# SECTION 3
# DATA TRANSFER (DX) MOVE FUNCTIONS

The MOVE functions copy data from registers and/or tables into other registers or tables. The data can then be examined or changed by the controller without altering the original data.

A register is a location in the controller's memory in which a numerical value is stored. This value can be binary or binary coded decimal (BCD). In a 584L PC, the maximum decimal value is 9999 and the maximum number of bits is sixteen.

A table is a group of consecutive registers or discretes. The maximum number of registers or groups of discretes in a table is 255 in a Level 1 584L PC or 999 in a Level 2 584L PC.

Each function block (except STAT) occupies three nodes in a 10 x 7 node network format and consists of a source, a destination, and a node specifying table length. The top input is the control input; when it receives power the function is performed. The top output passes power when the top input receives power. This allows function blocks to be cascaded within a network.

The input(s) to a function block can be a single relay contact, another function block, or a whole network of logic. The output(s) can be connected directly to coils, to other function blocks, to relay contacts, or left unconnected.

**NOTE**

If a single move operation is desired, use a transitional contact to control the top input.

The DX function blocks can perform functions with 30XXX input registers, 4XXXX holding registers, and 0XXXX and IXXXX discrete references.

If discrete references are used, remember the following:

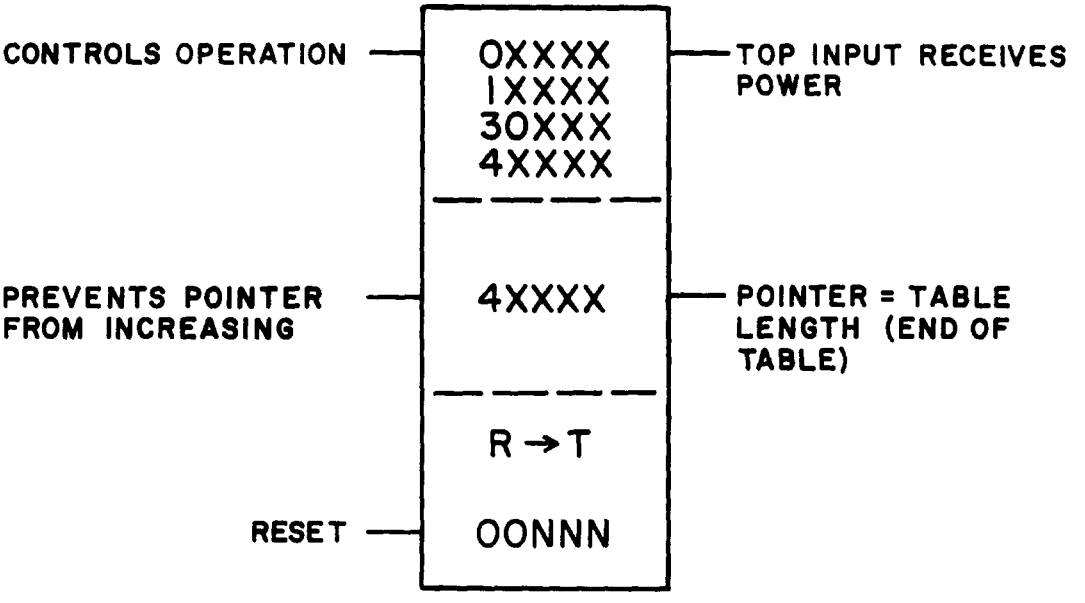
- Discretes are used in groups of sixteen.
- The reference number used is the first in the group; the other fifteen references are implied.
- Only certain reference numbers are valid; the number must be divisable by 16 with a remainder of 1. The valid reference numbers are: 00001, 00017, 00033, etc., and 10001, 10017, 10033, etc.

If a 0XXXX or a 1XXXX reference is used in a DX function block, it cannot be used anywhere else in the program. If discrete references are used to specify table length, the value refers to the number of groups of 16 discretes (i.e., 4 indicates 4 groups or 64 discretes).

## 3.1 REGISTER-TO-TABLE MOVE (R→T)

FUNCTION

The Register-To-Table Move function copies sixteen logic coils, sixteen discrete inputs, one input register, or one holding register into a single specific location within a table of registers.




FUNCTION BLOCK

- The top node is the source node. It can be one of the following references: a 0XXXX logic coil, a 1XXXX discrete input, a 30XXX input register, or a 4XXXX holding register. The source is a single 16 bit location (e.g., a register or group of sixteen discretes).

- The middle node is the destination node. It is a 4XXXX holding register which holds the pointer value. The pointer value plus one indicates the position of a register in a table (i.e., a pointer value of three indicates the fourth position in a table). The table starts at the next register (4XXXX + 1), not at the pointer.

- The bottom node contains the symbol R→T and a numerical value which specifies the table length. This constant can range from 1 to 255 in a Level 1 584L PC or from 1 to 999 in a Level 2 584L PC.

**NOTE**

If the pointer register is loaded with a value greater than the table length, the 584L PC sets the pointer value to the table length when the function block is solved.

INPUTS

- The top input controls the operation. When it is receiving power, the information in the source register is copied into a location in the table.

- The middle input, when receiving power, prevents the pointer from increasing.

- The bottom input, when receiving power, resets the pointer to zero. If the top input is also energized, the source register is copied to the first register in the table.

OUTPUTS

- The top output passes power when the top input receives power.

- The middle output passes power when the pointer equals the table length. This indicates that the table is full (end of table).

EXAMPLE

The following paragraphs provide a detailed explanation of the logic used in Figure 3-1.

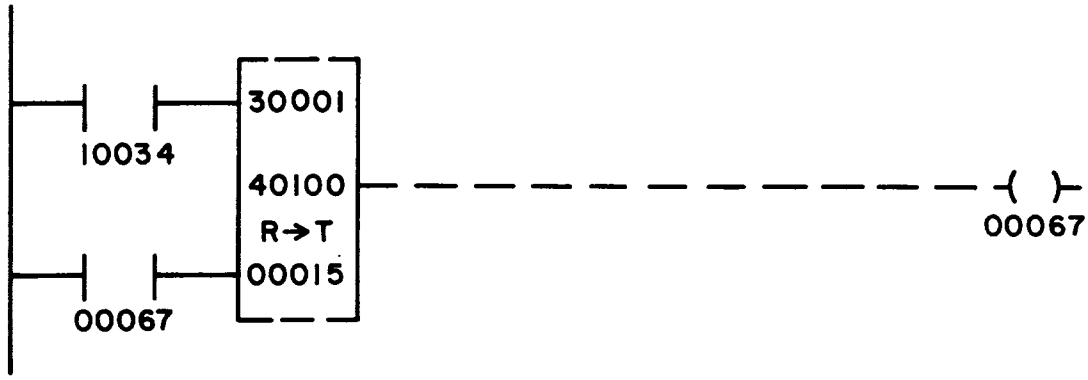



*Figure 3-1. Register-To-Table Move Logic*

When input 10034 is energized, the top input receives power and the content of input register 30001 is moved into table 40101-40115. Data is moved one entry per scan.

If the pointer value equals zero when the top input receives power, the content of register 30001 is moved into register 40101 and the pointer value increases to one.

On the next scan, provided the top input is still receiving power, the content of register 30001 is moved into register 40102 and the pointer value increases to two. Whenever the top input loses power, the move operation stops and the pointer holds its value.

Once the pointer has increased to the (table size as defined in the bottom node (e.g., 15), the middle output passes power and energizes coil 00067. On the next scan, the bottom input receives power because it is referenced to coil 00067. This input resets the pointer to zero, thereby de-energizing coil 00067.

Figure 3-2 is an illustration of the Register-To-Table Move described in the preceding paragraphs.

