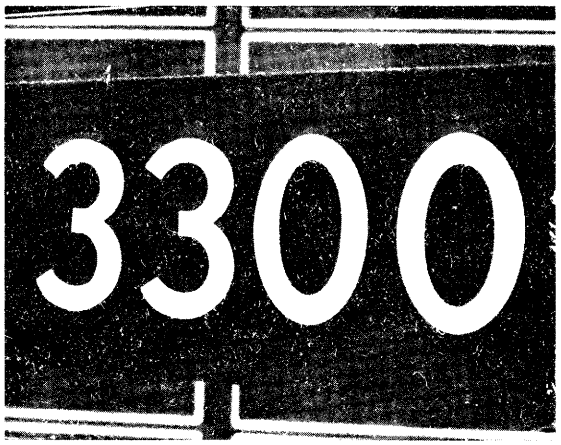
CONTROL DATA
EDUCATION INSTITUTES

CONTROL DATA
CORPORATION

SECOND EDITION



TRAINING MANUAL

Volume II

3300 COMPUTER TRAINING MANUAL

SECOND EDITION

FOR TRAINING PURPOSES ONLY

This manual was compiled and written
by instructional personnel of

CONTROL DATA INSTITUTE
CONTROL DATA CORPORATION

Publication Number 60158800A
August, 1968

This manual obsoletes the 3300 Computer Training Manual, Volumes II and III, Publication Number 60158800, Printed November, 1965.

CONTENTS

ILLUSTRATIONS

ILLUSTRATIONS (continued)

TABLES

STORAGE

STORAGE CAPACITY

The maximum internal storage capacity of a 3300 Computer System is 262,084 28-bit words. The minimum internal storage capacity is 8,192 28-bit words. The storage capacity of an 8K system may be increased to 16,384 words by adding an additional 8K module to the basic system. Any additional increments beyond 16,384 words must be made by adding 16K modules. Therefore, there are two types of storage modules available, the 3309 Storage Module (8K) and the 3302 Storage Module (16K).

3309 Storage Module

One 3309 (8K storage) is included as part of each processor. No more than two 3309's may be attached to any one system. Figure 51 shows the 3309 Storage Module.

3302 Storage Module

A 3302 provides 16,384 words of storage capacity; it has a 16K stack, not two 8K stacks. Therefore in this text the 3309 and 3302 must be treated separately. Figure 52 shows the 3302 Storage Module.

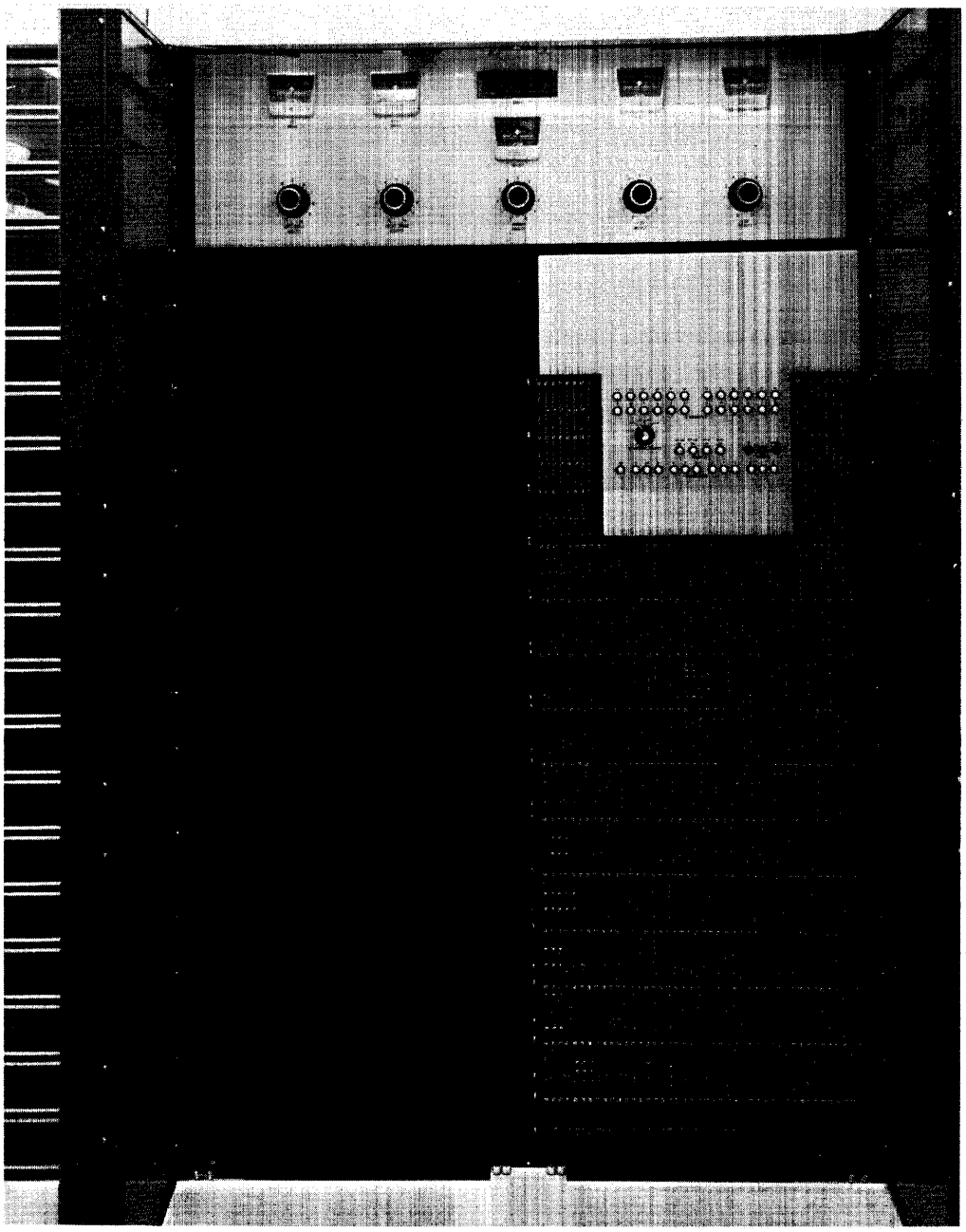


Figure 51. 3309 Storage Module

STORAGE WORD FORMAT

The 28-bit storage word (figure 53) contains 24 bits of data, which may be exchanged with the processor, and four parity bits, which are generated and checked only in the storage module. The storage word can be considered to be four 6-bit characters and four parity bits (one parity bit associated with each character).

STORAGE PARITY

Each character within a storage word has one parity bit associated with it (figure 53). During each write cycle, a parity bit is stored along with each character. When part or all of a word is next read from storage, parity is checked for a loss or gain of bits. The 3300 uses odd parity; that means the number of 1's in a character, plus the parity bit, always totals an odd number. Any failure to produce correct parity read operations cuases a memory fault indication on the storage module control panel and the console, followed immediately by a program halt. If the console PARITY STOP switch is inactive, parity

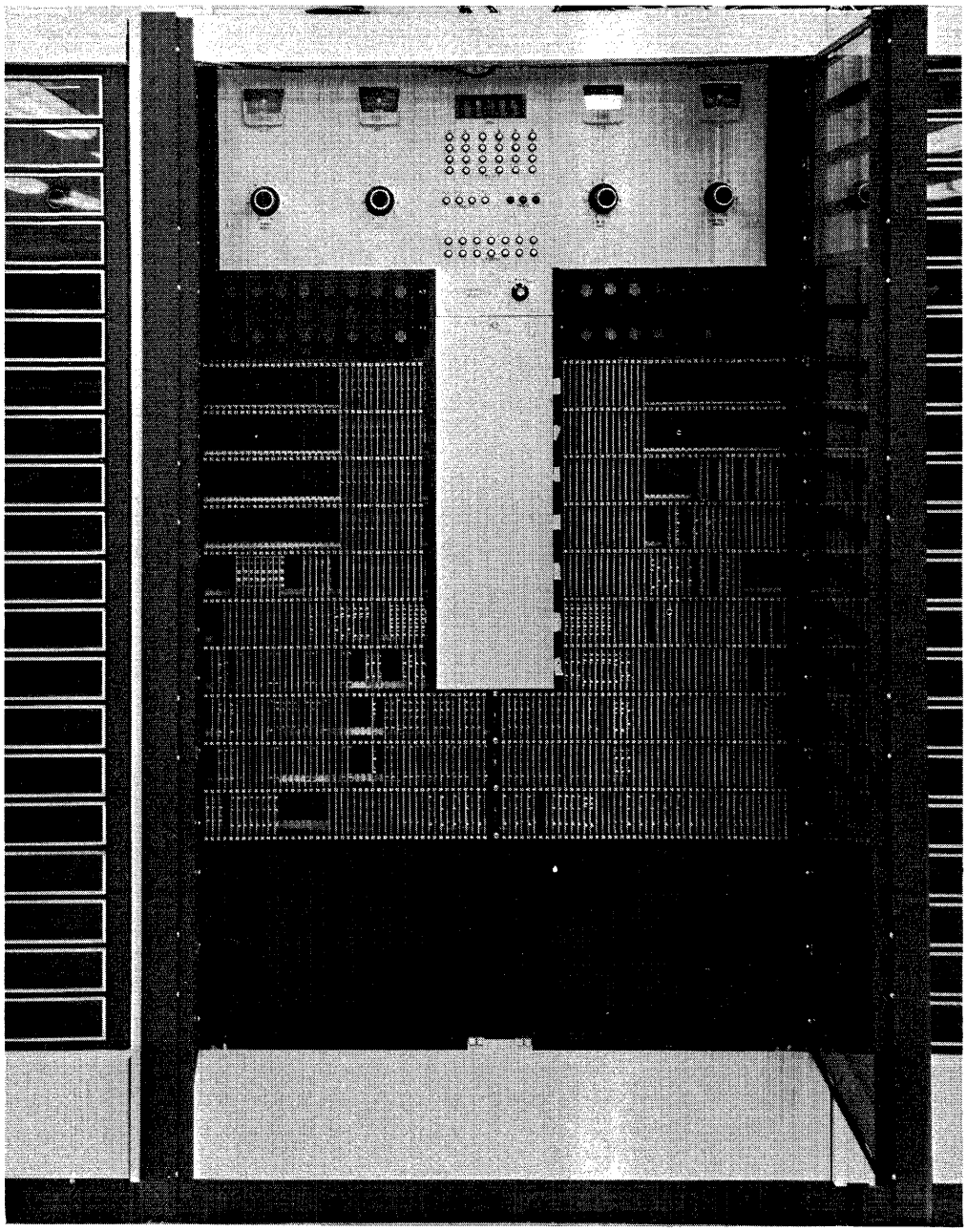


Figure 52. 3302 Storage Module

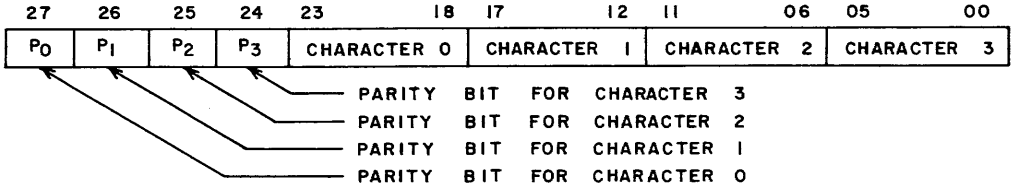


Figure 53. Storage Word Format

will still be generated, but the program will not halt because of a parity error.

The storage module control panels, figures 54 and 55, contain lights which display each bit of the current storage word, including four lights for display of the parity bits.

Complete the following table. Show whether the parity bits of each character is a 1 or a 0.

24-Bit Word To Be Stored	Parity Bit for Char 0	Parity Bit for Char 1	Parity Bit for Char 2	Parity Bit for Char 3
01234567				
20356016				
10667201				
00010233				
60731065				
06731026				

STORAGE MODULE DESCRIPTION

Storage Registers

Two registers, S and Z, are associated with each storage module.

1. S Register - The S register contains the address of the word currently being processed. The 3309 has a 13-bit S register; the 3302 has a 14-bit S register. The address is transmitted from the processor or the multiprogramming module.
2. Z Register - The 28-bit Z register is the storage restoration and modification register. A storage word may be read into Z and the 24 bits of data transmitted to the processor via the data bus or the processor may send a new 24-bit word (or portion of a word) over the data bus to storage where the word is placed in Z, then written into storage with the correct parity.

Read/Write Controls

During a normal memory read cycle, all bits of a word referenced by (S)* are read out of core storage in parallel, loaded into Z, used for some purpose, and written back into storage intact.

A memory write cycle is performed just like a read cycle except that certain groups of bits are blocked from Z register initially and new data from the processor is placed in these bits of Z. (Z) is then written back into storage. The correct parity for the new data is also generated and written into storage.

Five modes of storage modification (partial writes) are possible in the 3203/3209. In all cases, assume that Z is clear. Z register is cleared only at the beginning of each memory cycle (except when master cleared). If the program stops as the result of a parity error the operator can examine (Z) via the storage module control panel (figure 54).

The storage modification modes are:

1. Single-Character Mode - Any one 6-bit character may be ignored during read cycle. New data is then loaded into the corresponding character position of Z and the whole (Z) is stored.
2. Double-Character Mode - The upper, middle, or lower half (12 bits) of a word is ignored during read cycle. New data is loaded into the unfilled half of Z and the whole (Z) is stored.
3. Triple-Character Mode - Either of the two possible triple-character groups may be ignored during read cycle. New data is then loaded into the corresponding character positions of Z and the whole (Z) is stored.
4. Full-Word Mode - The whole 24-bit word is ignored during read cycle. A new word is entered into Z and (Z) is stored.
5. Address Mode - The lower 15 or 17 bits of a word may be ignored during read cycle. A new word or character address is then loaded into A and the whole (Z) is stored.

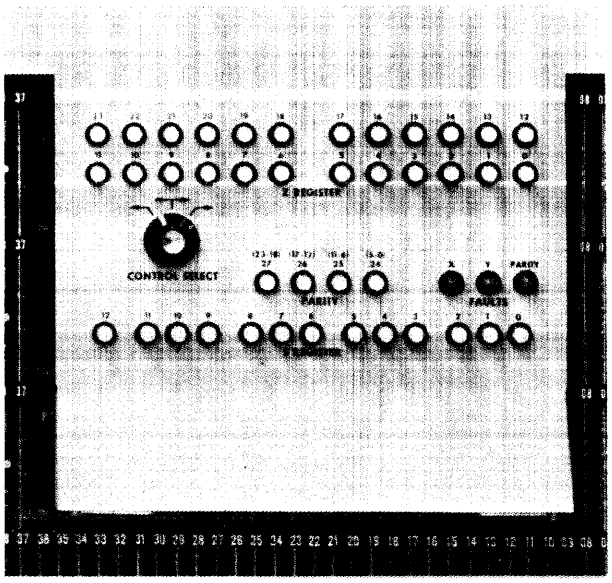


Figure 54. 8K Storage Module Control Panel

Storage Module Control Panel

Figure 54 shows the storage control panel which is mounted at the top of each 8K storage module. Figure 55 shows the storage control panel for the 16K storage module.

The drive voltage control permits the adjustment of the drive voltage to 26v. The drive voltage meter

*Parentheses signify "contents of."

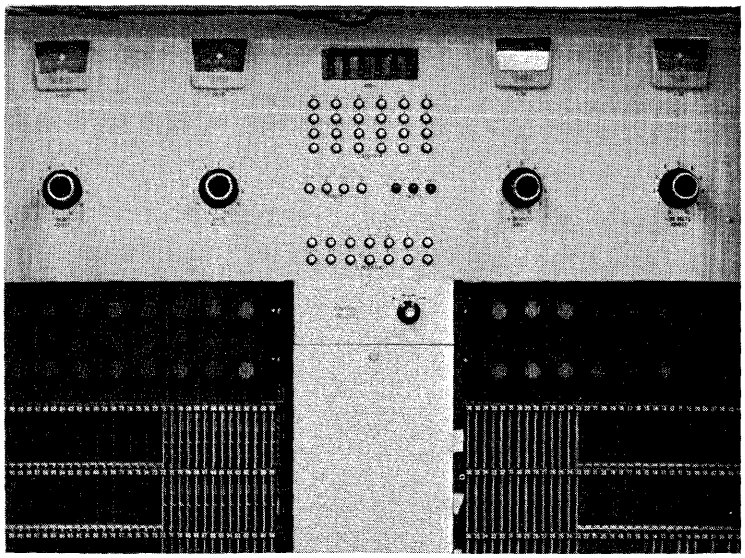


Figure 55. 16K Storage Module Control Panel

indicates drive voltage. It reads ±20% and is adjusted to read 0% (most modules require approximately 22.5v to provide the best margins).

The panel contains indicator lamps to display the contents of the S register, the contents of the Z register (including the four parity bits), and three storage faults: x or y drive line voltage failures and storage parity fault.

The panel also contains a three-position switch labeled CONTROL SELECT (figure 56). The storage module may communicate with one or both of the processors (or pieces of special equipment), depending on the setting of the CONTROL SELECT switch. The switch settings are:

1. The storage module communicates with the processor located to the left (as you face the control panel) of the module. Refer to figure 56 and note that the arrow for setting 1 points to the left.
2. The storage module communicates with the processor located to the right of the module. Note that the arrow for setting 2 points to the right.
3. The storage module may be shared by both processors. Note that arrows point in both directions for this switch setting (figure 56).

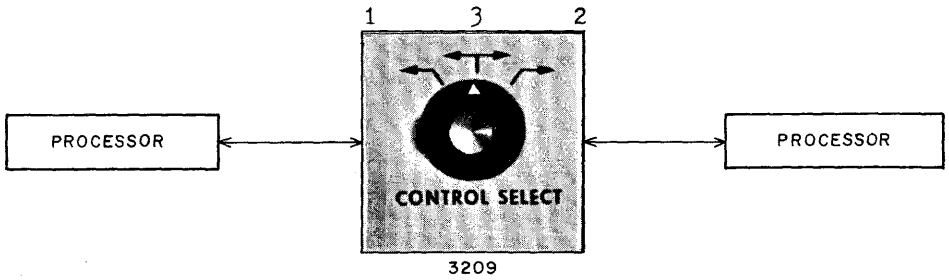


Figure 56. Use of Control Select Switch

A two-position scanner determines which processor communicates with the storage module at any given time. If the storage module is being used by one processor, a request from the other processor must wait. Control is then surrendered to the waiting processor at the end of the current cycle.

When a storage module is to be used by one processor only, the CONTROL SELECT switch should be set to point toward that processor. This disables the scanner and locks out the unused access path.

BUSSING SYSTEM

Figure 57 is a block diagram of an 8K storage module and shows the signals exchanged with a processor.

In figure 57 the dotted lines represent control operations while the solid lines represent transmissions. The heavy solid lines represent data transmissions.

Information and control signals are exchanged between a processor and storage modules over a system of twisted-pair wires known as the bus system. Figure 58 is a photograph of bus wires in chassis 1.

S Bus

The S bus system is an 18-bit bus which supplies a Storage Request signal and a 17-bit address to storage. The 17-bit address consists of a 3-bit Module Select code and a 14-bit Coordinate address

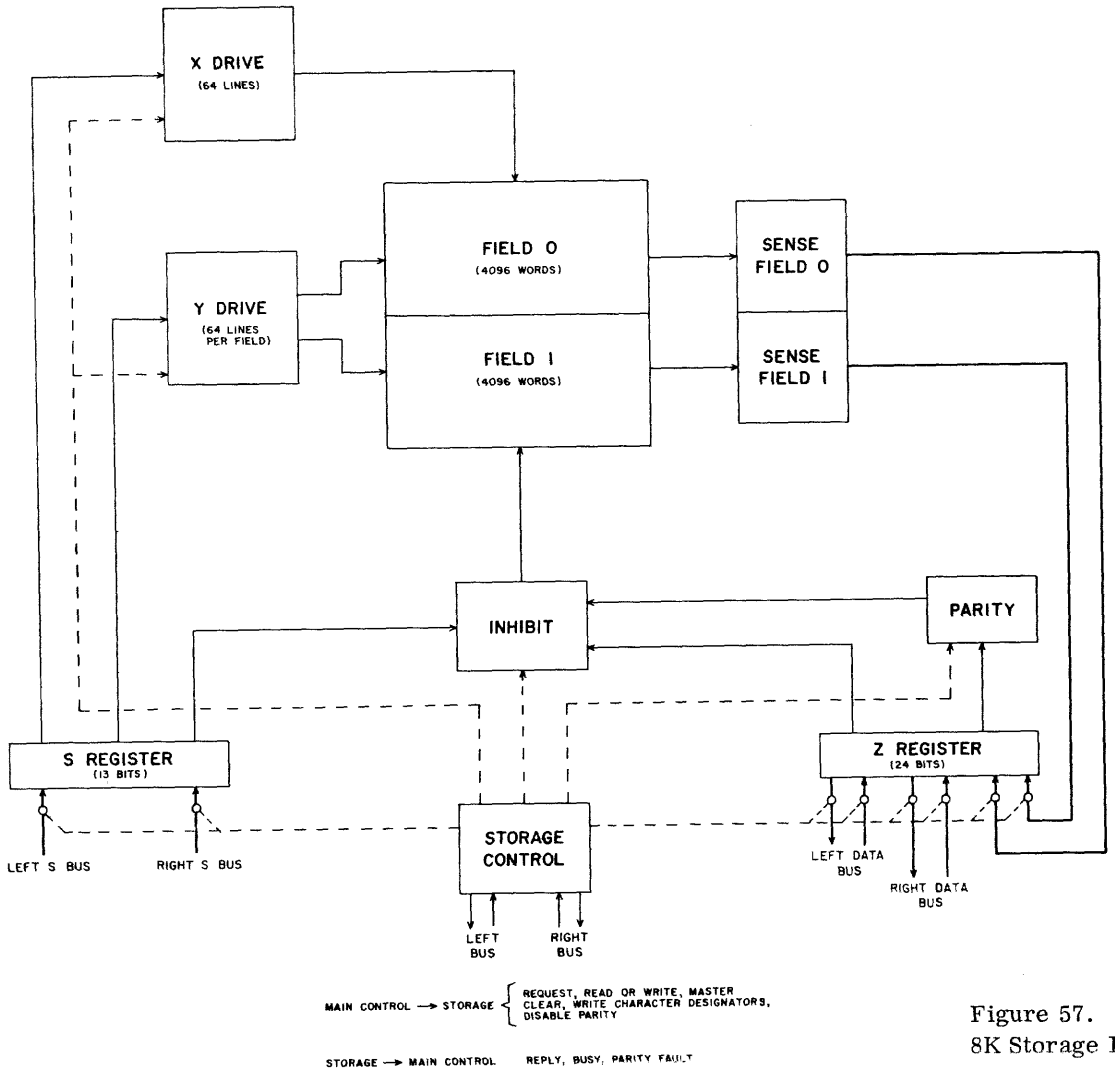


Figure 57.
8K Storage Block Diagram

for referencing a 3302. For referencing a 3309, the lower 17 bits of the S bus contain a 4-bit Module Select code and 13-bit Coordinate address. If the Multiprogramming module is not present in a 3300 system the following characteristics exist:

1. Maximum internal storage capacity is 131K.
2. Only one bus system exists.
3. The storage address and storage request go directly from the processor to storage.

If the Multiprogramming module is present the following characteristics exist:

1. Maximum internal storage capacity is 262K.
2. Two distinct buses exist, one defined as the right bus (allows access to addresses 000000 through 377777), the other defined as the left bus (which allows access to addresses 400000 through 777777).
3. The storage address and storage request are placed on the storage S bus by the Multiprogramming module.

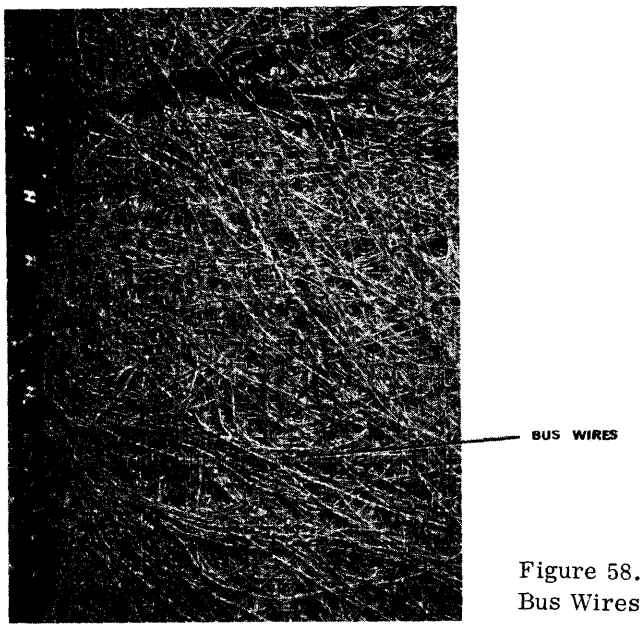


Figure 58.
Bus Wires

Figure 59 graphically represents the S bus scheme if the Multiprogramming module is not present whereas figure 60 represents the S bus scheme if the Multiprogramming module is present.

In figures 61 and 62, notice that any address placed on the storage S bus becomes available to all storage modules in the system; however, Storage Request (bit 17) goes only to the right or to the left.

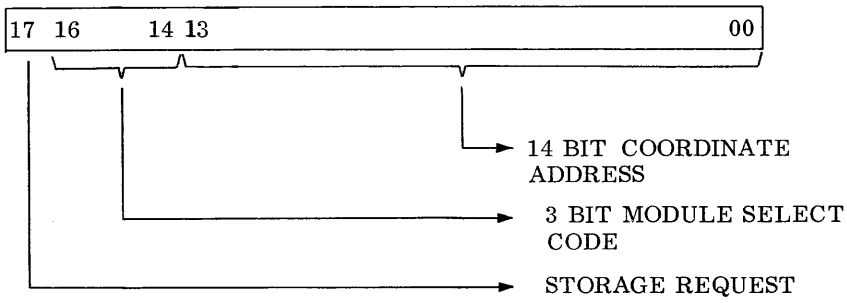


Figure 59. Function of S Bus Bits When Referencing 3302

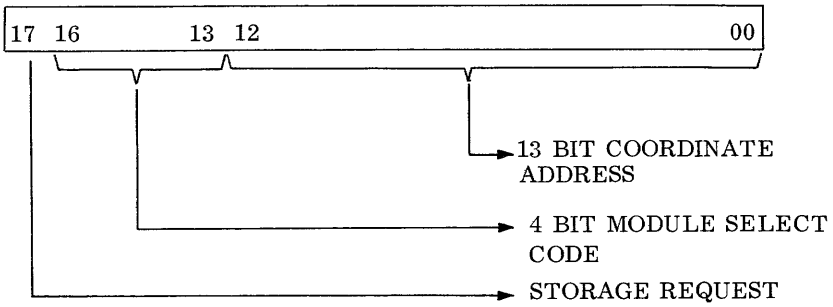


Figure 60. Function of S Bus Bits When Referencing 3309

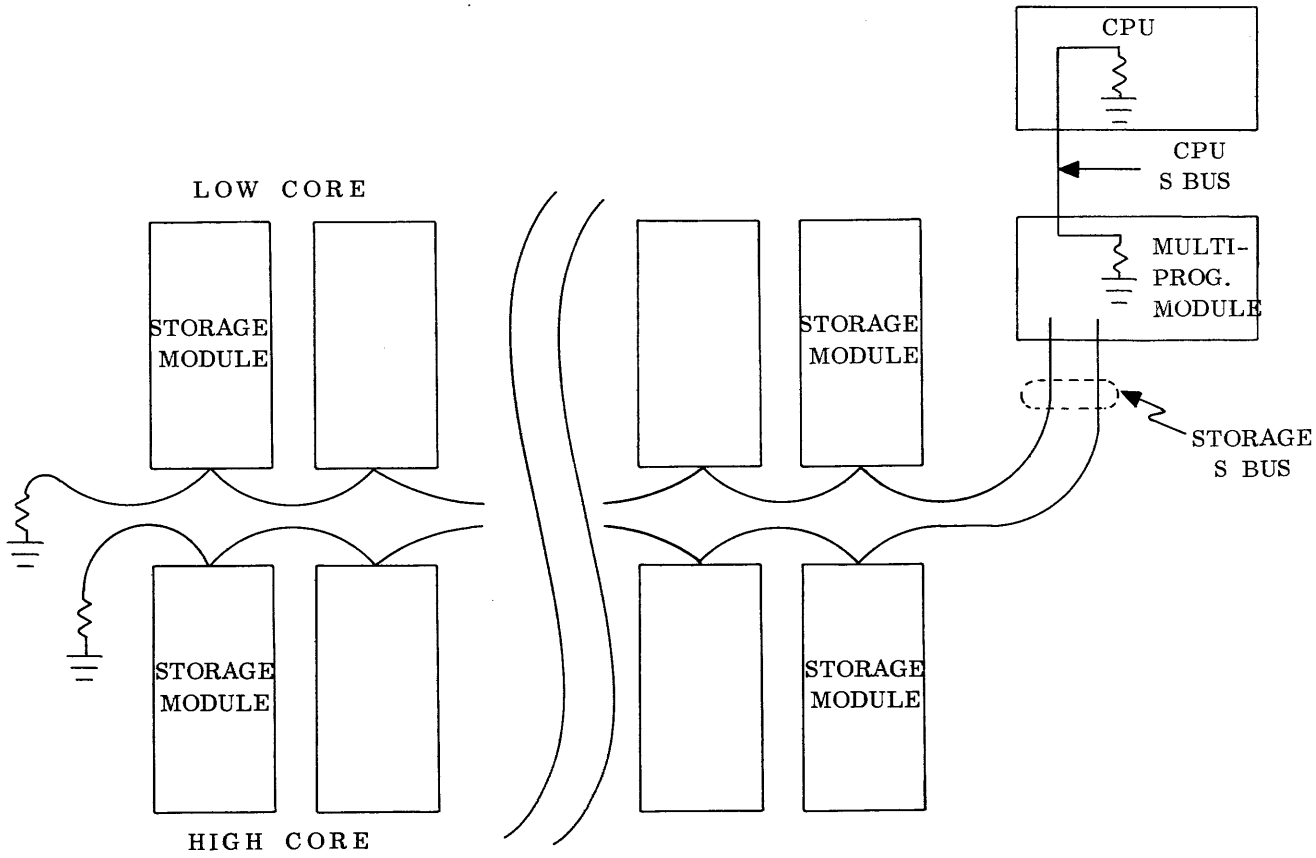


Figure 61. S Bus Scheme if Multiprogramming Module Present

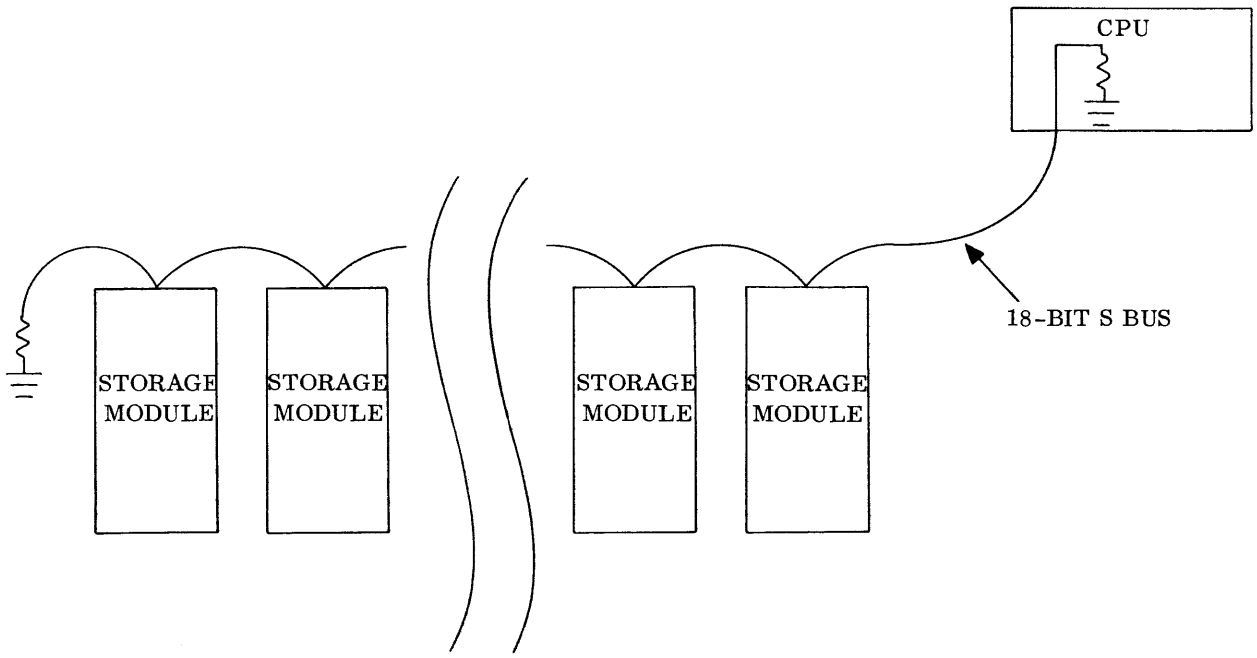


Figure 62. S Bus Scheme if Multiprogramming Module Not Present

For a unique module to honor a storage request, the Module Select code must match the setting of MODULE SELECT switches.

Data Bus

The data bus transmits 24 bits of data to and from the processor and storage register Z. As in the case of the S bus, the data bus is tied in parallel to all storage modules and is also tied into each communication channel. However, the data bus is used only by the module which receives a request from the processor.

STORAGE PROTECTION

Storage protection is the feature of the 3200 System that prevents alteration of a block of storage addresses. When logic in the computation section has determined that a reference is about to be made to a protected location, it changes the write signal (previously sent to storage but not yet used) into a read signal. Therefore, the contents of that location are not altered.

The storage module is given no other indication that any addresses are being protected. Figure 63 shows the system power panel with the STORAGE PROTECT switches.

3309 THEORY OF OPERATION

Two processors may share use of a storage module. To reference storage a processor sends the address of the word desired accompanied by a request signal. A scanner determines which computer has first access to the storage module. The request lines are monitored consecutively, insuring that each processor will be acknowledged in turn, free from interference. If only one processor is used the CONTROL SELECT switch locks the scanner to continually monitor only the right or left bus.

When a request is received storage control gates the 13-bit address from the S bus into the S register where it is translated to select the proper drive lines for a memory reference.

Each of the two memory operations--read from memory and write into memory--consists of:

1. A read phase, during which a word of information is removed from an address in memory, followed by
2. A write phase, during which a word of information (either the same word or a new word) is written into memory at the same address.

The processor will specify either read from memory or write into memory.

1. Read from memory. Memory performs a read/restore cycle in which a word is taken from memory, entered into Z register, and re-stored in its original memory location. While the word is in Z register it is also transmitted to the equipment via the data bus.
2. Write into memory. Memory performs a read cycle, with entry to Z register blocked for all or certain portions of the word from storage. The por-

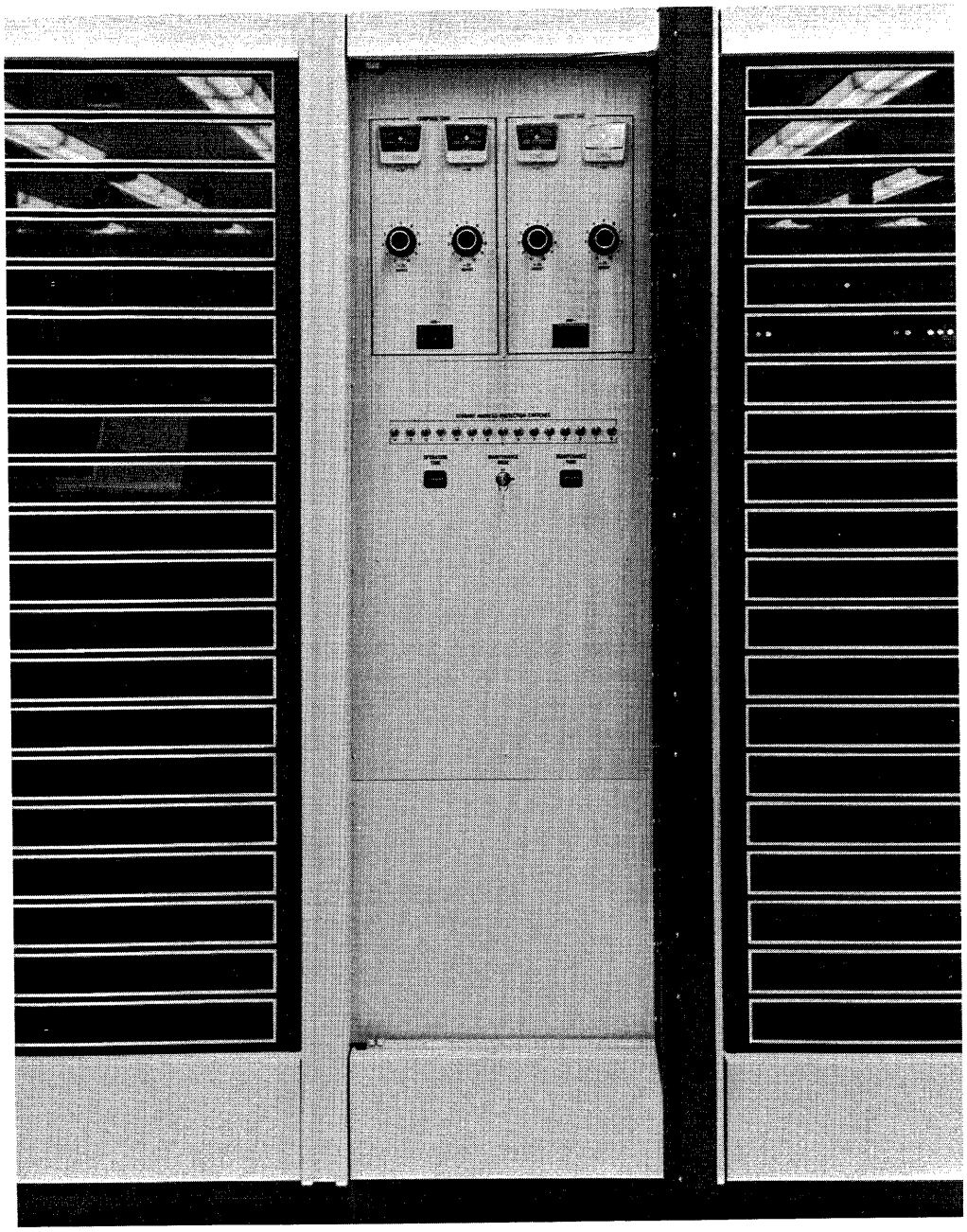


Figure 63. System Power Panel

tions of the word to be blocked from Z are determined by the type of partial write mode selected (selection depends on the instruction or cycle being executed).

The processor transmits updated information via the data bus into Z register and then into the original memory location.

Figure 64 maps out the theory of operation. It shows the basic steps in one complete memory cycle:

1. Set busy FF.
2. Turn on read current.
3. Gate the contents of a storage cell into Z register via the sense amplifiers.
4. Turn off read current.
5. Turn on inhibit current. (Prepare for re-storing information.)
6. Turn on write current. (Re-store information.)
7. Strobe for parity errors. (Parity is only checked early in the write cycle.)
8. Turn off write and inhibit currents. (Re-storing is completed.)
9. Clear busy FF. (Cycle completed.)

Information access time for memory is approximately 750 nsec. A complete storage reference cycle takes 1.25 usec. There are other timing pulses in the memory cycle but the ones listed above are most important.

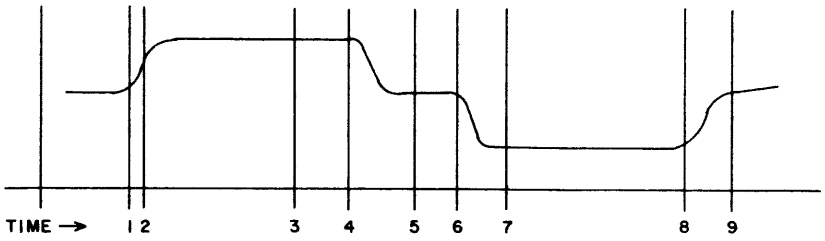


Figure 64.
Storage Reference Cycle Timing

The next portion of the chapter analyzes circuitry in each portion of the memory cycle.

CONTROL SELECT

As described earlier, a storage module may communicate with a processor via the right bus or left bus, or the module may be shared by two processors, one using the right bus and one using the left bus. The CONTROL SELECT switch (figure 65) is used to select right bus, left bus, or priority (a scanner controls use of right bus or left bus in this case).

In the logic shown in figure 65, CONTROL SELECT places a ground on the input of one of the G18x terms, allowing it to output a 1 to indicate which bus is selected. Note that the Y18x terms are sections of a filter card (HA17) and only remove noise from the signal. They do not delay or invert. The following discussion of the scanner explains the use of the signals from G18x.

Scanner

Two processor modules may share a storage module. One processor communicates with the storage module via the left bus, the other processor communicates with storage via the right bus. Storage requests are serviced on a first-come, first-served basis by the storage module. A two-position scanner determines which computer has access to the storage module at any given instant. The scanner first checks one line, then the other for a request. When a request is detected (R620 or R621 = 0), the scanner is disabled. K792/793 is then held clear for a left bus request or held set for a right bus request. (See figure 66.)

If the scanner recognizes a left bus request and the busy signal is not present, inverters G070-G072 output 1's, gating the address on the left address bus into S register where it is translated. G072 sets:

1. K796/797 - indicates left bus selected.
2. Reply, K780/781 - indicates acknowledgement of request.
3. Pulse generator, K750/751 - starts storage reference sequence.

When a right bus request is recognized and busy is down, inverters G080-G082 output 1's which gate an address from the right address bus to S register. The 1 from G082 sets:

1. Reply, K780/781, and
2. Pulse generator, K750/751.

It also clears K796/797 to indicate right bus selected. The scanner reactivates when the request is dropped (which occurs under central processor control after it receives the reply pulse from the storage module). The scanner is active only when CONTROL SELECT is set to priority. If only one processor is being used CONTROL SELECT should be set to select the related bus. This will disable the scanner.

Review the circuitry that has been covered so far by reading these statements:

1. Control select circuitry causes the following scanner actions according to switch settings:

   Position 1 disables the scanner (K792/793 clear, K794/795 clear) and selects left bus (G182 sends constant clearing input to K792/793).

   Position 2 disables the scanner (K792/793 set, K794/795 set) and selects right bus (G183 sends constant setting input to K792/793).

   Position 3 causes the scanner to flip back and forth constantly and to lock when a request comes from the processor. Receivers R620 and R621 statically have 0's out which will allow the scanner to run. Receipt of either request, left or right, causes the output of G176 or G177 to drop to a zero, stopping and locking out the scanner to one of the following positions:

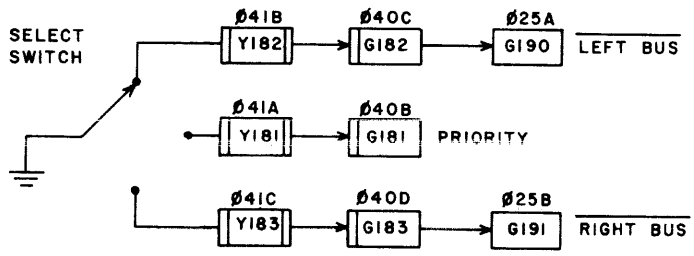


Figure 65. Control Select Switch and Logic

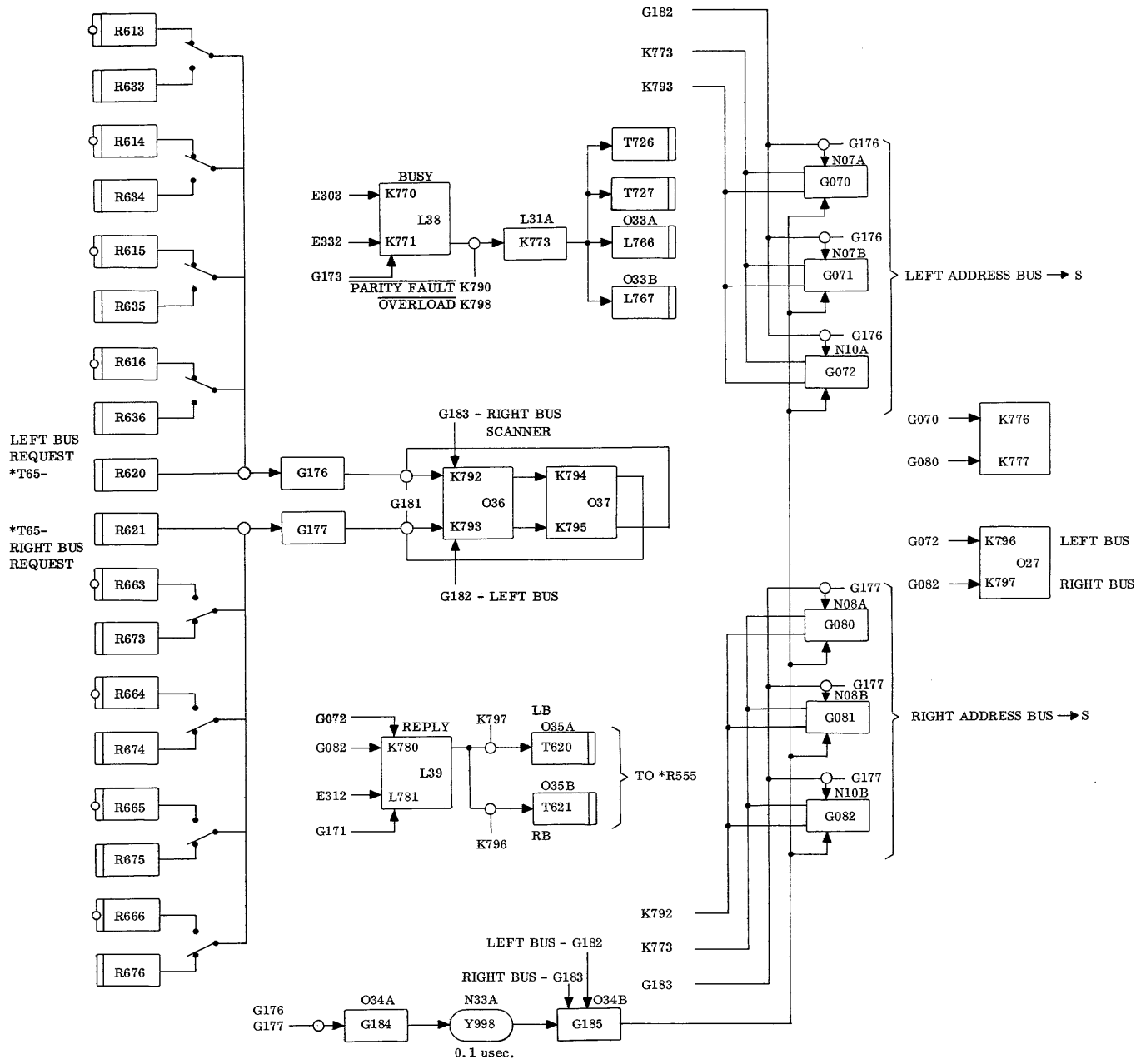


Figure 66. Left and Right Address Bus → S Control Logic

1. left request (G176 output going to a zero), the scanner will not set K792/793, leaving it cleared. K794/795 is also clear and the scanner remains that way until the request drops. This stopped position is that which the scanner assumes if CONTROL SELECT is in position 1.
2. The scanner locks into a position and a left or right request allows the outputs of G070, 071, 072, or G080, 081, 082 terms to come to a 1 level. This allows the address (13 bits) on the S bus to enter the S register.

   Assume Bit 00 is to be set. The central processor holds the 13-bit address on the S bus line (either left or right). When G070 or G080 outputs 1, information is transferred from the S bus via the receiver cards to the S register. The S register is cleared at the end of a memory cycle if there was no parity error or if parity error recognition has been disabled.
3. The output of G072 going to 1 sets the left/right bus FF (K796/797). The output of G082 going to 1 clears the left/right bus FF (K796/797). The output of either G072 or G082 sets the reply FF (K790/791). The reply FF informs the processor that storage has received and is processing the request. Within the CPU, the reply signal drops the request to storage and causes sampling of the data bus if necessary. With the request dropped, the scanner again runs if CONTROL SELECT is in the priority position.

   Outputs of either G072 or G082 going to a 1 also cause the memory cycle delay line to start.

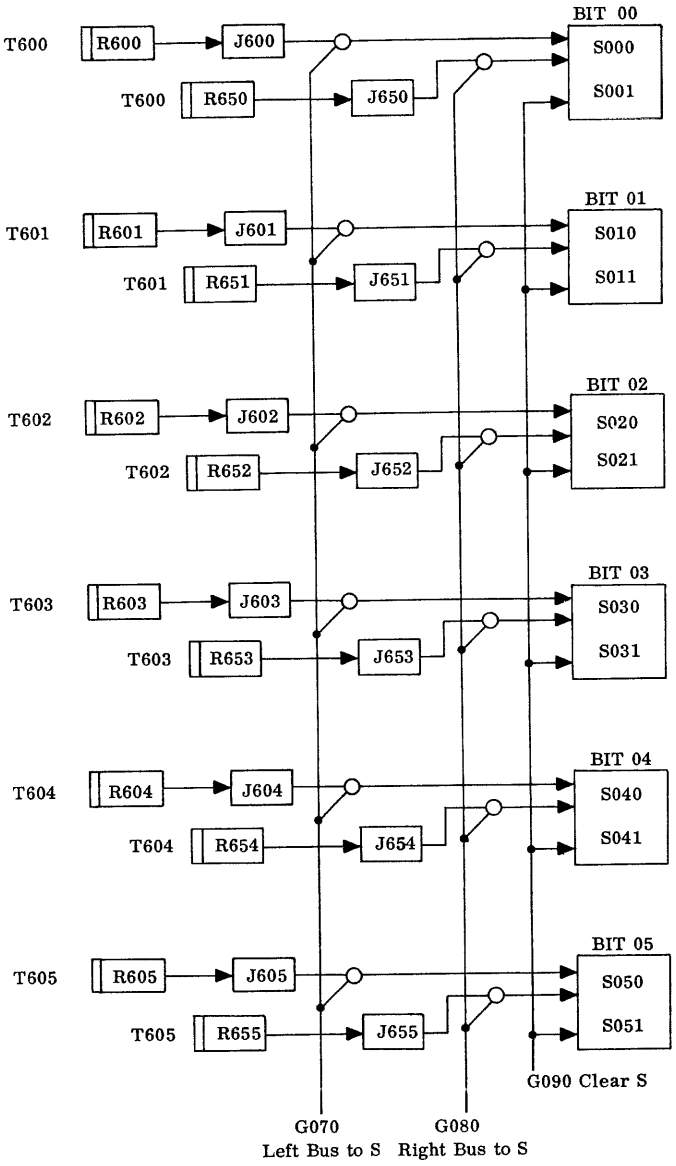


Figure 67. S Register Input Logic

DELAY LINE

The memory delay line supplies necessary time intervals to accomplish reading out of information from memory and writing back of information into memory.

Before proceeding, review the major timing intervals of a storage reference cycle (figure 64).

The 3300 Computer System uses coincident storage principles. When read or write current is turned on, a previous gating or selecting process must be used to select the X and Y drive lines that will carry read or write current. These and other operations will be developed as a pulse is followed down the delay line.

Each operation will be covered when the pulse reaches that point in the delay line. Considerable time will be spent in each area and all circuitry involved then will be covered.

It is a good idea to "step back" occasionally to keep track of our objective, which is to read the contents of a memory location and re-store it in memory. It is also a good idea to review operations and circuitry previously discussed.

Assume, for example, that the processor needs to use storage. Our study of the circuitry has shown:

1. How the request was received.
2. How storage notified the processor that the request had been received and was being processed (reply).
3. How the necessary information from the S bus, the field and coordinate address, was sent to S register.

Now let's take a look at the actual memory cycle delay line.

Delay Line Timing

The timing for a storage reference cycle is provided by a pulse traveling down a delay line. Signals are available at various taps on the line as the pulse passes by them. These signals are used to time the storage reference cycle. The input to the delay line is normally held at a steady 1 by D150 (see figure 68A). When the pulse generator FF is set (by either G072 or G082), its set output drops the output of D150 (a CA08 card) to a 0, starting a pulse down the delay line. (For further information refer to 3000 Printed Circuits Manual. The clear output of the pulse generator simultaneously causes Z to clear.

The first tap (A3) from the delay line (time 1 of figure 64) will:

1. Set Activate Read FF (K700/701, figure 68B) which enables the selected line drivers.
2. Set Gate FF (K710/711, figure 68C) which enables the selected gates to the drive transformers to begin the read phase of the cycle.
3. Set Busy FF (K770/771) which turns on the storage active status display.

The Busy FF (K770/771) clear output is inverted by K773 and disables the B07x and G08x inverters to prevent a new address from being transferred off bus to S. This can be seen in figure 66.

Figure 69 is a circuit within the 3309 that generates a Master Clear if the module has been busy longer than two microseconds. This logic also prevents Master Clears from occurring during a memory cycle. The output of G169 will be a 1 only if storage is Not Busy ($\overline{\text{Busy}}$) or if the Busy signal has been up longer than 2 usec.

Return for a moment to the Activate Read FF (K700/701, figure 68). When this FF sets, the selected line drivers are turned on. Gate FF (K710/711) turns on the gates. How the drivers and gates were selected is the next topic to be discussed.

ADDRESS SELECTION

Information stored in memory is held in cores located at intersections of X and Y half-current drive

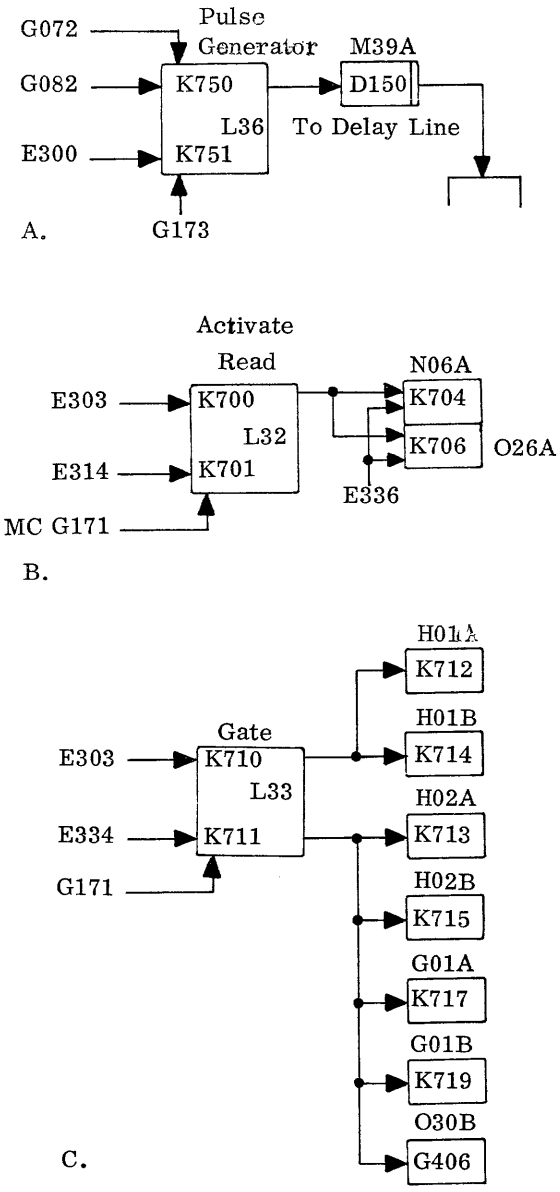


Figure 68. Pulse Generator, Activate Read, and Gate Logic

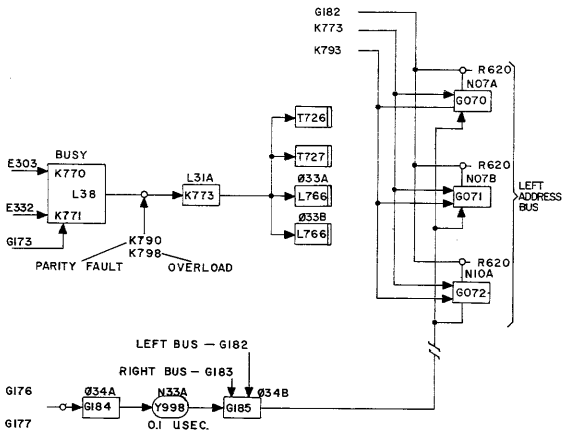


Figure 69. Busy FF and Left Address Bus → S Control Logic

lines. The 4096-word memory module has 64 X and 64 Y drive lines. In order to reference a location, the 13-bit address in S register must be translated so that one X and one Y drive line are energized.

The outside appearance of an 8K memory stack is shown in figure 71. An 8K stack is made up of two identical 4K fields. In figure 71 visually follow the cable coming up from the lower right corner and note the flanged area immediately behind it. This flange separates the two fields. They can be disassembled by removing the cables and then the connecting bolts.

MEMORY PLANES AND WAFERS

The wires threading the memory cores are strung across the center areas of square mounting frames to form memory planes. The wires used to drive and sense the cores are also used to hold the cores in place.

A memory plane is a matrix of 4096 cores representing one bit of the 28-bit computer word. Each plane has 64 X drive lines passing through 64 columns of cores with each drive line threading all 64 cores in one column. The Y drive lines are threaded similarly but at a 90° angle to the X drive lines.

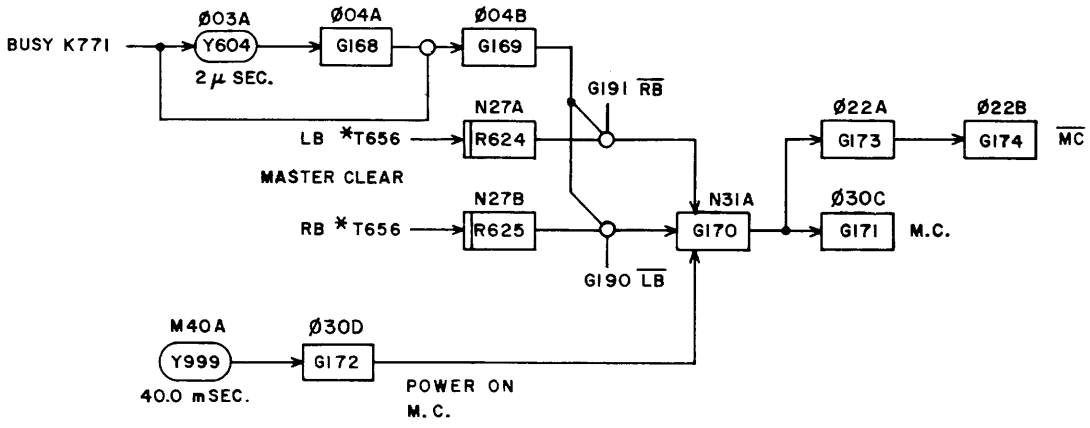


Figure 70. 3309 Master Clear Control Logic

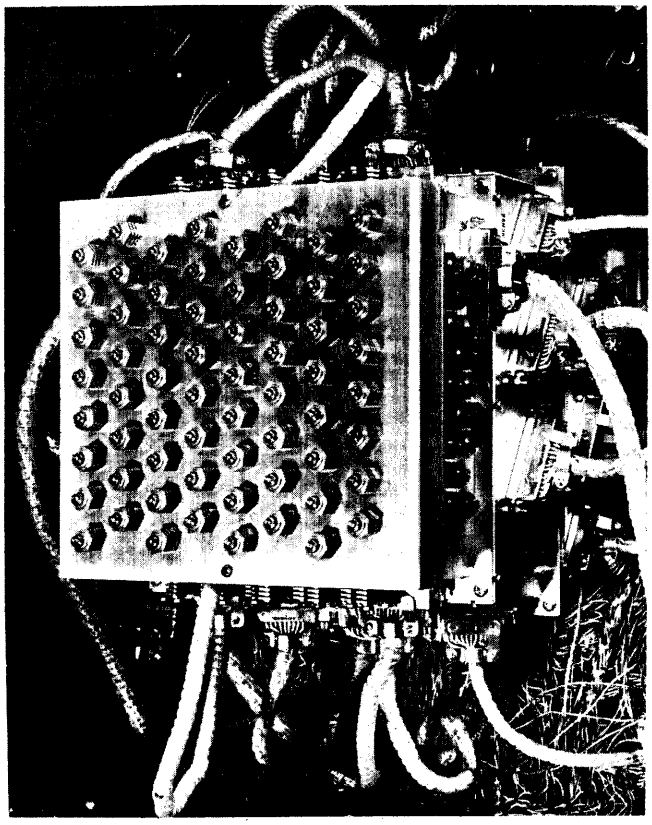


Figure 71. 8K Memory Stack

A memory wafer is formed by wiring a four-section frame so that it contains four distinct bit planes (figure 72). Each memory plane contains one bit of a word, so a wafer contains four bits.

MEMORY STACK

The cubic memory stack is formed by nine wafers mounted one behind another. The front wafer, the input or terminal wafer, serves as a terminal point for the X and Y drive lines. The bit plane wafers follow and may be numbered sequentially from front to back beginning with 0. Bits 0-23 are contained in the first six bit plane wafers (0-5). Wafer 6 contains the bit planes for the four parity bits. The last wafer serves as a transposition plane where X and Y drive lines are transposed from one section of the stack to another. The 64 X or Y drive lines are arranged in four stripes of 16 lines each, eight even and eight odd. Each stripe is associated with one inhibit line. To reduce noise the odd drive lines within each stripe are transposed at the transposition wafer. The first four odd drive lines change relative positions with the last four so they return to the input wafer threaded between different even-numbered drive lines. Exploded views of the stack are shown in figures 73 and 74.

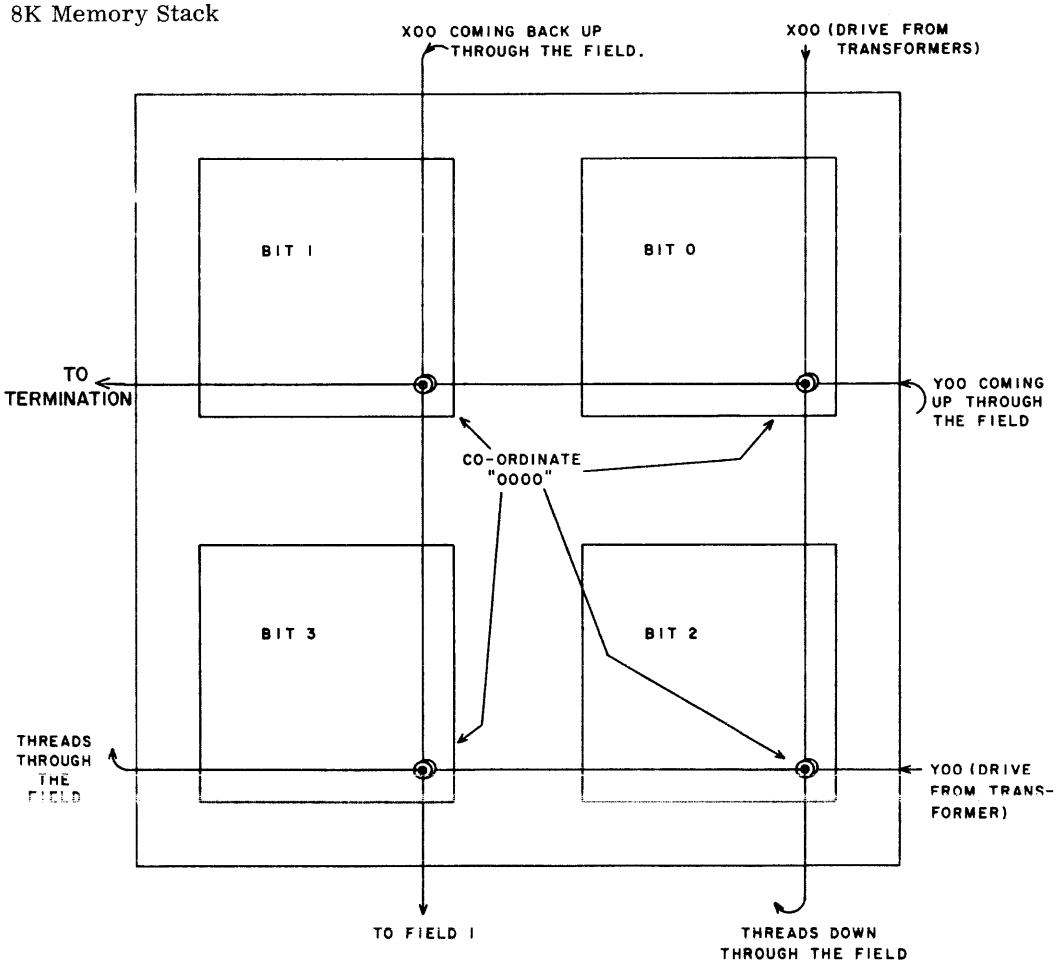
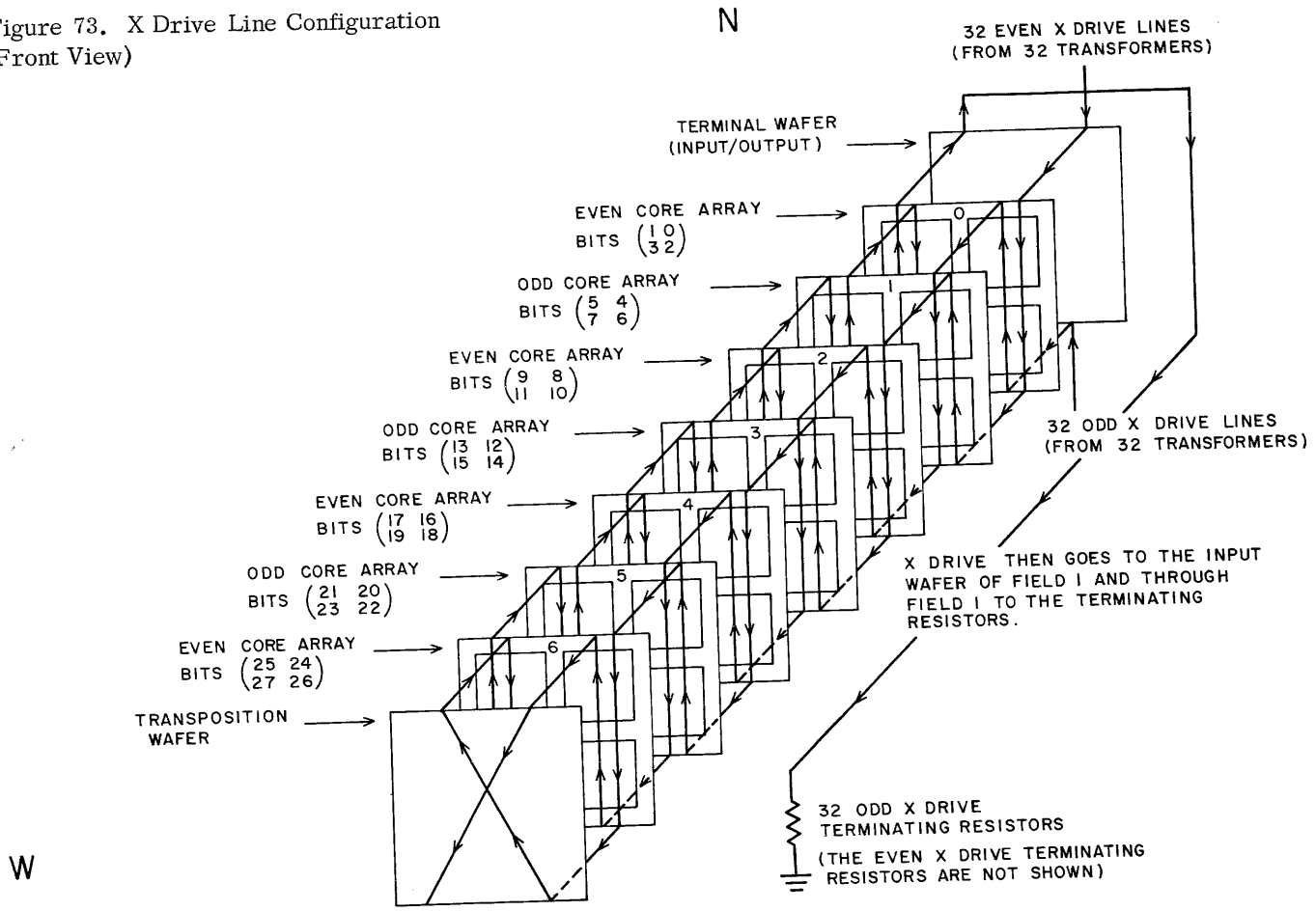


Figure 72. Memory Wafer

Figure 73. X Drive Line Configuration
(Front View)



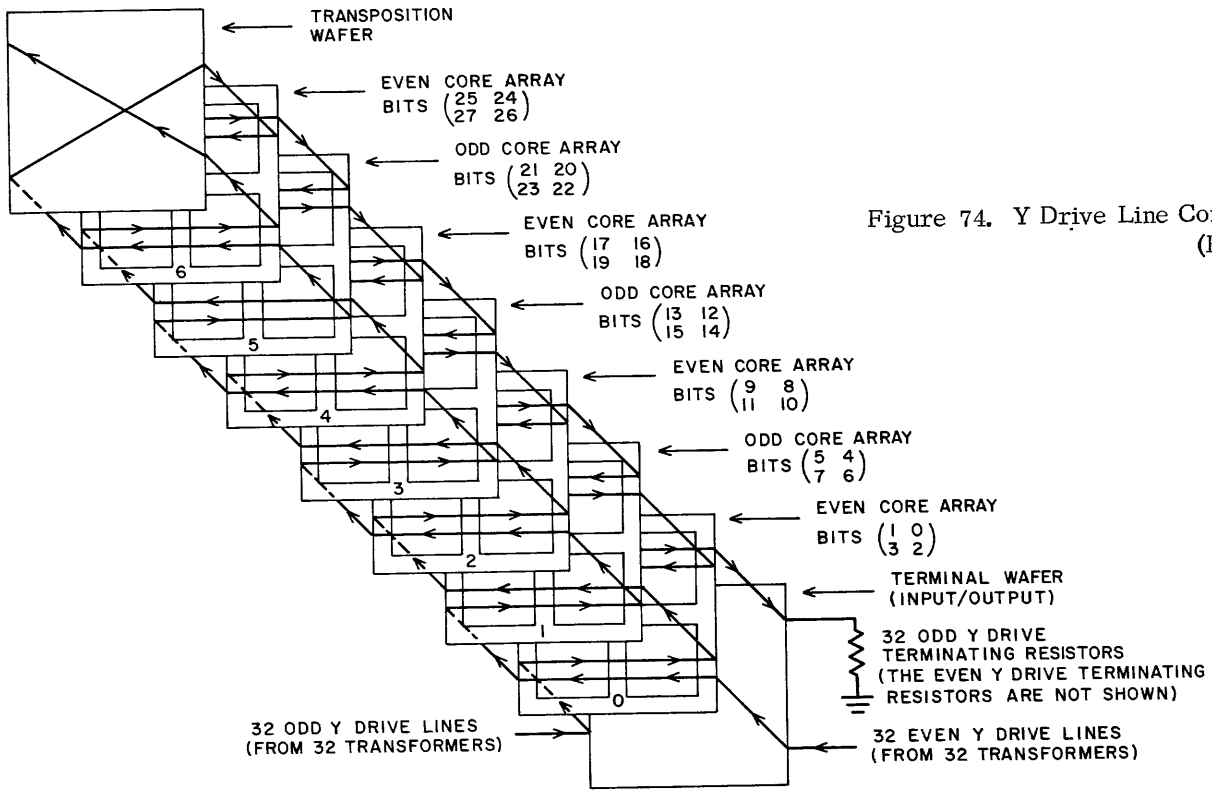


Figure 74. Y Drive Line Configuration
(Rear View)

Two 4096-word stacks (fields 0 and 1) are mounted face to face, making one 8192-word stack. These memories work together with common X drive lines and separate Y drive lines. Although the two stacks are physically identical, for easier wiring the bit planes of field 1 are rotated 90° clockwise from those of field 0. The X drive lines of field 1 are inverted from those of field 0 (figures 75 and 76). The odd Y drive lines enter fields 0 and 1 from the left. The even Y drive lines enter fields 0 and 1 from the right (figures 75 and 76).

Terminating resistors for the X drive lines are mounted on the rear panel of the stack and those for the Y drive lines are on the side panels.

Figure 72 is a simplified layout of four bits and the selection method via X and Y coordinates in field 0. Note that the Y drives terminate in this field (0) while the X drive line continues into the next field (1).

Separate connections are made to the sense and inhibit lines of each memory bit plane. Each sense line extends to a pair of tabs at a corner of the memory board. These tabs are connected to the sense amplifier circuits by twisted-pair wiring. Each inhibit line is brought out to a pair of tabs on the memory board. These tabs are connected to the inhibit circuits by twisted-pair wiring.

Drive Lines

The 64 X drive lines enter the two right planes of wafer 0 from the terminating wafer and thread through all the cores in the 14 right planes. Each drive line threads all the cores in one column of a plane, then is jumpered to the corresponding column of cores in the next plane.

At the last wafer the X drive lines are transposed and returned to the input wafer through the left planes of the memory wafers. The X drive lines are common to fields 1 and 0.

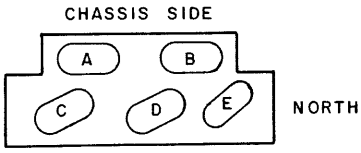
Fields 1 and 0 have separate Y drive lines. Only the Y drive lines for the selected field are energized at any one time. These lines thread the wafers in much the same manner as the X drive lines are threaded. The Y drive lines in field 0 run from input wafer to transposition wafer through the lower two planes and return through the upper two planes. The Y drive lines for field 1 are threaded from input wafer to transposition wafer through the upper two planes and return through the lower two planes.

For easier construction the even-numbered drive lines enter one side of a wafer while the odd-numbered drive lines enter from the opposite side (figures 75 and 76).

The half-current drive lines are energized in the same manner as transmission lines. Characteristic impedance of a drive line is approximately 130Ω, so each line is terminated with a 130Ω, 10w, noninductive resister. The lines are energized by pulses of 22-25v and the resulting half currents are about 340 ma.

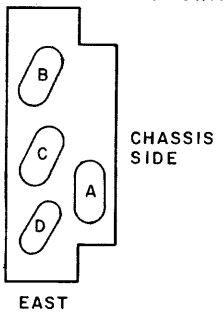
STACK CONSTRUCTION (field 0, 4K)

1. Each connector on the north (top) side of field 0 serves the following purpose:



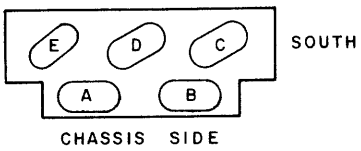
A. Odd X drive to field 1
B. Even X drives from transformers
C. Inhibit stripes 2 and 3 (28 wires)
D. Inhibit stripes 0 and 1 (28 wires)
E. Sense (7 twisted pairs)

2. Each connector on the east (right) side of field 0 serves the following purpose:



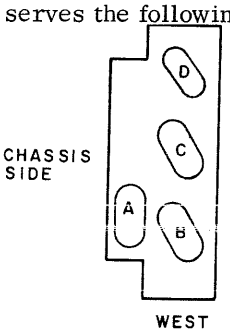
A. Even Y drive lines input from transformers
B. Inhibit stripes 2 and 3
C. Inhibit stripes 0 and 1
D. Sense

3. Each connector on the south (bottom) side of field 0 serves the following purpose:



A. Even X drive output to field 1
B. Odd X drive from transformers
C. Inhibit stripes 0 and 1
D. Inhibit stripes 2 and 3
E. Sense

4. Each connector on the west (left) side of field 0 serves the following purpose:



A. Odd Y drive lines from transformers
B. Inhibit stripes 0 and 1
C. Inhibit stripes 2 and 3
D. Sense

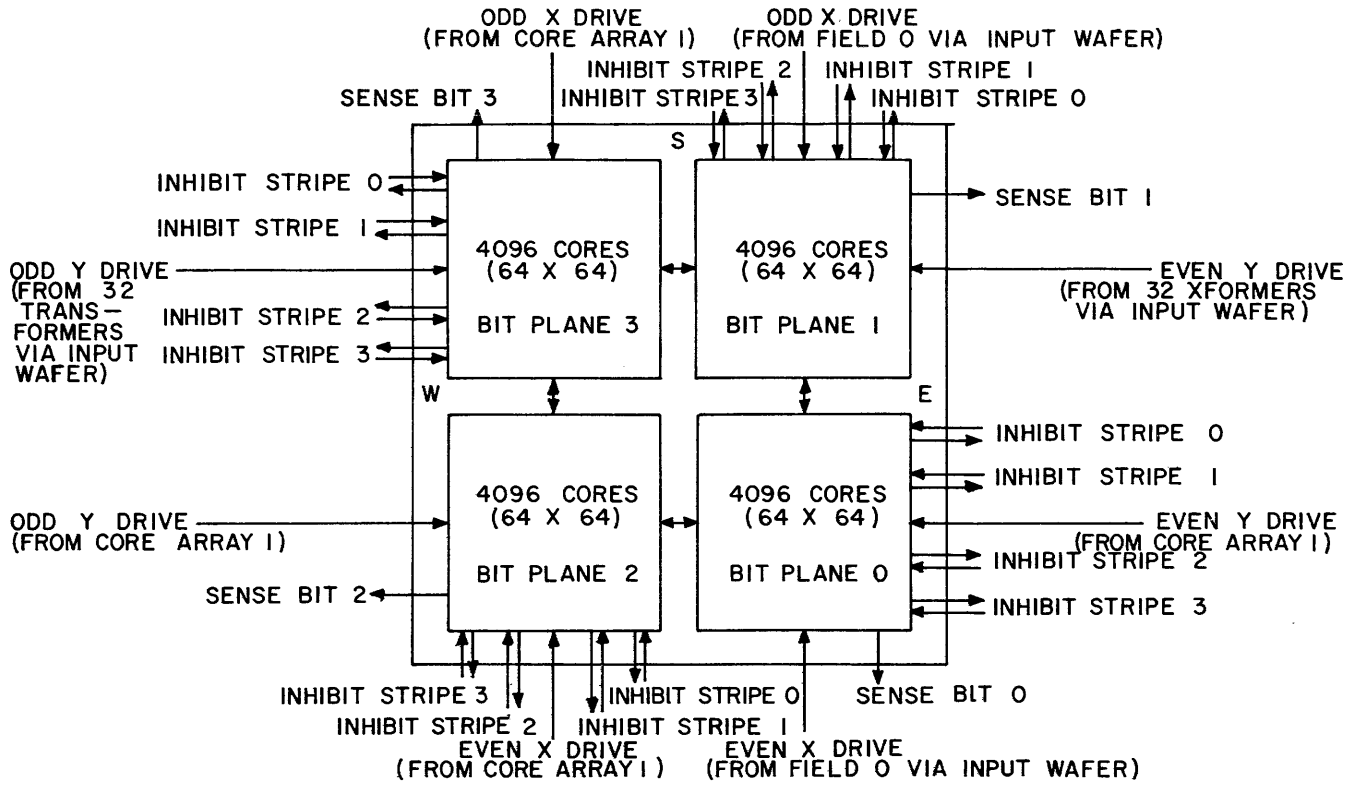


Figure 75. Even Memory Core Array, Field 0

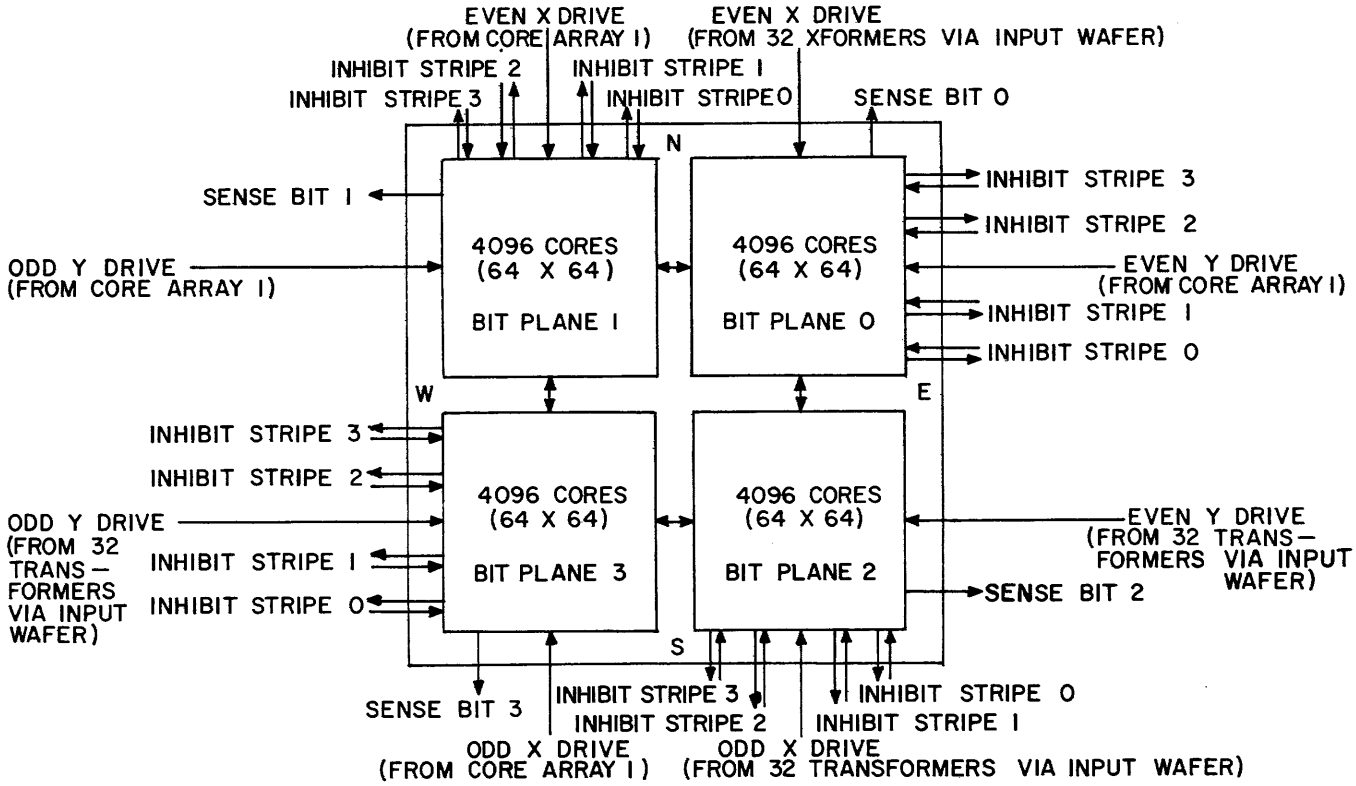


Figure 76. Even Memory Core Array, Field 1

Drive Compensator

In order to hold a constant load on the memory power supplies, load compensator circuits are enabled when the memory drive circuits are not carrying current.

When neither read nor write is being performed the X and Y drive lines are turned off and the load compensator is enabled. The load compensator consists of two C03 line driver cards (two sections per card). Each section drives a current of approximately 400 ma through a 50Ω, 10w, noninductive load resistor.

Inhibit Lines

Since the drive lines affect simultaneously all 28 cores in a selected storage location it is necessary to negate those lines affecting any bit position which must remain in the clear state. The inhibit line carries a 340 ma current parallel to one of the half-current drive lines, but in a direction opposite to the half-write current. Therefore, if the inhibit current is flowing during write phase it will cancel one of the half-write currents and will prevent any core through which it passes from being switched to the set state.

The inhibit lines in each plane are arranged in four groups or stripes. Each stripe is 16 cores wide and extends all the way across the plane; thus, each inhibit line threads 1024 cores in a 4K field. Inhibit stripes are common to fields 1 and 0; thus, each inhibit stripe threads 2048 cores in an 8K stack.

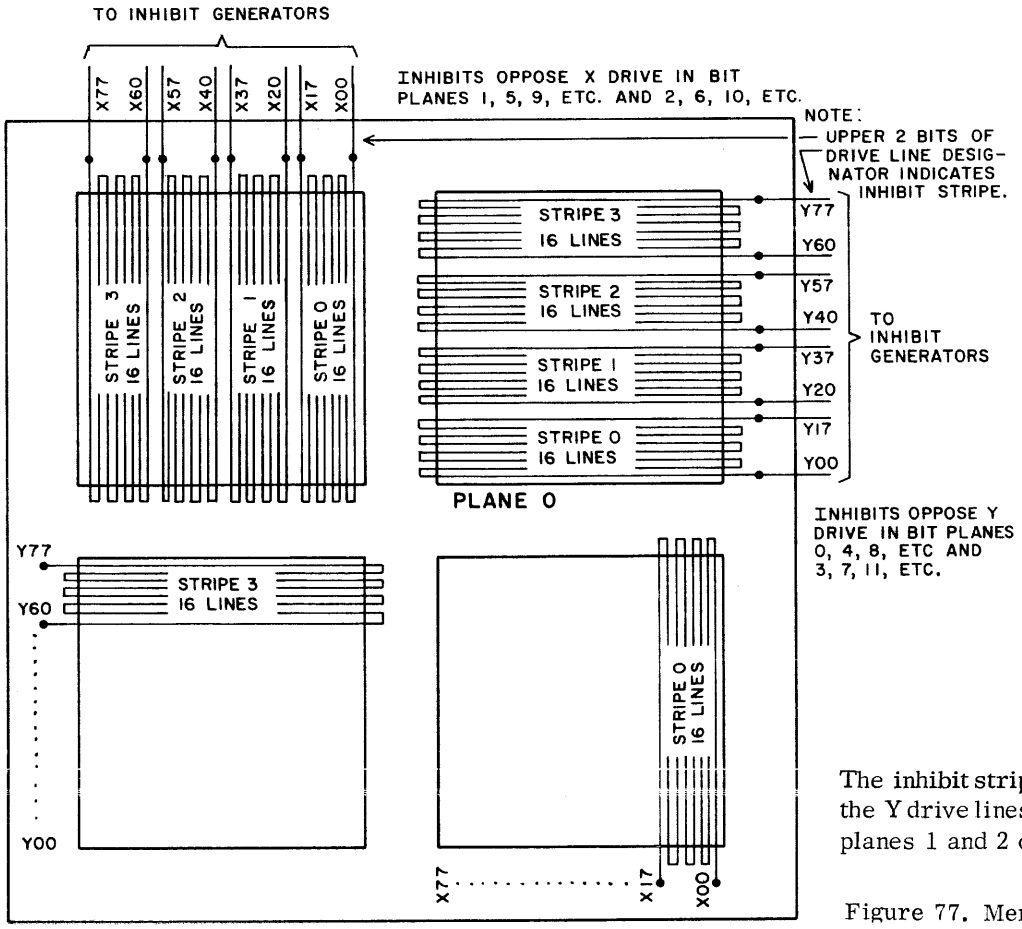
Four inhibit lines are needed to thread the 4096 cores for each plane (figure 77). The inhibits are not jumpered together and each inhibit winding appears in one plane only.

Use of four stripes per plane reduces the number of cores disturbed by an inhibit pulse, thus reducing noise generated in the sense amplifier windings.

Inhibit Compensator

As in the case of the drive lines, load compensator circuits are enabled during times when the inhibit circuits are not carrying current. This holds a constant load on the memory power supplies.

The inhibit circuits include a load compensator for each plane in the inhibit driver selection scheme. Four inhibit driver circuits are required per plane and a fifth circuit is included as a compensator. This fifth circuit is an inhibit compensator, card type C09. The inhibit circuits are selected one at a time through translations from S and Z registers. When all four inhibits to a plane are turned off, the inhibit compensator is enabled. The compensator circuit drives 340 ma current



The inhibit stripes in planes 0 and 3 oppose the Y drive lines while the inhibit stripes in planes 1 and 2 oppose the X drive lines.

Figure 77. Memory Wafer-Inhibit Stripes

through a 133Ω, 25w, noninductive resistor. The 133Ω resistor is approximately equal to the sum of the 13Ω DC resistance and 120Ω terminating resistance of an inhibit line.

Sense Lines

A simplified example of a sense winding is shown in figure 78. One sense line threads all cores in a memory plane. The ends of this continuous line are extended to terminals near a corner of the plane.

The rapid flux change which occurs when a core switches state induces a voltage of approximately 35 mv on the sense line. This voltage appears across the differential amplifier inputs of the sense amplifier (card type HA18) and results in a logical 1 at the amplifier output terminals. This output occurs when any core in the plane switches state in either direction, but the output is sampled only during read phase.

The flux density of half-selected cores is changed slightly by a read drive pulse and induces noise voltage on the sense line. Noise voltage is reduced by threading the sense line through the cores of a plane so that noise signals from half-selected cores cancel each other.

Using the previous material as an introduction to physical characteristics of a memory stack, we now come to the actual electronics of the stack. Each circuit will be studied completely and actual prints will be used to show selections.

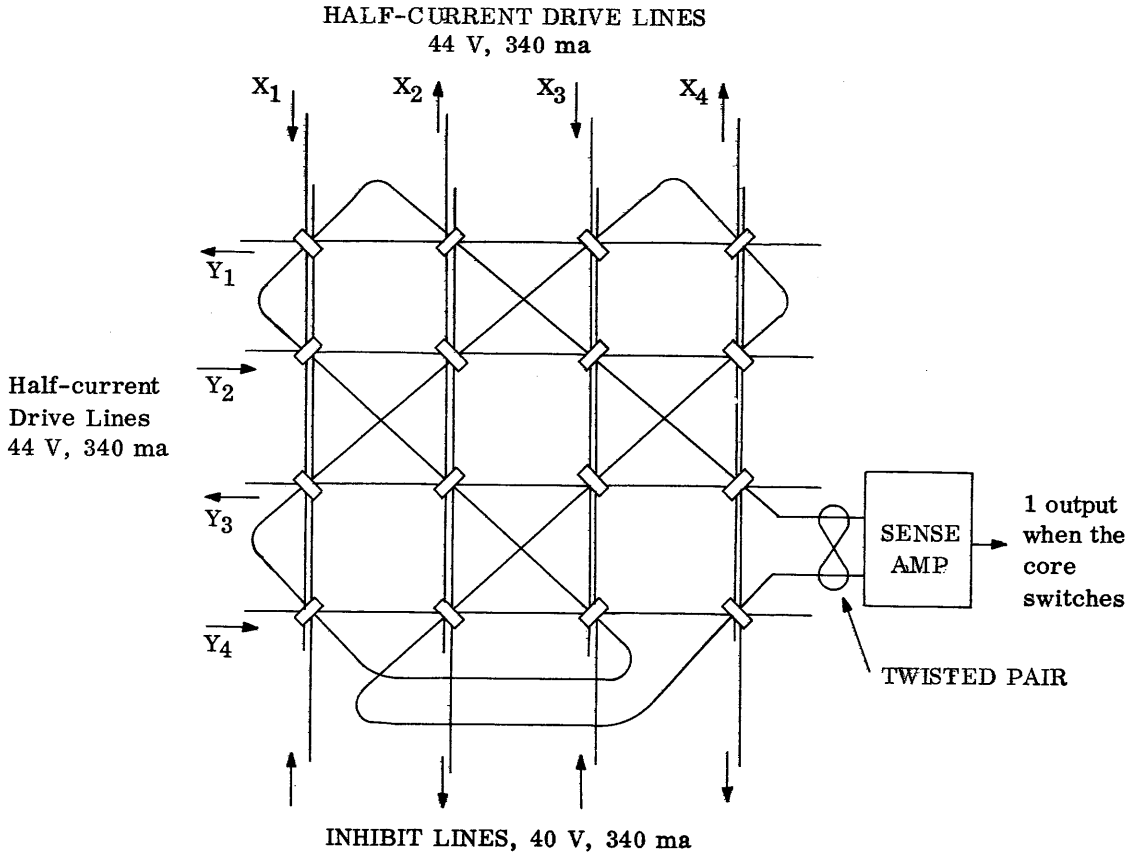


Figure 78. Simplified Sense Quadrant

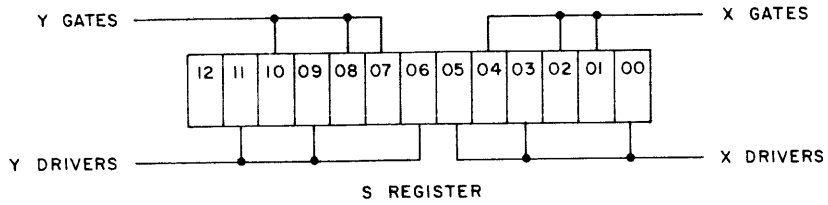
S Register

The S (storage address) register holds the storage address during a storage reference. The register consists of 13 single-rank flip-flop stages and has no properties other than storage. An address may be entered into S from either the right or the left address bus. These addresses are gated into S (by the G070, G080 terms) at the same time a signal is sent to begin delay line timing. S is cleared at the end of each memory cycle if there is no parity error.

Line Driver Selection

Three bits from S register select the line driver circuits (figure 79) and either read or write. Bits 0, 3, and 5 select the two circuits in the X coordinate. Bits 6, 9, and 11 select the circuits in the Y coordinate. The translation is such that the selected pair of line driver circuits will have all 0 inputs.

The circuitry in figure 80 selects the X drivers and gates: Bits 0-5 of S register determine which X line is selected. For the time being, it will require persistence to get the outputs from the terms in the X gates and X transformer drivers. Inverters G300-G305 must output 0's to be considered turned on (assume E336 is 0). The X gate inverters (G000-G007) must have 0's



The storage address converts directly, giving the selected drive line numbers in octal. Bits 00 through 05 select the X drive line; bits 06 through 11 select the Y drive line; bit 12 selects the Field.

Figure 79. Storage Address Bit Assignment

in to turn on. The X transformer drivers (D000-D015) also need all 0's in to turn on. One gate and two drivers must be turned on to select a drive line.

Work the following problem through the circuitry for address 03765:

1. Which X gate card would output a 1? (The K712 and K715 terms are from the outputs of gate FF, K710/711, discussed earlier.) _____

2. What would be the outputs of these terms? (The K701 term comes from activate read FF discussed earlier. The K731 term comes from activate write FF which will be presented later; for now consider the output of K731 to be 0.)

   G300 = _____ G302 = _____ G304 = _____
   G301 = _____ G303 = _____ G305 = _____

3. Which X transformer drivers would be outputting logical 1's? (K700 has been discussed previously and K730 will be discussed later. For this problem, K700 will be 0 and K730 will be 1.) _____ and _____.

Carry these outputs to the following circuits, showing the X drive transformer. Determine which T card will turn on. To turn it on, you will need the X gate and two X transformer drivers.

You should have arrived at T326. If you did not, check your work. There are Y drive lines to consider yet, and we may as well enter into these.

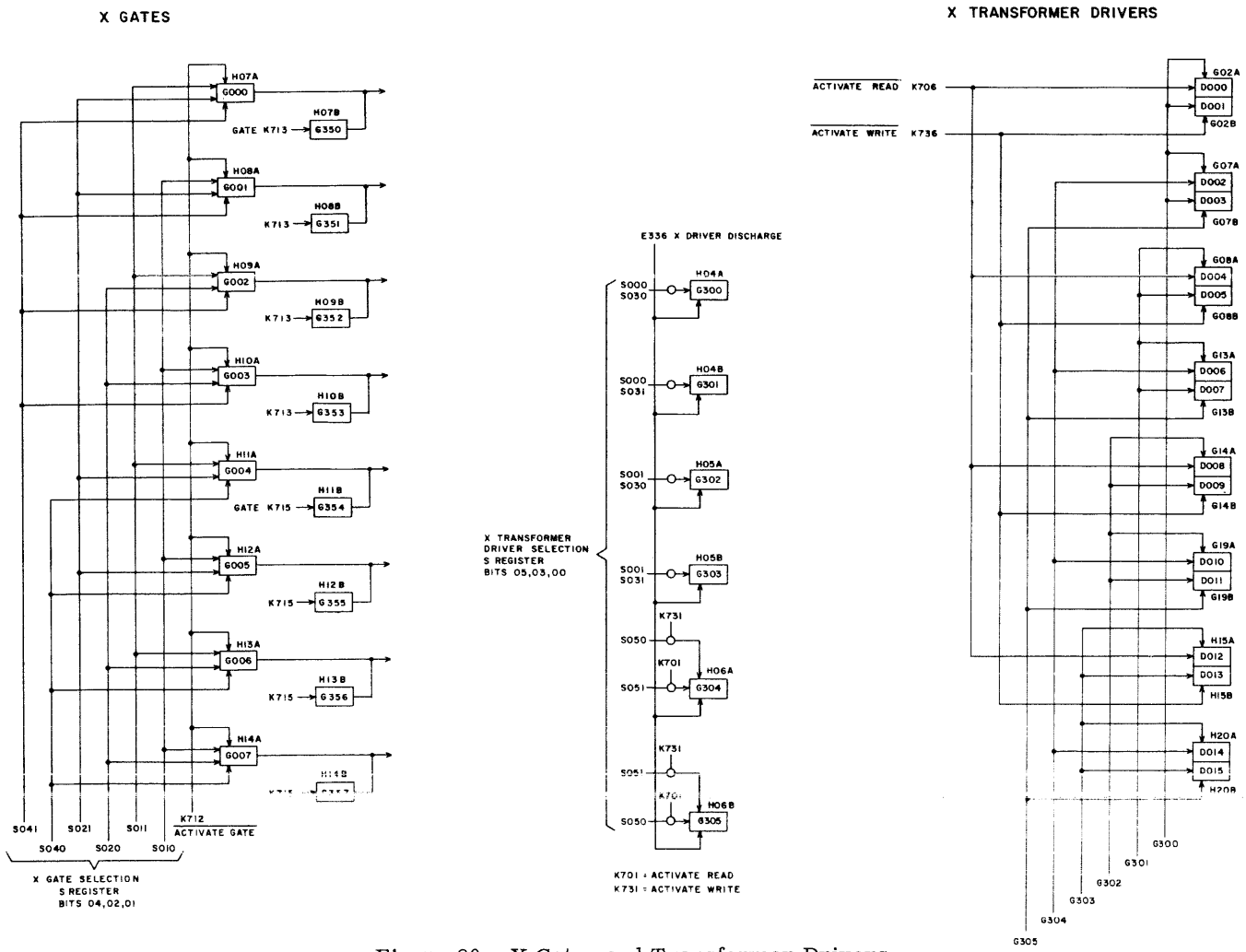
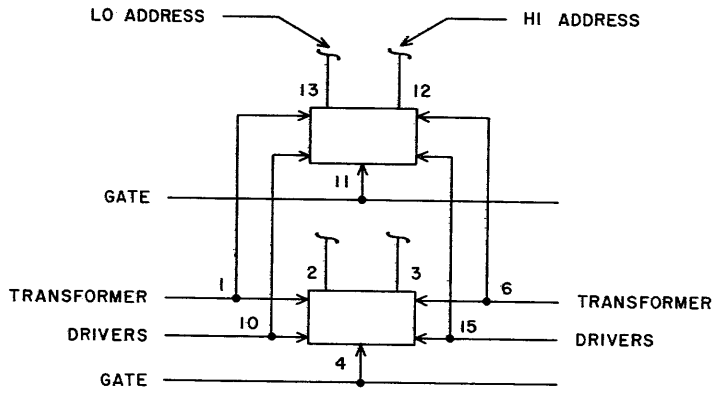


Figure 80. X Gates and Transformer Drivers

Address Translation

Logical translation is such that the number of the selected X drive line is identical to the number contained in bits 0-5 of the 13-bit storage address. Bits 6-11 specify the Y drive line number. This may easily be seen by converting to octal. Bit 12 of the storage address selects the field to be used. Bit 12 is a 1 to select field 1 and a 0 to select field 0.



Look at the top of T326 again. On the right side at the top you will find a 53 (decimal) and a 65 (octal). The same will be true for any address line for either X or Y.

To further break the transformer card down, a circuit is described in figure 81. It is a transformer card with the pins numbered.

Input Pins	Addresses
1 and 15	Read lo
10 and 6	Write lo
1 and 6	Read hi
10 and 15	Write hi

Figure 81. Transformer Card with Pin Numbers

Gate numbers are quickly determined by lifting out the specific bits that select the gate and aligning them in binary, then reading them in octal. The example that was worked through used G006. Take bits 4, 2, and 1 from S register for the address we want (65) and consider what they represent:

```
      110        101
       ↗         ↗ ↖
Bit 4      Bit 2  Bit 1
  1          1      0     = 6 or G006
```

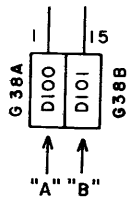
Find the transformer number that will be used for Y lines for address 03765. Use the following circuitry and the octal shortcut:

Transformer for read ____ (Were you in the right field?)
for write ____
Gate for read _____ (Were you in the right field?)
for write _____
Transformer drivers for read _____ and _____
for write _____ and _____

You should have had T437, G107 for both read and write, D112, D115 for read, D113, D114 for write. If you didn't, check yourself by working the actual circuit.

The only aspect of circuitry left to simplify is finding some easy way of determining which transformer drivers will be turned on. The following table will help, if you remember Hi and Lo address lines (Lo = 00-37, Hi = 40-77).

Table 10 shows that driver A is always on during a read operation and driver B is always on during a write operation. C or D determines whether a Hi or Lo address is read or written.



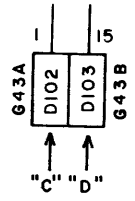


Table 10. DRIVE TRANSFORMER SELECTION

OPERATION DESIRED	TRANSFORMER DRIVERS ON			
	A	B	C	D
Read Lo address	Yes	No	No	Yes
Write Lo address	No	Yes	Yes	No
Read Hi address	Yes	No	Yes	No
Write Hi address	No	Yes	No	Yes

It can be seen in the table that Driver A is always on in a Read operation, and Driver B is always on in a Write operation. If Driver A is always on for a Read operation, the difference between reading a Hi or Low address is determined by the selection of C or D. The same principle applies to writing. B is always on during a Write operation and Hi or Lo is controlled by C and D.

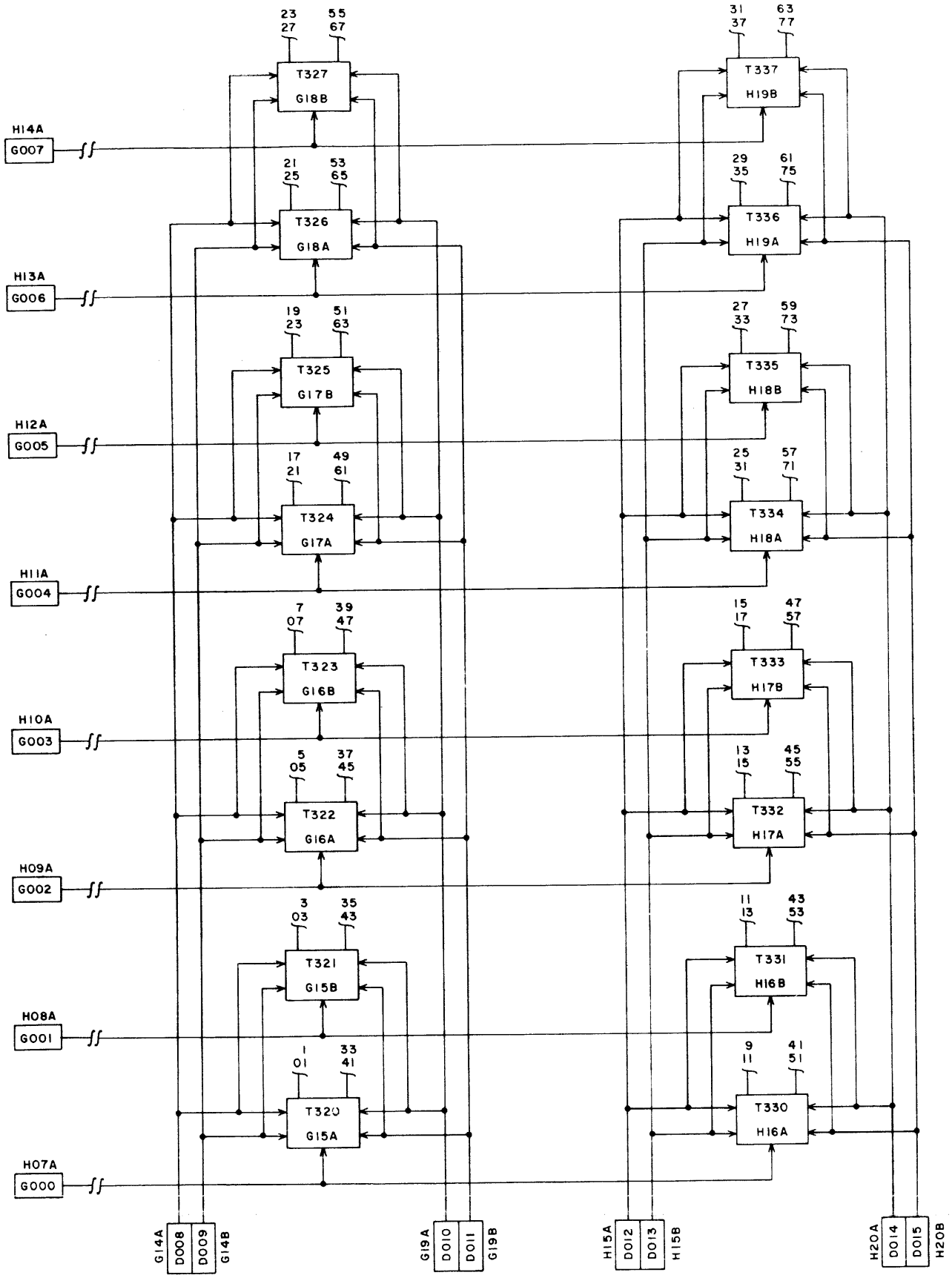


Figure 82. X Drive Line

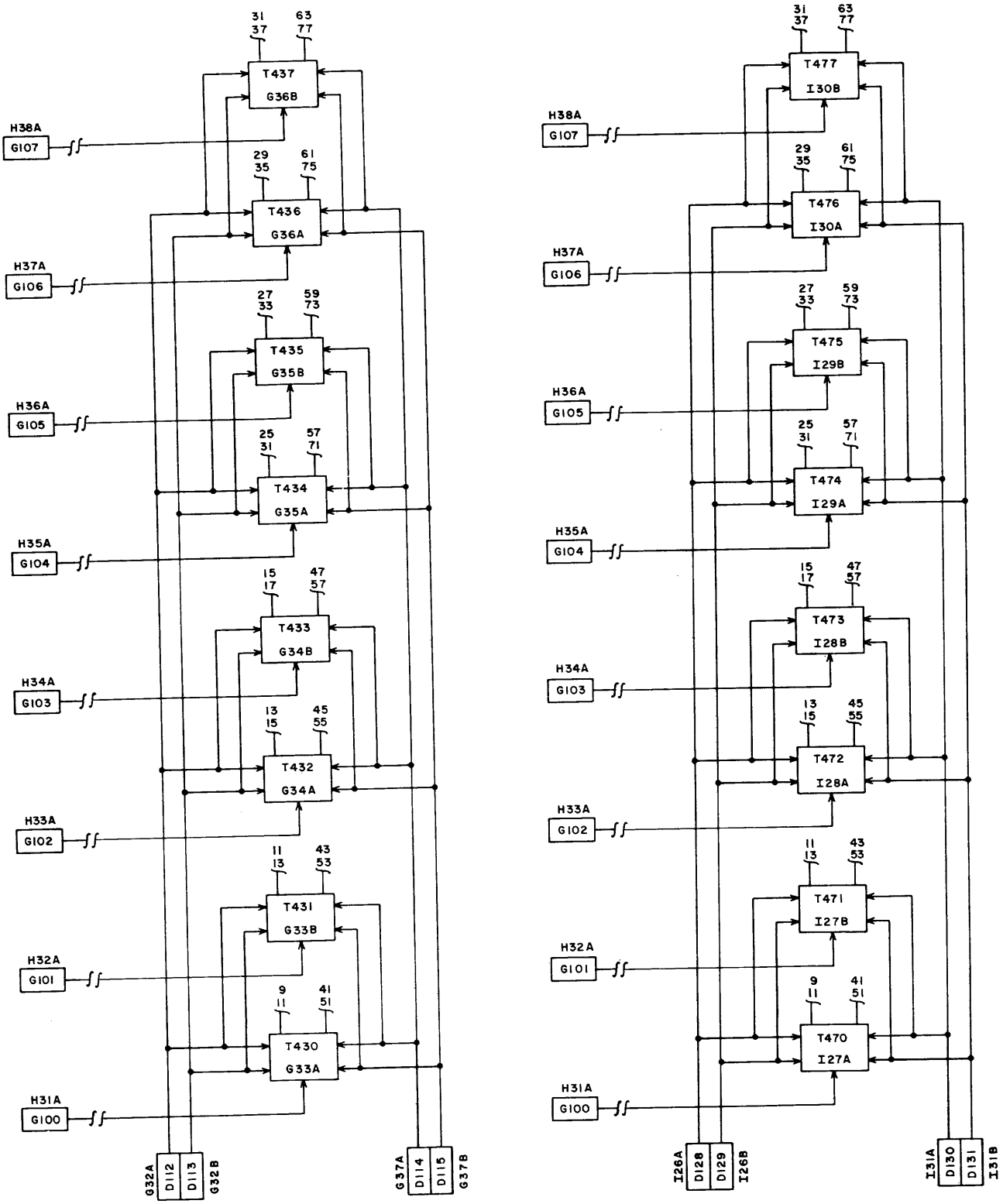


Figure 83. Y Drive Line

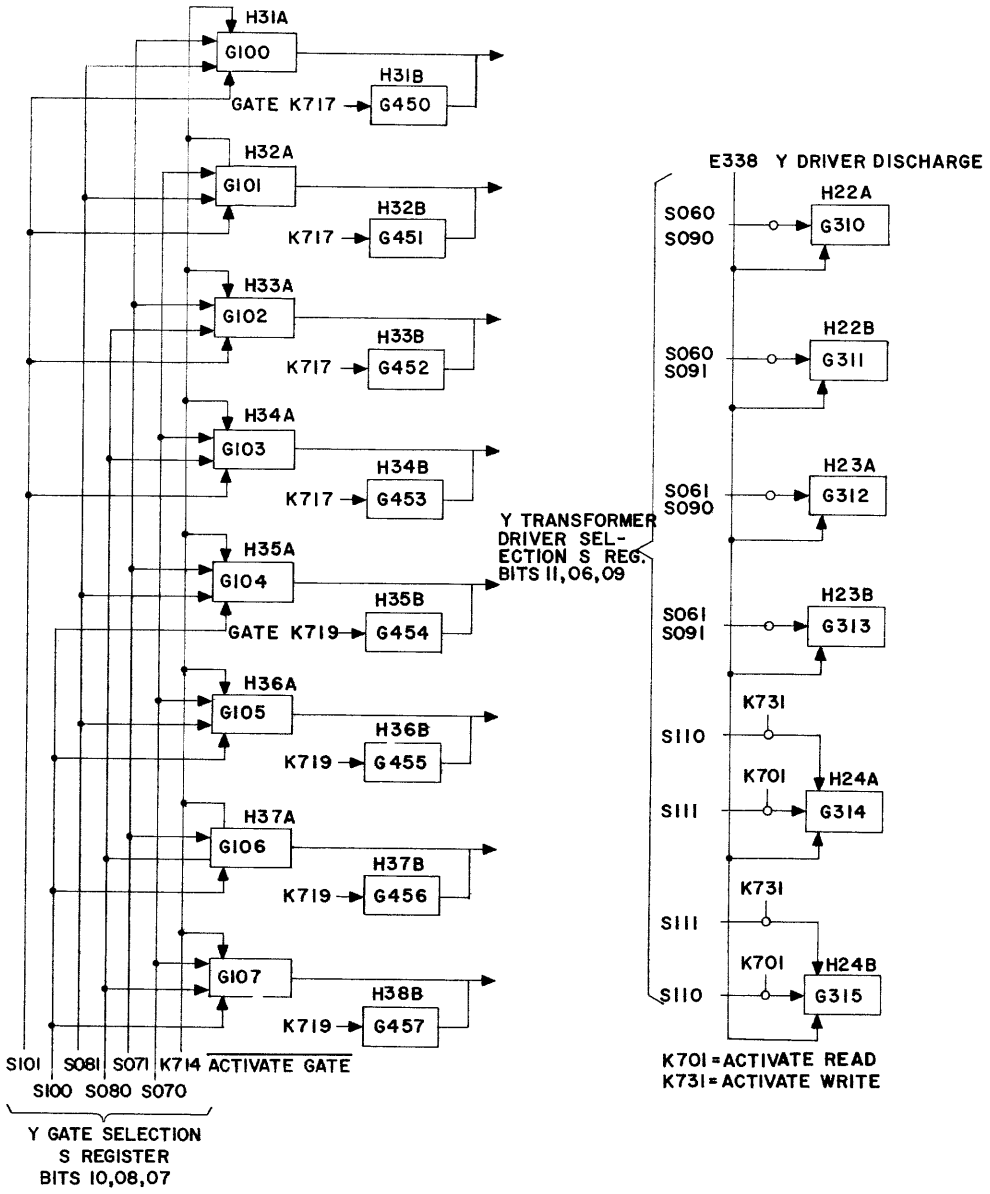


Figure 84. Y Gates

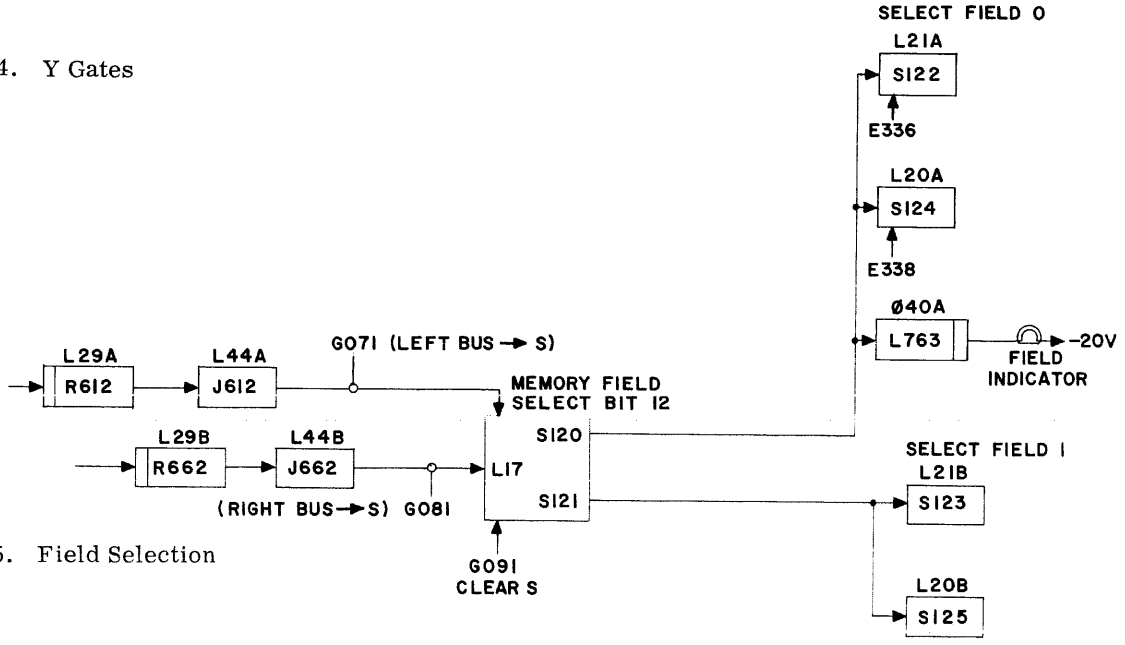


Figure 85. Field Selection

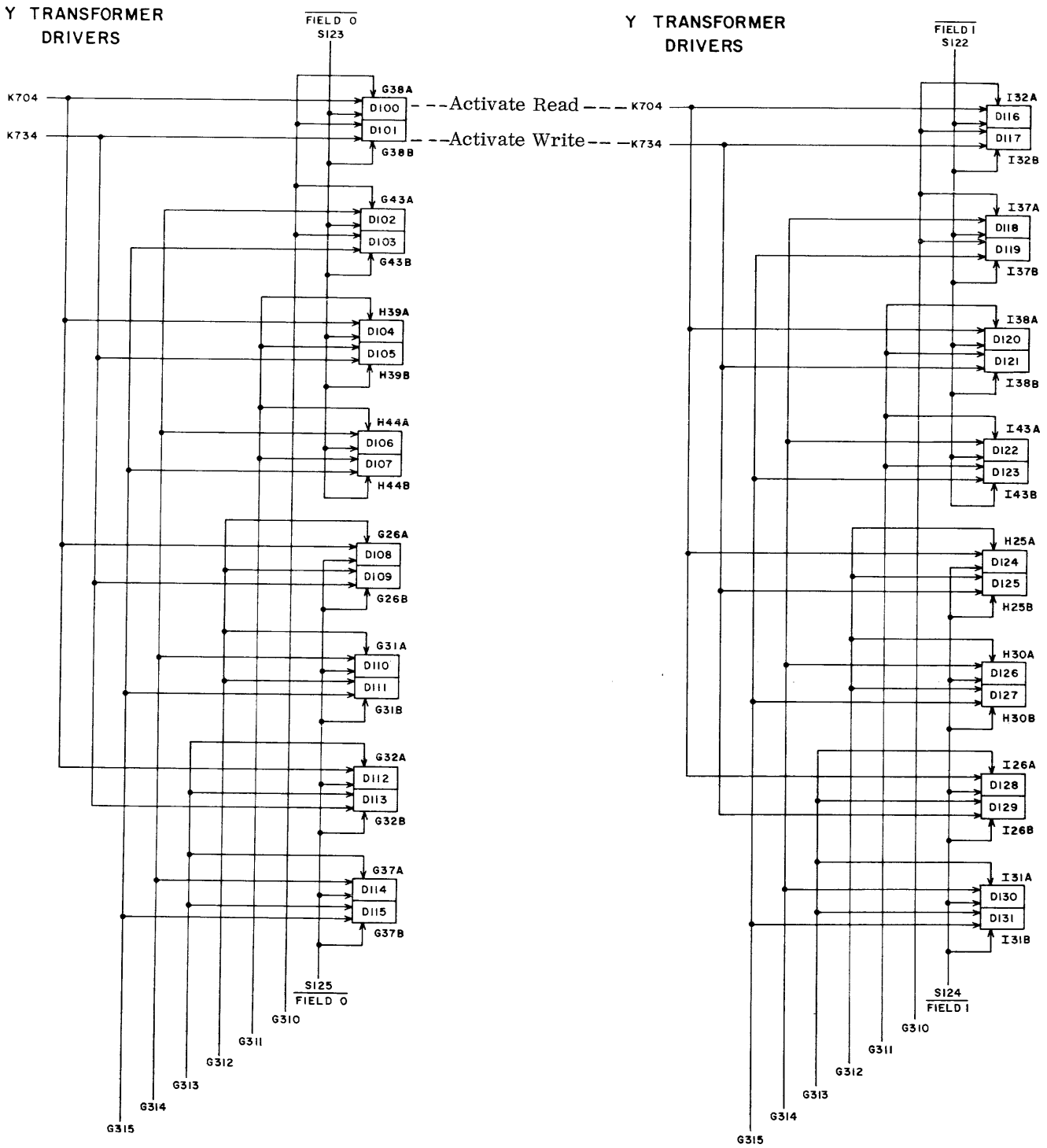
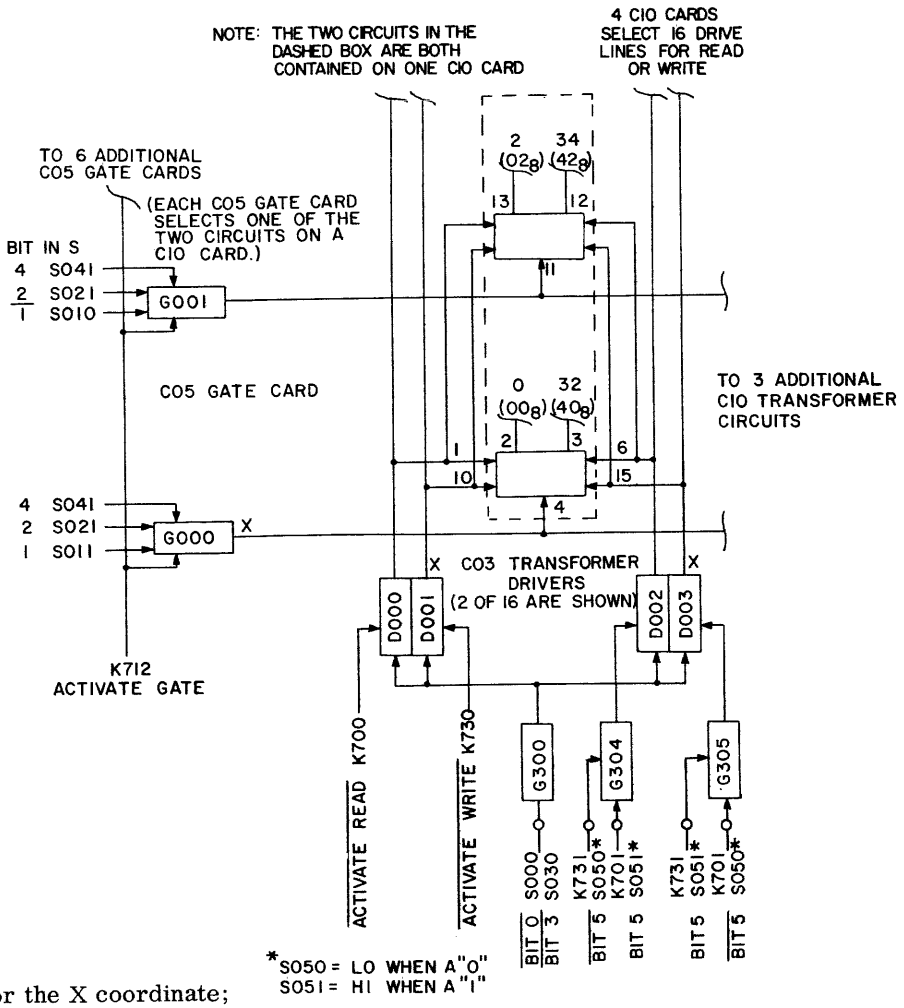


Figure 86. Y Transformer Drivers

Drive Line Selection

Drive line transformers, card type C10, can drive any one of the four X and Y drive lines. Two C03 line driver cards (two sections per card) and two C05 gate cards are needed to select the four drive lines which may be energized by one drive line transformer card. One gate card selects one of the two sections of the drive line transformer card, and two line driver sections select one of the two lines in that section.

In figure 86, if the lines marked with an X are selected, -44v will appear at pin 3 of the C10 card and drive line 32 (40_8) will be energized.



Selection is shown for the X coordinate; Y coordinate is similar.

Figure 87. Drive Line Selection

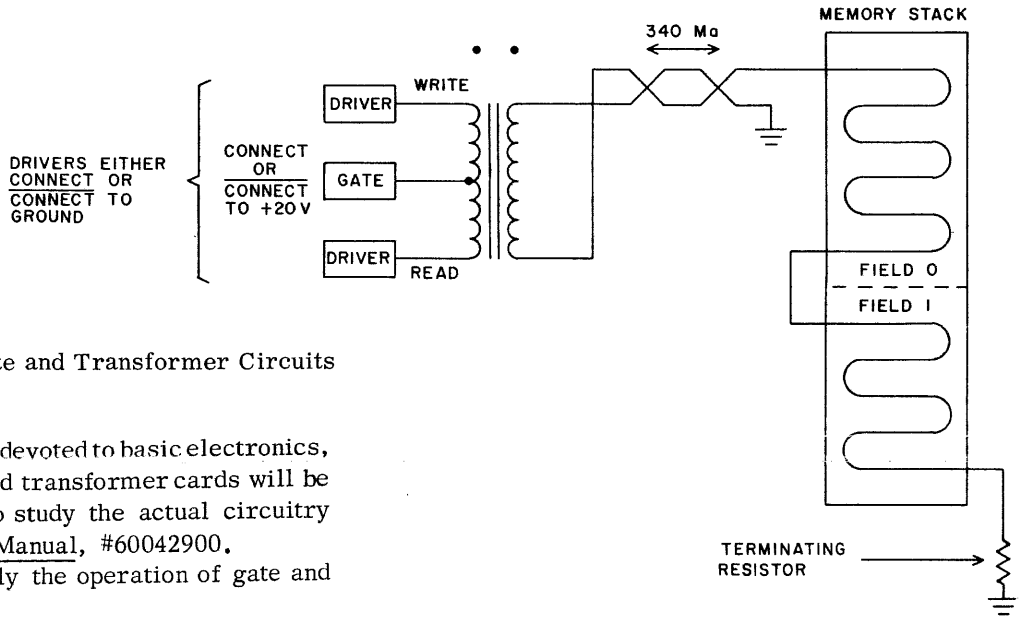


Figure 88. Simplified Gate and Transformer Circuits

Since this manual is not devoted to basic electronics, operation of gate cards and transformer cards will be covered very simply. To study the actual circuitry refer to Printed Circuits Manual, #60042900.

Figure 88 depicts simply the operation of gate and transformer cards.

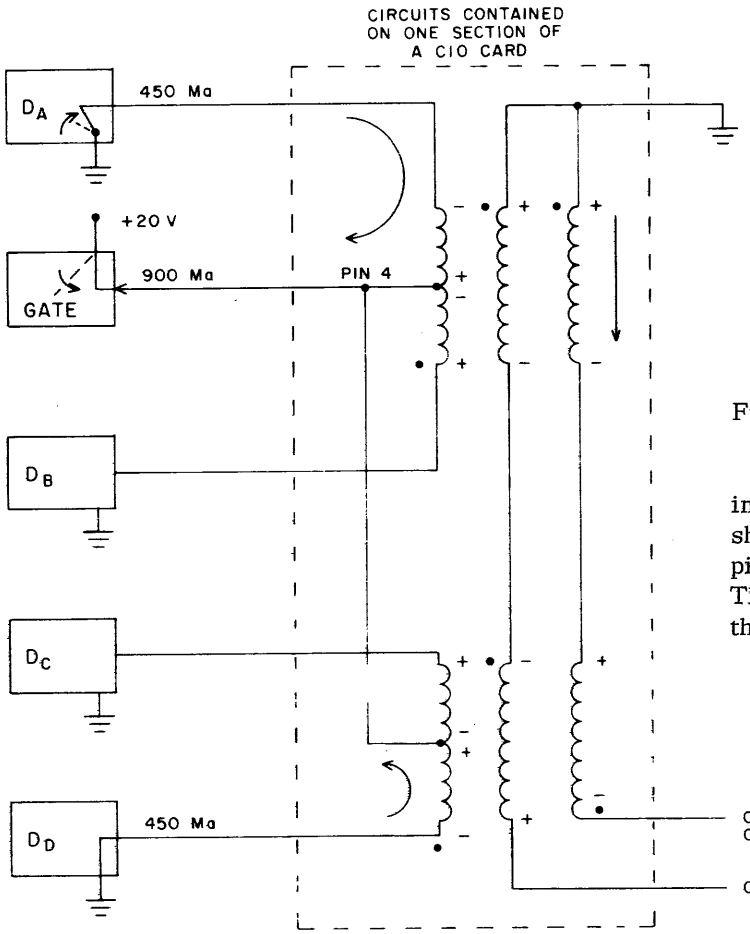


Figure 89. Drive Transformer Operation for Read

If DA and DD and the gate are on, current will flow in the primary and secondary of the transformers as shown. The winding in the secondary that is tied to pin 2 would actually be series winding, thus the -44v. The other winding is actually series opposing, thus there is little or no voltage or current.

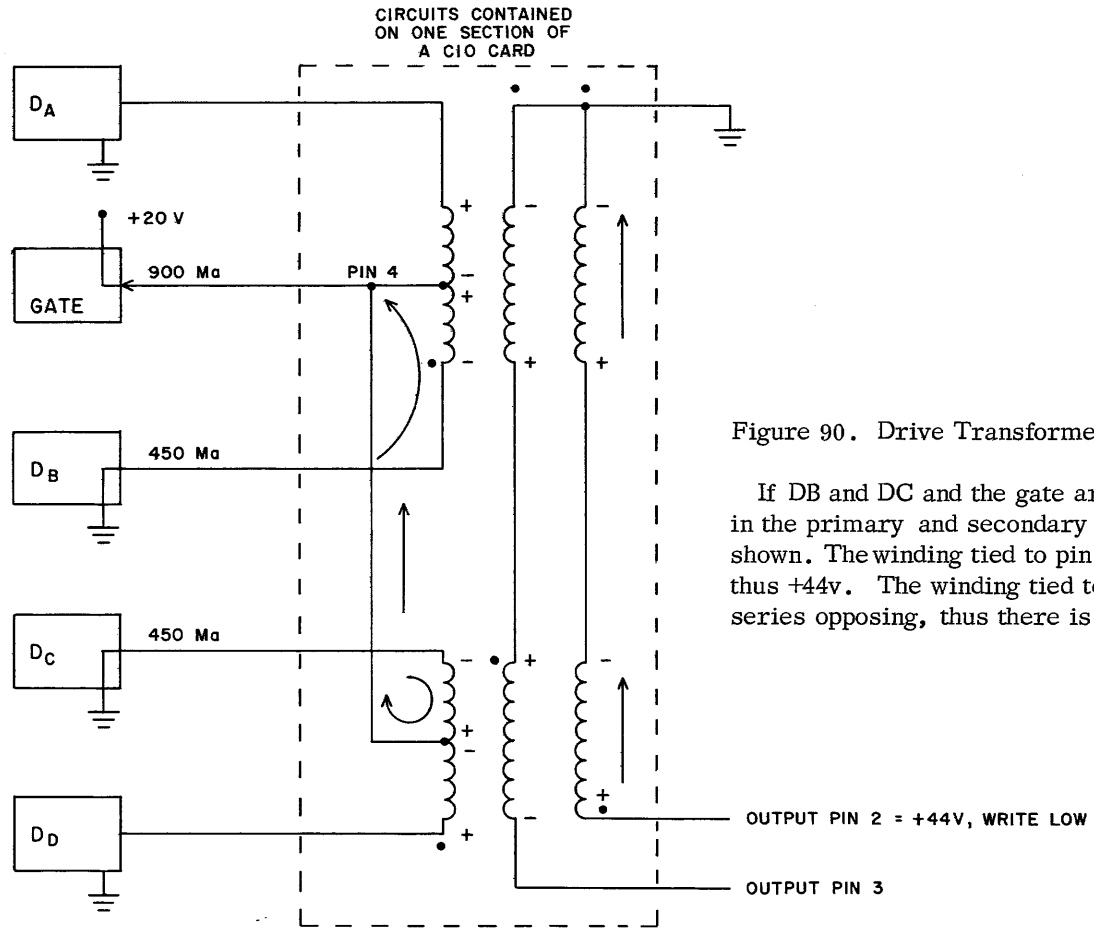


Figure 90. Drive Transformer Operation for Write

If DB and DC and the gate are on, current will flow in the primary and secondary of the transformer as shown. The winding tied to pin 2 is now series aiding, thus +44v. The winding tied to pin 3 is now acting as series opposing, thus there is little or no current.

Worksheet: STORAGE ADDRESS SELECTION

1. During an RNI cycle with (P) = 01571, indicate the terms that would be used within storage module 0 by filling in the following table:

	X Gate	Y Gate	X Xformer	Y Xformer	X Drivers	Y Drivers
Read phase						
Write phase						

2. During an RAD cycle with (F) = 20425673 and 16K storage, indicate the terms that would be used within the appropriate module by filling the following table: (Assume that the system contains two 3309 Storage modules.)

	X Gate	Y Gate	X Xformer	Y Xformer	X Drivers	Y Drivers
Read phase						
Write phase						

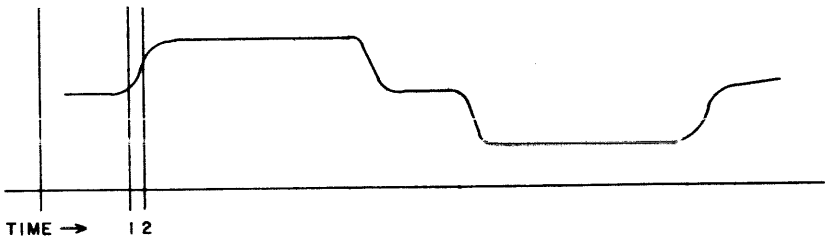
3. During an ROP cycle with (F) = 30057434 and 32K storage, indicate the terms that would be used within the appropriate module by filling in the following table: (Assume that the system contains four 3309 Storage modules.)

	X Gate	Y Gate	X Xformer	Y Xformer	X Drivers	Y Drivers
Read phase						
Write phase						

4. During an STO cycle with (F) = 40073746 and 32K storage, indicate the terms that would be used within the appropriate module by filling in the following table: (four 3309 modules)

	X Gate	Y Gate	X Xformer	Y Xformer	X Drivers	Y Drivers
Read phase						
Write phase						

5. For questions 2, 3, and 4, what changes would have to be made in the selected terms if the systems had contained only 8K storage? _____



Up to this point, we have:

1. Set busy FF.
2. Turn on read current.

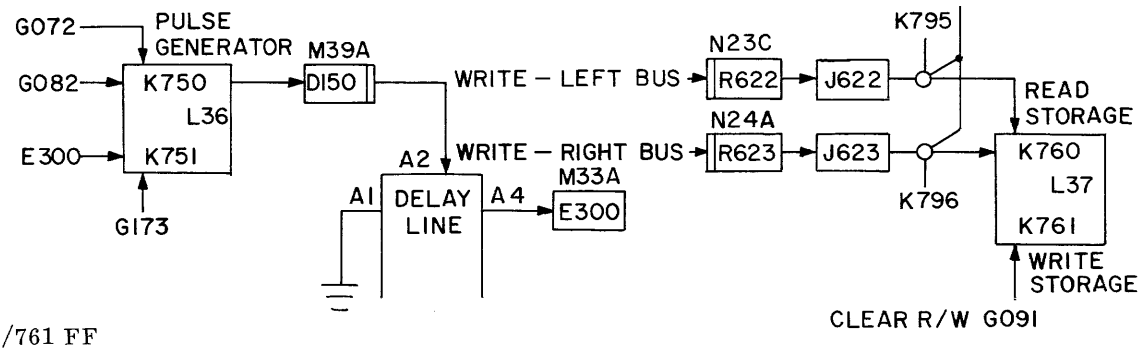


Figure 91. K760/761 FF

When E303 (A3) outputs a 1, the Activate Read, Gate and Busy FF's Set.

The first tap off the delay line feeds E (HA12 card), a delay line amplifier. Statically, the delay line is fed a constant 1 (-10v) and thus the constant outputs of the delay line amplifiers are 0's. Now, however, we will have a 1 out of the delay line amplifier because 0 (near ground) has been put into the delay line by the inverted set output of pulse generator FF.

As the pulse travels down the delay line the following signals are produced:

E300, tap A4, 50 nsec

1. E300 clears pulse generator FF, thus defining the trailing edge of the delay line pulse.

E309, tap A10, 200 usec. -

2. K760/761 (read/write storage) is set for any read or remains clear for any write. This flip-flop is set at this time if the memory cycle is a write into a protected area. (Main control senses this and changes write to read.) The protected area could be the auto load/auto dump area or an address block protected by the STORAGE PROTECTION ADDRESS switches. Attempting to set K760/761 at E allows sufficient time for address comparison (protected addresses compared to actual addresses referenced). Setting the flip-flop at this time is necessary because K760/761 is forced clear by a pulse occurring later along the delay line.

Read/Write Storage

Read/write storage FF (K760/761) is set by a read signal transmitted from either processor via the right or left bus. It is cleared at the end of a storage cycle. The set output of this flip-flop is used as an input to Z to data bus FF to determine whether or not the data in Z is to be gated to a data bus.

The clear (write) output is used to determine whether or not data is to be gated from either data bus to Z. When K760, E312, and K797 are all 1's, left data bus to Z is enabled. K760, E312, and K796 enable right data bus to Z. The clear output of read/write storage is ANDed with G174 and inverted to provide a clear input to the write designator bit FFs.

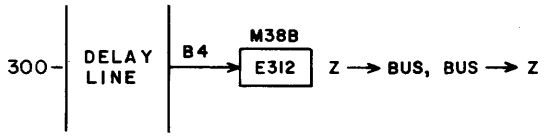
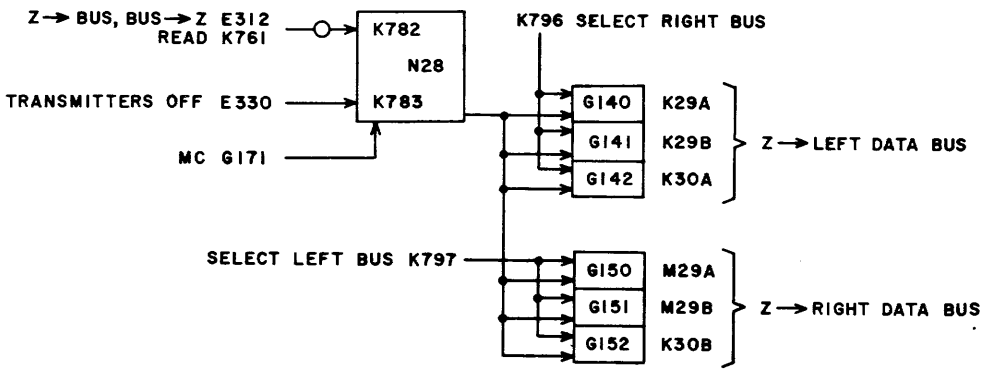


Figure 92. Delay Line Time B4

Tap B4, E312:

1. Enables (Z) to the transmitters that lead to the data bus register. (Although the information has not yet been read into Z register, it will be read into Z long before the data bus is sampled by the central processor.)
2. Clears the reply FF.

Figure 93. Data Bus



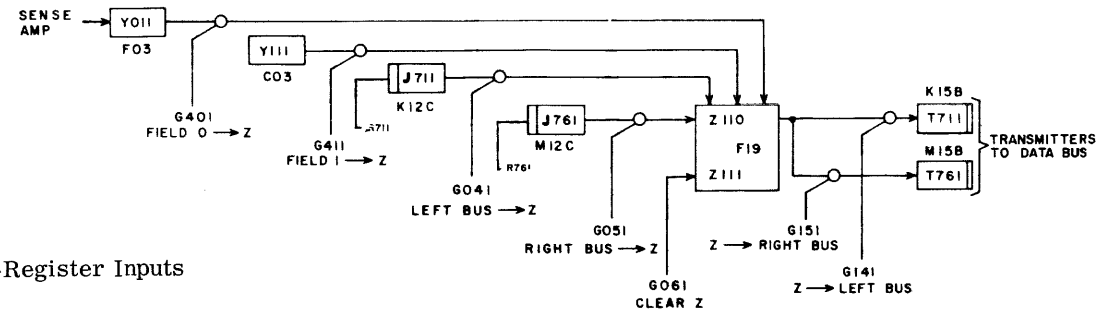


Figure 94. Z-Register Inputs

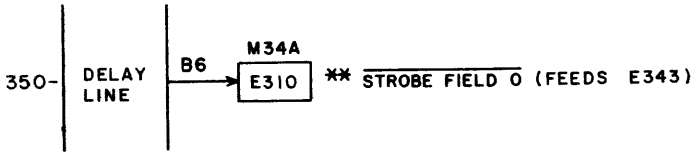


Figure 95. Delay Line Time B6

Tap B5, E310:

Partially enables G40- gates to gate the contents of field 1 sense amplifiers to Z (only if field 1 is being referenced). E311 is a C07 card (emitter follower) which does not invert. E343 is a C85 card which feeds R774 (a receiver). These two cards form a strobe-shaper network (figure 96). R774 outputs 0 for approximately 50 nsec during the sense → Z time.

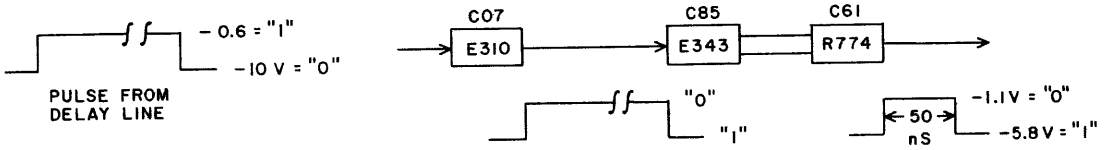


Figure 96. Strobe-Shaper Network

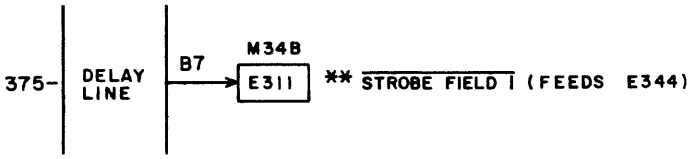


Figure 97. Delay Line Time B7

Tap B6, E311:

Partially enables G41- gates to gate the contents of field 1 sense amplifiers to Z (only if field 1 is being referenced). E311 is a C07 card (emitter follower) and does not invert. E344 and R775 (receiver for field 1) also form a strobe-shaper network. The theory of its operation is the same as that of E343 and R774.

Z REGISTER AND READ/WRITE CONTROLS

Z Register

The 28-bit Z register is the storage re-storing and modifying register. Data can be entered into Z from field 0 or field 1 sense amplifiers and the left or right data bus.

Read Control

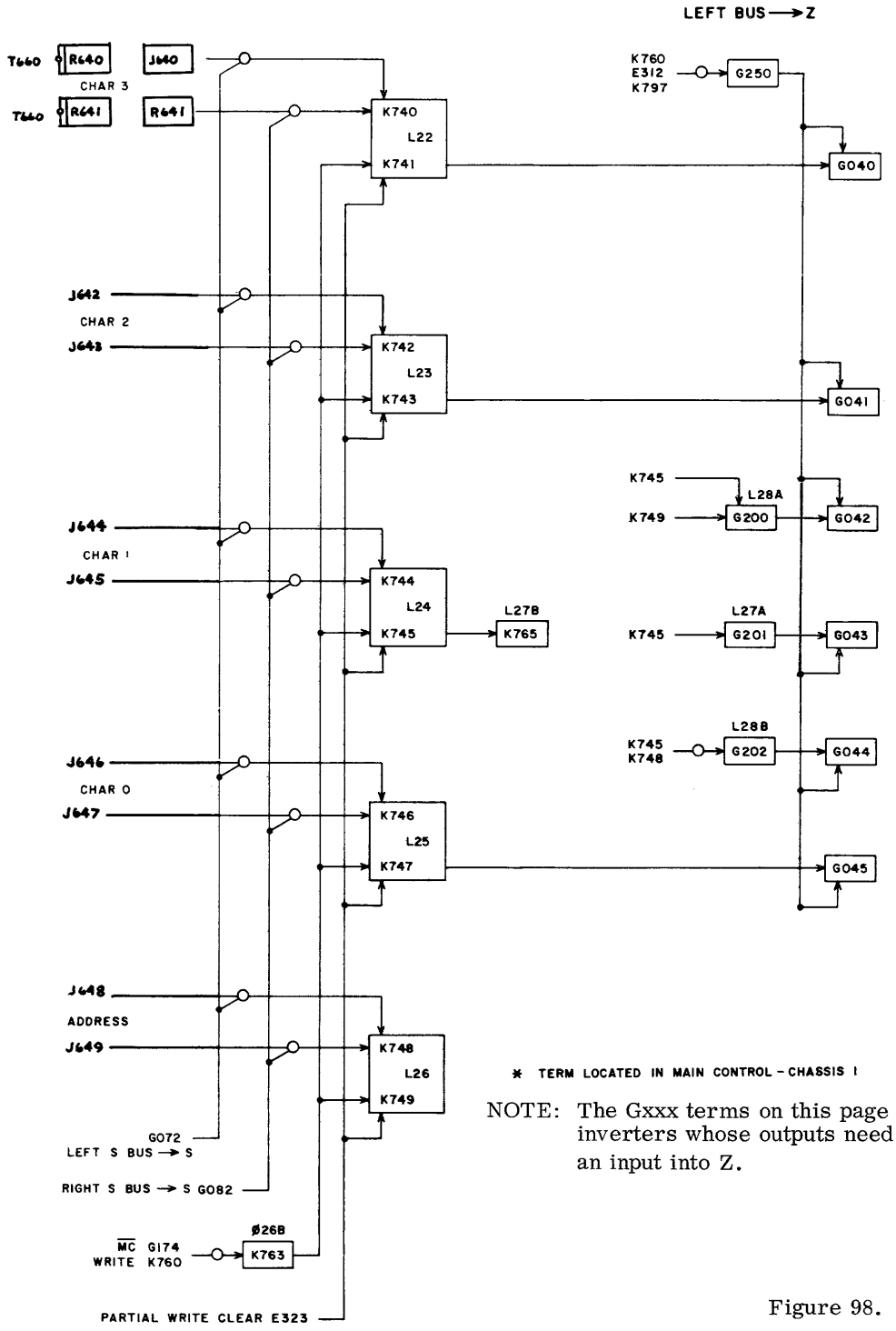
During a normal memory cycle all bits of the word referenced by (S) are read out of core storage in parallel and gated from the sense amplifiers into Z. Data from field 0 is gated to Z by the G40X inverters while data from field 1 is gated by the G41X inverters. The word in Z is placed on the selected data bus, then is written back into the storage location under reference.

Write Control

When a new word is to be written into storage, (Z) must be modified during the read phase of the write cycle. The modified word in Z is then written back into storage during the write phase. Five modes of storage modification (partial writes) are possible:

1. Single-character mode - any one of the four characters (six bits per character) of the word in Z may be replaced by a new character.
2. Double-character mode - the lower, middle, or upper two characters of a word in Z may be replaced by new data.
3. Triple-character mode - the upper three or lower three characters of the word in Z may be replaced by new data.
4. Full-word mode - a complete new word (four characters) may be placed in Z, then written into storage. The previous contents of this storage location are discarded.
5. Address mode - the lower 15 or 17 bits of the word in Z may be replaced by a new address.

Storage is modified (partial write) by blocking the input sense gates to Z for a character or group of characters during a read cycle, thus holding the corres-



NOTE: The Gxxx terms on this page are not gates, but inverters whose outputs need to be 1's to enable an input into Z.

Figure 98. Partial Write Bits

ponding bits of Z in clear. New data is then gated from the data bus into unfilled bits of Z and the whole (Z) is stored.

Write character signals determine storage modification by entering storage via R64x rank of receivers. Table 11 shows the signals necessary for various partial writes. These signals set write designator bit FFs (K74x/74x). The outputs of these flip-flops determine the gating of data to Z register by selectively enabling and blocking input gates to Z. The flip-flop outputs control G40x inverter rank which gates data from field 0 to Z, G41x rank which gates field 1 to Z, G04x rank which gates information from the left data bus to Z, and G05x rank which gates the right data bus to Z.

In a full-word re-store cycle all 24 bits are read and returned thus:

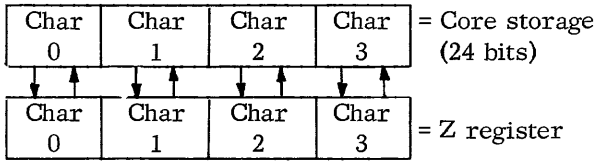
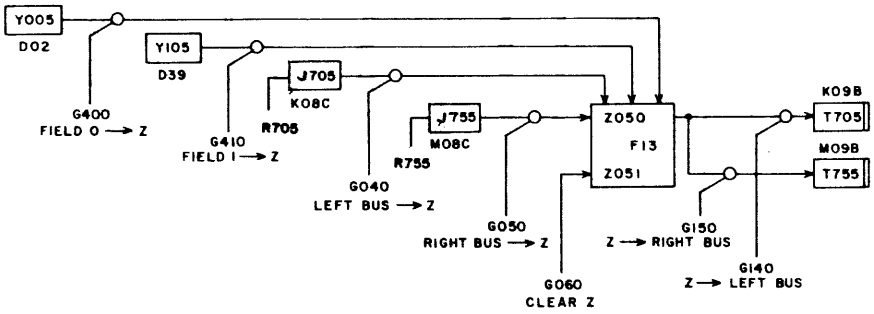


Table 11. PARTIAL WRITES

PARTIAL WRITE	BITS	WRITE CHAR 3 R640-R641	WRITE CHAR 2 R642-R643	WRITE CHAR 1 R644-R645	WRITE CHAR 0 R646-R647	WRITE ADDR R648-R649
Character 3	00-05	X				
Character 2	06-11		X			
Character 1	12-17			X		
Character 0	18-23				X	
Lower 12 bits	00-11	X	X			
Middle 12 bits	06-17		X	X		
Upper 12 bits	12-23			X	X	
Lower 18 bits	00-17	X	X	X		
Upper 18 bits	06-23		X	X	X	
All bits	00-23	X	X	X	X	
Word address	00-14	X	X			X
Character address	00-16	X	X	X		X



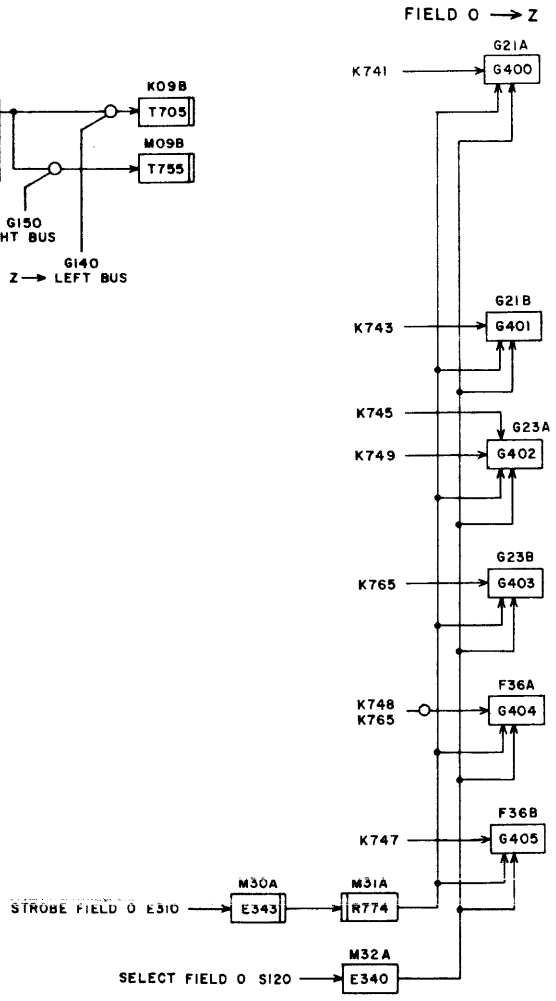
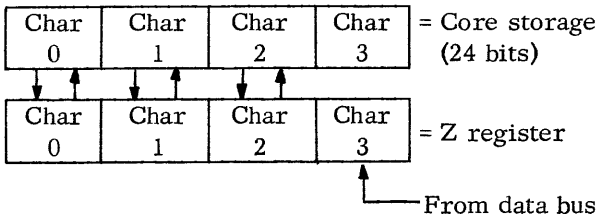


Figure 99. Possible Methods for Setting the Z Register

In a read cycle, with one character changed by instruction execution, 24 bits are read. However, the lower character (3) is not allowed to feed Z register. Instead, the character from the data bus is allowed to enter Z register and during the re-store cycle the newly created 24-bit word will be stored.



The write designator FFs are set by the central processor, depending on what instruction or cycle is being executed.

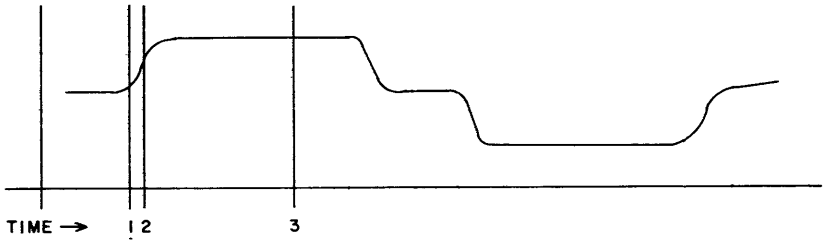
The following circuitry represents the write designator FF and enabling of either the sense amplifier or the data bus to Z register.

1. Fill in the following chart with the write designator bits required for the 12 indicated storage operations. Also indicate with a 1 which flip-flop is set:

WRITE OPERATION	2^4 K748/749	CHAR 0 2^3 K746/747	CHAR 1 2^2 K744/745	CHAR 2 2^1 K742/743	CHAR 3 2^0 K740/741
Character 0					
Character 1					
Character 2					
Character 3					
Upper 12					
Middle 12					
Lower 12					
Upper 18					
Lower 18					
Lower 15					
Lower 17					
Full 24					

Using the table above answer these questions regarding the write designator FFs:

2. Under what conditions and for what instruction would you expect storage to receive write designator bits of 01110_2? _____
3. For what operations would you expect storage to receive write designator bits of 00011_2? _____
4. Would a combination of write designator bits such as 01101_2 ever be employed within the 3300 ____
5. List all of the instructions and special sequences that would employ write designator bits of 10011_2.



We are now at stage 3:

1. Set busy FF.
2. Turn on read current.
3. Gate information to Z register.

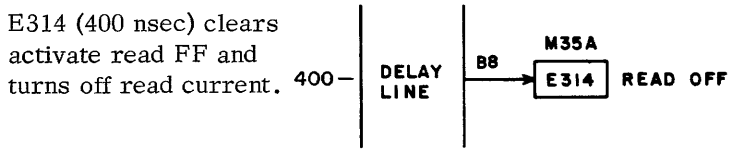
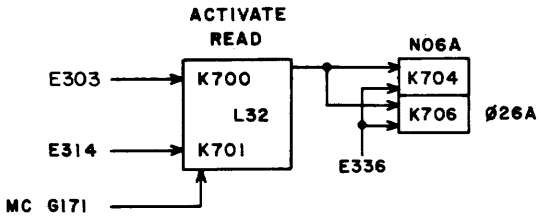


Figure 100. Delay Line Times



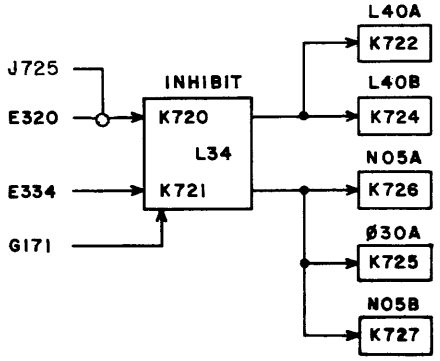


Figure 101. Delay Line Times

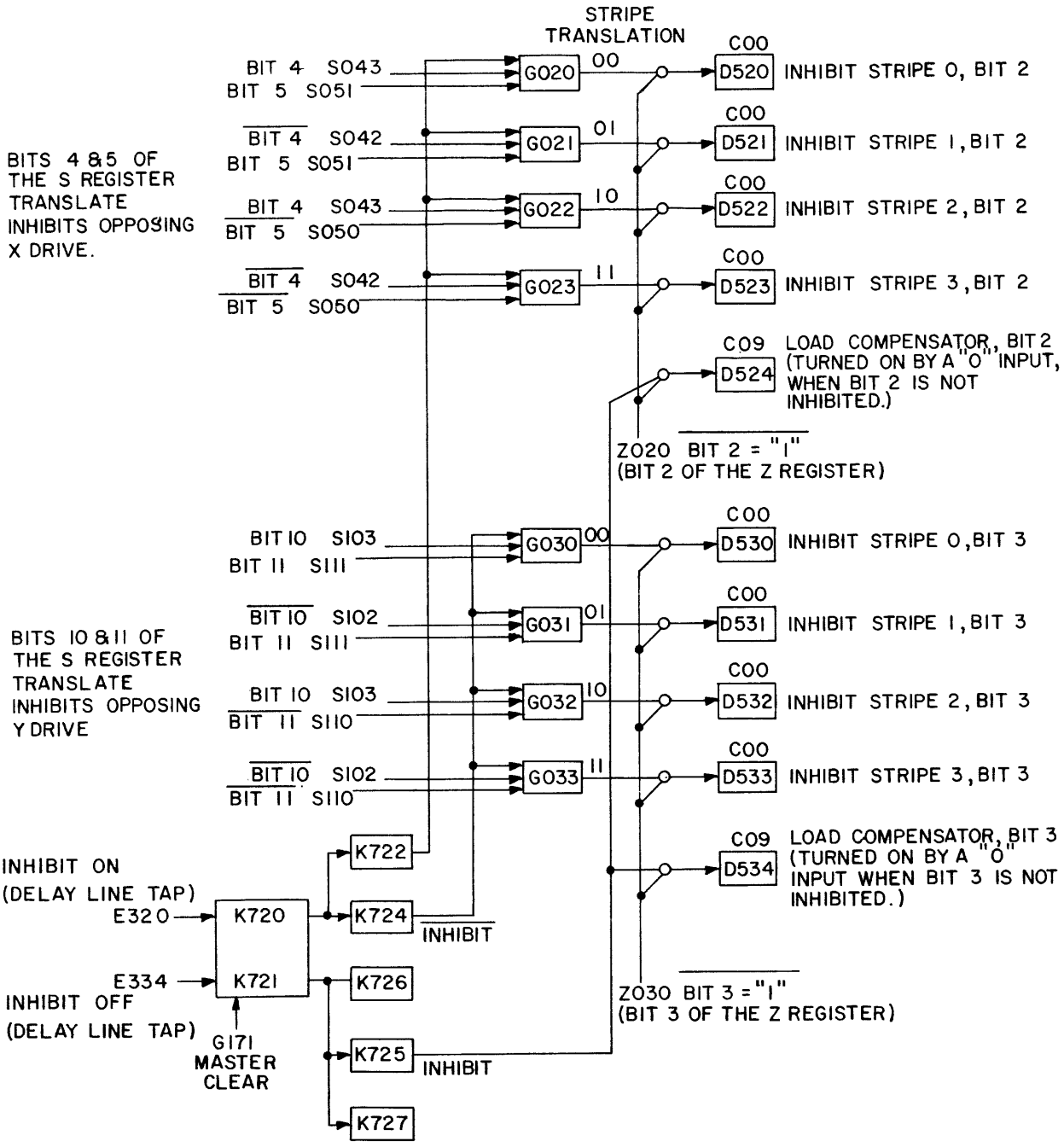


Figure 102. Inhibit Scheme for Bits 2 and 3 in Detail

An inhibit line is energized by inhibit driver, card type C00. (Inhibit driver C00 turns on with a 1 input.) The AND gates to the four inhibit drivers in a bit plane are partially enabled when 0 is to be written into that bit position. The clear output of Z register FF for that bit partially enables these gates. The inhibit stripe to be energized is determined by either bits 4 and 5 (for X inhibit) or bits 10 and 11 (for Y inhibit) of the address in S. The set and clear outputs of these flip-flops either enable or block the G02X and G03X inverters. The outputs of these inverters are ANDed with the clear output of Z register FF to enable the inhibit drivers. An inhibit is blocked by K722 and K724 when inhibit FF (K720/721) is clear (figure 102).

The inhibit lines are treated as transmission lines in a manner similar to the X and Y drive lines. The characteristic impedance of an inhibit line is approximately 140Ω. A 10w terminating resistor of 120Ω is placed in series with each inhibit line. Although this value produces a slight impedance mismatch the resulting current flow from a +40v supply is approximately the required 340 ma.

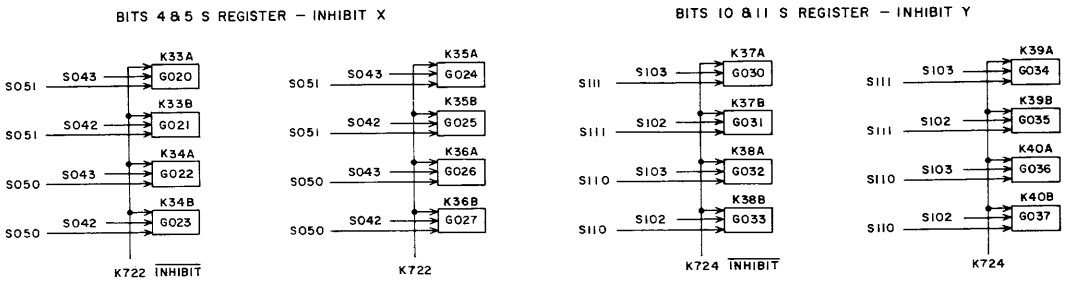


Figure 103.
Inhibit Selection Logic

It should be obvious from the description of the inhibits that the circuits will have either an inhibit or an inhibit compensator turned on. Both cannot be on at the same time. The inhibit compensator is a C09 card and turns on with 0 input.

Remember that parity is checked during the write cycle, so parity inhibits are also involved then.

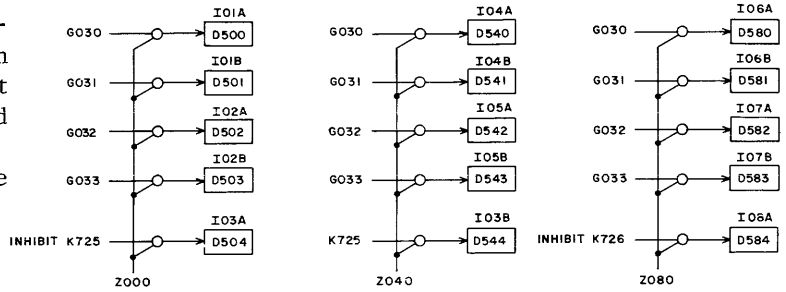


Figure 104. Partial Listing of Inhibit Line Selection

The combination of P304 and P306 translates character 3 as having an odd number of bits and indicates that inhibit should be turned on. P309 is the same as P304 and P306.

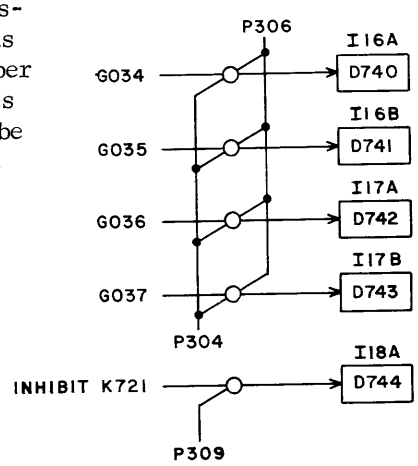
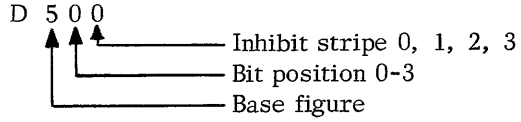
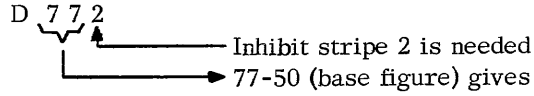


Figure 105.
Parity Bit
Inhibit Selection

The numbers assigned to inverters have special meaning. G020, for example, is selected for any X addresses from 00-17, G021 for any addresses from 20-37, etc. The same holds true for the Y inhibits, only the base number is 30. Thus G030 is for any Y addresses from 00-17, etc.

The G02X and G03X terms select a stripe and the D500 terms feed the specific bit. There is a numbering system here also. For example:





D772 would be turned on if character 0 required 0 stored in the parity position.

Work the following problems:

1. Program control has initiated a read operation at address 17603. The contents of this address is 47623C12. During the write phase of read, which inhibit generators and inhibit compensators will pass current to provide proper re-storing at this storage location?

Bit 0 = _____	Bit 10 = _____	Bit 20 = _____
Bit 1 = _____	Bit 11 = _____	Bit 21 = _____
Bit 2 = _____	Bit 12 = _____	Bit 22 = _____
Bit 3 = _____	Bit 13 = _____	Bit 23 = _____
Bit 4 = _____	Bit 14 = _____	
Bit 5 = _____	Bit 15 = _____	
Bit 6 = _____	Bit 16 = _____	Bit 24 = _____
Bit 7 = _____	Bit 17 = _____	Bit 25 = _____
Bit 8 = _____	Bit 18 = _____	Bit 26 = _____
Bit 9 = _____	Bit 19 = _____	Bit 27 = _____

2. List inhibit generators and compensators used during the write phase of STO cycle with the following initial conditions:

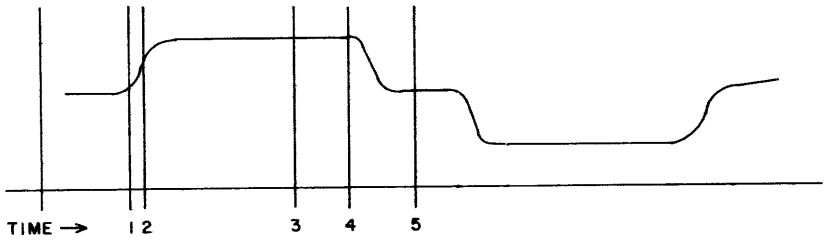
(F)=42062552 (A)=23476534 (ML 14532)=76431535

Bit 0 = _____	Bit 10 = _____	Bit 20 = _____
Bit 1 = _____	Bit 11 = _____	Bit 21 = _____
Bit 2 = _____	Bit 12 = _____	Bit 22 = _____
Bit 3 = _____	Bit 13 = _____	Bit 23 = _____
Bit 4 = _____	Bit 14 = _____	
Bit 5 = _____	Bit 15 = _____	
Bit 6 = _____	Bit 16 = _____	Bit 24 = _____
Bit 7 = _____	Bit 17 = _____	Bit 25 = _____
Bit 8 = _____	Bit 18 = _____	Bit 26 = _____
Bit 9 = _____	Bit 19 = _____	Bit 27 = _____

3. In an 8K stack, how many cores are threaded by the closed loop output of an inhibit generator such as D500? _____
4. What advantage, if any, is obtained by driving the inhibit generators for bits 24, 25, 26, and 27 with the output of the parity generator instead of the corresponding flip-flops in Z register? _____

_____

_____



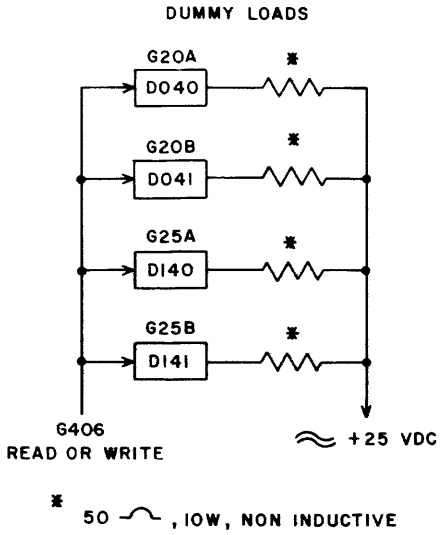


Figure 106. Dummy Load Enables

Another quick review--you supply the titles

1. _____
2. _____
3. _____
4. _____
5. _____

The next operation which should occur is:

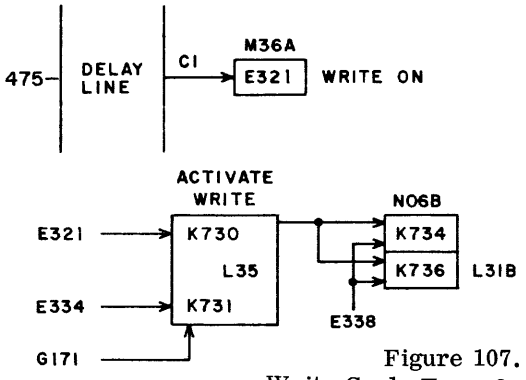


Figure 107. Write Cycle Turn-On

E321 sets activate write FF to begin the write phase of the cycle by enabling the selected line drivers. (This is the reverse of the transformer driver.)

E322 (figure 116) drops to a 0, so G192 may come up to partially enable AND gates to the set side of parity fault FF. Parity fault is then set if a parity error has been detected. At E322 time, G192 will output a logical 1 if the processor originating the request does not have DISABLE PARITY pressed. E322 is a C07 card (emitter follower).

Work the following problems:

For the parity checker/generator logic circuit associated with the three pairs of bits making up character 3, match each of the following terms with its output translation. CLUE: Only two combinations of pair translations will result in the generation of a 1 parity bit, only one pair consists of like bits, or all three pairs consist of like bits.

1. P300 ___
2. P301 ___
3. P302 ___
4. P303 ___
5. P304 ___
6. P305 ___
7. P306 ___
8. P307 ___
9. P308 ___
10. P309 ___

A. All pairs consist of like bits.
B. One pair consists of like bits + two pairs consist of like bits + three pairs consist of like bits.
C. Generate a parity bit of 1.
D. Lowest pair consists of unlike bits.
E. One pair consists of unlike bits + two pairs consist of unlike bits + three pairs consist of unlike bits.
F. Parity error when reading character 3.
G. Upper pair consists of unlike bits.
H. Generate a parity bit of 0.
I. Middle pair consists of unlike bits.
J. Two pairs consist of like bits + three pairs consist of like bits + three pairs consist of unlike bits.

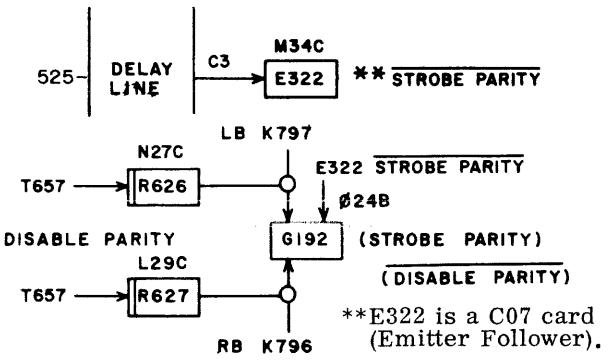


Figure 108. Parity Strobe

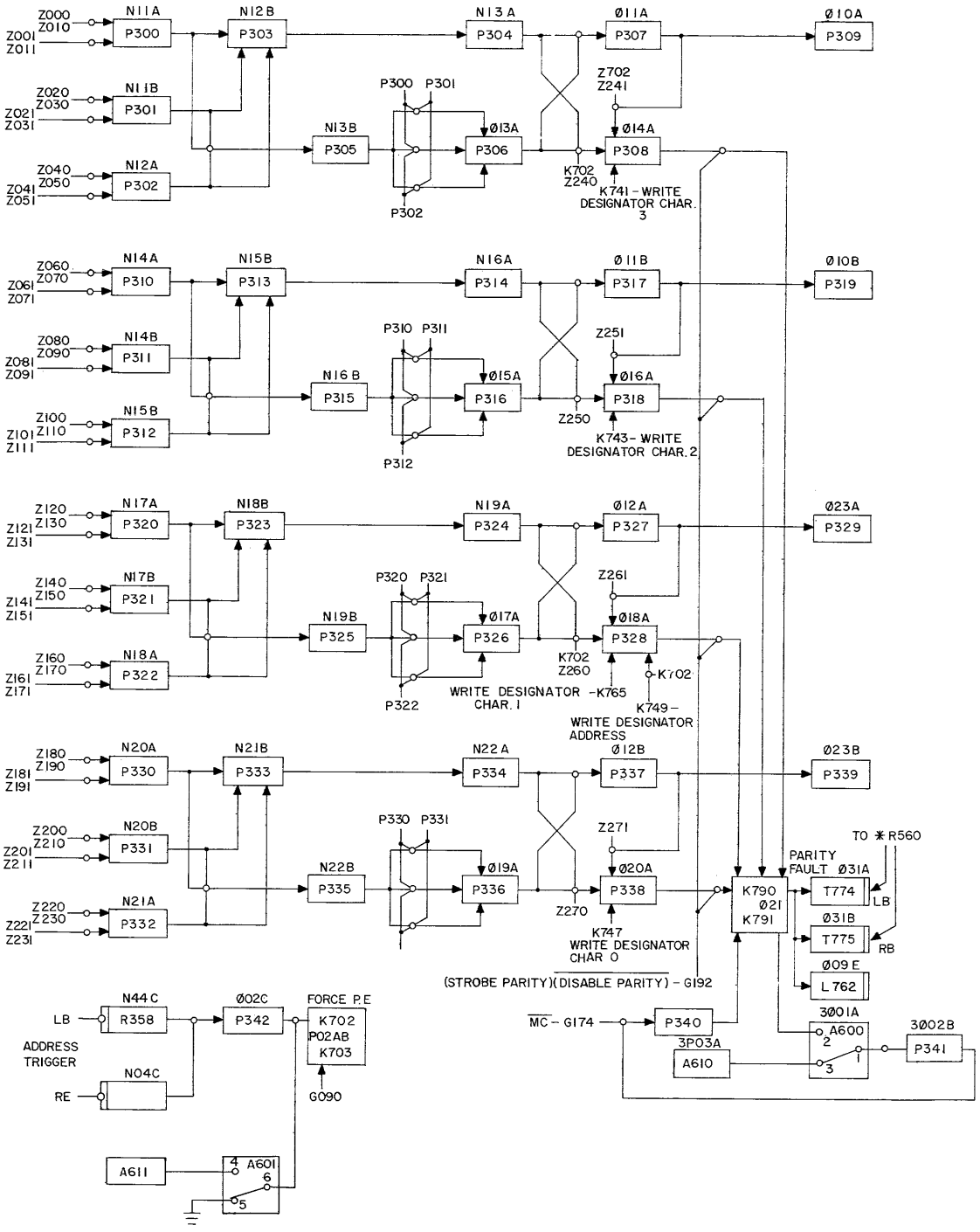
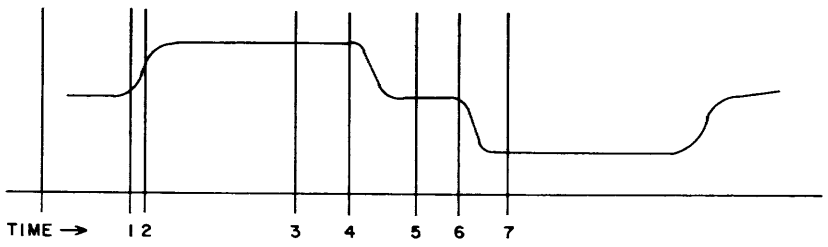


Figure 109. Circuitry Showing Method Used to Determine if a Parity Bit must be Stored

---

Another quick review:

1. Set busy FF.
2. Turn on read current.
3. Gate information to Z register.
4. Shut off read current.
5. Turn on inhibit current.
6. Turn on write current.
7. Strobe for parity errors.



