SM 2003

SM 2003

SERVICE MANUAL for DMA/DISC STORAGE UNIT CONTROLLER

September 15, 1975

REVISED: July 15, 1976

basic/four corporation
an MAI company ®

Printed in U.S.A.

DMA/DISC STORAGE UNIT CONTROLLER

LIST OF EFFECTIVE PAGES

Page	Revision Date
1-1	07-15-76
1-2 thru 1-6	Original
2-1 thru 2-40	Original
3-1 thru 3-5	07-15-76

TABLE OF CONTENTS

SECTION 1. GENERAL DESCRIPTION Page

SECTION 2. MAINTENANCE

LIST OF TABLES

LIST OF ILLUSTRATIONS

Section 1

GENERAL DESCRIPTION

1.1 SCOPE

This manual contains the general description and maintenance data for the DMA/Disc Storage Unit Controller manufactured by Basic/Four Corporation, 18552 MacArthur Boulevard, Santa Ana, California 92707. The DMA/Disc Storage Unit Controller (controller) is an integral part of BASIC/FOUR® Model 600 data processing systems. The controller provides the interface between the Central Processing Unit (CPU) and the Models 2224-200 and 2324-200 Disc Storage Units (Disc Drives).

1.2 PURPOSE

The controller is used to interface up to four Disc Drives to the CPU. The controller decodes instructions from the CPU to select the appropriate Disc Drive, Disc, and read/write head then command the selected Disc Drive to position the read/write head over the appropriate track and write data on or read data from the disc. The controller also transfers status and data directly to the CPU memory and, on command, searches the data read from a track to locate specific data identified by a key code. All of these operations are performed automatically by the controller once they have been initiated by the CPU.

1.3 PHYSICAL DESCRIPTION

The controller consists of two circuit boards connected by a flat cable at the rear edges of the boards. The front edges of the boards plug into adjacent slots on the CPU motherboard. One of the circuit boards is the interface between the CPU and the Disc Drives and has a second rear-edge connector for the cable to the first Disc Drive. The other board contains the DMA (direct memory access) and the microprocessor that controls the automatic operations of the controller.

1.4 FUNCTIONAL DESCRIPTION (figure 1-1)

The controller transfers data between the CPU and the Disc Drive on command from the CPU. Certain commands from the CPU are decoded and routed directly to the Disc Drive, however, commands for data transfer in either direction and keysearch

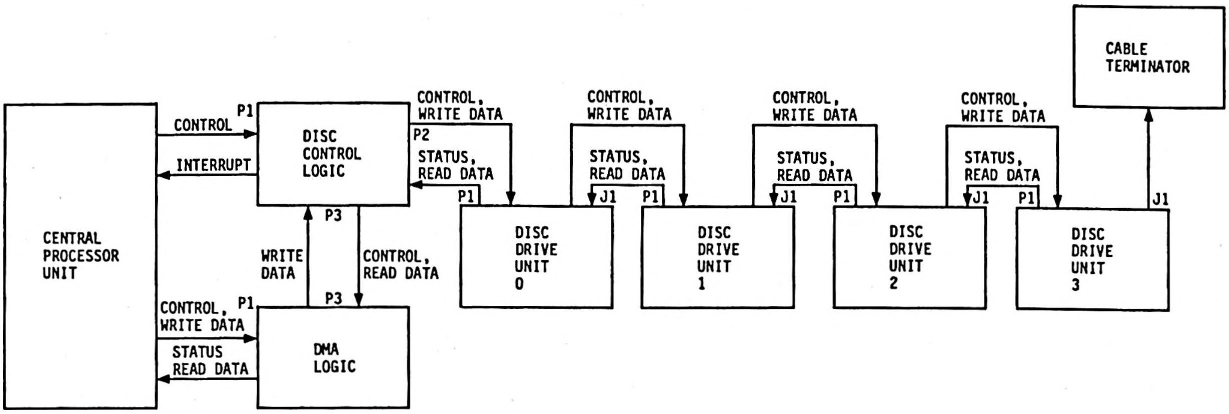


Figure 1-1. DMA/Disc Drive Unit Controller Information Flow Diagram

commands actuate microprocessor routines in the controller. Therefore, it is the microprocessor that actually controls data handling in the controller.

Control signals from the CPU establish initial conditions in the disc control logic and select the specific locations for the commanded operation. These control signals result in the selection of a Disc Drive Unit, and placing its read/write heads over/under a specific cylinder. The selected Disc Drive Unit returns status signals and index/sector pulses. The disc control logic determines from the index/ sector pulses when the specific commanded location is available for a read, write or keysearch operation. The status signals are converted to a status or alternate status byte and returned through the DMA logic to the CPU on command.

When the commanded operation is write, the write data is supplied to the DMA logic one byte at a time. The DMA logic supplies this data to the Disc control logic one bit at a time where it is converted to double frequency write date and transferred to the selected Disc Drive Unit.

When the commanded operation is read, data is read from the disc one bit at a time and assembled into a byte by the disc control logic. The byte is transferred through the DMA logic to the CPU.

When the commanded operation is a keysearch, the next higher key is transferred to the DMA logic. A block containing keys is read and each key is compared to the reference key from the CPU. The first key read from the block is also compared to zero. If the first key is less than the reference key, it becomes the trial best key. Subsequent keys are read and compared with the best key and the present trial best key. As a key is read that is less than the reference key and greater than the trial best key, it becomes the trial best key. After all keys in the block have been compared, an interrupt is sent to the CPU. The CPU then initiates a keyresult operation and the number of the best key is returned to the CPU.

Signals between the DMA/Disc Drive Unit Controller and the Disc Drive Units are cabled in a daisy-chain arrangement. That is, the first cable is from P3 on the disc board to Disc Drive Unit 0 in the CPU cabinet; the second cable is from J1 on Disc Drive Unit 0 to P1 on Disc Drive Unit 1; and so on. There must always be a cable terminator assembly on J1 of the highest numbered drive in the system, whether it is drive 0 or another number up to 3.

1.4.1 DISC LOGIC BOARD (figure 1-2)

All controller operations are initiated by the CPU and the commands for these operations are processed on the disc board. Signals on the control bus are decoded and two controls are recognized that sequentially enter commands and addresses from the I/O bus into the disc logic circuits. Commands that are addressed to the controller select the Disc Drive Unit, the disc, the read/write head, the cylinder (track) number, the sector number, and the mode. The commands may also include status requests. Track number selection requires the transfer of four bytes from the I/O bus. Disc, head, sector number and format selection requires one transfer of two bytes from the I/O bus. Other commands require the transfer of one byte from the I/O bus.