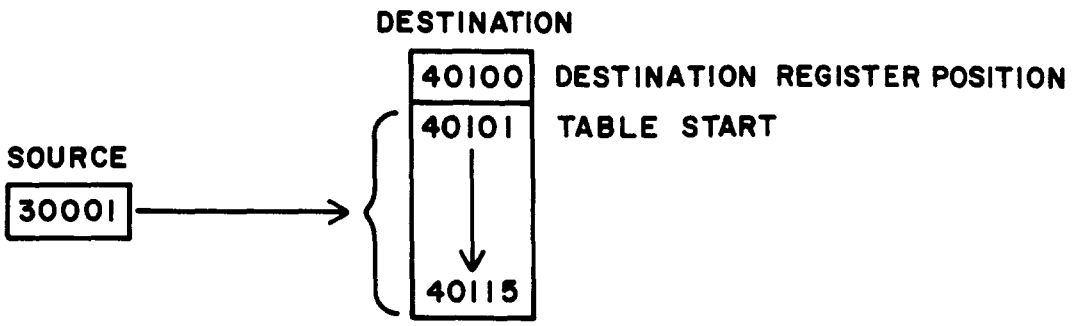
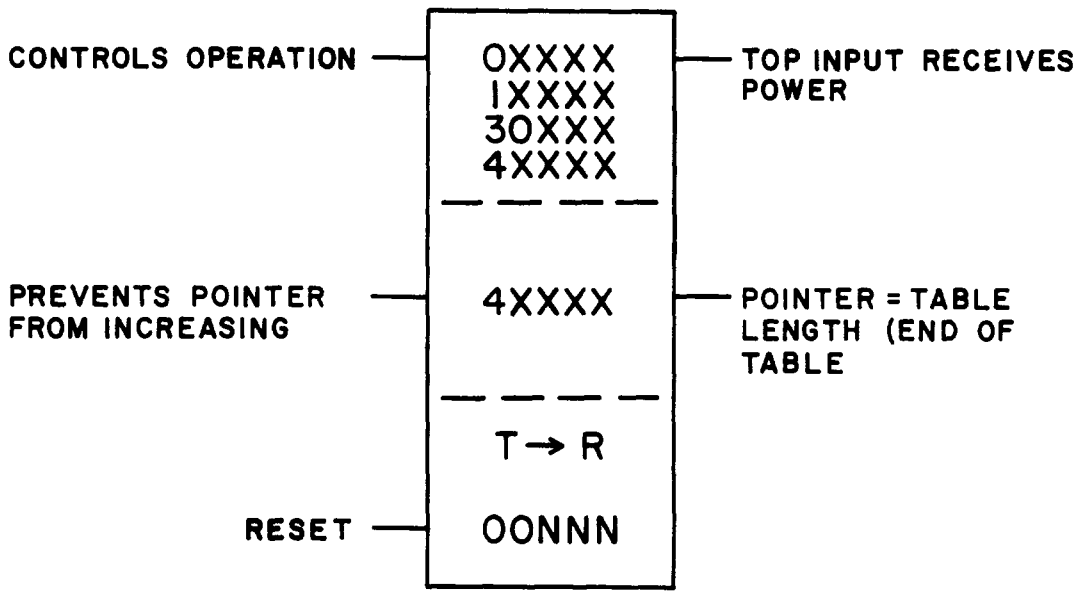



*Figure 3-2. Register-To-Table Move*

## 3.2 TABLE-TO-REGISTER MOVE (T ⟶ R)

FUNCTION

The Table-To-Register Move function copies one register or group of sixteen discretes from a table into a single holding register.




FUNCTION BLOCK

- The top node is the source node. It can be one of the following references: a 0XXXX logic coil, a 1XXXX discrete input, a 30XXX input register, or a 4XXXX holding register. The source is a table of 16 bit locations. Its size is defined in the bottom node.

- The middle node is the destination node. It is a 4XXXX holding register which holds the pointer value. The pointer value plus one indicates the position of a register in a table (i.e., a pointer value of three indicates the fourth position in a table). The next consecutive holding register (4XXXX + 1) receives the data. The pointer does not receive the data.

- The bottom node contains the symbol T→R and the numerical value that specifies the source table length. This constant can range from 1 to 255 in a Level I 584L PC, or from 1 to 999 in a Level 2 584L PC.

**NOTE**

If the pointer register is loaded with a value greater than the table length, the 584L PC sets the pointer value to the table length when the function block is solved.

INPUTS

- The top input controls the operation. When it is receiving power, the information in the source table's register is copied into a single holding register.
- The middle input, when receiving power, prevents the pointer value from increasing.
- The bottom input, when receiving power, resets the pointer to zero. If the top and bottom inputs are energized at the same time, the first register in the table is moved to the destination register.

OUTPUTS

- The top output passes power when the top input receives power.
- The middle output passes power when the pointer value equals the table length (end of the table).

EXAMPLE

The following paragraphs provide a detailed explanation of the logic used in Figure 3-3.

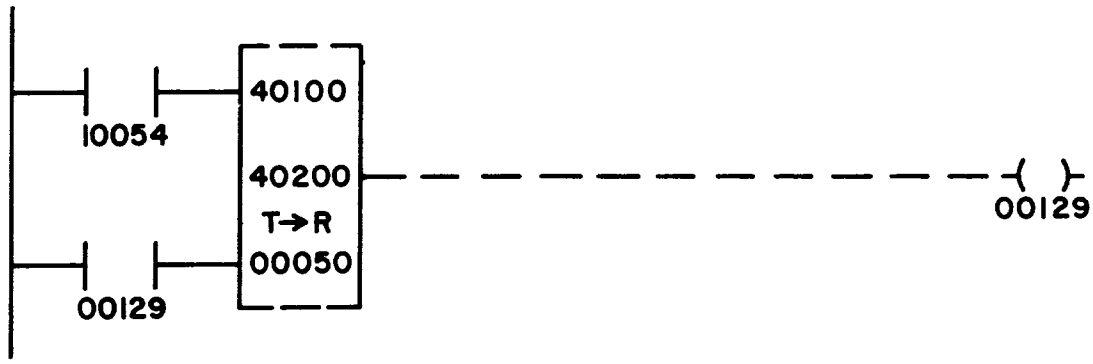



*Figure 3-3. Table-To-Register Move Logic*

When input 10054 is energized, the top input receives power and the content of the table starting at 40100 is moved into register 40201. Data is moved one register per scan. The pointer value increases after each move.

If the pointer value equals zero when the top input receives power, the content of register 40100 is moved into register 40201 and the pointer value (register 40200) increases to one. On the next scan, provided the top input is still receiving power, the content of register 40101 is moved into 40201 (the same register as before) and the pointer value (register 40200) increases to two. On this second scan, the value of register 40101 takes the place of the value of register 40100 in register 40201.

When input 10054 is energized, the top input receives power and the register moved is 40100 plus the pointer value (e.g., if the pointer value in register 40200 is 32, the register moved is 40132). When the pointer register 40200 equals 50, coil 00129 is energized. The bottom input is energized because it is referenced to coil 00129. The pointer is reset to zero, thereby de-energizing coil 00129.

Figure 3-4 illustrates the Table-To-Register Move described in the preceding paragraphs.

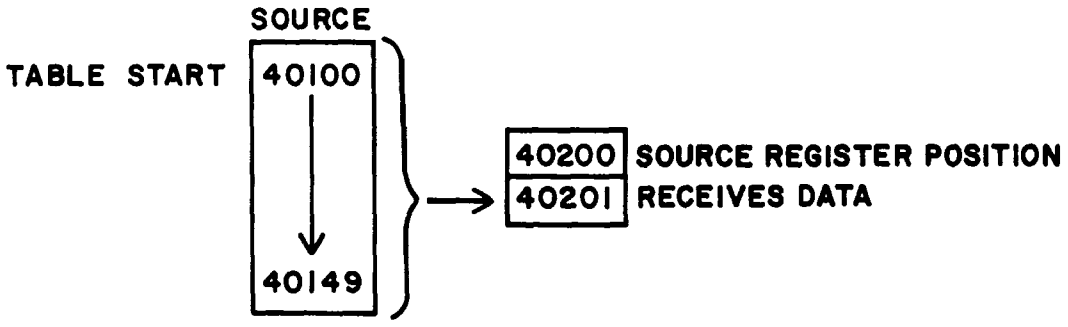
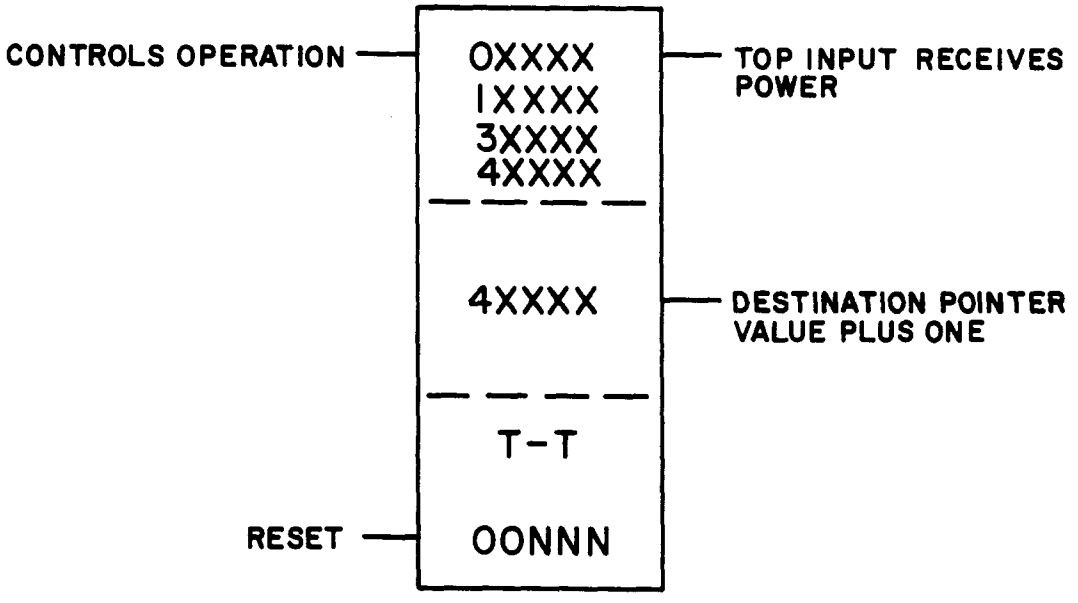