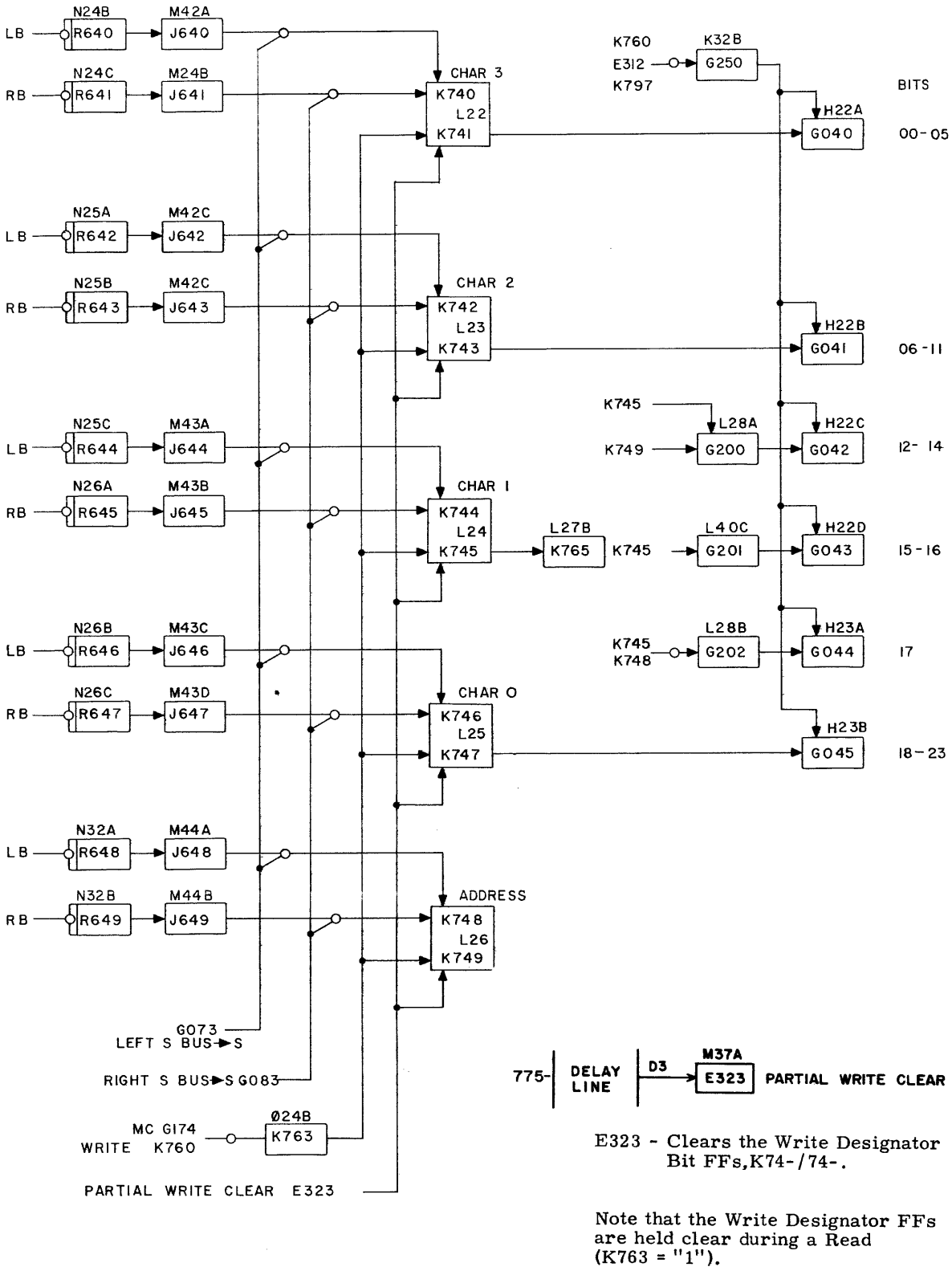
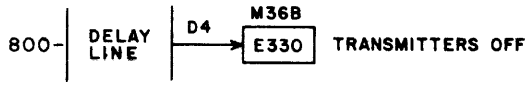


Figure 110. Write Designator Bit Clear Enable

Figure 111. Data Bus Transmitters Off



E330 (transmitters off) clears K782/783.

See figure, See figure 112 for remainder of drawing.

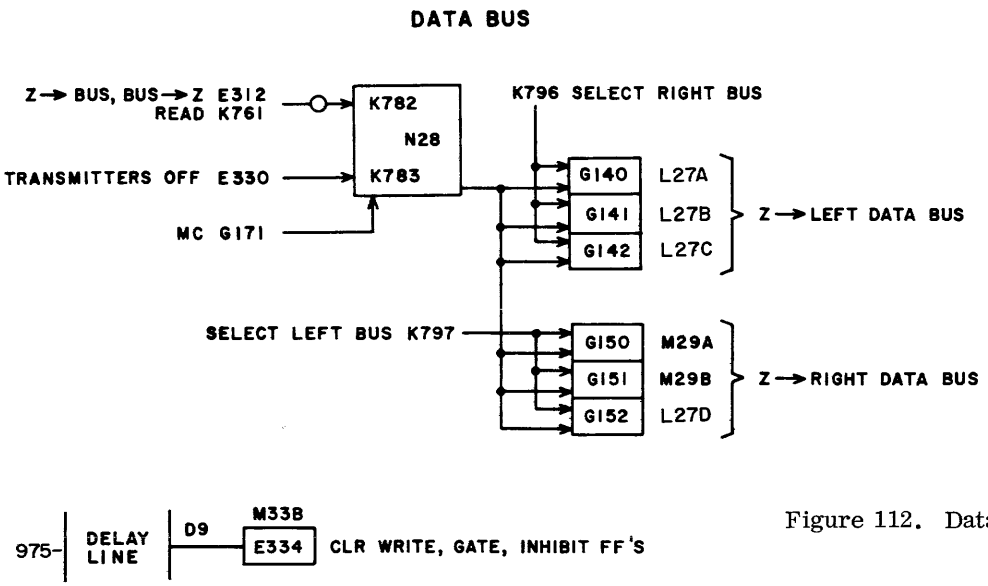


Figure 112. Data Bus Transmitter Enables

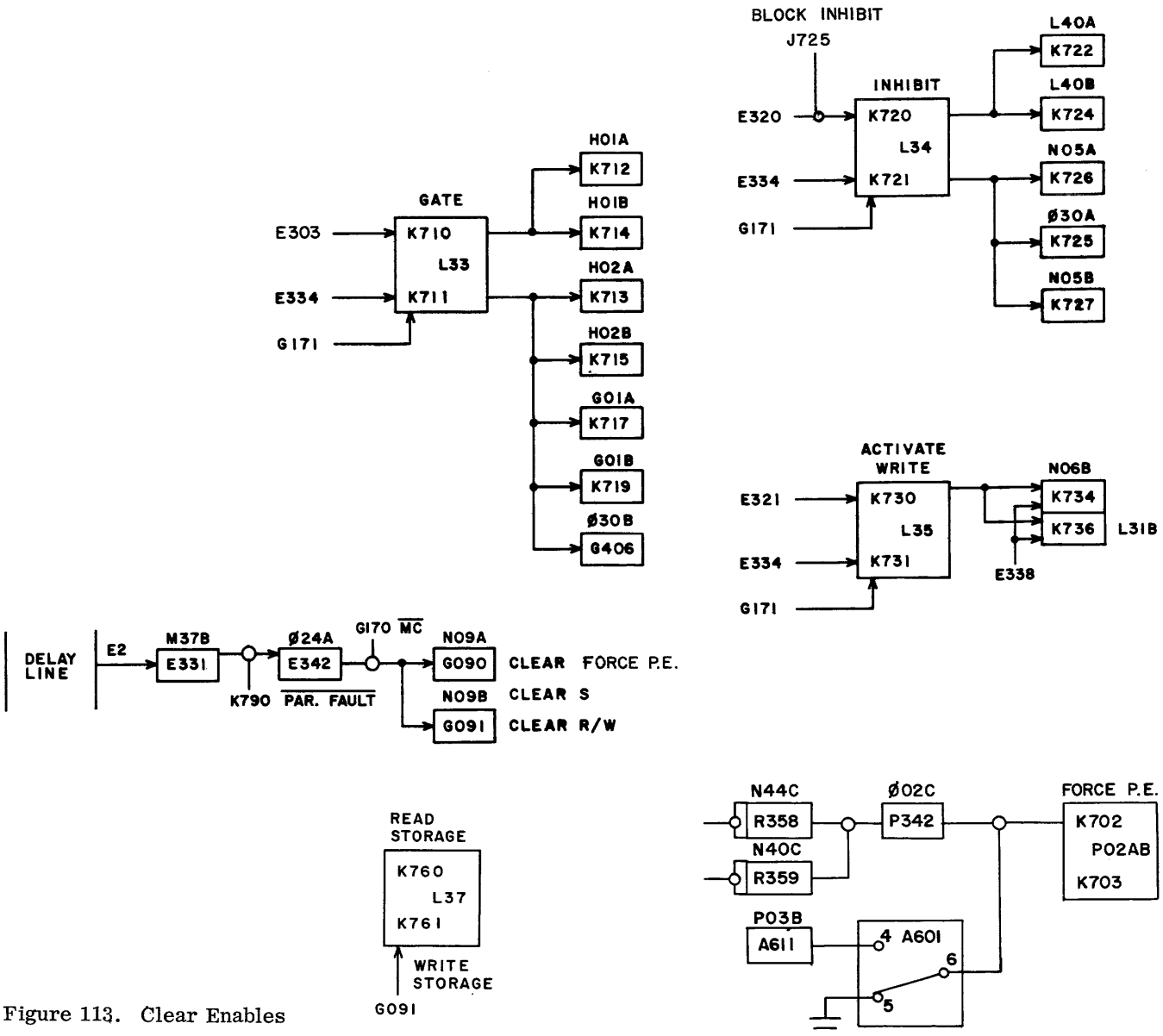


Figure 113. Clear Enables

E331 brings up signals which clear S register, parity fault, and read/write storage if no parity error exists or if a parity error is to be ignored. Read/write storage is always cleared at this point on the assumption that the next cycle will be write.

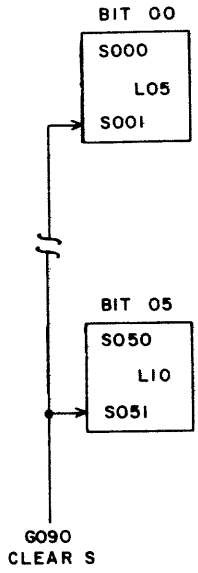


Figure 114. Clear "S" Enables

The purpose of the driver discharge is to ground both sides of all transformer primaries.

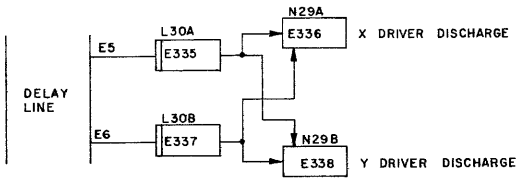


Figure 115. Driver Discharge Enable

By turning on all drivers with the gate off, all points on the transformer will be grounded.

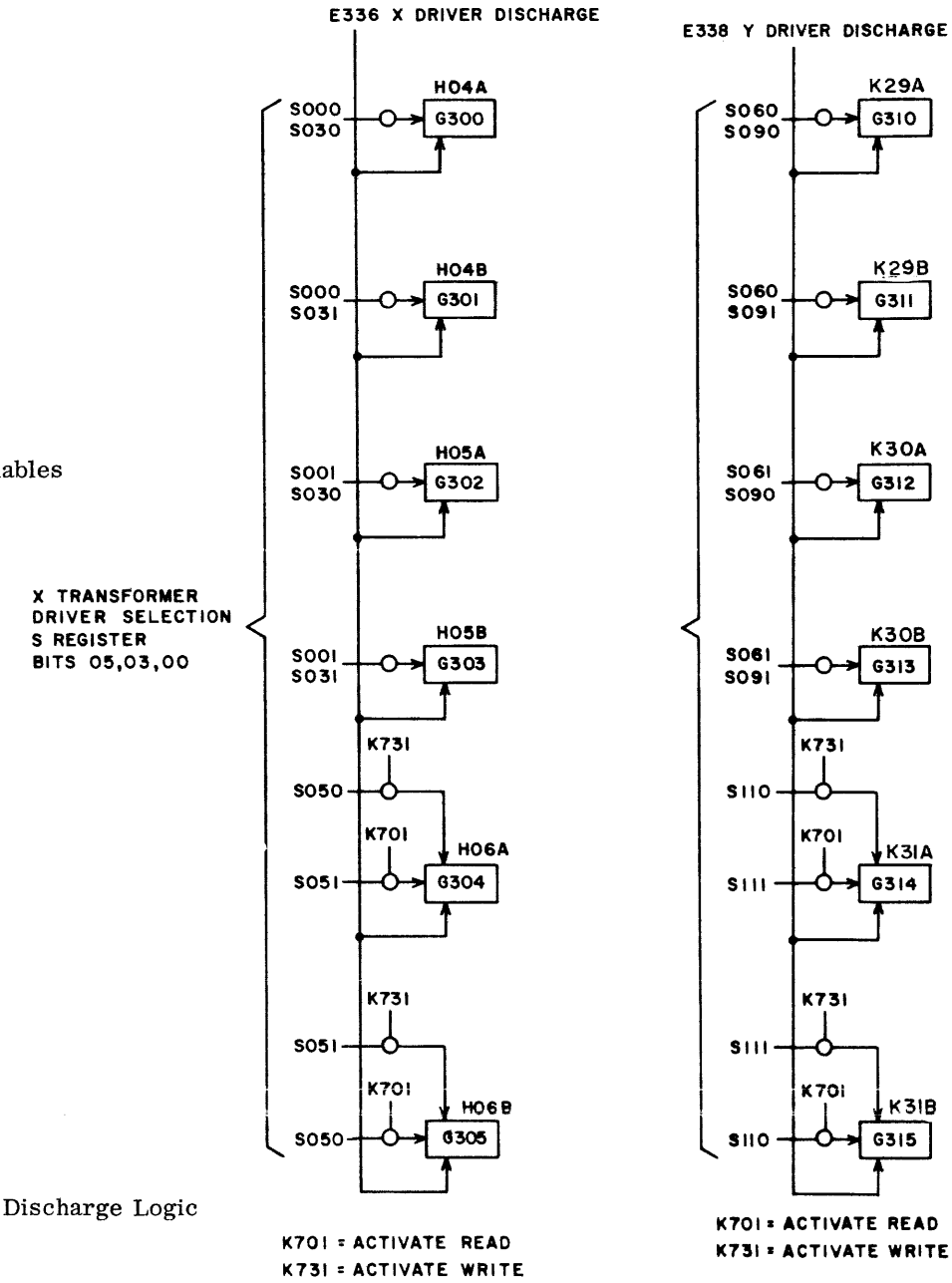


Figure 116. Driver Discharge Logic

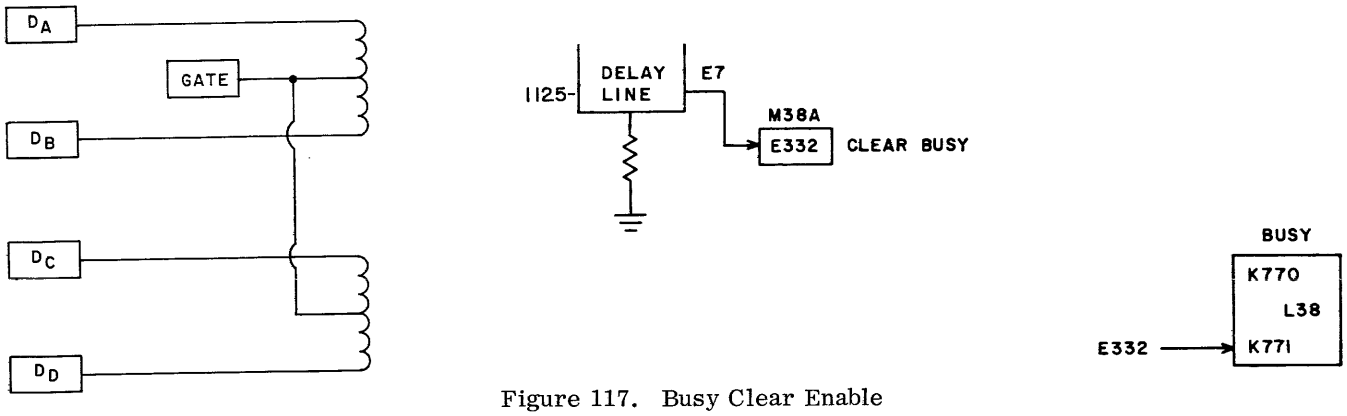
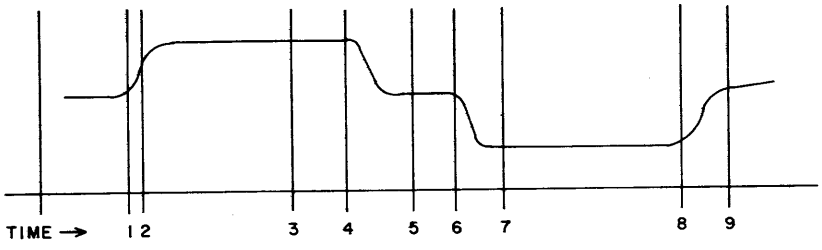


Figure 117. Busy Clear Enable

The previous discussion covered one complete memory cycle with all times and circuits analyzed. The worksheet on the next page reviews the cycle.

The previous material covered all the circuitry involved with each time tapped off the delay line. The following problems will serve not only as a review but should help cement in your mind the operations of the storage modules.



1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____

READ/WRITE TIMING Worksheet

Number each of the following operations in order of occurrence during the storage reference cycle indicated. (Operations that occur simultaneously should have the same number.)

READ CYCLE

_____ Test for parity error.

_____ Sample appropriate bus for write designators.

_____ Clear read FF.

_____ Set busy FF.

_____ Set inhibit FF.

_____ Clear Z register.

_____ Clear gate FF.

_____ Transmit (Z) to appropriate data bus.

_____ Sample appropriate S bus for an address.

_____ Clear busy FF.

_____ Set gate FF.

_____ Clear inhibit FF.

_____ Clear S register.

_____ Drop transmission of information in Z.

_____ Clear write FF.

_____ Transmit reply.

_____ Drop transmission of reply.

_____ Set read FF.

_____ Gate sense amps into Z register.

_____ Set write FF.

_____ Clear K760/761.

_____ Set pulse generator FF.

_____ Clear pulse generator FF.

_____ Apply clear pulse to write designator FFs.

WRITE CYCLE

_____ Clear write designator FFs.

_____ Clear pulse generator FF.

_____ Enable input to K760/761.

_____ Set pulse generator FF.

_____ Set write FF.

_____ Gate information from appropriate data bus into Z register.

_____ Drop transmission of reply.

_____ Set read FF.

_____ Transmit reply.

_____ Clear write FF.

_____ Gate sense amps to Z if partial write.

_____ Clear S register.

_____ Set gate FF.

_____ Clear busy FF.

_____ Sample appropriate S bus for an address.

_____ Clear gate FF.

_____ Set inhibit FF.

_____ Clear Z register.

_____ Set busy FF.

_____ Clear read FF.

_____ Sample appropriate bus for write designators.

_____ Test for parity error if partial write.

READ STORAGE TIMING

G176 Request from left bus and module select code compare locks scanner CC (6-3)

After .1 us G070→G072 = 1 if busy (6-3)
Set K796/797, left bus selected (6-3)
Set K780/781, reply and transmit reply (6-3)
Enable left S bus→S register (6-7)
Set K750/751, pulse generator (6-3)
Clear Z register (6-9)

A3 E303 Set K710/711 gate FF (6-5); Set K700/701 activate read FF (6-5) } Enable X and Y gates and drivers
Set K770/771 busy FF (6-5) (6-17 & 6-21)
K773=1 and transmits busy signal and disables G07X & G08X inverters (6-3)

A4 E300 Clear K750/751, pulse generator (6-5)
enable setting of K760/761 on read (6-3);
static clear on write designator bit FFs (6-9)

B4 E312 Clear K780/781, drop reply (6-3);
set K782/783, Z→data bus (6-9).

B6 E310 Enable field 0→Z if address bit 12 = 0 (6-9)

B7 E311 Enable field 1→Z if address bit 12 = 1 (6-9)

B8 E314 Clear K700/701, activate read (6-5);
turn off X and Y drivers (6-17) and (6-21)

B10 E320 Set K720/721, inhibit FF (6-5);
partially enable inhibit drivers (6-27)
if block inhibit signal down.

C1 E321 Set K730/731, activate write (6-2);
enable X and Y drivers (6-17) and (6-21).

C3 E322 Partially enable setting of K790/791, parity error FF (6-5 and 6-15), if disable parity

D3 E323 Clear write designator bit FFs (6-9) (redundant on a read STO)

D4 E330 Clear K782/783, Z→data bus (6-9)

D9 E334 Clear K710/711, gate FF; K720/721, inhibit FF; and K730/731, activate write FF (6-5).
End of write cycle

E2 E331 If no partial error or disable partial error, clear "S" reg (6-7)
If no partial error or disable partial error, clear K760/761 R/W FF (6-3)

E6 E336, E338 Turn on all drivers (6-17 and 6-21) to discharge transformers

E7 E332 Clear K770/771, busy FF (6-5)

*Page number in Logic Diagrams

WRITE STORAGE TIMING

G176 Request from left bus and module select code compare locks scanner CC (6-3).

After .1 us G070→G072 = 1 if busy (6-3)
Set K796/797, left bus selected (6-3)
Set K780/781, reply and transmit reply (6-3)
Enable left S bus→S register (6-7)
Enable write designator bits to K74X FFs (6-9)
Set K750/751, pulse generator (6-5)

A3 E303 Set K710/711 gate FF (6-5); Set K700/701 activate read FF (6-5) } Enable X and Y gates and drivers
Set K770/771 busy FF (6-5) (6-17 & 6-21)
K773=1 and transmits a busy signal (6-5) and disables G07X and G08X inverters (6-3)

A4 E300 Clear K750/751, pulse generator (6-5);
A10 E339 enable clearing of K760/761 on write storage (6-3)

B4 E312 Clear K780/781, drop reply (6-9);
enable left bus→Z (only those characters to be written new as determined by write designator bit FFs (6-9).

B5 E310 Enable field 0→Z if address bit 12 = 0 (6-9) (used only on a partial write for those characters not being replaced)

B6 E311 Enable field 1→Z if address bit 12 = 1 (6-9) (used only on a partial write for those characters not being replaced)

B8 E314 Clear K700/701, activate read (6-5); turn off X and Y drivers (6-17 and 6-21)

B10 E320 Set K720/721, inhibit FF (6-5); Partially enable inhibit drivers (6-27)

C1 E321 Set K730/731, activate write (6-5); enable X and Y drivers (6-17 and 6-21)

C3 E322 Partially enable setting of K790/791, parity error FF (6-3 and 6-15), if disable parity (only on a partial write to check parity on those characters not being replaced)

D3 E323 Clear write designator bit FFs (6-9)

D4 E330 Clear K782/783, Z→data bus (6-9) (redundant on write storage)

D9 E334 Clear K710/711, gate FF; K720/721, inhibit FF; and K730/731, activate write FF (6-5).
End of write cycle

E2 E331 If no partial error or disable partial error, clear "S" reg (6-7)
If no partial error or disable partial error, clear K760/761 R/W FF (6-3)

E6 E336, E338 Turn on all drivers (6-17 and 6-21) to discharge transformers

E7 E332 Clear K770/771, busy FF (6-5)

INTERFACE

There are a number of paths for transfer of data and control signals between processors and the storage module. Signals are received by the storage module on receiver cards designated RXXX. Signals are sent back to the processor via transmitter cards designated TXXX. These signals are listed in table 12.

Table 12. INTERFACE SIGNALS

LEFT BUS	RIGHT BUS	FUNCTION OF SIGNAL
R620	R621	Memory Request
T620	T621	Reply
R622	R623	Write
R624	R625	Master Clear
R626	R627	Disable parity
R640	R641	$\overline{\text{Write character 3}}$
R642	R643	$\overline{\text{Write character 2}}$
R644	R645	$\overline{\text{Write character 1}}$
R646	R647	$\overline{\text{Write character 0}}$
R648	R649	$\overline{\text{Write address}}$
F600-R612	R650-R662	Bus to S, bits 0-12
R700-R723	R750-R773	Bus to Z, bits 0-23
T700-T723	T750-T773	Z to bus, bits 0-23
T774	T775	Parity fault

Signals required for direct access to 3309 Storage Module.

1. "Not" (inverted) state of 24 data bits to storage designated $\overline{Z}_X$.
2. "True" (not inverted) state of 24 data bits from storage designated Z_X.
3. "Not" (inverted) state of 13 address bits to storage designated $\overline{S}_X$.
4. Five partial write signals designated W_X dictate writing into a storage location according to the following assignments:

WRITE	W_0	W_1	W_2	W_3	W_4
Bits 00-05	1	-	-	-	0
Bits 06-11	-	1	-	-	0
Bits 12-17	-	-	1	-	0
Bits 18-23	-	-	-	1	0
Bits 00-14	1	1	0	0	1
Bits 00-16	1	1	1	0	1

5. $\overline{\text{Read}}$, master clear, disable parity, request signals to the storage module.
6. Reply, parity error signals from the storage module.
7. "True" (not inverted) state of 24 data bits from storage on same lines as data to storage.
8. Busy, drive fault, module present, 8K present signals not used as logic except in case of static. Busy and drive fault light an indicator. For the stated condition the line goes to ground, busy drives 100 ma at -20v, drive fault drives 250 ma at -20v. Module present and 8K present are wired directly to ground for the condition stated.

All signals except those listed in 8 above are standard 3300 - type transmitter/receiver pairs. Timing for signals is shown in figure 118. Times before discontinuity relate to the request. Discontinuity is a wait caused by priority scanning and/or storage busy. Times after the discontinuity relate to the reply. Shaded areas represent variations which can or do exist.

Signals defined in 1, 3, 4, and 7 above, in addition to read, request, and reply are necessary. All others may or may not be used.

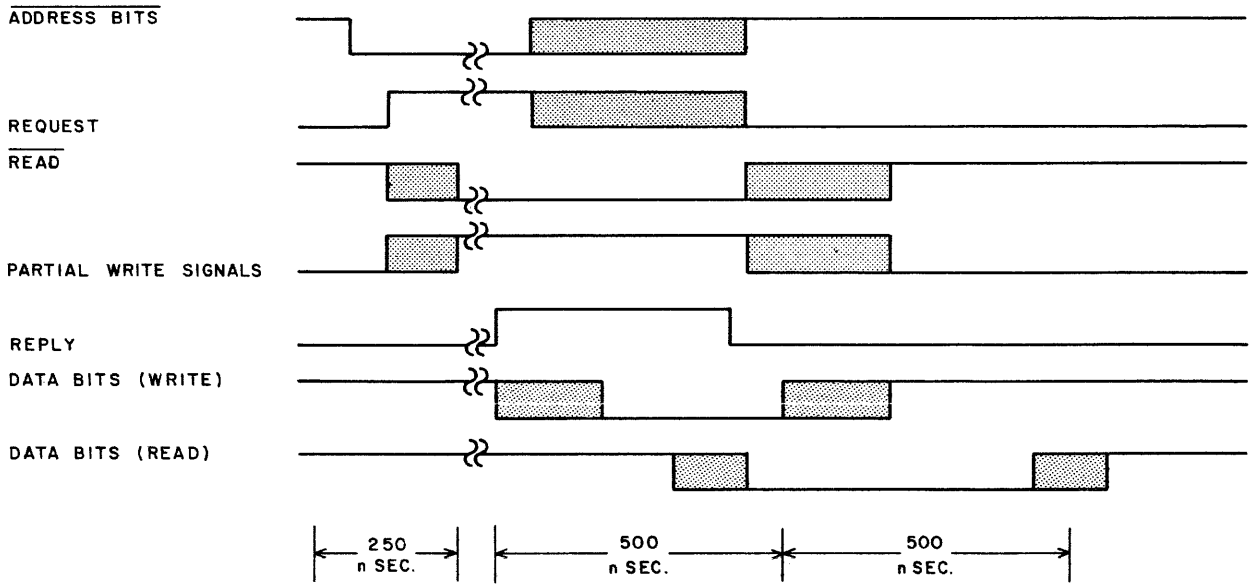


Figure 118. Storage Reference Timing

PIN NUMBERS	CARD POSITIONS 04	03	02	01	ROW
1	+	+	+	+	
2	- $\overline{Z18}$	- $\overline{Z12}$	- $\overline{Z6}$	- $\overline{Z0}$	
3	+	+	+	+	
4	- $\overline{Z19}$	- $\overline{Z13}$	- $\overline{Z7}$	- $\overline{Z1}$	
5	+	+	+	+	
6	- $\overline{Z20}$	- $\overline{Z14}$	- $\overline{Z8}$	- $\overline{Z2}$	
7,8,9	UNUSED				M
10	+	+	+	+	
11	- $\overline{Z21}$	- $\overline{Z15}$	- $\overline{Z9}$	- $\overline{Z3}$	
12	+	+	+	+	
13	- $\overline{Z22}$	- $\overline{Z16}$	- $\overline{Z10}$	- $\overline{Z4}$	
14	+	+	+	+	
15	- $\overline{Z23}$	- $\overline{Z17}$	- $\overline{Z11}$	- $\overline{Z5}$	
1	+	+	+	+	
2	- W0	- $\overline{S12}$	- $\overline{S6}$	- $\overline{S0}$	
3	+	+	+	+	
4	- W1	- $\overline{Read}$	- $\overline{S7}$	- $\overline{S1}$	
5	+	+	+	+	
6	- W2	- Reply	- $\overline{S8}$	- $\overline{S2}$	
7,8,9	UNUSED				N
10	+	+	+	+	
11	- W3	- Master clear	- $\overline{S9}$	- $\overline{S3}$	
12	+	+	+	+	
13	- W4	- Disable parity	- $\overline{S10}$	- $\overline{S4}$	
14	+ Interrupt	+	+	+	
15	- from other processor	- Parity error	- $\overline{S11}$	- $\overline{S5}$	

NOTE: The right bus wiring enters the module via the flex-prints in the card positions shown. The wiring leaves the module via flex-prints in card positions 41-44. The signal assignments for position 01 correspond to those for position 44, etc.

Figure 119. 8K Memory Module Flexprint Assignments

3300 Memory Power Supply

Power supply is not covered in this manual.

Circuits not covered so far in this manual are shown in figure 120.

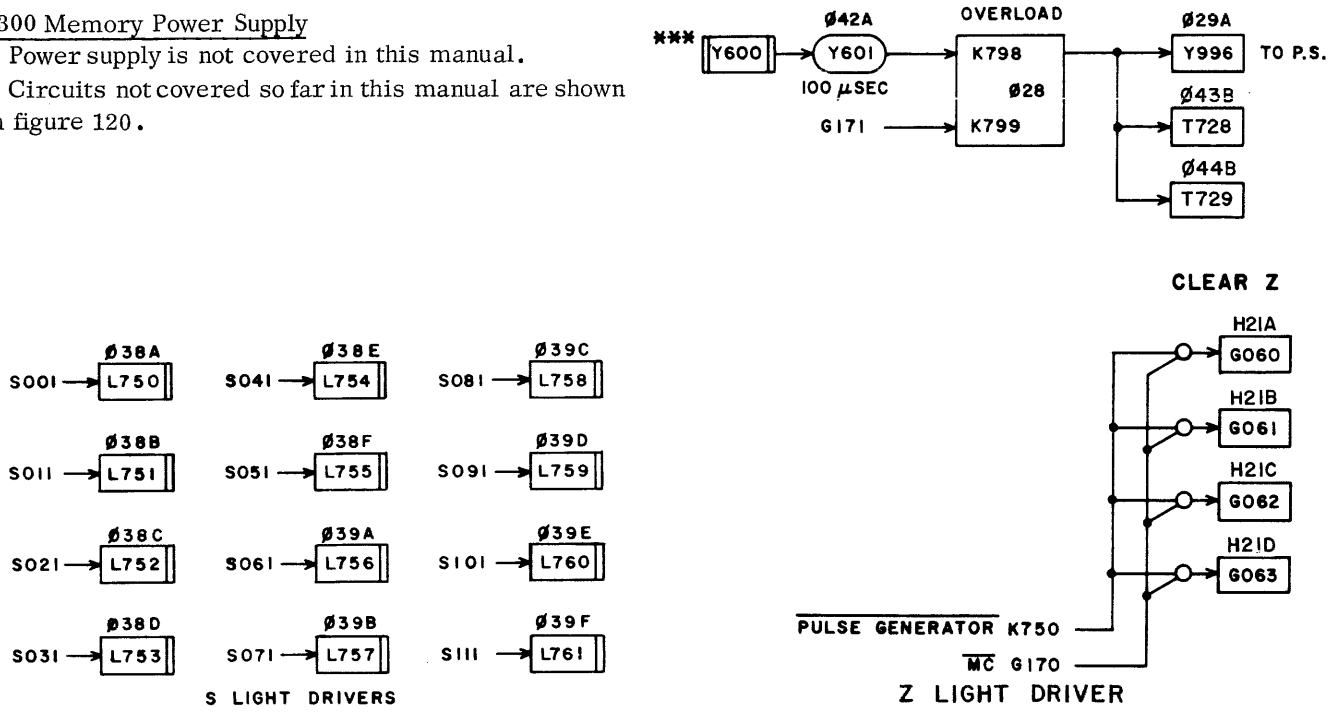


Figure 120. Additional 3300 Circuits

3302 STORAGE MODULE

Two processors may share the use of a 3302 Storage module. To reference a 3302 storage module, either the processor or the Multiprogramming module must send a Storage Request signal accompanied by: 1) a 3-bit module select code and 2) a 14-bit coordinate address to the storage module. A scanner controlled by a switch located on the storage module control panel determines which processor has access to storage.

When a request is recognized, storage control gates the 14-bit coordinate address from the S bus into the S register, where it is translated to select the proper drive lines for a memory reference.

A memory cycle is then started which consists of:

1. A Read phase, during which time a word of information is removed from an address in memory, followed by
2. A Write phase, during which time a word of information (either the same word or a new word) is written into memory at the same address.

The processor will specify either read from memory or write into memory.

1. Read from Memory. Memory performs a Read re-store cycle in which a word is taken from memory, entered into Z, and re-stored in its original memory location. While in Z, the word is transmitted to the equipment via the data bus.
2. Write into Memory. Memory performs a read cycle, with entry to Z blocked for all or certain portions of the word from storage. Portions of the word to be blocked from Z are determined by the type of partial write mode selected (selection depends on the instruction or cycle being executed).

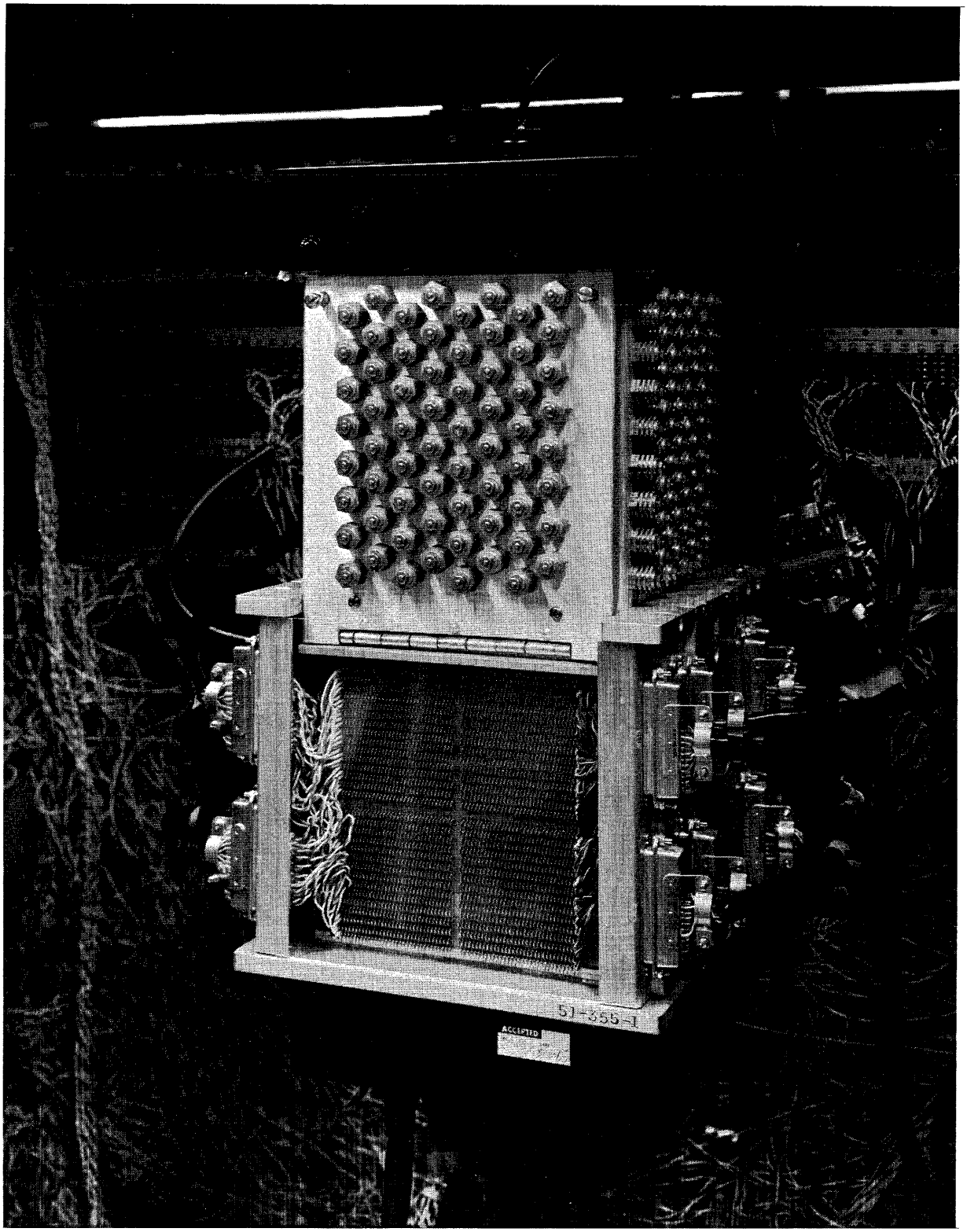


Figure 121. 16K Stack

STORAGE Worksheet #1

A 3304 with four 3309's and no Multiprogramming module is in use at this installation. The system operates Non-executive. The morning maintenance check uncovered a malfunction in the system. Indications were as follows:

After writing 1's throughout storage (enter continuous operation) subsequent sweep operations indicated that all 0's were being read from locations 40004, 40005, 40014, 40015, 40104, 40204, and some higher-numbered locations. The symptoms established a pattern.

Answer the first questions below and follow the directions given!

1. What memory module is causing the trouble? Module # ____. Go to step 1.

2. (Answer yes or no) Can a sense amplifier be at fault? _____. Go to step 2.

3. (Answer yes or no) Could the error be in the inhibit circuitry? _____. If your answer is yes, go to step 3. If your answer is no, go to step 4.

(1) Module 2 is causing the trouble. This can be seen by looking at the addresses (in binary notation) that failed:

40004 = 100 000 000 000 100
40005 = 100 000 000 000 101
40014 = 100 000 000 001 100
40015 = 100 000 000 001 101
40104 = 100 000 001 000 100
40204 = 100 000 010 000 100

The upper 2 bits of the address always indicate the module number. In this case, 10 is considered a 2; therefore, the trouble exists in module 2.

Another way of remembering which modules contain which addresses is to remember this table:

Module Number	Contains Addresses
0	00000-17777
1	20000-37777
2	40000-57777
3	60000-77777

What module is causing the trouble if addresses 50004, 50005, 50014, 50015, and 50104 are dropping bits? Module #____.

If your answer was other than 2, go back and review the step--now! If your answer was 2, go to question 2 in the first part of this worksheet.

(2) NO! A sense amplifier could not be at fault. It would require at least ____ sense amplifiers malfunctioning to cause a complete word to read up 0's after 1's were stored in that location. Remember there is a sense amplifier for every bit of the word, not a sense amplifier for each word.

If you answered this question with yes, it would be advantageous for you to go back in the storage text to the area covering the sense circuitry and study it some more.

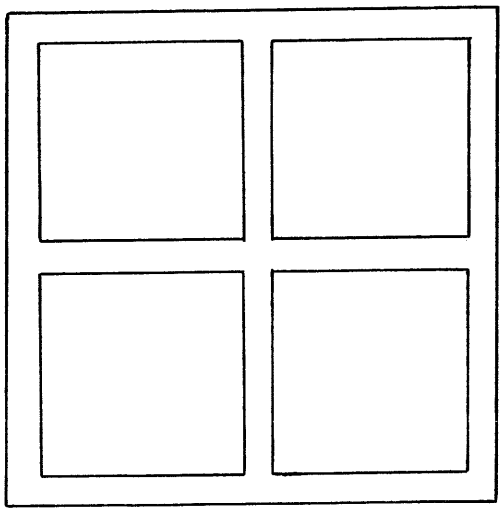
A sense amplifier (can or cannot) _____ cause the loss of 24 bits.

If you put can on the line above, you did not go back to the text and study as was suggested.

If you put cannot on the line above, go to question 3 at the beginning of this worksheet.

(3) NO! The inhibit circuitry is not at fault. It couldn't be! Remember how the inhibit circuits operate. They are in stripes throughout memory, aren't they? How many stripes are there? _____ How many drive lines are involved or paralleled by inhibit lines? _____

Draw the inhibit lines on this representation of a core plane.



There are four inhibit stripes per plane and each stripe parallels 16 drive lines. Now, how could the inhibit circuits cause the loss of all 24 bits in locations 40004, 40005, 40014, 40015, 40104, and 40204? Go to step 4.

(4) You are right--the inhibit circuits could not cause the trouble. The reason they couldn't be at fault is: _____

Several possible reasons would indicate that the inhibit circuits could not be at fault. Some of the reasons are:

a. The inhibit circuitry is associated with one bit but the error involves 24 bits from different memory locations; therefore, the inhibit circuitry is not at fault.
b. The inhibit circuitry affects stripes in memory or certain memory locations, which applies to this problem, but the stripes could not cause locations 40004, 40005, 40014, 40015, 40104 and 40204 to fail. This can be verified by reviewing the operation of the inhibit lines. If you have forgotten just how the inhibit wires operate it is suggested that you quickly review the area in this chapter that covers this circuitry.

Go to step 5.

(5) Now the problem is localized to memory module 2 by checking the appropriate bits in the failing addresses.

a. The sense amplifier circuits have been eliminated because they affect only one bit per word, not all 24 which is the problem.
b. The inhibit circuits have been eliminated for much the same reason that the sense amplifiers were. The inhibit circuits are normally associated with one bit in every word, not all 24 bits. The stripes have no effect on more than one bit. The stripes do affect a series of locations but not a series of bits.

Could the trouble be drive line transformers?_____
If you answered no, go to step 6 . If you answered yes, go to step 7 .

(6) You're right. The drive line transformers could not be at fault! Why?_____

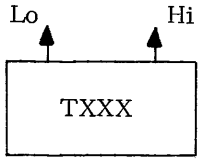
How many drive lines can one transformer feed? _____ What does Lo and Hi addresses remind you of? _____

Each transformer feeds two drive lines, one Lo address and one Hi address. In this chapter it was shown that the transformer is turned on if that particular address is needed. You will recall that transformers are selected by a unique gate and driver combination. There are X transformers and Y transformers and they supply the direction of current flow for the drive lines during both read and write.

Another circuit area in the storage module has now been eliminated. Go to step 8 .

(7) Now tell me--how could the transformers be at fault? _____

Think a moment. Remember how the transformers were presented in this chapter:



Lo referred to low addresses and Hi referred to high addresses. How many Lo and Hi addresses are involved per transformer?_____

One Lo and one Hi address is involved per transformer with one transformer for Y lines and one for X lines. It must be remembered that the transformers supply the direction of current flow on the drive lines for both read and write operations and that the transformers can be turned on more than twice in any run through memory from addresses 0000 through 7777. That is, the Y coordinates could be for any one address from 00 through 77 and the X coordinate transformer for address 00 would be turned on for each of the following addresses:

Different Y coordinates, thus different Y transformers	00 00 01 00 02 00 . . . 77 00	Same X coordinates, thus same transformer

Look again at the addresses that failed:

40004	=	100	000	000	000	100
40005	=	100	000	000	000	101
40014	=	100	000	000	001	100
40015	=	100	000	000	001	101
40104	=	100	000	001	000	100
		Y transformer		X transformer		

It's easy to see that four different X transformers are involved in these addresses. At least two different Y transformers are needed for the addresses. Therefore, the transformers could not be the problem. Go to step 8 .

(8) What circuitry in memory module 2 would you suspect as being faulty at this point? _____

_____

_____

Go to step 10 .

(9) Your choice is valid but you should be a bit more specific than you were. Can the circuitry be broken or divided more than you have done? Where would you look in S register for a failure? _____

_____

_____

If you were able to answer this question then you are ready to go back to step 8 and try again.

(10) Several different suggestions might have been made at this point. Obviously some area of circuitry in the storage module is at fault. If in step 8 you listed S register as your answer, go to step 9 . If you had any answer other than S register, go to step 11 .

(11) There were only four other logical choices you could have made at step 8 . The error is known to exist in module 2. The sense amplifiers, inhibits, and the drive line transformers have been eliminated. This leaves the:

X gates
Y gates
X drivers
Y drivers
} all controlled by S register.

One of the four listed above is faulty. Which one do you think it is? _____

We'll see. To determine which one is faulty it is necessary to analyze S register. Certain bits in S register select the circuits listed above.

What bits select each of these circuits:

X gates = bits _____, _____, _____
Y gates = bits _____, _____, _____
X drivers = bits _____, _____, _____
Y drivers = bits _____, _____, _____

The addresses that failed are written below in binary (only the lower 12 bits).

Address	Bit Number											
	11	10	9	8	7	6	5	4	3	2	1	0
40004	0	0	0	0	0	0	0	0	0	1	0	0
40005	0	0	0	0	0	0	0	0	0	1	0	1
40014	0	0	0	0	0	0	0	0	1	1	0	0
40015	0	0	0	0	0	0	0	0	1	1	0	1
40104	0	0	0	0	0	1	0	0	0	1	0	0
40204	0	0	0	0	1	0	0	0	0	1	0	0

Bits 11, 9, and 6 do not always stay the same for all failing addresses, therefore it seems practical at this point to say that the problem is probably not the _____. Bits 10, 8, and 7 change, therefore it probably is not the _____. Bits 5, 3, and 9 change also, therefore it probably is not in the _____ either. Bits 4, 2, and 1 are always in the same configuration for every address that fails, therefore the _____ seem the most logical circuit to investigate first--but which X gate? _____.

Remember in this chapter the easy method that was shown for determining the gate numbers? Just read the bits that select the gate in binary, convert to octal, and you have the gate number in question. In this case, bits 4, 2, and 1 in S register select the gate. Each time a memory location fails, the bits involved are 0, 1, and 0 or 010 or X gate 2.

Refer to the complete set of 3209 prints and supply the location number of the bad gate. X gate _____ located at _____ at module____.

Go to step 12 .

(12) You should have G002 located at H09A in module 2. This circuit just about has to be the problem. Review the procedure for troubleshooting this problem by reading through the flow-charted summary. (figure 133).

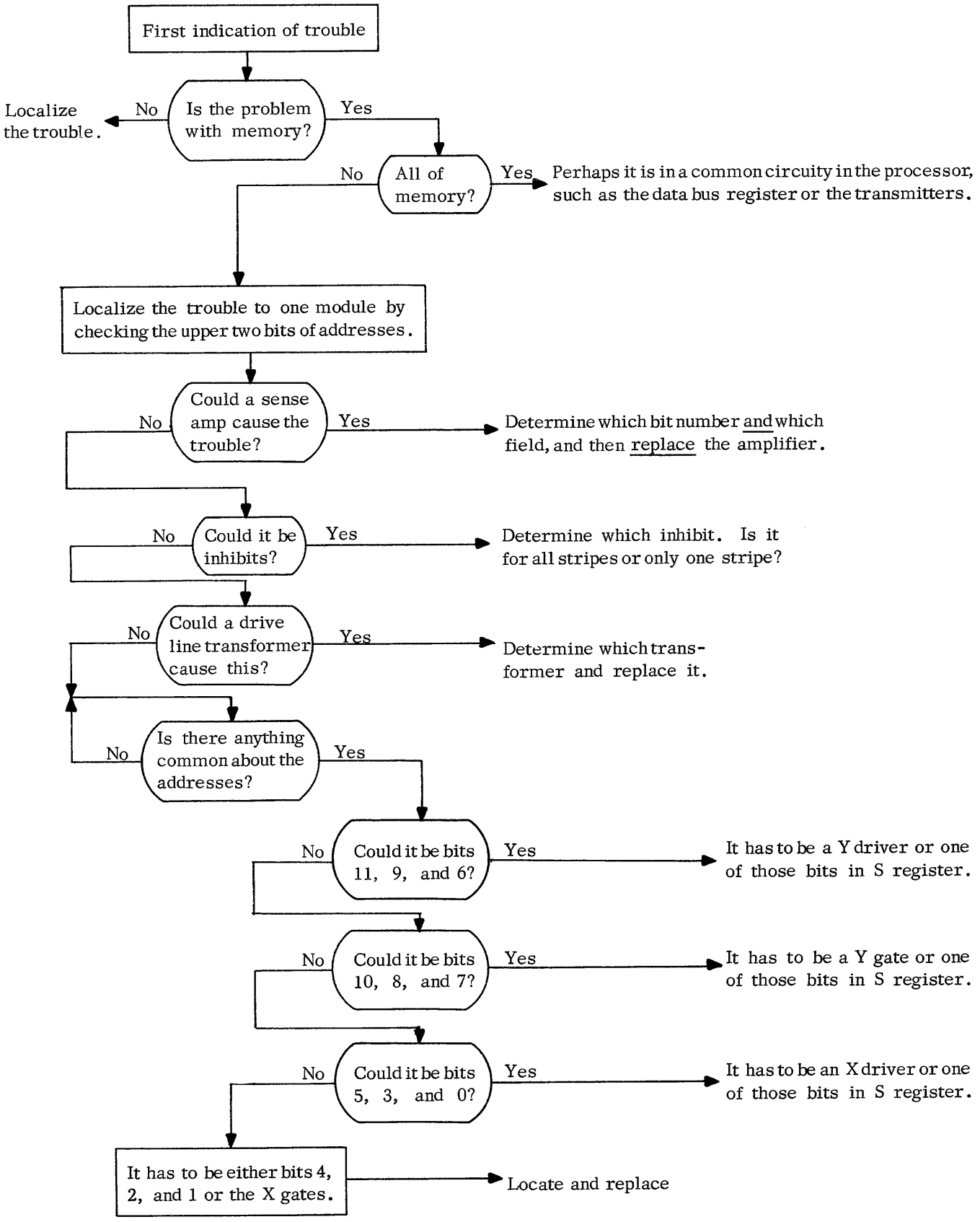


Figure 122. Troubleshooting Flow Chart #1

STORAGE Worksheet #2

A 3304 with four 3309's and no Multiprogramming module is in use at this installation. The system operates Non-executive. During morning maintenance is first located when the console typewriter types out the following information while running a test routine:

Location 20040: Bit 2 set; should be clear.

After typing out, the test routine stops.

NOTE: For purposes of this worksheet any faults that would cause the computer to interrupt or halt are ignored. The purpose of this worksheet is to familiarize the student with general operation of some specific circuit. The fault circuits are covered in other problems.

Answer each question and follow the directions given.

What memory module is causing the trouble? Module # ____. Go to step 1 , answer each question, and follow the directions given.

(1) You could not know at this point if the fault is in a memory module! The most you know right now is that one memory location in storage possibly is at fault. It is not wise to make such a hasty decision with so little information. You could be right; the error might be in a memory module--but you might be wrong.

Have you verified that there is an error? List the steps of the operation you would perform at this time to verify that an error does exist that perhaps is common to all of storage.

List the steps of the operation you would perform to verify the trouble. _____

_____

_____

_____

_____

_____

Step 2 contains some logical first steps, one of which you may have written above. Look through the list for the operation you would have performed. If the operation you listed is there, follow the directions given.

(2)
a. You entered (via keyboard) a load A from location 20040. You then executed the load A. If a load A was your choice, go to step 3 .
b. You swept contents of location 20040 to C register for observation. If this was your choice, go to step 4 .
c. You entered (via keyboard) all 7s into C register and stored (C) in memory location 20040. Then you executed a sweep from that location in storage. If this was your choice, go to step 5.
d. You cleared location 20040 and then did a sweep to C for observation. If this was your choice, go to step 6 .
e. You cleared all of memory, then did a sweep or check of all addresses in storage. If this was your choice, go to step 7 .
f. If the operation you performed was not listed above, perhaps your choice was not one of the most logical ones you could have made. Analyze your operation. What help is it to you in verifying that the system does not work? Perhaps the operation you chose would have helped you and it was just not listed above.

Either way, now that you have read through the list of logical choices supplied above, which one of them do you feel is the most logical? Reread the problem, make another choice (from a to e above), and follow the directions given.

(3) After executing the load A from memory location 20040, (A) = 12106037.

Knowledge gained: Bit 2 coming to A register as a 1 could be significant. How do you know that the faulty bit (bit 2) in memory is a 1? You do not know, do you? By the same token, what if some circuitry is faulty--data bus for example--and a 0 actually is in bit position 2 in storage, but because of the faulty circuitry becomes a 1 between storage and A register?

You wasted precious time by doing this load A operation. You do not know what is in storage, so how can you analyze what comes to A register? (If you noticed the possibility of a parity error refer to the note on the first page of this worksheet.)

Go back to step 2 and make another choice. Remember you are trying to verify the existence of an error.

(4) After you sweep the contents of location 20040 to C register for observation you will find that C contains 12106037.

Knowledge gained: Bit 2 coming to C register as a 1 could be significant. How do you know that the faulty bit (bit 2) in memory is a 1? You do not know, do you?

By the same token, what if some circuitry is faulty--data bus for example--and a 0 actually is in bit position 2 in storage, but because of the faulty circuitry the 0 becomes a 1 between storage and C register?

You wasted precious time by doing a sweep at this time. You do not know what is in storage, so how can you analyze what comes to C register? (If you noticed the possibility of a parity error refer to the note on the first page of this worksheet.)

Go back to step 2 and make another choice.

Remember you are trying to verify the existence of an error.

(5) When you sweep up the 7s you entered in location 20040 you will find that C register contains all 7s.

What did you expect? The original indication stated that bit 2 was set and should be clear. You stored all 7s and they came back as 1's, so you proved nothing.

This was a waste of precious time. Go back to step 2 and make a better choice.

(6) When you do a sweep to location 20040, after you clear it you will find (C) = 00000004.

Knowledge gained: There definitely is a problem. You stored all 0's in location 20040, but bit 2 comes back as a 1. This indicates that at least one memory location is faulty--but do the others react the same way? A good way to find out would be to do step 2 , item e. Read it and follow the directions given.

(7) Clear all of memory and then, as you auto step through memory, bring contents of locations to C for observation. Use auto step here (sweep continuous could not give a visual display of C register) or write a small program to check the contents of each location for 0's. Then the following occurs:

Whichever operation you perform (auto step or a program to check for 0's in all locations, the first indication of an error occurs at location 20040. The contents of that location is 00000004; it should be all 0's. Bit 2 of address 20040 comes back a 1 when it should come back a 0. (If you wonder about the possibility of a parity error refer to the note on the first page of this worksheet.)

What is the most logical operation you could perform now to see if other locations also fail?

_____

_____

_____

_____

_____

_____

_____

_____

Go to step 9 .

(8) Module 1 is causing the trouble. This can be seen by looking at the addresses (in binary notation) that failed.

The upper two bits of the address always indicate module number. In this case 01 is considered a 1; therefore, the trouble exists in module 1.

Another way of remembering which modules contain which addresses is to remember this table:

Module Number	Contains Addresses
0	00000-17777
1	20000-37777
2	40000-57777
3	60000-77777

What module is causing the trouble if addresses 20004, 20005, 20014, 20015, and 20104 were dropping bits? Module #____.

If your answer was 1, go to step 10 . If your answer was other than 1, go back and review the step--now!

(9) Probably the most logical operation to perform at this time would be to run your sweep operation or checking program again to determine if other locations besides 20040 are getting a bit in position 2 when they should not.

Assuming you do check memory either via continuous sweep or a program operation you will find the same errors (bit 2 is a 1) in the following addresses:

20040 ⟶ 20057
20140 ⟶ 20157
20240 ⟶ 20257
etc.
37640 ⟶ 37657
37740 ⟶ 37757

All other addresses contain all 0's and all bits except bit 2 in the above addresses are 0's.

The faulty circuity exists in only one storage module. Which one?____

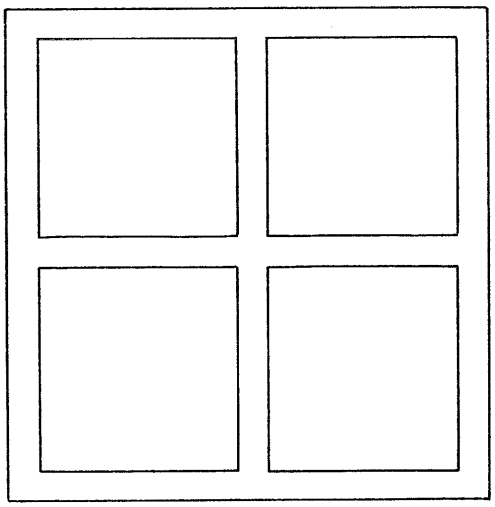
If you answered 1, go to step 10 . If you answered 0, 2, or 3, go to step 8 .

(10) Storage module 1 is the faulty one. The only addresses that failed are located in module 1. It is known that module 1 is faulty in some manner. Could a sense amplifier in storage module 1 be faulty? (yes or no) _____

If you said yes, go to step 11 . If you said no, go to step 12 .

(11) No, a sense amplifier could not be at fault. In storage module 1 only selected addresses are failing, not all addresses. The same sense amplifier is used for bit 2 for every location in module 1.

You should have remembered this from the description of the storage module in this chapter. For review draw the sense lines on the following representation of a core plane.



You should have four different sense windings, one for each bit position. Do you remember what addresses are involved in each quarter of the core plane? All addresses are involved! Go to step 12 .

(12) A sense amplifier could not be at fault because not all addresses in module 1 are failing, just selected ones.

It has been determined that only addresses ending in 40 to 57 in module 1 are failing (see step 9 for complete listing) and only one bit is incorrect in each of those addresses.

What circuitry would you suspect as being at fault at this time? _____

_____

Go to step 13 .

(13) If only addresses ending in 40 to 57 are failing and only one bit in each of those locations is incorrect, you should have listed the inhibit circuits as your choice for step 12 . This is the most logical choice because it is the only circuit left in the storage module that is associated with only one bit.

If you listed the inhibit circuits as your choice for step 12 go to step 14 .

If you do not see how the inhibit circuits could be the logical choice at this time read on for a moment.

The sense amplifier could not be at fault. All locations use the same sense amplifier for bit 2 and all locations do not fail. Gates, drivers, transformers, or S register would affect more than one bit in any location, thus they are not at fault. Z register could not be at fault because all addresses use Z register and all addresses do not fail.

Unless there is an outlandish problem like the stack being bad or a cable or wire not making contact the only logical choice that could have been made was the _____ circuits.

Go to step 14 .

(14) The logical choice is the inhibit circuitry--but what part of the inhibit circuitry?

Some of the addresses that failed are listed in binary below:

Some Addresses	Bit Numbers (Lower 6 Bits Only) 5	4	3	2	1	0
20040	1	0	0	0	0	0
20045	1	0	0	1	0	1
30156	1	0	1	1	1	0
32351	1	0	1	0	0	1
33454	1	0	1	1	0	0
37757	1	0	1	1	1	1

Note that bits 5 and 4 are the same for each address that failed. The upper two bits (5 and 4) of the X series of bits (5 through 0) select the inhibit stripe during reference to those locations. It seems most logical that the fault is in an inhibit stripe. What inhibit stripe is faulty? Stripe #____ .
Go to step 15 .

(15) You should have listed stripe 2! Bits 5 and 4 read in octal are a 2 (10_2 is a 2_8). It is known that inhibit stripe 2 is bad. It is also known that it is only common to bit 2. Therefore, the faulty circuit is not the inhibit gate, which controls bits 0, 1, 2, and 3. The fault must be in the inhibit _____ for bit 2.

Open your manual of storage prints to the page containing circuitry for the inhibit drivers and load compensator drivers. In the lower left corner you will find the inhibit gates, controlled by bits 5 and 4 from S register. Only two of these eight gates (G020 to G027) will be outputting a 1. Which two? G_____ and G_____.

G022 and G026 will be outputting a 1 because they are the gates for X addresses from 40 to 57.

The trouble is not in the gate; it is in the bit itself. Look up the left side of the same page in the prints until you find the inhibit generator for bit 2. (It will be controlled by bit 2 from Z register and either G022 or G026.)

What inhibit generator is at fault? _____

_____. Where is it physically located? _____

Go to step 16 .

(16) You should have D522 located at J02A for your answers to step 15 . Go to the summary flow chart (figure 123).

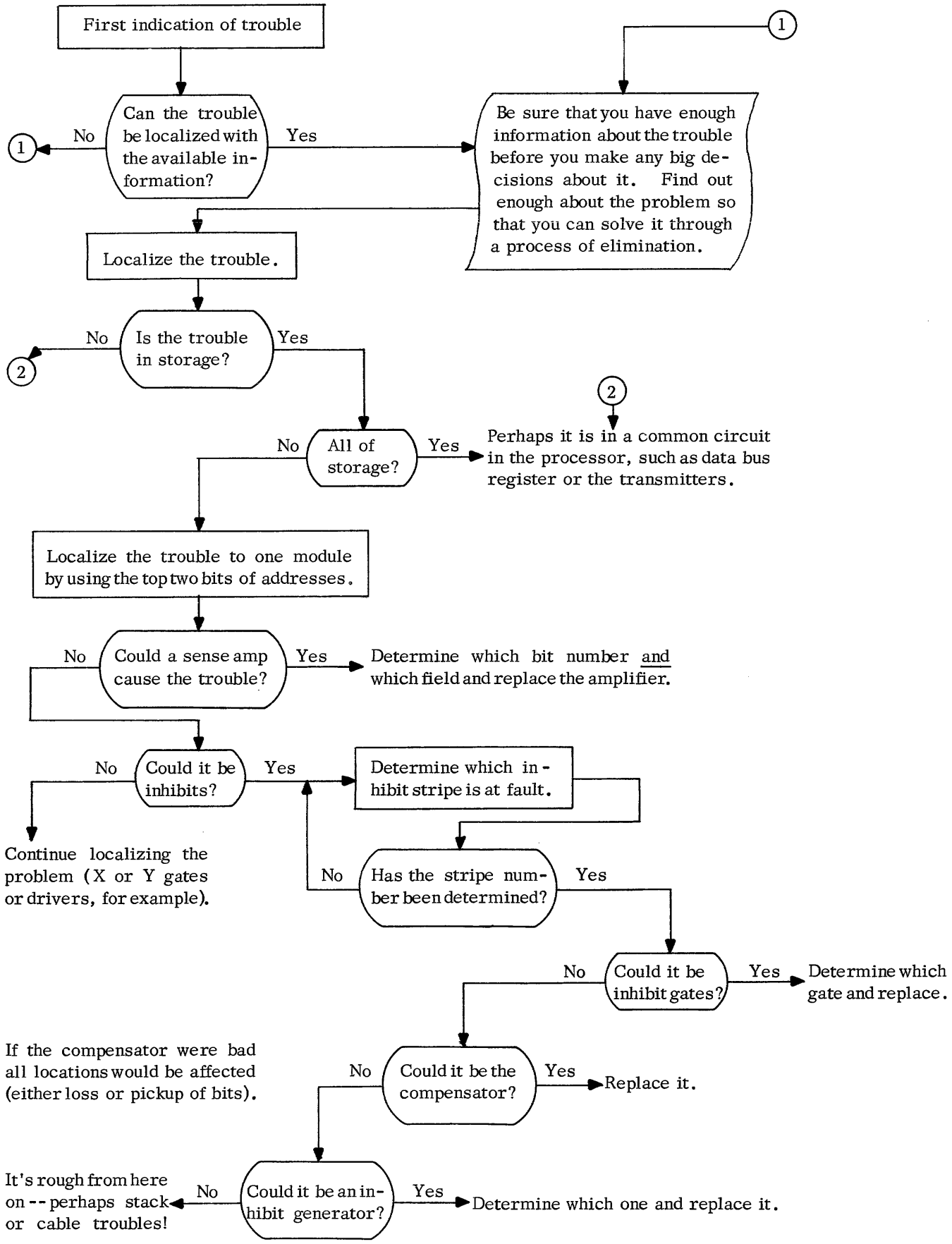


Figure 123 Troubleshooting Flow Chart #2

SELF-EVALUATION QUIZ ON CHAPTER 4

TRUE OR FALSE (T or F)

1. The 3302 consists of two 3309s.
2. The 3309 stack has four planes per wafer.
3. The 3309 checks parity on every write operation.
4. Stop on storage parity error is switch selectable.
5. Each inhibit driver drives a 16 x 64 matrix of cores.
6. Both the 3302 and the 3309 check parity for each character of a storage word.
7. Bit 27 of the storage word is the parity bit for character 0.
8. A master clear within the 3309 cannot occur until 2 usec after storage reference has been initiated.
9. A master clear may be performed from the control panel of the 3309.
10. Adjacent storage modules are connected via flex jumpers.
11. If desired, 32 3309s may be used to form 262K storage capacity.
12. The 3309 senses for illegal write conditions.
13. A _____ bit module select code is necessary for the 3302.
14. A 3309 may be used by more than one processor.
15. The 3302 contains twice the control logic of the 3309.
16. Address is supplied to the storage modules via the S bus.
17. Data is supplied to the storage modules via the data bus.
18. There are twice as many R cards for bit 14 of the S bus as there are for bit 07.
19. The physical placement of the storage modules in relation to the processor is determined by the module number.
20. There are 28 wafers in a 3309 memory stack.

Score yourself:
Missed none or one? Not bad--matter of fact, excellent!
Missed two? Satisfactory
Missed three or more? WAKE UP!

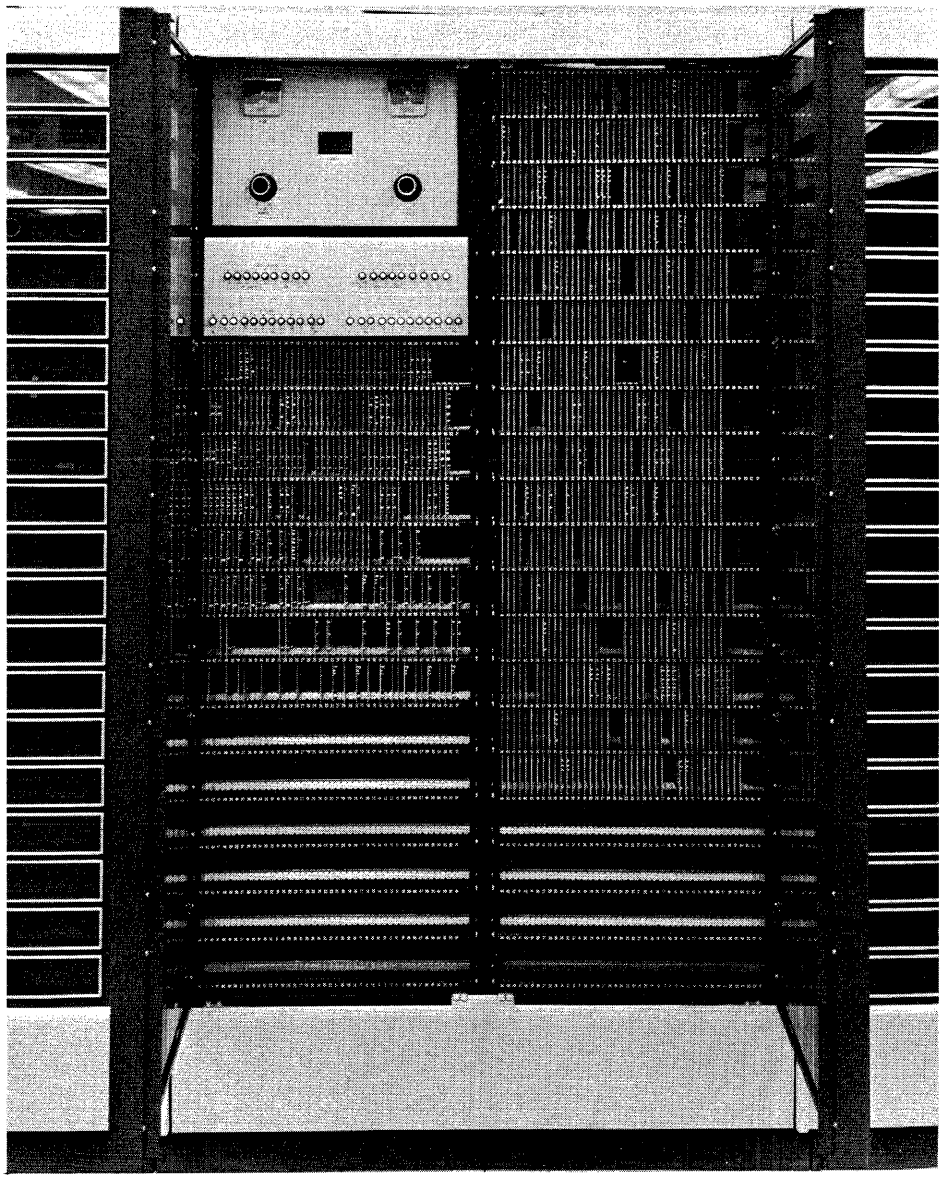


Figure 124. 3311 Multiprogramming Module

CHAPTER 5

3311 MULTIPROGRAMMING MODULE

INTRODUCTION

The Multiprogramming module is optional to the 3300 System and if incorporated into the system it will:

1. Allow a maximum of 262K of core storage
2. Segment core storage into 2K pages
3. Allow dynamic relocation
4. Sense for Illegal Storage Reference

The 3311 is physically located in Chassis 5 and consists primarily of a 24-bit, 64-word, word-organized storage--called the Page Index File--and control logic. Figure 124 is a photograph of the

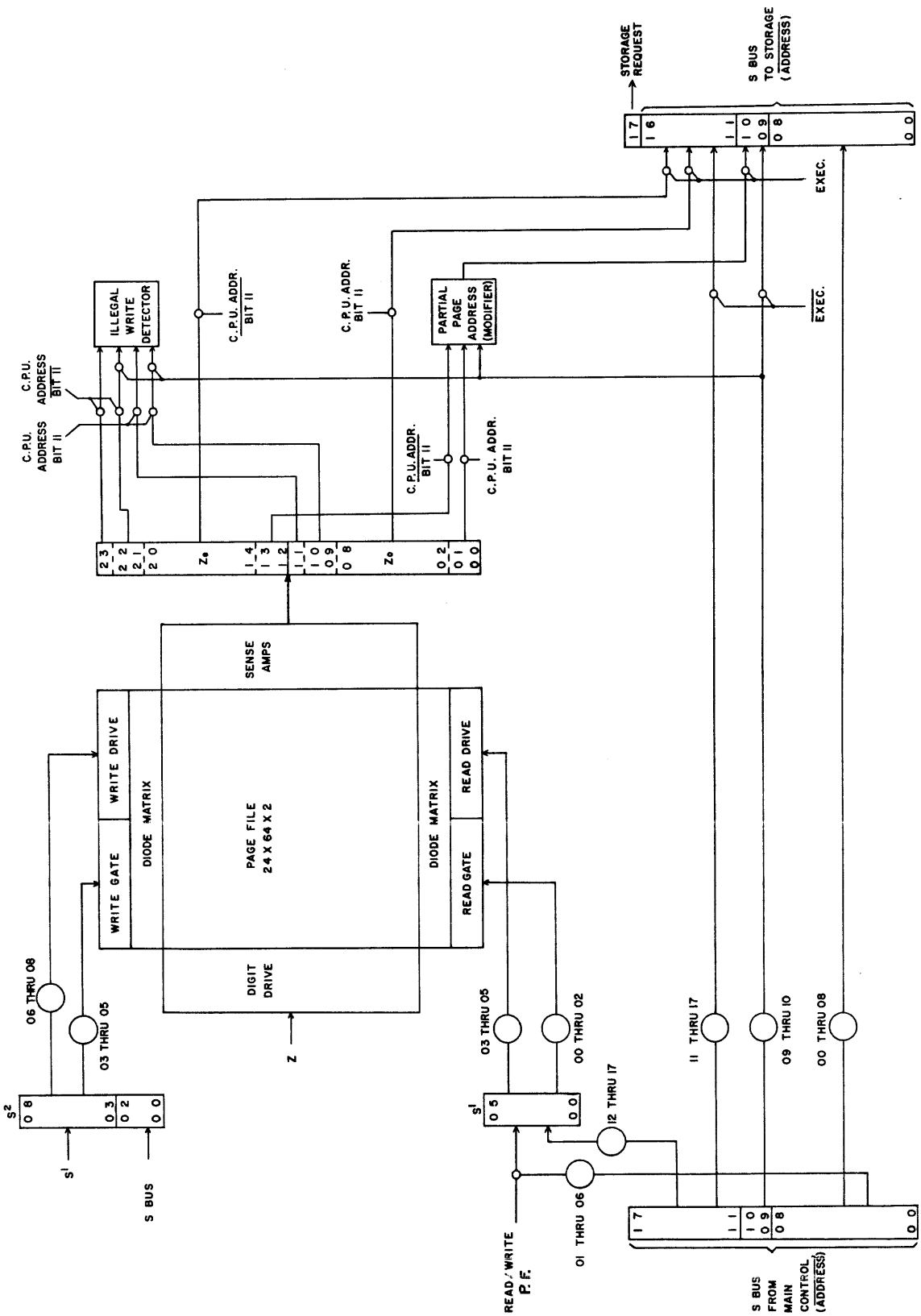


Figure 125. Page Index File Block Diagram

Multiprogramming module. Note that binary register displays are available on the control panel. Figure 125 shows the physical location of the Multiprogramming module.

If the system is operating in the Non-Executive mode, the 3311 will not relocate addresses. If the system is operating in the Executive mode, the 3311 is used for every storage reference. Before discussing the overall operation of the 3311 it is necessary to understand the principles of word-organized storage.

The approach to presenting material in this chapter is based on the assumption that the reader has familiarized himself with theories of magnetic core storage as contained in chapter 4.

GENERAL INFORMATION

The Page Index file is a 64-word (24 bits per word) rapid-access memory. The Page Index file uses the word-organized rather than the coincident current technique. The primary advantage of word-organized memory lies in the ability to use larger read drive currents than are possible in coincident current memories. These larger read currents produce faster core switching speeds and much faster access time (250 nsec).

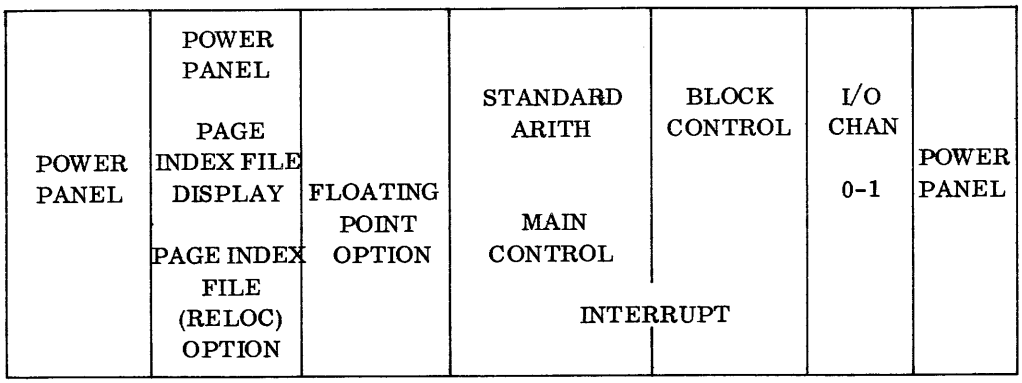


Figure 126. 3300 Physical Layout

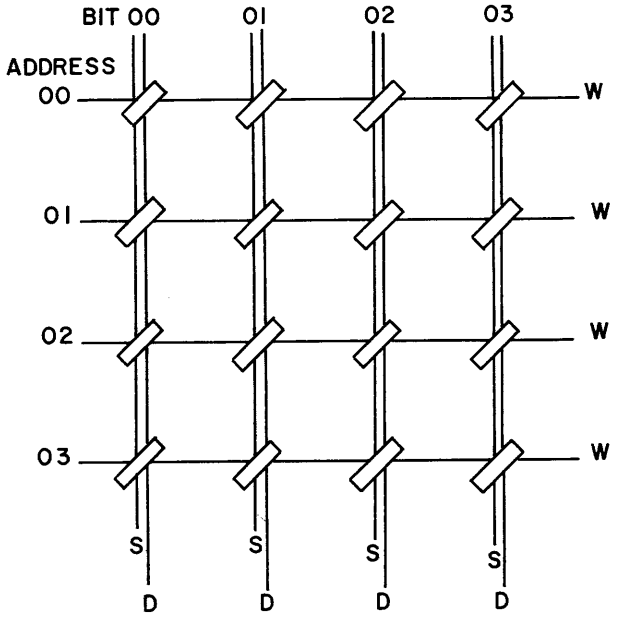
WORD-ORGANIZED MEMORIES

A word-organized memory differs from coincident current memory in that all of the cores for a given address lie along only one drive (word) line. A word-organized memory has its cores arranged in a matrix as shown in figure 127.

A 4-bit, four-address matrix is illustrated; however, the matrix assembly would be the same regardless of address capacity or word length.

As is shown in figure 127, a word line is common to all cores in one address. A sense line is common to all cores in one bit as is the digit line. The cores are magnetized in either direction by a combination of word and digit current. In this manner the cores are able to store a logical 1 or a logical 0. The direction of the core's flux field will indicate whether the core is storing a 1 or a 0.

To extract data from an address, a read current is applied to the word line. This current aligns the fields of the cores on that word line in the same direction. Each core that was storing a 0 will already have its field in this direction and does not switch as a result of the read current. The field of each core that was



D = Digit line S = Sense line W = Word line

Figure 127. Word-organized Memory Matrix

storing a 1 is forced to switch direction by the large amount of drive current.

The switching of a core induces a voltage on the sense line which is detected by a sense amplifier. It should be noted that there is one sense line for each bit position (figure 127) and that it is common to all of the addresses for that bit. The read current is going only to those cores at the selected address; therefore, although a sense line threads all of the cores in all addresses for one bit position, it does not receive an induced voltage from any source other than the core at the selected address. During the read operation the digit line is not used.

Writing into a word-organized memory retains the principles of coincident current memories and is always done with the cores at the selected address in the clear state. Writing is done with an augment-inhibit system using a digit line (figure 127) to increase or decrease the write drive current that is applied to the core. The write drive current is always applied in the opposite direction of the read drive current. To write a 1 into a bit position, the core would have to be switched as it is in the Q state from the preceding read operation. When the write drive current is applied, the digit current must be in a direction that will augment the drive current and cause the core to switch. If a 0 is to be written the digit current must be in a direction that will effectively counteract (inhibit) enough of the write drive current to prevent the core from switching. Thus the digit driver must be capable of providing a bidirectional current on the digit line.

The digit current is represented by +I and -I. When a 1 is to be written into a bit position, the applied digit current is in the +I direction and produces a total applied current of 3I when added to the write current, more than sufficient to switch the core. If a 0 is to be written, the applied digit current is -I. The write drive current, 2I, is inhibited by the digit current of -I and results in a total effective current of I, which is not of sufficient amplitude to switch the core.

As mentioned earlier, the primary advantage of a word-organized memory lies in its ability to use large drive currents to produce faster core switching. Because the cores at the unselected addresses do not receive drive current, as opposed to their receiving half drive current in coincident current memories, generally the only factor limiting the size of the read drive current is the current-carrying capacity of the transistor switches that select the word lines.

Two-Core Bits

A disadvantage common to both word-organized and coincident current memories is the fact that a different number of cores are switching each time data is read or written. This results in a varying load on the drive line and requires extensive compensating circuits.

An effective means for overcoming this disadvantage is to use two cores for every bit position. A word-organized memory using two cores per bit, as in the register file, has its cores arranged in a matrix as shown in figure 128.

The center horizontal line through the cores (figure 129) is the word line which is doubled so as to loop through each core twice. Only one is shown. This has been done in order to double the effective current that is applied to the core. Note that the sense and digit lines at core B are threaded in opposition to the word line while in core A they are in the same direction. Therefore, when current is applied to the sense and digit lines, the digit current will be in the same direction as the drive current in one core while it will be in opposition to the drive current in the other.

Regardless of the storage of a 1 or a 0 in a bit position, the flux state of the two cores are in opposite directions. If the bit being stored is a 1, the flux field of core A (figure 129) will be in the direction indicated and core B will have its flux field in the opposite direction. If the stored bit is a 0, both flux fields will be the opposite to that when holding a 1.

To read, a read current pulse is applied to the word line at the selected address. This read current will attempt to align all of the cores along that word line with their fields in the R direction. Note that before this, one of the two cores at each bit position already had its field in the R direction. When the read current is applied, the remaining core at each bit position will switch to the R direction. Core switching will induce a voltage on the sense line. The polarity of this voltage will depend on which of the two cores switched. Had the bit position been storing a 1 when the read current was applied, core B would have switched. If a 0 had been at the bit position, core A would have switched. With this type of read out, sense amplifier design is greatly simplified, requiring only a differential amplifier to detect the change in polarity.

Writing uses the same principle discussed earlier and is done with the fields of all of the cores at the selected address in the R direction following the read. With two cores per bit, however, the digit line must be threaded in such a manner that the digit current will be able to augment the drive current in one core while it inhibits the drive current in the other. The direction of the applied digit current determines which of the two cores will produce effective drive current sufficient to cause it to switch.

Writing into the register file uses a two-thirds write scheme. With this, the drive current (2I) makes up two-thirds of the total current that will be applied to the core if that core was selected to be switched. The write current is of less magnitude than the read current and is in the opposite direction. In order to switch a core the digit line provides the remaining one-third of the applied current. The digit line is threaded through the two cores (figure 129) so that while the digit

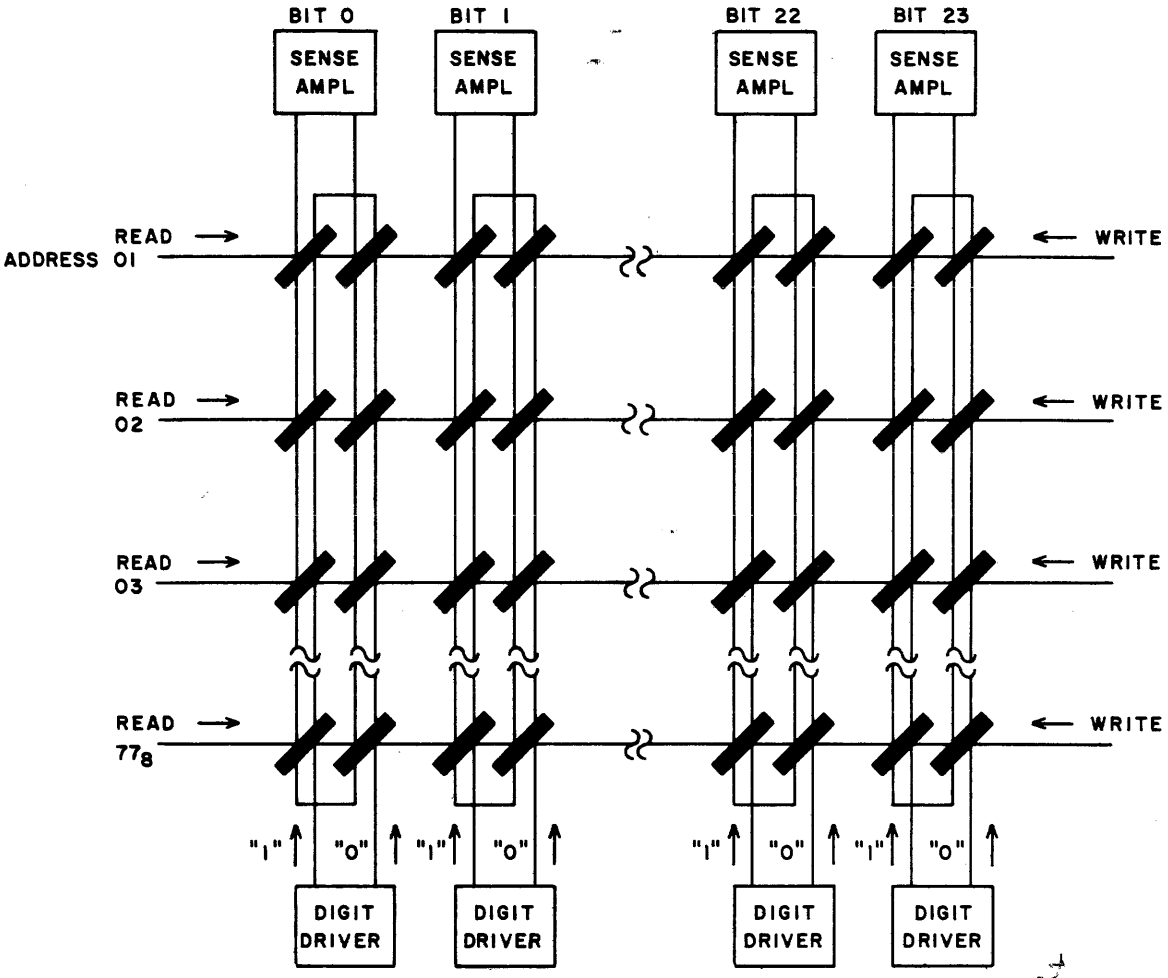


Figure 128. Matrix of Two-core Bits

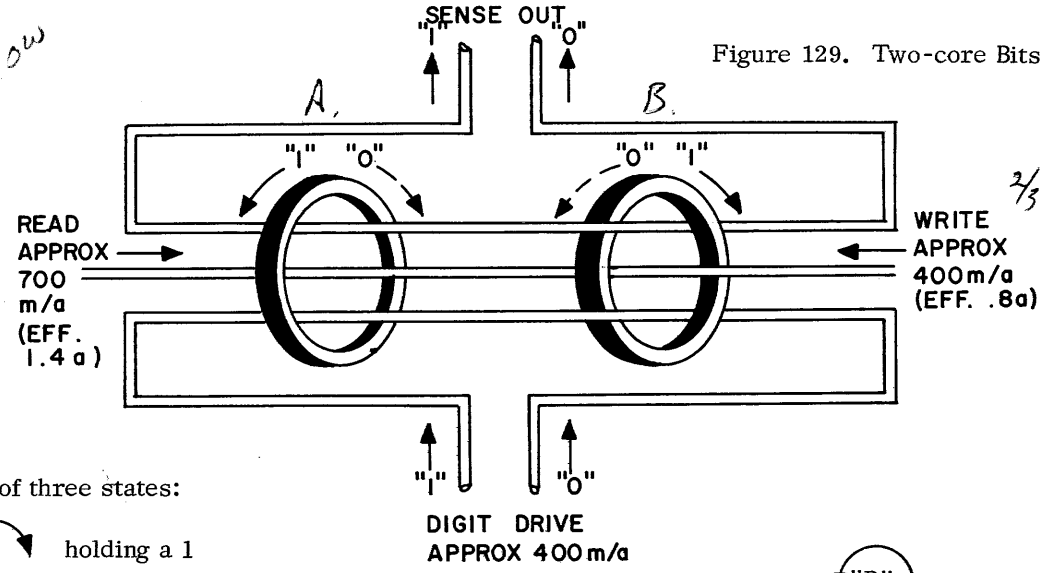
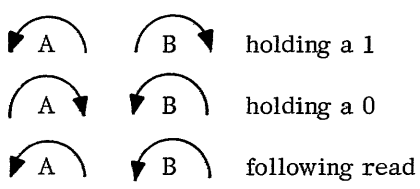


Figure 129. Two-core Bits

The cores can be in any of three states:



direction referred to in text

S1 - READ
S2 - WRITE

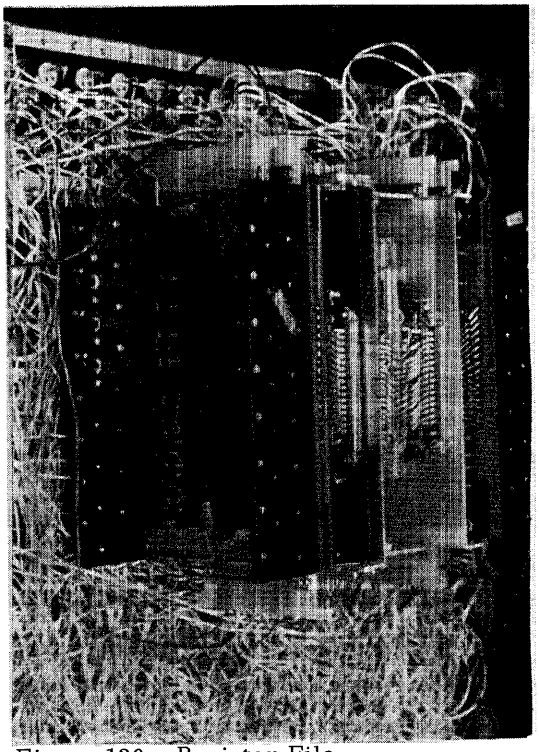


Figure 130. Register File

current is augmenting the drive current in one core, it will inhibit the drive current in the other core preventing it from switching. To switch a core the drive current, 0.8 effective, is augmented by the digit current of 0.4 amp and results in a total effective drive current of 1.2 amp. The magnitude of 1.2 amp is more than sufficient to switch the core. The other core receives a current of 0.8 amp and -0.4 amp, a total effective current of 0.4 amp which is not sufficient to switch the core.

With a 24-bit word using 48 cores, no matter what data is read or written the same number of cores, 24, will switch for any given read or write operation. A uniform number of cores switching each time will produce a constant load on the word line.

Another important advantage of a two-core-per-bit system has already been mentioned: The bipolar output on the sense line simplifies sense amplifier design and provides a more positive method of recognizing the difference between a 1 and a 0.

ORGANIZATION

A photograph of the register file is shown in figures 130 and 131. It consists of the following basic sections:

1. Memory stack - contains the magnetic cores and the sense, digit and word lines.
2. Read and write drive boards - contain the circuits for the gates and drivers that will send the drive current into the stack.
3. Diode matrices - used to select drive lines and to couple the drive current into the stack.
4. Logic translators - translate the output of S register in order to enable the proper gate and driver.

Other logic associated with the register file are:

1. S register - a 6-bit double-ranked register used to hold the address to be referenced. The first rank (S1) holds the address for the read logic translators and the second rank (S2) holds the address for the write logic translators.
2. Sense amplifiers - 24 sense amplifiers that detect and amplify the voltages induced on the same line during read.
3. Digit drivers - 24 digit drivers that augment or inhibit the switching of cores in order to write a 1 or a 0 into each bit position.
4. Z register - a 24-bit single-ranked register that serves as an exchange register for reading or writing.

A block diagram of the register file is shown in figure 132. The following sections will explain each component shown on the diagram.

STACK

The memory stack has two planes (0 and 1). Each plane has 1755 cores arranged in a 65 by 27 array. There is one spare address and three spare bits at each address which are not used.

Figure 133 shows how the word lines are threaded through the cores. Only the word lines for addresses 0 and 77 are shown. The writing scheme, however, is the same for all of the addresses. Read and write drive currents are applied to the word lines.

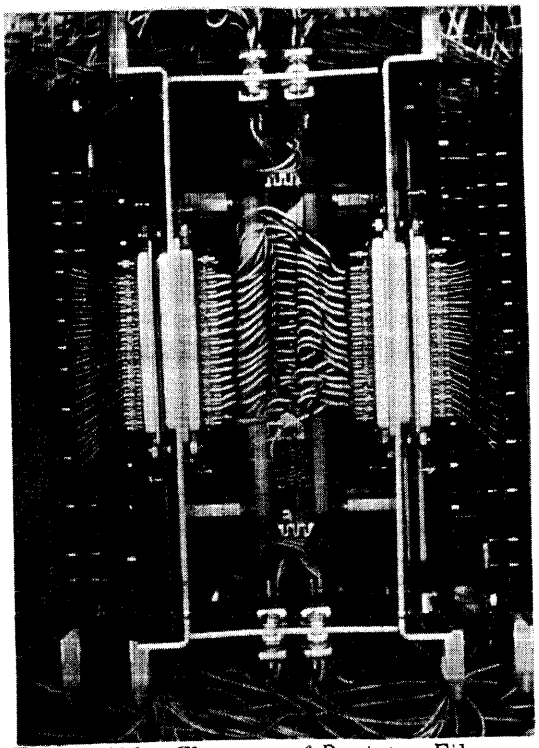


Figure 131. Close-up of Register File

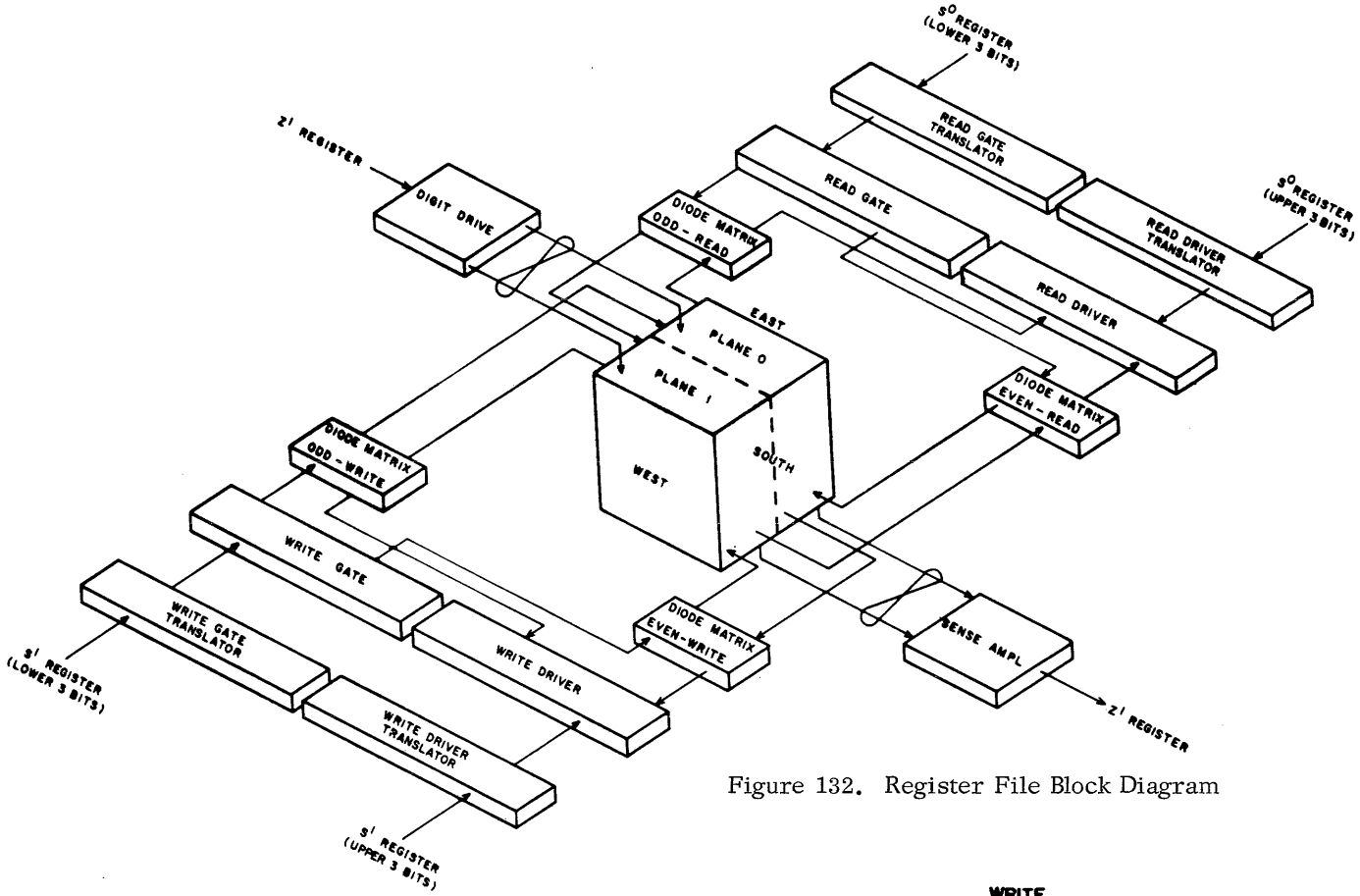


Figure 132. Register File Block Diagram

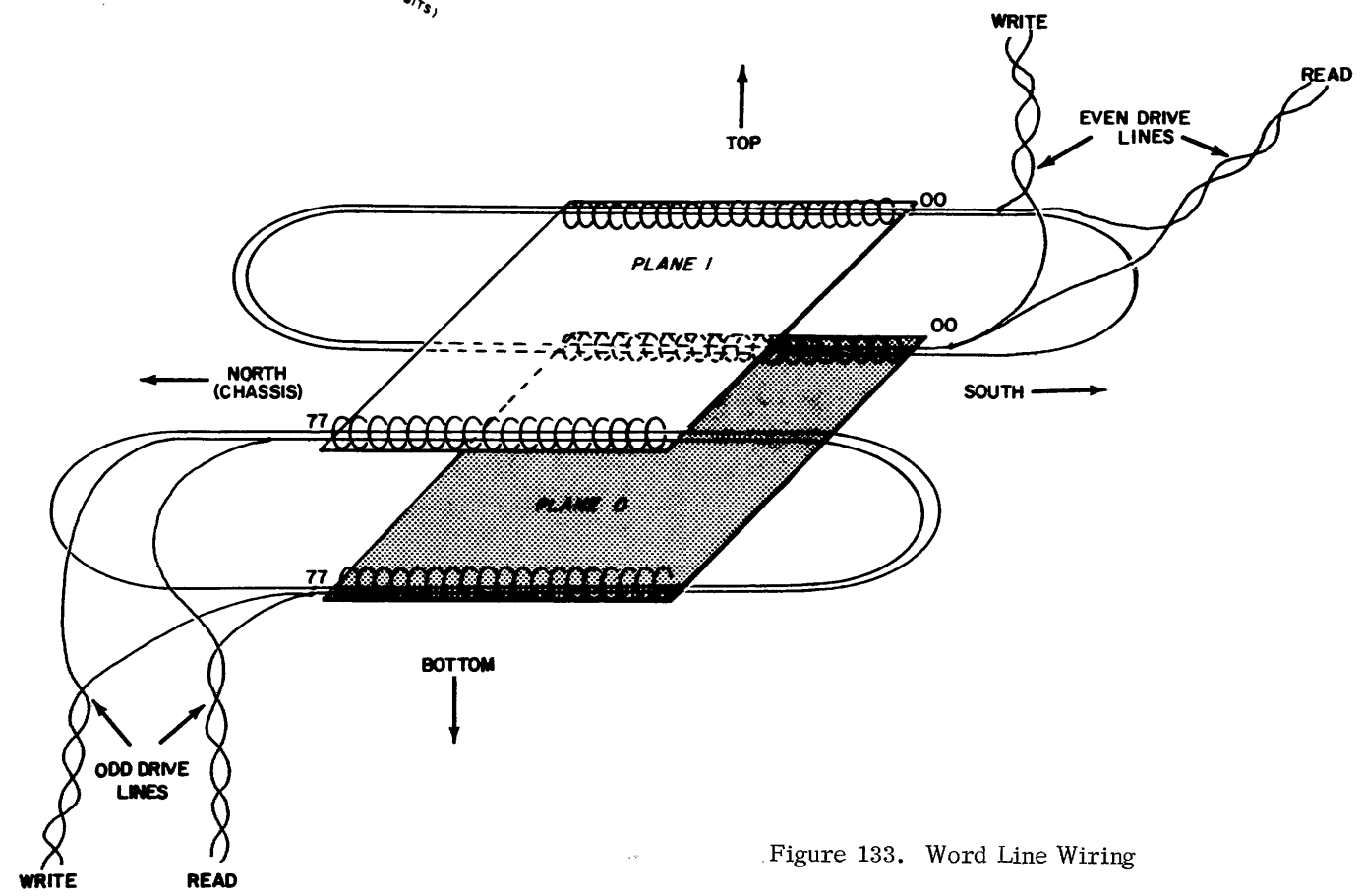


Figure 133. Word Line Wiring

It should be noted that the word lines for the odd-numbered addresses enter from the north side of the stack while the even address word lines enter from the south. Small tabs are provided along the edges of the stack for connecting the word lines and to provide jumpers between planes. At the tabs where the word lines connect there are two lines attached. One of these wires goes to the read drive board and the other goes to the write drive board.

Within the stack a word line is provided for each address. A word line enters a plane and passes through all of the cores at that address. The word line leaves one plane and passes to the next by way of a jumper tab on the edge of the stack. The word line threads all of the cores on this plane for the same address, returns to the first plane, and repeats again. In this manner each core is threaded twice by the same word line to double the effective drive current applied to the cores. The actual drive current flowing through the word line during a read operation is 0.7 amps, but due to the word line passing through the cores twice, the current through the core is 1.4 ampere turns. During write, the word line current is 0.4 amps while the current through the core is 0.8 ampere turns.

The digit and sense lines enter from the top and bottom of the stack, respectively, and are threaded vertically through the cores. Figure 134 shows the digit and sense wiring for bits 0 and 23 only. The wiring scheme, however, is the same for the entire stack. Note that the two cores for each bit position are mounted on separate planes.

A digit and sense line pass through all of the cores in all addresses for a particular bit. For example, the digit and sense lines for bit 0 will pass through all of the bit 0 cores in the 77_8 addresses.

The cores that make up the magnetic storage elements are referred to as 20 mil cores. They have an outside diameter of 0.020 in., an inside diameter of 0.013 in., and are 0.0035 in. thick.

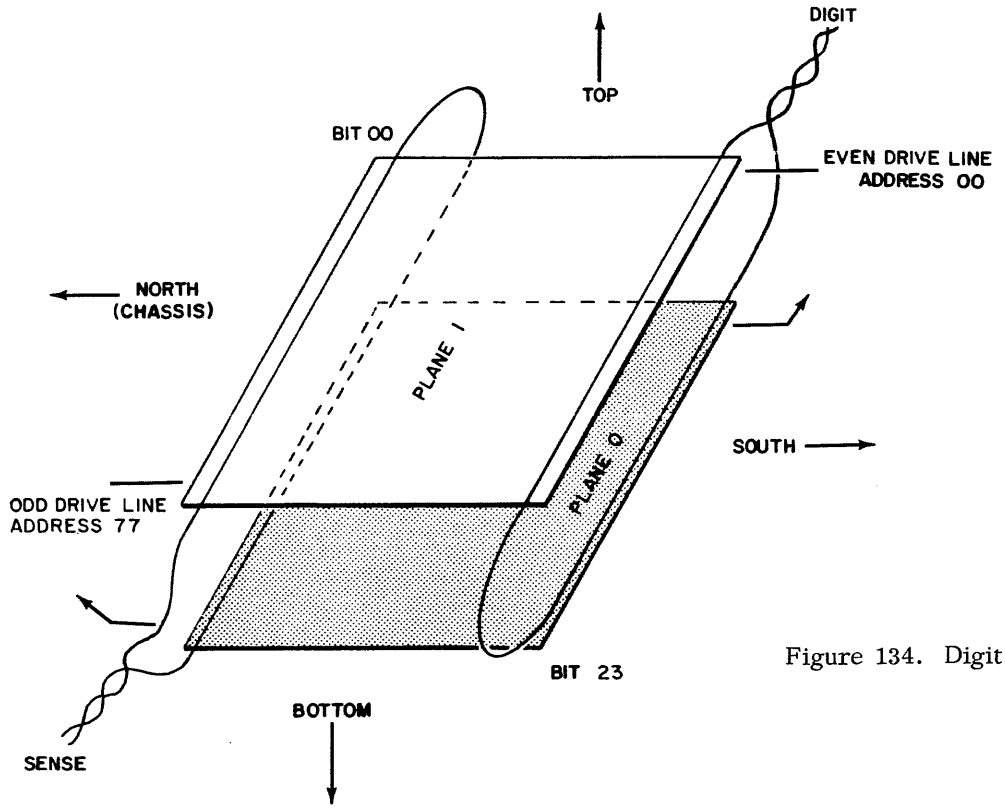


Figure 134. Digit and Sense Line Wiring

WORD LINE SELECTION

A word line is selected whenever a driver and a gate are enabled. The drivers and gates serve as switches to allow the drive current to flow through a particular word line. A simplified diagram of this is shown in figure 135.

When reading, a pair of logic translators will enable a read gate and a read driver. Current will flow in the word line from -20v through the word line to +20v. When writing, another pair of logic translators will enable a write gate and driver allowing current to flow through the drive line in the opposite direction. When reading or writing, the gate serves as a +20v switch and the driver serves as a -20v switch.

A logical representation of the drive line selection system for all 77 addresses is shown in figure 136.

The heavy lines between the diodes represent the word lines. The address number appears beside each

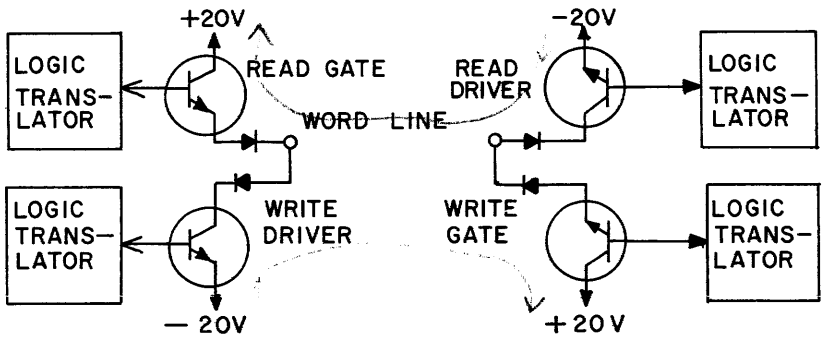


Figure 135. Word Line Selection

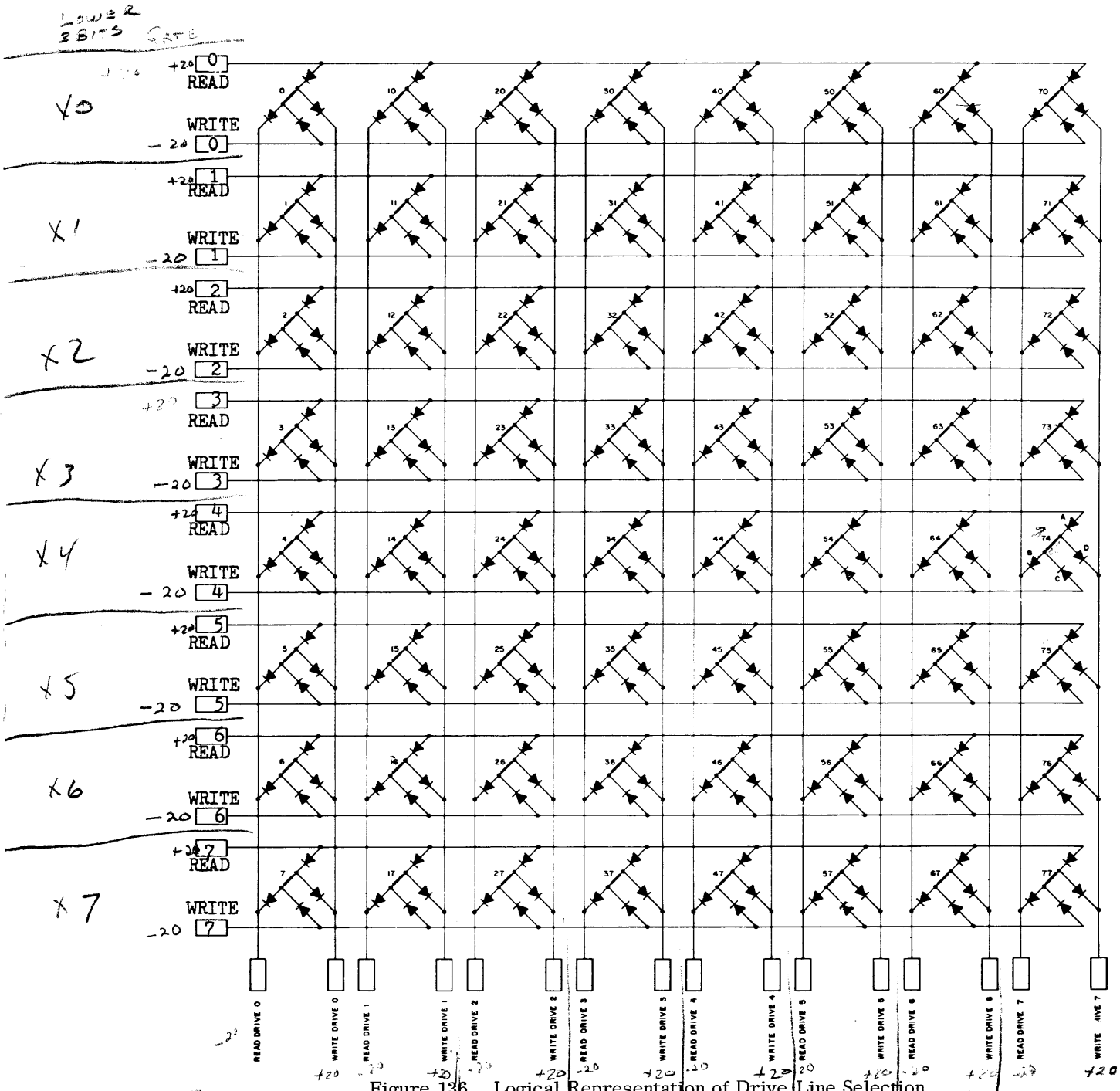


Figure 136. Logical Representation of Drive Line Selection

word line. To select any one of these word lines a driver and a gate must be enabled. To select address 74, for example, a logic translator would enable read gate 4 while another enabled read drive 7. When the current is applied, it will flow from -20v (read drive 7), through diode B along the word line, ~~through diode B along the word line~~, through diode A to +20v (read gate 4). When the write current is applied, write gate 4 and write drive 7 are enabled, allowing current to flow through the word line from diode D to diode C. Diodes A and B are the same diode matrix as are diodes C and D. The other diode matrix serves in the same manner except that they are connected to all of the odd address word lines.

Read and Write Drivers

Mounted on either side of the register file (figures 130 and 131) are the read and write drive boards. The circuits for the two are identical except for pin numbering. Figure 137 illustrates a driver and a gate.

The control signal input is the output of a logic translator. The logic translator will enable a path from ground to the +20v connected to the primary of the interstage transformer and force the transistor into conduction. The conduction of the transistors will allow the drive current to flow from the -20v at the driver

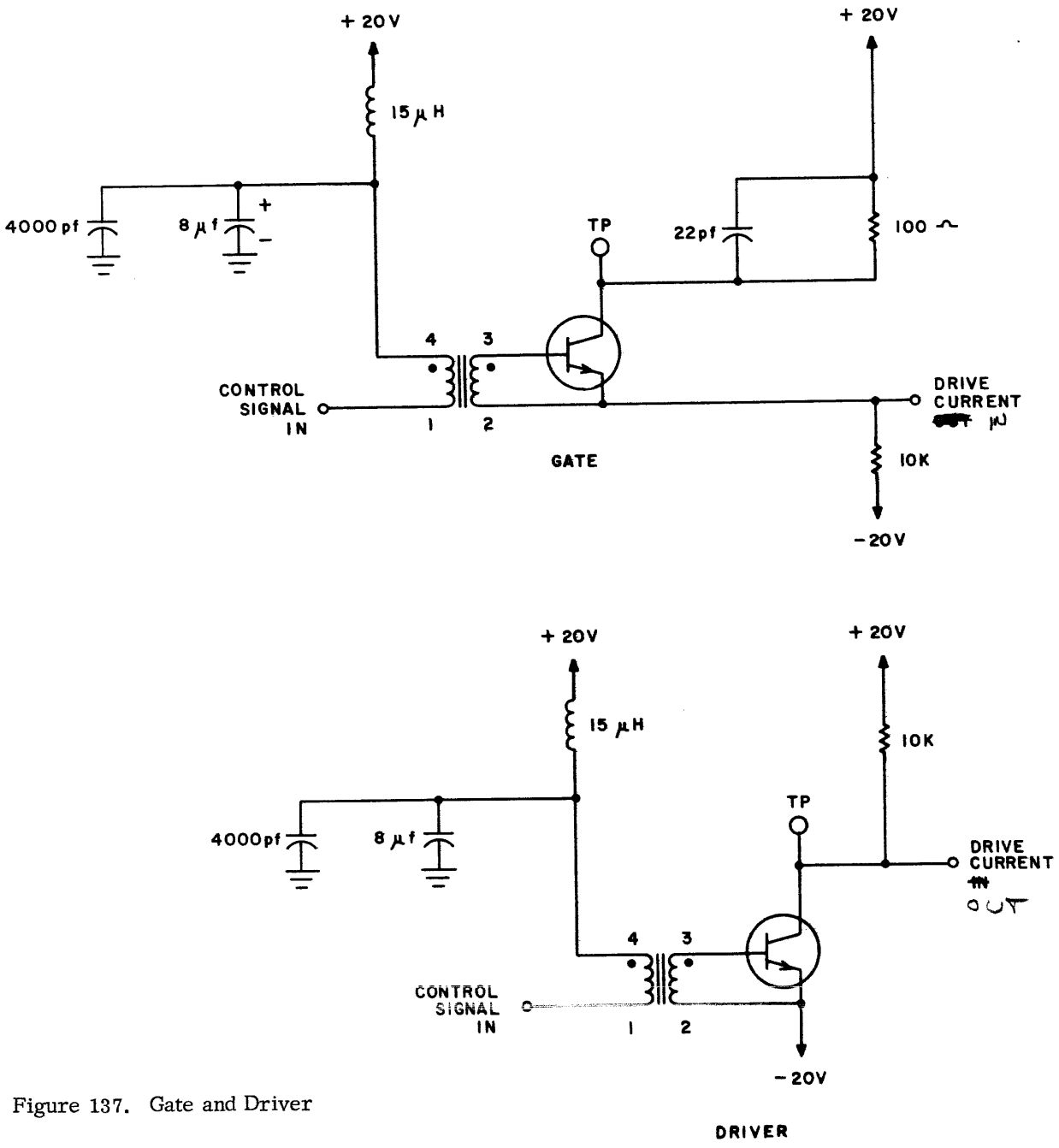


Figure 137. Gate and Driver

through the diode matrix (one diode), through the stock, through the diode matrix (the other diode of the pair), and through the gate to +20v.

LOGIC TRANSLATOR

Four identical logic translators are used in the register file. Their purpose is to decode the output of S register in order to enable the proper gate and driver. A section of the logic translator is shown in figure 138.

Transistor Q1 is used as a switch to allow current to flow from +20v through the primary of the gate or driver transformer to ground. Resistors R4, R5, and R6 form a current-limiting network for the circuit, while R3 reduces the collector voltage to prevent the transistor from burning out. When the transistor first conducts, C1 will bypass the current-limiting network and allow a large initial flow of current. At this time the current will be limited by the inductance of the transformer.

The base circuitry, R1, R2, and CR5, provides biasing for the transistor and level shifting for operating from -1.1v and -5.8v logic levels. The diodes CR1, CR2, and CR4 provide a means of translating S register contents. The biasing is arranged such that if a logical 0 (-1.1v) appears at the cathodes of all the diodes, a positive voltage is applied to the base of Q1 and it conducts. If the cathodes of any of the diodes have -5.8v (1) on them, the transistor will remain cut off. The arrangement of the diodes and the transistor is known as a NOR circuit. Therefore, the inputs to the translator will be $\overline{\text{data}}$.

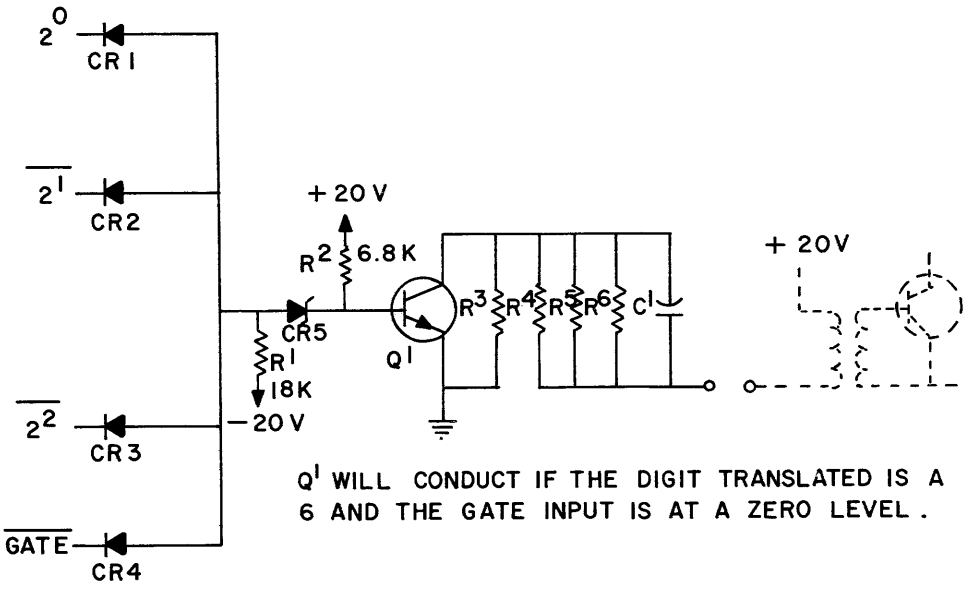


Figure 138. Logic Translator (Section)

S REGISTER

The 6-bit double-ranked S register (Logic Diagrams, page 1-95) is used to hold the address to be referenced. The first rank (S1) will send its outputs to the read logic translators and to the second rank (S2). S2 will hold the address for the write logic translators. The lower three bits of S1 and S2 send their outputs to the logic translators that enable the gates while the upper three bits go to the driver translators.

The function of S register in regard to individual instructions is explained later in this chapter.

SENSE AMPLIFIERS

The 24 sense amplifiers are type HA16 cards. A detailed description of their function is found in the Printed Circuits Manual, Publication #60042900.

A logical representation of the sense amplifier is shown in figure 139. The sense amplifier detects and amplifies the 0.1v output from a switching core. A strobe pulse (sense Z1) may be applied through an inverting circuit to the AND gate in order to gate the sense amplifiers to the Z register. If pin 4 is positive relative to pin 5 when the strobe pulse is applied, the AND gate will be made. Making the AND gate will send a 1 to the corresponding flip-flop in Z1 register. If pin 4 is negative relative to pin 5, the AND gate will not be made and the output of the sense amplifier will be a 0. The sense amplifiers are in Logic Diagrams, page 1-98.

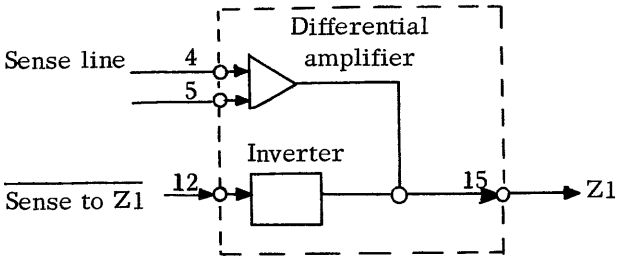


Figure 139. Sense Amplifier

DIGIT DRIVERS

The 24 digit drivers are type HA14 cards. A detailed description of their function is found in Printed Circuits Manual. A logical representation of the digit driver is shown in figure 140.

If transistors A and D are switched on, current will flow from pin 14 through the digit line to pin 13. If transistors B and C are turned on, current will flow from pin 13 through the digit line to pin 14. In this manner the digit driver is able to provide a bidirectional output on the digit line. The digit drivers are in Logic Diagrams, page 1-98.

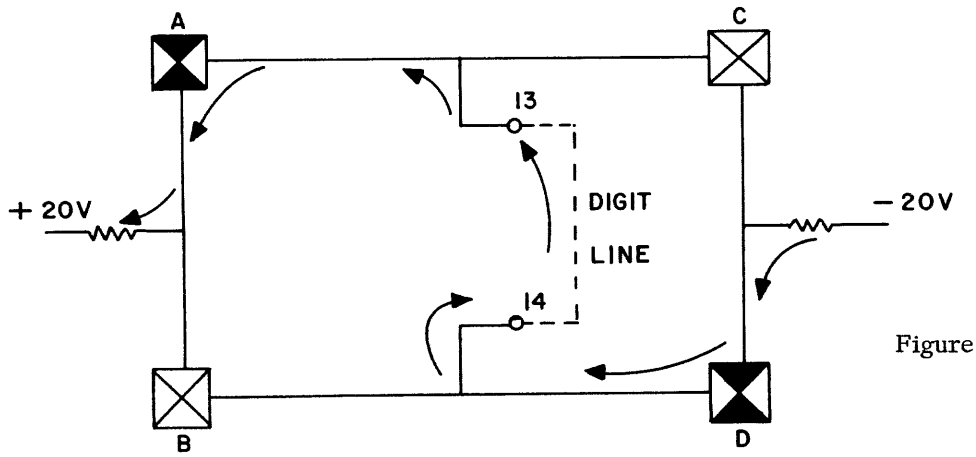


Figure 140. Digit Driver

Z REGISTER

Z register is a 24-bit double-ranked register that is used for moving data into or out of the Page Index file. Pages 1-99 through 1-102 of Logic Diagrams show the two ranks of Z register.

Z1 receives its inputs from:

1. The sense amplifiers (for reading data out of the register file).
2. The data bus (for writing in new data).

OVERALL SELECTION

The lower three bits of S1 are sent to a logic translator which in turn activates a gate. On the other side of the word line a driver has been activated by a translator which received its input from the upper three bits of S1. Read drive current flows from the driver to the gate.

When the write portion of the cycle is started, the lower three bits of S2 have enabled a gate the upper three bits have enabled a driver. Again current flows from driver to gate but in the opposite direction.

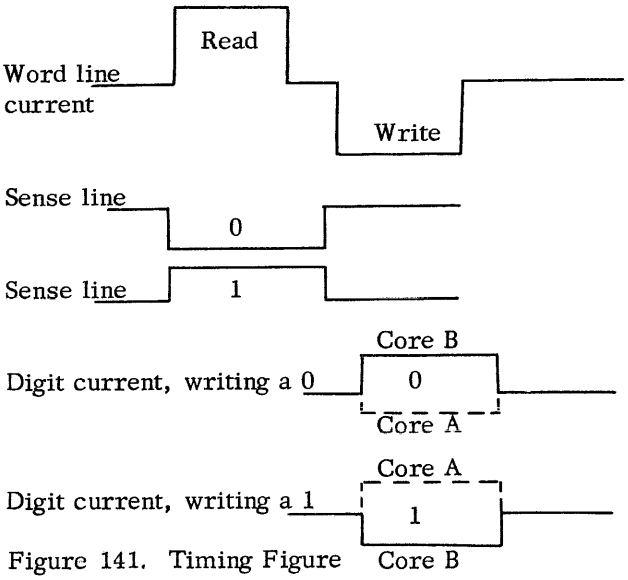


Figure 141. Timing Figure

TIMING

Figure 141 shows the overall timing and current directions for a memory cycle.

When read current is applied to the word line, the sense amplifier will receive an input as shown on figure 141. Note that the sense line voltage is the same magnitude for a 1 as for a 0 but in the opposite direction. At the conclusion of the read operation the fields of the cores at the selected address are in R direction (figure 129).

When write current is applied the digit drivers will also be turned on. The direction of the digit current will determine which core will be returned to the R state (figure 129). When writing a 0, core B receives inhibiting digit current and core A receives augmenting digit current. When writing a 1, core B receives the augmenting digit current and switches while core A does not.

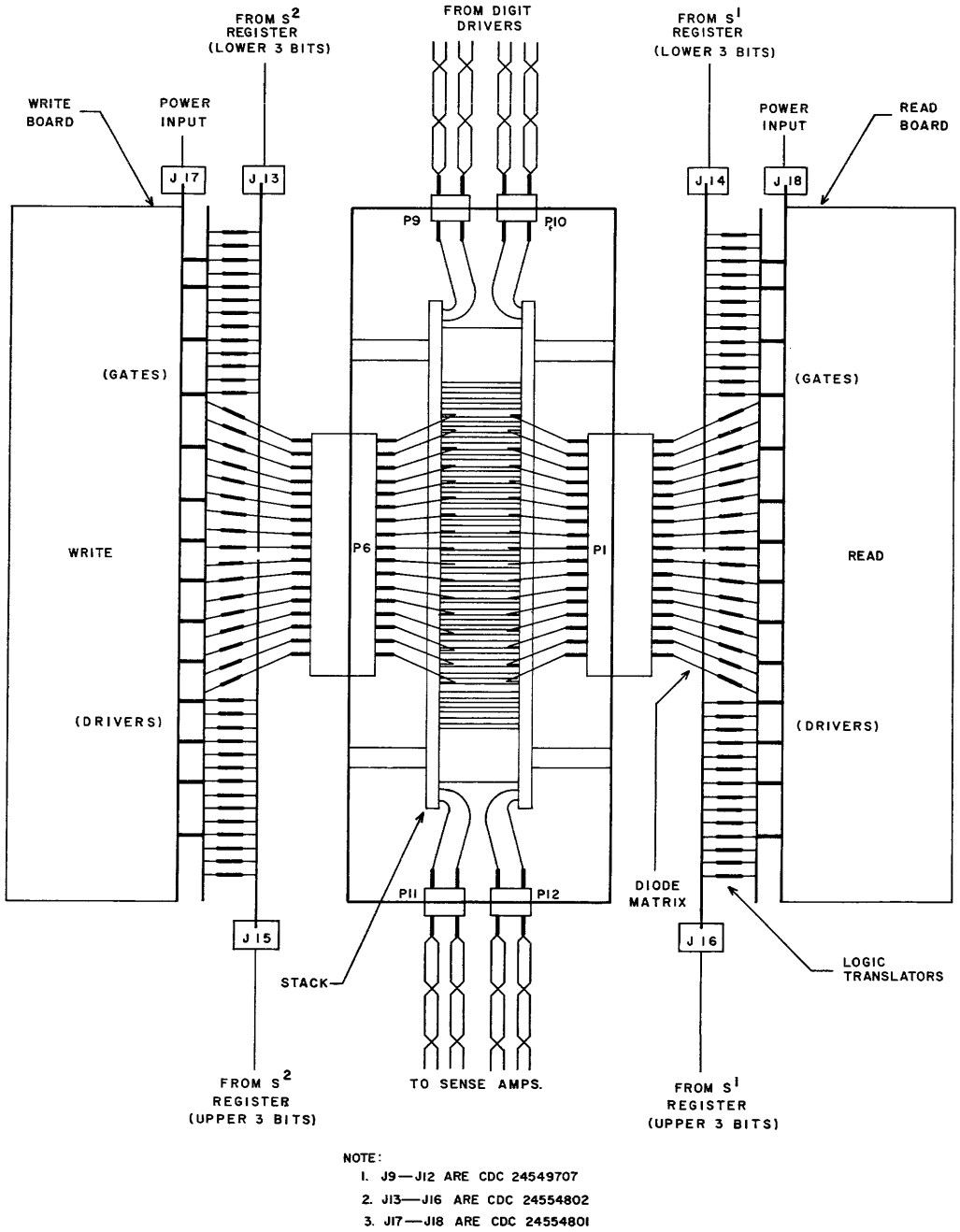


Figure 142. Page Index File Assembly and Designations

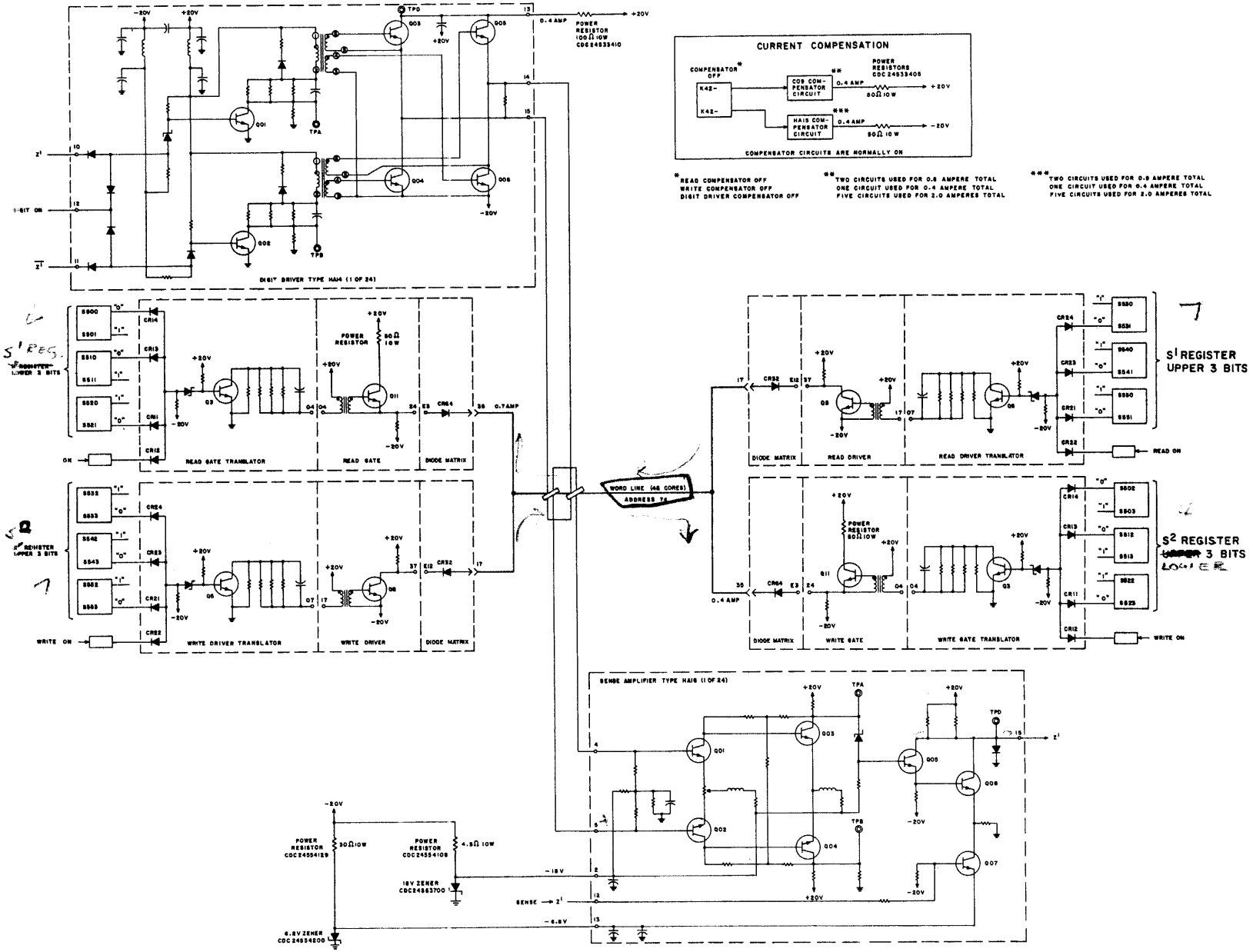
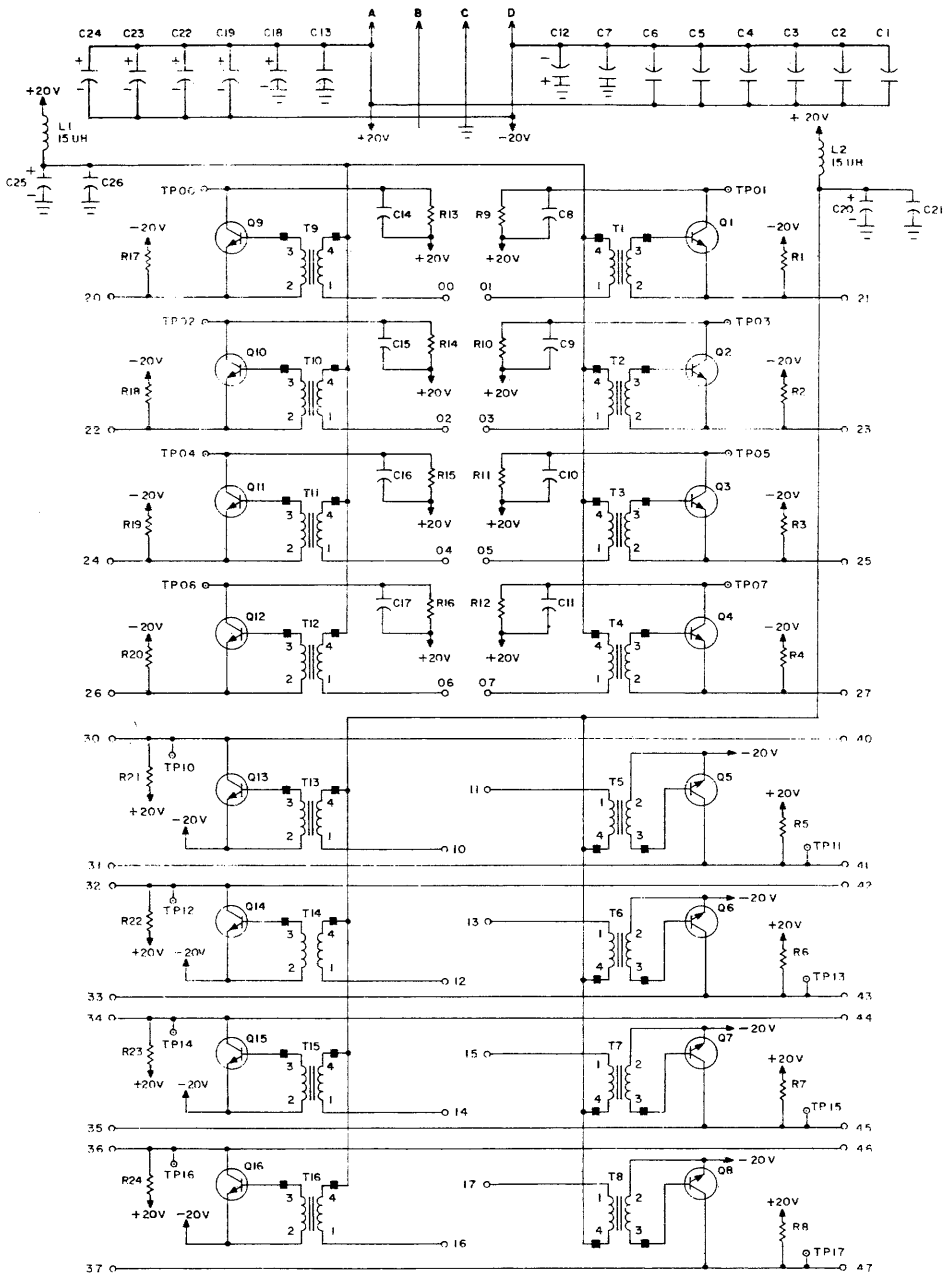


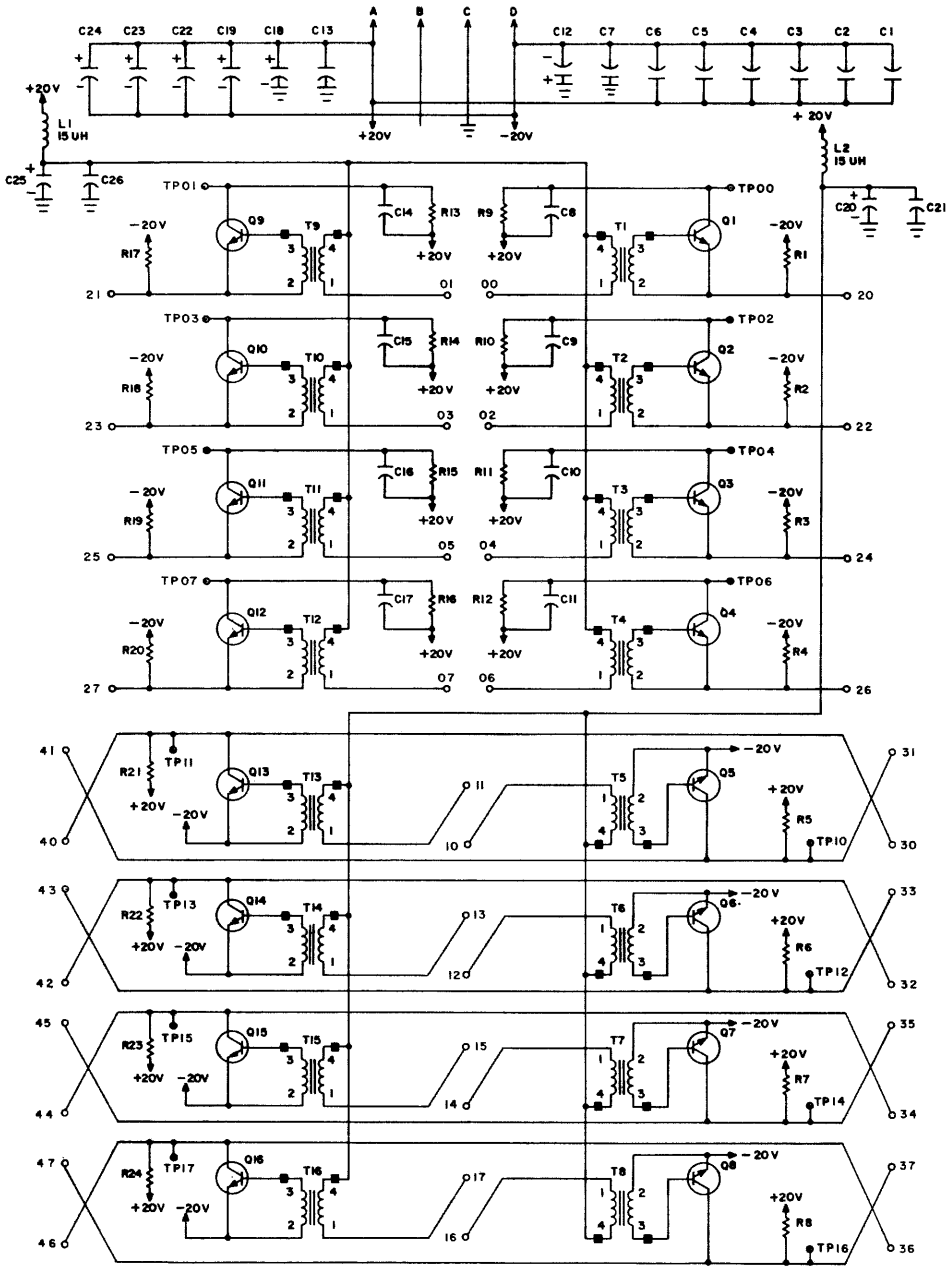
Figure 143. Register File Operation Example



NOTE:

1. R1-R8, R17-R24 ARE 10,000 Ω, 1/4 W.
2. R9-R16 ARE 50 Ω, 10 W.
3. C1-C7, C13, C21, C26 ARE 4000 PF.
4. C8-C11, C14-C17 ARE 250 PF.
5. C12, C18-C20, C22-C25 ARE 8μF.
6. TRANSISTORS ARE CDC 245513.
7. TRANSFORMERS ARE CDC 245518.

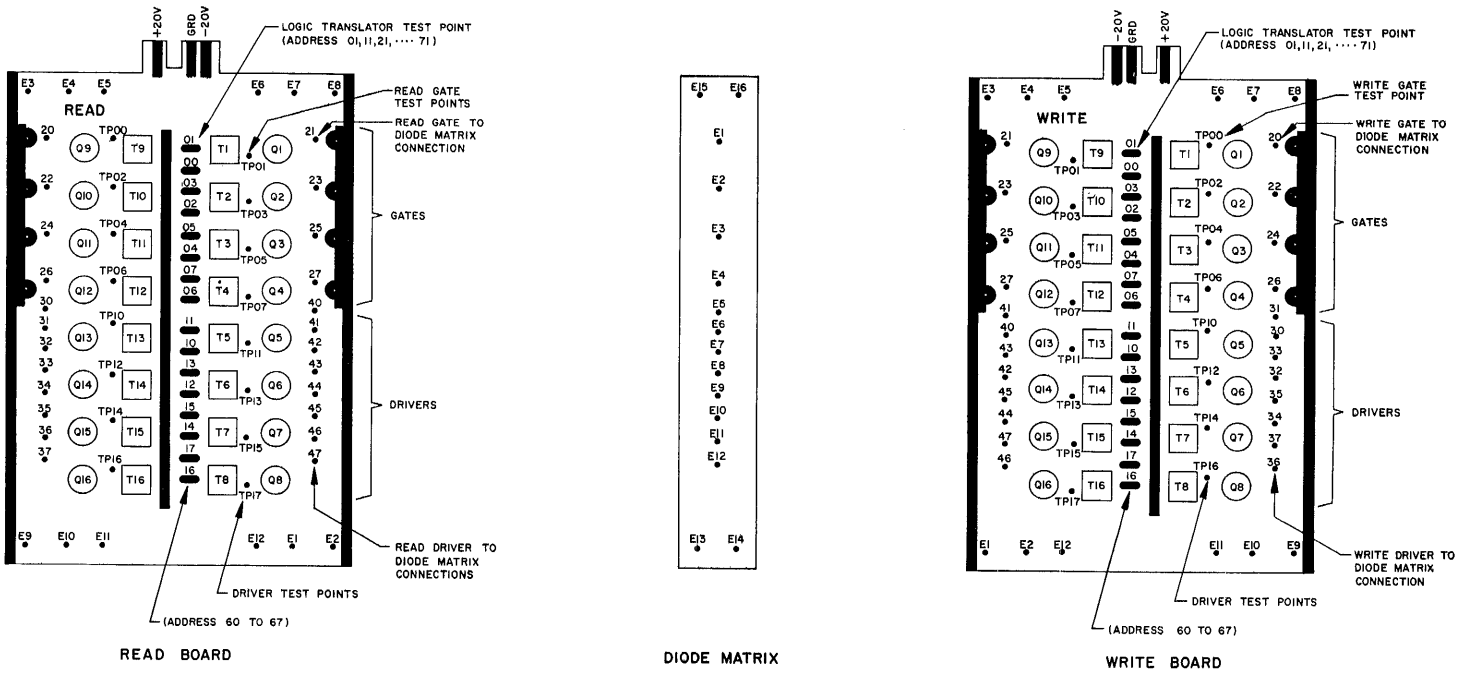
Figure 144. Read Board Schematic -- Type 24419200

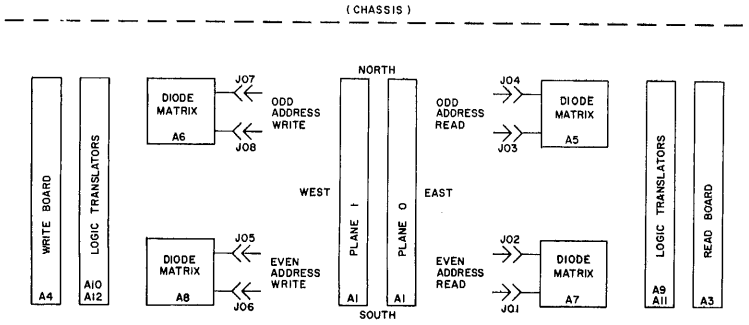


NOTE:

1. R1-R8, R17-R24 ARE 10,000 Ω, 1/4 W.
2. R9-R16 ARE 100 Ω, 10W.
3. C1-C7, C13, C21, C26 ARE 4000 PF.
4. C8-C11, C14-C17 ARE 22 PF.
5. C12, C18-C20, C22-C25 ARE 8μF.
6. TRANSISTORS ARE CDC 245513.
7. TRANSFORMERS ARE CDC 245518.

Figure 145. Write Board Schematic -- Type 24419500





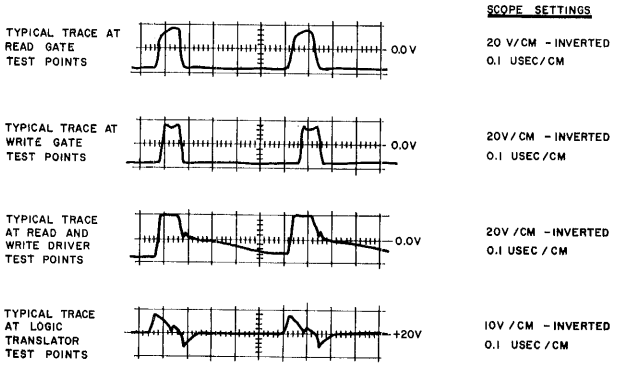


Figure 146. Register File Test Points

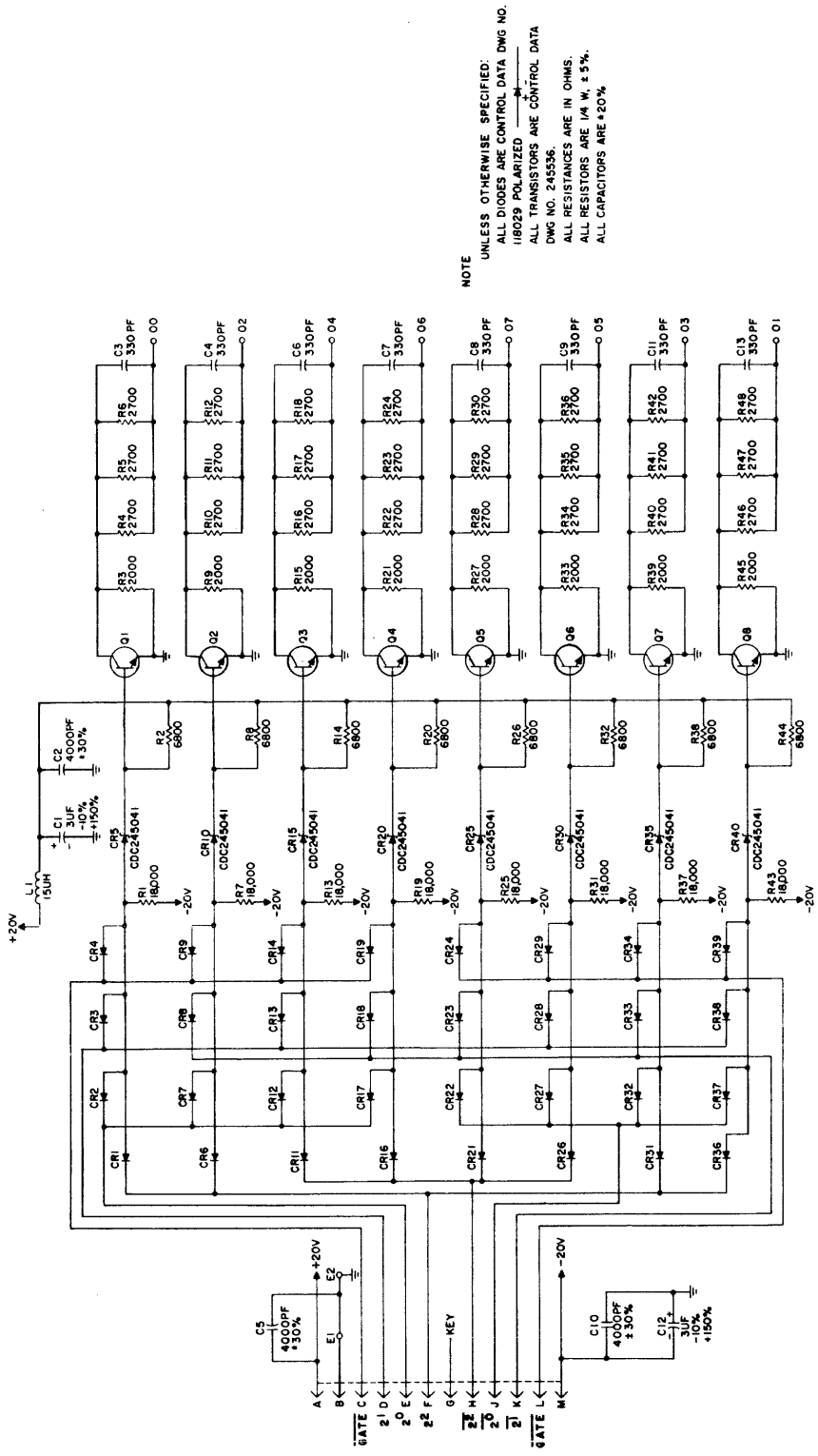
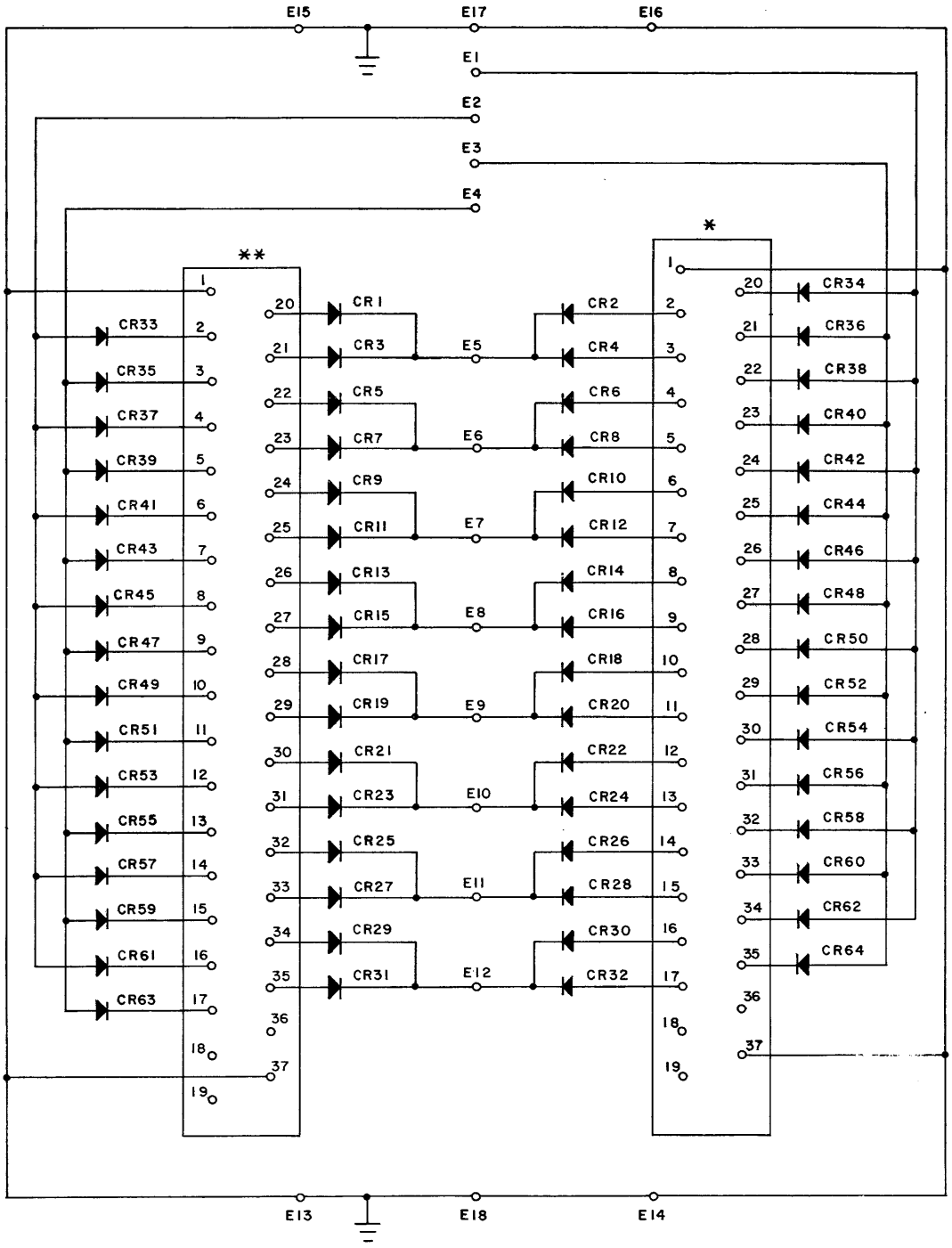


Figure 147. Logic Translator Schematic -- Type 24418700



NOTE: ALL DIODES ARE CDC 245535.

* J01 IF ASSY A7
J03 IF ASSY A5
J05 IF ASSY A8
J07 IF ASSY A6

** J02 IF ASSY A7
J04 IF ASSY A5
J06 IF ASSY A8
J08 IF ASSY A6

Figure 148. Diode Matrix Schematic -- Type 24418900

All operations in the Multiprogramming module are based on reading from or writing into the Page Index File. Timing for read or write page index file is obtained from a tapped delay line. Timing is as follows:

Storage Request--Set K000/001 (Start Delay Line)
Set K014/015 (Digit Dummy), Gate Address to S^1

T25 Set K010/011
Enable Read Drive

T50 Sense Amps to Z if Read
Data Bus lower 12 to Z if Write
Clr K000/001

T100 Clr S^3

T125 S^1 to S^2
Clr K010/K011

T275 Storage S Bus Address to S^3
Set K012/013

T375 Enable Digit Drive
Enable Write Drive

T500 Clr K012/013
Clr K014/015

T800 Clr K004/005

From this timing it is apparent that the only difference between a read or write operation is at T50. Note that for a write only the lower 12 bits of the data bus are sampled and this quantity will be gated to Z_e or Z_o.

Reasons for accessing the 3311 are to set up the Page Index File or to effect relocation. Let's first examine setting up the Page Index File.

The 77.64 instruction writes the contents of A lower 12 (hereafter written as A_{L12}) into the page index specified by the lower seven bits of the instruction. The 77.65 instruction reads the contents of the page index specified by the lower seven bits of the instruction and places that quantity into A_{L12}. Either of these instructions may be indexed by B^2 if bit 11 of the instruction word is set. If the system is in the Non-Executive mode, these instructions become no-op's. If the system is in Program State, attempting to execute these instructions will generate an executive interrupt.

WRITE PAGE INDEX

Figure 149 shows the necessary data and control signal transmissions for a write page index operation. For this operation the desired address is the lower seven bits of the S bus; also, the 3311 has access only to the lower 12 bits of the data bus.

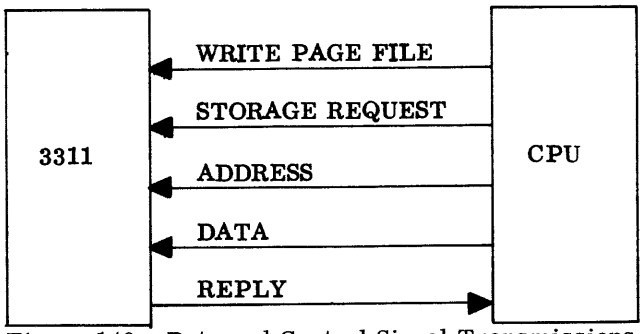


Figure 149. Data and Control Signal Transmissions for Write Page File

READ PAGE INDEX

Figure 150 shows the necessary data and control signal transmissions for a read page index operation. For this operation the desired address is the lower seven bits of the S Bus; also the 3311 has access only to the lower 12 bits of the data bus.

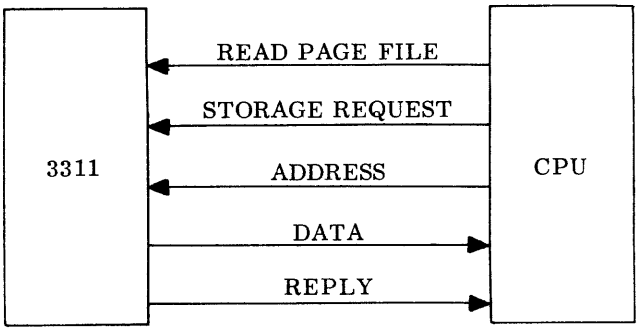


Figure 150. Data and Control Signal Transmissions for Read Page File

Figure 125 is a block diagram of the Page Index File and its control logic. Note that for a read or write page file operation, bits 01-06 of the S bus are gated to S^1 to select a word line. Bit 00 determines which register (Z_e or Z_o) is to be referenced.

Figure 151 shows the address flow if the system is operating in Executive mode. The upper seven bits of the CPU S bus determine which Page Index is referenced. Figure 125 shows that bits 12-17 are gated to S^1 to select a word line while bit 11 determines whether Z_e or Z_o is sampled. In the lower left corner of figure 151 is the Illegal Write Detector. This detector senses for an illegal storage reference during read or write. If the 3311 is being referenced for relocation or to read a page index, a 24-bit word is read and gated into Z; however, only Z_e or Z_o will be sampled.

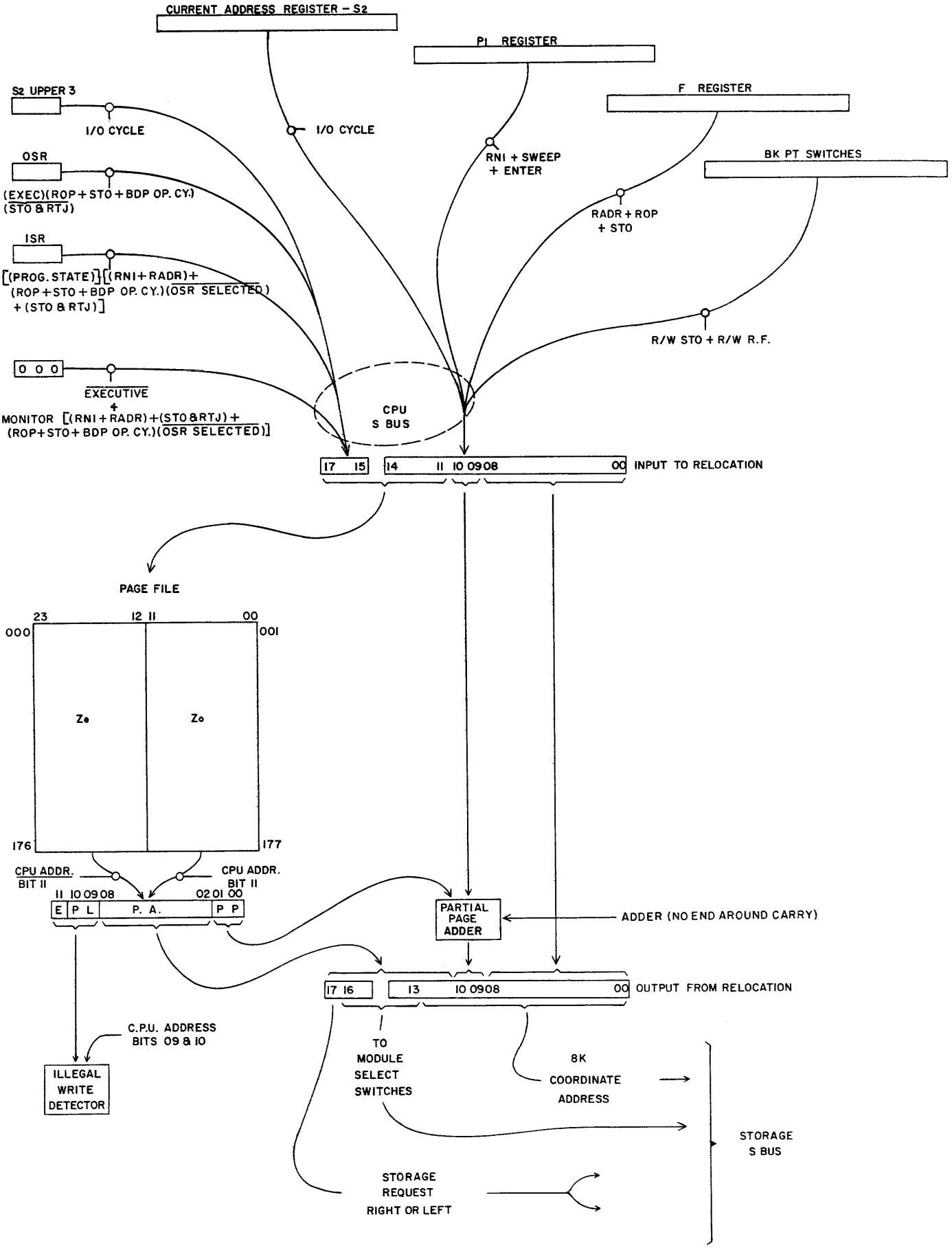


Figure 151. Address Flow for Executive Mode

ILLEGAL STORAGE REFERENCE DETECTION

The 3311 senses illegal storage references when:

1. E = 1, all other bits of the page index equal zeros, and the system is in Program State (see figure 152).
2. E = 1, a write is specified, and the system is in Program State (see figure 153).
3. The CPU address specifies a reference outside the limits set by PL (see figure 154). When the CPU S bus bits 09 and 10 are equal to or greater than PL, an illegal reference occurs (except when PL = DO).

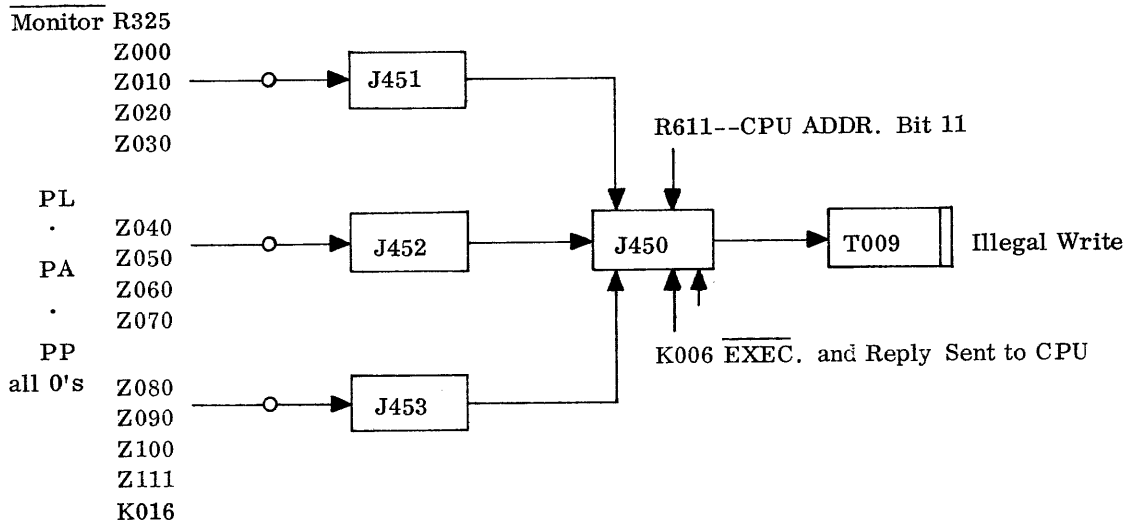


Figure 152. "Z Even" Illegal Write Test

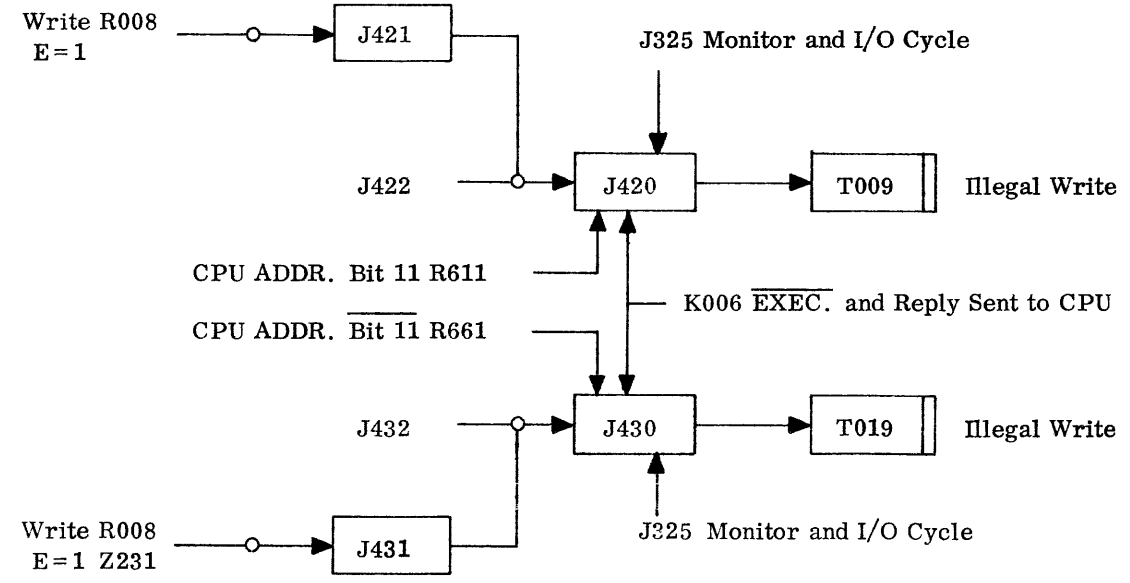


Figure 153. E Bit Illegal Write Test

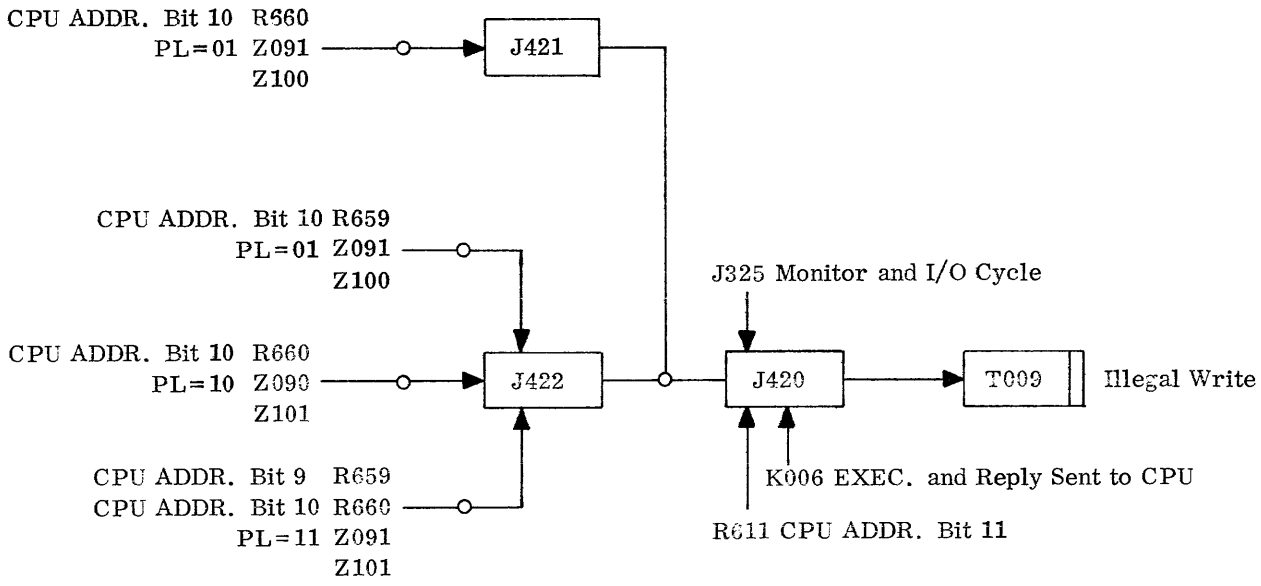


Figure 154. Z Even Page Length Test

READ PAGE INDEX 0

Whenever page index 0 is referenced, PA and PP must be read as zeros. This is to insure that the Executive Auto Load/Auto Dump operations and all interrupt sequences reference page 0. Anything may be written into page index 0 but the logic in figure 155 insures that PA and PP are read as zeros by blocking transfers from the sense amplifiers into bits 12-20 of Z. Note that J300 goes to 1 if page index 0 is referenced for relocation, whereas J303 goes to a 1 if page index 0 is referenced for a 77.65 instruction or for sweep page file.

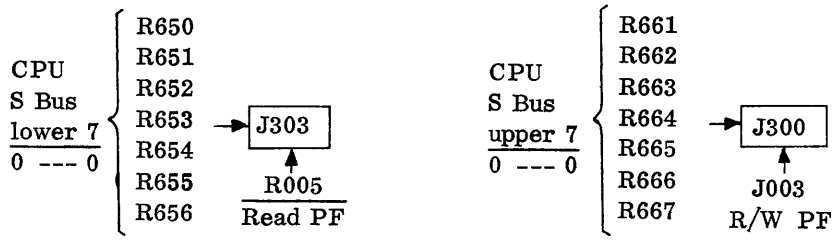


Figure 155. Page Index Zero Test

PARTIAL PAGE ADDER

The Partial Page Adder performs twos-complement addition of PP and bits 09 and 10 of the CPU bus. In twos-complement additive arithmetic end around carries are not sensed. Figure 156 is an addition chart for the Partial Page Adder.

PP bits of the page index define the top of the page; bits 09 and 10 of the CPU S bus specify which quarter is to be referenced relative to the top of the page. The Partial Page Adder consists of six inverters which drive Storage S bus T cards, bits 09 and 10. When examining the Partial Page Adder remember that address is placed on the S bus.

PP
(PAGE START ADDRESS)

Bits 9 & 10 of incoming address from Main Control	00	01	10	11
00	00	01	10	11
01	01	10	11	00
10	10	11	00	01
11	11	00	01	10

Resultant quarter page that will be referenced

Figure 156. Partial Page Adder Addition Chart

STORAGE REQUEST

The 3311 serves as a relay station for the Storage Request signal. If the system is operating in Non-Executive mode, the 3311 receives a Storage Request from the CPU and retransmits the request directly to low core. If the system is operating in Executive mode, the 3311 inspects the highest-order bit of PA and sends a Storage Request signal to: low core if the bit is a 0, to high core if the bit is a 1.

SPECIAL CYCLE

Special cycle occurs during each write to an even-numbered page index. Bit 24 of the preceeding 24-bit index is set on the special cycle if the even PIF location is being loaded with 4000.

When writing 4000 in P+2, special cycle causes P+2 to become decremented by 2 while bit 24 becomes a 1 in that location. If one operand of a double-precision instruction is in the page designated by P+1 and the other operand is located in the page designated by P+2, an Illegal Write condition occurs

24	EVEN	ODD
	P+2 = 4000	P+3
1	P	P+1

Figure 157. Special Cycle Bit 24 Test

during the first ROP or STO sequence of the double-precision instruction.

This special look-ahead bit becomes the 25th bit at each location.

NO CHANGE

Because instructions use sequential addresses, the upper address bits (09-17) may not change for as many as 1000_8 sequential references. If CPU S bus bits 09-17 remain the same, then relocation need not be performed each time, resulting in a time savings.

To accomplish this, two registers (S6 and S7) are used to hold relocated addresses. S6 contains the last relocated address from an RNI or RADR sequence; S7 contains the last relocated address from an ROP or STO sequence. Two registers are needed because operands are usually separated from instructions when stored in memory. S4 contains the upper nine CPU S bux bits from the last RNI or RADR sequence. S5 contains the upper nine CPU S bus bits from the last ROP or STO sequence. S4 or S5 are compared with the incoming S bus bits to detect either a change or a no-change condition. If no change is detected, the contents of S6 or S7 are gated to the Storage S bus.

Use of the no-change function is inhibited for one reference of each sequence type--RNI/RADR or ROP/STO--following a read or write to the Page Index File.

SELF-EVALUATION QUIZ ON CHAPTER 5

FILL IN THE BLANKS AND TRUE OR FALSE (T or F)

1. The 3311 increases the maximum storage capacity of the 3300 from 131K to 262K.
2. Page index 000 may not be written into.
3. The 3311 detects an illegal write if the system is in Executive mode.
4. The 3311 is chassis _____.
5. The 3311 connects to the CPU via flex jumpers.
6. The 3311 relays the Storage Request signal to high and low core.
7. The 3311 is referenced during each storage reference if the system is in Executive mode.
8. The Page Index File segments storage into 2K pages.
9. The Page Index File is a word-organized storage unit.
10. An advantage of all word-organized storage units is that the same number of cores switch for every storage reference.
11. There are 24 cores on each word line.
12. There are _____ cores on each digit drive line.
13. The Page Index File stack consists of a cubic matrix containing _____ x _____ x _____ bits.
14. A current of 2I is necessary to switch a core.
15. Odd parity is used in the 3311.
16. Each word line passes through a given core twice.
17. Two read/write cycles occur each time the Page Index File is referenced.
18. The 3311 samples _____ bits of the data bus.
19. If an Illegal Write is sensed by the 3311, the Write signal is converted to a Read signal.
20. The address of the page index to be referenced is obtained from the lower seven bits of the S bus for either read or write page file.

CHAPTER 6

LOGIC TIMING AND KEYBOARD ENTRY

COMPUTER TIMING

All computer operations must be timed and signals must occur in the proper sequence. Timing is provided by the clock and clock pyramid and by the resynchronizing circuits which are timed by the clock.

MASTER CLOCK AND CLOCK PYRAMID

The master clock operates continuously when power is applied to the computer; it provides the timing pulses used throughout the computer. Timing of all signals is determined directly or indirectly by this clock.

The clock system consists of a master oscillator feeding four ranks of clock amplifiers in an oscillator-amplifier pyramid. The oscillator and amplifiers are contained on the type C01 card. The pyramid connection and circuit operation are discussed in the Printed Circuits Manual, publication no. 60042900.

The oscillator operates at 8 megacycles and provides four sine wave outputs. Two of these are 180 degrees out of phase with the remaining two. One set of outputs is designated even raw clock, the other set is odd raw clock. The raw clock outputs used for timing logic are available only from the clock amplifiers in the fourth rank of the pyramid.

Clock signals are placed on one-way AND gates as well as multiple input AND gates to time the output of control delays and single inverters. These are always designated as HXXX, VXXX, or NXXX terms. When

the raw clock signal goes to a 0 (-3v to +1v), the inverter outputs a logical 1 for 62.5 nsec. The inverter circuits clip the raw clock signal to convert the sine wave to a rectangular wave. All odd numbered NXXX and VXXX terms have an odd raw clock input. The logical 1 from these terms occurs at odd time. Likewise, even numbered NXXX and VXXX terms are fed by even raw clock and output a logical 1 at even time.

The master clock and pyramid must be tuned to a frequency of 8 megacycles. Procedure for tuning the clock pyramid is presented in Printed Circuits Manual.

Ranks 1, 2, 3, and 4 are ranks of amplifiers. In terms of hardware, the circuits are the same for both the amplifiers and the master oscillator. The master oscillator drives two amplifiers and each amplifier may drive four more amplifiers.

RESYNCHRONIZING

External equipment and switches provide signals which are asynchronous to the timing of the computer and must be resynchronized. This involves insuring that only one synchronized pulse results from an asynchronous signal, regardless of the duration of such a signal. In addition, resynchronization prevents the possibility of marginal timing due to the occurrence of runt pulses.

Resync Counter

The resync counter is composed of a free-running chain of four control delays. This chain begins counting when the first odd clock pulse is produced by the clock pyramid. A 62.5 nsec pulse is produced by one of the control delays each phase time (figure 158). Each control delay produces a pulse every quarter microsecond.

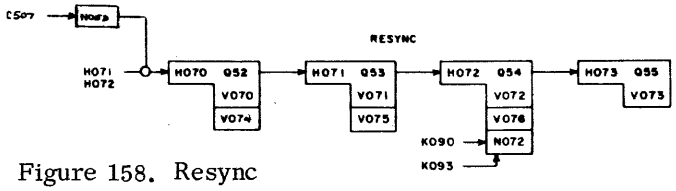


Figure 158. Resync

The first control delay of the timing chain has a three-way ANDed input.

1. N053: Odd clock slave outputs a logical one 62.5 nsec every odd phase time.
2. H071 and H072: These are outputs from the A side of the H07X term. Their output will go to 0 for 2 Ø times after the H07X term receives an input. This is true of all control delays.

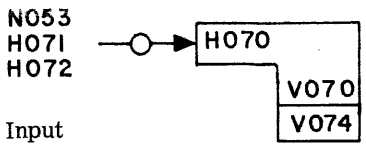


Figure 159. Resync Input

The resync pulses synchronize signals by conditioning AND gates throughout the computer. An asynchronous signal is sampled to form a synchronous (62.5 nsec) 1 when a resync pulse completes the AND gate. A synchronous pulse is not produced when a runt pulse is sampled, for this type of signal does not have sufficient amplitude to definitely indicate 1 or 0. The resync pulse occurs again in 0.25 usec. If the input has settled to a steady 1 by this time, a synchronous pulse is produced.

