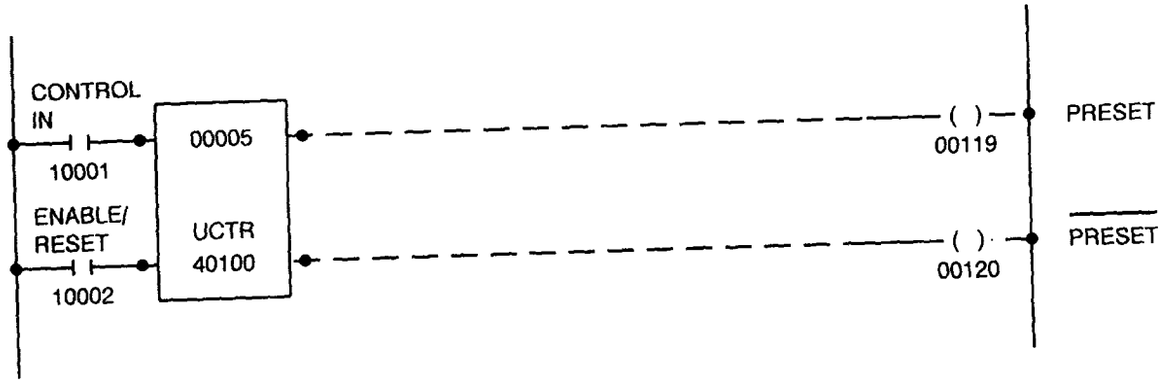
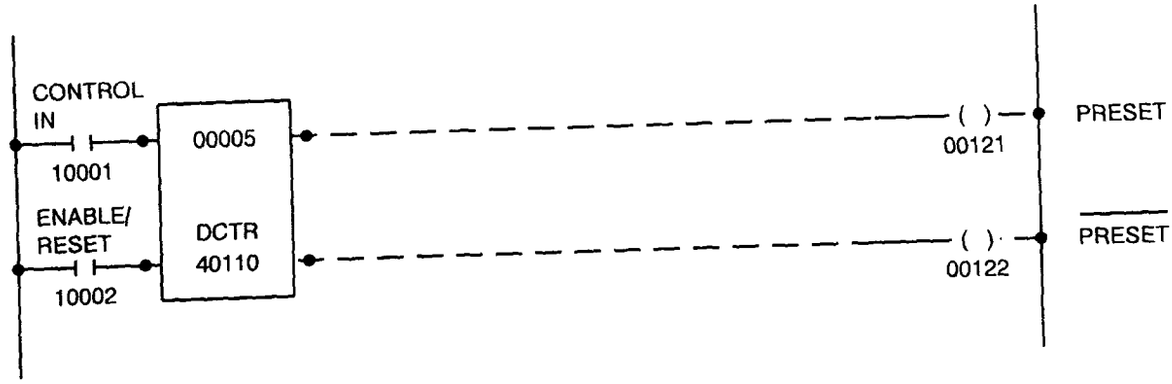


Figure 9-24 Down Counter Network Example



Network Description In the Up Counter example above, assume that 10002 is closed (counter enabled), and that the value contained in register 40100 = zero. Since 40100 does not equal the preset of 5 entered in the top node, coil 00119 will be off, and coil 00120 will be on.

First transition of 10001 from off to on will increment 40100 to 1
Second transition of 10001 from off to on will increment 40100 to 2
Third transition of 10001 from off to on will increment 40100 to 3
Fourth transition of 10001 from off to on will increment 40100 to 4
Fifth transition of 10001 from off to on will increment 40100 to 5, Coil 00119 will be on, and coil 00120 will be off.

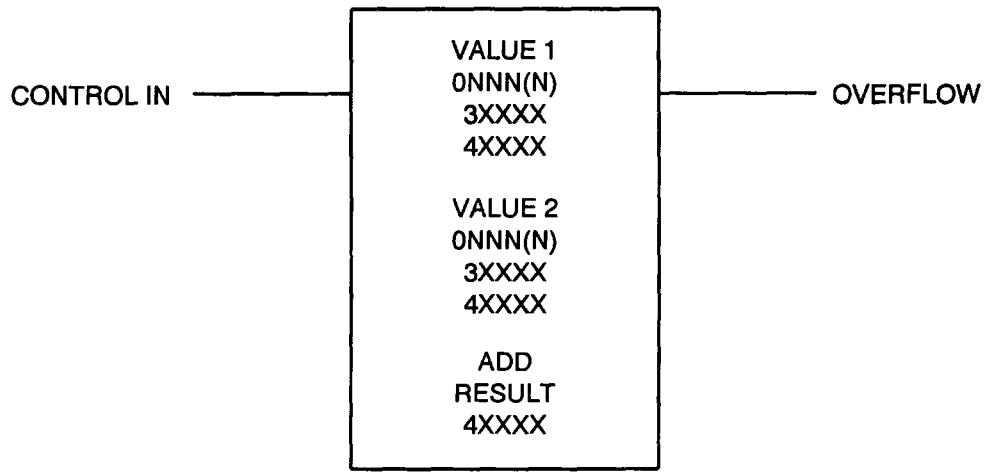
➤ **NOTE** When 10002 is opened 40100 will be reset to zero, coil 00119 will be off, and 00120 will be on. If the accumulated time does not equal the preset, the bottom output will pass power even though no inputs to the block are present. The down counter shown above operates identically to the up counter except that the count in register 40100 will decrement from the preset to zero.

Calculates

When selected from the main programming menu, the CALCS software label key provides the ADD, SUB, MUL, and DIV block functions.

ADDITION This function adds value 1 to value 2 and stores the result in a holding register.

Figure 9-25 Addition Function Block



Function Block The top node is value 1 and can be one of the following:
A decimal number ranging from 1 to 999 in 16 bit CPU's or 1 to 9999 in 24 bit CPU's
An input register (3XXXX)
A holding register (4XXXX)

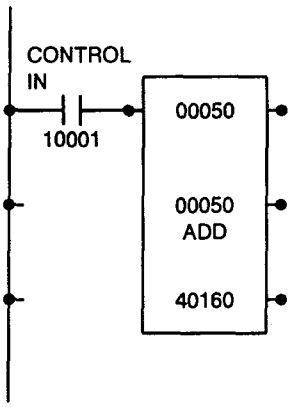
The middle node is value 2 and can be one of the following:
A decimal number ranging from 1 to 999 in 16 bit CPU's or 1 to 9999 in 24 bit CPU's
An input register (3XXXX)
A holding register (4XXXX)

The bottom node contains the symbol ADD and a holding register (4XXXX) where the result of the addition will be stored.

Inputs
Top: Control in. Every scan this node is powered, the addition is performed. A transitional contact should be used if single operations are desired.

Outputs
Top: Passes power when the result is (greater than) 9999.

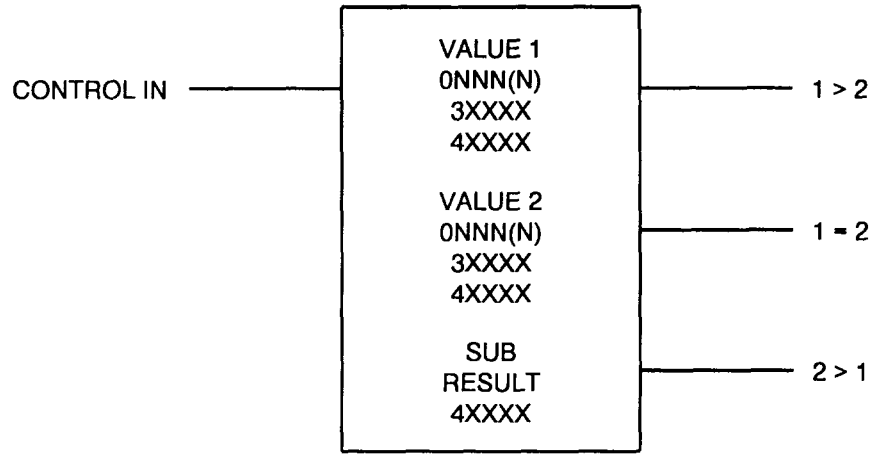
Figure 9-26 ADD Block Network Example



Network Description In the figure above, when 10001 is closed value 1 (50) will be added to value 2 (50) and the result (100) will be stored in register 40160.

Subtraction This function performs an absolute subtract, without sign, of values 1 and 2, and stores the result in a holding register. The block also functions as a comparator identifying whether value 1 is > (greater than), < (less than), or equal to value 2.

Figure 9-27 Subtraction Function Block



Function Block The top node is value 1 and can be one of the following:
A decimal number ranging from 1 to 999 in 16 bit CPU's or 1 to 9999 in 24 bit CPU's
An input register (3XXXX)
A holding register (4XXXX)

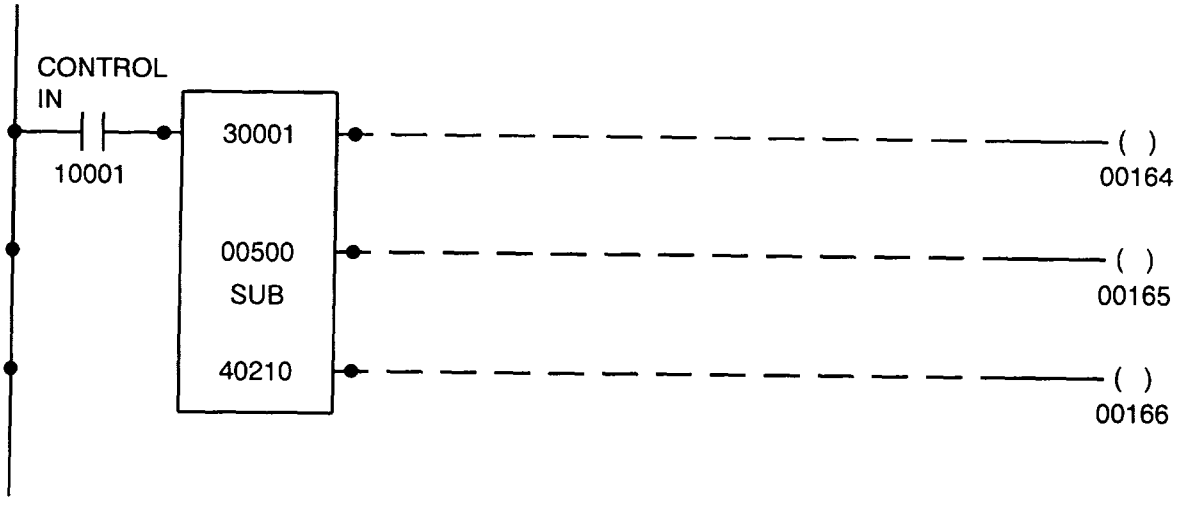
The middle node is value 2 and can be one of the following:
A decimal number ranging from 1 to 999 in 16 bit CPU's or 1 to 9999 in 24 bit CPU's
An input register (3XXXX)
A holding register (4XXXX)

The bottom node contains the symbol SUB and a holding register (4XXXX) where the result of the subtraction will be stored.

Inputs
Top: Control in. Every scan this node is powered, the subtraction is performed. A transitional contact should be used if single operations are desired.

Outputs
Top: Passes power when the top input is powered and value 1 is > value 2.
Middle: Passes power when the top is powered and value 1 = value 2.
Bottom: Passes power when the top input is powered and value 2 is > value 1.

Figure 9-28 SUB Block Network Example



Network Description In the figure above, when 10001 is closed value 2 (500) will be subtracted from value 1 (input register 30001). The result will be stored in register 40210.

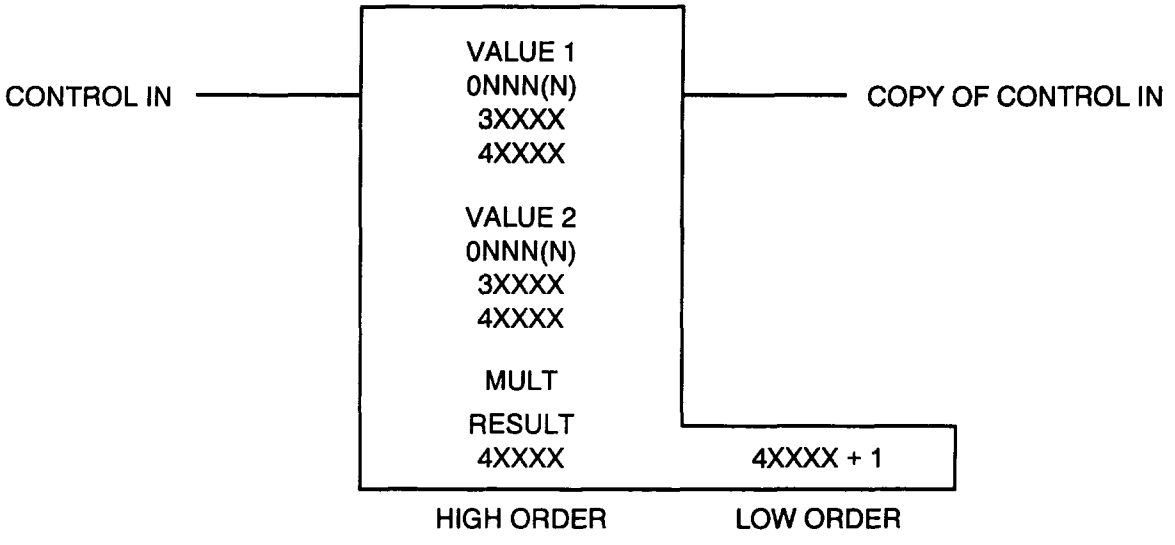
If 10001 is closed and the value of 30001 is > 500, then the coil connected to the top output will be on.

If 10001 is closed and the value of 30001 is equal to 500, then the coil connected to the middle output will be on.

If 10001 is closed and the value of 30001 is < 500, then the coil connected to the bottom output will be on.

Multiplication This function multiplies value 1 by value 2 and stores the result in a holding register.

Figure 9-29 Multiplication Function Block



Function Block The top node is value 1 and can be one of the following:
A decimal number ranging from 1 to 999 in 16 bit CPU's or 1 to 9999 in 24 bit CPU's
In input register (3XXXX)
A holding register (4XXXX)

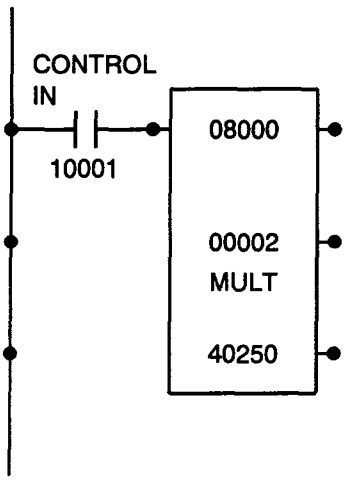
The middle node is value 2 and can be one of the following:
A decimal number ranging from 1 to 999 in 16 bit CPU's or 1 to 9999 in 24 bit CPU's
An input register (3XXXX)
A holding register (4XXXX)

The bottom node contains the symbol MUL and a holding register (4XXXX). The result of the multiplication is stored in two consecutive registers. The high order digits are stored in the register specified in the bottom node, and the low order digits are stored in the next sequential register (4XXXX+1).

Inputs
Top: Control in. Every scan this node is powered, the multiplication is performed. A transitional contact should be used if single operations are desired.

Outputs
Top: Passes power when the top input is powered.

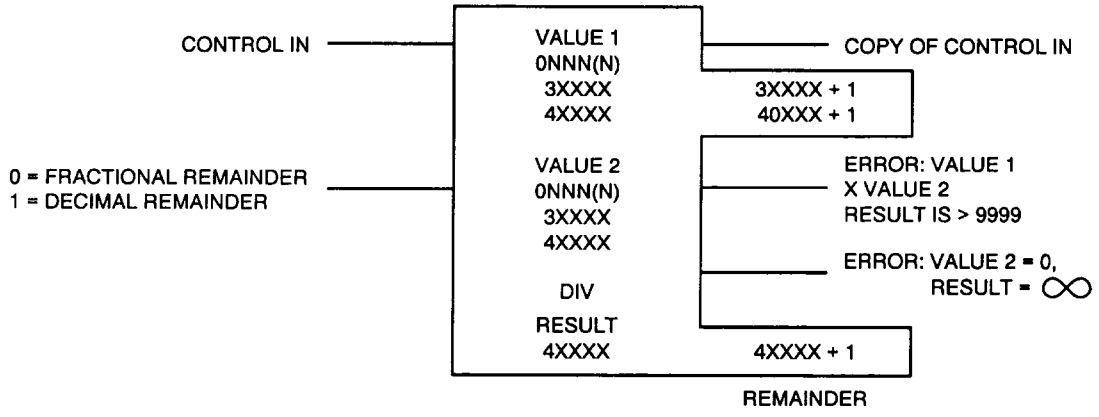
Figure 9-30 MUL Block Network Example



Network Description When 10001 is closed value 1 (8000) will be multiplied by value 2 (2). The result (16,000) will be stored in two sequential registers; in this example 40250 will contain the high order digits (0001) and 40251 will contain the low order digits (6000).

Division This function divides value 1 by value 2 and stores the result and remainder in two consecutive holding (4XXXX) registers.

Figure 9-31 Division Function Block



Function Block The top node is value 1 and can be one of the following:
A decimal number ranging from 1 to 999 in 16 bit CPU's or 1 to 9999 in 24 bit CPU's
Two input registers (3XXXX = high order digits, 3XXXX + 1 = low order digits)
Two holding registers (4XXXX = high order digits, 4XXXX + 1 = low order digits)

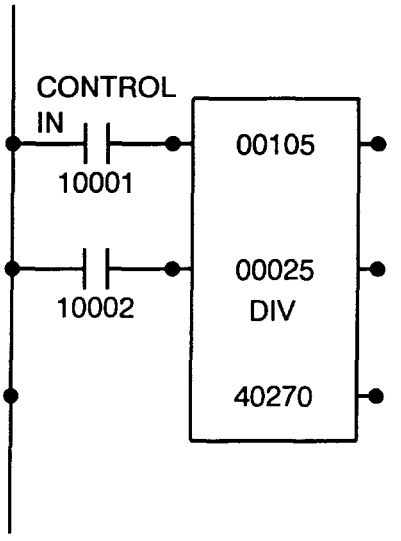
The middle node is value 2 and can be one of the following:
A decimal number ranging from 1 to 999 in 16 bit CPU's or 1 to 9999 in 24 bit CPU's
An input register (3XXXX)
A holding register (4XXXX)

The bottom node contains the symbol DIV and a holding register (4XXXX). The result of the division is in the register specified in the bottom node, and the remainder is stored in the next sequential register (4XXXX + 1).

Inputs
Top: Control in. Every scan this node is powered, the division is performed. A transitional contact should be used if single operations are desired.
Middle: When not powered, the remainder will be a whole number. When powered, the remainder will be a decimal fraction.

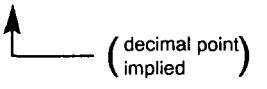
Outputs
Top: Passes power when the division is successful.
Middle: Passes power when the result is 9999.
Bottom: Passes power when value 2 = 0, (result = infinity.)

Figure 9-32 DIV Block Network Example



Network Description When 10001 is closed value 1 (105) will be divided by value 2 (25). The result (4 with a remainder of 2) will be stored in two sequential registers. 40270 will contain the result, and 40271 will contain the remainder.

If 10002 is open, the remainder will be a whole number (5 in this example). If 10002 is closed, the remainder will be decimal (0.2 in this example), displayed in register 40271 as .2000,



Data Transfer (DX) Move Functions

Data transfer or "DX" move functions copy data (16 bit words) from one memory area to another. The data can then be operated on without altering the original data. These functions are available under the DX portion of the programming menu and include:

register to table move (R → T)

table to register move (T → R)

table to table move (T → T)

block move (BLKM)

first in (FIN)

first out (FOUT)

table search (SRCH)

system status (STAT)

Table A group of consecutive 16 bit words is referred to as a table. Several examples of tables are shown in the figure below, each has a "table length" of 5. Note, however, that each discrete table location contains the status ("1 = ON, "0" = OFF) of 16 coils or inputs.

Figure 9-33 Table Examples

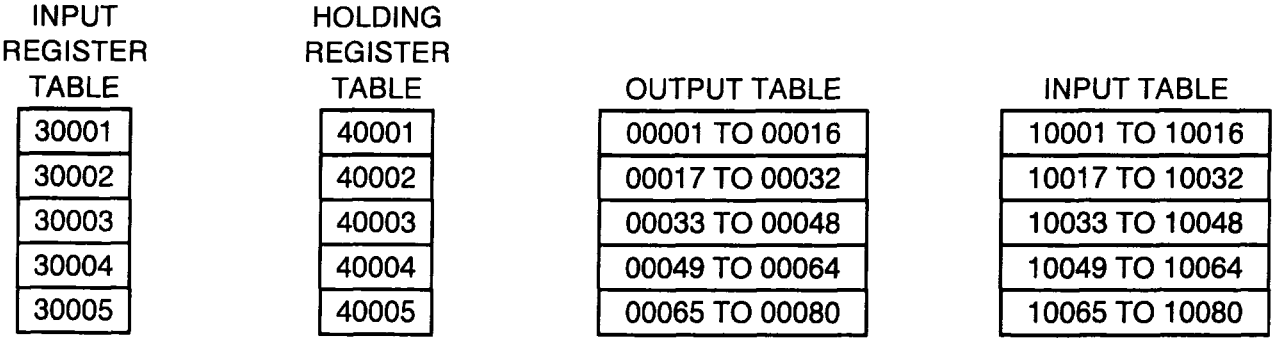
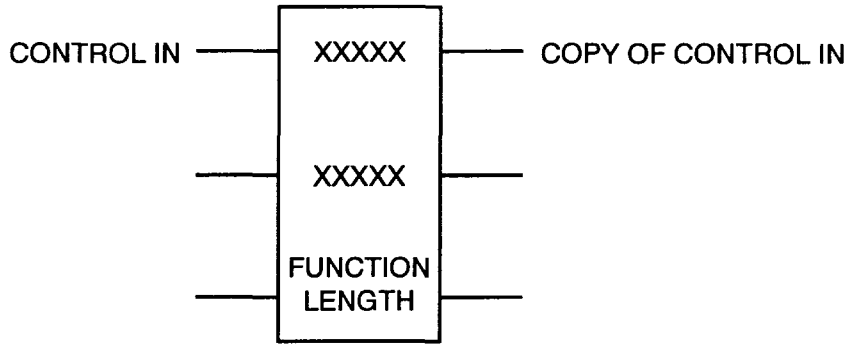


Figure 9-34 General Format



The figure above illustrates the general format of a data transfer move function block. The block consumes 1 horizontal and 3 vertical nodes (except system status which consumes 1 horizontal and 2 vertical nodes). The bottom node within the block specifies the function and the table length.

The top is always the "control in" input and the top output is always a copy of the "control in" input. This allows DX blocks to be cascaded. The top and middle nodes within the block, and all other inputs and outputs vary with each function.

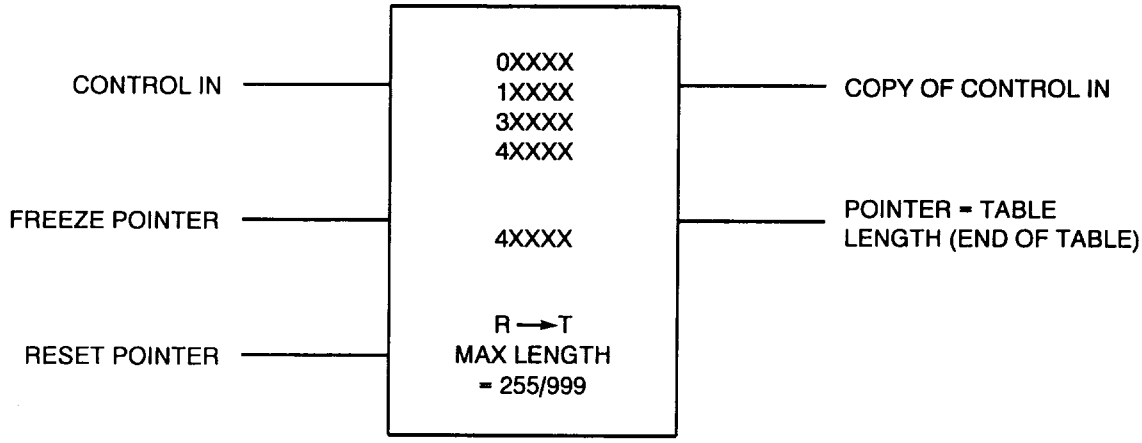
Pointer Some of the DX move functions dedicate a register which indicates which table position the data was:
copied from (T → R, T → T & FOUT),
written to (R → T & FIN),
found (SRCH).

This register is termed the "pointer register". The pointer register value will never exceed the table length. Zero is a valid pointer value, and typically indicates that the next operation of the DX block will copy data from or write data to the first table position. The pointer register may be manipulated by other logic functions (timers, counters, arithmetic operations, etc.)

Discrete Tables Discrete references are used in groups of 16. The reference number used is the first in the group; the other 15 references are implied. Additionally, only reference numbers which appear in the "first of 16" table shown in Table 9-1 may be used, e.g. 1, 17, 33, etc.