*Figure 3-4. Table-To-Register Move*

## 3.3 TABLE-TO-TABLE MOVE (T → T)

FUNCTION

The Table-To-Table Move function copies discretes or registers from one table to a table of holding registers.



FUNCTION BLOCK

- The top node is the source node. It can be one of the following references: a 0XXXX logic coil, a 1XXXX discrete input, a 30XXX input register, or a 4XXXX holding register. The source is a table of 16 bit locations. Its size is defined in the bottom node.

- The middle node is the destination node. It is a 4XXXX holding register which holds the pointer value. The pointer value plus one indicates the position of a register in a table (i.e., a pointer value of three indicates the fourth position in a table). The table starts at the next register (4XXXX + 1), not at the pointer.

- The bottom node contains the symbol T—►T and the numerical value that specifies the length for both tables. This constant can range from 1 to 255 in a Level I 584L PC, or from 1 to 999 in a Level 2 584L PC.

**NOTE**

If the pointer register is loaded with a value greater than the table length, the 584L PC sets the pointer value to the table length when the function block is solved.

INPUTS

- The top input controls the operation. When it is receiving power, the information in one register of the source table is copied into the corresponding register in the destination table.

- The middle input, when receiving power, prevents the pointer value from increasing.

- The bottom input, when receiving power, resets the pointer value to zero. If the top and bottom inputs are energized at the same time, the first register in the table is moved to the first register in the destination table.

OUTPUTS

- The top output passes power when the top input receives power.

- The middle output passes power when the pointer value equals the table length (end of table).

EXAMPLE
The following paragraphs provide a detailed explanation of the logic used in Figure 3-5.

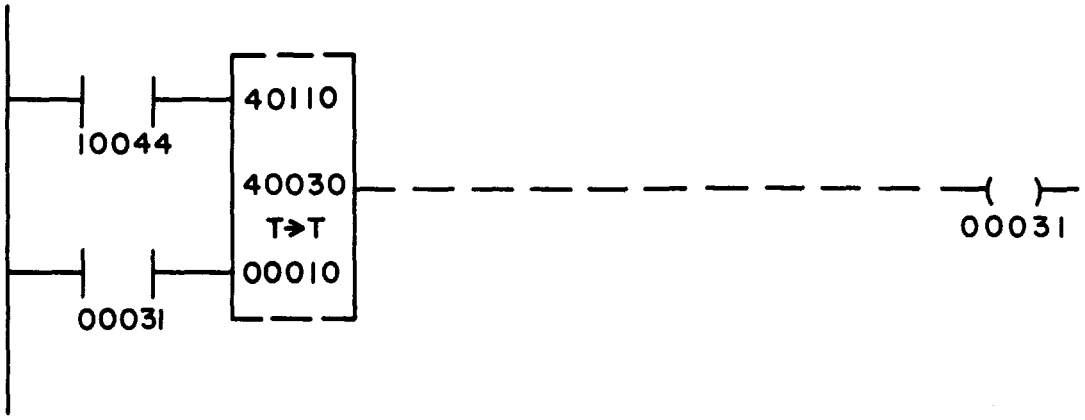



*Figure 3-5. Table-To-Table Move Logic*

When input 10044 is energized, the top input receives power. A register from the table which starts at register 40110 is moved into the register at the corresponding position in the table which starts at 40031. Register 40030 holds the pointer value.

One register is moved per scan and the pointer value increases by one on each scan. The middle output passes power when the pointer value equals the table size 10 in this example, thus energizing coil 00031. On the next scan, the bottom input receives power because it is referenced to coil 00031. The pointer value is reset to zero; therefore, coil 00031 is de-energized OFF.

At the start of the move, if the pointer is at zero, the content of register 40110 is copied into register 40031, and the pointer increases to one. On the next scan, register 40111 is copied into register 40032, and the pointer increases to two. If the function block stops receiving power, the pointer holds its value and recalls it when power is returned.

Figure 3-6 illustrates the Table-To-Table Move described in the preceding paragraphs.

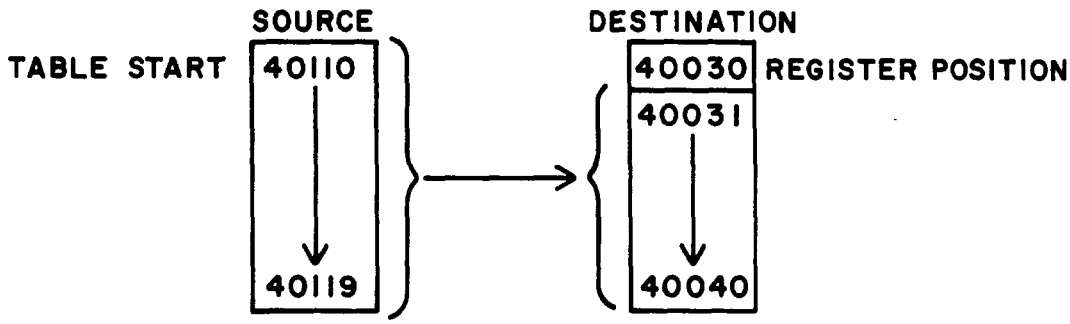
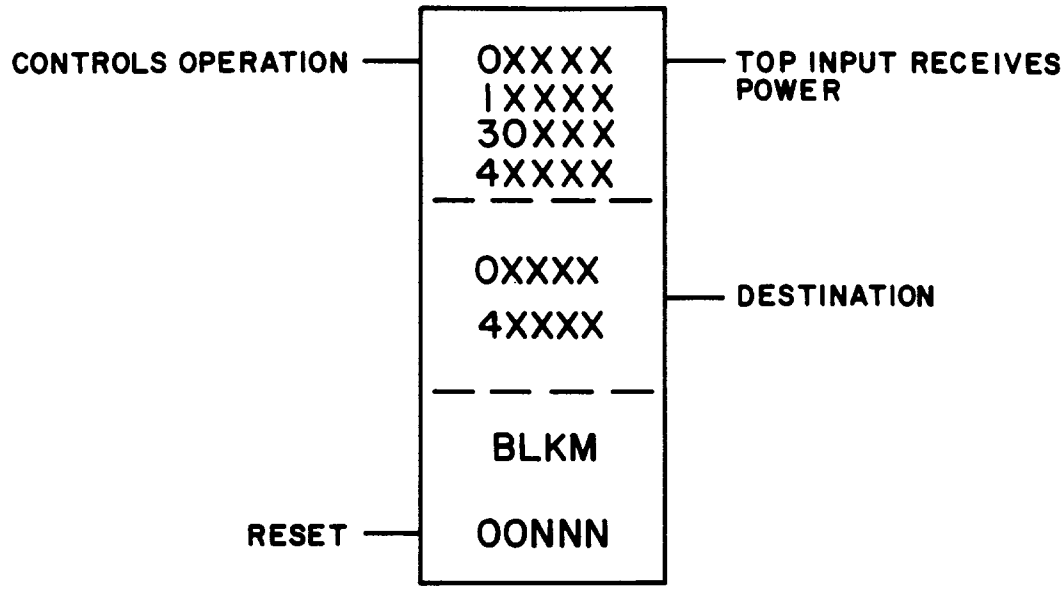



*Figure 3-6. Table-To-Table Move*

## 3.4 BLOCK MOVE (BLKM)

FUNCTION

The Block Move function copies the entire contents of a table of registers or discretes into another table on one scan. This function does not use a pointer register.