Since there may be up to four Disc Drive Units in a system, each operation involving a Disc Drive Unit must begin with a drive select command. This causes the disc logic to send a drive select order to the Disc Drive Units. Usually the next command is a status request to confirm that the drive is ready. The next command is to move the read/write heads over a new track. This causes the disc logic to send a cylinder address and seek order to the Disc Drive Unit. Another status request may be issued to verify that the heads are correctly positioned. The next command selects the disc and head, contains the sector number, and specifies if the read operation is formatted or not. This causes the disc logic to order the Disc Drive Unit to enable the head for the selected disc and to compare the sector number with sector counter outputs. This sequence is common to all operations involving the Disc Drive Unit. After this preliminary sequence, the CPU commands

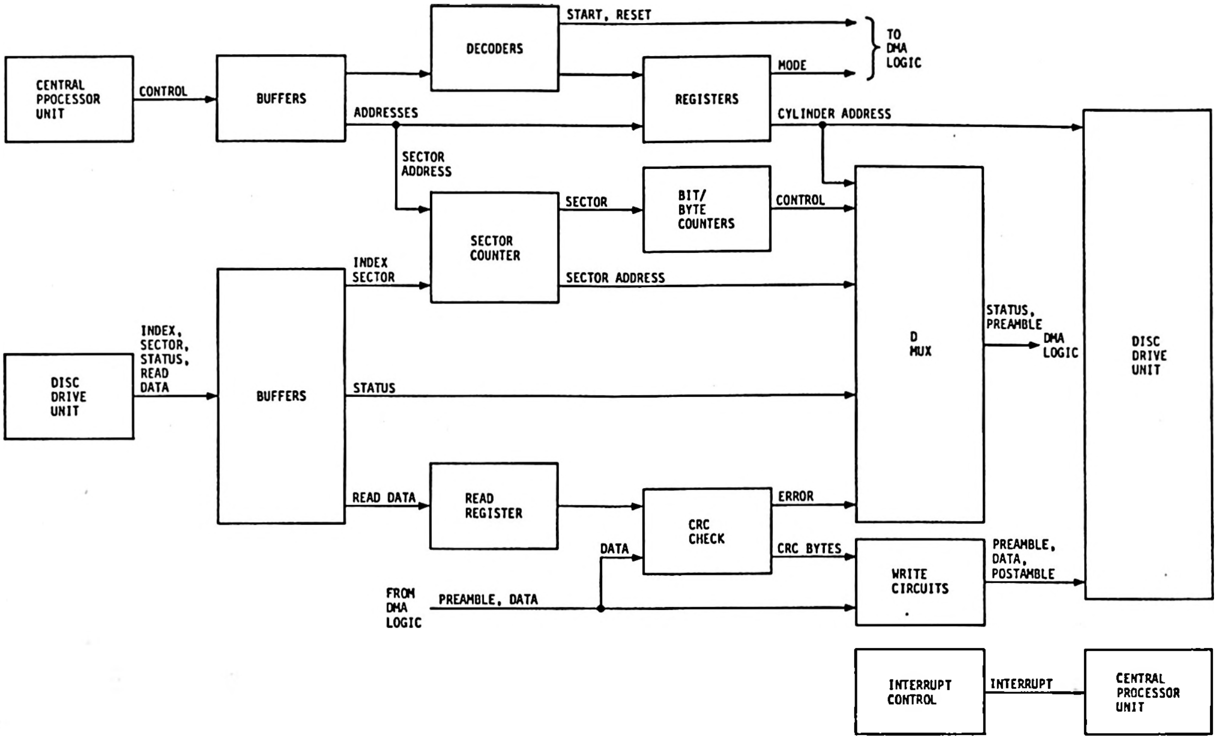


Figure 1-2. Disc Control Logic (Disc Board) Block Diagram

a read (only) mode, a write mode, a keysearch mode or a keysearch result mode. Once a mode is commanded, the operation of the controller is controlled by the DMA logic on the microprocessor board even though the read and write signals are partially processed by the disc control logic.

The information recorded in each sector on a disc consists of a preamble, a data field, and a postamble. The data field is, of course, the stored information for which the disc and the Disc Drive Unit exists. The information in the preamble and postamble is for housekeeping purposes. The preamble uniquely defines that sector and contains a bit/byte pattern that can be recognized for synchronization purposes. The postamble contains a cyclic redundancy check (CRC) polynomial which is used to verify read/write accuracy and a "dead spot" to absolutely mark the end of the sector. The disc control logic assembles the formatting and information for storage in the preamble and postamble.

1.4.2 DMA LOGIC BOARD (figure 1-3)

All transfers between the CPU memory and the controller are through the DMA logic. The DMA logic gets the CPU memory start address and end address which are the boundaries of a read or write data block, the data to be written on the disc, and keysearch parameters from the CPU memory. The DMA logic enters status, read data, and keysearch result bytes in the CPU memory.

The microprocessor contains subroutines for status transfers, read, write, keysearch, and keyresult operations. The program counter is conditionally jumped to the appropriate subroutine as a result of the mode selected and subsequently jumped as a result of conditions encountered during transfers to and from the Disc Drive Unit. Once a read or write operation begins, the start address is incremented at the end of each byte transfer until the end address is reached. The microprocessor then issues a halt which causes a DMA interrupt to the CPU.

During a keysearch the keysize, number of keys to be examined and the number of sectors to be read for keys is entered from the CPU and stored in the scratch pad memory. A reference key is then transferred from the CPU memory to the KD first-in-first-out (FIFO) memory and the KBO FIFO memory is set to all zeroes. As each key is read from the search area, it is compared byte for byte with the KD and KB keys. Key bytes are examined in descending order of their significance (MSB first) so the "best key" decision need only be made once for each key. A best key is less than KD and greater than KB. Since any real key read must be greater than zero, the first key read which is less than KD automatically becomes the trial best key (KB) so subsequent keys examined must be greater than it and less than KD to become the trial best key. The best key is the trial best key left after the defined area has been searched.

An example of possible keys illustrates how the keysearch works. Assume a group of files identified by alphameric keys from 1240AB through 1249SX and file 1243DE is to be updated. KD would be 1243DF. Since the keys are stored in ascending sequential order, the first key read (1240AB) would become the trial best key only to be replaced by the next key read and so on until 1243DE is read and becomes the trial best key. All keys read thereafter would be equal to or greater than 1243DF and would not become trial best keys because they failed to satisfy one of the two comparison tests.