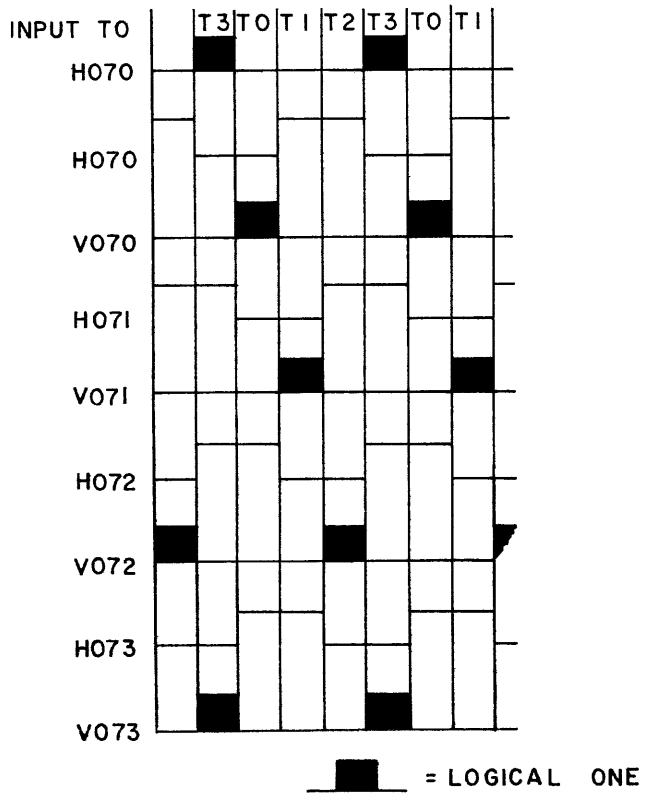


Figure 160. Resync Timing

Start

The RNI sequence provides for most manual and program starts and stops. The computer may be started by pressing GO switch if normal program operation is desired, or by pressing SW/EN CONT switch if a sweep or enter operation is to be performed. These switches are mutually exclusive although they both set Initiate Go FF (K304/305). The output of Initiate Go is timed with a resync pulse to set the Go sync FF. The output of this FF then sets Go (K090/091) and the output of Go allows a start pulse to begin an RNI to read the first instruction of a program. (The logic discussed above is found in Logic Diagrams, page 2-15).

The computer may also be started from halt by pressing CYCLE STEP, INSTRUCTION STEP, or AUTO STEP switches on the console. Operation of the computer in these modes is discussed elsewhere in this manual.

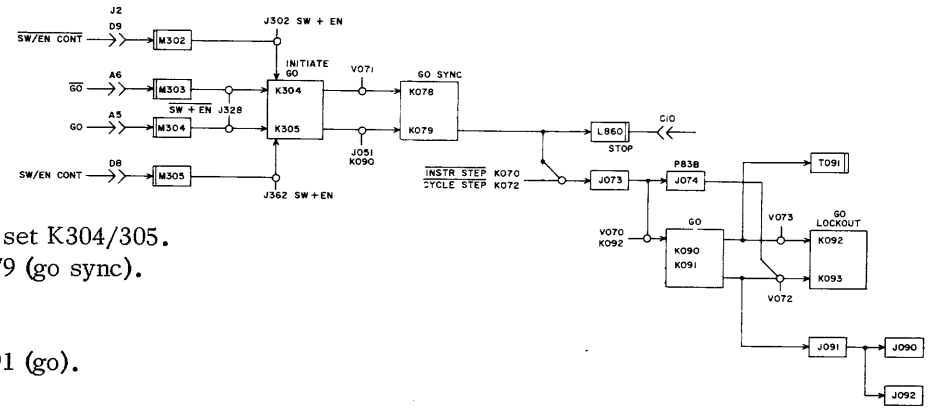


Figure 161. Go Logic

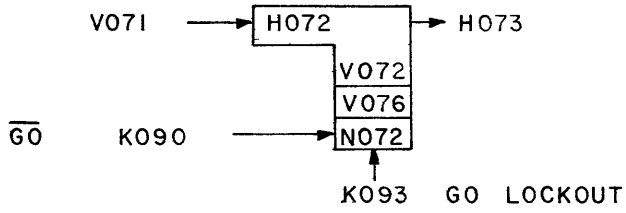
Operator pushes GO, set K304/305.

V071 (t1): Set K078/079 (go sync).
(t2)
(t3)

V070 (t0): Set K090/091 (go).
(t1)

N072 (t2): Outputs a logical 1. This is the start pulse for the processor.

V073 (t3): Set K092/093 (go lockout).



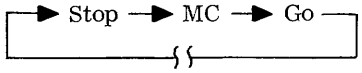
N072 will have only a 1 out at t2 after setting of K090/091 and before setting K092/093.

Figure 162. Time 2 of Resync Timing

TEST MODE

Test mode is available with the 3300 Computer for the convenience of maintenance personnel; it performs a three-step operation consisting of Stop, Master Clear, and Go.

The repetition rate of this three-step operation may be switch-selected at a 1 msec. rate or at an auto-step rate. TEST MODE switches are at location 1T23 in the main frame. The C section of the switch card controls the repetition rate; the D section enables Test mode.



Turn on Test Mode
M338 and M339 = 1. Clear Go Initiate FF (K304/305) and Stop FF (K306/307).

Real Time clock (J793 = 1) or Auto Step (Y859 = 1)
- Set K386/387
  - M329 = 1. Set Stop FF (K306/307).
  - M339 = 0. Remove clear from Stop FF.
  - after 2 msec duration:
  - M337 = 0, M309 = 1 (Master Clear)
- After 5 msec duration:
- Set K388/389
  - M328 = 1. Set Fo Initiate FF (K304/305).
  - M339 = 1. Clear Stop FF (K306/307).
  - M338 = 0. Remove clear input from Go Initiate FF (K304/305).
  - M329 = 0. Remove output of M309 from input to J280, dropping Master Clear.
- Next V071, Set Go Sync FF (K078/079).
- Next V070, Set Go FF (K090/091).
- Continue in ≈ Go condition for time determined by Real Time Clock or after 5 msec duration:
- Clear K386/387, K388/389
  - M328 = 0. Remove set input to Go Initiate FF (K304/305).
  - M338 = 1. Clear Go Initiate FF (K304/305).

STATIC CONTROLS

The static controls are used to enter data from the keyboard. These include the keyboard entry controls and the breakpoint controls.

KEYBOARD ENTRY CONTROLS

The keyboard entry controls are used to control (1) entry of data into C register from the keyboard, and (2) transfer of this data to the various registers.

COMMUNICATIONS REGISTER CONTROLS

Entry of data from the keyboard to C register is controlled by C register controls which include the digit counter, the sequence counter, the digit sequence chain, and various timing networks.

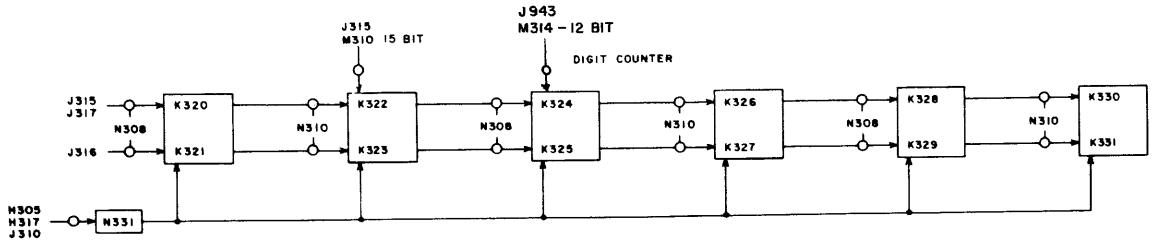


Figure 163. Digit Counter

The digit counter is a chain of six flip-flops which controls the entry of successive digits into C. The counter operates by moving a combination of setting and clearing inputs down the chain.

The various combinations are translated by the J320 to J327 translators. A translator outputs a 0 when its inputs are satisfied. This allows a corresponding J340 to J347 term to output a 1 to turn on the proper light driver card to light the blue background indicator in the desk console display. This background indicator shows the digit position to be filled next.

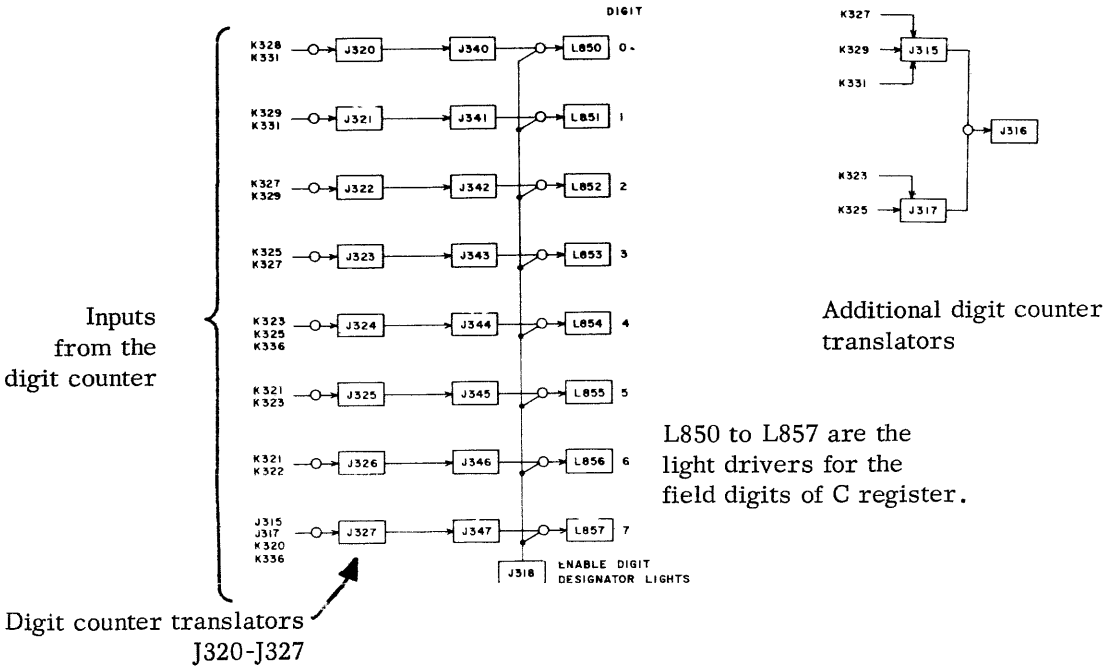


Figure 164. C Register Enables

The flip-flop combinations necessary to select each digit position are shown below.

Table 13. DIGIT POSITION SELECTION

Digit Position	Flip-flops in Set State
7 (First to be filled for a 24-bit entry)	All flip-flops clear
6	First flip-flop in the chain, K320/321
5	First and second
4 (First to be filled for a 15-bit entry)	Second and third
3 (First to be filled for a 12-bit entry)	Third and fourth
2	Fourth and fifth
1	Fifth and sixth
0	Sixth

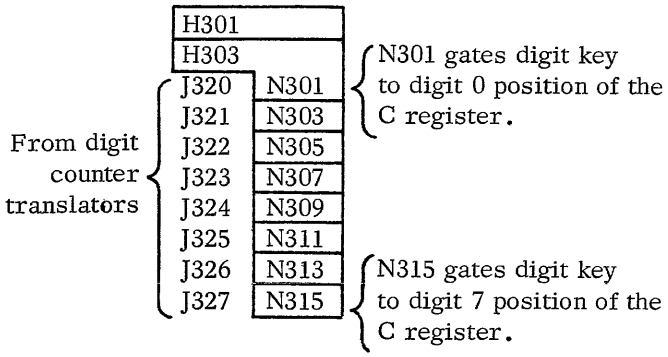


Figure 165. C Register Digit Enables

Depending on the state of the digit counter, one of the J32X (digit counter translators) terms produces a logical 0. This output is used to permit gating the digit from the pressed digit key into C register.

Removing the blocking input to the N3XX gate which then comes up and gates the digit position in C register.

The digit sequence is a chain of three flip-flops and a control delay.

The control delay, H301, H303 is drawn complete in figure 166.

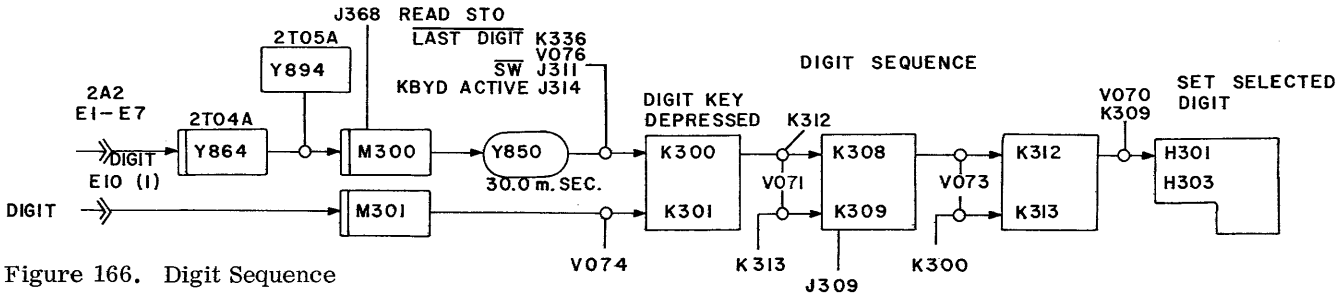


Figure 166. Digit Sequence

When a digit key is pressed, K300/301 (digit key depressed FF) is set. The signal is advanced down the chain to pulse H301 which then clocks out a pulse to set the selected digit in the digit position indicated by the digit counter and as translated by the digit counter translators.

The sequence counter is a pair of flip-flops which indicate whether an odd or even number of digits has been entered into C. When the digit key is pressed, the first flip-flop is set or cleared depending on the state of the second flip-flop. After the first flip-flop has stabilized, its state is copied by the second flip-flop. Both flip-flops are set to indicate an odd number of digits entered.

N308 and N310 translate the state of the sequence counter FFS. After each digit is entered in C, a pulse is clocked from one or the other of these inverters to advance the digit counter.

When the last digit has been loaded into C the last digit FF is set. This prevents the end-around entry of C by breaking the input to the digit key pressed FF. The set output of last digit is gated to control delay H305 which clears the digit counter. Last digit is cleared

Selection of a 15-bit register may be overridden by selection of a 24-bit register. K354/355 is set when a 15-bit register is selected. If a 24-bit register selection is then made before C has been fully loaded, K356/357 sets to complete an AND gate to H305. H305 then clears the keyboard controls. Clearing of keyboard controls will reset the digit counter. This will reset the digit designator field light.

when the contents of C are transferred to the selected register. It is also cleared by a keyboard clear or a master clear. When last digit flip-flop has set, the pressing of any digit key or keys will not affect the contents of C register and all digit designator field lights will be out.

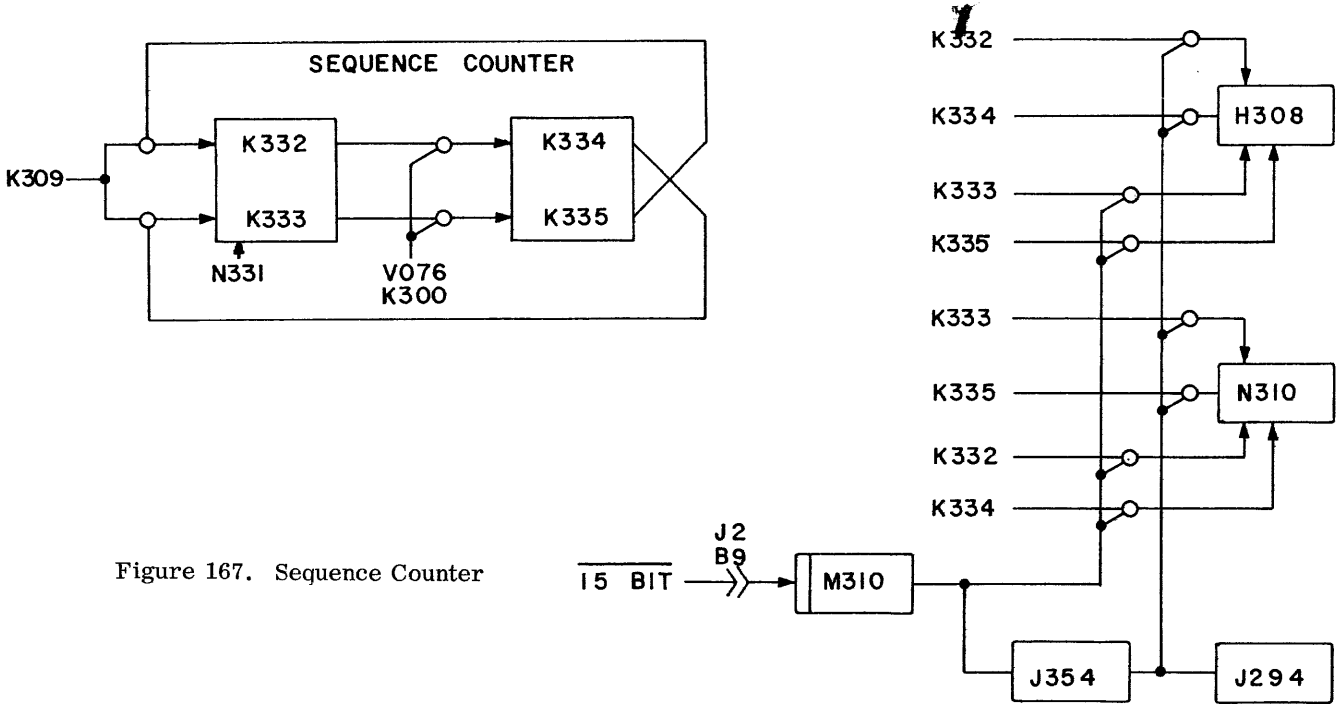


Figure 167. Sequence Counter

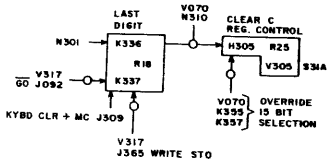


Figure 168. Last Digit Logic

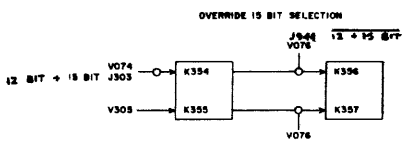
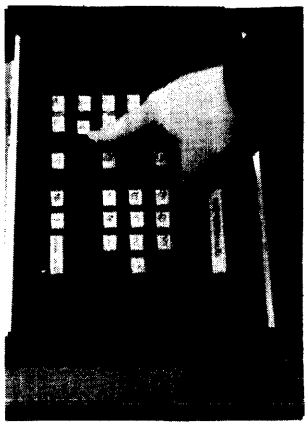


Figure 169. 12 or 15 Bit Override Logic

Register Selection

Data may be entered from the keyboard into P, B1, B2, B3, A, and Q registers via the communications register.

A register is selected by pressing one of the register switches on the keyboard. Selecting a register enables the keyboard active signals, the digit indicators (blue background lights), and the console display of C register.



In this example manual entry is being made into A register.

Figure 170. Manual Entry into "A"

Selecting A register enabled the indicator for digit 7 of C register.

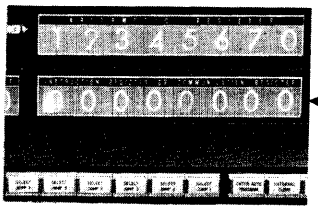


Figure 171. C Register Display for Digit 7

Except for P register, a register may be selected only when the machine is stopped. P register may be selected when the machine is running or stopped and will be displayed in both cases. When the computer is running and P has not been selected, all keyboard switches but STOP, READ STO, WRITE STO, and KYBD CLEAR are disabled.

Consider manual entry into A register as an example of register entry. This will be a 24-bit entry, as A is a 24-bit register. (B1, B2, B3 and P are 15-bit entries.)

Entry into A register will be discussed in two parts.

1. Digits from digit keys to C register
2. Transfer of C register to selected register (A register in example)

Entry of Data into C

Entry of data into C is performed using keyboard switches and C register controls.

Manual entry timing is as follows:

1. Computer must be stopped in order to perform manual register entry.
2. Press the keyboard clear switch to clear K366/337 (last digit FF), the digit counter FFs, K332/333 (sequence counter 1 FF), and C register.
   Note: N310 = 1.

Figure 172. Keyboard

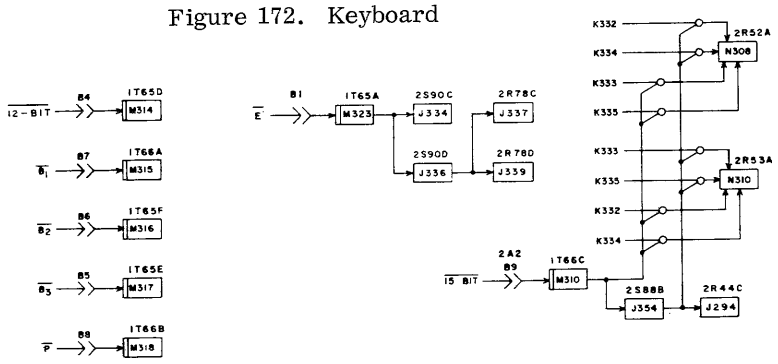


Figure 173. Register Selection Logic

A keyboard clear is performed to initialize the C register controls.

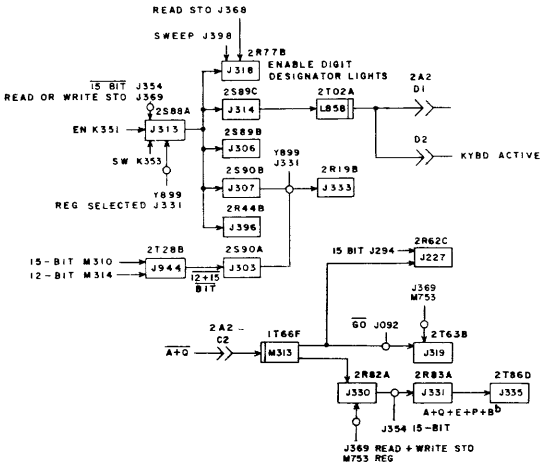
The A, Q, E_U, or E_L register (24-bit registers) switch on the console keyboard is pressed to select the register and activate the keyboard.

The A register is selected for the example.

Since a 24-bit register has been selected, all digit counter FFs are in the clear state. The digit counter translation drops J327 to a 0, allowing J347 to output a 1 to the L857 light driver. This light driver lights the blue background light in digit position seven of the C register console display.

Selecting a register causes a keyboard active indication. This is required for the following timing to take place.

Note that selecting a register will not cause a keyboard active condition if the computer is running. The AND gate into J313 will be broken as J090 = 0 when the computer is running.



The following timing chart shows the entry of a single digit into C. Format will be that of the command timing charts. This will aid in understanding the commanding timing charts for the instruction set of 3300 Computer. (Refer to 3300 CE Diagrams, page 2-21.)

TIMING FOR 24-BIT ENTRY

TIME	TERM	COMMAND	CONDITION	REMARKS
Async*		Start		One of the digit keys (0-7) is pressed.
	Y850			Provides 30 ms delay.
V076(t2)	K330/301	Set digit key pressed FF	$\overline{\text{(Sweep)}}$ $\overline{\text{(Last Digit)}}$	
(t3)				
(t0)				
V071(t1)	K308/309	Set digit sequence 1		When K309 = 1, set K332/333 (first FF of the sequence counter).
(t2)				N308 = 0, N310 = 0; disables advance of digit counter.
V073(t3)	K312/313	Set digit sequence 2		
V070(t0)	H301	Set selected digit		

TIMING FOR 24-BIT ENTRY (Cont)

TIME	TERM	COMMAND	CONDITION	REMARKS
V071(t1) (Odd time)	K308/309			Locks out double pulses. N315 outputs a 1 at odd time to set the selected digit in position 7, (flip-flops C930/931, C920/921, and C910/911, or digit 5, 3, or 1 if not first pass).
Async				Operator releases digit switch.
V074(t0) (t1)	K300/301	Clear digit key pressed FF		
V076(t2)	K334/335	Set second FF of sequence counter		When this flip-flop sets, the translation of the sequence counter is such that N308 outputs a 1. All digit counter FFs are clear so J315 and J317 also = 1. These terms are ANDed to set K320/321, the first flip-flop of the digit counter. Digit counter translation drops J326 to a 0, allowing J346 to output a 1 to the L856 light driver. This moves the indicator from digit seven position to the digit six position (or 5 to 4 or 3 to 2 or 1 to 0).
V073(t3)	K312/313	Clear K312/313		Next digit may now be entered.
Async	Y850			One of the digit switches (1-7) is pressed. Provides 30 ms delay.
V076(t2) (t3) (t0)	K300/301	Set digit key press FF	$\overline{\text{(Sweep)}}$ $\overline{\text{(Last digit)}}$	
V071(t1) (t2)	K308/309	Set digit sequence 1		When K309 = 1, clear K332/333 (first flip-flop of the sequence counter); N308 = 0, N310 = 0.
V073(t3)	K312/313	Set digit sequence 2		
V070(t0)	H301	Set selected digit		
V071(t1)	K308/309	Clear digit sequence 1		Locks out double pulses. N313 (for digit 6) outputs a 1 at odd time to set the selected digit in position 6 or position 4, 2, or 0 if not first pass. When the last digit is entered (digit 0) K336/337 (last digit FF) is set. This prevents the end-around entry into C register.
Async				Operator releases digit key.
V074(t0) (t1)	K300/301	Clear digit key pressed FF		
V076(t2)	K334/335	Clear second flip-flop of sequence counter		When this flip-flop clears, translation of the sequence counter is such that N310 outputs a 1. Advnace digit counter.
V073(t3)	K312/313	Clear digit sequence 2		Return to * for next odd digit if K336/337 is clear.

TIMING FOR 24-BIT ENTRY (Cont)

TIME	TERM	COMMAND	CONDITION	REMARKS
V070(t0)	H305	Clear C register		Input occurs if K336/337 (last digit) is set. The indicator in C register will have moved right one digit position and disappear after entering digit 0 position.
N331		Clears digit counter and sequence counter		

At this time C register will hold the data to be placed in A register. If the operator made an error, KYBD CLEAR will clear C register and controls. The logic will be initialized to the same state as when A register was first selected.

If the data in C register is correct, the next operation required is transferring (C) to the selected register (A in this case). Figure 174 is the waveform timing for 24-bit entry. This could be the first part of manually enter A or Q. This will be the same also for enter or write storage which will be discussed later.

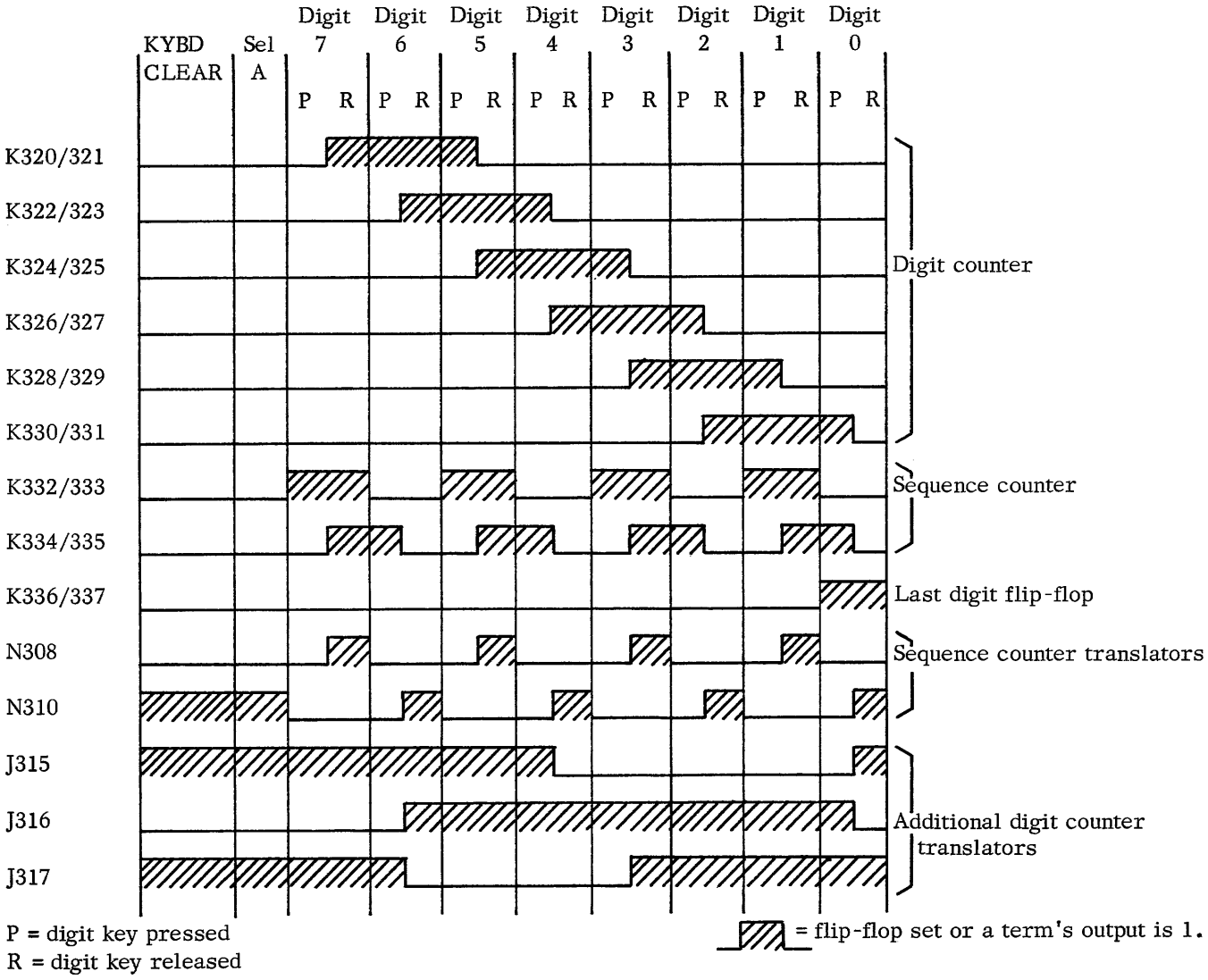


Figure 174. Waveform Timing for 24-bit Entry

Entry of a 15-bit quantity into C is similar to 24-bit entry. The major difference is that the digit counter is set to enter the first digit in digit position 4. This is done by setting K322/323 and K324/325 when a 15-bit register is selected.

1. The computer must be stopped in order to perform manual entry into the registers.
2. Press KYBD CLEAR switch to clear K336/337 (last digit), the digit counter FFs, K332/333 (sequence counter 1), and C register. Note that N310 = 1.
3. Select P, B1, B2, or B3 at the keyboard. Clear K334/335 (sequence counter 2) at 12. Set K332/323 (digit counter FF); N308 = 1. Set K324/325 (digit counter FF) and set K354/355 (15-bit selection record).

TIMING FOR 15-BIT ENTRY INTO B1, B2, B3, or P*

TIME	TERM	COMMAND	CONDITION	REMARKS
**Async				One of the digit keys (0-7) is pressed.
	Y850			Provides 30 ms delay.
V076(t2)	K300/301	Set digit key press FF	$\overline{\text{(Sweep)}}$ $\overline{\text{(Last digit)}}$	
(t3)				
(t0)				
V071(t1)	K308/309	Set digit sequence 1		When K309 = 1, set K332/333 (first flip-flop of the sequence counter).
(t2)				N308 = 0, N310 = 0, disables advance of digit counter.
V073(t3)	K312/313	Set digit sequence 2		
V070(t0)	H301	Set selected digit		
V071(t1)	K308/309	Clear K308/309		Locks out double pulses. N30X term gates M33X terms to digit position 4, 2, or 0 in C register.
				Set K336/337 (last digit) if entering digit 0.
Async				Operator releases digit key.
V074(t0)	K300/301	Clear digit key pressed		
(t1)				
V076(t2)	K334/335	Set second flip-flop of sequence counter		N310 = 1, advance digit counter
V073(t3)	K312/313	Clear digit sequence 2		
				The next digit may now be entered if K336/337 (last digit) not set.
V070(t0)	H305	Clear C register control		Input occurs if K336/337 (last digit) set.
V305				Clear digit counter and sequence counter.
Async				Operator presses digit key.
	Y850			Provides 30 ms delay.
V076(t2)	K300/301	Set digit key Press FF	$\overline{\text{(Sweep)}}$ $\overline{\text{(Last digit)}}$	
(t3)				
(t0)				
V071(t1)	K308/309	Set digit sequence 1		When K309 = 1, clear K332/333 (first flip-flop of the sequence counter).
(t2)				N308 = 0, N310 = 0.
V073(t3)	K312/313	Set digit sequence 2		
V070(t0)	H301	Set selected digit		

TIMING FOR 15-BIT ENTRY INTO B1, B2, B3, or P* (Cont)

TIME	TERM	COMMAND	CONDITION	REMARKS
V071(t1)	K308/309	Clear digit sequence 1		Locks out double pulses. N30X term gates M33X terms to digit position 3 or 1 in C register.
Async				Operator releases digit key.
V074(t0)	K300/301	Clear digit key pressed		
(t1)				
V076(t2)	K334/335	Clear second flip-flop of sequence counter		When this flip-flop clears, the translation of the sequence counter is such that N308 outputs a 1 advance digit counter. The indicator in the C register will have moved right one digit position. Return to ** for next digit.

MANUAL ENTRY TIMING Worksheet: 15-Bit Register

In the following chart indicate state of flip-flops and output of the terms listed as five digit keys are pressed sequentially.

			Digit 4		Digit 3		Digit 2		Digit 1		Digit 0	
	MC	Sel P	P	R	P	R	P	R	P	R	P	R
K320/321												
K322/323												
K324/325												
K326/327												
K330/331												
K332/333												
K334/335												
K336/337												
N308												
N310												
J315												
J316												
J317												

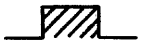
P = Digit key pressed R = Digit key released  = flip-flop set or a term's output is 1.

Figure 175. Waveform Timing for 15-bit Entry

Transfer Sequence

After data has been entered into C, it must be transferred to the selected register. Pressing TRANSFER initiates the transfer sequence. This sequence is performed using the manual timing chain and the keyboard controls (Logic Diagrams, page 2-19).

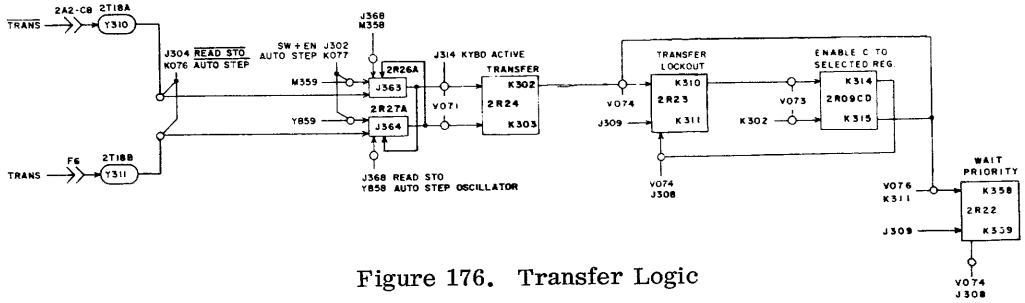


Figure 176. Transfer Logic

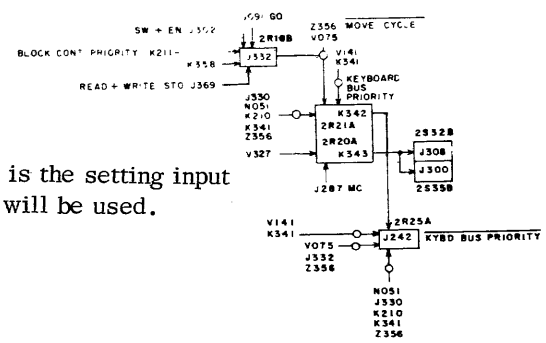
For transfer to A or Q; press TRANSFER to set transfer FF. The signal then passes down the chain of flip-flops to set keyboard bus priority.

V071 (t1): Set K302/303 (transfer).
(t2)
(t3)
V074 (t0): Set K310/311 (transfer lockout).
(t1)
V076 (t2): Set K358/359 (wait priority).
V073 (t3): Set K314/315 (enable C register).

The processor must be stopped for manual entry into a register. Main control will not be using the bus system at the time but the block control could be using it. An input or output operation could be taking place even with the processor stopped. Since DB register will be used during the transfer sequence, it will be necessary to request the bus system and wait until priority is granted.

Priority granted is indicated by setting of K342/343 (keyboard bus priority).

(t0)
V075 (t1) If K210/211 (block control bus priority) is set, test again at resync t1. If K210/211 is clear, set K342/343 (keyboard bus priority) and continue. When K342/343 is set, gate (C) to the C7X2 inverters. Force the EXX2 inverters to 1's.



J332 is the setting input that will be used.

Figure 177. Keyboard Bus Priority

J332 = $\overline{(go)}\,\overline{(SW+EN)}\,\overline{(block\ control\ priority)}\,\overline{(wait\ priority)}\,\overline{(read + write\ storage)}$.

The flow for manual entry to A register is: Digit keys to $\overline{C}$ to C7X2 to $\overline{DBR}$ to I^4 to X to adder to I^0 to A register. Trace this path on the block diagram.

Setting of K342/343 (keyboard bus priority) drives

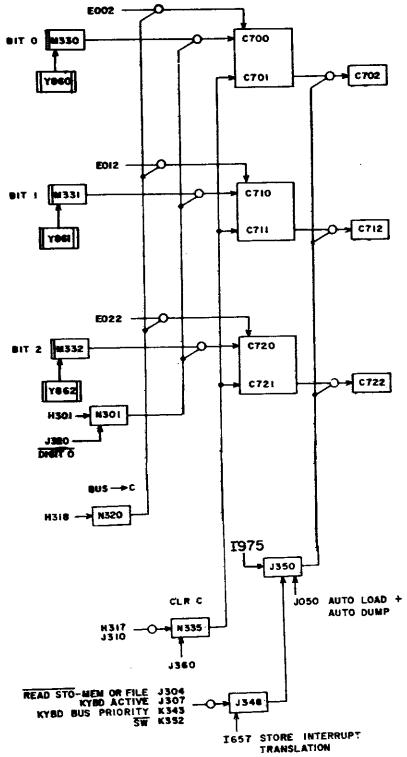


Figure 178. C Register Enables

J348 to a logical 0. J350 will output a logical 1 enabling the clear side of C register ($\overline{C}$) to the C7X2 inverter rank.

The J350 term's output of a logical 1 will disable the inputs to the EXX2 inverter rank, the output of which will be logical 1's. N440 will be one of the gating terms which gate C7X2 inverters to DB register.

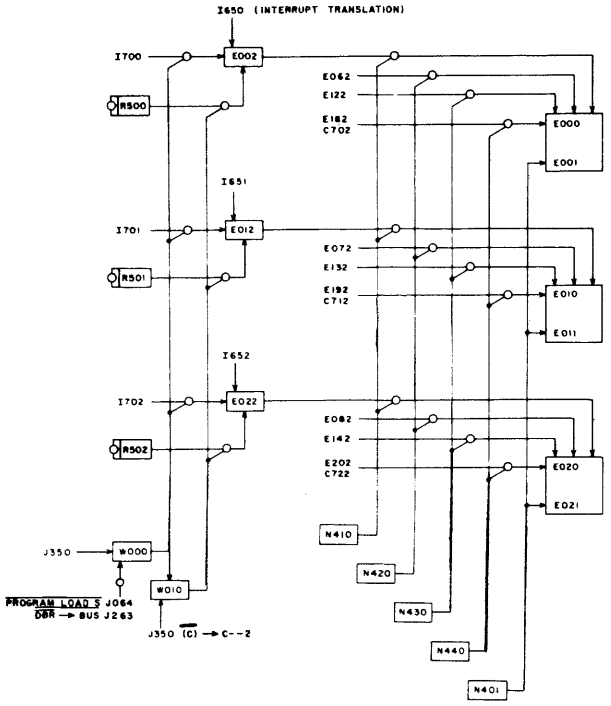


Figure 179. DBR Enables

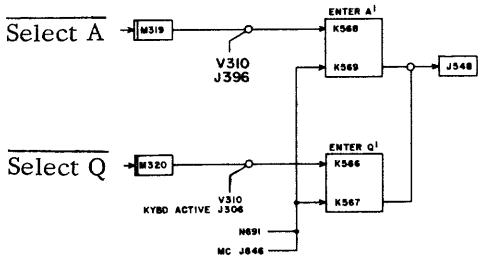


Figure 180. AQ Entry Logic

(t2)
V073 (t3): Input to H310.

V310: Clear K310/311 (transfer lockout) and K358/359 (wait priority); set K568/569 (enter A) or K566/567 (enter Q) and K570/571 (F1 to F2).

Setting of enter A FF or enter Q FF will force a 20 or 21 into arithmetic register F2. The arithmetic section will simulate a load A or load Q instruction.

MANUAL TIMING CHAIN

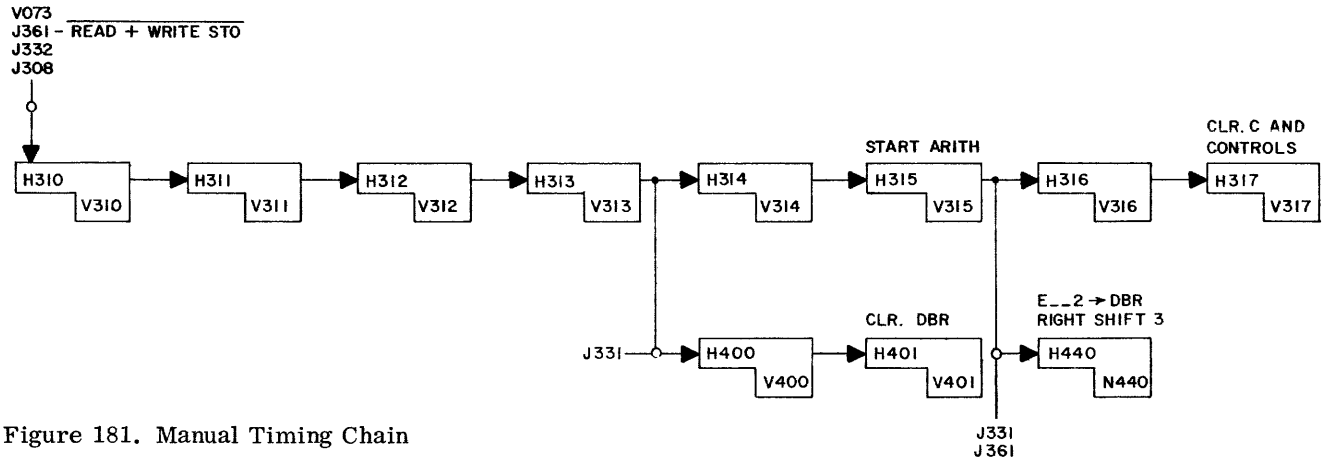


Figure 181. Manual Timing Chain

V311
V312
V313: Input H400, Input H126.
V314: Input H401 (clear DBR)
V126: Set K104/105 (start arith 2)
V315: Clear DBR, Input H440.
V316: EXX2 · C7X2 in DBR. The arithmetic section will execute a load A or load Q sequence, DBR to X to A or DBR to X to Q. (Arithmetic timing is discussed in chapter 12.)

V317: Clear K342/343 (keyboard bus priority), clear C register and its associated controls, and clear K336/337 (last digit).

Async: Operator releases TRANSFER.

V071 (t1): Clear K302/303 (transfer).
(t2)
V073 (t3): Clear K314/315 (enable C to selected register.

If the register selected is B1, B2, or B3 the timing for the transfer sequence will be the same to V310 time.

The following timing is for manually entering a B (index) register during V310 through V317 time.

Operator presses TRANSFER. Transfer timing is the same as for A register.

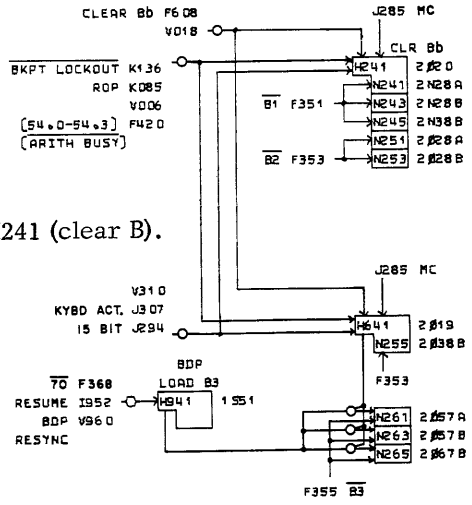


Figure 182. B Register Entry Logic

V310: Input H241 (clear B).

One of the terms F351, F353, or F355 will have a 0 for its output. This is caused by pressing the respective B1, B2, or B3 button on the keyboard when selecting the register.

V311: Clear B (index register).
V312: Input H245 (C to B^b).
V313: C7X2 to B^b. Static enable for $\overline{C}$ to C7X2.
V314
V315
V316
V317: Same as for enter A register.

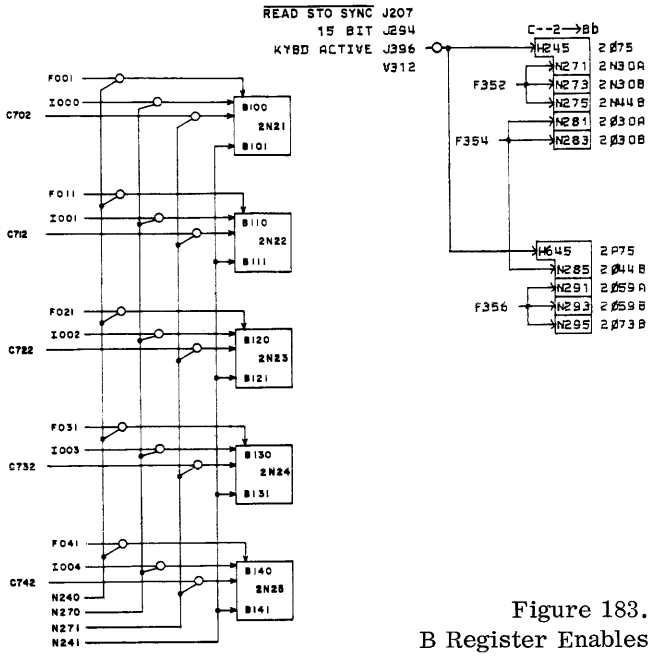


Figure 183.
B Register Enables

If the register selected is P, timing for the transfer will be the same as manually enter A to V310 time. The following timing is for manually enter P register from V310 to V317 time.

V310
V311
V312: Input H215 (clear P1).
V313: Clear P1, input H210 (complement to P1).
V314: C7X2 to P1 (static enable for $\overline{C}$ to C7X2). Input H221 (P1 to P2).
V315: P1 to P2 is a forced transfer to equalize a two-rank register.
V316
V317: Clear K342/343 (keyboard bus priority), clear C register and its associated controls, and clear K336/337 (last digit).

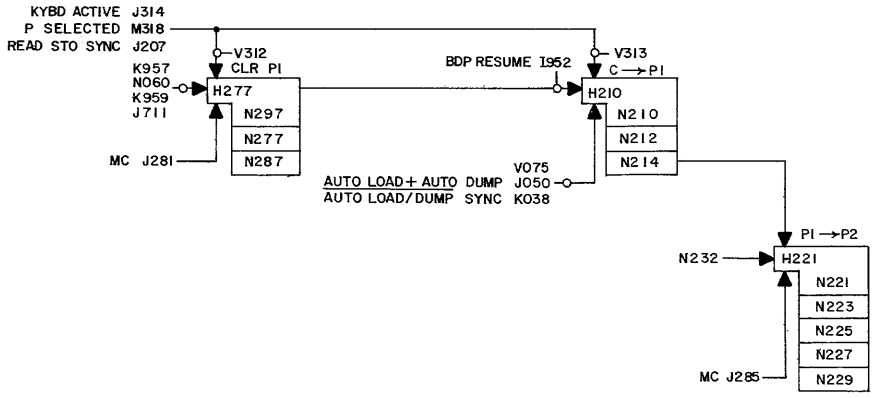


Figure 184. P Register Entry Logic

Figure 184

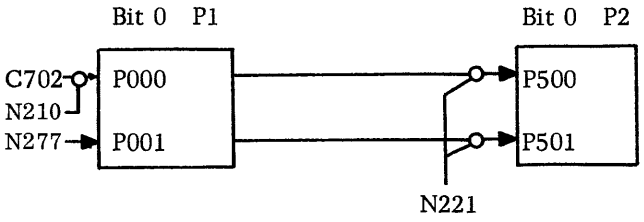


Figure 185. P Register Enables

Review of Transfer Sequence

After data has been entered into C, it must be transferred to the selected register. Pressing TRANSFER initiates the transfer sequence. This sequence is performed using the manual timing chain and the keyboard controls (Logic Diagrams, page 2-19).

For transfer to B1, B2, B3, or P, press TRANSFER switch to set transfer FF. The signal then passes down the flip-flop chain to set keyboard bus priority. H310 is pulsed to start the manual timing chain. As the signal passes down the timing chain: DB register is cleared. B^b (if selected) is cleared. Contents of C is gated to DB register. P1 (if selected) is cleared. Contents of C are gated to P1 or to selected B register. P1 is gated to P2 (if P is selected), C register and controls are cleared.

For Transfer to A or Q: Press TRANSFER to set transfer FF. The signal passes down the flip-flop chain to set keyboard bus priority. H310 is pulsed to start the manual timing chain. As the signal passes down the timing chain: Enter A1 FF is set and F2 register is forced to a 20 (function code for load A) or enter Q1 FF is set and F2 is forced to a 21 (function code for load Q). DB register is cleared; (C) is gated to DB register. The arithmetic timing chain is started. X and A2 registers are cleared. DB register is gated through I^4 to X; nothing is gated to A2. (X) and (A2) begin to propagate through the adder. C register and its controls are cleared. A1 or Q1 is cleared. (X) is gated through I^1 to Q1 or the $\overline{\text{sum}}$ from the adder is gated to A1 through I^0.

1. Indicate the state of the digit counter FFs for each position of C register by filling in the following chart.

	K320/321	K322/323	K324/325	K326/327	K328/329	K330/331
Digit 7						
Digit 6						
Digit 5						
Digit 4						
Digit 3						
Digit 2						
Digit 1						
Digit 0						

2. Indicate three manual operations which will cause C register to be cleared. _____

3. With a select register key pressed on the keyboard, what condition must exist in order to obtain a keyboard active status? _____

4. What is the function performed by K336/337 (last digit FF)? _____

5. Briefly describe the need for flip-flops K354/355 and K356/357. _____

6. What function is performed by the flip-flop specified at the times given? _____

	P Register Previously Selected	A Register Now Selected
Resync T2		
Resync T3		
Resync T0		
Resync T1		
Resync T2		

7. What are the transfer operations between C register and the register selected at the keyboard? _____

8. With the keyboard active after the computer has been stopped and A register selected, what would occur if TRANSFER were pressed and released twice in succession? _____

9. What is the significance of the preceding answer during manual register entry operations performed at the keyboard? _____

10. With the computer stopped and a keyboard master clear already performed, list the sequence of switches that must be pressed to place the quantity 03, 000, 000 into Q register. _____

11. What prevents simultaneous selection of more than one register at the keyboard? _____

12. Why is it necessary to obtain bus priority during a transfer sequence (i.e., what malfunction could occur if bus priority were omitted)? _____

13. List the operations that occur within the program control section as a result of performing a master clear at the keyboard. (Assume that the machine is stopped.) _____

14. Under what conditions can go FF be set? _____

14. Under what conditions can go FF be cleared? (Include the output translation for J071.) _____

MANUAL READ STORAGE

Read storage enables the operator to monitor the contents of one particular storage location while the computer is either running or stopped.

During a read storage operation the storage location set on the BREAKPOINT Address Selector switches is read and its contents displayed in C register. This location is continually monitored at a rate determined by the auto step oscillator.

The command timing chart on the following page is for a manual read operation. Refer to Logic Diagrams as you progress through the operation.

To perform read storage:

1. Set the address to be referenced on the BREAKPOINT Address Selector switches. This is an address in either main memory or one of the 64 locations in register file memory.
2. Set the Mode Selector switch to STO if referencing main memory or to REG if referencing register file memory.
3. Press keyboard switch READ STO.
4. When the auto step oscillator puts out a signal, transfer FF is set. If the computer is running there is a delay while the current instruction is executed; if the computer is stopped there is no delay. The contents of the designated location are read and displayed in the C register by use of the manual timing chain and controls. When the data has been displayed, normal operation continues. (If the computer is not running, nothing occurs.) When the signal from the oscillator again comes up, the cycle is repeated. The designated location is continually monitored at the auto step rate until another keyboard switch is pressed to release READ STO.

Table 14. READ STORAGE TIMING CHART

TIME	TERM	COMMAND	CONDITION	REMARKS
				Set the BREAKPOINT Address Selector switches to the address to be referenced. (This may be a main memory address or one of the 64 registers in the register file.) Set the Mode Selector switch to STO to reference main memory or to REG to reference register file memory.
Async				Press keyboard switch READ STO
V073, N319	K316/317		Transfer FF clear	Set read storage sync (puts J368 to a 1).
V071	K302/303		M358 = 0	Set transfer when the auto step oscillator signal goes to 1.
V074	K310/311		K314/315 clear	Set transfer lockout.
V076	K358/359		K314/315 clear	Set wait priority.
- - - -	- - - - -		Computer running	Wait until the instruction currently being processed is completed.
V082, V084, or V086	K340/341		Computer running. RNI next.	Set K340/341 via J301 or via keyboard bus request.
V072	K340/341		Computer stopped.	Set K340/341 via J357.

READ STORAGE TIMING CHART (Cont)

TIME	TERM	COMMAND	CONDITION	REMARKS
V073	K314/315			Set enable C to selected register.
Odd time	K342/343	Request bus Transmit read Transmit storage address on S bus	K210 = 1	Set keyboard bus request if block control does not have bus priority. Transmit read via T655. Gate breakpoint address (M6XX and M7XX) to EXX8, from EXX8 to T6XX transmitters.
N050 and	H117		K212/213 clear, no register selected	Keyboard operation obtains bus priority.
V074	Clear K310/311		K342/343 set	Clear transfer lockout.
	Clear K358/359			Clear wait priority.
V117	Set K212/213			
	Set K116/117			Transmit a storage request to the selected module via one of the T65X transmitters.
Even time N050	H115			
V115	K012/013	Enable data bus		Set K012/013 to lock out multiple pulses.
— — —	— — — — — —	— —		Main control now waits for the selected storage module to provide a reply.
Async	R555	Reply		Reply received and used as an input to the resync circuit.
V061	H310			Resynchronized reply provided to start the manual timing chain.
V310				
V311	K004/005	Set address mode (word)		V311-V314 provide idle time while the word is read from storage and placed on the data bus.
V314	K340/341			Clear keyboard bus request.
V315				
V316	H317	Clear C and controls		
V316	H401	Clear DB register		

READ STORAGE TIMING CHART (Cont)

TIME	TERM	COMMAND	CONDITION	REMARKS
V317	K012/013			Clear storage request lockout.
V317	Clear K116/117			DB and C registers are cleared. Clear storage request.
N401	H410	EXX2 to DB register		
V317	Clear K342/343	Direct		
	H318	EXX2 to C		Clear keyboard bus priority.
N41X				Gate the data from the bus into DB register (EXX2 to DB register).
N32X				Gate the data from the bus into C register (EXX2 to C).
V071	Clear K302/303		Y858 = 0	Clear transfer when the auto step oscillator signal goes to a 0.
V073	Clear K314/315		K302/303 clear	Clear enable C to selected register.
				If processor is running when K342/343 is cleared (V317 time), program control may obtain the bus and continue processing instructions. Read storage again occurs on the next cycle of the auto step oscillator and will continue until a keyboard switch is pressed to realease READ STO.

MANUAL WRITE STORAGE

Write storage enables the operator to store information in one particular storage location in main memory or in register file memory. Write storage may occur while the computer is either running or stopped.

During a write storage operation, information contained in C register is stored in the location specified by the BREAKPOINT Address Selector switches. Storage occurs when TRANSFER is pressed.

The command timing chart on the following page is for a manual write operation. Refer to Logic Diagrams as you progress through the operation. Then answer the questions following the chart.

To perform write storage:

1. Set the address to be entered on the BREAKPOINT Address Selector switches. This is either an address in main memory or one of the 64 locations in register file memory.
2. Set the Mode Selector switch to STO if referencing main memory or to REG if referencing register file memory.
3. Press keyboard switch WRITE STO.
4. Load the data to be stored into C register.
5. Press TRANSFER.
6. If the computer is running there is a delay until the current instruction has been executed; if the computer is stopped, there is no delay.

The manual timing chain and controls are used to store the data in the designated storage location. Then normal operation continues. Write storage may be performed again by pressing TRANSFER. Another keyboard selector switch must be pressed to release WRITE STO.

WRITE STORAGE TIMING CHART

TIME	TERM	COMMAND	CONDITION	REMARKS
				The address to be referenced must be set on the BREAKPOINT Address Selector switches. The Mode Selector switch must be set to STO or REG.
				WRITE STO is pressed.
V073, N319	K318/319		Transfer clear	Set write storage sync (puts J365 to a 1).
Async.				The data to be stored is loaded into C register.
Async.				Press TRANSFER.
V071	K302/303			Set transfer.
V070-4	K310/311		K314/315 clear	Set transfer lockout.
V072-6	K358/359			Set wait priority.
-----	-------	---	Computer running	Wait until the instruction currently being processed is completed.
V082, V084, or V086	K340/341		Computer running RNI next K358/359 set	Set K340/341 via J301 or via keyboard bus request
V072	K340/341		Computer stopped	Set K340/341 via J357.
V073	K314/315			Set enable C to selected register.
Odd time N051	K342/343	Request bus	K340/341 set K210 = 1	Obtain bus priority if block control does not have bus priority.
		Transmit write		Transmit a write signal to storage via T655.
		Transmit write character designators		Transmit write character designators to storage via the T66X transmitters.
		Transmit storage address on S bus		Gate the breakpoint address to EXX8, EXX8 to T6XX transmitters.
Even time N050	H117		No register selected	
V117	K212/213			
	Set K116/117	Storage request		Transmit a storage request.
V070-4	Clear K310/311			Clear transfer lockout.

WRITE STORAGE TIMING CHART (Cont)

TIME	TERM	COMMAND	CONDITION	REMARKS
	K358/359			Clear wait priority.
Even time N050	H115			
V115	K012/013 H400 K004/005			Set lockout to prevent multiple pulses. Set address mode.
N400	H401	Clear DB register		
N407	H440	CXX2 to DB register		DB register is cleared.
N440				CXX2 is gated to the DB register ($\overline{C}$) to CXX2 was enabled earlier), (DBR) to the T5XX data bus transmitters. Force EXX2 to output 1s.
— — — —				Wait for a reply.
Async	R555	Reply		Reply received by resync circuit.
V061	H310			Resynchronized reply provided to start the manual timing chain. .
V310				V310-V314 provide idle time while the data on the bus is being accepted by the storage module.
V314	K340/341			Clear keyboard bus request.
V315				
V317	Clear K012/013 K116/117	Release bus Drop storage request		Program control may now continue normal program execution if the processor is running.
V317	Clear K342/343	Clear keyboard bus priority		
N33X		Clear C		
Async				READ STO, SW, EN, KYBD OFF, or a register switch must be pressed to release WRITE STO.
V073	Clear K318/319		K302/303 clear	Clear write storage sync.

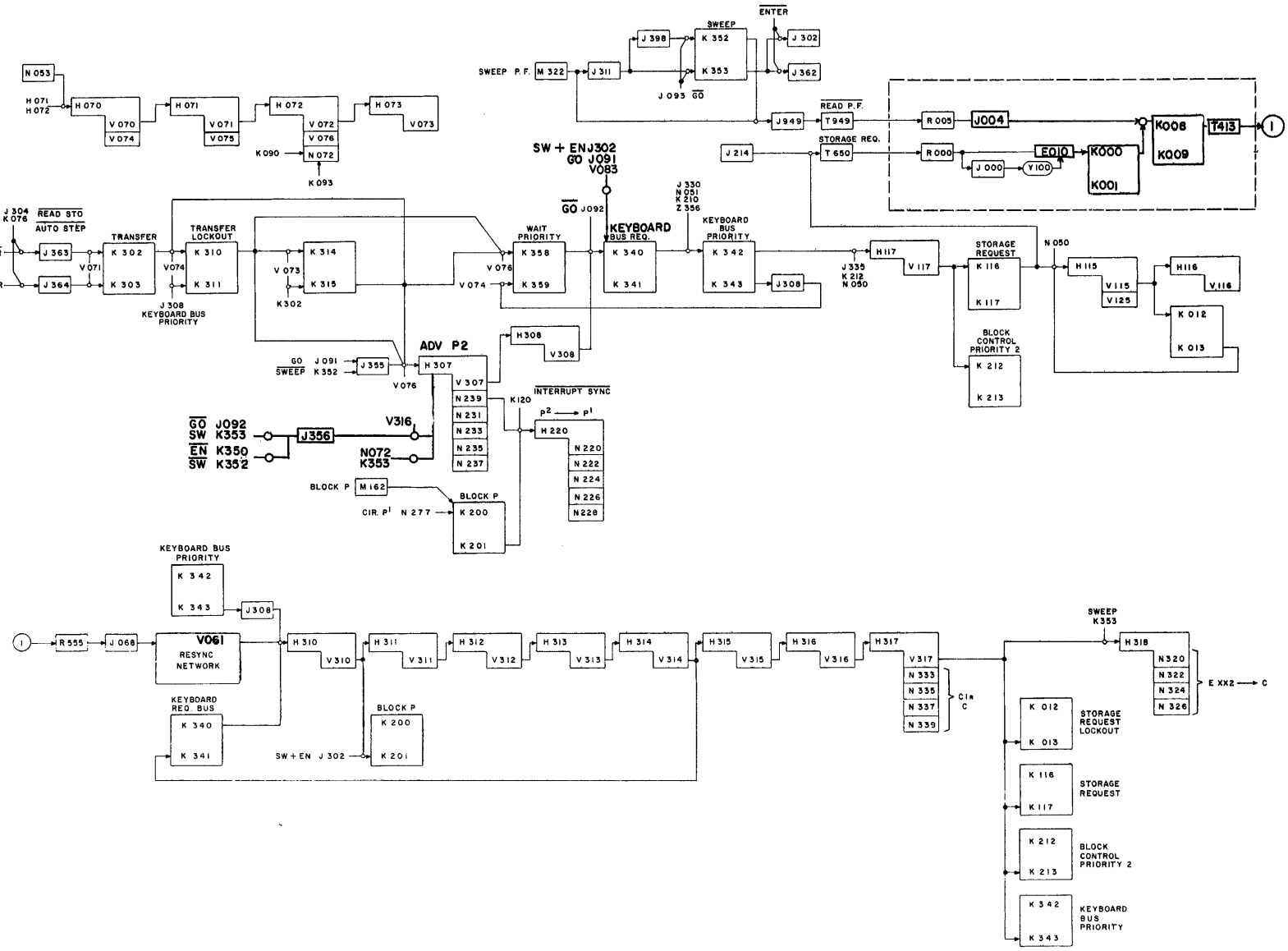


Figure 186. Sweep Page Index File

Questions on Manual Read/Write Operation

1. List the steps necessary to allow you to observe the contents of location 20603 while the computer is running and while the computer is stopped.

Running	Stopped

2. If you want to change the contents of location 23 in the register file, what steps must be performed? List in order the buttons that must be pushed and the operations that must be performed.

Enter Page Index File Sequence--timing begins with Enter PF switch depressed and C equal to the quantity to be entered.
Depress TRANSFER
V071: Set K302/303 (Transfer)
V072
V073
V070/V074: Set K310/311 (Transfer Lockout FF)
V071
V072/076: Set K358/359 (Wait Priority FF)
N072/K351: Set K340/341 (Keyboard Bus Request FF) (if Go) (start here for Go)
*N051/V073: Set K314/315, Set K342/343 (Keyboard Bus Priority FF)
N050: Input H117
V117: Set K116/117 (Storage Request FF) Set K212/213 (Block Control Priority 2 FF)
N050: Input H115
V115: Set K012/013 (Storage Request Lockout FF) Input H400
N400: Input H401
N401: Clear DBR, Input H440
N440: C7X2 to DBR, Input H411
N411: Set K214/215 (Delayed STO Request FF) Transmit a Storage Request to the Multiprogramming module.
V061: Resynced Storage Reply, Input H310
V310: Clear Block P K200/201 (first time)
V311
V312
V313
V314: Clear K340/341
V315
V316: Input H307
N239/V317: Advance P2, Clear: K012/013, K116/117, C; Input H308; Clear: K214/215, K212/213, K342/343; Input H220.
N220/V308: P2 to P1, Set K340/341 if Go and return to *.

SELF-EVALUATION QUIZ ON CHAPTER 6

TRUE OR FALSE OR FILL IN THE BLANKS

1. A different card type is used for the master clock and the clock amplifiers.

2. One cycle of the master clock defines one phase time.

3. The clock outputs are available any time power is on.

4. The output of the resync network is available only when the computer is running.

5. For a sweep continuous operation the Go FF will be set.

6. The correct sequence for Test mode is Master Clear, Go, then Stop.

7. The Test mode repetition rate is switch-selectable at 1 ms or at an auto-step rate.

8. If a change in selection is made from a 15-bit register to a 24-bit register, the background indicator will move from digit position 8^7 to 8^4.

9. The C register is part of the transfer path for all keyboard operation.

10. The presence of the background indicator specifies which digit position will be entered when the digit key is depressed.

11. The C register is the only register whose contents may be displayed when the computer is running.

12. When K342/343 is set, the keyboard has bus priority.

13. During the B2 manual entry sequence, _____ will be a 0 to allow gating from C to B2.

14. A P1 to P2 transfer is not necessary during manual entry into P.

15. The Arithmetic timing chain is not started for manual entry into A or Q.

CHAPTER 7

READ NEXT INSTRUCTION SEQUENCE

FOUR MAJOR FUNCTIONS

The 3300 Computer has four major storage sequences, RNI, RADR, ROP, and STO. The first sequence, RNI (read next instruction) has four main functions:

1. To read an instruction word from storage and place the instruction in F register.
2. To insure proper updating of P register.
3. To sense for all types of interrupts.
4. To start/stop.

Let's consider the first function of RNI, to read an instruction word from storage and place it in F.

In order to visualize the data flow path for the address look at figure 188. Notice the heavy black line originating at the P register. This line defines data flow for the address.

Now let's consider the data flow path for the instruction word coming from storage. Again look at figure 187, but this time notice the heavy dashed line originating at Z register of module 0. This line defines the data flow path for the instruction word.

The second function of RNI is to insure proper updating of P register. P may be updated at times other than when in RNI, but RNI must always update P. Four cases to be considered in updating P are:

1. P + 1.
2. P + 2. (SKIP)
3. Jump.
4. Block advance P.

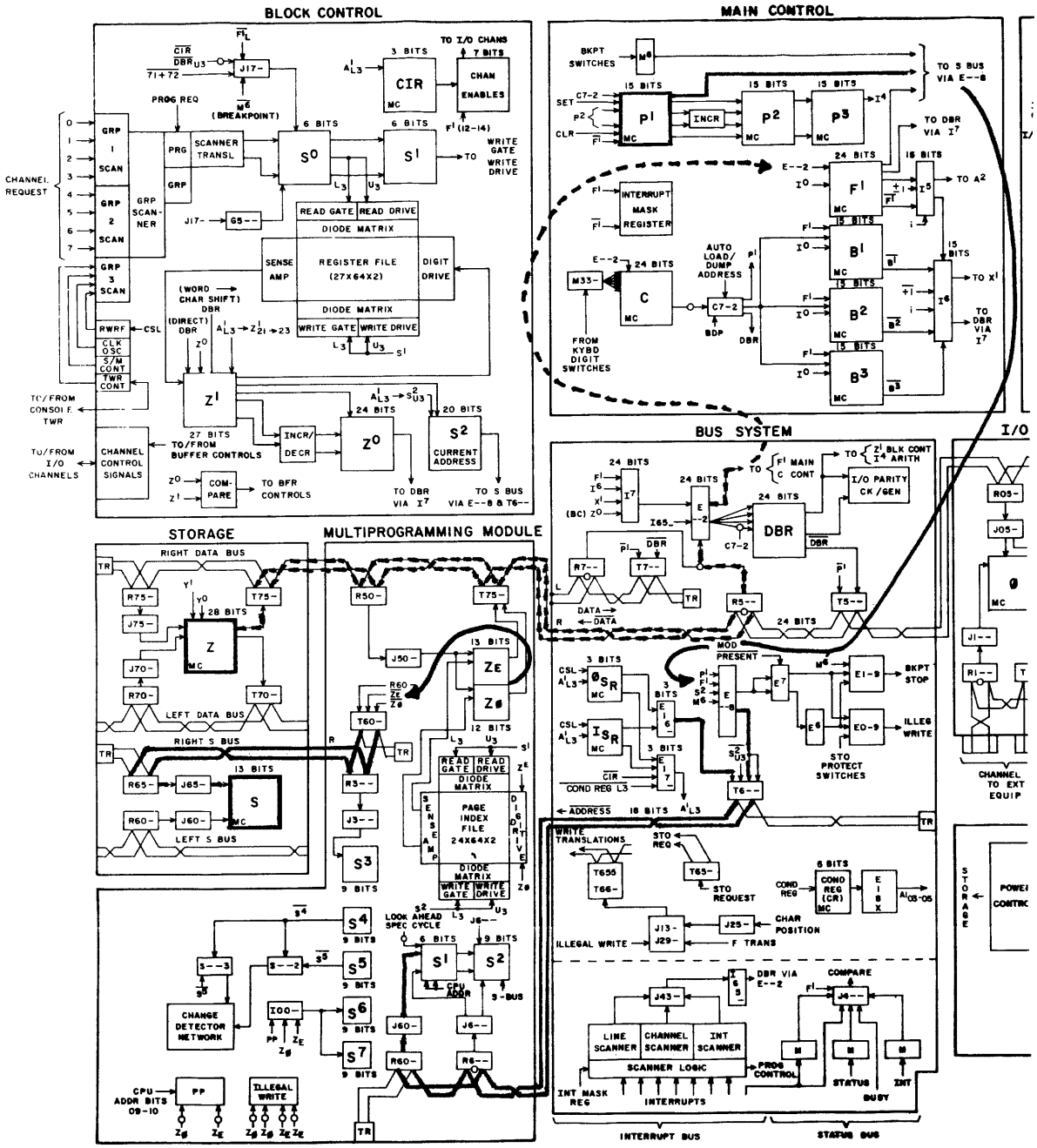


Figure 187. Data Flow Path

Each case will be discussed in detail at V014 time of RNI sequence in this text.

The third function of RNI is to sense for all types of interrupts. This includes powerfail, illegal write, STO parity error, normal interrupt, and trap sequence, all of which will be discussed in detail in chapter 11. Note that RNI timing assigns priority to interrupts. The RNI times associated with these are:

1. V004 - Sense for illegal storage reference, STO parity error
2. V006 - Sense for power fail
3. N010 - Sense for normal interrupt
4. V080 - Sense for trap

The fourth function of RNI is start/stop. When GO on the keyboard is pressed an RNI is normally the first sequence entered. The computer may be stopped by clearing go FF. With the exception of a stop during cycle step or after a storage parity error, all stops are made at the end of an instruction. A stop will occur:

1. When a program stop is required.
2. After one instruction is executed in instruction step mode.
3. When breakpoint stop occurs.
4. When the go sync FF is cleared by pressing STOP on the console keyboard.
5. After one memory sequence is executed in cycle step mode.

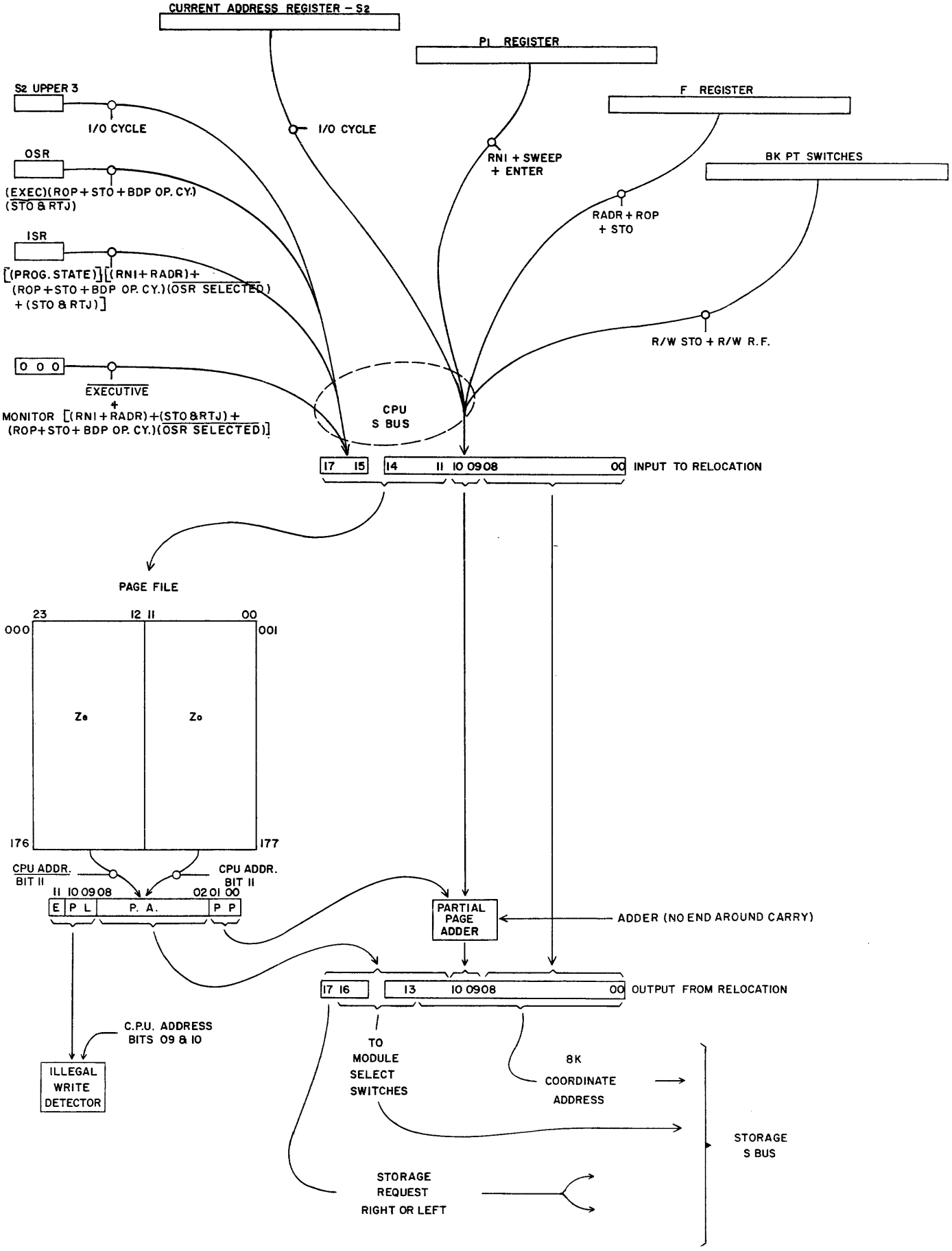


Figure 188. 3300 Address Flow

DETAILED TIMING

A flow chart of RNI sequence is shown in figure 189. It depicts the several steps involved in accomplishing the four major RNI functions. A time-ordered list of events pertinent to successful operation of RNI follows:

1. Request bus system. Advance P register if other than first RNI. (P register now holds the address of the next instruction to be executed.)
2. Obtain bus priority.
3. Transmit address in P to storage via the S bus.
4. Do not transmit a write signal. (Absence of a write signal is a read signal.)
5. Transmit a storage request (module request).
6. Test for BPI stop. Wait for reply.
7. Receive and resync reply signal from storage.
8. Clear request bus FF.
9. Time out the access (allows time for storage to read and place the word on the data bus).
10. Unconditionally release the bus system.
11. Stop if breakpoint stop; Clear F register if not BPI stop.
12. Gate data bus to F register via EXX2. (F register now holds translated instruction.)
13. Test for stop (cycle step mode or instruction step and end of instruction or STOP pressed and end of instruction).
14. Advance storage reference controls to RNI + RADR + ROP + STO.
15. Progress to the next storage reference if not stop.

Figure 190 is a block diagram of the RNI timing chain which indicates the times of occurrence for the RNI events. Not all times of the timing chain are listed and note that item 7 is associated with no specific time. This diagram will reappear throughout the detailed discussion of RNI timing to enable you to keep track of the overall timing of RNI.

ENTRANCE TO RNI

The RNI sequence provides for most manual and program starts and stops. The computer may be started by pressing GO if normal program operation is desired or by pressing SW/EN CONT if sweep or enter operation is to be performed. These switches are mutually exclusive although they both set initiate go FF (K304/305). The output of initiate go is timed with a resync pulse to set the go sync FF. The output of this FF then sets go (K090/091) and the output of go allows a start pulse to begin an RNI sequence to read the first instruction of a program. (The logic discussed above is found in 3300 Computer System Logic Diagrams, page2-15)

The computer may also be started from halt by pressing console switches STORAGE CYCLE STEP, INSTRUCTION STEP, or AUTO STEP. Operation of the computer in these modes is discussed elsewhere in this manual.

Figure 191 shows all the possible entry points to the RNI sequence. The inputs to H012, H016 and H014 are from the other sequences or operations. Most inputs occur at the end of a sequence or operation. The output of N014 initiates RNI.

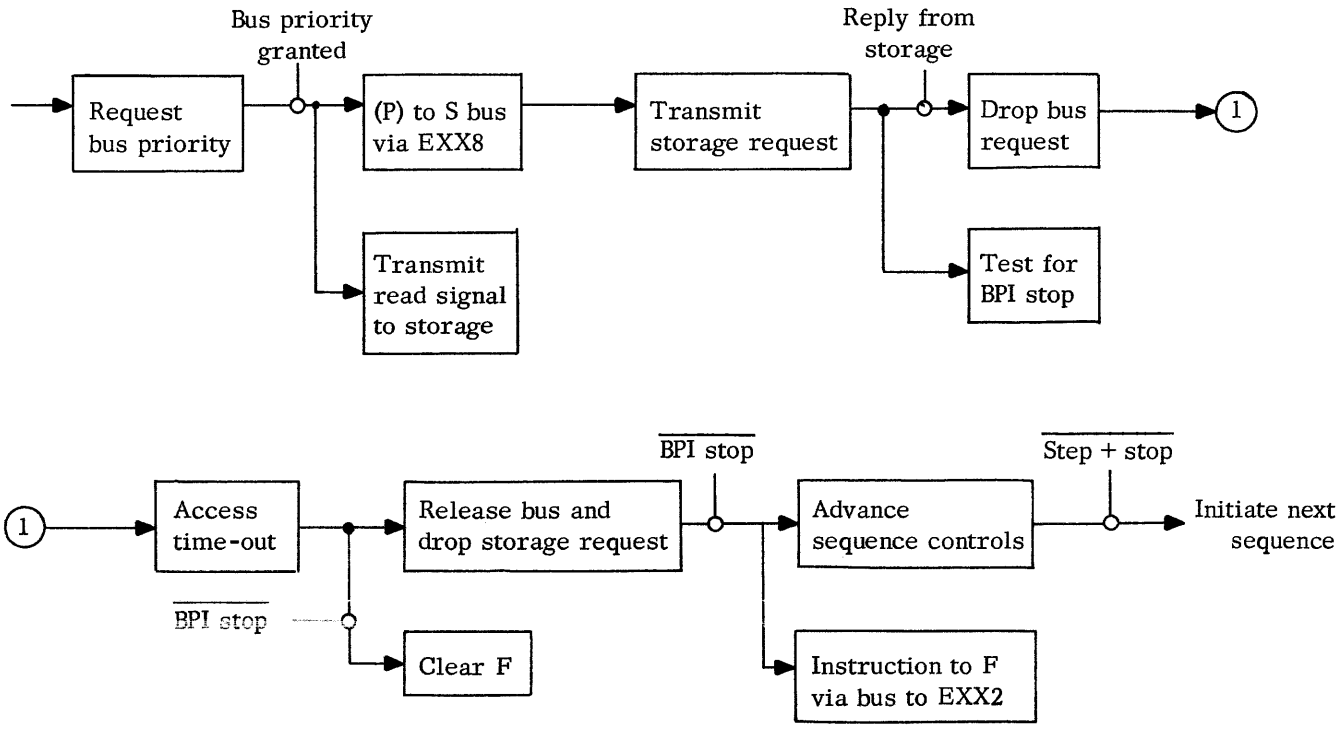
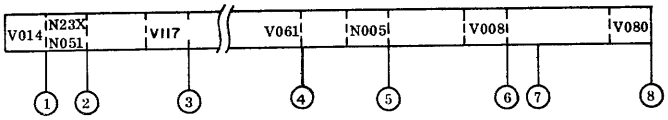


Figure 189. Flow Chart of RNI Sequence



(1) Initiate RNI, Request the Bus System

(2) Update P.

(3) Request storage.

(4) Reply from storage.

(5) Release bus system.

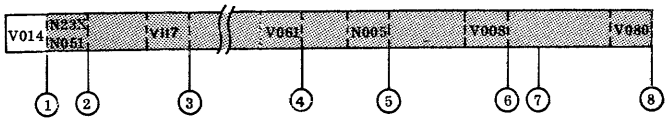
(6) Enable the Instruction to the F register

(7) Decode instruction.

(8) End RNI.

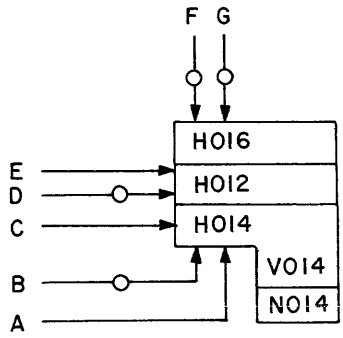
Figure 190. Major Timing of RNI Sequence

INITIATE RNI



(1) Initiate RNI, request bus system.

N014/V014: Set K000/001 (request bus) and K002/003 (RNI lockout). Input to H231 (advance P2) except first time after a master clear, a jump, entering P manually, or after DISABLE ADVANCE P console switch has been pressed.



GATE	TERMS	TRANSLATIONS OR REMARK
	K340	
A	V087	00.7 or 01.4 initial + 06 to 10.0
	K081	+ 2X to 52 + 54 + 56 + 67
B	N151	71 to 77.4 + 77.6
C	F701	$\overline{\text{BPD instruction and}}$
	F510	02 + $\underline{11}$ to 13.3 + 14 to $\underline{17}$ +
	K002	$\underline{(\text{IRT} \cdot \text{RF})}$ + (70.X + 55.X00)
	V109	(Trap)
D	F416	01 indexed + 03 to 05 + 10.1 to
	K002	10.7 + 13.4 to 13.7
	V103	
E	N217	from first advance P during skip or restart main control after read or write storage
F	F366	00.(0 - 6) + 01 $\overline{\text{indexed}}$ + (77.5X)
	F610	$\overline{(77.55)}$ + $\overline{(\text{Exec})}$ + 55.0 + 55.4 +
	F611	+ (77.6) $\overline{(77.60)}$ $\overline{(77.630 + 77.670}$
	K950	$\overline{+ 77.674 + \text{Exec})}$ $\overline{(77.64 + 77.65}$
	V011	$\overline{+ \text{Exec} + \overline{\text{PF Pres}})}$ + 77.7
G	F558	(IRT . RF) or second RNI of 71
	V171	to 76

Figure 191. Entry to RNI Sequence

K080/081 (RNI sequence control FF) is set by either completion of a sequence or a master clear. (It is set after a master clear because it is assumed that the next sequence will be RNI.)

The S bus and data bus circuits are shared by main control and block control. Each of these sections needs the bus system for its operations. The section requiring the bus requests it and priority is granted to the sections alternately. Main control and block control have equal priority when requesting the bus system, in that neither can take it away from the other and each has first chance to get the bus system after the other releases it. V014 occurs only when initiating an RNI sequence. K000/001 (request bus FF) serves as the main control request bus flip-flop. (The block control request FF will be covered in detail in chapter 15.)

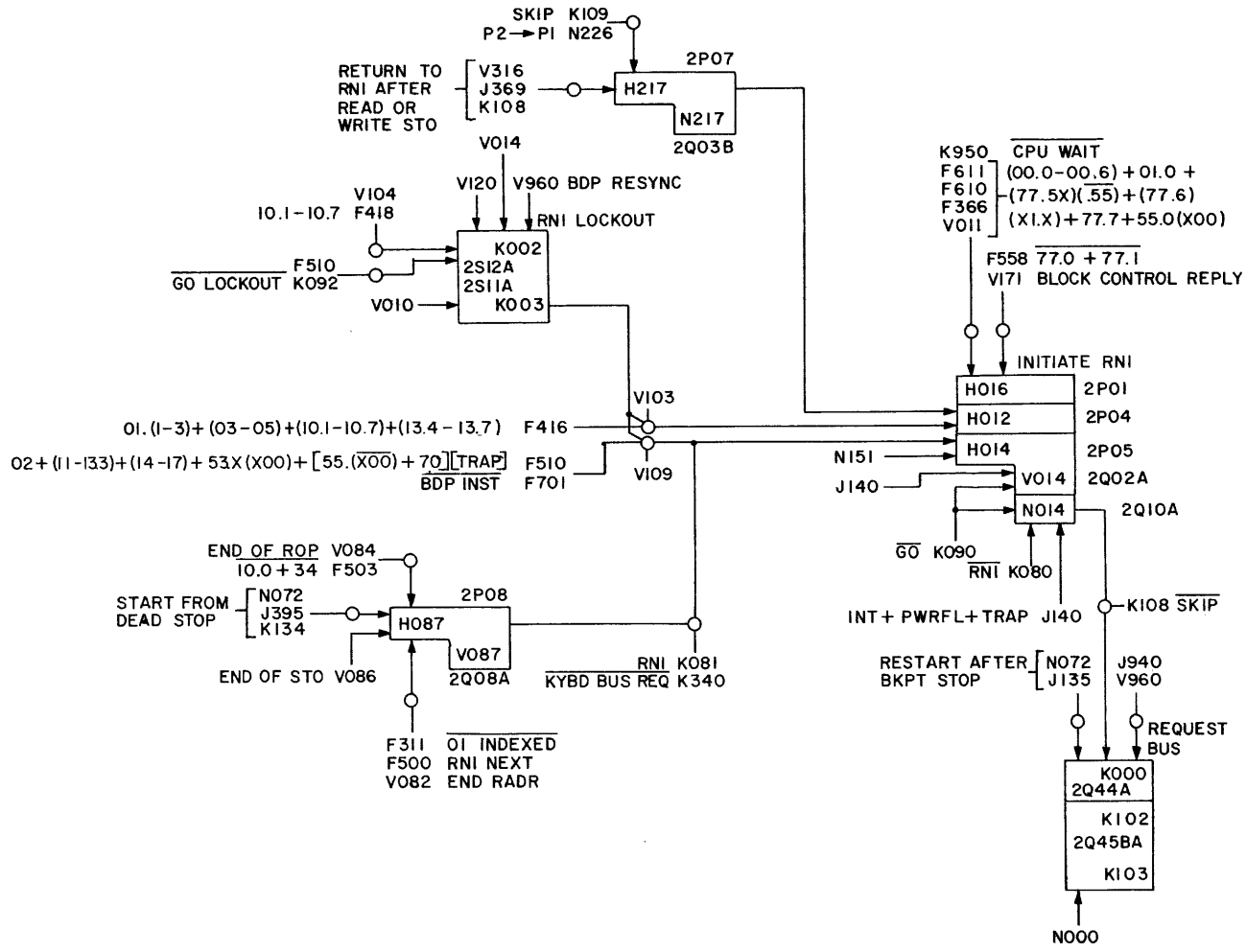
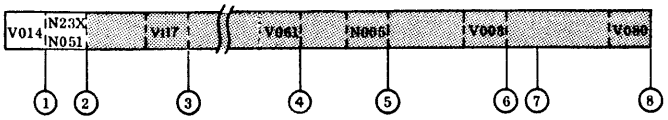


Figure 192. Initiate RNI Logic

UPDATE P



(1) Initiate RNI, Request the Bus System

(2) Update P.

The P register holds the address of the current instruction or the address of the first instruction at V014 time. At this time RNI must insure proper updating of P register. The four cases to be considered are:

1. P + 1.
2. P + 2 (skip).
3. Jump.
4. Block advance P.

To execute a P+1, refer to figure 193 and consider this timing:

V014: Set K000/001, input H231.
N231: Advance P2 (+1), input H220.
N220: P2 → P1.

To accomplish a P + 2 the skip FF (K108/109) must be set before V014 time. Instructions 04 to 10.7, 52, 71 to 76, 77.0 to 77.4, and 77.6 provide for P + 2 when a certain condition exists. For all but the 7X instructions the skip translators (logic diagrams 2-13) determine if a skip condition exists. For the 7X instructions the skip condition is tested in the interrupt or block control sections. When a skip condition exists during the execution of an instruction with skip capabilities, skip FF will be set. To understand a P + 2 refer to figure 193 and consider the timing below. Note that a skip (P + 2) adds 4 ∅ times to the RNI sequence.

K108/109 (skip) is set.
V014: Do not set K000/001; input H231.
N239: Advance P2 (+1), input H220.
N226: P2 → P1, input H229.
N229: Clear K108/109 (skip); input H012.
V014: Set K000/001; input H231.

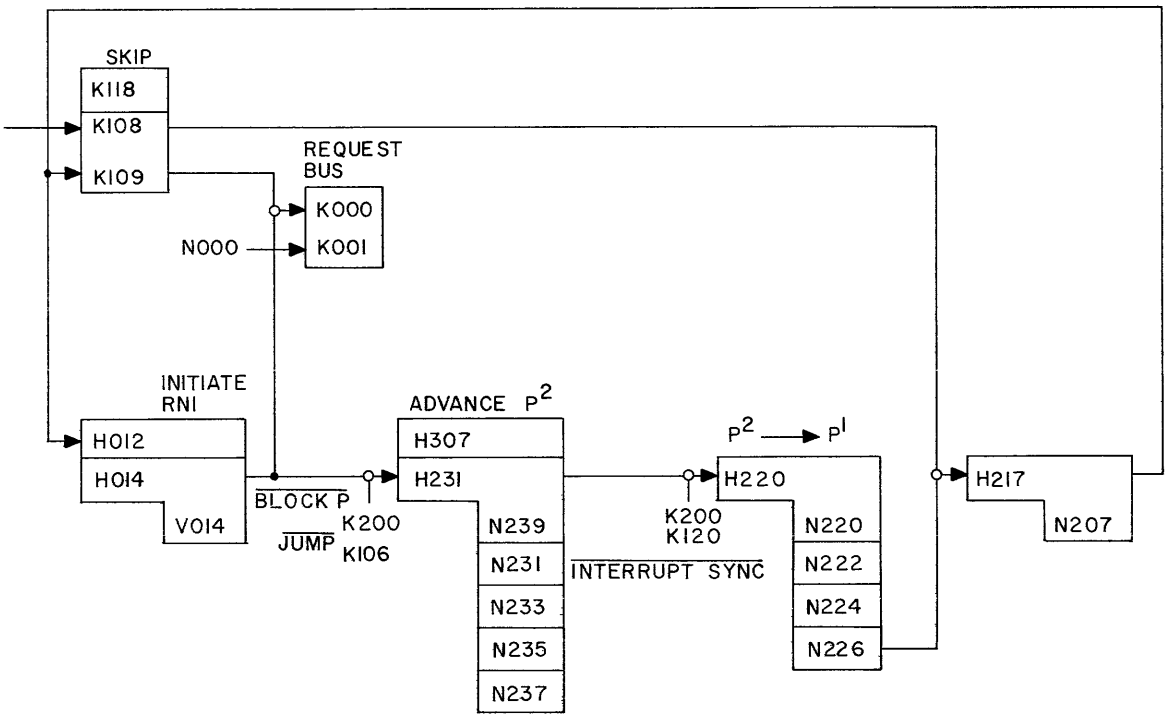


Figure 193. Update P (+1, +2) Logic

N239: Advance P2, input H220.
N226: P2→P1.

To accomplish a jump, the jump FF (K106/107) must be set before V014 time. The instructions 00.X-03.X and 70.(4 to 6) have jump capabilities. For setting of jump FF refer to Logic Diagrams, page 1-3. Executing a jump will not increase the time required for the next RNI. To understand how a jump is accomplished refer to figure 194 and consider the timing below:

V014: Set K000/001, input H211.
N211: Set P1, input H230.
N230: F1→P1, input H221. P2 P3.
N221: P1→P2.

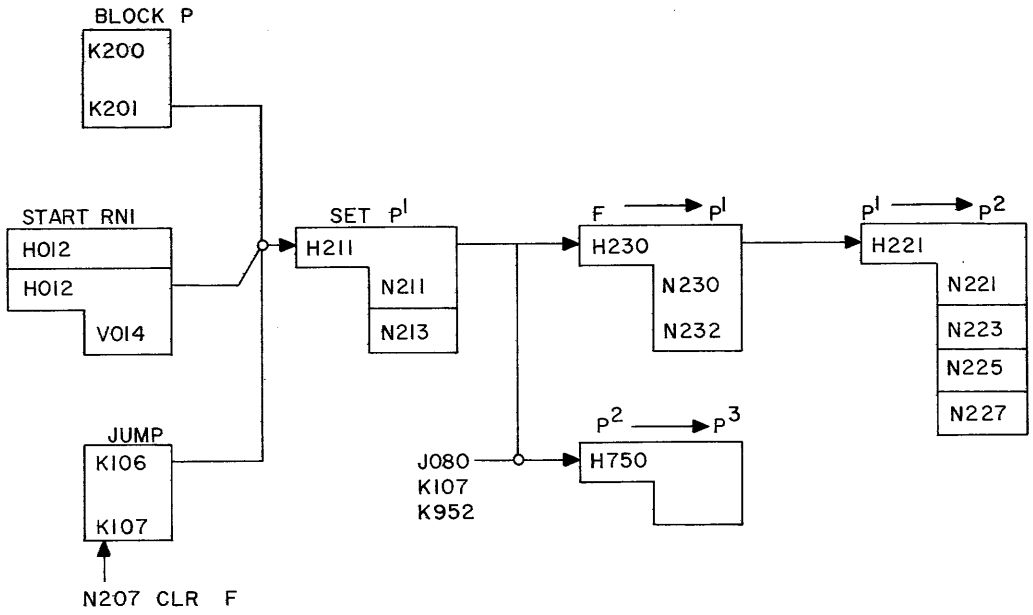


Figure 194. Jump Logic

To accomplish a block advance P the block P FF (K200/201) must be set before V014 time. Block P will be set for:

1. Master clear.
2. Manual entry into P register.
3. Console switch DISABLE ADVANCE P pressed.
4. Instructions 06 and 07.

When block P sets, inputs to H231 and H211 (Logic Diagrams, page 2-47) will be disabled. Therefore, P + 1, and P + 2, cannot be accomplished.

Obtain Bus Priority

N051: Set K010/011 (main control priority) if block control does not have priority. Input to H220 and advance P2 (Logic Diagrams, p. 2-47) except first time after a master clear, jump, etc. If K211 = 1, repeat preceding test at the next N051 time because block control has priority.

If K209= 1: Set K010/011 (program or main control priority), gate P1 to EXX8 to T6XX to S bus. This places an address on the S bus. Gate EXX8 to EXX7 to EXX6. This allows for generation of the proper module request.

T655 (READ) is off. (This absence of a write signal is in fact a read signal to storage.)

N050: Input to H117.

Setting K010/011 indicates that main control has bus priority. Setting K010/011 enables (P) to the S bus (see figure 204 for the flow path). The address on the S bus, now available to storage, is the address of the next instruction. (Note that if block control were using the bus system K010/011 would not be set until block control lost its priority and released the bus by clearing K208/209.)

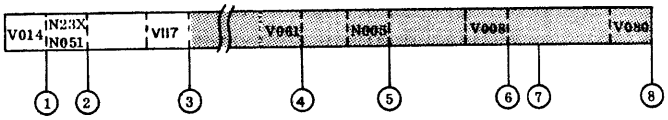
Address (S) Bus

The S bus (Logic Diagrams, page 1-19) provides a path for transmission of a storage address from main control to a storage module.

1. P register - The address in P specifies location of instruction to be read in an RNI sequence. The address from P will always be a word address.
2. F register - The contents of F specify location of the operand for an ROP sequence, location of the address for an RADR sequence, or address at which a word is to be stored in an STO sequence. (Each sequence is discussed in detail in its own chapter in this manual.)
3. S2 (current address register) in block control - An address is provided for block control access to storage.
4. M6XX-M7XX inverter rank - The BREAKPOINT Address Selector switch provides an address via this rank during a read/write storage operation. Breakpoint operations are discussed elsewhere. Either a character address or a word address may be provided by F register. P and M6XX-M7XX may provide only a word address. S2 will always provide a character address.

Addresses from these four sources are brought into the EXX8 inverter rank. The address is gated into EXX8 from the desired source by the W0XX gates. The complement of the storage address is then transmitted from EXX8 to the relocation chassis via the T6XX transmitter rank.

REQUEST STORAGE



(1) Initiate RNI, request the bus system

(2) Update P.

(3) Request storage.

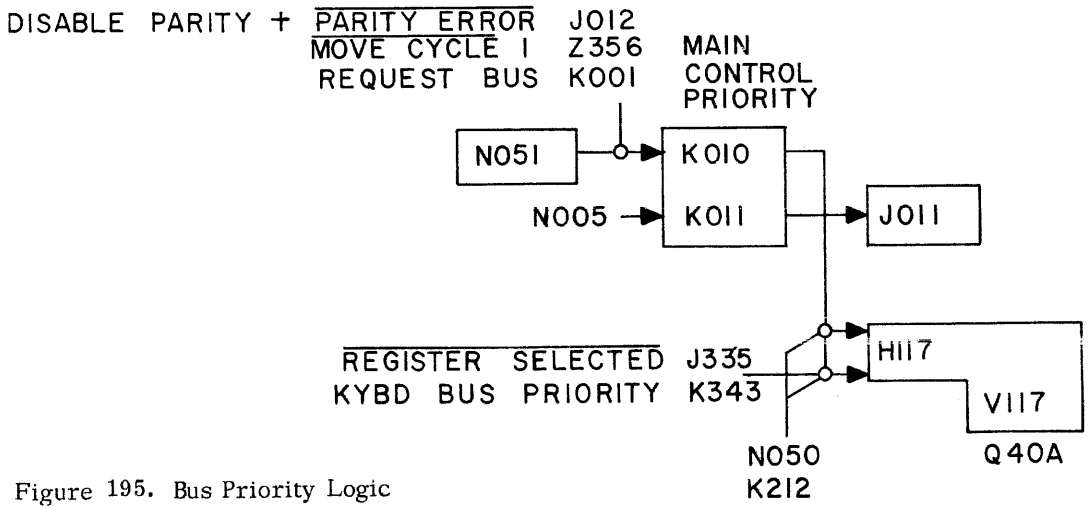


Figure 195. Bus Priority Logic

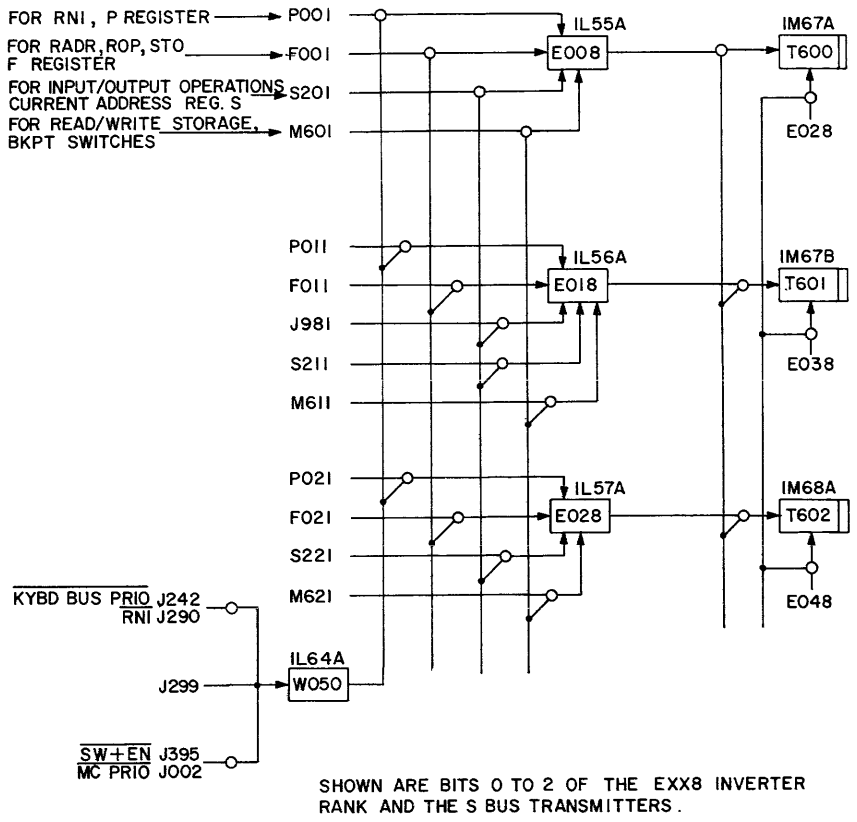


Figure 196. Address Transmission Logic

V117: Set K116/117 (storage request); transmit a request to the appropriate storage module.

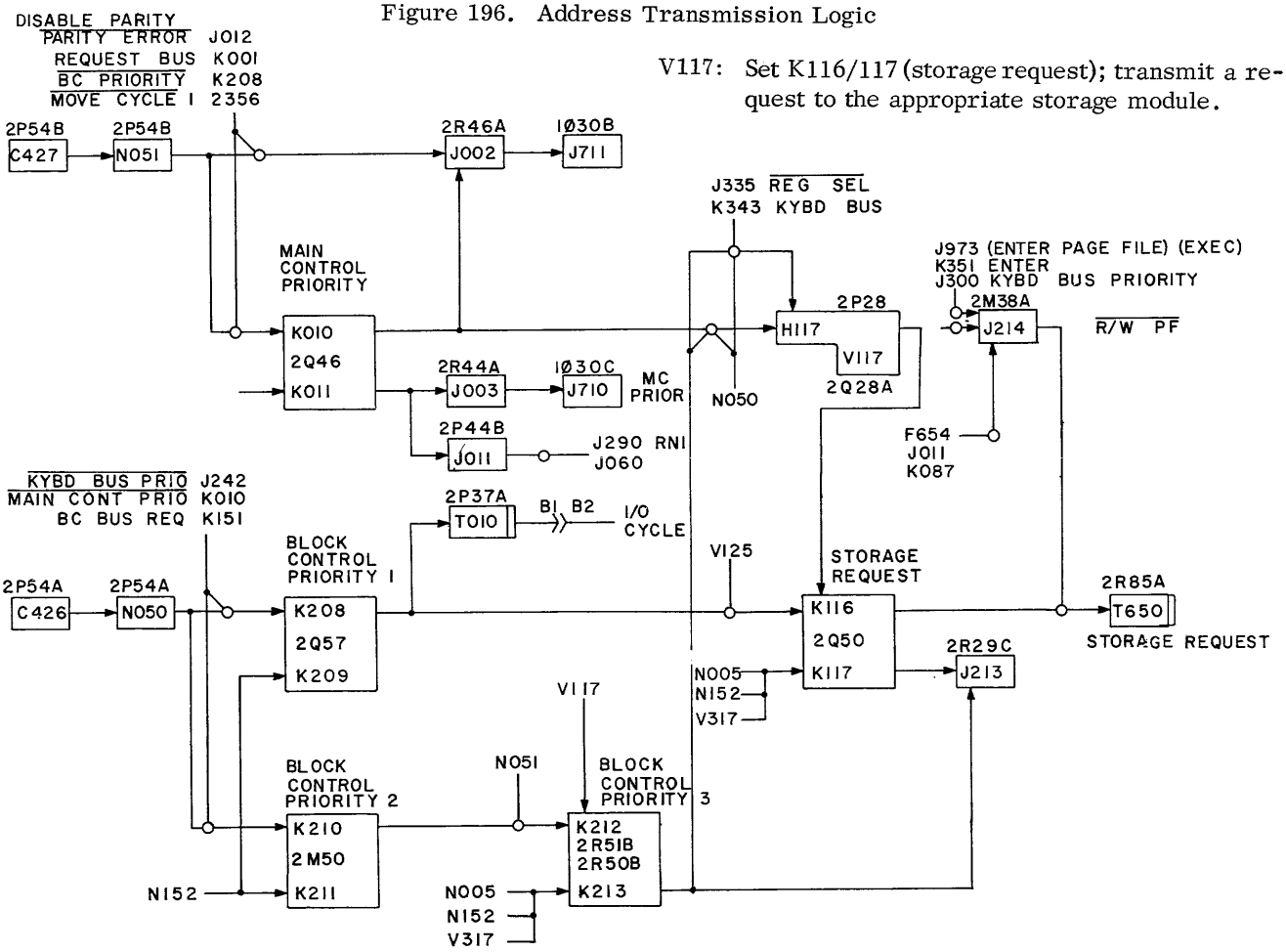


Figure 197. Storage Request Logic

While main control waits for the reply from storage, it will compare the breakpoint address with the address transmitted to storage for equality.

N050: Input to H115.

V115: Set K012/013 (request lockout) and K004/005 (word address mode). Input H116.

V116: Test for breakpoint comparison if BPI is selected and K136/137 is clear. (If breakpoint stop were selected and a stop occurred on previous RNI, K136/137 would now set.)

K012/013 (enable data bus) is set by V115. The purpose of this flip-flop is to break the inputs to H115 while K212/213 (priority delay) breaks the input to H117, thus preventing multiple outputs from the control delays. They are fed by N050, a raw clock which comes up every other phase time.

K004/005 (word address mode) is set unconditionally during RNI. It is used during ROP and STO and will be discussed at that time.

K134/135 (breakpoint stop) will set if the address on the S bus matches the breakpoint address, BPI is selected, and breakpoint lockout is not set. K136/137 (breakpoint lockout) being clear indicates that this is not the first storage reference after a breakpoint stop. NOTE: If it is the first storage reference after a breakpoint stop, K134/135 (breakpoint stop FF) will be prevented from setting to allow the computer to continue with the program. To say it another way, K134/135 will not be set during the comparison to prevent stopping the second consecutive time on the same address.

Main control then waits for the selected storage module to signal a reply. The waiting time varies with the status of the memory module.

To review what has occurred so far fill in the blanks:

1. (1) _____

   (2) _____

   (3) _____

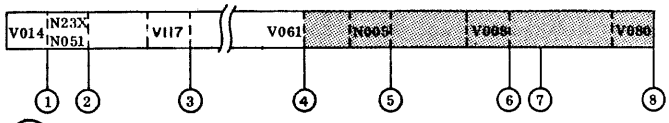
2. What is on the bus system at this time? _____

3. What is the state of main control at this time? _____

4. Considering major timing of the RNI sequence, what should occur next? _____

REPLY FROM STORAGE

Refer to figure 188 to reinforce your concept of how the logic accomplished the previous steps of RNI.



(1) Initiate RNI, request the bus system

(2) Update P.

(3) Request storage.

(4) Reply from storage.

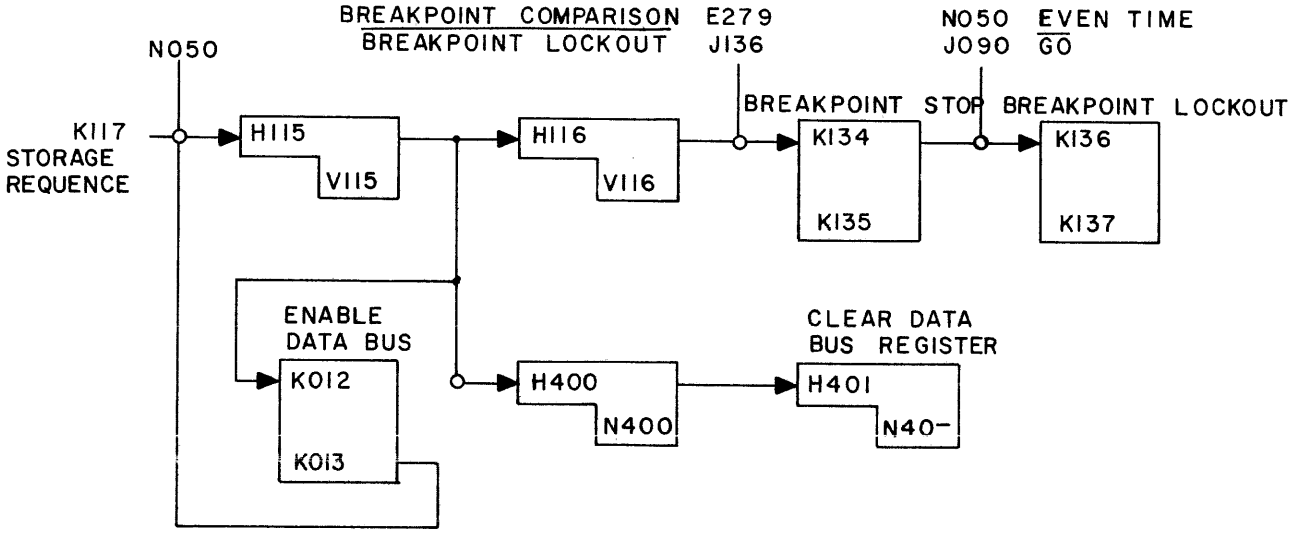


Figure 198. Breakpoint Logic

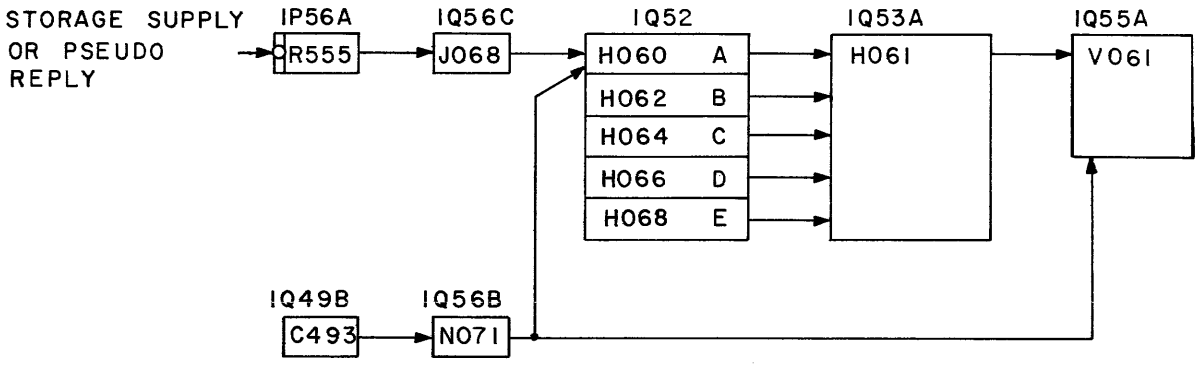


Figure 199. Resync Logic

Main control was hung up until the reply signal came from storage. The S bus held the contents of P register, waiting for pickup by the storage module.

The processor must wait for the reply signal from the storage module. The reply signal indicates that storage is processing the request and will have the information on the data bus after a predictable delay. (This delay time can be predicted because it is known that when the storage module sends the reply signal it is also referencing the location desired, and the time for a memory cycle is known.)

R555 (storage reply) (receiver)
V061 (resynced storage reply)

Reply signal R555 indicates that the storage module is processing the request and after access time the instruction will be available on the data bus.

A reply from any one of four storage modules is received by R555. The asynchronous reply (logical 1) is fed to H060 along with a timing pulse from N071. H060/62/64/66/68 and H061 convert the reply to a logic signal which produces a 1 output from V061 during an odd clock phase. This output is 62.5 nsec long and is not repeated regardless of duration of input.

The timing pulse is produced by C493, an independently tuned clock amplifier. This amplifier is fed by the clock pyramid but must be individually tuned so that an output from V061 occurs exactly at odd time. The procedure for tuning this amplifier is presented in the 3300 Printed Circuits Manual.

The resynchronized storage reply signals the main timing chain, the manual timing chain, or the block control timing chain that the storage module is processing the request and will accept a word during a write operation, or that the storage module will place a word on the data bus during a read operation.

V000-V005: Clear K000/001 (request bus) gives the storage module access time to store the information from the data bus.

The timing from V000 to V005 is called "time-out for access". Its purpose is to permit storage to reference an address and place a word on the data bus.

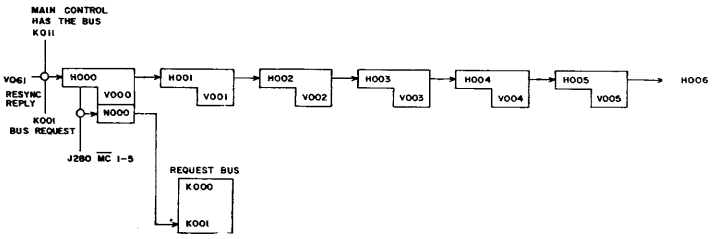
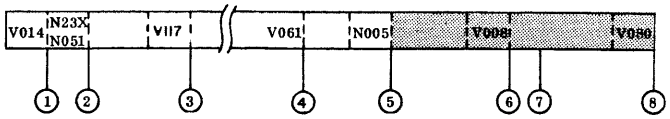


Figure 200. Access Time

One of the first things to occur after the reply signal has been received by the processor is the clearing of request bus FF. Clearing request FF neither clears nor releases the bus system. If main control needs the bus system, it requests it via K000/001 (request bus FF). If the bus system is not busy main control is given bus priority and sends a request for storage. Storage acknowledges the request by signaling that the request has been received and is being processed. However, the request for storage must be completely processed before storage can transmit the requested information. Essentially one might say that main control needs the bus, requests it, and storage starts processing the request. At that point the bus request --not the bus system--will be released.

Clearing K000/001 (request bus) allows block control the possibility of requesting the bus now. It is important to note that releasing the bus request is not the same as releasing the bus system. Main control still needs the bus system to get the information that is coming from the storage module, thus the bus system itself has not been released. After the access time-out and after main control has received the information it requested from storage, then the bus system will be released.

RELEASE BUS SYSTEM



(1) Initiate RNI, request the bus system

(2) Update P.

(3) Request storage.

(4) Reply from storage.

(5) Release bus system.

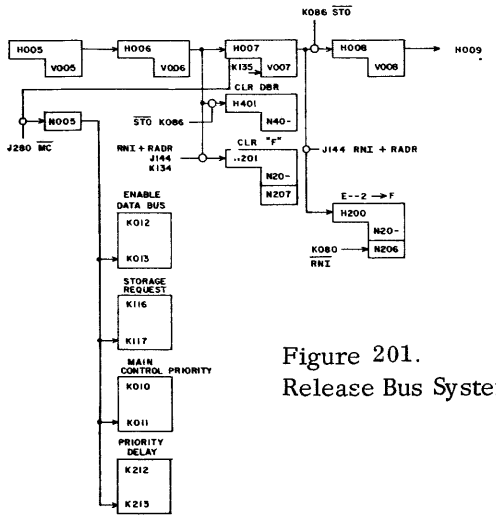


Figure 201.
Release Bus System

V006: Input to H201 (Logic Diagrams, page 2-29 if J138 = 1. Test for powerfail (see chapter 11).

V007, N005: Clear K010/011 (main control priority), F1, K212/213 (priority delay), K116/117 (storage request), and K012/013 (request lockout); input H200.

N050: Clear K122/123 (initiate storage request).

Operations depending on V007 will not occur if the breakpoint stop condition exists.

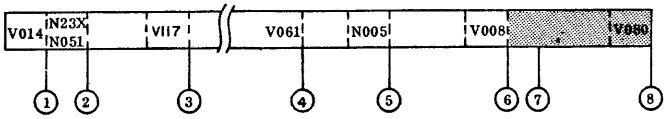
After access time has ela sed and as the information from storage is taken from the bus system, main control will release the bus system.

Releasing the bus system as soon as possible is important because block control and main control share the same bus system. If the other control area needs the bus system, the sooner it is released the sooner it can be used. Further discussion of this bus-sharing operation is supplied in chapter 15.

Breakpoint Stop

When BPI is selected the breakpoint address is continually compared to the storage address on the S bus. During an RNI sequence, if the storage address matches the breakpoint address, breakpoint stop FF (K134/135) is set. K135 is ANDed with V003 to clear the go FF and stop the computer. K135 also holds V007 to 0, disabling the main timing chain.

ENABLE DATA BUS TO F REGISTER



Refer to figure 188 and review all previous steps.

(1) Initiate RNI, Request the Bus System

(2) Update P.

(3) Request storage.

(4) Reply from storage.

(5) Release bus system.

(6) Enable the instruction to the F register

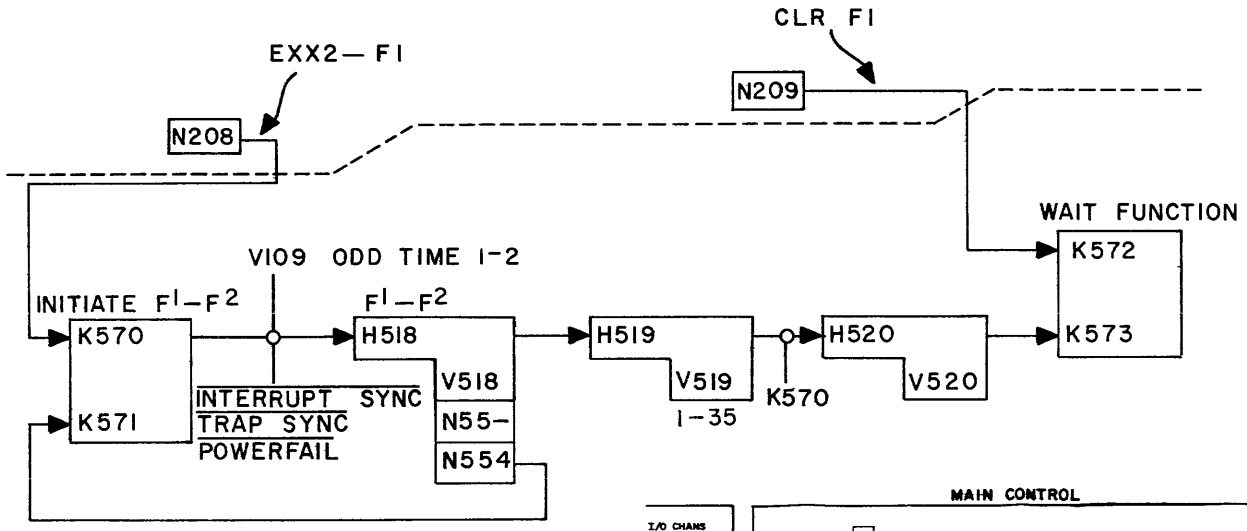
V007: Set K572/573 (wait function).
(N209)
V008: Set K570/571 (F1 F2 enable)
(N208) EXX2 to F1

These two flip-flops are used in the arithmetic section of the processor and will be discussed in chapter 12.

The output of N208 and N209 setting K570/571 (copy code FF) and K572/573 (wait function FF) occurs during each RNI sequence. The names of the flip-flops describe their functions. Near the end of RNI, after the instruction has been placed in F1 register, the upper nine bits of F1 are duplicated in F2 in the standard arithmetic section of the processor.

Since the arithmetic section can run independently during some arithmetic operations, F2 register provides the only source for the function translations. The function translations are needed to provide enables and permit gating during, for example, the multiply and divide operations.

The data bus to receivers to EXX2 is a static enable. See figure 204 for the block diagram flow. See figure 202 for the simplified logic. Note on figure 204 that EXX2 is in the flow path from storage to main control as well as to the standard arithmetic section. EXX2 could actually be considered the first branching point for the flow. In this case EXX2 feeds F register. Note that EXX2 also feeds the DB register.



Circuitry shown below the dotted line is discussed in greater detail in chapter 12.

Figure 202. F1 to F2 Logic.

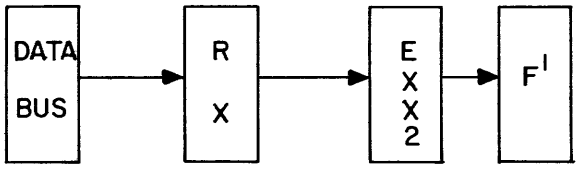


Figure 203. Instruction Data Path

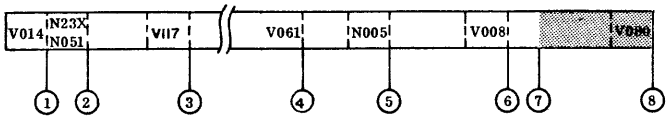
Figure 204. Data Flow for RNI.

Transfer to DB register occurs during all RNI sequences but is only used on the second RNI sequence for the 71 to 76 instructions. These instructions are 48 bits long and occupy two sequential memory addresses. Reading 71 to 76 instructions from memory follows this sequence during RNI:

1. First half of the 71 to 76 instruction is placed in F register and also in DB register.
2. The instruction in F register is translated and another RNI sequence is initiated.
3. P register is advanced and, during the second RNI sequence, the second half of the instruction word is read and placed in DB register. (The enable from storage to F register is not present during the second RNI of 71 to 76 instructions.)

These are the reasons for transfer to DB register during the RNI sequence. The execution of these 48-bit instructions will be discussed in chapter 15.

DECODE INSTRUCTION



(1) Initiate RNI, request the bus system.

(2) Update P.

(3) Request storage.

(4) Reply from storage.

(5) Release bus system.

(6) Enable the instruction to the F register.

(7) Decode instruction.

V009: Set K008/009 (sense interrupt during arithmetic) if arithmetic section is busy.
V010, N010: Test for an interrupt condition.
V011: Input to H080.

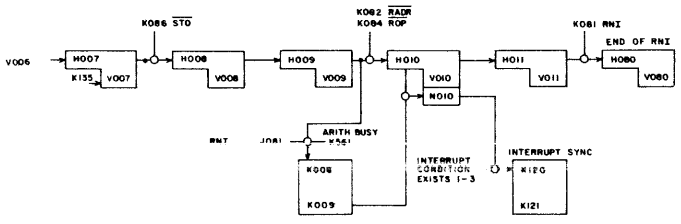


Figure 205. Decode Instruction and Sense Interrupt

Interrupt Recognition

An interrupt signal, generated by an external equipment or the control section of the computer, is a request for special action to handle a special condition which has occurred. The main program is suspended temporarily while a special routine of instructions is performed. The interrupt request may be acknowledged and an interrupt routine initiated during RNI or the RADR sequences.

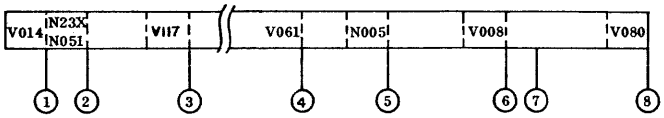
An interrupt may be sensed only at N010 time of the RNI sequence. When arithmetic section is not busy N010 time will automatically come up only once in the cycle 1 Ø time after an RNI pulse enters H010. This would mean that an interrupt could be sensed only during the one time N010 was up. K008/009 remedies this situation.

Suppose the RNI pulse enters H010; 1 Ø time later it senses for an interrupt and there is none. The RNI pulse continues down the timing chain and sets K112/113 but cannot enter H110 because the arithmetic section is busy.

Now, assume an interrupt signal occurs after the RNI pulse has reached this point. K008/009 permits the computer to accomplish an interrupt sequence while waiting for arithmetic to complete an operation.

If an interrupt is active N010 will set K120/121 (interrupt sync, see chapter 11) which will allow the interrupt to be processed. K008/009 would not have been set at V009 time if the arithmetic section were not busy, therefore N010 would have tested for an interrupt active only once. Figure 206 shows a flow chart of interrupt recognition.

END RNI



(1) Initiate RNI, request the bus system.

(2) Update P.

(3) Request storage.

(4) Reply from storage.

(5) Release bus system.

(6) Enable the instruction to the F register.

(7) Decode instruction.

(8) End RNI.

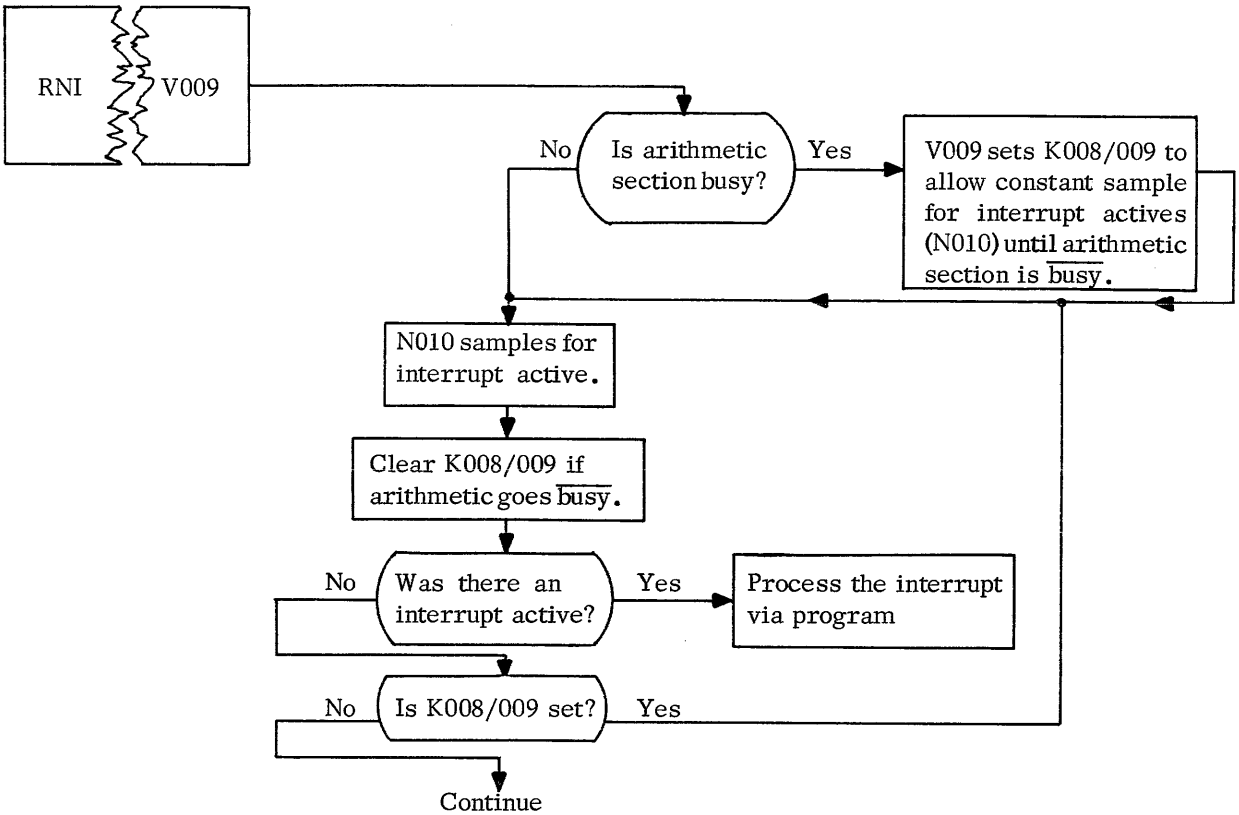


Figure 206. Interrupt Recognition

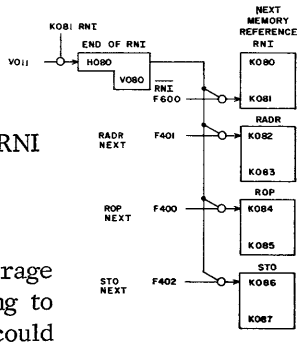


Figure 207. End RNI

V080: Progress to next storage reference according to (F1). NOTE: V080 could start the arithmetic section if necessary -- for example, RNI to RNI progressions such as shift or enter operations.

The arithmetic section would also be started if address modification were necessary. Address modification is called FADR (form address) and is discussed in chapter 12.

V080 may initiate a trap sequence if the instruction just read up is translated as floating point, double precision multiply or divide, or a BCD instruction (55 to 70), and the respective option is not present in the system.

The trap sequence does not execute these instructions. It performs an operation similar to return jump to a software routine that simulates the execution of the instruction. The trap sequence is discussed in chapter 14.

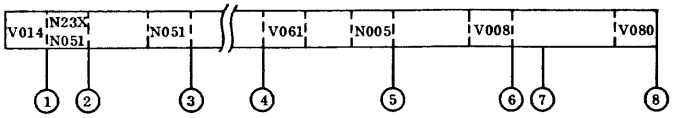
SUMMARY

This completes your excursion through the RNI sequence. So that you can see the forest in spite of the trees let's review the four major functions of RNI.

1. Read an instruction word from storage and place the instruction in F register.
2. Insure proper updating of P register.
3. Sense for all types of interrupts.
4. Start/stop.

REVIEW

Scan the by-now-familiar RNI sequence diagram and fill in the blanks without referring back to the text.



(1) _____

(2) _____

(3) _____

(4) _____

(5) _____

(6) _____

(7) _____

(8) _____

Now refer to figure 208, a different representation of RNI sequence. We will supplement this figure in the chapters on RADR, ROP, and STO so that you can visualize the areas unique to each sequence and common to all.

The next couple of pages provide a handy list of problems to fill your leisure hours.

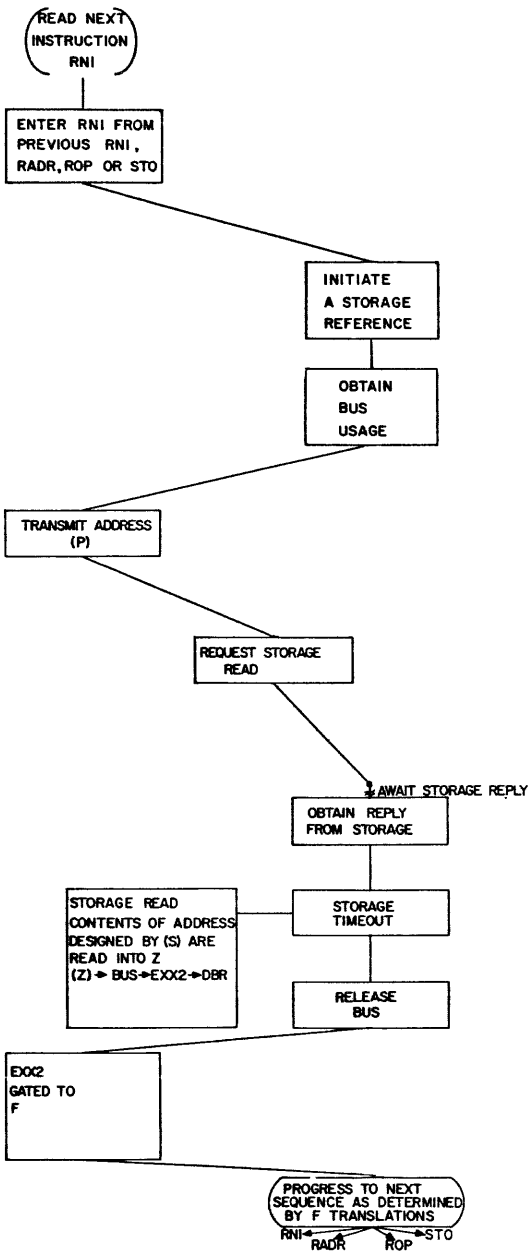


Figure 208. RNI Big Picture

1. Define the purpose and application of block P FF, (K200/201). _____

2. Why must the P1 to P2 transfer always follow gating of F to P? _____

3. During the RNI sequence at address 15000, what ranks of inverters, transmitters, and receivers provide a path from Z register to F register?

	Z register
a. Storage	
b. Transmitters	_____
c. Receivers	_____
d. Inverters	_____
e. Main control	_____

4. At best, how much time is necessary for an RNI sequence from initiation (V014) to end RNI (V080)? _____ Ø times, _____ usec.

5. What two factors can delay the amount of time it takes to complete an RNI sequence?
   a. _____
   b. _____

6. What would the console indication be if a reply signal didn't come back to the processor? _____

7. List as many items as you can that could prevent the reply signal from returning and entering H000?
   a. _____
   b. _____
   c. _____
   d. _____
   e. _____

8. What is the purpose for releasing the bus system at V007 time rather than at the end of RNI? _____

9. After executing a 00.0 15000 instruction located in memory at location 10077, what are the contents of the following registers?
   (F) = _____ (P) = _____

10. What registers, if any, will instruction 00.0 15000 affect besides P and F? _____

11. Would less time be required to execute a selective jump instruction if the switch were set than if it were not set? _____

12. At best, the selective jump instruction requires _____ Øtimes, _____ usec, to accomplish its execution.

13. Is the arithmetic section used for the $B^b \neq 0$ decision of the index jump instruction? _____

14. Does the increment / decrement of the 02.X instruction lengthen its execution time? _____

15. At best, the index jump instruction requires _____ Øtimes, _____ usec, to accomplish its execution.

16. Does execution of a 03.5 XXXXX require starting of the arithmetic section to accomplish its execution? _____

17. Does the 03.5 XXXXX instruction take more time for execution than the 03.2 XXXX instruction? _____

18. At best, the 03.X instruction requires _____ Ø times, _____ usec, to be executed.

The following problems are imaginary malfunctions in the RNI sequence. Study each carefully and, after analyzing the indications, list areas that should be checked. Use Logic Diagrams, 3200 Command Timing Charts, and this manual for references. NOTE: Consider each of the problems separately.

19. Indications:
   a. F register does not get cleared of the old instruction.
   b. Z register in the storage module shows that a new instruction was read out of storage.
   c. The following flip-flops remain set during the RNI sequence:
      K000/001 (request bus)
      K080/081 (RNI); RNI console indicator stays lit
      K002/003 RNI lockout FF remains set
      K010/011 (main control priority)
      K212/213 (priority 3)
      K116/117 (storage request)
      K122/123 (initiate storage request)
      K012/013 (enable data bus)

   There are at least four possible circuits that could give these indications. List each one with the reason why it could be at fault.

A. _____ because _____

B. _____ because _____

C. _____ because _____

D. _____ because _____

CLUES: Did you request storage?
Did you get a reply back from storage?
If the reply did get back, did the processor recognize it?

20. Indications:
    a. F register does not get cleared of the old instruction.
    b. Z register in storage shows that the new instruction was read out.
    c. The following flip-flops remain set during the RNI sequence:
       K080/081 (RNI)
       K002/003 (RNI lockout)
       K010/011 (main control priority)
       K212/213 (priority delay)
       K116/117 (storage request)
       K122/123 (initiate storage request)
       K012/013 (enable data bus)
    d. K000/001 (request bus) is clear.

The area of trouble that would give these indications must be between H_____ and V_____.

CLUES: Was a storage request initiated?
Did a reply come back from storage?
Did the processor acknowledge the reply?
If the reply did get back and the processor acknowledged it, how far did it get?

21. Indications:
    a. F register receives the new instruction.
    b. The following flip-flops are clear:
       K010/011 (main control priority)
       K012/013 (storage request lockout)
       K122/123 (initiate storage request)
       K116/117 (storage request)
       K212/213 (priority 3)
    c. K002/003 (RNI lockout) is set.

The area of trouble must be from H_____ to V_____.

CLUE: Storage was requested and the reply signal must have come back with the information or else F would not have received the new instruction.

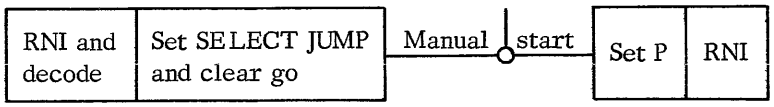
22. Indications:
    a. RNI FF stays set.
    b. The old instruction stays in F register.
    c. Z register in storage contains the data from the last storage reference.
    d. The following flip-flops stay set during RNI sequence:
       K000/001 (request bus)
       K002/003 (RNI lockout)
       K010/011 (main control priority)
       K122/123 (initiate storage request)
       K116/117 (storage request)
       K212/213 (priority 3)
       K012/013 (storage request lockout)

The area of trouble must be from H_____ to B _____. (No clues for this problem.)

NOTE: The times given for these answers were figured from the simplified logic page included in this chapter. If your answer does not agree exactly with the answer given here, consider the time for transmitter and receiver cards, flip-flop settings and clearings, etc.

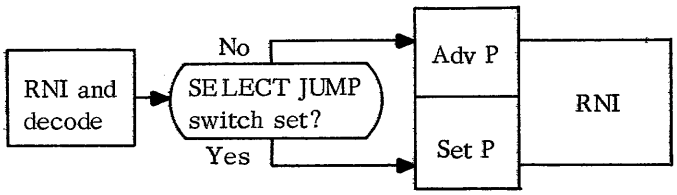
Some instructions are executed during RNI to RNI. A few of these are shown on this page with their flow charts. Read the flow charts and trace the instructions through on the logic diagrams, using the 3300 Command Timing Charts as a guide.

00.0 Unconditional stop and jump to "m" upon restart:



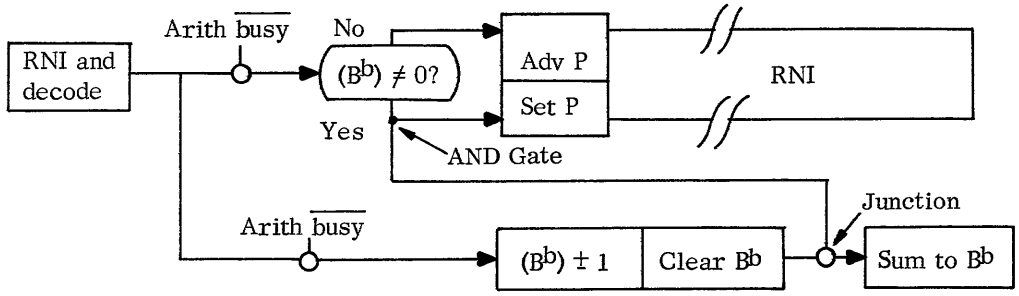
NOTE: This instruction is executed regardless of the state of the arithmetic section.

00.1 to 00.6 Selective jump:



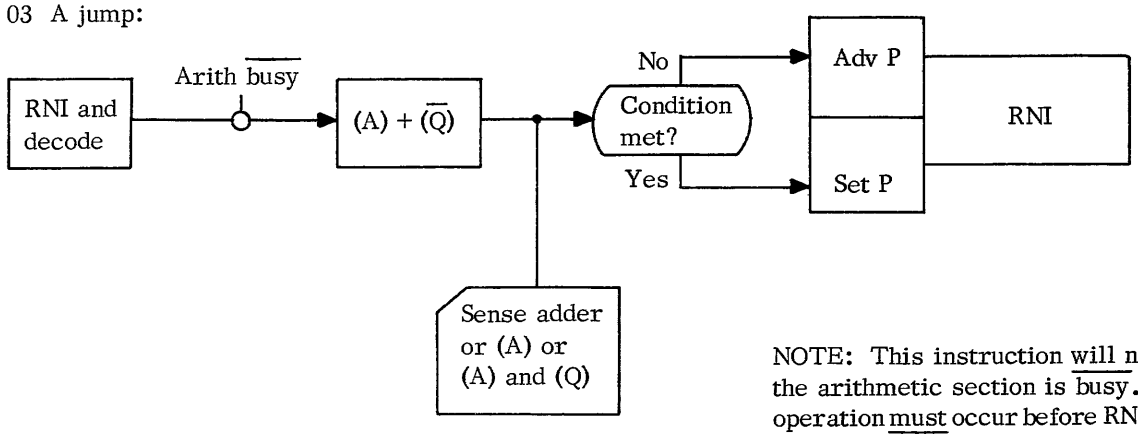
NOTE: This instruction is executed regardless of the state of the arithmetic section.

02 Index jump:



NOTE: This instruction will not execute unless the arithmetic section is busy. Notice that the arithmetic operation occurs parallel to RNI.

03 A jump:



NOTE: This instruction will not execute unless the arithmetic section is busy. The arithmetic operation must occur before RNI may be initiated.

SELF-EVALUATION QUIZ ON CHAPTER 7

TRUE OR FALSE:

1. P register may be updated only during RNI.

2. Updating P register by 2 adds 4 Ø times to instruction execution.

3. Setting K000/001 gives main control priority of the bus system.

4. The block advance P FF is set by a master clear to insure that the first RNI is at the address in P.

5. The execution of a jump adds 4 Ø times to instructions for which the jump is optional.

TRUE OR FALSE OR FILL IN BLANKS

6. The P3 register holds the last jump address (Program State).

7. If the system is in Non-executive mode, the upper 3 bits of the S bus, as sensed at the output of the T cards, will be _____.

8. The _____ rank of inverters has 2 inputs to the T6XX cards so that _____ to _____ address conversion may be accomplished.

9. The CPU sends an absolute address directly to storage if the system is in Executive mode and the Multiprogramming module is present.

10. The function of the Breakpoint Lockout FF is to insure that a restart may be accomplished after a breakpoint stop.

11. The reply from storage starts the main timing chain and indicates in all cases that storage has acknowledged the storage request.

12. F1 to F2 transfer is necessary because the Arithmetic Section can run independently.

13. Main control receives $\overline{\text{Data}}$ from storage during RNI.

14. The transfer of EXX2 to F1 occurs for every RNI.

15. Interrupt is sensed every V010 time if the RNI FF is set.

Score yourself:
Missed none or one? Excellent!
Missed two? Below average
Missed three or more? Too bad--you've failed!

CHAPTER 8

READ ADDRESS SEQUENCE

DESCRIPTION

The read address (RADR) sequence is used when indirect addressing, used only with instructions that use execution address m, is required. After an RNI (and possibly indexing), an RADR sequence is used to procure the execution address of an instruction. Several levels (or steps) of indirect addressing may be used to reach the execution address. Indirect addressing is specified for applicable instructions when bit 17 is a 1. Figure 209 shows the indirect addressing routine for the 3300.

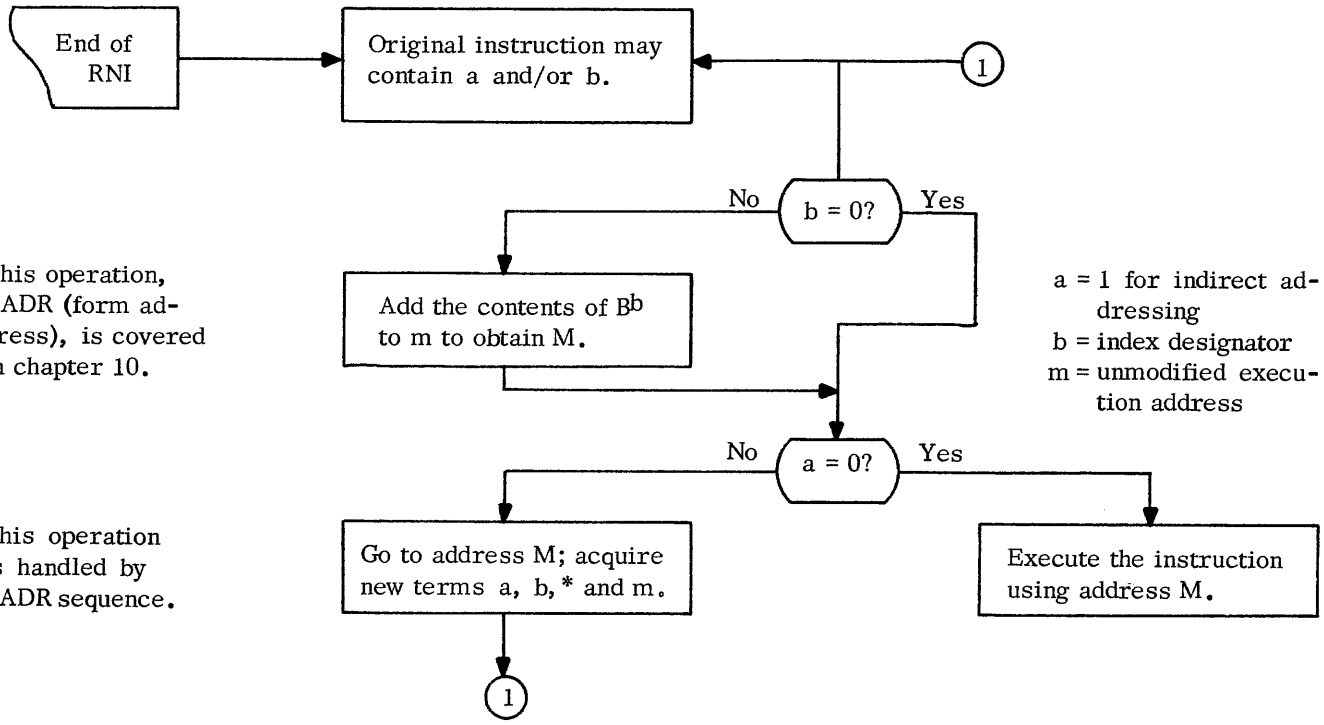
Figure 210 is a system block diagram. The heavy black line originating at F register indicates the transfer path for the lower 15 bits of the address. The upper 3 bits of the address will be 0's if the system is non executive or in monitor state. The upper 3 bits of the address will be (ISR) if in program state. Originating at Z register of module 0 represents transfer path for data from storage.

Steps in the RADR sequence are:

1. At the end of RNI request bus priority.
2. Obtain bus priority.
3. Transmit a read signal. Transmit storage address
4. Transmit storage request. Wait for selected storage module to reply. (Waiting time varies depending on memory module status.)
5. Receive and resync storage reply; use this signal to start main timing chain.
6. Clear lower 18 bits of F and clear DB register.
7. Gate data from the bus to F and DB register.
8. Test for interrupt.

9. End RADR. Set the correct storage sequence control FF to execute the next appropriate sequence. If another step of indirect addressing is required, RADR remains set and another RADR sequence is executed.

RADR is nearly identical to RNI. The flow of information is identical. The only differences are that the storage address comes from F register instead of P, and information from storage goes to only the lower 18 bits of F instead of all of F.



This operation, FADR (form address), is covered in chapter 10.

a = 1 for indirect addressing
b = index designator
m = unmodified execution address

This operation is handled by RADR sequence.

*The b designator is not changed on a 47 or 54 instruction.

Figure 209. Indirect Addressing Routine

DETAILED TIMING

To read an address from storage, several operations must occur:

1. Request bus system. (F lower 15 bits is the address to be referenced.)
2. Obtain bus priority.
3. Transmit address in the lower 15 bits of F to storage via the S bus.
4. Do not transmit write signal. (Absence of a write signal is a read signal.)
5. Transmit storage request.
6. Do not test for breakpoint stop.
7. Wait and resync reply from storage.
8. Clear request bus.
9. Time out access to allow storage to read and place word on data bus.
10. Sense for interrupt
11. Clear "F" lower
12. Gate data bus to "F" lower
13. Test for stop (cycle step mode, or instruction step and end of instruction, or STOP pressed and end of instruction).
14. Advance storage reference controls tc RNI+RADR + ROP + STO.
15. Progress to next storage reference if not stop.

NOTE: The lower 18 bits of F are cleared and EXX2 is gated to these bits for all instructions requiring indirect addressing except 47 and 54. For these two instructions only, bits 0-14 and 17 are cleared and replaced with respective bits from memory.

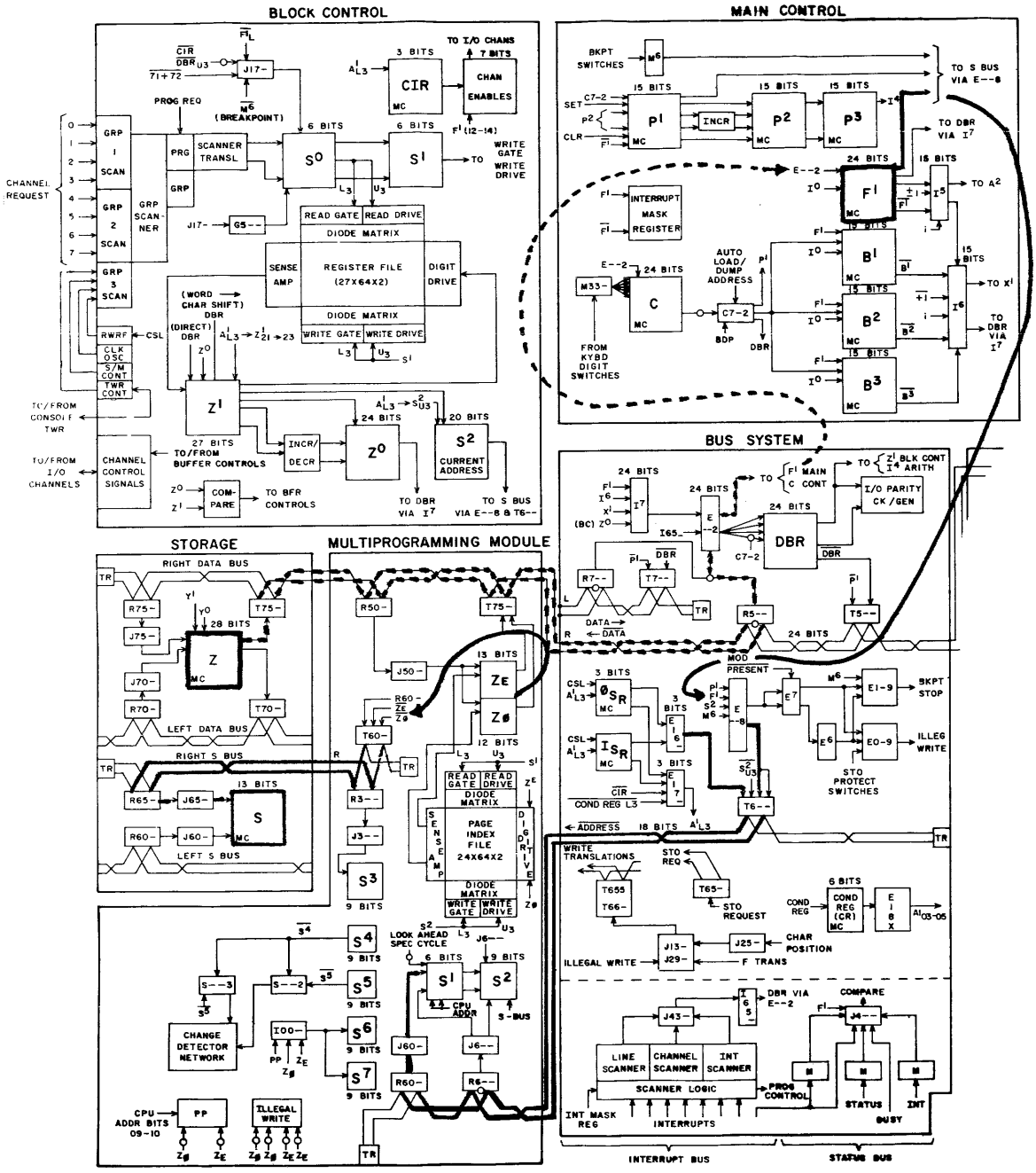


Figure 210. Data Flow for RADR

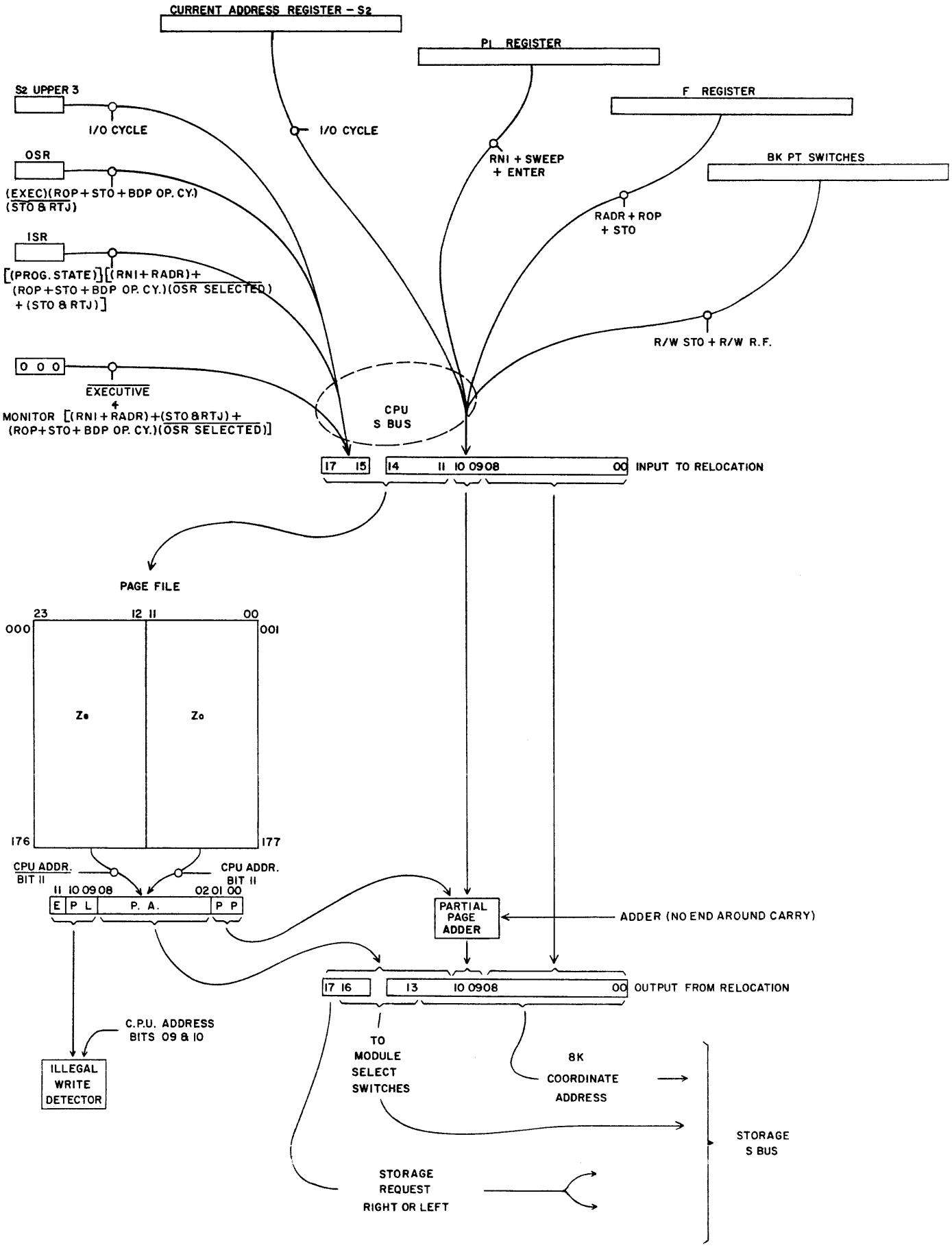


Figure 211. Address Flow

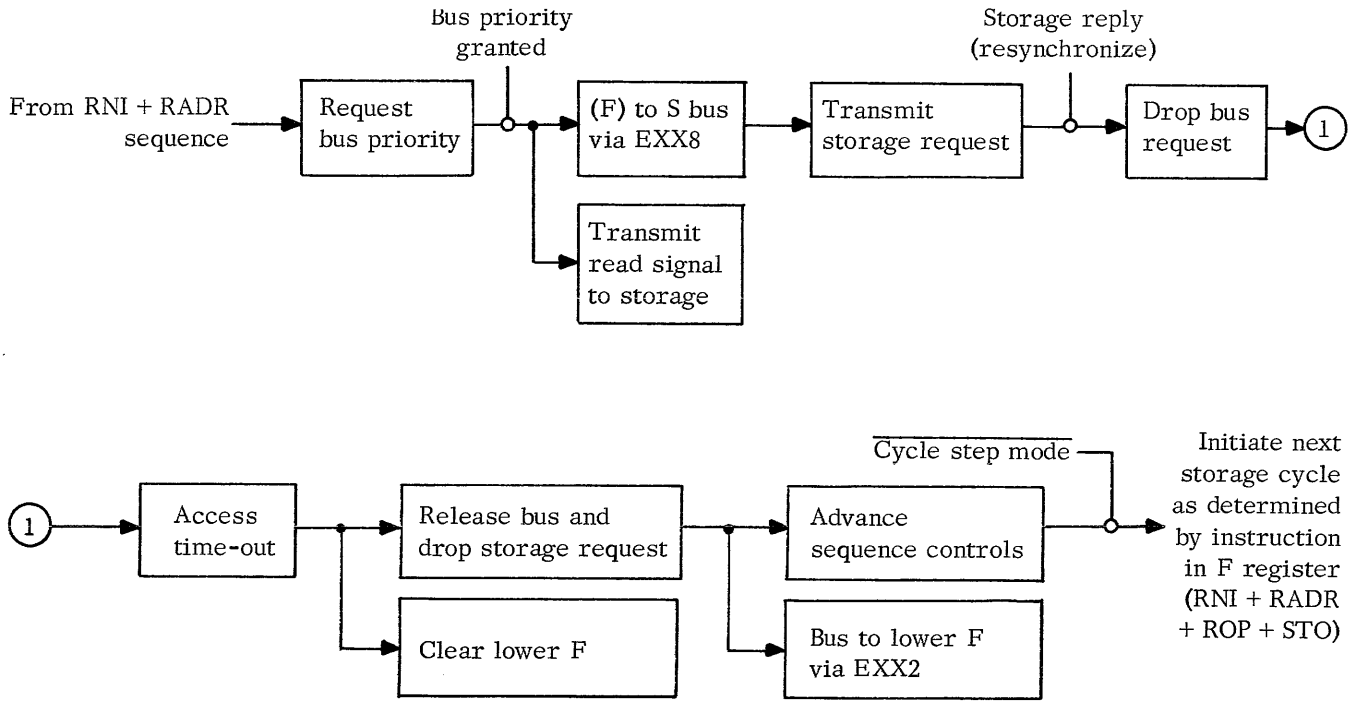


Figure 212. Flow Chart of RADR Sequence

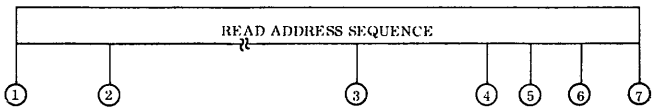
Figure 212 graphically shows the RADR sequence of operation.

READ ADDRESS STORAGE REFERENCE

This sequence will be used only for the instruction in which bit 17 is the addressing mode designator and bit 17 = 1.

READ ADDRESS SEQUENCE

The RADR sequence is used only for indirect addressing. It has no other purpose or function. It does not accomplish the execution of any instructions.



(1) Start RADR, request bus system

(2) Request storage

(3) Reply from storage

(4) Release bus system

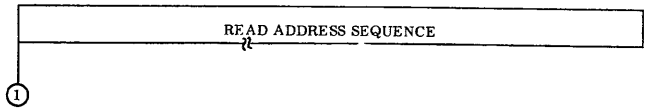
(5) Enable data bus to lower F register

(6) Redecode instruction

(7) End RADR

Figure 213. Major Timing of RADR Sequence

START RADR



(1) Start RADR, request bus system

If Indexing not required

V380 (end of RNI pulse): clear K080/081, set K082/083 (RADR), set K000/001 (Request bus).

If Indexing required

V380 (end of RNI pulse): clear K080/081, set K082/083 (RADR), set K100/101 (start Arith 1).
N659/V109: input H100, input H500
V500/V100: set K110/111 (enable I^0 to F)
V501:
V502:
V503: input to H110
V110/V210: set K000/001 (Request bus), clear K112/113 (no Index)

K000/001 (Request Bus Flip-Flop)

The S bus and Data bus circuits are shared by main control and block control sections of the CPU. Either

section requiring the bus system must request priority. Priority will be granted when the bus system is not in use.

Main control and block control have equal priority when requesting the bus system in that neither can take it away from the other, and each has first chance to get the bus system after the other releases it.

K000/001, Request Bus FF, serves as the main control request bus flip-flop.

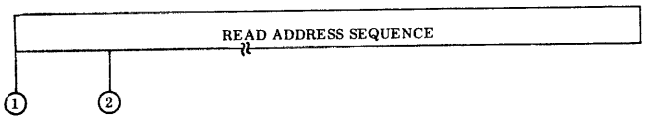
N051: Set K010/011 (Main Control Priority) if block control has priority.

If K209 = 1: Repeat this test at the next N051 time because block control has priority.

If K208 = 1: Set K010/011 (Program Control Bus Priority FF). Gate F_I to EXX8 to T6XX to S bus, placing a 15-bit address on the S bus. Upper three bits of the address will be zeros or (ISR). T655 ($\overline{READ}$) is off.

Setting of K010/011 enables the (F_L) to the S bus. (Note that if block control has some of the bus system, K010/011 will not set until block control loses priority; K208/209 cleared.)

REQUEST STORAGE



(1) Start RADR, request bus system

(2) Request storage.

N050: Input to H117

V117: Set K116/117 (Storage Request). Transmit a storage request

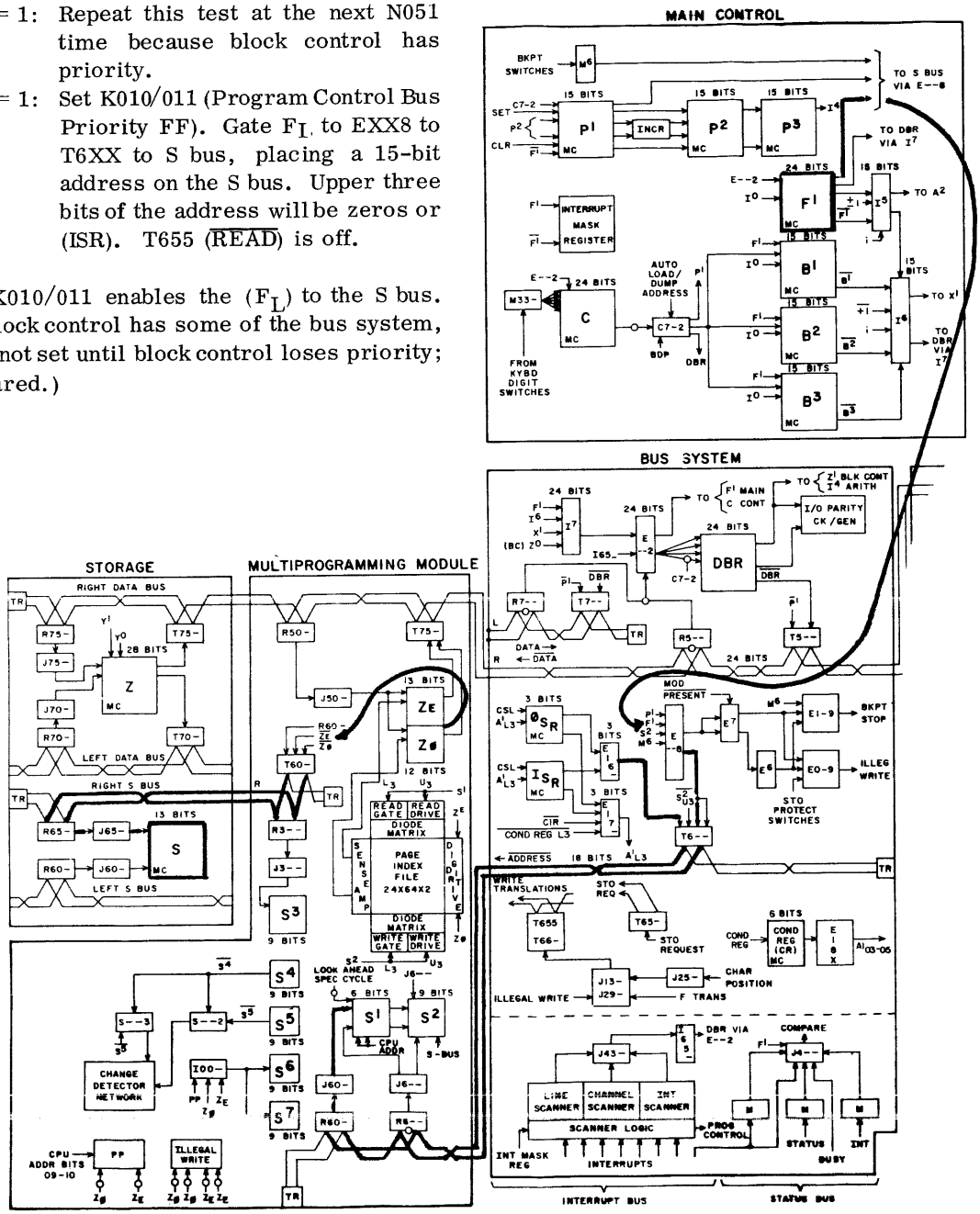


Figure 214. Address Transmission for RADR

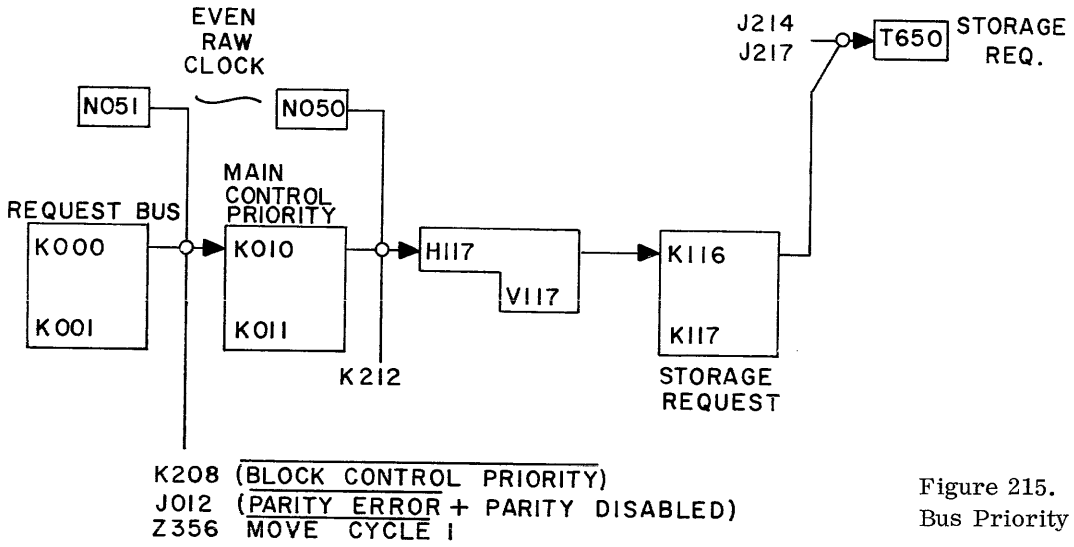


Figure 215.
Bus Priority Logic

Main control is now waiting for a reply from storage. There is no breakpoint comparison in RADR sequence.

N050: Input to H115.

V115: Set K012/013 (request lockout, p. 2-9) and K004/005 (word address mode, p. 2-71).

V116:

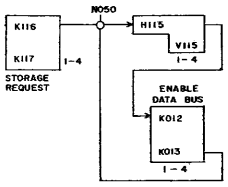


Figure 216.
Storage Request Logic

Review what has occurred in RADR sequence so far.

1 _____

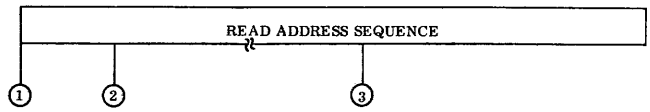
2 _____

Main control is hung up. There is nothing that it can do until the reply comes back from storage.

The processor must wait for a reply signal from storage. The signal indicates that storage is processing the request and will soon have the information on the data bus.

Item 3 of RADR sequence is _____.

REPLY FROM STORAGE



(1) Start RADR, request bus system

(2) Request storage.

(3) Reply from storage.

R555 (storage reply, p. 1-2)
V061 (resynchronized storage reply)

The reply signal (R555) indicates that the storage module is processing the request and, after access time, the instruction will be available on the data bus.

Resync circuit is discussed in chapter 7.

V000-V005: Clear K000/001 (request bus) to give the storage module access time to store the information from the data bus.

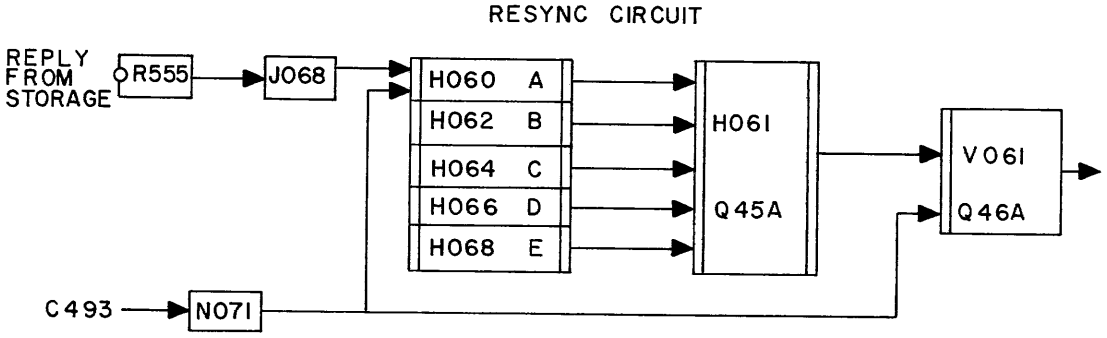


Figure 217. Reply Resync Logic

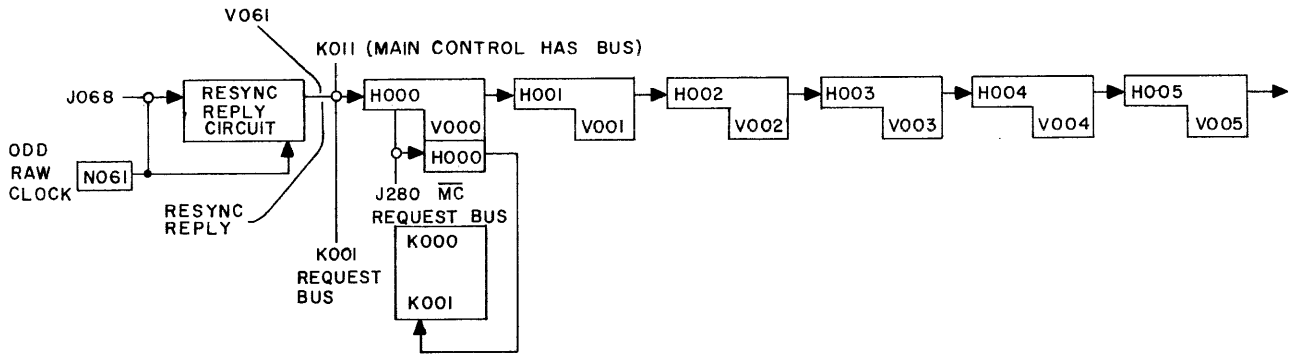


Figure 218. Clear Bus Request

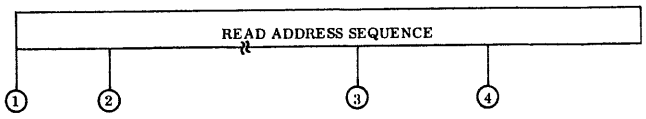
As soon as the reply signal from storage is resynchronized access time starts. One of the first things to occur after the reply signal has been received by the processor is clearing of the request bus FF. Clearing the request bus FF neither clears nor releases the bus. Main control needs the bus system and requests it via K000/001 (request bus FF). If the bus system is not busy main control is given priority and sends a request for storage. Storage acknowledges the request by sending back a reply signal indicating that the request has been received and is being processed. However, the request for storage must be completely processed before storage can transmit the requested information. Essentially, one might say that main control has the bus, thus does not need to request it, so K000/001 is cleared. Main control must still use the bus system, thus it has not been released.

V007, N005: Clear K010/011 (Main Control Priority), clear the DB register, set K494/495 (sense interrupt during RADR); input to H410; clear K212/213 (priority 2); clear K116/117 (storage request), clear K012/013 (request lockout); clear F1 (lower 18 bits only), input H200.

After access time due to the reply signal has elapsed and as the information from storage is taken from the bus system, then main control will release the bus system.

It is important that the bus system be released as soon as possible. Remember that block control and main control use the same bus system. If the other control area needs the bus system, the sooner it can be released, the sooner it can be used. Further discussion of this time-saving operation is supplied in chapter.

RELEASE THE BUS



(1) Start RADR, request bus system

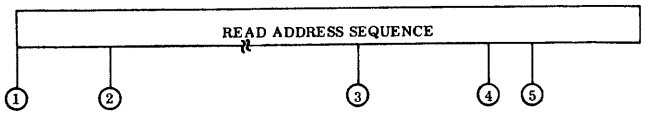
(2) Request storage.

(3) Reply from storage.

(4) Release bus system

V006: Input to H401; input to H201.

ENABLE DATA BUS TO LOWER F REGISTER



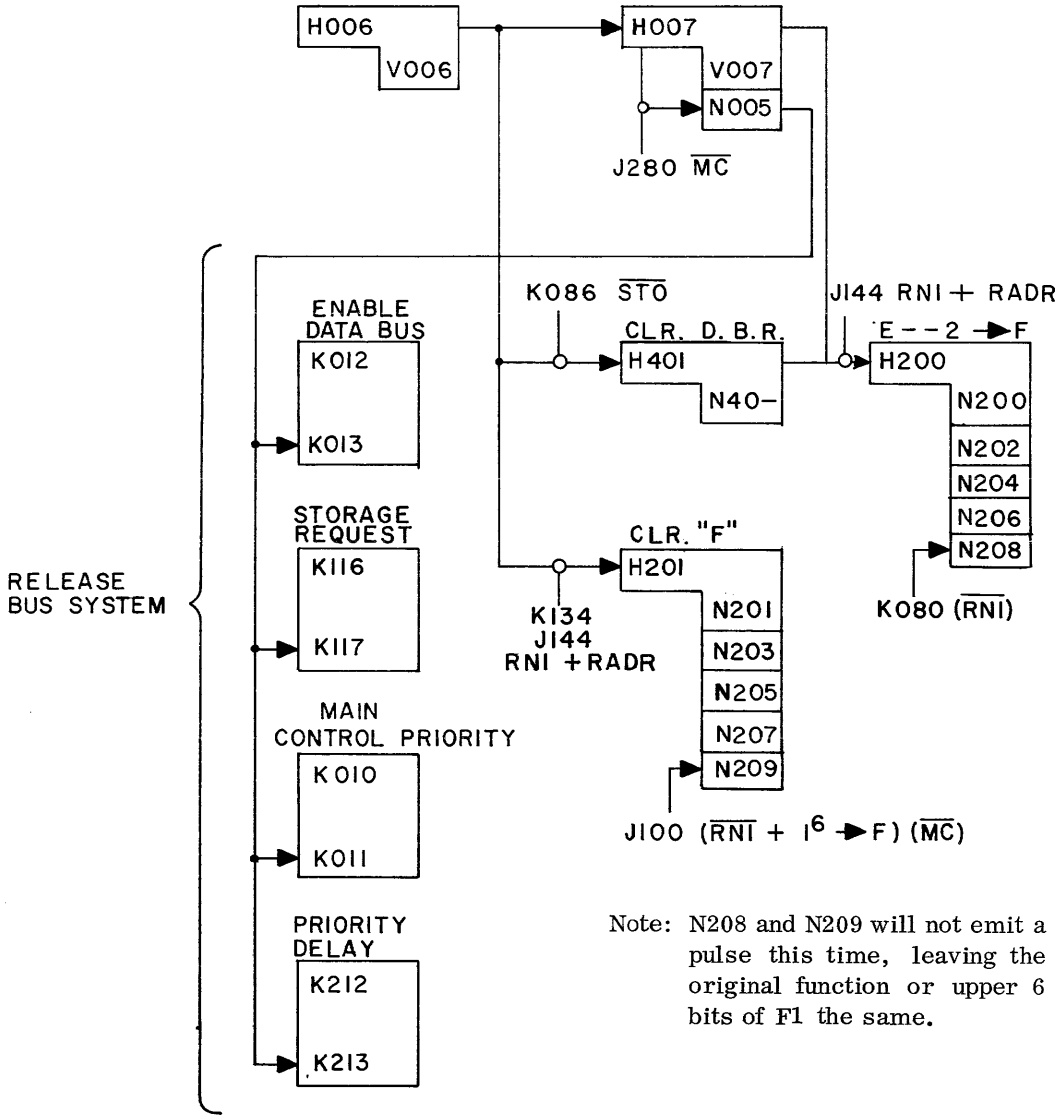
(1) Start RADR, request bus system

(2) Request storage.