FUNCTION BLOCK

- The top node is the source node. It can be one of the following references: a 0XXXX logic coil, a 1XXXX discrete input, a 30XXX input register, or a 4XXXX holding register. The source is a table of 16 bit locations.

- The middle node is the destination node. It can be either a 0XXXX logic coil reference or a 4XXXX holding register reference. The destination is a table of 16 bit locations, the same size as the source.

**WARNING**

The Block Move function overrides the disable state of a coil used in the destination node of the function block. This can cause personal injury if the user assumes a coil has disabled an operation and repairs are being made, because the coil's state can change as a result of the BLKM function.

- The bottom node contains the symbol BLKM and a numerical value that specifies the table length for both the source and the destination. This constant can range from 1 to 100.

INPUT

- The top input controls the operation. When it receives power, one table of registers or discretes is copied into another table of the same length.

OUTPUT

- The top output passes power when the top input receives power.

**NOTE**

Only the top input and top output are used.

EXAMPLE
The following paragraph provides a detailed explanation of the logic used in Figure 3-7.

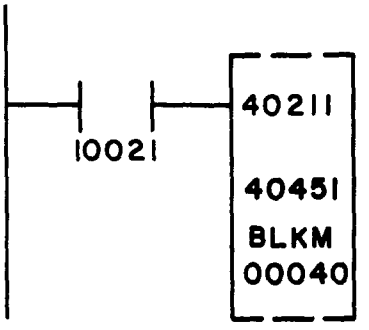



*Figure 3-7. Block Move Logic*

When input 10021 receives power, the contents of registers 40211-40250 are copied into registers 40451-40490. All the registers are moved in one scan, each scan the top input receives power. No output is required for this function.

Figure 3-8 illustrates the Block Move described in the preceding paragraphs.

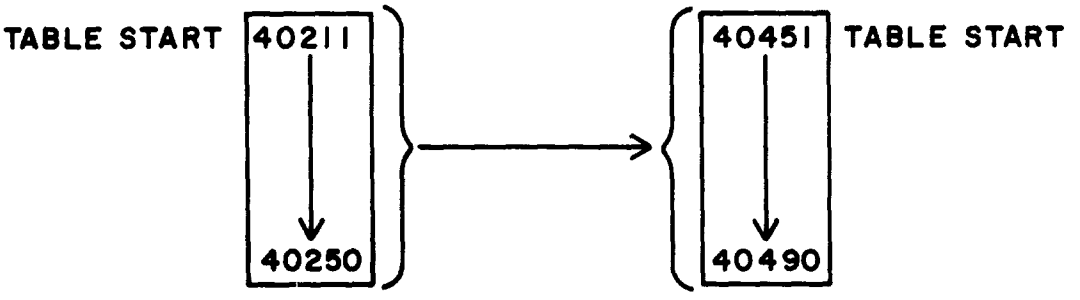



*Figure 3-8. Block Move*

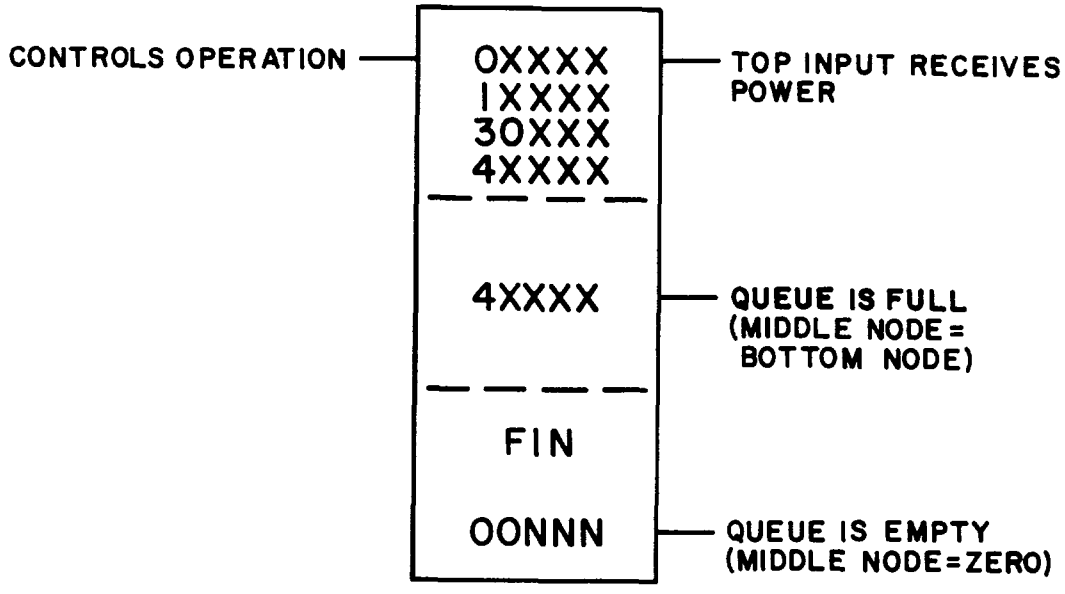
## 3.5 FIFO OPERATIONS

A FIFO queue is a table of 4XXXX holding registers. Individual 16 bit data items come out of the queue in the same order they entered the queue. A common analogy to a FIFO queue is a paper cup dispenser — cups are removed in the same order they were inserted.

Generally, both a First-In and a First-Out function block are used for each queue. The same pointer register is used for both the First-In and First-Out function blocks, and the queue length in each must be the same.

### 3.5.1 First-In (FIN)

FUNCTION
The First-In function inserts new data into the queue.




FUNCTION BLOCK

- The top node is the source node. It can be one of the following references: a 0XXXX logic coil, a 1XXXX discrete input, a 30XXX input register, or a 4XXXX holding register. The source is a single 16 bit location (e.g., a register or a group of sixteen discretes).

- The middle node is the destination node. It is a 4XXXX holding register which is used as the pointer. It also holds the pointer value (number of registers in the queue). The data is placed in the queue starting at 4XXXX + 1.

- The bottom node contains the symbol FIN and the numerical value that specifies the queue length. This constant can range from 1 to 100.

**NOTE**

If the pointer register is loaded with a value greater than the queue length, the 584L PC sets the pointer value to the queue length when the function block is solved.

INPUT

- The top input controls the operation. When it receives power, the information in the source register is copied to the first location in the queue and the pointer value is increased.

OUTPUTS

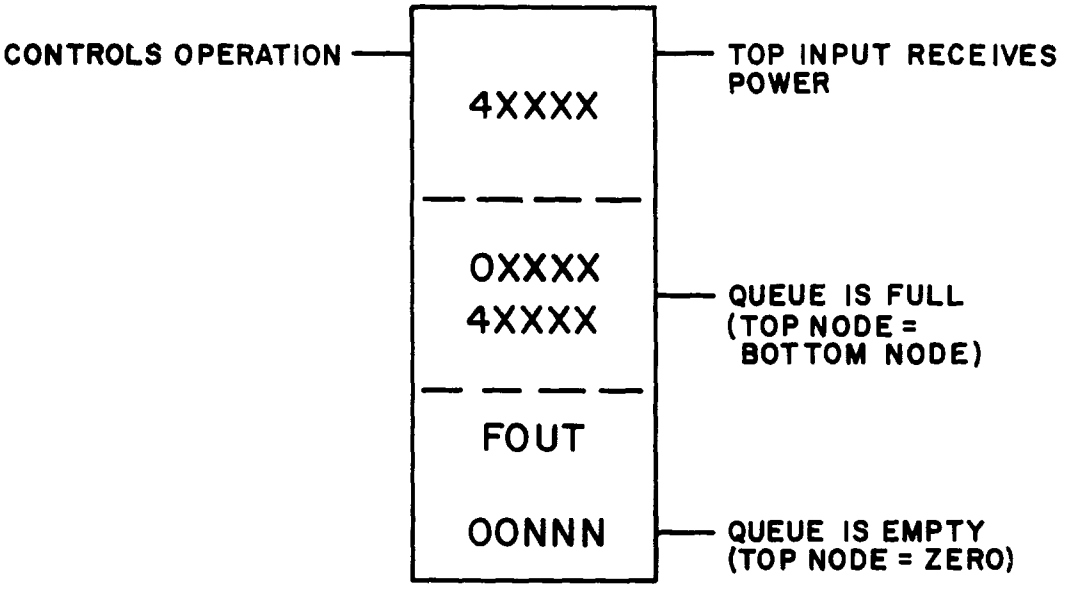
- The top output passes power when the top input receives power.