Register To Table Move This function copies the bit pattern of any register or 16 discretes to a specific register located within a table.

Figure 9-35 Register To Table Function Block



Function Block

The top node is the data source and can one of the following:
the first of 16 logic coils (0XXXX)
the first of 16 inputs (1XXXX)
an input register (3XXXX)
a holding register (4XXXX)

The middle node must be a holding register (4XXXX). This register is the pointer to the destination table which starts with the next consecutive register address (4XXXX + 1).

The bottom node contains the symbol R → T and a number specifying table length. Table length may range from:
1 to 255 in 16 bit controllers
1 to 999 in 24 bit controllers

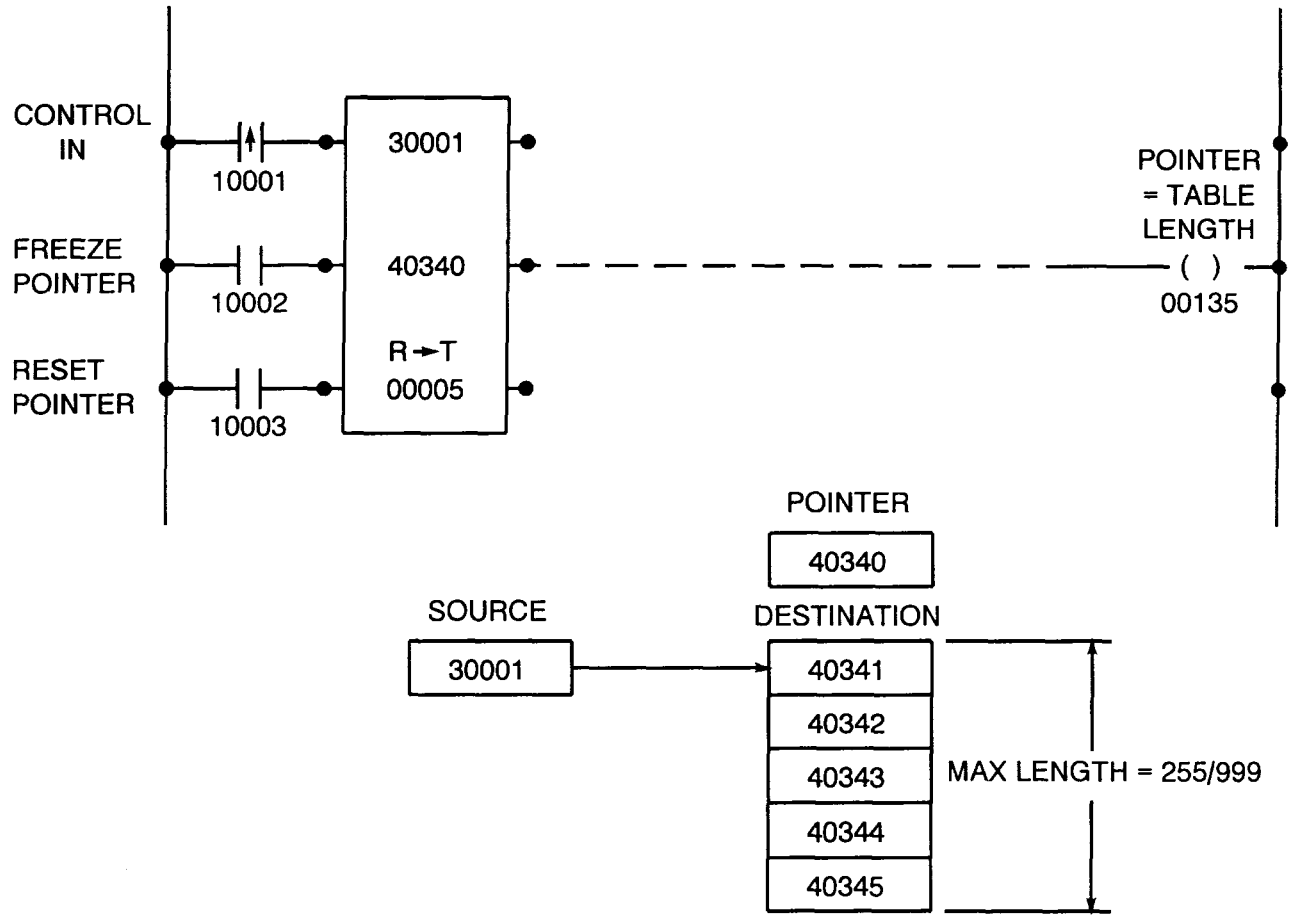
Inputs

Top: Control in. Every scan this node is powered, the R → T move is performed, and the pointer is incremented until the pointer value equals the table length. A transitional contact should be used if single operations are desired.
Middle: When powered, prevents pointer from increasing.
Bottom: When powered, resets pointer to zero.

Outputs

Top: Passes power when top input is powered.
Middle: Passes power when the pointer equals table length (end of table). No R → T operations are possible while the pointer value equals the table length.

Figure 9-36 Register To Table Network Example



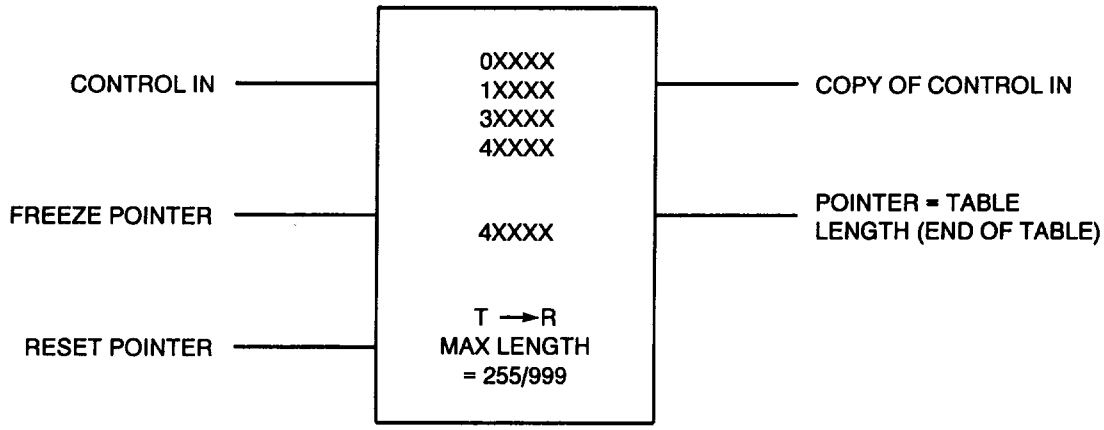
Network Description

The 1st transition of 10001 copies 30001 to 40341, increments pointer to 1.
The 2nd transition of 10001 copies 30001 to 40342, increments pointer to 2.
The 3rd transition of 10001 copies 30001 to 40343, increments pointer to 3.
The 4th transition of 10001 copies 30001 to 40344, increments pointer to 4.
The 5th transition of 10001 copies 30001 to 40345, increments pointer to 5, and, since the pointer value = table length, the middle output pases power, energizing coil 00135. No R → T operations are possible while the pointer value = the table length.

If, after the 2nd transition of 10001, 10002 were energized, this would prevent the pointer value from changing, and all subsequent transitions of 10001 would cause the value in 30001 to be copied to 40343. The pointer register will be reset to zero when 10003 is energized.

Table To Register Move This function copies the bit pattern of any register or 16 discretes located within a table to a specific holding register.

Figure 9-37 Table To Register Function Block



Function Block The top node is the data source and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 1XXXX reference number in a table of input references
the first 3XXXX reference number in a table of input registers
the first 4XXXX reference number in a table of holding registers

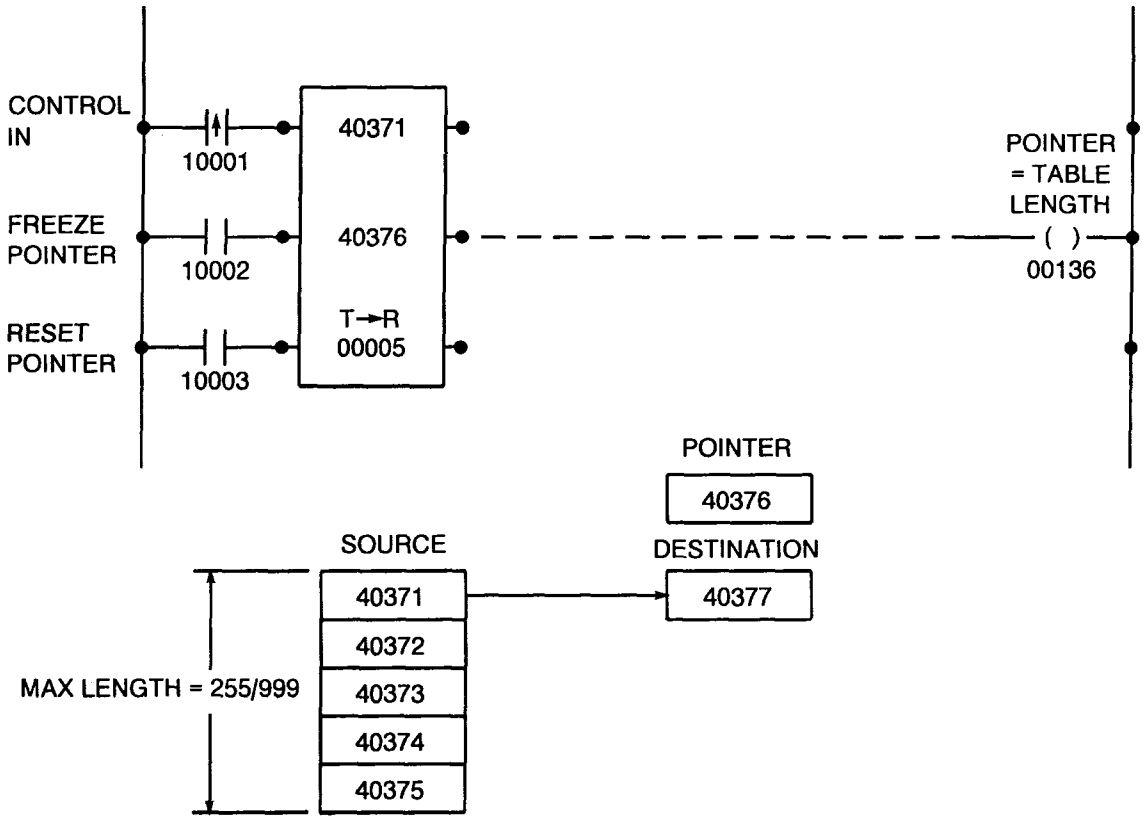
The middle node must be a holding register (4XXXX). This register is the pointer to the source table specified in the top node. The next consecutive register (4XXXX +1) is the destination register.

The bottom node contains the symbol T → R and a number specifying table length. Table length may range from:
1 to 255 in 16 bit controllers
1 to 999 in 24 bit controllers

Inputs
Top: Control in. Every scan this node is powered, the T → R move is performed, and the pointer is incremented until the pointer value equals the table length. A transitional contact should be used if single operations are desired.
Middle: When powered, prevents pointer from increasing.
Bottom: When powered, resets pointer to zero.

Outputs
Top: Passes power when top input is powered.
Middle: Passes power when the pointer equals table length (end of table). No T → R operations are possible while the pointer value equals the table length.

Figure 9-38 Table To Register Network Example



Network Description

The 1st transition of 10001 copies 40371 to 40377, increments pointer to 1.

The 2nd transition of 10001 copies 40372 to 40377, increments pointer to 2.

The 3rd transition of 10001 copies 40373 to 40377, increments pointer to 3.

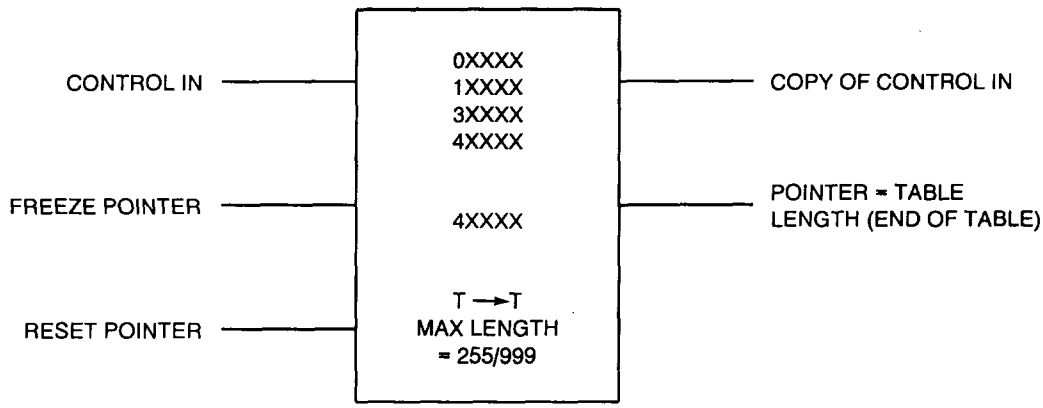
The 4th transition of 10001 copies 40374 to 40377, increments pointer to 4.

The 5th transition of 10001 copies 40375 to 40377, increments pointer to 5, and, since the pointer value = table length, the middle output passes power, energizing coil 00136. No T → R operations are possible whiie the pointer value = the table length.

If, after the 2nd transition of 10001, 10002 were energized, this would prevent the pointer value from changing, and all subsequent transitions of 10001 would cause the value in 40343 to be copied to 40377. The pointer register will be reset to zero when 10003 is energized.

Table To Table Move This function copies the bit pattern of any register or 16 discretes from a position within one table to the same position in a second table of holding registers.

Figure 9-39 Table To Table Function Block



Function Block The top node is the data source and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 1XXXX reference number in a table of input references
the first 3XXXX reference number in a table of input registers
the first 4XXXX reference number in a table of holding registers

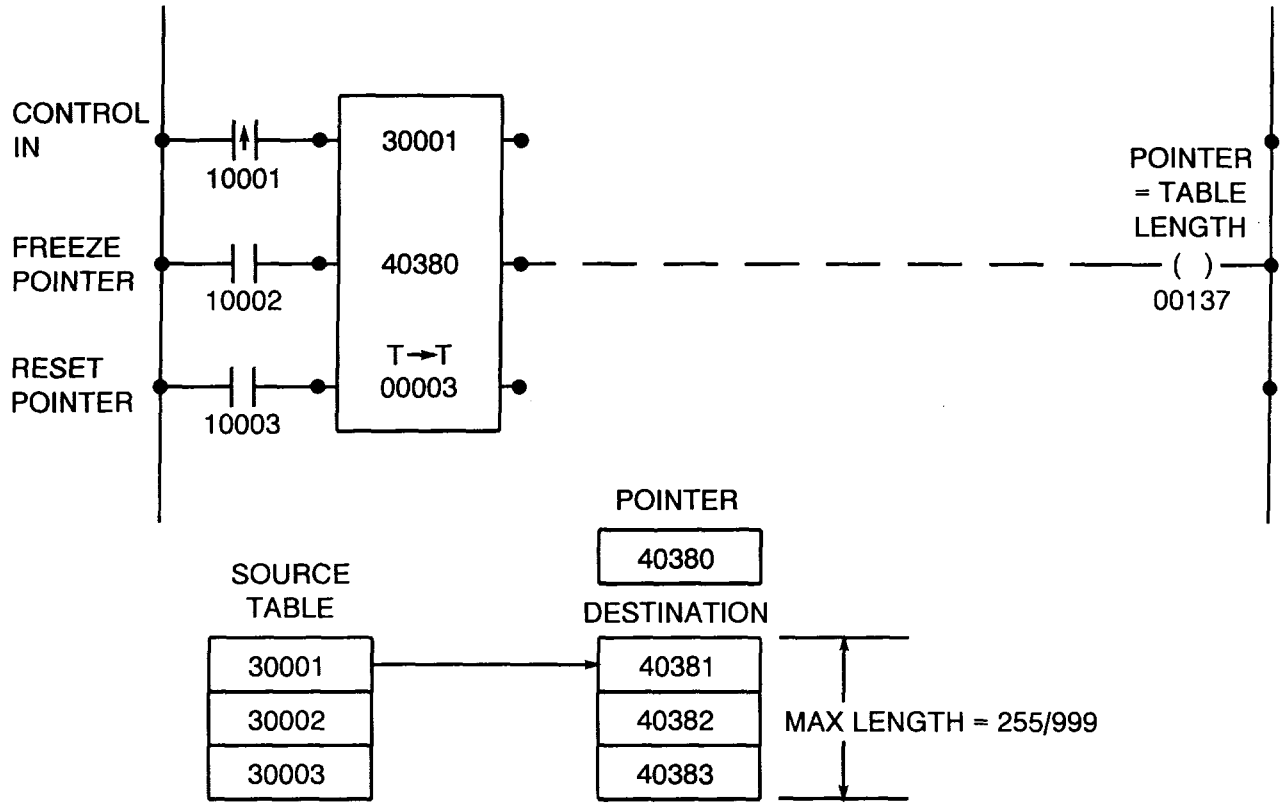
The middle node must be a holding register (4XXXX). This register is the pointer to both the source and destination tables. The next consecutive register (4XXXX +1) is the first register in the destination table.

The bottom node contains the symbol T → T and a number specifying table length. Table length may range from:
1 to 255 in 16 bit controllers
1 to 999 in 24 bit controllers

Inputs
Top: Control in. Every scan this node is powered, the T → T move is performed, and the pointer is incremented until the pointer value equals the table length. A transitional contact should be used if single operations are desired.
Middle: When powered, prevents pointer from increasing.
Bottom: When powered, resets pointer to zero.

Outputs
Top: Passes power when top input is powered
Middle: Passes power when the pointer equals table length (end of table). No T → T operations are possible while the pointer value equals the table length.

Figure 9-40 Table To Table Network Example



Network Description

The 1st transition of 10001 copies 30001 to 40381, increments pointer to 1.

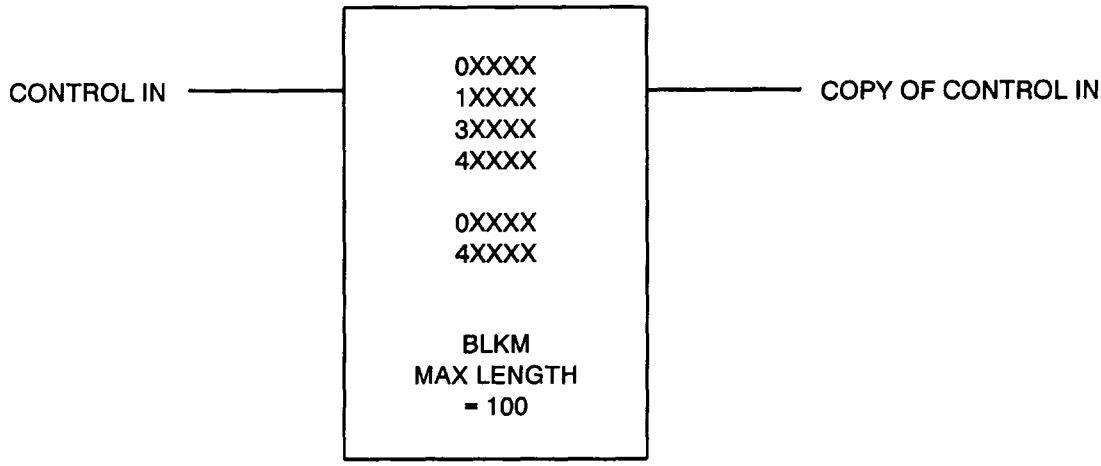
The 2nd transition of 10001 copies 30002 to 40382, increments pointer to 2.

The 3rd transition of 10001 copies 30003 to 40383, increments pointer to 3, and, since the pointer value equals table length, the middle output passes power, energizing coil 00137. No T → T operations are possible while the pointer value equals the table length.

If after the 2nd transition of 10001, 10002 were energized, this would prevent the pointer value from changing, and all subsequent transitions of 10001 would cause the value in 30003 to be copied to 40383.

Block Move This function copies, during one scan, the entire contents of any table to a table of outputs or holding registers.

Figure 9-41 Block Move Function Block



Function Block The top node is the source table and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 1XXXX reference number in a table of input references
the first 3XXXX reference number in a table of input registers
the first 4XXXX reference number in a table of holding registers

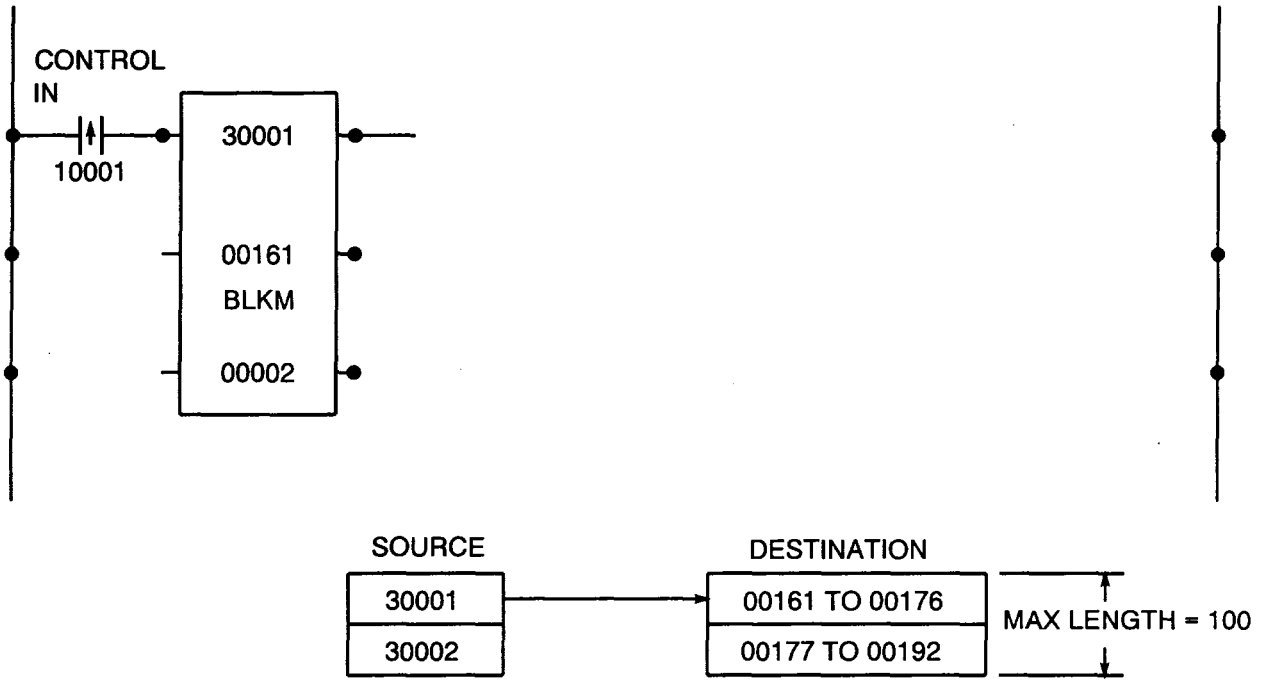
The middle node is the destination table and can be one of the following: the first 0XXXX reference number in a table of output registers or the first 4XXXX reference number in a table of holding registers.

The bottom node contains the symbol BLKM and a number specifying table length. Table length may range from 1 to 100.

Inputs
Top Control in. Every scan this node is powered, the block move is performed. A transitional contact should be used if single operations are desired.

Outputs
Top: Passes power when top input is powered.

Figure 9-42 Block Move Network Example



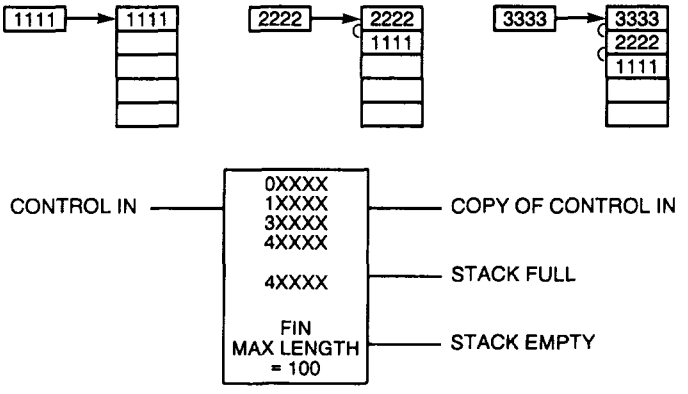
Network Description When 10001 is energized, the contents of the source table will be copied into the destination table.

In the case shown above, the destination is a table of outputs. These coils, like all coils, may only be used once. Attempts, for example, to program coil 165 in another network will result in a P190 error message: "coil used". A "search" for coil 165 via the PI90 will result in the display of the network the block move is programmed within.

➤➤ **WARNING The destination for this function can be a table of outputs (0XXXX). This function will override any disabled coils within this table without enabling them.**

First In This function copies the bit pattern of any register or 16 discretes to the first register in a table of holding registers. This register is the start of a queue or stack. Each entry to the stack causes the previous entry to move to the next position in the stack. This block typically is used with the first out block. Together they form a fifo (first in, first out) stack which is analogous to a paper cup dispenser - cups are removed in the same order they were inserted. Three successive first in operations are illustrated in the figure below.