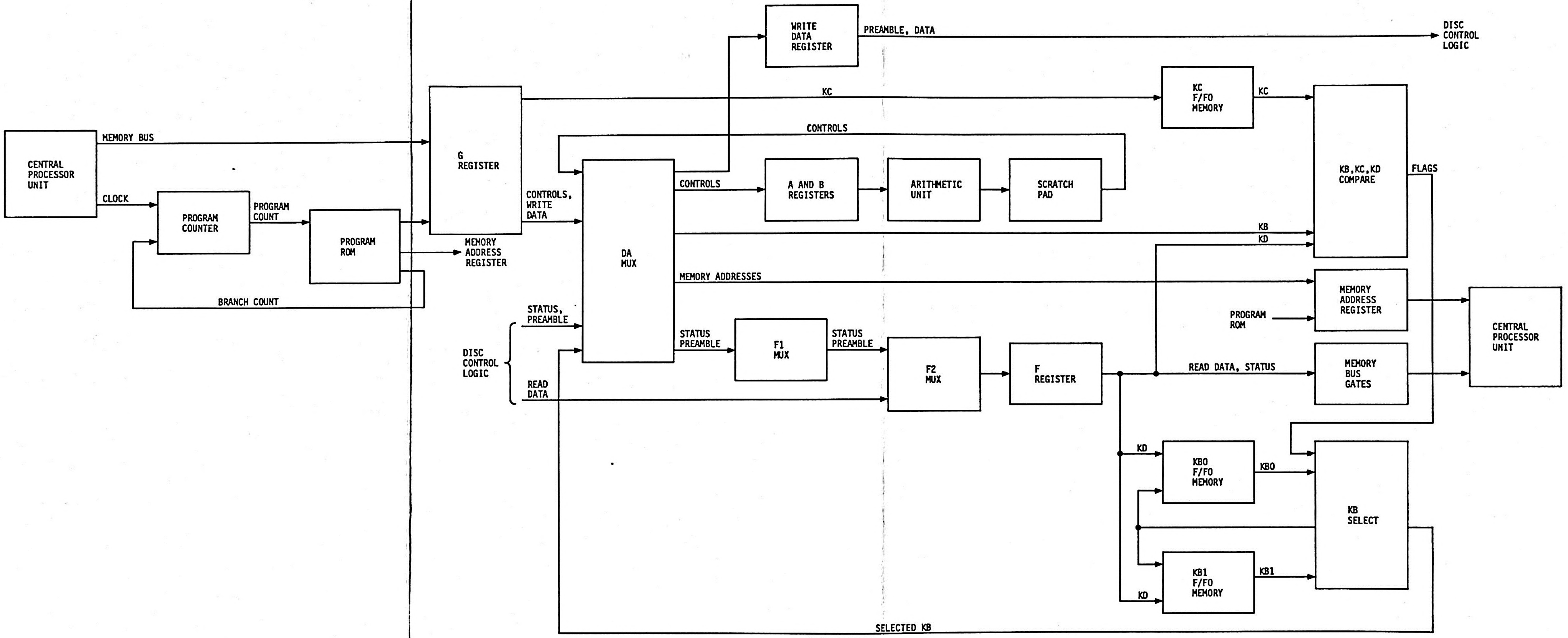


Figure 1-3. DMA Logic (Microprocessor Board) Block Diagram

(controller clear). CTLRST- resets routine and control flip-flops and the keysearch memory in the DMA logic. This established the conditions in the controller that are necessary for the start of any of the sequential operations performed by the controller.

2.5 CONTROLLER OPERATIONAL SEQUENCES (figure 2-2)

Before there can be any transfer of data to or from a disc, the Disc Drive Unit containing the disc must be selected then the cylinder (track) number must be selected. After the Disc Drive Unit has positioned the read/write heads over the correct cylinder, the appropriate beginning sector, disc, and head must be selected. When all of the selections listed above have been made, the controller may cause data from the CPU to be written on the disc, cause data to be read from the disc for use by the CPU, or conduct a keysearch.

A flow diagram of the operations which must be performed before a data transfer is shown in figure 2-2, sheet 1. After master reset has ensured that the controller is in the start condition the CPU must issue a series of commands to access the area where data is to be written on or read from a disc. Each of these commands must be addressed to the controller. After each command the CPU normally issues a status request through the command logic in order to verify that the command has been successfully executed. When the appropriate head is positioned over the correct track by the selected Disc Drive Unit and the sector counter indicates the sector just before the beginning sector, the read gate is made active so that sector will be read as an operational check. When the next sector pulse is detected by the controller, the operations dictated by the commanded mode are commenced. Since the accuracy of a write operation is verified by read-after-write, all sectors included for the operation are read. During keysearch mode the sectors are read to find the best key.

2.6 CPU COMMAND TRANSFER (figure 2-2)

Each time the CPU sends a command to the controller, it must send a control out (COXX) instruction on the control bus and the controller address in the first four bits (OD00 through OD03) of the I/O bus. The actual command is coded in bits OD04 through OD07. The control bus bits (I01X- through I03X-) are decoded to detect the control out and the controller address is detected by a gate. Coincidence of these two signals sets the connect latch (CONN) and loads the command bits (OD04 through OD07) into the command register. The CPU must then put a data out (DOXX)

Section 2

MAINTENANCE

2.1 SCOPE

This section contains the theory of operation and troubleshooting information for the controller. There are no adjustments on the controller and no preventive maintenance is specified.

2.2 THEORY OF OPERATION

The theory of operation is intended to provide the repairman with sufficient information to be able to identify when it is the controller that is failing and then which board contains the source of the failure. The text is supported by the block diagrams at the end of this section. Both the text and illustrations are arranged in a functional flow sequence for (1) preparing the controller and Disc Drive Unit for one of the operational modes and (2) for data transfers in each mode. Logic diagrams for the controller are contained in LD 2003. The blocks in the block diagrams reference the applicable logic diagrams by including the LD 2003 page number in parentheses.

2.3 CLOCK AND SIGNAL DISTRIBUTION (figure 2-1)

There is a clock pulse buffer and inverter on both boards. DCPH2+ and DCPH2- are used on the disc logic board to synchronize the DMA interrupt, sector pulse detection, and read data transfers to the CPU clock. BCPH2+ and BCPH2- are used on the DMA board to synchronize the program counter, the memory address register, DMA requests to the CPU, and memory interface register to the CPU clock.