- The middle output passes power when the queue is full, pointer value equals queue length.

- The bottom output passes power when the queue is empty, pointer value equals zero.

### 3.5.2 First-Out (FOUT)

FUNCTION
The First-Out function removes the oldest data from the queue.




FUNCTION BLOCK

- The top node is the source node. It is a 4XXXX holding register which is used as the pointer. The queue starts at 4XXXX + 1. This register also holds the pointer value (number of registers in the queue).

- The middle node is the destination node. It can be either a 0XXXX logic coil reference or a 4XXXX holding register reference. It is a single 16 bit location such as a register or group of 16 discretes.

**WARNING**

The FOUT function overrides the disable state of a coil used in the destination node of the function block. This can cause personal injury if the user assumes a coil has disabled an operation and repairs are being made, because the coil's state can change as a result of the FOUT function.

- The bottom node contains the symbol FOUT and the numerical value that specifies the queue length. This constant can range from 1 to 100.

**NOTES**

The FOUT function is the only function which uses the source register as the pointer.

If the pointer register is loaded with a value greater than the queue length, the 584L PC sets the pointer value to the queue length when the function block is solved.

INPUT

- The top input controls the operation. When it receives power, the oldest information in the queue is removed and placed in sixteen logic coils or a holding register. Also, the pointer value is decreased by one.

OUTPUTS

- The top output passes power when the top input receives power.
- The middle output passes power when the queue is full, pointer value equals queue length.
- The bottom output passes power when the queue is empty, pointer value equals zero.

EXAMPLE

The following paragraphs provide a detailed explanation of the logic used in Figure 3-9.

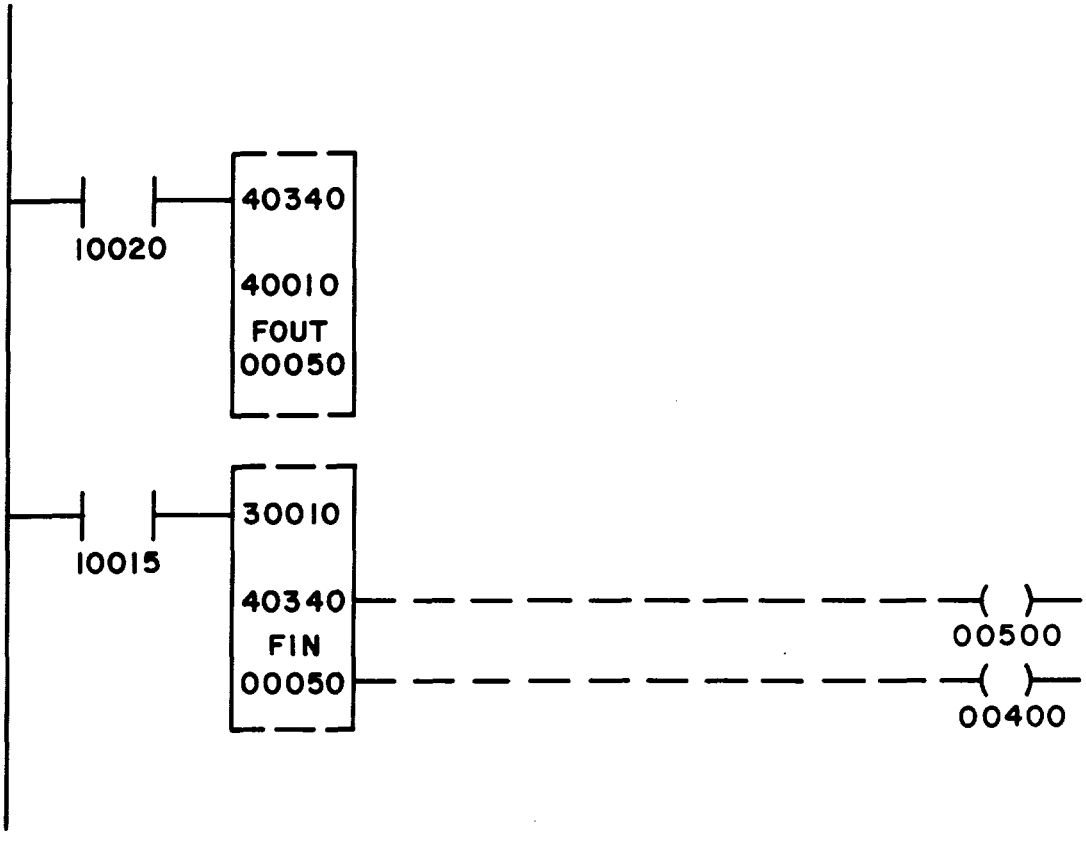



*Figure 3-9. First-In/First-Out Logic*

When input 10020 is energized, the oldest data in the queue is removed and placed in register 40010, and the pointer is decreased.

If the queue is full (value 0050 in register 40340), coil 00500 is energized. Attempts to insert new data are ignored. If the queue is empty, coil 00400 is energized and attempts to remove old data are ignored.

When input 10015 is energized, the top input receives power and register 30010 is copied into register 40341 in the queue (40341-40390). The pointer value in register 40340 increases by one. Each scan the top input receives power, a new value enters the queue. As new data is entered, old data is "pushed down" one register; therefore, register 40341 always contains the most recent entry.

The FOUT function block is placed before the FIN function block to ensure that, if the queue is full, the oldest data is removed before new data is entered. If the FIN block came first, an attempt to enter new data would be ignored if the queue was full. Placing the FOUT block first ensures that this will not happen; there will always be at least one empty register for new data.

Figure 3-10 illustrates the FIFO moves described in the preceding paragraphs.

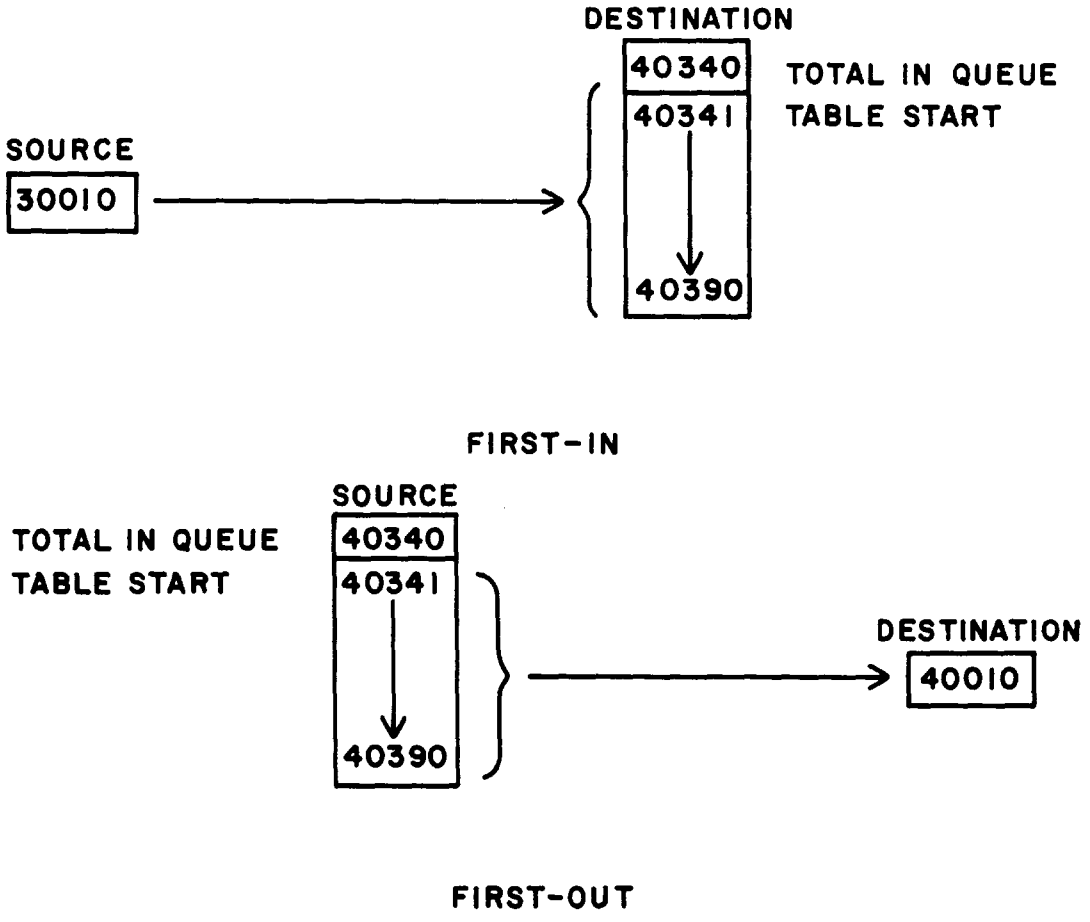
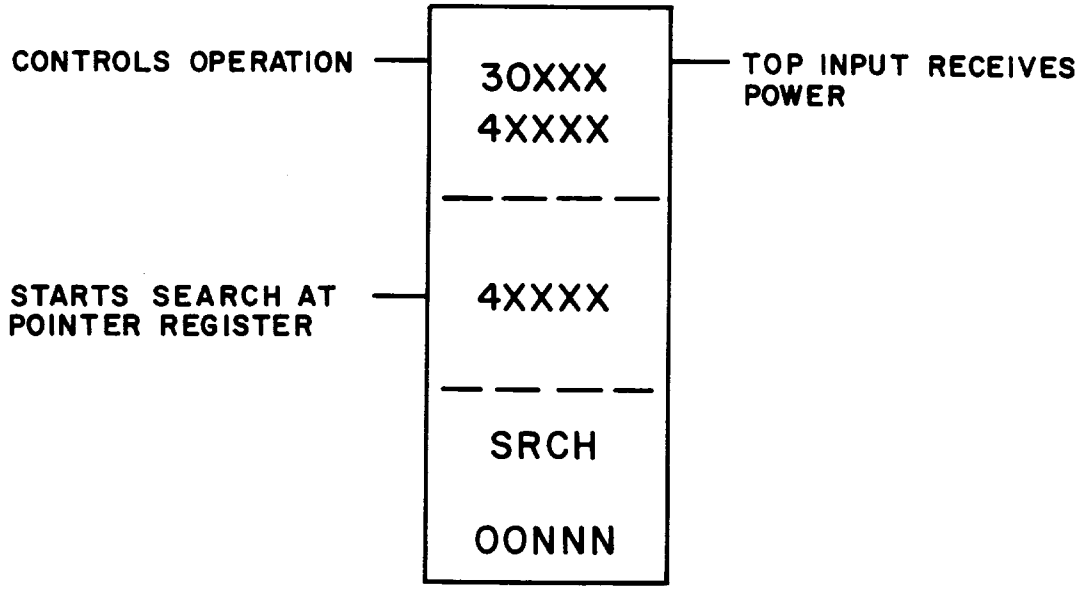