(3) Reply from storage.

(4) Release bus system.

(5) Enable data bus to lower F register.



Note: N208 and N209 will not emit a pulse this time, leaving the original function or upper 6 bits of F1 the same.

Figure 219. F Register Enables

V008: EXX2 to the lower 18 bits of F1; EXX2 to DB register; test interrupt

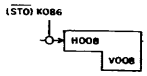


Figure 220. V008 Time

The data bus to receivers to EXX2 path is maintained by a static enable. Note on the block diagram that EXX2 is in the flow path from storage to main control as well as to standard arithmetic. EXX2 could actually be considered the first branching point for data flow. In this case EXX2 feeds F register. You will note EXX2 feeding DB register also, but the transfer is not used for this sequence.

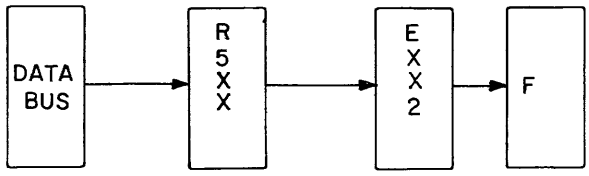


Figure 221. RADR Data Path

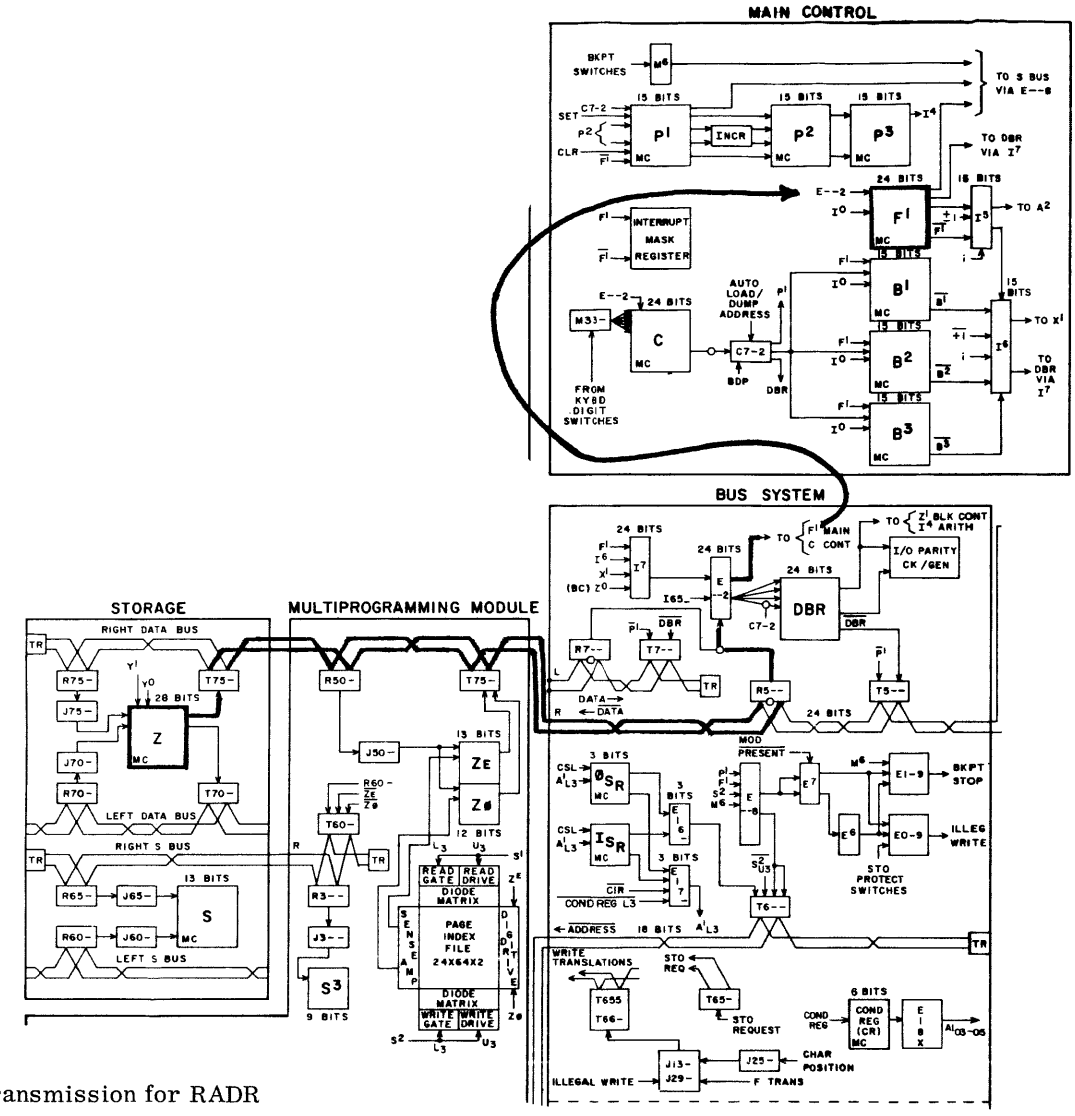
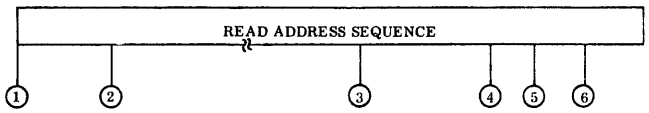


Figure 222. Data Transmission for RADR

REDECODE INSTRUCTION



(1) Start RADR, request bus system

(2) Request storage.

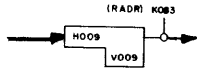
(3) Reply from storage.

(4) Release bus system.

(5) Enable data bus to lower F register.

(6) Redecode instruction.

V009: Input to H082, test stop.

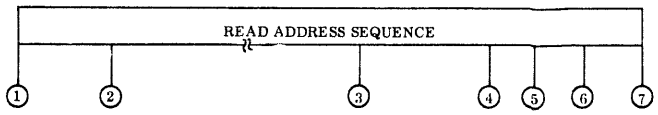
V008: 

H082

Figure 223. End RADR Time

Redecode time is less than decode time from RNI because function translation is static; the only new bit that can affect the translators is bit 17. Bit 17 is translated rapidly (fewer inverters) and will be translated by V082 time.

END RADR



(1) Start RADR, request bus system

(2) Request storage.

(3) Reply from storage.

(4) Release bus system.

(5) Enable data bus to lower F register.

(6) Redecode instruction.

(7) End RADR

V082: Progress to the next storage reference cycle according to the (F1).

V082 will set K112/113 (no index) which will allow another sequence to be initiated. Which sequence is initiated depends on which sequence control FF is set. RADR FF will not be cleared if another RADR sequence is required. After RADR sequence the sequence progression can occur in any of the following ways:

RADR to RADR if another level of indirect addressing is required; RADR to RNI if the instruction was initially a 01.4; RADR to ROP or RADR to STO for all instructions which use ROP + STO and can use RADR.

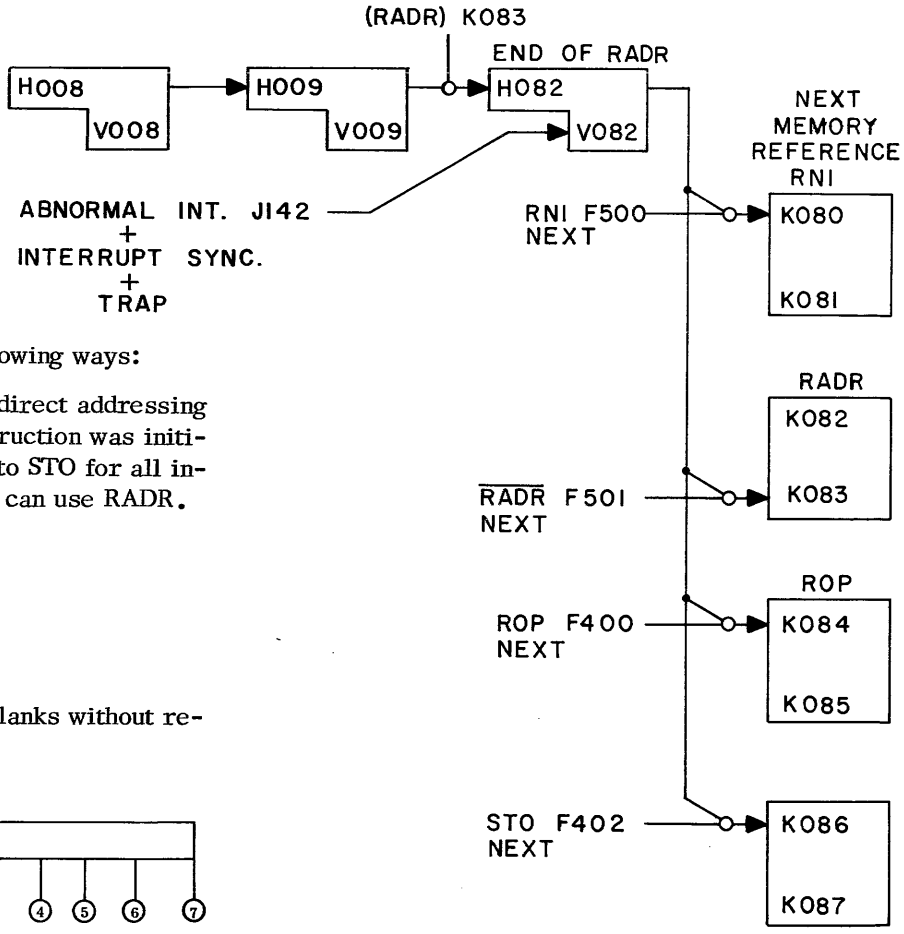
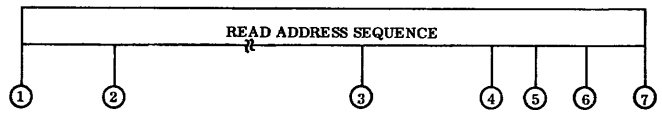


Figure 224. Sequence Controls Advance

REVIEW

You should be able to fill in the blanks without referring back to the text.



(1) _____

(2) _____

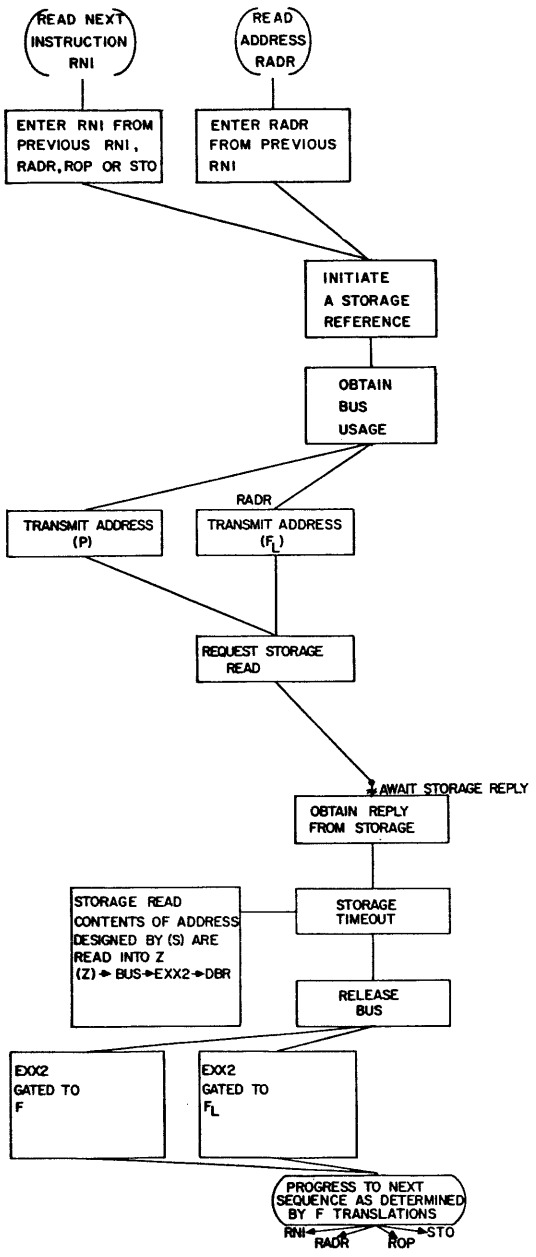
(3) _____

(4) _____

(5) _____

(6) _____

(7) _____



Notice the similarity between RNI and RADR.

Figure 225. RADR Big Picture

SELF-EVALUATION QUIZ ON CHAPTER 8

TRUE OR FALSE:

1. RADR sequence is available for all instructions in the 3300 repertoire.
2. RADR sequence may be entered only from RNI sequence.
3. If the system is in Program State, the upper three bits of address are supplied by ISR.
4. For a 47 or 54 instruction, RADR does not modify bits 15 and 16 of F.
5. An illegal write signal may never occur during RADR.
6. Interrupts are not sensed during RADR at V010 time.
7. During RADR, storage places only the lower 18 bits of the selected address on the data bus.
8. If indexing and RADR are both specified by the instruction in F, indexing will occur first.
9. Indexing will never occur after RADR.
10. More than one level of indirect addressing is permissible with the 3300.

Score yourself:
You should not have missed a single one.
If you missed one, you probably were careless.
If you missed two or more, you'd better study the chapter again.

CHAPTER 9

READ OPERAND SEQUENCE

INTRODUCTION

During the read operand sequence the lower 15 bits of address is obtained from F lower 15 while the upper 3 bits may be obtained from OSR, ISR, or they may be 0s depending on the Mode and State of the computer. In many cases the lower 15 bits of address may be modified by indexing. For more information about specific instructions refer to the 3300 Reference Manual.

The steps in ROP sequence are:

1. At the end of RNI or RADR or a previous ROP sequence, start ROP and request bus priority.
2. Obtain bus priority.
3. Transmit a READ signal and transmit the storage address via the CPU S bus.
4. Transmit a storage request.
5. Test breakpoint. Set Breakpoint Stop if BPO is selected and the breakpoint address equals the operand address. (Main control waits for the selected storage module to reply.)
6. Receive and resynchronize a storage reply; use this signal to start the main timing chain.
7. Stop if Breakpoint Stop is set.
8. Clear the DBR.
9. Gate data from the bus to the DBR.
10. Start arithmetic section. Arithmetic section then performs the operation specified by F code.
11. End ROP. Set the correct storage sequence control FF to execute the appropriate sequence next. If another ROP is necessary, as after the first cycle of a 48-bit precision load, ROP remains set and another ROP sequence is performed.

Figure 226 is a System Block Diagram. Note the heavy black line originating at the F register which illustrates the transfer path for the address to storage. Also note the heavy dashed line originating at the Z register of the Storage module. This line which terminates at the A register depicts the transfer path for data if we were doing a Load A instruction.

Figure 227 is a simplified Address Flow diagram for the 3300 if relocation is to take place.

DETAILED TIMING

ROP is used for load, add, subtract, multiply, and divide instructions.

To read an operand from storage, several steps must occur:

1. Request Bus System (K000/001) (F Lower holds address of the operand.)
2. Obtain Bus Priority (K010/011).
3. Transmit Address in F Lower on S Bus to the Multiprogramming module.
4. Do not transmit Write signal (T655 = 0).
5. Transmit Storage Request (K116/117).
6. Test for breakpoint stop if BPO (K134/135).
7. Wait and resync reply from storage (V061).
8. Clear request bus (K000/001 by N000).
9. Time out access to allow storage to read and place word on data bus.
10. Clear DB register (V007).
11. Stop if BPO.
12. Gate data bus to DB register (bus to EXX2 to DB register). Set start arith 2.
13. Test for stop (cycle step mode or instruction step and end of instruction or STOP pressed and end of instruction).
14. Advance storage reference controls to RNI + ROP + STO.
15. Progress to next storage reference if not stop.

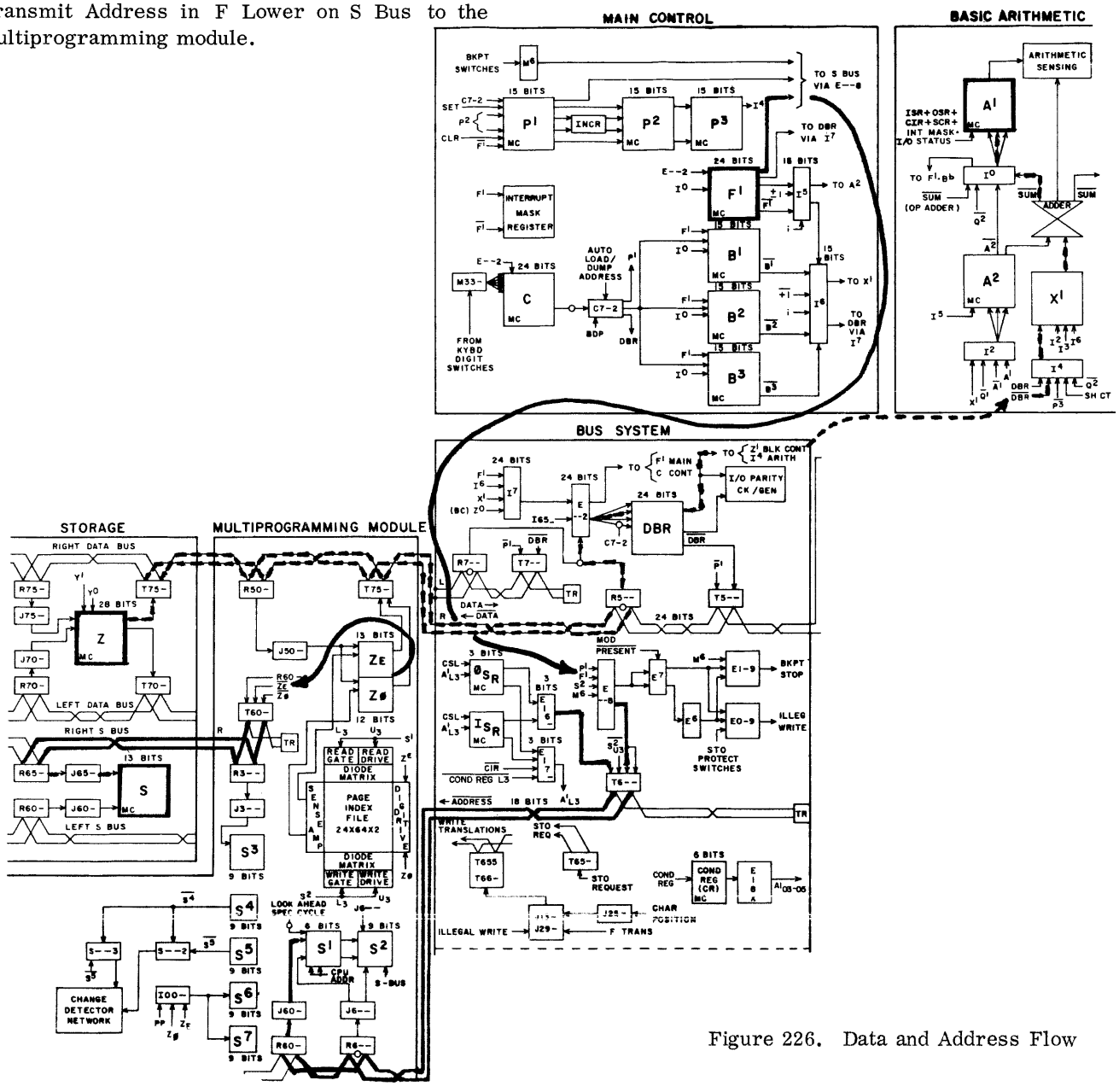


Figure 226. Data and Address Flow

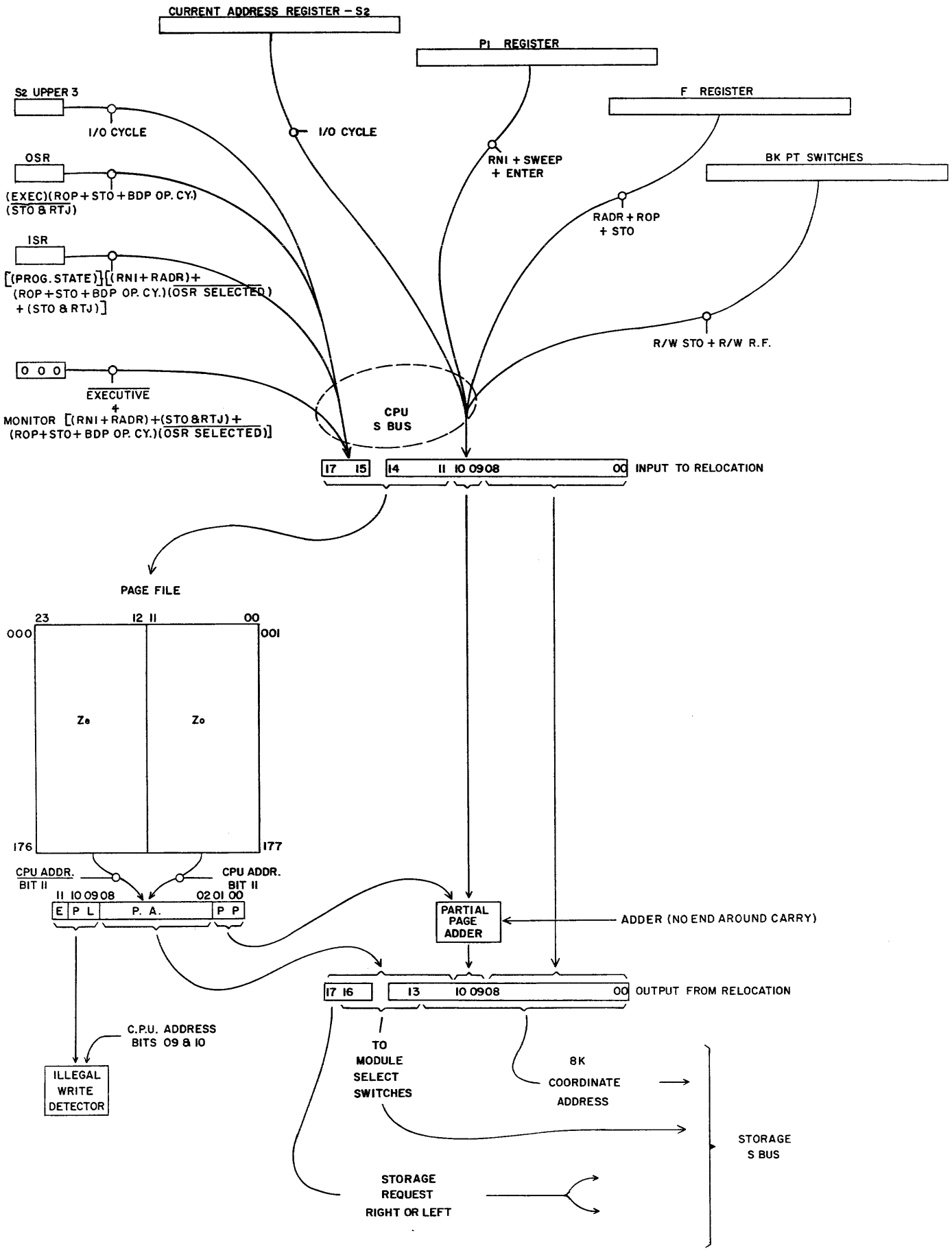


Figure 227. 3300 Address Flow

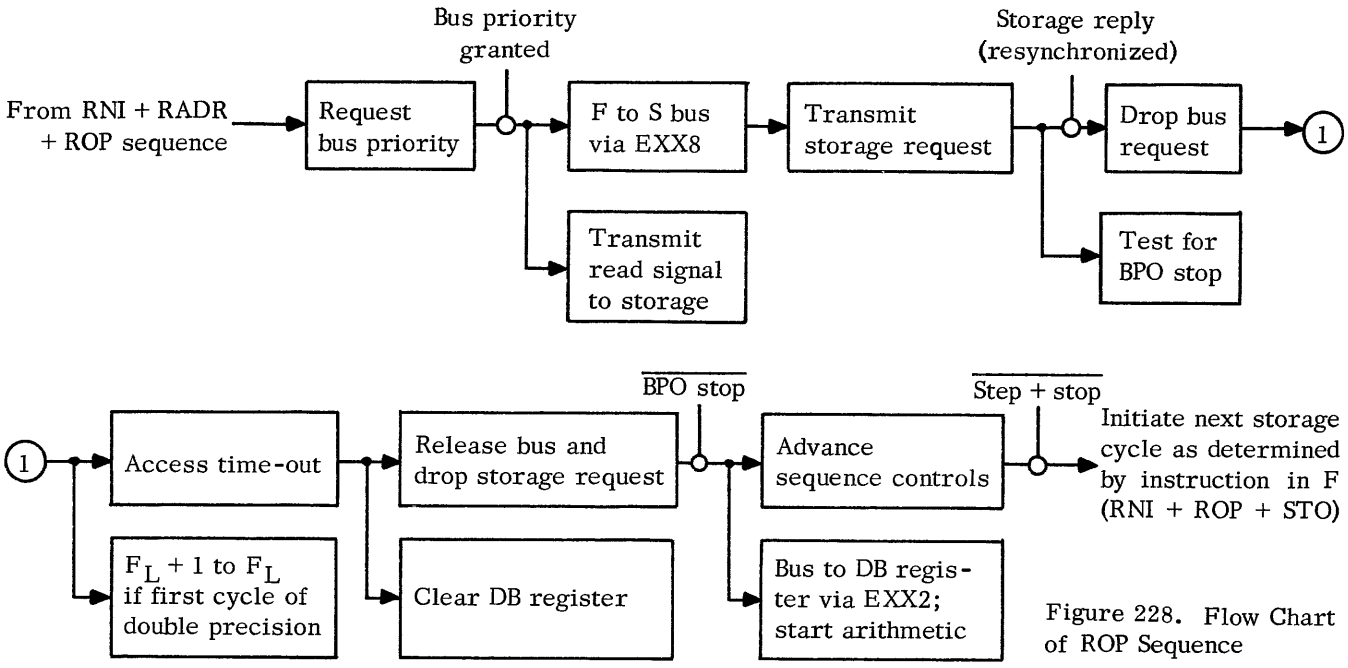
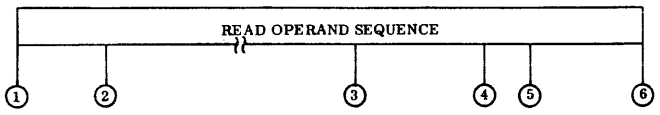


Figure 228. Flow Chart of ROP Sequence

Figure 228 graphically shows the ROP sequence of events.

During ROP sequence main control must acquire an operand from memory, place it in the DB register, and start the arithmetic section.

The major timing for an ROP sequence is shown in figure 229.



(1) Start ROP; request bus system.

(2) Request storage.

(3) Reply from storage.

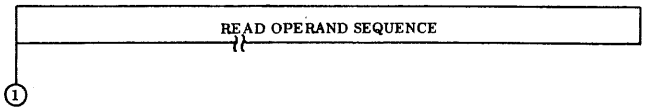
(4) Release bus system.

(5) Enable data bus to DB register; start arithmetic.

(6) End ROP.

Figure 229. Major Timing of ROP Sequence

START ROP; REQUEST BUS SYSTEM



(1) Start ROP; request bus system.

If (End RNI · $\overline{\text{Indexing}}$ · Go)
V380: End of RNI; set K000/001 (Request Bus) and K084/085 (ROP).

If (End RNI · $\overline{\text{Indexing}}$) + (End RADR · $\overline{\text{Indexing}}$)
V082: End of RADR.
V380: End of RNI. set K112/113 (No Index)
V109: Input H110 (If Arithmetic $\overline{\text{Busy}}$)
V210: Set K000/001 (Request Bus); clear K112/113.

If (End RNI · Indexing) + (End RADR · Indexing).
V380 or V082: Set K100/101 (Start Arith 1).
V659/V109: Input H100, Input H500.
V530/V100: Clear K100/101, Input H501, set K110/111 (Enable I^0 to F).
V501
V502
V503: Input H110
V210: Set K000/001 (Request Bus) and clear K112/113.

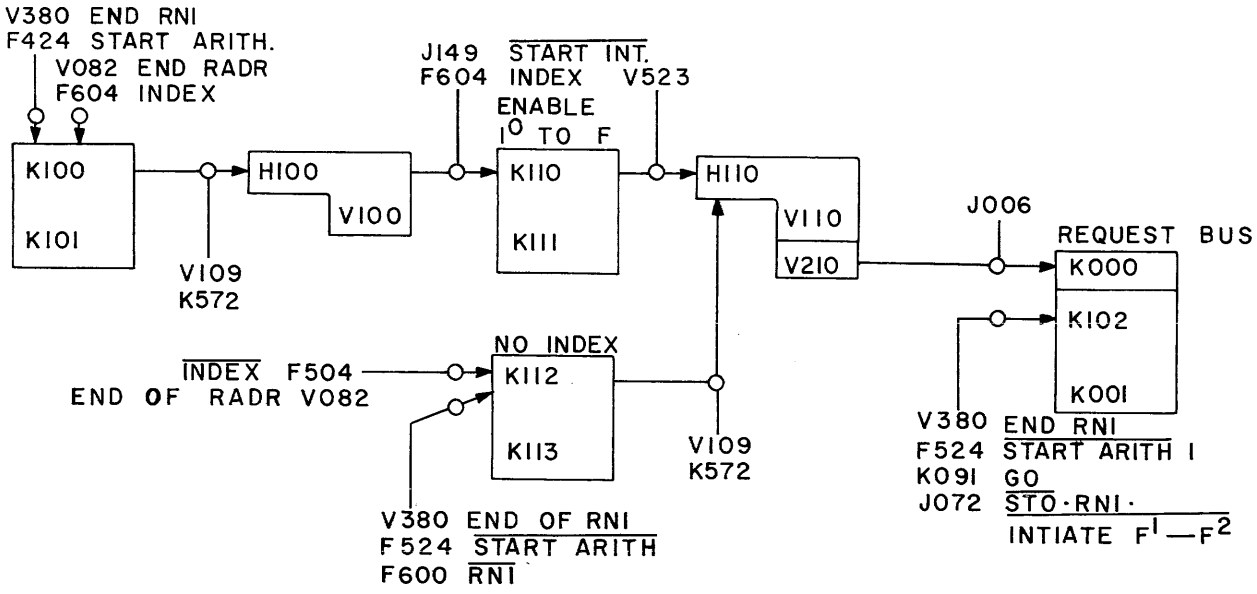


Figure 230. Sequence Progression from End of RADR

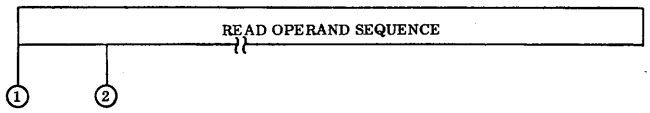
Input to H110 cannot be made as long as the arithmetic section is busy with a previous instruction. There is no way that the arithmetic section could be busy at this time if the progression is RNI to RADR to ROP.

N051: Set K010/011 (Main Control Bus Priority FF) if K208 = 1. If K209 = 1, repeat this test at the next N051 time. Gate F1 to EXX8 to T6XX to S bus, placing the lower 15 bits of address on the S bus. The upper three bits will be determined by Mode and State. Transmitter T655 ($\overline{\text{Read}}$) is off.

Setting of K010/011 indicates that main control has bus priority and enables the address to the S bus. The address on the S bus, now available to storage, is the address of the operand desired.

Note: If block control has bus priority, K010/011 will not set until the bus system is released by block control (K208/209 going clear).

REQUEST STORAGE



(1) Start ROP; request bus system.

(2) Request storage.

N050: Input to H117.

V117: Set K212/213 (Priority 2); Set K116/117 (Storage Request); Transmit request to Multiprogramming module.

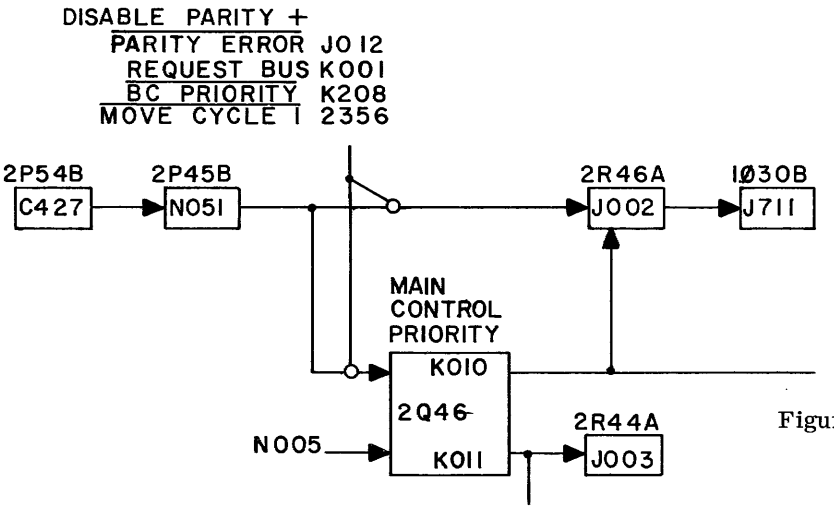


Figure 231. Bus Priority

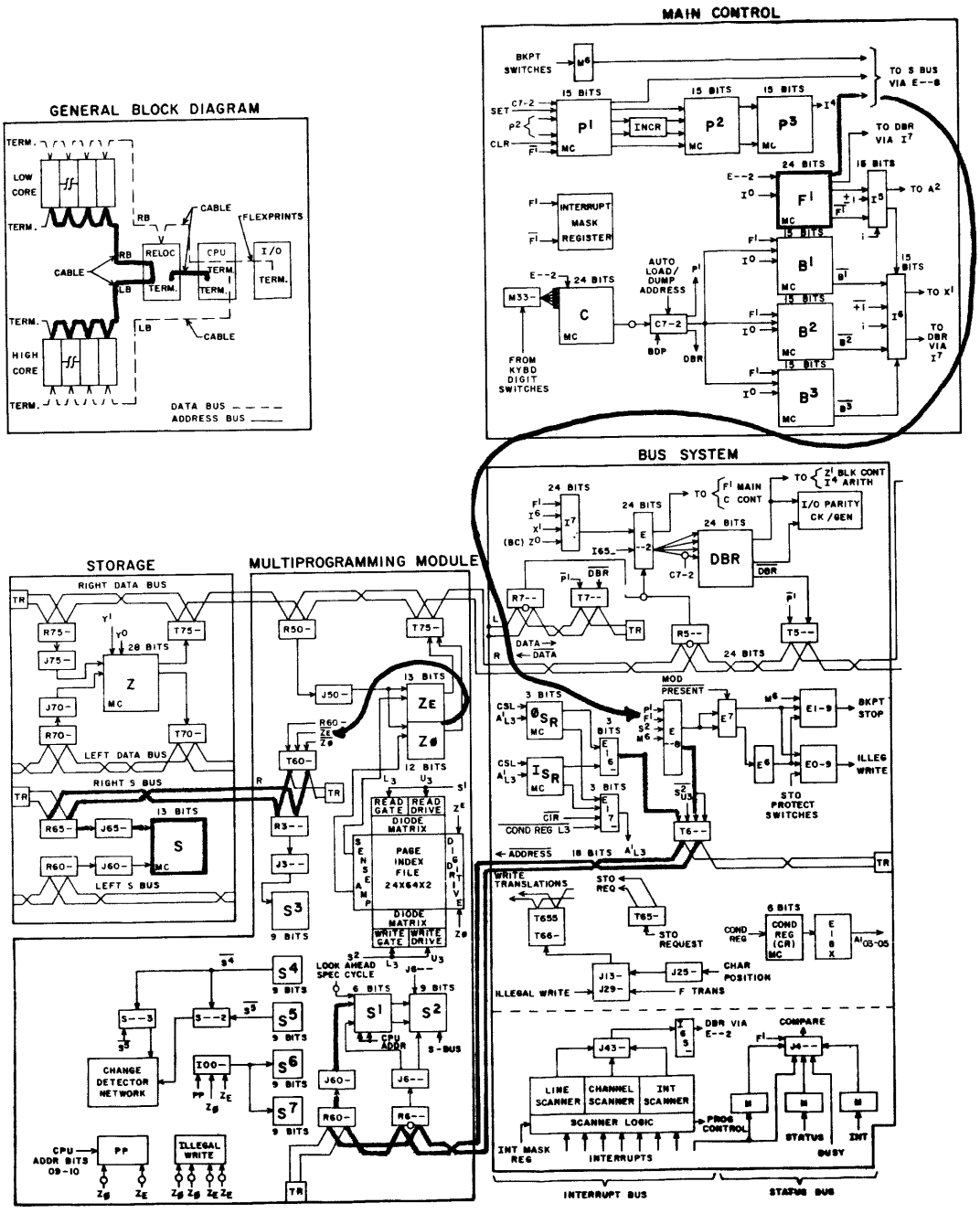


Figure 232.
Address Flow

Main control is now waiting for a reply from storage. Main control will compare the breakpoint address with the address transmitted to storage.

K012/013 (enable data bus) is set by V115. This flip-flop breaks the inputs to H115 and K212/213 breaks the input to H117, thus preventing multiple outputs from the control delays.

K004/005 (address mode) is set unconditionally during RNI and RADR because both sequences operate in word-addressing mode. During ROP it is possible for word-addressing or character-addressing modes to be used. This is determined by the instruction being executed (for example, whether it is load A or load A character). Setting or clearing of K004/005 occurs at V115 time of every sequence.

K134/135 (breakpoint stop) will set if the address on the S bus matches the breakpoint address and BPO has been selected, and if breakpoint lockout is not set. If K136/137 (breakpoint lockout) is not set, it means this is not the first storage reference after a breakpoint stop. NOTE: If it is the first storage reference after a breakpoint stop, setting of K134/135 (breakpoint stop) must be blocked to prevent consecutive stops on the same address.

If BPO is set, the breakpoint address is compared to the address on the S bus. If these addresses match

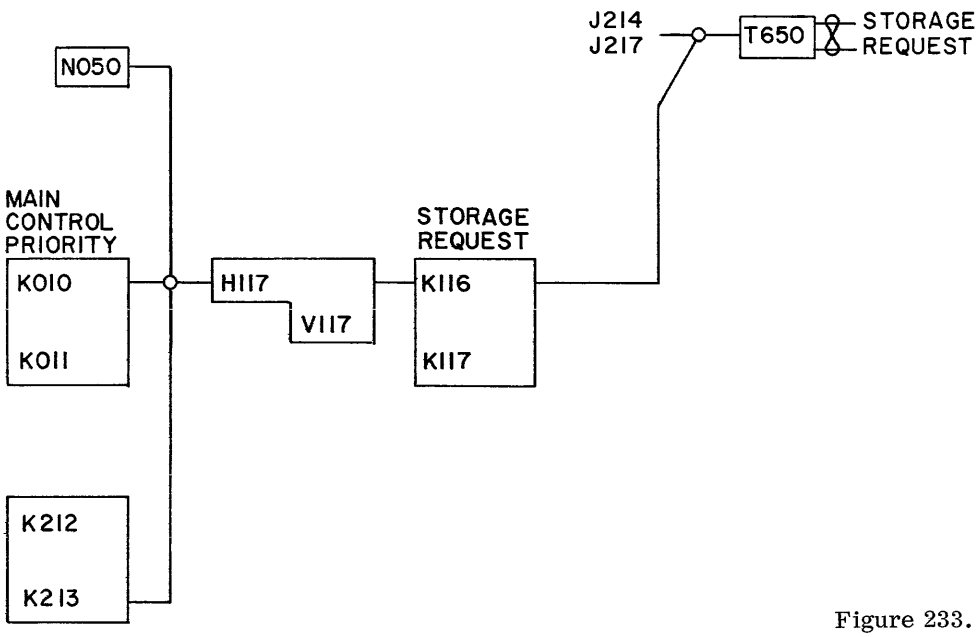
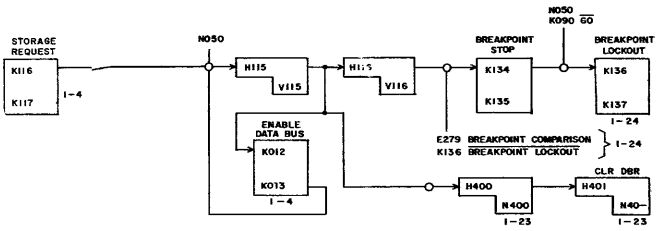


Figure 233. Storage Request



V050: Input to H115.

V115: Set K012/013 (enable data bus); set K004/005 (address mode, p. 2-71*) if word-addressed instruction or clear K004/005 if character-addressed instruction.

V116: Test for breakpoint comparison if BPO has been selected and K136 = 1 (p. 2-81).

Figure 234. Breakpoint Test

during an ROP or STO sequence the computer is stopped. It should be noted that the 13-bit address on the S bus is always a word address.

DB register will be cleared by H401 (at this time, N40X) as it is during RNI, RADR, or ROP read sequence. It has no special value at this time but its existence should be pointed out here.

DB register will be cleared before the operand arrives from storage. (This will be explained later.)

Main control then waits for the selected storage module to reply. Waiting time varies depending on the status of the memory module.

Review what has occurred in ROP so far:

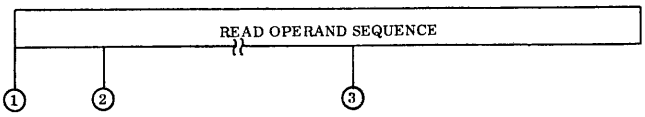
(1) _____

(2) _____

Main control is hung up. There is nothing that it can do until storage replies. The reply signal indicates that storage is processing the request and should place the operand on the data bus after a predictable delay.

Step 3 of ROP sequence is _____.

REPLY FROM STORAGE



(1) Start ROP; request bus system.

(2) Request storage.

(3) Reply from storage.

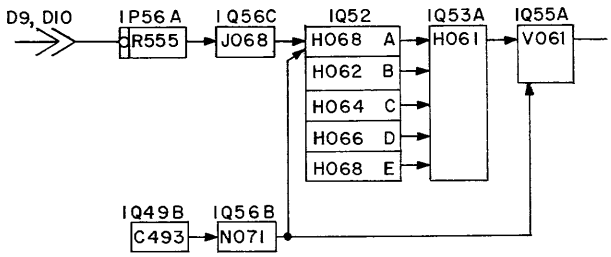


Figure 235. Reply Resync Logic

*Logic Diagrams

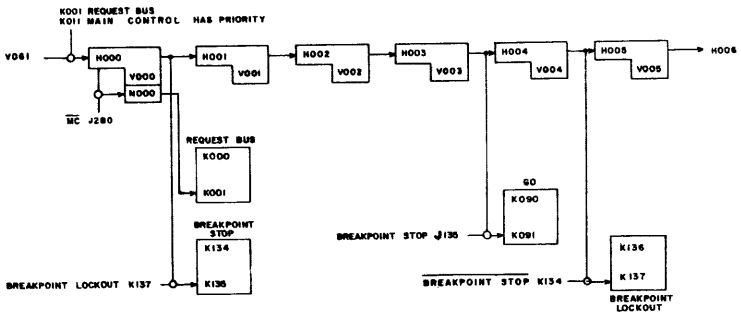


Figure 236. Breakpoint Stop

V000-V005: Clear K000/001 (request bus) to give storage module access time to store the information from the data bus.

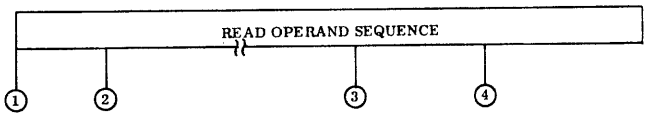
The reply signal (R555) indicates that storage is processing the request and after access time the operand will be available on the data bus.

The resync circuit is discussed in chapter 7.

Note that access time starts as soon as the reply signal from storage is resynced. One of the first things to occur after the processor receives the reply is clearing of request bus FF. If BPO is set, the breakpoint address is continually compared to the storage address on the S bus. During an ROP sequence, if the storage address matches the breakpoint address, breakpoint stop FF (K123/135) is set. K135 is ANDed with V003 to clear go and stop the computer.

Clearing request bus FF neither clears nor releases the bus. When main control needs the bus system, it requests it via K000/001 (request bus FF). If the bus system is not busy main control receives priority and sends a storage request. Storage acknowledges the request by signaling that the request is being processed. At this point the information requested has not yet been transmitted via the bus system. In other words, main control has the bus, thus has no need to request it, so K000/001 is cleared. Main control must still use the bus system, thus it has not been released. Main control priority (K010/011) remains set until V007 time on main control timing.

RELEASE BUS SYSTEM



(1) Start ROP; request bus system.

(2) Request storage.

(3) Reply from storage.

(4) Release bus system.

V006: Input to H401.

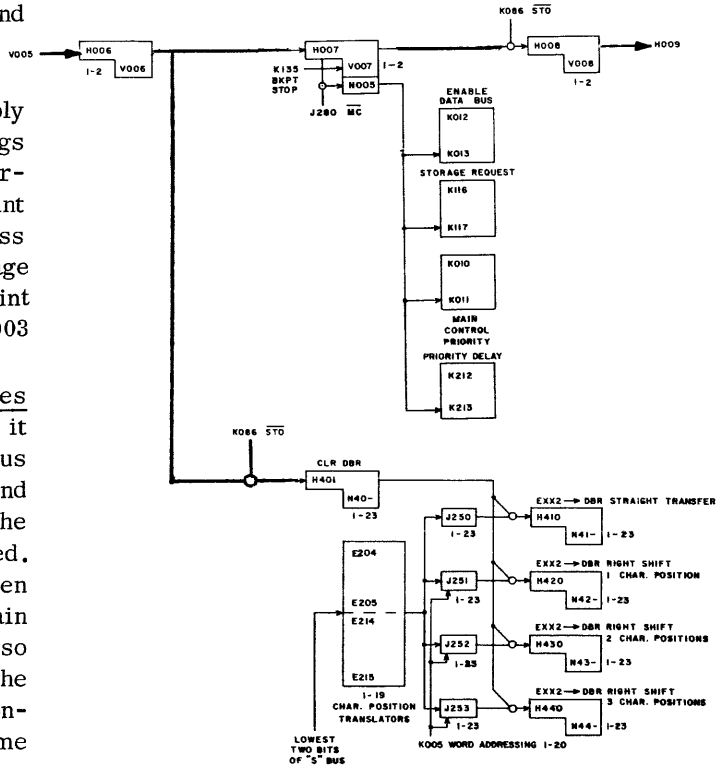


Figure 237. DBR Enables

K135 holds V007 to a 0, disabling the main timing chain

V007/N005: Clear DB register; operations depending on V007 will not occur if breakpoint stop exists. Clear K212/213 (priority 2), K010/011 (program control bus priority), K116/117 (storage request), nad K012/013 (request lockout).

V008/N050: Set K104/105 (start arith) and enter H000.

ENABLE DATA BUS TO DB REGISTER AND START ARITHMETIC

Data Bus to DB Register

All transfer of data between the processor, I/O, and memory require use of the data bus, the path for exchange of 24-bit data. All memory and I/O modules are tied to the bus so all data transmitted on the bus is available to each module. A unique line from the processor to each module carries the signal that grants bus priority.

Data from an I/O or memory module is taken from the lines via the R5XX receiver rank and fed to the EXX2 inverter rank (figure 228). EXX2 is also fed by the I^7 inverter rank of the processor. This provides a path for transmission of data from processor to bus. Data from EXX2 or CXX2 (C register) is gated into DB register.

Character shifting is performed between the EXX2 rank and DB register. Data can be transferred directly or right-shifted (end-around) one, two, or three character positions. These shifts are controlled by the H4X0 control delays. During character addressing one of these delays is pulsed according to the translation of the lower two bits of the character address. This then gates the selected character from EXX2 into the lower six bit positions of DB register.

The contents of DB register are transmitted over the bus by the T5XX transmitters. Information which may be transmitted includes the complement of the contents of DB register, the complement of the contents of P (used for a return jump or interrupt), and the complement of the BCD characters contained in DB register.

Shifted transfers (EXX2 to DB register) are enabled

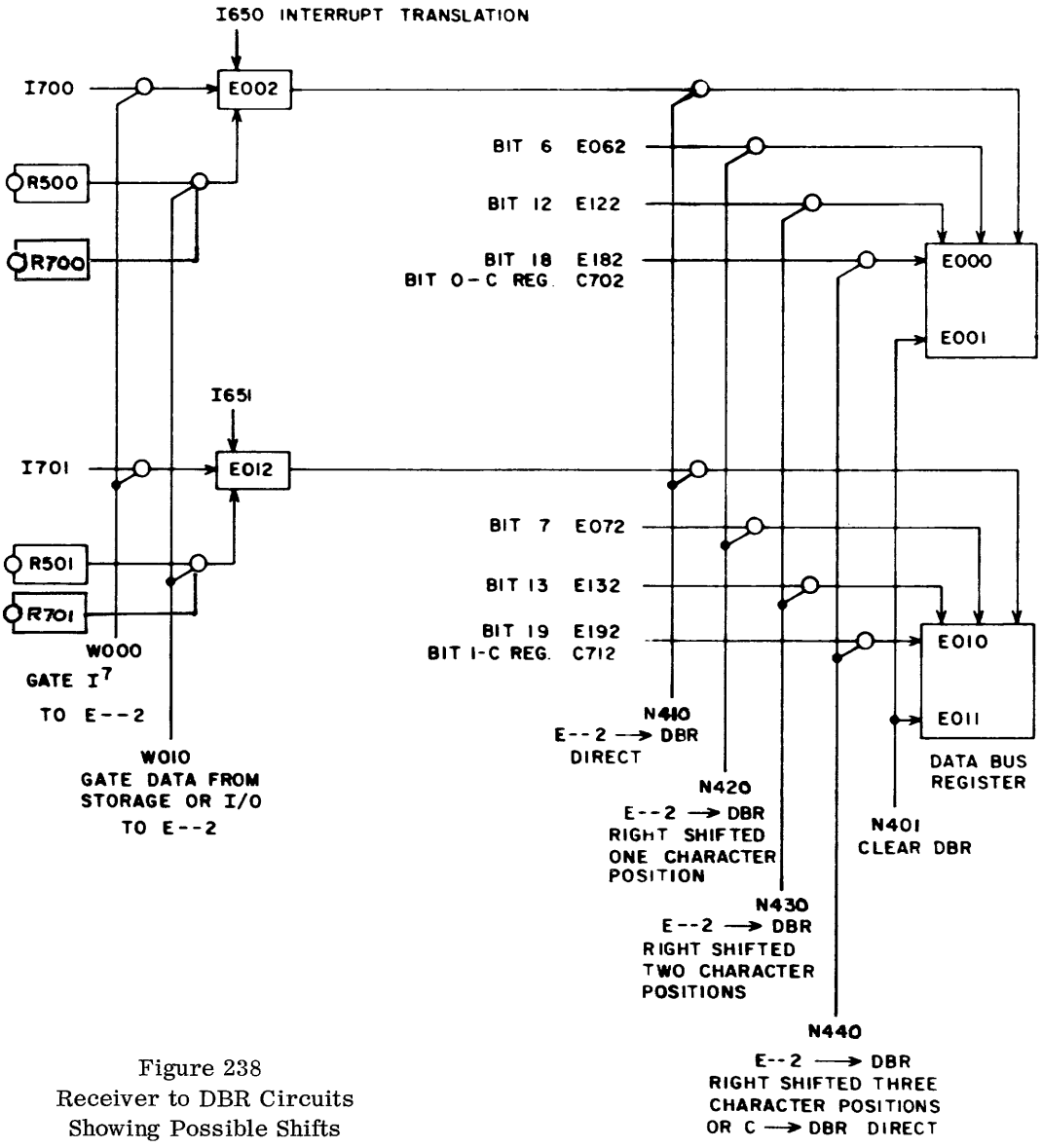


Figure 238
Receiver to DBR Circuits
Showing Possible Shifts

for character-addressed instructions by clearing K004/005 (address mode) at V115 of the storage reference (see figure 239).

K004/005 is set at V115 time of the storage reference for word-addressed instructions. This permits only a straight transfer from EXX2 to DB register. Both straight and shifted transfers of EXX2 to DB register are 24-bit transfers with the shifted transfers being right end-around (except for I/O).

K042/043 is set by V115 and cleared three phase times later by an N4XX term. This allows correct translation of the character address bits for a Read operation.

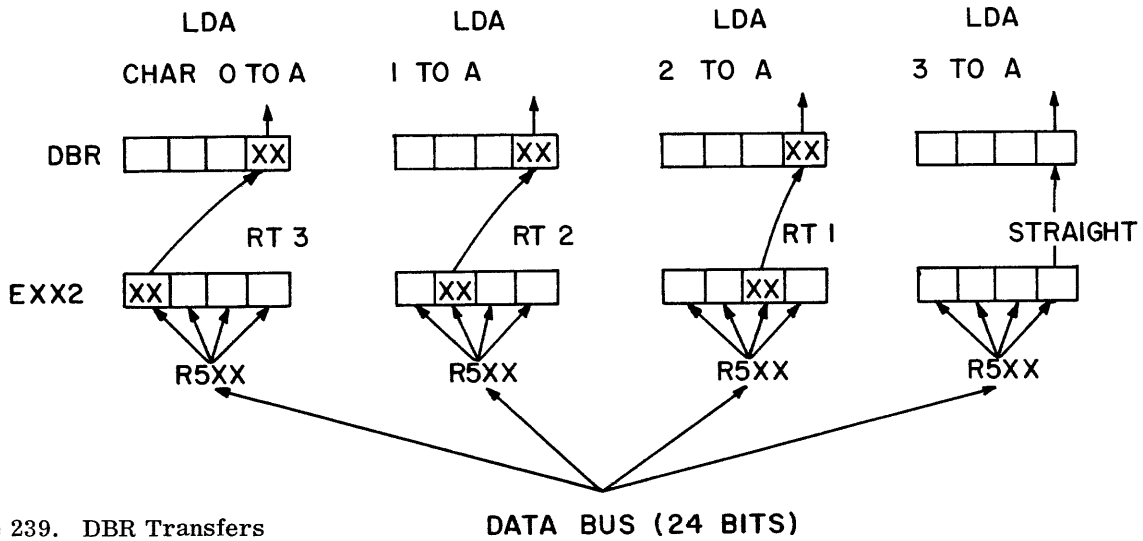
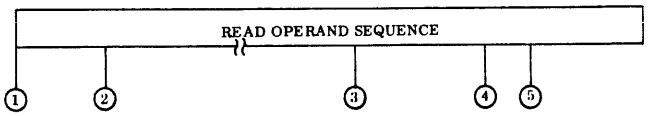


Figure 239. DBR Transfers

START ARITHMETIC



(1) Start ROP; request bus system.

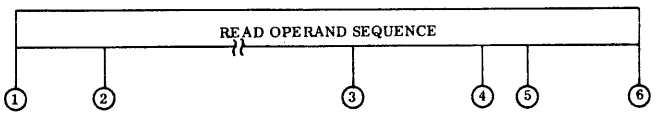
(2) Request storage.

(3) Reply from storage.

(4) Release bus system.

(5) Enable data bus to DB register; start arithmetic.

END ROP



(1) Start ROP; request bus system

(2) Request storage.

(3) Reply from storage.

(4) Release bus system.

(5) Enable data bus to DB register; start arithmetic.

(6) End ROP.

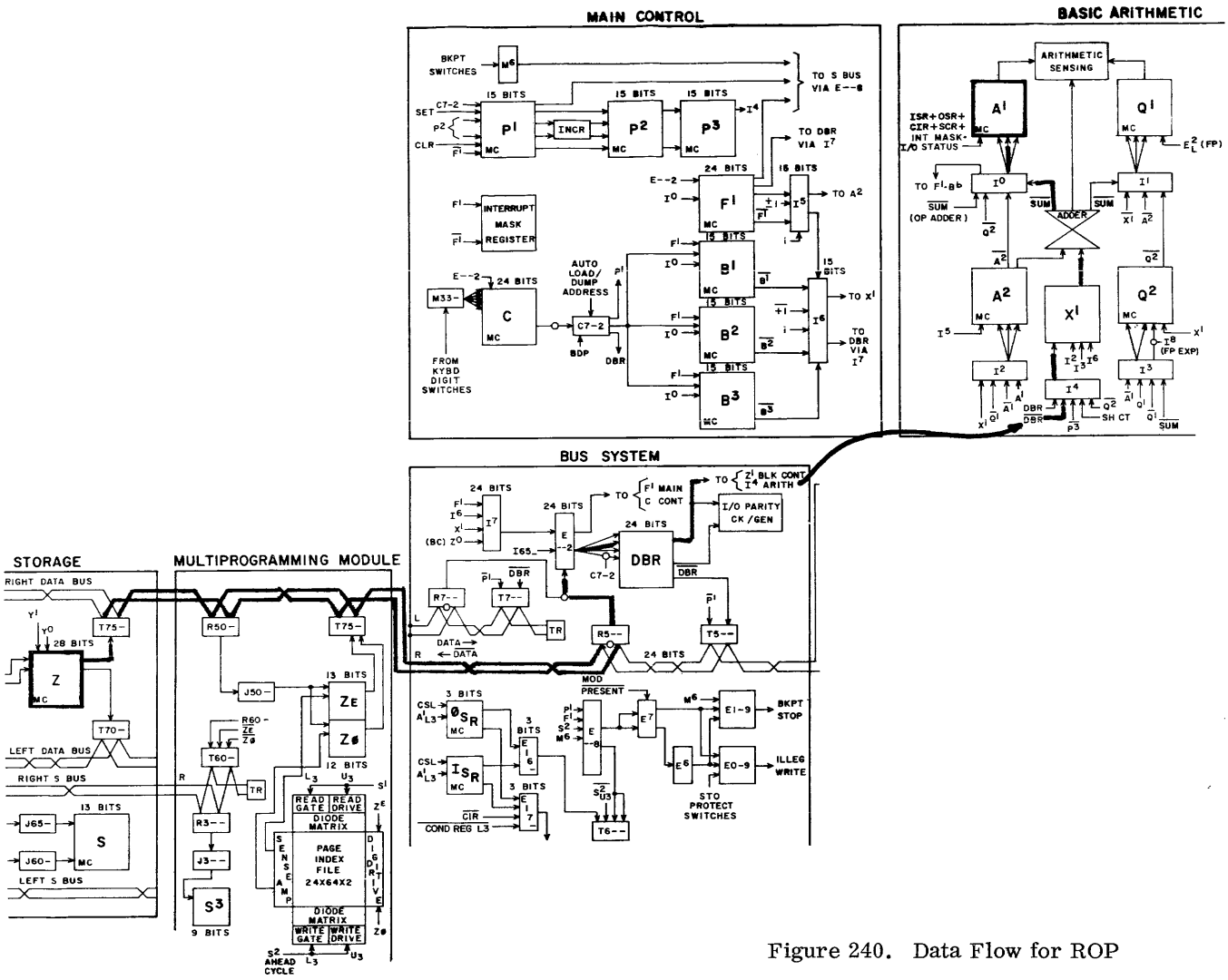
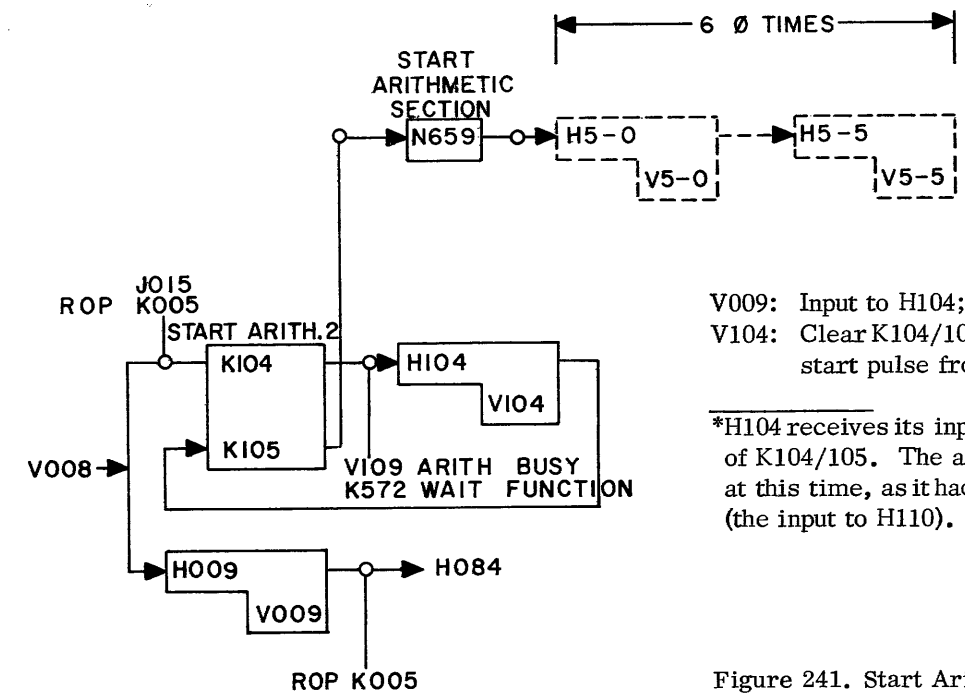


Figure 240. Data Flow for ROP



V009: Input to H104;* input to H084; test stop.

V104: Clear K104/105 (start arith II) permits only one start pulse from N691.

*H104 receives its input the next odd time after setting of K104/105. The arithmetic section cannot be busy at this time, as it had to be free to start ROP sequence (the input to H110).

Figure 241. Start Arithmetic on ROP

V084: Progress to next sequence according to (F1).*

It should be noted that end ROP (V084) cannot set RADR because there is no progression of instructions from ROP to RADR.

Sequence progresses to RNI or STO. For progression to RNI see initiate RNI control delay in chapter 7. (Note the input that would be used at the end of ROP.) If, instead, progression is from ROP to STO, two replace instructions, 10.0 (storage shift) and 34.X (replace add), are used.

V009:

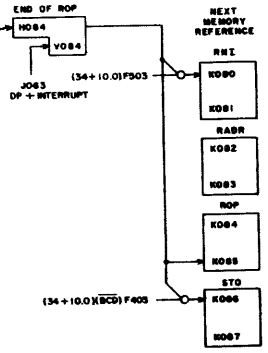
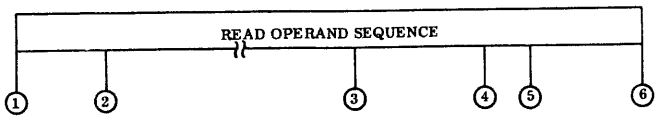


Figure 242. End ROP Timing

NOTE: V008 started the arithmetic section for actual loading (adding, etc.) of operand. Refer to figure 242 which shows the load A flow operation. The arithmetic area will run by itself without control from a sequence. For a complete discussion of the arithmetic section operations refer to chapter 12.

REVIEW

You should be able to fill in the blanks without referring back to the text.



(1) _____

(2) _____

(3) _____

(4) _____

(5) _____

(6) _____

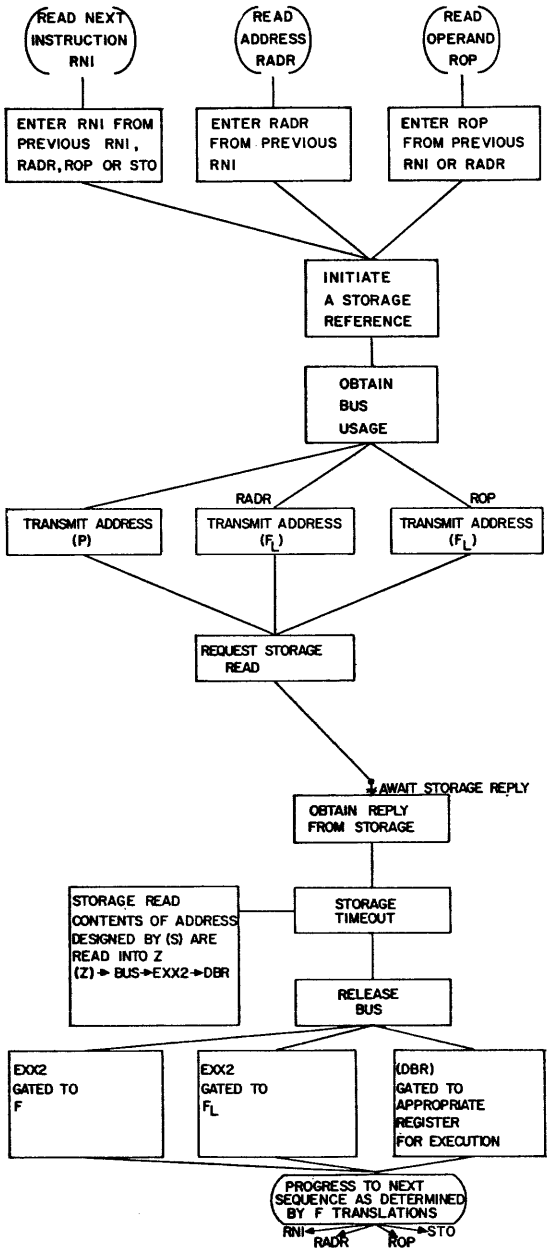


Figure 243. ROP Big Picture

*V084 would not occur if this were the first ROP cycle of a double-precision instruction. J063 will be a 1 to block the end of ROP if more than one ROP sequence is required to accomplish this instruction.

SELF-EVALUATION QUIZ ON CHAPTER 9

TRUE OR FALSE (T or F):

1. ROP sequence may be entered from RNI, RADR, or ROP.
2. ROP always exits to RNI or another ROP.
3. ISR is never referenced for the upper three bits of address during ROP.
4. During ROP, storage will place either six or 24 bits of data on the data bus.
5. During ROP the gating of EXX2 to DB register may be a right end-around shift to allow for character handling.
6. Interrupt may never be sensed during ROP.
7. A BPO is possible on the second ROP of double-precision instructions.
8. P is never advanced during ROP.
9. V110 must always be used to set request bus FF for ROP.
10. Go FF will never be cleared during ROP.

Score yourself:
Only a perfect score is passing.
If you missed one or more, you've failed!

CHAPTER 10

STORE OPERAND SEQUENCE

INTRODUCTION

During the Store Operand (STO) sequence, new information (taken from an arithmetic or control register) is stored in the memory location specified by the 18-bit address placed on the CPU S bus. The lower 15 bits are obtained from F lower 15, while the upper 3 bits may be obtained from OSR, ISR or may be 0s, depending on the Mode and State of the system. The STO sequence may be entered from an RNI, RADR, or ROP sequence. STO is entered following an RNI when initiating an interrupt routine. If Store AQ is being executed the STO FF remains set at the end of the first STO and another STO is performed.

The steps in STO are:

1. Set STO at the end of RNI, RADR, or ROP, or continue with another STO.
2. Perform indexing if required.
3. Request bus priority.
4. Obtain bus priority. (Wait if bus is being used.)
5. Transmit a write signal and the write designator signals and transmit the storage address from F via the S bus.
6. Start arithmetic.
7. Transmit a storage request.
8. Clear X and A2.
9. For a store A, gate I2XX to X. $\overline{\text{(A1}}$ to I2XX is enabled when F code is translated by F2 translators during RNI.)
10. Enable $\overline{\text{DBR}}$ to the T5XX transmitters of the data bus for all instructions but return jump (for STO sequence of a return jump and STO occurring on initiation of interrupt enable $\overline{\text{P}}$ to the T5XX transmitters).

11. Test breakpoint. Set breakpoint stop if BPO has been selected and the breakpoint address matches the operand address on the S bus.
12. Test storage protect. Set illegal write if a protected storage location is to be referenced.
13. Clear DB register.
14. Gate EXX2 to DB register, ($\overline{\text{DBR}}$) to T5XX. (For an STO involving A or Q, data from X register is enabled to I7XX to EXX2 as soon as the function code is translated during RNI. If a store index instruction is to be performed, B^b to I6XX, I6XX to I7XX to EXX2 is enabled when the function code is translated.) Main control now waits for the selected storage module to signal a reply. Waiting time depends on status of memory module.
15. Receive and resynchronize a storage reply; use this signal to start the main timing chain.
16. Stop if breakpoint stop is set.
17. Clear DB register.
18. End STO. If in the first cycle of a 48-bit precision store instruction, STO remains set and another store sequence is executed. If another STO is not necessary, RNI is set.

DETAILED TIMING

This sequence is used by instructions which place data in core storage. Refer frequently to figures 244 and 245 as you progress through the detailed timing of STO.

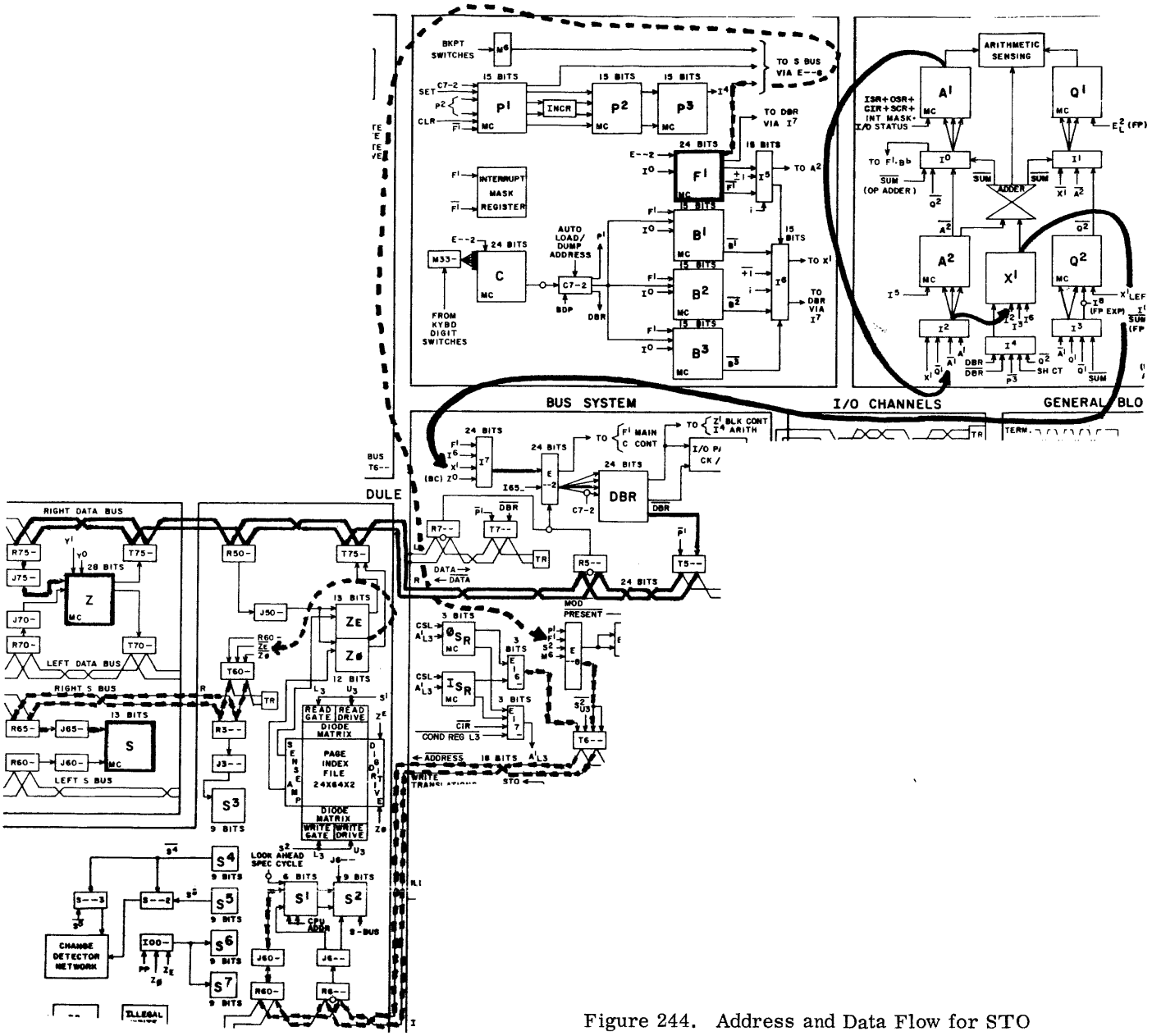


Figure 244. Address and Data Flow for STO

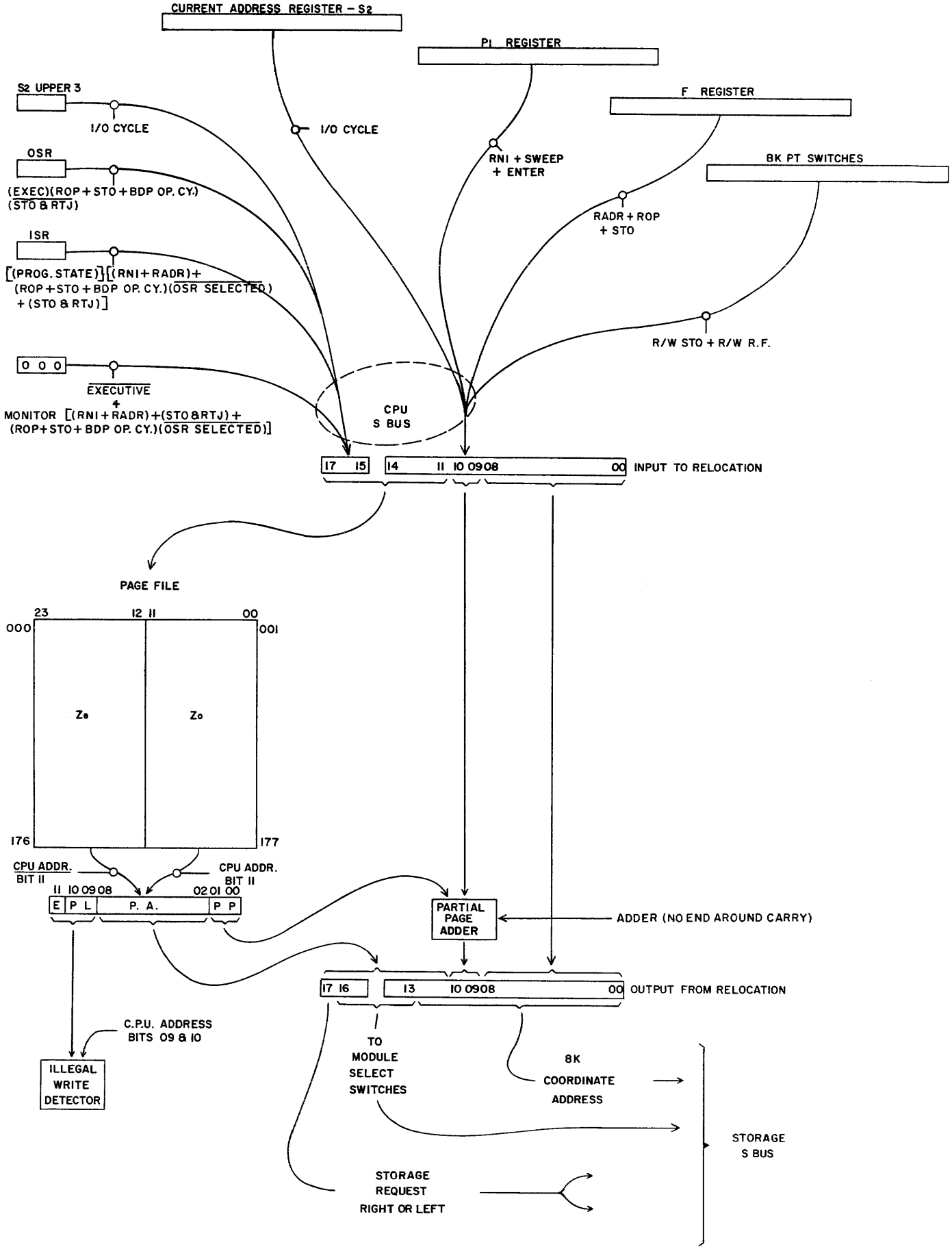


Figure 245. Address Flow

To store an operand in core, several steps must occur:

1. Request bus system (K000/001); set start arith. Lower F holds address of operand.
2. Obtain bus priority (K010/011).
3. Transmit address from lower F on S bus to storage.
4. Transmit write signal and write designators (T655 = 1).
5. Transmit storage request (K116/117).
6. Enable DB register to bus.
7. Test for breakpoint stop if BPO (K134/135); drop write signal if breakpoint stop.
8. Clear DB register.
9. Gate EXX2 to DB register. I^7 is enabled to EXX2.
10. Wait and resync reply from storage (V061).
11. Clear request bus (K000/001).
12. Time out access to allow storage module to store word.
13. Stop if BPO.
14. Advance storage reference controls to RNI + STO.
15. Progress to next storage reference if not stop.

Figure 246 graphically shows the STO sequence of events.

During STO main control must place an operand on the data bus, making it available to storage. Main control will start the arithmetic section about the same time as storage is requested.

The arithmetic section will move the contents of A or Q register to X register from where it will be enabled to I^7 to EXX2 and then gated to DB register.

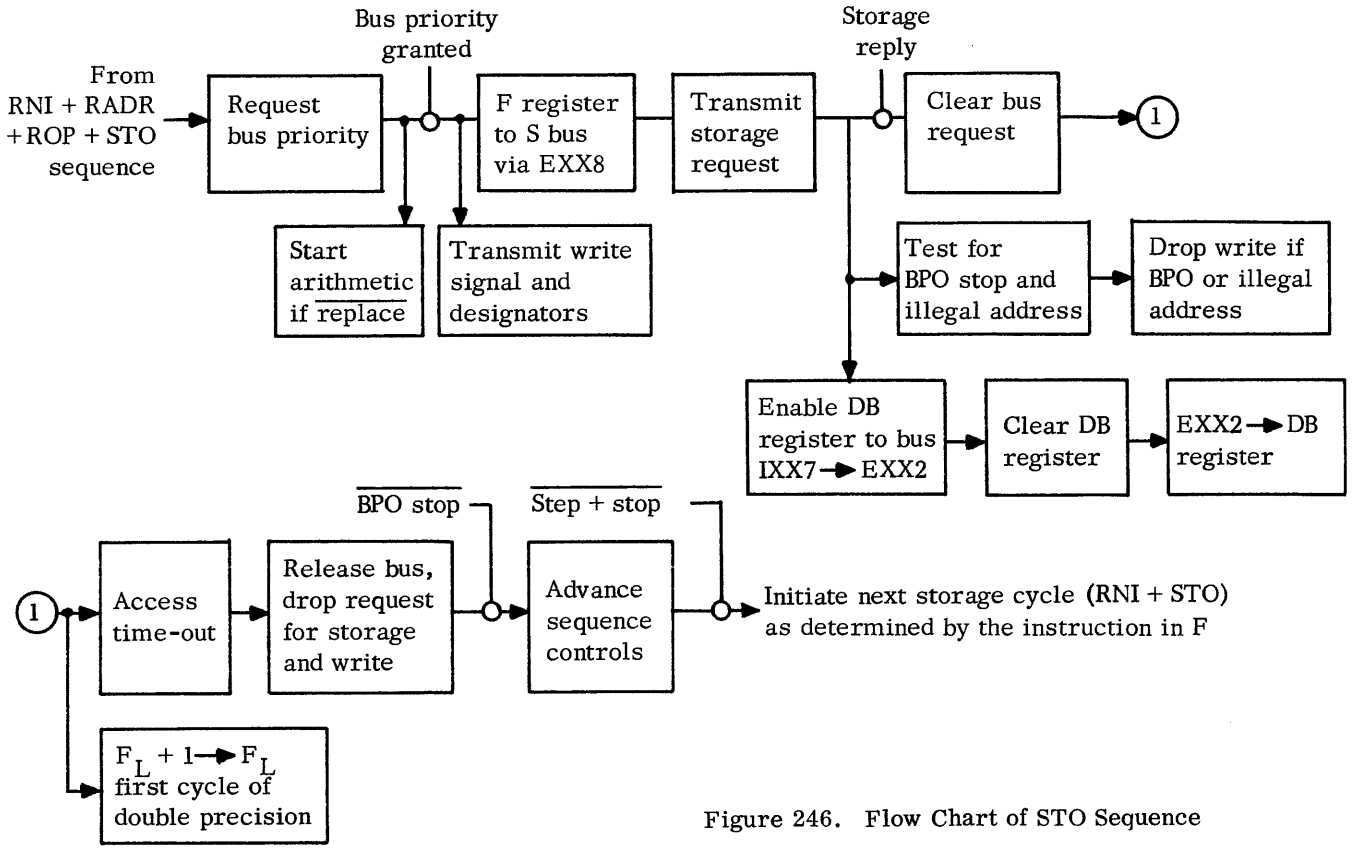


Figure 246. Flow Chart of STO Sequence

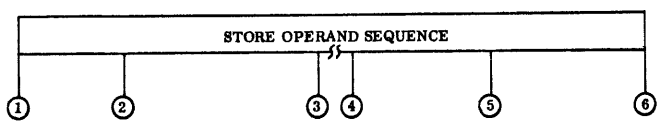


Figure 247. Major Timing of STO Sequence

(1) Start store (STO); request bus system.

(2) Request storage; start arithmetic.

(3) Word to bus.

(4) Reply from storage.

(5) Release bus system.

(6) End STO.

START STO; REQUEST BUS SYSTEM

(1) Start store (STO); request bus system.

V380/V080 (end of RNI) or V082 (end of RADR) or V084 (end of ROP): Set K086/087 (STO); clear K084/085 (ROP); set K112/113 ($\overline{\text{RNI}}$ $\overline{\text{Index}}$). Note that the end of any previous sequence could start STO.

V109: Input to H110

V210: Set K000/001 (request bus); clear K112/113 input to H105.

If no address modification is needed, K112/113 (no index) will set at the end of RNI or RADR or for the two replace instructions at the end of ROP.

Notice that STO is the third sequence which requires V210 to set request bus FF. What were the other two sequences? _____

What term did the one remaining sequence, STO, require to set K000/001 (request bus)? _____

Input to H110 cannot be made if the arithmetic section is busy from a previous instruction.

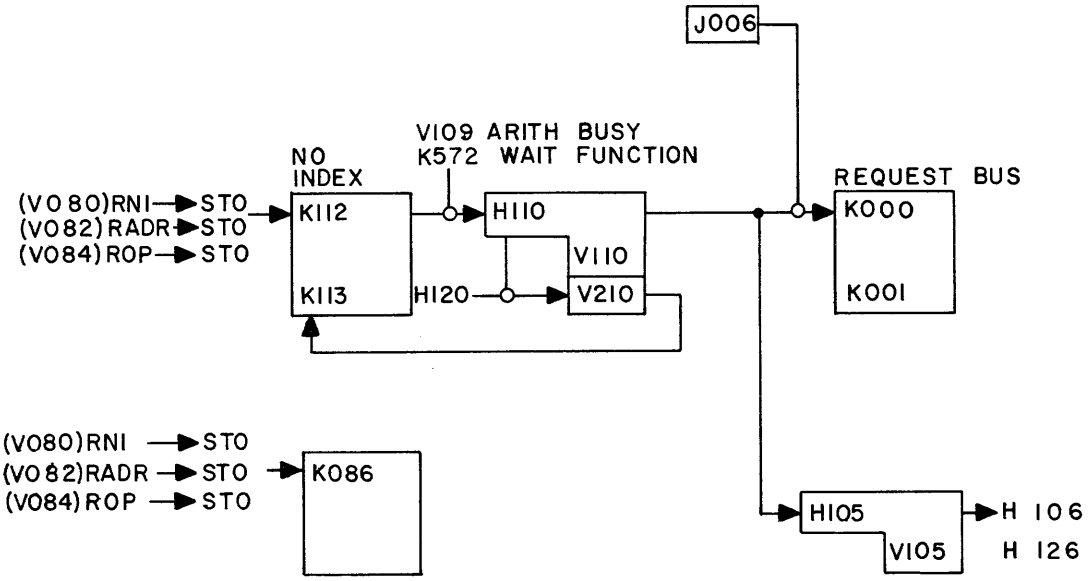
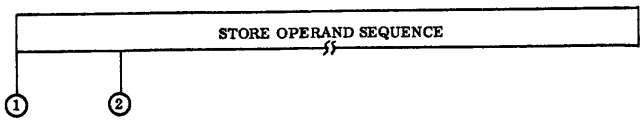


Figure 248. STO Sequence Initiation

REQUEST STORAGE AND START ARITHMETIC



(1) Start store (STO); request bus system.

(2) Request storage; start arithmetic.

Start Arithmetic

N051: Set K010/011 (p. 1-3*) if K208 = 1. If K209 = 1, repeat this test at next N051 time: When priority is granted (indicated by setting of K010/011) gate F1 to EXX8 to T6XX to S bus; gate EXX8 to EXX7 to EXX6; T655 ($\overline{\text{read}}$) is on; transmit write designators of 01111_2.

V126: Set K104/105; start arith 2.

The arithmetic, once started, will transfer the contents of A register or Q register to X register. Main control can cause the enables for X to I^7 to EXX2.

*Logic Diagrams

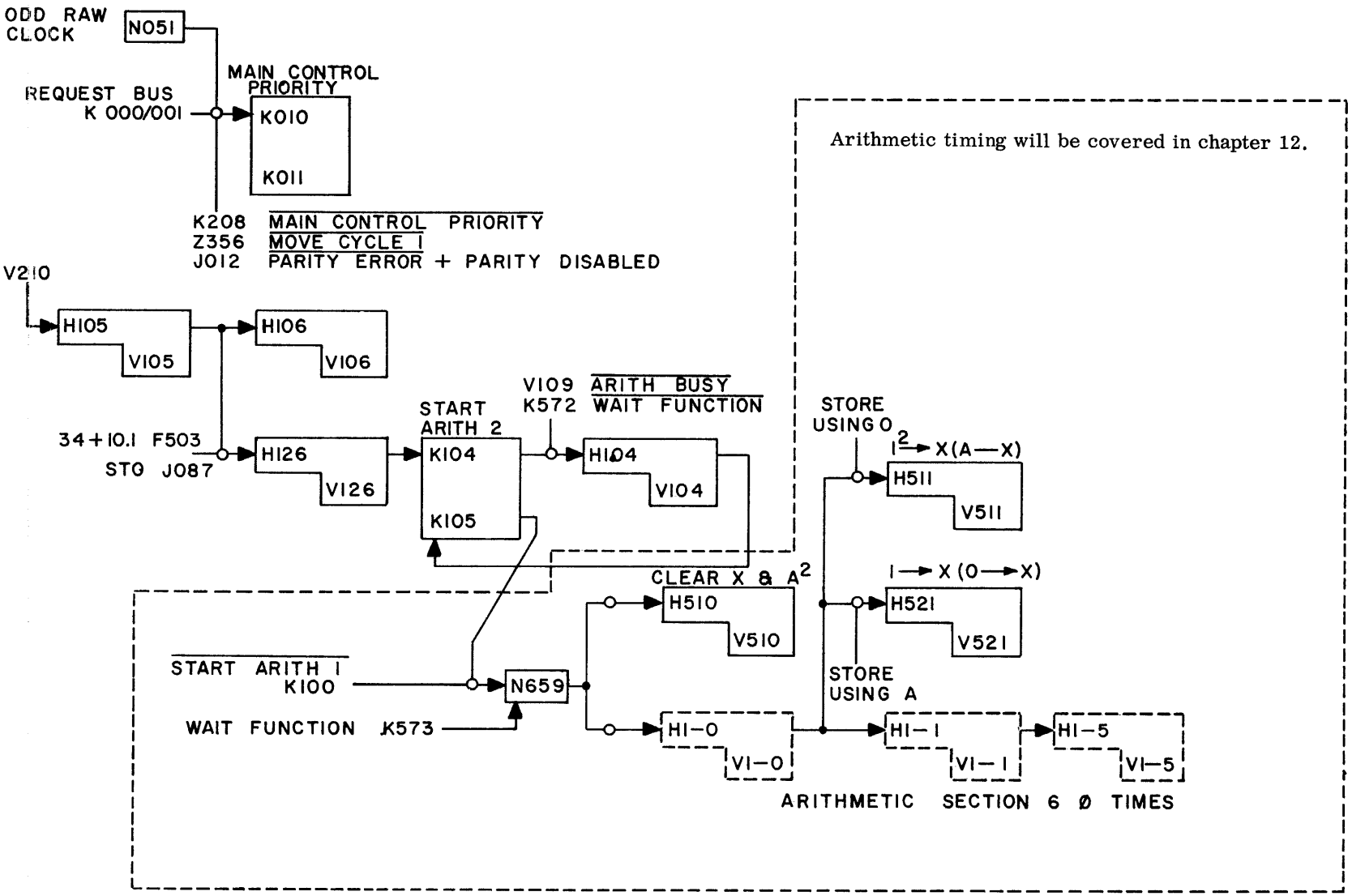


Figure 249. Arithmetic Start on STO

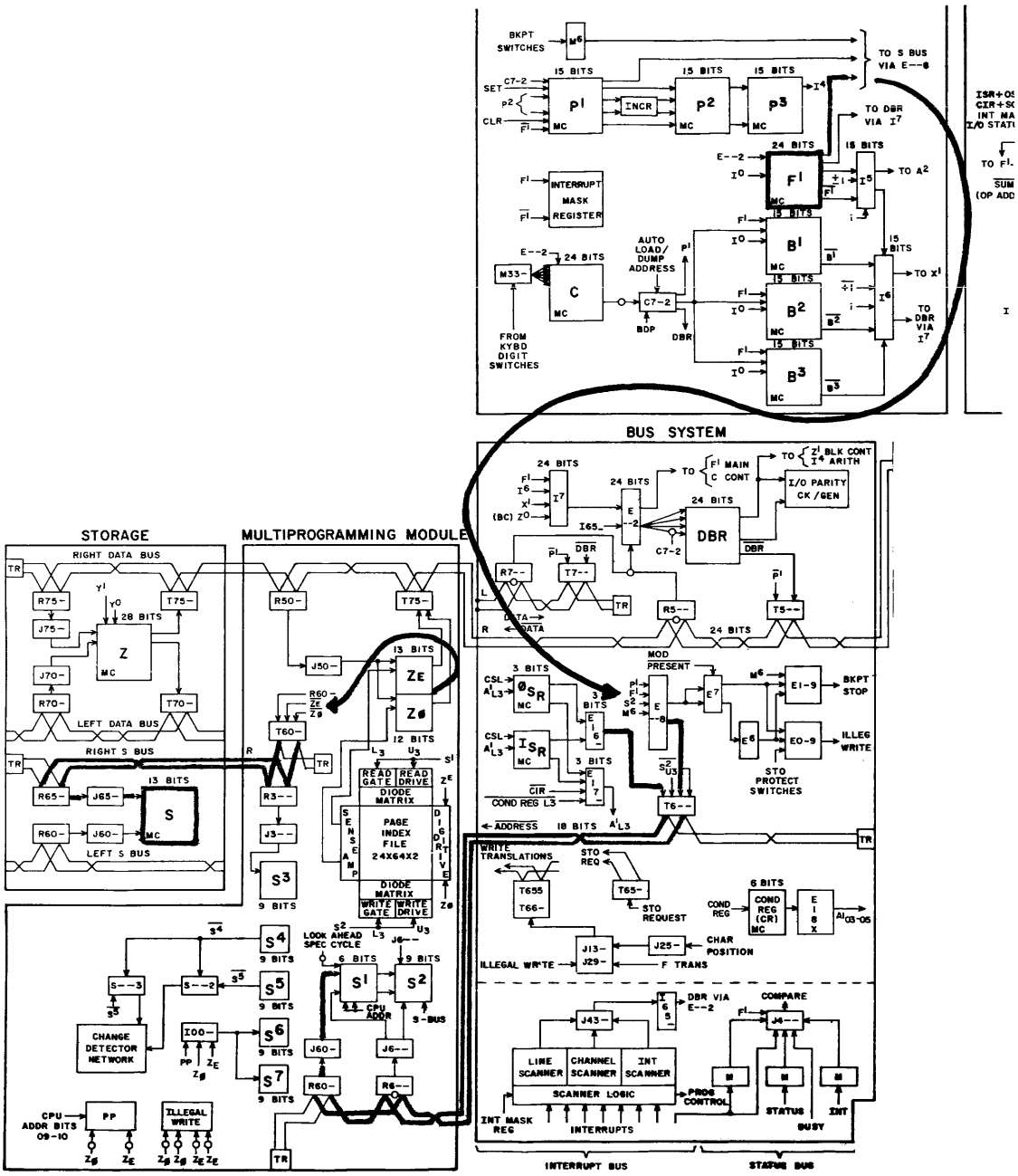


Figure 250. Address Transmission for STO

Request Storage

N050: Input to H117; at this time V126 is setting K104/105 (start arith 2).

V117: Set K212/213 (Priority 2). Set K116/117 (storage request). Transmit a request to the Multiprogramming module.

What has happened up to this point of the STO sequence?

1. Arithmetic section has been started and is gating (A) to X register.
2. A storage request has been transmitted along with a write signal and the write designators to the storage module.
3. Main control is comparing the breakpoint address with that on the S bus for possible BPO stop.
4. The address on S bus is being compared with the STORAGE PROTECT switches; checking for illegal write. The Multiprogramming module will also check for illegal write.

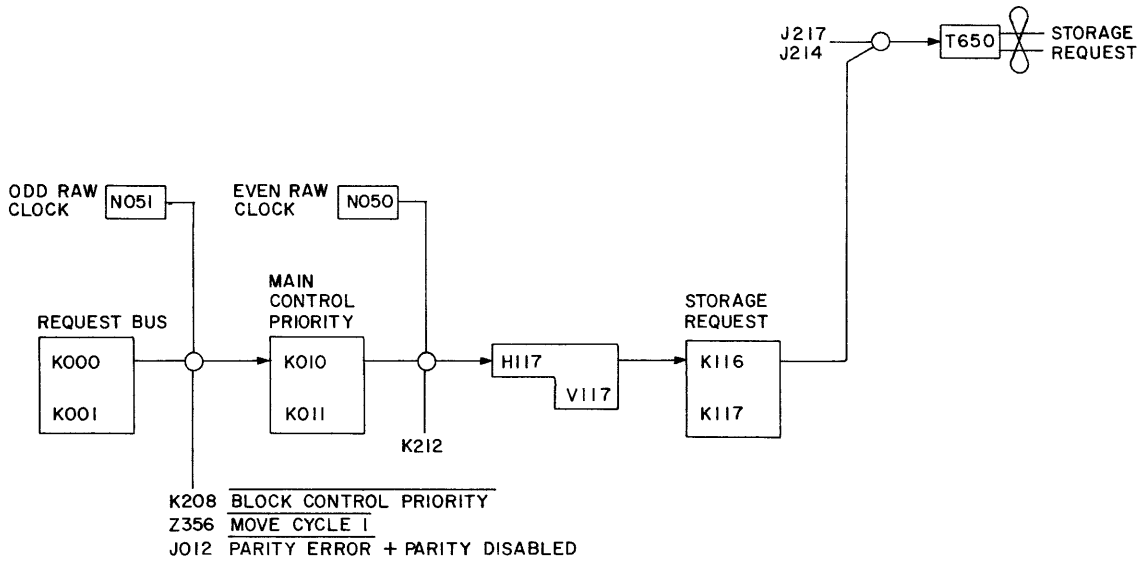


Figure 251. Storage Request on STO

Write Signal and Illegal Write

During a storage reference in which data is to be stored, the storage module receives a write signal which is generated in the CPU and relayed by the Multiprogramming module. If this write reference is illegal, the write signal will be changed to a read signal. If the system is in Executive mode when an illegal write occurs, an abnormal interrupt will be generated. In the CPU, J132 senses for an illegal write; in the Multiprogramming module, J440 senses for an illegal write.

J132 goes to 0 when:

1. (Exec) (Disable Storage Protect) (storage protect switches compare with the Address on the CPU S bus).
2. (Exec) (address on the CPU S bus is in the Auto Load or Auto Dump Area).
3. (Program State 0) (Disable Storage Protect) (storage protect switches compare with the lower 15 bits on the CPU S bus).
4. (Exec) (Keyboard Bus Priority) (address on the CPU S bus is in the Executive Auto Load or Auto Dump area).

J440 goes to 0 when:

1. E = 1.
2. Bits 09 and 10 of the CPU S bus $\geq$ PL.

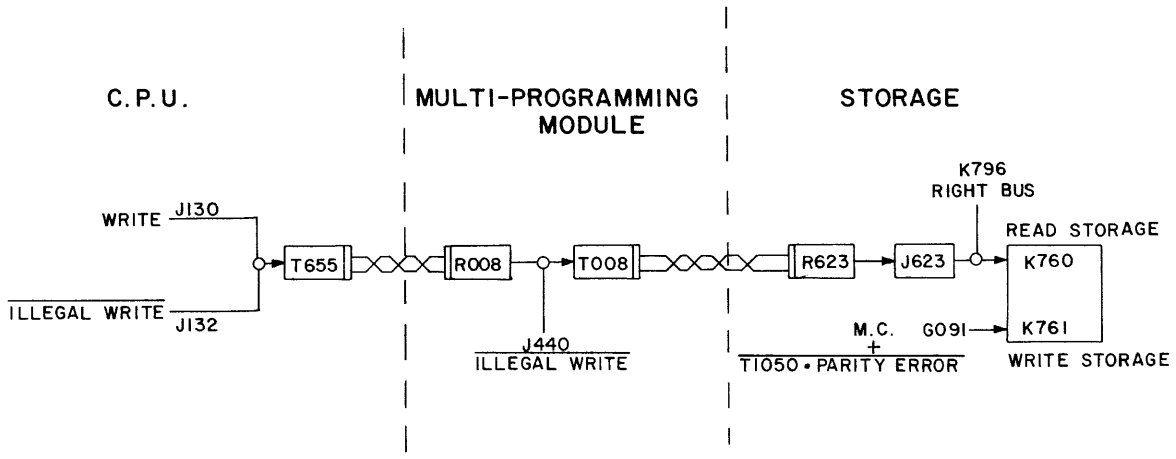


Figure 252. Write Signal Path

Write Designators

For instructions using character addressing, the character translations from E008, E018, E208, and E218 determine the write character designators (0-3) which are transmitted to storage to determine the character(s) to be written during a store character operation.

Referring to figure 253: During STO sequence of a word-addressed instruction, T660 to T663 transmitters will be turned on. This will be a full-word write of all four character positions. The J295 to J298 terms will have their inputs all at a zero level.

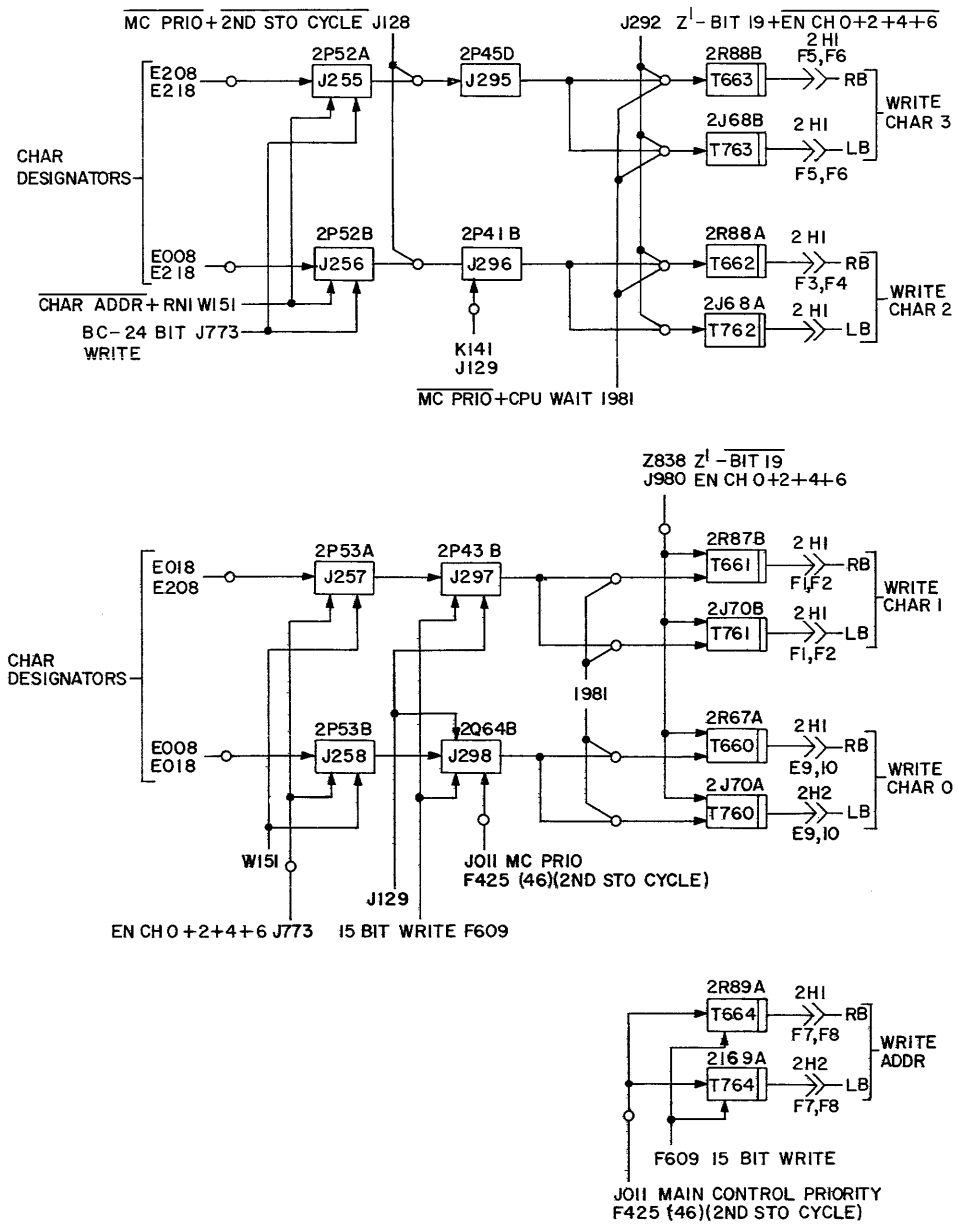


Figure 253. Partial Write Bit Transmission

For a store A character, the position at which the character is to be stored determines the transmitter that will be turned on. E008 and E018 are the complement of the character position. E208 and E218 are character position.

The other terms are used with store word address, store character address, interrupt sequences, or input operations

If BPO is set, the address on the digit switches is compared to the address on the S bus. If these addresses are equal during STO, the computer will stop. Sensing of Breakpoint Stop takes place at V116 time.

K134/135 (Breakpoint Stop) sets if: The address on the S bus compares with the address in the Breakpoint switch, BPO is selected, and Breakpoint Lockout is not set. If K136/137 (Breakpoint Lockout) is not set this indicates this is not the first storage reference after a breakpoint stop. Note: If it is the first storage reference after a breakpoint stop, setting of K134/135 (Breakpoint Stop FF) is blocked to prevent stopping a second time at the same address.

The character translators also determine setting or clearing of E204/205 and E214/215. During a write character operation, the outputs of these flip-flops determine the character transposition necessary to gate a character from the EXX2 to the lower bit positions of DB register. EXX2 is also fed by the I^7 rank of the processor. This provides a path for transmission of data from the processor to DB register. The data from EXX2 is gated into DB register.

Character shifting is performed between the EXX2 rank and DB register. Data can be transferred directly

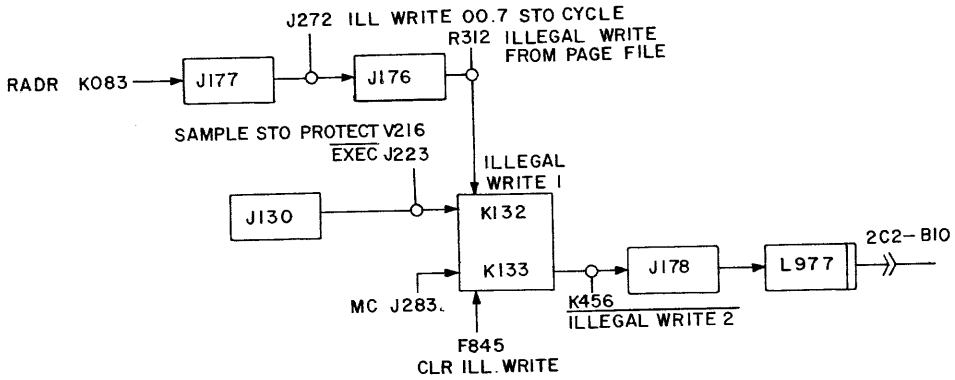
or right shifted (end-around) one, two, or three character positions. These shifts are controlled by the H4X0 control delays. During character addressing, one of these delays is pulsed on the basis of the translation of the lower two bits of the character address. This then gates the selected character from EXX2 into the lower six bit positions of DB register.

N050: Input to H115

V115: Set K012/013 (request lockout) and K004/005 (word addressing mode); input to H400. Set K042/043.

V116: Test for breakpoint comparison if BPO has been selected and K136 = 1; input to H401.

N401: Clear DB register; input to H410 (if 24-bit store).



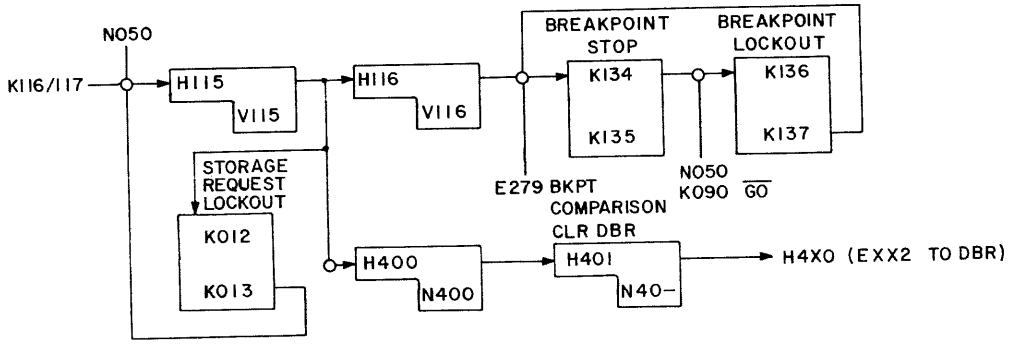
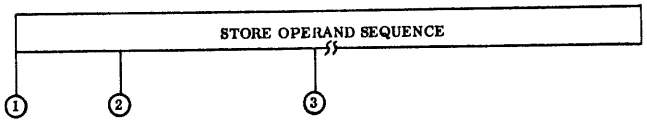


Figure 254. Breakpoint and Illegal Write Test

WORD TO BUS



(1) Start Store (STO); request bus system.

(2) Request storage; start arithmetic.

(3) Word to bus.

N410: Gate EXX2 to DB register (static enable of X to I^7 to EXX2); transmit the contents of DB register on the data bus. Clear K042/043.

The transfer from EXX2 to DB register may be straight or end-around right one, two, or three characters.

The store sequence for a word-addressed instruction will always be a straight or direct transfer from EXX2 to DB register (as, for example, in 40, 41, and 44-47 instructions).

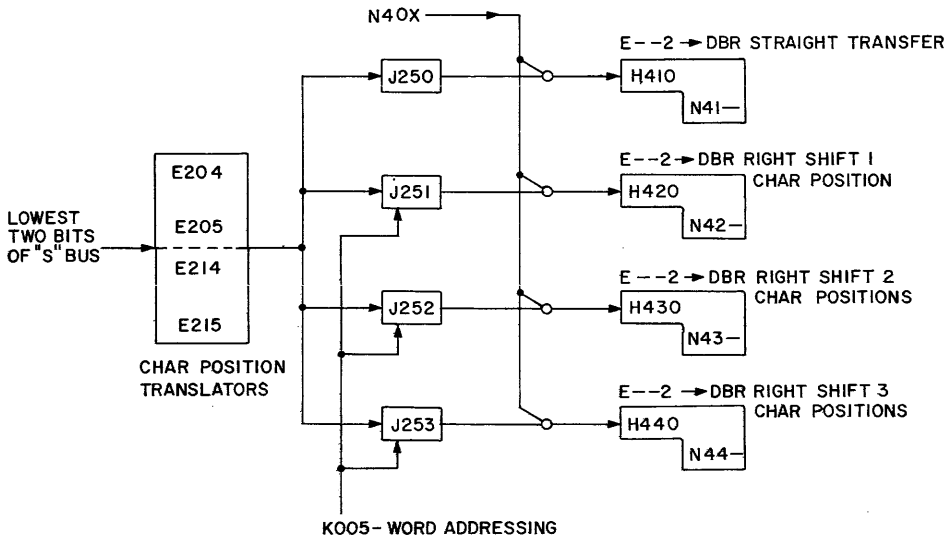


Figure 255. DBR Transfers

Character-addressed instructions may require any of four possible transfers between EXX2 and DB register. Which transfer path is used is determined by the position into which the character will be stored. Examples of the four possible transfers are shown below. The store A character is used but store Q character would be the same.

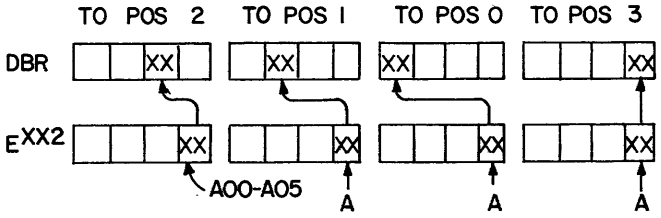


Figure 256. Operand Transmission for STO Sequence

$\overline{\text{DBR}}$ to the T5XX and T7XX transmitter enables is turned on when K012/013 (Enable Data Bus FF) sets.

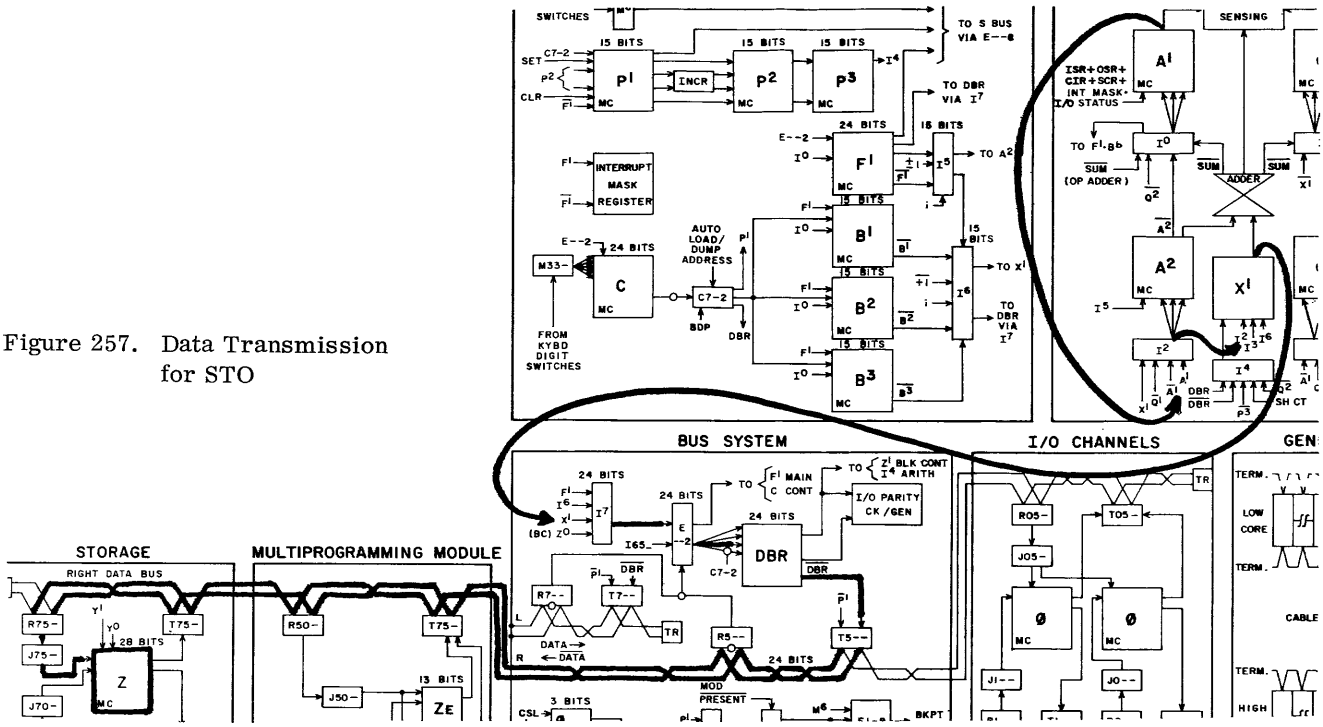


Figure 257. Data Transmission for STO

Review the progress of STO to this time:

(1) _____

(2) _____

(3) _____

Main control is hung up at this point. It cannot accomplish anything else until the reply signal arrives from the storage module.

(1) Start Store (STO); request bus system.

(2) Request storage; start arithmetic.

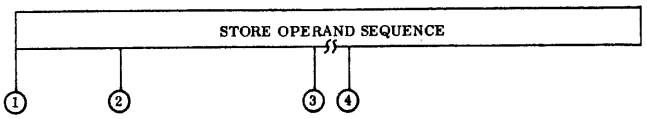
(3) Word to bus.

(4) Reply from storage.

V061 (resynced storage reply)

V000-V005: Clear K000/001 (request bus) gives the storage module access time to store the information from the data bus.

REPLY FROM STORAGE



Storage Reply Resync Circuit

A reply from any one of four storage modules is received by R555. The asynchronous reply (logical 1) is fed to H060 along with a timing pulse from N071. H060/62/64/66/68 and H061 convert the reply to a logic signal which produces a 1 output from V061 during an odd clock phase. This output is 62.5 nsec long and is not repeated regardless of duration of input.

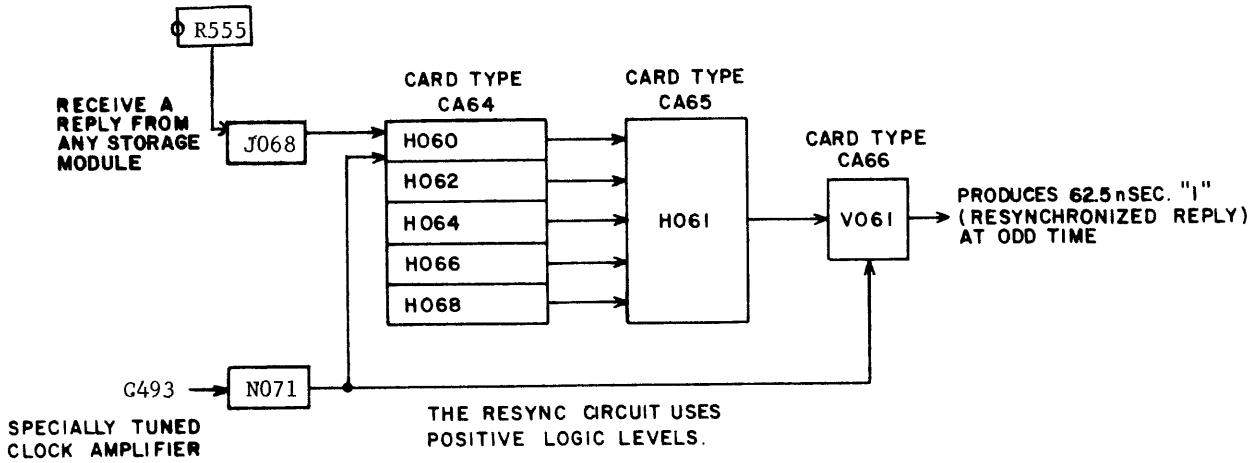


Figure 258. Storage Reply Resync Circuit

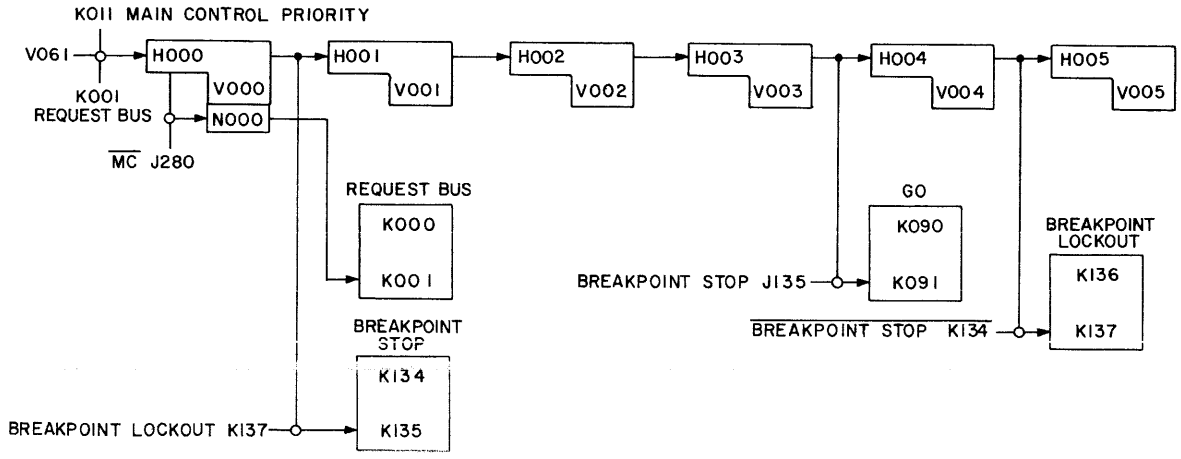


Figure 259. Breakpoint Stop Logic

The time from V000 to V005 is called "time-out for access". It permits storage to reference the data bus and to store the information. One of the first things to occur after the reply signal has been received by the processor is clearing of request bus FF. Clearing request bus FF neither clears nor releases the bus. Main control and block control share the same bus system. Main control needs the bus system, requests it via K000/001 (request bus FF), and storage starts processing the request. After the requested information is returned the bus request is released, but bus priority is retained.

V003 time will clear K090/091 (go FF) if a breakpoint stop is required. When BPO is set, the breakpoint address is continually compared to the storage address on the S bus. During an STO sequence, if the storage address matches the breakpoint address, breakpoint stop (K134/135) is set. K135 is ANDed with V003 to clear go and stop the computer.

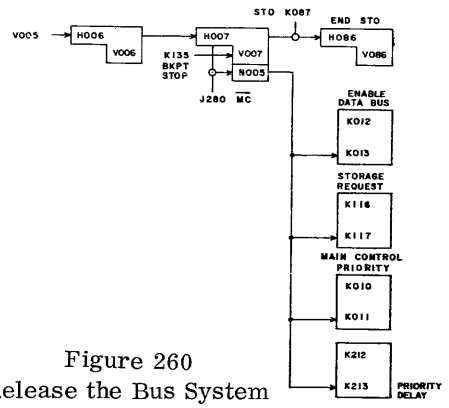
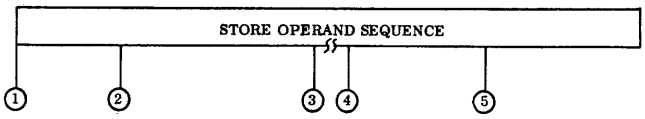


Figure 260
Release the Bus System

RELEASE BUS SYSTEM



(1) Start Store (STO); request bus system

(2) Request storage; start arithmetic.

(3) Word to bus.

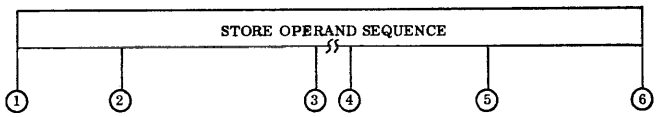
(4) Reply from storage.

(5) Release bus system.

V006
V007: Input to H086; clear K010/011, K116/117, and K012/013; test stop; end transmission of the DB register to Storage; clear K212/213.

After storage reply and access time has elapsed, main control will release the bus at V007 time. Releasing the bus system as soon as possible is important. Block control and main control use the same bus system. The enable set up earlier (DB register to storage) will drop out now.

END STO



(1) Start Store (STO); request bus system.

(2) Request storage; start arithmetic.

(3) Word to bus.

(4) Reply from storage.

(5) Release bus system.

(6) End STO.

V086: Clear K086/087 (STO); set K080/081 (RNI); input to H087.
V087: Input to H014; perform RNI if computer is going.

K340/341 blocks the progression from STO to RNI during manual store operation. For a discussion of manual store operation and timing see chapter 6.

The input from V007 to H014 must make it through the AND gate with K081 (RNI FF). This AND gate can be made if the next sequence is RNI. Other inputs that come into H087 (not illustrated) require the AND gate for other operations (discussed elsewhere in this manual).

The sequence progression FFs are always affected the same way at the end of the STO sequence. STO is cleared and RNI is set. There is no other progression path possible except STO to RNI, unless it is a double-precision operation which would be RNI - STO - STO - RNI at which time J063 (DP) will be a 1 blocking the output of V086 until the double-precision operation is complete.

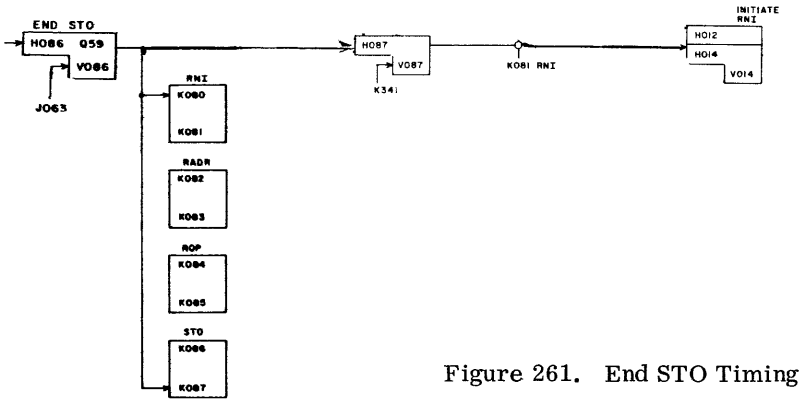


Figure 261. End STO Timing

REVIEW

You should be able to fill in the blanks without referring back to the text.

(1) _____

(2) _____

(3) _____

(4) _____

(5) _____

(6) _____

Figure 262 is a graphic representation of all four basic sequences. Note the areas unique to each sequence and common to all.

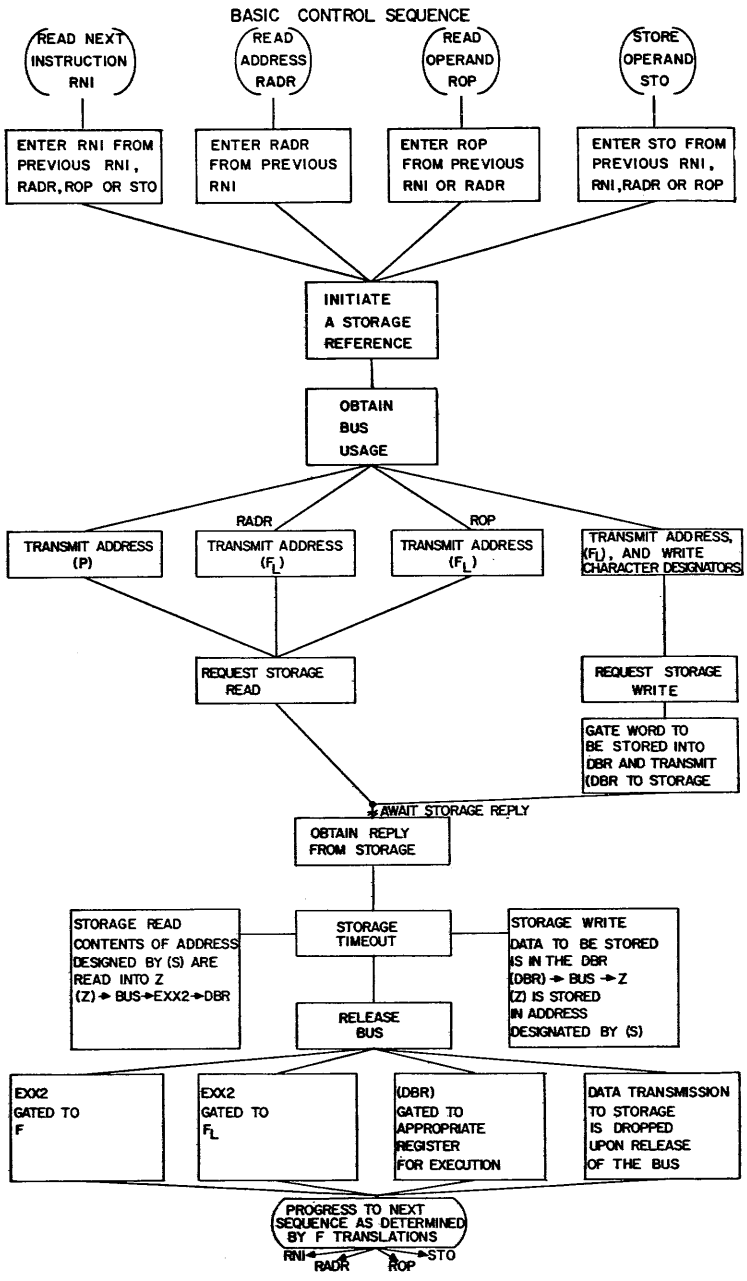


Figure 262. STO Big Picture

SELF-EVALUATION QUIZ ON CHAPTER 10

TRUE OR FALSE

1. The STO sequence may be entered from RNI, RADR, ROP, or another STO sequence.

2. If the system is in the Monitor state and operand relocation is not selected, ISR will be referenced for the upper three address bits during STO.

3. If the system is in the Monitor state an illegal write cannot occur during STO.

4. The Write signal, which is generated by main control during STO, is relayed to storage by the Multiprogramming module.

5. If an Illegal Write is sensed during STO, storage reference is aborted and a pseudo-reply is generated.

6. If the system is in the Program state, storage protect switches can be used to protect logical addresses $0XXXXX_8$.

7. For character store operations, it is the responsibility of main control to properly position the character to be stored.

8. For every STO, main control must send a Write signal and Write Designators to storage.

9. The only exit from STO is to RNI.

10. For a store character 2 operation, gating of EXX2 to DB register will be right 3.

11. During STO, main control sends $\overline{\text{Data}}$ to storage via the data bus cable.

12. A BPO is possible on the second STO of a store AQ instruction.

13. If the system is in Non-Executive mode, an STO in the upper 40_8 locations of available storage would be an illegal write.

14. If the system is in Executive mode, an STO in the upper 100_8 locations of page 0 would generate an illegal write.

15. A storage request is always sent to storage as soon as K116/117 sets during STO.

Score yourself:
After studying chapters 5, 6, and 7 you shouldn't have missed any questions.
If you missed one or more, you've failed!

CHAPTER 11

SEQUENCE PROGRESSIONS

This chapter lists all possible cycle progressions for the 3300 Computer. It will be to your advantage to use 3300 Command Timing Charts and 3300 Computer System Logic Diagrams in conjunction with this chapter. You will find that sequences of the 3300 will take on more meaning if you try a few sample cases of sequence progression.

SEQUENCE PROGRESSIONS

Chapters 7 through 10 individually covered the sequences, RNI, RADR, ROP, and STO. This chapter emphasizes the progression of sequences used during execution of instructions. Some instructions use only one sequence, others use two or more, and one instruction could use all four sequences.

The following pages are groupings of instructions that use a definite sequence progression in their execution.

These instructions use only RNI:

Stop	00.0	Unconditional stop
Jumps	00	Miscellaneous jump
	02	Index jump (IJ)
	03	AQ jump (AZJ, AQJ)
Skips	04	A, Q, or B^b = Y*, skip
	05	A, Q, or B^b = Y, skip
	10	Index skip
Shift or scale (may be indexed)	12	Shift A (SHA) Shift Q (SHQ)
	13	Shift AQ (SHAQ) Scale AQ (SCAQ)
No address, arithmetic, enter, or logic	11	Enter A with 17-bit address
	14	Enter A, Q, or B^b with Y
	15	Increase A, Q, or B^b by Y
	16	Selective complement (exclusive OR) of A, Q, or B^b with Y
	17	Logical product (AND) of A, Q, or B^b with Y
IRT	53	Interregister transfer (24-bit precision)
	55	Interregister transfer (48-bit precision)
	55.0	Select ISR
	55.4	Select OSR
BCD-shift E, E jump, set D	60	Shift E, E zero jump, set D
77 instruction**	77	Sensing, selecting, interrupt, and control functions

These instructions use RNI to RNI. The first RNI reads the first instruction word, the second RNI reads the second instruction word.

Block operations	71	Search
	72	Move
	73	Input, character
	74	Input, word
	75	Output, character
	76	Output, word

The instruction progression in one instruction is RNI to RADR.

Unconditional jump	01.4	Unconditional jump

*Y = operand (lower 15 bits of the instruction word)

**77.64 (Write PF) and 77.65 (Read PF) fall in this group only if Multiprogramming module is not present. 77.610000 (Test Memory Availability) is excluded from this group.

These instructions use RNI to ROP progression. The RADR sequence is optional and depends on the programmer's choice. Indexing may be used to modify the indirect address and/or the execution address.

Loads	20	Load A (LDA)
	21	Load Q (LDQ)
	22	Load A character (LACH)
	23	Load Q character (LQCH)
	24	Load A complement (LCA)
	27	Load logical (LDL)
	54	Load index (LDI)
Arithmetic or logical	30	Add (ADA)
	31	Subtract (SBA)
	35	Selective set (SSA)
	36	Selective complement (SCA)
	37	Logical product (LPA)
	50	Multiply (MUA)
	51	Divide (DVA)
	52	Compare (CPR) (within limits test)
	77.61	Test memory availability
	77.65	Read Page File

These instructions use RNI to STO progression. The RADR sequence is optional for the 4X instructions but neither indexing nor RADR is possible for the 00.7 instruction. Indexing may be used to modify the indirect address and/or the execution address.

	00.7	Return Jump
Store	40	Store A (STA)
	41	Store Q (STQ)
	42	Store A character (SACH)
	43	Store Q character (SQCH)
	44	Store lower 15 bits (SWA)
	46	Store lower 17 bits (SCHA)
	47	Store index (STI)
	77.64	Write Page File (APF)

One instruction uses RNI to ROP to STO sequence progression. Indirect addressing does not apply to this instruction. The instruction is:

Storage shift	10.0	Storage shift (SSH)

One instruction has a sequence progression of RNI to RADR to ROP to STO. Inclusion of RADR is for indirect addressing. It would not always be necessary; it depends on programmer's choice. Indexing may be used to modify the indirect and/or execution address.

Replace add	34	Replace add (RAD)

Double-precision load instructions use RNI to ROP to ROP sequence progression. RADR sequence could be used with indirect addressing. Indexing may be used to modify the indirect and/or execution address. The instructions are:

Load or arithmetic, 48-bit precision	25	Load AQ (LDAQ)
	26	Load AQ, complement (LCAQ)
	32	Add to AQ (ADAQ)
	33	Subtract from AQ (SBAQ)
	56	Multiply AQ (MUAQ)
	57	Divide AQ (DVAQ)
Floating-point arithmetic	60	Floating-point addition to AQ (FAD)
	61	Floating-point subtraction from AQ (FSB)
	62	Floating-point multiplication of AQ (FMU)
	63	Floating-point division of AQ (FDV)

There is one double-precision store instruction. It uses RNI to STO to STO sequence progression. RADR sequence could be used with indirect addressing. Indexing may be used to modify the indirect and/or execution address. The double-precision store instruction is:

48-bit store	45	Store AQ (STAQ)

Two instructions have optional sequence progressions. Normal progression is RNI to ROP to ROP...to ROP during the search operations; however, if an interrupt occurs, the ROP to ROP sequence is broken and RNI is initiated. These two instructions are:

Search	06	Masked Equality Search (MEQ)
	07	Masked Threshold Search (MTH)

Figure 263 is a block diagram of the possible sequence progressions for instructions in the 3300 Computer.

There are several AND gates (A through R) shown on figure 263. These AND gates (based on instruction function translators) determine which sequence progression will be used by the instruction being executed.

Table 15 is a worksheet based on figure 263, the 3300 Computer System Logic Diagrams, and the previous pages of this chapter. NOTE: AND gates A and C have been filled in for you and are meant to serve as a guide in filling out the other AND gates.

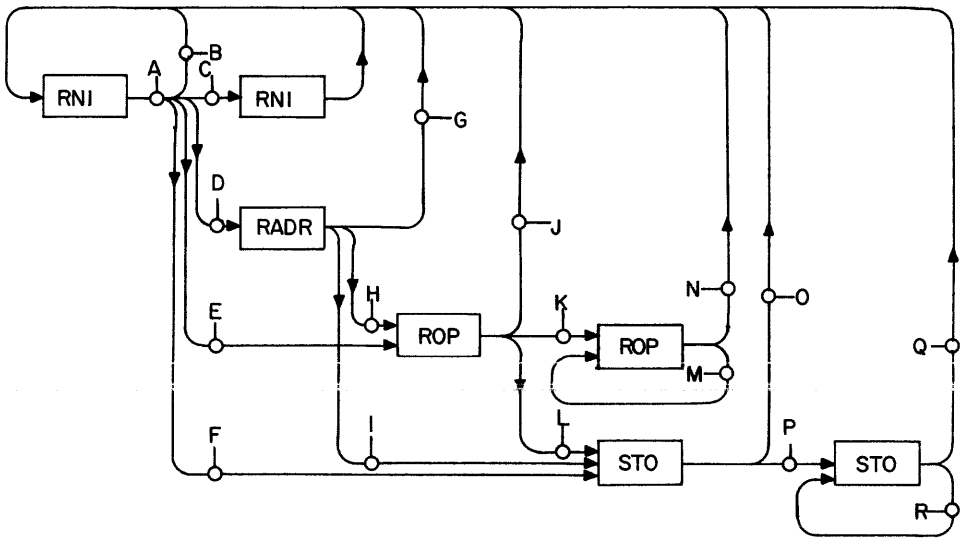


Figure 263. Block Diagram of Sequence Progressions

Table 15. AND GATES FOR SEQUENCE PROGRESSIONS

GATE	TRANSLATION	LOGIC TERM	LOGIC PAGE
A	Arithmetic not busy	V109	2-7
B			
C	71 through 76	F558 and V171	2-5
D			
E			
F			
G			
H			
I			
J			
K			
L			
M			
N			
O			
P			
Q			
R			

The following information traces one instruction through its entire execution. Check it for accuracy against the prints in Logic Diagrams and answer the questions.

Timing for a replace add instruction with one indirect address:

RNI
- N072 Resynced manual start
- V087 Input to RNI
- V014 Request bus on RNI
- N051 Obtain bus priority
- N050 Address stabilization
- V117 Request storage
- N050 Breakpoint comparison timing
- V115 Breakpoint comparison timing
- V116 Test BPI
- N051
- N050
- V061 Storage reply
- V000 Drop bus request
- V001 Access time-out
- V002 Access time-out
- V003 Access time-out
- V004 Access time-out
- V005 Access time-out
- V006 Access time-out
- V007 Clear F and release bus
- V008 Instruction to F
- V009 Instruction decode
- V010 Test interrupt
- V011 Instruction decode
- V080 End of RNI

RADR
- V109 Arithmetic not busy
- V110 Request bus on RADR
- N051 Obtain bus priority
- N050 Address stabilization
- V117 Request storage
- N050
- V115
- V116
- N050
- N051
- V061 Storage reply
- V000 Drop bus request
- V001 Access time-out
- V002 Access time-out
- V003 Access time-out
- V004 Access time-out
- V005 Access time-out
- V006 Access time-out
- V007 Clear lower F and release bus
- V008 Bus to lower F via EXX2
- V009 F decode
- V082 End of RADR

ROP

V109	Arithmetic not busy	
V110	Request bus on ROP	
N051	Obtain bus priority	
N050	Address stabilization	
V117	Request storage	
N050	Breakpoint comparison timing	
V115	Breakpoint comparison timing	
V116	Test BPO	
N051		
N050		
V061	Storage reply	
V000	Drop bus request	
V001	Access time-out	
V002	Access time-out	
V003	Access time-out	
V004	Access time-out	
V005	Access time-out	
V006	Access time-out	
V007	Clear DB register and release bus	
V008	Bus to DB register and start arithmetic	
V009		N691
V084	End of ROP	V500

STO

V109	Arithmetic not busy	V501
V110	Request bus on STO	V502
N051	Obtain bus	V503
N050	Address stabilization	V504
V117	Request storage	V505
N050	Breakpoint comparison timing	V530
V115	Breakpoint comparison timing	V531
V116	Test BPO; N400	
N051	N401 (Clear DBR)	
N050	N41X (EXX2 → DBR, direct)	
V061	Storage reply;	
V000	Drop bus request;	
V001	Access time-out	
V002	Access time-out	
V003	Access time-out	
V004	Access time-out	
V005	Access time-out	
V006	Access time-out	
V007	Release bus	
V086	End of STO	

V087 Input to RNI
V014 Request bus on RNI

For problems 1 through 14 assume Multiprogramming module not present and system not in Executive mode.

1. At best, RNI requires _____ Ø times, _____ usec.
2. At best, RADR requires _____ Ø times, _____ usec.
3. At best, ROP requires _____ Ø times, _____ usec.
4. At best, STO requires _____ Ø times, _____ usec.
5. At best, the total is _____ Ø times, _____ usec.
6. Without indirect addressing the total is _____ Ø times, _____ usec.
7. At best, bus priority per cycle is _____ Ø times, _____ usec.
8. Would the best time you determined for RNI apply to all RNI sequences?
9. Would the best time you determined for RADR apply to all RADR sequences? _____
10. Would the best time you determined for ROP apply to all ROP sequences? _____
11. Would the best time you determined for STO apply to all STO sequences?
12. At best, how much more time would be added to the execution of the replace add instruction if two levels of indirect addressing were used? _____ Three levels? _____
13. Question 5 asks what the best total time would be for an RAD instruction. What different factors could make the total time longer, not counting RADR sequences?
    a. Obtaining the bus during RNI.
    b. Storage reply.
    c. Arithmetic busy at entry to ROP sequence.
    d. _____
    e. _____
    f. _____
    g. _____
14. Why would obtaining the bus cause so much delay?

    _____

    _____

    _____

15. How much time would the use of the relocation feature of the 3300 Computer add to a sequence?

Remember, there is only one bus system. It is shared by program control and block control, each having equal priority to obtain it. Figure 264 shows how the bus system could be used.

It is important to keep in mind that figure 264 is only meant to show how the bus system could be used. It is not meant to show all the uses of the system; the bus system never follows that type of progression.

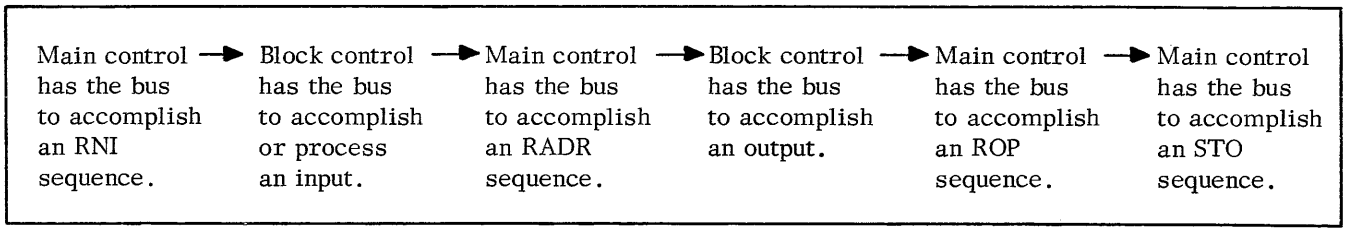


Figure 264. Example of Bus System Usage

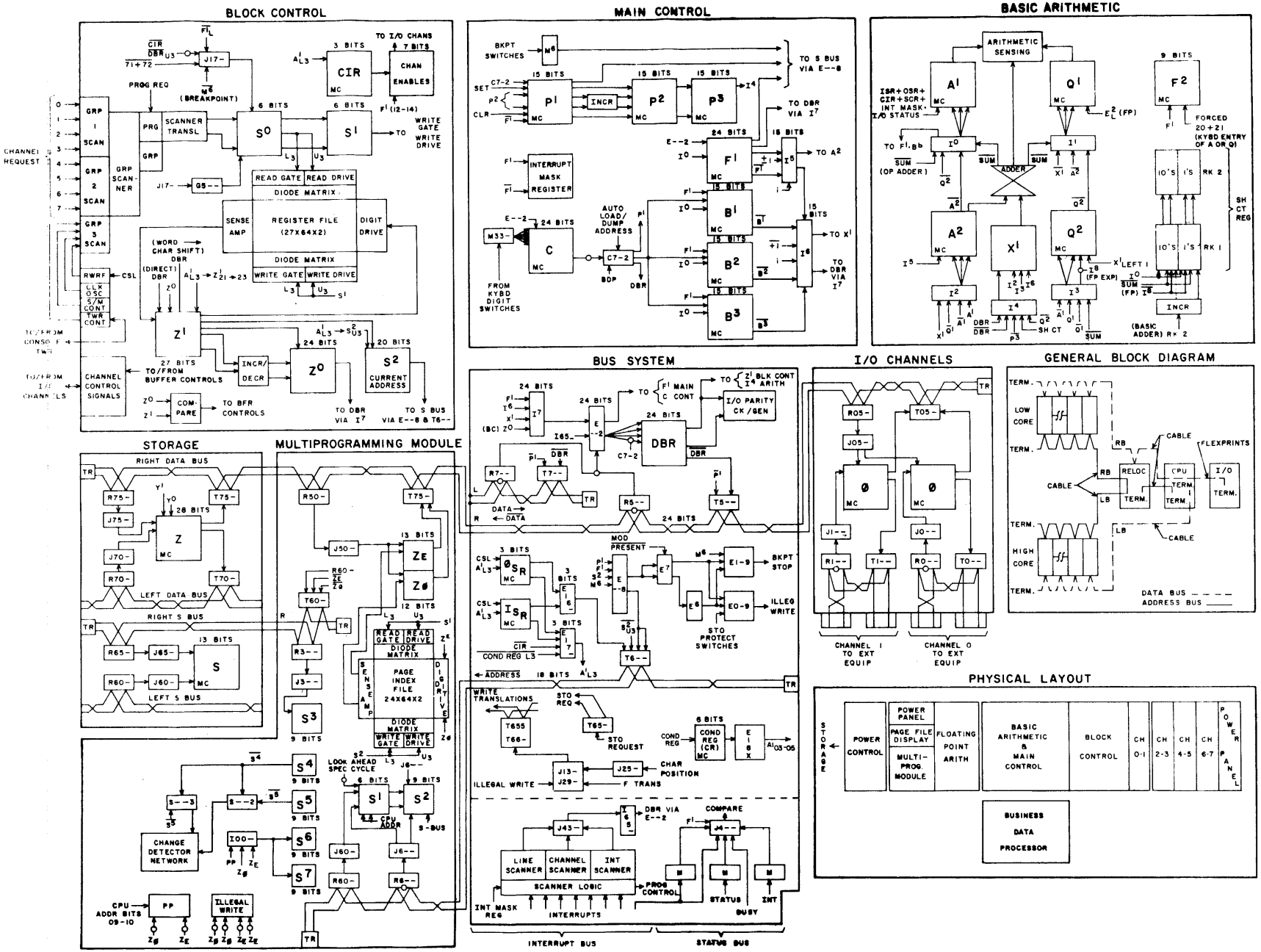


Figure 265. 3300 Block Diagram

SEQUENCES Worksheet

Assume that the Multiprogramming module is not present and the system is not in Executive mode for this problem.

You, the maintenance man, are summoned to the console by the computer operator. He tells you that the load Q instruction from memory location 14063 is not operating properly. He executes a 21 0 14063 and says, "See, the uppermost bit, bit 23, always comes up a 0." (Q) is 12106034 is all the information he can supply.

One of the first things a good maintenance man would do would be to verify that a load Q will not work.

Figure 265, (page 242) is an unmarked block diagram of the system. Draw in the data flow path for an operand moving from storage to Q register. Do this first, then continue.

Study the block diagram, then list the steps of the operation you would perform to verify the trouble.

The following list contains some logical first steps, one of which you might possibly have written above. Look through the list for the operation you would have performed. If the operation you listed is there, follow the directions given.

(1) a. You entered (via keyboard to storage) a load A from location 14063 (the same address referenced by the load Q). You then executed the load A. If load A was your choice, go to step 2, item a.

b. You manually entered (via keyboard) all 7s to Q register (at least to the upper digit position). If this was your choice, go to step 2, item b.

c. You swept contents of location 14063 to C register for observation. If this was your choice, go to step 2, item c.

d. You entered (via keyboard) all 7s into A register and stored the contents of A in memory. Then you executed a load A from that location in storage. If this was your choice, go to step 2, item d.

e. If the operation you performed was not listed above perhaps your choice was not one of the most logical ones you could have made. Analyze your operation. What help is it to you in verifying that a load Q does not work? Perhaps the operation you chose would have helped you and it was just not listed above.

Either way, now that you have read through the list of logical choices supplied above, which one of them do you feel is the most logical?

Reread the problem, make another choice (from a to d above), and follow the directions given.

(2) (a) After execution of the load A from memory location 14063 the contents of A register is 12106034.

Knowledge gained: The uppermost bit coming to A register as a 0 could be significant. How do you know that the uppermost bit (bit 23) in memory is a 1? You do not know, do you? You possibly wasted a bit of time here, especially if the bit in memory is a 0!

All is not wasted though. It's possible that you have learned some important things, even if they are not obvious to you at this point. If an error does exist it has to be in some circuitry that is common to both a load A and a load Q instruction. This knowledge is important!

What circuitry is common to both a load A and a load Q? (List the common flow paths.)

You should have listed storage, transmitters, receivers, EXX2, DB register, I^4, and X1. Study the block diagram in figure 265 and verify that these are the only common circuits. Keep in mind what you might have learned in this step, go back to step 1, and make another choice. Remember, you are trying to verify the existence of an error.

(2) (b) After keyboard entry of all 7s to Q register, (Q) is 37777777. C register contained all 7s when TRANSFER was pushed.

Knowledge gained: Q itself could be bad. The flip-flop for bit 23 possibly does not hold a 1 even if it gets there. There is circuitry other than Q involved during keyboard entry to Q. List the steps of the flow path for keyboard entry to Q. Use figure 265 for reference.

It is known that an error exists. Further, it is known that the error has to be in one of the following circuits in the Q flow path: Keyboard, MXXX, C, C7X2, DB register (EXX2 enable), I^4, X1, I^1, or Q1. Because the 7s got to C correctly, three circuits can be eliminated. On figure 265 draw a line through the circuits that can be eliminated.

The original problem was a load Q not operating. It is known that an error exists now because manual entry failed. C7X2 is used on the manual entry to Q path, but not on a load Q, thus C7X2 can be eliminated also. (It should be crossed out on figure 265, as should M33X and C register.

The error must exist in one of the following circuits: EXX2, DB register, I^4, X1, L1 and Q1.

It is also known that storage is okay (unless there are two errors) because keyboard entry does not involve storage; the enter operation is failing, thus the error could not be in storage. Go to step 3 .

(2)(c) After you swept the contents of location 14603 to C register for observation you found that C contained 52106034.

Knowledge gained: You know memory contains a 1 bit in position 23. You also know that the path from storage to C register is okay. Referring to figure 265, what is the path for information traveling from storage to C?

_____

_____

_____

_____

You should have listed storage, transmitters, receivers, EXX2, and C. You now know there is a fault. Further, you know it is not in the storage circuits because the information comes to C register correctly. What would be a logical choice of operations now?

_____

_____

_____

_____

Look through the following list for the operation you chose and for the directions given:

1) You entered (via keyboard to storage) a load A from location 14063 (the same address referenced by the load Q). You then executed the load A. If load A was your choice, go to step 5 .
2) You manually entered (via keyboard) all 7s to Q register (at least to the upper digit position). If this was your choice, go to step 2 , item b.
3) You entered (via keyboard) all 7s into A register and stored (A) in memory. Then you executed a load A from that location in storage. If this was your choice, go to step 2 , item d.

If the operation you performed was not listed above perhaps your choice was not one of the most logical ones you could have made.

Analyze your operation. What help is it to you? Perhaps the operation you chose would have helped you, but it was just not listed above. Either way, now that you have read through the list of logical choices supplied above, which one of them do you feel is the most logical? Reread the problem, make another choice (from a to c above), and follow the directions given.

(2)(d) As you enter (via keyboard) all 7s to A register you notice that A always comes up with a 37777777, yet C is always all 7s at the time of transfer. There is no need to do the store A or load A because something is wrong in the path from the keyboard to A, and this path does not use storage. Referring to figure 265, list the steps in the flow path for keyboard entry to A register.

_____

_____

_____

_____

It is known that an error exists. Further, it is known that the error has to be in one of these circuits in the A flow path: keyboard, MXXX, C, C7X2, DB register (EXX2 enable), I^4, X1, adder, I^0, A1. Because the 7s got to C correctly, three circuits can be eliminated. On figure 265, draw a line through the three that can be eliminated.

The original problem was a load Q not operating properly. It is known that an error exists now because manual entry failed. C7X2 is used on the manual entry to A path but not on the load Q path, thus C7X2 can be eliminated also. The error must exist in one of the following circuits: EXX2, DB register, I^4, X1, adder, I^0 or A1.

The adder is used for entry to A but not for a load Q, thus the adder can be eliminated, as can I^0 and A1. This leaves just EXX2, DB register, I^4, or X1. Go to step 4 .

(3) You know an error exists because you executed an enter to Q (via keyboard) and it did not function properly. You have already localized the problem and started eliminating circuitry.

You know the error has to be in one of the following circuits: EXX2, DB register, I^4, X1, I^1, or Q1.

What operation could you perform that would eliminate I^1 and Q1? _____

_____

_____

To inject a thought here--why not try keyboard entry to A with all 7s? If it fails, then the fault could not be I^1 or Q1; instead it would have to be in some circuitry common to both an enter to A

and an enter to Q. If it does not fail, then the error has to be in either I^1 or Q1.

Assuming that you took the bait, when you tried to enter to A, C got all 7s but A got 37777777. I^1 and Q1 cannot be faulty. Cross them out on Figure 265.

Now it is known that both an enter to A and an enter to Q fail. The error has to be in circuitry common to both. The following common circuitry could be at fault: C7X2, DB register (EXX2 enable), I^4, and X1.

C7X2 was eliminated in step 2 , item b. EXX2 can be eliminated for a slightly different reason. EXX2 is in the load Q path, but not in the enter path which also fails, therefore EXX2 could not be at fault. (Cross it out on figure 265.) This leaves DB register, I^4, and X1 to be eliminated.

Go to step 6 .

(4) You know an error exists because you executed an enter to A (via keyboard) and it did not function properly. You have already localized the problem and started eliminating circuitry.

You know the error has be be in one of the following circuits: EXX2, DB register, I^4, X1.

EXX2 can be eliminated for a reason similar to the one for which C7X2 was eliminated earlier.

EXX2 is in the load Q path, but it's not in the enter to A path which also failed. Therefore, EXX2 could not be at fault. This leaves DB register, I^4, and X1 to be eliminated.

Go to step 6 .

(5) After execution of the load A from memory location 14063, (A) is 12106034.

Knowledge gained: The uppermost bit coming to A register as a 0 has eliminated considerable circuitry. An error does exist and it has to be in some circuitry common to both a load A and a load Q instruction. This knowledge in itself is important.

What circuitry is common to a load A and a load Q? (List the steps in the common flow path.)

_____

_____

_____

_____

_____

You should have listed storage, transmitters, receivers, EXX2, DB register, I^4 and X1.

Study figure 265 and verify that these are the only common circuits.

The sweep operation that you performed earlier eliminated the storage circuits and an inverter rank. What three circuits have not been eliminated? _____

_____

_____

Go to step 6 .

(6) The bad circuit has been narrowed down to DB register, I^4, or X1.

Keep in mind that the fastest troubleshooting procedure is to eliminate as much circuitry as possible with operations executed from the console.

What operation could be performed to point at or to eliminate X1? _____

_____

_____

Another hint: X is fed by inverter ranks 2, 3, 4, and 6. (Look at figure 265.) It doesn't figure that X could be eliminated if an operation that used X also used I^4; so choose an operation that does not use I^4. Ranks 1, 2, and 3 are in the A and Q circuits, and so far it has not been possible to get bit 23 into either of these registers, so that pretty much leaves you with I^6.

I^6 feeds X1 and I^6 is fed by the B boxes and I^5. I^5 seems like a good bet. Enter an enter Q instruction with a negative number and sign extension, then execute it.

Doing an enter Q leaves all 7s in Q. X1 must be okay (cross it out on figure 265), leaving DB register and I^4 as possibles.

Go to step 7 .

(7) Some more facts are known. The error has to be in either DB register or I^4.

It is possible to get all 7s into Q via the enter instruction with sign extension.

Considerable benefit comes from being able to get all 7s into the register. Now the 7s can be stored from Q and further circuits can be eliminated.

When a store is followed by a sweep to C, all 7s return to C. Which circuit, DB register or I^4, can be eliminated? _____

Cross it out on figure 265. DB register should be okay because the path for a store Q is:

$\overline{Q1} \rightarrow I^3 \rightarrow X1 \rightarrow I^7 \rightarrow EXX2 \rightarrow \overline{DBR} \rightarrow$ storage

$\overline{Q1}$, I^3, X1, I^7, and storage are not involved in the circuits that are to be eliminated. Thus the error has to be in _____ .

Go to step 8 .

(8) YEP! I^4.

Now what? Locate the logic card and replace it. What manual would you use to find the physical location of the 23rd bit of I^4? _____

The logic diagrams for the 3304 Basic Processor would be a good bet. Look in the table of contents for I^4 (standard arithmetic section). Turn to that page in the diagrams and locate the 23rd bit and list its physical location.

You should have 2K88A. Go ahead, replace that card and try a load Q from location 14063. After execution, Q contains a 52106034, which is correct. Congratulations for fixing the trouble.

There were many steps to this problem. You possibly did not agree with the sequence of steps through which you were led in diagnosing the problem but if you remember these important things, you'll do well.

1. Always prove or disprove the first assumption before you fix anything.
2. Localize the problem to one area of the system as rapidly as possible by performing operations that use common circuits.
3. First try to eliminate all possible common circuitry via the console. It's faster than probing around with a scope. Besides, if you can push a few buttons, you make a better impression than when you state it must be so-and-so and replace it without even using a scope.
4. Repair the faulty component.

Only a very few instructions in the entire repertoire may be executed. All other instructions "die" leaving the console in the following configuration depending on which sequence the failing instruction would use to accomplish its execution. If RADR, ROP, or STO sequence were required, the console would be dark except for the appropriate sequence indicator being lit.

Instructions that can be executed are all common to RNI sequence:

Stops
Jumps
Skips
Shifts or scale
No-address arithmetic instructions, enters or logic
Intraregister transfers
77 series (connect, senses, etc.)

It is also possible to initiate block operation.

It should be already apparent that RNI is okay. Whatever is wrong must be common to all other sequences.

CLUES:

1. Could storage be at fault? _____ If the program were written so that all of the instructions were in storage module 0 and all operands were in module 1, could storage be at fault? _____
   _____
   _____

2. This problem is common to several sequences. The most logical place to look for the fault would be in control. Most of the common circuitry for the sequences is shown on page _____ of Logic Diagrams.

3. Could the arithmetic section be at fault? _____
   If RADR and RNI worked and only ROP and STO failed, could the arithmetic section be at fault? _____
   Why? _____
   _____
   _____
   _____

In your own words relate what you think would be at fault: _____
_____
_____
_____
_____

In diagnosing a problem you have discovered the following facts:

1. The arithmetic area works; it adds, subtracts, multiplies and divides. It also shifts correctly.
2. You can store and retrieve information from memory because you keyed in a STA and a LDQ instruction and they both operated correctly.
3. The computer reads and executes instructions correctly. Proof is in item 2.
4. Instruction address modification works because you b-modified an instruction and it worked.

From the facts you have learned in the four operations you performed you feel quite sure the error is in one sequence.

What sequences are checked by item 2? _____ and _____
What sequence is checked by item 3? _____ and _____
Are any additional sequences checked by item 4?
_____
What sequence has not been checked? _____
Supply one instruction that will check the remaining sequences.

It appears that a modified load A instruction is not operating correctly. The instruction arrives in F register properly, in program control.

When the instruction is executed the number that arrives in A register is not the proper one. Listed below are the internal sub-sections of the 3300 processor. If the sub-section could be at fault, write in yes.

If not, write no. Whether your answer is yes or no explain why.

Block Control? _____, because _____

Program Control? _____, because _____

Standard arithmetic, first pass? _____, because _____

Standard arithmetic, second pass? _____, because _____

Storage? _____, because _____

Multiprogramming module? _____, because _____

I/O? _____, because _____

S bus? _____, because _____

D bus? _____, because _____

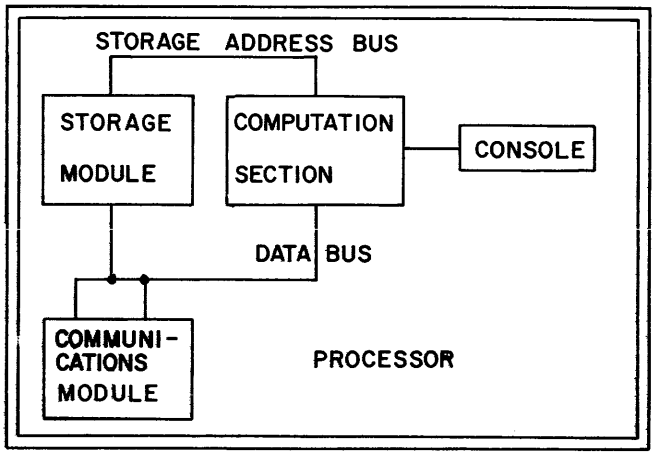


Figure 266. System Block Diagram

CHAPTER 12
ARITHMETIC CONTROLS

GENERAL DESCRIPTION

The standard arithmetic section of the processor is located within the computer section of the system (figure 266).

The system block diagram of the 3204 Processor is shown in figure 267. Note the inputs to and outputs from the standard arithmetic section.

Figure 268 shows the detailed block diagram of the standard arithmetic section. Throughout this chapter references are made to figures 267 and 268. Study them now and refer back to them each time they are mentioned.

The arithmetic section performs all loads, stores, shifting, scaling, logical, and arithmetic operations. This chapter discusses the logical operation of the arithmetic registers, the adder pyramid, the arithmetic timing controls, and the arithmetic sensing logic.

Figure 268 is a block diagram of the arithmetic section which shows all paths for data exchange within the arithmetic section.

REGISTERS

The arithmetic registers provide temporary storage for the operands used in program execution. The arithmetic section has five registers: A1, A2, Q1, Q2, and X. The contents of A1 and Q1 are normally displayed on the console when the computer is stopped.

As each register is described in the following discussions, refer to figure 268 to place it in the proper perspective to the entire section. Also, refer to it in 3300 Logic Diagrams to see all bits represented.

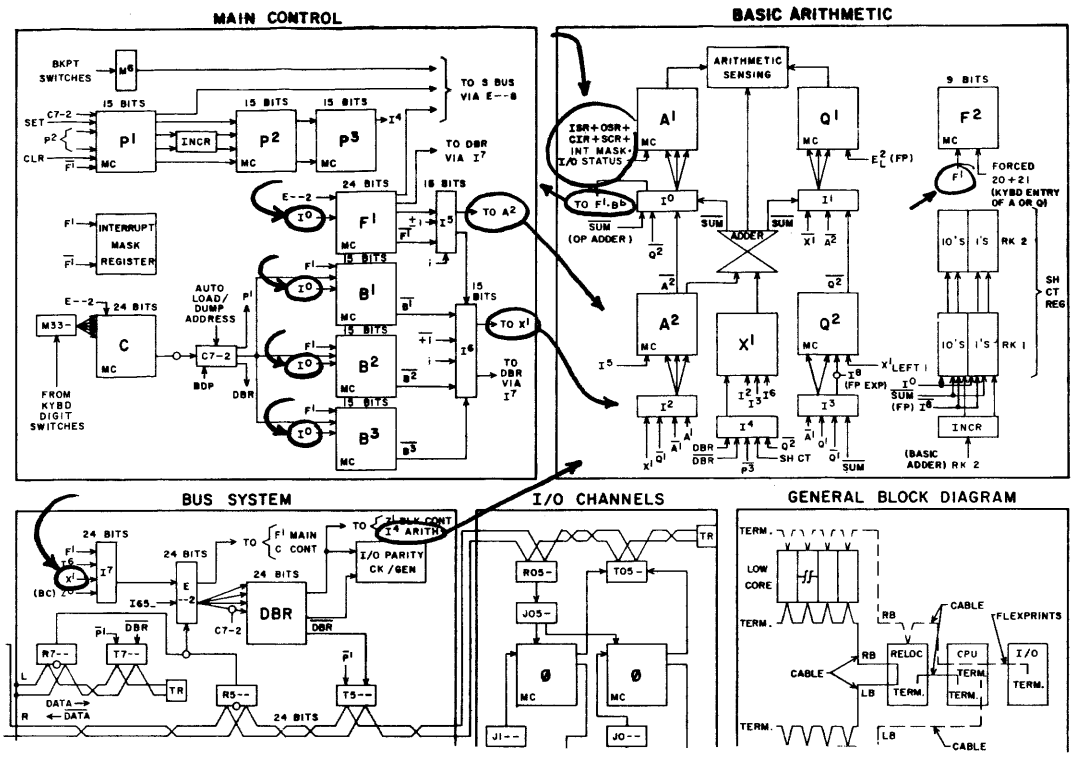


Figure 267. Arithmetic Inputs and Outputs

A1 Register

The 24-bit A1 (arithmetic) register is used in nearly all arithmetic and logical operations. (A1) may be transferred to A2 and (A2) added to the contents of a storage location or the contents of another register. The sum is then placed in A1 or it can be placed in storage. (A1) may be shifted right or left, and separately or in conjunction with Q1. A1 may also hold a control quantity which conditions certain jump, skip and search instructions.

Inputs to A1 are made from:

1. J4XX inverters and M4XX FFs. These inputs transfer the internal status code to bits 00-11 of A1 and the contents of the interrupt mask register to bits 12-23.
2. I^0 inverter rank. The sum from the main adder, sum from the optional adder, and (Q2) are transferred through I^0 into A1 directly. (A2) is transferred through I^0 into A1 either directly, left shifted (end-around) one place, or right shifted (end-off, sign extended) one place.
3. I^1 inverter rank. Bit 23 of I^1 feeds bit 0 of A1 during a left shift AQ or left shift A.
4. E175-E177 and E185-187 inverters. These inputs are used to transfer the contents of the condition register, ISR, and OSR to A_L.

Refer to Logic Diagrams, page 2-127, for bits 0-7 of A1 register and I^0 inverter rank.

Q1 Register

The 24-bit Q1 register (auxiliary arithmetic) assists A1 register in performing arithmetic and logical operations. (Q1) may be shifted right or left, separately or in conjunction with A1. Q1 may also be used with A1 register to form double-length register AQ. Q1 serves as a mask register for certain instructions.

Inputs to Q1 are made from:

1. E_L (optional arithmetic) register. The inputs are E4XX, E5XX, and E6XX.
2. I^1 inverter rank. The sum from the main adder and (X) are transferred through I^1 directly into Q1. (Q2) and (A2) are transferred through I^1 into Q1 either directly, left shifted (end-around) one place, or right shifted (end-off, sign extended) one place.
3. I^0 inverter rank. This rank is used together with the I^1 rank when shifting Q or AQ.

Refer to 3300 Logic Diagrams, page 2-133, for bits 0-7 of Q1 register and I^1 inverter rank.

A2 Register

The 24-bit A2 register is used primarily as one of the adder feeders. Data placed in A2 and X immediately begins to propagate through the adder. A2 is also used in conjunction with A1 register for operations involving a shift of A or AQ and to allow an exchange of (A) and (Q) during the execution of multiply and divide instructions.

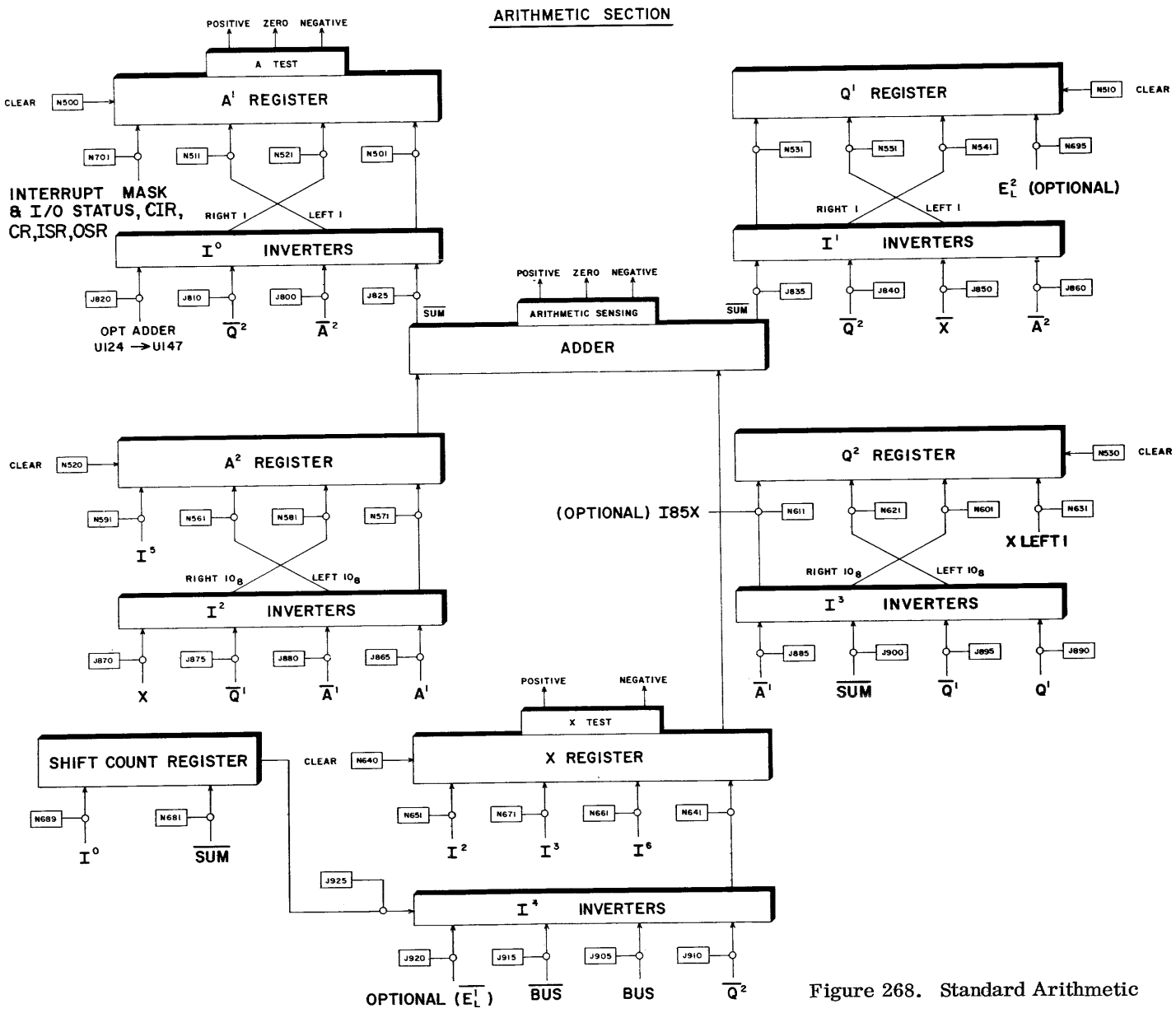


Figure 268. Standard Arithmetic

Inputs to A2 are made from:

1. I^5 inverter rank. (F_L), ($\overline{F_L}$), +1, -1, and the complement of the interval designator are transferred from the main control section through I^5 into A2.
2. I^2 inverter rank. ($\overline{X}$) and ($\overline{A1}$) are transferred through I^2 directly into A2. (A1) and (Q1) are transferred through I^2 and I^3 into A2 either directly, left shifted (end-around) 10_8 places, or right shifted (end-off, sign extended) 10_8 places.
3. I^3 inverter rank. This rank is used with I^2 when shifting A or AQ.

Refer to Logic Diagrams, page 2-139 for bits 0-7 of A2 register and the I^2 inverter rank.

Q2 Register

The 24-bit Q2 register is used in conjunction with Q1 register for operations involving a shift of Q or AQ and for an exchange of the contents of A and Q (swap AQ). Q2 is also used with X register to perform a left shift 1 of (X).

Inputs to Q2 are made from:

1. X register. (X) is always transferred to Q2 left shifted one place.
2. I^3 inverter rank. The sum from the main adder and ($\overline{Q1}$) are transferred through I^3 directly into Q2. (A1) and (Q1) are transferred through I^3 and I^2 into Q2 either directly, left shifted (end-around) 10_8 places, or right shifted (end-off, sign extended) 10_8 places.

3. I^2 inverter rank. This rank is used with I^3 when shifting Q or AQ.

Refer to Logic Diagrams, page 2-145, for bits 0-7 of Q2 register and I^3 inverter rank.

X Register

The 24-bit X register is an adder feeder register like A2. X register is the only arithmetic register with provisions for exchanging information with storage.

Inputs to X are made from:

1. I^2 inverter rank. (A1) are transferred to X via I^2.
2. I^3 inverter rank. (Q1) and $(\overline{Q1})$ are transferred to X via I^3.
3. I^6 inverter rank. (B^b) and (F_L) are transferred to X via I^6. The actual path for F_L is $F_L \rightarrow I^5 \rightarrow I^6 \rightarrow X1$.
4. I^4 inverter rank. (DBR), $\overline{(DBR)}$, (P_3), and (Q2) are transferred to X via I^4. The shift count is transferred into bits 0-5 of X via I^4.

Refer to Logic Diagrams, page 2-151, for bits 0-7 of X register and I^4 inverter rank.

INVERTER RANKS

Most of the parallel transmission paths between registers involve a rank of inverters. The major parallel transmission paths and inverter ranks of the arithmetic section are shown in figure 268.

I^0, I^1, I^2, I^3, and I^4 are the arithmetic section inverter ranks which serve as the main feeders to registers A1, Q1, A2, Q2, and X.

The various transfer paths for the inverter ranks are listed in table 16.

Table 16. INVERTER RANK TRANSFER PATHS

RANK	INPUTS	OUTPUTS
I^0	$\overline{Sum}$, $\overline{\text{optional sum}}$, $\overline{A2}$, $\overline{Q2}$: sign to Q230, right 1→bit 0; Q730 to Q000, left 1→bit 23; quotient bit (to bit 23)	A1 (direct, left 1, or right 1); shift count register; bit 23→bit 0 of Q1; bit 0→bit 23 of Q1; lower 15 or 17 bits to F_1 index registers B1, B2, B3
I^1	$\overline{Sum}$, $\overline{Q2}$, $\overline{X}$, $\overline{A2}$	Q1 (direct, left 1, or right 1); bit 23→bit 0 of A1
I^2	$\overline{A1}$, $\overline{Q1}$, X, A1; sign of Q2 for right 10_8 (to bits 0-7 of I^2); block bits 15-23 of I^2 for transferring $\overline{A1}$ lower 15 → I^2 during 04 or 05 inst.	X, A2 (direct, left 10_8, or right 10_8); bits 16-23→bit 0-7 of Q2; bits 0-7→bits 16-23 of Q2
I^3	$\overline{A1}$, $\overline{Q1}$, $\overline{sum}$, Q1	X, Q2 (direct, left 10_8 or right 10_8); bits 16-23→bits 0-7 of A2
I^4	$\overline{Q2}$, $\overline{DBR}$, $\overline{E1_L}$, DBR; shift count→bits 0-5	X

ADDER

The 3300 Adder is an inverter network which forms the sum (or difference), logical product (AND function), or selective complement (exclusive OR function) of two 24-bit quantities. The 3300 Adder is an additive accumulator whereas the adders of most other Control Data computers are subtractive accumulators.

There are 24 stages in the adder, one stage associated with each bit in the two feeder registers, A2 and X. Inputs to the adder are made only from these registers. The data is gated into the feeder registers. Operations within the adder are not clocked, so four phase times (including the time that data is being placed in the feeders) are allowed for adder propagation before the output of the adder is sampled.

The 3300 Adder is designed so that it may be extended in size to 48 bits without increasing adder propagation time. The optional floating-point/double-precision hardware package contains the logic necessary to do this.

The expanded adder is used only for the floating-point/double-precision instructions. When any other instruction is being executed the optional adder is disregarded.* One operation that occurs quite frequently in the 3300 computer is that of using the adder as a data path in which case a value is added to zero. The sum (equal to original value) is transferred to A register (example-the LDA Inst.). No real addition has occurred but DATA has been moved. *Note: The double precision add or subtract do not require optional hardware.

BINARY ARITHMETIC

The binary number system is the basis for the representation and manipulation of all information within the computer. The 3300 Computer uses 1's complement binary arithmetic.

The adder forms the sum of two numbers by adding them directly. For example:

```
 0111   augend
+0100   addend
 0011   partial sum
 1      carry
 1011   sum
```

Subtraction is performed by the "adding the complement" method. The difference of two numbers is found by first complementing the number to be subtracted, then adding the complement of the subtrahend to the minuend.

For example:

```
 1100   minuend
-0010   subtrahend
 1010   difference
```

The computer performs the subtraction thus:

```
 1100   minuend
+1101   complement of the subtrahend
 1001   partial difference
+   1   end-around carry
 1010   difference
```

These references contain a further description of number systems, including binary arithmetic: 3300 Reference Manual (see appendix section) and An Introduction to Digital Computers.

THEORY OF THE ADDER

The adder generates and recognizes the carry signals which occur when two numbers are added.

There are four cases which must be considered in binary addition. These are:

*The double-precision add or subtract do not require optional hardware.

	1. Satisfy	2. Pass
Augend	0	0
Addend	+0	+1
Sum	0	1

3. Pass	4. Generate
1	1
+0	+1
1	0 with a carry of 1 to the next higher stage

Satisfy: For case 1, the stage will satisfy an incoming carry. That is, the stage does not generate a carry during addition and, if a carry input is received, it will not be passed on to the next higher stage.

Pass: For cases 2 and 3, a carry input cannot be satisfied within the stage and must be passed on to the next higher stage. Thus, a pass is generated by all stages with unlike input signals from A2 and X.