2.4 SYSTEM START (figure 2-2)

When the CPU is turned on the operator presses the front panel switches CLOCK, RESET, RUN in that sequence. This generates master reset (MRST-) which is buffered on both boards to produce resets for the Disc Control and DMA logic circuits. Buffered master reset (BMRST-) resets the connect latch and the command register. BMRST- or CLRC- (clear controller) generate SRST- (system reset) which is used to clear flip-flops and registers in the Disc Control logic, MRST- is combined with the CLRC- + SPDMA- (stop DMA) signal from the Disc Control to generate CTLRSTinstruction on the control bus. This is decoded by the control decoder to obtain DOXX. DOXX and CONN strobe the command decoder to decode the command stored in the command register. When the command involves selecting from a choice such as selecting one of 4 drives or one of 400 tracks (cylinders), the selection (address) is indicated by the logic states of some or all of the bits on the I/O bus during the DOXX command.

The first command in operations involving the Disc Drive Unit is drive select (DRSEL-). During a DRSEL- command, bits OD00+ and OD01+ are the address bits for the drive. OD00+ and OD01+ are stored in the drive select register and decoded by the drive select decoder. This drive select signal (one of DRSEL0- through DRSEL3-) remains active throughout the operation to be performed and until the next DRSEL- is decoded or until system reset is active. One drive select signal is always active and after system reset drive 0 is selected. The drive select signal is routed through the daisy-chain cabling and activates the selected Disc Drive Unit. If the selected drive is turned on, the cartridge door locked, the heads loaded, and if the initial time delay has expired, the selected drive returns an active drive ready (DRRDY-) and the sector and index pulses are gated to the controller. The sector and index pulses are used by the controller after the read/write head is positioned over the selected cylinder. Drive ready is a status signal.

Normally the CPU will request a status report after selecting a Disc Drive Unit. A status request is processed like a command; (1) COXX, the controller address, and the request are issued by the CPU; (2) the connect latch is set and the request is stored in the command register; (3) DOXX is issued by the CPU; (4) the request is decoded by the command register. Either status request generates device status (DEVSTAT-) which sets the status enable (STATEN+) flip-flop on the DMA board. STATEN+ causes the program counter to jump to the status routine in the DMA logic and causes the D multiplexer (MUX) in the disc control logic to transfer a status word, D0+(0) through D7+(0), to the DA MUX in the DMA logic. When the status of the selected Disc Drive Unit is "drive ready", bits D1+(0) and D2+(0) are high and the state of the other bits is not significant. The program ROM sets the memory address register to address 58 and sets the DMA control flip-flops to generate DMA write (DMAW-) and the strobe (DMAS-) that gates the data word into memory.

2.7 SELECT CYLINDER (figure 2-2)

Nine bits are required to identify all 400 cylinders available on a disc so two bytes must be transferred to address a track. The CPU issues a cylinder select high byte command concurrent with control out then transfers the most significant cylinder address bit concurrent with data out. The controller decodes CYLHI- which stores the most significant bit in the cylinder address register then waits for the next CPU command. The next CPU transfer will be a cylinder select low byte command concurrent with control out followed by the eight least significant bits concurrent with data out. The controller decodes CYLLO- which stores the low order cylinder address byte in the cylinder address register and triggers the seek one-shot. Seek strobes the cylinder address into the Disc Drive Unit. The controller (and CPU) must then wait for the Disc Drive Unit to return a seek complete or a seek incomplete status. Typically, seek complete status is returned in 35 milliseconds, however, seek incomplete status will not be returned for 200 milliseconds. The CPU will issue a status request and if neither seek complete nor seek incomplete are active the CPU will have to reissue the status request. If seek incomplete becomes active, the CPU will have to issue a cylinder select low command and the low order byte again. The sector select sequence follows a seek complete status.

2.8 SELECT MODE, DISC, HEAD, SECTOR, AND FORMAT (figure 2-2)

The CPU issues one of four mode commands to the controller concurrent with COXX. Then, except for Keyresult mode, the CPU transfers disc select, head select, format select, and the five-bit sector number to the controller concurrent with DOXX. DOXX- generates STRB+ which completes the input to the command decoder and the commanded mode is stored in the mode register. Any mode selected and seek complete generates enable DMA which is sent to the DMA logic.

2.8.1 SECTOR ADDRESS REGISTER LOADING

DRW+ is generated when the selected mode is read, write, or keysearch. DRW+ clocks the disc selection, head selection, format selection, and sector number into the sector address register. Disc select and head select are transmitted to the Disc Drive Unit. Format may not be selected for keysearch mode but is optional for read and write modes. During a formatted read mode the preamble and postamble bytes are transferred to the CPU. During non-formatted write mode the preamble and postamble bytes are generated by the controller. Conditions in the controller have now been established so when the sector immediately preceding the appropriate sector (sector number minus one) passes under the selected head, reading can begin to verify satisfactory operation.

2.8.2 SECTOR NUMBER IDENTIFICATION (figure 2-2)

There are two identical sector counters on the disc control board; one for the fixed disc and one for the removable disc. Each counter consists of an enable latch, an alternate sector flip-flop, and a six-bit counter. The index pulse resets the enable latch, the counter to all zeros, and the alternate sector flip-flop so the output being used is high. Since there are 48 sector pulses per disc revolution and the CPU divides a cylinder into 24 sectors, the least significant bit of the counter is not used. The alternate sector flip-flop signal, ASCTF- or ASCTR-, is set high by the index, low by each odd numbered sector pulse, then high again by each even numbered sector pulse. The outputs from the fixed disc sector counter, FSADD0+ through FSADD4+, and the outputs from the removable disc sector counter, RSADD0+ through RSADD4+, are inputs to the selected disc MUX. ASCTF- gates every other fixed sector pulse, SPLSF+, into the selected disc MUX and ASCTR- gates every other removable sector pulse into the MUX.

At the beginning of each transfer operation when the CPU selects the Disc Drive Unit, the DRSEL- decoded from the command resets the sector synchronizing latches. The first sector pulse from the disc sets the latch. The outputs from the synchronizing latches, FSNC+ and RSNC+, are also inputs to the selected disc MUX.