Figure 9-43 First In Operations



Function Block The top node is the data source and can be one of the following:
the first of 16 logic coils (0XXXX)
the first of 16 inputs (1XXXX)
an input register (3XXXX)
a holding register (4XXXX)

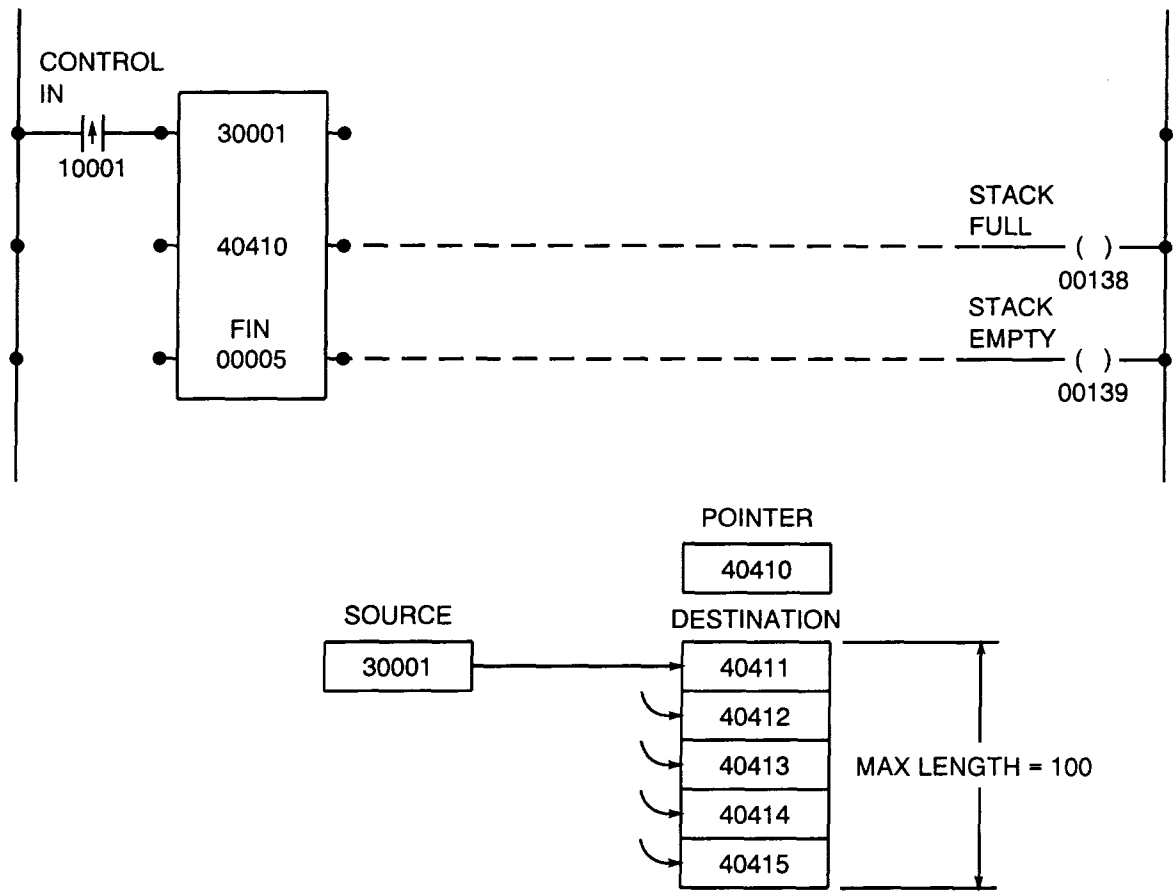
The middle node must be a holding register (4XXXX). This register is the pointer to the destination table which starts with the next consecutive register (4XXXX +1). The value contained in the pointer register equals the current number of entries in the table or stack.

The bottom node contains the symbol FIN and a number specifying table length, which should equal FOUT block. Table length may range from 1 to 100.

Inputs
Top: Control in. Every scan of this node is powered, the First In move is performed, and the pointer is incremented until the pointer value = the table length, A transitional contact should be used if single operations are desired.

Outputs
Top: Passes power when top input is powered
Middle: Passes power when the stack is full. No additional entries to the stack are possible while this output is on.
Bottom: Passes power when the stack is empty.

Figure 9-44 First In Network Examples



Network Description In the figure above, the 1st transition of 10001 copies 30001 to 40411, increments pointer to 1, de-energizes 00139 (stack not empty)

The 2nd transition of 10001 moves contents of 40411 to 40412, copies 30001 to 40411, increments pointer to 2.

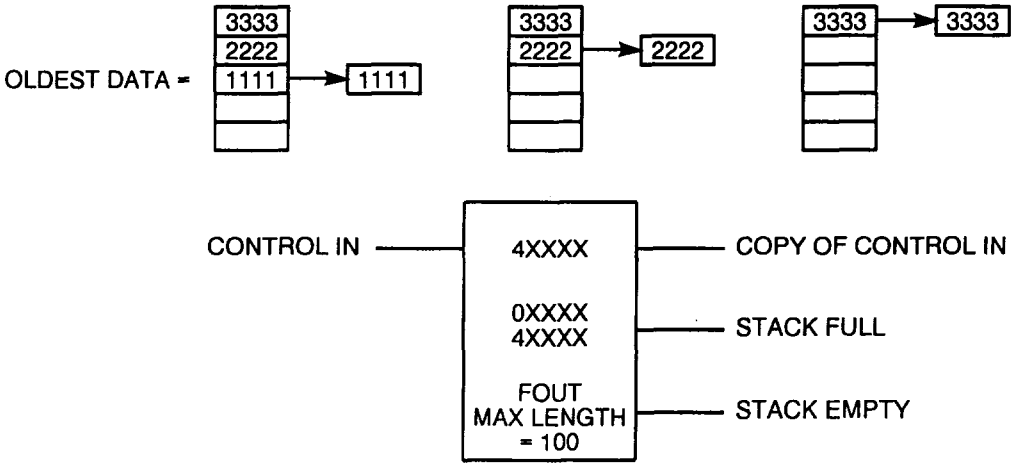
The 3rd transition of 10001 moves contents of 40412 to 40413, moves contents of 40411 to 40412, copies 30001 to 40411, increments pointer to 3.

The 4th transition of 10001 moves contents of 40413 to 40414, moves contents of 40412 to 40413, moves contents of 40411 to 40412, copies 30001 to 40411, increments pointer to 4.

The 5th transition of 10001 moves contents of 40414 to 40415, moves contents of 40413 to 40414, moves contents of 40412 to 40413, moves contents of 40411 to 40412, copies 30001 to 40411, increments pointer to 5, and energizes 00138 (stack full - no more entries possible while pointer equals table length).

First Out This function copies the bit pattern of a register within a table of holding registers. The register from which the data will be copied is the register with the oldest data (the highest register number in the table with data in it). This block typically is used with the First In block. Together, they form a FIFO (First In, First Out) stack which is analogous to a paper cup dispenser - cups are removed in the same order they were inserted. Three successive first out operations are illustrated in the figure below.

Figure 9-45 First Out Operations



Function Block The top node must be a holding register (4XXXX). This register is the pointer to the source table which starts with the next consecutive register (4XXXX +1). The number contained in the pointer register indicates which table position the data will be removed from (the current number of entries in the stack).

The middle node is the data destination and can be one of the following:
the first of 16 logic coils (0XXXX)
a holding register (4XXXX)

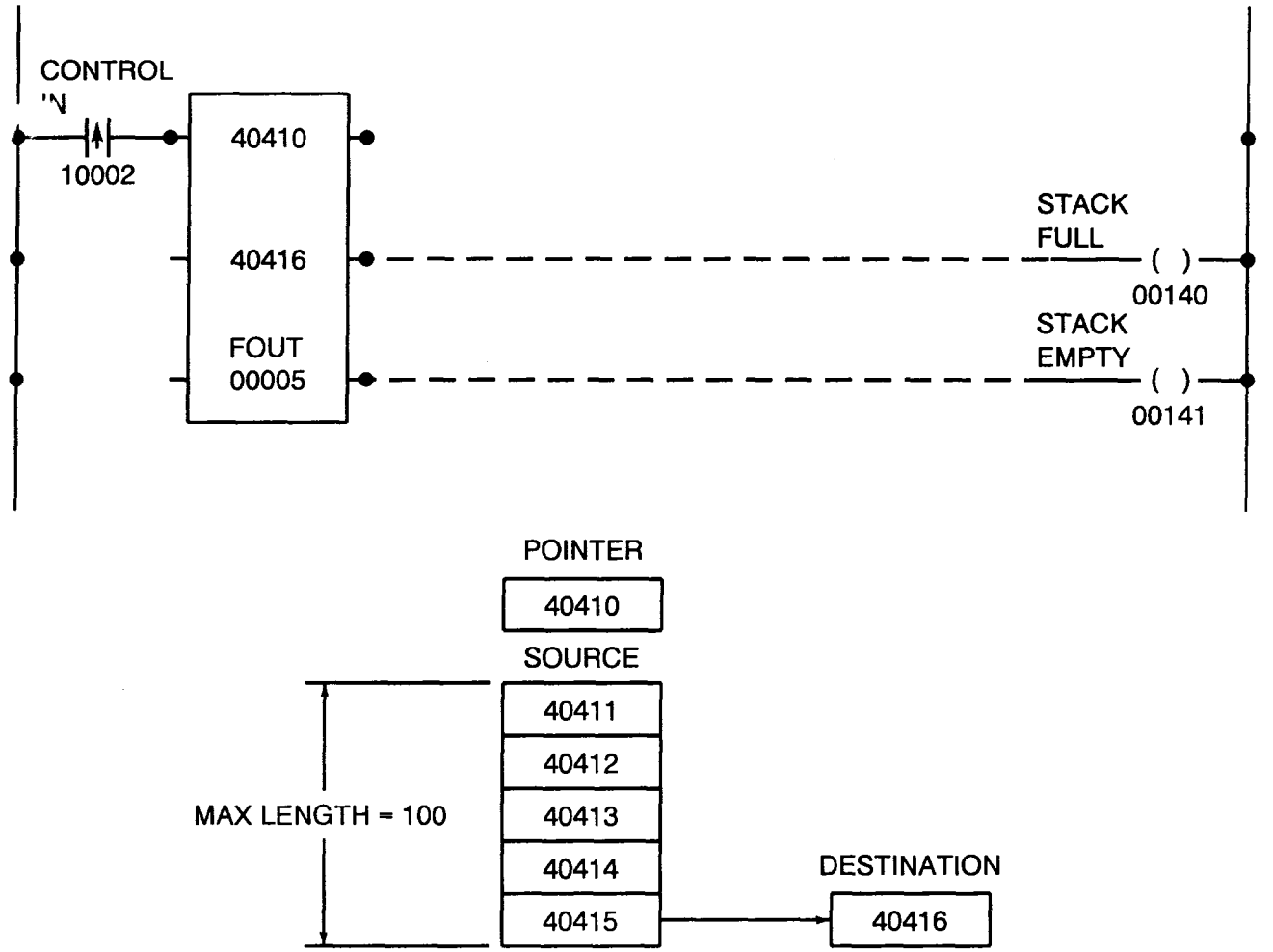
The bottom node contains the symbol FOUT and a number specifying table length. Table length may range from 1 to 100.

Inputs
Top: Control in. Every scan this node is powered, the First Out move is performed, and the pointer is decremented until the pointer value equals zero.

Outputs
Top: Passes power when top input is powered
Middle: Passes power when the stack is full. No additional entries to the stack are allowed while this output is on.
Bottom: Passes power when the stack is empty.

Figure 9-46 First Out Network Example

FIN works best with FOUT as a FIFO pair if the same pointer address is used. 40410 in these examples.



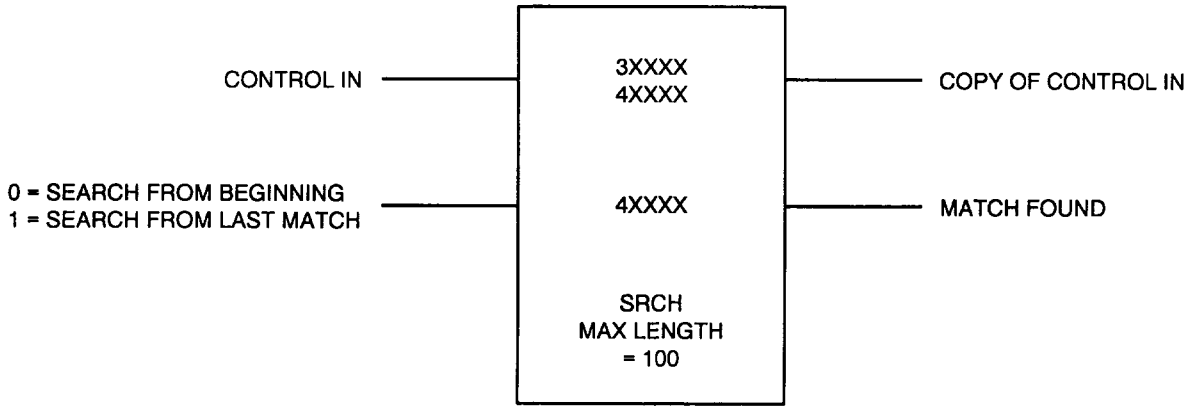
Network Description In the figure above, if the stack is full, pointer value = 5, and coil 00140 is energized, then:

The 1st transition of 10002 copies 40415 to 40001, decrements pointer to 4, de-energizes coil 00140 (stack not full).
The 2nd transition of 10002 copies 40414 to 40001, decrements pointer to 3.
The 3rd transition of 10002 copies 40413 to 40001, decrements pointer to 2.
The 4th transition of 10002 copies 40012 to 40001, decrements pointer to 1
The 5th transition of 10002 copies 40011 to 40001, decrements pointer to 0, energizes 00141 (stack is empty).

➤ ➤ **WARNING The destination for this function can be a group of 16 outputs. This function will override any disabled coils within this group.**

Search This function searches a table of registers for a specific bit pattern.

Figure 9-47 Search Function Block



Function Block The top node is the table to be searched and can be one of the following:
the first 3XXXX reference number in a table of input registers
the first 4XXXX reference number in a table of holding registers.

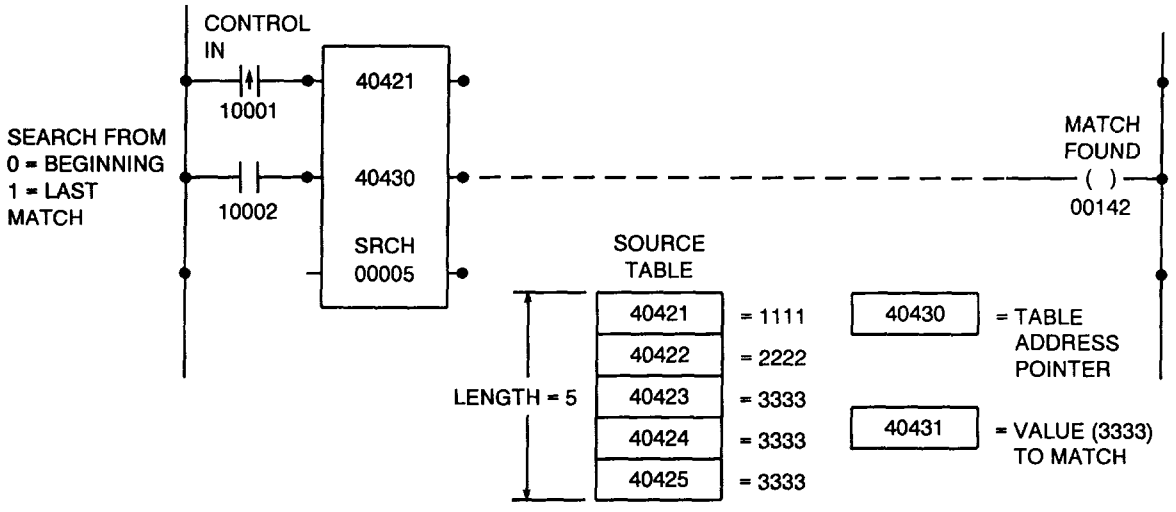
The middle node must be a holding register (4XXXX). This register is the pointer to an address in the table specified in the top node. The next consecutive register 4XXXX +1 contains the value or bit pattern to be searched for.

The bottom node contains the symbol SRCH and a number specifying table length. Table length may range from 1 to 100.

Inputs
Top: Control in. Every scan this node is powered, a search is performed. A transitional contact should be used if single operations are desired.
Middle: When powered, table will be searched from the last match when not powered, table will be searched from the beginning.

Outputs
Top: Passes power when top input is powered.
Middle: Passes power when a match is found.

Figure 9-48 Search Network Example



Network Description In the figure above, if 10002 is not energized, every scan that the "control in" node receives power, the source table will be searched for a "3333". The search will:
increment the pointer address,
look for a match at the pointer address,
find a match in register 40423,
stop searching for the remainder of scan,
energize coil 00142 for one scan,
reset counter address to zero.

If 10002 is energized 10001 transitions from off to on, the source table will be searched for a "3333". The search will:
increment the pointer address,
find a match in register 40423,
stop the search,
energize coil 00142 for 1 scan,
leave the pointer address at 3.

The 2nd transition of 10001 will:
increment the pointer address,
find a match in register 40424,
stop the search,
energize coil 00142 for 1 scan,
leave the pointer address at 4.

The 3rd transition of 10001 will:
increment the pointer address,
find a match in register 40425,
stop the search,
energize coil 00142 for 1 scan,
reset the pointer address to zero.

Stat

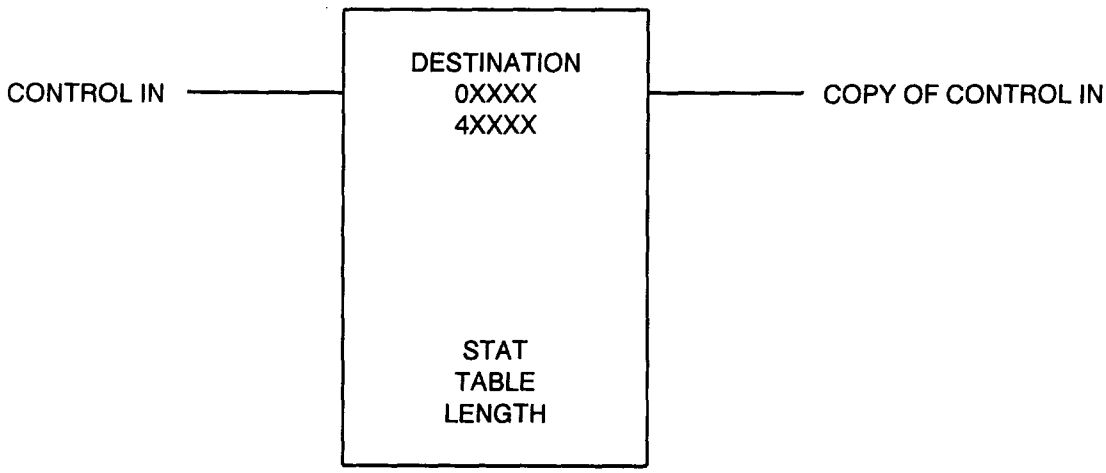
Copies the contents of the controller's status table to a table of holding registers or discrete output references.

Performs the full transfer from the status table to the destination table during one scan.

Holding registers should be used for destinations. If discrete outputs are specified for destinations, a large number of them will be required.

Will override any disabled coils within the destination table without enabling them.

Figure 9-49 Status Block



Nodes Top: 0XXXX: The first coil in a table of logic coils.
4XXXX: The first register in a table of holding registers.

Bottom: Symbol STAT with decimal value specifying the "table length" (number of registers) to be transferred. Can range from 1 to the maximum length of the controller's status table. 984 controller status table lengths (decimal):
16-bit controllers with S901: 71
24-bit controllers with S901: 75
16-bit controllers with S908: 255
24-bit controllers with S908: 277

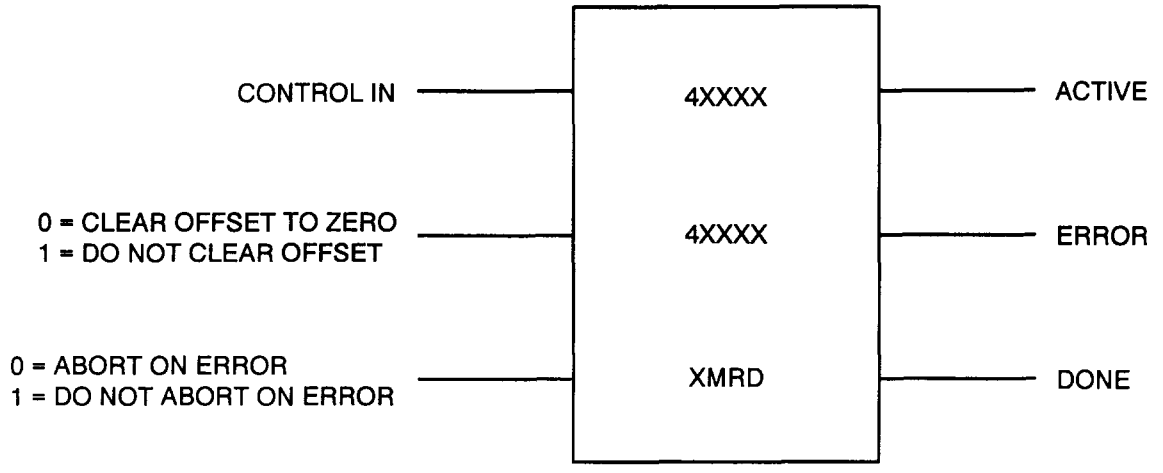
Inputs
Top: The STAT is performed on every scan when this input is powered.
Bottom: Not used

Outputs
Top: Passes power when the top input is powered.
Bottom: Not used

➤ **NOTE** Refer to Chapter 12 of this manual for a more detailed discussion of how the Stat Block is used with the 984/S901 and 984/S908 I/O processors.

Extended Memory Read This function block copies a table of extended memory registers (6XXXX) in a 984 to a table of holding registers (4XXXX).

Figure 9-50 Extended Memory Read Function



Function Block The top node must be a holding register (4XXXX) address. This register is the first in a table of 6 holding registers, which control the parameters of the file transfer.

The middle node is the destination table and must be the first 4XXXX reference number in a table of holding registers which will receive the data transferred from the extended memory storage registers (6XXXX).

The bottom node contains the symbol XMRD, and the value 00001 is automatically displayed when the block is programmed.

Inputs
Top: Control in. Every scan this node is powered, the data transfer will occur.
Middle: When powered, the offset value will not be cleared, When not powered, the offset value will be cleared to zero before the instruction is solved.
Bottom: When powered, the 984 will not automatically power down when a parity diagnostic error is detected.

Outputs
Top: Active. Passes power from the time the top input is initially asserted, until the specified number of registers have been transferred.
Middle: Error. Passes power if the top input is powered and an error is detected.
Bottom: Done. Passes power for one scan when all registers have been transferred.

Top Node Register Usage The top node defines a control table of 6 holding (4XXXX) registers as follows: 4XXXX = Status Word.

Figure 9-51 Bit Definition for First Top Node Address (Status Word). See expanded definitions on page 9-74.

Bit #	Definition
0	File parameter error
1	Starting address parameter error
2	Count parameter error
3	Offset parameter error
4	Max parameter error
5	
6	Non-existent SRAM
7	
8	
9	Offset parameter too large on entry
10	File boundary crossed
11	Done
12	Busy
13	Non-existent x-mem
14	X-mem parity error
15	Power up diagnostic failure

4xxxx + 1 = File Number
The extended memory area is divided into 2, 4, 7, or 10 files depending on 984 memory size. This register specifies which file is to be used.

4xxxx + 2 = Start Address
Specifies the starting address (0 to 9999) in the file specified by the previous register. 0 = first register, 9999 = 10,000th register.

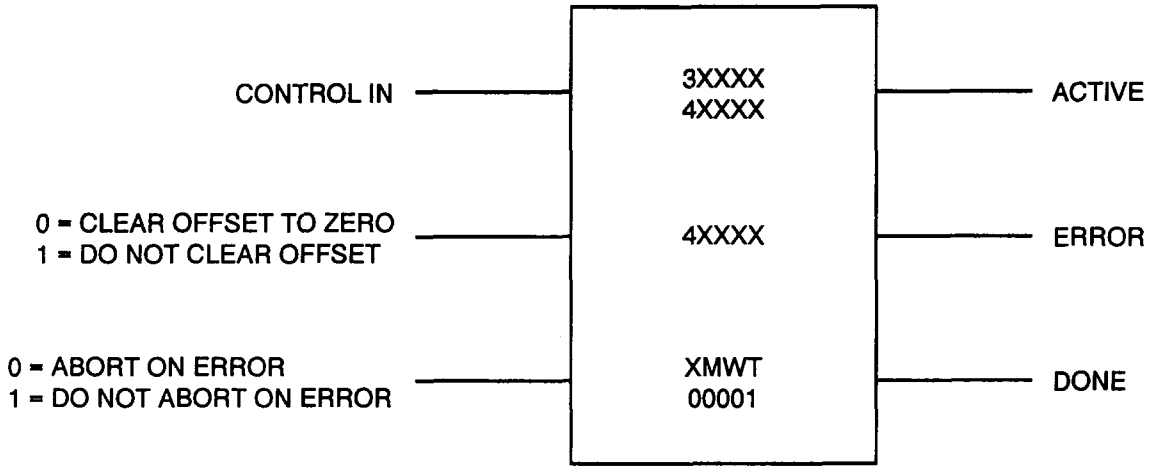
4xxxx + 3 = Number of Registers to be Transferred per Scan
This register specifies the number of registers to be transferred per scan. Range = 0 to 9999. 0 = 1 register, 9999 = 10,000 registers.