Generate: For the addition of 1 + 1, case 4, a partial sum of 0 is obtained and a carry is generated and passed on to the next higher stage. A carry input is not satisfied by a stage of this type for it always generates a carry of its own.

Signals within the adder form a double pyramid as shown in figure 269. The adder has 24 input stages, one for each bit of A2 and X. These stages are combined into eight 3-bit groups. Stages and groups are checked for carry generation and ability to pass a carry. Multiple-group generate and pass signals are then formed. Group carry inputs and stage carry inputs are generated next. Finally, the stage-generated signals are compared with the carry input signals at a rank of output inverters. These inverters produce the complement of the sum of X and A2.

The generation of individual signals is treated more fully in the discussion which follows.

If the double-precision/floating point option is present, an identical 24 bit adder unit is wired to the left, or above the standard arithmetic adder.

This provides: 48 stages (0-47)
16 groups (0-15)
4 sections (0-3)

The standard arithmetic adder is sometimes referred to as the right adder and the expanded adder as the left adder. In any case the adder works as a 48-bit adder, the output of which is always $\overline{\text{sum}}$.

Stage-Generated Signals

During addition* the first level of adder translation

*The description to follow is concerned with the use of the adder in true addition only. The adder can also form the logical product and the selective complement of two numbers, but this is discussed elsewhere.

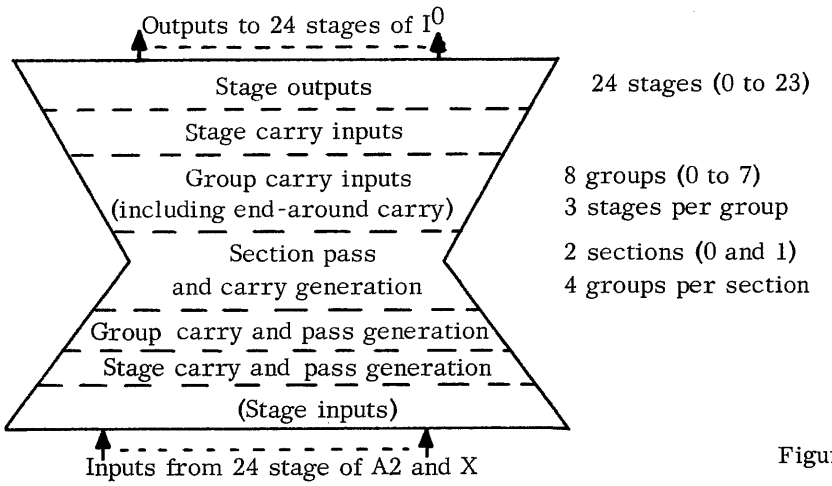
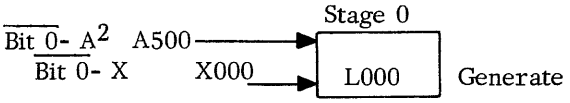


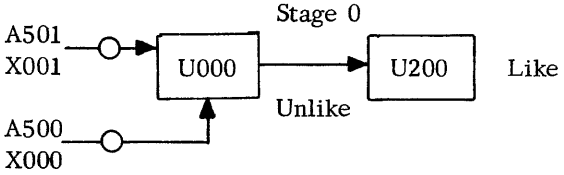
Figure 269. Adder Pyramid

occurs at the individual stages. All 24 input stages generate three types of signals:

1. Generate: A stage carry is generated by the L0XX terms (where XX is the number of the stage) when the corresponding bits of A2 and X are both 1's.



2. Unlike: The U0XX inverters output a 1 when the corresponding bits of A2 and X are unlike. This signifies a stage pass condition. In this case, this bit in the true sum will be a 1 unless a carry input is made.

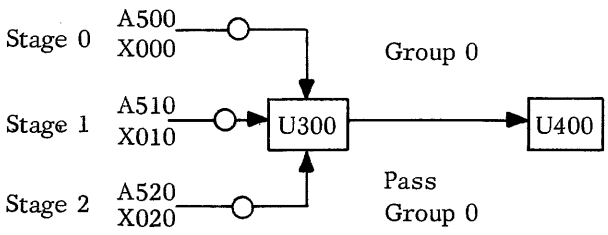


3. Like: The U2XX inverters output a 1 when the corresponding bits of A2 and X are both alike. There is no distinction made between both 1's or both 0's. In this case, the bit in the true sum will be a 0 unless a carry input is made.

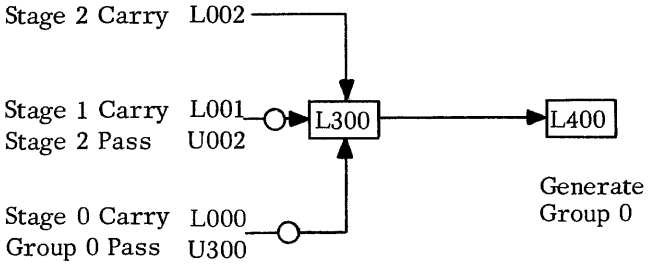
Group-Generated Signals

Second-level translations are made on eight groups of three stages each. Two types of translations are made:

1. Group Pass: A group pass (U3XX = 1) occurs only when there is no satisfy within the group. If both bits of any stage within the group are 0, U300 is driven to a 0. There is then a group satisfy.



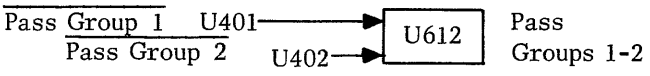
2. Group Generate: A group carry is generated (L40X = 1) only if one of the stages within that group generates a carry and no higher stage within the group generates a satisfy.



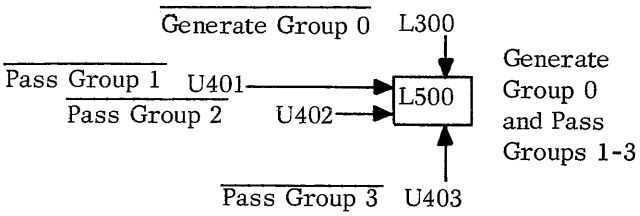
Multiple-group Pass and Carry Generation

Third-level translations are made on multiples of groups. The translations are:

1. Multiple-group Pass: The U6XX inverters generate multiple-group pass signals. These indicate that several consecutive groups contain no satisfy.



2. Group Generate and Multiple-group Pass: The eight groups are considered as two sections, one containing groups 0-3, the other containing groups 4-7. Pass translations from groups 8-15 are present



only when floating-point/double-precision hardware is in the computer. The L50X inverters come up only when a group generates a carry and all higher order groups in that section pass the carry.

End-Around Carry

An end-around carry is generated when any group produces a carry and that carry is passed by all higher groups, including group 7 (group 15 for floating-point/double-precision). Figure 270 shows the logic which recognizes and inserts the end-around carry.

Special logic is required to handle the end-around carry for 48-bit precision add and subtract (32 and 33 instructions).

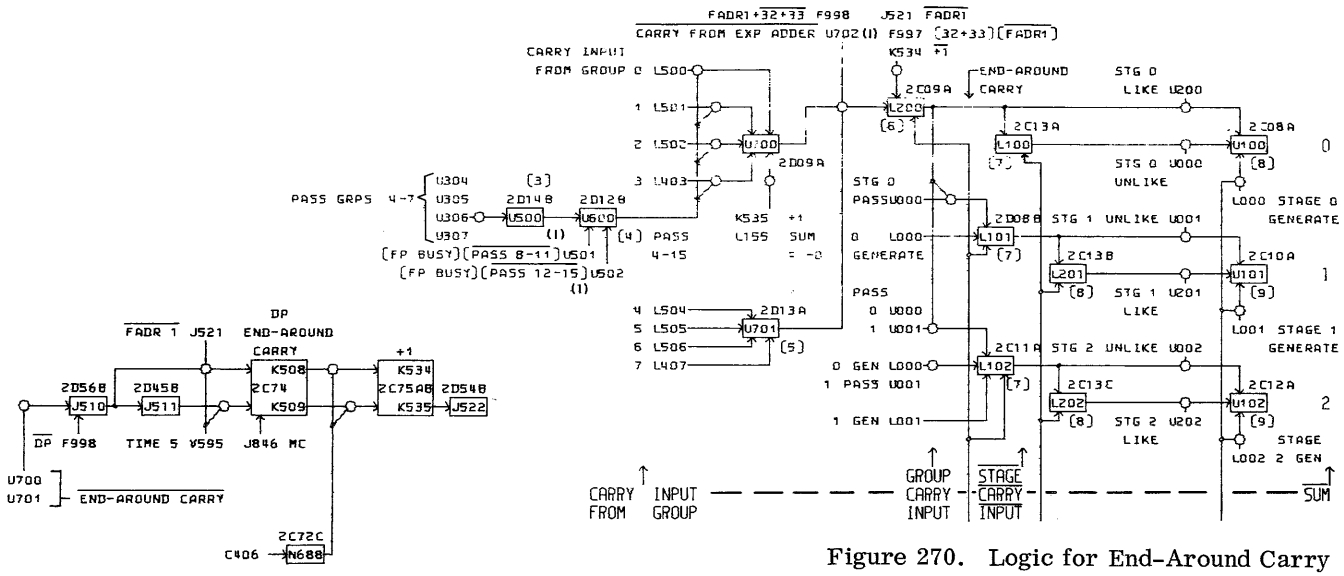


Figure 270. Logic for End-Around Carry

Group Carry Inputs

The fanout of adder signals begins with these fourth-level translations. The carries and group pass signals have now been determined. At this level, the multiple-group pass and generate signals are translated for each output group by inverter networks (shown in the diagrams), to determine if that group has an incoming carry from a lower group. The end-around carry is the group carry input for group 0. (See figure 270).

Stage Carry Inputs

Carry inputs are made to the rank of output inverters. These are compared with the original stage translations to determine the final sum.

The group carry input is inserted into the lower stage of each group, thus effectively serving as the stage carry input.

A stage carry input to the second stage of each group is made if the lower stage generates a carry or passes an incoming group carry.

A stage carry input to the third stage of each group is made if the lower stages generate and pass a carry, or if they pass an incoming group carry.

Stage carry translators are shown in figure 270 (L10X, L20X).

Generation of Final Outputs

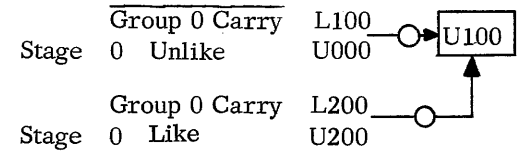
The output of the adder is obtained from the U1XX inverter rank (figure 270). This output is the complement of the true sum.

The determination of the final output is made at this rank. The original stage translations, like and unlike, are compared with the group and stage carry translations.

Four possible cases must be translated (two AND gates required) to determine the output.

Two conditions produce a true sum of 1 (U1XX = 0):

1. A2 and X unlike (U000 = 1), no carry input (L100 = 1).
2. A2 and X like (U200 = 1), carry input (L200 = 1).



The other two possible conditions produce a true sum of 0 (U100 = 1):

1. A2 and X unlike (U000 = 1), carry input (L100=0).
2. A2 and X like (U200 = 1), no carry input (L200=0).

Note that the last two conditions will cause both gates into U100 to be disabled by zeros.

ADDER Worksheet

The following two programs are intended to provide static additions that will help in troubleshooting the adder. All of the terms in each rank will provide the same output under normal conditions. Indicate the output for each rank for each program by filling in the table below.

Master clear and press GO for each of the following separate programs.

SIGNAL	PROGRAM #1 00000: 14.400000 00001: 15.400000 00002: 77.770000	PROGRAM #2 00000: 14.400000 00001: 15.477777 00002: 77.777070
L0XX: Stage generate		
U0XX: Stage unlike		
U2XX: Stage like		
L3XX: Not group generate		
L4XX: Group generate		
U3XX: Group pass		
U4XX: Not group pass		
L5XX: Section generate		
U5XX: Not groups pass		
U6XX: Groups pass		
U7XX: Not group carry input		
L2XX: Group carry input		
Stage carry input		
L1XX: Not group carry input		
Not stage carry input		
U1XX: Not sum		
I0XX: Sum		

This worksheet will familiarize the student with the adder and, when completed, the sheet will aid in troubleshooting adder malfunctions.

Referring to Logic Diagrams (Adder, pages 2-119 to 2-125 and the previous discussion in this chapter, circle the proper answers for each question based on the following problem.

Problem: Add 77777777
to 77777777

1. All L0XX terms will be outputting 1's/0's because all stages/groups are/are not generating.
2. The U0XX terms will be outputting 1's/0's because all bits are/are not the same.
3. The U2XX terms will be outputting 1's/0's because all stages are/are not identical.
4. The U3XX terms will be outputting 1's/0's because they are/are not satisfying in the stages/groups.
5. The L50X terms have 1's/0's out because all/none of the stages/groups are/are not generating.

DOUBLE-PRECISION EAC

A special case of end-around carry is double-precision add or subtract explained below.

A 48-bit add or subtract is performed using two, three, or four passes through the adder. The steps necessary for the add portion of the instructions are:

1. The upper 24 bits of the two 48-bit operands [(A) and (M)] are added during the first pass through the adder. An end-around carry is blocked because the +1 FF is clear, holding L200 to a 0 (figure 269). If an end-around carry occurs (U700 or U701 = 0), EAC (end-around carry) FF and the +1 FF are set.

2. On the second pass through the adder the lower 24 bits of the two operands [(Q) and (M+1)] are added. A carry from the upper bits (if present) is inserted into group 0 of the adder (+1 FF is set, so L200 = 1). This addition may also produce an end-around carry. If so, EAC and +1 are set; otherwise they are cleared.

3. A third pass through the adder then begins, but is allowed to continue only if an end-around carry has occurred. On this pass, +1, the carry from the lower 24 bits is added to the upper 24 bits of the 48-bit sum. Once again this addition can produce an end around carry. EAC and +1 are allowed to remain set if carry occurs, or cleared if it does not occur.

4. A fourth pass is now made. The carry from the upper bits is added to the lower 24 bits of the sum. There can be no more carries so EAC and +1 are cleared. A fifth adder pass begins but it is blocked. An example of 48-bit addition that takes four passes through the adder is shown below;

Upper 24 bits	Lower 24 bits	
77777777_8	77777777_8	
00000000_8	00000001_8	
77777777_8		First Pass
	00000000_8	Second Pass
Carry $+1_8$		
Carry 1_8 $\overline{00000000}_8$		Third Pass
	Carry $+1_8$	
	$\overline{00000000}_8$	Fourth Pass
00000000	00000001_8	Final Sum

LOGICAL PRODUCT

The adder is used to form the logical product (AND function) of two numbers for instructions 17, 27, 37, and the search 2 cycle of the 06 and 07 instructions.

The logical product is formed according to the following rules:

A2	0	0	1	1
X	0	1	0	1
Logical product	0	0	0	1

The translation of a 17, 27, or 37 allows I^{070}-I^{078} to output 1's (figure 271). All stage and group carry inputs are held to 0 by these inverters, locking out all interaction between the 24 stages. The logical product is determined then only by input stage translations.

As shown in figure 271, a bit in the logical product is a 1 only if a carry is generated by the input stage.

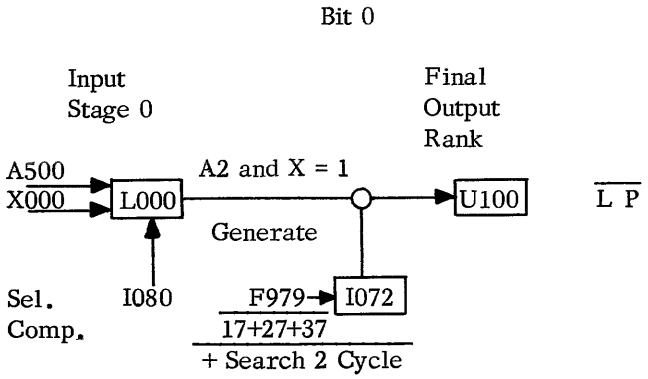


Figure 271. Logical Product Generation

SELECTIVE COMPLEMENT

The adder is used to form the selective complement (exclusive OR function) for instructions 16 and 36. Exclusive OR means one term or the other but not both).

The selective complement is formed according to the following rules:

A2	0	0	1	1
X	0	1	0	1
Selective complement	0	1	1	0

The translation of a 16 or 36 allows I^{080}-I^{082} to output 1's (figure 272). These inverters prevent input carry generation by holding all L0XX (generate) terms to 0. Since stage carries are locked out, there is no interaction between adjacent stages.

As shown in figure 272, a bit in the selective complement is a 1 only if A2 and X are unlike. Adder output is the complement of the selective complement.

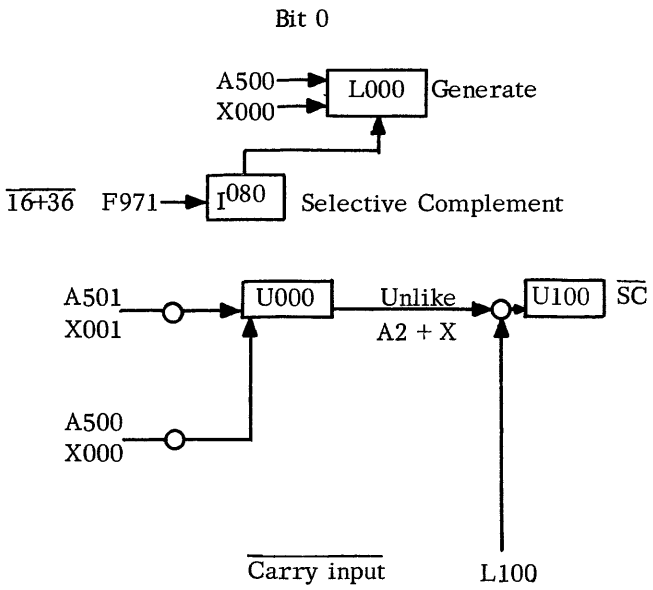


Figure 272. Selective Complement Generation

ARITHMETIC CONTROLS

All arithmetic operations must be timed and signals must occur in proper sequence. Timing and sequencing is determined by arithmetic controls which include F2 register, arithmetic function translators, arithmetic timing chain, shift cycle timing chain, complement and swap cycle timing chain, inverter enables, flip-flop translations, and the shift count register.

Although the arithmetic section contains independent timing logic, it is directed by the main control section. Main control reads an instruction from storage and, if the execution of that instruction involves an arithmetic operation, main control sends a start pulse to the arithmetic section at the proper time.

Once indexing and/or indirect addressing have been performed, the final execution of enter, shift, increase, logical, load, and arithmetic instructions is performed entirely by the arithmetic section with no aid from main control. This frees main control to read the next instruction while arithmetic simultaneously completes the current instruction.

A relatively long time is required to process certain instructions. It is possible that main control will have read another instruction before arithmetic has finished processing the current one. Therefore arithmetic busy FF is set (for multiply, divide, shift, BCD, and floating-point and some double-precision instructions) to lock out any further start pulses. If the new instruction requires use of the arithmetic section, operations in main control halt after RNI and do not continue until arithmetic operations are finished and arithmetic busy signal drops. If an arithmetic reference is not required by the new instruction, main control continues to operate and will execute that instruction.

F2 REGISTER

The contents of the upper nine bits of F1 are duplicated in F2 arithmetic register when an instruction is read from memory. This takes place during every RNI. F2 register which controls the arithmetic enables is decoded separately from the main control function translations. To understand the necessity for using F2 for arithmetic function translations consider this case:

1. Main control reads an instruction which requires a lengthy arithmetic operation.
2. Main control duplicates (F1) in F2 and starts the arithmetic timing chain. Main control has no further operations to perform in executing this instruction.
3. Therefore main control initiates a memory reference for a new instruction word while the arithmetic section is executing the present instruction.
4. The new instruction word is received by main control and gated into F1 while the arithmetic section is still executing the previous instruction. (F1) is not duplicated in F2 until the arithmetic section has completed execution of the previous instruction.

Note that step 3 effectively decreases program execution time, while step 4 insures proper instruction execution of the new instruction in F1.

The function code in F1 is duplicated in F2 by the logic shown in figure 273. The gating signal which transfers the new word into F1 (N206) also sets initiate F1→F2 and the wait function. Wait function locks out a start arithmetic pulse until the upper nine bits of F1 have been duplicated in F2 and decoded.

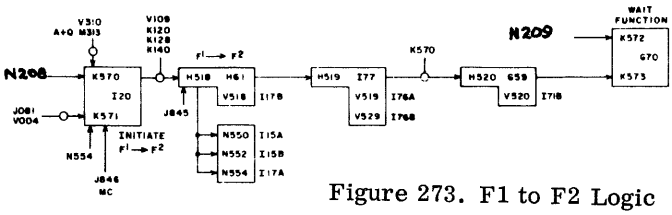


Figure 273. F1 to F2 Logic

During the first odd time after initiate F1→F2 FF is set, a pulse H518 begins the F code transfer. However, this pulse is blocked under two conditions:

1. Interrupt sync, trap sync, or powerfail FFs are set. The current instruction is superseded by one of these operations and is reread when the interrupt has been processed.
2. Arithmetic is busy (V109 = 0). F1→F2 is blocked until busy signal drops. If main control must start arithmetic before doing the next RNI, it hangs up (operations halt) and waits until busy drops, F1→F2 occurs, and wait function is cleared. This hangup occurs at the N6XX terms (start arithmetic).

If it is not necessary to use arithmetic, main control continues instruction execution (figure 274).

Shown are three bits of F2 which duplicate bits 18, 19, and 20 of F1 register. N550 is the gating term. N552 gates bits 21, 22, and 23. N554 gates bits 15, 16, and 17. Thus, every RNI sequence will cause F1 to be duplicated in F2. the contents of the upper nine bits of F1 are put into F2 whether or not the arithmetic section is to be used.

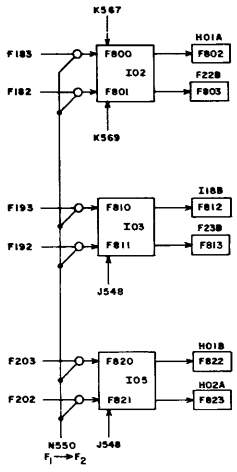


Figure 274.
F1 to F2 Duplication

As the signal passes down the chain (figure 275) the control delay slave inverters gate F1→F2. The timing chain consisting of H518, H519, and H520 serves to initialize the arithmetic section.

Initializing of the arithmetic section involves clearing shift count register and sensing the signs of A and Q registers. If (A) or (Q) were negative the fact would be recorded in flip-flops. This initializing of arithmetic occurs each RNI sequence even if arithmetic is not to be used.

The arithmetic section function translators (F8XX and F9XX) translate the 6-bit function code (from F2) and the three designator bits. The results of the translations are used to gate the commands which carry out the required arithmetic operations. (See Logic Diagrams, page 1-35.)

The major arithmetic timing consists of six control delays. In most cases one pass for arithmetic timing will be six Ø times long.

Shown are some of the common transfers that are part of arithmetic timing. Note that clearing of registers occurs at an even time and transfer occurs at an odd time. Only in a few special cases will you find the opposite.

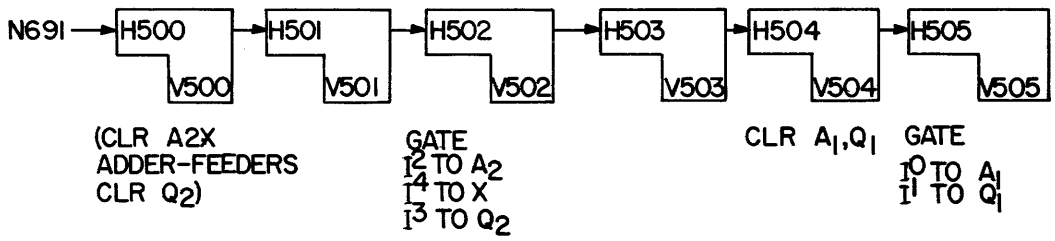


Figure 275. Arithmetic Timing Chain (Simplified)

The arithmetic timing chain can be started by main control, block control, or manual timing. In any case, to start the arithmetic timing chain one of two flip-flops must be set. Either K100/101 (start arith 1) or K104/105 (start arith 2) will be set to cause the start pulse from N687, N691, N693, and N659 (figure 276). K100/101- (start arith 1) and K104/105 - (start arith 2) are shown in Logic Diagrams, page 2-7.

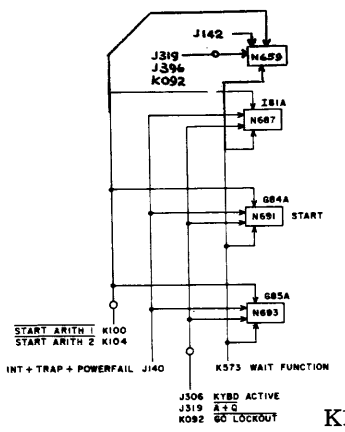


Figure 276.
K100/101 OR K104/105 Set

ADDRESS MODIFICATION (FADR)

There are two uses for the FADR cycle. The first is the adding together of address portion of an instruction word and contents of an index register (B^b) to form a new address. The sum of m and B^b is gated back to the lower portion of the F register to be used as the address for a storage reference.

$$m + B^b = M$$

The index register to be used can be determined by the b designator of the instruction word (e.g., 20.1 or 20.2) or by the function code (e.g., fixed-index instructions 22.4 or 42.4).

Indexing may be required for a RADR, ROP, or STO sequence. Thus, indexing will be initiated only at the end of RNI or RADR sequence.

Address modification will be referred to as FADR (form address).

For the FADR operation there will be a function translation of indexing. This will cause a static enable of the designated B index (clear side) to the I^{6XX} inverter rank in main control. There will also be a static enable of the lower 15 or 17 bits of F1 register to I^{5XX} inverter

rank. These enables will be static during the FADR operation (figure 277).

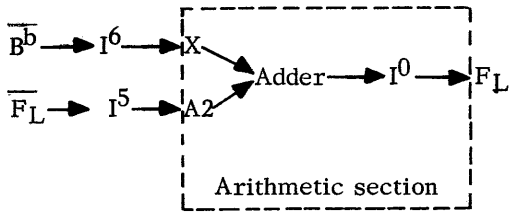


Figure 277. The Flow Path for FADR

The lower 15 bits of F1 register are used for a word-addressed instruction and the lower 17 bits are used with a character-addressed instruction. For all FADRs main control will extend the sign on the value from B^b and F_L:

Example:	Values added
B1 = 34567	00034567 = B1 with sign
F = 20.177760	77777760 = F_L with sign

TIMING FOR FADR

TIME	TERM	COMMAND	CONDITION	REMARKS
V080 or V082	K100/101	Start arith 1	Indexing	Provides entry to the indexing sequence at the end of RNI or RADR. When K101=1, it will set K536/537 (FADR 1) and K596/597 (FADR 2) FFs.

The arithmetic section will perform the same operation determined by translation of the instructions function code. If F1 and F2 registers hold a 50.2 the translators tell arithmetic to do a multiply. This is not what arithmetic should do to form address M for an FADR. When set, the purpose of FADR 1 and FADR 2 FFs is to cancel function translations for the arithmetic sections. The flip-flops remain set during FADR.

N691 N691	H100 H510	Clear X and A2	$\overline{\text{(Arith busy)}}$ (Wait function)	
V510/V530	H501			X and A2 are cleared.

A2 is cleared by N520, N522, and N524 (Logic Diagrams, page 1-29). The output of the HXXX terms on the AND gate to these N52X terms is a logical 1 in a stable state. H510 going to a 0 because of its input will permit N520, N522, and N524 to output 1's, clearing A2 at the next even time (effectively, V500).

N640, N642 and N644 will clear X register.

At the same time the following is taking place:

V530	H571	I^{5XX} to A2	
V530	H541	I^{6XX} to X1	
V100	K110/111	I^{0XX} to F	
V100	Clear K100/101		Clear start arith 1.
V100	K110/111	Enable I^0 to F	Permits gating sum (M) to F_L later.
V501	H502		I^{6XX} is gated to X and I^{5XX} is gated to A2. $X=B^b$ with sign extension and A2 = F_L with sign extension

The sum of A2 and X will be formed by the adder. A static enable from the adder's output to the I^0 inverter rank is caused by J825, J826, and J827 (Logic Diagrams, page 1-34).

TIMING FOR FADR

TIME	TERM	COMMAND	CONDITION	REMARKS
V502	H503			
V502	H201	Clear F (lower 15 or 17 bits).		H201, clear F, 1-10*
V503/V523	H504		Word address--	Lower 15 bits of F are cleared.
V523	H110		Character address	Lower 17 bits of F are cleared.
V554	K596/597			Clear FADR 2 FF
V504	H505			
V110	K000/001	Set request bus		
V110	H271	I^{0XX} to F_L		Initiate a storage reference.
V110	H105			H271 (I^0 to F, 1-10)
V505	H102			Gate I^{0XX} to F; $F_L = M = m+B^b$
V105	H106			End arithmetic at V505 time.
N051	K010/011	Set main control priority		Indexing is completed and the selected sequence is continued.
V102	Clear K536/537	Clear FADR 1		
V106	Clear K110/111	Clear enable $I^0 \rightarrow F$		

While the arithmetic section was doing FADR, main control was awaiting arithmetic time 3 (V523) to input H110. V110 is required to set K000/001 (request bus) whether FADR has or has not been completed.

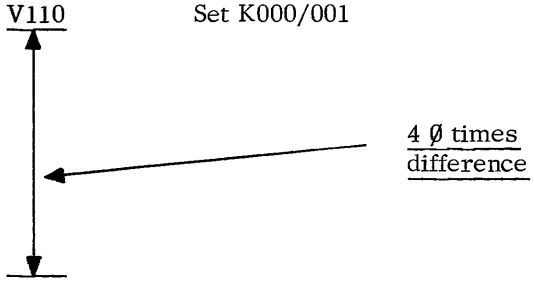
Let's compare the major points of timing for the 20.1XXXXX and 20.0XXXXX instructions.

20.1XXXXX (FADR)

TIME	COMMAND
V080	Set start arith 1
N691	Start arithmetic pulse
V500	Clear A2 and X (adder feeders)
V501	I^5 to A2 ($\overline{F}$ to I^5 static) I^6 to X (B^b to I^6 static)
V502	
V503	Clear F_L and input H110
V504/V110	Set K000/001 (request bus)
V505	I^0 to F_L (adder to I^0 static)

20.0XXXXX ($\overline{FADR}$)

TIME	COMMAND
V080	Set K112/113
V109	Input H110
V110	Set K000/001



Four phase times are added to the instruction execution time for each FADR. How many microseconds is this? _____

This process involves adding +1 to (F_L) to form an address M+1.

The second operation is the forming of the second operand address for the 32 and 33 instructions (also for double-precision and floating-point arithmetic if the option is present).

OPERATIONS IN THE ARITHMETIC SECTION

Instructions for the 3300 may be divided into several groups in which the instructions require similar arithmetic operations. These are, in order of complexity: jumps and skips, copy, interregister transfer, storage shift, enter, load, increase, logical, replace add, store, masked search, compare, shift and scale, add,

*Logic Diagrams

subtract, multiply, and divide.

The arithmetic operations required by each of these groups are discussed in this chapter. In most cases the main control operations involved for each instruction are not detailed, as it is assumed that you have read about main control.

The 3300 command timing charts provide a complete phase-time by phase-time description of the execution of each instruction.

The instructions in the optional ECD and floating-point/double-precision packages also require use of the standard arithmetic section.

JUMP AND SKIP INSTRUCTIONS (02-05, 10.1-10.7)

During execution of jump and skip instructions the arithmetic section is used to add two operands to form a quantity which conditions certain operations. The addition performed here is not for the purpose of forming a usable sum; rather, a comparison is made by using the addition process. These instructions are executed during RNI, so indexing and indirect addressing cannot be performed at that time.

A, Q, B^b Jumps and Skips (03-05)

The jump and skip instructions listed in table 17 require one pass through the adder during instruction execution. The general steps for executing these instructions and the arithmetic operation for each is shown in table 17.

1. Jump or skip instruction is obtained from storage and decoded during RNI.
2. Start arithmetic.
3. Clear X and A2 (adder feeders).
4. Gate operands to be added to form control quantity (listed for each instruction in table 17 into A2 and X.
5. Form sum in adder. (Sum is actually difference because the complement of one operand is gated into X.)
6. Test sum for required condition. Set jump FF or skip FF if test is satisfied. Sum is not gated from adder but is tested by jump or skip sensing network.
7. Perform a jump or skip exit to next RNI if condition is satisfied. Normal exit is made to next RNI if test condition is not satisfied.

Incremental/Decremental Index Jumps and Skips (02.1-3, 02.5-7, 10.1-7)*

The index jump and index skip instructions are conditional on the contents of a selected index register.

*02.0 and 02.4 are no-ops.

Table 17. A, Q, AND B^b JUMP AND SKIP INSTRUCTIONS

CODE	INSTRUCTION	TEST MADE	CONTROL QUANTITY FORMED BY ADDER
03.(0-3)	Compare A with zero, jump	Test* for (A1) equal, unequal, greater, or less than zero.	Arithmetic timing chain is started, but adder is not used. (A1) is sensed directly.
.(4-7)	Compare A with Q, jump	Test* for (A1) equal, unequal, greater, or less than (Q1).	Sum = (A1) + $(\overline{Q1})$ Ø = equal + = greater than - = less than
04.	Skip next instruction if		
.(0-3)	$(B^b) = y$	$(B^b) - y = 0?$	Sum = $(B^b) + \overline{y}$
.(4,6)	(A) = y	(A1) - y = 0?	Sum = (A1) + $\overline{y}$**
.(5,7)	(Q) = y	(Q1) - y = 0?	Sum = (Q1) + $\overline{y}$**
05.	Skip next instruction if		
.(0-3)	$(B^b) = y$	$(B^b) - y = 0?$	Sum = $(B^b) + \overline{y}$
.(4-6)	(A) = y	(A1) - y = 0?	Sum = (A1) + $\overline{y}$**
.(5-7)	(Q) = y	(Q1) - y = 0?	Sum = (Q1) + $\overline{y}$**

*Test condition determined by designator j, bits 15-17.
**y is sign extended for XX.4 and XX.5.

The state of this index register is sensed to determine if a jump or skip is to be taken, and also if an increment/decrement index operation should be performed.

Index Jump: (B^b) is sensed for equality with zero. If (B^b) = 0, the test condition is not satisfied and a normal RNI from P+1 is performed. If (B^b) ≠ 0, B^b is incremented (02.1-02.3) or decremented (02.5-02.7), and a jump to address m is performed.

The steps in execution of index jump are:

1. Instruction is obtained from storage and decoded during RNI.
2. Set jump FF if $B^b \neq 0$. (State B^b is sensed by index-sensing networks.)
3. Start arithmetic timing (occurs regardless of state of B^b).
4. RNI from address m if (B^b) ≠ 0 and jump FF is set. RNI from P+1 if (B^b) = 0. Arithmetic section now completes execution of index jump while main control simultaneously performs RNI.
5. Clear X and A2.
6. Gate (B^b) to X via I^6. Gate output of I^5 to A2 (+1 for incremental index jump, -1 for decremental jump).
7. Form sum.
8. Clear B^b. (This command is blocked when jump FF is not set.)
9. Gate sum through I^0 into B^b. (This command also is blocked when jump is clear.)

Index Skip: (B^b) is compared with y for an index skip. If (B^b) = y, B^b is cleared and a skip to P+2 is performed. If (B^b) ≠ y, B^b is incremented (10.1-10.3) or decremented (10.5-10.7) and an RNI is made from P+1. (This instruction will skip only if y = 00000 or if y = 77777 for 10.4.)

The steps in execution of index skip are:

1. Instruction is taken from storage and decoded during and RNI enable.
2. (F) → I^5 and ($\overline{B^b}$) → I^6.
3. Start arithmetic timing.
4. Clear X and A2.
5. Gate $\overline{F_L}$ into A2 via I^5; gate (B^b) into X via I^6.
6. Add (B^b) and $\overline{y}$. $\overline{Y}$ is F_L. (Both F and B^b are sign extended). This operation subtracts (Y) from (B^b); If sum = 0, they were equal.
7. Enable +1 (incremental skip) or -1 (decremental skip) from I^5.
8. Set skip FF if sum = 0 (B^b = y). Output of adder is sensed by skip-sensing network.
9. Set skip 2 FF is skip FF is set.
10. Start arithmetic.
11. RNI from P+1 if skip condition is not met ((B^b) ≠ y), from P+2 if skip condition is met (B^b = y). Arithmetic simultaneously completes execution of index skip.
12. Clear X and A2.
13. Gate (B^b) into X via I^6. Gate +1 (incremental skip) or -1 (decremental skip) into A2 via I^5.
14. Form the sum of (B^b) + 1.
15. Clear B^b.
16. Gate the sum from adder to B^b via I^0. (This command is blocked when skip 2 is set.)

COPY INSTRUCTIONS (77.2 ch 0000, 77.3 ch 0000)

The copy instructions load the contents of interrupt mask register into the upper 12 bits of A1 and the external status code for 77.2 (ch) 0000 instruction or internal status code for a 77.3 (ch) 0000 instruction into the lower 12 bits of A1.

The steps in executing copy instructions are:

1. Obtain instruction from storage.
2. Set status → A FF.
3. Clear A1.
4. Gate contents of interrupt mask register and status to A1. (External status code from I/O channel ch for 77.2 (ch) 0000, internal status code for 77.3 (0000).
5. Clear status → A FF.
6. RNI from P + 1.

SINGLE-PRECISION INTERREGISTER TRANSFER (53)

The interregister transfer instruction is used to move data between A1 and Q1 registers, index registers, and register file.

The single-precision interregister transfer instruction is composed of eleven subinstructions, each of which performs a separate transfer.

These instructions and the steps in their execution are:

1. Transfer (B^b) to A1 (53.0 (1-3) 0).
2. Obtain instruction from storage (RNI).
3. Start arithmetic, then RNI from P+1 while arithmetic completes execution of instruction.
4. Clear X and A2.
5. Gate (B^b) to X via I^6. A2 remains clear.
6. Form sum, (B^b) + 0 = (B^b).
7. Clear A1.
8. Gate sum, (B^b), into A1 via I^0.
9. Transfer (A1) to B^b (53.1 (1-30) 0).
10. Obtain instruction from storage.
11. Start arithmetic, then RNI from P+1 while arithmetic completes execution of instruction.
12. Clear X and A2.
13. Gate (A1) into X via I^2. A2 remains cleared.
14. Form sum.
15. Clear B^b.
16. Gate sum, (A1), into B^b via I^0.
17. Transfer (register m) to Q (53.01).
18. Obtain instruction from storage.
19. Start arithmetic, then RNI from P+1 while arithmetic completes execution of instruction.
20. Register file reads (register m) into Z0.
21. Clear DB register.
22. Gate (Z0) through I^7 and EXX2 into DB register.
23. Clear X and A2.
24. Gate (DBR) and (register m) into X via I^4.
25. Clear Q1.

26. Gate (X) into Q1 via I^1. Q1 now contains (register m).
27. Transfer (Q) to register m (53.41).
28. Obtain instruction from storage.
29. Start arithmetic.
30. Clear X and A2.
31. Gate (Q1) into X via I^3. Meanwhile, block control prepares register file to receive word by loading address into S, issuing a register file write, and starting delay line timing.
32. RNI from P+1 while block control simultaneously completes execution of instruction.
33. Clear DB register.
34. Gate (X) into DB register via I^7 and EXX2. ((Q1) now in DB register.)
35. Clear Z1 (block control register).
36. Gate (DBR) into Z1.
37. Write word, (Q1), from Z1 into register m.
38. Transfer (register m) to A1 (53.02).
39. Obtain instruction from storage.
40. Start arithmetic, then RNI from P+1 while arithmetic completes execution of instruction.
41. Register file reads (register m) into Z0.
42. Clear DB register.
43. Gate (Z0) into DB register via I^7 and EXX2.
44. Clear X and A2.
45. Gate (DBR) into X via I^4.
46. Clear A1.
47. Gate (X) through the adder to I^0 and A1. A1 now contains (register m).
48. Transfer (A) to register m (53.42).
49. Obtain instruction from storage.
50. Start arithmetic.
51. Clear X and A2.
52. Gate (A1) into X via I^2. Meanwhile, block control prepares register file to receive word by loading address into S, issuing a register file write, and starting delay line timing.
53. RNI from P+1 while block control simultaneously completes execution of instruction.
54. Clear DB register.
55. Gate (X) into DB register via I^7 and EXX2. ((A1) now in DB register.)
56. Clear Z1 (block control register).
57. Gate (DBR) into Z1.
58. Write word, (A1), from Z1 into register m.
59. Transfer (register m) to B^b (53. (1-3) 3).
60. Obtain the instruction from storage.
61. Start arithmetic, then RNI from P+1 while arithmetic and block control complete execution of instruction.
62. The register file reads (register m) into Z0.
63. Gate (Z0) into DB register via I^7 and EXX2.
64. Clear X and A2.
65. Gate (DBR) into X via I^4. A2 remains cleared.
66. (X) propagates through adder and is statically enabled to I^0.
67. Clear B^b.
68. Gate sum into B^b via I^0. B^b now contains (register m).
69. Transfer (B^b) to register m (53. (5-7) 3).
70. Obtain instruction from storage.
71. (B^b) to I^6, I^6 to I^7, I^7 to EXX2 are statically enabled.
72. Start arithmetic (arithmetic timing not used for this instruction), then RNI from P+1 while block control completes execution of instruction.
73. Clear DB register.
74. Gate (B^b) from EXX2 into DB register.
75. Clear Z1.
76. Gate (DBR) into Z1.
77. Write word, (B^b), from Z1 into register m.
78. Transfer (A) plus (Q) to A (53.04).
79. Obtain instruction from storage.
80. Start arithmetic, then RNI from P+1 while arithmetic completes execution of instruction.
81. Clear X and A2.
82. Gate (Q1) to X via I^3. Gate (A1) to A2 via I^2.
83. Form sum.
84. Clear A1.
85. Gate sum into A1 via I^0. A1 now contains (A1) + (Q1).
86. Transfer (A) plus (B^b) to A (53. (1-3) 4).
87. Obtain instruction from storage.
88. Start arithmetic, then RNI from P+1 while arithmetic completes execution of instruction.
89. Clear X and A2.
90. Gate (B^b, -sign extended) into X via I^6. Gate (A1) into A2 via I^2.
91. Form sum.
92. Clear A1.
93. Gate sum into A1 via I^0. A1 now contains (B^b) + (A1).
94. Transfer (A) plus (B^b) to B^b (53. (5-7) 4).
95. Obtain instruction from storage.
96. Start arithmetic, then RNI from P+1 while arithmetic completes execution of instruction.
97. Clear X and A2.
98. Gate (B^b, -sign extended) into X via I^6. Gate (A1) into A2 via I^2.
99. Form sum.
100. Clear B^b
101. Gate sum into B^b via I^0. B^b now contains (A1) + (B^b).

STORAGE SHIFT (10.0)

The storage shift instruction reads a word (m) from storage, senses the sign bit and sets skip if the sign is negative, shifts the word one place left (end-around), then replaces it in storage.

The next instruction is read from P+1 if the sign is positive. If the sign is negative and skip is set, a skip to P+2 is performed.

The steps in the execution of the storage shift are:

1. Instruction is obtained during RNI. ROP is then

performed to read word at location m. This word is placed in DB register.
2. Begin STO.
3. Start arithmetic. While arithmetic completes shift of (m), main control prepares for return of word to storage by obtaining bus priority and transmitting a write and the storage address to memory.
4. Clear X and A2.
5. Gate (DBR) to X via I^4.
6. Set skip if the sign of (m) is negative. (Sign was checked in DBR).
7. Gate (X) into Q2 left. This transfer always places (X) shifted one place end-around into Q2.
8. Clear X and A2.
9. Gate (Q2) into X via I^4. This places (m - left 1) back in X ready for return to storage.
10. Enable ($\overline{DBR}$) to data bus transmitters.
11. Clear DB register.
12. Gate (X) into DB register via I^7 and EXX2. Because ($\overline{DBR}$) has already been enabled to transmitters, (m - left 1) is immediately transmitted to storage.
13. Main control then waits until reply from storage starts storage time-out chain.
14. Drop bus priority and storage request.
15. RNI from P + 1 if skip is clear (original sign of (m) positive), or RNI from P+2 if skip is set (sign of (m) negative).

ENTER INSTRUCTIONS (11, 14)

The 14 instructions place a quantity y (lower bits of the instruction word) into A1, Q1, or B^b. The 11 instruction places a quantity r (lower 17 bits of the instruction word) into A1.

The steps in execution of enter character address into A (11) are:

1. Instruction is obtained and decoded during RNI.
2. Start arithmetic, then RNI from P+1 while arithmetic completes execution of instruction.
3. Clear X and A2.
4. Gate lower 17 bits of F (sign extended for bit 17 = 1) into A2 via I^5. X remains cleared.
5. Form sum (F_{L17}).
6. Gate sum, (F_L), into A1 via I^0.

The steps in execution of enter A, Q, or B^b with y (14) are:

1. Instruction is obtained and decoded during RNI.
2. Start arithmetic, then RNI from P+1 while arithmetic completes execution of instruction. (Arithmetic timing occurs but is not used for enter B^b with y.)
3. Clear X and A2.

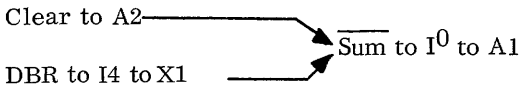
To execute enter A with y (14.4 or 14.6), gate y, lower 15 bits of F (sign extended for 14.4), into A2 via I^5. X remains clear. Form sum. (This is done in all cases, but output of adder is used only for enter A.)

To execute enter Q with y (14.5 or 14.7), gate y, lower 15 bits of F (sign extended for 14.5), into lower X via I^6. A2 remains clear.

Enter B^b with y (14.0-14.3). Clear B^b.
Enter A1 with y (14.4 or 14.6). Clear A1.
Enter Q1 with y (14.5 or 14.7). Clear Q1.
Enter B^b with y. Gate lower 15 bits of F into B^b.
Enter A1 with y. Gate sum, (F_L), into A1 via I^0.
Enter Q1 with y. Gate (X) and (F_L) into Q1 via I^1.

LDA (20) Instruction

Setting of K104/105 (start arith 2 FF) at V008 time will cause N691 to start the arithmetic timing chain. The arithmetic timing chain will enable:



LACH (22) Instruction

Identical to LDA (20) instruction with one exception: I^4_{L6} to X instead of I^4 to X. (Refer to Logic Diagrams, page 2-93). F967 will knock down N643, N645, and N647.

LCA (24) Instruction

Identical to LDA (20) instruction with one exception: DBR→I^4 instead of $\overline{DBR}$ to I^4.

(Refer to Logic Diagrams, page 2-151). F802, F823, and F882 input a 1 to J904, knocking it down and causing an output from J905 to J908.

LDAQ (25) Instruction

In this instruction we are concerned with an ROP to ROP. In the first ROP two things occur:

1. Clear A2 → $\overline{Sum}$ to I^0 to A
   $\overline{DBR}$ to I^4 to X1 → $\overline{Sum}$ to I^0 to A
2. An FADR operation is started by setting K100/101 (start arith 1 FF) at V002 time. K060/061 enabled setting of K100/101 and also forced I^6 to output a +1. FADR consists of adding +1 to M to form the address for loading (Q1).

In the second ROP, $\overline{DBR}$ to I^4 to $\overline{X1}$ to I^1 to Q1 is caused by setting of K104/105 at V008 time. NOTE: K502/503 (originally called load op, now LDAC + LDAQ) helps to enable data paths on the second ROP.

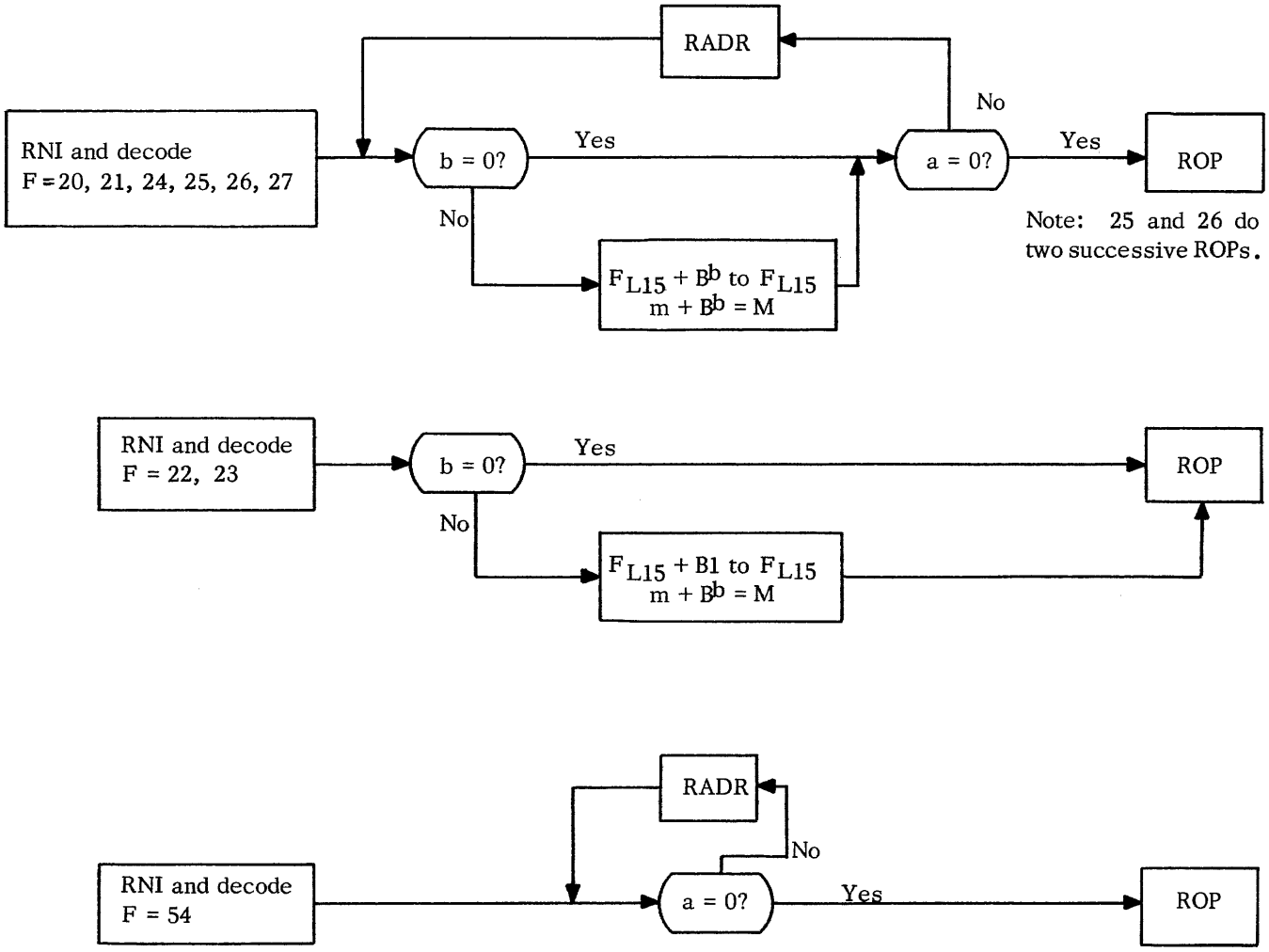
LCAQ (26) Instruction

Identical to LDAQ (25) instruction with one exception: DBR to I^4 instead of $\overline{DBR}$ to I^4. (Refer to Logic Diagrams, page 2-151,) F802, F823, and F882 input a 1 to J904, knocking it down and causing an output from J905 to J908.

LDL (27) Instruction

Identical to the LDA (20) instruction with two exceptions:

1. $\overline{Q1}$ to I^2 to A2 instead of 0's to A2. (Refer to Logic Diagrams, page 2-139.) K511, F940, F915, and



LDI 54****
LDA 20
LACH 22*
LCA 24**
LDAQ 25
LCAQ 26**
LDL 27***

Clear A2 → $\overline{\text{Sum}}$ to I^0 to A1

$\overline{\text{DBR}}$ to I^4 to X1

LDQ 21
LQCH 23*
LDAQ 25
LCAQ 26**

$\overline{\text{DBR}}$ to I^4 to $\overline{\text{X1}}$ to I^1 to Q1

Substitute:

*$\overline{I^4}_{L6}$ to X4
**$\overline{\text{DBR}}$ to I^4
***$\overline{\text{Q1}}$ to I^2 to A2
****I^0 to B^b

NOTE: Since instructions 25 and 26 are double precision the data paths to both A and Q are shown.

Figure 278. Load Instructions Sequences, and Data Paths

K575 are 0's causing the $\overline{Q1}$ to I^2 inverter to output 1's. J504, V500, and K596 input to H551 (I^2 to A2) at V500 time. (Refer to Logic Diagrams, page 2-95.)

2. Input to H515 to J508, V504, O817, and L175, not N544 and J555.

LDQ (21) Instruction

Identical to Q portion (DBR to I^4 to X1 to I^1 to Q1) of a 25 (load AQ) instruction.

LDCH (23) Instruction

Identical to LDQ (21) instruction with one exception: I^4_{L6} to X instead of I^4 to X. (Logic Diagrams, page 2-93.) F967 will knock down N643, N645, and N647.

LDI (54) Instruction

This instruction did not originally use the arithmetic timing chain. It had a data path of EXX2 to B^b. The 54 instruction is now identical to the LDA (20) instruction except that it is I^0 to B^b instead of I^0 to A1 at arithmetic time 5. (Refer to Logic Diagrams, page 2-5.) K006, V523, and K560 combine to input a 1 into H018. The output of this control delay feeds into H019. (Logic Diagrams, page 2-53.) V019 and F420 combine to input to H244 (I^0 to B^b).

With the 20 instruction H531 received two inputs at the same time. With the 54 instruction, the input is from the bottom. (This is of no consequence; it is merely pointed out.)

There is an input to H514 (clear A1) at V503 time and an input to H515 at N544 time for the 20 instruction but not for the 54 instruction.

STA (40) Instruction

Setting of K104/105 (start arith 2 FF) at V126 time will cause N691 to start the arithmetic timing chain. The arithmetic timing chain will enable $\overline{A1}$ to I^{2XX} to X.

SACH (42) Instruction

Identical to STA (40) instruction.

SWA (44) Instruction

Identical to STA (40) instruction.

SCHA (46) Instruction

Identical to STA (40) instruction.

STAQ (45) Instruction

In this instruction we are concerned with an STO to STO. In the first STO two things occur:

1. $\overline{A1}$ to I^{2XX} to X caused by setting of K104/105 at V126 time. This is identical to the 40 instruction.
2. An FADR operation is started by setting K100/101 (start arith 1 FF) at V002 time. K060/061 enabled setting of K100/101 and also forced I^6 to output a +1. FADR consists of adding the +1 to M to form the address for storing (Q1). In the second STO, $\overline{Q1}$ to I^{3XX} to X is caused by setting of K104/105 at V126 time. NOTE: The only purpose for K540/541 (double-precision STO) is to tie into J500 and J501 to determine whether it is I^2 to X or I^3 to X pages 2-93.

STQ (41) Instruction

Setting of K104/105 (start arith 2 FF) at V126 time will cause N659 to start the arithmetic timing chain. The arithmetic timing chain will enable $\overline{Q1}$ to I^{3XX} to X.

SQCH (43) Instruction

Identical to STQ (41) instruction except for number of bits stored.

STI (47) Instruction

Even though arithmetic timing is started it is not used. $\overline{B^b}$ to I^6 to I^7 to EXX2 is statically enabled.

COMMAND TIMING CHART

CODE	INSTRUCTION	FUNCTION
20	load A	Place (M) into the A register

SEQUENCE: 0's to A → $\overline{SUM}$ to I^0 to A1

$\overline{DBR}$ to I^4 to X → $\overline{SUM}$

TIME	PAGE*	TERM	COMMAND	CONDITION	REMARKS
N659	2-93	H500		J536, N659, and J514 combine to input a 1.	
	2-93	H510	Clear X, A2.	J514 and N659 combine to input a 1.	
V500	2-93	H501		J519, V500, and K596 combine to input a 1.	X and A2 now cleared. K581, which ties into the clear A2 inverters, can be set only during a 52 or BCD instruction.
	2-95	H531	I^4 to X	J502, V500, and K596 combine to input a 1.	$\overline{DBR}$ to I^4 enabled by F translations. Refer to pages 2-151. J904 makes the gate into J913. F901, F972, and K500 make the gate into J914. Both J913 and J914 output ones.
V501		H502		1 out of V501 and V531	$\overline{DBR}$ to I^4 to X now occurs. This places M in X. (A2) = 0. Since A2 and X tie directly into the adder, adder propagation will now begin.
V502		H503		V502 and J512 combine to input a 1.	
V503		H504		1 out of V503	
		H514	Clear A1.	J506, V533, and F924 combine to input a 1.	
V504	2-95	H505		1 out of V544	A1 is now cleared. K543 can only be a 1 during a divide instruction. (V514)
(N544)	2-97	H515	I^0 to A1	N544 and J637 combine to input a 1.	$\overline{Sum}$ to I^0 is statically enabled. Refer to page 2-127. All inputs to J825 to J827 are zeros. K531 breaks one AND gate and F984 breaks the other AND gate.
N505	2-97				$\overline{Sum}$ to I^0 to A1. This places (M) in A1. J560, J558, and K557 are 0's.

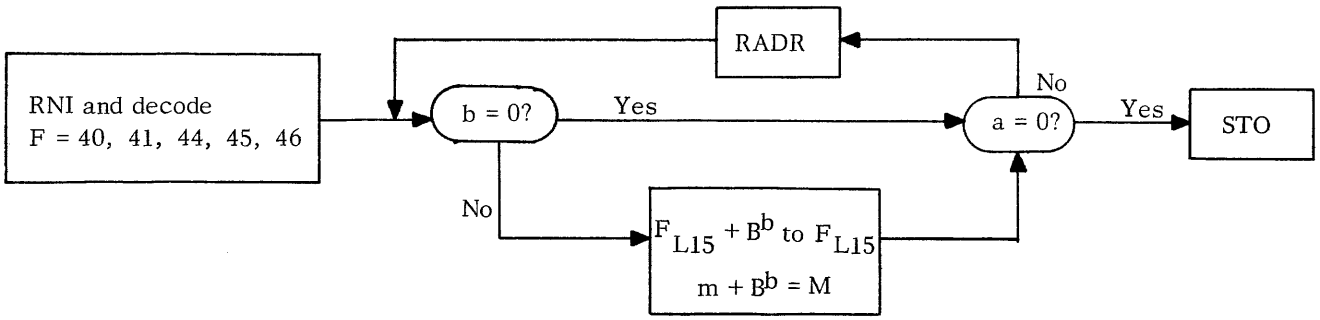
* In 3300 Computer Logic Diagrams

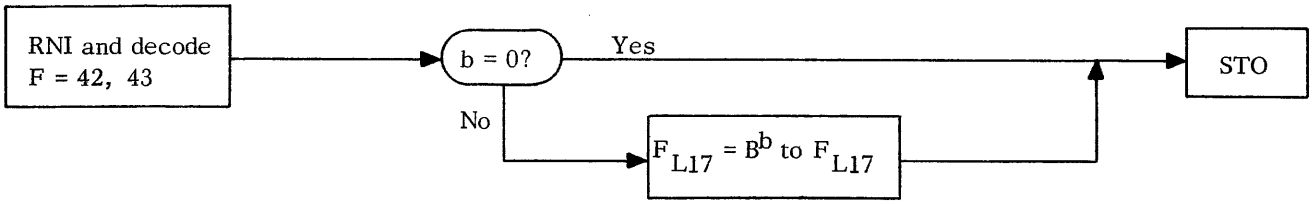
COMMAND TIMING CHART

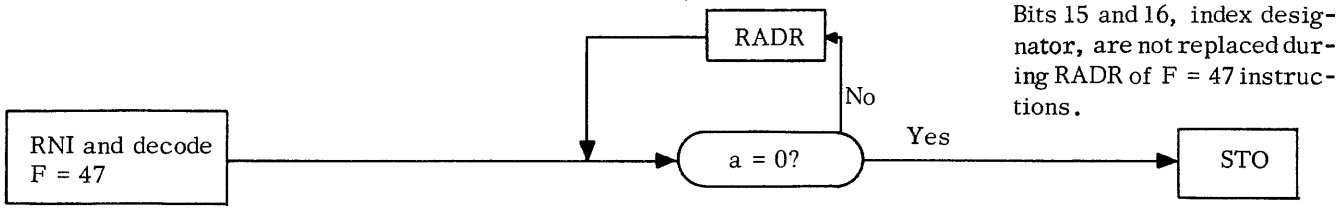
CODE	INSTRUCTION	FUNCTION
21	load Q	Place (M) into the Q register

SEQUENCE: $\overline{DBR}$ to I^4 to $\overline{X1}$, X1 to I^1 to Q1

TIME	PAGE	TERM	COMMAND	CONDITION	REMARKS
N659	2-93	H500		J536, N659, and J514 combine to input a 1.	
		H510	Clear X, A2	J514 and N659 combine to input a 1.	
V500		H501		J519, V500, and K596 combine to input a 1.	X and A2 now cleared. K581, which ties into the clear A2 inverters, can be set only during a 52 or BCD instruction.
	2-95	H531	I^4 to X	J502, V500, and K596 combine to input a 1.	$\overline{DBR}$ to I^4 enabled by F translations. J904 makes the gate into J913. F901, F972, and K500 make the gate into J914. Both J913 and J914 output 1's.
V501		H502		1 out of V501 and V531	$\overline{DBR}$ to I^4 X now occurs.
V502		H503		V502 and J512 combine to input a 1.	
V503		H504		1 out of V503	
	2-95	H524	Clear Q1	J507, V503, and J538 combine to input a 1.	
V504	2-95	H505		1 out of V504	Q1 now cleared. Refer to page 2-95. K593 is a 0. It can only by set during 51 instruction.
	2-97	H555	I^1 to Q1	J509, V504, and L175 combine to input a 1.	$\overline{X}$ to I^1 is statically enabled. Refer to page 2-133. The AND gate into the X to I^1 inverters is broken by F935.
		N53X		K559, K593, and H555 are 0's.	X to I^1 to Q1 occurs.







Bits 15 and 16, index designator, are not replaced during RADR of F = 47 instructions.

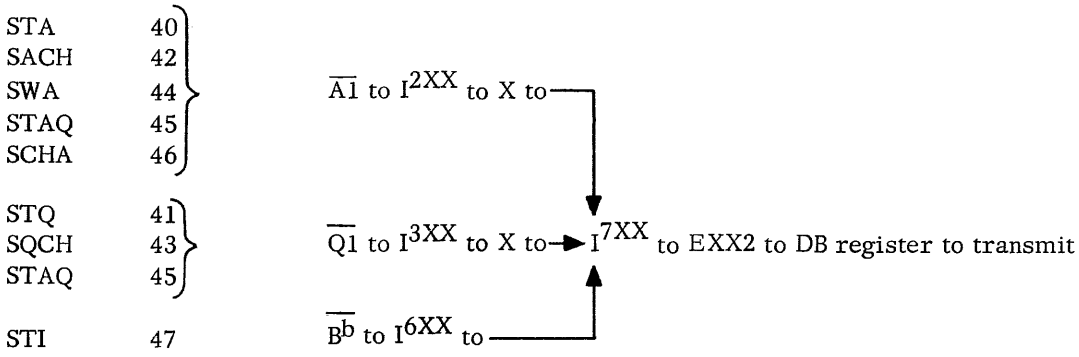


Figure 279. Store Instructions, Sequences, and Data Paths

COMMAND TIMING CHART

CODE	INSTRUCTION	FUNCTION
40	store A	Arithmetic timing portion of a store A instruction.

SEQUENCE: A1 to I^{2XX} to X

TIME	PAGE	TERM	COMMAND	CONDITION	REMARKS
N659	2-93	H500		J536, N659, and J514 combine to input a 1 to H500.	
	2-93	H510	Clear X1, A2	J514 and N659 combine to input a 1 to H510.	X1 must be cleared because it is in the data path that (A1) must take to be stored. Clearing A2 occurs but is of no consequence.
V500	2-93	H501		V500, K596 and J519 combine to input a 1 to H501.	F884 will not knock down J519 because F2 is not used.
	2-93	H511	I^2 to X	V570, K536, and J500 combine to input a 1.	X and A2 are now cleared. K581, which ties into the clear A2 N-terms, is a 0 for all store instructions.
					$\overline{A1}$ to I^2 is statically enabled. Refer to page 2-139. J880, J881, and J882 all have zeros coming into them. F900 will break the gate consisting of F918, F900, F966, and F994.
V501	2-95	H502		V501 inputs a 1.	$\overline{A1}$ to I^2 to X. (A1) now in X.
V502	2-95	H503		V502 and J512 combine to input a 1 to H503	Nothing happens from H502 through H505. From this point in time, program control takes over again.
V503	2-95	H504		V503 outputs a 1.	The arithmetic has already accomplished its objective.
V504	2-95	H505		V504 outputs a 1.	
V505	2-95				Arithmetic timing chain stops.

COMMAND TIMING CHART

CODE	INSTRUCTION	FUNCTION
41	store Q	Arithmetic timing portion of a store Q instruction.

SEQUENCE: $\overline{Q1}$ to I^{3XX} to X

TIME	PAGE	TERM	COMMAND	CONDITION	REMARKS
N659	2-93	H500		J536, N659, and J514 combine to input a 1.	The purpose of starting arithmetic timing is to get (Q) to X.
		H510	Clear X, A2	J514 and N659 combine to input a 1.	
V500	2-93	H501		J519, V500, and K596 combine to input a 1.	X and A2 are now cleared $\overline{Q1}$ to I^3 are statically enabled. Refer to page 2-145. K517 and K589 are zeros. F902 breaks the AND gate into J895 to J897.
		H521	I^3 to X	J501, Ø818, V500, and K536 combine to input a 1.	
V501		H502		1 out of V501	$\overline{Q1}$ to I^3 to X. (Q1) now in X. Page 2-151.
V502	2-95	H503		J512 and V502 combine to input a 1.	From this point in time program control takes over again.
V503		H504		1 out of V503	
V504		H505		1 out of V504	
V505					End arithmetic

ARITHMETIC INSTRUCTIONS

ADA (30) Instruction

This instruction is identical to that of the LDA (20) instruction except for $\overline{A1}$ to I^2 to A2 instead of 0's to A2. (Refer to Logic Diagrams, page 2-139. F966 breaks the AND gate and K505, K517, K545, and K589 are 0's going into the $\overline{A1}$ to I^2 J-terms, J880 to J881. (Logic Diagrams, page 2-95. J504, V500, and K596 combine to input a 1 to H551 (I^2 to A2) at V500 time.

SBA (31) Instruction

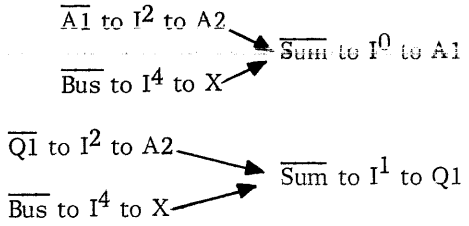
This instruction is identical to that of the LDA (20) instruction except for $\overline{A1}$ to I^2 to A2 instead of 0's to A, and DBR to I^4 instead of $\overline{DBR}$ to I^4. The explanation for the $\overline{A1}$ to I^2 to A2 is identical to that of the ADA (30) Instruction. (Refer to Logic Diagrams, page 2-151) F803, F822, and F883 combine to input a 1 to J904, knocking it down and enabling a 1 out of the DB register to I^4 J-terms (J905 to J908).

ADAQ (32) Instruction

This instruction is similar to the LDAQ (25) instruction in that it will set the first and second cycle FFs:

1. To perform an FADR operation for M+1 and
2. To allow an ROP to ROP sequence.

The data path is as follows:



ROP to ROP occurs during operation of the 32 instruction. The following is an example of what happens:

1. First ROP: (A1) + (M) to A1
   Second ROP: (Q1) + (M + 1) + carry (if available) to Q1 } 2s complement, sense carry

   If an EAC occurs during the first ROP, K508/509 sets causing K534/535 +1 FF to set. If K534/535 was set it is used during the second ROP (2-101).
2. If another EAC occurs after a second ROP the following will happen:
   (A1) + Q1 carry to A1
3. If another EAC is generated after step 2, the following will happen:
   (Q1) + A1 carry to Q1

The arithmetic timing chain will be started three to five times including FADR.

First time: FADR
Second time: First ROP
Third time: Second ROP
Fourth time: This is self-starting and depends on the conditions specified in step 2 which concern the second ROP. (Was there a carry?)
Fifth time: This is self-starting and depends on the conditions specified in step 3. (Was there a carry?)

NOTE: K506/507 (double precision) is responsible for the self-starting of the arithmetic timing chain the fourth and fifth times. (Refer to Logic Diagrams, page 2-95.) K506 breaks the AND gate into J537, causing it to output a 1. K504/505 (2+4 cycle double precision) helps enable the data paths and also is responsible for setting K506/507 (double precision FF). Just as setting K504/505 enables pass 4 and 5 of arithmetic timing, clearing it prevents a 6th and subsequent passes.

Overflow is checked twice. It is first checked during the first ROP sequence. Overflow is checked again if (A1) + Q1 carry to A1 is being done. K552/553 will set if (A) is negative. K548/549 will set if (A2) and (X) have like signs. (Refer to Logic Diagrams, page 2-103 and 2-105.) J545, J547, K549, and V515 are responsible for setting overflow FF.

SBAQ (33) Instruction

This instruction is identical to the ADAQ (32) instruction except for DBR to I^4 instead of $\overline{DBR}$ to I^4.

STATIC ENABLES AND TIMED TRANSFERS Worksheet #1: FADR Sequences and Instructions 02 and 03

References: 3300 block diagram, command timing charts, Logic Diagrams

For each of the following sequences insert the appropriate letters (A-K) in the proper order to obtain the desired operation.

1. FADR _____, _____, _____ and _____, _____
2. 02.0 _____, _____, _____ and _____
3. 02.1 to 02.3 _____, _____, _____ and _____, _____
4. 02.4 _____, _____, _____ and _____
5. 02.5 to 02.7 _____, _____, _____ and _____, _____
6. 03.0 to 03.4 _____
7. 03.4 to 03.7 _____, _____, _____

A. Clear A2 and X
B. I^5 to A2, I^6 to X
C. I^2 to A2, I^3 to X
D. $\overline{A1}$ to I^2, Q1 to I^3
E. $\overline{+1}$ to I^5
F. $\overline{-1}$ to I^5
G. $\overline{B^b}$ to I^6
H. 1's to I^6
I. Clear B^b, I^0 to B^b
J. $\overline{F}$ to I^5
K. Clear F_L, I^0 to F_L

The proof is in the prints--check your answers!

STATIC ENABLES AND TIMED TRANSFERS Worksheet #2: Instructions 04 and 05

References: 3300 block diagram, command timing charts, Logic Diagrams

For each of the following instructions insert the appropriate letters (A-O) in proper order to obtain the desired operation.

1. 04.0 _____, _____ and _____, _____ and _____
   05.0
2. 04.1 to 04.3 _____, _____ and _____, _____ and _____
   05.1 to 05.3
3. 04.4 _____, _____ and _____, _____ and _____
   05.4
4. 04.5 _____, _____ and _____, _____
   05.5
5. 04.6 _____, _____ and _____, _____ and _____
   05.6
6. 04.7 _____, _____, _____ and _____
   05.7

A. Clear X and A2
B. I^5 to A2 (1's extended)
C. I^5 to A2 (sign of Y extended)
D. I^2 to A2 ($\overline{A1}$ to I^2)
E. I^2 to A2, 0's extended ($\overline{A1}$ to I^2)
F. I^6 to X (1's extended)
G. I^6 to X (0's extended)
H. I^6 to X (sign of Y extended)
I. I^3 to X ($\overline{Q1}$ to I^3)
J. I^3 to X, 0's extended ($\overline{Q1}$ to I^3)
K. F to I^5
L. $\overline{F}$ to I^5
M. $\overline{B^b}$ to I^6
N. 1's to I^6
O. I^5 to I^6

The proof is in the prints--check your answers!

STATIC ENABLES AND TIMED TRANSFERS Worksheet #3: Instruction 10 Variations

For each of the following function codes insert the appropriate letters (A-N) in the proper order to obtain the desired operation.

1. 10.0 _____, _____, _____, _____, _____, _____

2. 10.1 to 10.3
   a. 1st cycle _____, _____ and _____, _____ and _____
   b. 2nd cycle _____, _____ and _____, _____ and _____, _____, _____

3. 10.4
   a. 1st cycle _____, _____ and _____, _____ and _____
   b. 2nd cycle _____, _____ and _____, _____ and _____

4. 10.5 to 10.7
   a. 1st cycle _____, _____ and _____, _____ and _____
   b. 2nd cycle _____, _____ and _____, _____, _____, _____, _____

A. Clear A2 and X
B. I^4 to X ($\overline{DBR}$ to I^4)
C. I^6 to X
D. I^5 to A2
E. Clear Q2
F. X to Q2 (left 1)
G. I^4 to X ($\overline{Q2}$ to I^4)
H. $\overline{F}$ to I^5 (sign of Y extended)
I. $\overline{B^b}$ to I^6 (sign of B^b extended)
J. 1's to I^6
K. Clear B^b
L. If skip: I^0 to B^b
M. $\overline{+1}$ to I^5
N. $\overline{-1}$ to I^5

The proof is in the prints--check your answers!

STATIC ENABLES AND TIMED TRANSFERS Worksheet #4: Instructions 11 and 14

References: 3300 block diagram, command timing charts, Logic Diagrams

For each of the following sequences, insert the appropriate letters (A-L) in the proper order to obtain the desired operation.

1. 11.0 to 11.3 _____, _____ and _____, _____

2. 11.4 to 11.7 _____, _____ and _____, _____

3. 14.1 to 14.3 _____, _____

4. 14.4 _____, _____ and _____, _____

5. 14.5 _____, _____ and _____ and _____, _____

6. 14.6 _____, _____ and _____, _____

7. 14.7 _____, _____ and _____ and _____, _____

A. Clear A2 and X
B. I^5 to A2
C. I^6 to X1
D. $\overline{F}$ to I^5 (0's extended)
E. $\overline{F}$ to I^5 (sign of Y extended)
F. Clear B^b
G. F to I^5
H. I^5 to I^6 (0's extended)
I. I^5 to I^6 (sign of Y extended)
J. Clear A1, I^0 to A1 ($\overline{A2}$ to I^0)
K. Clear Q1, I^1 to Q1 ($\overline{X1}$ to I^1)
L. F to B^b

The proof is in the prints--check your answers!

STATIC ENABLES AND TIMED TRANSFERS Worksheet #5: Instructions 15, 16, and 17

References: 3300 block diagram, command timing charts, Logic Diagrams

For each of the following variations of the 15, 16, and 17 instructions insert the appropriate letters (A-O) in the proper order to obtain the desired operation.

15: Increase the (r) by y
16: Take the exclusive OR of (r) and y to r
17: Take the logical product of (r) and y to r

1. .1 to .3 ____, ____ and ____, ____ and ____, ____, ____
2. .4 ____, ____, ____ and ____, ____, ____
3. .5 ____, ____, ____ and ____ and ____, ____, ____
4. .6 ____, ____, ____ and ____, ____, ____
5. .7 ____, ____, ____ and ____ and ____, ____, ____

A. Clear A2 and X
B. I^5 to A2 (0's extended)
C. I^5 to A2 (sign of Y extended)
D. I^2 to A2 ($\overline{A1}$ to I^2)
E. I^6 to X (0's extended)
F. I^6 to X (sign of Y extended)
G. I^3 to X ($\overline{Q1}$ to I^3)
H. $\overline{F}$ to I^5
I. F to I^5
J. I^5 to I^6
K. B^b to I^6
L. I^6 to X (sign of B^b extended)
M. Clear A1, I^0 to A1 (adder to I^0)
N. Clear Q1, I^1 to Q1 (adder to I^1)
O. Clear B^b, I^0 to B^b (adder to I^0)

The proof is in the prints--check your answers!

STATIC ENABLES AND TIMED TRANSFERS Worksheet #6: Instruction Series 2X

For each of the following function codes insert the appropriate letters (A-G) in proper sequence in order to obtain the proper operations.

1. 20 LDA ____, ____, ____
2. 21 LDQ ____, ____, ____
3. 22 LACH ____, ____, ____
4. 23 LQCH ____, ____, ____
5. 24 LCA ____, ____, ____
6. 25 LDAQ
   a. 1st cycle ____, ____, ____
   b. 2nd cycle ____, ____, ____
7. 26 LCAQ
   a. 1st cycle ____, ____, ____
   b. 2nd cycle ____, ____, ____
8. 27 LDL ____, ____, ____, ____

A. Clear X and A2
B. I^4 to X ($\overline{DBR}$ to I^4)
C. I^4 to X (DBR to I^4)
D. I^2 to A2 ($\overline{A1}$ to I^2)
E. Clear A1, I^0 to A1 (adder to I^0)
F. Clear Q1, I^1 to Q1 ($\overline{X1}$ to I^1)
G. Within the adder force all stage generate (L0XX) terms to 0 outputs.

The proof is in the prints--check your answers!

STATIC ENABLES AND TIMED TRANSFERS Worksheet #7: Instruction Series 3X

For each of the following function codes insert the appropriate letters (A-J) in proper sequence in order to obtain the proper operation.

1. 30 ADA _____, _____ and _____, _____, _____

2. 31 SBA _____, _____ and _____, _____, _____

3. 32 ADAQ
   1st cycle _____, _____ and _____, _____, _____
   2nd cycle _____, _____ and _____, _____
   3rd cycle _____, _____, _____, _____
   4th cycle _____, _____, _____

4. 33 SBAQ
   1st cycle _____, _____ and _____, _____, _____
   2nd cycle _____, _____ and _____, _____
   3rd cycle _____, _____, _____, _____

5. 34 RAD _____, _____ and _____, _____, _____, _____, _____

6. 35 SSA _____, _____, _____

7. 36 SCA _____, _____, _____, _____, _____

8. 37 LPA _____, _____, _____, _____, _____

A. Clear X and A2
B. I^4 to X ($\overline{DBR}$ to I^4)
C. I^4 to X ($\overline{DBR}$ to I^4)
D. Clear A1
E. I^0 to A1 (adder to I^0)
F. I^2 to A2 ($\overline{A1}$ to I^2)
G. Clear Q2, I^3 to Q2 (adder to I^3)
H. I^4 to X ($\overline{Q2}$ to I^4)
I. Clear Q1, I^1 to Q1 (adder to Q1)
J. Force a logical operation within the adder.

The proof is in the prints--check your answers!

STATIC ENABLES AND TIMED TRANSFERS Worksheet #8: Instruction Series 4X

For each of the following function codes insert the appropriate letters (A-H) in proper sequence in order to obtain the required data transfers.

1. 40 STA _____, _____, _____, _____

2. 41 STQ _____, _____, _____, _____

3. 42 SACH _____, _____, _____, _____

4. 43 SQCH _____, _____, _____, _____

5. 44 SWA _____, _____, _____, _____

6. 45 STAQ _____, _____, _____ and _____, _____, _____, _____

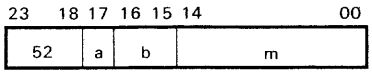
7. 46 SCHA _____, _____, _____, _____

8. 47 STI
   a. .0 _____, _____, _____, _____
   b. .1 to .3 _____, _____, _____, _____

A. Clear X and A2
B. I^2 to X (A1 to I^2)
C. I^3 to X (Q1 to I^3)
D. X to I^7 to E2
E. 1's to I^6 to I^7 to E2
F. B^b to I^6 to I^7 to E2
G. E2 to DBR (straight only)
H. E2 to DBR (straight or right shifted)

The proof is in the prints--check your answers!

CPR COMPARE (WITHIN LIMITS TEST)



a = addressing mode designator
b = index register designator
m = storage address

Figure 280. CPR Compare

Instruction Description: The quantity stored at address M is tested to see if it is within the upper limits specified by A and the lower limits specified by Q. The testing proceeds as follows:

1. Subtract (M) from (A). If (M) > (A), RNI from address P + 1; if not,
2. Subtract (Q) from (M). If (Q) > (M), RNI from P + 2; if not,
3. RNI from address P + 3.

Comments: The final state of the (A) and (Q) registers remains unchanged. (A) must be ≥ (Q) initially or the test cannot be satisfied. 77777777 is not sensed as negative zero.

It should be noted that the symbol means more positive than for this instruction.

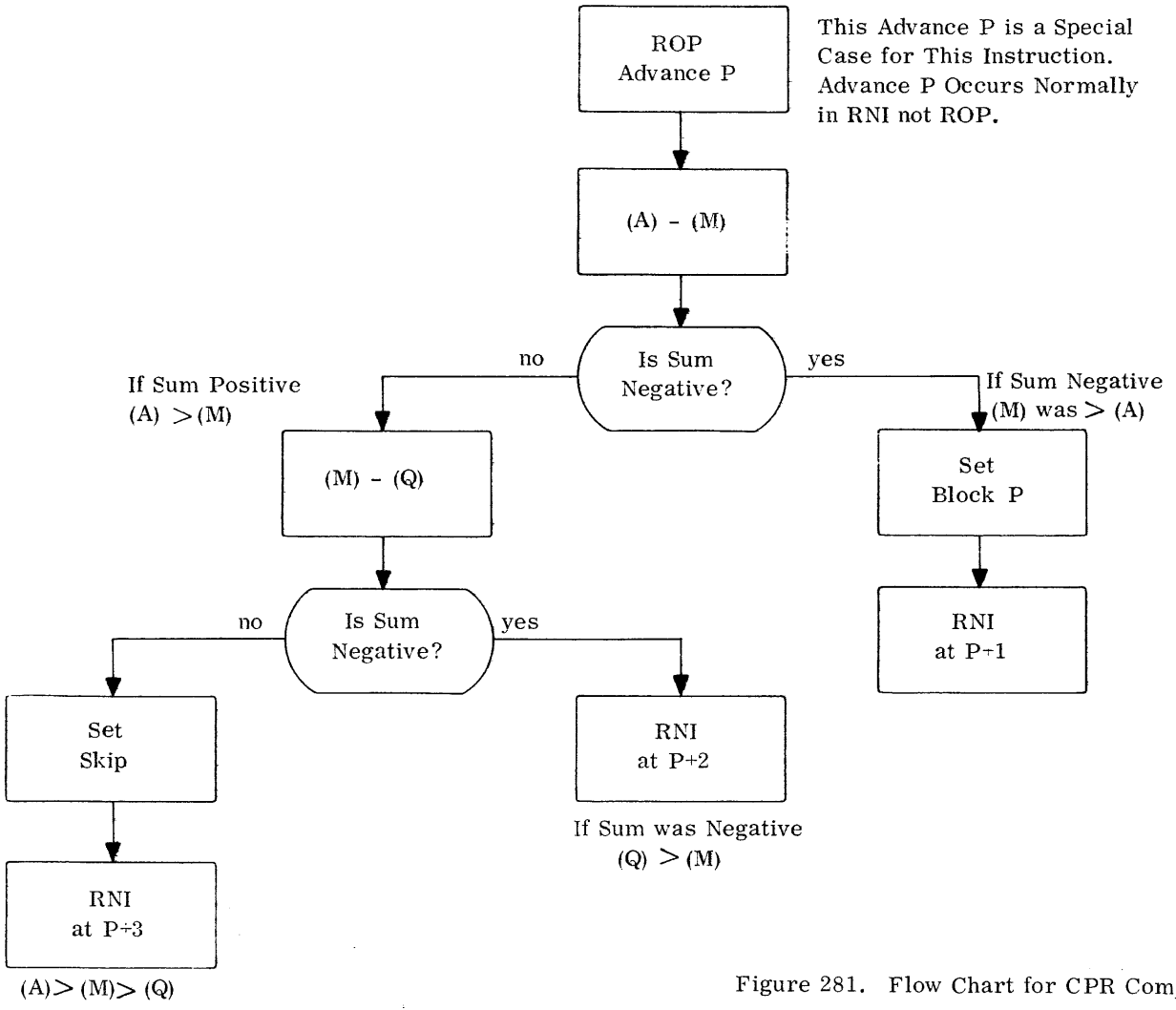


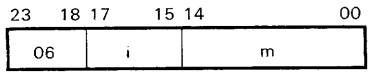
Figure 281. Flow Chart for CPR Compare

The following items are of note for a 52 instruction:

1. At V004 time of ROP, H231 is pulsed to advance P.
2. At V008 time ROP the Arithmetic timing chain is started to perform (A) - (M). If the sum is negative but not -0, (M) is greater than (A). This is sensed by the hardware by K584/585 (CPR Out of Limits) remaining set at Arithmetic time 5. In this case the Block P FF will be set and an RNI will be initiated at P + 1.
3. If (M) ≤ (A) a second Arithmetic pass, which is self-starting, will occur to perform (M) - (Q). If the sum is negative but not -0, (M) is less than (Q). This is sensed by the hardware by K584/585 (CPR Out of Limits) remaining set at Arithmetic time 5. In this case a normal exit to RNI is performed and RNI will occur at P + 2.

4. If the sum is positive or -0 at time 5 of the second Arithmetic pass (M) is within limits. In this case the Skip FF is set and an RNI is performed at P + 3.
5. If (M) (A) the second Arithmetic pass is blocked at the input to H501 by K584/585 (CPR Out of Limits) being set.
6. The CPR Out of Limits FF is set at time 4 of each Arithmetic pass. It is cleared at time 5 if a within limits condition exists by J564 and V585.

MEQ MASKED EQUALITY SEARCH



i = interval designator, 0 to 7
m = storage address

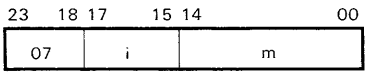
Figure 282. MEQ

Instruction Description: (A) is compared with the logical product of (Q) and (M). This instruction uses index register B1 exclusively. M is modified just prior to step 3 in the test below.

Instruction Sequence:
1. Decrement (B1) by i. (Refer to table below.)
2. If (B1) changed sign from positive to negative, RNI from P + 1; if not,
3. Test to see if (A) = (Q) (M). M = m + (B1). If (A) = (Q) (M), RNI from P + 2; if not,
4. Repeat the sequence.

Comments: i is represented by 3 bits, permitting a decrement interval selection from 1 to 8. Address modification may not be used.

MTH MASKED THRESHOLD SEARCH



i = interval designator, 0 to 7
m = storage address

Figure 283. MTH

Instruction Description: (A) is compared with the logical product of (Q) and (M). This instruction uses index register B2 exclusively. m is modified just prior to step 3 in the test below.

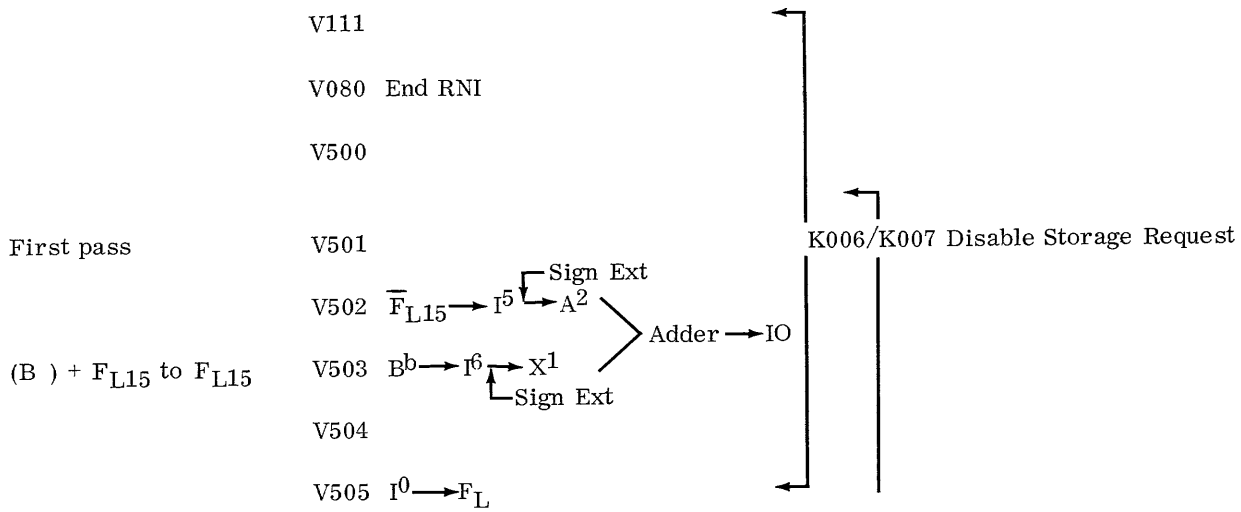
Instruction Sequence:
1. Decrement (B2) by 1. (Refer to table below.)
2. If (B2) changed sign from positive to negative, RNI from P + 1; if not,
3. Test to see if (A) $\geq$ (Q) • (M). M = m + (B2). If (A) $\geq$ (Q) • (M), RNI from P + 2; if not,
4. Repeat the sequence.

Comments: i is represented by 3 bits permitting a decrement interval selection from 1 to 8. Address modification may not be used.

Table 18. MTH COUNT EXAMPLE

Designator i	Decrement interval
1	1
2	2
3	3
4	4
5	5
6	6
7	7
0	8

The following is a brief summary of the first five arithmetic passes for an 06 or 07 instruction. They do not include a pass for indexing which could be specified by the instruction. The data flow and set times of important FFs are noted.



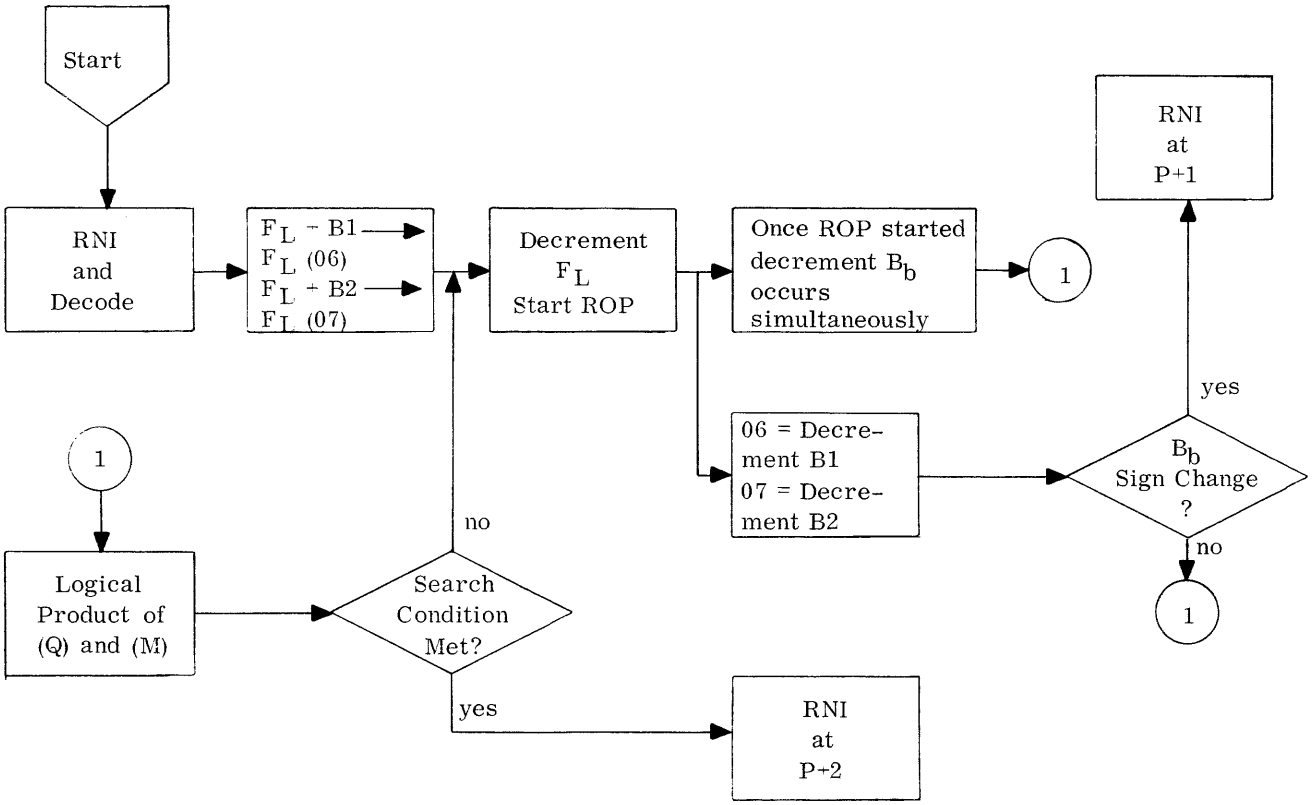
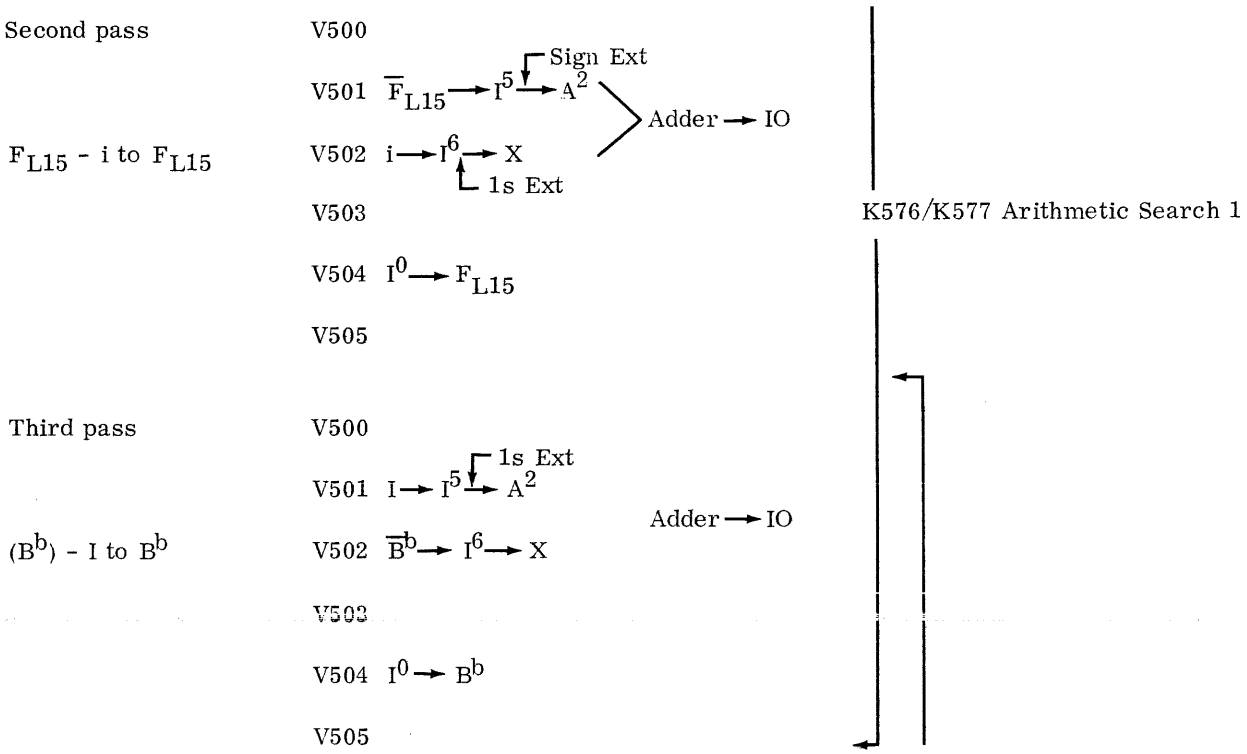
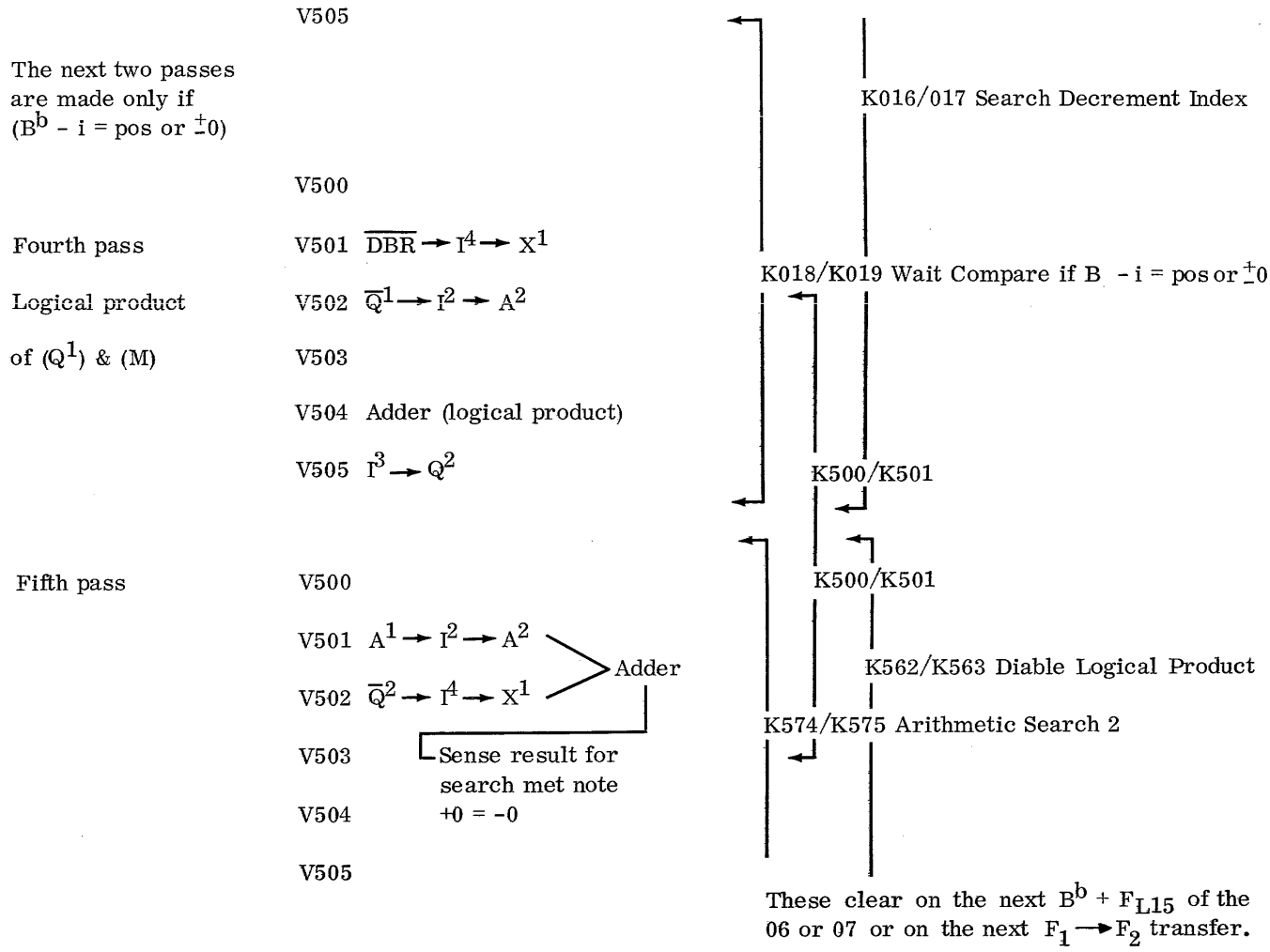


Figure 284. Flow Charts for 06 and 07 Instructions (the 06 uses B1 and the 07 uses B2)

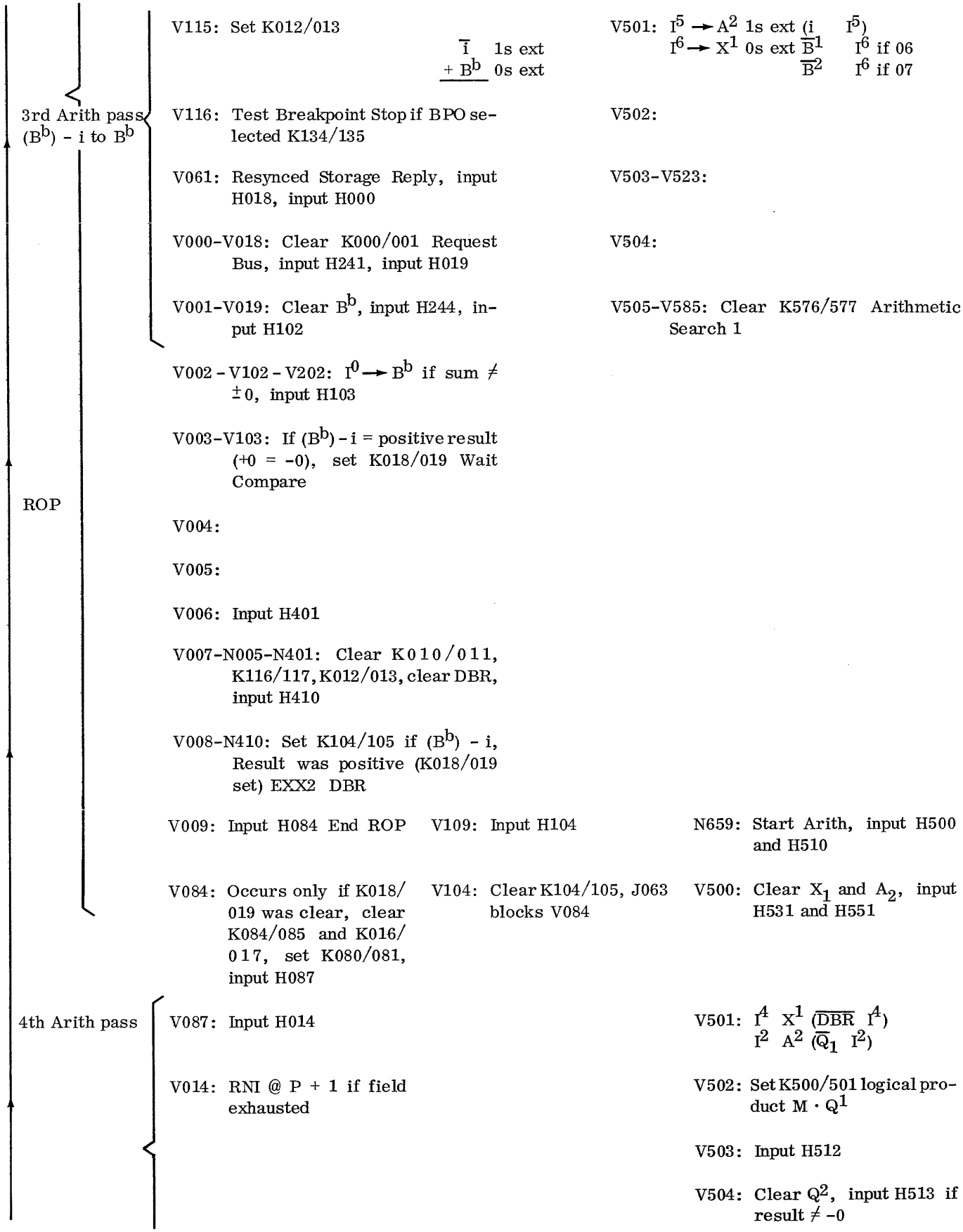


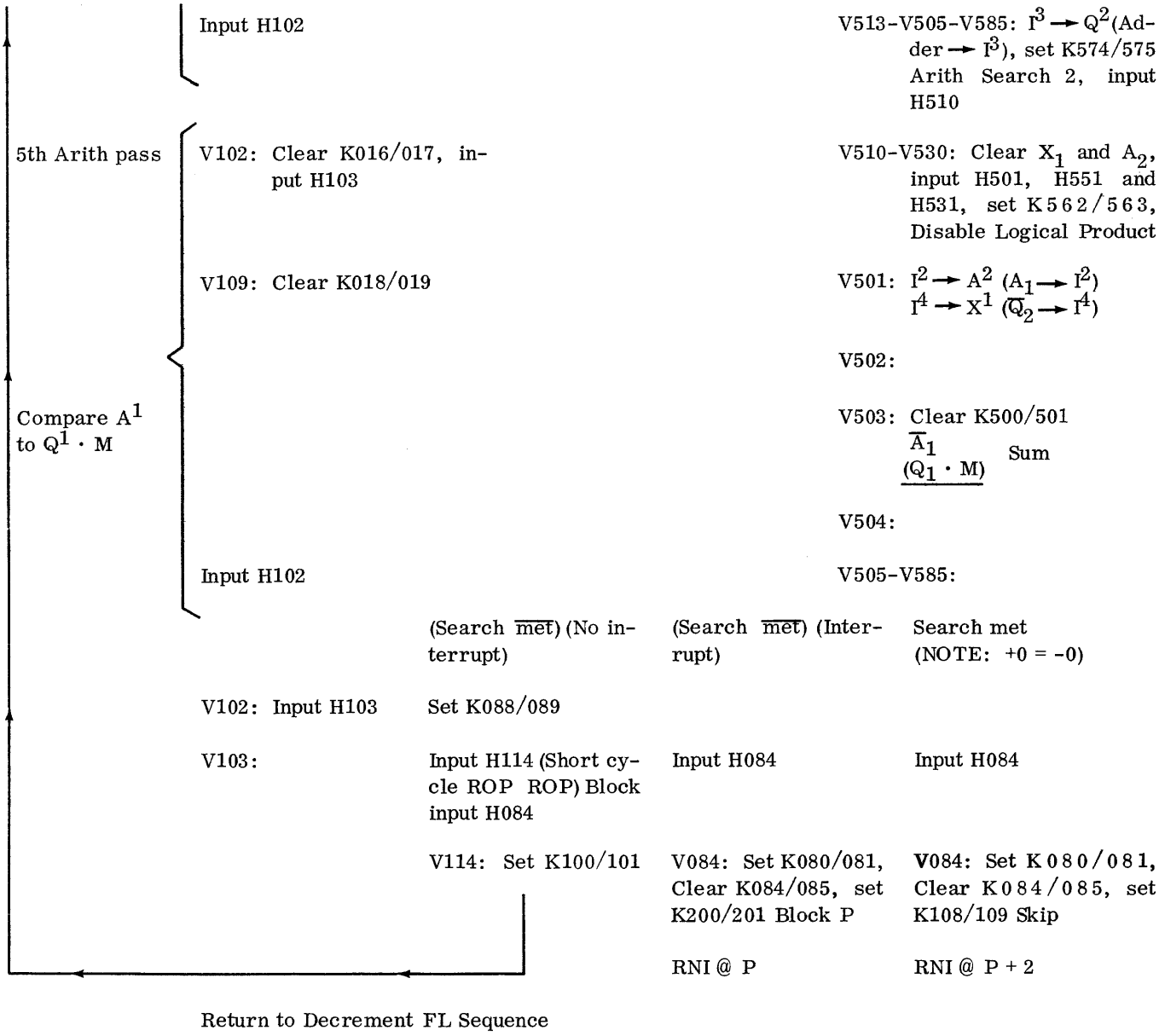


Timing for Search instructions, 06 and 07, beginning with V008 of RNI. These instructions provide a static indexing translation, (F604 = 1) and F424 = 1).

	V007:	Set K572/573
	V008: EXX2 → F_1	Set K570/571
	V009: If Arith Busy, set K008/009	V109: If Arith $\overline{Busy}$, input to H518
	V010: Test interrupts, clear K002/003	V518: $F_1 \rightarrow F_2$, clear K570/571
	V011: Input H080, set K006/007	V519:
	V080: Clear K080/081, set K084/085, set K100/101	V520: Clear K572/573
	V109: Input H100 if Arith $\overline{Busy}$	N659: Start Arith, input H500 and H510, set K536/537 FADR 1, set K596/597 FADR 2
1st Arith pass	V100: Clear K100/101, set K110/111	V500: Clear X_1 and A_2, input H541, H571

$(B^b)+F_{L15}$ to F_{L15}		V501: $I^5 \rightarrow A^2$ ($\overline{F} \rightarrow I^5$) $I^6 \rightarrow X_1$ ($\overline{B}^1 \rightarrow I^6$ if 06) ($\overline{B}^2 \rightarrow I^6$ if 07)
		Clear K574/575, set K576/577 Arithmetic Search 1
	Input H201 if $\overline{\text{Breakpoint Stop}}$	V502:
	N201: Clear F_{L15}, input H110	V503:
	V110-V210: Input H449, H105, Block setting of K000/001 Request Bus FF	V504: Clear K596/597 FADR 2
	V105: $I^0 \rightarrow F_{L15}$ if sum = ± 0, input H102, input H114	V505-V585:
	V106-V102-V202-V114: Clear K110/111, clear K006/007, set K100/101, input H103	Clear K536/537 FADR 1
	V103-V109: Input H100	N659: Start Arith, input H500 and H510, set K536/537 FADR 1, set K596/597 FADR 2
2nd Arith pass (F_{L15}) - i to F_{L15}	V100: Clear K100/101, set K110/111	V500: Clear X_1 and A_2, input H541, H571
		V501: $I^5 \rightarrow A^2$ sign ext ($\overline{F}_{L15} \rightarrow I^5$) $I^6 \rightarrow X^1$ (1s ext ($\overline{i} \rightarrow I^6$) as J018 = 0, this forces F521 and F522 to output 1s which in turn forces J281 and J282 to output 0s blocking a B^b -- I^6
	$\overline{i}$ 1s ext $+F_{L15}$ sign ext	
	Input H201	V502:
	N201: Clear F_{L15}, input H110, input H018 as K006/007 is clear	V503-V523:
	V110-V210-V018: Input H019, set K000/001 Request Bus (to start the ROP), input H449	V504: Clear K596/597 FADR 2
	V019-V105: $I^0 \rightarrow F_{L15}$ if sum $\neq \pm 0$, input H126 and H102, set K010/011, transmit F_{L15} on S bus	V505-V585:
	V102-V202-V126: Clear K110/111, set K104/105 Start Arith 2, set K016/017, input H117 and K103	Clear K536/537 FADR 1
	V109-V103-V117: Input H104, set K116/117 send Storage Request T650 = 1	N659: Start Arith, input H500 and H510
	V104-N050: Input H115, Clear K104/105	V500: Clear X_1 and A_2, input H541 and H571


V115: Set K012/013
$\overline{i}$ 1s ext
$+ B^b$ 0s ext
V501: $I^5 \rightarrow A^2$ 1s ext (i I^5)
$I^6 \rightarrow X^1$ 0s ext $\overline{B}^1$ I^6 if 06
$\overline{B}^2$ I^6 if 07
3rd Arith pass
$(B^b) - i$ to B^b
V116: Test Breakpoint Stop if BPO selected K134/135
V502:
V061: Resynced Storage Reply, input H018, input H000
V503-V523:
V000-V018: Clear K000/001 Request Bus, input H241, input H019
V504:
V001-V019: Clear B^b, input H244, input H102
V505-V585: Clear K576/577 Arithmetic Search 1
V002 - V102 - V202: $I^0 \rightarrow B^b$ if sum $\neq \pm 0$, input H103
V003-V103: If $(B^b) - i$ = positive result (+0 = -0), set K018/019 Wait Compare
ROP
V004:
V005:
V006: Input H401
V007-N005-N401: Clear K010/011, K116/117, K012/013, clear DBR, input H410
V008-N410: Set K104/105 if $(B^b) - i$, Result was positive (K018/019 set) EXX2 DBR
V009: Input H084 End ROP
V109: Input H104
N659: Start Arith, input H500 and H510
V084: Occurs only if K018/019 was clear, clear K084/085 and K016/017, set K080/081, input H087
V104: Clear K104/105, J063 blocks V084
V500: Clear X_1 and A_2, input H531 and H551
4th Arith pass
V087: Input H014
V501: I^4 X^1 ($\overline{DBR}$ I^4)
I^2 A^2 ($\overline{Q}_1$ I^2)
V014: RNI @ P + 1 if field exhausted
V502: Set K500/501 logical product $M \cdot Q^1$
V503: Input H512
V504: Clear Q^2, input H513 if result $\neq -0$



SHIFT INSTRUCTION

There are a number of instructions which require shifting such as shift, scale, multiply, and divide. The instructions to be discussed here will be the 12.0 to 13.3 series. This includes shift A, shift Q, and shift AQ. The shift instructions are described in the reference manual.

Execution of a shift instruction can be divided into two parts:

1. Forming the shift count.
   a. Number of places to shift a register.
   b. Direction of shift, left or right.
2. The shift cycle.

Forming shift count is similar to FADR.

Form shift count: $K = k + B^b$

FADR: $M = m + B^b$

The lower 15 bits of the instruction are k. B^b and k will be totaled in the adder and both k and B^b will be sign extended.

The shift count may come from I^0 to SC register or from the adder to SC register. Which path is taken depends on the sum of k and B^b.

Referring to figure 241, you will note that the SC register will hold the complement of the shift count. This is true for shift instructions 12.0 to 13.3.

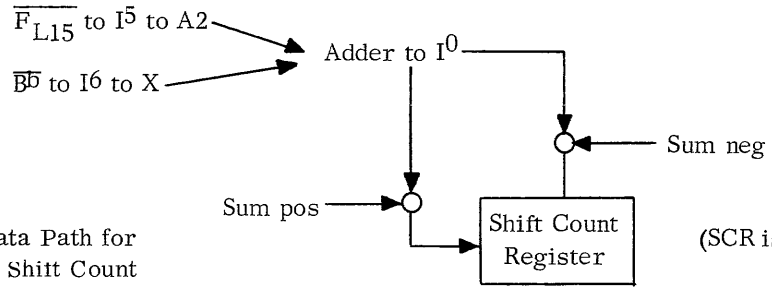


Figure 285. Data Path for Forming Shift Count

(SCR is on p 2-115 in Logic Diagrams.)

Translations are made from SC register to determine when the shift is complete.

As an example: Instruction 12.000001 is executed, a shift of one place. The value that would go to SC register would be 76 which is the 6-bit complement of the shift count. When the shift 1 takes place, SC register is advanced by 1 and then holds a 77 which indicates shift complete. Shift count register is a 6-bit counter, but for the shift instructions is considered as two 3-bit counters.

In Logic Diagrams, page 2-115, the left-hand group of flip-flops (B700/701 to B722/723) is the 1's counter. When the contents of a register is shifted 1 place left or right, this counter will be advanced by 1. When the 1's counter holds 7 it indicates that the 1's shifts are complete.

The right-hand flip-flops (same page) are the 10s counter (B730/731 to B752/753). When the contents of a register is shifted 10_8 octal places, this counter will be advanced by 1.

Using the shift cycle timing chain, shifting will be accomplished by a number of passes through the shift cycle.

One pass can shift a quantity 1 place, 10_8 places or 11_8 places. There is no other combination possible.

As you go through the timing which follows, refer to figures 242 and 243 and Logic Diagrams. Verify the paths. Look for the end-around path for left shift A and Q and AQ. Also trace the path for drawing sign on the right shift.

An example of how SCR counts is significant. For the example assume a desired shift of 13_8. The count shown for SCR is the count that would exist at the end of that pass.

Table 19. SCR COUNT EXAMPLE

SCR	Comments
64	Initial
75	1st pass--Shift 11_8
76	2nd pass--Shift 1
77	3rd pass--Shift 1
77	4th pass--used to clear Arith Busy FF

Note that only three passes were needed to complete the shifting. The additional pass is a "housekeeping" pass and would occur any time a 1s shift occurs as the last shift.

Timing for Shift Instructions

The RNI sequence for a shift instruction has taken place.

V380 (End RNI): Set K100/101 (start arith 1).

N687, N691, N693: Start arithmetic, input H500 (start arithmetic timing chain). Input H510 (clear A2 and X).

V500: Clear adder feeders A2 and X. Input H541 (I^6 to X) and H571 (I^5 to A2).

V501: Gate I^5 to A2 (static enable $\overline{F}$ to I^5). Gate I^6 to X (static enable $\overline{B^b}$ to I^6).

If the instruction is 12.0 or 13.0 (no index) the output of I^6 will be 0's.

V502

V503: Set K546/547 (transfer 10s enable).

Shift count is gated to rank 1 of SC register and then must be transferred to rank 2 so that advance SC register will take place properly. Transfer of 1's always happens at the first time of shift cycle, and on the first pass transfer of 10s must also take place to initialize the 10s counter.

V504

V505: If K is positive, gate adder to SC register (p. 1-3). Set K512/513 (shift left) or, if K is negative, gate I^0 to SC register. Set K510/511 (shift right). Input H600 and H610 (start shift cycle). Bit 23 of the sum is sensed to determine left or right shift.

Shift Cycle

V600: Clear A2 and Q2. Transfer 1's count, transfer 10s count first pass only. Set K560/561 (busy), set K578/579 shift cycle. If SCR $\neq$ 7X, 10s complete. Input to H561 OR H523. If SCR = 7X, 10s complete. Input to H551 OR H513.

In Logic Diagrams, page 2-99, note the input to H561 (I^2 to A2 shifted). This input will be made if 10s is not complete and the instruction is shift A or shift AQ.

The input to H523 (I^3 to Q2 shifted) will be made if 10s is not complete and the instruction is shift Q or shift AQ.

The 10s shift takes place between I^2, A2 and/or I^3, Q2. If 10s is complete the transfer from I^2I^3 to A2Q2

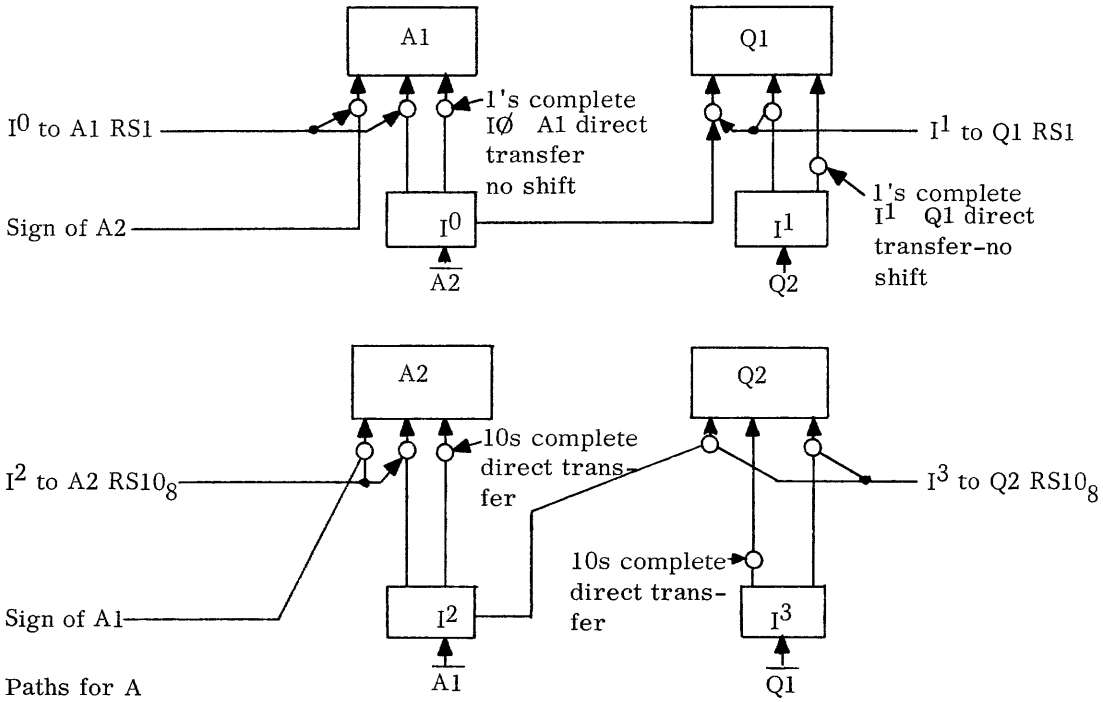


Figure 286. Shifting Paths for A

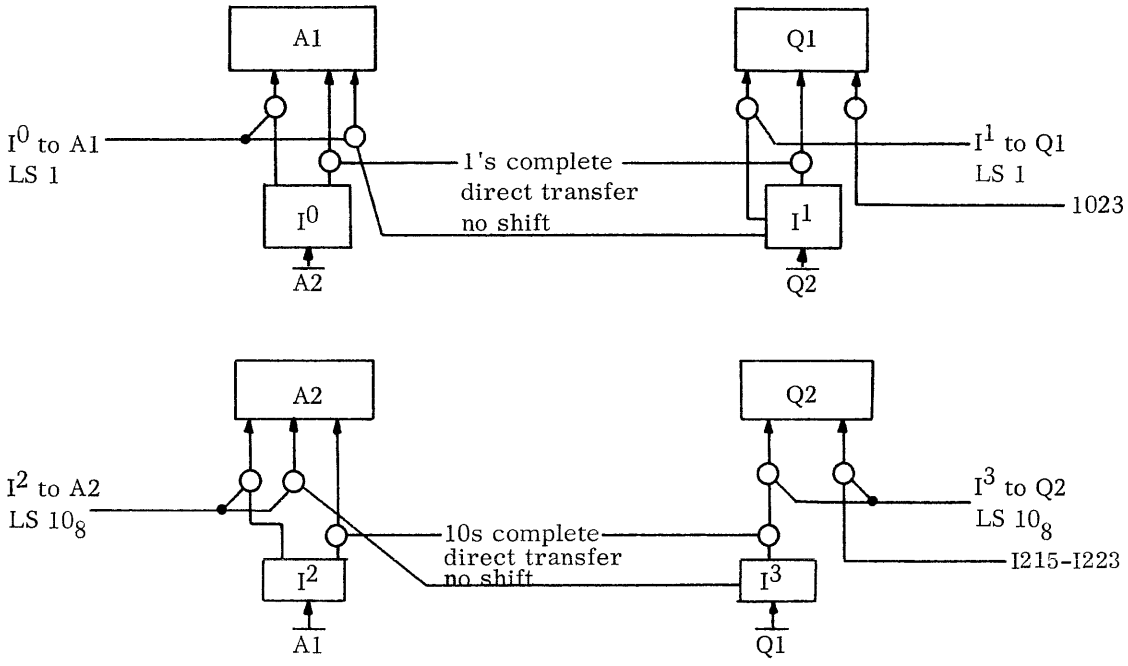


Figure 287. Shifting Paths for Q

will be direct. See page 2-95 for inputs to H551 and H513.

V601: If 10s complete, advance 10s counter and gate I^2 to A2 (enable $\overline{A1}$ to I^2) and I^3 to Q2 (enable $\overline{Q1}$ to I^3). The transfers will be right or left 10_8 enabled by shift left FF or shift right FF being set (p. 2-99). If 10s complete, gate I^2 to A2 ($\overline{A1}$ to I^2) and I^3 to Q2 ($\overline{Q1}$ to I^3). Clear K546/547 (transfer 10s enable).

V602: Clear A1 and Q1. Transfer 10s count. If SCR ≠ X7 (1's complete), input H525 (I^0 to A1 shifted 1) and H565 (I^1 to Q1 shifted 1), p. 1-31. If SCR = X7 (1's complete), input H515 (I^0 to A1) and H555 (I^1 to Q1). If SCR = 77 (shift complete), clear K560/561 (busy).

V603: If 1's complete, gate I^0 to A1 (enable $\overline{A2}$ to I^0) and I^1 to Q1 (enable Q2 to I^1) shifted 1. Advance 1's counter. If 1's complete, gate I^0 to A1 and I^1 to Q1 straight. The enables to I^0 and I^1 are the same whether shifted or not shifted. If SCR ≠ 77 (shift complete), input H600 to repeat shift cycle.

Single-Precision Multiply

The multiply A instruction, function code 50, will multiply the contents of A register and the contents of a 24-bit memory location, leaving a 48-bit product in AQ registers. (Chapter 3 describes this instruction.)

The multiply operation can be subdivided into three parts. Refer to figure 244.

1. Initialization.
   a. Makes (A) positive if negative and places it in Q register. This is the multiplier.
   b. Makes multiplicand from memory positive if negative and places it in X register.
   c. Records if original signs of multiplicand and multiplier were unlike.
   d. Checks first multiplier bit (MB); the multiplier bit is the least significant bit of the multiplier.
2. Multiply step.
   a. If MB = 1, add X and A2. Enable $\overline{sum}$ to I^0.
   b. If MB = 0, enable $\overline{A2}$ to I^0.
   c. Shift I^0I^1 right 1 to A1Q1.
   d. Check next MB
   e. Return to step 2a 23_{10} times for the 24_{10} multiply steps.

The multiply step will form the 48-bit product in AQ registers. After the multiply step, A and Q registers will be swapped for programing convenience. The product is formed through a series of additions and shifts.

3. Complement and/or swap
   a. Complement AQ, the product if unlike signed operands were multiplied. Note: Since the arithmetic section can multiply only positive numbers, the product must be positive at this time. If either or both the multiplier and multiplicand were negative before the multiply step, the numbers were complemented during initialization.
   b. Swap A and Q so that the most significant bits of the product are in Q. This will always take place.

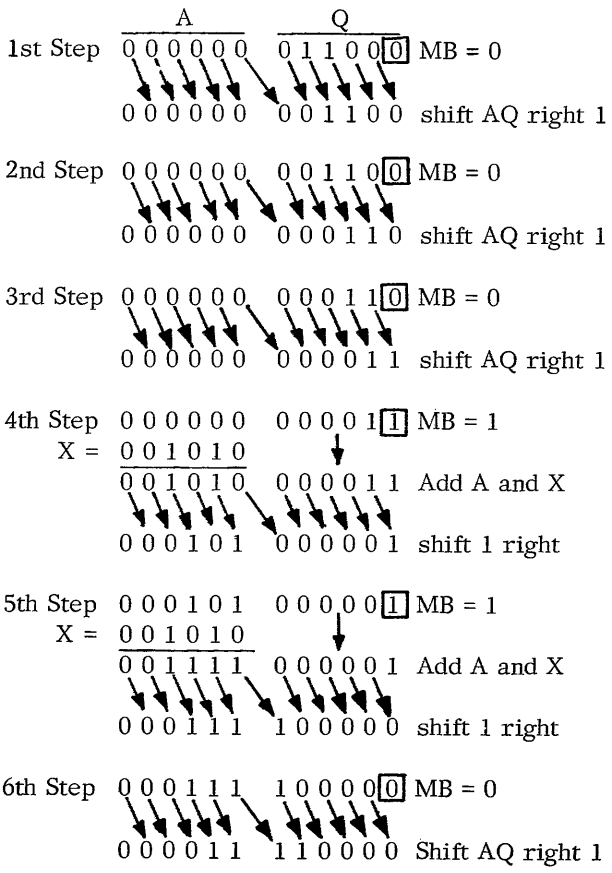
In the following example of the multiply step 6-bit registers will simply be used. Finding the product will require six multiply steps.

Example 1

$12_8 \times 30_8 = ?$

(A) = 30 = multiplier (M) = 12 = multiplicand

Initialization is complete: A = 00, Q = 30, X = 12



AQ = 0360 before swap.

The pencil and paper method:

$$\begin{array}{r} 12 \\ \times 30 \\ \hline 00 \\ 36 \\ \hline 360 \end{array}$$ The answers match.

Example 2 (Negative times positive)

A = 60 (M) = 35

Initialization is complete: A = 00, Q = 17, X = 35

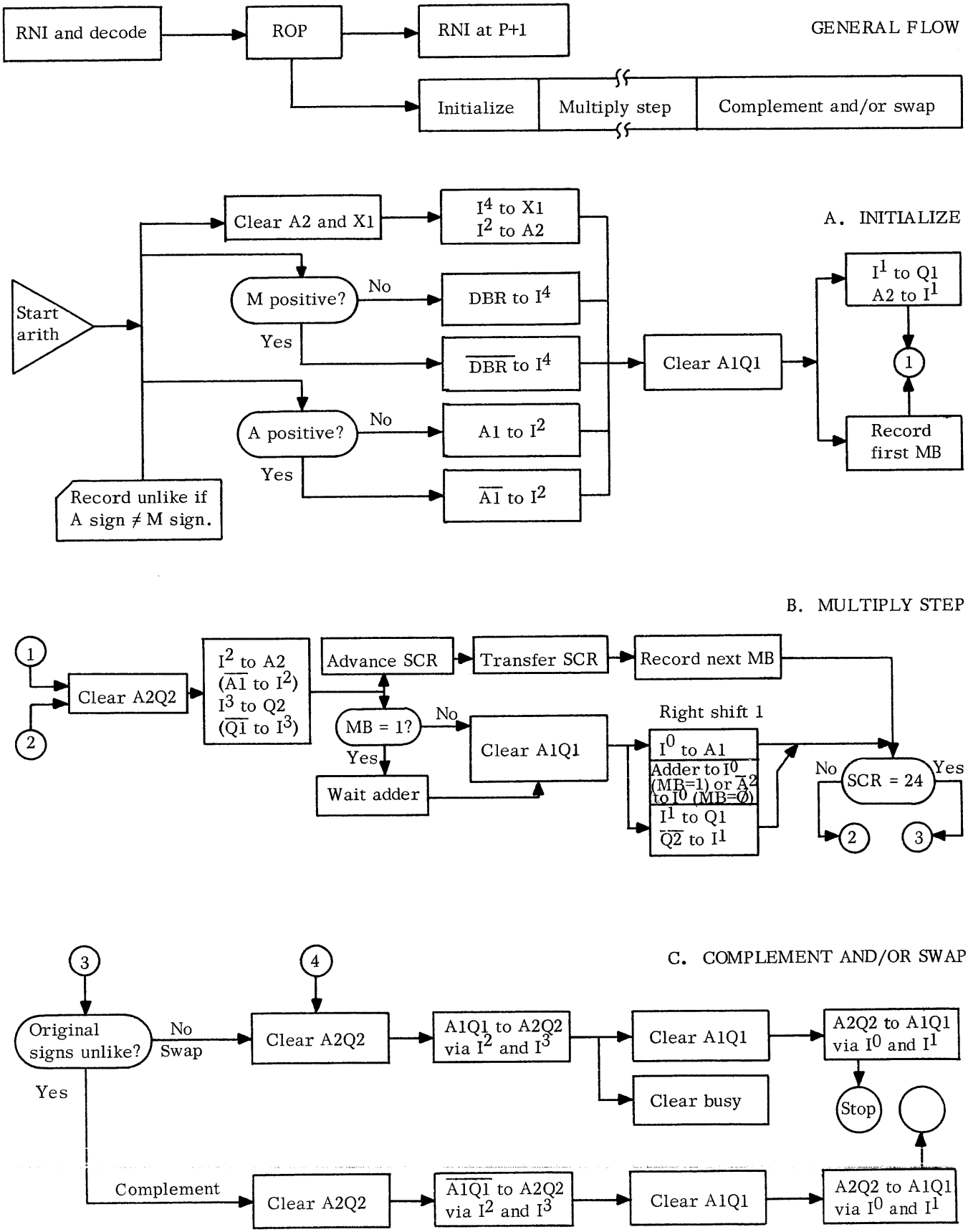


Figure 288. Flow Charts for Multiply A

Note: Using 6 bit registers, bit 05 is the sign bit. Thus 60= - 17. Initialize cycle makes all numbers positive.

```
                A          Q
1st Step  0 0 0 0 0 0   0 0 1 1 1 [1] MB = 1
   X =    0 1 1 1 0 1
          0 1 1 1 0 1   0 0 1 1 1 1  Add A and X
          0 0 1 1 1 0   0 0 1 1 1 1  shift 1 right

2nd Step  0 0 1 1 1 0   1 0 0 1 1 [1] MB = 1
   X =    0 1 1 1 0 1
          1 0 1 0 1 1   1 0 0 1 1 1  Add A and X
          0 1 0 1 0 1   1 1 0 0 1 1  shift 1 right

3rd Step  0 1 0 1 0 1   1 1 0 0 1 [1] MB = 1
   X =    0 1 1 1 0 1
          1 1 0 0 1 0   1 1 0 0 1 1  Add A and X
          0 1 1 0 0 1   0 1 1 0 0 1  shift 1 right

4th Step  0 1 1 0 0 1   0 1 1 0 0 [1] MB = 1
   X =    0 1 1 1 0 1
          1 1 0 1 1 0   0 1 1 0 0 1  Add A and X
          0 1 1 0 1 1   0 0 1 1 0 0  shift 1 right

5th Step  0 1 1 0 1 1   0 0 1 1 0 [0] MB = 0
          0 0 1 1 0 1   1 0 0 1 1 0  shift 1 right

6th Step  0 0 1 1 0 1   1 0 0 1 1 [0] MB = 0
          0 0 0 1 1 0   1 1 0 0 1 1  shift 1 right
```

AQ = product = 0663

AQ holds a positive product which must be complemented to equal -0663.

The pencil and paper method:

```
  +35
x -17
 313
  35
-663
```

After the complement and swap cycle AQ would hold 7114 with the most significant bits in Q and least significant bits in A register.

Swapping AQ to leave the least significant bits in A is justified in that, in most case, the product with sign can be contained in A register. Thus, in most cases, the programmer can follow a multiply instruction with an add or subtract A instruction.

What is the decimal equivalent of the largest signed number that can be displayed in a 24-bit register?___

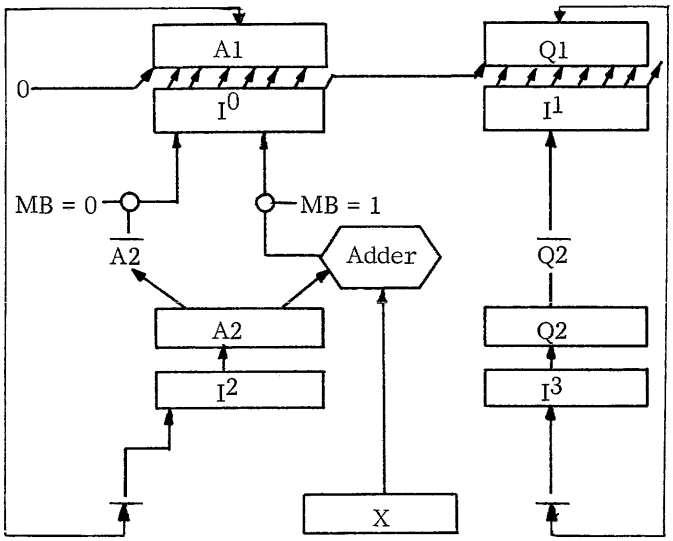


Figure 289. Data Flow for Multiply Step

The multiplier bit sampled to determine the input to I^0 inverter rank is effectively the least significant bit of Q1, which holds the multiplier.

The first MB is sampled during initialization (bit 0 of I^1). The remaining MB, bit 1 of Q2 register, will be sampled at Q510 and Q511. This is necessary because the sampling occurs prior to the shift. Bit 01 will be shifted into bit 00 position after the sample but before the next addition.

The multiply will be performed in 24_{10} multiply steps. Shift count register will count the steps and will terminate multiply step when SCR = 30_8.

Timing for Multiply A Instruction

Refer to figure 244.

Main control's responsibility for single-precision multiply is:

1. To read the instruction from memory (RNI sequence) and decode.
2. To read the operand from memory (ROP sequence) and gate the operand to DB register.
3. To start the arithmetic section when the operand has arrived.

Main control will then initiate an RNI sequence for the next instruction. Complete timing is given in the command timing charts.

In the arithmetic section, main control has just performed the RNI sequence. We will pick up the timing at V008 of ROP sequence.

V008: Gate EXX2 to DBR (DBR = (M)).
Set K104/105 (start arith 2).

The next odd time that arithmetic is not busy and not wait function, the arithmetic timing chain will receive a start pulse, N687, N691, or N693 (p. 1-29).*

*Logic Diagrams

(N687, N691, N693, and N659 are the start arithmetic terms.) Refer to the logic diagrams pages 2-93.

N659, N687, N691 & N693: Start Arith.
Input H510 (clear adder feeders A2 and X).
Set K518/519 (A2 to I^1) to provide a path for the multiplier during initialization.

The flow path for a positive multiplier is:
$\overline{A1}$ to I^2 to $\overline{A2}$ to I^1 to A1

The flow path for a negative multiplier is:
A1 to I^2 to $\overline{A2}$ to I^1 to Q1

Note: An inverter rank inverts its input.
Set K532/533 (unlike signs 1) via J520 if sign of multiplier is not the same as sign of multiplicand.
Set K516/517 (A1 to I^2) if multiplier is negative. Provides for complementing (A).

V510, V530: Clear adder feeders A2 and X.
Set K560/561 (arithmetic busy) to prevent another start pulse to the arithmetic section during multiply operation.
Input H531 (I^4 to X1).
Input H551 (I^2 to A2).

V531, V551: Gate I^2 to A2 (A positive: static enable $\overline{A1}$ to I^2; A negative: static enable A1 to I^2 (p. 2-139).
If K516/517 is set, J865, J866, and J867 = 1's, and J880, J881, and J882 = 0's. If K516/517 is clear, the opposite occurs.
Gate I^4 to X1 (M positive: static enable $\overline{DBR}$ to I^4; M negative: static enable DBR to I^4 (p. 2-151).

For the multiply instruction J904 translates a 1 for 50 (M positive). The J904 output of a 1 will drop DBR to I^4 enable (J905-J908 = 0's) and bring up the $\overline{DBR}$ to I^4 enable (J915-J918 = 1's).
Gate K532/533 to K514/515 (unlike signs 1 to unlike signs 2).
Input H514 (clear A1).
Input H524 (clear Q1).

V514, V524: Clear A1 and Q1.
Input H555 (I^1 to Q1).
Clear K516/517 (A1 to I^2). Drop enable to complement multiplier.

V555: Gate I^1 to Q1 (static enable A2 to I^1). Q1 now holds the positive multiplier.

Sense first MB; if MB = 1, set K528/529.
Input H500 to start multiply step.
Input H512 (clear Q2).
Input H522 (clear A2).

The arithmetic section has completed initialization.
1. Q1 register holds a positive multiplier.
2. X1 register holds a positive multiplicand.
3. A sign flip-flop "remembers" if like or unlike signed operands are to be multiplied.

MULTIPLY STEP

Static enables that will be up during multiply step are:

$\overline{A1}$ to I^2

$\overline{Q1}$ to I^3

A2 and X to adder (These transfer paths exist at all times and are never disabled).
$\overline{Q2}$ to I^1

V500, V512, V522: Clear A2 and Q2.
Set K510/511 (right shift); multiply uses right shift only.
Clear K518/519 ($\overline{A2}$ to I^1) used only on initialize.
Input H513 (I^3 to Q2).
Input H551 (I^2 to A2).

Input H503 if MB = 0 (short cycle) or
input H501 if MB = 1 (long cycle) to allow adder propagation time (figure 290).

Referring to figure 289, you will notice that if MB = 0, $\overline{A2}$ is enabled to I^0 inverter rank. All that happens when the MB = 0 is a right shift of AQ, 1 bit position.

V501, V513, V551: Set K530/531 (static enable $\overline{sum}$ to I^0).
Gate I^2 to A2 (static enable $\overline{A1}$ to I^2) and I^3 to Q2 (static enable $\overline{Q1}$ to I^3).

V502: Wait for adder.

V503: Advance SCR.
Input H514 (clear A1).
Input H524 (clear Q1).
V513, V531 if MB = 0; gate I^2 to A2 and I^3 to Q2.

V504: Clear A1 and Q1.
Transfer SCR.
Set K520/521 (static enable $\overline{A2}$ to I^0) if MB = Ø.
Input H525 (I^0 to A1 shifted 1).
Input H565 (I^1 to Q1 shifted 1).
} Occurs if MB = Ø or 1

V505, V575: Gate I^0 to A1 RS1. (Static enable: if MB = 0, $\overline{A2}$ to I^0; if MB = 1, adder to I^0.)
Gate I^1 to Q1 RS1 (static enable $\overline{Q2}$ to I^1).
Test bit 1 of Q2 for next MB.
Set K528/529 if MB = 1.
Input H500 to restart timing chain.
Input H512 (clear Q2).
Input H522 (clear A2).

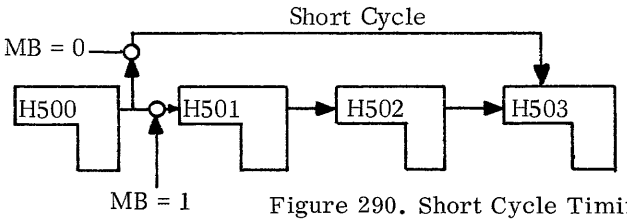


Figure 290. Short Cycle Timing

If SCR $\neq 30_8$ (24_{10}), repeat multiply step. If SCR = 30_8, set K520/521 ($\overline{A2}$ to I^0 enable). Multiply step is complete.

COMPLEMENT AND/OR SWAP

If sign of multiplier is the same as the multiplicand, set K588/589 ($\overline{A1}$ to I^3 and $\overline{Q1}$ to I^2) static enables for swap cycle. The timing for swap is continued following the complement cycle timing.

If unlike signed operands were multiplied, set K516/517 (static enable A1 to I^2) and K526/527 (static enable Q1 to I^3). This provides the enables to be used in complementing the product.

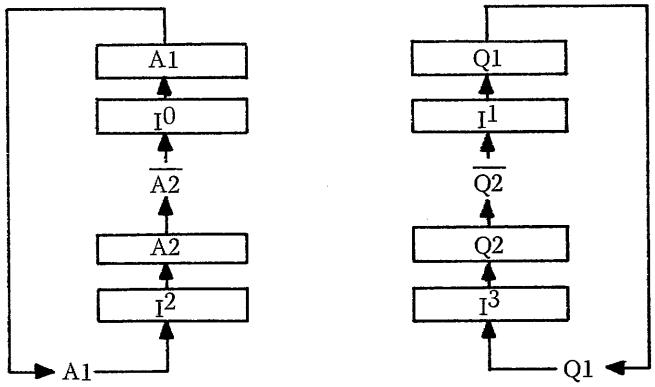


Figure 291. Data Flow for Complement Cycle

V500: Clear A2 and Q2.
Input H611 (p. 2-101) to start complement cycle timing.
Block input to H501 (stops multiply step timing).

V611: Gate I^2 to A2 (static enable A1 to I^2) and I^3 to Q2 (static enable Q1 to I^3).
Set K556/557 (block I^0 to A1) if AQ = 0.
Set K558/559 (block I^1 to Q1) if AQ = 0.

Setting of these two flip-flops will occur if the product in AQ is 0. At this time the swap has not taken place nor has the sign been corrected (complement cycle). This will prevent a -0 as a product even if unlike signed operands were multiplied.

V612: Clear A1 and Q1.

Clearing A1 and Q1 is done by N500, N502, N504 and N510, N512, N514, respectively (p. 2-95).

NOTE: H612 on their input AND gates.
Clear K516/517 (drop enable A1 to I^2, p. 2-139) and K526/527 (drop enable Q1 to I^3, p. 2-145).
Set K588/589 (enable $\overline{A1}$ to I^3 and $\overline{Q1}$ to I^2).

The enables now present are for swap cycle which must follow complement cycle.

V613: Gate I^0 to A1 (static enable $\overline{A2}$ to I^0) and I^1 to Q1 (static enable $\overline{Q2}$ to I^1).

This transfer will place the complemented product in AQ but will not take place if complementing would produce a 48-bit -0.

Input H620 to start swap cycle.

SWAP CYCLE

V620: Clear A2 and Q2.
Set K586/587 (complement cycle lockout) to stop timing chain when swap has been completed. This is also V500 time if like signed operands were multiplied.

V500: Set K586/587 (complement cycle lockout).
Block input H501.
Input H611 (starts swap cycle).

V611: Gate I^2 to A2 (static enable $\overline{Q1}$ to I^2) and I^3 to Q2 (static enable $\overline{A1}$ to I^3).
Set K556/557 (block I^0 to A1) and K558/559 (block I^1 to Q1). These two flip-flops are discussed at V611 of complement cycle. The flip-flops could have been set during complement cycle (p. 2-101).

V612: Clear A1 and Q1.
Clear K560/561 (arithmetic busy).

After duplicating F1 into F2 and clearing wait function FF, the arithmetic section will be available should the next instruction require it. Remember that arithmetic has been running independent of main control. Main control has been performing RNI for the next instruction.

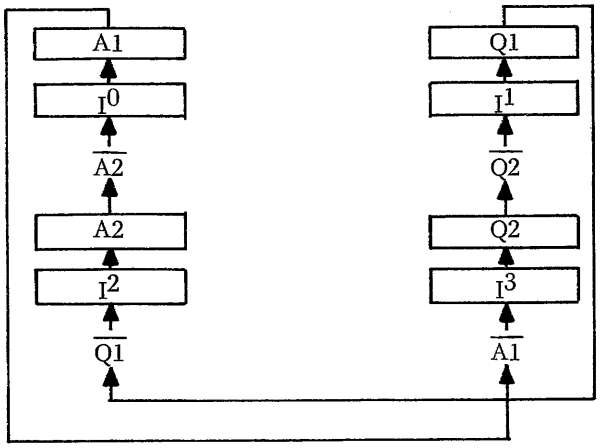


Figure 292. Data Flow for Swap Cycle

Summary of arithmetic phase times for a multiply operation:

Initialize = 5 Ø times
Multiply step = 96 to 142 Ø times depending on MBs
Swap = 4 Ø times

V613: Gate I^0 to A1 ($\overline{A2}$ to I^0) and I^1 to Q1 ($\overline{Q2}$ to I^1). These transfers are blocked if AQ is holding all 0's at V611 time.

The multiply instruction has now been executed and the product with proper sign is in AQ register.

COMPARISON OF MAIN CONTROL AND ARITHMETIC TIMING

V008	(M) to DBR				
V009			N691		
V084	End ROP of multiply instruction		V510		Initialize, 1 pass
V087			V531		5 Ø times
V014	Set request bus		V514		
N051	Set main control priority		V555		
N050			V500	short	Multiply step, 24 passes
V117	Set storage request		V501	cycle	6 Ø times, long cycle
N050		long	V502		4 Ø times, short cycle
V115	Set lockout	cycle	V503		
V116			V504		
N051			V505		
N050		like	V500		Complement cycle, optional
V061	Resync reply	signs	V611		1 pass
V000			V612		4 Ø times
V001			V613		
V002			V620		
V003			V611		
V004			V612	clear busy	Swap cycle, 1 pass
V005			V613		4 Ø times
V006					
V007					
V008	Next instruction to F1 (best case)				

Summary of arithmetic phase (Ø) times for a multiply operation:

Initialize = 5Øt
Multiply step = 96 to 142Øt depending on MBs
Complement = 4Øt
Swap = 4Øt

MULTIPLY TIMING Worksheet

Indicate what complementing is required after multiply step and the number of short cycles.

1. (A) = 10000000 Complement required: ____
   (M) = 57777777 Number of short cycles: _____

2. (A) = 37777777 Complement required: ____
   (M) = 37777777 Number of short cycles: _____

3. (A) = 77777777 Complement required: ____
   (M) = 00000001 Number of short cycles: _____

4. (A) = 77777777 Complement required: ____
   (M) = 40000000 Number of short cycles: _____

5. (A) = 40000000 Complement required: ____
   (M) = 00040000 Number of short cycles: _____

List the five preceding multiply operations by number below in order of least time to most time required.

Least time _____
_____
_____
_____
Most time _____

SINGLE-PRECISION DIVIDE

Divide A instruction, function code 51, will divide (AQ) registers by a 24-bit divisor located at address M in memory. On completion of divide, the quotient with sign will be in A register and the remainder with sign will be in Q register.
(See chapter 3 for description of instruction.)

The divide operation can be sub-divided into three parts. Refer to the flow chart for divide.

1. Initialization.
   a. Makes (AQ) positive if negative and returns it to AQ as part of the first divide step. This is the dividend.
   b. Makes the divisor negative, if positive, and places it in the X register. (A divide is a series of subtractions, thus, divisor must be negative).
   c. Records if original signs of divisor and dividend are unlike.
   d. Records sign of dividend. This determines sign of remainder.
2. Divide Step.
   a. Subtracts divisor from dividend.
   b. If result is negative, enable $\overline{A_2}$ to I^0.
      If result is positive, enable sum to I^0.
   c. Shift I^0 I^1 left 1 to A1Q1.
   d. Form quotient bit (QB) of D if result was negative. Form QB of 1 if result was positive and insert QB into least significant end of Q reg. bit 00. The quotient is assembled in the Q register.
   e. Continue divide step. A total of 24 passes will be made.
3. Complement and/or swap. (Complement cycle is optional but used when necessary; results of executing divide instruction follow the rules of mathematics.)
   a. Complement quotient if unlike signed operands were divided.
   b. Complement remainder if dividend was negative.
   c. Prevent -0 for both quotient and remainder.
   d. Swap A and Q to place the quotient in A and the remainder in Q.

Swapping A and Q registers at the end of a divide instruction is done for programing convenience. The multiply A instruction can follow directly. In many cases the remainder is not important and the quotient can be incremented or decremented following divide.

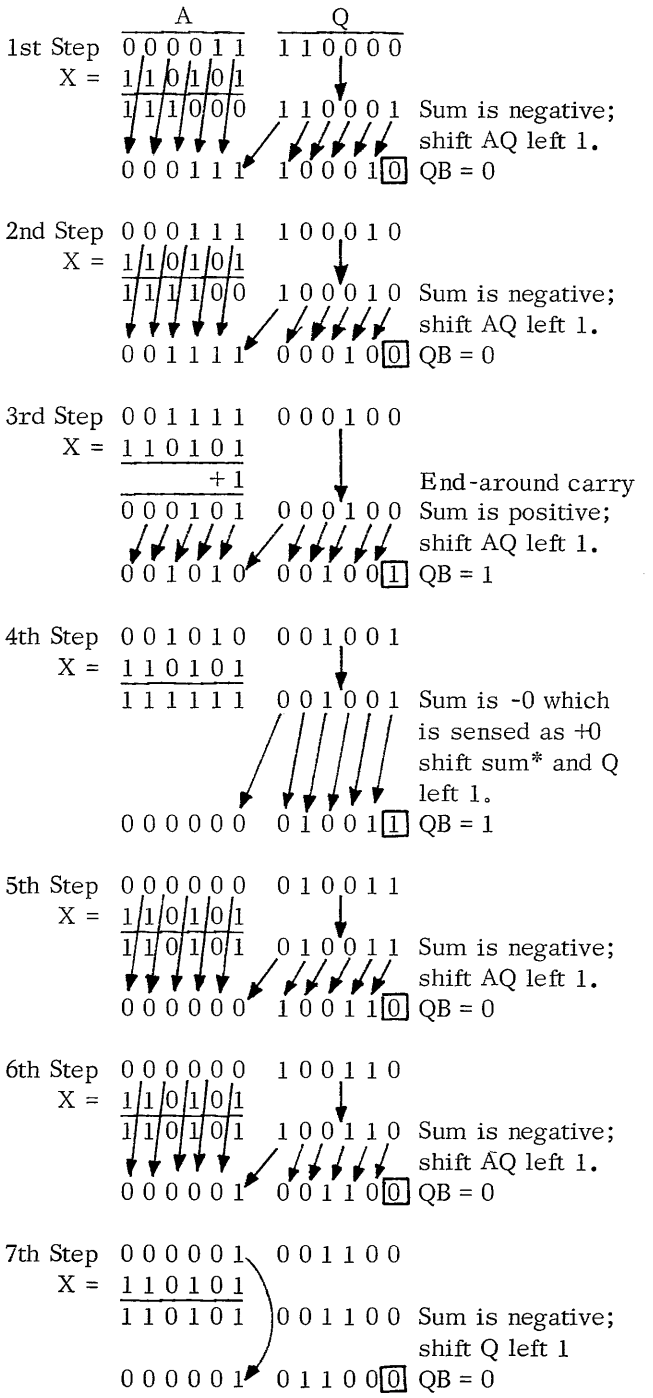
The divide operation will be performed with successive subtraction and left shifting. The arithmetic adder is additive, so to subtract it is necessary to complement and add. That is the reason for insuring that the divisor is negative during initialization.

The following is an example of divide step. Six-bit registers will again be used to simplify the example. Six divide steps plus one partial step will be needed to divide a number.

Problem: 361 ÷ 12 = ?
AQ = 0361 = dividend
M = 12 = divisor

After initialization, which is part of the first divide step:

X = 110101



*The sum, -0, will be gated back to A register during the shift as +0.

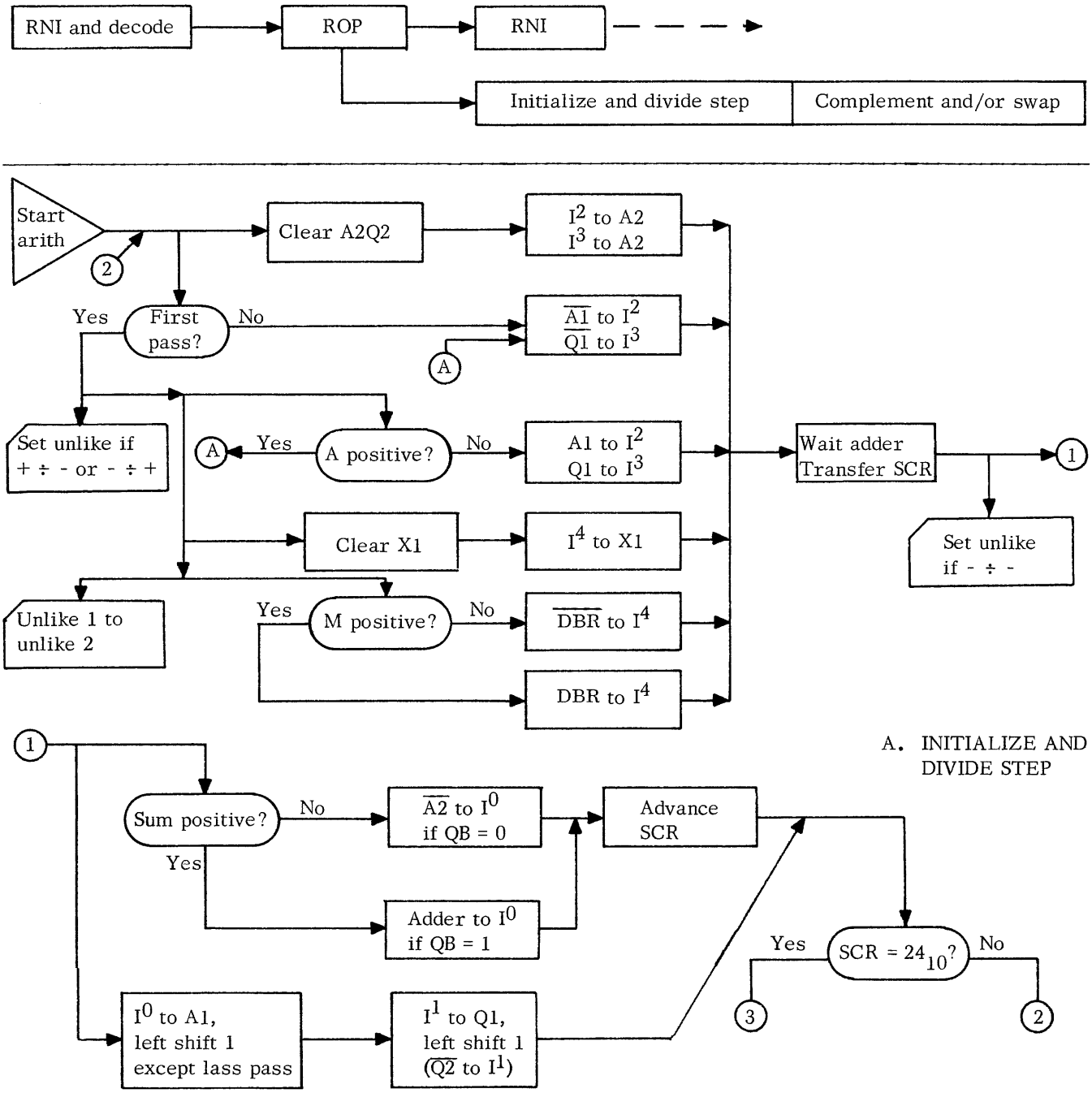


Figure 293. Divide Flow Charts

The last divide step is different only in that A is not shifted. The sensing of sum positive or negative, the forcing of quotient bit, and the shifting of Q are all exactly the same. This is necessary in order to leave the remainder in tact in "A" yet shift the final QB into Q.

Using the pencil and paper method, check the answer.

$$12\overline{)361}\quad 30$$
$$\underline{36}$$
$$01$$

Q = 30 and A = 01 is a quotient of 30 with a remainder of 1.

Since complement is not needed in this case, swap cycle would follow divide step. Swap cycle would present the result as the programmer would see it.

Initialization insures a positive dividend and a negative divisor.

Static enables for divide step:

1. $\overline{A1}$ to I^2
2. $\overline{Q1}$ to I^3
3. $\overline{Q2}$ to I^1
4. A2 and X to adder
5. If the sum is positive, enable adder to I^0 (the output of the adder is $\overline{sum}$) or, if the sum is negative, enable $\overline{A2}$ to I^0.

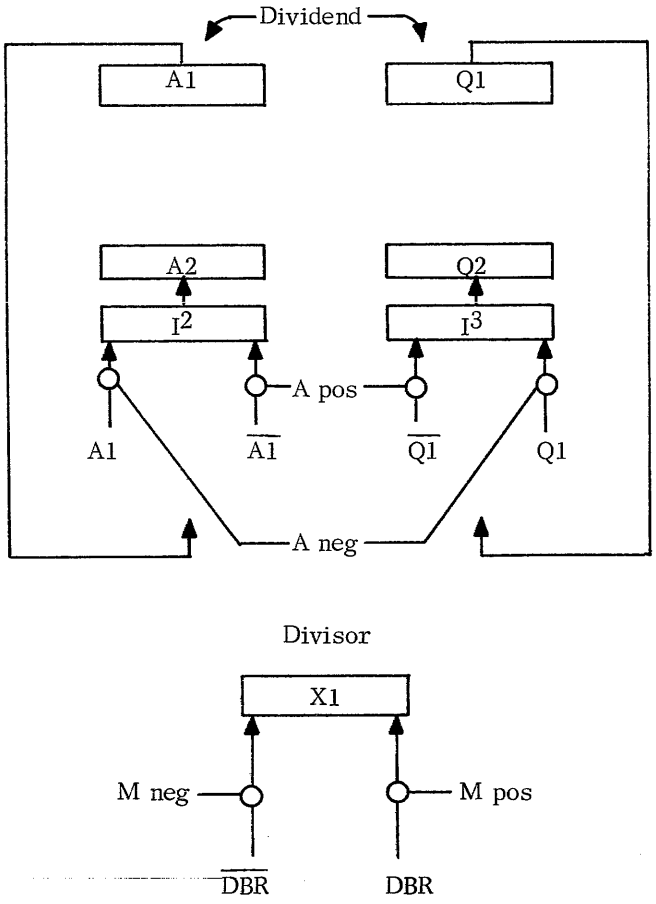


Figure 294. Data Flow for Initialization

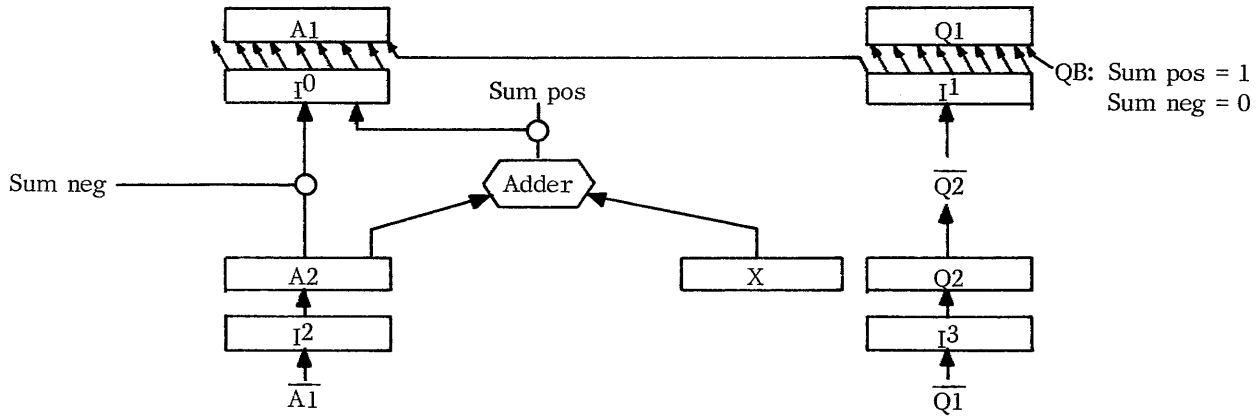


Figure 295. Data Flow for Divide Step

A divide fault may occur when the divide instruction is executed. The arithmetic section will sense for a divide fault on the first and second divide step.

Divide fault will occur if the sum is positive on the first or second divide step. Both of these two passes must find the sum negative and force a QB = 0. (The divide step does use the left shift but it is not end-around.)

Divide fault occurs when quotient with sign cannot be contained in a 24-bit register.

The following two examples show a divide fault on the first pass.

1. 00 / 1745 or + 0 divided into any number

```
        A          Q
     001111     100101
 X = 111111
    ┌001110
    └──────► +1
     001111  positive on first pass
```

2.

```
       307
   5 / 1745
       17
        2
```

```
        A          Q
     001111     100101
 X = 111101
    ┌001100
    └──────► +1
     001101  positive on first pass
```

In the 3300 Computer when a divide fault occurs on the first pass, it indicates that the quotient would be more than 24 bits in length.

Below is an example of divide fault on the second pass:

```
                 42
             3 / 146
```

```
              A           Q
1st Pass   000001      100110
   X =     111100         ↓
           111101      100110   Sum negative;
           ↓↓↓↓↓      ↙↙↙↙↙↙    shift AQ left 1
           000011      001100   QB = 0

2nd Pass   000011      001100
   X =     111100         ↓
           111111      001100   Sum sensed pos;
                      ↙↙↙↙↙↙    shift 0's to A, Q L1
           000000      011001   QB = 1
```

When the divide fault is sensed on first or second pass, that pass is completed. In the 3200 Computer, a divide fault on the second pass indicates that the quotient would have been 24 bits long excluding sign.

Timing for the Divide A Instruction

Refer to figure 293. Main control's responsibility for single-precision divide:

1. Read the instruction from memory (RNI sequence) and decode.
2. Read the divisor from memory (ROP sequence) and gate divisor to DB register.
3. Start arithmetic section.

Main control will then initiate an RNI sequence for the next instruction. Complete timing is given in the command timing charts. For this discussion, timing covered will that of the arithmetic section. The RNI sequence of a 51.X and the ROP sequence to V008 time has taken place.

V008: Gate EXX2 to DB register (DBR = (M). Set K104/105 (start arith 2).

The next odd time that the arithmetic section is not busy and not wait function, the arithmetic timing chain will receive a start pulse, N687, N691, N693 and N659. (p. 1-29).

N687, N691, N693: Start arithmetic. Input H500. Input H510 (clear adder feeders A2 and X). Input H512 (clear Q2). (Initialization occurs during the first divide step.) Set K516/517 (enable A1 to I^2) and K526/527 (enable Q1 to I^3) if A1 is negative; this will permit complementing a negative dividend. Set K532/533 (unlike signs 1) if dividing unlike signed numbers. Clear K538/539 (divide fault).

If the divide fault FF is set at this time, it indicates that a fault occurred on a previous divide instruction. Since the flip-flop is cleared during initialization of the divide, its being set at the end of the divide indicates that this instruction produced the divide fault.

V500: Clear A2 and Q2. Clear K530/531 (enable $\overline{\text{sum}}$ to I^0). If SCR $\neq$ 30, clear K520/521 (enable $\overline{A2}$ to I^0). Input H513 (I^3 to Q2). Input H551 (I^2 to A2).
First pass only: Clear X. Set K512/513 (shift left). Set K560/561 (busy). Input H531 (I^4 to X).

If K538/539 (divide fault) is set, block input to H501 and clear K560/561 (busy). This stops arithmetic timing and terminates the divide operation.

V501: I^2 to A2 ($\overline{A1}$ to I^2) and I^3 to Q2 ($\overline{Q1}$ to I^3).
First pass only: If A is negative, gate A1 to I^2, block $\overline{A1}$ to I^2, gate Q1 to I^3, block $\overline{Q1}$ to I^3. I^4 to X1: If M is positive, DBR to I^4; if M is negative, $\overline{\text{DBR}}$ to I^4. Gate K532/533 to K514/515. (unlike signs 1→unlike signs 2). At this time there will be a positive. Dividend in A_2 Q_2 and a negative divisor in X.

V502: Transfer SC register (p. 2-115).

V503: Input H514 (clear A1). Input H524 (clear Q1).
First pass only: Set K532/533 if dividend is negative and dividing like signs. Clear K526/527 (Q1 to I^3) used if complemented dividend.
Set K550/551 (block last shift A) if SCR rank 2 = 24_{100}; actually, this would be the 25th half pass, which is the same as the 24 previous passes except A is not shifted. (Last QB shifted into Q00).

V504: Clear A1Q1.

First pass only: Clear K516/517 (A1 to I^2), used if complemented dividend.

Input H565 (I^1 to Q1 shifted). If K550/551 is clear, input H525 (I^0 to A1 shifted). If K550/551 is set, input H515 (I^0 to A1 direct); this would be the last pass.

V505: I^0 to A1 (LS1, except last pass). If sum is positive, adder to I^0; if sum is negative, A2 to I^0. I^1 to Q1 (LS1), Q2 to I^1. If sum is positive, insert quotient bit of 1; if sum is negative, insert quotient bit of 0.

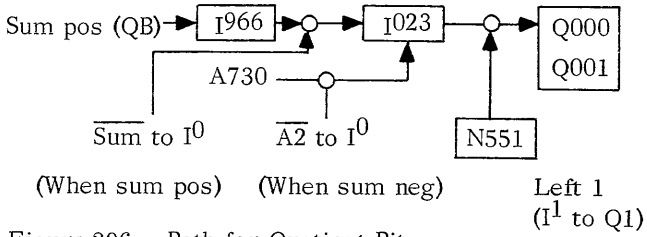


Figure 296. Path for Quotient Bit

For the divide, I^{023} = QB.

The path for the quotient bit looks complex, but it allows for other transfers also, such as AQ end-around left shift.

Input H500 (to continue timing). Input H512 (clear Q2). Input H522 (clear A2).

If SCR rank 2 $\neq 24_{10}$, repeat divide step. If SCR rank 2 = 24_{10}, continue to complement and/or swap. If K532/533 (unlike signs 1) is clear, proceed to swap (V500) as divided positive by positive.

If K532/533 (unlike signs 2) is set, continue with complement cycle. The flip-flop's being set indicates unlike signed operands where divisor and/or dividend were negative.

Set K516/517 (A1 to I^2) enable for the complement. Set K526/527 (Q1 to I^3).

V500: Clear A2 and Q2. Input H611 (starts complement cycle timing). Block input to H501 (stops main arithmetic timing).

V611, V621: (See p. 2-101). Gate I^2 to A2 (A1 to I^2 enable) and I^3 to Q2 (Q1 to I^3 enable). Actual complementing of data occurs right here. If dividend (original (A)) is positive, set K542/543 (block clear A1) to leave the remainder positive. If dividend (original (A)) is negative, set K592/593 (block clear Q1) to leave the quotient positive. If A1 = 0, set K556/557 (block I^0 to A1) to prevent a remainder of -0. I^0 contains complemented (A1) at this time). If Q1 = 0, set K558/559 (block I^1 to Q1) to prevent a quotient of -0. I^1 contains complemented (Q1) at this time).

V612: Clear A1 if K542/543 is clear. Clear Q1 if K592/593 is clear. Clear K516/517 (A1 to I^2) and K526/527 (Q1 to I^3) as complement enables no longer needed. Set K588/589 ($\overline{A1}$ to I^3 and $\overline{Q1}$ to I^2) enables for swap.

V613: Gate I^0 to A1 ($\overline{A2}$ to I^0) if K542/543 and K556/557 are clear. (The N5XX gating terms are on p. 2-95 and the flip-flops are on p. 2-127.) Gate I^1 to Q1 ($\overline{Q2}$ to I^1) if K592/593 and K558/559 are clear. Input H620 to start swap cycle timing.

V500: Block input H501 if complement cycle not required or

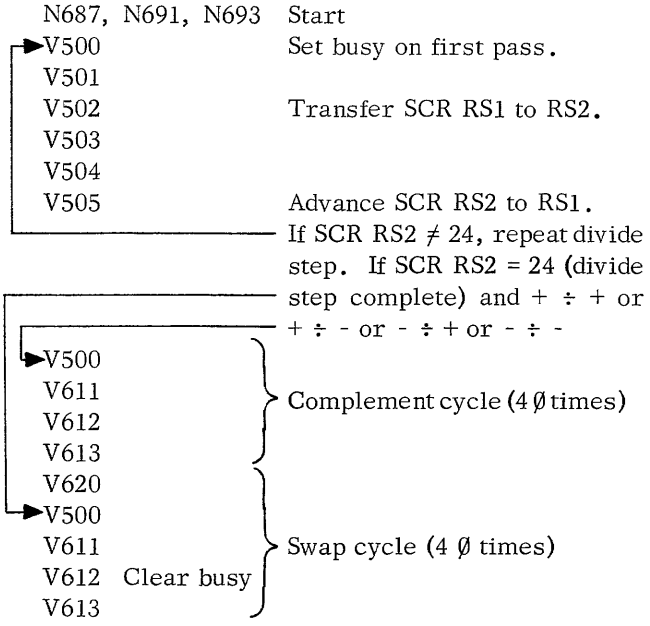
V620: if complement cycle just completed. Clear K542/543 (block clear A1) and K592/593 (block clear Q1). Clear A2 and Q2. Set K586/587 (complement cycle lockout). Input H611.

V611: Gate I^2 to A2 ($\overline{A1}$ to I^2) and I^3 to A2 ($\overline{A1}$ to I^3).

V612: Clear A1 and Q1; clear K560/561 (busy), K556/557 (block I^0 to A1), and K558/559 (block I^1 to Q1).

V613: Gate I^0 to A1 ($\overline{A2}$ to I^0) and I^1 to Q1 ($\overline{Q2}$ to I^1).

General Arithmetic Timing for Divide



1. How long would K560/561 (arithmetic busy) be set if divide fault were sensed on the first divide step? _____ On the second divide step? _____
2. How long would K560/561 (arithmetic busy) be set if the divide operation included the complement cycle? _____

SELF-EVALUATION QUIZ ON CHAPTER 12

TRUE OR FALSE:

1. The arithmetic section of the 3300 is partially independent of main control since it has its own timing chain and its own F register.

2. The 3300 Adder is subtractive in nature and yields an output of $\overline{sum}$.

3. If -0 were added to -0 in the 3300 Adder, the sum would be +0.

4. A selective complement is an exclusive OR function.

5. A logical product is a boolean AND function and does not require the adder.

6. Optional hardware is necessary to perform a 48-bit precision add in the 3300.

7. The adder is not used in the execution of a single-precision load instruction.

8. The adder forms part of the transfer path for the enter A instruction.

9. An I^0 to B^b transfer is necessary for the load I instruction.

10. The transfer path $\overline{X1}$ to I^1 to Q1 is used for the load Q instruction.

11. The 52 instruction will advance P during its first ROP.

12. The compare out-of-limits FF will set for each arithmetic pass of the 52 instruction.

13. An interrupt may not be sensed during execution of an 06 instruction because ROP to ROP cycle progression is used.

14. A +0 does not equal -0 for the 07 instruction.

15. A swap cycle will always occur at the end of the single-precision multiply instruction.

16. The multiplier for the single-precision multiply instruction is located at M and M + 1 and this quantity will affect the amount of time necessary to execute the instruction.

17. The divide step of a single-precision divide instruction require 31_8 arithmetic passes.

18. A swap cycle is optional at the end of the single-precision divide instruction.

19. Adder propagation requires 4 Ø times.

20. Both the 50 and 51 instructions set K560/561 (arithmetic busy) to insure that main control will not generate additional start pulses while the instructions are being executed.

Score Yourself:
This was a good quiz.
If you missed none or one, congratulations!
If you missed two or three, you're okay.
If you missed four or more--I don't know you!

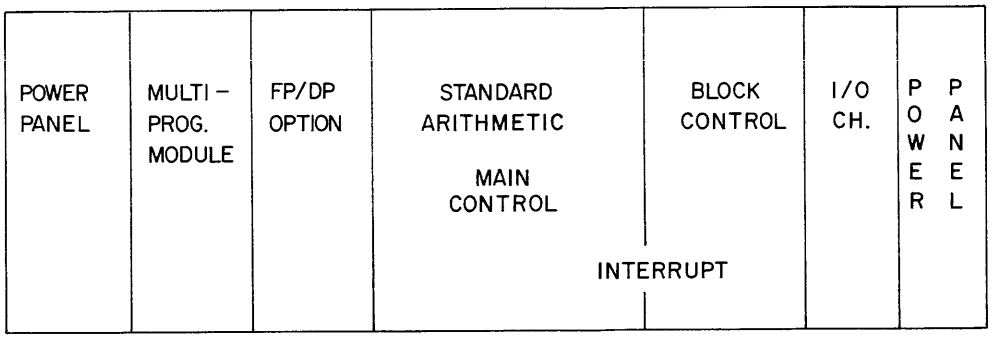


Figure 297. Physical Location of the FP/DP Option

CHAPTER 13

FLOATING-POINT/ DOUBLE-PRECISION OPTION

The FP/DP Option gives the 3300 System hardware capabilities for executing the instructions listed in table 20. The FP/DP Option is designated 3310 and becomes part of the CPU when added to the system.

Physically, the FP/DP option is located in chassis 4 and consists primarily of a 24-bit optional adder and a 48-bit E register. The optional or left adder is disabled during single precision operations but during double precision operation the optional adder operates on the higher order 24 bits. Figure 297 shows the physical location of the FP/DP option. Figure 298 is a block diagram of the arithmetic section with the FP/DP option present.