*Figure 3-10. FIFO Moves*

## 3.6 TABLE SEARCH OPERATION (SRCH)

FUNCTION

The Table Search function searches a table of registers for a specified value. When a matching value is found, the operation stops and the pointer value indicates the register in which the match is.




FUNCTION BLOCK

- The top node is the source node. It can be either a 30XXX input register reference or a 4XXXX holding register reference. This register is the first register of a table.

- The middle node is the destination node. It is a 4XXXX holding register which is used as a pointer and it also holds the pointer value. This value indicates the location of the register containing a match. The next consecutive holding register, 4XXXX + 1, contains the value being searched for; this value can be a 4-digit number up to 9999, a 16 bit binary pattern, or two ASCII characters.

- The bottom node contains the symbol SRCH and the numerical value that specifies the table length. This constant can range from 1 to 100.

INPUTS

- The top input controls the operation. When it receives power, each register in a table is examined to see if it contains a specified value. The search begins at the first register in a table unless the middle input is receiving power and the pointer value is greater than zero.

- The middle input, when it receives power, begins the search operation at the register whose location is specified in the pointer register, or continues the search operation from the register in which the match was found. If the pointer value is greater than zero and the middle input does not receive power, the search begins at the first register in the table.

OUTPUTS

- The top output passes power when the top input receives power.
- The middle output passes power when a match is found. If no match is found in the scan of an entire table, this output does not pass power and the pointer is reset to zero.

EXAMPLE

The following paragraphs provide a detailed explanation of the logic used in Figure 3-11.

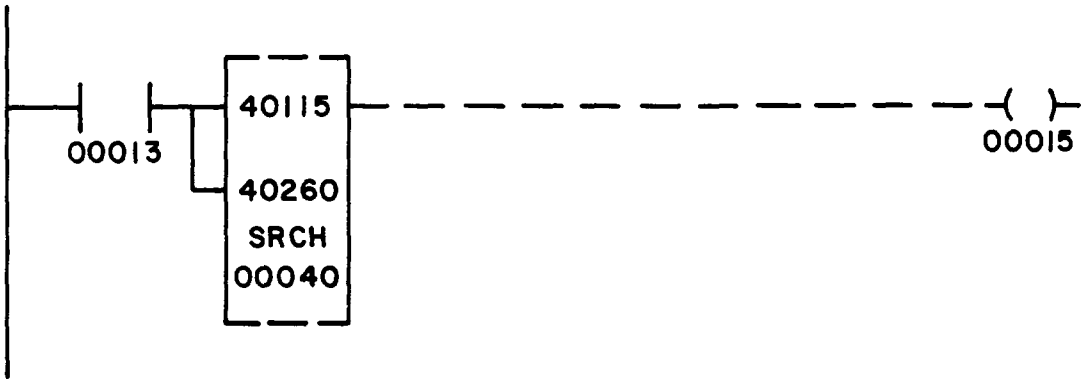



*Figure 3-11. Table Search Logic*

When contact 00013 is energized, the top input receives power. The table, which starts at register 40115, is searched to see if it holds the same value as in register 40261. Register 40260 holds the pointer value.

If a match is found, the search stops, the position number of the register is placed in 40260, and the middle output passes power.

To search for additional matches, both the top and middle inputs must receive power. If this condition is true, the search continues at the next consecutive register after the one containing a match. If no match is found, the middle output does not pass power and the pointer is reset to zero.

Figure 3-12 illustrates the Table Search described above.

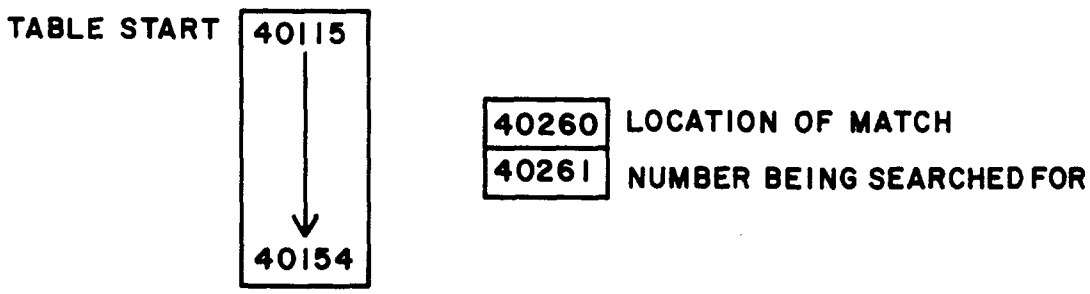
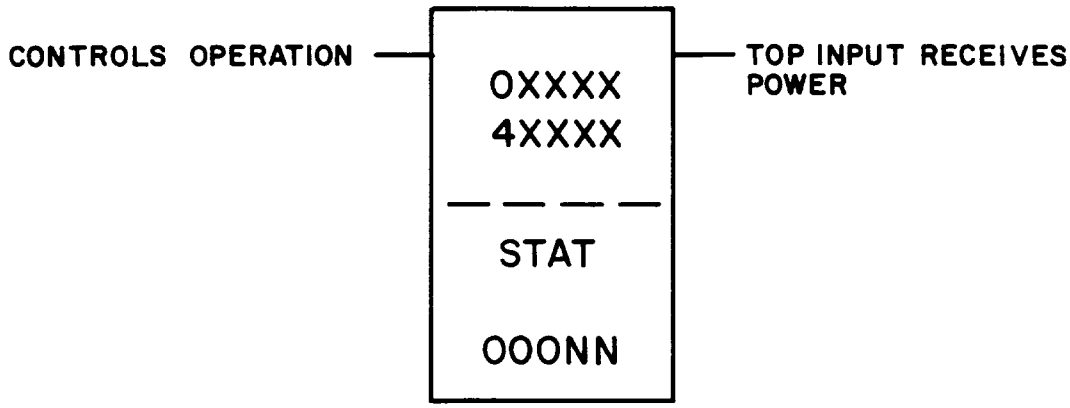



*Figure 3-12. Table Search*

## 3.7 GET CONTROLLER SYSTEM STATUS (STAT)

FUNCTION

The Get Controller System Status function obtains vital information about the controller, such as: memory protect status, battery status, I/O error, loss of active lights, and J200 Remote I/O Interface status. The information is placed in a table of registers or discretes.