4xxxx + 4 = Offset
This is the running total of the number of registers which have been transferred thus far.

4xxxx + 5 = Total Number of Registers to be Transferred
This register specifies the maximum number of registers to be transferred when the block is powered. Range = 0 to 9999 (which may span several scans).

Extended Memory Write This function block copies a table of holding (4XXXX) or input (3XXXX) registers to a table of 6XXXX registers located within the extended memory area of a 984.

Figure 9-52 Extended Memory Write



Function Block The top node is the source table and can be either: the first 3XXXX reference number in a table of input registers or the first 4XXXX reference number in a table of holding registers

The middle node must be a holding register (4XXXX). This register is the first in a table of 6 holding registers, which control the parameters of the file transfer.

The bottom node contains the symbol XMWT, and the value 00001 is automatically displayed when the block is programmed.

Inputs
Top: Control in. Every scan this node is powered, the data transfer will occur.
Middle: When powered, the offset value will not be cleared. When not powered, the offset value will be cleared to zero before the instruction is solved.
Bottom: When powered, the 984 will not automatically power down when a parity diagnostic error is detected.

Outputs
Top: Active. Passes power from the time the top input is initially asserted, until the specified number of registers have been transferred.
Middle: Error. Passes power if the top input is powered and an error is detected.
Bottom: Done. Passes power for one scan when all registers have been transferred.

Extended Memory Write

Middle Node Register Usage The middle node defines a table of 6 holding (4XXXX) registers as follows: 4XXXX = Status Word

Figure 9-53 Extended Memory Write Bit Definition For First Middle Node Address (Status Word), see page 9-74.

Bit #	Definition
0	File parameter error
1	Starting address parameter error
2	Count parameter error
3	Offset parameter error
4	Max parameter error
5	
6	Non-existent SRAM
7	
8	
9	Offset parameter too large on entry
10	File boundary crossed
11	Done
12	Busy
13	Non-existent x-mem
14	X-mem parity error
15	Power up diagnostic failure

4XXXX + 1 = File Number
The extended memory area is divided into 10 files. Each file has 10,000 registers. This register specifies which file (1-10) is to be used.

4xxxx + 2 = 6XXXX Start Address
Specifies the starting address (0 to 9999) in the file specified by the previous register. 0 = first register, 9999 = 10,000th register.

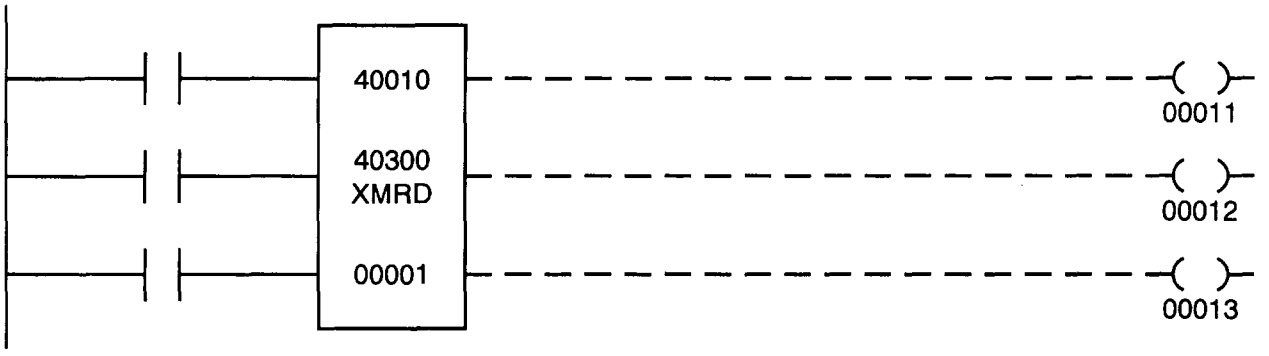
4xxxx + 3 = Number of Registers to be Transferred per Scan
This register specifies the number of registers to be transferred per scan. Range = 0 to 9999. 0 = 1 register, 9999 = 10,000 registers.

4xxxx + 4 = Offset
This is the running total of the number of registers which have been transferred thus far. Termed "offset".

4xxxx + 5 = Total Number of Registers to be Transferred
This register specifies the maximum number of registers to be transferred when the block is powered. Range = 0 to 9999.

The bottom node contains the symbol XMWT, and the value 00001 is automatically displayed when the block is programmed.

Figure 9-54 Extended Memory Read Example



In the figure above example, a block of 1400 6XXXX registers are read in two scans (700 per scan) from extended memory. The block starts at register 3000 in file three and is written to a block starting at register 40300.

The control table, displayed in the figure below, begins at register 40010. The value in 40011, 0003, is the extended memory file number. The value 3000 in register 40012 is the starting register in the extended memory block that is read. The value 0700 in register 40013 is the number of registers transferred per scan. Register 40014 contains the offset count which changes throughout the function. The value in register 40015, 1400, is the total number of registers being transferred.

Coil 00011 is energized when the function block is active and coil 00013 is energized for one scan when the read is complete. Coil 00012 is used to indicate the error.

Figure 9-55 Control Table

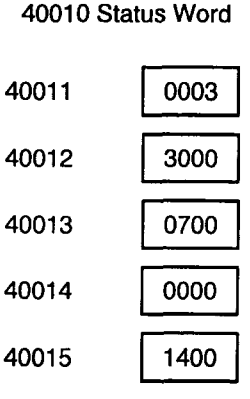
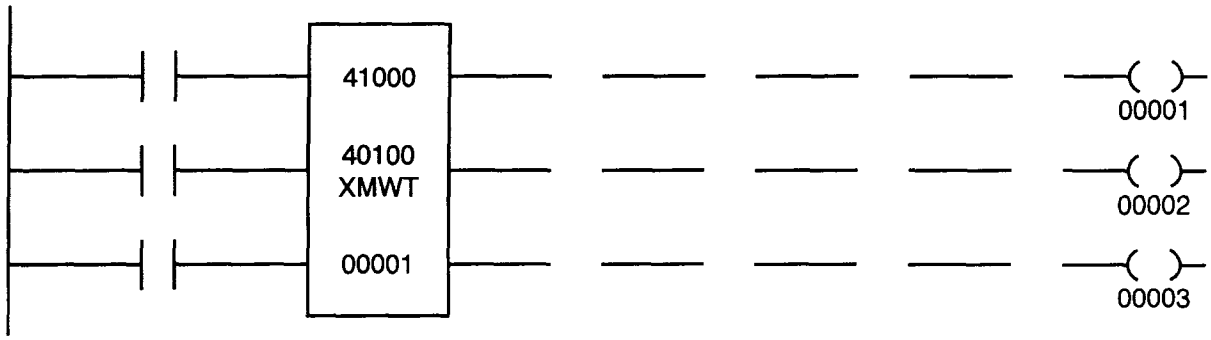


Figure 9-56 Extended Memory Write Network Example



In the example above, a block of 1000 4XXXX registers is written in one scan to extended memory. The block starts at register 41000 and is written to a block starting at register 2000 in file two.

The control table, displayed in the figure below, begins at register 40100. The value in register 40101, 0002, is the extended memory file number, The value 2000 in register 40102 is the location in extended memory that the block of registers is written to. The value 1000 in register 40103 is the number of registers transferred per scan. Register 40104 contains the offset count which changes throughout the function. The value 1000 in register 40105 is the total number of registers being transferred.

Coil 00001 is energized when the function block is active and coil 00003 is energized for one scan when the write is complete. Coil 00002 is used to indicate an error.

Figure 9-57 Control Table

40100 Status Word

40101	0002
40102	2000
40103	1000
40104	0000
40105	1000

Extended Memory Status Word Format

Bit 0 -- File Parameter Error
The second register in the control block (4XXXX + 1) contains an illegal value for the number of files. Legal values range from 1 to 10.

Bit 1 -- Starting Address Parameter Error
The third register in the control block (4XXXX + 2) contains an illegal value for the starting address. Legal values range from 0 to 9999 except in the last file. The legal values for the last file number and its last address are:

984 Size	XMEM Size	Last File	Last Address
48K	16K	2	6383
64K	32K	4	2767
96K	64K	7	5535
128K	96K	10	8403

Bit 2 -- Count Parameter Error
The fourth register in the control block (4XXXX + 3) contains an illegal value. Valid count values range from 0 to 9999.

Bit 3 -- Offset Parameter Error
The fifth register in the control block (4XXXX + 4) contains an illegal value. Valid offset values range from 0 to 9999.

Bit 4 -- Maximum Register Parameter Error
The sixth register in the control block (4XXXX + 5) contains an illegal value. Valid maximum register values range from 0 to 9999.

Bit 5 -- Not Used.

Bit 6 -- Non-existent State RAM
This bit is set when the 4XXXX or 3XXXX register, specified in the node as source or destination, plus the maximum number of registers to transfer will exceed the last 3XXXX or 4XXXX register available in the 984 machine configuration. The bit is set the first time the node is active even in the multiple scan mode. This bit is set even if the actual transfer that would cause the error occurs in the latter scan. No data is transferred as long as the condition exists.

Bit 7 -- Not Used.

Bit 8 -- Not Used.

Bit 9 -- Offset Parameter Too Large On Entry
If you are in the do-not-clear-offset mode upon entering the node and the offset value is equal to or greater than the maximum number of registers to be transferred, this bit is set. No data transfer is performed.

Bit 10 -- File Boundary Crossed
This bit is set during the scan that results in data being transferred to to or from extended memory and the address crosses a file boundary. On single scan transfers it is set for the scan that the node is active and the file crossing occurs. During multiple scan transfers the bit is set during the scan that the transfer occurs and remains on for all transfers to the upper file. The file number in the control block is not changed.

Bit 11 -- Done
This bit is set when a transfer is complete even if a parity error was detected during the transfer.

Bit 12 -- Busy
This bit is set on multiple scan transfers, when the scan node is active and the transfer is not complete. The busy bit is not set if all validation checks have not been successful.

Bit 13 -- Non-existent XMEM
This bit is set if the read/write data transfer will reference a register address that does not exist in the current configuration, On a single scan transfer, if the sum of the starting address and the count references a register address that does not exist in the current configuration, the bit is set. On multiple scan transfers, if the sum of the starting address and the maximum number of registers to be transferred references a value outside the configuration, the bit is set. The setting of the bit inhibits data transfer even if the transfer to or from the non-existent registers will not occur during the current scan.

Bit 14 -- XMEM Parity Error
This bit is set when the logic detects a parity error when attempting to read an extended memory location.

Bit 15 -- Power-up Diagnostic Failure
This bit is set when the node is active and an error was previously detected during the extended memory power-up diagnostic.

Subroutine Functions – Models 984 - x8x Only

Three additional DX subroutine nodes (JSR, RET, and LAB) are included in the controller's basic functionality. JSR is a Jump to Subroutine command, RET is a RETurn from subroutine command, and LAB is a LABel command that marks the entry point of a subroutine network.

To use them, configure the system for one more segment than is required for the user logic. Then, remove the last segment entry from the segment scheduler's order-of-solve table. All subroutines must be built in the last segment of user logic.

The JSR logic element transfers execution to the LABel node. Then the subroutine executes normally until one of the following conditions is met:

1. An energized RET node,

2. An End of Logic node,

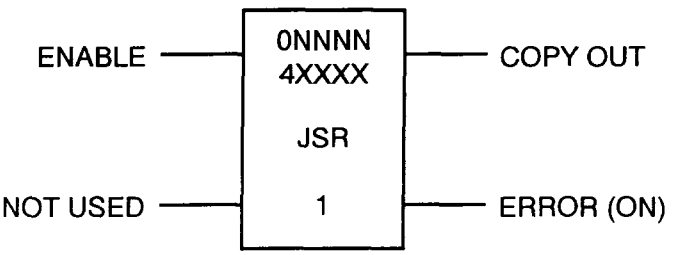
3. The next LAB node.

Subroutine nesting and ladder logic recursion are allowed, provided the subroutine stack does not exceed 100. (Infinitely recursive subroutine loops are broken at the l00th invocation. They will not crash the user logic - although the watchdog timer could expire before the l00th invocation.)

Jump to Subroutine (JSR) The jump to subroutine function is a two node block that conditionally jumps to the subroutine given by the source parameter.

The JSR block can appear anywhere within the logic, even within a subroutine (jump to a subroutine from within a subroutine). This is known as nesting, or looping if a subroutine calls itself.

Figure 9-58 JSR Function Block



Source Node
The source node, or top node value, indicates to which subroutine to jump, this node can be a constant or a single holding register value between 1 and 255, inclusive.

Size
The size, or bottom node, must be a constant of one.

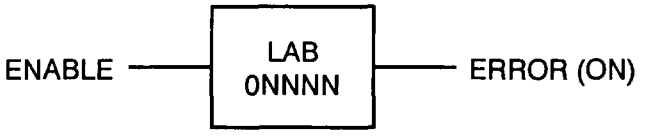
Inputs
The top input of the block, Enable, is used. When this input receives power, logic solving will jump to the subroutine specified by the source node value.

Outputs
The top output is a copy of the Enable input. If the Enable input receives power, this output will be energized. If the Enable input does not receive power, this output will be de-energized. Note, this output is only energized after logic solving has returned from the subroutine.

The bottom output, Error, is energized if the jump cannot be executed for either of the following reasons: The label node of the target subroutine does not exist, or the maximum subroutine nesting stack level of 100 has been exceeded. In all other cases this output will be off.

Label (LAB) The Label function is a one node block that is used to label the starting point of a subroutine. The label block also functions as a default return for the subroutine in the preceeding networks. This function must be programmed in row 1, column 1 of a network in the last segment.

Figure 9-59 LAB Function Block



Size

The Label size value assigns a number to the following subroutine and must be a constant between 1 and 255, inclusive. This means the user can create up to 255 different subroutines. Each label function must have a unique number. If multiple networks in the last segment have the same label value, the network with the lower number is used as the starting point for that subroutine.

Input

There is only one input to the label block. This input is the Enable and is always on since the block must be programmed in row 1, column 1.

Output

The only output of this block is the Error output. This output will be turned on if the return from this subroutine cannot be executed because the subroutine stack is empty. In all other cases this output is off.

Return (RET) The return function is a one node block that is used to conditionally return logic execution to the node immediately following the most recently executed JSR instruction. This node only has effect if it is placed in a network in the last segment.

If a subroutine does not contain a RET instruction, either the next LAB instruction or the end of logic, which ever comes first, will act as a default return.

Figure 9-60 RET Function Block

Size
The size must be a constant of one.

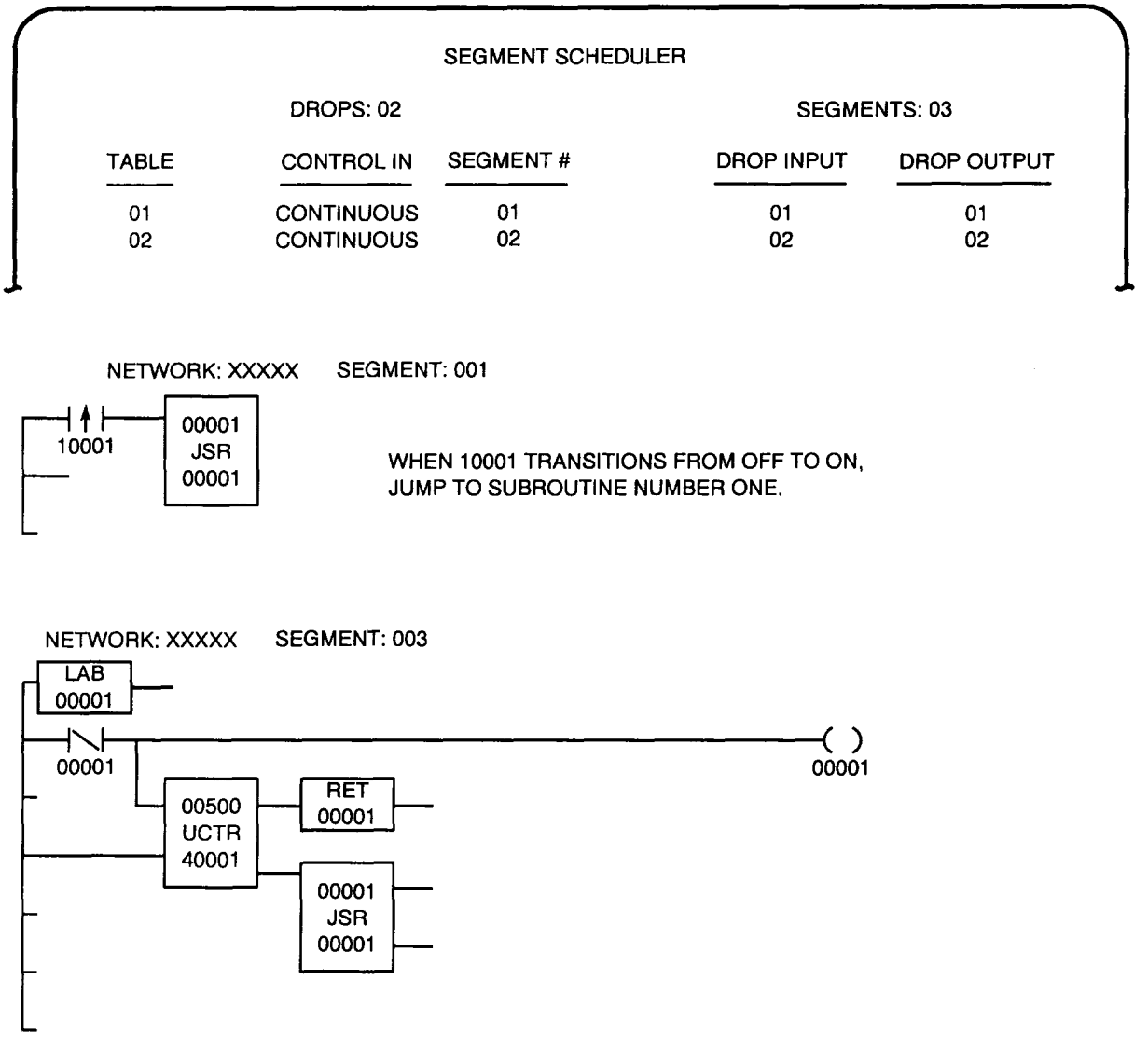
Input
The Enable input, when energized, returns logic execution to the node immediately following the most recently executed JSR.

Output
The Error output is turned on only if the return function cannot be executed because the subroutine stack is already empty.

Subroutine Ladder Logic Example The figure below is a simple ladder logic example of looping. Although the counter preset is 500, it will only count to 50 because the subroutine stack is limited to l00; 100 scans equals 50 counts. Note that segment three has been removed from the segment scheduler.

Figure 9-61 Subroutine Looping Example

Each scan, contacts 00001 oscillates, causing the counter to be incremented by one every other scan. If 40001 is less than 500 (bottom output on), jump to beginning of subroutine number 1 (again). When 40001 equals 500, (top output on) return to main logic. However, 40001 will never equal 500, because the subroutine stack is limited to 100 which energizes the error output.



Data Transfer (DX) Matrix Functions Overview

DX Matrix Functions

A matrix is a sequence of data bits formed by consecutive 16 bit words derived from tables. The figure below depicts the bit matrix derived from 3 consecutive holding registers. Note the bit numbering scheme.

Figure 9-62 Bit Matrix

HOLDING REGISTER TABLE

		BIT MATRIX															
40001	=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
40002	=	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
40003	=	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48

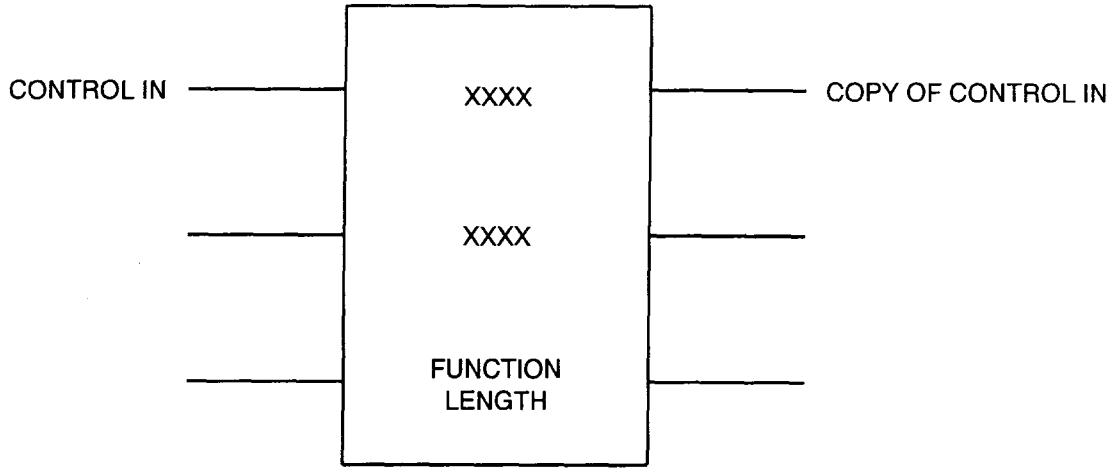
Format

The figure below illustrates the general format of a data transfer matrix function block. The block consumes 1 horizontal and 3 vertical nodes. The bottom node within the block specifies the function and the matrix length in 16 bit Words.

The top input is always the "control in" input and the top output is always a copy of the "control in" input. This allows DX blocks to be cascaded.

The top and middle nodes within the block, and all other inputs and outputs vary with each function.

Figure 9-63 Matrix Function Block



Length
Minimum table length is one (one 16 bit word or register). Maximum table length is dependent on the DX function and on the controller type (16 or 24 bit CPU). See figure below.

Figure 9-64 Table Lengths

Controller Model	AND	OR	XOR	COMP	CMPR	MBIT	SENS	BROT
16 Bit CPU	100	100	100	100	100	255	255	100
24 Bit CPU	100	100	100	100	100	600	600	100

Pointer
The compare, bit modify, and bit sense functions utilize a pointer to indicate which bit in the matrix is being operated on. A register used to hold the pointer value for the bit modify or bit sense function may be manipulated by other logic functions (timers, counters, arithmetic operations, etc.).

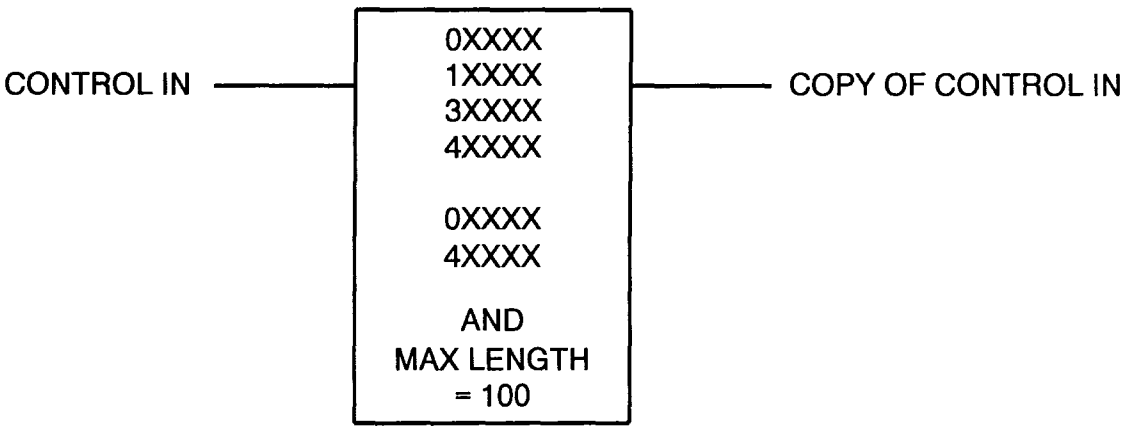
Discrete Tables
Discrete references are used in groups of 16. The reference number used is the first in the group; the other 15 references are implied. Additionally, only reference numbers which appear in the "first of 16" table may be used, e.g. 1, 17, 33 etc.

AND This function logically "ands" each bit in a source matrix with its corresponding bit in a second (destination) matrix, if both bits are a "1", the bit in the destination matrix will be set to a "1". if either or both bits are a "0", the bit in the destination matrix will be cleared to a "0". A truth table and function block are shown in the figures below.