ENABLING THE 48-BIT ADDER

During the 56, 57, and 60-63 instructions the 48-bit adder is enabled by setting K804/805 (optional arithmetic busy). Figure 299 is a simplified logic diagram showing how optional arithmetic busy affects the adder. Reference figure 299 and consider the following cases:

Case 1. Assume optional arithmetic busy set and a 48-bit operation in which group 8 is a generate and groups 9-15 are passes. See the diagram below. Note that these conditions generate a carry input to group 0. The hardware operation begins with optional

Table 20. FLOATING-POINT/DOUBLE-PRECISION INSTRUCTIONS

OPERATION FIELD	ADDRESS FIELD	INTERPRETATION
55 ELQ	- - - -	Transfer (E_L) to Q
QEL	- - - -	Transfer (Q) to E_L
EUA	- - - -	Transfer (E_U) to A
AEU	- - - -	Transfer (A) to E_U
EAQ	- - - -	Transfer (E) to AQ
AQE	- - - -	Transfer (AQ) to E
56 MUAQ, I	m, b	Multiply AQ
57 DVAQ, I	m, b	Divide AQ
60 FAD, I	m, b	FP addition to AQ
61 FSB, I	m, b	FP subtraction from AQ
62 FMU, I	m, b	FP multiplication of AQ
63 EMD, I	m, b	EP division of AQ

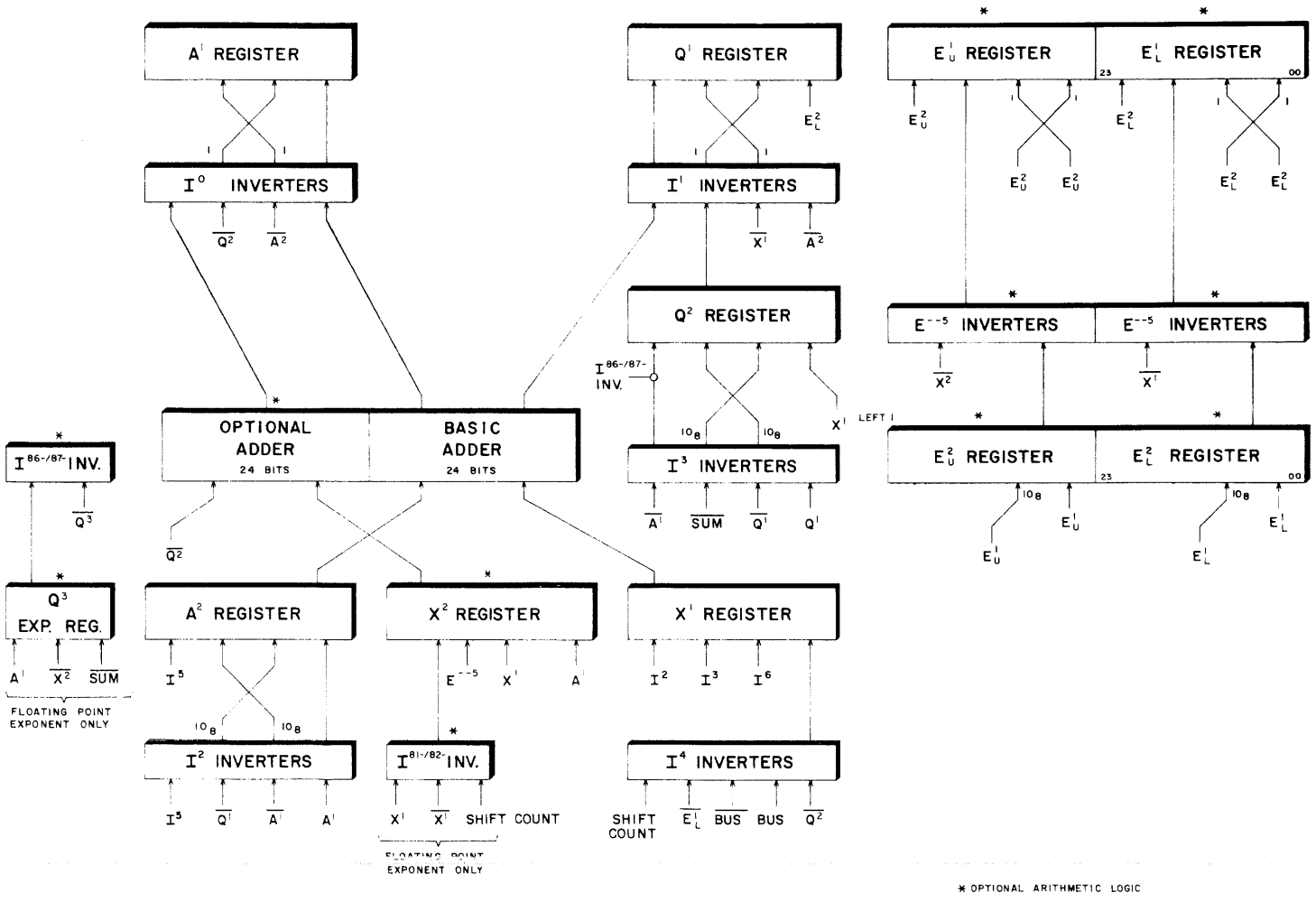


Figure 298. Block Diagram of Arithmetic Section With Floating-Point/Double-Precision Option Present

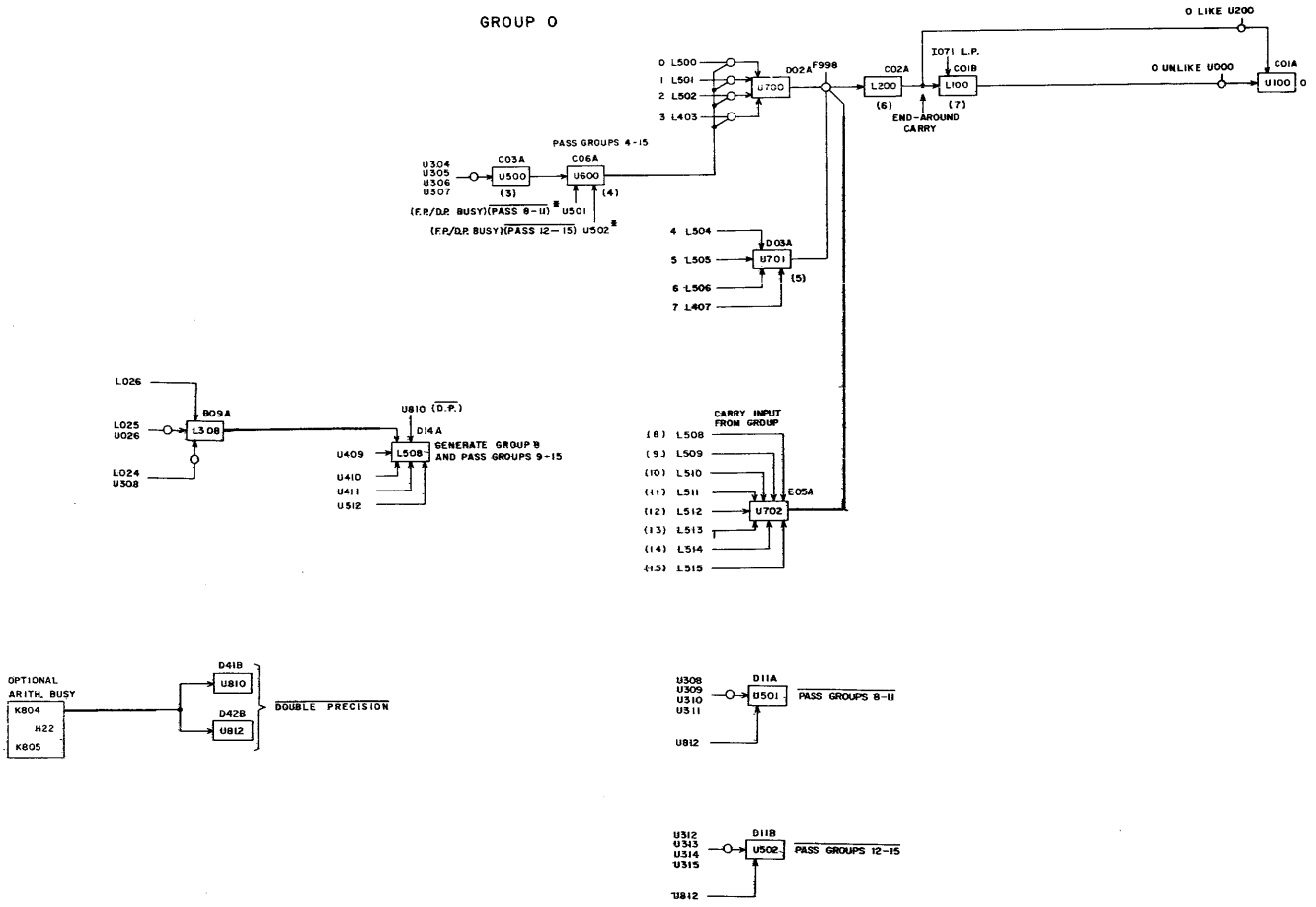
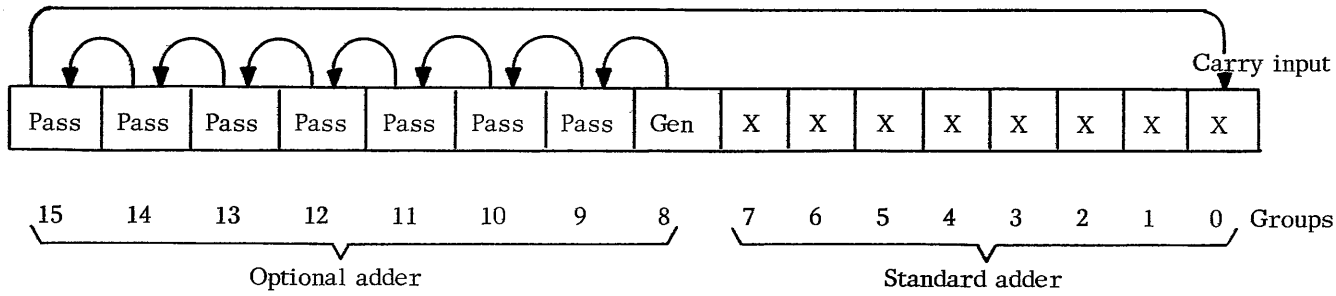


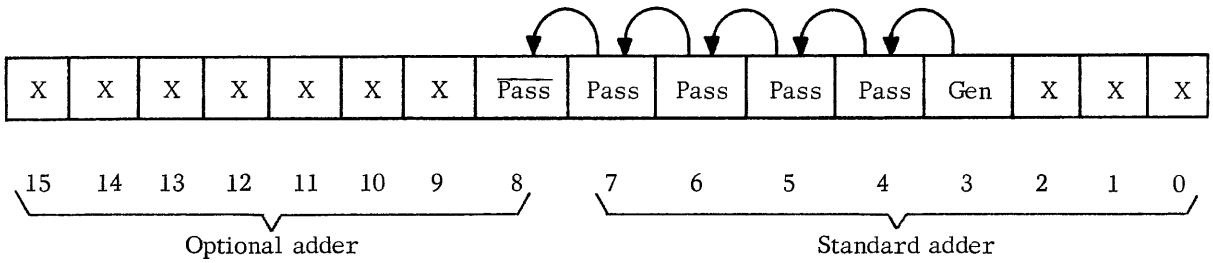
Figure 299. Enabling the 48-Bit Adder

arithmetic busy being set which drives U810 and U812 to 0. U810 being a 0 enables L508 through L515 to sense for generate and pass conditions in the optional adder. For this case L508 = 1 which forces U702 = 0. U702 breaks the input to L200 and L200 = 1 which indicates that group 0 has a carry input.



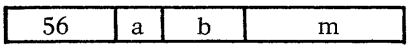
Case 2. Assume optional arithmetic busy set and a 48-bit operation in which group 3 is a generate, groups 4 through 7 are passes and group 8 is a pass. See diagram below. These conditions would generate a carry input for a 24-bit adder but not for the 48-bit adder.

First consider case 2 for the 24-bit adder. Optional arithmetic busy is clear therefore U812 = 1 forcing U501 and U502 to 0. Groups 4 through 7 being passes drives U500 to 0 and U600 goes to 1. L403 = 1 since group 3 is a generate; therefore L403 and U600 drives U700 to a 0 which breaks the input to L200. L200 goes to 1 which says carry input to group 0. If performing a 48-bit add, optional arithmetic busy is set, therefore U812 = 0 allowing U501 and U502 respectively to sense for pass groups 8-11 and pass groups 12-15. For case 2 U501 would sense group 8 pass and go to 1 which drives U600 to 0 and group 0 would not sense a carry input for this case.



DOUBLE-PRECISION MULTIPLY

MUAQ multiply AQ



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description

Multiply (AQ) by the 48-bit operand in addresses M and M + 1. The 96-bit product is displayed in AQE. Refer to figure 300 for operand formats. For every 0 bit in the multiplier the evecution time is decreased by 125 nsec.

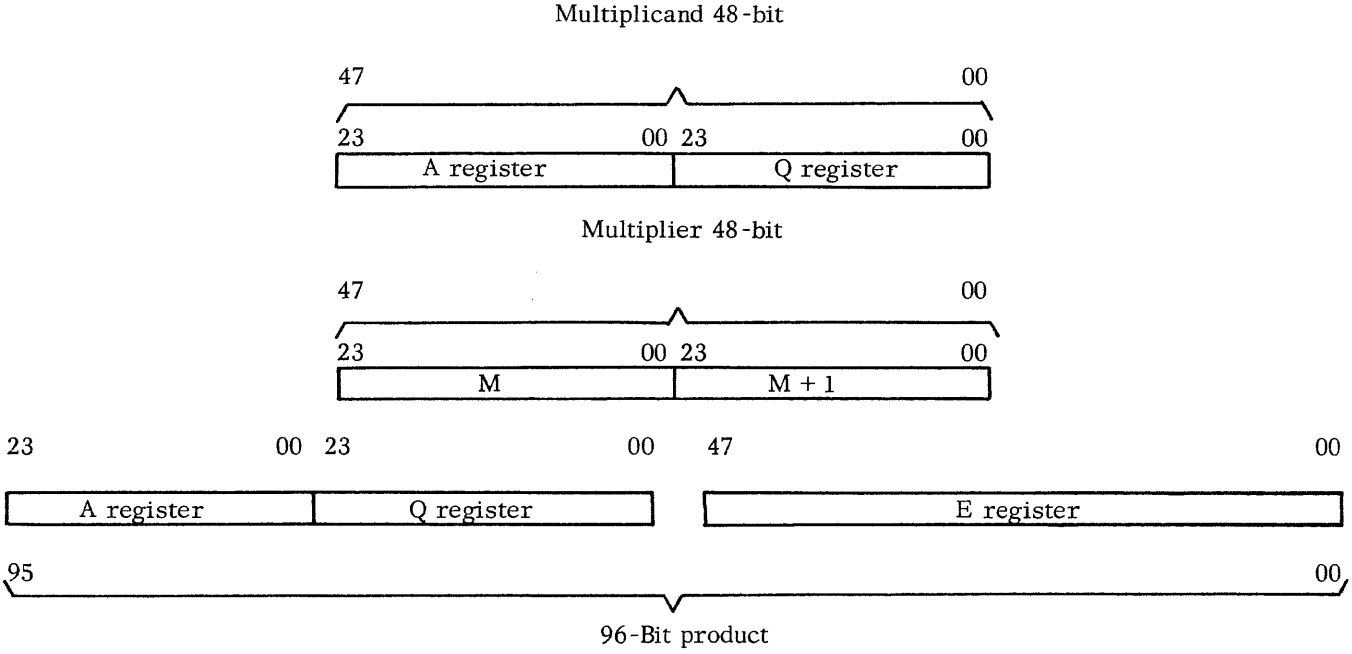


Figure 300. Operand Format for Double-Precision Multiply

SIX-BIT EXAMPLE FOR MULTIPLY

In order to realize the steps that the computer must take to execute a multiply instruction assume you have a 6-bit machine and attempt a double-precision multiply with pencil and paper. The following conditions exist:

(A) = 40 (Q) = 25 multiplicand
(M) = 35 (M + 1) = 05 multiplier

With pencil and paper, position the operand thus:

4025_8 multiplicand
3505_8 multiplier

The multiplicand is a negative number and must be complemented before performing a multiply. If the multiplier had been negative it would have been necessary to complement it as well. The computer must perform the multiply with positive operands. The computer would have performed the same steps. The problem has been initialized and appears as:

(-)3752_8 multiplicand
(+)3505_8 multiplier

Perform the multiply step.

$$
\begin{array}{rl}
 & (-)3752_8 \text{ multiplicand} \\
\times & (+)3505_8 \text{ multiplier} \\
 & 23622 \\
 & 23622 \\
 & 13676 \\
\hline
(-) & 16304022
\end{array}
$$

The product shown is correct in actual value, but since it is negative it must be complemented to be represented in the computer. The third and final step of a double-precision multiply becomes complement. The final result would be:

6 1 4 7 3 7 5 5

Note that a three-step operation was necessary.

1. Initialize - gives positive form of multiplier and multiplicand properly positioned for multiply.
2. Multiply step - performs actual multiplication and always yields a positive product.
3. Complement - necessary any time the product is to be negative.

DETAILED TIMING

Figure 301 is a detailed flow chart of double-precision multiply. In order to accomplish the three major steps of double-precision multiply, several steps are involved and some operations take place in parallel. Figure 302 is a graphic representation of parallel timing chains involved.

DETAILED TIMING FOR INITIALIZE

Detailed timing starts at V007 time of RNI.

V007
V008: EXX2 to F1.
V009: If arithmetic busy, set K008/009.
V010: Test interrupts, clear K002/003.
V011: Input to H080.

First ROP

V080: Set K084/085 (ROP); set K112/113 (no index); clear K080/081 (RNI).
V109: Input to H110.
V110: Set K000/001 (request buss); clear K112/113.
N051: If K210 = 1, set K010/011, gate F to the s bus, T655 off.
N050: Input to H117.
V117: Set K212/213 (priority 2); set K116/117 (storage request); transmit a storage request.
N050: Input to H115.
V115: Set K012/013.
V116: Test breakpoint stop if BPO selected.
N207: Set K572/573 (wait function).
N206: Set K570/571 (initiate F^1 to F^2).
V109: If arithmetic not busy, input to H518.
V518: F_1 to F_2, clear K570/571.
V519: Clear K552/553 (sign of A)
V520: Clear K572/573; clear SCR; set K552/553 if (A) negative.
V061: Resynced storage reply, set K060/061.
V000: Clear K000/001 (request bus)
V001:
V002: Set K100/101 (start arithmetic 1).
V003; V109: Input to H100.
V004; V100: Clear K110/111 (main control priority); (enable I^0 to F).
V005: Set K062/063 (2nd cycle).
V006: Input to H401, input to H201.
V007: Clear K010/011, K012/012, K116/117; clear DBR, clear F_{L15}, input to H410; input to H110.
V008; V110: EXX2 to DBR, input to H449.

Second ROP

Set K000/001, set K104/105 (start arithmetic 2).

V009; V105; N051: Input to H084, input to H102; if K210 = 1, set K010/011; gate F to S bus, T655 off.

At V008 time (M) is gated to DBR, which places the highest order bits of the multiplier at an entrance point to the arithmetic section. Also, at V008 (V110) time the new address, M + 1, is gated to F lower 15 so that at V009 time the proper address for the 2nd ROP is on the S bus.

FADR

Set K536/537, set K596/597.
N687, N691, N693, N659: start arithmetic.
V500: Clear X_1 and A_2.
V502: static enables: $\overline{+1}$ to I^6
V503: $\overline{F_L}$ to I^5

V504: Clear K596/597;
set K850/851 if (M) negative;
set K850/851 (DBR→I^4 enable) if (M) negative.
V505: N687, N691, N693: start arithmetic;
if sign of A ≠ M, set K532/533;
input to H104.

The arithmetic timing shown above forms the address M + 1 and records whether or not numbers with unlike signs are being multiplied.

V106; V104; N050: (V102) Clear K110/111, clear K104/105; input to H117, block V084.

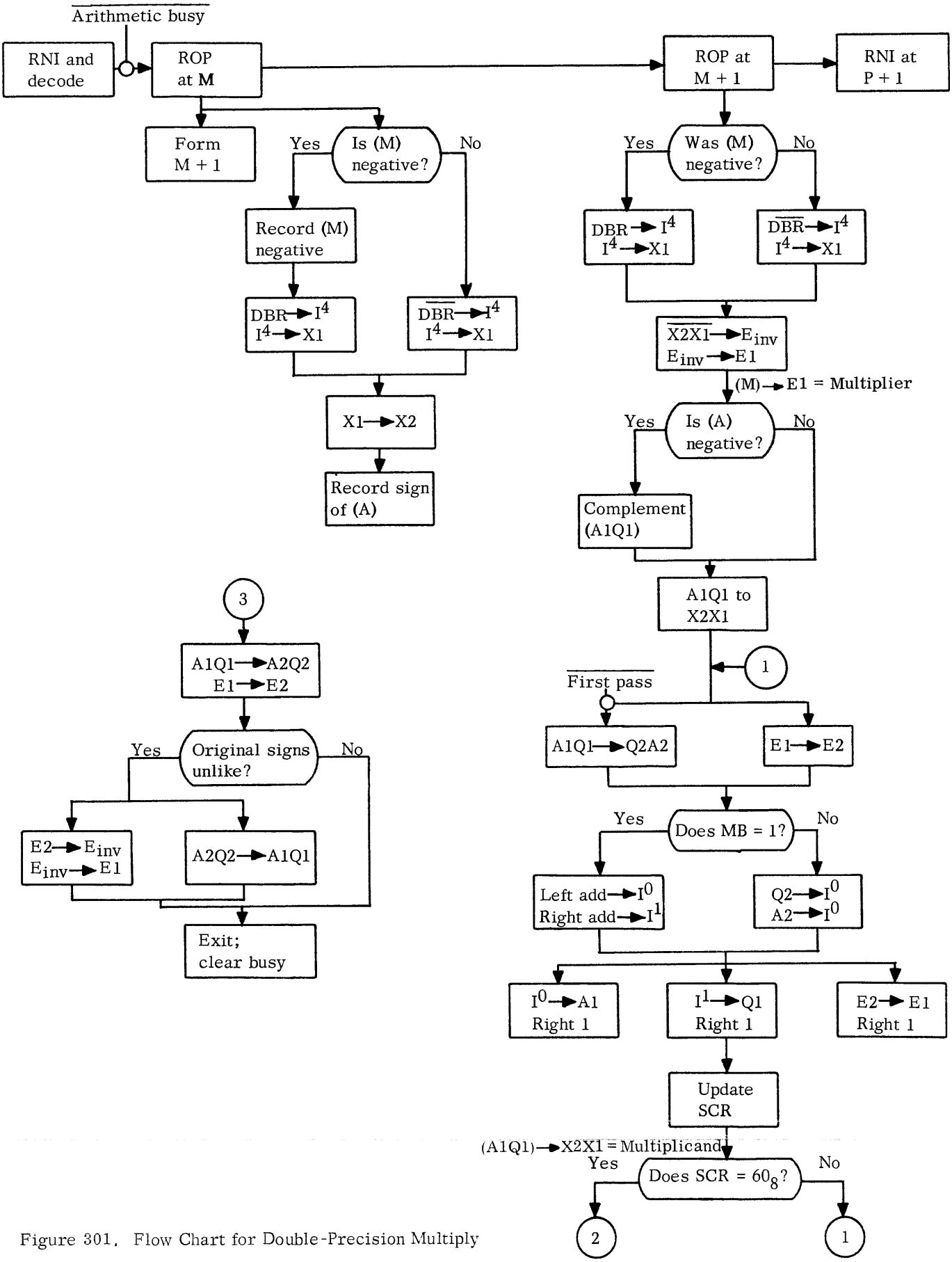


Figure 301. Flow Chart for Double-Precision Multiply

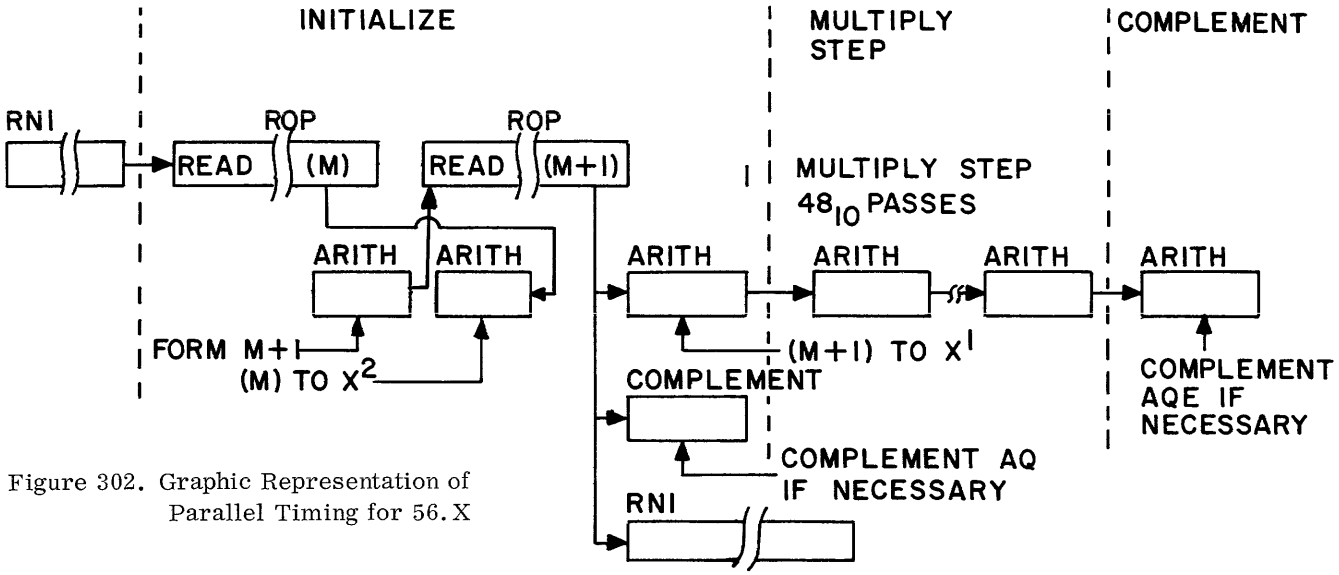


Figure 302. Graphic Representation of Parallel Timing for 56.X

V117: Set K212/213 (priority 2); set K116/117 (storage request) transmit a storage request.
N050: Input to H115.
V115: Set K012/013 (enable data bus).
V116: Test breakpoint stop if BPO selected.

At the first N051 time above, storage is requested for (M+1). Notice that at V116 time a breakpoint operand is possible.

V500: Clear X_1 and A_2; clear K536/537.

V501: I^4 to X_1: M positive, $\overline{DBR}$ to I^4; M negative, $\overline{DBR}$ to I^4; input to H854.
V502: Clear X_2, input to H855.
V503: X_1 to X_2, set K800/801 (wait word 2).
V504:
V505:

The arithmetic timing shown above places the positive value of M in X^2.

V061: Resynced Storage Reply.
V000: Clear K000/001 (request buss).
V001:
V002:
V003: Clear K060/061 (1st cycle).
V004:
V005: Clear K062/063 (2nd cycle).
V006: Input to H401.
V007: Clear K019/011, K012/013, K116/117; clear DBR, input to H410.
V008: EXX2 to DBR, set K104/105 (start arithmetic 2).
V009; V109: Input to H104 and H084; the arithmetic section receives a start pulse that is coincident with V009.
V084: Clear K084/085, clear K104/105, set K080/081 (RNI is to be next); input to H087.
V087: Input to H014.
V014: RNI at P + 1.

Access is timed out for the 2nd ROP and at V008 (M+1), which is the lower order bits of the multiplier, is gated to DBR. The arithmetic start which occurs at V009 time will be used to complete the initialization phase. At V087 time V014 is entered to initiate RNI. If the next instruction requires the use of the arithmetic section its execution will be held up until the completion of the double precision multiply.

N687; N691; N693: Input to H500, H510 and H512; start arithmetic coincident with V009 time of the 2nd ROP cycle; if a negative set K852/853 (Complement AQE).
V500: Clear X_1 and A_2Q_2; input to H611. If K852/853 is set, input H531, set K560/561 arithmetic busy.
V501: I^4 to X_1: If K850/851 set, DBR to I^4; if K850/851 clear, DBR to I^4.
V502: Clear K850/851 (DBR to I^4 enable); set K802/803 (word 2).
V503: Input H834; input to H612 if K852/853 is set.
V504: Set K804/805 (busy), clear E_1; input to H915.
V505: Einv to E_1 ($\overline{X_2X_1}$ to Einv.); (M and M+1) → X2X1 → E1 = multiplier).
V611: I^2 to A_2 (A_1 to I^2); I^3 to Q_2 (Q_1 to I^3).
V612: Clear A_1Q_1; input to H613.
V613: I^0 to A_1 (A_2 to I^0); I^1 to Q_1 (Q_2 to I^1); first pass only: input to H810.

The arithmetic timing is used to place the multiplier in its positive form in E^1. Refer to figure 303 for a simplified data flow diagram. At V501 time of the

The arithmetic timing is used to place the multiplier in its positive form in E1 (figure 303). At V501 time of first pass of multiply step the positive form of the multiplicand will be placed in X2 X1 and initialization is completed (figure 305). At V500 time K560/561 (arith busy) is set, which will exclude main control from the arithmetic section until execution of this instruction is completed.

The above timing involves the complement and swap timing chain. This timing would occur only if it were necessary to complement the multiplicand in A1Q1 (figure 304).

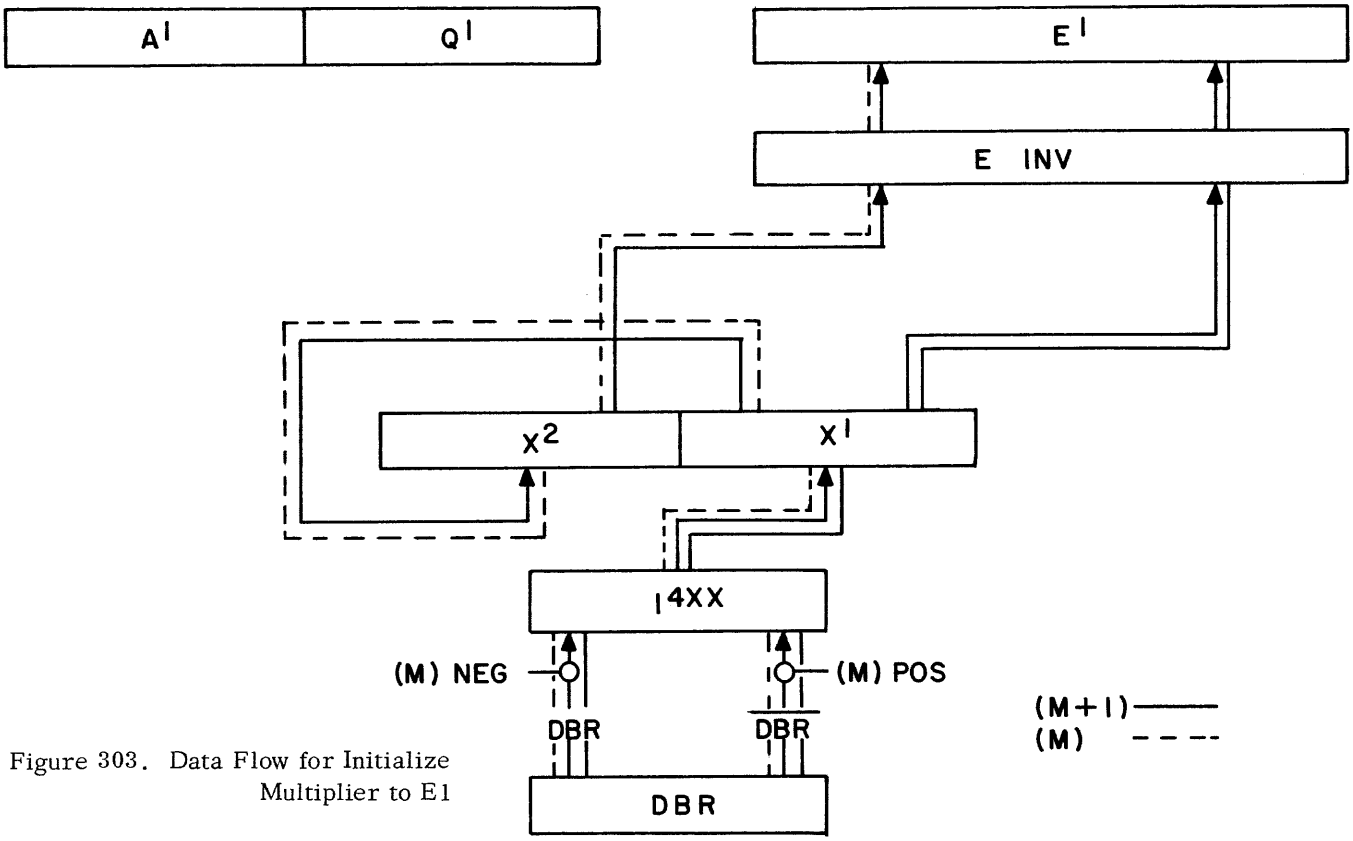


Figure 303. Data Flow for Initialize Multiplier to E1

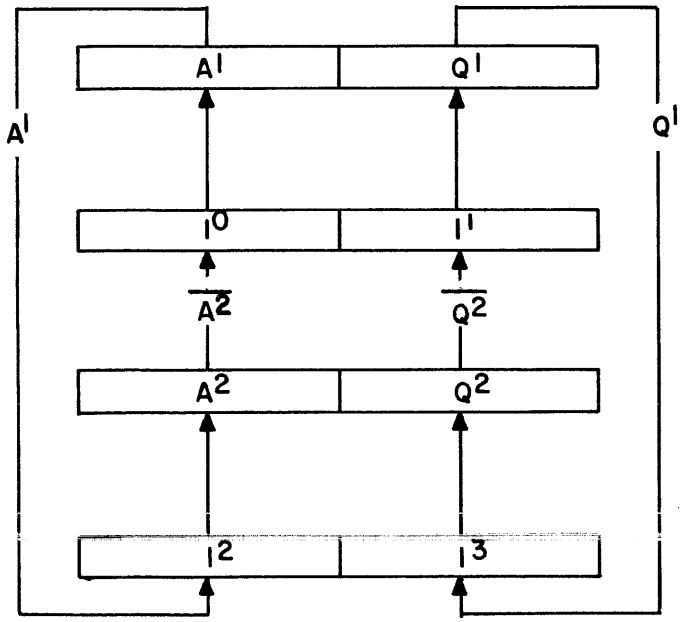


Figure 304. AQ Negative (Positive Form of Multiplicand to A1Q1)

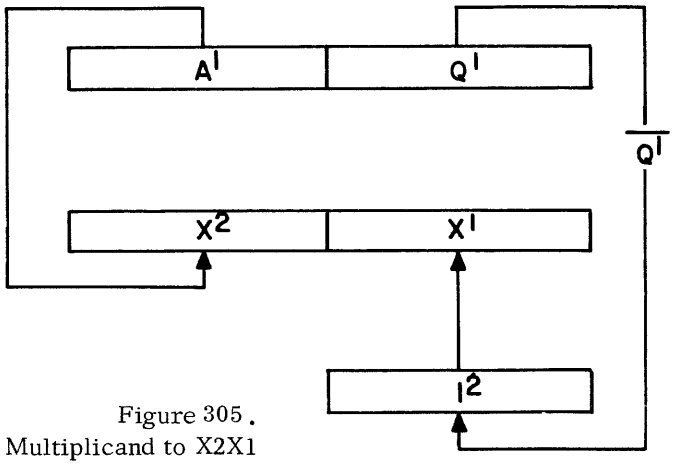
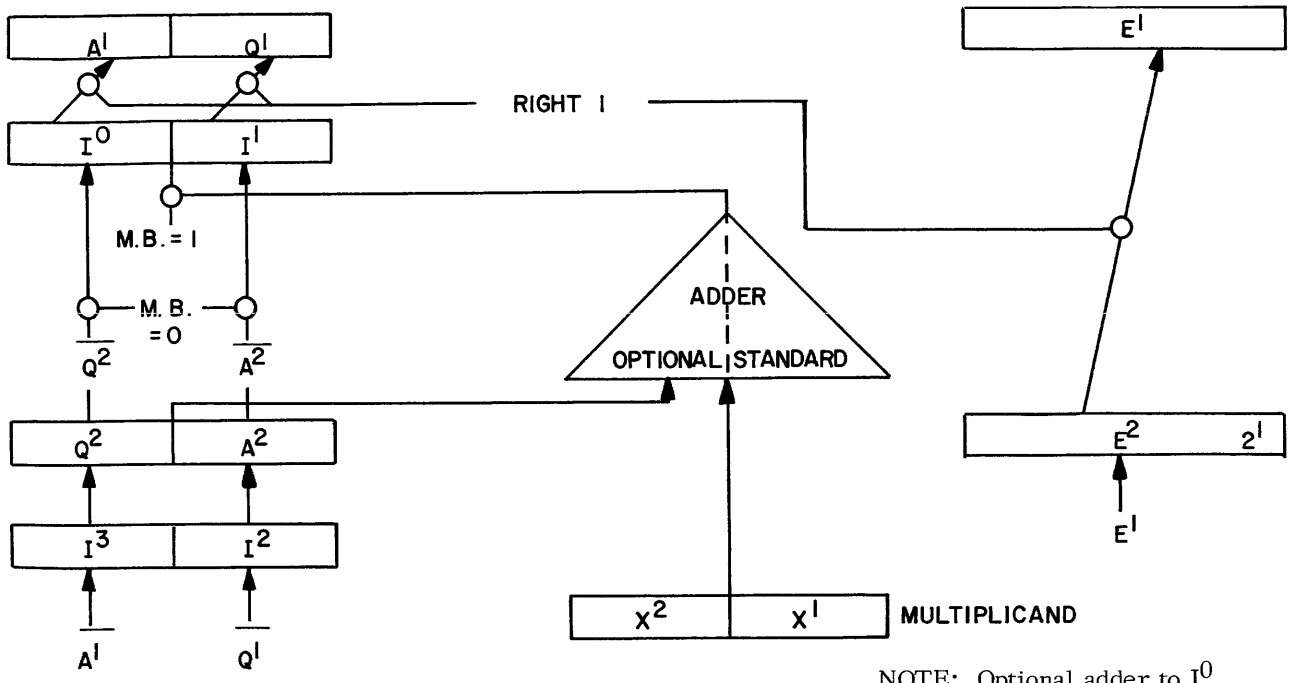


Figure 305. Multiplicand to X2X1

Figure 305 shows the enables and transfer paths for multiply step. Become familiar with this before going through the detailed timing of multiply step.



NOTE: Optional adder to I^0
Standard adder to I^1
X2 and Q2 to optional adder
X1 and A2 to standard adder

Figure 306. Enables for Multiply Step

MULTIPLY STEP

V500: Clear E2, input to H871;
clear A2Q2; input to H611 (blocked first pass).
*Clear X1 and X2, input to H811;
clear K800/801 (wait word 2).
*A1 to X2, I^2 to X1 (Q1 to I^2).

V501: E1 to E2;
short I^2 to A2 (Q1 to I^2) blocked first pass;
cycle I^3 to Q2 (A1 to I^3) blocked first pass.

V502

V503: Clear K898/899 (short cycle), advance SCR;
input to H612 and H895.
Clear K802/803 (word 2);
set K810/811 (multiply 2).

V504: Clear A1Q1E1: transfer SCR;
input to H525, H565, and H895;
set K898/899 (short cycle) if bit 1 of E2 = 1.

V505: E2 to E1 RS1, I^0 to A1 RS1, I^1 to Q1 RS1;
if MB = 0, Q2 to I^0, A2 to I^1;
if MB = 1, left adder to I^0, right adder to I^1;
input to H500, H620, and H820;
if SCR $\neq 60_8$, repeat multiply step;
if SCR = 60_8, set K852/853 (complement AQE);
clear K898/899 (short cycle).

*First pass only.

Refer to arithmetic chapter for a detailed analysis of a computer multiply. The main items in multiply step are:

1. Long and short cycle capabilities. A long cycle consists of 6 Ø times and is necessarily long to allow for adder propagation. The short cycle consists of 4 Ø times and is used when the multiplier bit is a 0. The time for a 56 instruction is dependent on the multiplier (M, M + 1).

2. Shift count register. 48_{10} or 60_8 passes must be made during multiply step. It is the responsibility of the SC register to keep count of the passes. A count of 60_8 indicates terminate.

3. A long cycle will always occur on the first pass. On passes 1-47, E201 will be tested for short cycle on the next pass. MB for the present pass will always be E200. This determines the enables into I^0I^1; e.g., $\overline{Q2A2}$ or $\overline{sum}$.

4. On the first pass A2Q2 is cleared and A1Q1 is not gates to Q2A2. This insures that if the first multiplier bit were a 1 that the multiplicand would be added to all zeros.

COMPLEMENT STEP

V500: Clear A2Q2E2, clear K806/807 (multiply 1);
clear K810/811 (multiply 2);
input to H611 and H871.

V501: E1 to E2;
I^2 to A2 (A1 to I^2);
I^3 to Q2 (Q1 to I^3).

V502

V503: If signs unlike, input to H612 and H834.

V504, Clear A1Q1E1;
input to H915 and H613 if AQE ≠ 0.

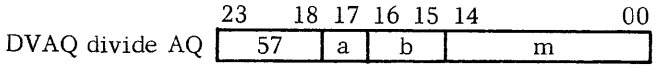
V505: Input to H864.
E_{inv} to E1 (E2 to E_{inv});
I^0 to A1 ($\overline{A2}$ to I^0)
I^1 to Q1 ($\overline{Q2}$ to I^1).

V684: Clear K560/561 (arith busy).
Clear K804/805 (optional arith busy).

The clearing of K560/561 at V684 time frees the arithmetic section for use by main control. The arithmetic timing shown will occur regardless of whether or not a negative product must be formed. The data transfers shown below occur only if the complement is necessary.

E1 to E2 to E_{inv} to E1;
A1 to I^2 to $\overline{A2}$ to I^0 to A1;
Q1 to I^3 to $\overline{Q2}$ to I^1 to Q1.

DOUBLE-PRECISION DIVIDE



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description

Divide (AQE) by the 48-bit operand in addresses M and M + 1. The quotient is displayed in AQ. The remainder with its sign extended is displayed in E.

If a divide fault occurs, program execution advances to the next address. The final contents of AQ and E are meaningless if a divide fault occurs. Refer to figure 307 for operand formats.

SIX-BIT EXAMPLE FOR DIVIDE

In order to visualize the steps that the computer must take to execute a divide instruction assume you have a 6-bit machine and attempt a double-precision divide with pencil and paper. The following conditions exist:

(A) = 61 (Q) = 47 (E) = 3756 Dividend
(M) = 35 (M + 1) = 05 Divisor

With pencil and paper, position the operands thus:

divisor 3505 $\overline{)\,61473756}$ dividend

The dividend is a negative number and must be complemented before performing a division. If the divisor had been negative, it would have been complemented for the pencil and paper example as well. The com-

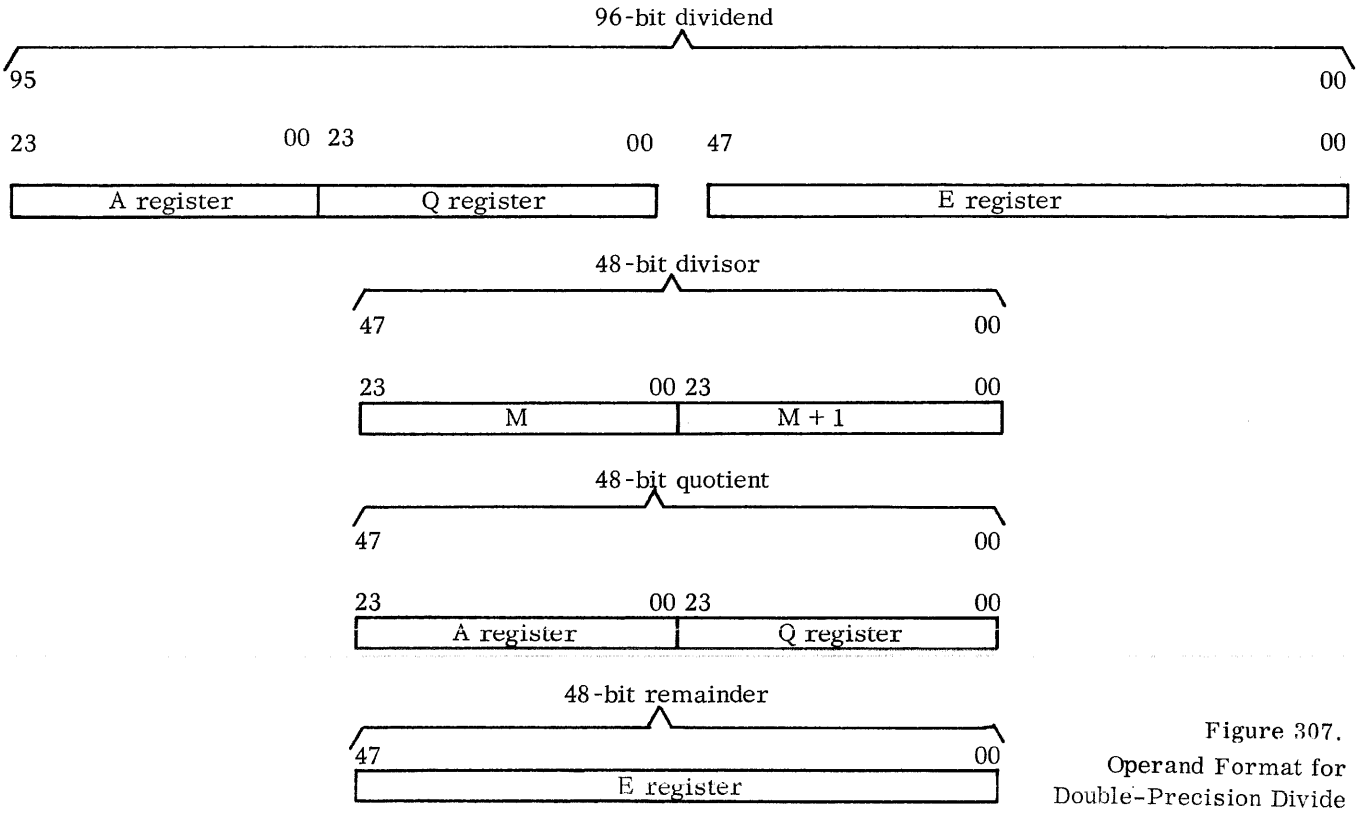


Figure 307.
Operand Format for
Double-Precision Divide

puter divides by a series of subtractions, thus always requires the negative form of the divisor. For this example the computer has to perform the same steps as above plus complementing the divisor. The problem would appear as:

divisor (+)3505)(-)16304021 dividend

Perform the divide step.

```
                    3751   quotient
divisor  +3505 ) (-)16304021  dividend
                  12717
                  33650
                  31343
                  23052
                  22131
                   7211
                   3505
                   3504   remainder
```

The main items in divide step are:

1. When the computer executes the divide step, the quotient is in E and the remainder in AQ. The programmer prefers the quotient be in AQ so hardware performs a step called swap. Swap, which follows divide step, places the quotient in AQ and the remainder in E. After swap we have:

   (AQ) = 3751 quotient
   (E) = 3504 remainder

2. The computer always performs the divide step with positive dividend. At the termination of divide step the positive value of the operands is available in AQ and E. The law of signs for division states: Only when dividing unlike signs is the quotient negative; the remainder always has the same sign as the dividend. In order to form a negative answer, the computer must perform a complement step. After complement step the final result is:

   (AQ) = 4026 quotient
   (E) = 4273 remainder

For this example note that four-step operation was necessary.

1. Initialize - yields the positive form of the dividend and divisor properly positioned for the divide step.
2. Divide step - performs the actual division and always yields a positive quotient and a positive remainder.
3. Swap - places the positive form of the quotient in AQ and the positive value of the remainder in E.
4. Complement - forms a negative quotient and/or negative remainder.

DETAILED TIMING

Figure 308 is a detailed flow chart of the double-precision divide. In order to accomplish the four major steps of the double-precision divide, several operations are involved and some take place in parallel. Figure 309 shows parallel timing and the individual timing chains involved.

DETAILED TIMING FOR INITIALIZE

Detailed timing starts at V007 time of RNI.

V007:
V008: EXX2 to F_1.
V009: If arith busy, set K008/009 (sense interrupt during arithmetic busy).
V010: Test interrupts, clr K002/003 (RNI lockout).
V011: Input to H080
V080: Set K084/085 (ROP), set K112/113 (no index) clear K080/081 (RNI).
V109: Input to H110.
V110: Set K000/001, (request bus) clear K112/113
N051: If K210 = "1", set K010/011, (main control priority) gate F to the "S" bus, T655 OFF (read request)
N050: Input to H117
N117: Set K212/213 (priority 2)
Set K116/117, transmit a storage request (T650).
N050: Input to H115.
V115: Set K012/013 (enable data bus).
V116: Test breakpoint stop if BPO selected.
N207: Set K572/573 (wait function).
N206: Set K570/571 (initiate F1 to F2).
V109: If arith not busy, input to H518.
V518: F1 to F2, clear K570/571.
V519: Clear K552/553 (sign of A).
V520: Clear K572/573.
Clear SCR
Set K552/553 if (A) negative
V061: Resynced storage reply, set K060/061 (1st cycle).
V000: Clear K000/001 (request bus).
V001:
V002: Set K100/101.
V003, V109: Input to H100.
V004, V100: Clear K100/101, set K110/111.
V005: Set K062/063 (second cycle).
V006: Input to H401, input to H201.
V007: Clear K010/011, K012/013, K116/117, clear DBR, clear FL_{15}, input to H410, input to H110.
V008, V110: EXX2 to DBR, input to H271 2nd ROP set K000/001, (request bus) set K104/105
V009, V105, N051: Input to H084, input to H102 if K210 = "1", set K010/011. Gate F to "S" bus, T655 off (read) input H104.

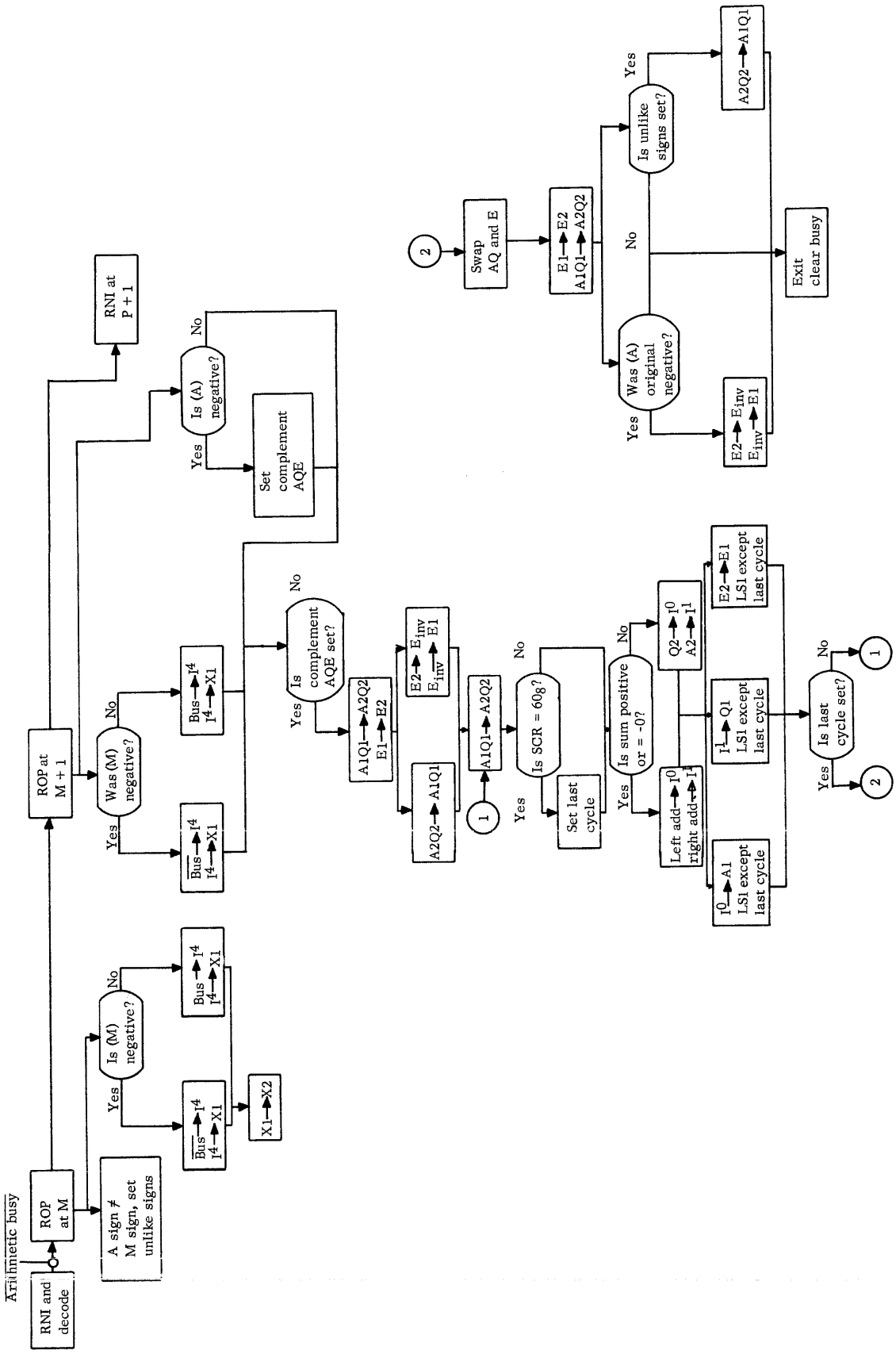


Figure 308. Flow Chart for Double-Precision Divide

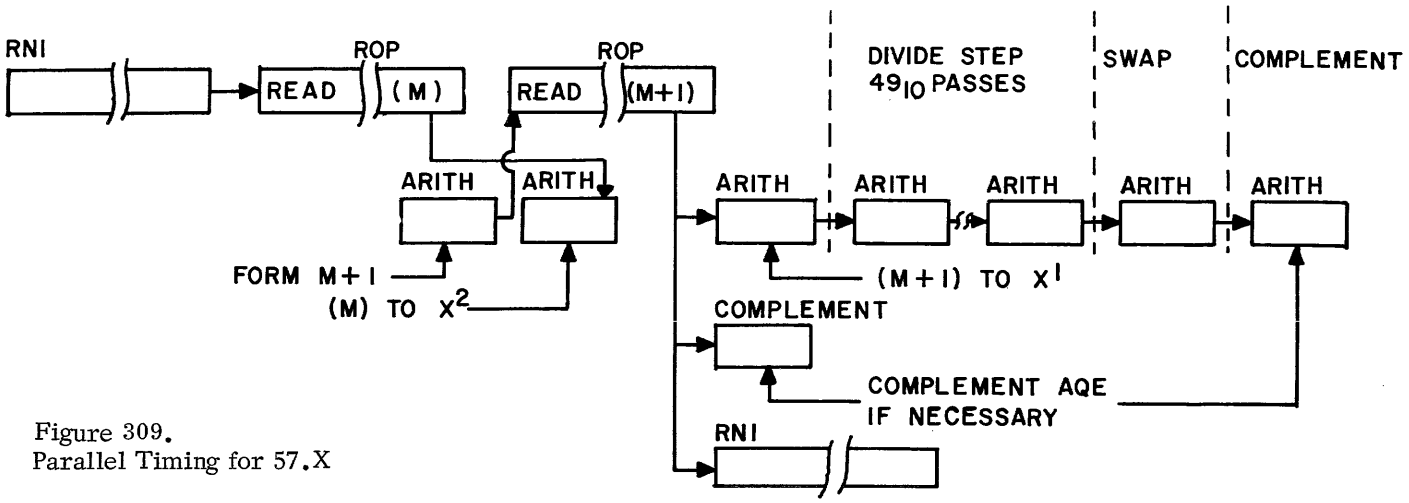


Figure 309.
Parallel Timing for 57.X

FADR

Set K536/537, set K596/597.
N687, N691, N693: Start arith.
V500: Clear X_1 and A_2.
V501: I^5 to A_2, I^6 to X_1
V502: Static enables: $\overline{+1}$ to I^6.
V503: $\overline{FL}$ to I^5.
V504: Clear K596/597, if (M) positive set K850/851
V505: N687, N691, N693: Start arithmetic, if sign of A = M, set K532/533.

The arithmetic timing shown above forms the address M + 1 and records whether or not operands with unlike signs are being divided.

At V008 time (M) is gated to DBR, which places the highest order bits of the dividend at an entrance point to the arithmetic section. Also at V008/V110 time the new address, M + 1, is gated to F lower 15 so that at V009 time the proper address for the second ROP is on the S bus. The setting of K060/061 and K062/063 (first and second cycle FF's) allows J063 (logic diagrams 2-5) to go to "1". J063 being "1" blocks V084 (end ROP) and allows ROP to ROP cycle progression by preventing the clearing of the ROP FF.

V106, V104, N050: Clear K110/111, clear K104/105.
(V102) Input to H117, block V084.
V117: Set K212/213 (priority 2).
Set K116/117, transmit a storage request (T650).
N050: Input to H115.
V115: Set K012/013 (enable data bus).
V116: Test breakpoint stop if BPO selected.

At the first N051 time shown above, storage is requested for (M + 1). Notice that at V116 time a breakpoint operand is possible.

V500: Clear X_1 and A_2.
Clear K536/537.
V501: I^4 to X_1: (M) negative, $\overline{DBR}$ to I^4
(M) positive, DBR to I^4
Input to H584.
V502: Clear X_2, input to H355.
V503: X_1 to X_2, set K800/801 (wait word 2).
V504:
V505:

The arithmetic timing shown above places (M), the highest order bits of the divisor, in X_2.

V061: Resynced storage reply.
V000: Clear K000/001 (request bus).
V001:
V002:
V003: Clear K060/061 (first cycle).
V004:
V005: Clear K062/063 (second cycle).
V006: Input to H401.
V007: Clear K010/011, K012/013, (enable data bus) K116/117 (STO request).
Clear DBR, input to H410.
V008: EXX2 to DBR, set K104/105.
V009: V109: Input to H104 and H084.

The arithmetic section receives a start pulse that is coincident with V009. The timing continues on the next page.

V084: Clear K084/085, clear K104/105, set K080/081, input to H087.
V087: Input to H014.
V014: RNI at P + 1.

Access is timed out for the second ROP and at V008 time (M + 1), which is the lower order bits of the divisor, is gated to DBR. The arithmetic start which

occurs at V009 time will be used to complete the initialization phase. At V087 time V014 is entered to initiate RNI. If the next instruction requires the use of the arithmetic section its execution will be held up until the completion of the double precision divide.

N659, N687, N691, N693: Start arithmetic coincident with V009 time of the second ROP. If (A) negative set K852/853.
Input to H500, H510, H512, H820.

V500: Clear A_2 Q_2 E_2 and X_1.
Input to H871, set K560/561 (arithmetic busy)

V501: I^4 to X_1: If K850/851 set, DBR to I^4.
If K850/851 clear, $\overline{DBR}$ to I^4.
E_1 to E_2.

V502: Clear K850/851 (bus to I^4 enable).
Set K802/803 (word 2)

V503: Input to H612 if K852/853 (comp AQE FF) is set.
Input to H834 if K852/853 is set.

V554: Set K804/805 (busy).

V505: Set K808/809 (divide I).
Input to H500, H620, H820.
Clear K852/853 (compl AQE).

Input H611 if K852/853 (comp AQE FF) is set.

V611: I^2 to A_2 (A_1 to I^2).
I^3 to Q_2 (Q_1 to I^3).

Clear A_1, Q_1, E_1.
Input to H612 and H915 if K853 is set.
I^0 to A_1 ($\overline{A_2}$ to I^0).
I^1 to Q_1 ($\overline{Q_2}$ to I^1).
Einv to E^1 (E^2 to einv).

The arithmetic timing shown above completes the initialization phase by placing (M + 1) in X_1. Refer to figure 310 for the data flow of the divisor into X_1X_2. At V500 time K560/561 (arithmetic busy) is set, which insures that main control cannot get into the arithmetic section until the execution of this instruction is complete.

The above timing involves the complement and swap timing chain. This timing would occur only if it were necessary to complement the dividend which is in $A_1Q_1E_1$. Refer to figure 311 for a simplified data flow diagram.

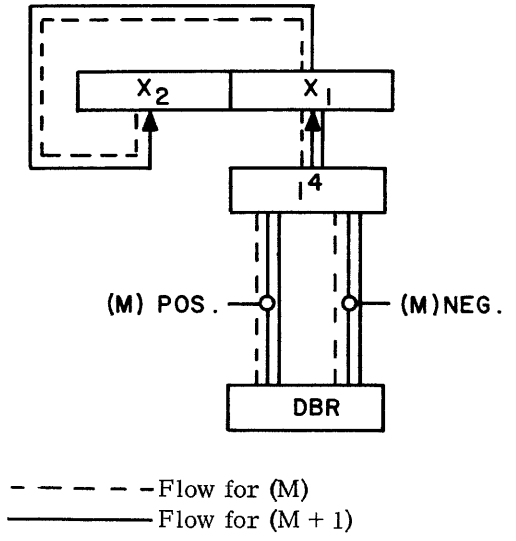


Figure 310. Divisor to X2X1

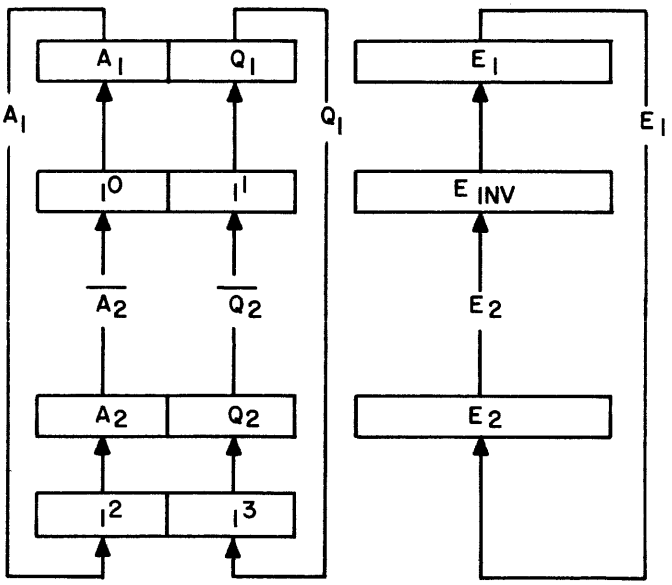


Figure 311. Complement AQE

DIVIDE STEP - Remainder in AQ; Quotient in E1

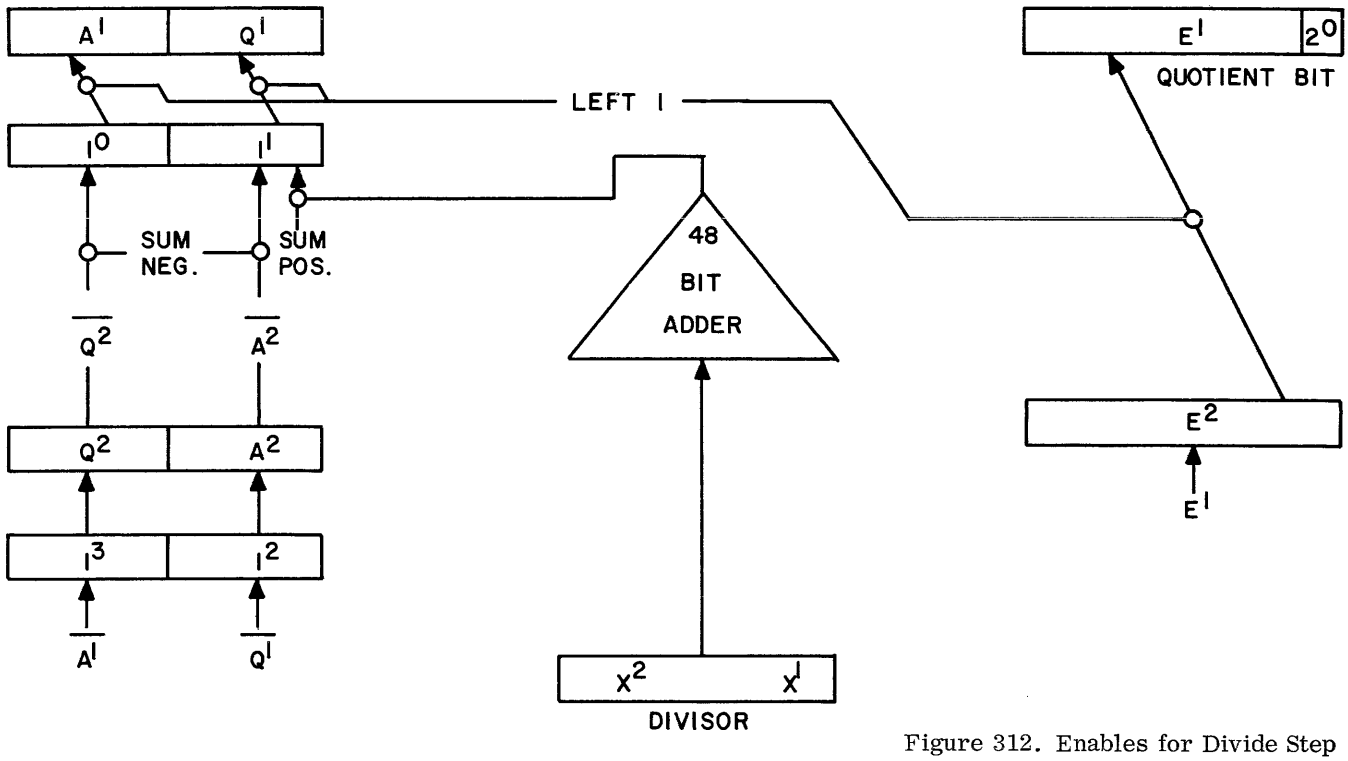


Figure 312. Enables for Divide Step

Figure 312 shows the enables and transfer path for divide step. Become familiar with this before proceeding through the detailed timing of divide step.

V500: Clear A2, Q2, E2; clear K800/801 (wait word 2); input to H611 and H871.

V501: I^2 to A2 ($\overline{Q1}$ to I^2); I^3 to Q2 ($\overline{A1}$ to I^3); E1 to E2.

V502: If last cycle and E2 bit 46 or 47 is set, set K538/539, (divide fault).

V503: Input to H612 and H834. Clr K802/803 (word 2).

V504: Clear A1, Q1, and E1; input to H905; if $\overline{\text{last cycle}}$ input H525 and H565.

V505: E2 to E1 LSI; I^0 to A2 LSI except last cycle, I^1 to Q1 LSI except last cycle. If sum positive: QB = 1, O881 to E400; left adder to I^0, right adder to I^1. If sum negative: QB = 0, O881 to E400; $\overline{Q2}$ to I^0, $\overline{A2}$ to I^1. Input to H500, H620 and H820; if $\overline{\text{last cycle}}$, repeat divide step; if last cycle, input to H810 and proceed to swap.

Note: Optional Adder to I^0
Standard Adder to I^1
X2 and Q2 to Optional Adder
X1 and A2 to Standard Adder

If SCR = 60_8, set K812/813 (last cycle).

Advance SC register.
Transfer SC register.
If last cycle (sum $\neq$ -0), input to H613; set K844/845 (swap AQE 1).

For a step-by-step analysis of a computer divide, refer to the arithmetic chapter of this text. The main items for divide step are:

1. Shift count register - Arithmetic passes 49_{10} or 61_8 are necessary for divide step. The quotient is positioned correctly on the sixty-first pass. The SC register keeps count of the passes. A count of 61_8 says terminate.
2. Quotient bit - The divide step consists primarily of a series of subtractions and left shifts. With each subtraction the result is sensed; if it is positive the quotient bit is a 1. Figure 313 shows how the quotient bit is set during divide step. Translations of the inputs to O881 and examples of the cases represented are shown. During divide step a result of -0 must be treated as a positive result.

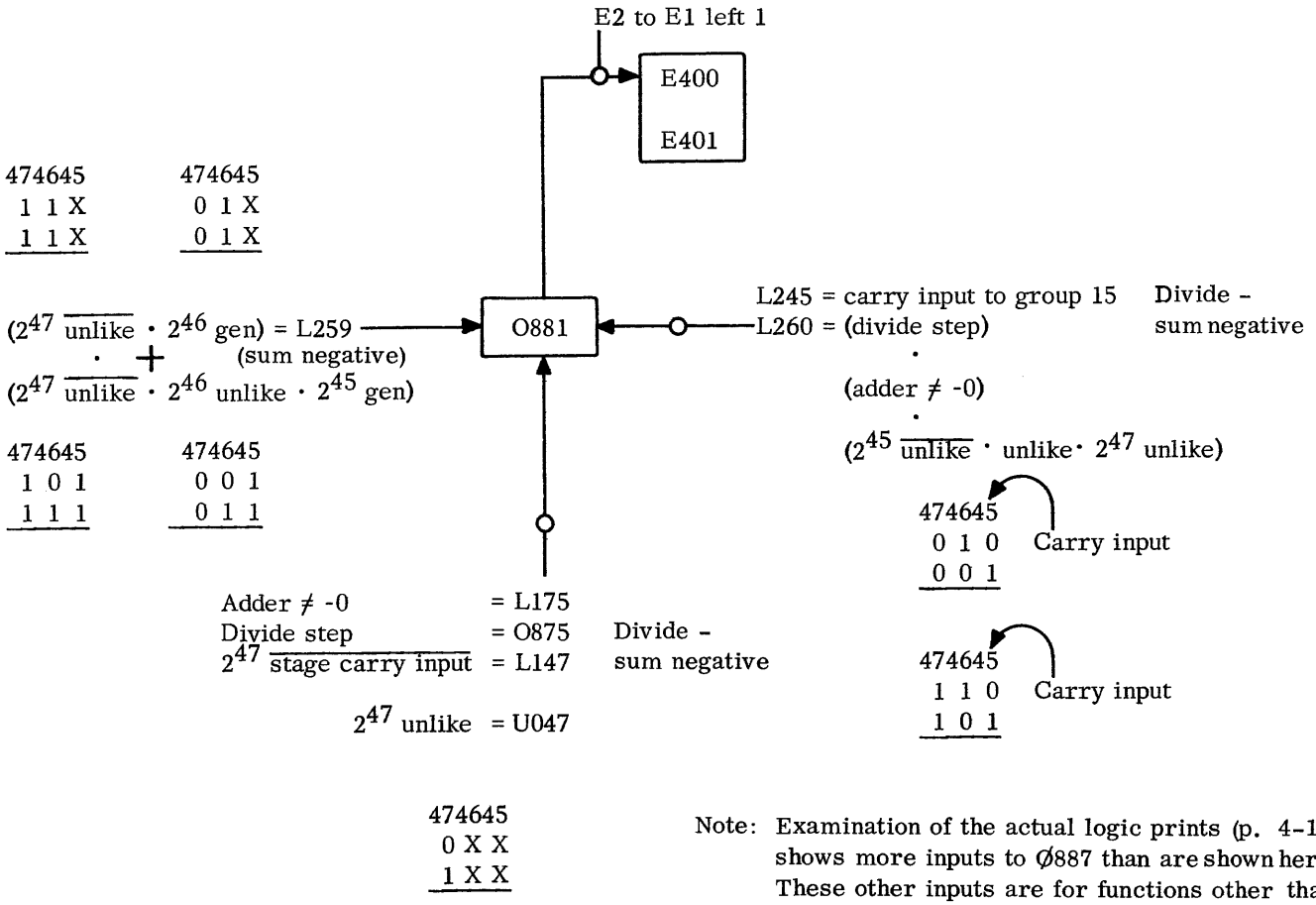


Figure 313. Quotient Bit Translations

Note: Examination of the actual logic prints (p. 4-11) shows more inputs to Ø887 than are shown here. These other inputs are for functions other than D. P. divide and as such are of no consequence during this discussion.

3. Divide fault - Occurs when the upper 48 bits of the dividend left shifted 1 is greated than or equal to the divisor. Two 6 bit examples of divide fault are shown below.

In double-precision divide the quotient is being formed in E register. During the last cycle $E2_{47}$ or $E2_{46}$ being set will cause the divide fault FF to be set. (Logic Diagrams 2-103).

```
  100
01)0100  this quotient could not be expressed in a 6-bit register.

  40
10)0040  this quotient could not be expressed in a 6-bit register.
```

SWAP

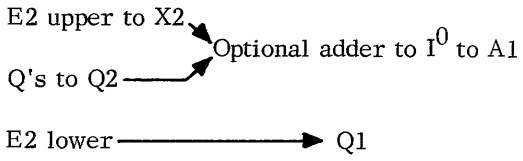
V500: Clear A2, Q2, E2, X2, and X1; clear K812/813 (last cycle).
V501: E1 to E2.
V502: Set K846/847 (swap AQE 2); input to H811.
V503: A1 to X2; I^2 to X1 ($\overline{Q1}$ to I^2); input to H834.
V504: Clear E1; input to H915.
V505: E_{inv} to E1 ($\overline{X2X1}$ to E_{inv}); clear K844/845 (swap AQE 1); input to H500, H620, and H810.

The arithmetic timing shown above accomplishes the following:

1. The quotient in E1 is transferred to E2. E2 serves as a holding register for the quotient while the remainder is transferred to E1.
2. The remainder in AQ at the end of divide step is transferred to E1 during this pass. The transfer is: A1 Q1 to $\overline{X2X1}$ to E_{inv} to E1

V500: Clear A2, Q2, X2, and X1; input to H821.
V501: Set K548/549 (swap AQE 3); E2 upper to X2.
V502: Clear K846/847 (swap AQE 2).
V503: Input to H814 and H844.
V504: Clear A1 and Q1; input to H515.
V505: I^0 to A1 (left adder to I^0); E2 lower to Q1; set K852/853 (complement AQE); input to H500, H620, and H820.

At V500 time the quotient is in E2. This pass will be used to place the quotient in A1 Q1. The data transfers are:



COMPLEMENT

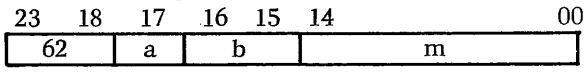
V500: Clear A2 Q2 and E2; clear K848/849 (swap AQE 3); input to H611 and H871.
V501: E1 to E2; I^2 to A2 (A1 to I^2); I^3 to Q2 (Q1 to I^3)

V502

	Complement quotient	Complement remainder
V503: Input to H864.	Input to H611 if (unlike signs) (AQ ≠ 0).	Input to H834 if A1 original was negative and E 1 ≠ 0.
V504: Clear K804/805 (optional arith busy).	Clear A1 Q1; input to H613.	Clear E1; input to H915.
V505: Input to H864.	I^0 to A1 ($\overline{A2}$ to I^0); I^1 to Q1 ($\overline{Q2}$ to I^1).	E_{inv} to E1 (E2 to E_{inv}).
V864: Clear K560/561 (arith busy)		
At V504 time the clearing of optional arith busy enable the adder to function as a 24-bit adder. Clearing arith busy allows main control to use the arithmetic section again.	The timing shown above is used only if it is necessary to express a negative quotient.	The timing shown above is used only if it is necessary to express a negative remainder.

FLOATING-POINT MULTIPLY

FMU FP multiplication of AQ



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description

Multiply the 48-bit floating-point operand in AQ by the floating-point operand located at storage addresses M and M + 1. The rounded and normalized product is displayed in AQ.

Bits 12-48 of E hold the lower 36 bits of the 72-bit unnormalized product. The sign of this residue is the same as that of the product. See operand formats in figure 314 .

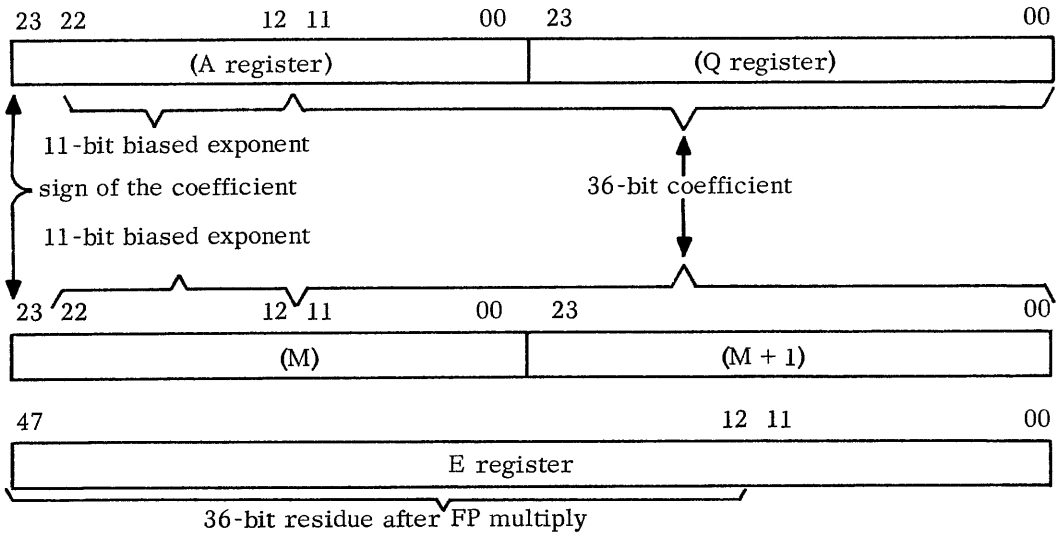


Figure 314. Operand Format for Floating-Point Multiply

The following is a 24-bit example of how floating-point multiplication could be performed by pencil and paper method. You should become familiar with this example before proceeding through the detailed timing of the 62 instruction. The original octal numbers are in the upper right-hand corner of the page. The heavy black lines define the flow path for the problem. The computer would start with the operands packed in floating-point format. The steps from this point are:

1. Obtain positive form of operands.
2. Toggle bit 22 to obtain real exponents.
3. Form the sum of the 11-bit exponents.
4. Place the sum of the exponents in a holding register.
5. Multiply the coefficients.
6. Round the product by adding +1.
7. Normalize the product.
8. Adjust the exponent.
9. Merge the exponent and coefficient.
10. Toggle the bias bit to form biased exponent.
11. Complement if a negative product is to be formed.

In the lower left-hand corner of the page the original octal numbers are multiplied and the product is packed into floating-point format. The answer, 5767,3207, is the same one the computer would obtain.

DETAILED TIMING

Phase I

N687: Start arithmetic coincident with V003 time of the first ROP; set K536/537 (FADR 1); set K596/597 (FADR 2); input to H500 and H510.

V500: Clear A2 and X1.

V501: $I^5 \rightarrow$A2, ($\overline{F_{L15}} \rightarrow I^5$); $I^6 \rightarrow$X1, ($\overline{+1} \rightarrow 16$); input to H942.

V502: Clear Q3; input to H943.

V503: A1 exponent $\rightarrow$ Q3; clear F_{L15}; set K870/871 (add exponent 1).

V504: Clear K596/597 (FADR 2); set K104/105 (start arith 2); set K850/851 if (M) is negative.

V505: $I^0 \rightarrow F_{L15}$

At V501 time the enables come up to form M + 1. At V505 time M + 1 is gated to F_{L15} to be available for the second ROP. At V503, A1 exponent is placed in the holding register Q3. During the transfer the bias is toggled to place the real exponent in Q3 if A1 is positive or the complement of the real exponent in Q3 if A1 is negative.

Phase 2: First Arithmetic Pass

N687: Start arithmetic coincident with V009 time of the first ROP and V505 time of phase 1; set K532/533 (unlike signs 1) if sign of A $\neq$ sign of M; input to H500, H510, and H512.

V500: Clear A2, Q2, X1; input to H513 and H531.

V501: $I^4 \rightarrow$X1 (DB register $\rightarrow I^4$ if (M) negative); (DB register $\rightarrow \overline{I^4}$ if (M) positive); $I^3 \rightarrow$Q2 ($\overline{A1} \rightarrow I^3$) bits 0-11 and bit 23; 186X/87X Q2 bits 12-22 (Q3 $\rightarrow$ 186X/87X if (Q3) negative); (Q3 $\rightarrow$ 186X/87X if (Q3) positive); input to H854.

V502: Clear X2; input to H821 and H855.

V503: X1 $\rightarrow$ X2 bits 0-11 and bit 23; $\overline{I81X/82X} \rightarrow$ X2 bits 12-22 ($\overline{X1} \rightarrow$ I81X/82X); set K800/801 (wait word 2); set K872/873 (add exponent 2); input to H834.

V504: Clear E1.

V505: Clear K870/871; input to H500 and H942.

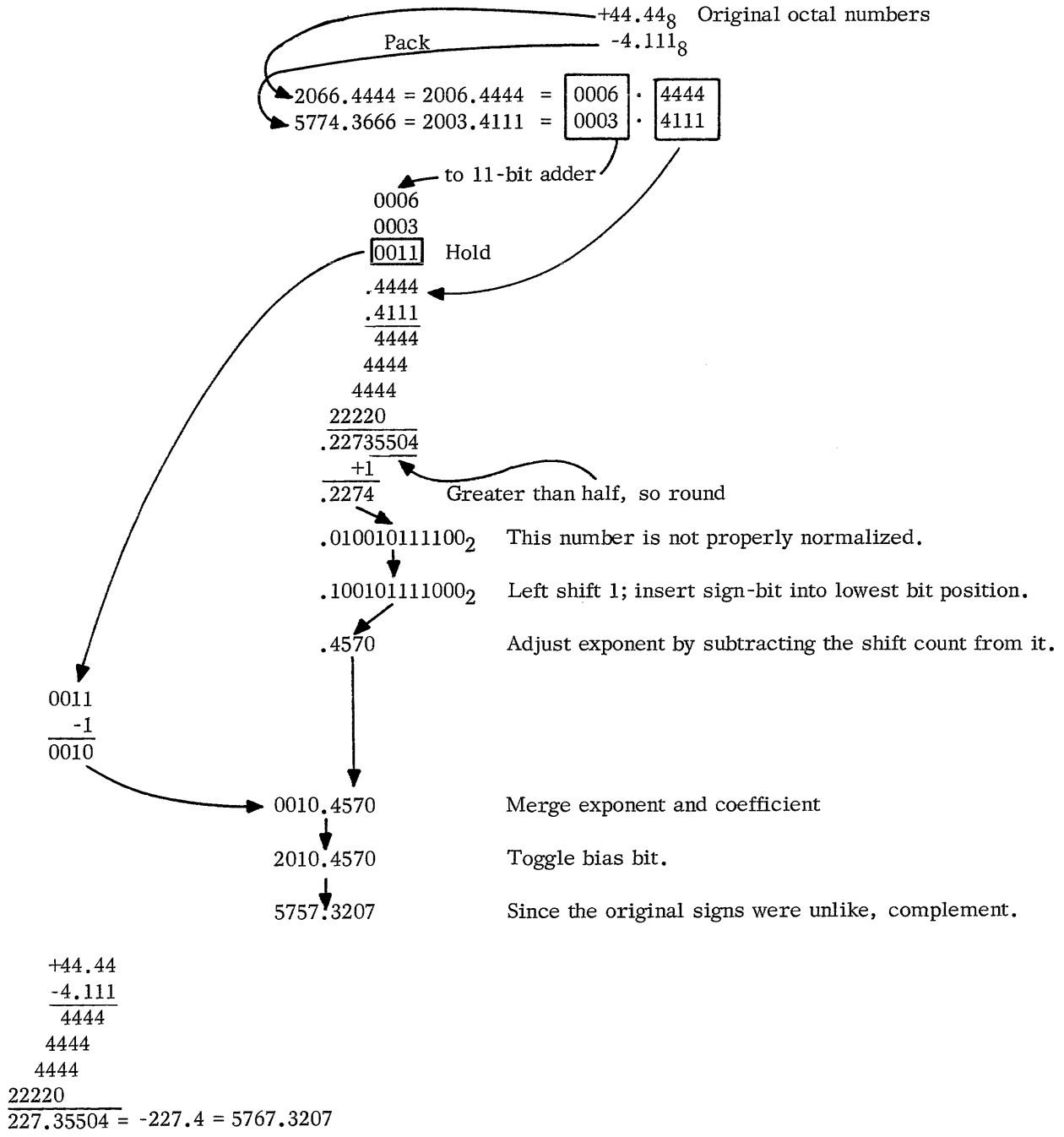
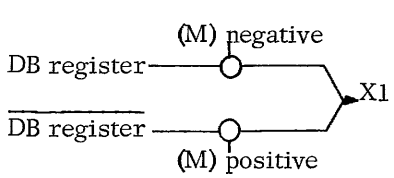
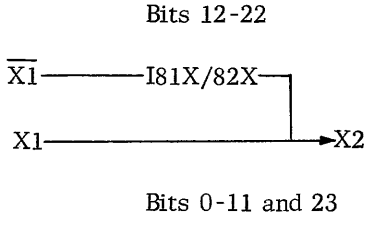


Figure 315. Pencil and Paper Example of Floating Point Multiplication

Phase 2 consists of two arithmetic passes and has three major functions:

1. (M)→X2 - the detailed transfer path is shown.





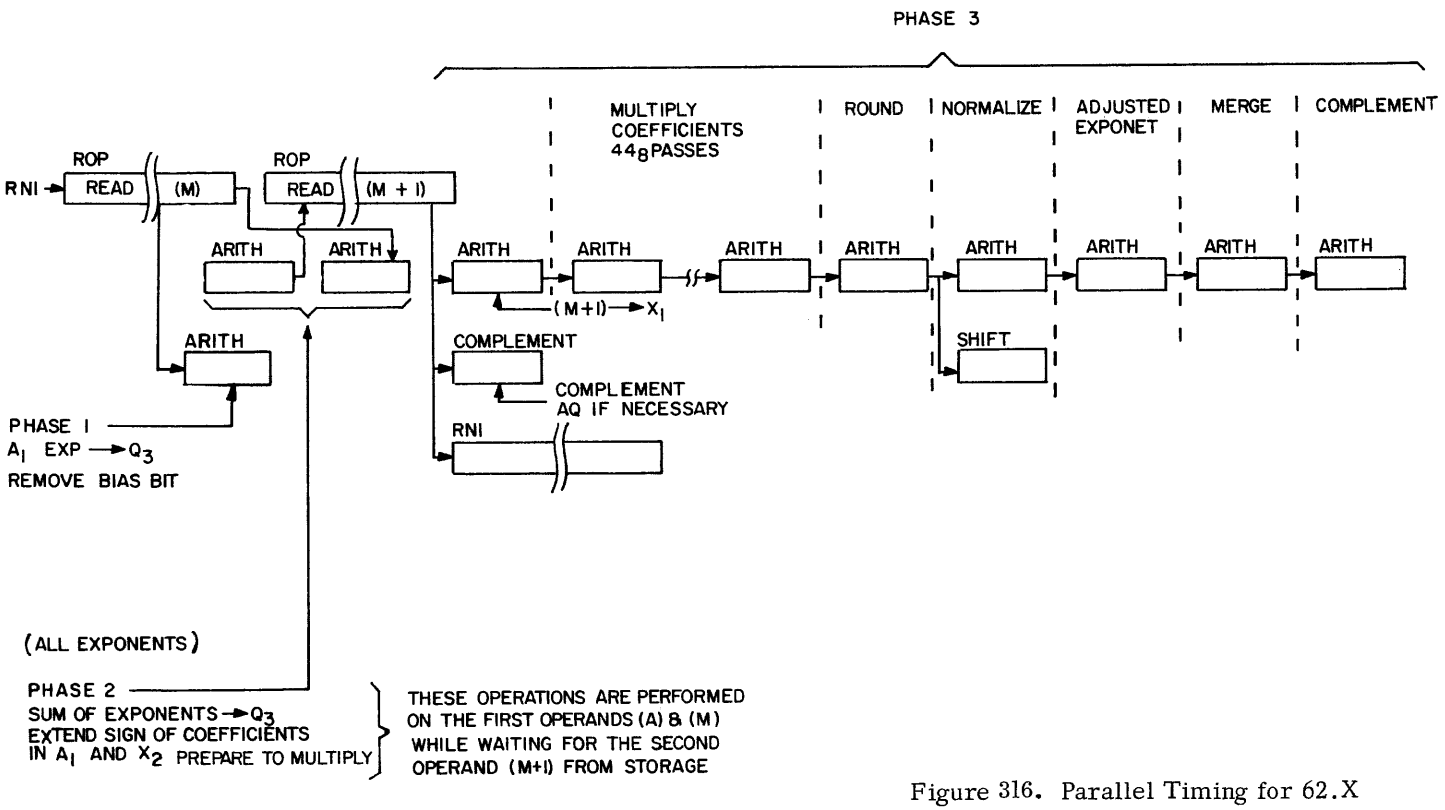


Figure 316. Parallel Timing for 62.X

The transfer of $\overline{X1} \rightarrow$ I81X/82X toggles the bias bit bit so that bits 12-22 of X2 hold the real exponent at V503 time.

2. Add Exponents - The sum of the exponents is formed by forcing the 48-bit adder to appear as an 11-bit adder. This is accomplished by setting K872/873 (add exponent 2) at V503 time which causes groups 0-11 to appear as passes and not generates. Figure 317 is a simplified logic diagram of how group 12 of the adder is affected by the setting of add exponent 2. For the time interval defined by add exponent 2, U800 = 1, and U801 = 0. This insures that all inputs to U746, U747, and U748 are disabled; while all inputs to U749 are partially enabled. Therefore group 12 only sense for carry inputs from groups 12, 13, 14, and 15. Figure 318 shows the data flow for add exponents. The operands are presented to the adder at V503 time of this arithmetic pass and sum will be sampled at V501 time of the next arithmetic pass.

3. Extend sign - The second arithmetic pass of phase 2 will be used to extend the sign of the coefficient for both the multiplier and multiplicand. This sign extension will allow the computer to perform the multiplication of the coefficients in the same manner as the multiply step of double-precision multiply. At V505 time of the first arithmetic pass the

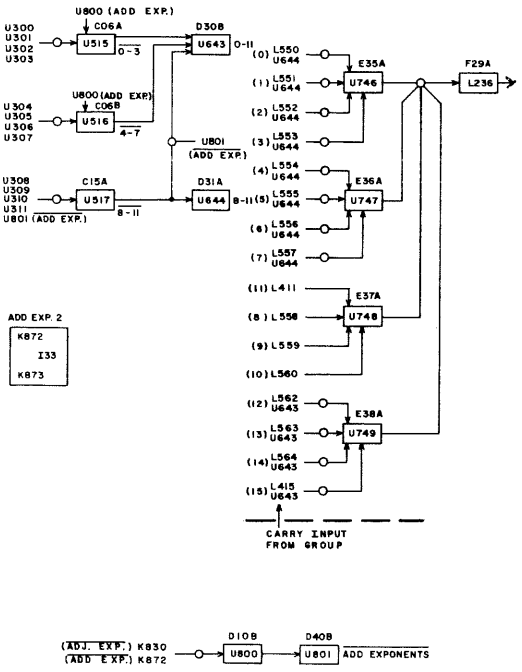
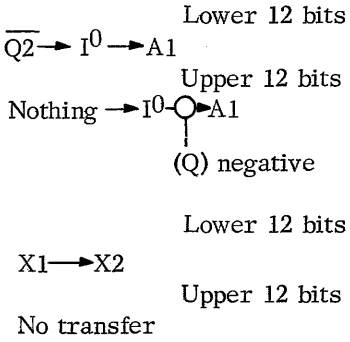


Figure 317. Enabling the 11-Bit Adder

positive form of (M) is in X1 and bits 0-11 and 23 of the original value in A are in Q2. The enables for extend sign are:



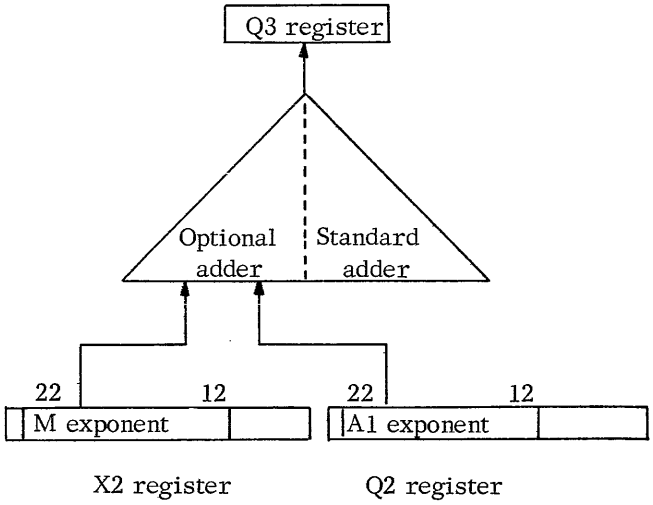


Figure 318. Enables for Add Exponents

Phase 2: Second Arithmetic Pass

V500: Clear Q3; input to H943.

V501: $\overline{Sum}$ to Q3 (set bit 11 of Q3); set K874/875 (extend sign).

V502: Clear K872/873 (add exponent 2).

V503

V504

V505: Input to H844 and H854.

V844, V854: Clear A1, and X2; input to H515 and H855.

V515, V855: $I^0 \rightarrow A1$ ($\overline{Q2} \rightarrow I^0$), lower 12 bits only; if Q2 negative $I^0 \rightarrow A1$ (nothing to I^0), upper 12 bits only; X1 $\rightarrow$ X2, lower 12 bits only. Set K800/801 (wait word 2).

All of the above operations have been performed on (A) and (M) while the CPU is waiting for (M+1) from storage.

Phase 3: Initialize

N687: Start arithmetic - coincident with V009 of second ROP; set K852/853 (complement AQE) if (A1) negative; input to H500, 510, and 512.

V500: Clear A2, Q2, and X2; clear K874/875 (extend sign); input to H611 if (A1) negative.

V501: $I^4 \rightarrow X1$ (DBR $\rightarrow I^4$ if (M) negative; ($\overline{DBR} \rightarrow I^4$ if (M) positive); set K802/803 (word 2) (indicates word 2 has arrived from storage); clear K850/851) DBR $\rightarrow I^4$ enable).

V502

V503: Input to H834 and H612.

V504: Clear E1; set K804/805 (optional arithmetic busy).

This timing occurs only if complement AQ is necessary.

V611: $I^2 \rightarrow A2$ (A1 $\rightarrow I^2$); $I^3 \rightarrow Q2$ (Q1 $\rightarrow I^3$)

V612: Clear A1 and Q1; input to H613.

V505: $E_{inv} \rightarrow E1$ ($\overline{X2X1} \rightarrow E_{inv}$);
clear K852/853 (complement AQE);
set K806/807 (multiply 1); and input to H500, H620, and H820.

The preceding timing is used to place the positive value of (M + 1) in X1 at V501 time. This is the lower 24 bits of the coefficient. Phase 2 placed the upper 12 bits of the coefficient in X2, sign extended. At V505 time X1X2 is transferred to E1 and becomes the multiplier for multiply step. K560/561 is set at V500 time insuring that main control cannot proceed to the arithmetic section until execution of this instruction is complete. At V504 time K804/805 is set, enabling the 48-bit adder.

Phase 3: Multiply Step

V500: Clear A2, Q2, and E2.
Input to H271;
input to H611 (blocked first pass).

V501: $I^2 \rightarrow A2$ ($\overline{Q1} \rightarrow I^2$) blocked first pass;
$I^3 \rightarrow Q2$ ($A1 \rightarrow I^3$) blocked first pass;
$E1 \rightarrow E2$.

V502

V503: Clear K898/899 (short cycle);
advance SC register; input to H834 and H612.

V504: Clear A1, Q1, and E1;
set K898/899 (short cycle) if MB = 0;
input to H525, H565, and H895.

V505: $E2 \rightarrow E1$ right 1;
$I^0 \rightarrow A1$ right 1;
$I^1 \rightarrow Q1$ right 1;

MB = 0	MB = 1
$\overline{Q2} \rightarrow I^0$	left adder $\rightarrow I^0$
$\overline{A2} \rightarrow I^1$	right adder $\rightarrow I^1$;

input to H500, H620, and H820;
if SC register $\neq 44_8$, repeat multiply step;
if SC register = 44_8, set K822/823 (round);
clear K898/899 (short cycle); input to H810.

Refer to chapter 12 for a detailed analysis of a computer multiply. The main items of multiply steps are:

1. Long and short cycle capabilities - A long cycle consists of 6 Ø times and is necessary to allow for adder propagation. The short cycle consists of 4 Ø times and is used when the multiplier bit is a 0. The execution time for the 62 instruction depends on the multiplier (M, M + 1).
2. The multiplier bit determines if each pass will be long or short cycle. If at V504 time $E2_{01}$ is a 0, short cycle FF will be set to allow short cycle on the next pass.
3. Shift count register - During multiply step, 36_{10} or 44_8 passes must be made. SC register keeps count of the passes; a count of 44_8 indicates terminate.
4. On the first pass Q2A2 is cleared and A1Q1 is not gated to Q2A2. This insures that should the first multiplier bit be a 1, the multiplicand would be added to all 0s.

At the completion of multiply step the product of the coefficients would be in AQE positioned as shown:

V613: $I^0 \rightarrow A1$ ($\overline{A2} \rightarrow I^0$);
$I^1 \rightarrow Q1$ ($\overline{Q2} \rightarrow I^1$).

Remember that the computer must work with positive operands during the multiply step. The preceding timing will insure that the coefficient of the floating number in AQ will be in positive form for the multiply step. The positive form of the coefficient from storage is insured during the first and second ROP by transfer of $\overline{DB}$ register to I^4.

The following events are unique to the first pass.
Clear X2X1;
clear K800/801 (wait word 2);
input to H811.
$A1 \rightarrow X2$;
$I^2 \rightarrow X1$ ($\overline{Q1} \rightarrow I^2$). Place the multiplicand in X2X1.
(E1 contains multiplier and A1Q1 are cleared to receive partial products).
Clear K802/803 (word 2);
set K810/811 (multiply 2).

At V501 time above A1Q1 is transferred to X2X1. This places the positive form of the multiplicand in X2X1.

Refer to figure 306 for a graphic representation of the transfers and enables for multiply step.

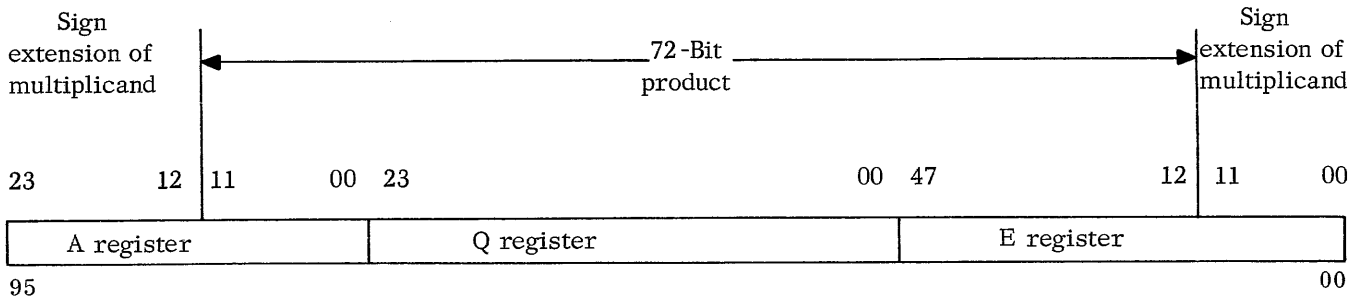


Figure 319. (AQE) After Multiply Step

ROUND

V500: Clear A2, E2, Q2, X1, and X2; clear K806/807 (multiply 1); clear K810/811 (multiply 2); input to H611.

V501: $I^2 \rightarrow$ A2 ($\overline{Q1} \rightarrow I^2$); $I^3 \rightarrow$ Q2 ($\overline{A1} \rightarrow I^3$); input to H944; set $X1_{00}$ if $E1_{47} = 1$. (This is the "1" used for rounding up). Propagate through the adder.

V502: Clear SC register.

V503: Input to H612.

V504: Clear A1 and Q1; set K898/899 if adder $\neq$ -0; input to H613.

V505: Set K826/827 (normalize); remove constant clear input from K880/881 (shift left 10s).

Adder Output From Round

V613: $I^0 \rightarrow$ A1 (left adder $\rightarrow I^0$); $I^1 \rightarrow$ Q1 (right adder $\rightarrow I^1$).

The timing for round occurs whether or not any adjustment of the coefficient is necessary. The only possible adjustment for the 62 instruction is plus 1. Plus 1 is accomplished by forcing X2X1 to 0$\rightarrow$01, placing the coefficient in Q2A2, and performing a 48-bit add. Figure 320 is a block diagram of the enables during round. Figure 321 is a simplified logic diagram of forcing X2X1 to 0$\rightarrow$01. The timing for all floating-point instructions is common for normalize, adjust exponent, merge, and complement. Go to page 244 to complete the timing for this instruction.

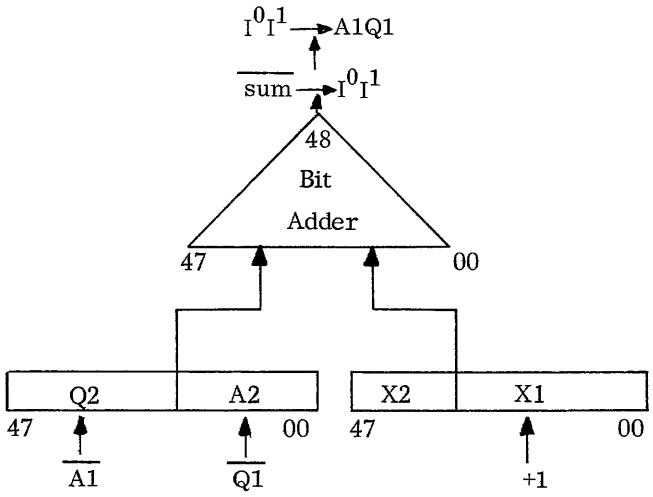


Figure 320. Enables for Round

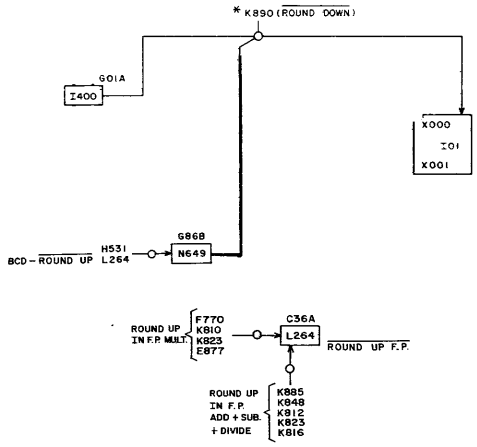
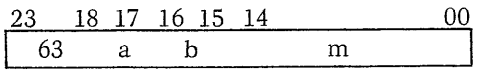


Figure 321. Plus 1 Logic for Round

FLOATING-POINT DIVIDE

FDV FP division of AQ



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description. Divide the floating-point operand in AQ by the 48-bit floating-point operand located at storage addresses M and M+1. The rounded and normalized quotient is displayed in AQ. The remainder with sign extended appears in the F register.

Comments. The sign of the remainder is the same as that of the dividend. Refer to figure 322 for operand formats.

Note: The divisor must be properly normalized or a divide fault will result.

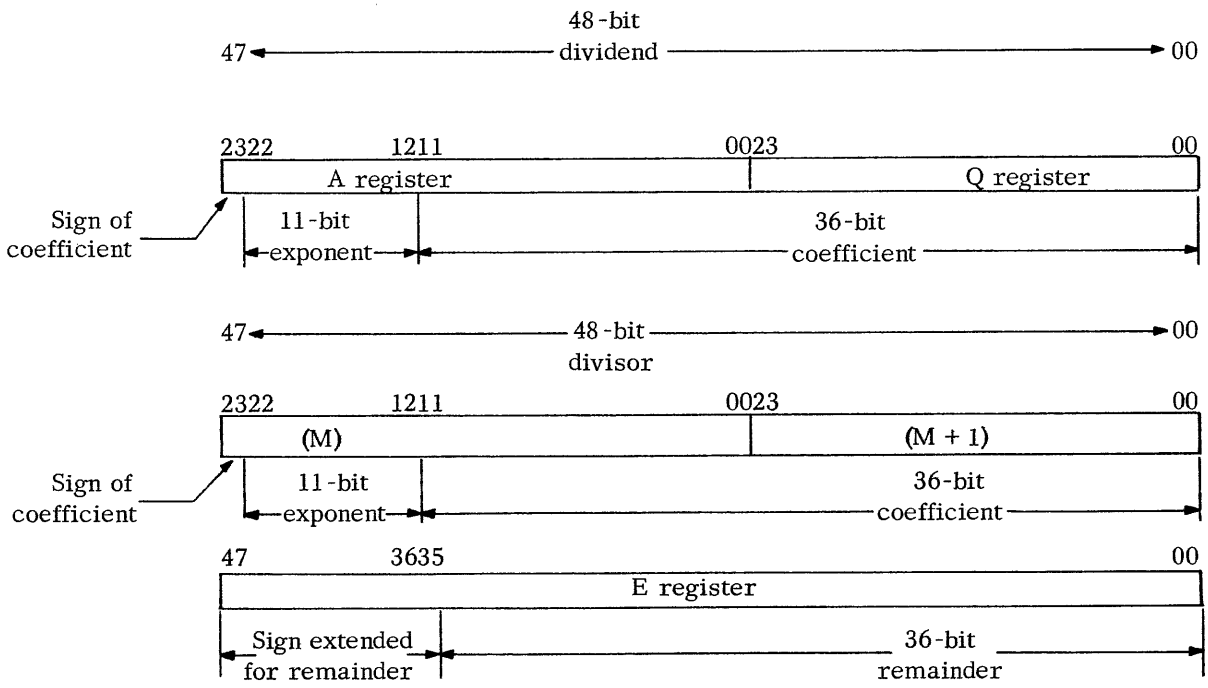
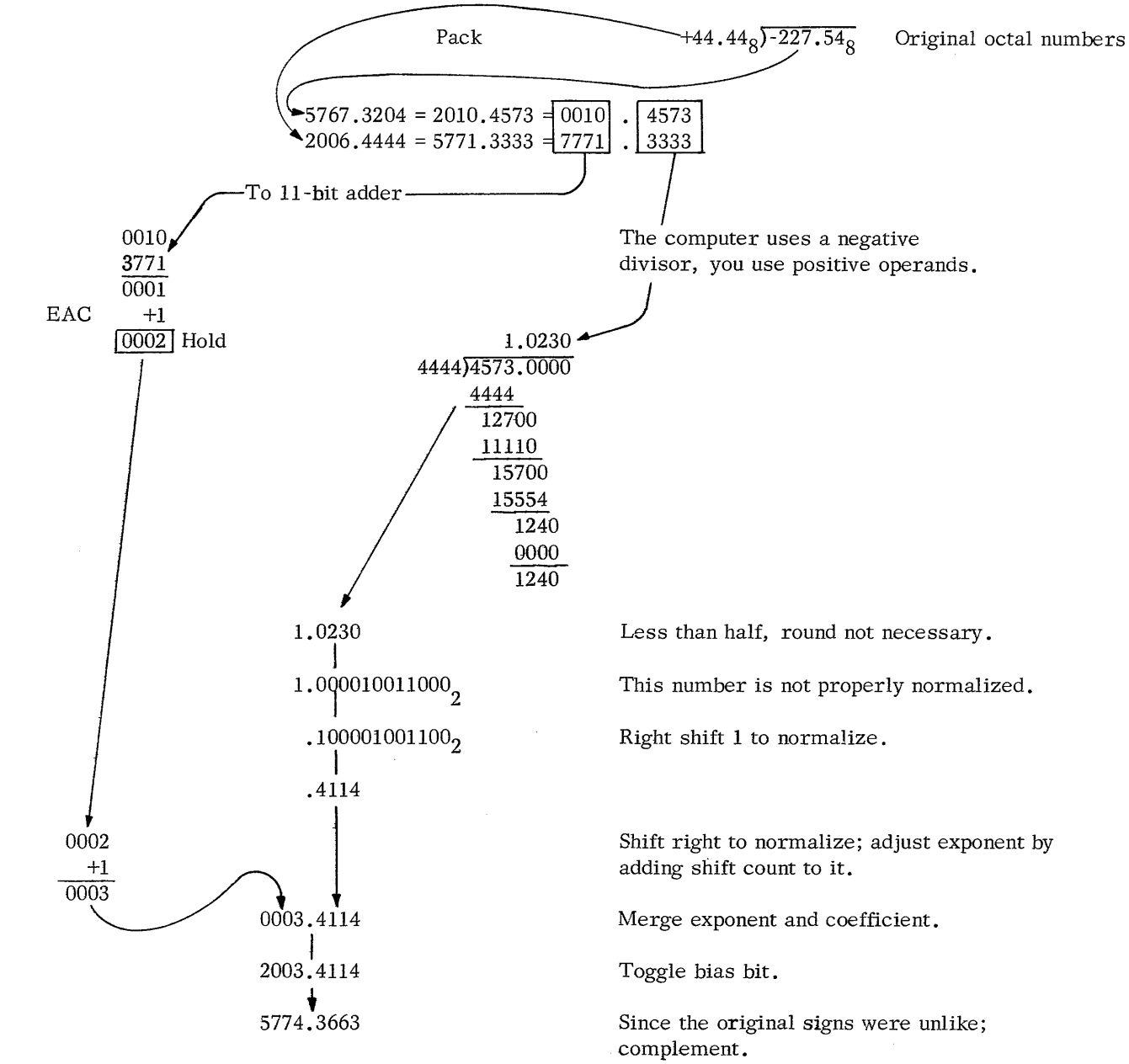


Figure 322. FDV Operand Format

The pencil and paper example on the opposite page shows how a floating-point divide could be performed. The original octal numbers are in the upper right-hand corner of figure 323. The heavy black lines define the flow for the problem. The computer would start with the operands packed in floating-point format. The steps from this point are:

1. Obtain the positive form of the dividend and the negative form of the divisor.
2. Toggle bit 22 to obtain real exponents.
3. Form the difference of the 11-bit exponents.
4. Place the difference of the exponents in a holding register.
5. Divide the coefficients. The computer performs this step using a positive dividend and a negative divisor. Use the positive form of each.
6. Rounding is not necessary.
7. Normalize the quotient.
8. Adjust the exponent.
9. Merge the exponent and coefficient.
10. Toggle the bias bit to form the biased exponent.
11. Complement to form the negative quotient.

In the lower left-hand corner of the page, the original octal numbers are divided and the quotient is packed in floating-point format. Answer 5774.3663 is the same one the computer would obtain. Figure 324 displays the parallel timing for the 63 instruction.




```
          -4.1141 = 4.1141 = 5774.3663
+4444) -22754.0000
        22220
         5340
         4444
          6740
          4444
          22740
          22220
           5200
           4444
            534
```

Figure 323.
Pencil and Paper Example of Floating Point Divide

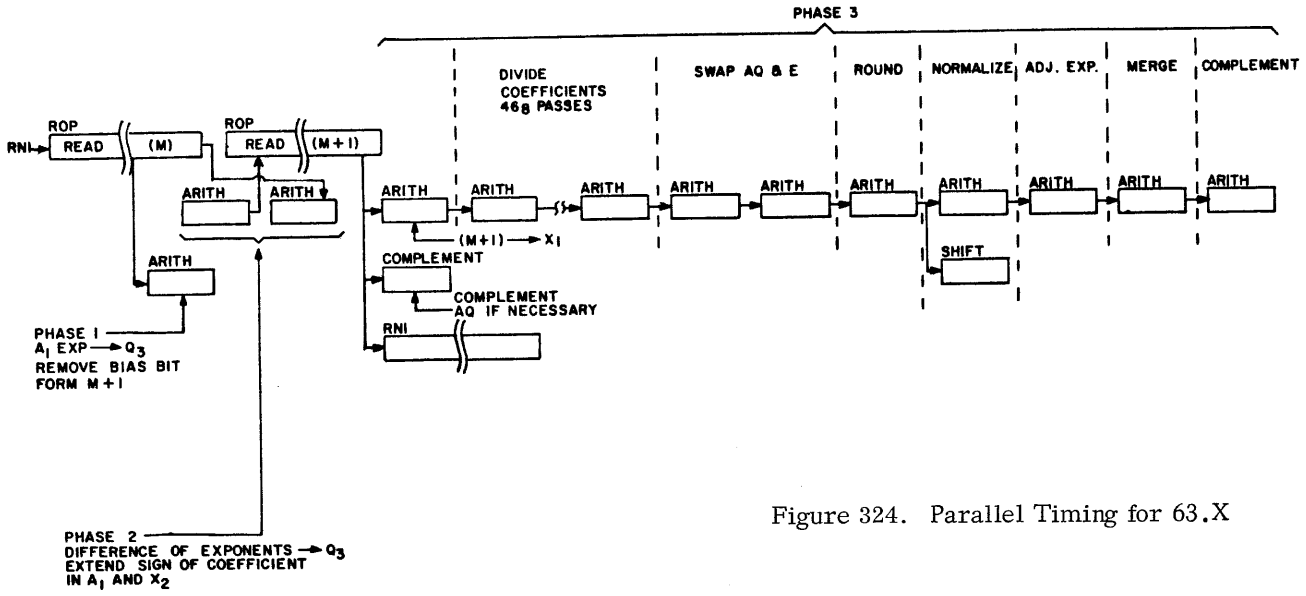


Figure 324. Parallel Timing for 63.X

DETAILED TIMING

Phase 1

N687: Start arithmetic coincident with V003 time of the first ROP;
set K536/537 (FADR 1);
set K596/597 (FADR 2);
input to H500 and H510.

V500: Clear A2 and X1

V501: I^5 to A2, ($\overline{F_{L15}} \rightarrow I^5$);
I^6 to X1, ($+1 \rightarrow I^6$);
input to H942.

V502: Clear Q3; input to H943.

V503: A1 exponent → Q3;
clear F_{L15}; set K870/871 (add exponent 1).

V504: Clear K596/597 (FADR 2);
set K104/105 (start arithmetic 2);
set K850/851 if (M) is positive.

V505: $I^0 \rightarrow F_{L15}$.

At V501 time the enables come up to form M+1. At V505 time M+1 is gated to F lower 15 so that it is available for the second ROP. At V503 time A1 exponent is placed in the holding register Q3. During the transfer, A1 exponent → Q3 the bias bit is toggled, which places the real exponent in Q3 if A1 is positive, or the complement of the real exponent is Q3 if A1 is negative.

Phase 2: First Arithmetic Pass

N687: Start arithmetic coincident with V009 time of the first ROP and V505 time of Phase 1;
set K532/533 (unlike signs 1) if the sign of A ≠ sign of M; input to H500, H510, and H512.

V500: Clear A2, Q2, and X2;
input to H513 and H531.

V501: $I^4 \rightarrow$ X1 ($\overline{\text{DB register}} \rightarrow I^4$ if (M) positive);
(DB register → I^4 if (M) negative);
$I^3 \rightarrow$ Q2 (A1 → I^3) bits 0-11 and bit 23,
I86X/87X → Q2 bits 12-22 (Q3 → I86X/87X if (Q3) negative); ($\overline{Q3}$ → I86X/87X if (Q3) positive); input to H854.

V502: Clear X2; input to H821 and H855.

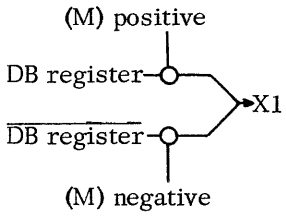
V503: X1 → X2 bits 0-11 and bit 23;
I81X/82X → X2 bits 12-22 ($\overline{X1}$ → I81X/82X);
set K800/801 (wait word 2); set K872/873 (add exponent 2); input to H834.

V504: Clear E1.

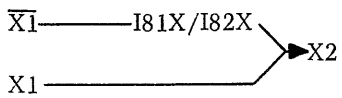
V505: Clear K870/871 (add exp 1); input to H500, H492.

Phase 2 consists of two arithmetic passes and has three major functions listed as follows:

1. (M) → X2 The detailed transfer path is:



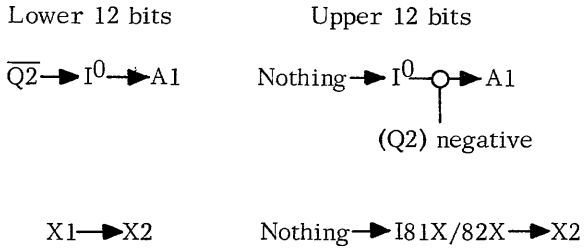
Bits 12-22



Bits 0-11
and bit 23

The transfer of $\overline{X1}$ to I81X/82X toggles the bias bit so that bits 12-22 of X2 hold the real exponent at V503 time.

2. Subtract exponents - The difference of the exponents is formed by forcing the 48-bit adder to appear as an 11-bit adder. This is accomplished by setting K872/873 (add exponent 2) at V503 time which effectively causes groups 0-11 to appear as passes and not generates. Figure 421 is a simplified logic diagram of how group 12 of the adder is affected by the setting of add exponent 2. For the time interval defined by add exponent 2, U800 = 1 and U801 = 0. This insures that all inputs to U746 and U748 are disabled while all inputs to U749 are partially enabled. Therefore, group 12 only senses for carry inputs from groups 12, 13, 14, and 15. Figure 317 shows the data flow for subtract exponents. The operands are presented to the adder at V503 time of the arithmetic pass and $\overline{sum}$ will be sampled at V501 time of the next arithmetic pass.
3. Extend sign - The second arithmetic pass of phase 2 will be used to extend the sign of the coefficient for both divisor and dividend. This sign extension will allow the computer to perform the divide step in the same manner as the double-precision divide. At V505 time of the first arithmetic pass the negative form of (M) is in X1 and bit 0-11 and bit 23 of the original value in A are in Q2. The enables for extend sign are:



Phase 2: Second Arithmetic Pass

V500: Clear Q3;
input to H943.
V501: $\overline{Sum}$ to Q3 (set bit 11 of Q3);
set K874/875 (extend sign).
V502: Clear K872/873 (add exponent 2).
V503
V504
V505: Input to H844 and H854.
V844, V854: I^0 → A1, ($\overline{Q2}$ → I^0) lower 12 bits only;
If Q2 negative, I^0 → A1 (nothing → I^0),
upper 12 bits only;
X1 → X2, lower 12 bits only;
I81X/82X → X2 (nothing to I81X/82X)
upper 12 bits only.

During phase 2, program control has been performing a second ROP at M+1.

Phase 3 - Initialize

N687: Start arithmetic coincident with V009 of second ROP;
if A1 negative, set K852/853 (complement AQE);
input to H500, H510, H512, and H820.
V500: Clear A2, Q2, E2, and X1; set K560/561 (arithmetic busy); clear K874/875 (extend sign); input H871; if K852/853 is set, input to H611.
V501: I^4 to X1 (DBR to I^4 if (M) was positive);
($\overline{DBR}$ to I^4 if (M) is negative);
E1 to E2;
set K802/803 (word 2);
clear K874/875 (extend sign).
V502: Clear K850/851 (bus to I^4 enable).
V503: Input to H612 if K852/853 is set
V504: Set K804/805 (optional arithmetic busy).
V505: Set K808/809 (divide 1);
clear K852/853 (complement AQE);
input to H500, H620, and H820.

The preceding timing initializes the registers for the divide step. At V500 time K560/561 (arithmetic

The timing shown is used only if complement of AQ is necessary.

V611: I^2 to A2 (A1 to I^2);

V612: Clear A1 and Q1; input H613 and H915.
V613: I^0 to A1 ($\overline{A2}$ → I^0);
I^1 to Q1 ($\overline{Q2}$ to I^1).

The preceding timing insures that the positive form of the dividend is in A1Q1 previous to divide step.

busy) is set, which insures that main control cannot enter the arithmetic section until execution of this instruction is complete. At V504 time K804/805 (optional arithmetic busy) is set, which enables the 48-bit adder. At V501 time the negative form of (M + 1) is gated into X1; E1, now clear, is gated into E2. Therefore, going into divide step, E1 is cleared, X2X1 holds the divisor sign extended, and A1Q1 holds the positive form of the dividend.

DIVIDE STEP

V500: Clear A2, Q2, and E2; clear K800/801 (wait word 2); input to H611 and H871.

V501: I^2 to A2 ($\overline{Q1}$ to I^2); I^3 to Q2 ($\overline{A1}$ to I^3); E1 to E2.

V502: If cycle 37_{10}, clear K808/809 (divide 1) which sets K882/883 (shift right 1) to block further operations with E.

V503: Clear K802/803.

V504: Clear A1, A1, and E1; input to H525, H565, and H905.

V505: E2 to E1, LS1;
I^0 to A1, LS1 except last cycle; I^1 to Q1, LS1 except last cycle (for last cycle these are right shift 1).
QB = 1 (O881 to E400);
left adder to I^0; right adder to I^1.
If sum negative or cycle 38:
QB = 0 (O881 to E400);
$\overline{Q2}$ to I^0 to I^1; $\overline{A2}$ to I^2.
Input to H500, H620, and H820.
If not cycle 38_{10}, repeat divide step;
if cycle 38_{10}, set K884/885 (round up) if sum positive or -0; input to H810.
If SCR = 44_8, set K812/813 (last cycle).

Advance SCR.
Transfer (SCR);
if cycle 38, set K844/845 (swap AQE 1).

Figure 325 is a block diagram of the enables and transfers for divide step. At the initiation of divide step the positive form of the coefficient of the dividend is in A1Q1, the negative form of the coefficient of the divisor is in X2X1, and E1 is clear. At the completion of divide step the lower 37 bits of E1 holds the positive form of the quotient and the lower 36 bits of A1Q1 holds the remainder.

The following are major items for divide step:

1. Quotient bit - the quotient bit is formed in 2^0 of E1. On each arithmetic pass except the 38th, at V505 time 2^0 of E1 is set if the sum is positive. O887 (Logic Diagrams, page 4-11) translates for the quotient bit.
2. Pass 37 - Pass 37_{10} is necessary because the actual division is a 36-bit divisor into a 72-bit dividend. Therefore the quotient is 37-bit. At V505 time of this pass the lower 37 bits of E1 holds the quotient and the lower 37 bits of AQ1 hold the remainder left shifted 1 place.
3. Pass 38 - the function of the 38_{10} pass is to ascertain if a round operation is necessary. A subtract is performed on this pass and, if the result is positive, K884/885 (round up) will be set at V505 time. The result of this subtract will not in any case be gated into A1Q1. Instead A2Q2, which hold the proper remainder left shifted 1 place, will be right shifted 1 into A1Q1. See figure 326 for an example of why passes 37 and 38 are needed.
4. Shift count register - SC register was cleared during copy F1 to F2 time of RNI and will be counted up to keep track of the arithmetic passes for divide step. At V500 time of pass 45_8 (SC register = 44_8) K812/813 (divide last cycle) will be set.
5. Termination - occurs after 38_{10} passes. At V500 Time of pass 37 divide last cycle is set but since SC register = 44_8 at V502 time divide 1 cannot be cleared; therefore the enables for divide step remain up. On pass 37 SC register advances to 45_8, so at V502 time of pass 38 divide 1 is cleared, dropping the divide step enables.

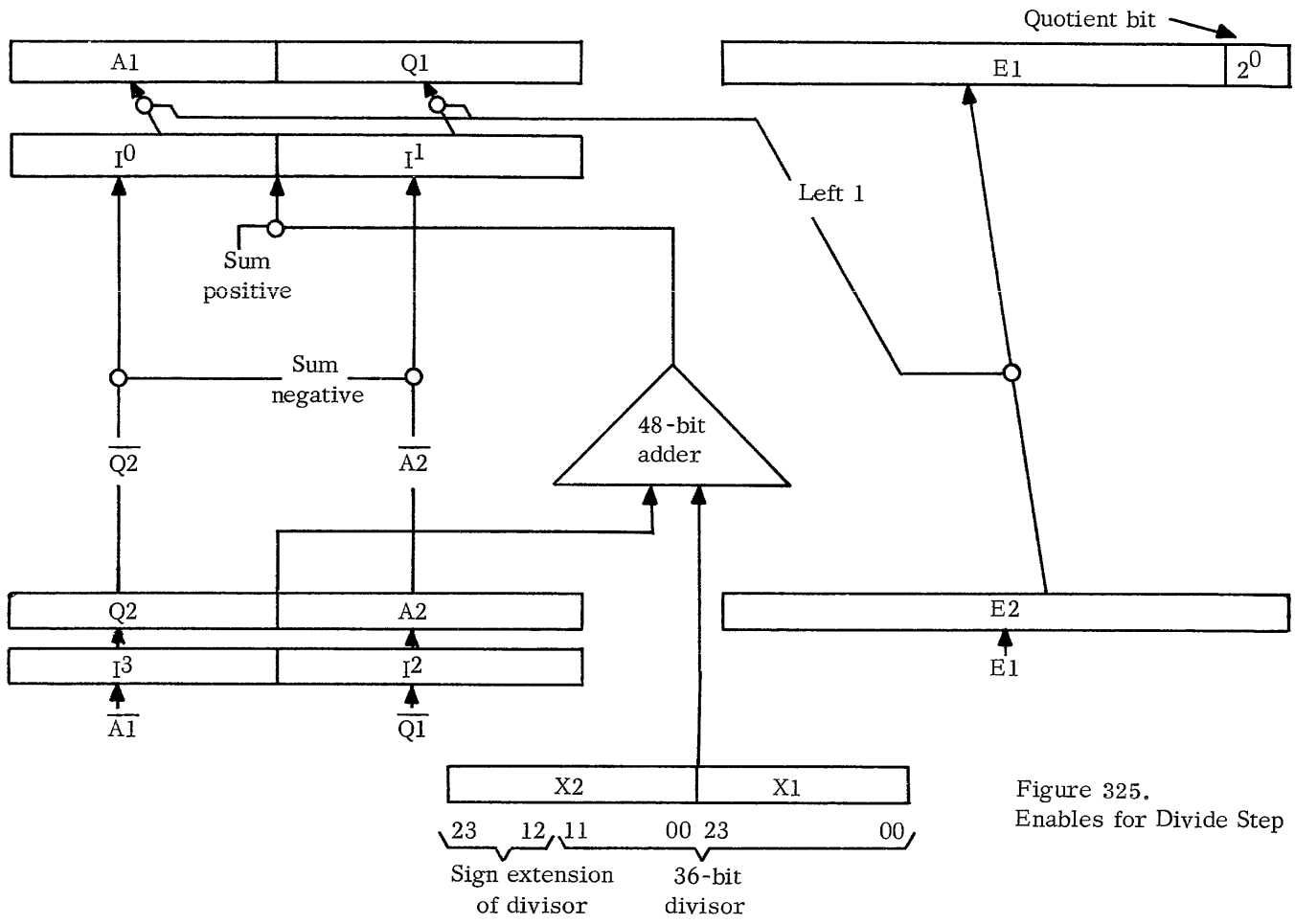


Figure 325.
Enables for Divide Step

Swap AQ and E: First Pass

V500: Clear A2, Q2, E2, X2, and X1,
clear K812/813 (divide last cycle);
input to H871.
V501: E1 to E2.
V502: Set K846/847 (swap AQE 2):
input to H811.
V503: A1 to X2;
I^2 to X1 (Q1 to I^2); input to H834.
V504: Clear E1; input to H915
V505: $\overline{X2X1}$ to E_{inv} to E1;
clear K844/845 (swap AQE 1);
input to H500, H620, and H810.

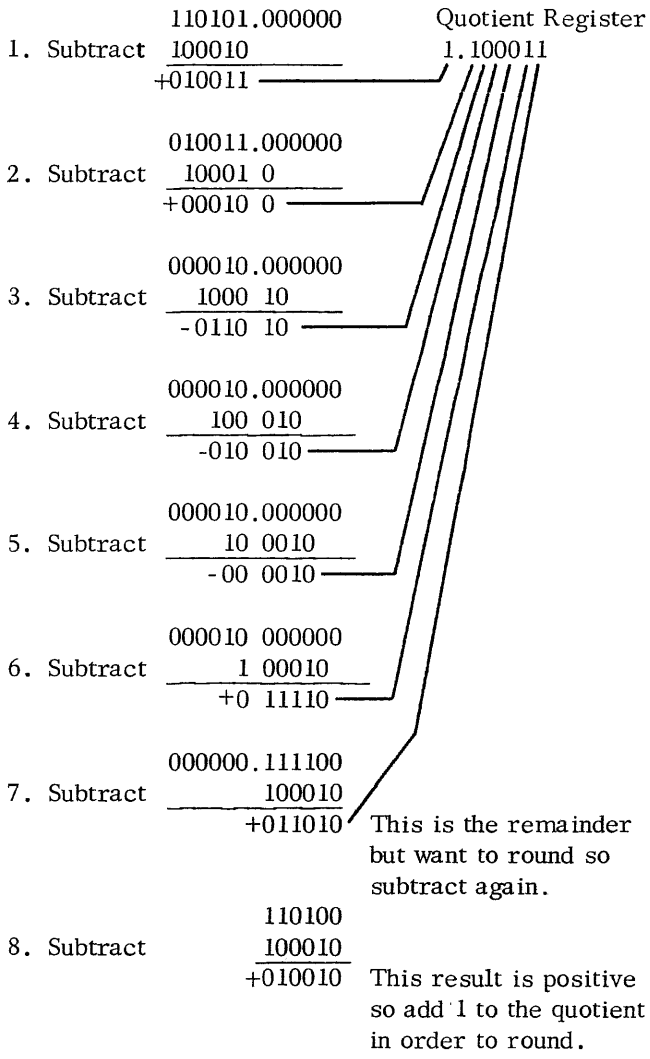
Swap AQ and E consists of two arithmetic passes and is used to place the coefficient of the quotient in AQ and the remainder in E. The timing shown above is the first pass of swap AQ and E and it accomplishes the following:

1. Places the quotient in E2.
2. Places the remainder in E1.

The data flow for this pass is shown above right.

E1→E2

Quotient to E2,
A1→X2, X2X1→E_{inv}→E1,
Q1→I^2→X1.
Remainder to E1



Let's look at a 6-bit example to determine why passes 37 and 38 are necessary for the 63 instruction. Since the computer must subtract to divide so must you. The subtraction is with real signed numbers and each time the difference is positive the appropriate bit in the quotient register is set. Seven steps were required to arrive at the correct quotient one step more than the number of bits. The eighth step was used to see if the answer should be rounded. After the eighth step the remainder must be shifted right 1 to bring it back to its proper value.

Figure 326. Example of Divide Step

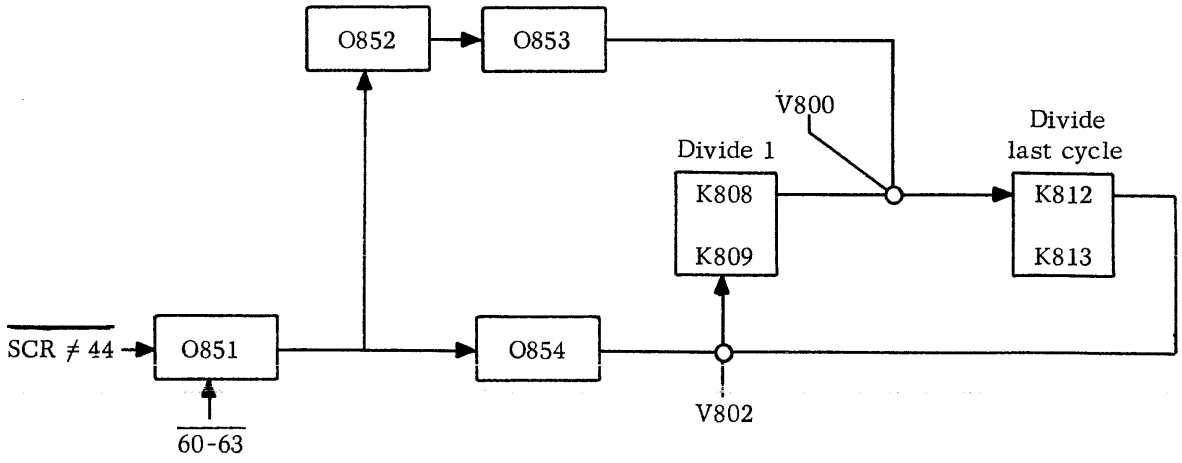


Figure 327. Terminating Divide Step

Swap AQ and E: Second Pass

V500: Clear A2, Q2, X2, and X1;
input to H821.
V501: E2 upper to X2;
set K848/849 (swap AQE 3);
clear K882/883 (shift right 1).
V502: Clear K846/847 (swap AQE 2).
V503: Input to H814 and H844.
V504: Clear A1 and Q1;
input to H515.
V505: I^0 to A1 (left adder to I^0);
E2 lower to Q1;
set K822/823 (round);
input to H500, H620, H810, and H820.

Timing shown completes swap AQ and E by placing the quotient in A1Q1. Figure 328 is a block diagram of the transfer paths used.

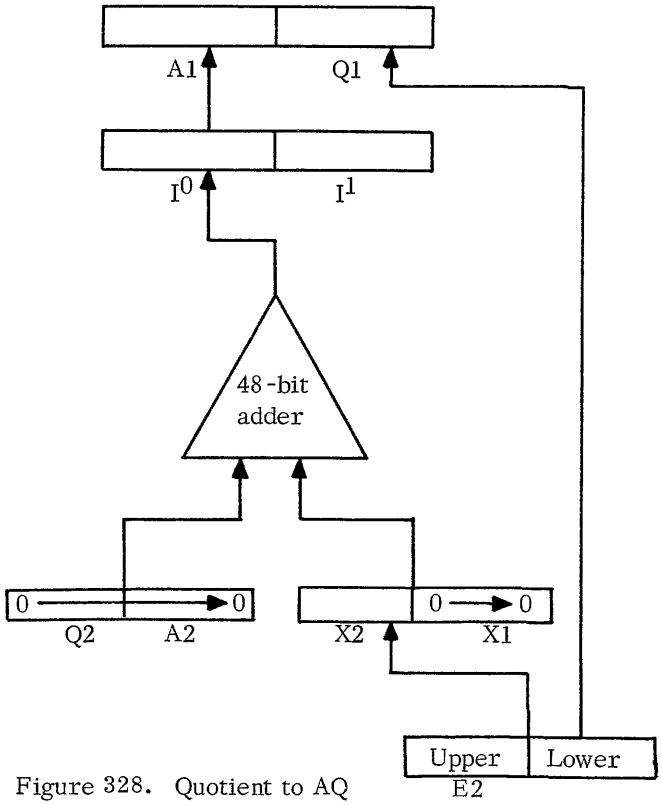


Figure 328. Quotient to AQ

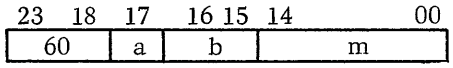
ROUND

V500: Clear A2, Q2, E2, X2, and X1;
input to H611.
V501: I^2 to A2 ($\overline{Q1}$ to I^2);
I^3 to Q2 ($\overline{A1}$ to I^3).
X2X1 forced to a +1 if round up is set, otherwise X2X1 = 0's. Adder propagation begins.
V502
V503: Input to H612.
V504: Clear A1 and Q1;
set K898/899 (short cycle);
input to H613 if sum $\neq$ -0.
V505: If sum $\neq$ -0:
I^0 to A1 (left adder to I^0);
I^1 to Q1 (right adder to I^1);
set K826/827 (normalize) which removes constant clear from K880/881 and K886/887.
Input to H500, H600, H620, and H820.

Timing for round occurs whether or not an actual round operation is to take place. The only possible case for rounding is plus 1. To accomplish plus 1, A1Q1 is transferred to Q2A2 and presented to the adder. X2X1 is forced to 0→01 and presented to the adder. The logic for forcing X2X1 is shown in figure 321. If no rounding is required, X2X1 is cleared and a 48-bit add of the quotient and all zeros takes place, which preserves the original quotient. The timing for all floating-point instructions is common for normalize, adjust exponent, merge, and complement. Go to page 244 to complete the timing for this instruction.

FLOATING-POINT ADD

FAD Floating-point Addition to AQ



a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description

Add the 48-bit combined contents of M and M + 1 to (AQ). The rounded and normalized sum appears in AQ. The upper order bits of E hold the portion of the operand that was shifted into E during the equalization of exponents. Refer to figure 329 for operand format. The pencil and paper example of figure 330 shows how the computer performed a floating-point addition. The original octal numbers are in the upper right-hand corner of the page and the flow for the problem is defined by the heavy black lines. The computer would start with the operands packed in floating-point format. The steps are as follows:

1. Toggle the bias bit to obtain the real exponents.
2. Sent the exponents to an 11-bit adder and perform a subtract. The magnitude of the difference indicates how much the smaller operand must be right shifted to align the octal points. The sign of

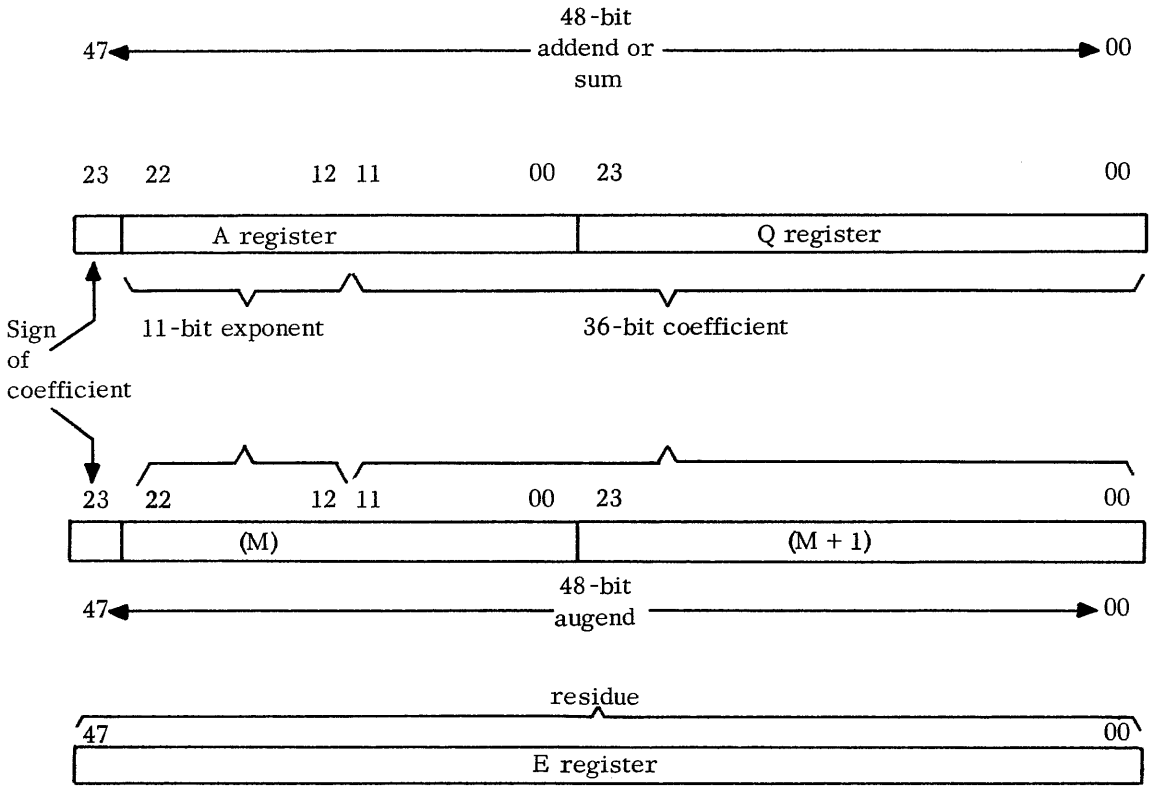


Figure 329. Operand Format for Round

the difference indicates which operand is smaller. The exponent of the larger operand will be retained to form the result of the floating-point addition.

3. Extend the sign of the coefficients.
4. Shift the coefficient of the smaller number right to align the octal points. The bits shifted off are retained so that they may be used to determine if a round is necessary. These bits form the residue.
5. Perform the addition of coefficients.
6. Round. The residue is inspected for the round under the following rules:
   a. If the operand shifted right was negative and the highest order bit of the residue is a zero, add -1 to round. This operation is referred to as round down.
   b. If the operand shifted right was positive and the highest order bit of the residue is a 1, add +1 to round. This operation is referred to as round up.
   c. In any other case add all 0's to round.
   d. Round down (-1) is the same as round up except it deals with negative numbers. Adding a -1 to a negative number and complementing is the same as complementing first and then adding a +1.
7. Normalize is not necessary for this example.
8. Adjust exponent is not necessary since normalize did not take place.
9. Merge the exponent and coefficient.
10. Toggle the bias bit.

In the lower left-hand corner of figure 330 the original octal numbers are added and the sum is packed in floating-point format. Note that this is the same answer arrived at by using the computer method. Parallel timing for the instruction is shown in figure 331.

DETAILED TIMING

Phase 1

N687: Start arithmetic coincident with V003 time of first ROP;
set K536/537 (FADR 1);
set K596/597 (FADR 2);
input to H500 and H510.

V500: Clear A2 and X1.

V501: $I^5 \rightarrow$ A2 ($\overline{FL_{15}} \rightarrow I^5$);
$I^6 \rightarrow$ X1 ($\overline{+1} \rightarrow I^6$);
input to H942.

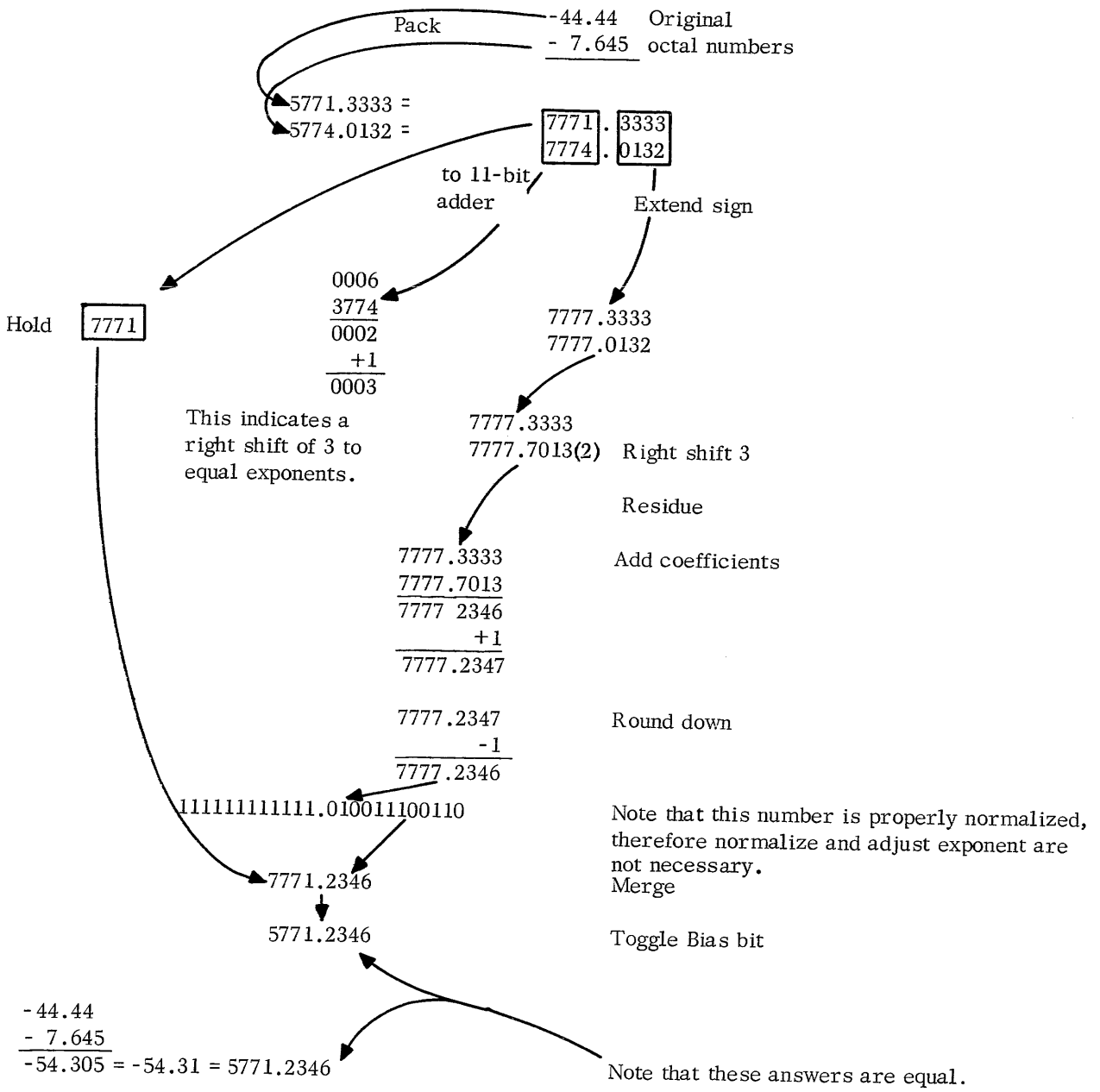


Figure 330. Pencil and Paper Example of Floating Point Addition

V502: Clear Q3;
input to H943.
V503: A1 exponent → Q3;
clear F_{L15};
set K870/871 (add exponent 1).
V504: Clear K596/597 (FADR 2);
set K104/105 (start arithmetic 2).
V505: $I^0 \rightarrow F_{L15}$.

At V501 time the enables come up to form M + 1. At V505 time M + 1 is gated to F_{L15} so that it is available for the second ROP. At V503 time A1 exponent is placed in the holding register Q3. During the transfer the bias bit is toggled, which places the real exponent in Q3 if A1 is positive or the complement of the real exponent in Q3 if A1 is negative.

Phase 2: First Arithmetic Pass

N659: Start arithmetic coincident with V009 of the second ROP and with V505 of phase 1;
input to H500, H510, and H512.
V500: Clear A2, Q2, and X1;
input to H513 and H531.
V501: $I^4 \rightarrow$ X1 ($\overline{DBR} \rightarrow I^4$);
$I^3 \rightarrow$ Q2 ($\overline{A1} \rightarrow I^3$) bits 0-11 and 23;
I86X/87X → Q2 bits 12-22 (Q3 → I86X/87X if (Q3) negative); ($\overline{Q3} \rightarrow$ I86X/87X if (Q3) positive); input to H854.

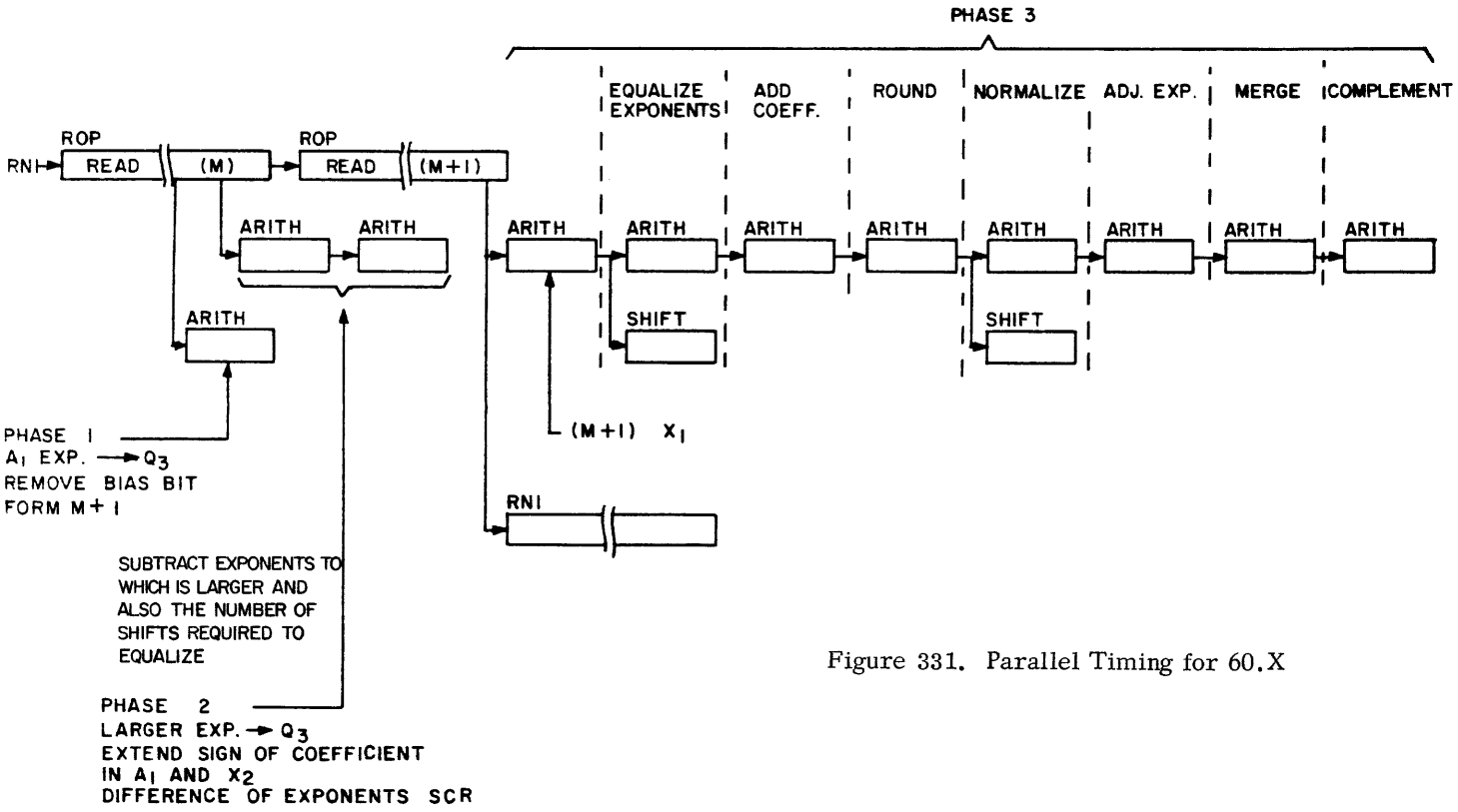
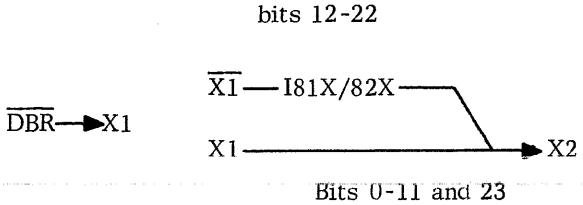


Figure 331. Parallel Timing for 60.X

V502: Clear X2;
input to H821 and H855.
V503: X1→X2, bits 0-11 and 23;
I81X/82X→X2, bits 12-22 (X1→I81X/82X if (X1) positive);
($\overline{X1}$→I81/82X if (X1) negative);
set K800/801 (wait word 2);
set K872/873 (add exp 2); enable 11-bit adder;
input to H834.
V504: Clear E1.
V505: Clear K870/871 (add exponent 1);
input to H500 and H942.

Phase 2 consists of two arithmetic passes and has six functions listed as follows:

1. (M)→X2 - The detailed transfer path is:



Bits 0-11 and 23

The transfer of $\overline{X1}$→I81X/82X toggles the bias bit so that bits 12-22 of X2 hold the real exponent at V503 time.

2. Difference of exponents to SC register - The difference of the exponents is formed by forcing the 48-bit adder to appear as an 11-bit adder. This is accomplished by setting K872/873 (add exponent 2) at V503 time, which causes groups 0-11 to appear as passes and not generates. Figure 332 is a simplified logic diagram of how group 12 of the adder is affected by the setting of add exponent 2. For the time interval defined by add exponent 2, U800 = 1 and U801 = 0. This insures that all inputs to U746, 747, and 748 are disabled while all inputs to U749 are partially enabled. Therefore, group 12 only senses for carry inputs from groups 12, 13, 14, and 15. Figure 333 shows the data flow for subtract exponents. The operands are presented to the adder at V503 time of the first arithmetic pass of phase 2 and $\overline{sum}$ is sampled at V501 time of the second arithmetic pass of phase 2. At V503 time of the second arithmetic pass the complement of the required number of shifts is placed in SC register so that the octal points can be aligned before the addition of coefficients. If the difference between exponents is greater than 77_8, I873 (Logic Diagrams, page 4-25) will block the transfer of $\overline{Q3}$→SC register. Therefore, 77_8 shifts take place and the final answer will be the larger operand.
3. Larger Exponent to Q3 - At V505 time of the second arithmetic pass of phase 2 one of the following will occur:

a. If the sum of subtract exponents was positive, A1 exponent is the larger exponent. Therefore A1 exponent is gated to Q3 so that it will be available for adjust exponent. If (A1) is positive, the real exponent is placed in Q3; if (A1) is negative, the complement of the real exponent is placed in Q3.

b. If the sum of subtract exponents was negative, M exponent is the larger exponent. The complement of M exponent is in X2 from subtract exponents. Therefore, gating $\overline{X2}$ to Q3 places the M exponent in Q3 so it will be available for adjust exponent. The real exponent is placed in Q3.

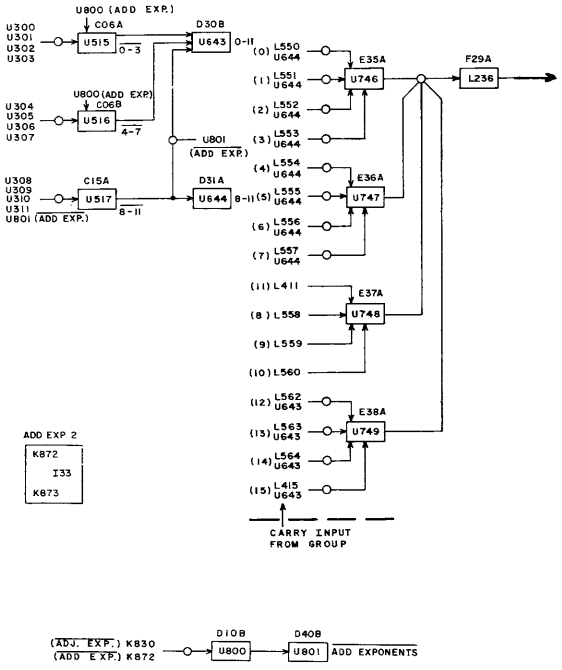


Figure 332. Enabling the 11-Bit Adder

4. Smaller number to A1 = If the sum of subtract exponent was positive, (M, M+1) is smaller than (AQ) and must be right shifted to align the octal points. Since only AQE have shift capabilities (M, M + 1) must be placed in AQ and (AQ) will be placed in X2X1. It is important to note that during subtract exponents the standard adder is functioning as a 24-bit adder. The enables for placing (M) in A1 are:

Right adder $\rightarrow I^3 \rightarrow$ Q2;

$\overline{Q2} \rightarrow I^0 \rightarrow$ A1

The exchange of (AQ) and (M, M + 1) will be completed during the first arithmetic pass of phase 3.

5. Larger Number to X1 - If the sum of subtract exponents was positive (M, M + 1) is smaller than (AQ) and must be right shifted to align the octal points. Since only AQE have shift capabilities (M, M + 1) must be placed in AQ as in item 4. In order to make room for this transfer (AQ) must be placed in X2X1. During phase 2 (A) is transferred to X1 and then X1 is transferred sign extended into X2. The transfer path for A1 to X2 is:

$\overline{A1} \rightarrow I^2 \rightarrow$ X1 $\rightarrow$ X2

The exchange of (AQ) and (M, M + 1) will be completed during the first arithmetic pass of phase 3.

6. Extend sign - At V505 time of the second arithmetic pass of phase 2 the operands are positioned with the upper 24 bits of the larger operand in X1 and the upper 24 bits of the smaller operand in Q2. The enables for extend sign are:

Lower 12 bits

$\overline{Q2} \rightarrow I^0 \rightarrow$ A1

Upper 12 bits

Nothing $\rightarrow I^0 \rightarrow$ A1

(Q2) negative

Lower 12 bits

X1 $\rightarrow$ X2

Upper 12 bits

Nothing $\rightarrow$ I81X/82X $\rightarrow$ X2

(X1) negative

Nothing (zeros) into the inverter ranks I^0 and I^8 equals all 1's out. If a positive sign extension (0's) is required, the gates $I^0 \rightarrow$ A1 and $I^8 \rightarrow$ X2 are simply not enabled.

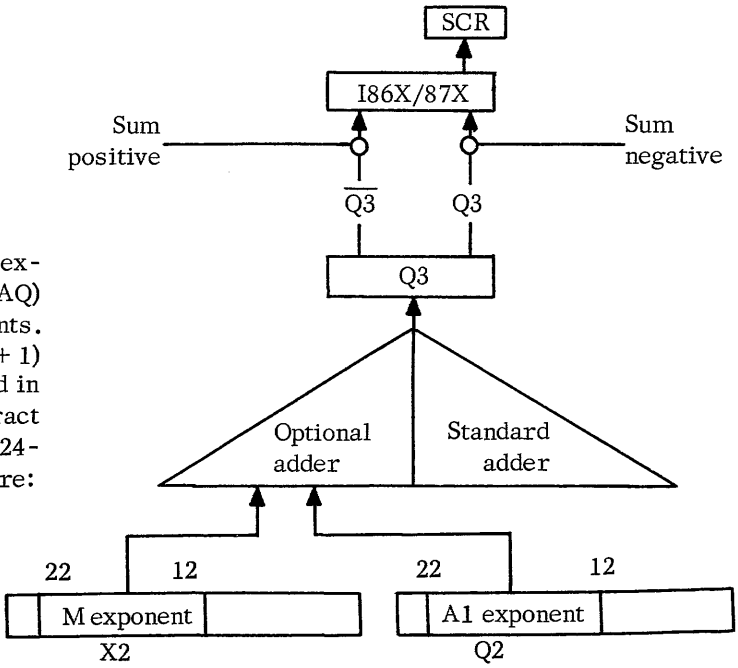


Figure 333. Data Flow for Subtract Exponents

Phase 2: Second Arithmetic Pass

	If sum of subtract exponent positive A exponent $\geq$ M exponent	If sum of subtract exponent negative M exponent $\geq$ A exponent
V500: Clear Q3; input to H943		
V501: Left adder→Q3 (extend bit 10); set K874/875 (extend sign); input to H944.	Set K818/819 (swap A1 and X2); input to H620.	
V502: Clear SC register; clear K872/873 (add exponent 2); input to H945.	Clear A2 and Q2; input to H513.	
V503: I86X/87X→SC register; set K814/815 (equalize exponent); clear K532/533 (unlike signs 1); input to H942.	$\overline{Q3}$→I86X/87X; I^3→Q2 (right adder→I^3); input to H810.	Q3→I86X/87X.
V504: Clear Q3; input to H943.	Clear X2 and X1; input to H511.	
V505: Input to H844 and H854.	I^2→X1 ($\overline{A1}$→I^2); A1→Q3.	$\overline{X2}$→Q3.

V844, V854: Clear A1 and X2; input to H555 and H855; if X1 negative, input to H821.

V515, V855: I^0→A1, bits 0-11 ($\overline{Q2}$→I^0); if Q2 negative, I^0→A1, bits 12-23 (nothing to I^0); X1→X2, bits 0-11 and 23; if X1 negative, I81X/82X→X2, bits 12-23 (nothing to I81X/82X); set K820/821 (swap Q1 and X1); set K800/802 (wait word 2).

Phase 3

	If A exponent $\geq$ M exponent
N687: Start arithmetic coincident with V009 of second ROP; input to H500, H510, and H512; clear K818/819 (swap A1 and X2).	
V500: Clear A2, Q2, and X1; set K560/561 (arithmetic busy); input to H531. Clear K874/875.	Input to H513.
V501: I^4→X1 ($\overline{DBR}$→I^4); set K802/803 (word 2); set K898/899 (short cycle).	I^3→Q2 ($\overline{Q1}$→I^3); input to H524.
V502: Set K802/803 (word 2); set K898/899 (short cycle).	Clear Q1; input to H555.
V503:	I^1→A1 ($\overline{X1}$→X1); input to H810.
V504: Set K804/805 (optional arithmetic busy); transfer 1's count; transfer 10s count.	Clear X1; input to H531.
V505: Set K814/815 (equalize exponent); clear K820/821 (swap Q1 and X1); input to H500, H600, H620, and H820.	I^4→X1 ($\overline{Q2}$→I^4)

The preceding arithmetic timing places (M + 1) in X1 at V501 time. At V504 time the shift count register is equalized so that it will count properly during equalize exponents. Remember that the complement of the required number of shifts was placed in rank 1

The preceding enables occur only if A1 exponent $\geq$ M exponent which required that (M, M+1) be placed in AQ to allow shifting. During phase 2 (A1) was placed

of SC register during phase 2. The setting of optional arithmetic busy at V504 time enables the 48-bit adder. The setting of K560/561 at V500 time insures that main control cannot get into the arithmetic section until the execution of this instruction is complete.

Phase 2: Equalize Exponents

V500: Clear A2, Q2, and E2;
clear K800/801 (wait word 2);

V503: Clear K802/803 (word 2).

V504

V505: Input to H500, H600, H620, and H820;
if SC register ≠ 77, repeat shift cycle;
if SC register = 77, set K816/817 (add/subtract coefficient);
clear K898/899 (short cycle).

The preceding arithmetic timing is a short cycle (4 Ø times) since K898/899 (short cycle) was set at V501 time of the previous arithmetic pass.

Phase 3: Add Coefficients

V500: Clear A2, Q2, E2;
clear K814/815 (equalize exponents);
input to H611.

V501: $I^2 \rightarrow$ A2 ($\overline{Q1} \rightarrow I^2$);
$I^3 \rightarrow$ Q2 ($\overline{A1} \rightarrow I3$);
set K884/885 (round up) if $A2^{23} = 0$ (A is positive) and $E2^{47} = 1$.

V502: Set K890/891 (round down) if $A2^{23} = 1$ (A is negative) and $E2^{47} = 0$.

V503: Input to H612.

V504: Clear A1 and Q1;
input to H613.

in X2 and (M) was placed in A1. This timing places (M + 1) in Q1 and (Q1) in X2.

V600: If SC register ≠ 7X, input to H523, H561, and H881;
if SC register = 7X, input to H513, H551, and H871.

V601: Input to H514, H524, and H834;
if SC Register ≠ 7X:
$I^2 \rightarrow$ A2, right shift 10 ($\overline{A1} \rightarrow I^2$);
$I^3 \rightarrow$ Q2, right shift 10 ($\overline{Q1} \rightarrow I^3$);
E1 → E2, right shift 10;
advance 10s count.
If SC register = 7X:
$I^2 \rightarrow$ A2 ($\overline{A1} \rightarrow I^2$);
$I^3 \rightarrow$ Q2 ($\overline{Q1} \rightarrow I^3$);
E1 → E2.

V602: Clear A1, Q1, and E1; transfer 10's count;
If SC register ≠ X7, input to H525, H565, and H895;
If SC register = X7, input to H515, H555, and H885.

V603: If SC register ≠ X7:
$I^0 \rightarrow$ A1, right shift 1 ($\overline{A1} \rightarrow I^0$);
$I^1 \rightarrow$ Q1, right shift 1 ($\overline{Q2} \rightarrow I^1$);
E2 → E1, right shift 1, advance 1's count.
If SC register = X7:
$I^0 \rightarrow$ A1 ($\overline{A1} \rightarrow I^0$);
$I^1 \rightarrow$ Q1 ($\overline{Q2} \rightarrow I^1$);
E2 → E1.

The preceding timing is from the shift timing chain. At the completion of equalize exponents the coefficient of the smaller operand will be positioned in AQE sign extended. That portion of the operand contained in E is called residue and will be used to determine if a round operation is necessary. If the difference of exponents is $\geq 36_{10}$, AQ will contain only the sign extension.

V505: I^0→A1 (left adder→I^0);
I^1→Q1 (right adder→I^1);
set K822/823 (round);
input to H500, H620, and H810;
input H510 if round down.

During this arithmetic pass the coefficients are added. Figure 334 shows the data flow for add coefficients. During this pass the enables for round are determined by the setting or not setting, round up or round down. If the residue is ≥ one-half, a round is necessary.

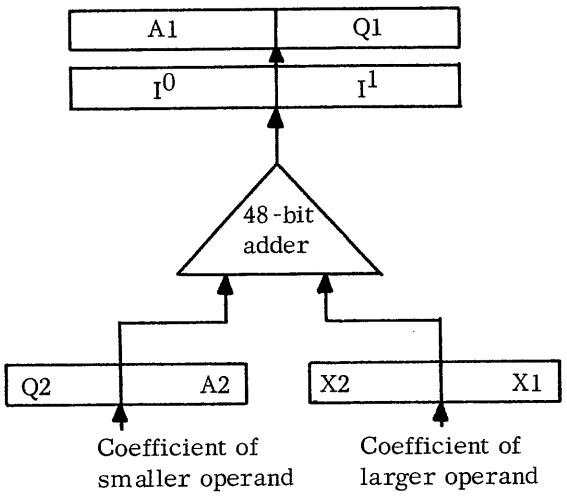


Figure 334. Enables for Add Coefficients

Phase 3: Round

V500: Clear A2, Q2, X1, and X2;
clear K816/817 (add/subtract coefficient);
input to H611.

V501: I^2→A2 ($\overline{Q1}$→I^2);
I^3→ Q2 ($\overline{A1}$→I^3);
input to H944.

V502: Clear SC register.

V503: Input to H612.

V504: Clear A1 and Q1;
set K898/899 (short cycle);
if sum = -0, input to H613.

V505: If sum ≠ -0:
I^0→A1 (left adder→I^0);
I^1→Q1 (right adder→I^1);
set K826/827 (normalize);
remove clear from K880/881 and K886/887;
input to H500, H600, H620, and H820.

If round down,
input to H531 and H821.

I^4→X1 (nothing to I^4);
I400→X000 block by round down;
I81X/82X→X2 Upper (nothing→I81X/82X);
set X2 lower. (A -1 is thus forced into X2X1.)

If round up.

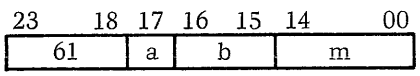
A +1 is forced to X2X1.

In both cases above, adder propagation occurs; $\overline{sum}$ is gated to I^0I^1.

The timing for round occurs whether or not an adjustment of the coefficient must be made. Round can be +1 or -1 for the 60 instruction. For a +1 operation the sum of the coefficients is placed in Q2A2, X2X1 is forced to 0→01, and a 48-bit add is performed. For a -1 operation the sum of the coefficients is placed in Q2A2, X2X1 is forced to 7→76, and a 48-bit add is performed. If neither a +1 nor a -1 is required, X2X1 is cleared prior to the 48-bit add and the original sum is preserved. The timing for all floating-point instructions is common for normalize, adjust exponent, merge, and complement. Go to page 244 to complete the timing for this instruction.

FLOATING-POINT SUBTRACTION

FSB Floating-point Subtraction From AQ



a = addressing mode designator
b = index register designator
c = storage address; M = m + (B^b)

Instruction Description

Subtract the 48-bit floating-point operand located at storage addresses M and M+1 from the floating-point operand in AQ. The rounded and normalized difference is displayed in AQ.

The upper order bits of E hold the portion of the operand that was shifted into E during the equalization of exponents. Refer to figure 335 for operand format.

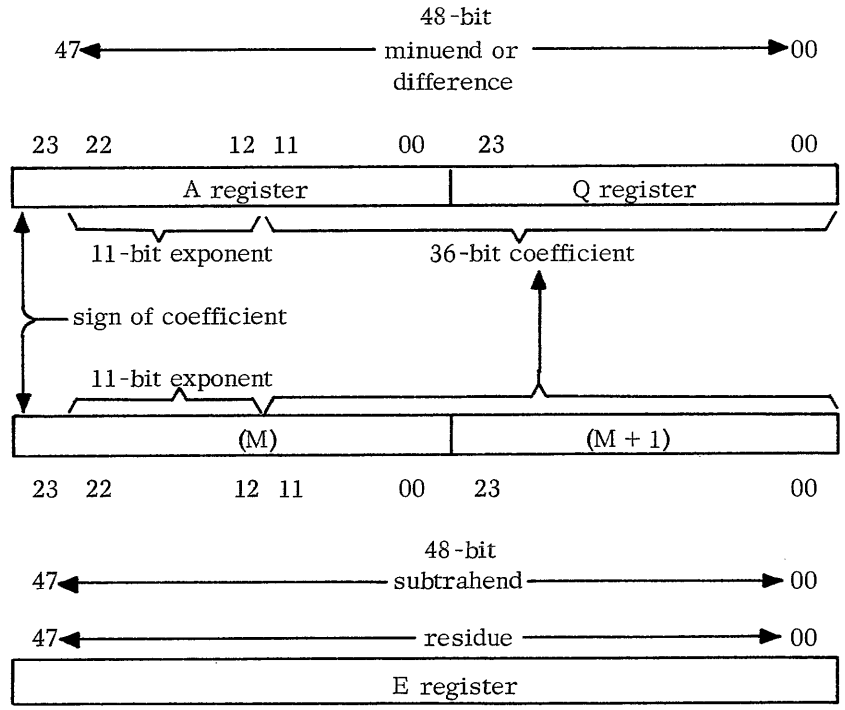


Figure 335. Operand Format for Floating Point Subtraction

The pencil and paper example, figure 336 shows how the computer performs a floating-point subtraction. The original octal numbers are in the upper right corner of the page and the flow for the problem is defined by the heavy black lines. The computer would start with the operands packed in floating-point format. The steps are as follows:

1. Obtain the complement of the subtrahend.
2. Toggle the bias bit to obtain the real exponents.
3. Send the exponents to an 11-bit adder, subtracting the exponent of the subtrahend from the exponent of the minuend. The magnitude of the difference indicates how much the smaller operand must be right shifted to align the octal points. The sign of the difference indicates which operand is smaller. The exponent of the larger operand will be retained to form the result of the floating-point subtraction.
4. Extend the sign of the coefficients.
5. Shift the smaller coefficient right to align the octal points. Note that the bits shifted off are retained so that they may be used to determine if a round is necessary. These bits form the residue.
6. Perform the subtraction.
7. The residue is inspected for the round under the following rules:
   a. If the operand shifted right was negative and the highest order bit of the residue is 0, add -1 to round.
   b. If the operand shifted right was positive and the highest order bit of the residue is 1, add +1 to round.
   c. In any other case add all 0's to round.
8. Normalize; a left shift of 1 is required for this example.
9. Adjust exponent by subtracting 1 since the shift was left to normalize.
10. Merge the exponent and coefficient.
11. Toggle the bias bit.

In the lower left-hand corner the original octal numbers are subtracted and the difference is packed in floating-point format. Note that this is the same answer arrived at by using the computer method.

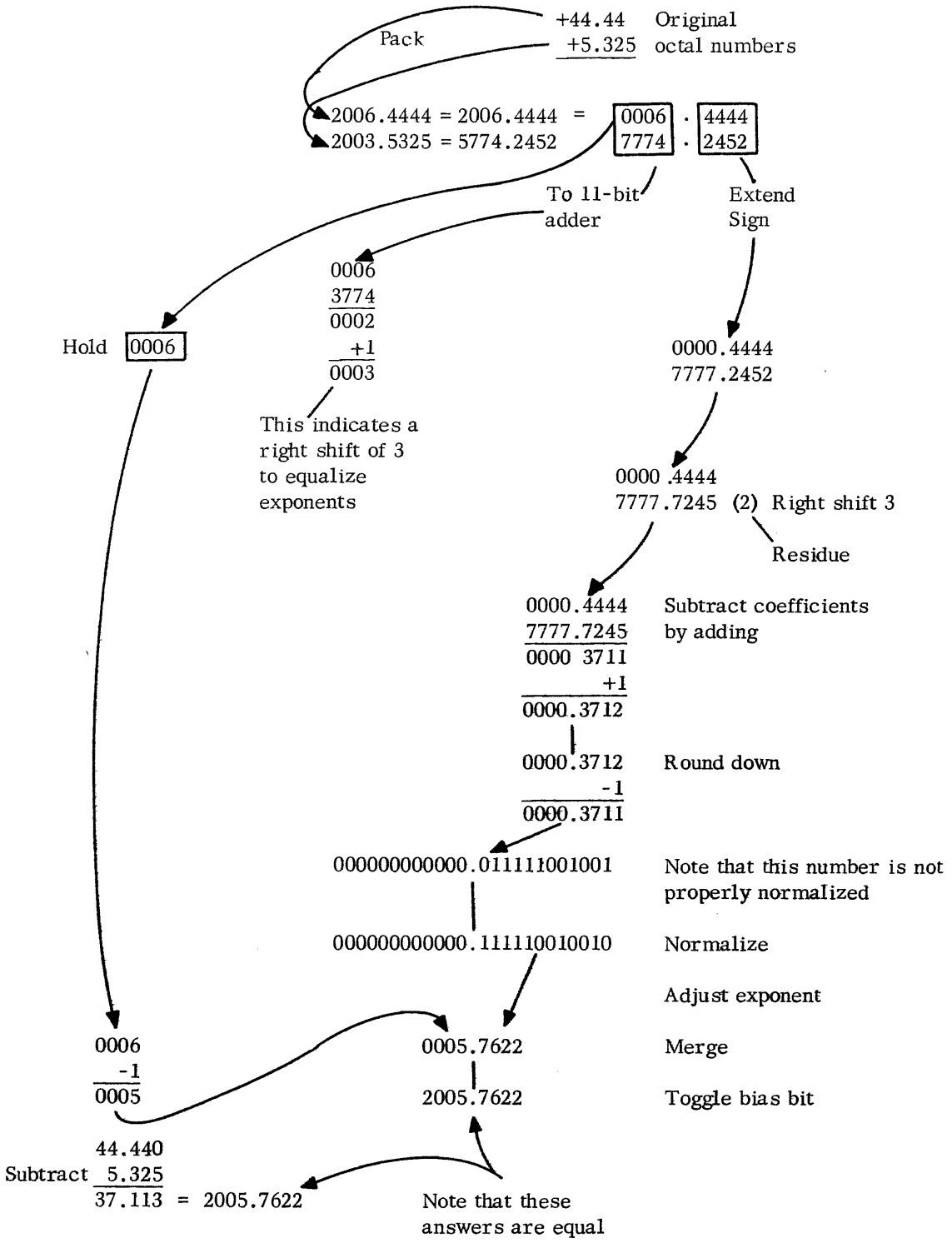


Figure 336. Pencil and Paper Example of Floating Point Subtraction

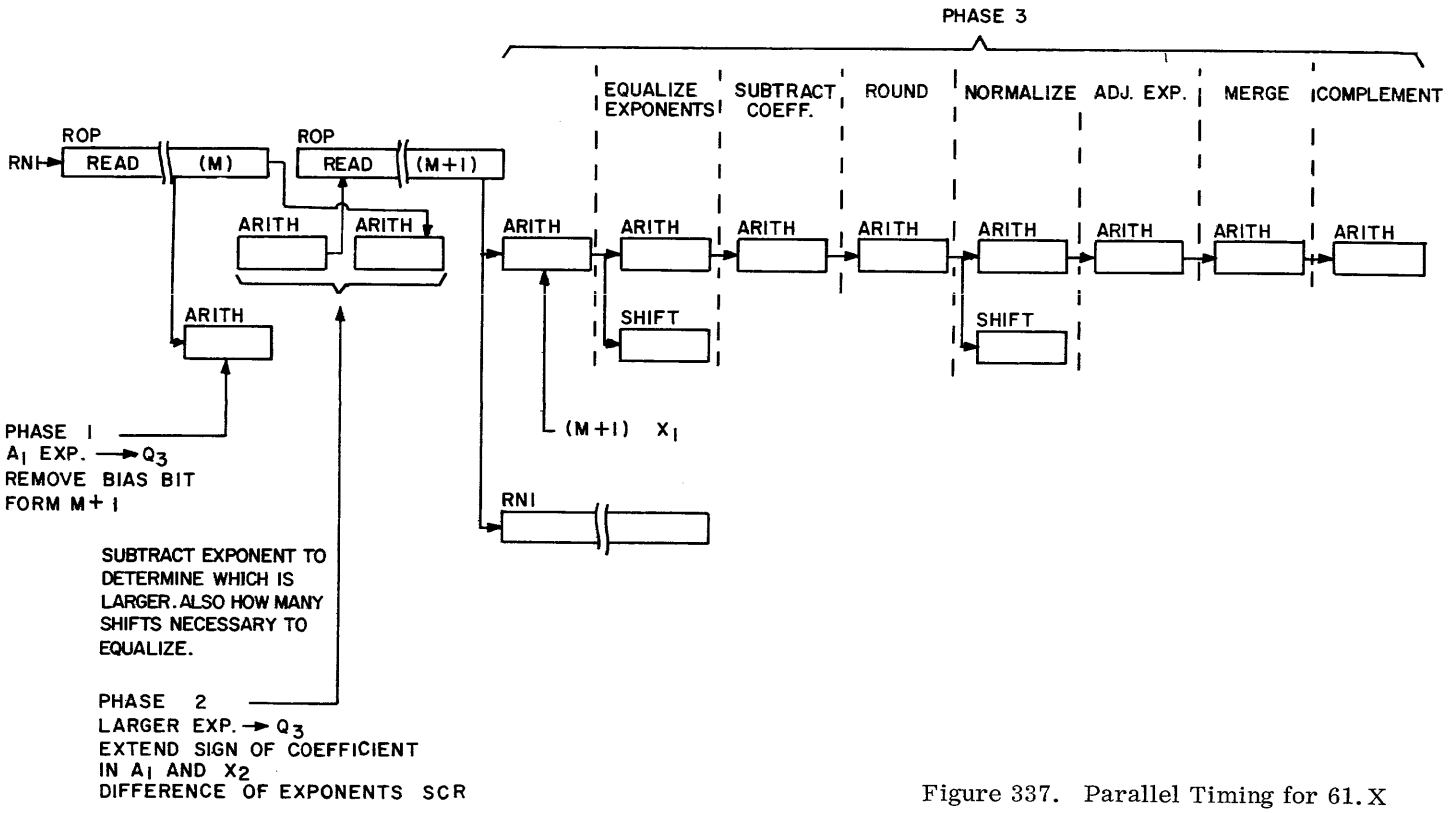


Figure 337. Parallel Timing for 61.X

DETAILED TIMING

Phase 1

N687: Start arithmetic coincident with V003 time of first ROP;
set K536/537 (FADR 1);
set K596/597 (FADR 2);
input to H500 and H510.