After the CPU has selected a disc, the sector related signals for the disc are gated through the MUX by DSCSEL-. The sector count (SADD0+ through SADD4+) are compared with the selected sector number (SSADD0+ through SSADD4+) and are also routed to the DMA logic. The selected sector number is always the sector before the one where data transfers are to occur. Therefore, the output of the comparator is sector minus one (SM1+). When SM1+ goes high, the read circuits become operative so this sector can be read as an operational check before the sector where the reading or writing is to occur becomes available to the selected read/write head.

2.8.3 DMA START AND END ADDRESS LOADING

When the mode command is loaded in the mode register the ENDMA+ sets the DMA routine flip-flop which results in the DMARUT+ and ROUTEN+ signals becoming active. ROUTEN+ enables the program counter and DMARUT+ enables the program memory. The program counter advances at the CPU clock rate which successively steps through the counts to enable the program to load the start address into the A register and the end address into the B register. The program loads 60 into the memory address register and issues a request to read, RQSTR+, which sets the read request latch. Memory busy (MBUSY-) is active when something is being written into or read from the CPU memory. As soon as MBUSY- becomes inactive, DMAR- and DMAS- are sent to the CPU. As soon as the high byte of the CPU memory start address is transferred to the memory bus, MBSY- goes active again. While the request latch or the DMA strobe latches are set, the program counter is disabled by PAUSE-. Three CPU clock pulses after MBSY- goes active (after transfer of the start address high byte) the program counter is enabled again. The next program step loads the start address high byte into the memory read register (G REGISTER). The byte is transferred through the DA MUX and into the A register. The next two program steps transfer the start address low byte from memory address 61 to the A register. Then the next four program steps load the high byte of the end address from memory addresses 62 and 63 to the B register. The start address is then moved from the A register to page 0 of the scratch pad memory.

2.9 READ CHECK OF SECTOR MINUS ONE (figure 2-2)

The sector pulse from the selected disc MUX is applied to the sector synchronizing circuit where the synchronized sector pulse, SYNSPULSE, signals are generated. The SYNSPULSE signals are active for one CPU clock period and occur from the second to the third CPH2 after the trailing edge of each selected disc sector pulse. SYPLS- is routed to the DMA logic. SYPLS+ and SM1+ generate the synchronized sector minus one pulse, SYNM1-, which is also routed to the DMA logic. SYNM1- causes a program memory branch which advances the program counter to the next step causing the program ROM to issue DRDT+, R6+, and R7+. These signals set the operate flip-flop (OPLCH+ active) and the enable read flip-flop. The enable read signal, ENRD+, enables the read delay counter. The read delay counter is clocked by the 2.5 megahertz clock. When the read delay count reaches 8 it latches itself at that count and generates the read gate (RDGT-). RDGT- enables the read clock and read data signals from the Disc Drive Unit (figure 2-4), the first one-bit contained in the read data sets the sync bit flip-flop (SYNBT- low) which loads 28 into the preamble counter and enables the bit counter. Each read clock thereafter advances the bit count and each read data bit is loaded into the read byte register. When the bit count reaches 8 the bit counter is reset, the preamble counter is advanced, and the byte ready signal is active and byte read clock (BRDCLK-) is generated.

BRDCLK- transfers the contents of the read byte register to the F register in the DMA logic. The byte is then available to the CPU memory. BRDCLK- resets the bit counter. The bit counter is advanced each count clock until the count of 8 when BRDCLK- is generated, the bit counter is reset, and the cycle repeats until the preamble counter advances to a count of 32. The next time the bit count is 8 the data field counter is advanced. The data field counter advances as each byte is read from the disc until it reaches the count 255. The next byte ready signal from the bit counter advances the postamble counter. The postamble counter advances to 4 which generates a reset read signal which resets the enable read flip-flop, the sync bit flip-flop, the read delay counter and the read byte register. The program counter then waits until the next sector pulse which signals the beginning of the first sector indicated by the start address.

2.10 DATA TRANSFER BUS (figure 2-2)

The DA MUX is the center of data transfers in the DMA logic. Data to be transferred from the CPU memory to the disc for storage enters through the G register and is transferred through the DA MUX and loaded into the serial data out register one byte at a time. Status from the Disc Drive Unit and the disc control logic as well as the cylinder address and sector address are transferred through the D MUX on the disc control logic board through the DA MUX and into the F multiplexers in the DMA logic. The status bytes are loaded on command through the F MUX, into the F Register and written directly into the CPU memory. The address bytes are loaded into the preamble check comparator. The contents of the scratch pad memory are transferred through the DA MUX to the A or B register. During keyresult operation the best key is transferred from the KBO MUX through the DA MUX to the F register.

2.11 ERROR CHECKING

Although the information read from sector-minus-one is processed through the controller right up to the F register, the contents of this sector are not transferred to the CPU memory. As preamble words P0 and P1 are read from the disc they are compared to the P0 and P1 words created in the D MUX. Any time there is not a bit-for-bit comparison, an error latch is set. At the end of the postamble and error latch is set if the read CRC polynomial generator does not have a zero output. These error conditions may be transmitted to the CPU in a status report, however, any error condition detected in sector minus one read inhibits write operation in the next sector.

If no error is detected in the sector-minus-one read, the commanded operation will be started when the next sector pulse is received.

2.12 CONTROLLER READY

The CPU initiates a data (read/write) or information (Keysearch/Keyresult) transfer operation by selecting a Disc Drive Unit (or in the case of a single drive system, by issuing a new cylinder address). The DMA/Disc Drive Unit Controller stores the cylinder address, sector number, disc select, head select, and mode selection commanded by the CPU. The controller also stores the CPU memory start address and end address which defines the number of bytes to be written on the disc, read from the disc, or contained in the reference key used in a keysearch operation. Then as the sector preceding the start of the commanded operation passes under the selected read/write head, it is read to check for errors as an operational check of the controller and the Disc Drive Unit. When no errors are detected, the commanded operation is executed. When errors are detected, the write operation is inhibited, however, the read and keysearch operations are executed. The CPU program may have contingency routines when errors are detected during read or keysearch.

2.13 WRITE LOOP

The write loop refers to the microprocessor ROM program which controls the logic on the DMA Logic board and the logic on both boards that is involved in transferring data from the CPU memory to the Disc Drive Unit. The program ROM outputs are divided into three types of controls: operational commands, selector counts, and addresses. The selector counts control the arithmetic unit operations and the branch multiplexer selection. The addresses are branch addresses for the program counter and location addresses for the scratch pad memory. The transfer may be a formatted or a non-formatted write. The difference is not in the information transferred to the Disc Drive Unit, but in the source of the preamble and postamble information. During a formatted write, this information is supplied from the CPU memory. During a non-formatted write, the information is generated in the DMA/Disc Drive Unit Controller.