Figure 9-65 AND Truth Table

Bit 1 Source	Bit 1 Destination Before Operation	Bit 1 Destination After Operation
0	0	0
0	1	0
1	0	0
1	1	1

Figure 9-66 AND Function Block



Function Block The top node is the data source and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 1XXXX reference number in a table of input references
the first 3XXXX reference number in a table of input registers
the first 4XXXX reference number in a table of holding registers

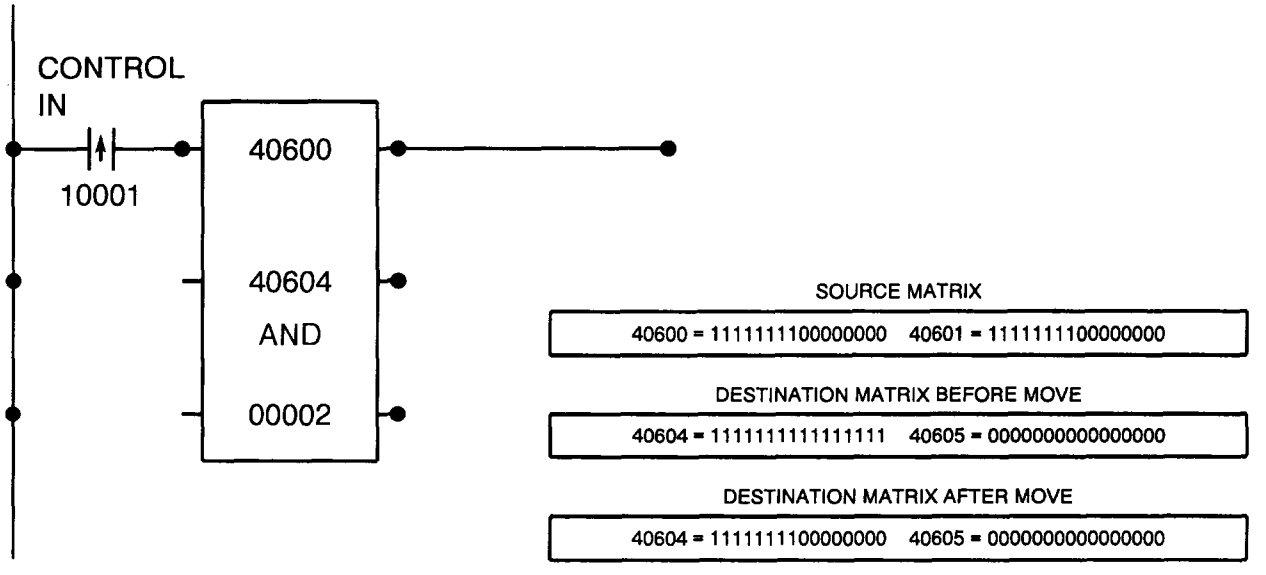
The middle node is the destination and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 4XXXX reference number in a table of holding registers

The bottom node contains the symbol and a number specifying table length. Maximum table length = 100 registers (maximum number of bits in matrix = 1600)

Inputs
Top: Control in. Every scan this node is powered, the AND function is performed. A transitional contact should be used if single operations are desired.

Outputs
Top: Passes power when top input is powered.

Figure 9-67 AND Network Example



Network Description
In the figure above, whenever I0001 passes power, the bit matrix formed by registers 40600 and 40601 will be = ANDed with the bit matrix formed by registers 40604 and 40605. The result of this operation will be copied into registers 40604 and 40605.

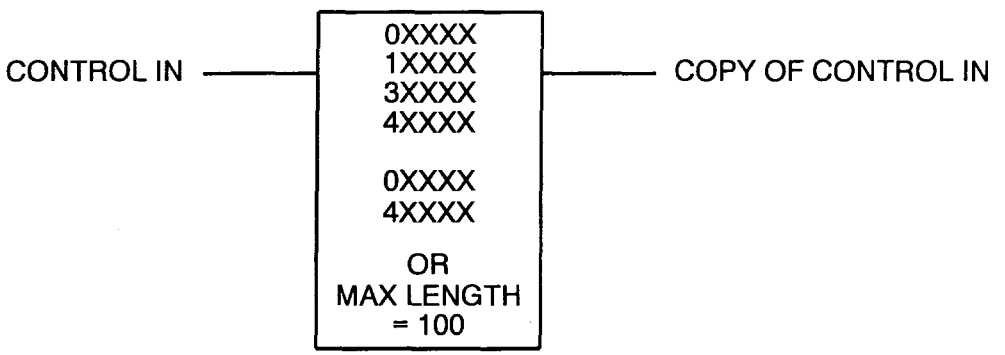
➤➤ **WARNING The destination for this function can be a group of outputs. This function will override any disabled coils within this group without enabling them.**

OR This function logically "ORs" each bit in a source matrix with its corresponding bit in a second (destination) matrix. If either or both bits is a "1", the bit in the destination matrix will be set to a "1". If both bits are a "0", the bit in the destination matrix will be cleared to a "0". A truth table and function block are shown in the figures below.

Figure 9-68 OR Truth Table

Bit 1 Source	Bit 1 Destination Before Operation	Bit 1 Destination After Operation
0	0	0
0	1	1
1	0	1
1	1	1

Figure 9-69 OR Function Block



Function Block The top node is the data source and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 1XXXX reference number in a table of input references
the first 3XXXX reference number in a table of input registers
the first 4XXXX reference number in a table of holding registers

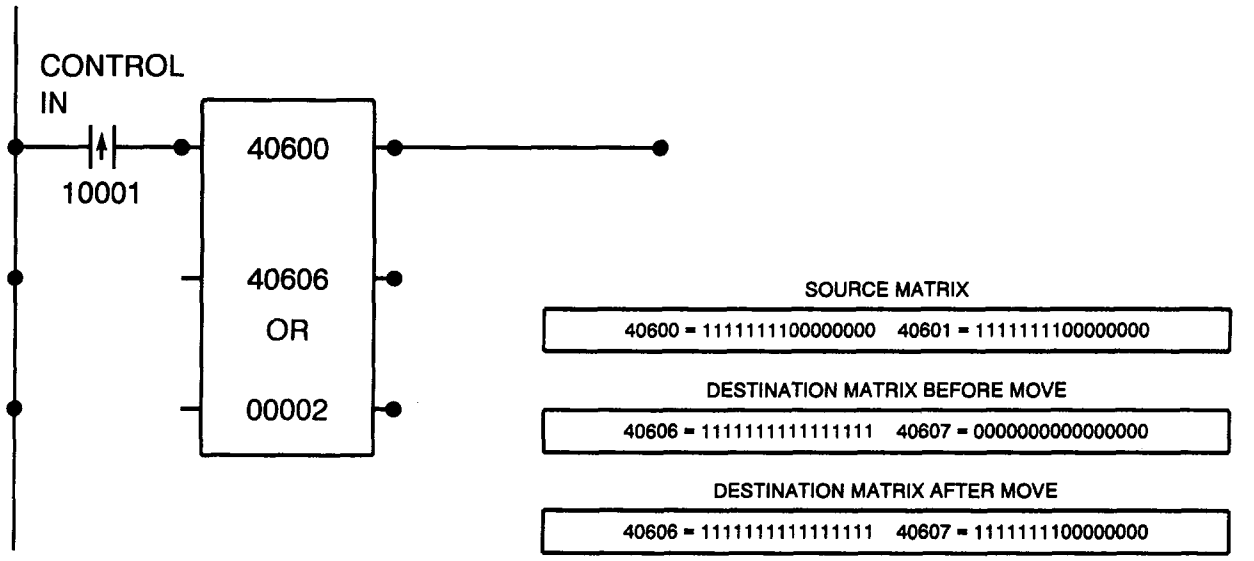
The middle node is the destination and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 4XXXX reference number in a table of holding registers

The bottom node contains the symbol OR and a number specifying table length. Maximum table length = 100 (maximum number of bits in matrix = 1600)

Inputs
Top: Control in. Every scan this node is powered, the OR function is performed. A transitional contact should be used if single operations are desired.

Outputs
Top: Passes power when top input is powered.

Figure 9-70 OR Network Example



Network Description
In the figure above, whenever 10001 passes power, the bit matrix formed by registers 40600 and 40601 will be ORed with the bit matrix formed by 40606 and 40607. The result of this operation will be copied into registers 40606 and 40607.

➤➤ **WARNING The destination for this function can be a group of outputs. This function will override any disabled coils within this group without enabling them.**

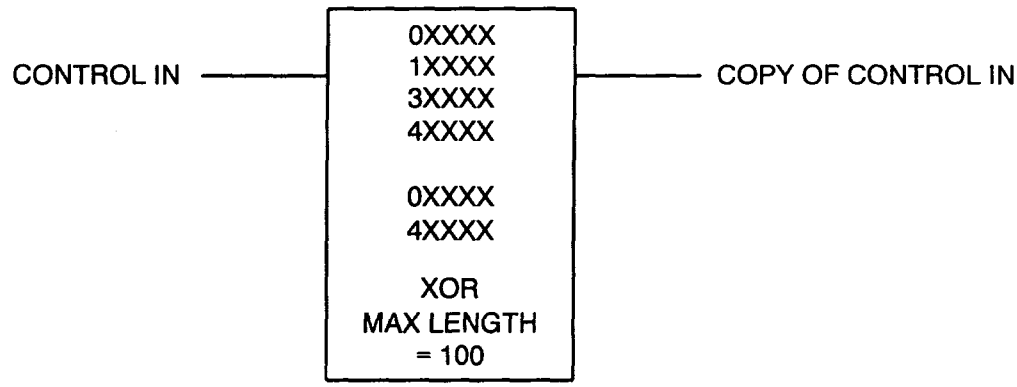
➤ **CAUTION Outputs cannot be turned off with this function.**

Exclusive OR This function logically "XORs" each bit in a source matrix with its corresponding bit in a second (destination) matrix. If either bit is a "1", the bit in the destination matrix will be set to a "I". If both bits are a "1" or a "0", the bit in the destination matrix will be cleared to a "0". A truth table and function block are shown in the figures below.

Figure 9-71 Exclusive OR Truth Table

Bit 1 Source	Bit 1 Destination Before Operation	Bit 1 Destination After Operation
0	0	0
0	1	1
1	0	1
1	1	0

Figure 9-72 XOR Function Block



Function Block The top node is the data source and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 1XXXX reference number in a table of input references
the first 3XXXX reference number in a table of input registers
the first 4XXXX reference number in a table of holding registers

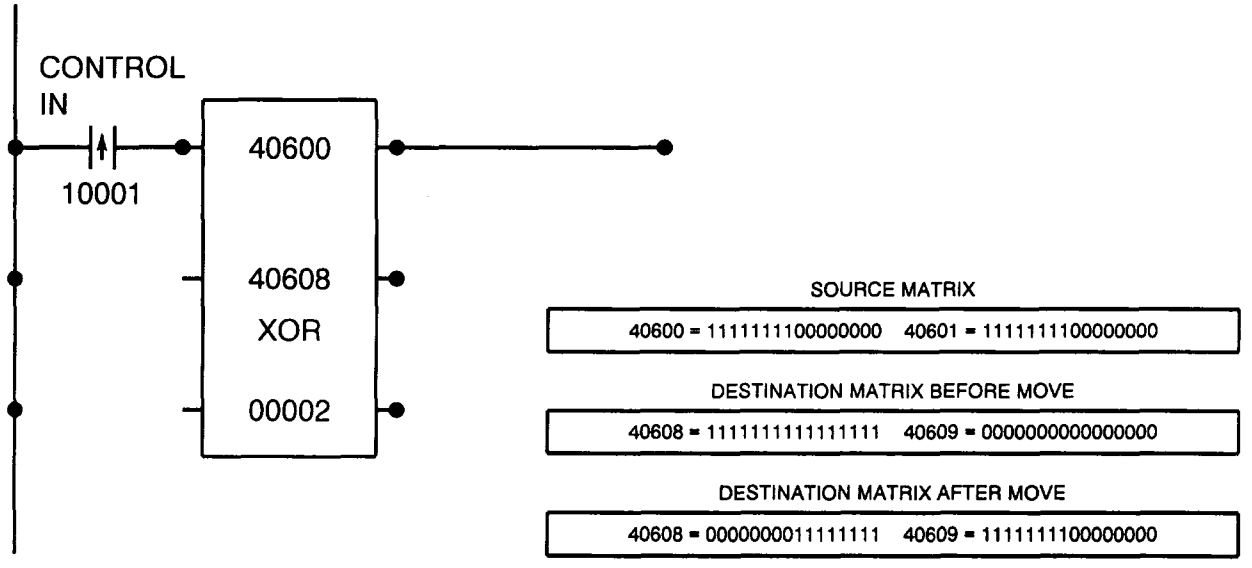
The middle node is the destination and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 4XXXX reference number in a table of holding registers

The bottom node contains the symbol XOR and a number specifying table length. Maximum table length = 100 (maximum number of bits in matrix = 1600)

Inputs
Top: Control in. Every scan this node is powered, the XOR function is performed. A transitional contact should be used if single operations are desired.

Outputs
Top: Passes power when top input is powered.

Figure 9-73 XOR Network Example

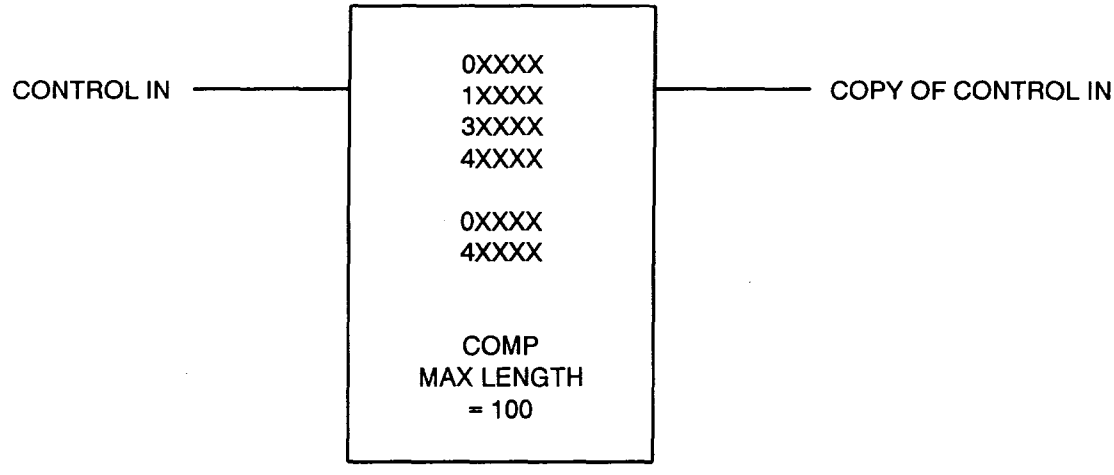


Network Description In the figure below, whenever 10001 passes power, the bit matrix formed by registers 40600 and 40601 will be exclusive ORed with the bit matrix formed by 40608 and 40609. The result of this operation will be copied into registers 40608 and 40609.

➤➤ **WARNING The destination for this function can be a group of outputs. This function will override any disabled coils within this group without enabling them.**

Complement This function copies the complemented or inverted bit pattern (all "1's" are replaced by "0's", all "0's" are replaced by "1's") of one matrix into a second matrix.

Figure 9-74 Complement Function Block



Function Block The top node is the data source and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 1XXXX reference number in a table of input references
the first 3XXXX reference number in a table of input registers
the first 4XXXX reference number in a table of holding registers

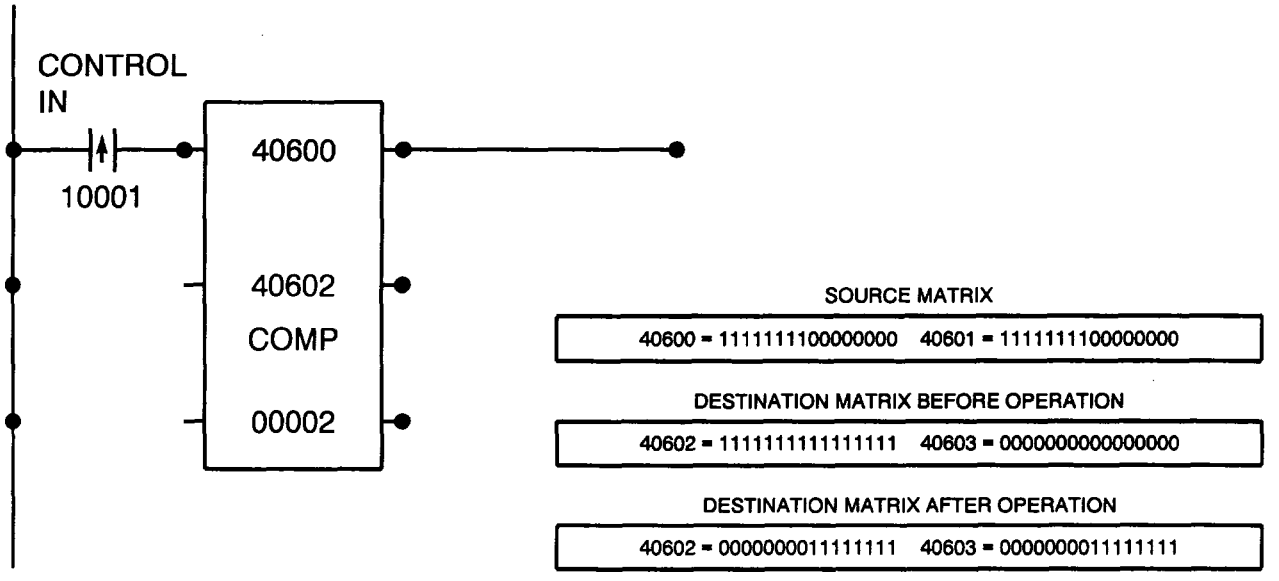
The middle node is the destination and can be one the following:
the first 0XXXX reference number in a table of output references
the first 4xxxx reference number in a table of holding registers

The bottom node contains the symbol COMP and number specifying table length. Maximum table length = 100 (maximum number of bits in matrix = 1600)

Inputs
Top: Control in. Every scan this node is powered, the complement function is performed. A transitional contact should be used if single operations are desired.

Outputs
Top: Passes power when top input is powered.

Figure 9-75 COMP Network Example

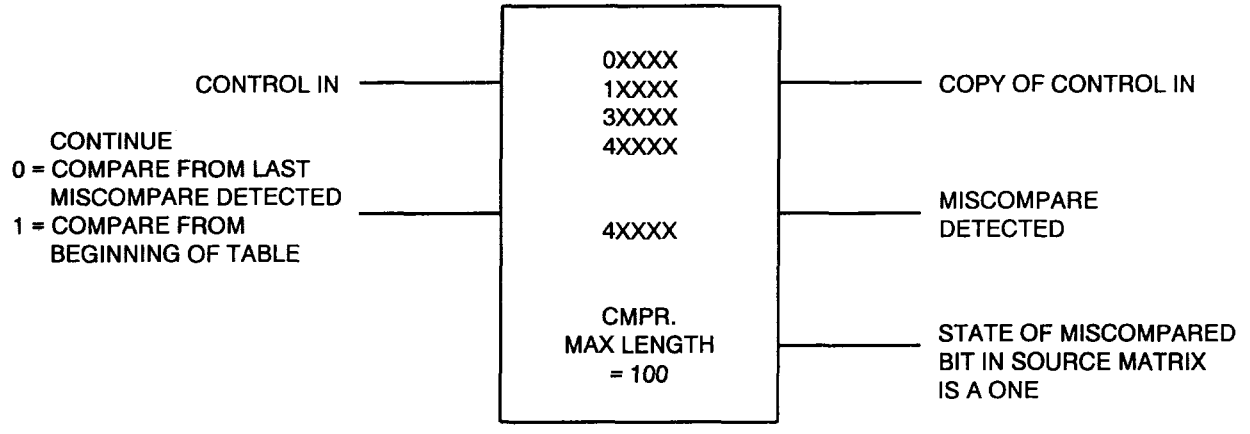


Network Description In the figure above, whenever 10001 passes power, the complement of the bit matrix formed by registers 40600 and 40601 will be copied into a second matrix formed by registers 40602 and 40603.

➤➤ **WARNING The destination for this function can be a group of outputs. This function will override any disabled coils within this group without enabling them.**

Compare This function compares the bit pattern of one matrix against the bit pattern of a second matrix for discrepancies.

Figure 9-76 CMPR Function Block



Function Block The top node is the data source (matrix "A") and can be one of the following:
the first 0XXXX reference number in a table of output references
the first 1XXXX reference number in a table of input references
the first 3XXXX reference number in a table of input registers
the first 4XXXX reference number in a table of holding registers

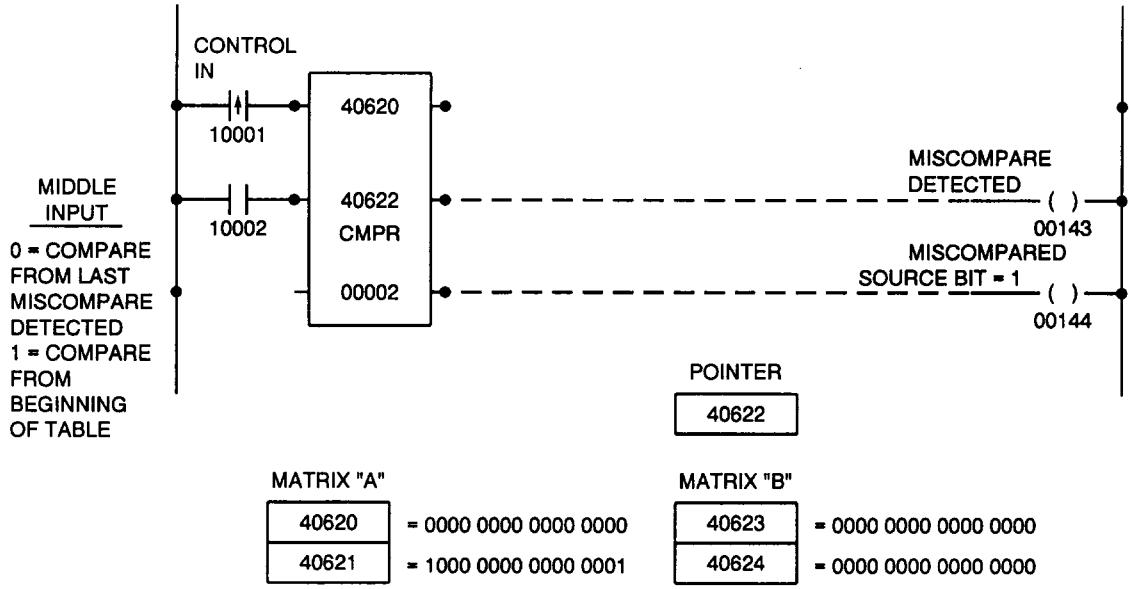
The middle node must be a holding register (4XXXX). This register is the pointer. It will display the miscompared bit number. The next consecutive register (4XXXX +1) is the start of the table which the source table will be compared against (matrix "B").

The bottom node contains the symbol CMPR and a number specifying table length. Maximum table length = 100 (maximum number of bits in matrix = 1600).

Inputs
Top: Control in. Every scan this node is powered, the compare function is performed. A transitional contact should be used if single operations are desired.
Middle: When powered, the search for miscompares starts at the beginning of the matrices. When not powered, the search for miscompares starts with the next bit after the previous miscompare.

Outputs
Top: Passes power when top input is powered.
Middle: Passes power when a miscompare is detected and the top input is powered.
Bottom: Passes power if the miscompared bit = "1" in the source (matrix "A") and the top input is powered.

Figure 9-77 CMPR Network Example



Network Description If 10002 is energized, then every scan the "Control In" node receives power, matrix "A" will be compared against matrix "B", which has all bits cleared to "0". This comparison is done bit-by-bit. If a bit in matrix "A" = a bit in matrix "B", both "0's" or both "1's", then the next two bits are compared. In the network above, this bit-by-bit comparison would continue until bit 17. At this point, because the two bits are not the same (matrix "A" = 1, matrix "B" = "0"):

Pointer register, 40622 = 17 (bit number in matrix)

Energize 00143 for one scan

Energize 00144 for one scan

Next compare at bit #1 of matrices

If 10002 is de-energized, the 1st transition of 10001 will:

Pointer register, 40622 = 17 (bit number in matrix)

Energize 00143 for one scan

Energize 00144 for one scan

Next compare at Bit #18 of matrices

The 2nd transition of 10001:

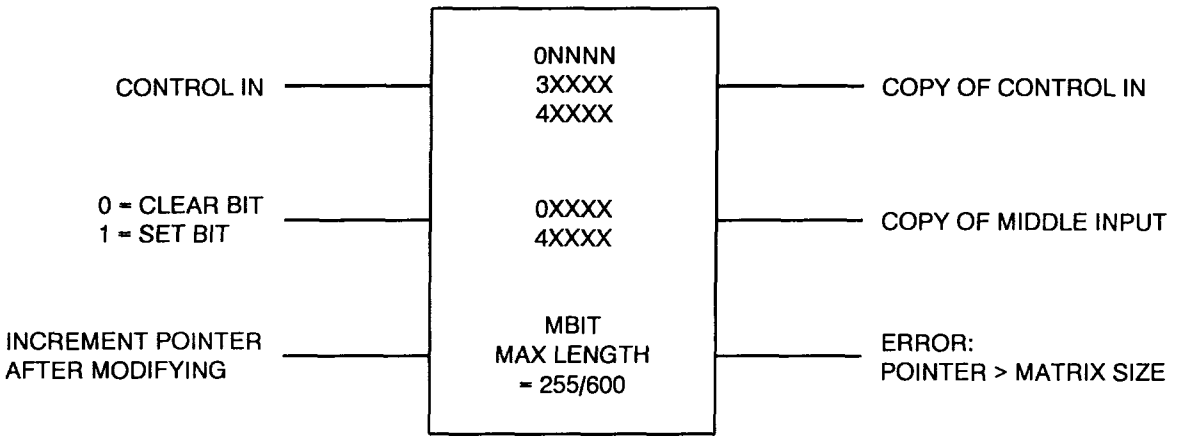
Pointer register, 40623 = 32 (bit number in matrix)

Energize 00143 for one scan

Energize 00144 for one scan

Bit Modify The function alters a specific bit within a matrix.