FUNCTION BLOCK

- The top node is the destination node. It specifies the registers or discretes which will hold the status information. It can be either a 0XXXX logic coil reference or a 4XXXX holding register reference. It is a table of registers or groups of discretes. Each register has 16 bit locations and discretes are in groups of sixteen.

- The bottom node contains the symbol STAT and the numerical value that specifies the table length. This constant can range from 1 to 71.

**WARNING**

The STAT function overrides the disable state of a coil used in the destination node of the function block. This can cause personal injury if the user assumes a coil has disabled an operation and repairs are being made, because the coil's state can change as a result of the STAT function.

INPUT

- The top input controls the operation. When it receives power the controller status is copied into the registers specified in the top node.

OUTPUT

- The top output passes power when the top input receives power.

STATUS INFORMATION

Status information is located in specific words of memory. Each word has its own holding register. The status information available and the words containing the information are listed in Table 3-1. The registers listed in this table are just examples. Any group of seventy-one consecutive 4XXXX holding registers can be used or seventy-one consecutive groups of 16 coils.

*Table 3-1. Status Words*

| Word(s) | Register(s) | Status |
|---|---|---|
| 1 | 40101 | Controller Status |
| 4 | 40104 | J200 Remote I/O Interface Status |
| 12-27 | 40112-40127 | Input Module Active Light Status |
| 28-43 | 40128-40143 | Output Module Active Light Status |
| 44-71 | 40144-40171 | Communication Status to J200 Drops (2 words/registers to each drop) |

Words 2-3, and words 5-11 are for future use.

The Controller Status Word (register 40101) bit assignments are listed in Table 3-2. If the bit in the left column equals one (ON), the condition in the right column is true.

Bits in a word are numbered 1 to 16 and progress from left to right. Bit 1 is the most significant bit (MSB). Bit 16 is the least significant bit (LSB).

**NOTE**

The STAT block puts status words for remote channels I-4 after the status words for channel 32, instead of before the status words for channel 5.

*Table 3-2. Controller Bit Status*

| Bit No. | Condition |
|---|---|
| 1 | Port 1 Set Up |
| 2 | Port 2 Set Up |
| 3 | Port 1 Dev # Entered |
| 4 | Port 2 Dev # Entered |
| 5 | Future Use |
| 6 | Enable Constant Sweep |
| 7 | Enable Single Sweep Delay |
| 8 | Max. 2048 Reference System |
| 9 | AC Power Failure |
| 10 | Run Light OFF |
| 11 | Memory Protect is OFF |
| 12 | Battery Backup Fault (CMOS Memory) |
| 13 | Future Use |
| 14 | Future Use |
| 15 | Future Use |
| 16 | Future Use |

Each register or word of the I/O Module Active Light Status (registers 40112-40143) contains the input or output information for two channels. Table 3-3 illustrates this breakdown of channels into words or registers.

*Table 3-3. I/O Module Active Light Word Status*

| Channels | Word | Register |
|---|---|---|
| 1,2 | 12 | 40112 |
| 3,4 | 13 | 40113 |
| 5,6 | 14 | 40114 |
| 7,8 | 15 | 40115 |
| 9,10 | 16 | 40116 |
| . | . | . |
| . | . | . |
| . | . | . |
| 31,32 | 27 | 40127 |
| 1,2 | 28 | 40128 |
| 3,4 | 29 | 40129 |
| 5,6 | 30 | 40130 |
| 7,8 | 31 | 40131 |
| 9,10 | 32 | 40132 |
| . | . | . |
| . | . | . |
| . | . | . |
| 31,32 | 43 | 40143 |

**NOTE**

Words 12 through 27 contain input information. Words 28 through 43 contain output information.

Each register of the I/O Module Active Light Status (registers 40112-40143) has individual bit assignments. These are listed in Table 3-4.

A one bit indicates that the I/O slot has been enabled in the Traffic Cop and the slot does not contain a working I/O module.

A zero bit indicates that the I/O slot has either been inhibited in the Traffic Cop or the slot contains a working I/O module.

*Table 3-4. I/O Module Active Light Bit Status*

| Odd Channels | |
|---|---|
| Bit No. | Slot No. |
| 1 | I/O Slot No. 1 |
| 2 | I/O Slot No. 2 |
| 3 | I/O Slot No. 3 |
| 4 | I/O Slot No. 4 |
| 5 | I/O Slot No. 5 |
| 6 | I/O Slot No. 6 |
| 7 | I/O Slot No. 7 |
| 8 | I/O Slot No. 8 |
| **Even Channels** | |
| Bit No. | Slot No. |
| 9 | I/O Slot No. 1 |
| 10 | I/O Slot No. 2 |
| 11 | I/O Slot No. 3 |
| 12 | I/O Slot No. 4 |
| 13 | I/O Slot No. 5 |
| 14 | I/O Slot No. 6 |
| 15 | I/O Slot No. 7 |
| 16 | I/O Slot No. 8 |

Words 44-71 (Registers 40144-40171) represent the communication status to J200 drops. There are 14 drops, 2 channels each. Each drop uses two status words.

The bit assignments for each drop are the same. Table 3-5 contains the bit assignments for the first word in each drop (i.e., words 44, 46, 48, ...68, 70). The bit assignments for the second word in each drop (i.e., words 45, 47, 49, ... 69, 71) are listed in Table 3-6.

*Table 3-5. Remote I/O Word One Bit Status*

| Bit No. | Condition |
|---|---|
| 1-3 | This field identifies the type of processing scheduled to be performed by the 584L for the addressed IOR. The following functions are defined:<br><br>000 — Normal I/O<br>001 — Restart Phase I (Communication Reset)<br>010 — Restart Phase II (Application Reset)<br>011 — Unassigned (INHIBIT IOR)<br>100 — INHIBIT<br>101 — Unassigned (INHIBIT IOR)<br>110 — Unassigned (INHIBIT IOR)<br>111 — Unassigned (INHIBIT IOR) |
| 4 | Sequence number received on response from the addressed IOR is different from the expected value. |
| 5 | Byte count underrun. The internal IOR response byte count is incompatible with the IOL transmitted byte count. |
| 6 | The current message response received by the 584L from the addressed IOR is not supported. |
| 7 | Future Use |
| 8 | The current 584L message has not been accepted by the addressed IOR. |
| 9-11 | The receive sequence number. Indicates the number (modulo 8) of the next information frame that is expected to be received from the addressed IOR. |
| 12 | Identifies the cable to be used by the IOL to communicate with the addressed IOR (0 = Cable 0, 1 = Cable 1). |
| 13-15 | The send sequence number. Indicates the number (modulo 8) of the current information frame that is being sent to the addressed IOR. |
| 16 | The current 584L message has been queued for processing by the addressed IOR. |

*Table 3-6. Remote I/O Word Two Bit Status*

| Bit No. | Condition |
|---|---|
| 1 | Future Use |
| 2 | The IOL sensed a character overrun error from the addressed IOR. |
| 3 | The IOL sensed a communication error from the addressed IOR. A communication error is a CRC failure; receives abort or residual bit count error. |
| 4 | The addressed IOR did not respond to the latest 584L command message. |
| 5 | Future Use |
| 6 | Link reject response issued by the addressed IOR indicating that the IOR has just gone through its power-up sequence. |
| 7 | Link reject response issued by the addressed IOR indicating that the current message has an incompatible sequence number. |
| 8 | Link reject response issued by the addressed IOR indicating that the current command is not being supported by the IOR. |
| 9-16 | This field counts the number of retries attempted by the 584L to the addressed IOR. The maximum value is $255_{10}$. |

EXAMPLE
The following paragraph provides a detailed explanation of the logic used in Figure 3-13.

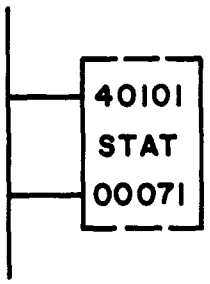



*Figure 3-13. Get Controller System Status (STAT)*

The status information is obtained every scan because the function block is next to the power rail. Since the length is 71, all the available status information is obtained in registers 40101 to 40171.