2.13.1 SECTOR FORMAT (figure 2-3)

The format for each sector on a disc is the same. Each sector is divided into three parts; the preamble, the data field, and the postamble. The preamble consists of 28 bytes in which a zero has been recorded for every bit (224 bits) followed by a sync byte which is seven zeroes and a one in the most significant bit. The sync byte is followed by two address bytes, P0 and P1. P0 is the eight least significant bits of the cylinder address. P1 consists of the five-bit sector address, the head (top or bottom), the disc (fixed or removable), and the most significant bit of the cylinder address. The last byte of the preamble is another byte of all zeroes. The preamble is followed by a 256 byte data field. Following the data field is the postamble. The postamble consists of two CRC bytes, a byte of zero bits, and eight bytes of erased (no flux change) track.

2.13.2 WRITE DATA PATH (figure 2-3)

During write mode there are two possible data paths, one for the preamble and postamble and the other for data from the CPU memory. In the case of a formatted write, everything is transferred as data from the CPU memory. During a non-formatted write, the bytes for the preamble originate in the D multiplexer on the disc control logic board. These bytes are transferred through the DA multiplexer to the disc write register. Bits are shifted from the disc write register as SERDA+ to the double frequency gates at a 2.5 megahertz rate. The output of the double frequency write gates is a 100-nanosecond pulse for each zero in SERDA+ and two 100-nanosecond pulses for each one in SERDA+. Data from the CPU memory is called for by the controller one byte at a time beginning at the start address transferred from the CPU memory. Each data byte from the CPU memory is stored in the G register until the previous byte is transmitted to the Disc Drive Unit. The byte is then transferred through the DA multiplexer to the disc write register. Data bits from the disc write register are entered into the CRC polynomial generator and the double frequency gates. At the end of each data field the CRC polynomial generator contains 16 bits which, when added to the bits in the data field result in a number which may be divided by 16 without a remainder. These sixteen bits are transferred to the double frequency write gates to become the CRC bytes of the postamble. The input to the double frequency write gates for the next eight bit times result in the zero byte. The double frequency write gates are then disabled for eight bytes to erase a block before the start of the next sector.

2.14 PREAMBLE WRITE (figure 2-3)

The write operation commences when the sector pulse marking the start of the selected sector reaches the disc control logic. The sector pulse is synchronized to the CPU clock by the disc control logic to become SYPLS+ and SYPLS-.
From this instant until the end of the postamble, both the disc control logic and the DMA logic perform different simultaneous functions to process data from the CPU memory to the Disc Drive Unit. Periodically the DMA logic must wait for the completion of an operation by the disc control logic before continuing. After the postamble, the circuits on both boards wait until the next sector pulse. In the

disc control logic the sector pulse resets the postamble counter then SYPLS+ sets the enable write flip-flop which enables the bit counter and gates the count clock which advances the bit counter at the 2.5-megahertz rate. If no error was detected during the sector-minus-one read operation, the write gate is sent to the Disc Drive Unit. If an error was detected, the controller processes continue but nothing is written on the disc. Since formatted write operations are a special case which is seldom done, the following discussion is for a non-formatted write.

The first section of the preamble word consists of 28 bytes of zero bits. The D multiplexer is disabled so its outputs are used to supply zeroes through the DA multiplexer to the disc write register. The bit counter counts off eight bit time intervals then BYTRY+ is generated. Each BYTRY+ advances the preamble counter. When the preamble counter reaches the count of 27, the output of the D multiplexer is changed so it consists of seven zeroes (D0 through D6) and a one from D7. When this byte is transferred through the disc write register, the one is the last bit shifted out. The preamble counter is then advanced to 28 and the preamble ROM issues a low P0-. This selects the CADD0+ through CADD7+ to be transferred through the D multiplexer. When the preamble counter advances to 29, the preamble ROM issues P1- which selects the SADD0+ through SADD4+, HDSEL+, DISCSEL+, and CADD8+ inputs to be transferred through the D multiplexer. This is the same address/selection information that located the sector for the write operation and so it uniquely describes this sector. The drive select is not pertinent since the data must be retrievable from the removable disc regardless of which Disc Drive Unit or System it may be used in. When the preamble counter advances to 30, the outputs from the D multiplexer are again zeroes so the last byte (the thirty-second) of the preamble is another zero byte.

SYPLS- advances the program counter to 15 in the DMA logic. The program ROM issues a request to read (RQSTR+) command which sets the request to read flip-flop. The request to read flip-flop disables the program counter (PAUSE-) so it will not advance until the first data byte is transferred from the CPU memory. The request to read may occur when the CPU memory is busy. When this happens, the DMA logic must wait until MBSY- goes high. As soon as MBSY- is high the DMA strobe flip-flop is set. DMAR- and DMAS- are sent to the CPU which causes the first data byte to be put on the memory bus. (The CPU memory start address was put in the memory address register before the program ROM issued the wait-for-sector-pulse command.) Once the data byte is put on the memory bus, MBSY- goes high enabling the clock pulse counter.

The two-count output resets the DMA strobe flip-flop so the program counter becomes enabled again and the third clock pulse advances the counter. The byte on the memory bus is loaded into the G register and the program counter waits until the disc control logic issues a data byte ready (DBRDY-).

2.15 DATA WRITE LOOP (figure 2-3)

Once the data field for the sector is reached, the write operation becomes a program loop which is repeated until the end of the data field or until the CPU memory end address is reached. The program loop steps are started by DBRDY- for each byte. DBRDY- advances the program counter to 17 and transfers the memory data byte from the G register through the DA multiplexer into the disc write register. The disc write register is a parallel-to-serial shift register so it takes 8 X 400 nanoseconds or 3.2 microseconds to transfer a byte through the register. The bits from the disc write register are entered into the CRC polynomial generator and gates one or two 2.5 MHz pulses through the double frequency gates. The 2.5 MHz pulses are as wide as CPU clock pulses so when the data bit is a one, the double frequency write signal looks like two CPU clock pulses. In contrast, when the data bit is zero, only one pulse is transmitted so a string of zeroes looks like the CPU clock signal where every other pulse is blanked. The CRC polynomial adds each one-bit in the write data. At the end of the data field it generates a two-byte polynomial which, when added to the total number of ones in the data field, results in a number that is evenly divisible by 16.