Figure 9-78 MBIT Function Block



Function Block The top node is the pointer and can be one of the following:
A decimal number ranging from 1 to 4080 in 16 bit CPU's or 1 to 9600 in 24 bit CPU's
An input register (3XXXX)
A holding register (4XXXX)

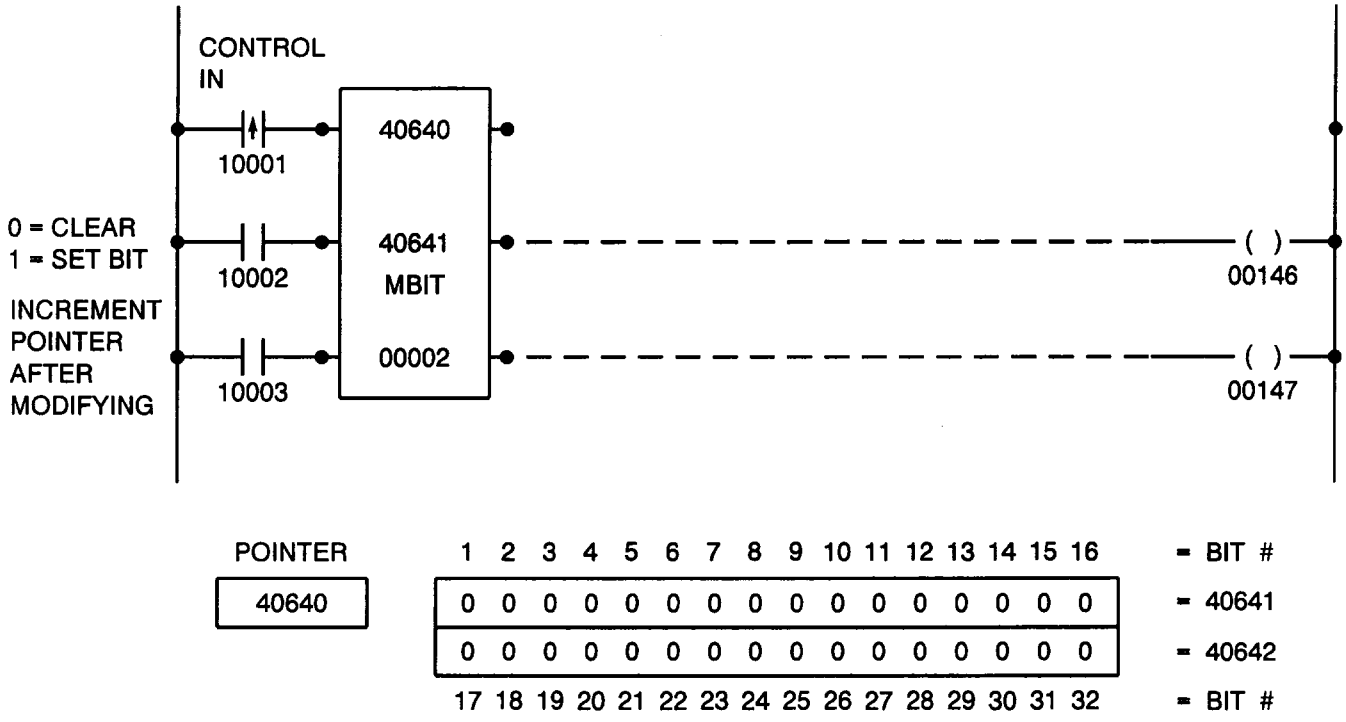
The middle node can be one of the following:
The first 0XXXX reference number in a table of output references
The first 4XXXX reference number in a table of holding registers

The bottom node contains the symbol MBIT and a number specifying table length. Table length may range from:
1 to 255 in 16 bit controllers
1 to 600 in 24 bit controllers

Inputs
Top: Control in. Every scan this node is powered, the bit modify function is performed. A transitional contact should be used if single operations are desired.
Middle: When powered, the "control in" input will cause the bit number specified in the top node to be set to a "1"
When not powered, the "control in" input will cause the bit number specified in the top node to be cleared to a "0"
Bottom: When powered, and if the pointer is a holding register (4XXXX), the pointer will increment each scan the top input receives power.

Outputs
Top: Passes power when top Input is powered
Middle: Passes power when the top and middle inputs are present
Bottom: Error. Passes power if the pointer value is > the matrix size

Figure 9-79 MBIT Network Example



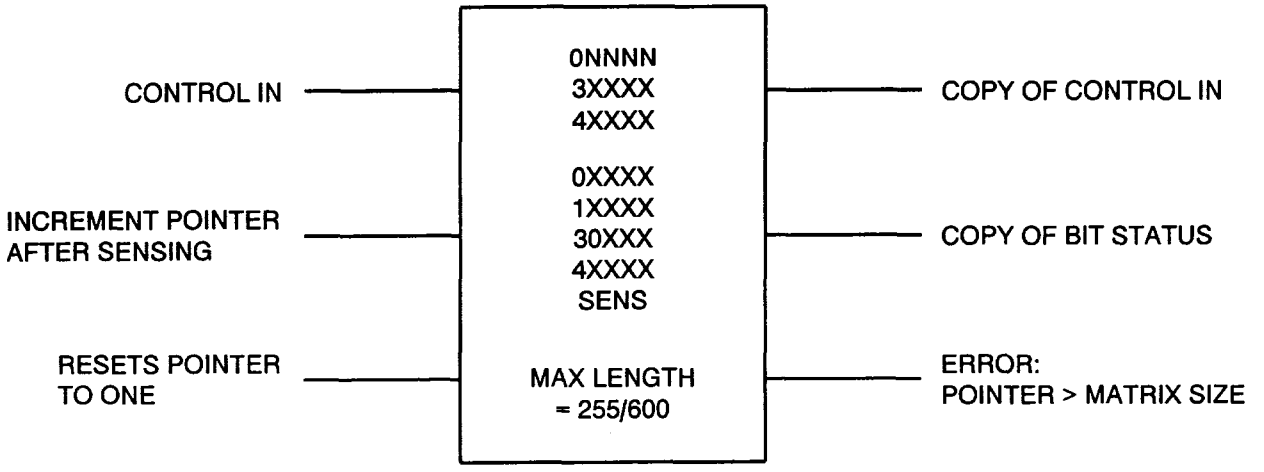
POINTER	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	= BIT #
40640	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 40641
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 40642
	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	= BIT #

Network Description If register 40640 = 16, and 10002 is closed, then, when 10001 transitions from off to on, bit #16 in register 40641 would be set to a "1", and coil 00146 would be energized for one scan. Also, if 10003 had been closed the pointer value in register 40640 would have incremented to 17.

➤➤ **WARNING The destination for this function can be a group of outputs. This function will override any disabled coils within this group without enabling them.**

Bit Sense This function determines the sense, "1" or "0", of a specific bit within a matrix.

Figure 9-80 SENS Function Block



Function Block The top node is the pointer and can be one of the following:
A decimal number ranging (1 to 4080 in 16 bit CPU's or 1 to 9600 in 24 bit CPU's)
An input register (3XXXX)
A holding register (4XXXX)

The middle node can be one of the following:
The first 0XXXX reference number in a table of output references
The first 1XXXX reference number in a table of input references
The first 3XXXX reference number in a table of input registers
The first 4XXXX reference number in a table of holding registers

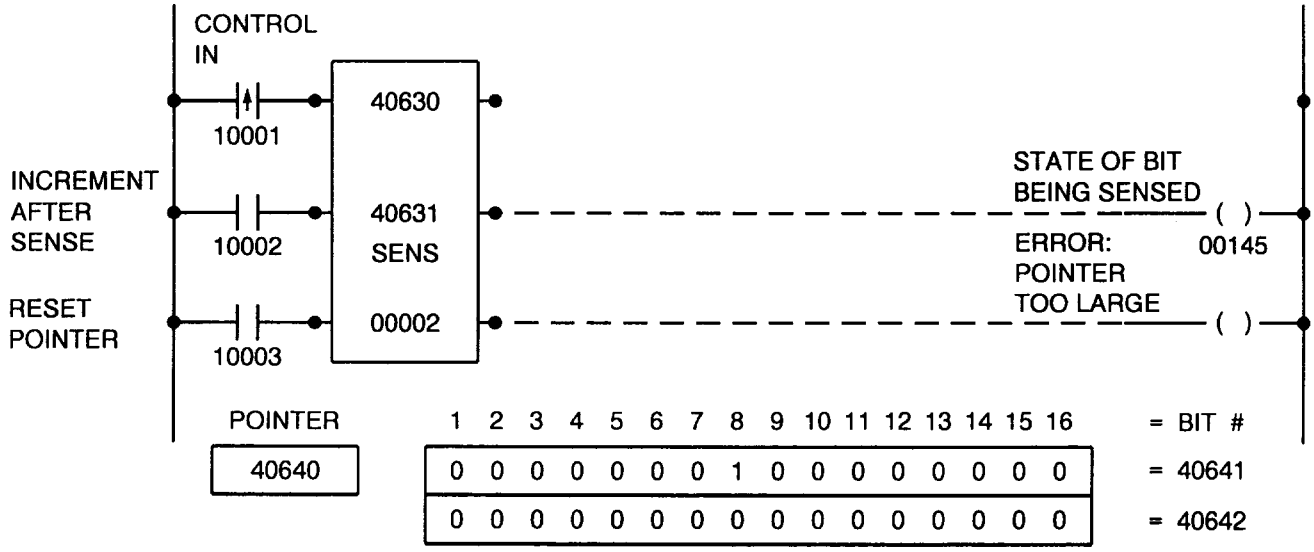
The bottom node contains the symbol SENS and a number specifying table length. Table length may range from:
1 to 255 in 16 bit controllers
1 to 600 in 24 bit controllers

Inputs
Top: Control in
Middle: When powered, and if the pointer is a holding register (4XXXX), the pointer will increment each scan that the top input receives power. If the pointer is incremented to the end of the table, it will automatically reset to one and continue incrementing.
Bottom: When powered, and if the pointer is a holding register (4XXXX), resets pointer value to 1. A transitional contact should be used if a single reset is desired.

Outputs
Top: Passes power when top input is powered
Middle: Passes power if the bit number specified by the pointer value = "1"
Bottom: Passes power if the pointer value is > the matrix size

If the middle node entry specifies coils, the controller erroneously marks those coils as used, and prevents their proper use elsewhere in the user program. If the coil reference is changed to a different coil reference or a register reference, then the original coils are freed for additional use.

Figure 9-81 Bit Sense Network Example



Network Description In the figure above, if register 40630 = 8, then, when 10001 transitions from off to on, coil 00145 will be on for one scan.

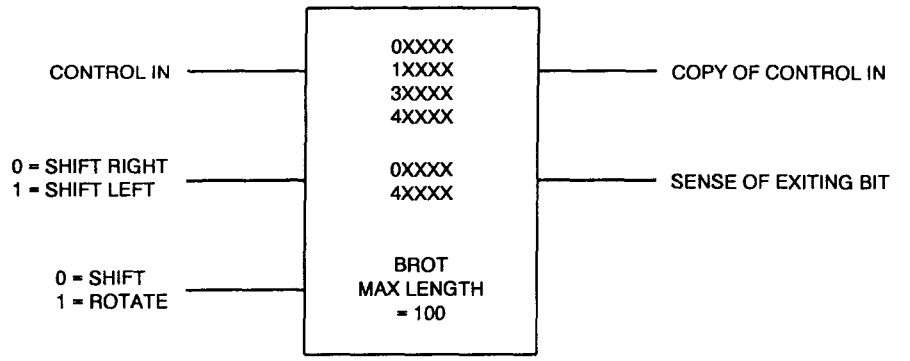
If 10002 had been closed when the above operation was performed, the pointer value in register would have incremented to 9.

If 10003 had been closed when the above operation was performed, the pointer would have been reset to 1.

If 10002 and 10003 had been closed when the above operation was performed, pointer would have been reset to 1, then incremented to 2.

Bit Rotate/Shift This function rotates or shifts the bit pattern of a matrix.

Figure 9-82 BROT Function Block



Function Block The top node can be one of the following:
the first 0XXXX reference number in a table of output references
the first 1XXXX reference number in a table of input references
the first 3XXXX reference number in a table of input registers
the first 4XXXX reference number in a table of holding registers.

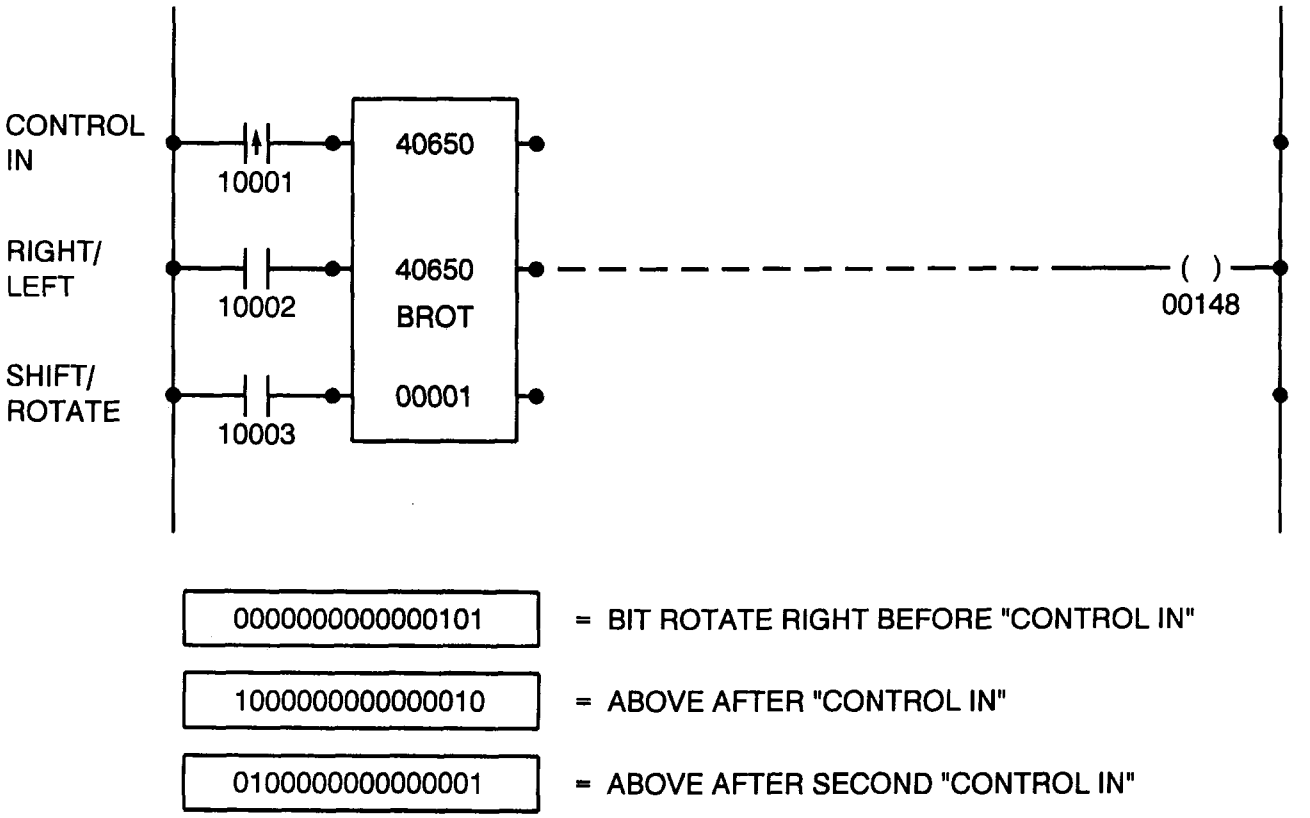
The middle node can be one of the following:
the first 0XXXX reference number in a table of output references (counts as the one time coil can be used) or
the first 4XXXX reference number in a table of holding registers.

The bottom node contains the symbol BROT and a number specifying table length. Maximum table length = 100 (maximum number of bits in matrix = 1600).

Inputs
Top: Control In. Every scan this node is powered, the bit rotate function is performed. A transitional contact should be used if single operations are desired.
Middle: When powered, the Control In input will cause the entire matrix to shift one bit to the left. When not powered, the Control In input will cause the entire matrix to shift one bit to the right.
Bottom: When powered, the Control In input will cause a copy of the exiting bit to be placed at the opposite end of the matrix. When not powered, the Control In input will cause a "0" to be placed at the opposite end of the matrix.

Outputs
Top: Passes power when top input is powered
Middle: Passes power if the exiting bit is a "1".

Figure 9-83 BROT Network Example



Network Description The first transition of 10001 will cause the bit pattern in register 40650 to shift one position to the right. Coil 00148 will be energized since the "exiting" bit will be a "1". 10003 is on when the above operation was performed, the "exiting" bit has "rotated" to the start of the matrix (bit 1). 10002 is off since 40650 register bits are shifting right.

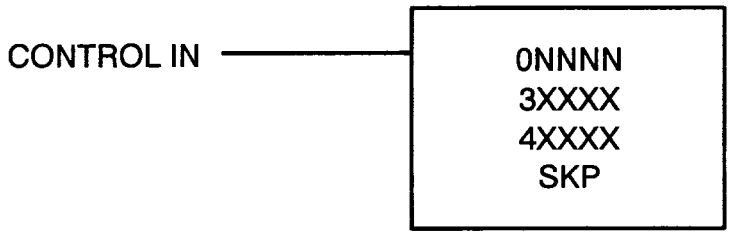
The rotated result of a 16 bit word can be returned to the original registers as in the example above or can be loaded into table of outputs or output registers as shown in the middle node on the preceding page.

➤➤ **WARNING The destination for this function can be a table of outputs (0XXXX). This function will override any disabled coils within this table without enabling them.**

Special Instructions

Skip Block The skip function allows networks to be bypassed and not solved. It can be used to reduce scan time by not solving seldom used program sequences (e.g. emergency stop) or to create sub-routines.

Figure 9-84 SKP Function Block

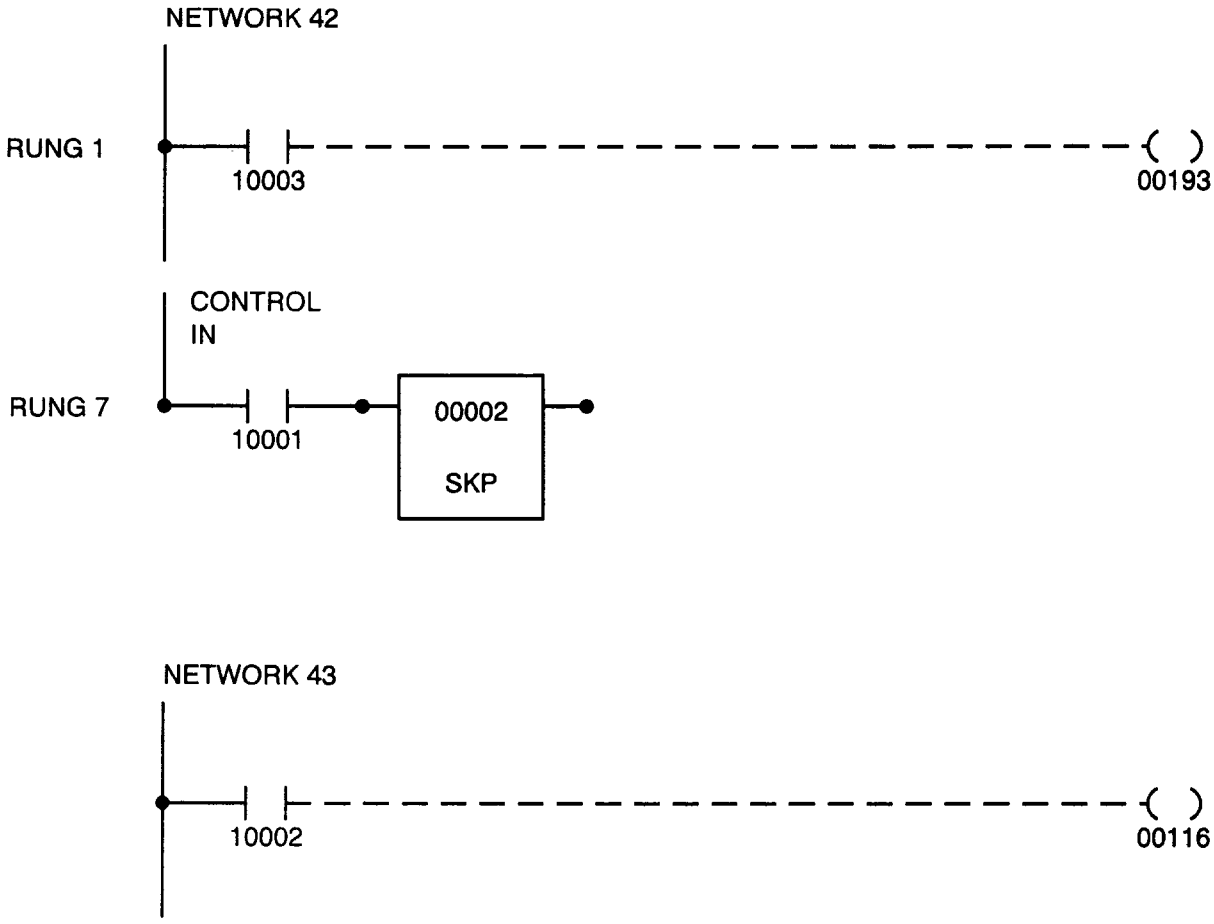


Function Block Occupies one node, and contains the symbol SKP. The number of networks to be skipped is specified here via the assembly register and may be one of the following:
a decimal number ranging from 1 to 999 in 16 bit CPU's or 1 to 9999 in 24 bit CPU's
an input register (3XXXX)
a holding register (4XXXX)

Input
Control in. Every scan this node is powered, the skip function is performed. This causes the remainder of the network containing the skip block to be skipped (counts as one network skipped), and skips additional networks until the total number of networks bypassed equals the value specified in the function block.

Output
There is no output with this function.

Figure 9-85 Skip Block Network Example



Network Description In the figure above, when 10001 is closed the remainder of Network 42 and Network 43 will be skipped. The power flow display for these networks will be invalid, and an information message stating this will appear on the P190 screen.

Note that coil 00193 will still be controlled by contact 10003 since the solution of coil 00193 occurs before the skip block. Coil 00116 will remain in whatever state it was in when the Network 43 was skipped.

The number of networks skipped must include the network in which the block is programmed.
If a value of zero is entered, the remainder of the logic within the segment will be skipped.
Regardless of how many networks are programmed to be skipped, the skip function cannot pass a segment boundary.
Logic within the network containing the skip block, but solved before it, will continue to be solved.
Power flow on the P190 screen is invalid for skipped networks, including the network containing the skip block, even though the logic before it is solved. The message "power display invalid -- network skipped" appears on the P190 screen to prevent misinterpretation of the display.
Skips within skips are not allowed. The controller could stop.
A skip block programmed other than in the 7th rung could cause the controller to stop.

➤➤ **WARNING The skip block is the most dangerous instruction in a programmable controller. Please note the following:**

1. It is critical to remember that changes of inputs, which would normally effect changes in control via the programmed logic, will not occur if the logic associated with these inputs is skipped.

2. Outputs associated with skipped logic will remain frozen in their last state.

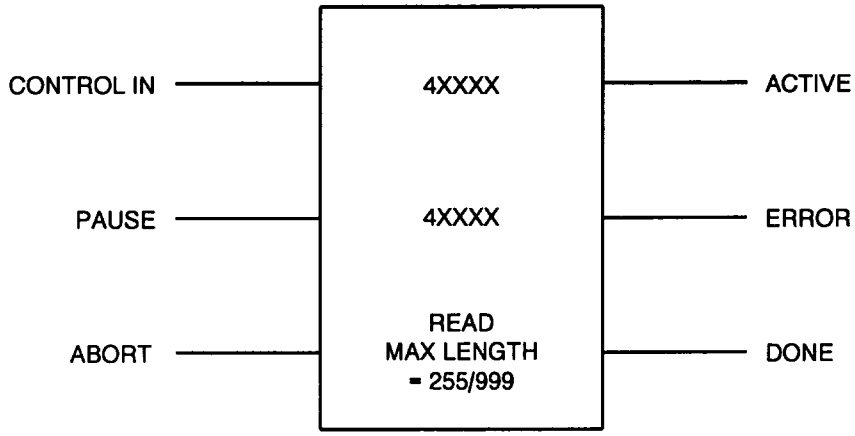
3. Logic controlling safety interlocks, motors, and critical alarm functions, should never be skipped. The results of skipping this type of logic includes overriding safety interlocks, inability of an operator to stop a motor or pump controlled by a hardwired stop button, and ignoring and not reporting critical alarms.