## 3.8 SUMMARY OF MOVE FUNCTIONS

| | Source | Destination | Max. Length | Top Input | Middle Input | Bottom Input | Top Output | Middle Output | Bottom Output |
|---|---|---|---|---|---|---|---|---|---|
| R→T | Any Reference | 4XXXX Pointer | 255/999 | Move, Increase Pointer | No Increasing | Enable/ Reset | Top Input Receiving Power | Table Full | Not Used |
| T→R | Any Reference | 4XXXX Pointer | 255/999 | Move, Increase Pointer | No Increasing | Enable/ Reset Power | Top Input Receiving | Table Full | Not Used |
| T→T | Any Reference | 4XXXX Pointer | 255/999 | Move, Increase Pointer | No Increasing | Enable/ Reset | Top Input Receiving Power | Table Full | Not Used |
| BLKM | Any Reference | 0XXXX, or 4XXXX | 100 | Move | Not Used | Not Used | Top Input Receiving Power | Not Used | Not Used |
| FIN | Any Reference | 4XXXX Pointer | 100 | Move, Increase Pointer | Not Used | Not Used | Top Input Receiving Power | Queue Full | Queue Empty |
| FOUT | 4XXXX Pointer | 0XXXX, or 4XXXX | 100 | Move, Decrease Pointer | Not Used | Not Used | Top Input Receiving Power | Queue Full | Queue Empty |
| SRCH | 30XXX, or 4XXXX | 4XXXX Pointer | 100 | Search | Start at Pointer | Not Used | Top Input Receiving Power | Value Found | Not Used |
| STAT | — | 0XXXX, or 4XXXX | 71 | Move | — | Not Used | Top Input Receiving Power | — | Not Used |

## 3.9 LOGIC EXAMPLES

### 3.9.1 Analog Multiplexing

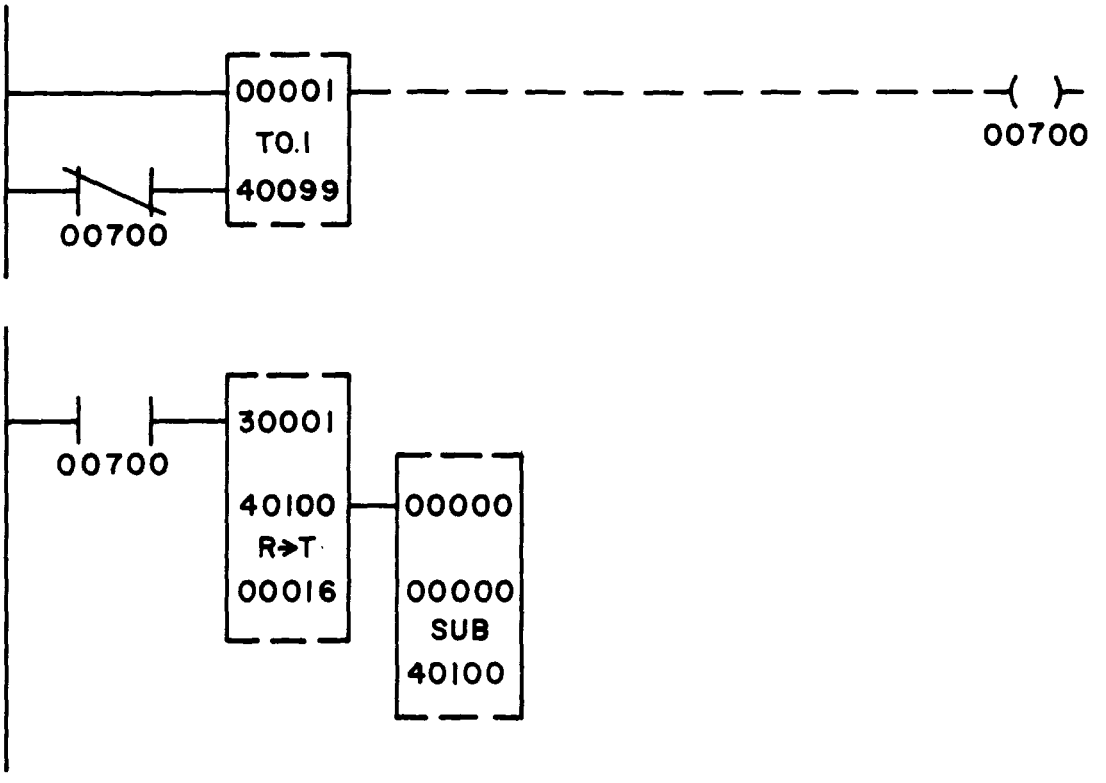



*Figure 3-14. Analog Multiplexing*

The B258 Analog Multiplexer module is used with a B243 Analog to Digital Converter module and the 584L PC. The B258 module multiplexes one of sixteen analog signals into one input of a B243 module.

The logic needed to perform this operation is shown in Figure 3-14. The first network contains a one-tenth of a second timer which is used to initiate the reading of the analog input. Every tenth of a second, coil 00700 is energized and the R⟶T function block in the second network receives power.

When this input receives power, the value in binary input register 30001 is moved into the first location (40101) in a table of sixteen registers starting at 40101. After the first move, the counter automatically increments so the next value is inserted into register 40102. This R⟶T move continues until the table (40101-40116) is full (pointer value equals sixteen). When the table is full the middle output passes power and the subtract block receives power. The subtract clears the pointer value (40100) to zero for the next scan.

**NOTE**

For this example, the 584L traffic cop must contain 30001 as a binary input register and 40100 as a BCD output register.

### 3.9.2 Recipe Storage

The following paragraphs provide a detailed explanation of the logic illustrated in Figure 3-15 to program a recipe storage function.

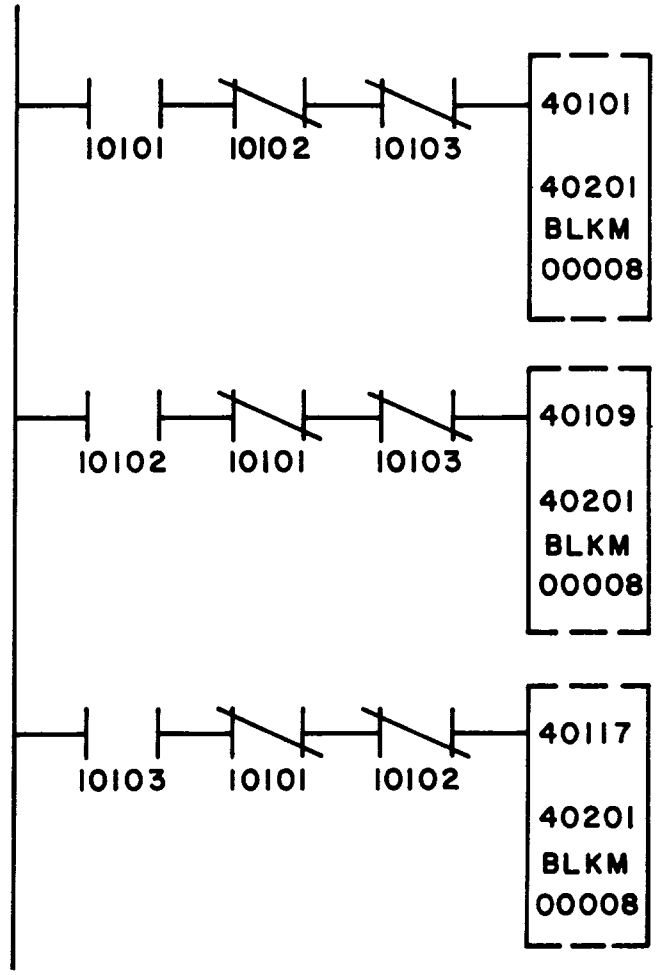



*Figure 3-15. Recipe Storage*

In some applications, information in various tables is needed, one table at a time, to perform various functions. An example of this is a manufacturer who manufacturers three products which are similar yet have specific differences. An example of this is soups. Soups are similar, but chicken soup is different from tomato soup.

The information or recipe for each soup is stored in a unique table. Since only one soup is made at a time, a working table is needed which can apply to any of the three soups. This is accomplished with a block move as shown in Figure 3-15. The tables for all the soups contain specific information in corresponding registers. These registers must also correspond with the working table's registers. For example, if the first register in one table contains cooking time, the first register in all the tables must contain cooking time.

The process is controlled from an operator panel. The panel can have three input switches; 10101, 10102, and 10103. To make soup A, the operator turns 10101 ON, and 10102 and 10103 remain OFF.

Following the logic in Figure 3-15, input 10101 is energized and passes power through normally closed contacts 10102 and 10103. The recipe for soup A is moved from table 40101-40108 to table 40201-40208. Table 40201-40208 is a working table. Each output register in this table is controlling a specific part of the operation.

If the original recipe tables are used as working tables, three individual programs are required. By using one working table as illustrated in this example, only one program is needed to control the output information.