As soon as DBRDY- advances the program counter to 17, the program ROM issues a request-to-write (RQSTW+) which results in no action during the write mode. The program ROM is advanced by the next CPU clock to 18. The contents of the A register (CPU memory start address) is compared to the contents of the B register (CPU memory end address). Of course, the start address should not be the same as the end address, but after the first byte has been processed, the A register contains the current CPU memory address so eventually the contents of the A and B registers will be equal. When the number in the A register equals the number in the B register, all of the data has been written for this operation so the program counter is branched to 63 and the program ROM issues HALT+.

When A does not equal B, the number in the A register is increased by one to become the next CPU memory address and placed in scratch pad location 0. The next CPU memory address is then transferred through the DA multiplexer to the A register and to the memory address multiplexer and register. The program counter is then reset to 15 and the porgram ROM issues another request to read. The program has now completed the write loop. The program continues looping until the current CPU memory address equals the CPU memory end address. Each time around the loop the following sequence occurs.

1. Transfer the next data byte from the CPU memory to the G register.
2. Wait for DBRDY-.
3. Transfer the data byte through the DA multiplexer to the disc write register.
4. Compare A with B.
5. Halt if A = B.
6. Add one to the current memory address (A) and store the result (next memory address) in the scratch pad.
7. Return the next memory address to the A register and the memory address register.

Excluding the time spent waiting for DBRDY-, the minimum time required to progress through the program is two microseconds. Since it takes 3.2 microseconds to write a byte on a disc, the program will always have to wait approximately one microsecond for DBRDY-.

It may happen that more than 256 bytes of data are to be written during a single transfer. In this case, the program counter will stay at count 16 during the time the postamble for the present sector and the preamble for the next sector are written because DBRDY- is not generated then.

2.16 POSTAMBLE WRITE (figure 2-3)

After the data counter counts to 256, the postamble counter is enabled. During postamble counts 0 and 1, the polynomial from the CRC generator replaces SERDA+ as the input to the double frequency write gates. During postamble count 3 a byte of all zeroes is encoded by the double frequency write gates. Then, during the next eight counts of the postamble counter, ERASE- from the postamble ROM disables the double frequency write gates so an 8-byte erase gap is created at the end of the sector. A postamble count of 12 disables the counter so all three counters (preamble, data, postamble) remain at their maximum count until they are reset by the next SPULSE-.

2.17 FORMATTED WRITE

Some circumstances make it necessary to use the formatted write option. In this case, all information written in a sector is transferred from the CPU memory and DBRDY- is generated for each byte in the sector up through the CRC bytes in the postamble. Otherwise, the operation is the same as for a non-formatted write.

2.18 BYTE COUNTER STATES

Table 2-1 lists the count for the preamble, data, and postamble counters for each byte written in a sector. In this table Data Byte means the byte written on the disc is transferred from the CPU memory. Note that after each counter reaches its maximum count it remains there until the start of the next sector.

Table 2-1. Write Operation Byte Counter States

COUNTER STATE			TRANSFER TYPE		INFORMATION SOURCE	
PRE-AMBLE COUNT	DATA COUNT	POST-AMBLE COUNT	WRITE	FORMAT WRITE	WRITE	FORMAT WRITE
0 ↓ 26	0 ↓ 0	0 ↓ 0	Write all-zero sync bytes	Write all-zero sync bytes	D MUX	D MUX
27	0	0	Write sync byte	Write synce byte	D MUX	D MUX
28	1	0	Write P0 byte	Write data byte	D MUX	CPU
29	1	0	Write P1 byte	Write data byte	D MUX	CPU
30	1	0	Write zero byte	Write data byte	D MUX	CPU
31	1	0	Write data byte	Write data byte	CPU	CPU
32 ↓ 32	1 1 2 ↓ 255	0 ↓ 0	Write data bytes	Write data bytes	CPU	CPU
32	255	0	Write CRC byte	Write data byte	CRC Gen.	CPU
32	255	1	Write CRC byte	Write data byte	CRC Gen.	CPU
32	255	2	Write zero byte	Write zero byte	D MUX	D MUX
32 ↓ 32	255 ↓ 255	3 ↓ 11	Create erase gap	Create erase gap	No change in write signal	No change in write signal
32	255	12	Reset write logic	Reset write logic		

2.19 READ LOOP

The initial steps to get the controller ready for a read operation are identical to the ones required for a write operation so at the start of a read operation the heads are positioned, sector minus one has been read and the information checked for errors, the CPU memory start address is stored in the A register and the end address in the B register, and the program counter is at count 14 waiting for the sector pulse. The program sequence is the same for a read operation as for a write operation; the program counter cycles from 15 through 22 and back to 15. The same counters keep track of the bit/byte position within the sector. However, what happens, the meaning of the counter states, and the direction of information flow are different in the two operations.

2.19.1 INFORMATION FLOW

Information, starting with the P0 byte is read from the disc. The P0 and P1 bytes are compared with the contents of the D multiplexer for each case and the detection of an error is stored in the P0 and/or P1 error latches. Data bits are read from the disc and accumulated in the read data register until a byte is assembled. DBRDY- is generated transferring the byte through the F2 multiplexer into F register. A request to write in the CPU memory is issued and, when the CPU memory is not busy, the byte is transferred into a memory location. The data bits are also counted by a CRC polynomial generator. If there are no errors in the data read from the disc, the CRC bytes from the disc make the CRC count correct for a zero remainder after division by 16. When a CRC error is detected, the CRC error latch is set. The read operation is not inhibited by the detection of an error (P0, P1, or CRC) but the fact that an error was detected is available as a status report.

2.19.2 FORMATTED READ

During a formatted read operation, the P0, P1, and CRC bytes are transferred to the CPU memory as data bytes. There is generally no reason to use a formatted read in an operational program.

2.20 PREAMBLE READ (figure 2-4)

The sector pulse marking the beginning of the selected sector resets the preamble, data, and postamble counters and is synchronized to the CPU clock to become SYPLS+ and SYPLS-. SYPLS- advances the program counter to 15 and the program ROM issues a request to read, RQSTR+. The request to read does not result in a CPU memory read because RQSTR+ is inhibited by enable read, ENRD-, being active.

The next CPU clock advances the program counter to 16 and waits for DBRDY- to indicate that a data byte has been assembled in the read data register.