4. Use of registers to control the number of networks skipped must be carefully considered. The contents of a register can be changed by MODBUS, the rap or the user program. This may result in unintentionally skipping logic which must not be skipped or only skipping a portion of logic. In addition, later editing of the program could result in someone using that register for other purposes.

5. If thumbwheel switches are used via a 30XXX input register to control the number of networks skipped, the switches could be read by the 984 while the operator is changing numbers. This could cause results similar to those described in "4" above.

ASCII Read Block This function block provides the ability to read data from an ASCII device into the 984's memory.

Figure 9-86 Read Function Block



Function Block The top node must be a holding register (4XXXX). This register is the first in a tabie of 7 holding registers which are allocated per the following figure.

Figure 9-87 Explanation of Top Node Table Usage

40681	BITS 0-5 = PORT NUMBER (1-32) BITS 6-11 = DROP INTERFACE STATUS BITS BITS 12-15 = HEX ERROR CODE
40682	= MESSAGE NUMBER
40683	= RESERVED FOR 984 USE (# OF REGISTERS TO SATISFY FORMAT)
40684	= RESERVED FOR 984 USE (# OF REGISTERS TRANSMITTED THUS FAR)
40685	= RESERVED FOR 984 USE (STATUS OF SOLVE)
40686	= RESERVED FOR 984 USE (UNASSIGNED)
40687	= RESERVED FOR 984 USE (CHECKSUM OF REGISTERS 1-7)

The middle node must be a holding register (4XXXX). This register is the first in a table of registers whose length is determined by the value in the bottom node. The "variable data" in a message will be written into this table. If, for example, the message was "please enter password: AAAAAAAAA", then the portion of the message which is underlined would be termed "embedded text" and the 10 character ASCII field, AAAAAAAAAA, would be termed "variable data field". Embedded text is stored in memory the same way that user logic is stored, but the variable data must be entered via the ASCII input device (e.g. terminal).

The bottom node contains the symbol READ and a number specifying table length. Table length may range from:
1 to 255 in 16 bit controllers
1 to 999 in 24 bit controllers

Inputs
Top: Control in. The read function is activated when power to this node transitions from a low to a high state and if:
1. No other read or write block is active within the same segment
2. No ASCII device is communicating with the specified port

Once activated, power may be removed from this node, but the block will remain activated for as many scans as necessary to complete the message transaction. Only the abort and pause inputs can stop this function.

Middle: Pause. When powered, deactivates the read function. This allows another read or write block within the same segment to become active. When power is removed, the read function will continue from where it was interrupted if the port was not communicated with. If the port was "talked to" the message will start at the beginning.

Bottom: Abort. When powered:
1. Deactivates the read function
2. The error (middle) output passes power for one scan
3. Load the upper byte of the register specified in the middle node (e.g. 40681) with an error code 6, "user initiated abort".

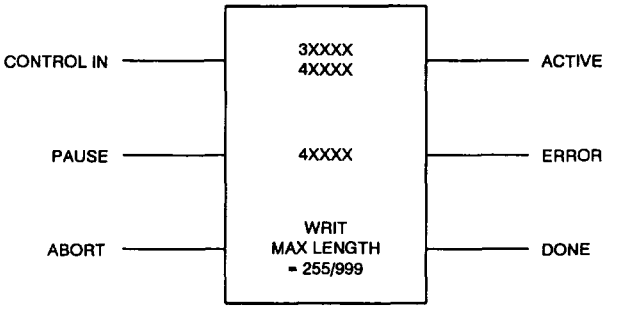
An abort requires that the "control in" be cycled from a low to a high state to restart the read function.

Outputs
Top: Active. Passes power when communications are established with the specified port. Middle: Error. Passes power for one scan whenever an error condition is detected. Loads the 10 most significant bits of the register specified in the middle node with an error code.

Bottom: Done. Passes power for one scan when the read function is successfully completed.

ASCII Write Block This function block provides the ability to send a message from the 984 to an ASCII device.

Figure 9-88 WRIT Function Block



Function Block The top node may be an input register (3XXXX), or a holding register (4XXXX). This register is the first in a table of holding registers whose length is determined by the value in the bottom node. This table will contain the data required to fill in "variable field data" in a message. if, for example, the message to be generated was "Vessel # 1 Temperature is: 132", then the portion of message which is underlined would be termed "Embedded Text" and the 3 digit Integer Field, 132, would be termed a "Variable Data Field". Embedded text is stored in memory the same way that user logic is stored, but the variable data is loaded, typically via DX moves, into a table of variable field data.

The middle node must be a holding register (4XXXX). This register is the first in a table of 7 holding registers which are allocated per the following figure.

Figure 9-89 Explanation of Middle Node Table Usage

40671	BITS 0-5 = PORT NUMBER (1-32) BITS 6-11 = DROP INTERFACE STATUS BITS BITS 12-15 = HEX ERROR CODE
40672	= MESSAGE NUMBER
40673	= RESERVED FOR 984 USE (# OF REGISTERS TO SATISFY FORMAT)
40674	= RESERVED FOR 984 USE (# OF REGISTERS TRANSMITTED THUS FAR)
40675	= RESERVED FOR 984 USE (STATUS OF SOLVE)
40676	= RESERVED FOR 984 USE (UNASSIGNED)
40677	= RESERVED FOR 984 USE (CHECKSUM OF REGISTERS 1-7)

The bottom node contains the symbol WRIT and a number specifying table length. Table length may range from:
1 to 255 in 16 bit controllers
1 to 999 in 24 bit controllers

Inputs

Top: Control in. The write function is activated when power to this node transitions from a low to a high state and if:
1. No other read or write block is active within the same segment
2. No ASCII device is communicating with the specified port

Once activated, power may be removed from this node, but the block will remain activated for as many scans as necessary to complete the message transmission. Only the abort and pause inputs can stop this function.

Middle: Pause. When powered, deactivates the write function. This allows another read or write block within the same segment to become active. When power is removed, the write function will continue from where it was interrupted if the port was not communicated with. If the port was "talked to" the message will start at the beginning.

Bottom: Abort. When powered:
1. Deactivates the write function
2. The error (middle) output passes power for one scan
3. Loads the upper byte of the register specified in the middle node (e.g. 40671) with an error code 6,"user initiated abort".
An abort requires that the "control in" be cycled from a low to a high state to restart the write function.

Outputs

Top: Active. Passes power when communications are established with the specified port.
Middle: Error. Passes power for one scan whenever an error condition is detected. Loads the 10 most significant bits of the register specified in the middle node with an error code.
Bottom: Done. Passes power for one scan when the write function is successfully completed.

Simple ASCII Overview

Used to input brief ASCII messages into the controller (984B only) and output messages to data terminal equipment (using a 984A or B).

Each message can be up to 62 characters in length

Messages are input and output through the ASCII/DAP port

Messages are stored in 4XXXX registers

A 4XXXX register can store 2 ASCII characters

A message is entered into 31 consecutive registers

The number of available 4XXXX registers determines how many messages you can have

Figure 9-90 Summary of Register Assignments

Register	Purpose	Contents
4XXXX	Status/Range	Message status and Number of characters
4XXXX + 1 through 4XXXX + 31	Message Data	2 ASCII characters per register

Separate tables are configured for ASCII message input and output.

Multiple outgoing messages can be stored in tables of 4XXXX holding registers. Each message can be moved to the ASCII message area when needed, using Block Move (BLKM) functions.

Incoming messages can be moved from the ASCII message area to another table of 4XXXX registers for action by the user program. This frees the message area for new messages.

During ASCII input, the Status/Range register's range count is incremented as each pair of ASCII characters is received.

During ASCII output, the Status/Range register's range count is decremented as each pair of ASCII characters is sent.

The Status/Range register's status bit (most significant bit) is set to 0 at the start of the operation. The ASCII device must maintain DTR (Data Terminal Ready), otherwise data transfer stops and the status bit is set to 1.

Simple ASCII Configuration

Two things must be specified in the configuration table of a 984A when configuring for simple ASCII: ASCII port parameters and ASCII output range register. A third parameter, the ASCII input range register, may be specified when using a 984B.

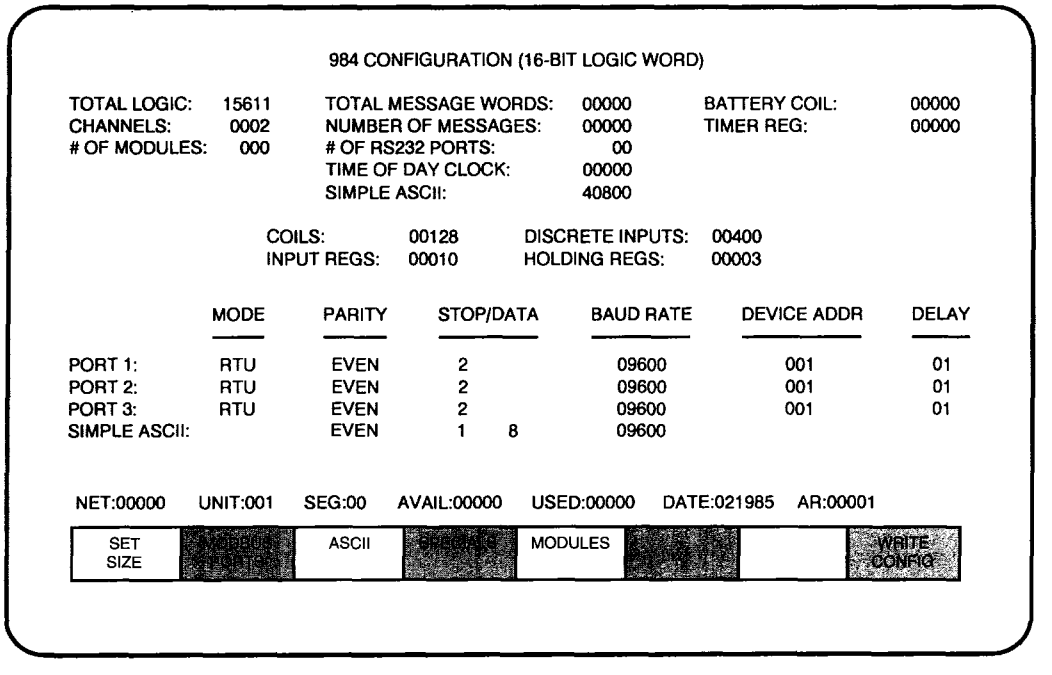
Port Parameters Match the simple ASCII port parameters to those of the data terminal equipment the 984 is transmitting to.

ASCII Input/Output Range Register Specify a 4XXXX register. The system automatically reserves the next consecutive 31 registers for holding up to 62 ASCII characters.

The figure below shows a configuration table (for a 984A) specifying register 40800 as the ASCII output range register; ASCII port parameters of even parity, 1 stop bit, 8 data bits and 9600 baud.

➤ **NOTE** The remainder of this overview will demonstrate how to output a message from a 984A.

Figure 9-91 Configuration Table (984A) ASCII Output Register Specified



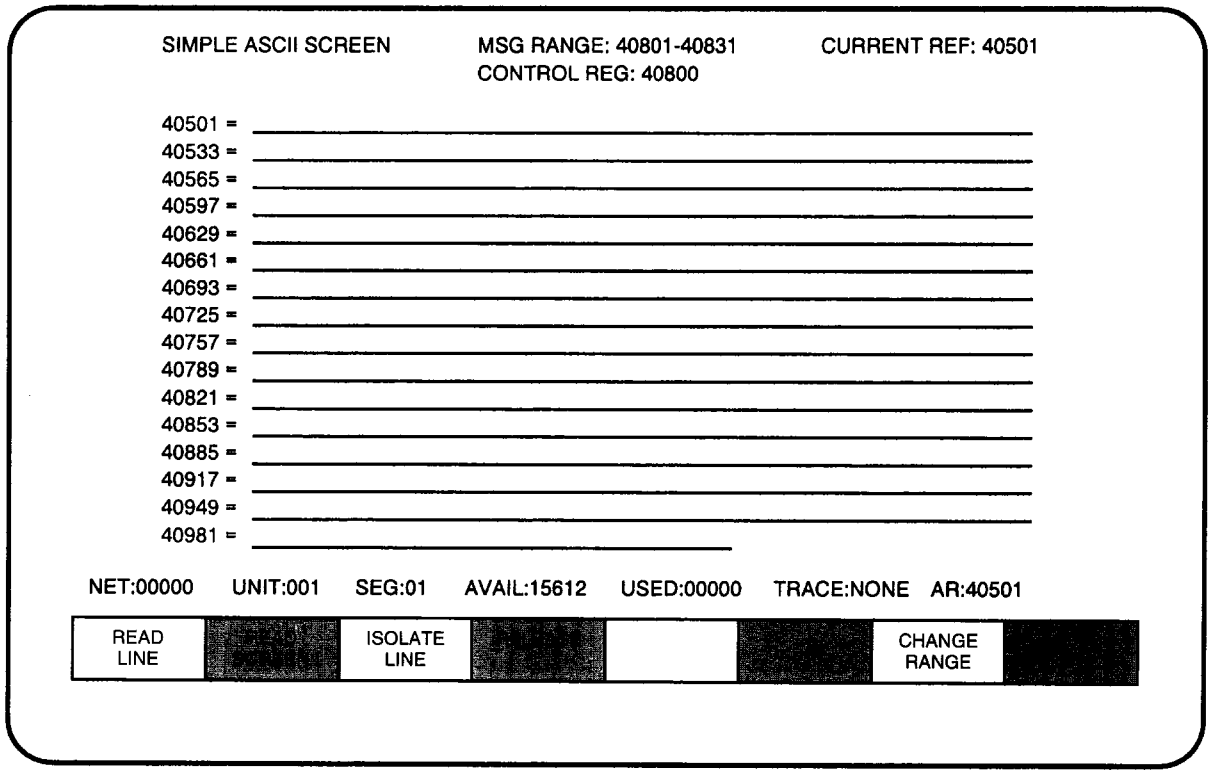
Simple ASCII Screen

The next step is to use a programmer loaded with programming software to go to the exit level. Once there, press CHG SCREEN and select the software label key <ASCII REFERENCE>.

Enter the 4XXXX Register in the AR which is to be the first of 31 sequential registers for holding your output message. You can either call up 1 group of 32 registers at a time by pressing <READ LINE> or sequential groups of 32 by pressing <READ SCREEN>.

In the figure below register 40501 is entered in the AR and the <READ SCREEN> is key pressed. Sixteen groups of 32 registers are displayed. Note that the last group, starting with 40981, only contains 20 registers (40981 to 41000) -- not 32. That's because the system in this example is configured for 1000 4XXXX registers.

Figure 9-92 Sample SIMPLE ASCII Screen



The <ISOLATE LINE> key allows you to highlight 1 particular group of 32 registers for easy reading. Press <RELEASE> to eliminate highlighting.

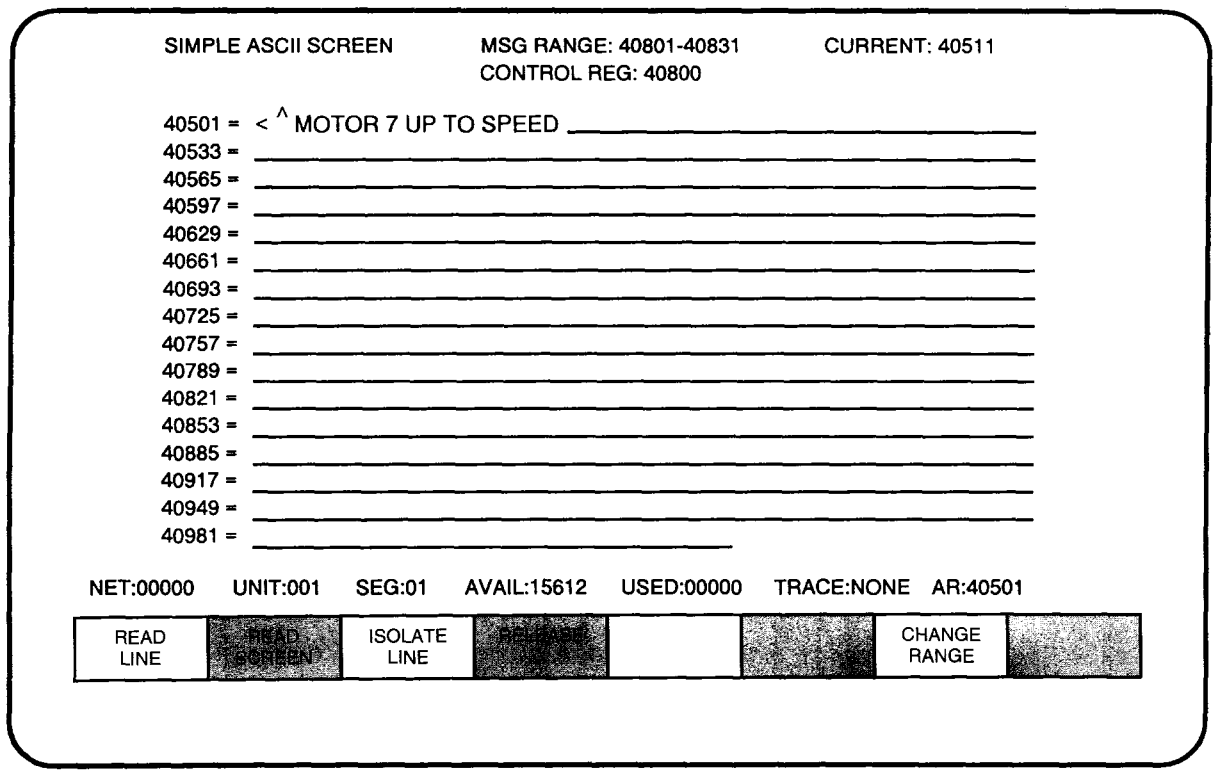
The <CHANGE RANGE> key allows you to change the control register without having to return to the configuration table.

Simple ASCII Message

To enter a message, position the cursor along the line which marks the group of registers you wish to enter the message into. Type the message in just like you would from the keyboard of a typewriter.

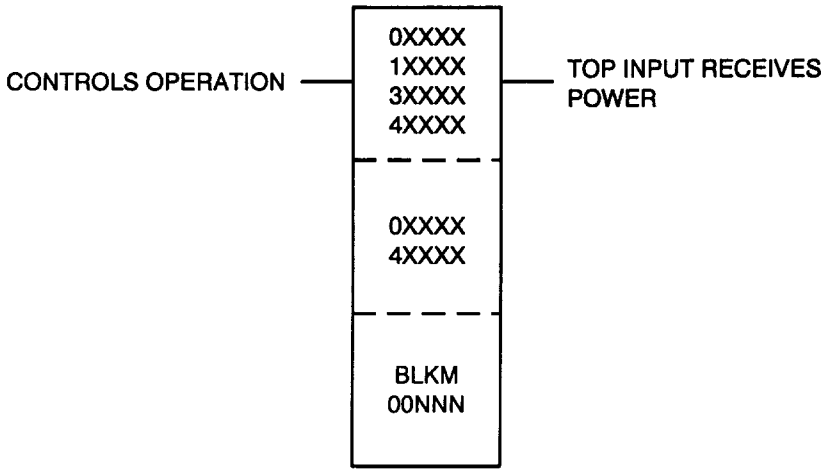
In the figure below, the message "Motor 7 up to speed" is entered into registers 40501-40511. A carriage return and line feed precede the message for formatting on the final printout.

Figure 9-93 Simple ASCII Screen - Sample Message Entered



Block Move It is important to have an understanding of how the Block Move function works, since it is the primary means of outputting messages. The figure below shows the Block Move function block.

Figure 9-94 Block Move Function Block



Top Node This is the source node. It can be one of the following references: 0XXXX, 1XXXX, 3XXXX, or 4XXXX holding register. The source is the first 16 bit word in a sequential table of 16 bit locations.

Middle Node This is the destination node. It can either be a 0XXXX or 4XXXX reference. The destination is the first 16 bit word in a sequential table of 16 bit locations into which the source table is to be moved. Source and destination tables are always the same size.

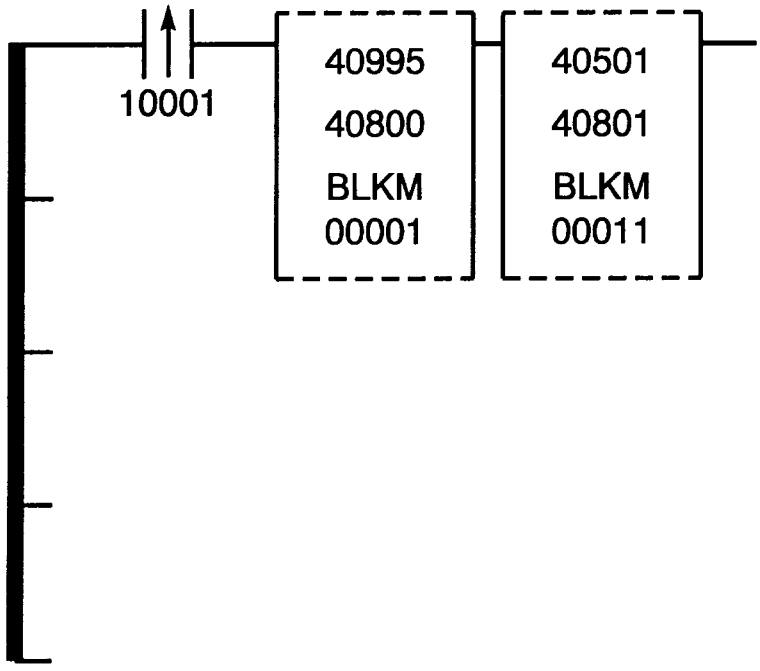
Bottom Node Contains a numerical value that specifies the table length for both the source and destination. This constant can range from 1 to 100 and indicates the number of 16 bit words in each table.

Top Input Controls the operation. When it receives power, the source table is copied into the destination table.

Top Output Passes power when the top input receives power.

Message Output Programming The figure below demonstrates how to output the message MOTOR 7 UP TO SPEED when contact 10001 goes passes power.

Figure 9-95 Sample Message Output Program



First Block Move The function of the first block move is to transfer a value into the output range register representing 1/2 the number of characters in the message and consequently the numbers of registers in the output range table to be loaded and output.

Top Node Register 40995 holds the constant 11, representing 1/2 the number of characters in the message. You, the user, must load a constant of 11 into this register.

Middle Node Register 40800 (control register) is the destination register which will hold the contents of 40995.

Bottom Node Since the source is 1 4XXXX register (40995) and the destination is 1 4XXXX register (40800), a table length of 1 is entered.

NOTE If address 40500 had been used to hold the number of characters in the message, instead of 40995, all of the registers moved as a block would be contiguous. In that case, only a single block move function would be necessary.

Second Block Move The function of the second block move is to transfer a group of 4XXXX registers containing the message into the group of 4XXXX registers reserved for ASCII.

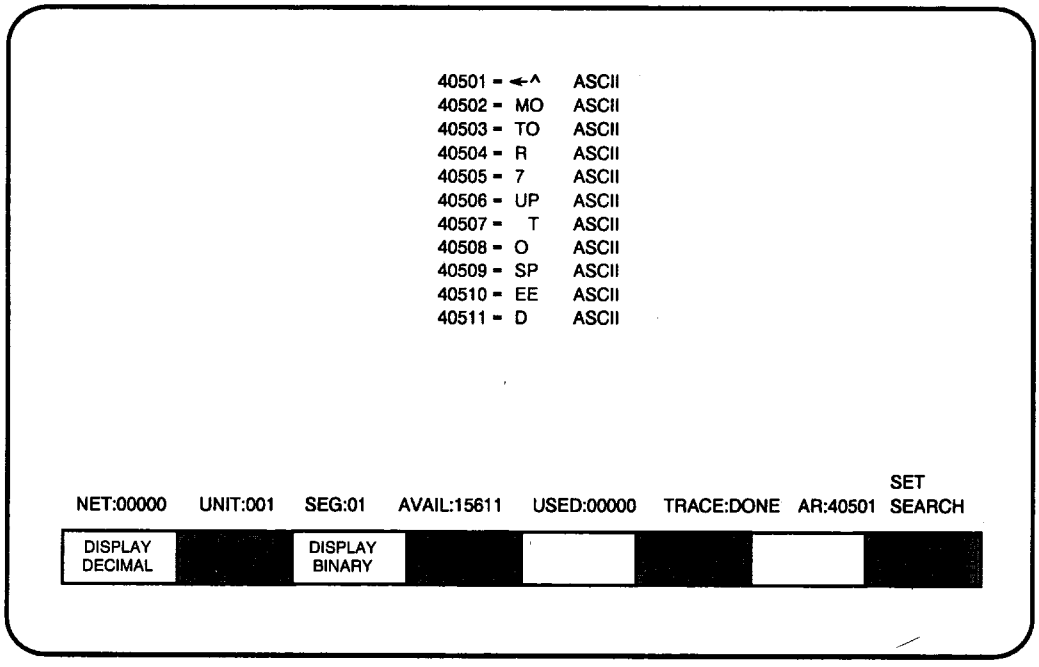
Top Node Register 40501 is the first register in the group of 4XXXX registers containing the message.