V500: Clear A2 and X1.

V501: $I^5 \rightarrow$A2 ($\overline{F_{L15}} \rightarrow I^5$);
$I^6 \rightarrow$X1 ($+1 \rightarrow I^6$);
input to H942.

V502: Clear Q3;
input to H943.

V503: A1 exponent$\rightarrow$Q3;
clear F_{L15};
set K870/871 (add exponent 1).

V504: Clear K596/597 (FADR 2);
set K104/105 (start arithmetic 2);
set K850/851 (DBR$\rightarrow I^4$ enable).

V505: $I^0 \rightarrow F_{L15}$ (sum $\rightarrow I^0$).

At V501 time the enables come up to form M + 1. At V505 time M + 1 is gated to F_{L15} so that it is available for the second ROP. At V503 time A1 exponent is placed in the holding register Q3. During the transfer the bias bit is toggled, which places the real exponent in Q3 if A1 is positive or the complement of the real exponent in Q3 if A1 is negative.

Phase 2: First Arithmetic Pass

N687: Start arithmetic coincident with V009 of the second ROP and with V505 of phase 1;
input to H500, H510, and H512.

V500: Clear A2, Q2, and X1;
input to H513 and H531.

V501: $I^4 \rightarrow$X1 (DBR$\rightarrow I^4$);
$I^3 \rightarrow$Q2 ($\overline{A1} \rightarrow I^3$), bits 0-11 and 23;
I86X/87X$\rightarrow$Q2, bits 12-23 (Q3$\rightarrow$I86X/87X if (Q3) is negative);
($\overline{Q3} \rightarrow$I86X/87X if (Q3) positive);
input to H854.

V502: Clear X2;
input to H821 and H855.

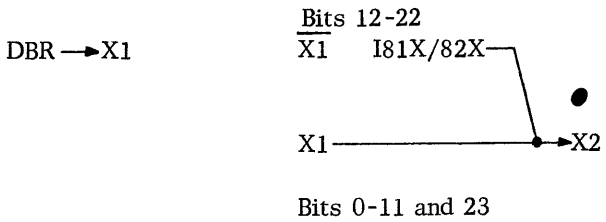
V503: X1$\rightarrow$X2, bits 0-11 and 23;
I81X/82X$\rightarrow$X2, bits 12-22
($\overline{X1} \rightarrow$I81X/82X if (X1) positive);
($\overline{X1} \rightarrow$I81X/82X if (X1) negative);
set K800/801 (wait word 2);
set K872/873 (add exponent 2);
input to H834.

V504: Clear E1.

V505: Clear K870/871 (add exponent 1);
input to H500 and H942.

Phase 2 consists of two arithmetic passes and has six functions listed as follows:

1. (M)→X2 - The detailed transfer path is shown below.



The transfer of X1→I81X/82X toggles the bias so that bits 12-22 of X2 hold the real exponent at V503 time.

2. Difference of Exponent to SC Register - The difference of the exponents is formed by forcing the 48-bit adder to appear as an 11-bit adder. This is accomplished by setting K872/873 (add exponent 2) at V503 time which effectively causes groups 0-11 to appear as passes and not generates. Figure 338 is a simplified logic diagram of how group 12 of the adder is affected by the setting of add exponent 2. For the time interval defined by add exponent 2, U800 = 1 and U801 = 0. This insures that all inputs to U746, U747, and U748 are disabled, while all inputs to U749 are partially enabled. Therefore, group 12 only senses for carry inputs from groups 12, 13, 14, and 15. Figure 339 shows the data flow for subtract exponents. The operands are presented to the adder at V503 time of the first arithmetic pass of phase 2 and $\overline{\text{sum}}$ is sampled at V501 time of the second arithmetic pass of phase 2. At V503 time of the second arithmetic pass the complement of the required number of shifts is placed in SC register so that the octal points can be aligned before the subtract step. If the difference between exponents is greater than 77_8, I873 (Logic Diagrams, page 4-25 will block the transfer of Q3 SC register. Therefore, 77_8 shifts take place and the final answer will be the larger operand.
3. Larger Exponent to Q3 - At V505 time of the second arithmetic pass of phase 2 one of the following will occur:
   a. If the sum of subtract exponents was positive, A1 exponent is the larger exponent. Therefore, A1 exponent is gated to Q3 so that it will be available for adjust exponent. Note that if (A1) is positive, the real exponent is placed in Q3; if (A1) is negative, the complement of the real exponent is placed in Q3.
   b. If the sum of subtract exponents was negative, M exponent is the larger exponent. The complement of M exponent is in X2 from subtract exponents. Therefore, gating $\overline{\text{X2}}$ to Q3 places M exponent in Q3 so that it will be available for adjust exponent. Note that the real exponent is placed in Q3.
4. Smaller number to A1 - If the sum of subtract exponents was positive (M, M + 1) is smaller than (AQ) and must be right shifted to align the octal points. Since only AQE have shift capabilities (M1, M + 1) must be placed in AQ and (AQ) will be placed in X2X1. It is important to note that during sub-

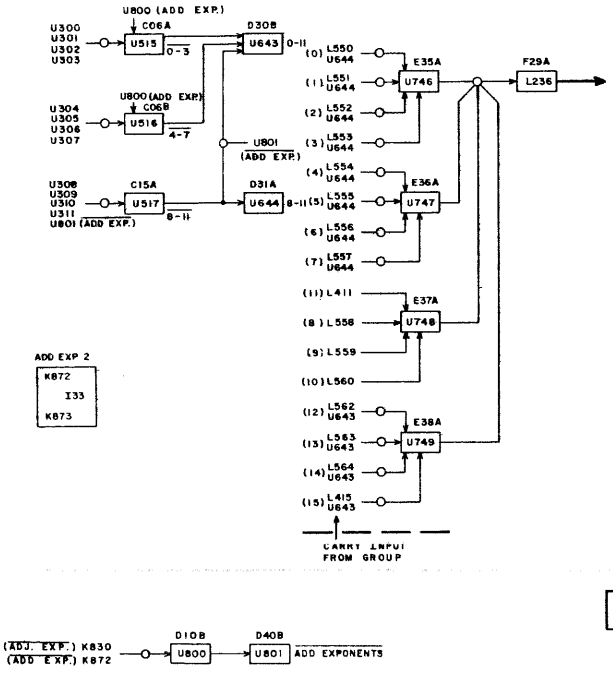


Figure 338. Enabling the 11-bit Adder

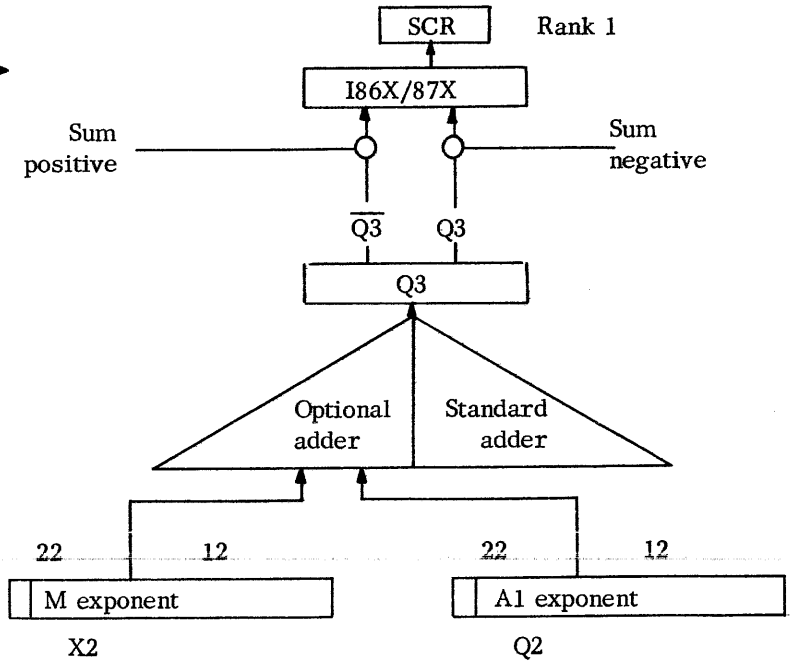


Figure 339. Enables for Difference of Exponents to SCR

tract exponents the standard adder is functioning as a 24-bit adder. The enables for placing (M) in A1 are:

Right adder → I^3 → Q2

Q2 → I^0 → A1

The exchange of (AQ) and (M, M + 1) will be completed during the first arith pass of phase 3.

5. Larger Number to X1 - If the sum of subtract exponents was positive, (M, M + 1) is smaller than (AQ) and must be right shifted to align the octal points. Since only AQE have shift capabilities (M, M + 1) must be placed in AQ (as in item 4). In order to make room for this transfer (AQ) must be placed in X2X1. During phase 2 (A) is transferred to X1 and then will be transferred sign extended into X2. The transfer path for A1 to X1 is:

$\overline{A1}$ → I^2 → X1 → X2

The exchange of (AQ) and (M, M + 1) will be completed during the first arithmetic pass of phase 3.

6. Extend Sign - At V505 time of the second arithmetic pass of phase 2 the operands are positioned with the upper 24 bits of the larger operand in X1 and the upper 24 bits of the smaller operand in Q2. The enables for extend sign are:

Lower 12 bits

$\overline{Q2}$ → I^0 → A1

Upper 12 bits

Nothing → I^0 → A1

(Q2) negative

Lower 12 bits

X1 → X2

Upper 12 bits

Nothing → I81X/82X → X2

(X1) negative

Phase 2: Second Arithmetic Pass

	If sum of subtract exponent positive, A exponent ≥ M exponent	If sum of subtract exponent negative, M exponent > A exponent.
V500: Clear Q3; input to H943.		
V501: Left adder → Q3 (extend bit 10); set K874/875 (extend sign); input to H944.	Set K818/819 (swap A1 and X2); input to H620.	
V502: Clear SC register; clear K872/873 (add exponent 2); input to H945.	Clear A2 and Q2; input to H513.	
V503: I86X/87X → SC register; set K814/815 (equalize exponent); clear K532/533 (unlike signs 1); input to H942.	$\overline{Q3}$ → I86X/87X; I^3 → Q2 (right adder I^3); input to H810.	Q3 → I86X/87X
V504: Clear Q3; input to H943.	Clear X2 and X1; Input to H511;	
V505: Input to H844 and H854.	I^2 → X1, ($\overline{A1}$ → I^2); A1 → Q3.	$\overline{X2}$ → Q3

V844, V854: Clear A1 and X2; input to H555 and H855; if X1 negative, input to H821.

V515, V855: I^0 → A1, bits 0-11 ($\overline{Q2}$ → I^0); if Q2 negative, I^0 → A1, bits 12-23 (nothing to I^0); X1 → X2, bits 0-11 and 23; if X1 negative, I81X/82X → X2, bits 12-22 (nothing to I81X/82X); set K820/821 (swap Q1 and X1); set K800/801 (wait word 2).

Phase 3: Initialize

N687: Start arithmetic coincident with V009 of second ROP;
input H500, H510, and H512.
clear K818/819 (swap A1 and X2).

V500: Clear A2, Q2, and X1; clear K874/875;
input to H531;
set K560/561 (arithmetic busy).

V501: $I^4 \rightarrow X1$ (DBR to I^4);
set K802/803 (word 2)
set K898/899 (short cycle);
clear K850/851 (bus $\rightarrow I^4$ enable).

V502: Set K802/803 (word 2); set K898/899 (short cycle); clear K850/851 (DBR $\rightarrow I^4$ enable).

V503:

V504: Set K804/805 (optional arithmetic busy);
transfer 1's count;
transfer 10s count.

V505: Set K814/815 (equalize exponent);
clear K820/821 (swap Q1 and X1);
input to H500, H600, H620, and H820.

If A exponent $\geq$ M exponent,

Input to H513.

$I^3 \rightarrow Q2$ ($\overline{Q1} \rightarrow I^3$);
Input to H524.

Clear Q1; input to H555.

$I^1 \rightarrow Q1$ ($\overline{X1} \rightarrow I^1$); input to H810.
Clear X1;
input to H531.

$I^4 \rightarrow X1$ ($\overline{Q2} \rightarrow I^4$)

The preceding arithmetic timing places (M + 1) in X1 at V501 time. At V504 time the shift count register is equalized so that it will count properly during equalize exponents. Remember that the complement of the required number of shifts was placed in rank 1 of the SC register during phase 2. Setting of optional arithmetic busy at V504 time enables the 48-bit adder. Setting of K560/561 at V500 time excludes main control from the arithmetic section until execution of this instruction is complete.

The enables shown above occur only if A exponent $\geq$ M exponent which required that (M, M + 1) be placed in AQ to allow shifting. During phase 2 (A1) was placed in X2 and (M) was placed in A1. This timing places (M + 1) in Q1 and (Q1) in X2.

Phase 3: Equalize Exponents

V500: Clear A2, Q2, and E2;
Clear K800/801 (wait word 2).

V503: Clear K802/803 (word 2).

V504

V600: If SC register $\neq$ 7X, input to H523, H561, and H881; if SC register = 7X, input to H513, H551, and H871.

V601: Input to H514, H524, and H834;
if SC register $\neq$ 7X:
$I^2 \rightarrow$ A2, right shift 10 ($\overline{A1} \rightarrow I^2$),
$I^3 \rightarrow$ Q2, right shift 10 ($\overline{Q1} \rightarrow I^3$),
E1 $\rightarrow$ E2, right shift 10;
advance 10s count.
If SC register = 7X:
$I^2 \rightarrow$ A2 ($\overline{A1} \rightarrow I^2$),
$I^3 \rightarrow$ Q2 ($\overline{Q1} \rightarrow I^3$),
E1 $\rightarrow$ E2.

V602: Clear A1, Q1, and E1; transfer 10's count.
if SC register $\neq$ X7, input to H525, H565, and H895.
If SC register = X7, input to H515, H555, and H885.

V505: Input H500, H600, H620, and H820;
if SC register ≠ 77, repeat shift cycle;
if SC register = 77, set K816/817 (add/subtract coefficient); clear K898/899 (short cycle).

The preceding arithmetic timing is a short cycle (4 Ø times), as K898/899 (short cycle) was set at V501 time of the previous arithmetic pass.

V603: If SC register ≠ X7:
$I^0 \rightarrow$ A1, right 1 ($\overline{A2} \rightarrow I^0$),
$I^1 \rightarrow$ Q1, right 1 ($\overline{Q2} \rightarrow I^1$),
E2 → E1, right 1; advance 1's count.
If SC register = X7:
$I^0 \rightarrow$ A1 ($\overline{A1} \rightarrow I^0$),
$I^1 \rightarrow$ Q1 ($\overline{Q2} \rightarrow I^1$),
E2 → E1.

The preceding timing is from the shift timing chain. At the completion of equalize exponents the coefficient of the smaller operand will be positioned in AQE sign extended. That portion of the operand contained in E is called residue and will be used to determine if a round operation is necessary. If the difference of exponents is $\geq 36_{10}$, AQ will contain only the sign extension.

Phase 3: Subtract Coefficients

V500: Clear A2, Q2, E2;
clear K814/815 (equalize exponents);
input to H611.

V501: $I^2 \rightarrow$ A2 ($\overline{Q1} \rightarrow I^2$);
$I^3 \rightarrow$ Q2 ($\overline{A1} \rightarrow I^3$);
set K884/885 (round up) if $A2^{23} = 0$ (A register positive) and $E2^{47} = 1$.

V502: Set K890/891 (round down) if $A2^{23} = 1$ (A is negative) and $E2^{47} = 0$.

V503: Input to H612.

V504: Clear A1Q1;
input to H613.

V505: $I^0 \rightarrow$ A1 (left adder $\rightarrow I^0$);
$I^1 \rightarrow$ Q1 (right adder $\rightarrow I^1$);
set K822/823 (round); input to H500, H620, and H810; input to H510 if round down.

During this arithmetic pass the coefficients are subtracted. Remember that, in order to subtract, the subtrahend must be complemented and addition performed. The complement of the subtrahend was obtained by setting K850/851 (DBR → I^4 enable) at V504 time of phase 1. This insures that (M, M + 1) is complemented as it is transferred into the arithmetic section. See figure 340 for data flow during subtract coefficients. During this pass the enables for round are determined by the setting or not setting of round up or round down. If the residue is $\geq 1/2$, a round is necessary.

Remember: round up forces a +1 to X2X1 and round down forces a -1 to X2X1. The adder propagates in both cases before transferring $\overline{sum} \rightarrow I^0$.

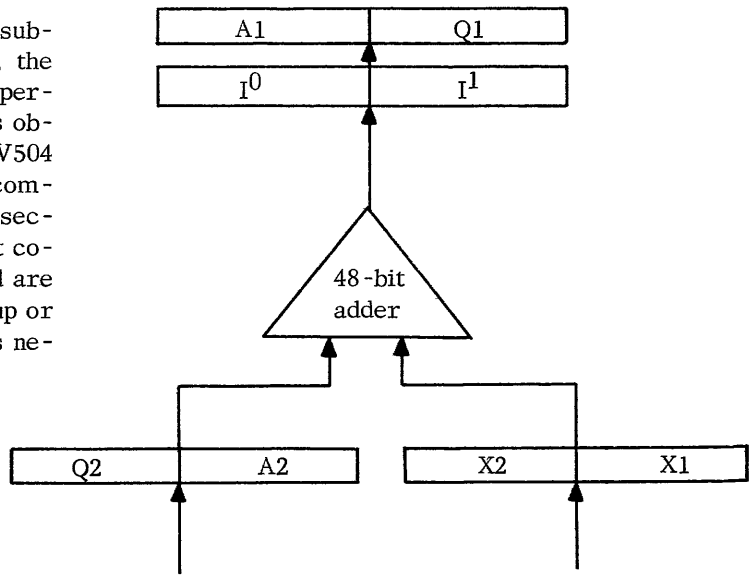


Figure 340. Enables for Subtract Coefficients

Phase 3: Round

V500: Clear A2, Q2, X1, and X2;
clear K816/817 (addition/subtraction coefficient); input to H611.

V501: $I^2 \rightarrow$A2 ($\overline{Q1} \rightarrow I^2$);
$I^3 \rightarrow$Q2 ($\overline{A1} \rightarrow I^3$);
input to H944.

V502: Clear SC register.

V503: Input to H612.

V504: Clear A1 and Q1;
set K898/899 (short cycle);
if sum $\neq$ -0, input to H613.

V505: If sum $\neq$ -0:
$I^0 \rightarrow$A1 (left adder $\rightarrow I^0$);
$I^1 \rightarrow$Q1 (right adder $\rightarrow I^1$);
set K826/827 (normalize);
remove clear from K880/881 and K886/887;
input to H500, H600, H620, and H820.

If round down,
input to H531 and H821.

$I^4 \rightarrow$X1 (nothing $\rightarrow I^4$), I400 $\rightarrow$ X000 blocked by round down, I81X/82X $\rightarrow$ X2 upper (nothing $\rightarrow$ I81X/82X); set X2 lower. This equals a -1 in X2X1.

If round up.

Force X2X1 to +1.

The timing for round occurs whether or not an adjustment of the coefficient must be made. Round can be +1 or -1 for the 61 instruction. For a +1 operation the difference of the coefficients is placed in Q2A2; X2X1 is forced to 0 $\rightarrow$ 01, and a 48-bit add is performed. For a -1 operation the difference is placed in Q2A2, X2X1 is forced to 7 $\rightarrow$ 76, and a 48-bit add is performed. If neither a +1 nor a -1 is required, X2X1 is cleared prior to the 48-bit add and the original difference is preserved. The timing for all floating-point instructions is common for normalize, adjust exponent, merge, and complement. Go to page 244 to complete the timing for this instruction.

Phase 3. Common Timing: Normalize
Set short cycle FF for normalize.

V500: Clear A2 and Q2.

V503: If K886/887 (left 10s shift enabled) is clear and A_{04-15} is all 1's or all 0's,
set K880/881 (left shift 10s);
if K880/881 (left 10s shift enabled)
is clear and $A_{11} = A_{12}$,
set K888/889 (shift 1's);
if K886/887 (left 10s shift enabled) is set
and Q_{20-23} and A_{00-05} is all 1's or all 0's,
clear K880/881 (shift left 10s);
if $A_{12} \neq A_{13}$, set K882/883 (shift right 1)
and set K888/889 (shift 1's);
if A_{10-13} is all 1's or all 0's,
set K878/879 (normalize complete);
If (AQ) = 0, set K878/879 (normalize complete);
transfer 1's count.

V504: Force transfer K880/881 to K886/887.

V600: If K880/881 (shift left 10) is set,
input to H523 and H561.
If K880/881 (shift left 10) is clear,
input to H513 and H551.

V601: If 10s enabled:
$I^2 \rightarrow$A2, left shift 10 ($\overline{A1} \rightarrow I^2$),
$I^3 \rightarrow$ Q2, left shift 10 ($\overline{Q1} \rightarrow I^3$);
advance 10s count.
If 10s enabled:
$I^2 \rightarrow$A2 ($\overline{A1} \rightarrow I^2$),
$I^3 \rightarrow$ Q2 ($\overline{Q1} \rightarrow I^3$).

V602: Clear A1 and Q1;
transfer 10s count;
if K888/889 (shift 1's) is set,
input to H525 and H565;
if K888/889 (shift 1's) is clear,
input to H515 and H555.

V505: Input to H500, H600, and H620.
If normalize complete:
set K830/831 (adjust exponent);
clear K898/899 (short cycle);
input to H810;
proceed to adjust exponent.
If normalize complete:
repeat shift cycle.

V603: If shift 1's and shift right 1:
$I^0 \rightarrow A1$, left 1 ($\overline{A2} \rightarrow I^0$),
$I^1 \rightarrow Q1$, left 1 ($\overline{Q2} \rightarrow I^1$);
advance 1's count.
If shift 1's and shift right 1:
$I^0 \rightarrow A1$, right 1 ($\overline{A2} \rightarrow I^0$),
$I^1 \rightarrow Q1$, right 1 ($\overline{Q2} \rightarrow I^1$);
advance 1's count.
If shift 1's:
$I^0 \rightarrow A1$ ($\overline{A2} \rightarrow I^0$),
$I^1 \rightarrow Q1$ ($\overline{Q2} \rightarrow I^1$).

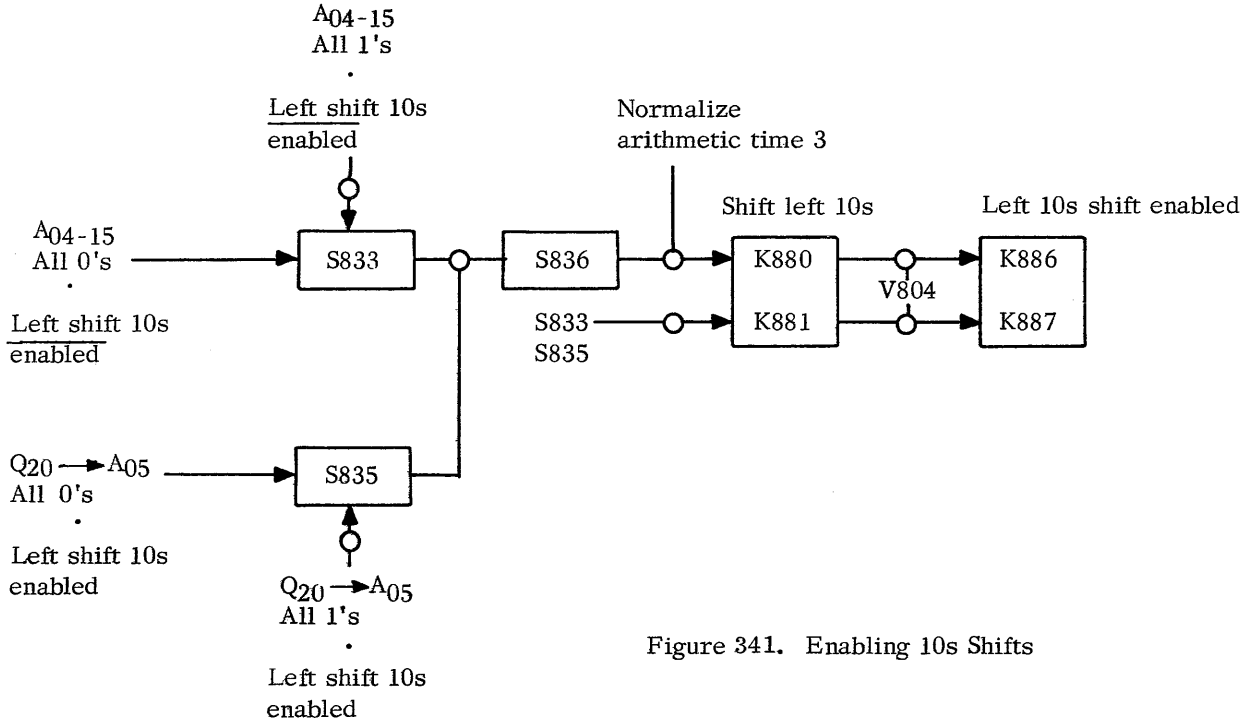


Figure 341. Enabling 10s Shifts

The timing for normalize is common to all floating-point instructions and consists of passes through the arithmetic and shift timing chains. The arithmetic passes are short cycles (4 Ø times) since short cycle was set at V504 time of round. The main items for normalize are:

1. Right shift will never be more than 1 place and could occur for a 60, 61, or 63 instruction. $A_{12} \neq A_{13}$ indicates that a right shift is necessary.
2. Left shift can be any number of shifts less than 36_{10} and could occur for any floating-point instruction.
3. 10s shifts will always be left and will always be completed before any 1's shifts take place. Figure 341 is a simplified logic diagram of setting shift left 10s. Note that S835 is forced to a 1 on the first pass by K886/887 and that S833 is forced to a 1 on all following 10s passes.
4. Normalize complete is defined by either one of two conditions: AQ = 0 or A_{10-13} all alike.

Phase 3: Adjust Exponent

V500: Clear A2, Q2, X1, and X2;
clear K826/827 (normalize);
if no shift or left shifts were required during normalize, input to H821.

V501: $I^3 \rightarrow Q2$ ($A1 \rightarrow I^3$), bits 0-11 and 23;
I86X/87X → Q2, bits 12-22;
if (Q3) is negative, $\overline{Q3} \rightarrow$ I86X/87X;
if (Q3) is positive, $\overline{Q3} \rightarrow$ I86X/87X.

V502

V503: Input to H942.

V504: Clear Q3;
input to H943.

V505: $\overline{Sum} \rightarrow Q3$, set bit 11 of Q3;
if exponent sum = -0 (A negative) or exponent sum ≠ -0 (A positive),
input to H500, H620, and H820;
set K834/835 (merge).

If left shift to normalize,	If right shift to normalize,
Set X620/621. (X2 register bits 12-22 set to a +1.)	I81X/82X → X2, bits 12-22; SC register → I81X/82X, lower 6; nothing → I81X/82X, upper 6. (Complement shift count to adder via X2 register.)

The following items are of note for adjust exponent:

1. If no shifts were required during normalize the real exponent is added to all 0's and gated back to Q3.
2. If a right shift was required during normalize, adjustment of the exponent requires a +1. This is accomplished by forcing bits 22-12 of X2 to 0 → 01 and performing an addition with the real exponent.
3. If the shift was left to normalize the shift count must be subtracted from the real exponents to adjust the exponent. This is accomplished by gating the complement of the shift count to X2 at V501 time and performing an addition with the real exponent.

At V505 time of the last pass of normalize K830/831 (adjust exponent) was set which enables the 48-bit adder to function as an 11-bit adder. K830/831 has the same effect on the adder as K872/873 (add exponent 2).

Phase 3: Merge

V500: Clear A2, Q2 and E2;
clear K830/831 (adjust exponent);
input to H611 and H871.

V501: $I^2 \rightarrow A2$ ($\overline{Q1} \rightarrow I^2$);
$I^3 \rightarrow Q2$, bits 0-11 and 23 ($\overline{A1} \rightarrow I^3$);
I86X/87X → Q2, bits 12-22;
if (Q3) positive ($\overline{Q3} \rightarrow$ I86X/87X);
if (Q3) negative ($\overline{Q3} \rightarrow$ I86X/87X).

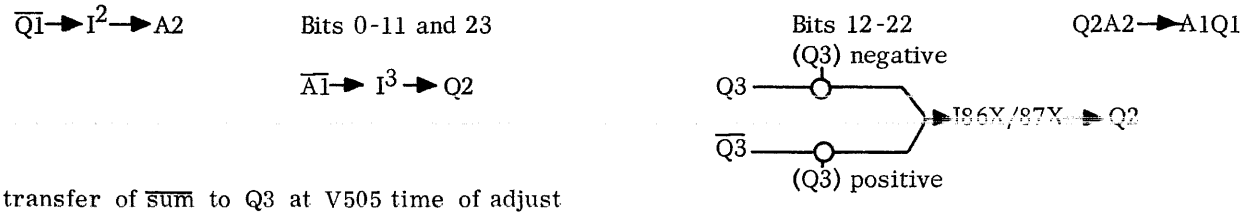
V502

V503: If AQ ≠ 0, input to H612.

V504: If AQ ≠ 0, clear A1Q1; input to H613.

V505: If AQ ≠ 0, $I^0 \rightarrow A1$ ($\overline{Q2} \rightarrow I^0$),
$I^1 \rightarrow Q1$ ($\overline{A2} \rightarrow I^1$);
set K852/853 (complement AQE or last cycle);
input to H500, H620, and H820.

The data flow for merge is:



The transfer of $\overline{sum}$ to Q3 at V505 time of adjust exponent toggled the bias bit so that the biased exponent is available for merge.

Phase 3: Complement

V500: Clear A2, Q2, and E2; clear K804/805 (optional arithmetic busy); input to H611, H871.

V501: $I^2 \rightarrow$ A2 (A1 $\rightarrow I^2$);
$I^3 \rightarrow$ Q2 (Q1 $\rightarrow I^3$);
E1 $\rightarrow$ E2.

V502

V503:

	If K532/533 is set (unlike signs),	If K552/553 (sign of A) is set and 63 instruction or if K532/533 (unlike sign 1) is set and 62 instruction,
	input to H612.	input to H834.
V504: Clear K804/805 (optional arithmetic busy).	Clear A1 and Q1; input to H613.	Clear E1; input to H915.
V505: Input to H864.	$I^0 \rightarrow$ A1 ($\overline{A2} \rightarrow I^0$); $I^1 \rightarrow$ Q1 ($\overline{Q2} \rightarrow I^1$).	$E_{inv} \rightarrow$ E1 (E2 $\rightarrow E_{inv}$).

V864: Clear K560/561 (arithmetic busy).

The complement of AQ would occur for floating-point multiply or divide when the signs of the coefficient were unlike. A complement of AQ will not occur for floating-point add or subtract. A complement of E would occur for a floating-point divide when the dividend was negative. K552/553 (sign of A) which indicates a negative dividend was set during copy F1 $\rightarrow$ F2 time of RNI if A23 was set. A complement of E would occur for a floating-point multiply if the product were to be negative. The clearing of arithmetic busy at V684 time frees the arithmetic section so that it may be used by main control.

PROBLEMS

1. May normalize be either a right or left shift for the 62 instruction? Why or why not?

2. Assume that (AQ) is negative. During phase 1, A1 exponent Q3 occurs. This places the complement of the real exponent in Q3. How are add exponents accomplished with read exponents in this case?

3. How is 2^{47} of the adder affected during add exponents?

4. At the completion of the multiply step where is the product? Can you predict what the lowest 12 bits of E1 will be?

5. Why is +1 the only possibility for round of the 62 or 63 instructions?

SELF-EVALUATION QUIZ ON CHAPTER 13

TRUE OR FALSE OR FILL IN THE BLANKS:

1. The FP/DP option is located in chassis _____.
2. The FP/DP option is necessary to execute the _____, _____, and _____ through _____ arithmetic instructions.
3. Primarily, the FP/DP option consists of a 48-bit adder and the 48-bit E register.
4. The optional adder is enabled by setting of _____ _____.
5. The optional adder may function as an 11-bit adder.
6. The multiplier for an MUAQ instruction is the quantity in AQ.
7. The first pass of multiply step for the MUAQ instruction must be a long cycle.
8. The multiply step of the MUAQ instruction requires 60_8 passes.
9. Complement is optional in the execution of the MUAQ instruction.
10. After the first multiply step of the MUAQ instruction, the multiplier bit is sensed at 2^1 of E2.
11. The amount of time necessary to execute a DVAQ depends on the number of 0 bits in the divisor.
12. The four steps required for execution of a DVAQ are initialize, _____, _____, and _____.
13. _____ arithmetic passes are required for divide step of a DVAQ.
14. The highest order bit of E is inspected to determine whether or not a round is necessary during the FMU.
15. Normalize for the FMU will always be right shift if any shifts are required.
16. If the product of an FMU has a negative coefficient and a positive exponent, complement will not occur.
17. Divide fault cannot occur during execution of the FDV since the hardware can normalize.
18. The function of pass 38_{10} of divide step on a FDV is to allow sensing for round.
19. Round on a FDV will always be -1.
20. If _____ is set, the optional adder will function as an 11-bit adder.

Score Yourself

This quiz was a good one. You may not agree, but you're entitled to that.
None wrong is way above average.
One wrong is above average.
Two wrong is average.
Three wrong--you're slipping!
Four or more wrong--you've slipped--into failure!

CHAPTER 14

INTERRUPT

GENERAL DESCRIPTION

The interrupt feature of the 3300 Computer System allows automatic sensing for specific internal or external conditions whose existence requires special action on the part of the program. To recognize an interrupt condition as soon as possible, the hardware senses for interrupt during RNI and RADR. If an interrupt is recognized, execution of the main program is terminated, the contents of P are stored at a fixed location, and program control is transferred to a fixed location.

TYPES

There are three major groups of interrupts in the 3300 system. These groups listed in order of priority are:

1. Abnormal interrupts.
2. Normal interrupts.
3. Trapped instruction interrupts.

ABNORMAL INTERRUPTS

There are three interrupt conditions that are classified as abnormal. Listed in order of priority these conditions are:

1. Interrupt on storage parity error or storage not available.
2. Interrupt on illegal storage reference.
3. Interrupt on power failure.

These interrupts are not masked and the interrupt system need not be enabled to recognize one of these interrupts.

NORMAL INTERRUPTS

In general, to recognize one of these interrupts the interrupts system must be enabled, the mask bit must be set, and the interrupt counters must have counted to the proper count. The exceptions to these conditions are:

1. Associated processor and manual interrupts are not mask.
2. Executive interrupt is exclusive of all three stated conditions.

External line interrupts and channel interrupts are equal in priority, that is, if the channel counter is active the line counter cannot be started; if the line counter is active, the channel counter cannot be started. To assign a priority to the normal interrupts, starting of the interrupt counter must be used as a time reference. The priority becomes:

1. Executive interrupt.
2. Arithmetic overflow or divide fault.
3. Floating point fault or BDP fault.
*4. External line or channel interrupts.
5. Search/move interrupt.
6. Real time clock interrupt.
7. Manual interrupt.
8. Adjacent processor interrupt.

TRAPPED INSTRUCTION INTERRUPT

The trapped instruction interrupt occurs whenever an instruction that requires optional hardware is read into F and the optional hardware is not present. This interrupt is not masked and the interrupt system need not be enabled to recognize this interrupt.

ABNORMAL INTERRUPT SEQUENCE

INTERRUPT ON STORAGE PARITY ERROR OR STORAGE NOT AVAILABLE

This is the highest priority interrupt in the 3300 system and does not require the interrupt to be enabled. Interrupt on storage parity error is switch-selectable from the console and recovery from this fault may not be possible.

If the system is operating non-Executive when this interrupt is recognized, the hardware interrupt sequence will store the (P) in the lower 15 bits of address 00020_8. The upper 9 bits of address 00020_8 are not altered by this operation. An identifying code is stored in the lower 12 bits of address 00021_8 and then RNI at address 00021_8.

If the system is operating Executive when this interrupt is recognized, the hardware interrupt sequence will:

1. Disable the Condition register.
2. Transfer to Monitor State.
3. Store the (P) in the lower 15 bits of address 000020_8; the upper 9 bits of address 000020_8 are not altered.
4. Store an identifying code in the lower 12 bits of address 00021_8.
5. RNI at address 000021_8.

ILLEGAL STORAGE REFERENCE INTERRUPT

This is the second highest priority interrupt in the 3300 system and does not require the interrupt system to be enabled. If the system is operating Non-executive, this interrupt does not exist.

If the system is operating Executive, there are four conditions which will generate an Illegal Storage Reference interrupt. They are:

1. Program State 0 and the STORAGE PROTECT switches compare with the lower 15 bits of the address placed on the S bus during a Write. This condition is sensed in Main Control.
2. Program State and the Page Index referenced has E = 1 and all other bits = 0. This condition is sensed in the Multiprogramming module and the interrupt would occur for either a Read or a Write.
3. Program State and the Page Index referenced has E = 1, any other bit = 1, and a Write is specified. This condition is sensed in the Multiprogramming module.
4. Program State and the available page length as specified by PL is exceeded. This condition is sensed in the Multiprogramming module and would occur for a Read or Write.

When this interrupt is recognized the hardware interrupt sequence will:

1. Disable the Condition register.
2. Transfer to Monitor State.
3. Store the (P) in the lower 15 bits of address 000014_8; the upper 9 bits of address 000014_8 are not altered.
4. RNI at address 000015_8.

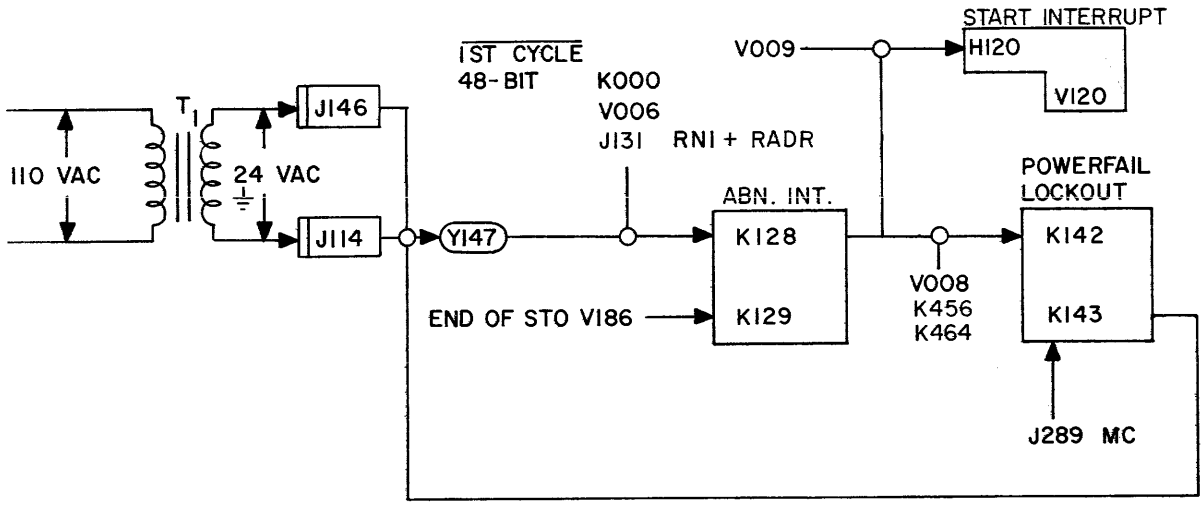
POWER FAIL INTERRUPT

This is the third highest priority interrupt in the 3300 system and does not require the interrupt system to be enabled. The purpose of this interrupt is to allow the computer to save any pertinent data before shutdown due to a loss of line voltage. The logic for detecting a power failure is shown in figure 342.

Source power of 110 vac provides power to the blowers in all the basic electronics cabinets. When this 110v power is brought up, a 24v holding relay (located

*Regardless of channel, a lower-numbered line has priority over a higher-numbered line; a lower-numbered channel has priority over a higher-numbered channel.

Figure 342. Power Failure Detection



on the power panel) is energized. This relay holds the power up after the POWER ON switch is released. The 24 volts from the relay transformer is applied to J146 and J147 to normally prevent them from both outputting 1s. If the 110 vac source power should drop, the input voltage to J146 and J147 falls from 24 vac to 0 vac, allowing both inverters to output 1s. The Abnormal Interrupt FF is then set at V006 time RNI (provided that the RNI is not the second RNI sequence for a 71 to 76 instruction) or at V006 of RADR.

Once Abnormal Interrupt has set, Powerfail Lockout will be set to prevent another input to the Abnormal Interrupt FF. A master clear must then be performed to again activate the powerfail detection circuit.

If the system is operating non-Executive when this interrupt is recognized, the hardware interrupt sequence will store the (P) in the lower 15 bits of address 00002_8 and RNI at address 00003_8. The upper 9 bits of address 00002_8 are not altered by this operation.

If the system is operating Executive when this interrupt is recognized, the hardware interrupt sequence will:

1. Disable the Condition register.
2. Transfer to Monitor State.
3. Store the (P) in the lower 15 bits of address 00002_8.
4. RNI at address 00003_8.

ABNORMAL INTERRUPT SEQUENCE

Timing begins with the Storage Reply during RNI or RADR.

V061: Resynced Storage Reply
V000: Clear K000/001 (Request Bus)
V001
V002
V003
V004: Set K128/129 if P.E. or illegal storage reference
V005
V006: If Power Failure Set K128/129 (Abnormal Interrupt)
V007: Clear K010/011 (Main Control Priority), K012/013 (Storage Request Lockout), and K116/117 (Storage Request)
V008: EXX2 to F, if (Power Failure) · $\overline{\text{(Illegal Storage Reference)}}$ · $\overline{\text{(Parity Error)}}$, set K142/143 (Powerfail Lockout).
V009: Input H120
V120: Set K086/087 (STO), K106/107 (Jump), and K124/125 (Special STO Cycle). Clear K080/081 (RNI) and K082/083 (RADR). Input H201; transfer Condition Register to Subcondition Register.
N120: Set Disable I FF K094/095.
N205: Clear IF lower 18; Input H122

N122: Clear K058/059 Program State II
V122: Set K000/001 (Request Bus) if Illegal Storage Reference; set K458/459 (Illegal Write Clear Enable); force F lower 15 to the appropriate interrupt address (00002 + 00014 + 00020).
N051: Set K010/011, transmit address → F lower 15 → EXX8 → T6XX → S bus; transmit Write signal and designators of 10011_2. Note that Storage Protect comparison is disabled.
N050: Input H117.
V117: Set K116/117 (Storage Request), set K212/213.
N050: Input H115.
V115: Set K012/013 (Storage Request Lockout); transmit $\overline{P}$ on the Data bus.
V116: Test Breakpoint Stop if BPO is selected.
V061: Resynced Storage Reply.
V000: Clear K000/001 (Request bus).
V001-V005: Access time.
V006: Set K126/127 (Second STO).
V007: Set F000/001; clear K498/499 (Interrupt Enabled)--this clears the line, Channel and Interrupt counters (if $\overline{\text{EXEC}}$ and $\overline{\text{Parity Error}}$); Clear K496/497 (Interrupt Detected) if K498/499 cleared; input H086.
V086, V186: Clear K086/087 (STO), K124/125 (Special STO Cycle), K126/127 (Second STO Cycle), K128/129 (Abnormal Interrupt), K464/465 (STO Parity Error); if Illegal Storage Reference Interrupt, clear K456/457 (Illegal Write); input H087.

For Storage PE interrupt, a second Store Cycle is required to store the identifying interrupt code. For the required timing refer to the Normal interrupt timing (time V007) on page 26.* The code stored at address 00021 varies with the type of sequence in progression and what control section had priority at time of interrupt.

V087: Input H014.
V014, N014: RNI at the interrupt address since Jump is set.

NORMAL INTERRUPT SEQUENCE

The normal interrupts are treated as a group due to the fact that the hardware interrupt sequence processes all of these interrupts in the same basic manner. For all of these interrupts, the following takes place during the hardware interrupt sequence:

1. The (P) is stored in the lower 15 bits of address 00004_8.
2. An Identification Code (I.D. code) is stored at address 00005_8.
3. RNI is performed at 00005_8.

It is the I.D. code which distinguishes between the normal interrupts. Table 21 lists the I.D. codes.

Parity Error Interrupts (Abnormal Interrupt) also use the Second STO Cycle to store an ID code. Table 22 lists the Parity Error Interrupt Codes.

Table 21. REPRESENTATIVE INT. I.D. CODES

Conditions	Codes
External interrupt	001Ch
I/O channel interrupt	010Ch
Realtime clock interrupt	0110
Arithmetic overflow fault	0111
Divide Fault	0112
Exponent overflow fault	0113
BCD fault	0114
Search/move interrupt	0115
*Manual interrupt	0116
*Associated processor interrupt	0117
*Executive interrupt	0120

*These Interrupts are not masked.

*See 3300 Command Timing

Table 22. PARITY ERROR INTERRUPT CODES

Reason for Interrupt	Priority	Operation	Code
Non-existent Memory	Block Control	73-76	00X0 X=ch
Parity Error	Block Control	73-76	00X2 X=ch
Non-existent Memory	Block Control	71, 72, or typewriter I/O	01X0 (X=0, Search), (X=1, Move), (X=3, TWR)
Parity Error	Block Control	71, 72, or typewriter I/O	01X2 (X=0, Search), (X=1, Move), (X=3, TWR)
Non-existent Memory	Main Control	RNI or RADR	00X1 (X=0, RNI), (X=2, RADR)
Parity Error	Main Control	RNI or RADR	00X3 (X=0, RNI), (X=2, RADR)
Non-existent Memory	Main Control	ROP or STO	0005
Parity Error	Main Control	ROP or STO	00X7 (X=0 or 1*)

*If X=1, the Parity Error occurred on the 00.7 STO Cycle.

EXECUTIVE INTERRUPT

The Executive interrupt occurs only if the system is operating Executive and Program State and one of the following instructions is executed.

```
00.0
53. (4 → 7)(1 → 3)(XX00 → XX37)
71.X
72.X
73.X
74.X
75.X
76.X
77.X but not 77.71 or 77.72
```

The interrupt is removed when the hardware interrupt sequence transfers to Monitor State. Executive interrupt is the highest priority of the normal interrupts and the interrupt system need not be enabled to recognize this interrupt. Also it is not masked and does not reference the interrupt counters. When this interrupt is recognized, the following events occur:

1. Disable the Condition register.
2. Transfer to Monitor State.
3. Store (P) in the lower 15 bits of address 000004_8.
4. Store the ID code in the lower 12 bits of address 000005_8.
5. RNI at address 000005_8.

OPTIONAL ARITHMETIC INTERRUPTS

An interrupt may be produced by a floating-point fault or a BCD fault. The fault is detected by the optional arithmetic hardware (when present) or, in the absence of the hardware, by the Simulator routine.

FLOATING-POINT FAULT

The Floating-point Fault FF, K420/421, is set to indicate that a floating-point fault has occurred. It may be set:

1. By the AND gate of V874 and S883 being made when floating-point hardware detects the fault, or
2. When instruction 77.71, Set Floating-Point Fault, is executed. The Simulator routine, used in lieu of hardware, recognizes any condition that would cause a fault and executes this instruction to set the FF. Thus, an interrupt may occur in the same manner as it would if floating point hardware were present in the computer.

BCD FAULT

The BCD Fault FF, K418/419, is set to indicate that a BCD fault has occurred. It may be set:

1. When a BCD fault has been detected by the BCD hardware, or
2. When instruction 77.72, Set BCD Fault, is executed. This instruction is also used by the Simulator routine.

INTERRUPT ADJACENT PROCESSOR

Instruction 77.57, Interrupt Adjacent Processor, allows two processors to interrupt each other.

When the function code of the instruction has been decoded, T148 outputs an Interrupt Adjacent Processor signal. This signal is received by the adjacent processor and sets K408/409 in the other processor. This FF remains set until the interrupt is recognized.

INTERNAL INTERRUPTS

Eight internal conditions may be set to cause an interrupt. These conditions are: Executive Interrupt, Arithmetic Overflow fault, Divide fault, Exponent Overflow fault, BCD fault, I/O channel interrupts, Search/Move interrupt, and Real Time Clock interrupt.

EXTERNAL INTERRUPTS

Three external conditions may cause interrupts. These are: External I/O interrupts (generated by a piece of I/O equipment), Manual interrupt (set by a switch on the console or a switch on the console typewriter), and the Associated Processor interrupt.

INTERRUPT MASK REGISTER

The programmer can choose to honor or ignore most normal interrupts by means of the Interrupt Mask register. The Mask register is a 12-bit, single-rank FF register with inputs from the lower 12 bits of the F register. The mask bit representing an interrupt condition must be set to a 1 for that interrupt condition to be recognized by the sensing network. The mask is selectively set with instruction 7752XXXX, and selectively cleared by instruction 7753XXXX (XXXX represents the mask bits). Master Clear does not affect the mask register. Table 23 gives the mask bit assignments.

Table 23. INT. MASK REGISTER BIT ASSIGNMENTS

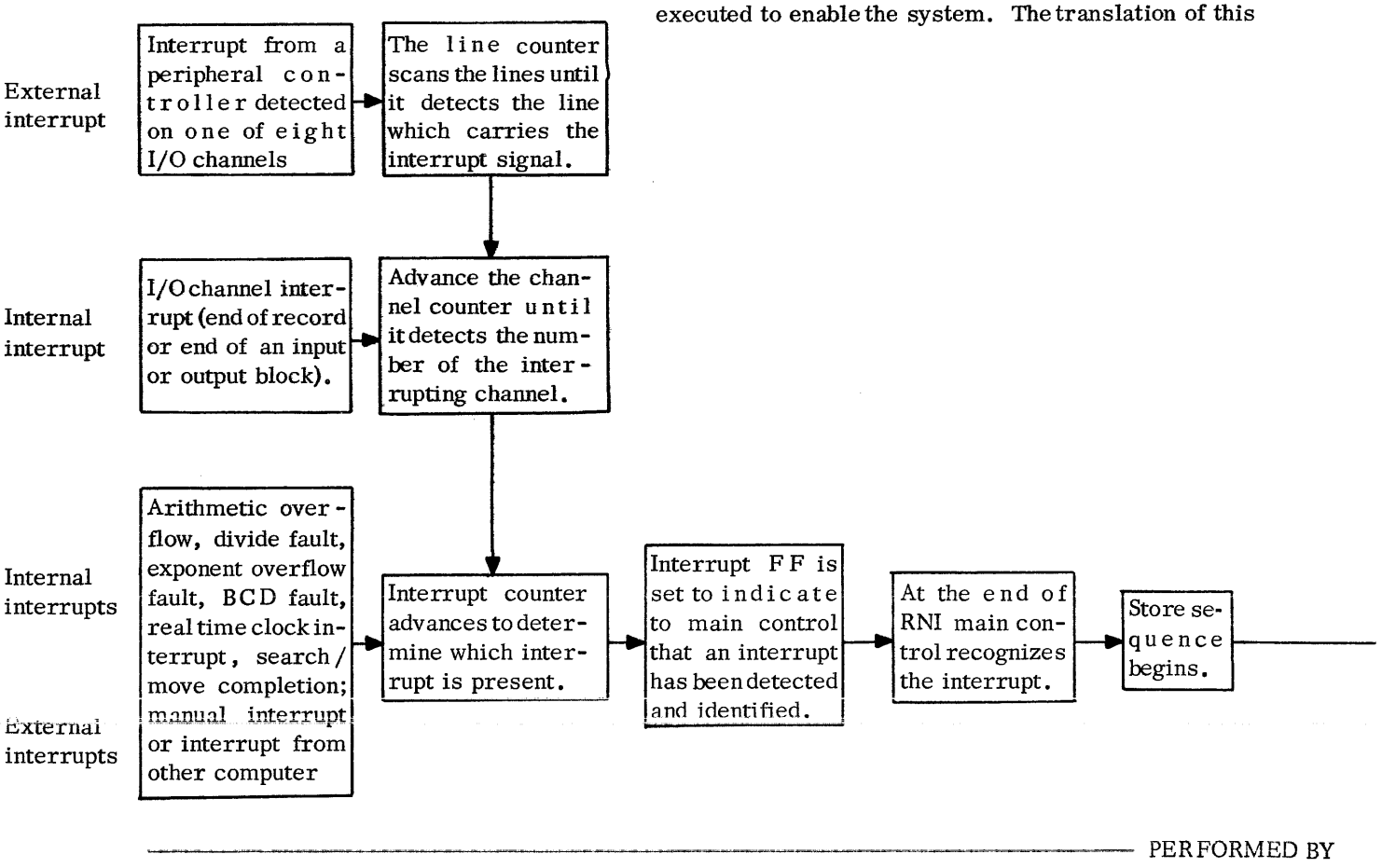
BIT POSITIONS	DEFINITIONS
00-07	I/O channel 0-7 interrupts (internal and external)
08	Clock interrupt
09	Exponent overflow or BCD fault
10	Arithmetic overflow or divide fault
11	Search/move completion interrupt

The manual interrupt and the associated processor interrupt are not masked and thus will always be recognized provided the interrupt system has been enabled. The interrupt system must be enabled for any normal interrupt except Executive to be recognized.

INTERRUPT SELECTION

The programmer has master control over the normal interrupt system. Instruction 7774---- must be executed to enable the system. The translation of this

Figure 343. Interrupt Cycle Flow Chart



instruction by M413 sets K454/455 at the end of the RNI sequence. The output of this FF, in turn, sets K498/499, the Enable Interrupt FF. In the clear state, this FF disables interrupt recognition by holding the inputs to the interrupt counters down.

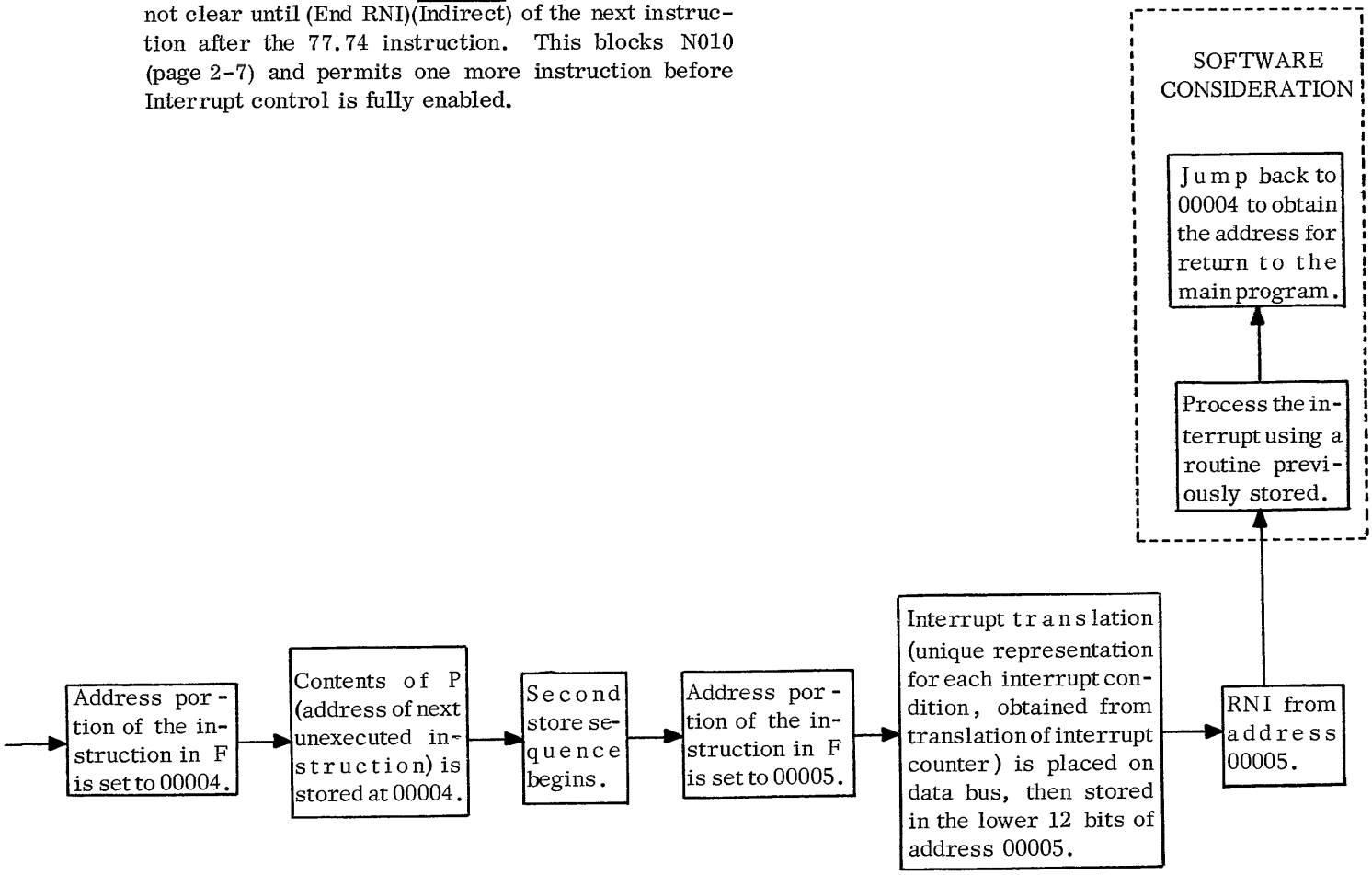
Instruction 7773---- clears Enable Interrupt. This FF is also cleared by J125 which comes up when Abnormal Interrupt, $\overline{PE}$, and $\overline{Disable}$ (J855) = 1. (The interrupt section has identified an interrupt before main control enters an interrupt cycle.) Thus, while the main control section is processing an interrupt, all other normal interrupts are disabled. When leaving the interrupt subroutine, the program must execute another 7774---- instruction to again enable the interrupt system.

The timing on setting Enable Interrupt is such that the interrupt system is not enabled immediately following the 7774---- instruction. Thus, after executing a 7774---- instruction, one more instruction is executed before another interrupt can occur. This insures that the computer will return to the original program before the next interrupt. If this were not the case, and interrupts were to follow each other immediately, the original return address might be lost. K455 does not clear until (End RNI)($\overline{Indirect}$) of the next instruction after the 77.74 instruction. This blocks N010 (page 2-7) and permits one more instruction before Interrupt control is fully enabled.

INTERRUPT RECOGNITION

Before an interrupt can be processed, the interrupt section must recognize the interrupt and determine its origin.

An order of priority exists between the various interrupt conditions. When an active interrupt is received, the priority scanner, composed of three counters (Line, Channel and Interrupt counter), begins checking the priority list. The sensing networks compare the priority count with the active interrupt. When the Interrupt counter starts a resync network is pulsed and the output of the resync sets K496/497, Interrupt detected. This signals main control that an interrupt condition has been detected and identified. Figure 344 is a flow chart of the steps in interrupt recognition.



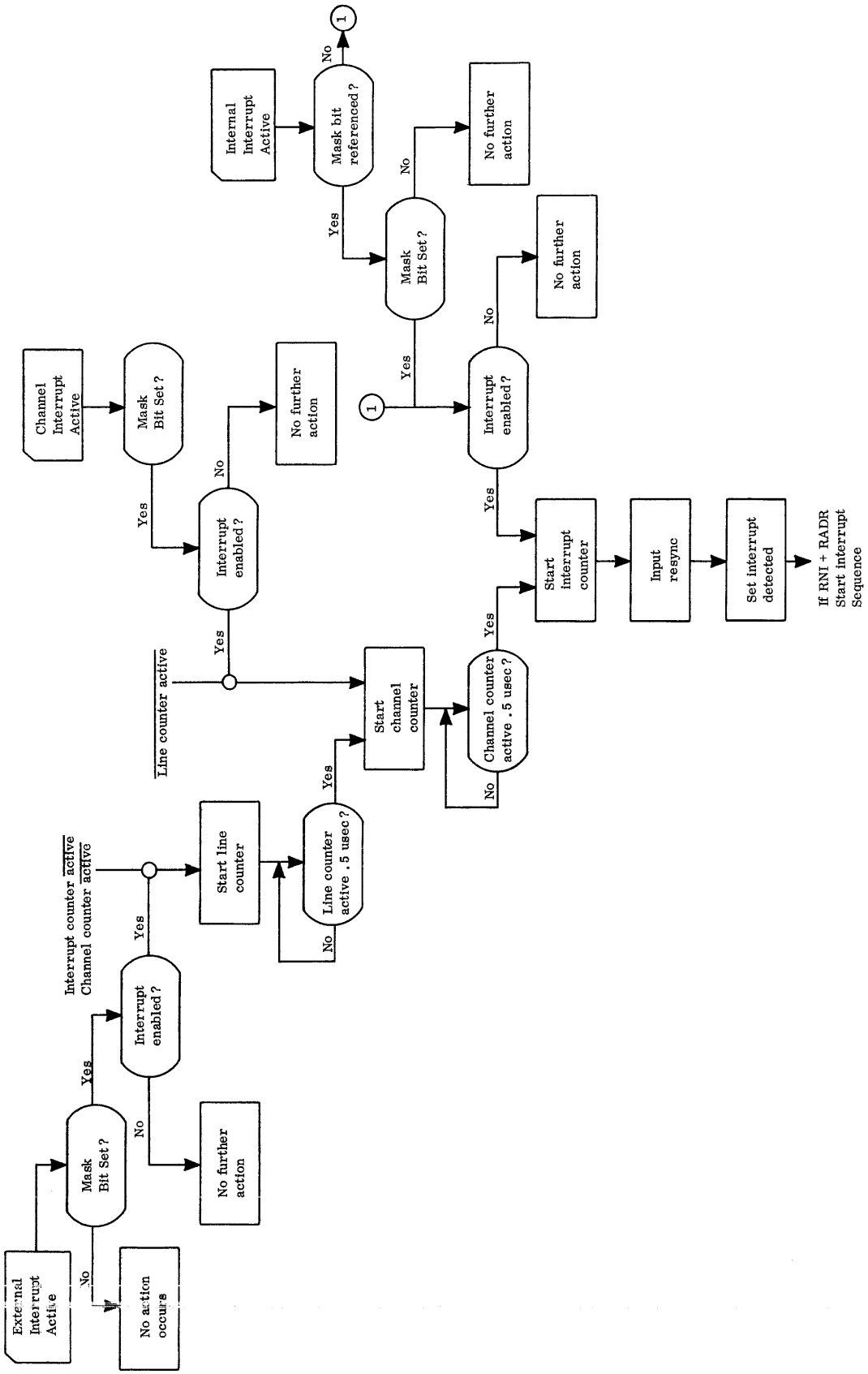


Figure 344. Interrupt Recognition Flow Chart

NORMAL INTERRUPT SEQUENCE

Timing begins with N010 which could be V010 time of RNI or V008 time of RADR.

N010: If Interrupt has been detected, set K496/497; set K120/121 (Interrupt Sync). Note that this blocks End RNI and End RADR pulses.
N053: Input H120.
V120: Set K086/087 (STO), K106/107 (Jump), K124/125 (Special STO), and K094/095 (Disable I); clear K080/081 (RNI) and K082/083 (RADR); Input H201.
N053, N205: Clear F; Input H122.
V122: Clear K058/059 (Program State II); Set K000/001 (Request Bus), and F020/021.
N051: Set K010/011 (Main Control Priority); transmit address 000004, F to EXX8 to T6XX to S bus; transmit Write signal and Write designators of 10011_2.
N050: Input to H117.
V117: Set K116/117; transmit a Storage Request.
N050: Input to H115. NOTE: storage protect comparison is disabled.
V115: Set K012/013, Transmit $\overline{P}$ on Data Bus ($\overline{P}$ to T5XX).
V116: Test Breakpoint Stop if BPO selected.
V061: Resynced Storage Reply.
V000: Clear K000/001.
V001-V005: Access Time.
V006: Set K126/127 (Second STO). Timing from this point on also applies to Parity Error Interrupt Sequence.
V007: Clear K010/011, K012/013, K116/117; input to H086, input to H122, set F000/001.
V122: Block output from V086 and V186 (J063); set K000/001.
N051: Set K010/011 if K210 = 1; transmit address 00005, F to EXX8 to T6XX to S bus; transmit Write signal and Write designators of 00011_2.
N050: Input to H117.
V117: Set K212/213, K116/117; transmit a Storage Request.
N050: Input to H115.
V115: Set K012/013 (Enable $\overline{DBR}$ to T5XX); input H116, H400.
V116, N400: Test Breakpoint Stop if BPO selected; input to H401. NOTE: Storage Protect Comparison is Disabled.
N401: Clear DBR; input to H410.
N410: EXX2 to DBR, ($\overline{ID.}$ to EXX2 to DBR to T5XX to Data bus).
V061: Resynced Storage Reply.
V000: Clear K000/001.
V001-V004: Access Time.
V005: Clear K124/125 (Special STO), J063 = 0; clear K464/465 Storage Parity Error (if PE sequence).
V006: Clear K498/499 (Enable Interrupt) via J125.
V007: Clear K010/011, K012/013, K116/117; input to H086.
V086, V186: Clear K126/127 (Second STO), K120/121 (Interrupt Sync), K086/087 (STO); set K080/081 (RNI); input to H087.
V087: Input to H014.
V014: Jump and RNI at address 00005.

TRAPPED INSTRUCTION INTERRUPT

All commands in the instruction repertoire of the 3300 Computer System may be used regardless of the size of the computer. However, a basic computer lacking the optional floating-point 48-bit precision and/or the BCD package is not capable of directly processing instructions that require this hardware. These instruction, implemented by software, are called trapped instructions.

Trapped instructions fall into two groups, those that are trapped when the floating-point 48-bit precision option is missing and those that are trapped when the BCD option is absent. Table 24 lists the instructions which may be trapped.

Table 24. LIST OF TRAPPED INSTRUCTIONS

CODE	MNEMONIC	DESCRIPTION
Floating-point 48-Bit Precision Package Missing		
55	----	IRT, 48-bit precision
56	MUAQ	Multiply AQ, 48-bit precision
57	DVAQ	Divide AQ, 48-bit precision
60	FAD	Floating-point add
61	FSB	Floating-point subtract
62	FMU	Floating-point multiply
63	FDV	Floating-point divide
BDP Package Missing		
64	LDE	Load E
65	STE	Store E
66	ADE	Add to E
67	SBE	Subtract from E
70	SFE	Shift E
	EZJ, EQ	E zero jump (E = 0)
	EZJ, LT	E zero jump (E = 0)
	EOJ	E overflow jump
	SET	Set D register

Translator J118 (figure 345) determines when an instruction is to be trapped. When the function code of any optional instruction has been decoded the left AND gate to J118 is broken. The other two AND gates then test whether the necessary optional arithmetic package is present. If it is present there will be no interrupt and the instruction is handled by the hardware. If the necessary hardware is not present, J118 outputs a 1 which sets trap sync FF at the end of RNI. A trapped instruction interrupt then proceeds in much the same manner as a normal interrupt.

The steps in the trapped instruction interrupt cycle are:

1. At V010 time of RNI a test is made for interrupt.
2. RNI ends; trap sycn FF is set if J118 indicates a trap cycle necessary.
3. Advance P2 to set the address of the next instruction in P2.
4. Start arith 1, start arith 2, no index, RNI, RADR, and ROP FFs are cleared. STO, special store cycle, and jump FFs are set. This sets up initial conditions for the trap cycle.
5. The lower bits of the F register are cleared.
6. (P2) is transferred to P1.
7. Set request bus FF.
8. F030/031 of F register is set. This places address 00010 in F.
9. Main control obtains bus priority. A write signal, write designators of 10011_2 (word address), and the storage address (00010) are transmitted to storage.
10. A storage request is sent to the selected module.
11. The complement of the contents of P (address of the current unexecuted instruction) is placed on the data bus. This address is then stored in location 00010.
12. A storage reply is received and resynchronized when the storage module has accepted the word. The resynchronized reply starts the storage timing chain.
13. Second store cycle FF is set.
14. The bus is released.
15. F000/001 is set. This advances the address in F to 00011.
16. Request bus is set.
17. Bus priority is once again obtained. A write signal, write designators of 00001_2 (lower 6 bits), and the storage address (00011) are transmitted to storage.
18. A storage request is transmitted.
19. Data bus register is cleared.
20. Gate EXX2 to DB register, shifted right three character positions. (F to I^7 to EXX2 to DB register. $\overline{\text{DB register}}$ to the T5XX transmitters of the data bus.) The storage module accepts the data and stores the upper 6 bits of F in the lower 6 bits of address 00011. The upper 18 bits of the word in 00011 are unchanged.
21. A storage reply is received and resynchronized. The resynchronized reply starts the timing chain.
22. Clear special store cycle FF.
23. The bus is released.
24. Second store cycle, trap sync, and SRO are cleared while RNI is set.
25. An RNI is now performed with a jump to address 00011. When the trapped instruction interrupt has been processed, a jump is made back to address 00010 where the address for return to the main program is found.

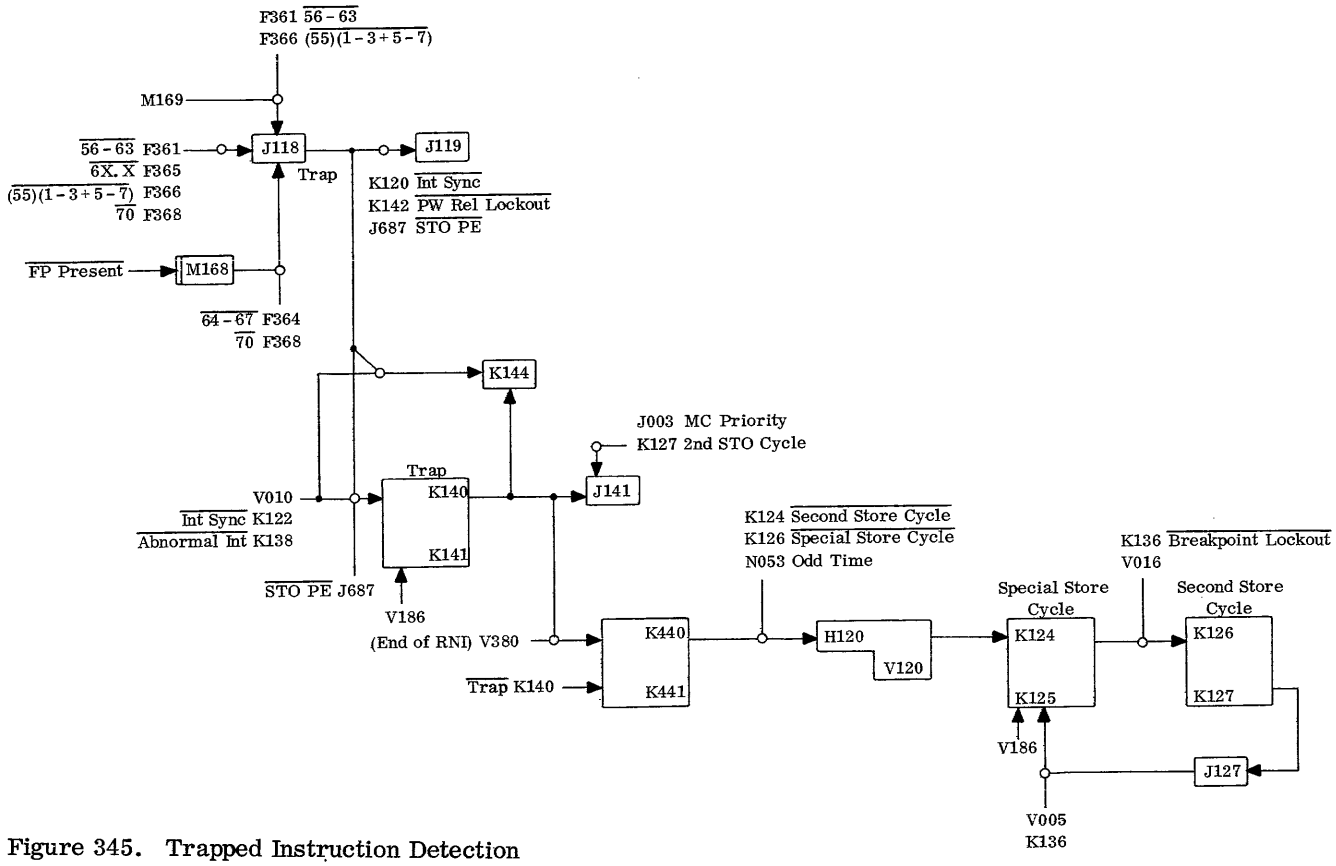


Figure 345. Trapped Instruction Detection

TRAP SEQUENCE

Timing begins at V008 of RNI.

V008: EXX2 to F1.
V009:
V010: Clear K002/003. Test Interrupt. Set K140/141 (Trap Sync).
V011: Input to H080.
V080: Advance Storage Sequence Controls to the appropriate cycle. Set K440/441.
N051: Input to H120, Block Start Arith Pulses (J142).
V120: Apply clear pulses to: K080/081 (RNI), K082/083 (RAD), K084/085 (ROP), K100/101 (Start Arith), K112/113 (Not RNI · Not INDEX). Block # setting of K110/111 (I^0 to F En.), K000/001 (Request bus); set K086/087 (STO), K124/125 (Special STO), K106/107 (Jump), K094/095; Disable I (if Exec); input to H201, H231.
N051, N205: Clear F_L, input to H122. P2 is advanced, input H220.
V122: Set K000/001 if K210 = 1. P2 to P1, clear K058/059 Program State II; set F030/031 F register bit 03 (Address 00010)
N051: Set K010/011 if K210 = 1. Transmit address 00010, F to EXX8 to T6XX to S bus. Transmit Write signal and Write Designators of 10011_2.
N050: Input to H117. NOTE: Storage Protect Comparison is Disabled.
V117: Set K116/117, K212/213; transmit a Storage Request.
N050: Input to H115.
V115: Set K012/013, transmit $\overline{P}$ on Data Bus ($\overline{P}$ to T5XX)
V116: Test Breakpoint Stop if BPO selected.
V061: Resynced Storage Reply, (Block setting of K060/061 if DP instruction).
V000: Clear K000/001.
V001-V005: Access Time.

V006: Set K126/127 (Second STO).
V007: Clear K010/011, K012/013, K116/117, K212/213; input to H086, input to H122, set F000/001.
V122: Block output of V086 and V186 (J063); set K000/001, request bus.
N051: Set K010/011 if K210 = 1. Transmit address 00011, F to EXX8 to T6XX to S bus. Transmit Write signal and Write Designators of 00001_2.
N050: Input to H117. NOTE: Storage Protect Comparison is Disabled.
V117: Set K116/117, K212/213; transmit a storage request.
N050: Input H115.
V115: Set K012/013 (Enable DBR to T5XX); input to H400; set K042/043, first transfer to bus.
N400: Input to H401.
N401: Clear DBR, input to H440.
N440: EXX2 to DBR (RS3), F to I^7 to EXX2 to DBR to T5XX to Data bus; clear K042/043.
V061: Resynced Storage Reply.
V000: Clear K000/001.
V001-V004: Access Time.
V005: Clear K124/125 (Special STO). NOTE: J063 = 0.
V006:
V007: Clear K010/011, K012/013, K116/117, K212/213; input to H086.
V086, V186: Clear K126/127 (Second STO), Clear K140/141 (Trap Sync), K440/441; input to H087, clear K086/087 (STO), set K080/081 (RNI).
V087: Input to H014.
V014: Jump and RNI at Address 000111.

INTERRUPT SENSING

Normal interrupts may be selectively sensed, independently of the interrupt mask register.

The interrupt status and fault sensing networks (Logic Diagrams, pages 2-207, 2-209 and 2-211) identify the various conditions. The instructions used for sensing are:

(c = I/O channel designator, XXXX = sense mask)
1. 77.2cXXXX - Sense status of I/O equipment.
2. 77.3cXXXX - Sense internal status.
3. 77.4cXXXX - Sense interrupt.

The various interrupt conditions are identified in the sensing network (figure 346 shows a portion of this network), then the condition identified is compared to its corresponding mask bit. If the mask bit is a 1 for the active line, one of the J45X* inverters outputs a 0. J455 of the comparison network also outputs a 0 to indicate comparison found. In this case, no further action is taken and program execution continues with an RNI from location P + 1.

Upon execution of sense instruction, if the mask bit for the active line is not set or if no lines are active, J455 outputs a 1 to indicate no comparison. This output is used to set skip FF. Program execution then proceeds with an RNI from P + 2.

*X = 2 + 4 + 6

PAUSE INSTRUCTION

The sensing network is also used for the pause instruction, 77.60XXXX (XXXX is the mask). The busy lines defined in table 25 are sensed and compared to the mask.

Table 25. BUSY COMPARISON MASK

COMPARISON MASK BIT POSITIONS	DEFINITION
00-07	Channel 0-7 busy
08	Typewriter busy
09	Type $\overline{\text{finish}}$
10	Type $\overline{\text{repeat}}$
11	Search/move control busy

When the pause instruction is decoded K438/439, enable pause, is set to allow instruction execution using the pause controls. If comparison between a busy line and its mask bit occurs J453 outputs a 1 which is delayed 10 ms before it sets pause FF. The output of pause is then applied to a 30 ms delay. J453 also holds J455 to a 0 to prevent the setting of skip FF.

If busy comparison drops at any time during the 40 ms of delay or if comparison is not originally detected, J455 outputs a 1 clearing pause via Y030. This signal also stops the free-running pause scanner. Stopping the scanner allows J010 to insert an input to the resync circuit. The signal from the resync circuit brings up N151 which, along with J455, sets skip FF. N151 then

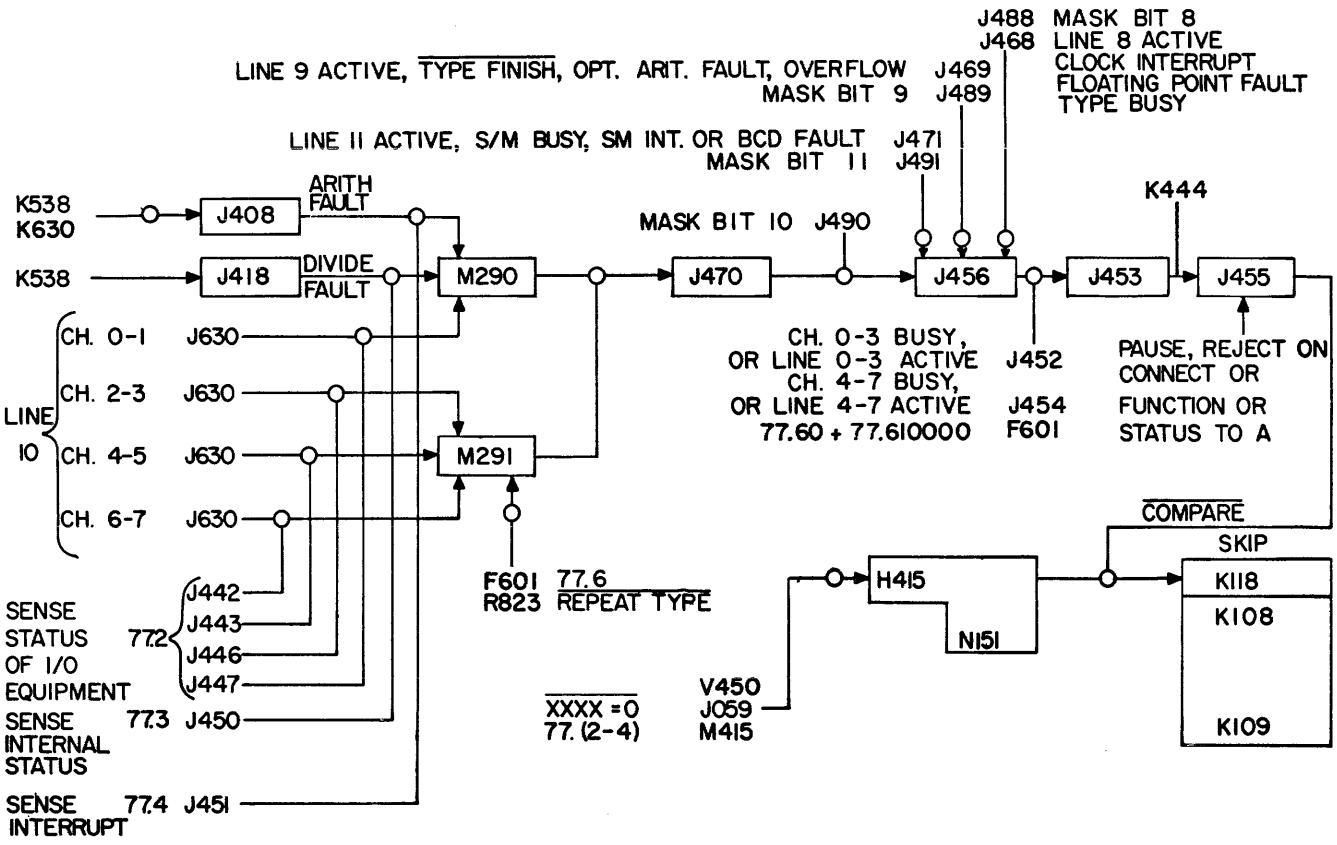


Figure 346. Sensing Network

pulses H014 to begin an RNI. The next instruction is then read from location P + 2.

When the entire 40 ms delay is exhausted before the busy signal drops, the delayed pause signal from Y029 stops the pause scanner and the resync circuit is pulsed in the manner described above. This time, however, comparison will not have dropped and J455 will output a 0. Thus, skip is not set and the RNI is made from P + 1.

An interrupt may occur at any time during the pause instruction. Setting of interrupt detected FF stops the pause scanner with both flip-flops set. The state of these flip-flops is translated by Y427 to insert a pulse into the resync circuit, to set block P FF, and to prevent the setting of skip by holding J455 to a 0. Since block P is set the P counter is not advanced and the RNI is made from location P. This means that the pause instruction is once again read but is interrupted before execution. Powerfail will cause the pause scanner to stop as though for an interrupt detected. The pause scanner permits an interrupt or powerfail to be processed before the end of the 40 ms delay being exhausted for the pause instruction.

CLEARING INTERRUPT

An interrupt may be cleared in several ways. The FFs associated with clock interrupt, exponent overflow, BCD fault, arithmetic overflow, divide fault, or a search/move completion are cleared when these faults are sensed using Sense Internal Status, 77.3, or Sense Interrupt, 77.4. Various inverters output the necessary clear signals when the function code of the instruction has been translated.

Instructions 77.50 and 77.51 selectively clear the interrupt conditions defined by the mask bits shown in table 26 below. When the mask bit for an interrupt is a 1, that interrupt condition is cleared by one of the F8XX inverters. The connect or function, 77.0 + 77.1, clears the interrupt to the selected I/O channel by allowing F572 to come up and gate a control enable signal to the selected I/O channel via one of the T14X transmitters.

Table 26. CLEAR INSTRUCTIONS - MASK BIT ASSIGNMENTS

BIT POSITIONS	MASK BIT DEFINITIONS	
	77.50 Instruction	77.51 Instruction
00-07	I/O channels 0-7 interrupts (internal and external)	I/O channels 0-7 interrupts (internal and external)
08	Clock interrupt	Clear type control
09	Exponent overflow or BCD fault	Not used
10	Arithmetic overflow or divide fault	Not used
11	Search/move completion interrupt	Search/move completion interrupt

INTERRUPT Worksheet

1. If interrupt mask register contained 7777 and every possible interrupt became active at once, list the order in which they would be recognized. (Concerning channel and external interrupts, assume that the channel interrupt is from channel 7 and that the external interrupt is on line 0, channel 0.)

   a.
   b.
   c.
   d.
   e.
   f.
   g.
   h.

2. If an interrupt has come in on line 3, channel 4, how many phase times will be consumed from the time start line counter sets until interrupt detected sets?

3. If the pause scanner was always in a clear/clear state how would this affect the output of N151? Would this cause any ill effects on computer operation on instructions other than a 77.6?

SELF-EVALUATION QUIZ ON CHAPTER 14

FILL IN THE BLANKS AND TRUE OR FALSE (T or F):

1. The three major groups of interrupt in the 3300 are _____, _____, and _____.

2. The Illegal Storage Reference interrupt is not available if the system is Non-executive.

3. Power failure interrupt references bit 9 or interrupt mask register.

4. All normal interrupts require the interrupt system to be enabled before they can be recognized.

5. Line interrupts and channel interrupts are equal in priority.

6. Stop on Storage Parity Error has priority over Interrupt on Storage Parity Error.

7. Storage Not Available interrupt will never be generated during the execution of a Test Storage Availability instruction.

8. The interrupt sequence provides a convenient method for transferring the system from Program State to Monitor State.

9. The normal interrupt counters are free-running as soon as power is applied to the system.

10. The state of the normal interrupt counters when an interrupt has been recognized determines what ID code will be stored in address 000005_8.

Score Yourself:
None wrong--excellent work!
One wrong--careless.
Two or more wrong--that is bad!

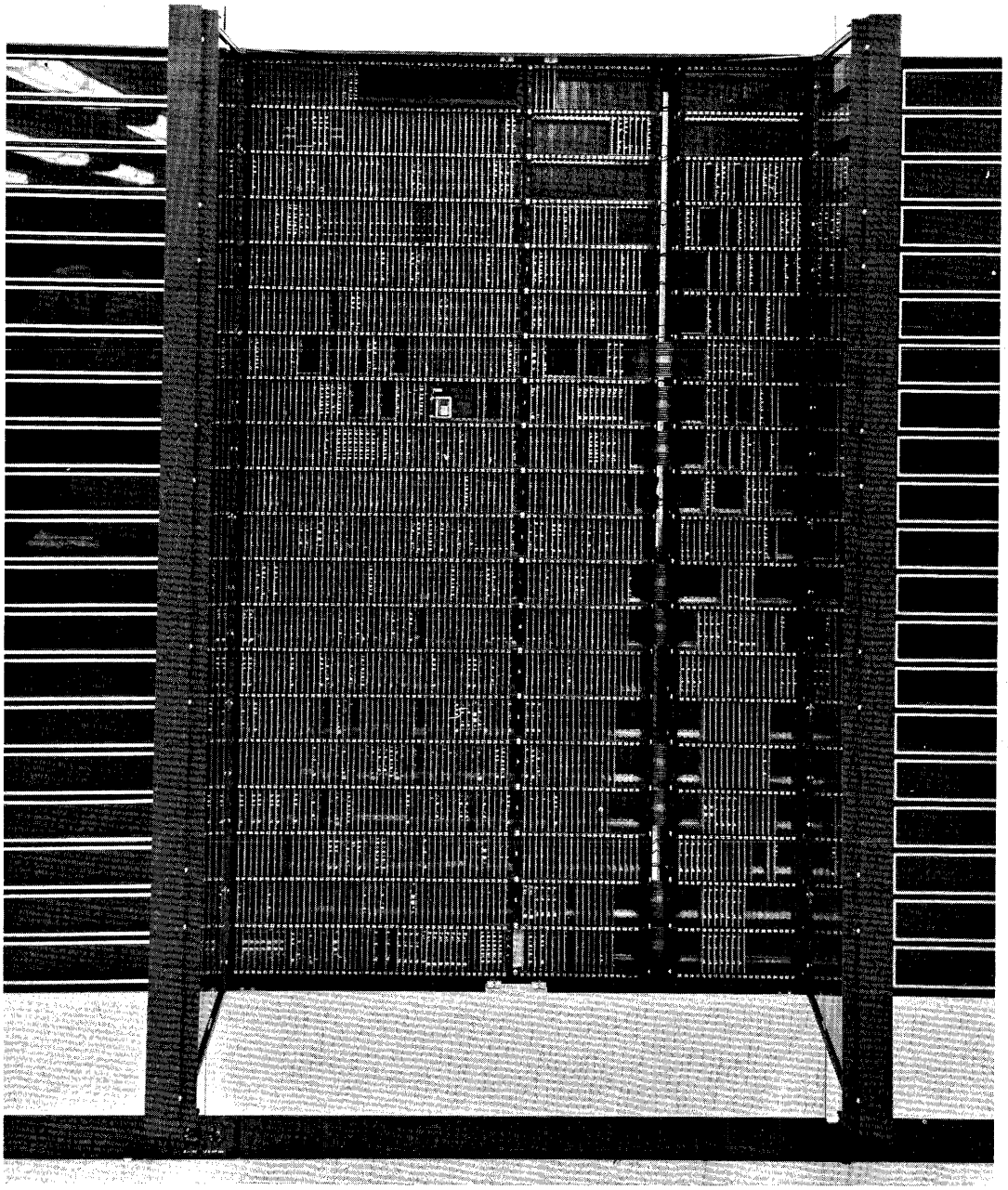


Figure 347. Block Control, Front View

CHAPTER 15

BLOCK CONTROL

Block Control is an integral part of the 3300 Computer. Its primary function is to control block operations. A complete list of Block Control functions is:

1. Control I/O.
2. Control Search or Move.
3. Update the Real Time Clock.
4. Control Typewriter operations.
5. Control inter-register transfers into or out of the register file.
6. Control keyboard operation into or out of register file.

Figure 347 is a front view of Block Control. Note the real time clock and delay line in the H row. Figure 348 is a block diagram of Block Control.

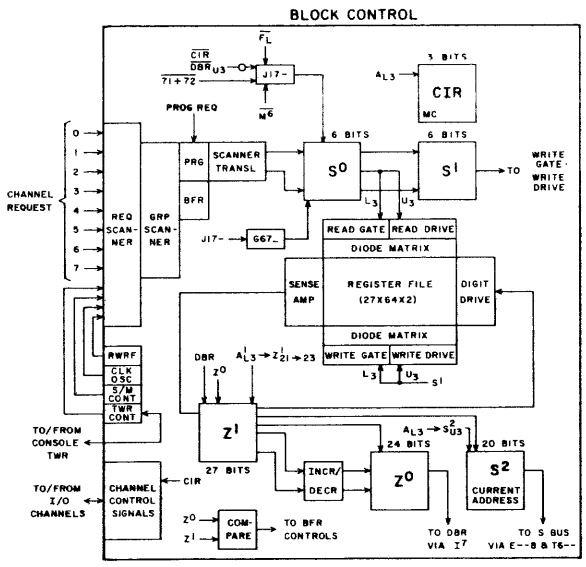


Figure 348. Block Control, Block Diagram

The register file is a 64_{10} word, 27 bit-per-word, word-organized storage which functions the same as the page index file. Now might be a good time for you to turn back to the chapter on the Multiprogramming module and review the principles of word-organized storage. Table 27 lists the register assignments for the register file.

Table 27. REGISTER ASSIGNMENTS

REGISTER	RESERVED FOR
00-07	Current character address (channels 0-7)
10-17	Last character address (channels 0-7)
20	Current character address (search)
21	Source address (move)
22	Clock, current time
23	Current character address (type)
24-27	Temporary storage
30	Last character address +1 (search)
31	Destination address (move)
32	Clock interrupt mask
33	Last character address + 1 (type)
34-37	Temporary storage

REGISTERS

CIR (3-bit Channel Index Register)

The contents of CIR are ORed with the channel designator of Connect, Function, Activate I/O and Sense instructions to obtain an absolute channel designation.

S^0, S^1 (6-bit, Double Rank Register)

S^0 controls the gates and drivers for the read cycle on register file references. S^1 controls the gates and drivers for the write cycle on register file operations.

S^2 (20-bit Current Address Register)

S^2 holds the current address during I/O or S/M buffer cycles. A character address is always used during buffer cycles; therefore, 20 bits are necessary for S^2.

Z^1 (27-bit Register)

All data read out of or written into the register file must pass through Z^1.

Z^0 (24-bit Register)

Z^0, in conjunction with Z^1, forms an increment/decrement network for updating the current address during a buffer cycle. Increment/decrement may be by 1, 2, or 4 octal.

BLOCK OPERATIONS

Operations within Block Control are basically two types: those initiated by Main Control and those initiated within Block Control. Operations initiated by Main Control are activate I/O, connect or function, activate S/M, activate type and inter-register transfers with the file.

Operations initiated within Block Control are I/O buffer cycles (including type), S/M buffer cycles, and update the real time clock.

Buffered operations (buffer cycles) take place in block control when accomplishing an I/O operation, doing a search or move, or during a type-in or type-out. During a buffered operation, the program is delayed only long enough to permit block control to make a memory reference (access to the bus system). One character or word will be moved on the data bus during this delay. The program then continues until another storage reference is required by block control. This sequence will continue until the block operation (buffer cycle) is completed.

One other buffered operation is the 1 ms incrementing of the real time clock. This operation does not require a memory reference and the program will not be delayed unless clock interrupt is set.

REQUESTS

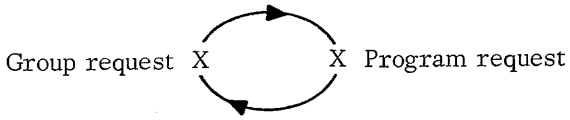
In order to obtain usage of the block control logic, the concerned section must transmit a block control request signal.

Requests are of two types, external and internal in respect to block control. External requests are generated by main control, a communications channel, the typewriter or keyboard for read/write register file. Internal requests are created by one of the controls within block control, e.g., search, move, or clock.

PRIORITY

A request priority network is established within block control by five scanners (Logic Diagrams, page 2-159).

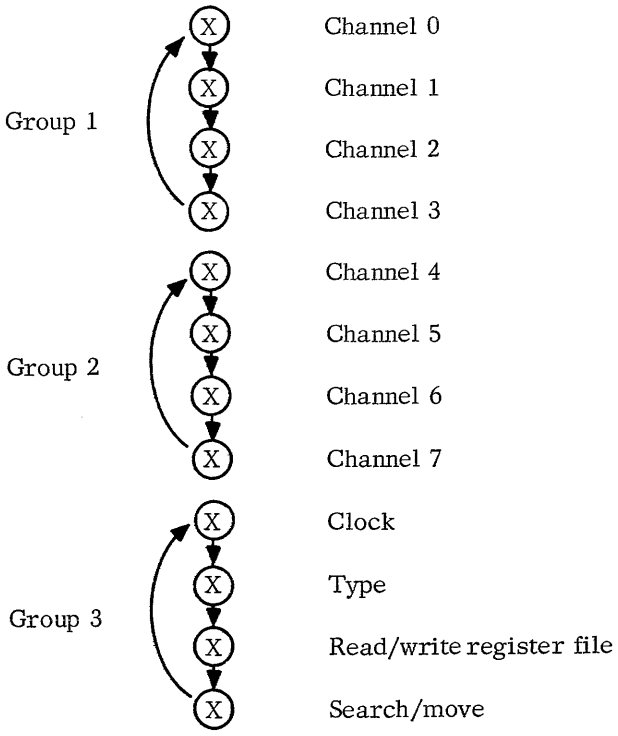
The first is the group/program scanner. As the group program scanner sets and clears it alternately checks for requests from main control (program request) and a request from any other source (group request). When both flip-flops are set the absence of a program request will allow them to clear. When cleared, the absence of a group request will allow them to set. The scanning action of the group/program scanner can be represented as:



The remaining requests (group requests) are divided into three groups:

Group 1. I/O channels 0-3
Group 2. I/O channels 4-7
Group 3. Clock, type, search/move, RWRF

The action of the group scanners can be represented as:

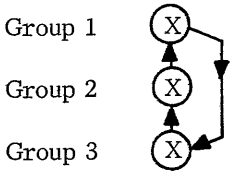


When the scanning action is interrupted by a request, the state of the flip-flops in that scanner will determine the source of the requesting signal. Table 28 lists the states of the scanners when stopped by a request.

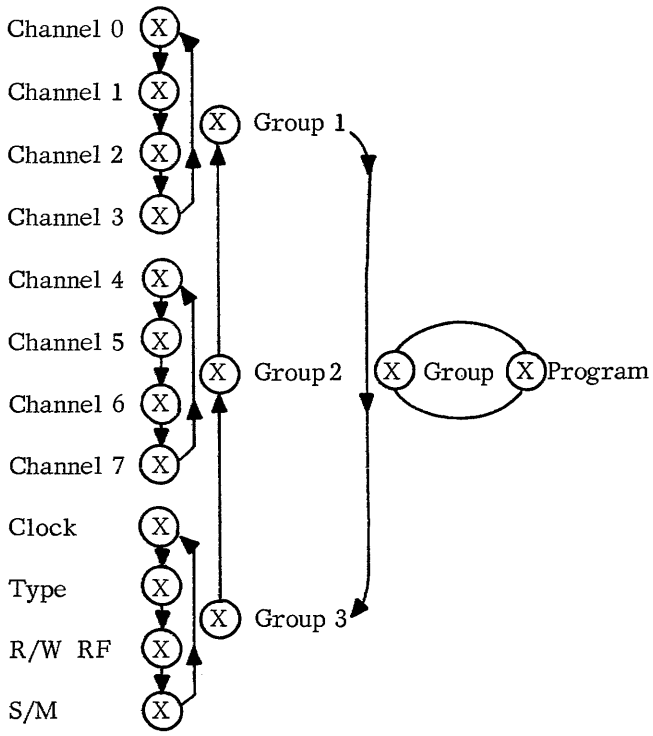
Table 28. SCANNER STATES

REQUEST	STATE	
Channel 0	Z308/309 set	Z310/311 set
Channel 1	Z308/309 clear	Z310/311 clear
Channel 2	Z308/309 clear	Z310/311 clear
Channel 3	Z308/309 set	Z310/311 clear
Channel 4	Z312/313 set	Z314/315 set
Channel 5	Z312/313 clear	Z314/315 set
Channel 6	Z312/313 clear	Z314/315 clear
Channel 7	Z312/313 set	Z314/315 clear
Clock	Z316/317 set	Z318/319 set
Type	Z316/317 clear	Z318/319 set
Read/write RF	Z316/317 clear	Z318/319 clear
Search/move	Z316/317 set	Z318/319 clear

The final scanner is the group scanner. As it runs it checks to see if the group 1, group 2, or group 3 scanner has been stopped by a request. The action of the group scanner can be represented as:



All of the scanners together can be represented as:



The group/program scanner gives equal recognition time to requests from the three groups. The individual scanners, however, are always reset by the rescan logic (Logic Diagrams, page). For example, if a request is received from channel 6, the group 2 scanner will stop. When the group scanner reaches the point where it can recognize the group 2 scanner, it too will stop. The group/program scanner stops with both flip-flops clear. After the request is processed, the group 2 scanner is reset and will recognize a request from channel 4 first. The group scanner is not reset and it will check group 1 next. The group/program scanner will restart and check for a program request if there is not another Active Group Request.

Program requests have equal priority with any other request, and within the groups, channels 0 and 4 and the clock have highest priority. From a programing standpoint, therefore, channels 0 and 4 should always have the faster I/O devices attached to avoid delay in waiting for priority over slower devices.

TIMING

All operations within block control are governed by a timing chain composed of nine control delays.

A delay line is provided to create the enables for the register file. The delay line is an inductor that is tapped at 25 nsec intervals. It is fed by a C08 delay line drive card (D160) that outputs a -0.6v level whenever it receives a 1 input. The setting of Z332/333 will start the delay line and $\approx$ 75 nsec. Later Z332/333 will be cleared by E388 on the A4 tap of the delay line. This will cause a 75 nsec wide pulse to travel down the delay line and create the various enables for the register file.

Figure 349 is the timing chart for the register file enables.

BUSY

Search/move (S/M) and I/O operations are mutually exclusive. That is, if one of these buffer operations is active (blocking control servicing it) and activating the same buffer is attempted, the computer will read the reject instruction at P + 2.

Inverter J192 (Logic Diagrams, page 2-165) will output a 1 whenever an I/O channel is selected by the program request and it is busy or when S/M has been selected and is busy.

When the busy condition is present, the AND gate between H151 and H152 is broken and the timing chain is bypassed (Logic Diagrams, page 2-165). The delay line will not be started and there will be no storage operations within the register file.

The busy condition will also prevent setting of skip FF in main control, and when the computer does the next RNI sequence it will read the reject instruction at P + 2.

BLOCK OPERATIONS

Block operations are of four types: search, move, I/O, and typewriter. These operations use the computer block controls and, except for I/O with A register, are buffered. After the search, move, I/O, or typewriter control has been activated the computer returns to the main program and continues until an interrupt is generated or the program senses for block operation completion.

To activate the search, move, or I/O control, three steps are necessary:

1. Load A lower 3 with the upper 3 bits of address.

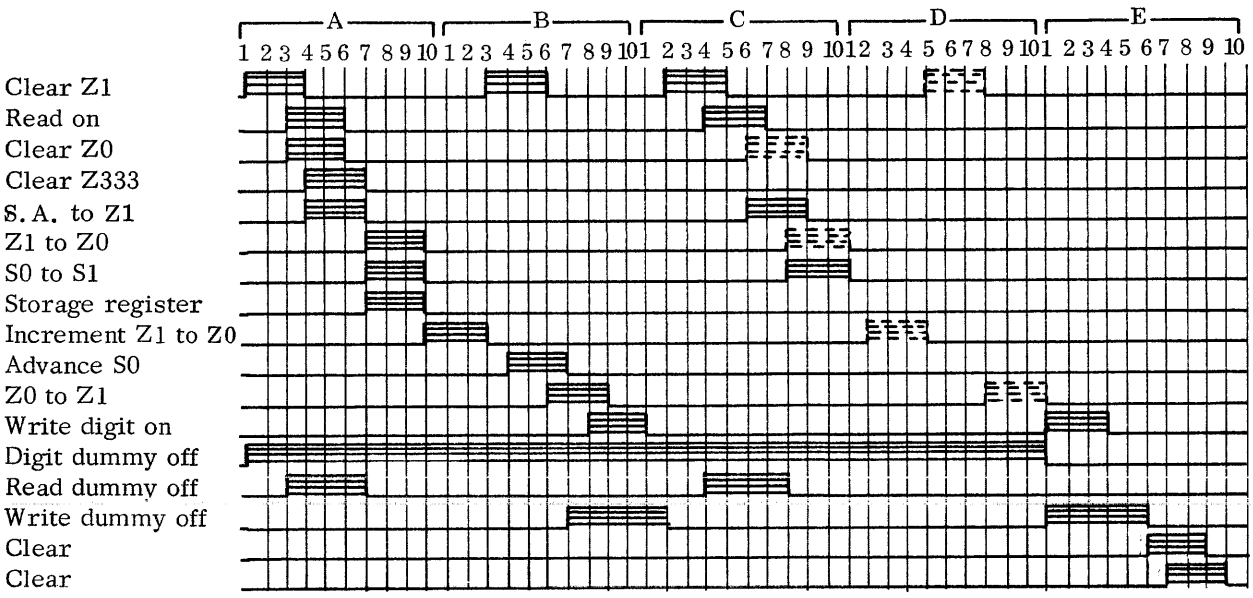
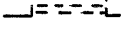


Figure 349. Timing for Register File Enables

 used for move only

2. First RNI. The instruction is read up and placed in F register. At the end of this RNI sequence, main control sends a program request signal to block control. After granting priority, block control responds with a resynchronized reply (Logic Diagrams, V171, page 2-159). This reply initiates a second RNI sequence at P + 1.
3. Second RNI. As this RNI sequence progresses, the second instruction word is read out of storage and placed in data bus register. At the end of this RNI, a signal is sent to block control that starts the timing chain and the delay line. Starting the delay line causes (P + 1)* and (P)* to be stored in the register file. If I/O channel is busy the words from P + 1 and P are not stored in the register file, and a portion of the block control timing chain is skipped. After the storage operations have been completed, block control is released and the computer continues with the program. An interrupt may not be recognized during this RNI.

The typewriter control is activated through a programed instruction (77.75 or 77.76) or by pressing TYPE LOAD or TYPE DUMP on the typewriter or on the console. Starting and terminating address + 1 must have been previously entered in registers 23 and 33 by an IRT instruction or write register file.

INPUT/OUTPUT

The I/O instructions, 73-76, permit the bidirectional flow of data between the computer and the peripheral equipment. The I/O instructions are divided into two separate types: those that deal with storage and those that deal with A register. Table 29 lists the I/O instructions.

Table 29. INPUT/OUTPUT INSTRUCTIONS

INAC INAW	Character input Word input	Unbuffered input to and output from A
OTAC OTAW	Character output Word output	
INPC INPW	Character input Word input	Buffered input to and output from storage
OUTC OUTW	Character output Word output	

The I/O instructions begin with a common set of steps:

(1) When an I/O instruction is sensed by the computer, (2) a program request signal is sent to block control. This signal will set K152/153 (Logic Diagrams, page 2-159) and drive the output of J179 to a 0. The 0 from J179 will stop the group/program scanner with both flip-flops set. (3) The second RNI sequence is started, and at V008 time K164/165 is set and the upper three bits of the DB register are sent to channel translator FFs (K154/155, K156/157, K158/159; Logic Diagrams, page 2-165). (4) The channel enable signal is now transmitted to the selected I/O channel via a T80X transmitter.

At the conclusion of the second RNI, the timing chain is started via the AND gate into H151 composed of V180, F664, and K169. (5) If a busy condition does not exist, N151 will set K180/181 (Logic Diagrams, page 2-171) and the delay line will be started. V280 will set skip FF in main control to bypass the reject instruction. V152 on the timing chain will set K174/175 (Logic Diagrams, page 2-163) and (6) the read or write signal will be transmitted to the communications channel followed by a clear O signal. If the I/O operation is to be an input (read) the read signal from block control will set the data signal FF in the communications channel. The receipt of the data signal from the communications channel will cause the controller to place a word or character in the communications channel O register. After the word or character is in O register, the communications channel will generate a block control request signal which will be placed in the block priority network. An output (write) will cause the block control request signal to be generated immediately.

During this time, (7) (DBR) has been sent to register 0X of the register file, (8) (F) has been sent to DB register and then stored in register 1X of the register file where X = the channels 0 to 7. At the conclusion of the storage operations, (9) block control is released, as is the data bus, and the program does an RNI sequence at P + 3.

During the second RNI in main control (reading P + 1), P2 is advanced by H231 (Logic Diagrams, page 2-47) and the second word of the instruction is read and placed in DB register. During this time, F (which contains (P)*) is prevented from clearing by setting K062/063 (Logic Diagrams, page 2-5) and driving J144 (page 2-29) to a 0. Also prevented by this is the EXX2 to F transfer. In this manner the first word of the instruction remains in F while the second word is in DB register.

I/O OPERATIONS WITH STORAGE

An I/O operation with storage begins when the communications channel receives the channel enable, read or write, and clear O signals from block control.

A read signal will cause the communications channel to send a data signal to the controller which will soon reply by placing a word or character in O register of the communications channel. Receipt of this word or

*Contents of address specified by P

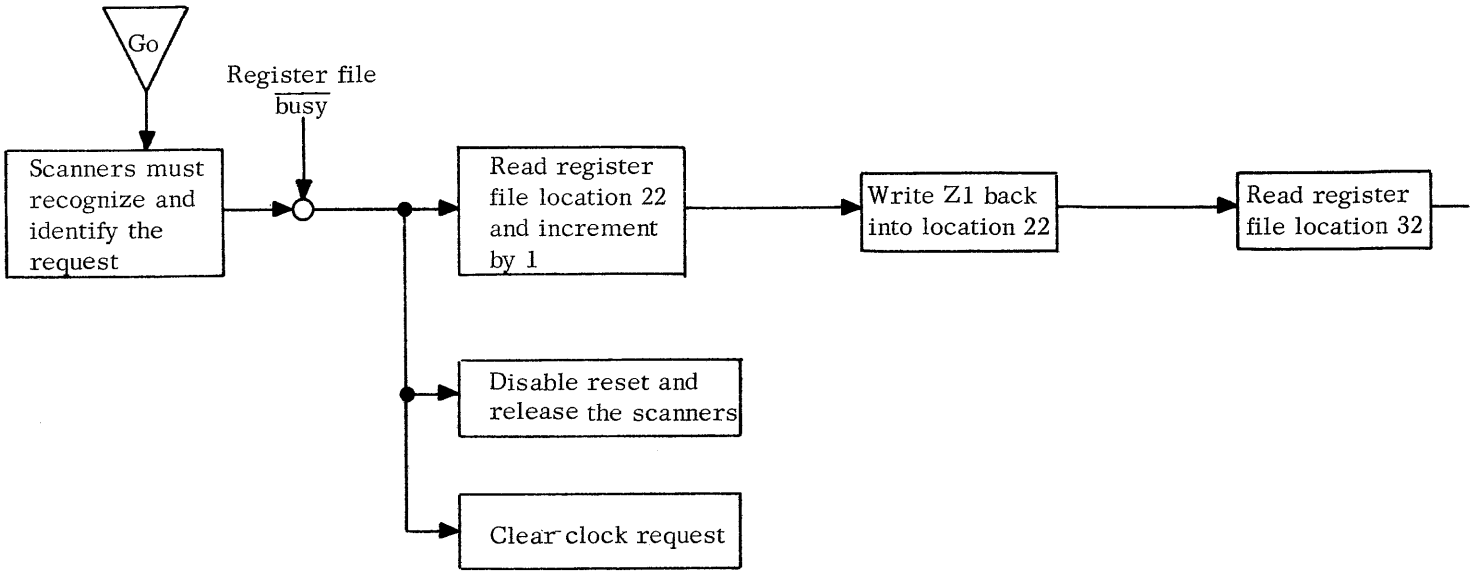


Figure 350. Flow Chart for Updating Real Time Clock

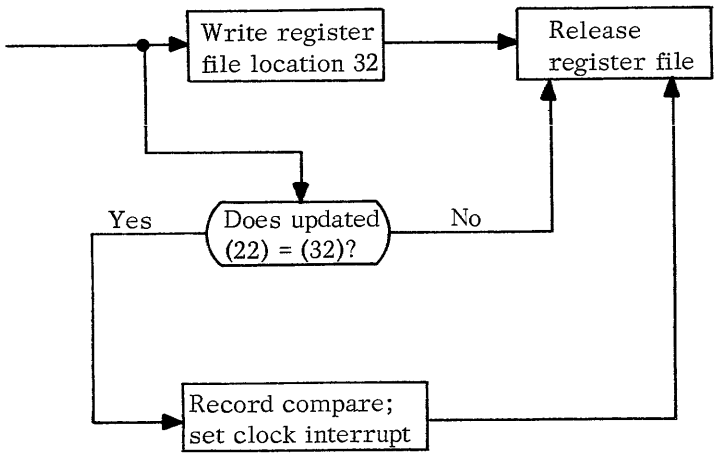
character will cause the communications channel to generate the block control request signal. A write signal causes the communications channel to generate the block control request signal immediately.

After block control has granted priority to the request, the delay line is started via the G659 (Logic Diagrams, page 2-177) input to Z332/333. Register 0X,* which contains M1, is read into Z1. The contents of Z1 are then sent to the current address register (S2) and to Z0 through the increment/decrement network. The network will increment or decrement M1 by 1, 2, or 4, depending on the type of instruction. At the same time, Z352/353 (Logic Diagrams, page 2-167) has been set in order to generate the storage request in main control.

Register 1X* (M2) is read out and placed in Z1. The comparison network (Logic Diagrams, page 1-103) will test for M1 = M2. If equality does exist the comparison FF Z350/351 (Logic Diagrams, page 2-167) will be set. Setting the comparison FF does not terminate the operation until after the word or character currently being processed is delivered to or from storage.

Transmission of the storage request will suspend the program while main control waits for bus system priority. The reply from memory originated by the request from block control will start the timing chain at H150 (Logic Diagrams, page 2-161). As each control delay is pulsed by the preceding one, certain enabling signals are transmitted. The data bus is connected to O register and the communications channel transmits or receives one word or character. After transmission of the word or character, the bus is released and the

*Contents of address specified by P



program resumes. The request procedure is repeated by the communications channel until the block of addresses is exhausted.

I/O OPERATIONS WITH A

An I/O operation with A begins in the same manner as those operations with storage. The request from the communications channel causes registers 0X and 1X to be read. A storage request is transmitted in order to gain access to the bus. A memory cycle is initiated but, as the lower 17 bits of the first instruction word are meaningless, the memory performs only a read. Arithmetic is started and the bus is enabled to I^4. After propagating through the adder, the word or character is placed in the lower bits of A. The upper bits remain clear. If the operation is an OTA, the data path is $\overline{A}$ to I2 to X_1 to I_7 to EXX2 to DBR to Data bus.

TYPEWRITER

A type operation begins when the computer senses a 77.75 (type-in), a 77.76 (type-out) instruction, or when the operator presses TYPE LOAD or TYPE DUMP on the console or typewriter. Before this, the current and terminating address of the block must have been stored in register 23 and 33 of the register file. This storage is accomplished through the IRT instruction.

Receipt by block control of a type-in or type-out signal (Logic Diagrams, page 2-169) will set a corresponding flip-flop (Z360/361 for type-in, Z362/363 for type-out).

A block control request signal is then placed in the scanner network (for type-in, a character must be typed to initiate the request). After granting priority the delay line is started and register 23 (M1) is read out. A storage request is generated and, after memory replies, the bus is connected to O register in the typewriter via transmitters and receivers in block control. A character is then sent to or from storage via the bus.

During type-in a strobe signal indicates to block control that a character has been typed and is in O register of the typewriter. During type-out a $\overline{\text{busy}}$ signal will be sent by the typewriter indicating that it is ready to accept a new character.

When the type load indicators on the typewriter and on the console go off it indicates type load operation (M1 = M2) is finished. The operator could continue to type but the data would be lost.

TIMING

See figure 350.

J793 = 1 for 200 nsec (when C051 starts its positive swing and Z380/381 (clock request FF) sets. J793 is a pulse shaper (C089 card).

G620 = 0; group 3 scanner goes set-set; G638 = 0; G641 = 1; group scanner goes set/clear; group/program scanner goes clear-clear.

If Z390/391 has been clear 100 nsec, G657 and G659 will have 100 nsec pulses since G651 = 1 and G656 = 0. S0 is forced to a 22; Z332/333 (start delay line FF) and Z348/349 (clock control FF) set.

Record buffer cycle with G642 = 1.

(A3) E353: Read current on, clear S2.

(A4) E388: Clear Z332/333 (start delay line FF).

E351: Sense amps → Z1, set Z398/399 to disable scanners.

(A7) E356: S0 → S1, Z1 → Z0, Z1 → S2; clear Z0.

(A10) E359: Z1 → Z0; clear Z380/381, set Z390/391; increment all 24 bits by 1.

(B3) E355: Clear Z1; set Z328/329 (rescan 3 FF).

(B4) E367: Advance S0 to 32.

(B6) E365: Z0 → Z1; clear rescan 1 and 2 FFs.

(B8) E360: Write and digit current on; clear rescan 3 FF.

(C2) E367: Clear Z1.

(C5) E370: Read current on.

(C6) E371: Sense amps → Z1.

(C9) E376: S0 → S1.

(D5) E375: If Z1 = Z0 ((22) = (32)), set Z350/351 which sets Z378/379 (clock interrupt FF).

(E1) E380: Write and digit currents on.

(E6) E387: Clear Z390/391 and Z350/351.

(E7) E389: Clear Z348/349 (clock control FF).

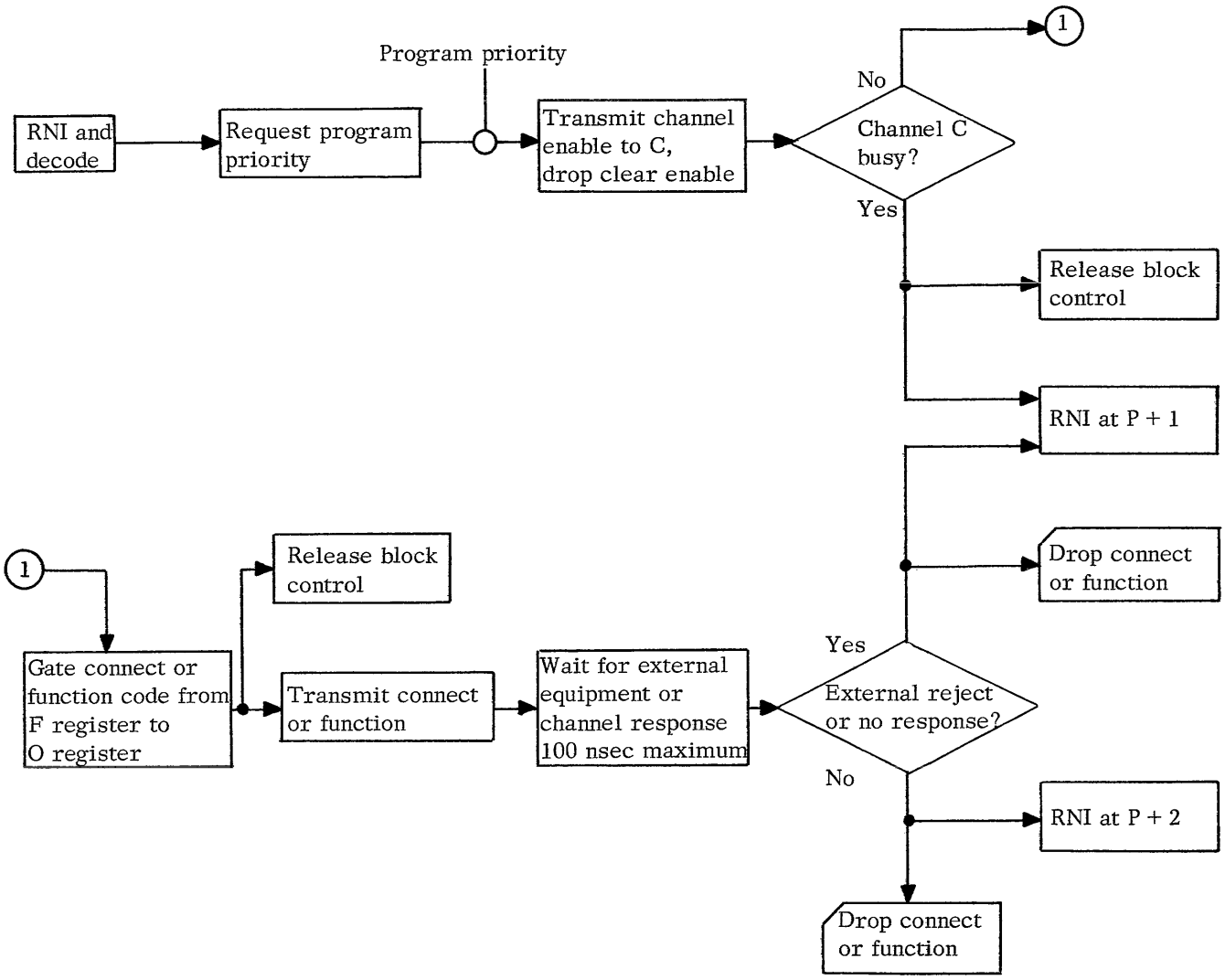


Figure 351. Flow Chart for Connect (77.0C) and Function (77.1C) Instructions

The flow charts on the following pages show the progression for instructions 71 to 77.1 and the single-precision interregister transfer instructions (figures 352-357). Complete timing for the instructions is given in the command timing charts.

The 71 to 76 instructions are covered in two parts, activate and buffer cycle. During activate the instruction is in F register. During buffer cycle main control may be executing the instructions of a program or may be stopped. Use the flow charts as you follow the respective instruction through the Command Timing Charts and Logic Diagrams.

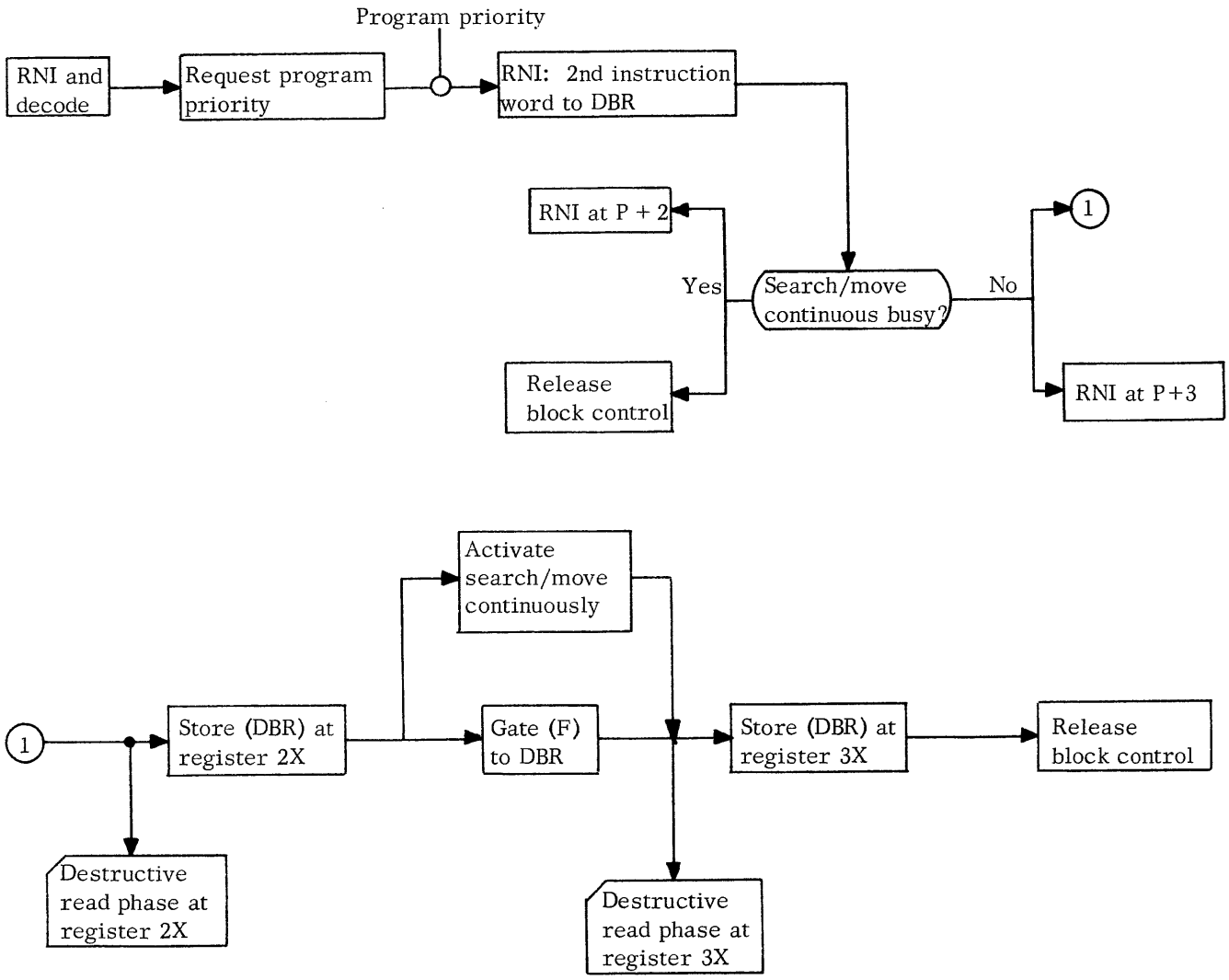


Figure 352. Flow Chart for Activate Search (71) or Move (72) Instructions

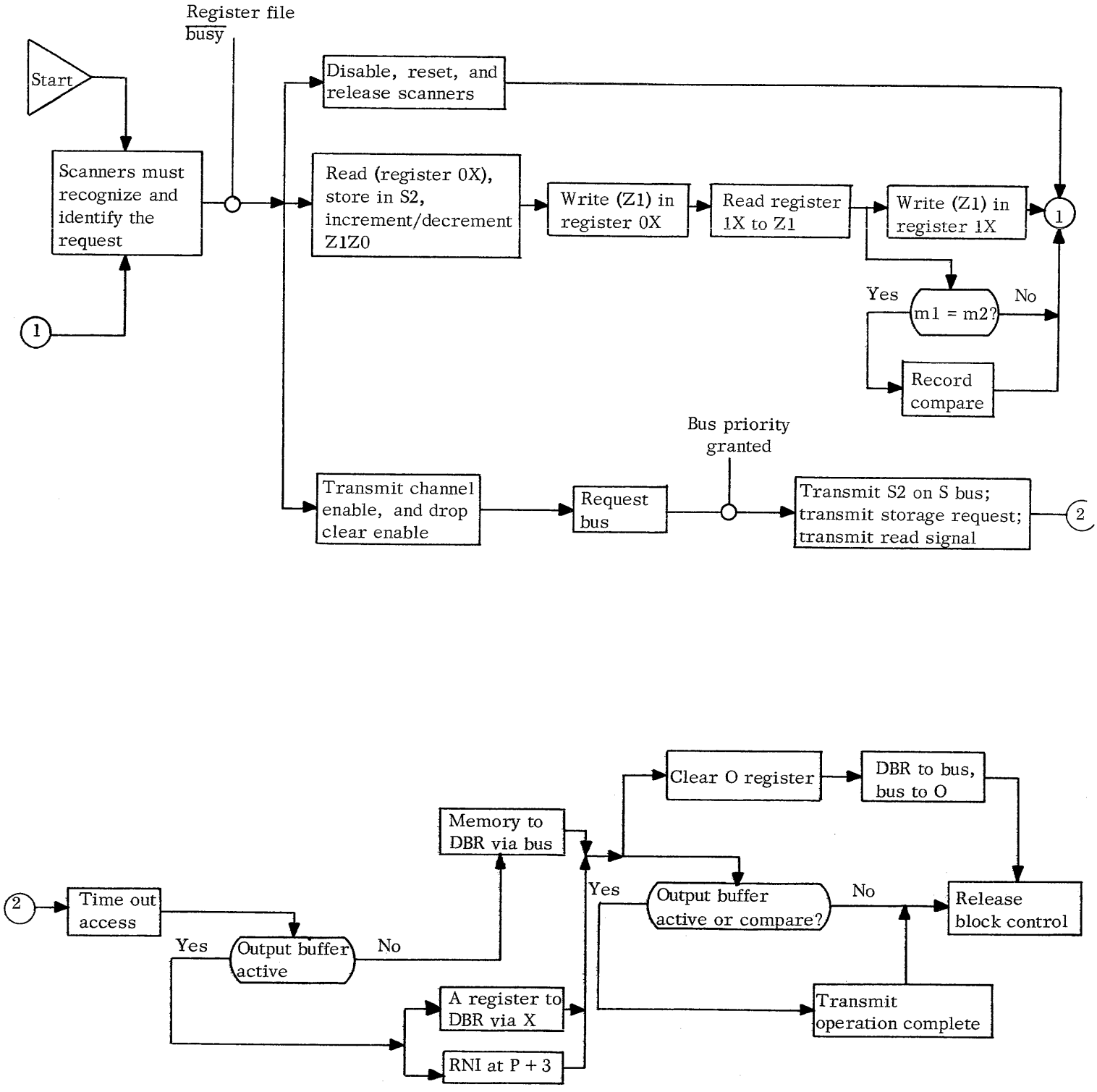


Figure 353. Flow Chart for Output Buffer Cycle

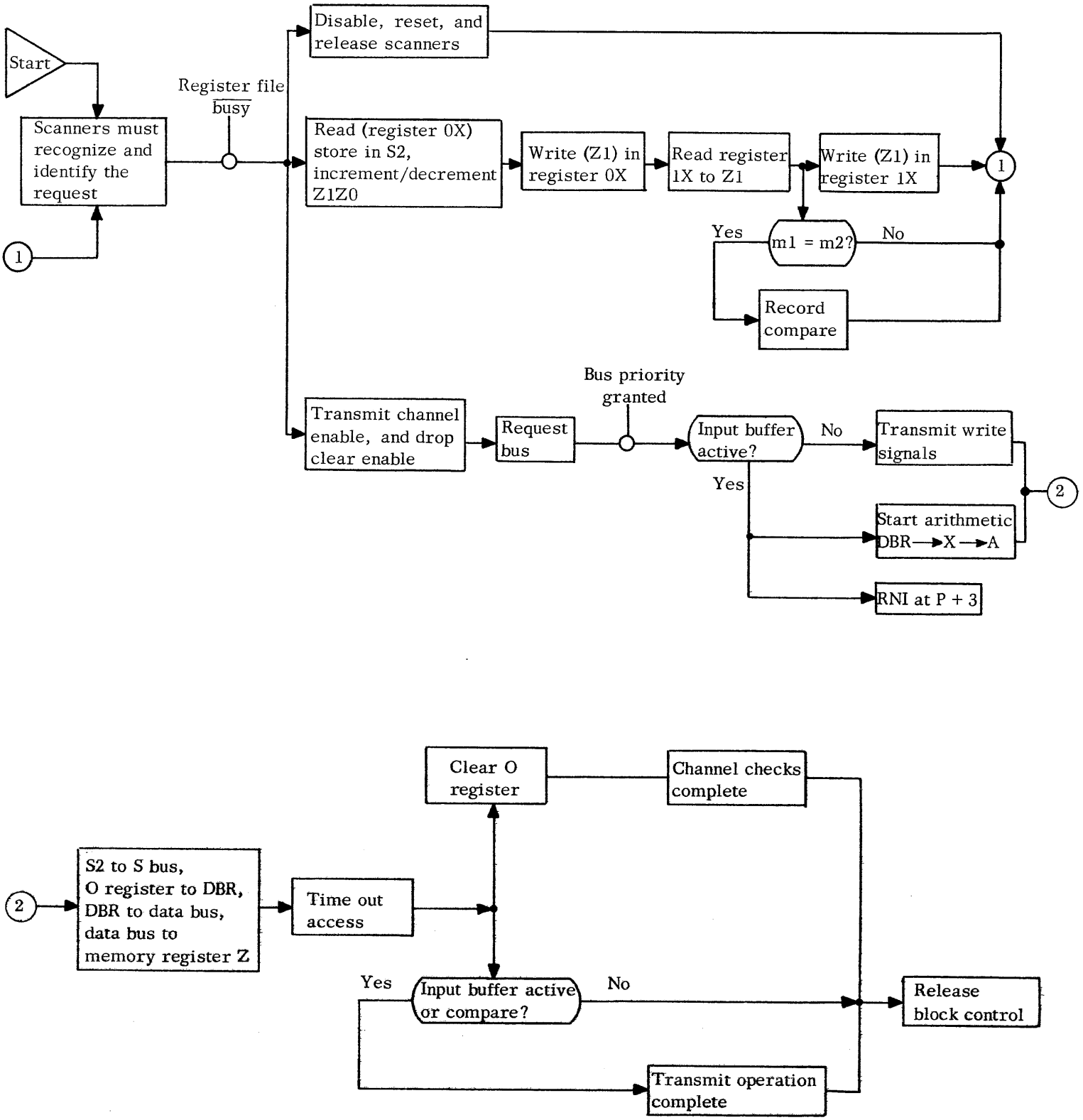


Figure 354. Flow Chart for Input Buffer Cycle

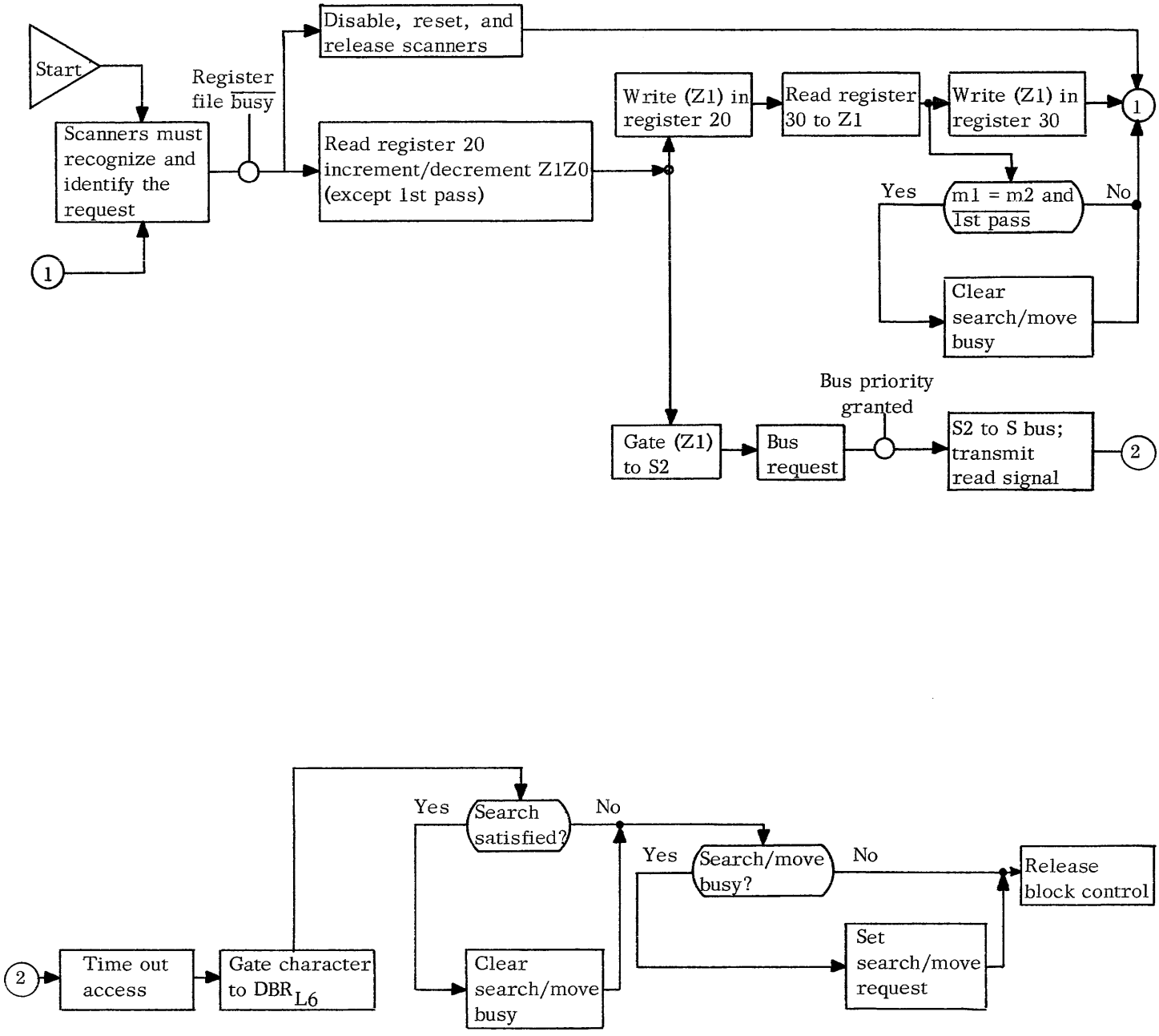


Figure 355. Flow Chart for Character Search Buffer Cycle

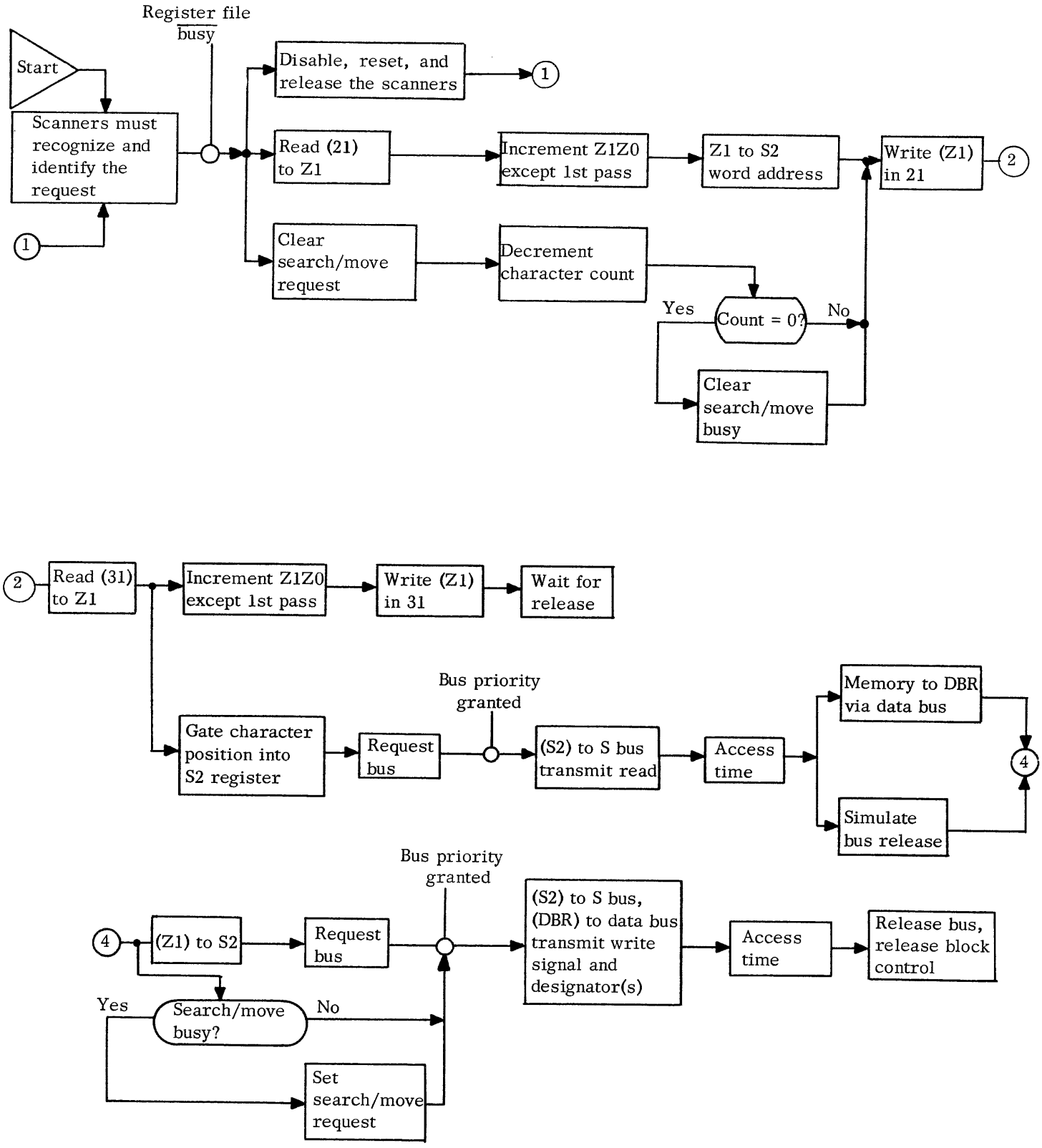
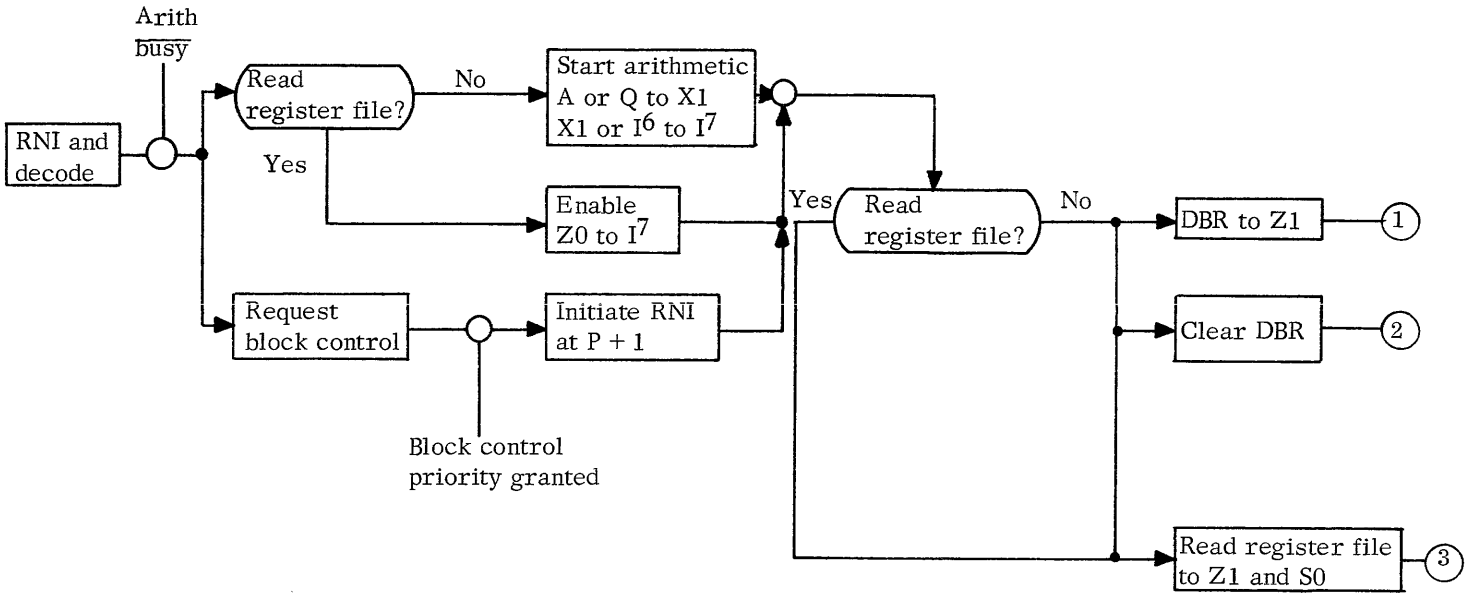


Figure 356. Flow Chart for Move Buffer Cycle



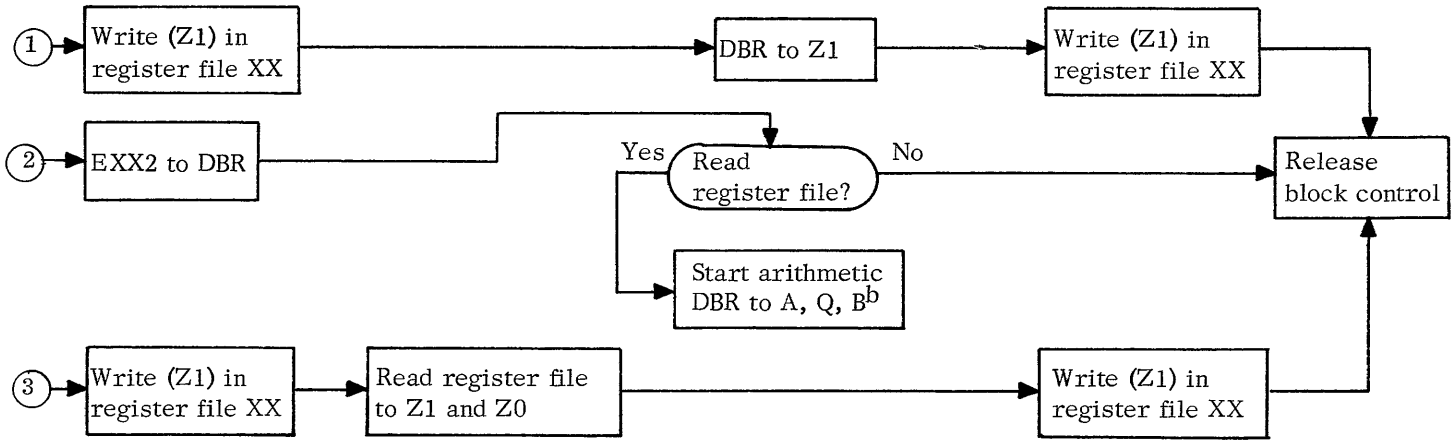


Figure 357. Flow Chart for IRT With Register File

I/O WORD MODIFICATION

Activate I/O and S/M are two-word instructions which are read from storage and placed into the register file so that Block Control can accomplish the actual block operation. So that these instructions can access any portion of 262K of core storage, an 18-bit word address or 20-bit character address is necessary. The instruction format for the 71 through 76 instructions only allow for a 15-bit word address or a 17-bit character address. The upper 3 address bits for this instruction group are obtained from the lower 3 bits of the A register. Therefore, the words placed in the register file on activate will consist of 27 bits and, in appearance, will differ appreciably from the Activate instruction.

For 73 through 76 instructions, on activate, (P) are stored in register 1X and (P + 1) are stored in 0X. Figure 358A and 358B illustrate the contents of 0X and 1X and the function which the bit positions represent. Note that bits 24 through 26 of 0X and bits 21 through 23 of 1X hold a quantity referenced as modified operation code. This quantity is derived from the original 6-bit operation code of the Activate instruction. The upper bit of the modified operation code is 0 for operations with storage and 1 for operations with A. The lower two bits of the modified operation code are the lower two bits of the original 6-bit operation code.

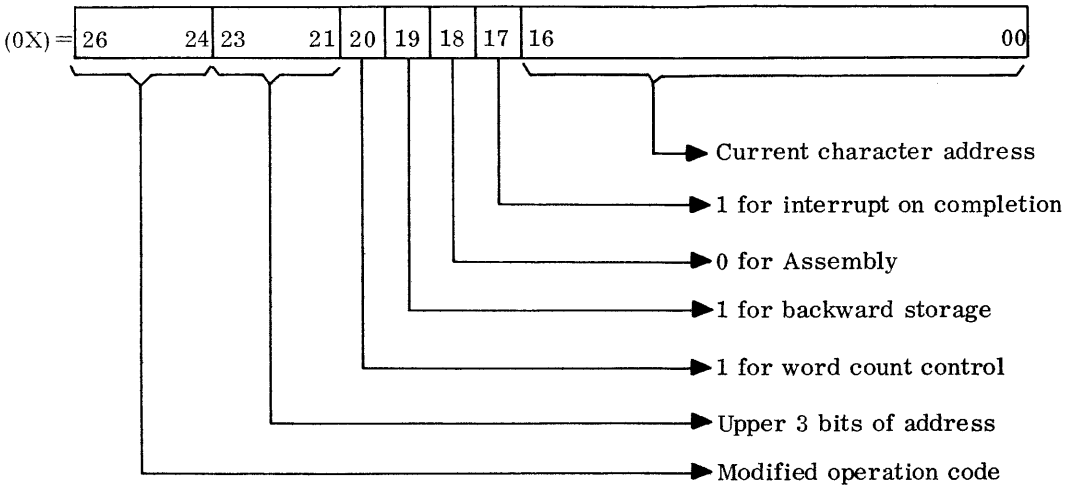


Figure 358A. Contents of 0X Register

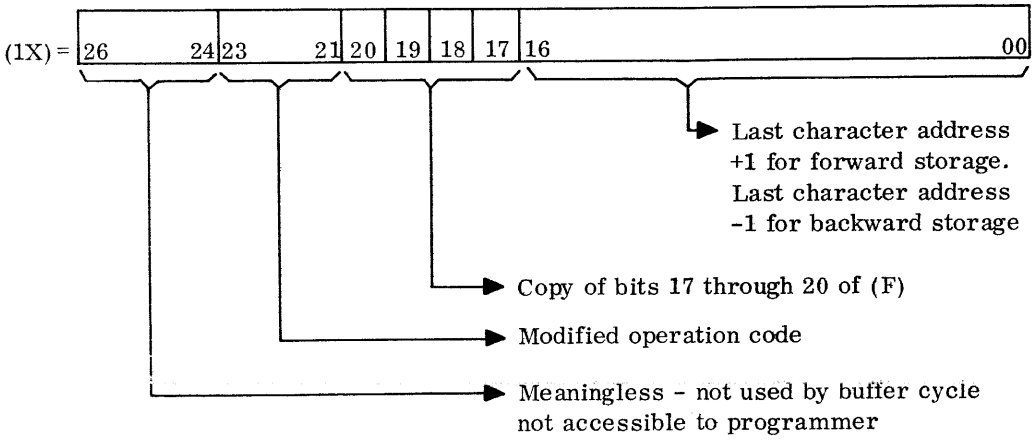


Figure 358B. Contents of 1X Register

BLOCK CONTROL Worksheet

1. Several sets of I/O instructions are shown below. Write them as they would appear in the register file.

   (A) = 00000001
   73 000105
   20 400100

   (A) = 00000005
   73 000105
   21 000100

   (A) = 77777774
   75 400105
   21 000100

   (A) = 00000000
   75 000105
   27 000105

   (A) = 00000006
   74 000104
   36 000100

   (A) = 00000003
   76 000104
   37 000100

   (A) = 000000000
   74 400104
   37 000100

   (A) = 00000002
   76 000104
   30 400100

2. A move buffer cycle is in progress. The address portion of the word in the register file is:

   RF 21 = 32400013
   RF 31 = 72000100

   What would be the pseudo-character address used during the read portion of the move buffer cycle in order to have the character that is read from memory positioned in the proper place in DB register before the store cycle? _____

3. On a search instruction, a character is read from storage and compared to a character which is part of the instruction word in register file location 30. When this comparison takes place the character from storage is located at character position _____ of _____ register and the character from the instruction word is in the upper six bits of _____ register.

4. If requests came into block control from program control, search/move, channel 0, channel 2, channel 7, and type, and block control is servicing the request from channel 0 first, in what order would the remaining requests be serviced?
   1.
   2.
   3.
   4.
   5.

Timing for Activate I/O, 73 through 76 instructions. Timing begins at V008 of first RNI.

V008: EXX2 to F1.
V009:
V010: Test interrupt.
V011: Input H080.
V080: Set K152/153 (Program Request); when J179 = 0 the Group/Program scanner will hang up set/set, indicating program priority.
Asyncronous: Input H170 when Program Priority and $\overline{\text{R. F. Busy}}$.
V171: Set K168/169 (Program Register File Priority); input H016; initiate RNI to read second word of the instruction.
N014: Set K000/001 (Request Bus); input to H231.
N231/N051: Set K010/011 (Main Control Priority); advance P2; input H220.
N220/N050: P2 to P1; input H117.
V117: Set K116/117 (Storage Request).
N050: Input H115.
V115: Set K012/013 (Enable Data Bus).
V116: Test BPI.

Main Control now waits for Reply from Storage.

V061: Resynced Storage Reply; set K060/061 (1st Cycle) which blocks sensing interrupt for this RNI.
V000: Clear K000/001 (Request Bus).
V001-V004: Access time.
V005: Set K062/063 (2nd Cycle).
V006: Input to H401; Block input to H201.
V007: Clear K010/011, K012/013, K116/117, DBR; input to H410; block input to H200.
V008: EXX2 to DBR, EXX2 to F1 does not occur; set K164/165 (Read and Write). Note that at this point in the timing, F holds (P) and DBR holds (P + 1). Set K176/177 (Gate Channel 0); remove clear inputs from K15X Channel designator FFs; gate channel designation to K15X channel designator FFs. Note that the actual channel designated is the inclusive OR of CIR and bits 21, 22, and 23 of DBR. When the contents of the Channel Designator FFs have been translated (Logic Diagrams, page 2-165) a Channel Enable signal will be transmitted to the appropriate channel.
V009:
V010:
V011:
V080: Input H151; if selected channel $\overline{\text{Busy}}$, set K108/109 (Skip); if selected channel Busy, set K044/045 (Reject Read and Write).

If Channel Busy

Block Control Timing

V151: Input H014; input H156.
V156: Clear K152/153 (Program Request); Clear K164/165 (Activate Read and Write); clear K176/177 (Enable channel 0); drop the Channel Enable.
V157:
V158: Clear K168/169 (Program Register File Priority).

Main Control Timing

N014: Set K000/011 (Request Bus); input H231.

N051/N231: Advance P2; set K010/011 (Main Control Priority); input H220.
N050: P2 to P1; input H117; RNI at P + 2 is in progress.

If Channel $\overline{\text{Busy}}$. For this case, if $\overline{\text{I/O with A}}$, three timing elements will be running simultaneously. This timing chart will not show the transfers associated with the Main Control timing but Main Control times will be listed to the left of the Block Control times.

N161/V151: Input H152, H014; if Word Addressed instruction (Backward + 3307 selected) Word Addressed instruction (Straight Transfer + Character Addressed Instruction (12 to 24), set K368/369 (Force Character Address); set K180/181 (Program Load S).

N014-V152:

N239-V153:

V154: Set K170/171
(End of 1st pass)

N217-V155: Input H150.

N014-V150:

N051-V151: Input H400.

N050-V152: Set K196/197 (Clear O); input H401; set K174/175.

V117-V153: Set K172/173 (2nd pass); clear DBR; input H410; set K048/049 (WCC) if (Read) (Bit 20).

N050-V154: Clear K196/197; EXX2 to DBR (F to I^7 to EXX2).

V115-V155: Set K194/195 (Bus to O).

V116-V156: Clear K180/181 (Program Load S); K170/171 (End of 1st Pass), K172/173 (2nd Pass), K152/153 (Program Request).

V061-V157: Clear K194/195 (Bus to O), K174/175 (Read + Write), K048/049 (WCC).

V000-V158: Clear K168/169 (Program Priority).

RNI in progress at P + 3.

The relationship of the timing shown below to Block Control timing is an approximation.

G680, G682, G683, G686 pulse for 100 nsec, gate channel to S0 lower 3. This is the OR of DBR upper 3 and CIR.
Set Z332/333 (Start Delay Line), Z334/335 ($\overline{\text{Increment}}$), Z370/371 (Register File Write), and Z390/391 (Register File Busy). Record Activate at G642/G643/G644 by forcing G643 = 1 (3 state FF) · Clear Z1.

(A3) E353: Read Current On, clear S_2.

(A4) E351: DBR to Z1, A_{L3} to Z1, clear Z0, Block Sense Amps to Z1.

E388: Clear Z332/333 (Start Delay Line).

(A7) E356: Z_1 to Z_0, Z_1 to S_2, S_0 to S_1.

(A10) E357: If Interrupt on Completion, set K192/193.

E359: Block Z1 to Z0 (Increment/Decrement).

(B3) E355: Clear Z1.

(B4) E367: Force S0 to 1X.

(B6) E365: Z0 to Z1.

(B8) E363: Write and Digit Currents On.

(C2) E368: Clear Z1.

(C5) E370: Read Current On.

(C6) E371: DBR to Z1.

(C9) E376: S0 to S1.

(D8) E385: Clear Z334/335 ($\overline{\text{Increment}}$).

(E1) E380: Write and Digit Current On.

(E6) E387: Clear Z390/391 (Register File Busy).

(E7) E389: Clear Z368/369 (Modify Address), K164/165 (Read + Write), K176/177 (Gate Channel 0), and K15X Channel Designator FFs and Drop Channel Enable.

TIMING FOR ACTIVATE SEARCH OR MOVE BUFFER, 71 OR 72 INSTRUCTIONS

Timing begins at V008 of the first RNI.

V008: EXX2 to F_1.
V009:
V010: Test interrupt.
V011: Input to H080.
V080, V180, V280: Set K152/153 (Program Request), J179 is driven to a 0. Block Control's Priority Scanner will eventually recognize the request by hanging up with Z300/301 and Z302/303 in their set state. J799 = 1, delay Y951 = 1, Resync via H170/171.
V171: Input to H016; set K168/169 (Program Priority Granted).
V014: Set K000/001 (Bus Request); input to H231.
N051: Set K010/011 (Bus Priority); Advance P_2; input to H220.
N050: Input to H117; P_2 to P_1.
V117: Set K116/117 (Storage Request).
N050: Input to H115.
V115: Set K012/013 (Enable Data Bus); input to H116.
V116: Test Breakpoint Stop if BPI selected.
V061: Resynced Storage Reply; set K060/061 (48-bit 1st cycle). This blocks Sensing for all interrupts.
V000: Clear K000/001.
V001:
V002:
V003:
V004:
V005: Set K062/063 (48-bit 2nd cycle).
V006: Input to H401; Block Input to H201.
V007: Clear K010/011, K012/013, K116/117, DBR; input to H410; Block Input to H200. NOTE: F_1 is not cleared.
V008: EXX2 to DBR. NOTE: EXX2 to F_1 does not occur.
V009:
V010:
V011: Input to H080.
V080, V180, V280: Input to H151; if S/M is not Busy, set K108/109 (Skip).

Timing for S/M Not Busy is continued on next page. below.

Timing for S/M Busy:
V151, N151: Input to H014 ... Input to H156.
V014: Set K000/001 (Bus Request); input to H231.
N051: Set K010/011 (Bus Priority); advance P_2; input to H220.
N050: Input to H117; P_2 to P_1; RNI at P+2 (effective).
V156: Clear K152/153 (Program Request). (Priority Scanner looks for BFR Request).
V157:
V158: Clear K168/169 (Program Priority Granted). Block Control has been released and is free to honor BFR Requests.

See next page for actual activation of a Search or Move Buffer.

The following sequence occurs at the end of the 2nd RNI Cycle providing Search/Move Control is not Busy.

N151: Input to H014.
V151: Input to H152.
N161: Set K180/181 (Program Load S), G682/G683/G686: 100 nsec pulses. Force 21 or 20 to S_0 (J171/G67X); set Z334/335 ($\overline{\text{Increment}}$), Z370/371 (Register File Write), Z390/391 (Register File Busy), Z332/333 (Pulse Generator) and Clear Z_1; Record Activate at G642/G643/G644.

(V014) V152:
(H239) V153:

(N226) V154: Set K170/171
(End of 1st Pass)
(N229) V155: Input to H150.
(V014) V150:
(N051) V151, N153: Input to H400; if Search, clear Z340/341, if Move, set Z340/341, Z342/343 (S/M Request), Z336/337 (S/M 1st Character), and Z338/339 (S/M Busy). If (Move) (m_1 and m_2 designate character 0) (Character count = $XXXXX00_2$), set Z330/331 (Word Move).
(N050) V152, N400: Input to H401; set K196/197 (Clear O).
(V117) V153, N401: Set K172/173 (2nd Pass); clear DBR; input to H410.
(N050) V154, N410: Clear K196/197 (Clear O); EXX2 to DBR (F to I^7 to EXX2).
(V115) V155: Set K194/195 (Bus to O).
(V116) V156: Clear K180/181 (Program Load S), K170/171 (End of 1st Pass), K172/173 (2nd Pass), and K152/153 (Program Request).

(V061) V157, V167: Clear K194/195 (Bus to O).
(V000) V158: Clear K168/169 (Program Priority).
(V001)

Access and Decode time for RNI at P + 3 (effective).

(A3) E353: Read Current On; Clear S_2.
(A4) E388: Clear Z332/333 (Pulse Generator).
(A4) E351: DBR to Z_1; Clear Z0; Block Sense Amps to Z_1.
(A7) E356: Z_1 to Z_0, Z_1 to S_2, S_0 to S_1.
(A10) E359: Block Z_1 to Z_0 (Increment/Decrement).
(B3) E355: Clear Z_1.
(B4) E367: Advance S_0 to 3X (30 + 31).

(B6) E365: Z_0 to Z_1.

(B8) E363: Write and Digit Currents On.

(C2) E368: Clear Z_1.
(C5) E370: Read Current On.
(C6) E371: DBR to Z_1; set Z354/355 if bit 17 = 1 (DBR); Block Sense Amps to Z_1.
(C7) E376: S_0 to S_1.

(D8) E385: Clear Z334/335 ($\overline{\text{Increment}}$).
(E1) E380: Write and Digit Currents On.
(E6) E387: Clear Z390/391 (Register File Busy).
(E6) Block Control is free to honor incoming request of highest priority.

SELF-EVALUATION QUIZ ON CHAPTER 15

TRUE OR FALSE OR FILL IN THE BLANKS:

1. Block control contains a word-organized storage unit which has a ____ x ____ x ____ matrix.

2. Each time the register file is accessed, two read/write cycles take place.

3. Locations 34_8 through 77_8 of the register file are available to the programmer for temporary storage.

4. Location 22_8 of the register file could be used by the programmer.

5. Location 17_8 of the register file could be used for temporary storage in many systems.

6. Both a search and a move may be in progress simultaneously.

7. The channel specified in coding an activate instruction is an absolute channel.

8. All 27_{10} bits of a register file are accessible to the programmer by means of the IRT instructions.

9. During buffer cycle the increment/decrement network is used to update all 20 bits of the current address.

10. The current address of I/O cycles is always a character address.

11. For 12 to 24 I/O operations increment/decrement is by 2_8.

12. Block requests and program requests have equal priority in accessing block control.

13. During the second RNI of an activate the transfer of EXX2 to F1 does not occur.

14. The upper 3 bits of address for activate instructions is obtained from the lower 3 bits of the A register.

15. The upper three bits of location 1X of the register file contain a modified code which specifies what type of input/output is in progress.

Score Yourself:

You could miss one of these and still be considered above average.

You could miss two and be considered average.

But if you missed more than two, don't spread it around because you're below average!

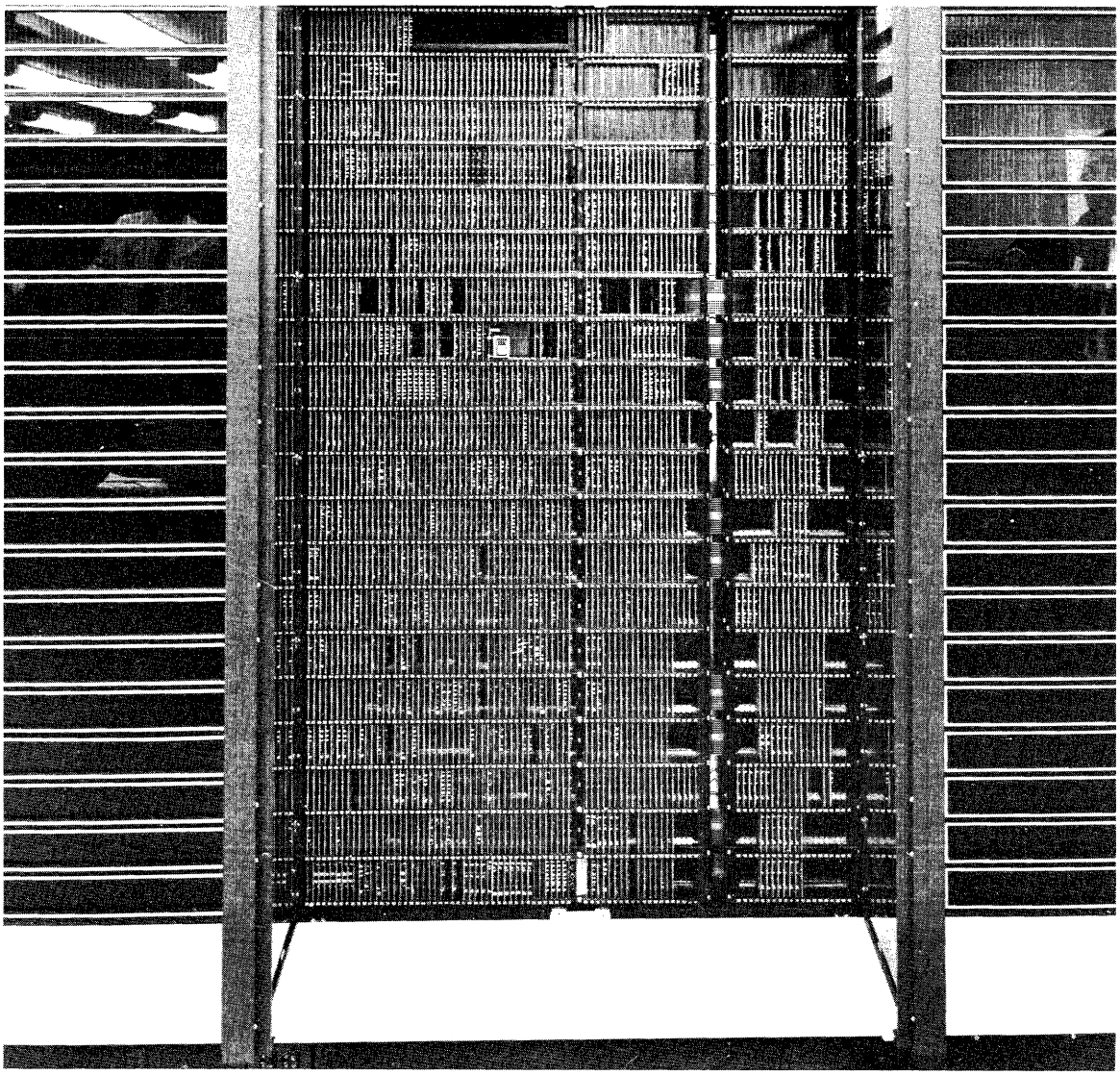


Figure 359. Communications Modules

CHAPTER 16

COMMUNICATIONS MODULES

GENERAL DESCRIPTION

Figure 359 is a front view of two Communications channels. A maximum of eight channels may be attached to any single CPU. There are two types of I/O channels available for the 3300 System. The 3306 is a 12-bit bidirectional channel; a maximum of eight 3306s may be attached to a single CPU. The 3307 is a 24-bit bidirectional channel; a maximum of four 3307s may be attached to a single CPU. If present in a system, 3307s will always be even-numbered channels. Each Communications module may contain two I/O channels. Figure 360 illustrates the physical position of the channels.

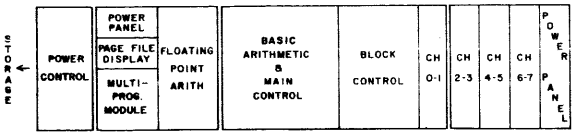


Figure 360. 3300 Computer Physical Layout

Figure 361 is a block diagram of a Communications module.

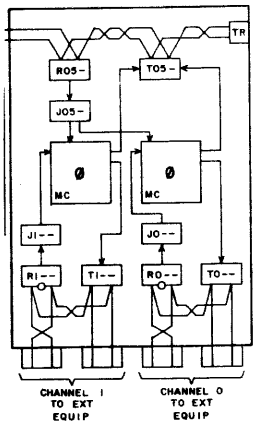


Figure 361. 3300 Communications Module Block Diagram

3306 COMMUNICATIONS CHANNEL

FUNCTIONAL DESCRIPTION

Communication channels provide a buffer between the computation section and various peripheral equipment controllers. They prevent the computation section from the external equipment. A program must go through several instructions prior to exchanging data with a piece of external equipment.

1. It must connect the equipment to the channel.
2. It must set up and direct the equipment with external function codes.
3. It could sense internal and external status conditions.
4. It must initiate a read or write operation.

A 3306 Communication Channel provides a 12-bit, bidirectional, buffered, input/output path between the processor and up to eight peripheral equipment controllers. Since there may be up to eight channels, one processor may communicate with up to 64 controllers. Each controller in turn may be attached to a number of units. For example, the 3329 Magnetic Tape Controller may be physically connected to eight tape transports. This makes it physically possible for one processor to communicate with 512 tapes, an unlikely situation, of course. All data is handled in bytes (groups of 12 bits) with one parity bit per byte.

OPERATION

An I/O module contains no indicators or controls, so all operation must be initiated by the program via the computation section of the computer.

PROGRAMMING

The 15 instructions listed in table 30 are directly related to the I/O operations of a 3300 Computer. For a detailed description of these instructions, refer to the 3300 Computer System Reference Manual.

Table 30. 3300 I/O RELATED INSTRUCTIONS

OPERATION FIELD	ADDRESS FIELD		INSTRUCTION
77.3 INS	x, ch	$x \neq 0$	Sense internal status
77.3 CINS	x, ch	$x \neq 0$	Copy internal status into lower 12 bits of A
77.0 CON	x, ch		Connect to external equipment
77.2 EXS	x, ch	$x \neq 0$	Sense external status
77.2 COPY	x, ch	$x \neq 0$	Copy external status into lower 12 bits of A
77.1 SEL	x, ch		Select function of external equipment
73 INPC, INT, B, H	ch, r, s		Input character block to storage
73 INAC, INT	ch		Input character to A
75 OUTC, INT, B, H	ch, r, s		Output character block from storage
75 OTAC, INT	ch		Output character from A
74 INPW, INT, B, N	ch, m, n		Input word block to storage
74 INAW, INT	ch		Input word to A
76 OUTW, INT, B, N	ch, m, n		Output word block from storage
76 OTAW, INT	ch		Output word from A
77.51 IOCL			Selectively clear I/O channel

For specific function codes and status bits refer to the 3000 Series Computer System Peripheral Equipment Codes.

THEORY OF OPERATION

This area of the chapter describes the theory of operation of a 3306 Communication Channel. It is broken down into six main parts:

1. Data paths
2. Parity checking
3. Status checking
4. Clearing circuits
5. Interrupts
6. I/O operations

The following paragraph is a rapid run-through of the sequence of events. Figure 362 is a flow chart of the sequence of events.

In response to programed instructions, the computation section connects an I/O channel and one of its external equipments, and sets up the equipment for a read or write operation with function codes. The channel then proceeds with the read or write operation, obtaining access to storage or A register via block control (in central processor) when necessary to fetch or store information. The entire operation is performed by block control and the channel, allowing the computation section to continue its main program.

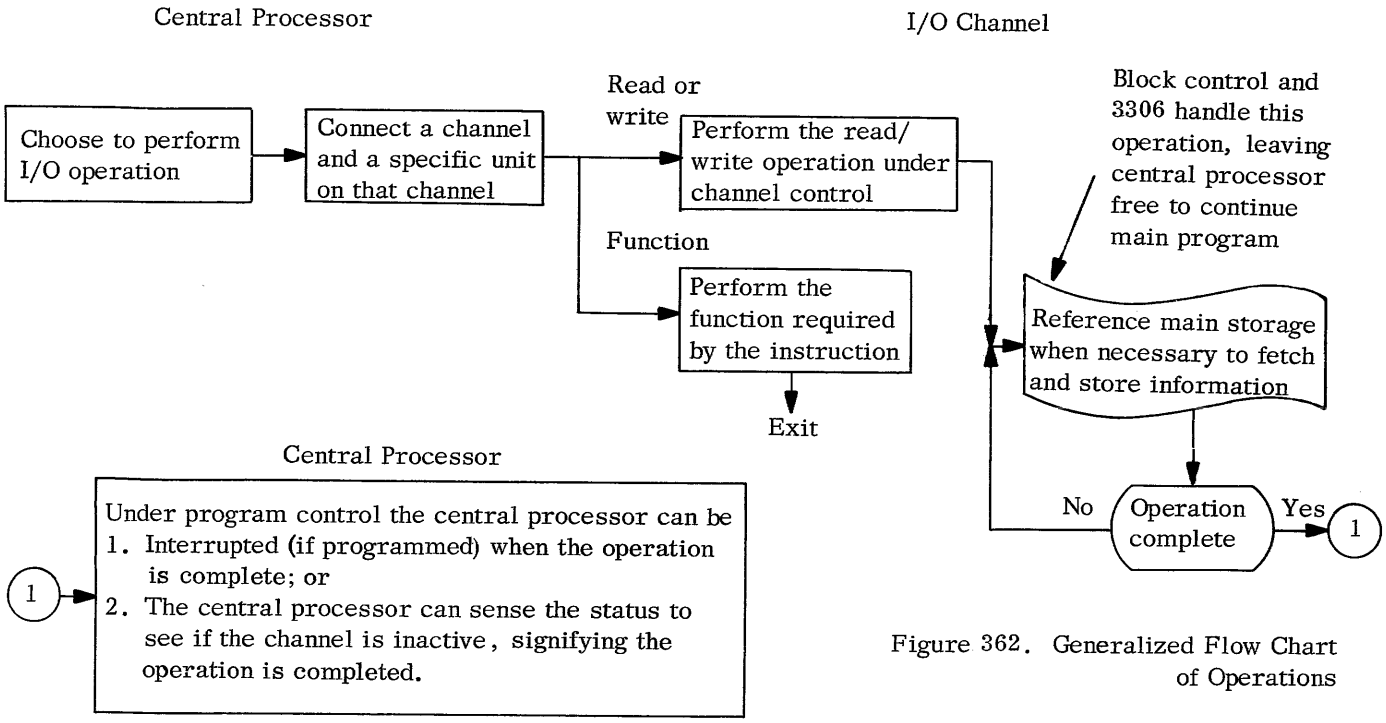


Figure 362. Generalized Flow Chart of Operations

All data exchanged between a 3300 and its external equipments is in the form of 12-bit bytes accompanied by one parity bit per byte. Odd parity is used exclusively. Six bit characters use only half of a channel the lower six bits. All assembly and disassembly of computer words is done by block control or the controller, not by the 3306.

Figure 363 is a more detailed flow chart of the sequence of operations.

DATA PATHS

In each standard I/O section, there is an even and an odd buffer (O) register, one for each 3306 Communication Channel. Each channel has its own set of receivers and transmitters between the O register and external equipment jacks. However, the pair of channels in each I/O section share the set of receivers and transmitters between the data bus and their respective O registers.

There is one other data path. The proper O registers receive information from and transmit information to the data bus. This is done by the circuits in figure 364.

Note that the communications channel uses common transmitters back to the data bus just as the O registers receive from the data bus via common receivers.

A quick glance at figure 365 will convey what has just been covered.

Data which may consist of a connect code, a function code, or an operand, is gated from the data bus (in the processor) to O register (in the communication channel) by a signal from block control. The data moving from the communications channel to external equipment (the (O) to external equipment signal) is controlled by each channel. Input data is gated from external equipment to O register by a signal within the channel (figure 366).