Middle Node Register 40801 is the first register in the group of 4XXXX registers reserved for ASCII messages.

Bottom Node Since the source is a group of 11 4XXXX registers (40501 - 40511) and the destination is a group of 11 4XXXX registers (40801 - 40811), an 11 is entered.

Alternate Screen Display ASCII The figure below displays the alternate screen and registers 40502 - 40511 in ASCII.

Figure 9-96 Alternate Screen Display ASCII



Function Blocks

Function blocks are used to program all instructions, except relays and coils.

A function block is inserted into a program by entering the value desired for the top node of the function block into the AR and pressing the appropriate software label key. The function block appears with the desired value in the top node, and question marks in the other node(s), How to edit a function block is explained in the table below.

Table 9-2 Editing a Function Block

Type of Edit	Instructions
To replace the question marks with a value.	1. Position the cursor over the question marks. 2. Enter a value into the AR. 3. Press the ENTER key.
To change a value in a function block.	1. Position the cursor over the value. 2. Enter the new value into the AR. 3. Press the ENTER key.
To change the type of function block.	1. Position the cursor over any node of block. 2. Press the desired software label key.
To delete a function block.	1. Position the cursor over any node of the block. 2. Press the DELETE NODE key.

Networks

Once a whole or partial network has been entered, spacing changes can be made. This section highlights the four types of changes. To make changes in a network:

1. Ensure that the Memory Protect key on the P190 is OFF and that Memory Protect on the 984 is OFF.

2. Display the network needing changes on the P190 screen.

3. Press the EXIT key.

4. Press the EDIT NETWORK software label key.

The following software labels are displayed:

EXPAND HORIZONTAL	EXPAND VERTICAL	COMPRESS HORIZONTAL	COMPRESS VERTICAL

To return to the programming software keys (e.g., RELAYS, COILS, etc.), press the CHG NODE (Change Node) key.

Expand Horizontal This software label key is used to create a column in a network. When the key is pressed, the programming elements at the cursor position, and below, above, and to the right of the cursor, are moved one column to the right.

The logic in the figure below shows a network before pressing the EXPAND HORIZONTAL software label key. The shaded area is the cursor. The revised network is shown in the figure on the next page.

➤ **NOTE** The cursor can be placed anywhere in the column to be created. It does not have to be over an element. There must be a blank column at, or to the right of, the cursor or the EXPAND HORIZONTAL is not allowed.

Figure 9-97 Network Editing Example (Before Pressing EXPAND HORIZONTAL)

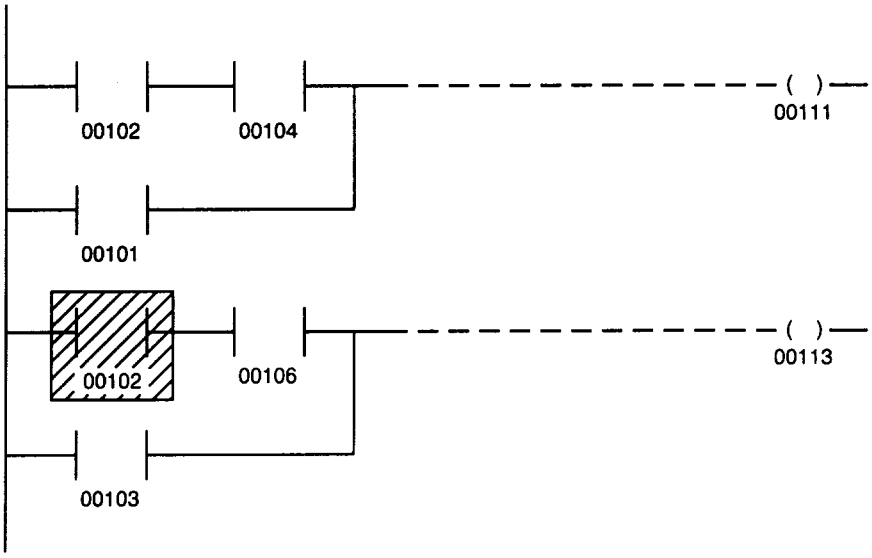
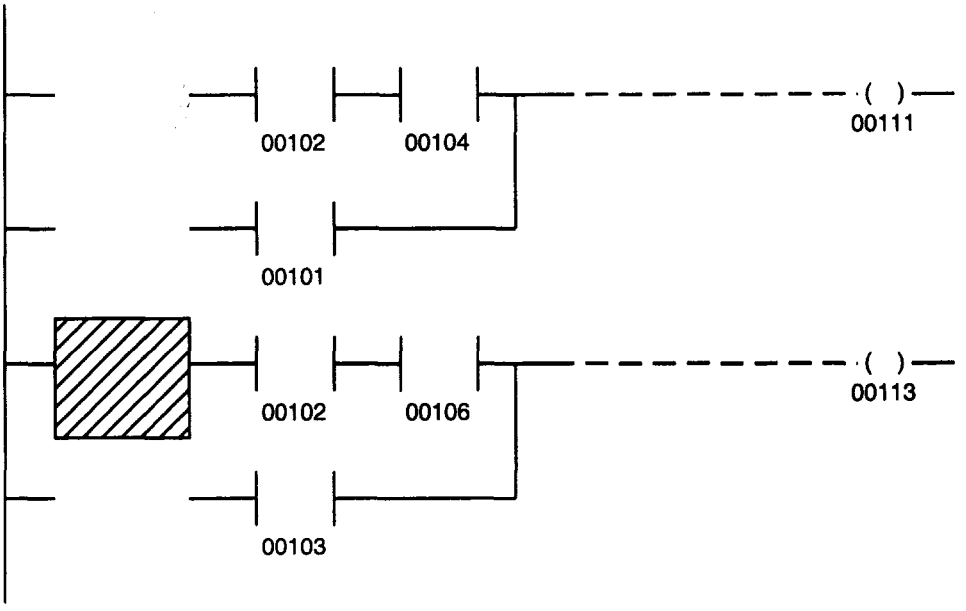


Figure 9-98 Compress Horizontal Example (Possible Result of EXPAND HORIZONTAL)



Compress Horizontal The result of pressing this software label key is the opposite of pressing EXPAND HORIZONTAL. When the key is pressed, the column in which the cursor is located is deleted. The logic in the figure above shows a network before pressing the COMPRESS HORIZONTAL software label key. The revised network is shown in the figure on the previous page.

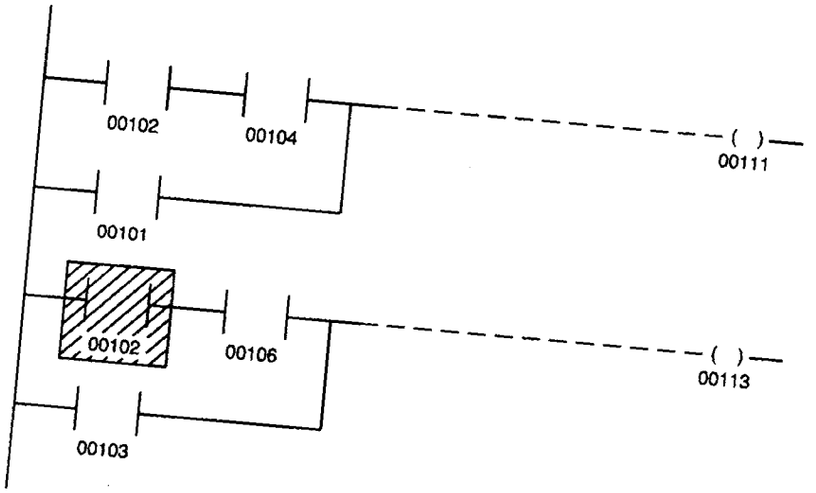
➤ **NOTE** COMPRESS HORIZONTAL is not allowed if the cursor is located over a programming element or if the column to be deleted contains a programming element.

Expand Vertical This software label key is used to create a row in a network. When the key is pressed, the programming elements at the cursor position, and below, to the right, and to the left of the cursor, are moved one row down.

The logic in the next figure shows a network before pressing the EXPAND VERTICAL software label key. the revised network is shown in the last figure on the next page.

➤ **NOTE** The cursor can be placed anywhere in the row to be created. It does not have to be over an element. Also, there must be a blank row under the last row containing programming elements or the EXPAND VERTICAL is not allowed.

Figure 9-99 Expand Vertical Network Example

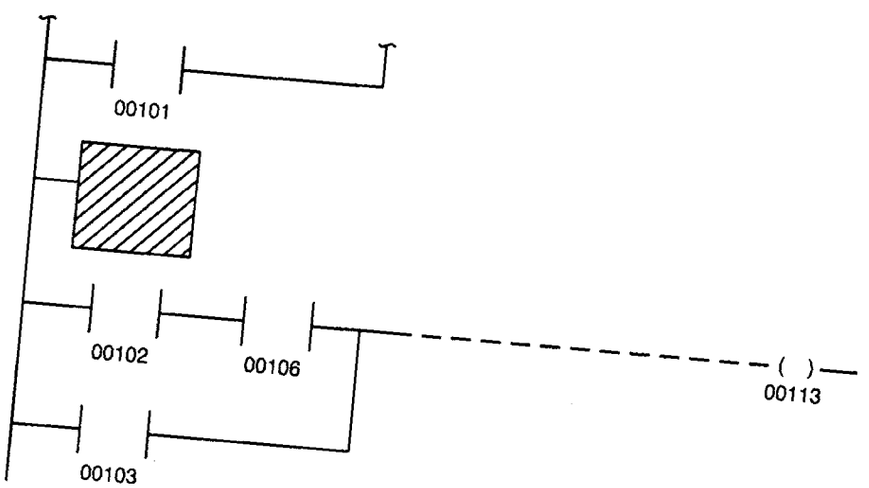


Compress Vertical The result of pressing this software label key is the opposite of pressing EXPAND VERTICAL. When the key is pressed, the row in which the cursor is located is deleted.

The logic in the figure below shows a network before pressing the COMPRESS VERTICAL software label key. The revised network is shown in the figure above.

➤ **NOTE** COMPRESS VERTICAL is not allowed if the cursor is placed over a programming element or if the row to be deleted contains a programming element.

Figure 9-100 Compress Vertical (Possible Result of EXPAND VERTICAL)



984 Optimize vs Run Modes

Run mode allows the user to :

Disable Coils/Inputs

Edit ladder logic while ladder program is being scanned

View power flow

All other typical editing features

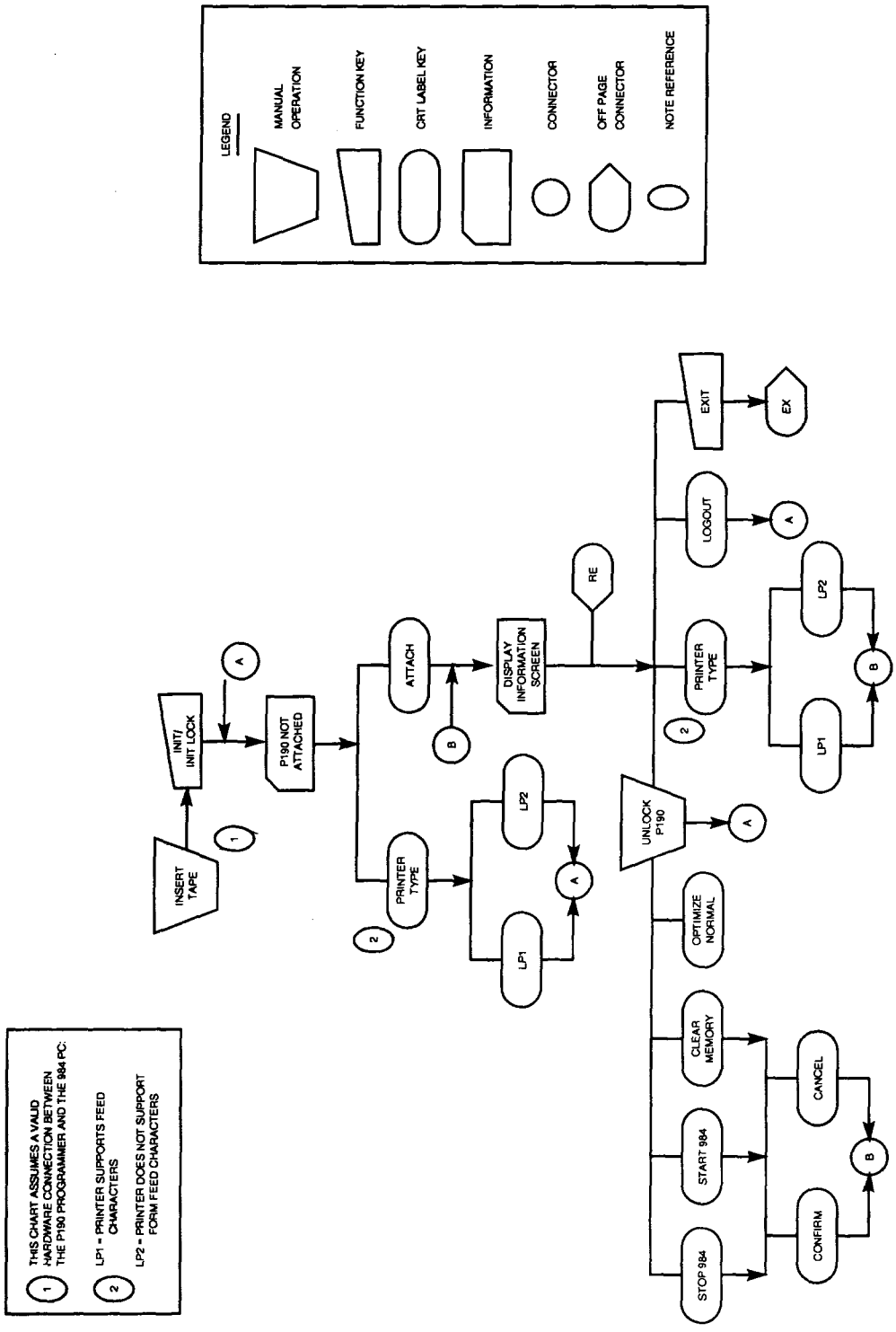
When in the Optimize mode, the user:

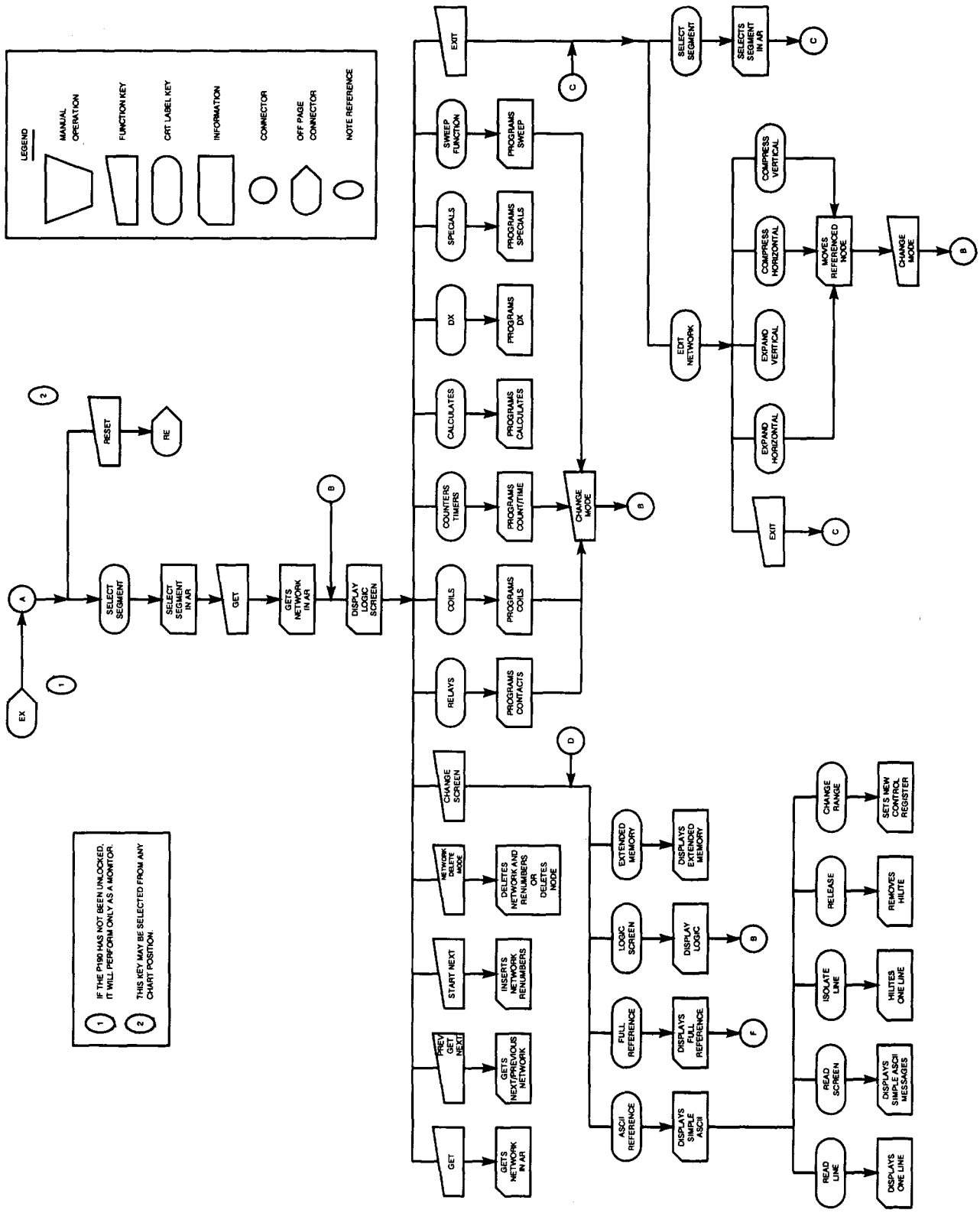
Cannot disable Coils (the controller will not start with any disabled points present)

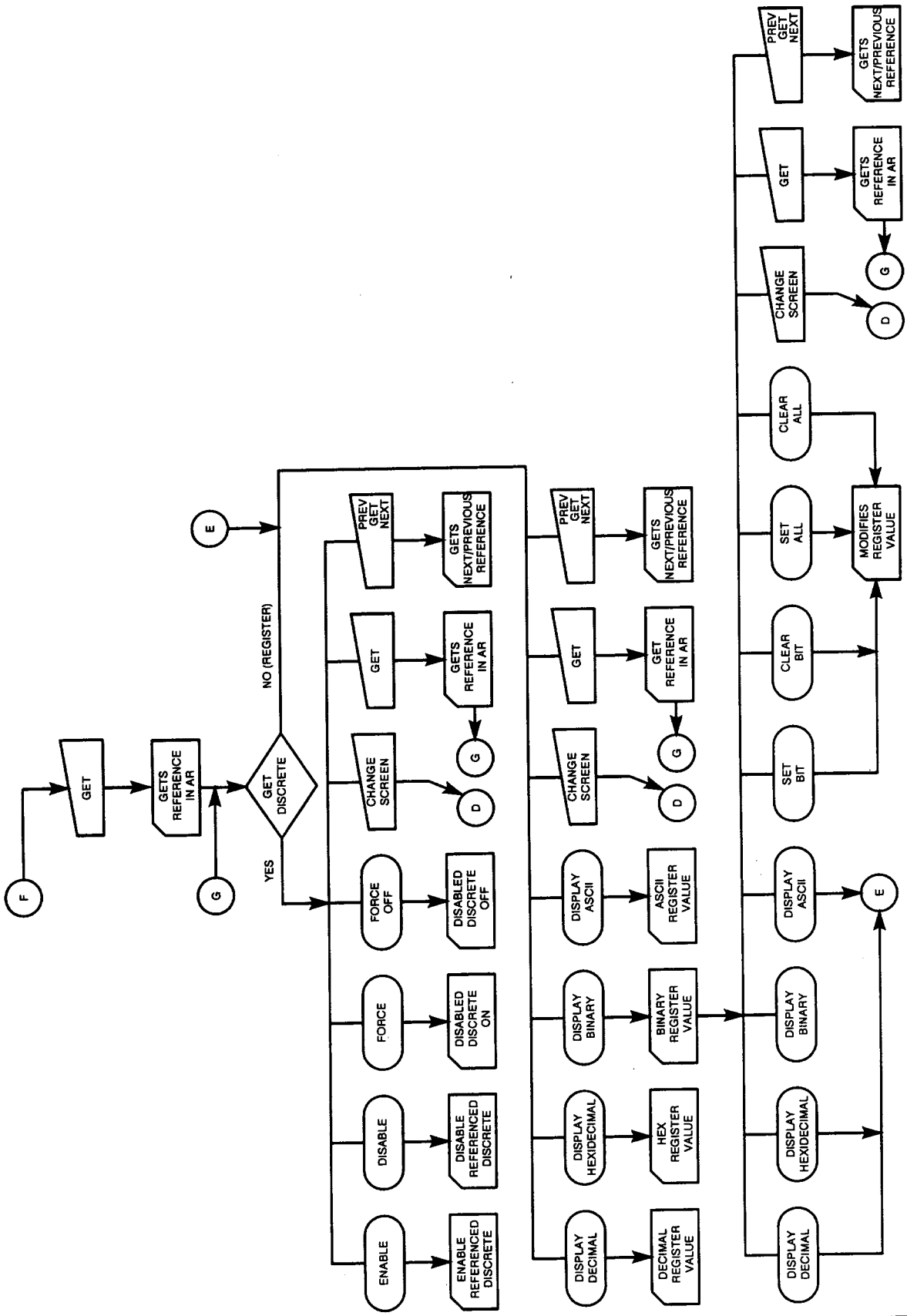
Can view power flow (984-X8X controller only; 984A, 984B, 984X can not view power flow)

In Optimize mode, a 984A, 984B, 984X will have a marginally faster scan time. In a 984-X8X, the scan time is appreciably faster when in Optimize mode.

984 Programmer Tape (Reset Level)




LEGEND
MANUAL OPERATION
FUNCTION KEY
CRT LABEL KEY
INFORMATION
CONNECTOR
OFF PAGE CONNECTOR
NOTE REFERENCE
EX
SELECT SEGMENT
SELECT SEGMENT IN AR
GET
GETS NETWORK IN AR
DISPLAY LOGIC SCREEN
RESET
RE
COILS
PROGRAMS COILS
RELAYS
PROGRAMS CONTACTS
COUNTERS TIMERS
PROGRAMS COUNT/TIME
CALCULATES
PROGRAMS CALCULATES
DX
PROGRAMS DX
SPECIALS
PROGRAMS SPECIALS
SWEEP FUNCTION
PROGRAMS SWEEP
EXIT
CHANGE MODE
EDIT NETWORK
SELECT SEGMENT
SELECTS SEGMENT IN AR
COMPRESS VERTICAL
COMPRESS HORIZONTAL
EXPAND VERTICAL
EXPAND HORIZONTAL
MOVES REFERENCED NODE
CHANGE SCREEN
NETWORK DELETE MODE
DELETES NETWORK AND RENUMBERS OR DELETES NODE
START NEXT
INSERTS NETWORK RENUMBERS
PREV GET NEXT
GETS NEXT/PREVIOUS NETWORK
EXTENDED MEMORY
DISPLAYS EXTENDED MEMORY
LOGIC SCREEN
DISPLAY LOGIC
FULL REFERENCE
DISPLAYS FULL REFERENCE
ASCII REFERENCE
DISPLAYS SIMPLE ASCII
CHANGE RANGE
SETS NEW CONTROL REGISTER
RELEASE
REMOVES HILITE
ISOLATE LINE
HILITES ONE LINE
READ SCREEN
DISPLAYS SIMPLE ASCII MESSAGES
READ LINE
DISPLAYS ONE LINE
1 IF THE P190 HAS NOT BEEN UNLOCKED, IT WILL PERFORM ONLY AS A MONITOR.
2 THIS KEY MAY BE SELECTED FROM ANY CHART POSITION.


F
GET
GETS REFERENCE IN AR
G
GET DISCRETE
NO (REGISTER)
YES
E
ENABLE
ENABLE REFERENCED DISCRETE
DISABLE
DISABLE REFERENCED DISCRETE
FORCE
DISABLED DISCRETE ON
FORCE OFF
DISABLED DISCRETE OFF
CHANGE SCREEN
D
GET
GETS REFERENCE IN AR
G
PREV GET NEXT
GETS NEXT/PREVIOUS REFERENCE
DISPLAY DECIMAL
DECIMAL REGISTER VALUE
DISPLAY HEXIDECIMAL
HEX REGISTER VALUE
DISPLAY BINARY
BINARY REGISTER VALUE
DISPLAY ASCII
ASCII REGISTER VALUE
CHANGE SCREEN
D
GET
GET REFERENCE IN AR
G
PREV GET NEXT
GETS NEXT/PREVIOUS REFERENCE
DISPLAY DECIMAL
DISPLAY HEXIDECIMAL
DISPLAY BINARY
DISPLAY ASCII
SET BIT
CLEAR BIT
SET ALL
CLEAR ALL
MODIFIES REGISTER VALUE
E
CHANGE SCREEN
D
GET
GETS REFERENCE IN AR
G
PREV GET NEXT
GETS NEXT/PREVIOUS REFERENCE