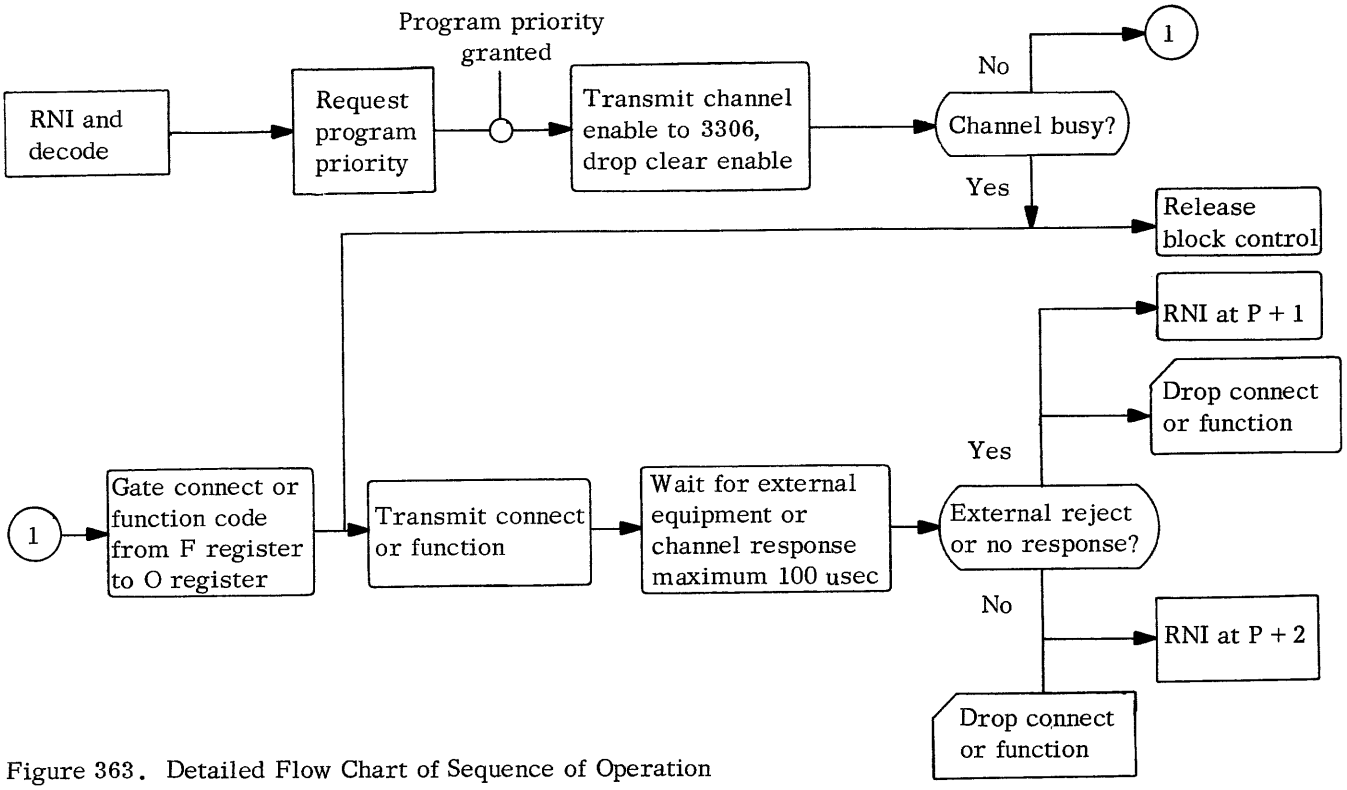


Figure 363. Detailed Flow Chart of Sequence of Operation

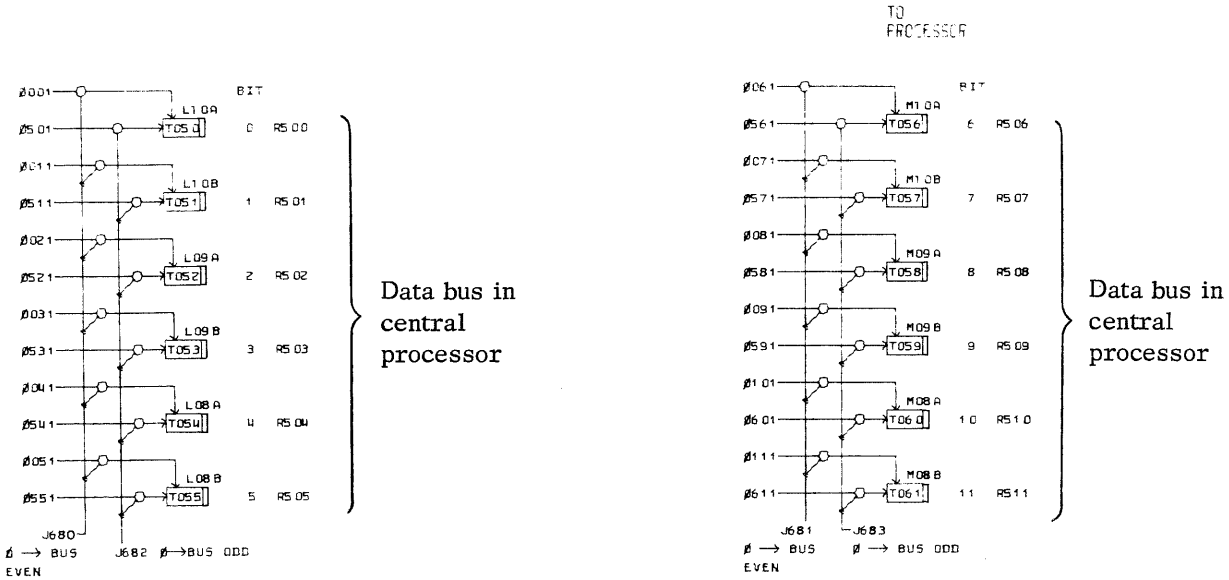


Figure 364. Data Bus Transmitter Logic

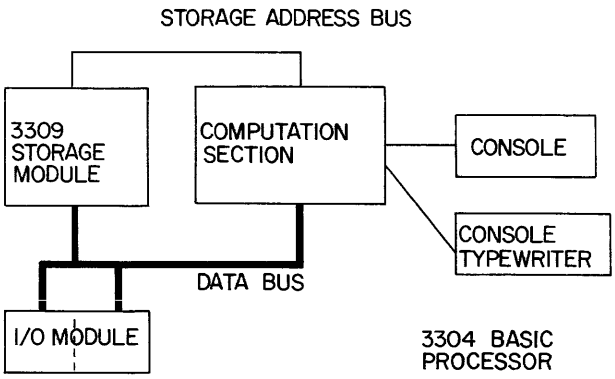


Figure 365. 3304 Processor with Data Bus Emphasized

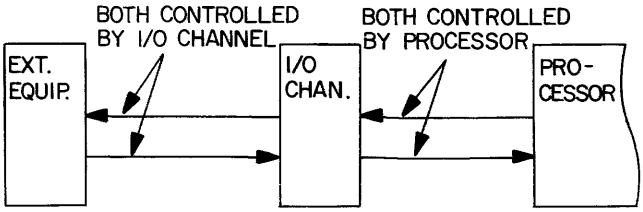


Figure 366. I/O Data Flow

THEORY OF OPERATION

The normal operation of a program might be:

1. Connect a specific piece of peripheral equipment to the channel.
2. Perform either a read or write operation.

In the next portion we will discuss in detail only the even channel. Remember that the even and odd channels are identical in operation and each channel contains identical circuits. The circuits common to both are the receivers and tramsmitters coming from and going to the data bus in the central processor.

The following section will cover, by individual circuit analysis, a connect or function operation on the even channel and a write operation on the even channel.

Connect or Function Operation

When the central processor executes either a 77.0 or 77.1 instruction the function translators (in the central processor) will supply a constant control enable to the designated channel. This constant translation of the 77.0 or 77.1 instruction (control enable) will remain up until another instruction is read to F register, which will start another RNI.

While the 77.0 or 77.1 instruction is being executed several separate things must take place. For a quick glance of what must take place see figure 367. Refer back to this figure as the operation of a connect is explained.

Figure 368 shows a function operation. Note the similarity between connects and functions. Only connect is discussed in detail in this manual. A function operation is identical to a connect operation. A connect establishes contact or connection between the processor and a piece of peripheral equipment. A function instruction accomplishes some function, such as having a magnetic tape handler perform a rewind operation. The function operation may prepare the controller for certain controlling functions such as BCD or binary mode, write a file mark, etc.

Function operation uses the same circuitry and the responses to a connect are the same as to a function. If a solid understanding of the connect operation is obtained in the next few pages, understanding a function operation will be second nature.

The purpose of the connect operation is to establish a connection to a piece of external equipment.

There are several signals that are transmitted to the communications channel from the processor, during the execution of the connect (or function) instruction Some signals are transmitted back to the processor from the communications channel also.

A list of these signals with a brief description of the purpose of each is supplied on the following pages. Read through them carefully to get the big picture. Remember, the basic operation to be performed is to establish a connection between the processor and a piece of external equipment.

Channel Control Enable: This signal comes from the processor to one of eight channels to enable connect, function, and interrupt clear.

Channel enable: The signal comes from the processor to one of eight possible channels.

Clear enable: Statically the signal is always here to enable clearing the channel. It disappears now to allow test for busy. Either the channel is busy or not busy and the connection can or cannot be made.

Clear O: This signal is from the processor and clears the O register.

Bus to O: This signal enables the information to come from the processor to the channel.

Connect or function: This signal from the processor tells which operation is to be done.

O to external equipment: This signal from the channel relays the O to the controller. The connect or function signal is relayed to the controller at this time.

Reply: Signal indicating connect or function operation has been accomplished.

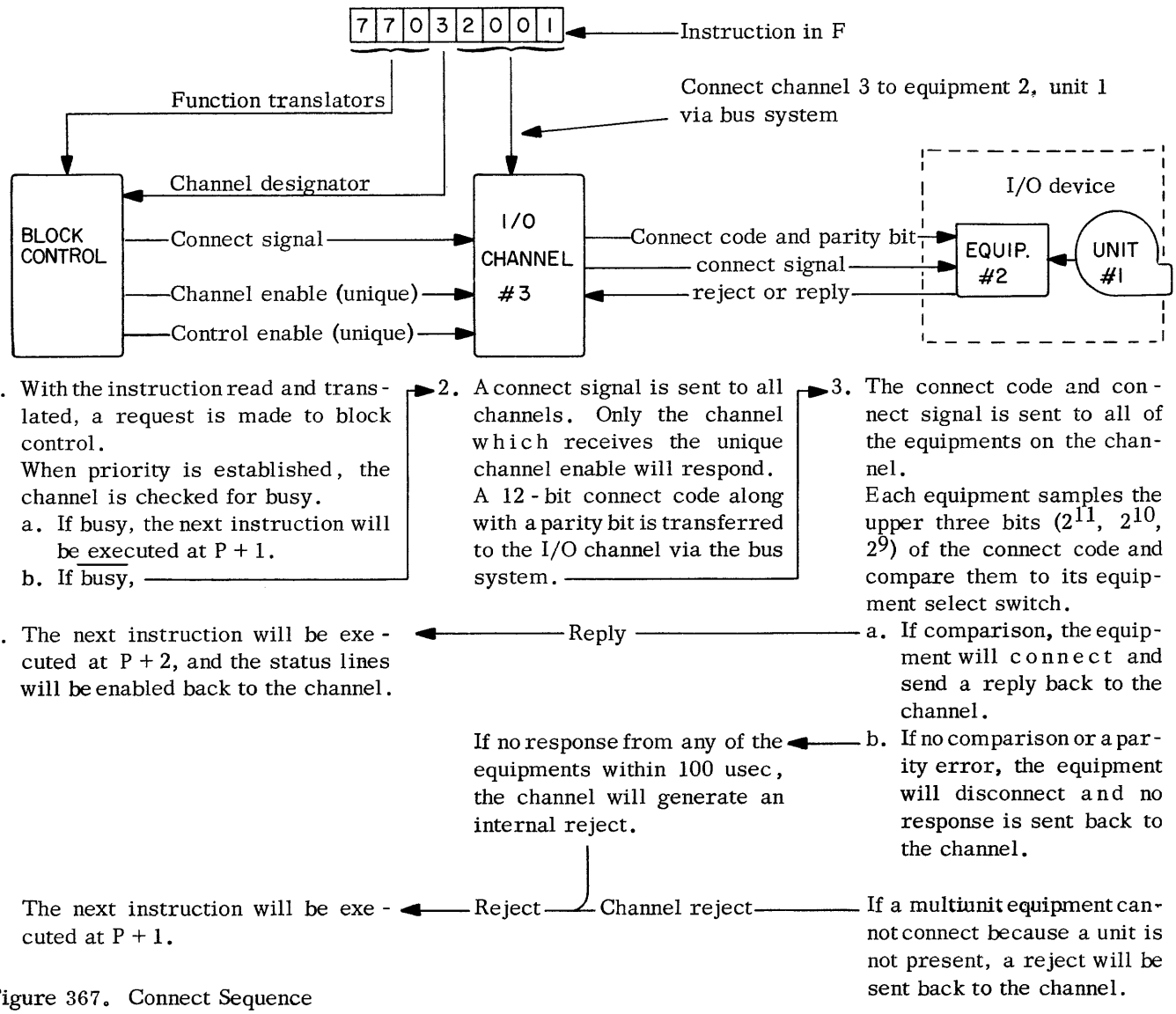


Figure 367. Connect Sequence

Reject: Signal indicating connection is impossible at this time (unit not ready, for example).

No response: Signal indicating connection is impossible (no unit exists, for example).

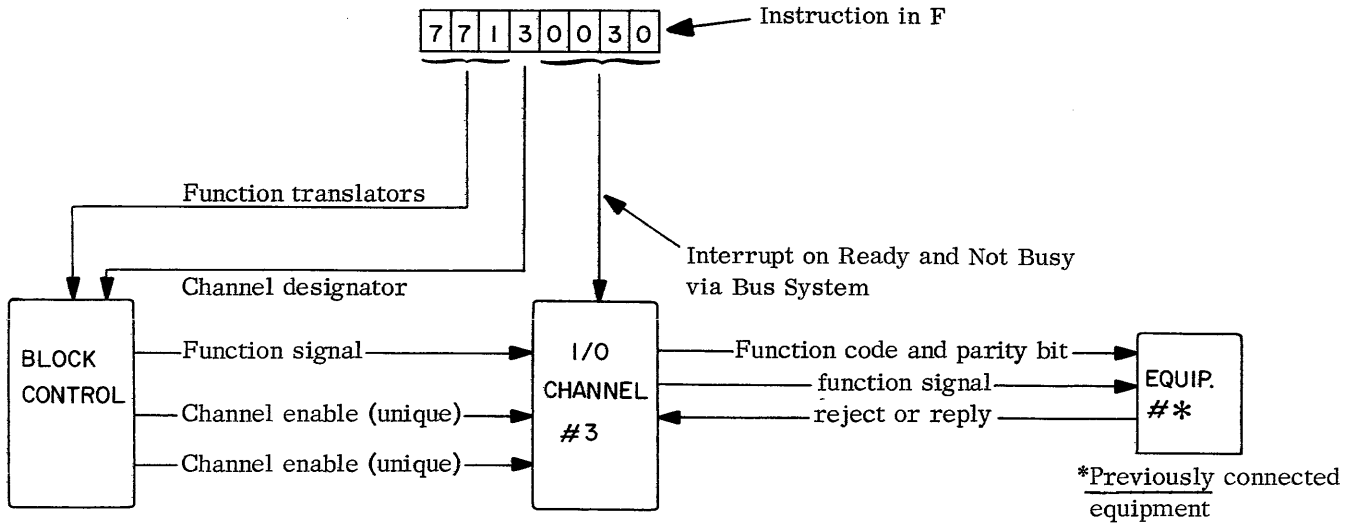
While the connect or function instruction is being executed several signals are transmitted to the I/O channel by the central processor. One of the first is:

1. Channel enable - The processor is executing an input/output instruction and is enabling further communications with this particular channel. Channel enable causes output of J074 = 1.

2. Clear enable - This signal drops at this time. Normally a constant clear enable is held at certain circuits within the channel, but during communications with the processor this signal must drop to allow valid checking of the channel busy.

Drop clear enable. R084 = 1 (see above).

NOTE: When the computer is on, clear enable is present at all times except when the processor is checking for channel busy.



1. With the instruction read and translated, a request to block control is made.
   When priority is established the channel is checked for busy.
   a. If busy, the reject instruction at P + 1 will be read.
   b. If not busy

2. A function signal is sent to all of the channels. Only the one which receives the unique channel enable will respond. A 12 - bit function code along with a parity bit is transferred to the channel via the bus system.

3. The function code and function signal are sent to all of the equipments tied to the channel, but only the previously connected equipment will respond.
   The function code is decoded.
   a. If the function can be accomplished, a reply will be sent back to the channel.
   b. If the function cannot be accomplished, a reject will be sent back to the channel or no response will be returned.

4. The next instruction will be executed at P + 2.
   The next instruction will be executed at reject address P+1.

Reply

Reject

Channel reject

If no response within 100 usec, the channel will generate an internal reject.

Figure 368. Function Sequence

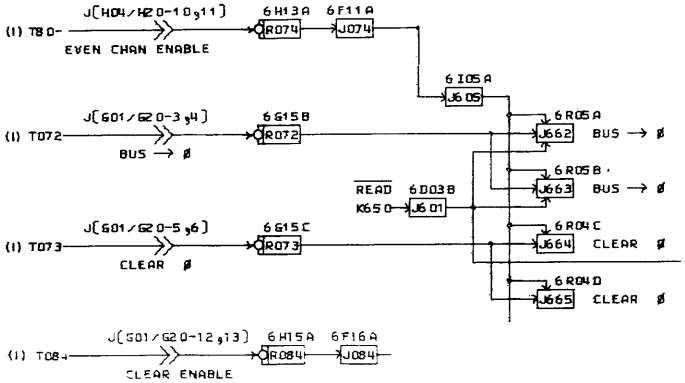


Figure 369. O Register Enables

3. At this point the processor has tried to establish communications with the channel. One of two things could have occurred: either the channel is busy and cannot communicate now, or it is not busy and can communicate. The channel would be busy if it were doing a read, write, or master clear.

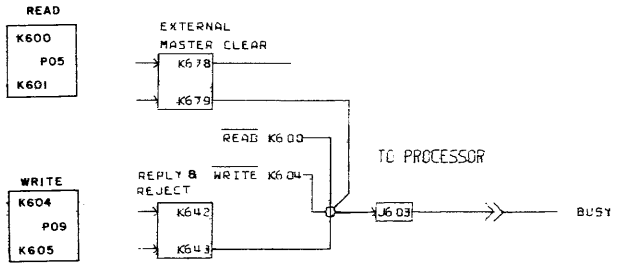


Figure 370. Channel Busy to Processor

If channel not busy (J603 = 0) the central processor will perform the following transfers to the data bus transmitters: F to I^7 to EXX2 to $\overline{DBR}$ to T5XX to transmit the connect or function code to the channel. (The channel is not busy if it is not doing a read, write, or master clear.) If channel busy, RNI at P + 1.

4. Clear O register: This signal clears the O register in preparation for the arrival of information from the data bus. On a connect or function it is the unique 12-bit code that will accomplish the operation. Clear O register from the block control (R073 = 0, J664 and J665 = 1).

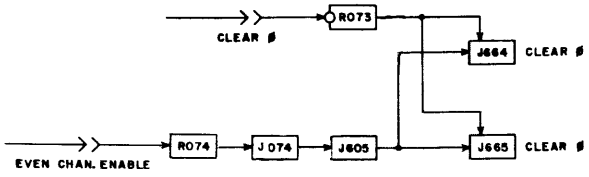


Figure 371. Clear O Register Enable

When clear O pulse arrives (R073 = 0), and when the even channel is enabled (R074 = 1, J605 = 0), then the output of J664/665 is a logical 1, clearing O register.

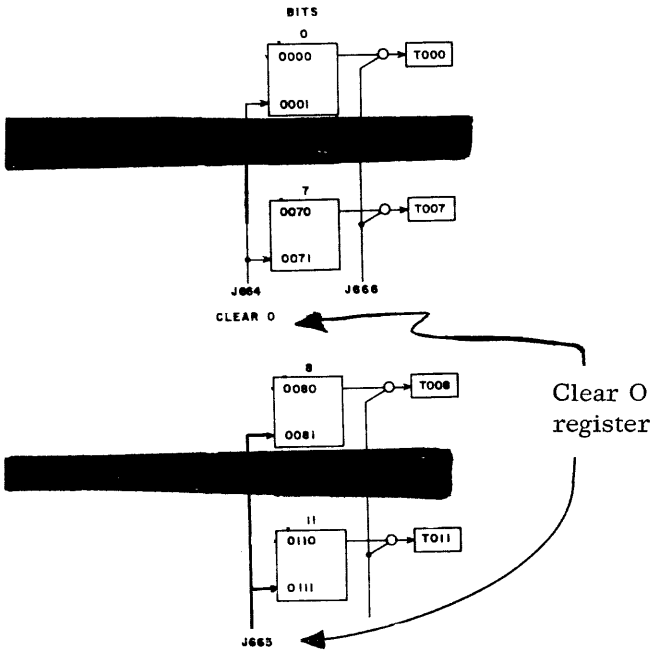


Figure 372. Clear O Register Enable

5. Bus to O: The signal comes from the processor and enables the specific code from the instruction to enter the O register in the channel. The clear O register signal will drop out at this time to allow the entry of the code to the O register.

Bus to O from block control (R072 = 0, J662 and J663 = 1).

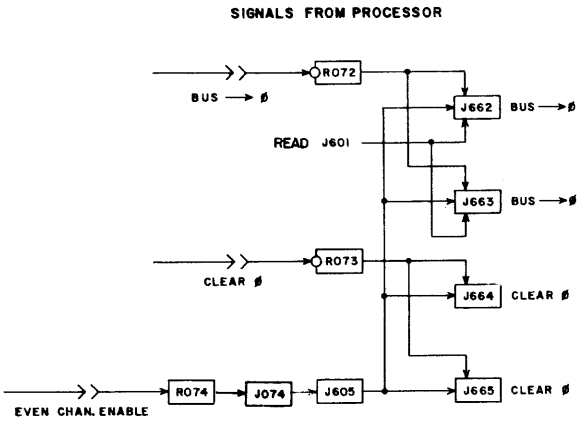


Figure 373. Bus to O Register Enable

6. Connect or function: This signal comes from the processor to the channel and then is relayed to the controller to indicate a connect or function code is on the lines.

Connect or function from the block control (R081 or R082 = 1 and R086 = 1 to cause J602 = 1).

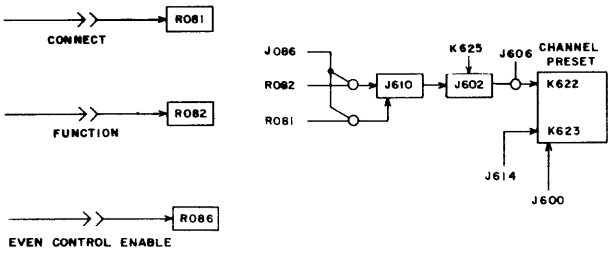


Figure 374. Channel Preset Status

At this point the connect or function instruction has been read by the processor and the connect or function code has been sent to the channel. The next action that must take place is the channel relaying the connect code to the external equipment controller and then the controller reacting. J604 = 1, clearing K620/621, K608/609, and K612/613.

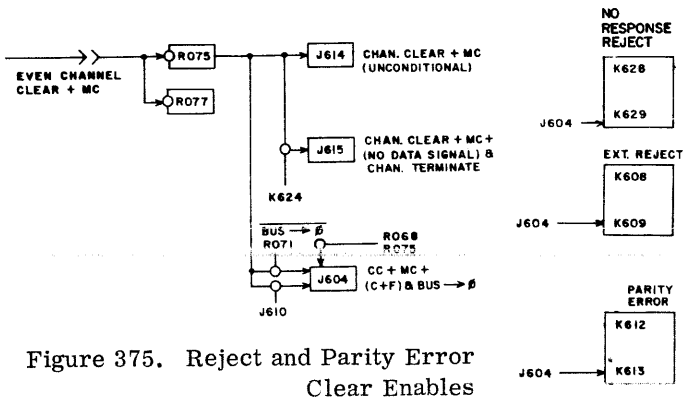


Figure 375. Reject and Parity Error Clear Enables

7. O to external equipment: This signal comes from block control to the channel. It enables the information (connect or function code) in the O register to the external equipment.

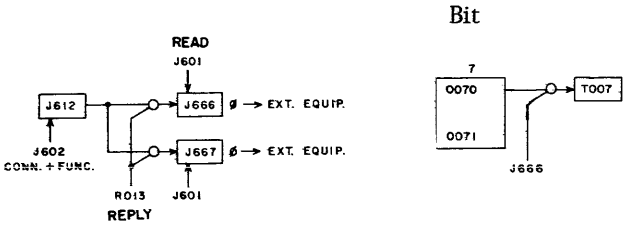


Figure 376. O Register to External Equipment Enables

During O to external equipment, J602 forces J612 to 0, J666 and J667 enable O to external equipment. Only one bit of the O register is shown.

After 0.2 usec, connect or function is sent to external equipment via T015 to T016.

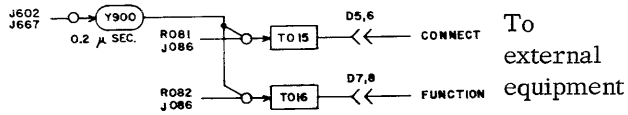


Figure 377. Connect and Function Signals

Main control hangs up, waiting for one of the following three responses:

1. Reply
2. External reject
3. No response reject

Block control is released and drops the channel enable. The clear enable will come back up.

Reply: In this instance the signal came back to the controller, indicating the controller is connected. The channel then relays this reply to the processor so the processor will know that the connection was made.

If external equipment will reply, R013 = 1 and J606 = 1. J606 being a 1 and the output of J602 still being a 1 will allow setting of K622/623.

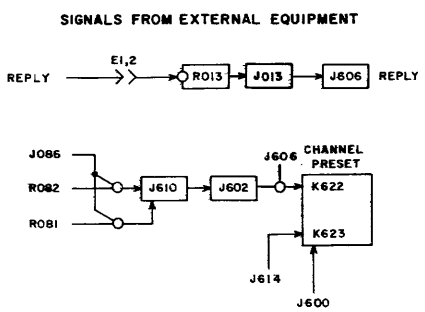


Figure 378. Channel Preset Enables

K622/623 going set would cause a 1 on sense line 6 if a sense instruction were executed, bringing J678 to a 1.

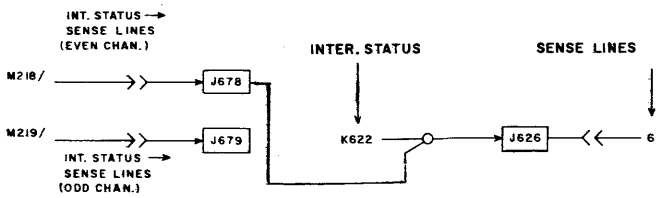
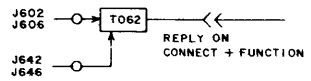


Figure 379. Channel Preset Status

T062 will transmit a reply to main control.

Figure 380. Reply on Connect or Function



Control drops control enable and connect + function. O to external equipment is dropped. Connect and function to external equipment is dropped. Main control does an RNI at P + 2.

External Reject: In most cases this signal comes back from the controller indicating that connect cannot be performed to a multiequipment controller or that the function cannot be performed. (The reason for this signal returning depends on the specific external equipment controller being used. For exact meaning of these signals refer to specific controller books.)

If external equipment rejects, R014 = 0 and J607, J608 = 1, set K608/609 (external reject). K608/609

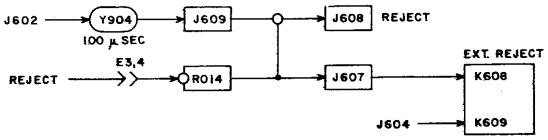
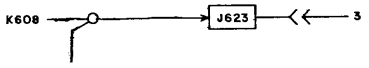


Figure 381. External Reject on Connect or Function

would cause a 1 on sense line 3 if a sense instruction were executed, bringing J678 to a 1.



J678 Figure 382. External Reject Status

T076 = 1, reject to main control.

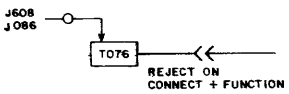


Figure 383. Reject Signal to CPU

Control enable and connect or function drop. Main control performs an RNI at P + 1.

No Response Reject: This signal, or lack of a signal means that the connect cannot take place or that the connect or function has a parity error. For specific details refer to the specific controller books.

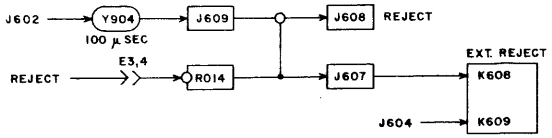


Figure 384. External Reject Enables

The I/O module is no response reject, after 100 usec J609 = 0, J607 and J608 = 1.

K608/609 set would cause a 1 on sense line 3. Y904 sets K628/629. K628/629 set would cause a 1 on sense line 4.

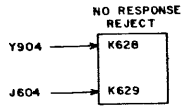


Figure 385. No Response Reject FF

The rest of the circuitry involved with no response is identical to that used for normal reject.

T076 = 1, reject to main control. Control enable and connect or function drop. Main control performs an RNI at P + 1.

The previous description for operation of a connect or a function instruction for even channel applies to odd channel, except that different circuitry is used.

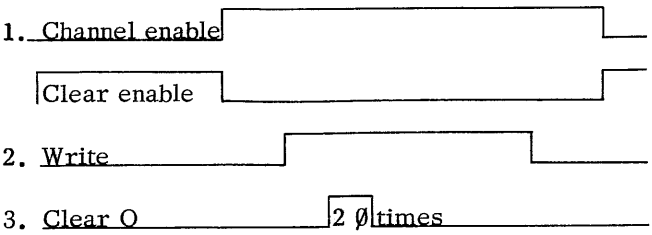
The channel has now been connected via instruction between the central processor and the external equipment controller.

The original connect or function code was sent to the channel from the processor. The channel relayed the code to the external controller. The controller accomplished either the connect or function and sent the appropriate response back to the channel, which relayed this back to the processor.

Study figure 388 for a quick review of the overall operation.

WRITE OPERATION

The next process that will be covered is write operation. Write operation could be directed to any output device such as a card punch, line printer, or magnetic type unit. The following signals are sent from the central processor to the 3306 Channel:



Channel enable is a 1 signal from block control that is sent only to the channel being used. Its purpose is to prevent the simultaneous transmission or reception of data by more than one channel at a time.

1. Channel enable: R068 = 0, J605 = 0. Drop clear enable, J084 = 0.

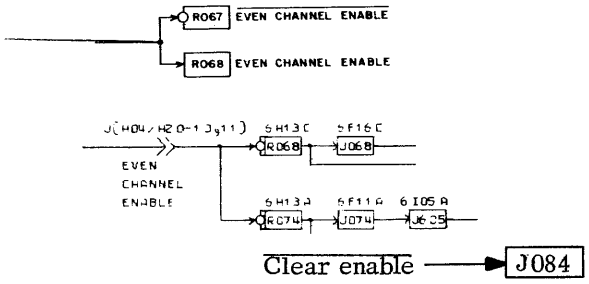


Figure 386. Channel Enable

Write is a 1 signal from block control to the channel to initiate the write operation. The signal is dropped by the reply or reject.

2. Write: J080 = 1, set K602/603 (block control request), T031 = 1. 100 nsec after K602/603 sets, K614/615 will set.

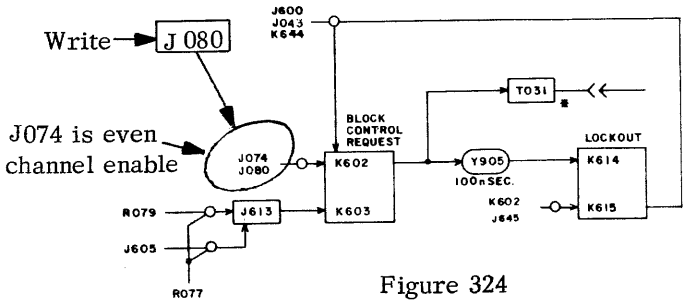
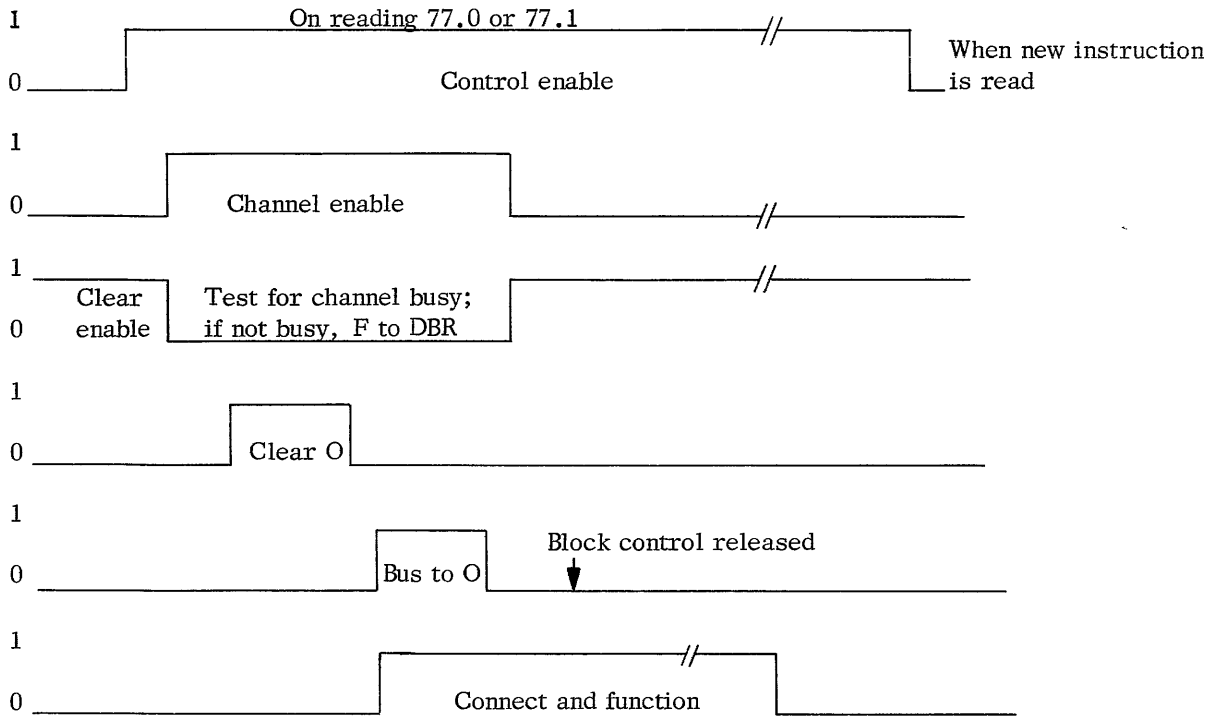


Figure 324

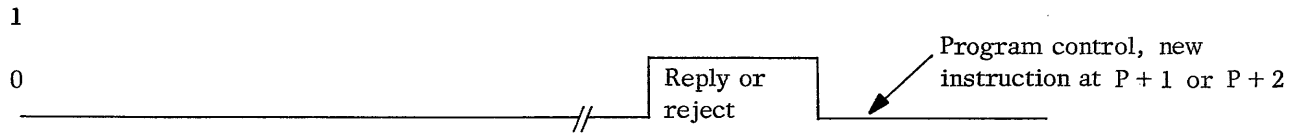
Figure 387. Block Control Request upon Activate Write

Clear O is a 1 signal from block control that clears all flip-flops in O register.

A. Signals for Block Control



B. Signals From External Equipment



C. Signals to External Equipment

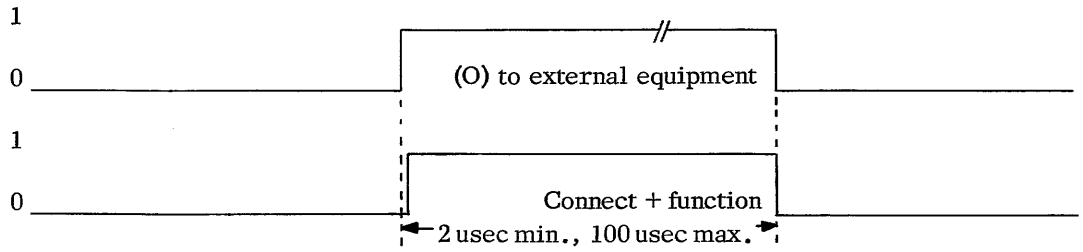
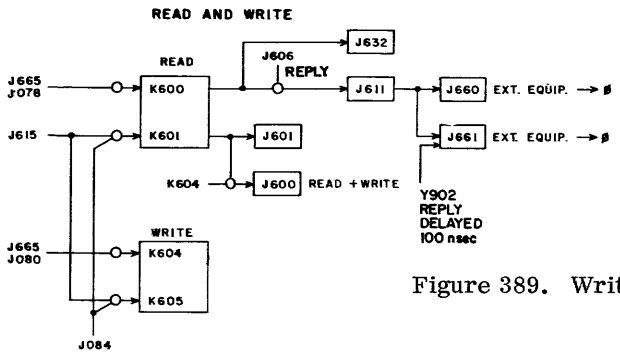


Figure 388. Timing for Connect or Function for Even Channel (77.0 + 77.1)

3. Clear O: J664 + J665 = 1; set K604/605 (write), T014 = 1 (write to external equipment).



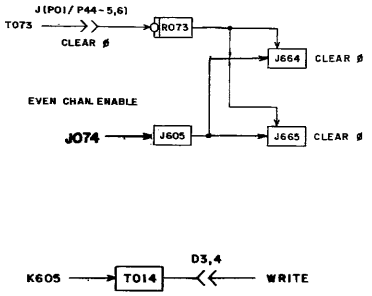


Figure 389. Write FF and Transmitter

J600 clears K622/623 (channel preset), which was set on the previous connect or function.

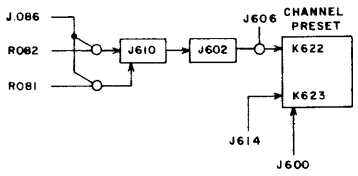
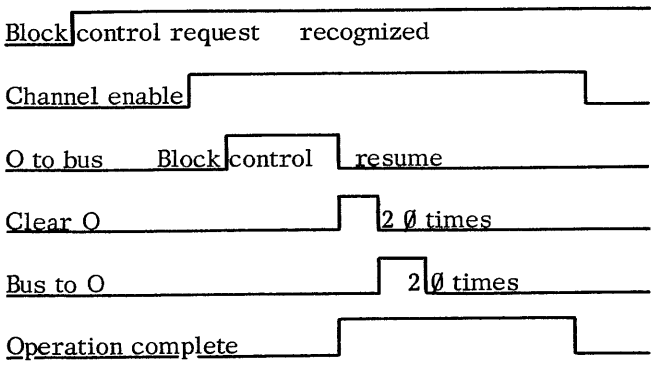


Figure 390. Channel Preset Clear Enable

Up to this point the write control signals have been sent to the external equipment controller and a block control request was sent back to the central processor requesting the information that is to be written.

Block control in the central processor will eventually recognize the request coming from T031 and will then accomplish the following action.

OUTPUT BUFFER CYCLE



Remember that the first cycle of write only set up the write controls in the 3306 and the external equipment controller. Execution of an output (write) instruction started the process. Now block control and the communications channel will communicate with each other and actually output data.

This portion of the cycle is to bring the first byte (word) of data from storage and place it in O register for further transmission. After the information is in the 3306, the 3306 will relay it on to the external equipment controller. The controller will receive the information and transmit it to the piece of peripheral equipment. After this word is sent to output unit (written on the output unit*), the output unit will initiate another block control request which starts the process over again. This will continue until block control returns an operation complete signal during an output buffer cycle.

Figure 391 shows only a write operation. Read will be covered later. These things should be noted in figure 391.

1. The top area (to point A) is the first cycle of a write operation. It is initiated by the execution of the output instruction. After the output instruction has been executed, block control and the 3306 will control the output operation while the central processor continues with the main program.
2. From point B on, the figure shows the operation of the 3306 and block control. The major points to recognize in this area are:
   a. The need for the 3306 to request entrance to block control, which assures output of the word, sends it to the 3306, and checks to see if it is the last word.
   b. The 3306 sends the word on to the controller.
   c. The controller relays the word to the peripheral equipment and generates a reply signal.
      1) If this is not the last word the reply signal will cause the channel to generate another block control request. (Go back to 2a and continue until last word.)
      2) If this is the last word the reply signal cannot cause the channel to request block control because the channel is disconnected from the processor.

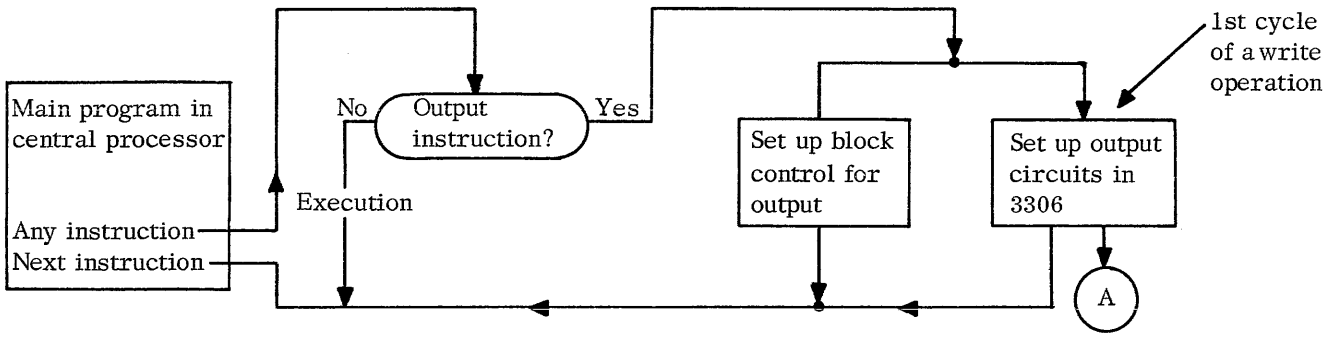
The following area of circuit analysis covers from point B on in figure 391.

Block Control Services a Request

(The block control request was from T031 in first cycle--remember?)

1. Channel enable: R068 = 0, R074 = 0, J605 = 0. The actual circuitry will not be shown here because it is identical to that shown previously. Refer back to the first cycle of a write operation or refer to the complete set of 3306 prints if you do not remember how it works.

*Written is used only to denote output; output mediums could have been card punch, line printers, etc. In most cases the controller stores the output data and returns a reply.



(A) While the main program continues, the 3306 and block control of the processor will accomplish the outputting of data to (B).

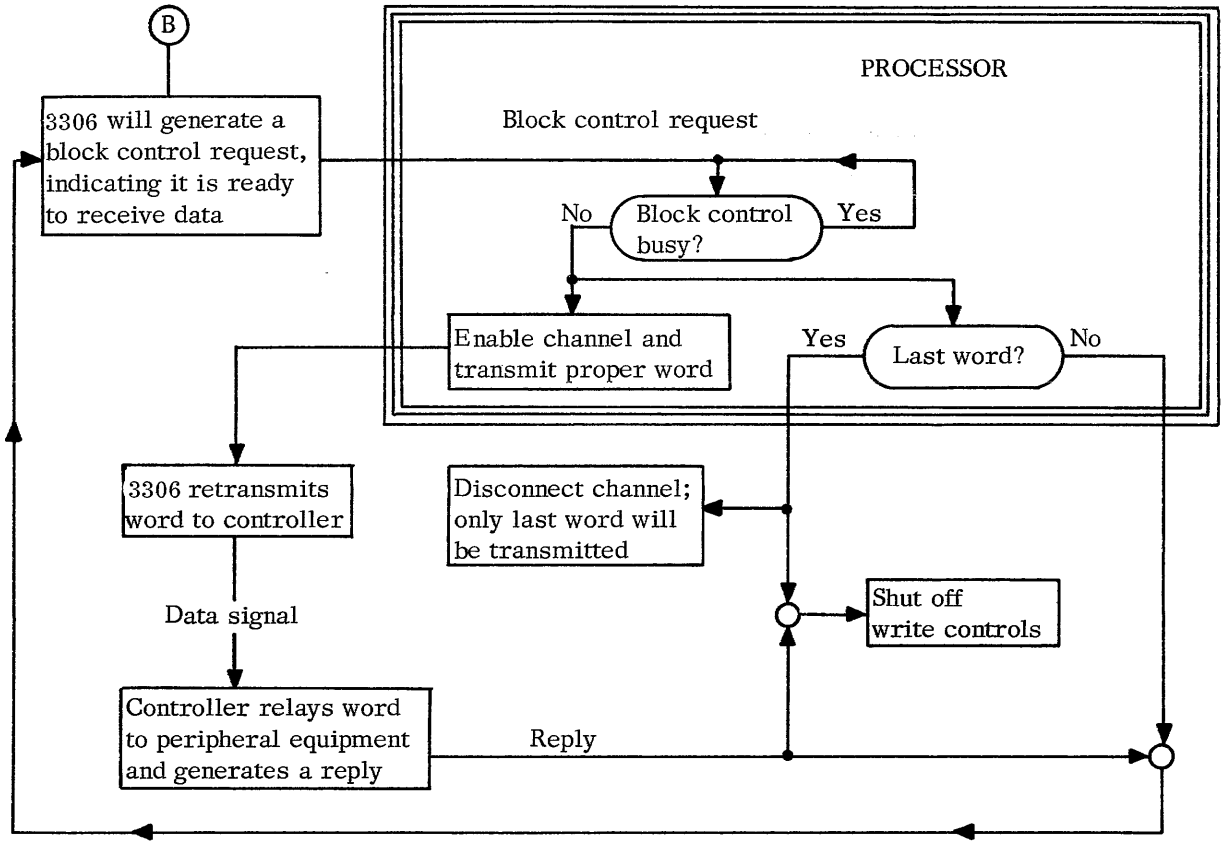


Figure 391. Graphic Representation of a Write Operation

2. Block control resume: R079 = 0, clear K602/603. K614/615 will clear on the trailing edge of the bus to O signal. K602/603 going clear removes the block control request from T031.

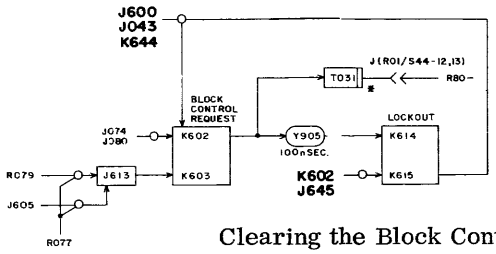


Figure 392. Clearing the Block Control Request

3. Clear O: R073 = 0, J664 and J665 clear O register. Again, the circuitry is not shown because it is identical to other clear O operations. O is being cleared in preparation for output of the word.
4. Bus to O: Transmitting the word to be output to O register in the channel. R072 = 0, J662 and J663 enable bus to O, parity bit to O120/121; J635 = 1, set K606/607 (data signal precondition).

The circuitry for a bus to O is not shown because it has been covered previously. A parity bit is transmitted at this time; however, the circuitry for it will be covered later when all of the parity circuits are covered.

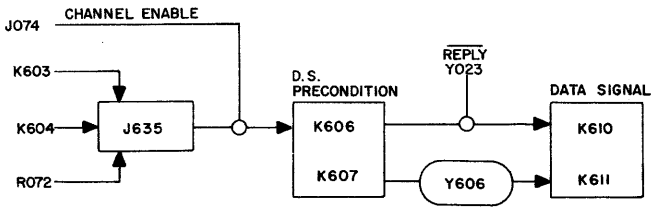


Figure 393. Data Signal FF Enables

The data signal causes J666 and J667 (enable O to external equipment). 100 nsec later T017 = 1, data signal.

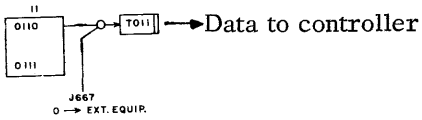


Figure 394. O Register to External Equipment Enable

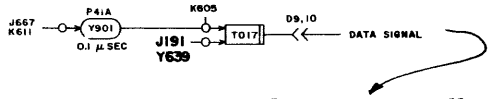


Figure 395. Data Signal Transmitter

Informs controller that the data is on the lines

Refer back to figure 391. Remember that block control sent the word to the 3306 and determined if this was the last word. The 3306 relayed the word to the controller whether it was or was not the last word, but if it was the last word the following would have occurred.

Operation Complete - (m1 = m2) or (r = s).

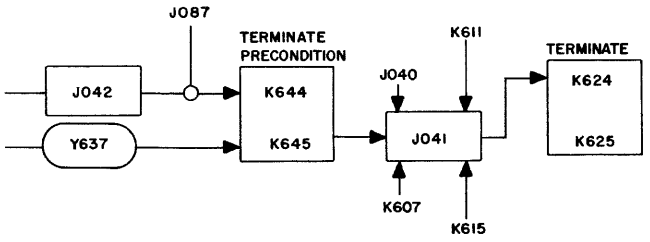


Figure 396. Terminate FF Enables

Channel terminate FF is set. Continue on for a moment to see how channel terminate either allows or disallows the next block control request.

The channel transmitted the word to the controller and the controller must now respond with a reply signal. The return of the reply signal to the 3306 will sample two paths: either the channel has not been terminated or it has been terminated. If it has not been terminated, initiate a block control request; if it has been terminated, clear out the write controls.

Reply From External Equipment - R013 = 1, J606 = 1. 100 nsec later K610/611 is cleared, dropping the data signal to the controller.

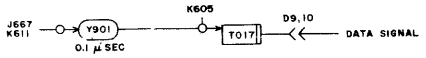


Figure 397. Data Signal Transmitter

Now the two paths are sampled: If the channel were not terminated when reply returned, K602/603 would set and 100 nsec later K614/615 (lockout) would set; T031 would send block control request, asking for more data.

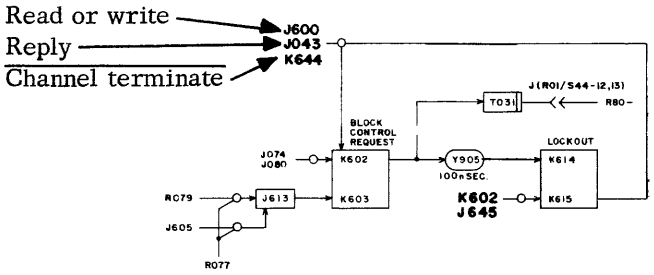


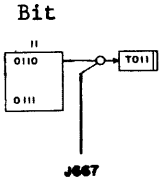
Figure 398. Block Control Request upon Reply

External equipment drops reply (loss of the data signal causes this operation). The actual circuitry that drops the reply is in the controller.

J666 and J667 = 0, dropping O to external equipment enable. At this point return to the area describing block control services a request and redoes the cycles until operation complete signal is generated.

If the channel was terminated when reply returned:

Figure 399. O Register to External Equipment Enable



Refer to the above circuit (see figure 398). It can be seen that the block control request FF cannot be set because K644/645 would have been set by the operation complete signal. All that remains is to clear out write controls.

Clear K604/605 by J615. J666 and J667 = 0, dropping O to external equipment enable.

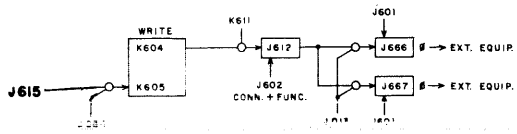
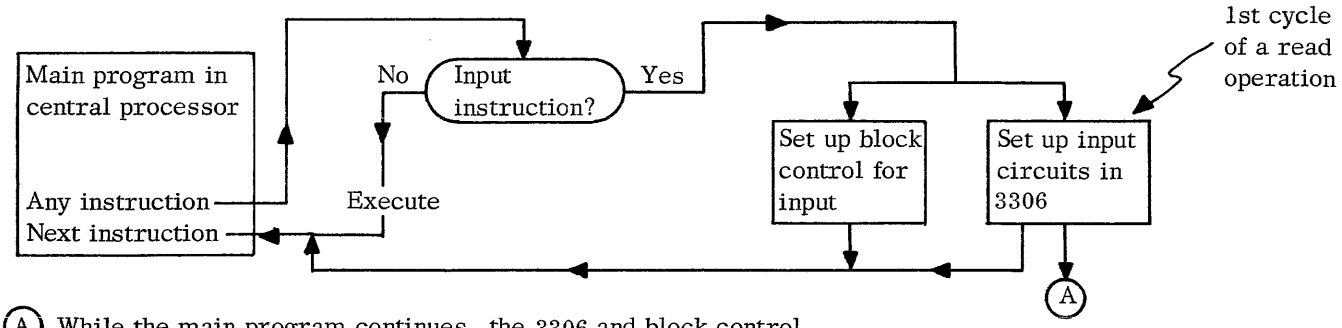


Figure 400. O Register to External Equipment Enable



(A) While the main program continues, the 3306 and block control of the processor will accomplish the inputting of data to (B).

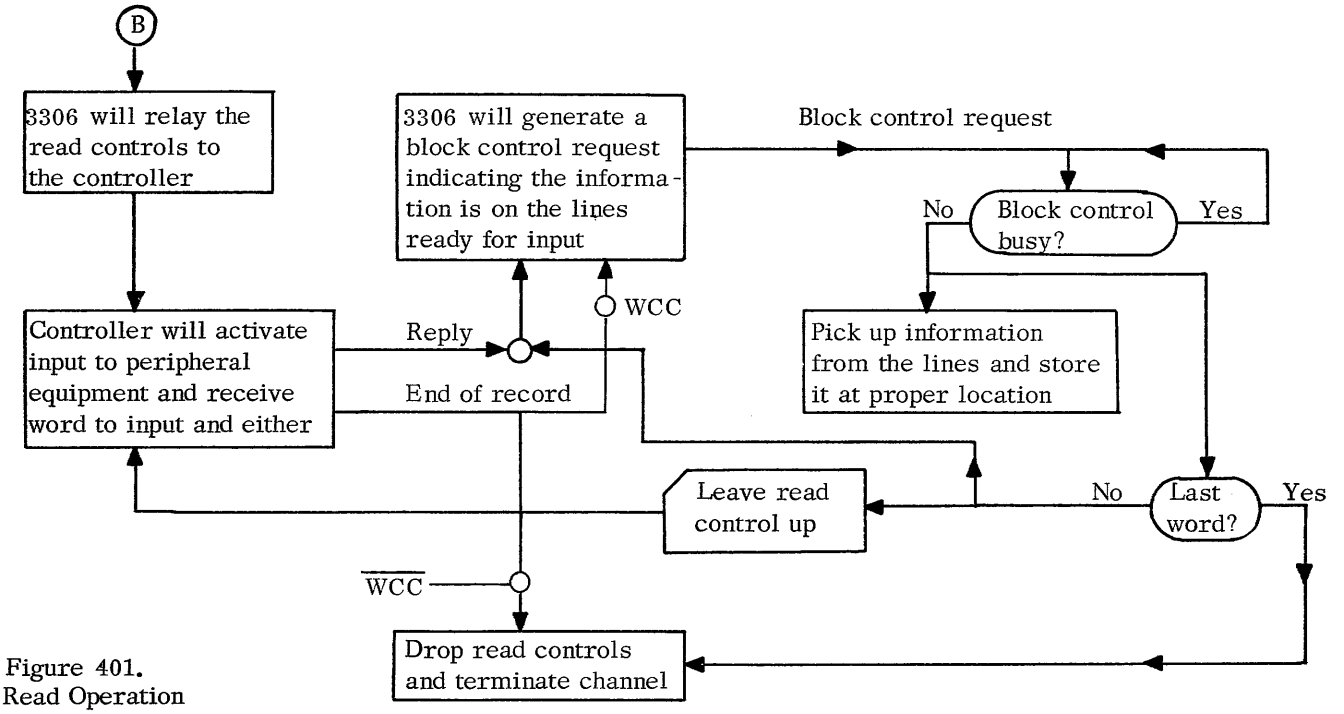


Figure 401.
Read Operation

This constitutes a complete write operation, from the original execution of an output instruction to the final termination of the channel. Refer back to figure 391 for a quick review of the operation.

It is important to recognize that the output operation itself was handled independent of the central processor except for initiation and occasional needs for memory references.

It is also important to recognize that it is the reply signal from the controller that initiates another block request, if one is necessary.

Another important item to note is that all of the circuit analysis has been on the even channel. There is an odd channel, identical to the even channel, with its own controlling circuits that can be operating at the same time.

The even channel control logic diagrams may be found on pages 7-3, 5, 7, 9 and the even channel O register and status logic is on pages 7-19, 21, 23.

Read Operation

A read operation is very similar to a write operation. The biggest differences are in direction of data flow and how the channel terminates.

Data flows are opposite. On output the flow of information is from the computer to the peripheral equipment. On input operations, the flow is from the peripheral equipment to the computer.

Channel termination on a write or output operation comes from the computer when the starting and terminating addresses are the same. Channel termination may come from an end of record signal from the external equipment or from the processor when the addresses are equal.

Previous to any read operation there must be a connect operation. This connect sequence is the same as the connect operation discussed previously so it will not be discussed at this time. Remember, connect establishes a connection between controller and channel.

Assume that a connect operation has been performed and that the connection was made to the odd channel. Assume that the next instruction is an input (read) and that the instruction was made to an odd channel. (The odd channel was selected for this description to show the common circuitry between the odd and even channels.)

Remember, the flow of information on input is:

External equipment→controller→channel→processor

Read through figure 401, keeping in mind:

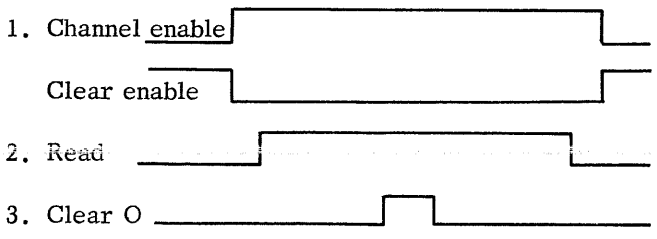
1. Execution of an input instruction sets up the 3306 channel and block control. The processor goes about accomplishing the remainder of the program while the channel and block control handle the input.

2. a. As the words are read by peripheral equipment and relayed to the controller, the controller will signal the channel by a reply signal. The reply signal is in response to the data signal by the channel. If the read controls stay on, the input unit will continue to read information and send it to the controller. Each time another word is ready to be sent to the channel the controller will transmit a reply signal if the channel is requesting data. This operation continues until block control signifies that the word that it received was the last one. This signal causes the channel to terminate.

   b. Another signal could be sent from the controller to the channel. This signal is called the end of record signal and it literally means that there is either no information to input or the input information ran out before block control signaled that the last word had been read in.

A graphic representation of read is shown in figure 401. Read through it carefully. Notice the similarity to the write (output) operation. Note the two ways that the channel can terminate.

Assuming the connect instruction did receive a reply and the next instruction is a read instruction, the following signals will be developed upon the execution of the input instruction. (Review figure 401 to point B (first cycle).



Channel Enable: R168 = 0, R174 = 0, J645 = 0. Drop the clear enable; J184 = 0.

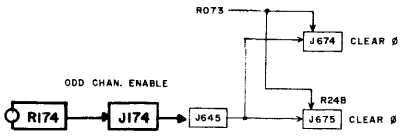
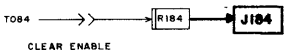


Figure 402.
Channel Enable



Read: J078 = 1.

Clear O: Because the actual clear O circuitry has been shown previously it will not be shown here. Clear O signal now prepares for the arrival of the information from the controller.

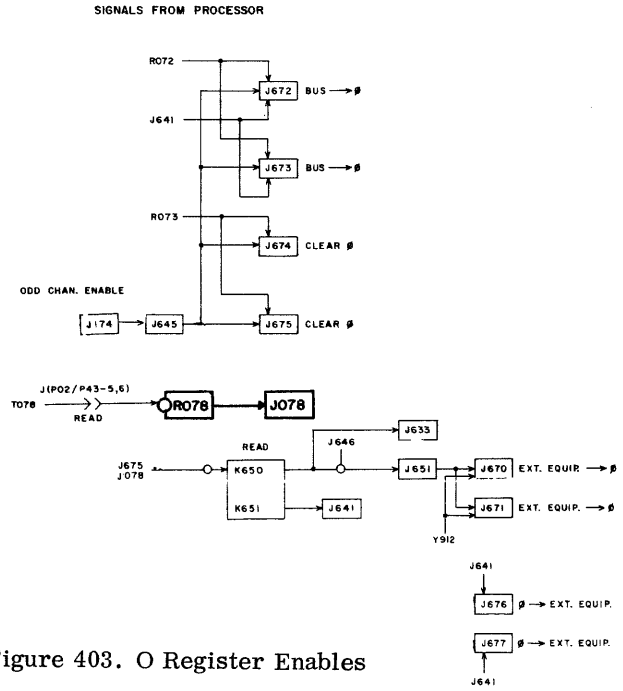


Figure 403. O Register Enables

The combination of read and clear O signals will set K650/651. Note that the O to external equipment signal cannot come up because of J641.

The set output of read FF will not enable external equipment to O until the reply signal (J646) comes back from the controller.

The channel preset FF will clear when read FF sets. It was set by the connect operation.

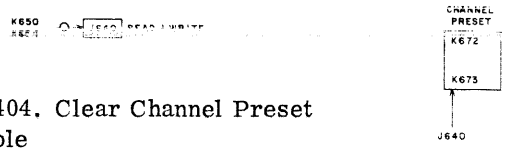


Figure 404. Clear Channel Preset FF Enable

The set output of the read FF will become the read signal to the controller.

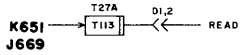


Figure 405. Read Signal to Controller

While the read signal is going to the controller, a data signal will also be sent to the controller. When K656/657 sets for the first time K660/661 and K690/691 will also set. K690/691 will remain set until the read is complete. K690/691 ANDed with K696 and J193 will transmit the data signal for a read operation.

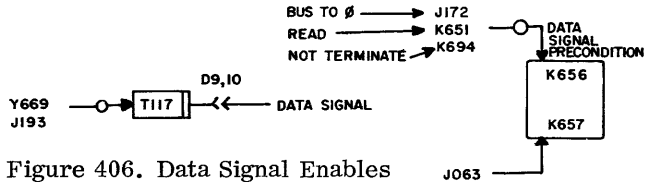


Figure 406. Data Signal Enables

The following is a description of figure 401 starting at point B.

Eventually the peripheral equipment will have read the word and relayed it to the controller. The controller will signal the channel via the reply signal indicating the input information is on the lines and ready to be placed in O register.

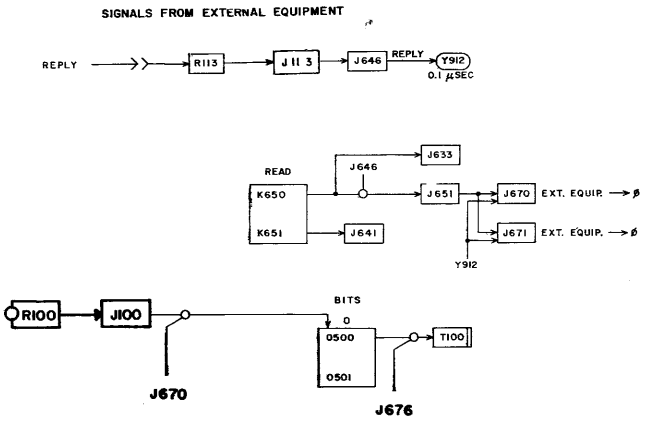


Figure 407. External Equipment to O Register Enables

When the channel receives the reply signal from the controller it will request block control.

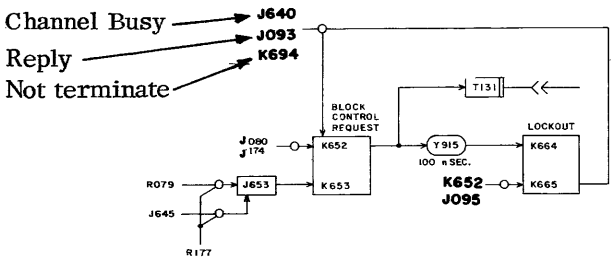
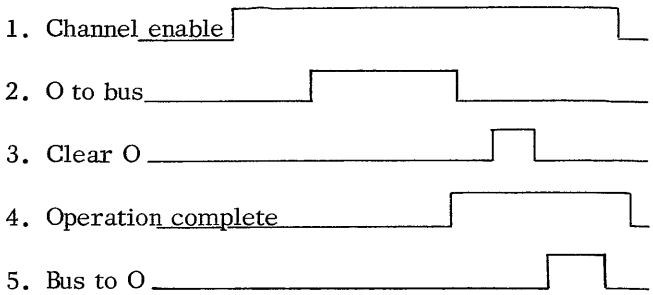


Figure 408. Block Control Request upon Reply

The block control request is accomplished by setting K652/653 and transmitting a block control request via T131.

The reply signal or end of record signal from the controller will clear the data signal FF.

Eventually block control will honor the request. The following shows the timing for the reading of the information from the channel to the processor.



Block control services the request.

Channel Enable: J168 = 1, J174 = 1, J645 = 0. The actual circuitry will not be shown here. Refer to first cycle of a read operation or a complete set of 3306 prints (off channel) for a review (p. 7-11, 13, 15, 17).

O to Bus Signal: R079 = 0, clears K652/653. K652/653 going clear will clear K664/665 on the trailing edge of the bus to O signal.

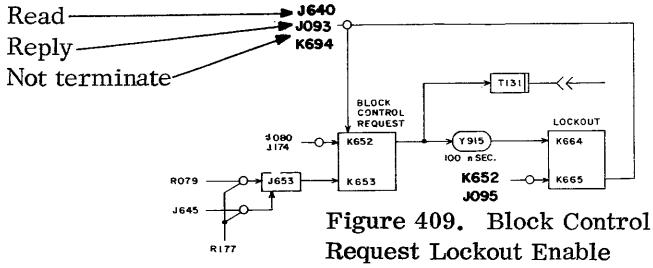


Figure 409. Block Control Request Lockout Enable

O to bus: J683 = 1 (moving the information from the channel to the processor). The O to bus signal will remain up until R079 goes to a 1.

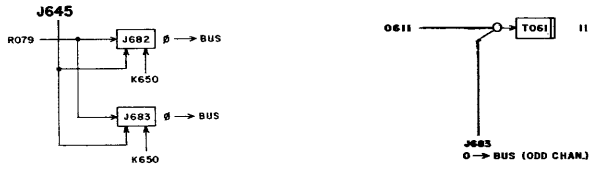


Figure 410. O Register to Bus Enable

A clear O signal will then arrive. O is cleared to prepare for the next word from external equipment.

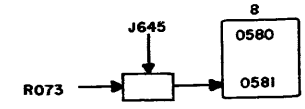


Figure 411. Clear O Register Enable

Refer back to figure 401. Remember that block control received the word from the channel and determined if this was the last word. If it was the last word the following would have occurred: Operation complete (r or M1 = s or m2). If J087 = 1, set K694/695 terminate precondition.

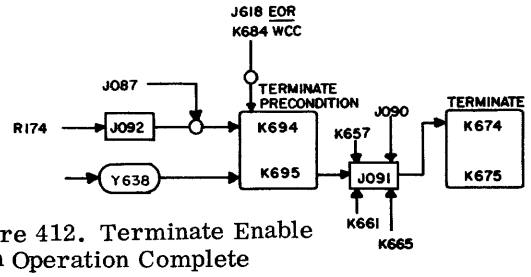


Figure 412. Terminate Enable upon Operation Complete

The set output of K694/695 will transmit a terminate signal to the processor.

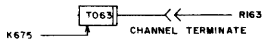


Figure 413. Channel Terminate Signal to CPU

The setting of K694/695 will not allow the block control request FF to set because of the zero on the AND gate.

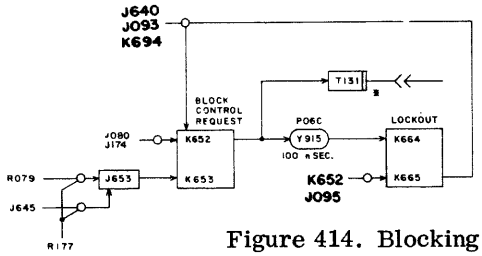


Figure 414. Blocking Block Control Requests upon Termination

Read controls are shut off by the clear enable and no channel terminate. The data signal FF will be prevented from setting by K694. The clear enable is up all the time after the activate cycle.

The clearing of the read FF drops the read signal that is constantly being transmitted to the controller.

The channel could be terminated by another method; this is the end of record if WCC is not selected. The end of record signal comes when the inputting device has not found any information to input.

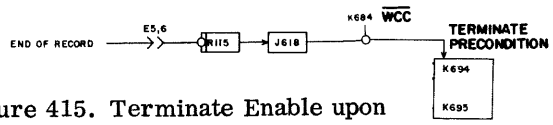


Figure 415. Terminate Enable upon End of Record

When end of record is received (and $\overline{WCC}$) the data signal is up and the channel is waiting for a reply. EOR (J618) is ANDed with K684 ($\overline{WCC}$) to clear K656/657. This causes K660/661 to clear 100 nsec later. With the drop of the data signal, the EOR signal drops, letting J091 set K674/675 (terminate FF). The read FF clears when K674 = 0 and the channel operation is complete.

WORD COUNT CONTROL

The word count control FF (684/685) allows the channel to terminate only with the operation complete signal from block control; not with the EOR signal from the controller. This allows the programmer to read several records from tape or several cards from the card reader with a single input instruction.

When an EOR signal is received (and WCC) the data signal and the read signal are dropped momentarily, but the corresponding FFs are not cleared and the signals are re-enabled. The controller interprets the new read and data signals as a new instruction and initiates another read operation.

This constitutes a complete read operation from the connect to the channel termination. Refer to figure 402 for a review of the read operation, paying particular attention to the block control request and channel termination.

The only difference in the termination of the channels is how channel terminate FF gets set.

This is a complete read operation, from connect to channel termination. Refer to figure 402 for a review of read operation, paying particular attention to the block control request and channel termination.

STATUS OPERATION

As was stated earlier, the central processor communicates with the I/O channel previous to actually using it. One of the first things that could have been done was to check the status of the channel or some equipment on the channel. Checking the status of the channel itself is called internal; checking the status of equipment attached to the channel is called external. (See figure 417.)

Figure 416 shows the connections of the status lines to the processor.

1. Eight external interrupt lines. One from each controller, total of eight per channel.
2. Internal status for the channel (five lines, 0-4, 6, 7).
3. External status for the channel, 12 external status lines parallel to all eight controllers. (Only the connected controller will enable its status to be sent on the lines.)
4. Appropriate sense enable received from processor.
5. Twelve sense lines for sensing internal or external status from odd or even channel.

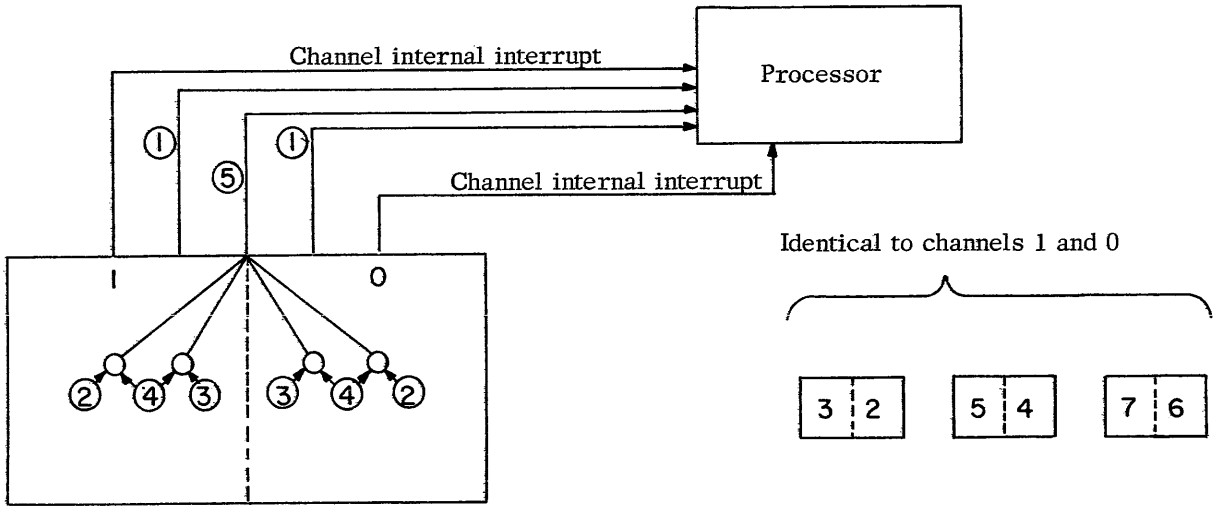
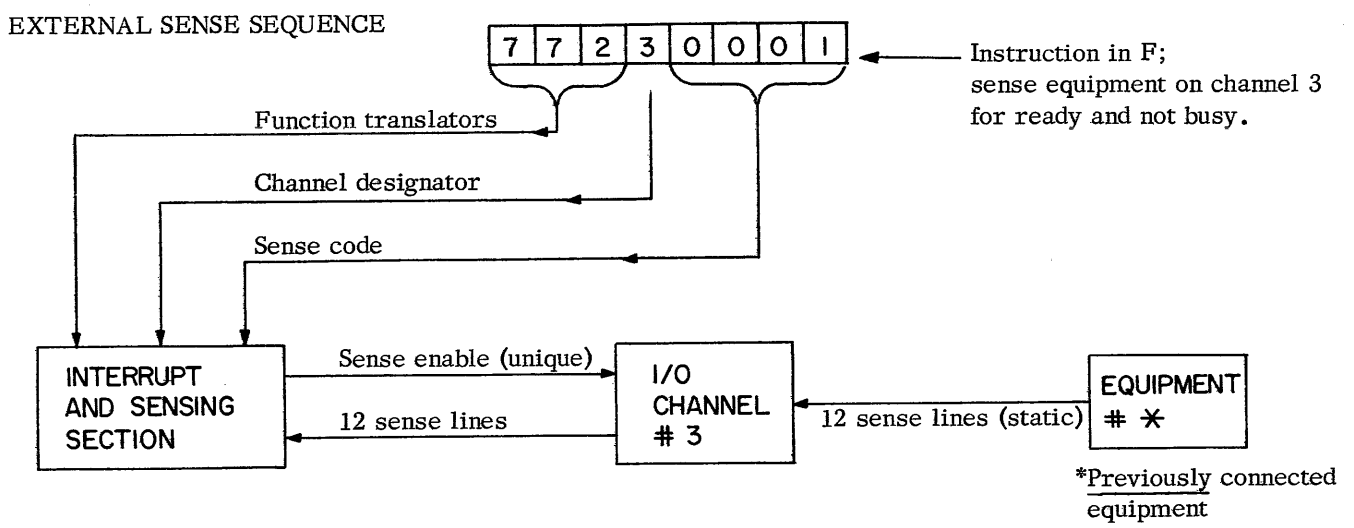


Figure 416. Status and Sense Lines



1. After the instruction is read and translated, a sense enable is sent to the I/O channel via the interrupt and sensing section.
2. The 12 sense lines from the connected equipment which are statically enabled to the I/O channel are sent to the interrupt and sensing section.
3. The 12 sense lines are compared to the sense code.
   a. If any bit compares, the instruction at P + 1 will be executed.
   b. If no bits compare, the instruction at P + 2 will be executed.

Figure 417. Graphic Representation of a Sense Operation

STATUS CHECKING

The communication channel status as well as the external equipment status may be checked by the program in the computation section.

Internal Status

Internal status of the I/O channel may be checked by the program at any time over seven of the 12 sense lines which lead from the I/O channel to the computation section. Strobing is done by signals from the computation section via J678 (J679). The instruction for this operation is 773(ch)XXXX, where ch represents the I/O channel and XXXX is a mask of the conditions being checked. Bits 0-7 in the mask correspond to sense lines 0-4, 6, 7. Bits 5, 8-11 of the mask represent arithmetic faults within the computation section. When XXXX is 0, the internal status information is copied into the lower 12 bits of the A register.

External Status

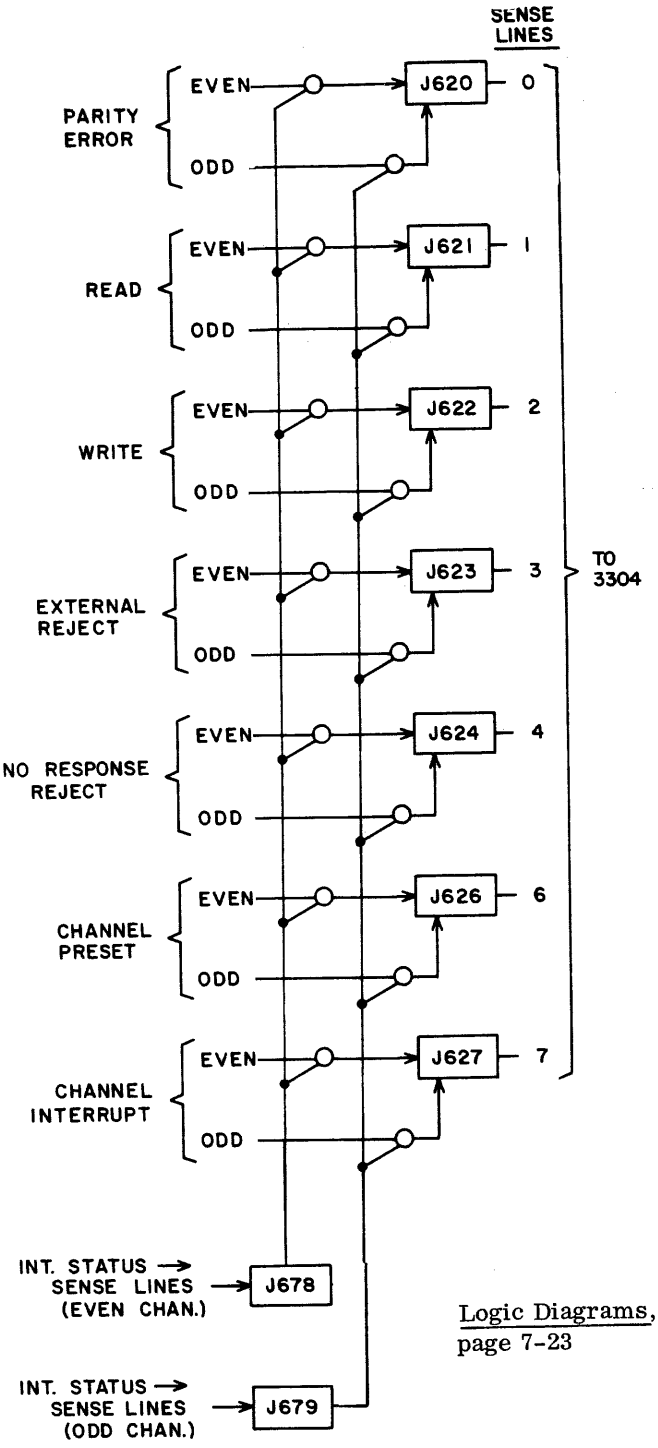
Status of the external equipment may be checked by the program at any time over the 12 sense lines which lead from the I/O channel to the computation section. Signals on these lines have meanings which generally are unique for each type of external equipment. The instruction for this operation is 772 (ch) XXXX, where ch represents the I/O channel and XXXX is a mask used for looking at individual lines or groups of lines. When XXXX is 0, the external status information is copied into the lower 12 bits of A register. During the time that status is being strobed by the computation section via J684-5 (J686-7), a 1 is sent to the external equipment over transmitter (T022/T122).

The status circuits are extremely important to the programmer. It is through these lines that the condition of the external equipment can be determined. Other checks can be made from the sense lines which reflect the condition of control within the channel. These conditions are set by the appropriate signals. The circuits for each and the sense lines affected are shown in figure 420.

The parity error FFs can be set by two different methods.

1. Output or Write. The controller receives a parity error during the output of the word from the channel. This is the external parity error signal and comes from the controller.
2. Input or Read. The channel transmitted the (O) register to the data bus in the processor and the data bus generates a parity bit. This is transmitted back to the channel and compared to the parity bit in the O register; if they disagree the parity error FF will set.

Parity error on connect and function codes works the same as output or write.

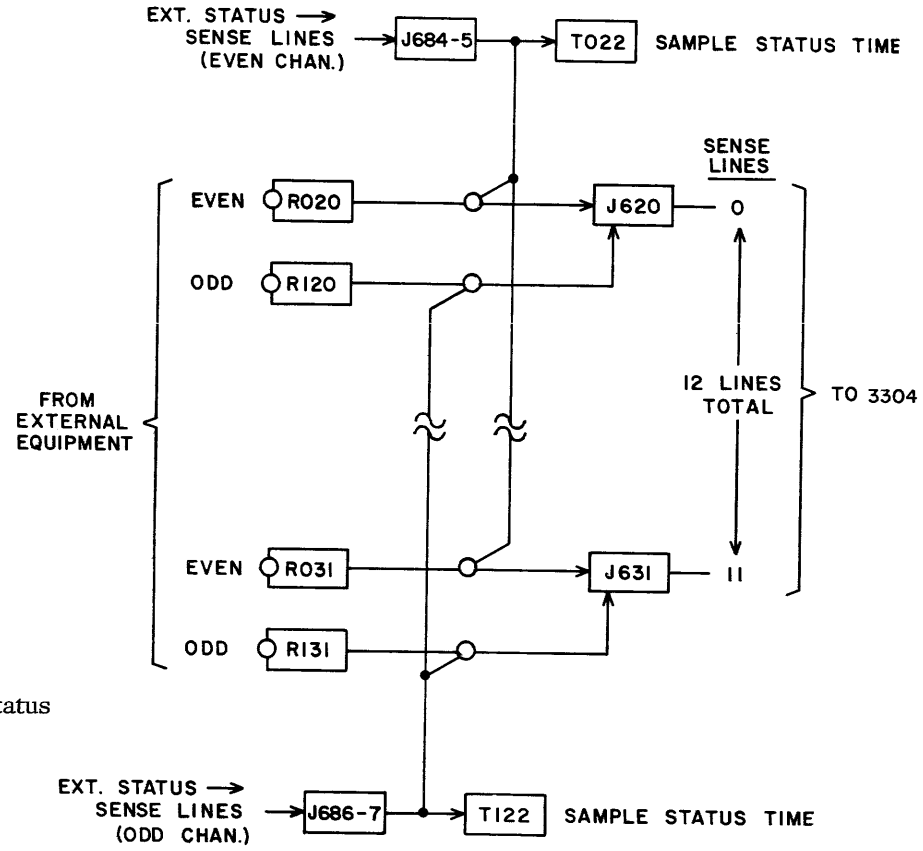


Logic Diagrams, page 7-23

Figure 418. Internal Status

PARITY CHECKING

Parity is checked by one method for connect, function, and write operations and by a second method for read operations.



Logic Diagrams, page 7-23

Figure 419. External Status

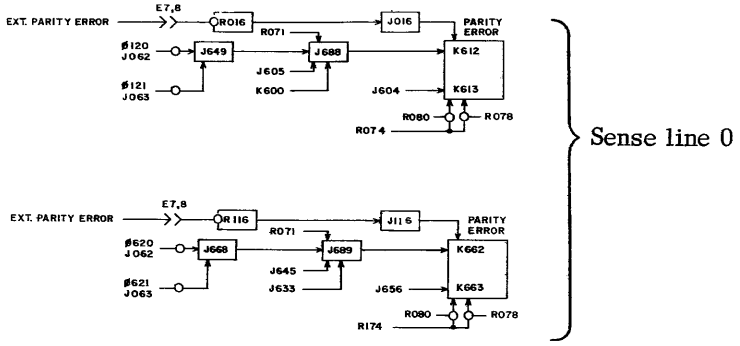


Figure 420. Parity Error Status

Connect, Function, and Write

During the connect, function, and write operations a parity bit is generated in the data bus circuit of the computation section and is sent to the external equipment via O120/121 FF in O register. A parity bit is generated in the external equipment and compared with the parity bit from the computer. If an error exists, an external parity error signal is sent back to R016 in the 3306. This sets the parity error FF (K612/613) and provides a 1 on sense line 0. K612/613 is cleared every time that an attempt is made to perform a connect, function, read, or write operation with this channel. (Not all equipments send parity signals on a connect operation. Refer to manuals covering specific synchronizer for details.) It may also be channel cleared by the program or master cleared by the operator.

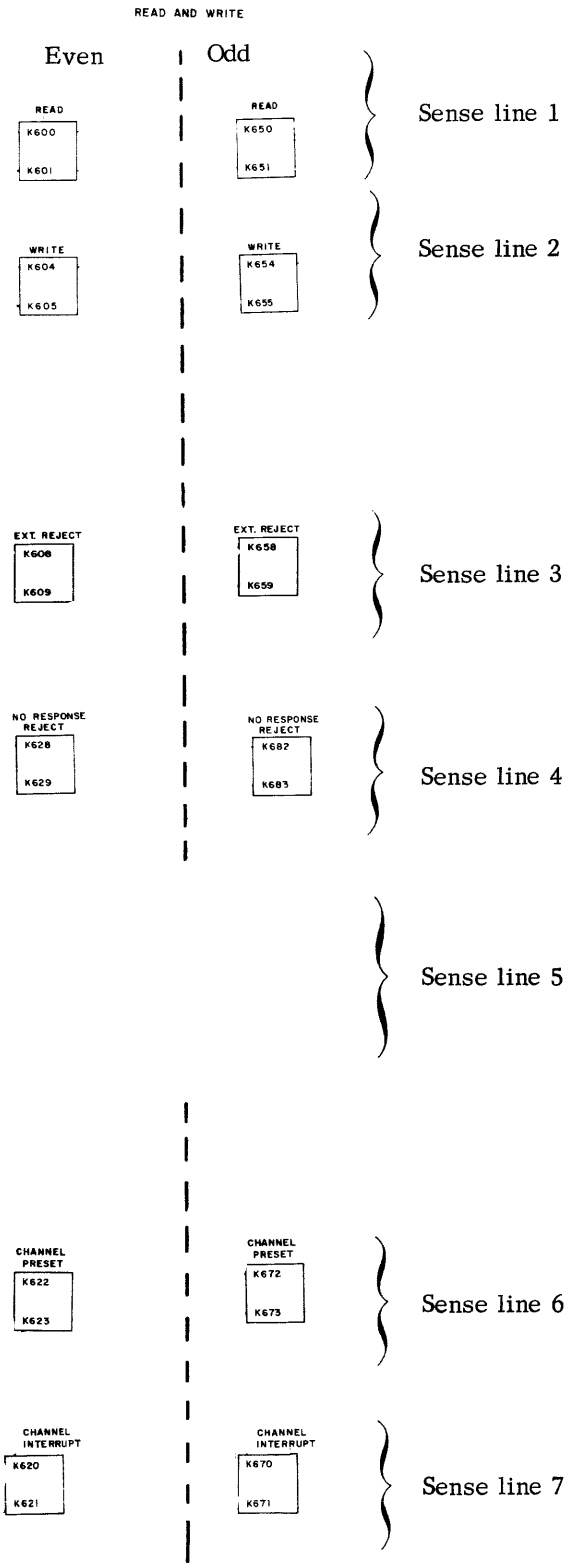


Figure 421. Internal Status

Read

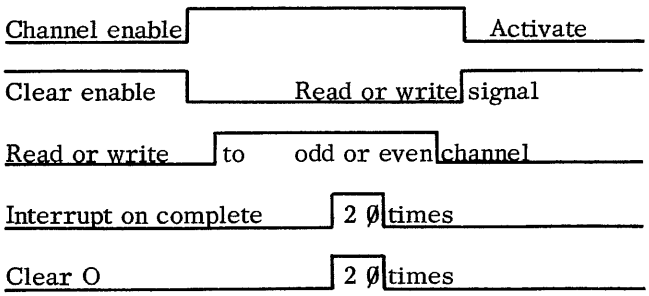
During a read operation, a parity bit is generated by the external equipment and sent to the 3306 along with the 12 bits of data. The parity bit is held in O120/121 FF of O register while the data is forwarded to the computation section. Parity is generated in the data bus circuit of the computation section and is sent back to the 3306. This new parity signal is compared with the parity signal, being held to O120/121, which was generated by the external equipment. If an error exists, setting of parity error FF (K612/613) is enabled by a bus to O signal from the computation section. This provides a 1 on sense line 0. K612/613 is cleared every time that an attempt is made to perform a connect, function, read, or write operation with this channel. It may also be channel cleared by the program or master cleared by the operator.

The previous description shows that the condition of the channels, either odd or even, internal or external, can be determined through the use of a sense instruction.

INTERRUPTS

Consider for a moment the activate input or output operation. If the activate instruction carries the appropriate other bits it is possible for the channel to send an interrupt signal to the processor when the operation is complete.

The following signals are identical to the ones presented previous to a read or write operation, with the addition of an interrupt on completion signal.



J074 and J174 are channel enables.

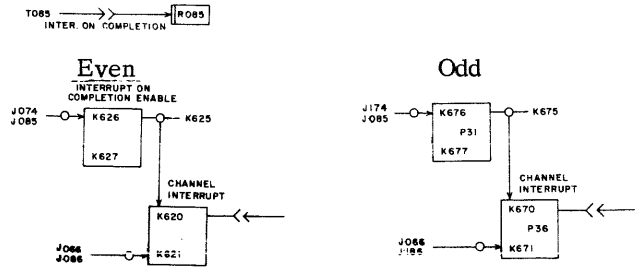


Figure 422. Channel Interrupt

Other interrupt signals can be developed. The circuitry for each is shown below.

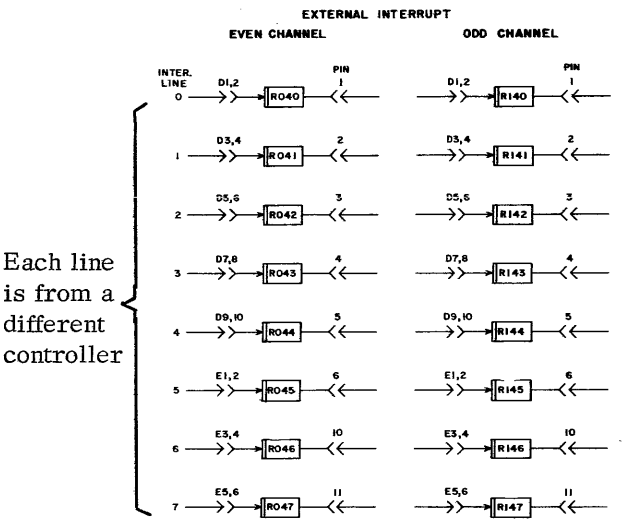


Figure 423. Interrupt Signal Circuitry

Two types of interrupts are associated with an I/O operation. These are the external I/O interrupts and the I/O channel interrupts.

External I/O Interrupts

The interrupt circuits in external equipment are enabled by the select function instruction 771(ch)XXXX. If an enabled interrupt becomes active, a signal is passed directly to interrupt control in the computation section via one of the receiver cards, R040-7, in the communication channel. There are eight interrupt lines per I/O channel, numbered 0-7, each corresponding to the equipment number switch setting on the peripheral equipment controllers. This interrupt signal remains active until the computation section clears it with a function code, or until the operator does an external master clear.

I/O Channel Interrupts

Each channel contains a channel interrupt FF whose set output is monitored by interrupt control in the computation section. This FF is set by the channel terminate FF provided that the select interrupt on completion FF has been previously set by the computation section. The select interrupt on completion FF is set when bit 17 of the second word of an I/O (73-76) instruction contains a 1. The channel interrupt FF is cleared by either:

1. A programmed channel interrupt clear
2. An operator's external master clear

The select interrupt on completion FF will be cleared if the channel is not busy and either:

1. An IOCL or CLCA instruction is executed
2. Channel terminate is set (after busy drops)
3. An operator's master clear

CLEARING CIRCUITS

Figure 424 shows the clearing circuits in a 3300 communication channel.

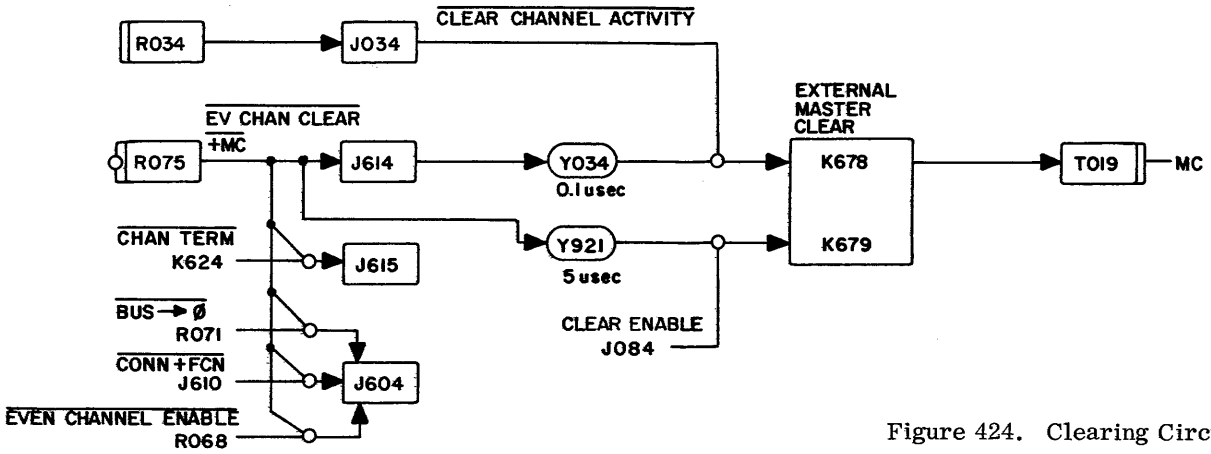


Figure 424. Clearing Circuits

NOTE: The clear enable is present at all times that the computer is on except for connect, function, and activate read or write operations.

There are other circuits within the 3306 that allow assembly/disassembly, suppression of word mark, etc. The theory of these circuits is left to the reader.

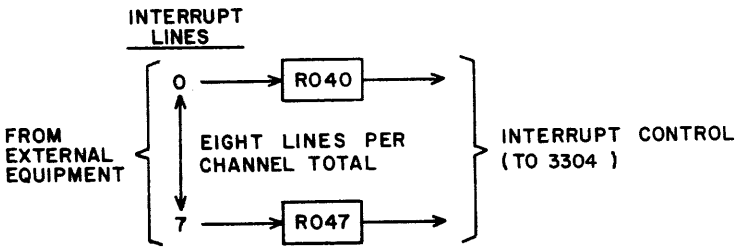


Figure 425. External I/O Interrupts

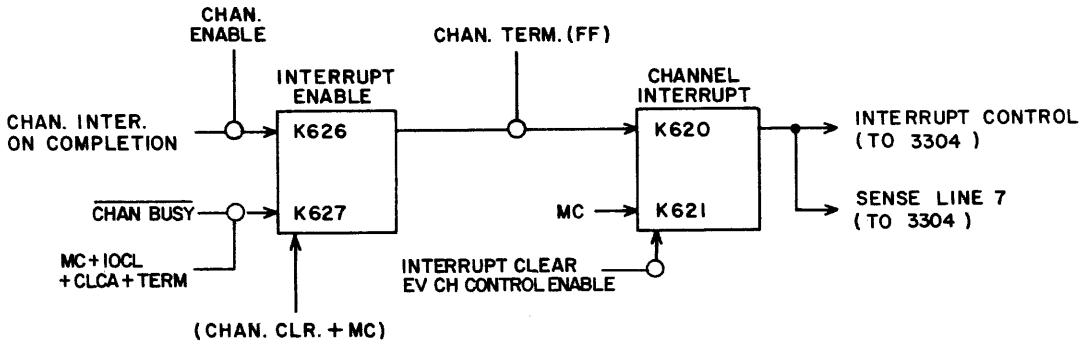


Figure 426. I/O Channel Interrupts

INPUT/OUTPUT CHANNELS Worksheet #1

Write the instruction which will accomplish each of the following operations, then answer the questions regarding each instruction.

Reference: Computer reference manual, instruction descriptions associated with I/O and interrupt, and 3306 theory text.

1. Connect tape handler 3 and controller 5 to channel 7.
   a. What signal would be transmitted to the processor if channel 7 were busy?

   b. What signal would be transmitted to the processor if channel 7 were not present in the system?

   c. What would happen if there weren't a synchronizer designated as 5 on channel 7?

   d. What would happen if there weren't a tape handler designated as 3 on synchronizer 5 on channel 7?

2. Sense for channel 1 busy. What sense line would carry the signal to the processor?

3. Provided that tape handler 3 on synchronizer 5 were connected to channel 7, what instruction would sense for that tape handler at load point? What type of signal would be developed to indicate this?

4. Sense for illegal write. What sense line carries the signal indicating an illegal write has occurred?

5. Clear channel 5 and its peripheral equipment.

INPUT/OUTPUT CHANNELS Worksheet #2

Any information you write on magnetic tape and then read in is wrong. You know the read circuitry itself is okay because you can correctly read tapes that were written several days ago.

The fault appears to be in the write operation, somewhere between the processor and the 3306.

List the steps you would perform to localize the trouble.

CLUES:

a. Remember, read operation works, but write operation fails. Perhaps a comparison of the flow paths for each would help eliminate some circuitry that is common to each.

b. The basic operation of the channel must be good because you can connect to the proper unit for read operations.

The 3307 is a 24-bit bidirectional I/O channel which must be an even-numbered channel. The purpose of the 3307 is two fold:

1. It facilitates convenient interface to a 24-bit I/O device.
2. It halves the number of storage references required when operating with a high speed 12-bit I/O device.

The first purpose of the 3307 is self explanatory since the 3307 has a 24-bit O register and 24 data lines. To understand the second purpose of the 3307, remember that with the 3306 a buffer cycle, which referenced storage, was necessary for every data transfer. However, the 3307 has an assembly/disassembly feature for word addressed I/O enabling assembly of 12 bits to 24 or disassembly of 24 bits to 12, allowing the 3307 to initiate a buffer cycle for every other transfer (74 + 76)(N = 0).

The 3307 uses odd parity exclusively. There is a parity bit for the lower 12 bits of the 3307 and a parity bit for the upper 12 bits.

In an I/O module that contains a 3307 and a 3306, the clear O signal is used as a control signal only and will not cause the O register to clear.

Refer to the 3300 Logic Diagrams, pages 7-27 to 7-39 for the 3307 control circuits and pages 7-49 to 7-55 for the 3307 O register.

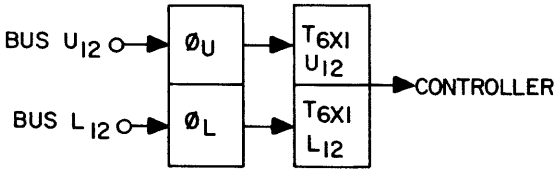


Figure 427. 3307 Data Flow on Output (Without Disassembly)

3307 TIMING FOR WRITE NO ASSY/DSSY:
INSTRUCTIONS (75)(H = 0 + 1) OR (76)(N = 1)

Activate I/O Operation

1. Even Channel Enable (J310 = 1).
2. Drop Clear Enable (J308 = 0).
3. Clear O register signal (R300 = 0).
4. Write signal (J307 = 1) sets; Write FF K302/303 which sends Write signal to external equipment (T306 = 1); Block Control Request Enable FF K366/367; and Block Control Request FF K332/333. Setting of K332/333 clears the Block Control Request Lockout FF K362/363 and sends a Block Control Request signal (T301 = 1).
5. If (75)(H = 0), suppress assembly/disassembly (J314 = 1) which sets the Suppress Assembly/Disassembly Synchronizer FF K308/309 and sends a Suppress Assembly/Disassembly signal (T313 = 1).
6. Wait for Block Control to service request.

Buffer Cycle

1. Even Channel Enable (J310 = 1).
2. Block Control Resume (J352 = 1), causing J456 to generate a 1 for 100 ns. This clears the Block Control Request FF K332/333 and the Block Control Request Enable FF K366/367.
3. If operation complete, R313 = 0 which sets the Terminate Precondition FF K334/335.
4. Clear O Register (R300 = 0) which sets Block Control Request Enable FF K366/367 (if operation complete).
5. Bus to O Register (J301 = 1). This gates Data bus to the O Register; static 0 to external equipment transmitters; and sets O Loaded FF K340/341.
6. Trailing edge of Bus to O signal sets Data Signal Precondition FF K342/343. After 100 ns this sets:
   a. the Data Signal FF K344/345, which sets the Block Control Request Lockout FF K362/363.
   b. the Data Signal II FF K368/369, which sends a Data signal (T309 = 1).
7. Wait for Reply from external equipment.
8. External Reply (R400 = 0, R450=1, J400 = 1, J451=1, J452 = 1). This clears the O Loaded FF K340/341, clears the Data Signal Precondition FF K342/343, and sets the Block Control Request FF K332/333 (if operation complete). After 100 ns Y400 = 1 which clears the Data Signal FF K344/345.

If Operation Complete

Repeat Activate I/O Operation, step 6.

If Operation Complete

1. Reply drops (R400 = 1). After 100 ns J450 = 0 which sets Terminate FF K336/337, clears K368/369, and clears Write FF K302/303.
2. 100 ns after K302/303 clears, the Terminate Precondition FF K334/335 and the Terminate FF K336/337 clear.

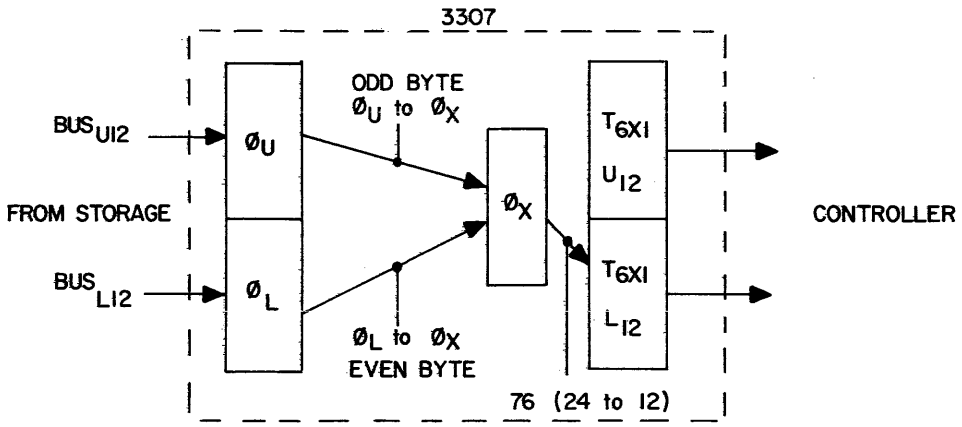


Figure 428. 3307 Data Flow on Output (With Disassembly)

GENERAL FLOW FOR A 3-WORD OUTPUT ON A 76 (24 to 12) DISASSEMBLY (FORWARD)

Step 1: Activate 76 (24 to 12) Forward
- a. Send Write signal to controller and request Block Control.
- b. Assembly/Disassembly counter = set, set.
- c. Set 12-24 A/D.

Step 2: Serviced by Block Control
- a. Bus to O_U/O_L. O_U to O_X to $T6X1_{L12}$.
- b. Data signal to controller.
- c. Assembly/Disassembly counter = clear, set.

Step 3: First Reply from Controller
- a. O_L to O_X to $T6X1_{L12}$ (Assembly/Disassembly = clear, clear).
- b. Data signal to controller, request Block Control.
- c. Assembly/Disassembly counter = set, clear.

Step 4: Second Reply from Controller
- a. Assembly/Disassembly counter = set, set.
- b. Wait for new word from Block Control.

Step 5: Serviced by Block Control
- a. Bus to O_U/O_L. O_U to O_X to T6X1.
- b. Data signal to controller.
- c. Assembly/Disassembly counter = clear, set.

Step 6: Third Reply from Controller
- a. O_L to O_X to $T6X1_{L12}$ (Assembly/Disassembly counter = clear, clear).
- b. Data signal to controller, request Block Control.
- c. Assembly/Disassembly counter = set, clear.

Step 7: Serviced by Block Control
- a. Bus to O_U/O_L.
- b. Send operation complete.

Step 8: Fourth Reply from Controller
- a. O_U to O_X to $T6X1_{L12}$. (Assembly/Disassembly counter = set, set).
- b. Data signal to controller.
- c. Assembly/Disassembly counter = clear, set.

Step 9: Fifth Reply from Controller
- a. O_L to O_X to $T6X1_{L12}$. (Assembly/Disassembly counter = clear, clear).
- b. Data signal to controller. (Don't request Block Control if operation complete.)
- c. Assembly/Disassembly counter = set, clear.

Step 10: Sixth Reply from Controller
- a. Assembly/Disassembly counter = set, set.
- b. Terminate operation, clear Assembly/Disassembly counter.

3307 TIMING FOR WRITE DISASSEMBLY: INSTRUCTION (76)(N = 0) FORWARD

Activate I/O Operation
1. Even Channel Enable (J310 = 1).
2. Drop Clear Enable (J308 = 0).
3. a. Clear O signal R300 = 0 which sets Block Control Request Enable FF K366/367. This FF clears the Block Control Lockout FF K362/363 and sends a Block Control Request (T301 = 1).
   - b. Write signal (J307 = 1) which sets:
     1) Write FF K302/303, Sending Write signal (T306 = 1).
     2) Block Control Request FF K332/333.
   - c. Assembly/Disassembly signal (J317 = 1) which sets Assembly/Disassembly FF K304/305 and A000/001 and B000/001.
4. Wait for Block Control to service request.

Buffer Cycle
1. Even Channel Enable J310 = 1.
2. Block Control Resume J352 = 1. This forces J456 to output a 100 ns logical 1 which clears:
   - a. Block Control Request FF K332/333.
   - b. Block Control Request Enable FF K366/367.
3. Clear O signal (R300 = 0) and set Block Control Request Enable FF K366/367.

4. Bus to O signal (J301 = 1 and R301 = 0). R301 gates Bus to the O register. J301 sets the O Loaded FF K340/341. After 150 ns, J462 outputs a 100 ns logical 1 which:
   a. Gates O_U to O_X (O_X to T static). This sets Suppress Word Mark FF K320/321 and drops Word Mark (T310 = 0).
   b. Sets the Data Signal Precondition FF K342/343. After 100 ns, this sets the Data Signal FF K344/345, K368/369, J462 outputs logical 1 for 100 ns which sends the Data signal (T309 = 1), clears A000/001, and sets the Block Control Lockout FF K362/363.
5. Wait for first Reply from external equipment.

First Reply

1. R400 = 0. After 100 ns Y400 = 1 which clears Data Signal FF K344/345.
2. J451 = 1 for 100 ns. This:
   a. Clears the O Loaded FF K340/341.
   b. Clears the Data Signal Precondition FF K342/343.
   c. Clears B000/001.
   d. Sets the Block Control Request FF K332/333, which sends a Block Control Request (T301 = 1) and clears the Block Control Request Lockout FF K362/363.
3. J400 = 1.
4. J452 = 1 for 150 ns, after which time
   a. J454 = 1 for 100 ns. This enables O_L to O_X (O to T static), clears the Suppress Word Mark FF K320/321 and sends Word Mark (T310 = 1).
   b. Sets the Data Signal Precondition FF K342/343 (K352). After 100 ns, this sets the Data Signal FF K344/345 (if Reply has dropped). J465 = 1 for 100 ns, which sets A000/001.
5. a. Wait for second Reply from I/O equipment.
   b. Wait for Block Control to service channel. (In this example, assume that the Reply comes up first.)

Second Reply

1. a. R400 = 0. After 100 ns Y400 = 1 which clears the Data Signal FF K344/345.
   b. J451 = 1 for 100 ns. This sets B000/001 and clears the Data Signal Precondition FF K342/343. (The O Loaded FF K340/341 is still clear (J462 = 0); thus, O_U will not gate to O_X, the Suppress Word Mark FF K340/341 will not set, the Data Signal Precondition FF K342/343 will not set, nad the Data Signal FF K344/345 will not set.)
   c. J400 = 1.
   d. J452 = 1 for 150 ns.
2. Continue the wait for Block Control to service channel.

Buffer Cycle

1. Even Channel Enable J310 = 1.
2. Block Control Resume (J352 = 1). This causes J456 to output a logical 1 for 100 ns which clears the Block Control Request FF K332/333 and Block Control Request Enable FF K366/367.
3. Clear O signal (R300 = 0) and set Block Control Request Enable FF K366/367.
4. Bus to O signal (J301 = 1, R301 = 0). R301 output gates Bus to O Register. J301 output sets the O Loaded FF K340/341 and after 150 ns, J462 = 1 for 100 ns. The output of J462:
   a. Gates O_U to O_X (O_X to T static) which sets Suppress Word Mark FF K320/321 and drops Word Mark (T310 = 0).
   b. Sets the Data Signal Precondition FF K342/343. After 100 ns, this sets the Data Signal FF K344/345 which causes J465 to output a logical 1 for 100 ns and sends the Data signal (T309 = 1). This, in turn, clears A000/001 and sets the Block Control Request Lockout FF K362/363.
5. Wait for third reply.

Third Reply

1. a. R400 = 0. After 100 ns this causes Y400 = 1 which clears the Data Signal FF K344/345.
   b. J451 = 1 for 100 ns. This:
      1) clears the O Loaded FF K340/341.
      2) clears the Data Signal Precondition FF K342/343.
      3) clears B000/001.
      4) sets the Block Control, Request FF K332/333 which sends a Block Control Request (T301 = 1), and clears the Block Control Request Lockout FF K362/363.
   c. J400 = 1.
   d. J452 = 1 for 150 ns after which time J454 = 1 for 100 ns. This:
      1) enables O_L to O_X (static O_X to T). This clears the Suppress Word Mark FF K320/321 and sends a Word Mark (T310 = 1).
      2) sets the Data Signal Precondition FF K342/343 (K352). After 100 ns, this sets the Data Signal FF K344/345 (if Reply has dropped). J465 = 1 for 100 ns which sets A000/001.
2. a. Wait for the fourth Reply.
   b. Wait for Block Control to service channel. (Assume I/O equipment slow to reply and that Block Control services request and send another 24-bit word to the channel.)

Buffer Cycle

1. Even Channel Enable (J310 = 1).
2. Block Control Resume (J352 = 1), causing J456 to output a logical 1 for 100 ns. This clears the Block Control Request FF K332/333 which clears the Block Control Enable FF K366/367.

3. Operation complete (R313 = 0, J363 = 1, and J313 = 1). J363 sets Terminate Precondition FF K334/335.
4. Clear O Signal (R300 = 0). (Do not set Block Control Request Enable FF K366/367 since operation complete is up.)
5. Bus to O signal (J301 = 1) and (R301 = 0). R301 gates the bus to the O Register. J301 sets O Loaded FF K340/341. Since O_X is holding 12 bits, J462 will not come up since B000/001 is still clear. Thus, no O_U to O_X transfer will take place until the next (fourth) Reply is received from the channel.
6. Continue wait for the fourth Reply.

Fourth Reply

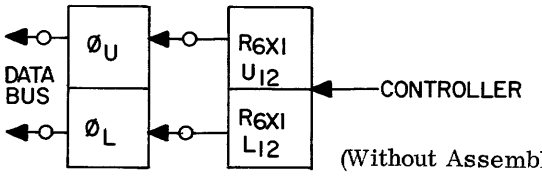
1. a. R400 = 0. After 100 ns Y400 = 1 which clears the Data Signal FF K344/345.
   b. J451 = 1 for 100 ns. This clears the Data Signal Precondition FF K342/343 and sets B000/001. After 150 ns, J462 = 1 for 100 ns since the O Loaded FF is still set. The output of J462:
      1) gates O_U to O_X (O_X to T static). This sets the Suppress Word Mark FF K320/321 and drops Word Mark (T310 = 0).
      2) sets the Data Signal Precondition FF K342/343. After 100 ns, the Data Signal FF K344/345 sets. This, in turn, causes J465 to output a logical 1 for 100 ns and sends a Data signal (T309 = 1). The output of J465 clears A000/001 and sets the Block Control Request Lockout FF K362/363.
   c. J400 = 1.
   d. J452 = 1 for 150 ns.
2. Reply drops (R400 = 1). After 100 ns Y450 = 1 and J450 = 0. (Cannot set the Terminate FF K336/337 at this time since the Data Signal FF is set).
3. Wait for the fifth Reply.

Fifth Reply

1. a. R400 = 0. After 100 ns Y400 = 1 which clears the Data Signal FF K344/345.
   b. J451 = 1 for 100 ns. This:
      1) clears the O Loaded FF K340/341.
      2) clears the Data Signal Precondition FF K342/343.
      3) clears B000/001.
      4) cannot set the Block Control Request FF K332/333 since the Terminate Precondition FF is set.
   c. J400 = 1.
   d. J452 = 1 for 150 ns. After 150 ns, J454 = 1 for 100 ns.
2. Reply drops (R400 = 1).
   a. Enable O_L to O_X (static O_X to T) which clears the Suppress Word Mark FF and sends Word Mark (T310 = 1). The Data Signal Precondition FF K342/343 sets, after 100 ns, this causes:
      1) J465 = 1 for 100 ns, which sets A000/001.
      2) Data signal to be transmitted (T309 = 1).
   b. After 100 ns Y450 = 1, which causes J450 = 0. (Cannot set the Terminate FF K336/337 yet since the Data Signal FF is set.)
3. Wait for the sixth (last) Reply.

Sixth Reply

1. a. R400 = 0. After 100 ns Y400 = 1 which clears the Data Signal FF K344/345.
   b. J451 = 1 for 100 ns. This clears the Data Signal Precondition FF K342/343 and sets B000/001. After 150 ns J462 cannot output a logical 1 because the O Loaded FF is set; therefore, the Data Signal Precondition FF will not set.
   c. J400 = 1.
   d. J452 = 1 for 150 ns.
2. Reply drops (R400 = 1). After 100 ns Y450 = 1 and J450 = 0. This sets the Terminate FF K336/337 since the Data Signal FF is clear. J450 output causes:
   a. J866 = 1, which clears A000/001, B000/001 and K368/369.
   b. J863 = 1 until the Terminate FF clears. J863 output clears the Assembly / Disassembly FF K304/305 and clears the Write FF K302/303. The Write FF, in turn, drops the Write signal (T306 = 0). After 100 ns the Terminate FF K336/337 and the Terminate Precondition FF K334/335 clear.



(Without Assembly)

Figure 429. 3307 Data Flow on Input

3307 TIMING FOR READ NO ASSY/DSSY:
INSTRUCTIONS (73)(H = 0 + 1) OR (74)(N = 1)

Activate I/O Operation

1. Even Channel Enable (J310 = 1).
2. Drop Clear Enable (J308 = 0).
3. Clear O signal (R300 = 0) and Read signal (J306 = 1).
   a. R300 sets the Block Control Request Enable FF K366/367.
   b. J306 sets:
      1) Read FF K300/301 which sends a Read signal (T305 = 1).
      2) K310/311 if word count control.
      3) O Unloaded FF K340/341.
      4) Data Signal Precondition FF K342/343. This FF sets the Data Signal FF K344/345 and, after 100 ns, sets K368/369 which sends a Data signal (T309 = 1).
4. Wait for first Reply.

First Reply

1. a. R400 = 0.
   b. J451 = 1 for 100 ns. This clears the Data Signal Precondition FF K342/343, clears the O Unloaded FF K340/341, and sets the Block Control Request FF K332/333. FF K332/333:
      1) clears the Block Control Request $\overline{\text{Lockout}}$ FF K362/363.
      2) sends a Block Control Request (T301 = 1).
      3) after 100 ns Y400 = 1, which clears the Data Signal FF K344/345.
2. Wait to be serviced by Block Control.

Buffer Cycle

1. Even Channel Enable (J310 = 1).
2. O to Bus signal (J352 = 1) and J456 = 1 for 100 ns. J456 output:
   a. clears the Block Control Request FF K332/333.
   b. clears the Block Control Enable FF K366/367.
   c. gates O_U/O_L to the Data bus.
   d. gates O_U/O_L parity bit into O716/717 and O836/837.

   Trailing edge of O to Bus signal forces J455 = 1 for 100 ns, which sets the O Unloaded FF K340/341.
3. Clear O signal (R300 = 0). This sets the Block Control Request Enable FF K366/367.
4. Bus to O signal (R301 = 0).
   a. R301 output controls the Check Parity Error FF K328/329.
   b. Trailing edge of the Bus to O signal forces J457 = 1 for 100 ns and:
      1) sets the Block Control Request $\overline{\text{Lockout}}$ FF K362/363.
      2) sets the Data Signal Precondition FF K342/343.
      3) set the Data Signal FF K344/345 if the Reply is down for 100 ns.
      4) send data signal (T309 = 1) if K344/345 sets.
5. Wait for the second Reply.

Second Reply

1. a. R400 = 0.
   b. J451 = 1 for 100 ns. This:
      1) clears the Data Signal Precondition FF K342/343.
      2) clears the O Unloaded FF K340/341.
      3) sets the Block Control Request FF K332/333. This FF clears the Block Control Request $\overline{\text{Lockout}}$ FF K362/363, sends a Block Control Request (T301 = 1), and after 100 ns causes Y400 = 1 which clears the Data Signal FF K344/345.
   c. J400 = 1.
   d. J452 = 1 for 150 ns.
   e. J471 = 1 which gates the external receivers to O_U/O_L.
2. Wait to be serviced by Block Control

Buffer Cycle

1. Even Channel Enable (J310 = 1).
2. O to Bus signal (J352 = 1 and J456 = 1 for 100 ns). J456 output:
   a. clears the Block Control Request FF K332/333.
   b. clears the Block Control Request Enable FF K366/367.
   c. gates O_U/O_L to the Data bus.
   d. gates O_U/O_L parity bits into O716/717 and O836/837.

   The trailing edge of the O to Bus signal forces J455 = 1 for 100 ns. This sets the O Unloaded FF K340/341.

Operation Complete Signal

1. R313 = 0.
2. J363 = 1 which sets the Terminate Precondition FF K334/335.
3. Operation complete.
4. Clear O signal (R300 = 0). Cannot set the Block Control Request Enable FF K366/367.
5. Bus to O signal (R301 = 0) which controls the Check Parity Error FF K328/329. Trailing edge of the Bus to O signal:
   a. clears O716/717 and O836/837 for 100 ns.
   b. sets the Block Control Request $\overline{\text{Lockout}}$ FF K362/363.
   c. sets the Data Signal Precondition FF K342/343.
   d. cannot set the Data Signal FF K344/345. This causes the Terminate FF K336/337 to set, forcing:
      1) J866 = 1, which clears K368/369 and the Data Signal Precondition FF K342/343.
      2) J863 = 1, which clears the Read FF K300/301. After 100 ns the Terminate Precondition FF K334/335 and the Terminate FF K336/337 clear.

No Operation Complete Signal

1. Operation $\overline{\text{Complete}}$.
2. Clear O signal (R300 = 0). This sets the Block Control Request Enable FF K366/367.
3. Bus to O signal (R301 = 0) which controls the Check Parity Error FF K328/329. Trailing edge of the Bus to O signal forces J457 = 1 for 100 ns. The output of J457:
   a. clears O716/717 and O836/837.
   b. sets the Block Control Request $\overline{\text{Lockout}}$ FF K362/363.
   c. sets the Data Signal Precondition FF K342/343.
   d. if Reply down for 100 ns, set the Data Signal FF K344/345 which sends a Data signal (T309 = 1).
4. Wait for third Reply.
5. If (End of Record) ($\overline{\text{Word Count Control}}$), continue at number 1.
6. If (End of Record) (Word Count Control), continue at number 2.

Number 1

1. End of Record signal (J402 = 1, R402 = 0, and J471 = 1. J402 output:
   a. sets the Terminate Precondition FF K334/335.
   b. cannot set the Block Control Request FF K332/333.
   c. clears the Data Signal Precondition FF K342/343.
   d. clears the O Unloaded FF K340/341 if no Word Count Control.
   e. After 100 ns Y413 = 1 which clears the Data Signal FF K344/345. End of Record signal drops (R402 = 1). After 100 ns J450 = 0 which sets the Terminate FF K336/337 if no Word Count Control.
      1) J450 output forces J866 = 1, clearing K368/369.
      2) Terminate FF forces J863 = 1, clearing the Read FF K300/301. After 100 ns the Terminate Precondition FF K334/335 and the Terminate FF K336/337 clear.

   J471 output gates the external receivers to O_U/O_L.

Number 2

1. a. End of Record signal J402 = 1 which:
      1) cannot set the Terminate Precondition FF K334/335.
      2) cannot set the Block Control Request FF K342/343.
      3) allows the O Unloaded FF K340/341 to remain set.
      4) after 100 ns Y413 = 1, clearing the Data Signal FF K344/345, and Y412 = 1, setting the End of Record I FF K338/339.
   b. R402 = 0.
   c. J471 = 1 which gates the external receivers to O_U/O_L.
2. End of Record signal drops (R402 = 1). After 100 ns Y403 = 1 and J450 = 0. The Terminate FF K336/337 cannot set. The cleared Terminate FF:
   a. forces J866 = 0, preventing K368/369 from clearing.
   b. forces J863 = 0, preventing the Read FF K300/301 from clearing.

   Y403 output:
   a. sets End of Record II FF K346/347 and forces J865 = 0, which drops the Read signal (T305 = 0).
   b. after 100 ns clears the End of Record I FF K338/339 and forces J865 = 1 which brings up the Read signal (T305 = 1).
   c. 100 ns after K338/339 clears, Y338 = 1 which sets the Data Signal Precondition FF K342/343. This, in turn, sets the Data Signal FF K344/345 and sends a new Data signal (T309 = 1).
   d. 100 ns after the Data signal was sent, Y464 = 1 which clears K346/347.
3. Wait for the next Reply.

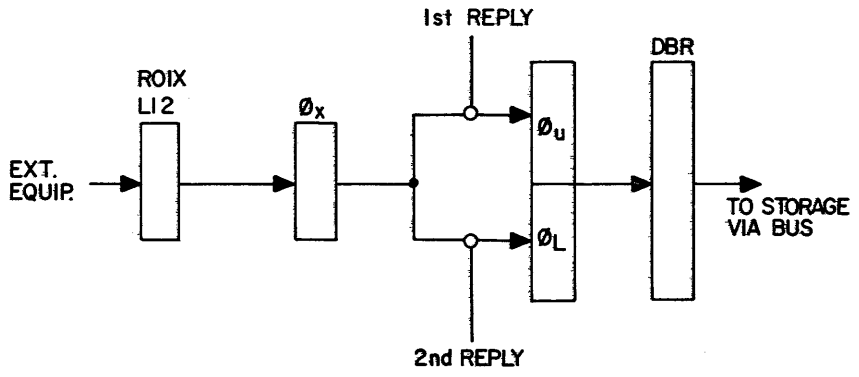


Figure 430. 3307 Data Flow on Input (With Assembly)

READ AND 12 TO 24 ASSEMBLY FORWARD

1. Activate 74 (12 to 24) Forward.
   Send Read Signal to Peripheral Equipment.
   Assembly/Disassembly Counter Clear, Clear.
2. First Data Signal to Peripheral Equipment.
   Assembly/Disassembly Counter Set, Clear.
3. First Reply.
   R16X to O_X to O_U.
   Assembly/Disassembly Counter Set, Set.
4. Second Data Signal to Peripheral Equipment.
   Assembly/Disassembly Counter Clear, Set.
5. Second Reply.
   R16X to O_X to O_L.
   Assembly/Disassembly Counter Clear, Clear.
   Request Block Control.

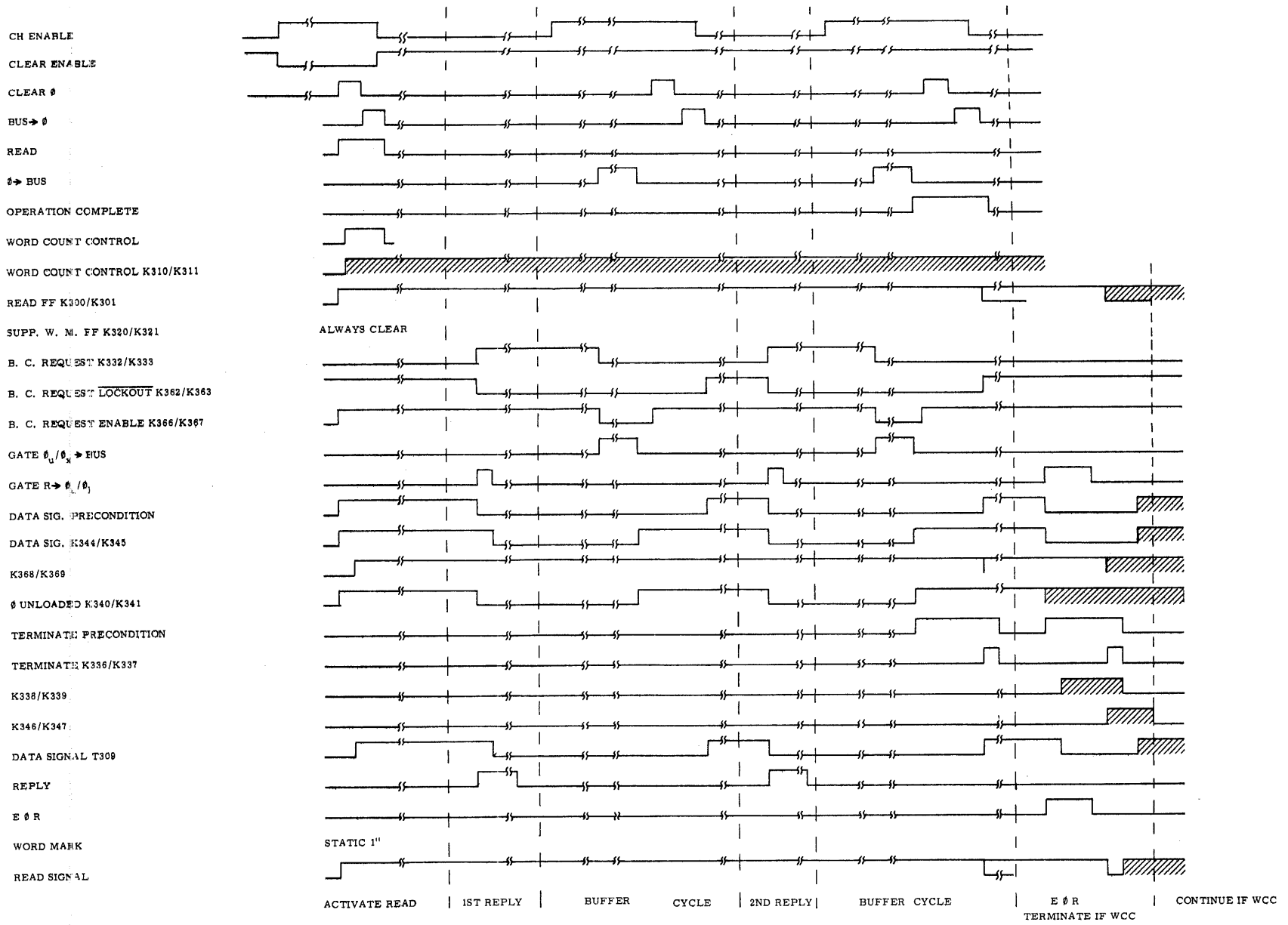


Figure 431. I/O Signals on Read (Assembly) with 3307

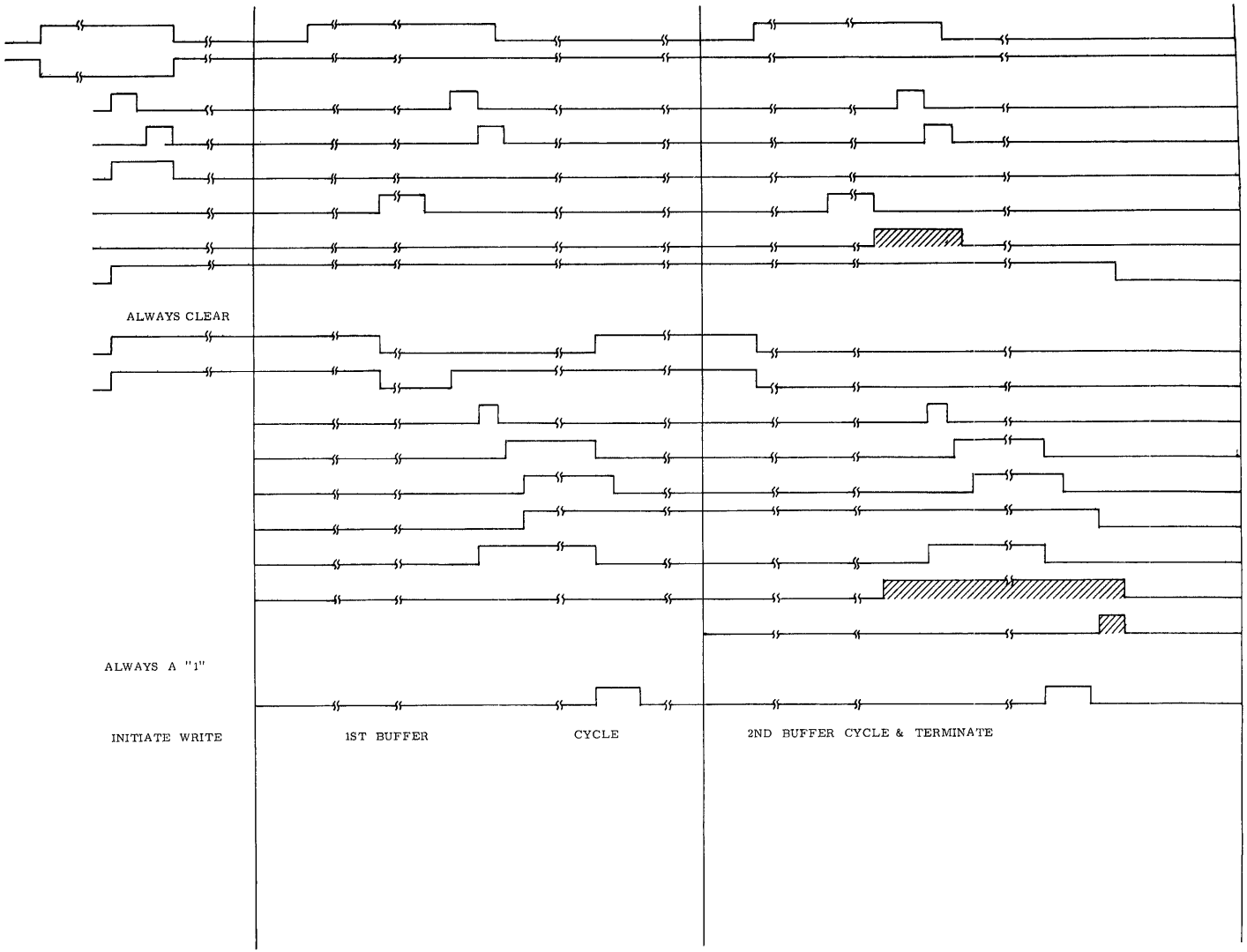


Figure 432. I/O Signals on Write (No Disassembly) with 3307

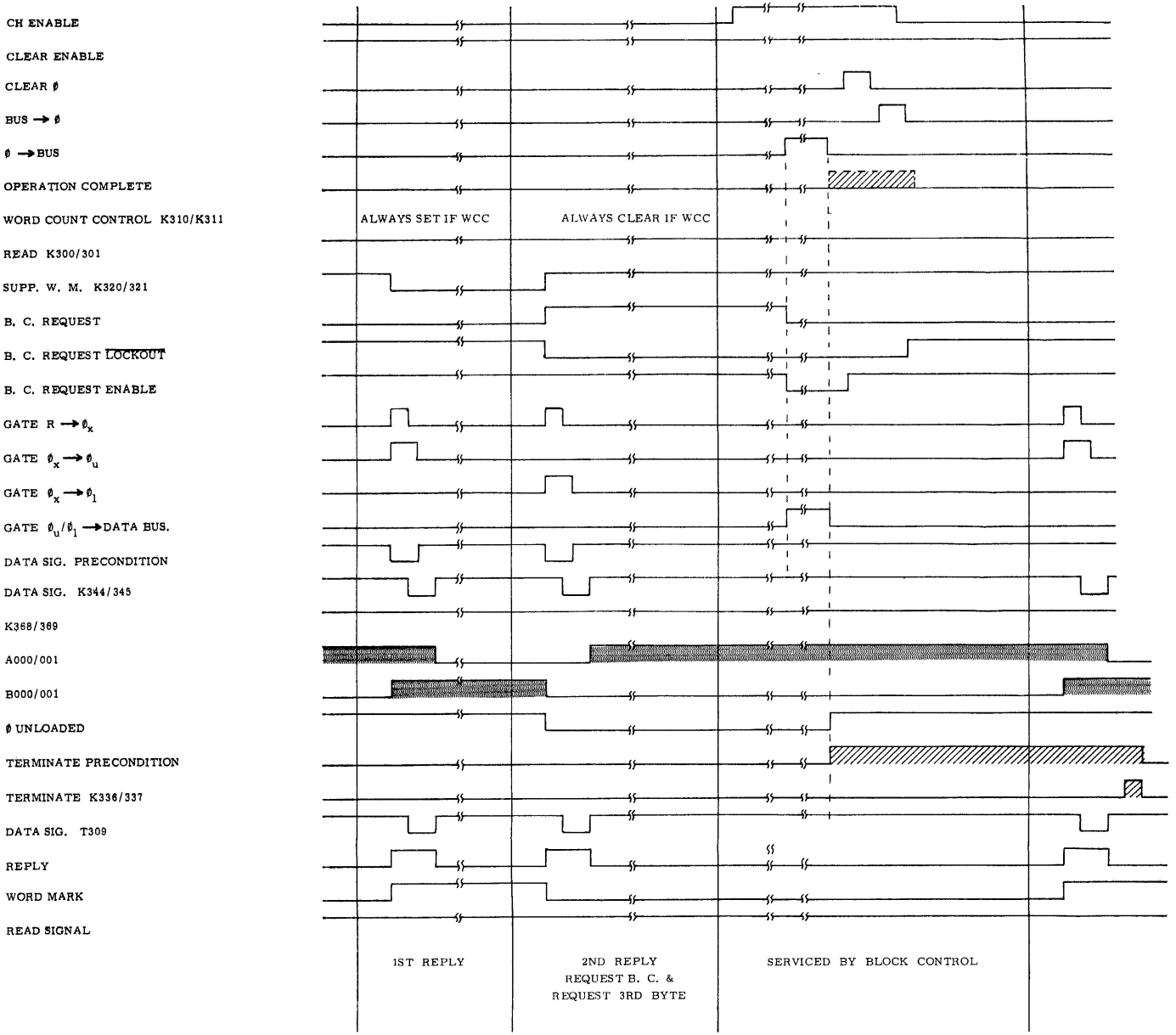


Figure 433. I/O Signals on Read Buffer Cycle (Assembly) with 3307

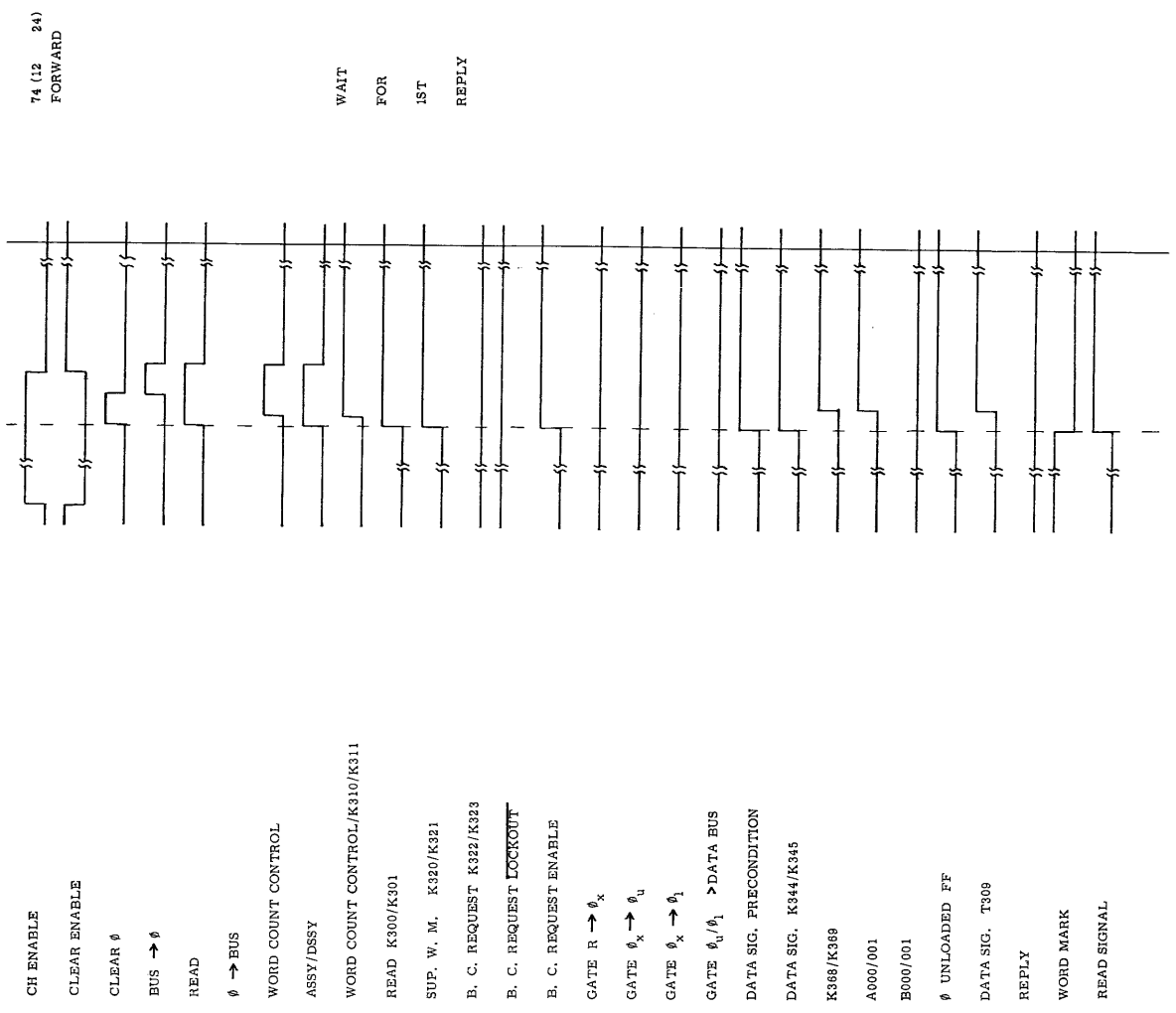


Figure 434. I/O Signals on Activate Read (Assembly) with 3307

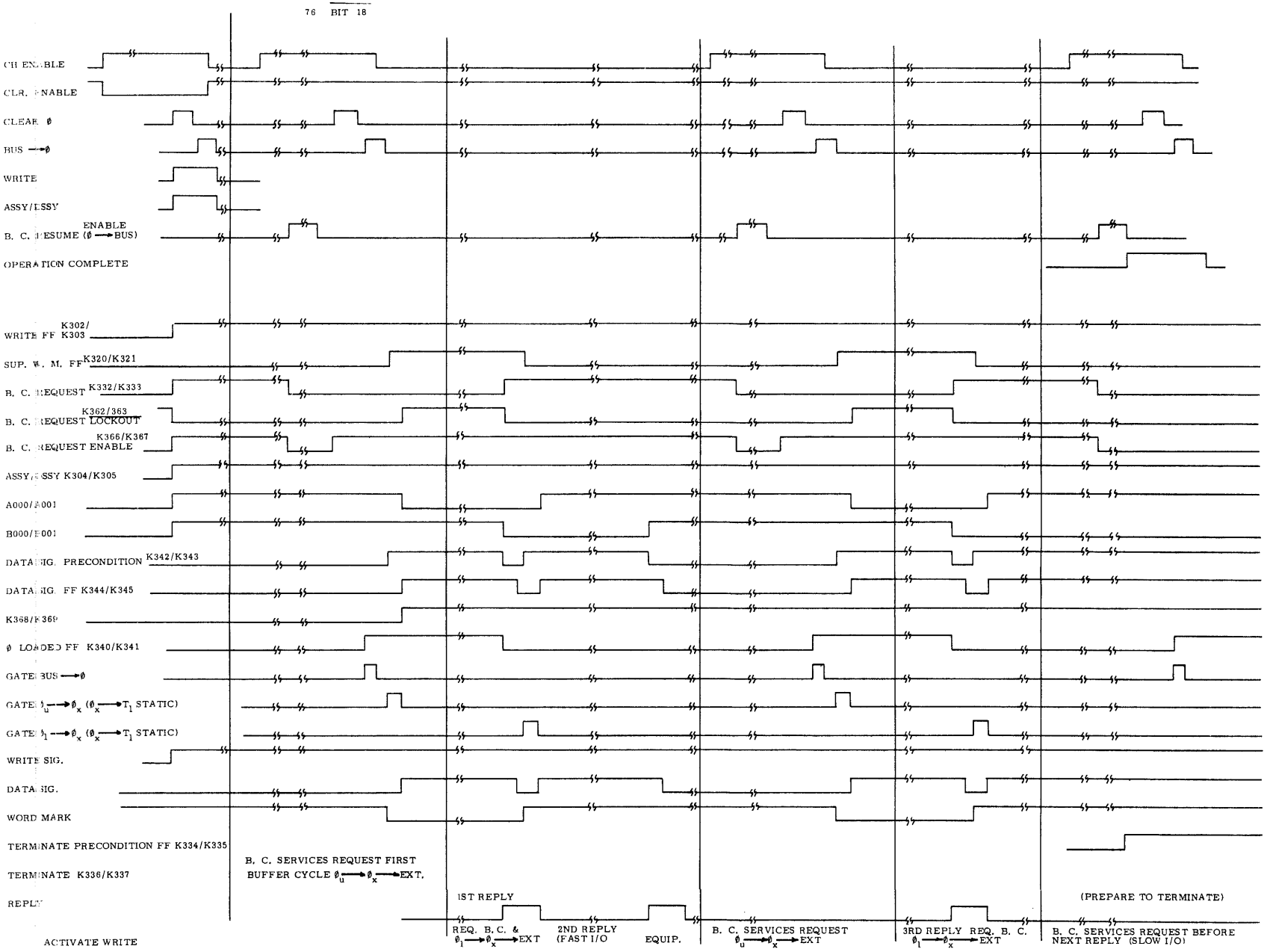


Figure 425. I/O Signals on Write Buffer Cycle (Disassembly) with 3307

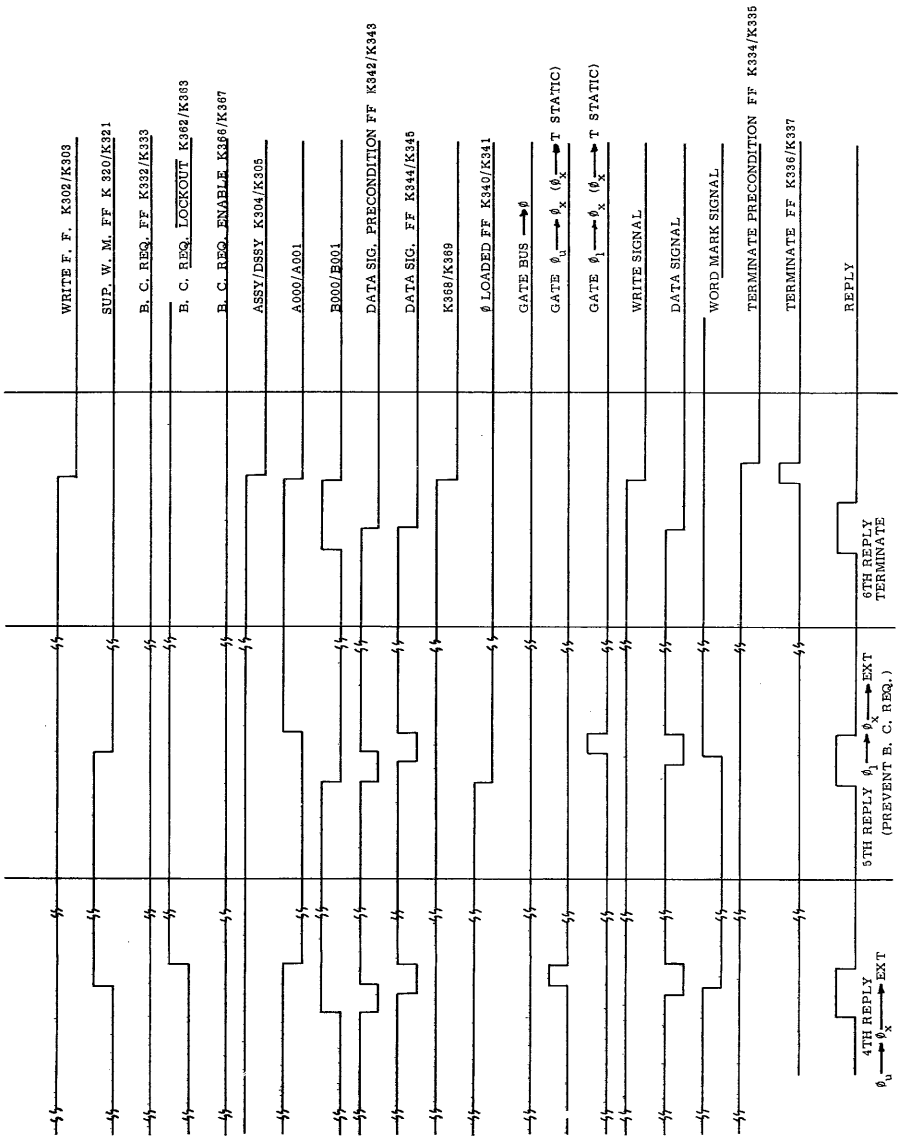


Figure 436. I/O Signals on Write Buffer Cycle (Disassembly) with 3307 (continued)

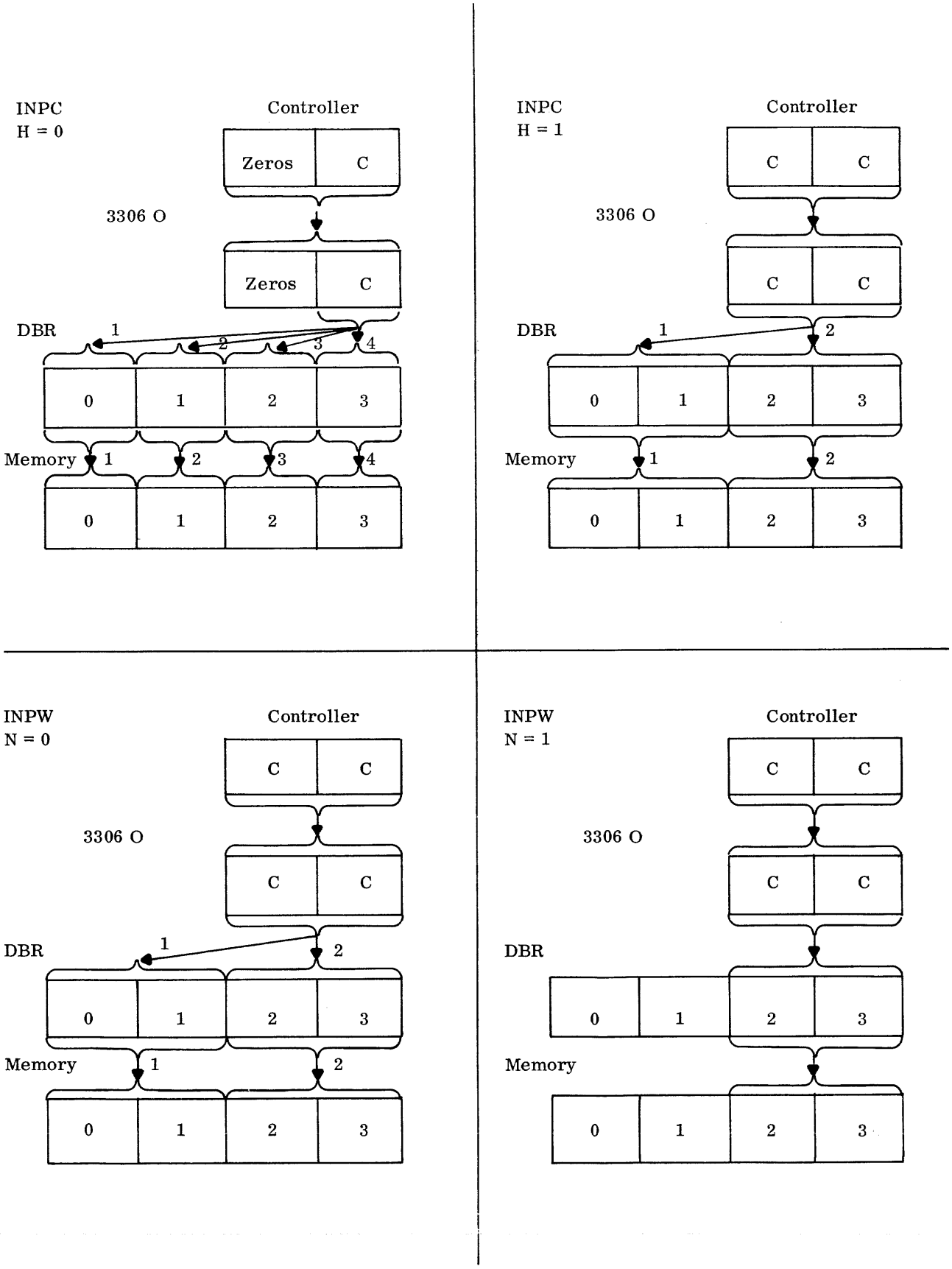


Figure 437. 3306 Forward (73 OR 74) Data Flow

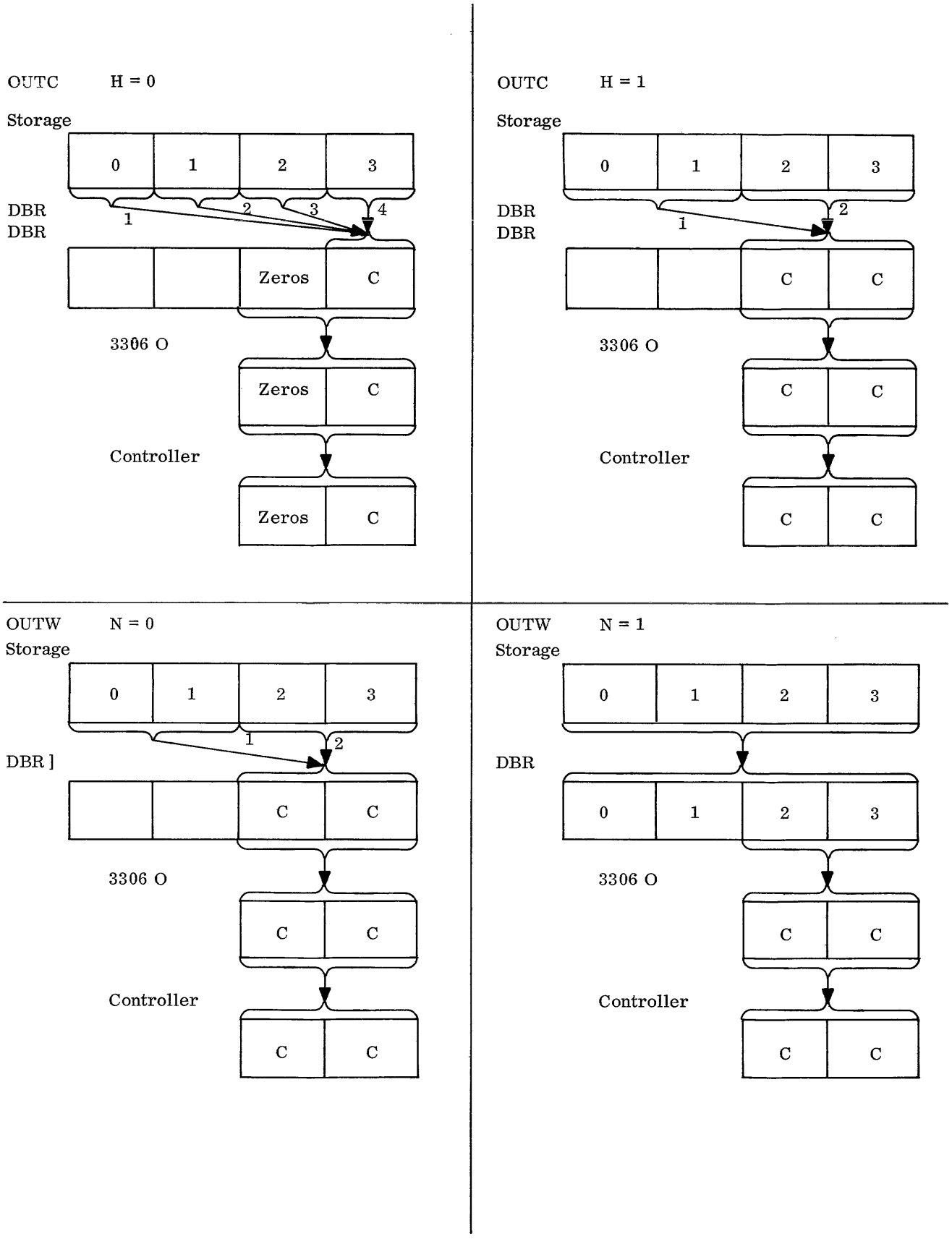


Figure 438. 3306 Forward (75 OR 76) Data Flow

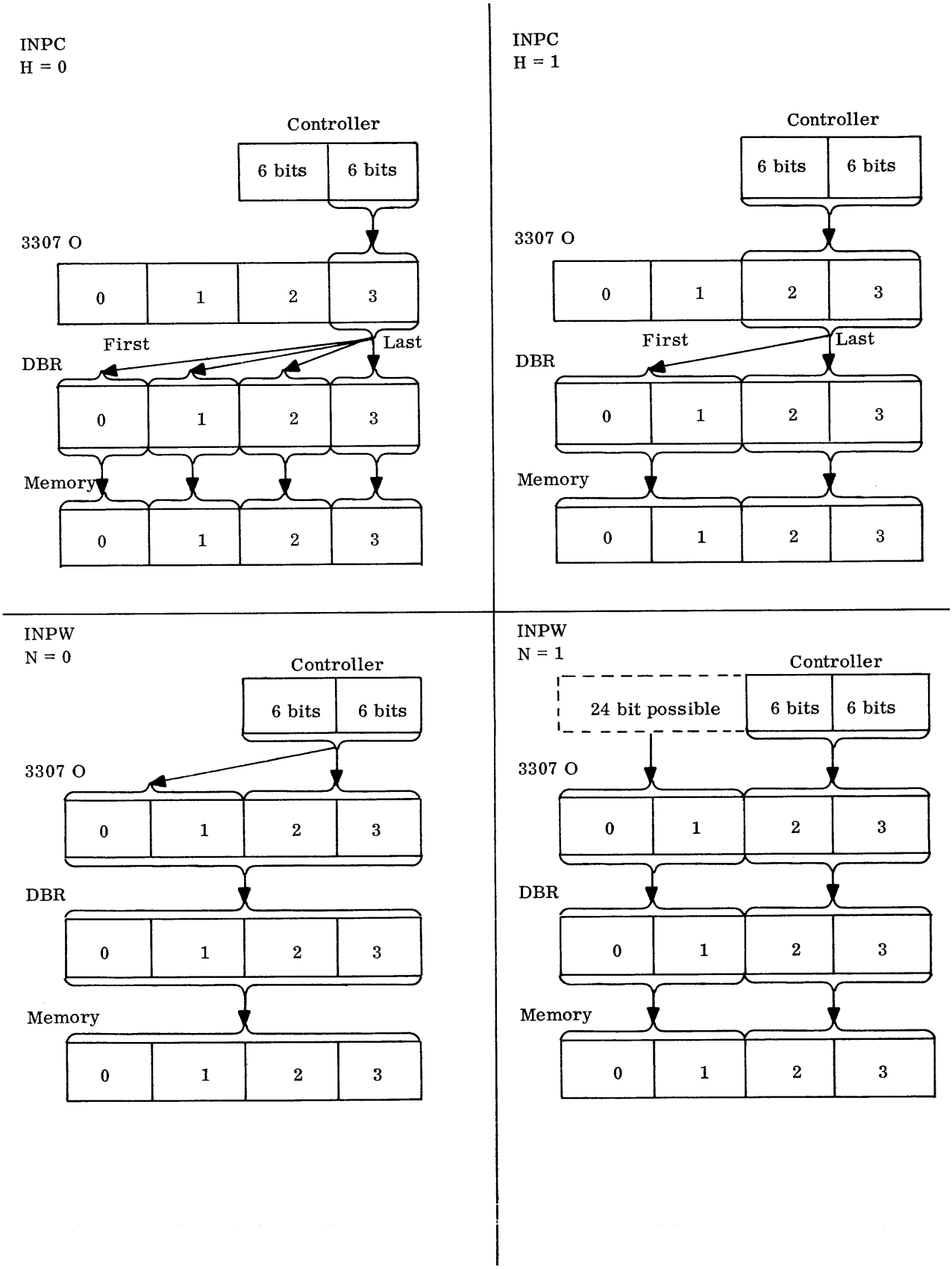


Figure 439. 3307 Forward (73 OR 74) Data Flow

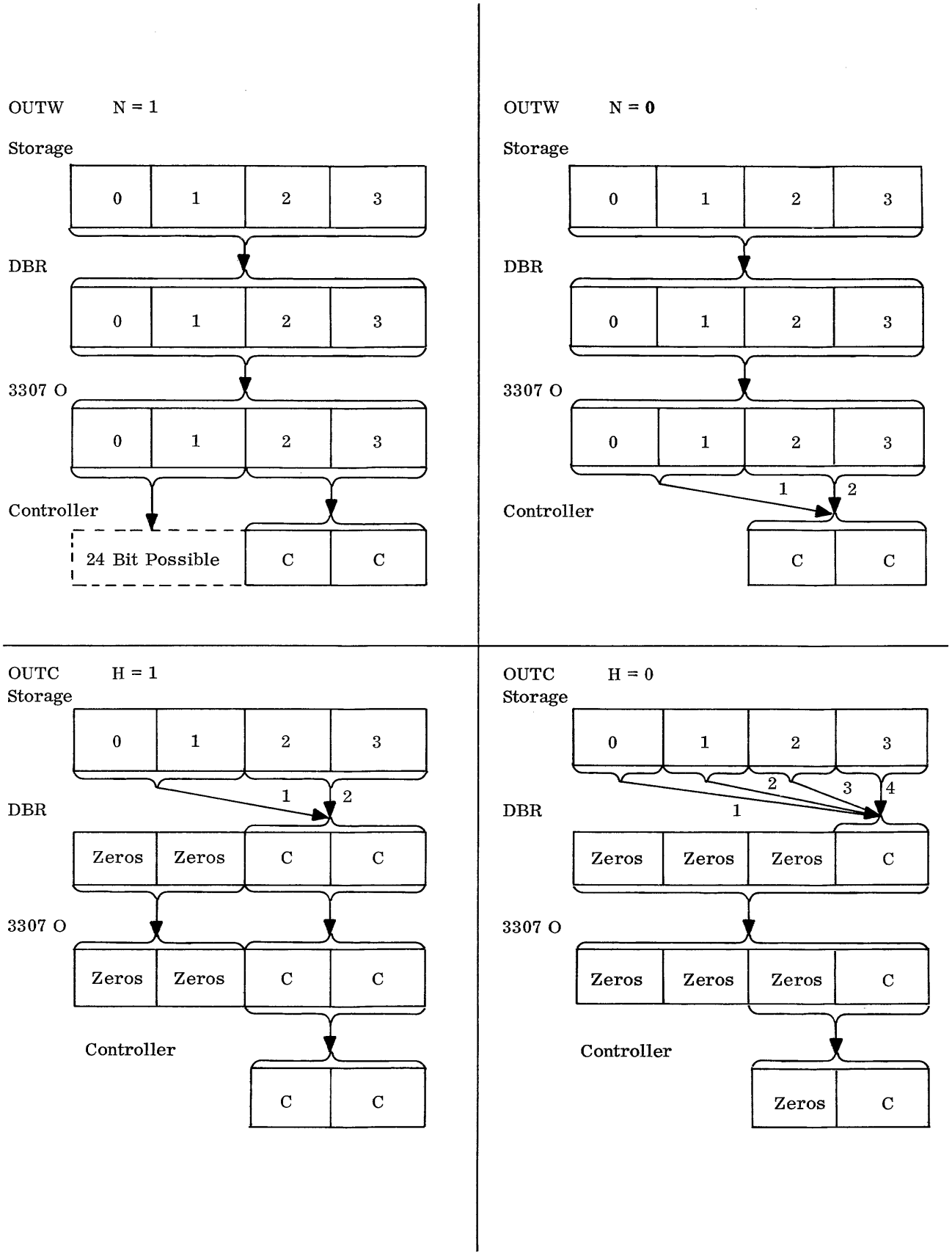


Figure 440. 3307 Forward (75 OR 76) Data Flow

Table 31. BIDIRECTIONAL SIGNALS

SIGNAL	DEFINITION
Data bits	Lines which carry data are bidirectional and perform as follows: 1. During a read (input) operation, data is transmitted from the external equipment to the channel. 2. During the write (output) operation, data is transmitted from the channel to the external equipment. 3. The connect code and function code are transmitted from the channel to the external equipment via the 12 data lines.
Parity bit	A parity bit accompanies each 12 bits transmitted between the channel and external equipment. Odd parity is used, so the total number of 1's transmitted is always an odd number. Parity bits accompany the connect code, the function code, and each 12 bits of data.

Table 32. SIGNALS FROM CHANNEL TO EXTERNAL EQUIPMENT

SIGNAL	DEFINITION
Connect	A 1 signal sent to external equipment when 12-bit connect code is available on data lines. If the equipment is available, it connects and returns the reply signal. If it is not available, it returns the reject signal. The connect signal and code drop when the external equipment returns the reply or reject. A channel may have a maximum of eight external equipments attached to its set of I/O cables. Eight equipments receive the 12-bit connect code and the connect signal but only one equipment (the one which has an equipment number switch setting matching the upper three bits of the connect code) will respond. (The other equipments do not return reject signals.) No response is returned by any of the equipments if the connect code contains a parity error; however, the transmission parity error indicators on all equipments will light. After 100 usec delay the channel generates its own internal reject. A connect code does not initiate action; it merely selects an external equipment. The upper three bits of the connect code select one of the eight possible equipments attached to the channel, and the lower nine bits specify the unit (if any) attached to that equipment. A connect code matching its equipment number switch setting will be accepted by an equipment if it is available, although it may be in the not reply condition. The connect code will be rejected only if the equipment is already connected to or reserved by another channel or is otherwise not available. The equipment will, however, enable its status lines to the channel which attempted to connect, so that the reason for the reject may be determined by using the copy status instruction. The status lines remain enabled to that channel until it transmits another connect code to any of its external equipments. Once an equipment is connected to a channel, it remains connected until the channel initiates a disconnect. Any connect code which does not match its equipment number switch setting will disconnect the equipment, including its status lines. The equipment must be capable of recognizing the code and disconnecting less than 1 usec after receiving the connect signal. During selection the equipment must not return a reply or reject sooner than 2 usec after receiving the connect signal.

Table 32 (continued)

SIGNAL	DEFINITION
Function	A 1 signal sent to external equipment when 12-bit function code is available on data lines. If the connected equipment is capable of executing the specified function when it receives the function signal, it initiates the function and returns the reply signal. If the equipment cannot perform the function, it returns the reject signal. The function signal and code drop when the external equipment returns the reply or reject. The 12-bit function code and function signal are received by the equipments attached to the channel, but only the connected equipments will respond. If no equipment is connected, the function signal and code will be completely ignored. After a 100 usec delay the channel generates its own internal reject. The specified function will not be performed if a parity error exists on the function code; however, a parity error signal is returned by the connected equipment. Also, the transmission parity error indicator on the connected equipment will light. Once a function code is accepted, all other function codes will be locked out until the first one is acted on. An equipment does not hold or stack up the function codes; a reply or reject is returned immediately. If the second function code received is identical to the first, the second function code will be rejected unless the function can immediately be performed a second time.
Channel busy	Static 1 signal sent to external equipment while data channel is active during a read or write operation. When the processor initiates the read or write operation the channel busy signal becomes 1 immediately and remains up until the operation is finished and no further operation is specified. It does not drop when an end of record signal is received unless the end of record actually terminates the operation. When the channel busy line goes to 1 the connected equipment immediately becomes busy (status response bit 1) unless it is already busy or is not ready. If the equipment is still busy from a previous operation, it finishes that operation before beginning the new operation. The equipment does not become busy if it is not ready; however, if the equipment becomes ready while the channel busy signal is up, the equipment will become busy.
Read	Static 1 signal directing the connected equipment to begin reading information from its storage medium and to continue as long as the read signal is present. The read operation always starts at the beginning of a record. If the read signal drops before the complete record is read, data transmission stops but the external equipment continues its action until the end of record is reached. If the read signal drops and comes back up within a record, data transmission stops and does not begin again until the beginning of the next record.
Write	Static 1 signal directing the connected equipment to begin writing information into its storage medium, and to continue as long as the write signal is present. The write operation always starts at the beginning of a record. Each time the write signal drops, external equipment automatically ends the record.
Data signal	A 1 signal used during read and write operations. Data signal drops when reply (or end of record) is received from external equipment. 1. During a read operation the data signal indicates that the channel is ready to accept data from the external equipment. 2. During a write operation the data signal indicates that the channel has placed output data on the data lines.

Table 32. (continued)

SIGNAL	DEFINITION
Word mark	A 1 signal sent to external equipment coincident with the data signal to indicate the last byte of the computer word. The word mark drops when the data signal drops (on receipt of reply or end of record).
Master clear	A 1 signal from the 3304 which returns channel and external equipment to 0 initial conditions and disconnects external equipment. Initiated by console EXTERNAL MC or keyboard MC switches.
Computer running	Static 1 when computer is running.
Suppress assembly/ disassembly	A 1 signal to the controller to force single character data transfers (73 + 75) (H = 0).
Clear external interrupt	A 1 signal initiated by INCL instruction and transmitted by channel. Not used by Control Data peripheral equipment.
Sample status time	A 1 signal transmitted by channel while it is gating external status to sense lines. Not used by Control Data peripheral equipment.

Table 33. SIGNALS FROM EXTERNAL EQUIPMENT TO CHANNEL

SIGNAL	DEFINITION
Reply	A 1 signal produced by external equipment in response to a connect, function, or data signal. Signal drops when connect, function, or data signal drops. 1. If connection can be made when connect signal is received, external equipment connects and returns a reply. 2. If specified function can be performed when function signal is received, external equipment initiates function and returns a reply. 3. During a read operation, external equipment sends a reply as soon as it has placed data on the data lines in response to the data signal. During a write operation, external equipment sends a reply as soon as it samples the data lines in response to the data signal. (If end of record is reached during a read operation, the reply is not returned in response to the data signal. Instead, the external equipment transmits the end of record signal.)
Reject	A 1 signal produced by external equipment in response to a connect or function signal if the connection cannot be made or the function cannot be performed at the time that the external equipment receives the respective signal.
End of record	A 1 signal produced (instead of reply) in response to the next data signal following the end of every record during a read operation. The end of record signal drops when the data signal drops. If the read signal drops before end of record, neither the end of record signal nor the remaining data in the record is transmitted although the external equipment continues its action to end of record. (This applies even though the read signal may have dropped and come back up again within a record. See definition for read signal.) Records of data written on magnetic tape are separated by blank spaces called interrecord gaps. In the case of punched cards, each card is a record.
Parity error	A 1 signal produced if channel does not send an odd number of 1's in 12 bits plus parity bit. (A parity bit accompanies each 12 bits. The external equipment checks the 12-bit portion and if there is an error it sends the parity error signal.) The following events occur when a parity error is detected. 1. Parity error on connect code: a. No equipment will connect. b. Any connected equipment will disconnect. c. No equipment returns a reply or reject. d. No equipment returns a parity error signal. e. The transmission parity error indicators on all equipments attached to the channel will light. 2. Parity error on function code: a. Nothing happens if no equipment is connected. b. If an equipment is connected, the following occurs: 1) It returns a parity error signal. 2) Its transmission parity error indicator will light. 3) It does not return a reply or reject. 4) It does not perform the function. 3. Parity error on data during write operation: a. Nothing happens if no equipment is connected. b. If an equipment is connected, the following occurs: 1) It returns a parity error signal. 2) Its transmission parity error indicator will light. 3) It uses the data. 4) It returns a reply.

Table 33 (continued)

SIGNAL	DEFINITION
Status bits	The external equipment indicates its operating conditions by placing information on the 12 status lines. Each equipment has its own particular set of status response codes, some of which are unique to that equipment. However, several status indications are normally common to all equipments and therefore occupy the same bit positions in the status response codes for all equipments. Ready (bit 0 = 1) An equipment is ready if, when properly connected, a read or write signal can initiate a read or write operation. Conversely, equipment is not ready if, when properly connected, a read or write signal cannot initiate a read or write operation. Once ready, an equipment remains continually ready until operation is no longer possible; it then becomes not ready. An equipment cannot become not ready while it is actually transferring information; information transfer must first be halted. Any equipment which requires manual intervention in its normal operation (such as loading tape, cards, paper, etc.) is provided with switches to put it in either a manual mode or a computer controlled mode. When in the manual mode the equipment is not ready. An equipment that has become not ready because of the need for manual intervention automatically goes into manual mode. It becomes ready again only after it has been attended to and manually switched back into the computer controlled mode. Busy (bit 1 = 1) Any equipment is busy when it is in operation. The equipment becomes busy immediately on initiation of the read or write operation. Normally, an equipment remains busy until it has finished all activity and is ready to perform another operation; it then becomes not busy. However, an equipment will become not busy if a condition arises due to which the equipment can no longer continue the operation. (An example of such a condition is becoming not ready because of running out of cards, paper, etc. An equipment cannot be busy if it is not ready.) Error (bit 10 = 1) This signal indicates information errors produced and detected by the external equipment. It does not indicate parity errors on information received from the channel; the parity error line is reserved for this. It is not an indication of malfunction such as paper tearing or printed circuit card failure. If a read or write operation includes more than one record, the error bit is not cleared between records, but will indicate an error anywhere in the operation. The error indication is cleared by beginning a new operation. Reserved (bit 11 = 1) This bit is used by multichannel peripheral equipment to indicate that the equipment is reserved by one of the channels to which it is attached. Once a multichannel equipment has been connected by a channel, it remains reserved by that channel even though the operation may terminate, the equipment may become not busy, and/or the channel may connect another equipment. No other channel can communicate with the equipment until the first channel releases the reservation. This may be done using a master clear switch, the clear channel instruction, or by issuing the appropriate function code.

Table 33 (continued)

SIGNAL	DEFINITION
Interrupt lines	A 1 signal on an interrupt line indicates that an external equipment has reached a predetermined condition. A channel may communicate with a maximum of eight equipments and each equipment uses one interrupt line. An interrupt signal may be dropped by reselecting the same selection or by clearing the selection. Each equipment has a set of conditions on which it will interrupt, if selected. Some of the interruptable conditions are common to all equipment and are described below. Interrupt on End of Operation With this selected, interrupt will occur the next time an operation ends. The operation may be in progress at the time of the selection or it may be initiated later. Interrupt will not occur from an operation which has ended before the selection is made. Interrupt on end of operation can occur both at the end of an I/O read or write operation and at the end of an operation specified by a function code. If a function code is accepted to initiate an operation that is already completed, an end of operation interrupt will occur, if selected. An example is a function code to rewind tape when tape is on load point. Normally, the end of operation interrupt for a read or write operation will occur when all data has been transferred, the channel busy signal has dropped, reading or writing of the current record is completed, and all error checking is completed. In some cases, this interrupt may occur before the equipment becomes not busy. If for any reason (such as becoming not ready) the equipment is unable to continue the activity, the equipment will end its operation and interrupt will occur. Interrupt on Abnormal End of Operation This directs the external equipment to interrupt if an operation ends under circumstances other than normal, such as becoming not ready or detecting an error. The operation may be in progress at the time of the selection or it may be initiated later. Interrupt will not occur from an operation which has ended before the selection is made, even though it may have ended under abnormal circumstances. However, if the equipment has become not ready before the interrupt is selected, an attempt to initiate another operation after the selection is made will cause the equipment to interrupt immediately. The equipment does not send the interrupt signal while information is being transferred. All activity and information transfer are stopped at the most consistent point (such as at the end of the current record); then the interrupt occurs. Automatic stopping on an error takes place only when this interrupt is selected. Interrupt on Ready, or Interrupt on Ready and Not Busy This interrupt indicates that the external equipment is ready to start a new operation. It is often used to indicate the completion of any manual intervention.

SELF-EVALUATION QUIZ ON CHAPTER 16

TRUE OR FALSE OR FILL IN THE BLANKS:

1. A maximum of _____ 3307s may be used in a system.

2. A channel clear will cause the channel to appear busy.

3. The 3306 is a _____-bit data channel and the 3307 is a _____-bit data channel.

4. The communications modules have direct access to the S bus.

5. The DB register has the capability of generating two parity bits to facilitate operations on the 24-bit data channels.

6. The 3307 has assembly/disassembly capabilities.

7. The communications modules ignore word count control on output operation.

8. The communications modules will generate an internal reject if no response is received from the external equipment during a read or write.

9. The no response reject will occur 100 usec after a connect or function if no reply is received.

10. The end of record signal may terminate an input but not an output.

11. The communications modules check parity for every data transfer on a write operation.

12. The 3307 is a truly independent buffer channel.

13. During a write operation the write signal pulses with the data signal.

14. On 24 to 6 write operations work mark accompanies each data transfer.

15. When doing a 12 to 24 operation with the 3307, block control will be requested for each data transfer.

Score Yourself:
None wrong is perfect, superior work
One wrong is above average.
Two wrong is average.
Three or more wrong is below average.

APPENDIX A
ANSWERS TO QUESTIONS IN VOLUME II

CHAPTER 4

Page 122, Self-Evaluation Quiz

1. False
2. True
3. False
4. False
5. False
6. True
7. True
8. True
9. False
10. False
11. False
12. False
13. 3 bit
14. True
15. False
16. True
17. True
18. True
19. False
20. False

CHAPTER 5

Page 146, Self-Evaluation Quiz

1. True
2. False
3. True
4. 5
5. True
6. False
7. True
8. True
9. True
10. False
11. False
12. 128
13. 2 x 24 x 64
14. False
15. False
16. True
17. False
18. 12
19. True
20. True

CHAPTER 6

Page 171, Self-Evaluation Quiz

1. False
2. False
3. True
4. False
5. True
6. False
7. True
8. True
9. True
10. True
11. False
12. True
13. F354
14. False
15. False

CHAPTER 7

Page 192, Self-Evaluation Quiz

1. False
2. True
3. False
4. True
5. False
6. True
7. 111_2
8. EXX2, character, word
9. False
10. True
11. False
12. True
13. False
14. False
15. False

CHAPTER 8

Page 204, Self-Evaluation Quiz

1. False
2. False
3. True
4. True
5. False
6. True
7. False
8. True
9. False
10. True

CHAPTER 9

Page 217, Self-Evaluation Quiz

1. True
2. False
3. False
4. False
5. True
6. True
7. True
8. False
9. False
10. False

CHAPTER 10

Page 234, Self-Evaluation Quiz

1. True
2. False
3. True
4. True
5. False
6. True
7. True
8. True
9. False
10. True
11. True
12. True
13. True
14. True
15. False

CHAPTER 12

Page 298, Self-Evaluation Quiz

1. True
2. False
3. True
4. True
5. False
6. False
7. Depends on which load instruction is executed
8. False
9. True
10. True
11. True
12. True
13. False
14. False
15. True
16. False
17. True
18. False
19. True
20. True

CHAPTER 13

Page 348, Self-Evaluation Quiz

1. 3310, 4
2. 56, 57, 60, 63
3. False
4. K804/805 (optional arithmetic busy)
5. True
6. False
7. True
8. True
9. True
10. True
11. True
12. Divide step, SWAP, complement
13. 61_8
14. True
15. False
16. False
17. False
18. True
19. False
20. K872/873 (add exp 2)

CHAPTER 14

Page 363, Self-Evaluation Quiz

1. Abnormal Interrupt, normal interrupt, trapped instruction interrupt
2. True
3. False
4. False
5. True
6. True
7. True
8. True
9. False
10. True

CHAPTER 15

Page 386, Self-Evaluation Quiz

1. 2, 27, 64
2. True
3. True
4. False
5. True
6. False
7. False
8. False
9. False
10. True
11. True
12. True
13. True
14. True
15. False

CHAPTER 16

Page Self-Evaluation Quiz

1. 4
2. True
3. 12, 24
4. False
5. True
6. True
7. True
8. False
9. True
10. True
11. False
12. False
13. False
14. True
15. False