SYPLS+ sets the enable read flip-flop which enables the ready delay counter. The delay counter is advanced at the byte-time rate until it reaches the count of eight (approximately a 26 microsecond delay). At that time, the counter is inhibited from counting and the read gate (RDGT+) is generated. RDGT+ presets the preamble counter to 28. RDGT- is sent to the Disc Drive Unit where it enables the read circuits. READCLOCK- and READDATA- are returned to the controller. The read clock (RDCLK+) generates CNTCLK+ as long as the read gate is enabled. The last bit (read bit 7) of the sync byte is a one so RZDAT+ sets the sync bit latch. SYNBT- removes the load signal from the preamble counter and causes the ENRD+ENWRT signal to go high which removes the reset signal from the bit counter. The bit counter advances at the count clock rate (a nominal 2.5 megahertz) until it reaches the count of eight at which time it resets itself to one and generates BYTRY+ which enables the preamble counter to advance with the next count clock. BYTRY+ AND RDCLK- generate BRDCLK- which transfers the contents of the read byte register through the F2 multiplexer and into the F register. The first byte assembled in the read byte register is the P0 byte.

When the preamble counter is set to 28, the preamble ROM issues P0- which presets the data counter to 1. The data counter is disabled from counting until the preamble counter counts to 32 so the data counter still will not reach 255 before data byte 256 is available from the data byte register. When the P0 byte is assembled in the data byte register, it is transferred through the F2 multiplexer to the F register. At the same time, the P0 byte assembled in the D multiplexer is routed through the DA multiplexer to the comparator. The outputs of the F register are always applied to the comparators. When P0 from the D multiplexer is different than P0 from the read byte register while the preamble ROM is issuing P0-, the P0 error latch is set. During the next preamble count the preamble counter issues P1- and the P1 bytes from the F register and the D multiplexer are compared. If there is a difference, the P1 error latch is set. The zero byte is read but ignored except during a formatted read operation.

2.21 DATA READ (figure 2-4)

Anytime DATA+ is active a data byte ready, DBRDY-, signal is generated when the bit counter count is 8. DATA+ is active during a formatted read operation and during a normal read operation from preamble count 31 through data count 255.

The byte ready signal (count 8 of the bit counter) also generates BRDCLK-. BRDCLK- and ENRD- transfer the data byte from the read byte generator through the F2 multiplexer and into the F register. Each time DBRDY- is active, the program counter advances from count 16 to count 17 and the program ROM issues a RQSTRW+. RQSTRW sets the write request and the memory data in flip-flops. Write request generates RQST- which causes program counter PAUSE- and, as soon as the CPU memory is not busy, sets the DMA strobe flip-flop (DMAS-) and generates DMA write (DMAW-). PAUSE- keeps the program counter from advancing. DMAW- and DMAS- are sent to the CPU where they cause the byte on the memory bus to be entered in the memory location indicated by the memory address bus. DMASF+ enables the DMA strobe counter which is advanced by the CPU clock. The DMA strobe counter count of two resets the request to write flip-flop and the other flip-flops are reset by the next CPU clock. PAUSE- is inactive so the program counter is advanced by the next CPU clock. The program ROM outputs R27+, R28+, R29-, R30-, and R31+ are all high so, if the contents of the A register (CPU memory start/present address) equal the contents of the B register (CPU memory end address) the end of block flip-flop is set and the program counter is advanced to 63. The output of the program ROM for 63 is HALT+. When A is not equal to B the program counter is advanced to count 20 and the program ROM issues the scratch pad enable (SPEN-) and write in scratch pad (WRTSP+) signals while holding outputs R27+ through R31+ low. This causes the arithmetic logic unit to add one to the number from the A register and the result to be written in scratch pad location 0. The next CPU clock advances the program counter to 21 and the program ROM issues SPEN-, LD+, LALB-, and LAHB+. SPEN- (and address 0 determined by R0+ through R3+) transfer the next CPU memory address from scratch pad location 0 to the DA multiplexer. LALB- selects the SP00+ through SP07+ and SPEN- selects the SP08+ through SP15+ inputs of the DA multiplexer. LD+ clocks DA0+ through DA15+ into the CPU memory address register. LALH+ and LALB+ combine with DCPH2- to clock DA0+ through DA15+ into the A register. The next memory address is now stored in both the A register and the CPU memory address register. The next CPU clock advances the program counter to 23 to 22 and the program ROM issues a branch program counter to 15. Now the program ROM issues a request to read which does not result in any action since this is a read operation and no input from the CPU memory is required. The next CPU clock advances the program counter to 16 and the program ROM issues BTEST+ which causes the counter to be stopped until the next DBRDY- indicates a complete byte has been stored in the read byte register.

Each time DBRDY- is active, the data byte counter is advanced until it reaches the count of 255. When the 256th byte is stored in the read byte register causing a DBRDY-, the data byte counter is disabled keeping it at the full count and the postamble counter is advanced. The 256th data byte is the last byte transferred to CPU memory unless this is a formatted read because DBRDY- is not generated again until the read byte register contains the first data byte from the next sector. Therefore, the program counter will remain at count 16 for approximately 40 byte times (130 microseconds) unless the last data byte of the sector corresponds with the end of the block to be transferred. (A = B or this address equals end address).

Each of the 256 data bytes in the selected sector, then, are read from the disc. The disc drive unit develops a read clock and a read data bit for each bit cell and transfers these signals to the controller. The read clock advances the bit counter and shifts the data bits into the read byte register and the read CRC polynomial generator. The bit counter counts to 8 at which time the read byte register contains an assembled read byte. The bit 8 count causes the byte to be transferred through the F2 multiplexer to the F register to await transfer to the CPU memory. The bit 8 count also activates the program counter which sequentially steps the program ROM through the process of transferring the byte to its CPU memory location; checking memory address to determine if this is the last byte to be read and, if it is not, incrementing the present memory address to the next location; and then placing the next address in the CPU memory address register and storing it in the A register. Each bit 8 count also increments the data byte counter so the disc control logic has a current record of the portion of the sector being read.

2.22 POSTAMBLE READ (figure 2-4)

The first two bytes of the postamble are the CRC polynomial. All of the bits in the data field were supplied to the CRC polynomial generator and each one-bit was counted. Each time the count equals the CRC divisor, the counter is reset so, at the end of the data field, the contents of the counter is the remainder from dividing the number of ones in the data field by the CRC divisor. When the number of one-bits in the CRC bytes is added to this remainder, the counter will be reset. Therefore, when the read data is correct, the remainder in the polynomial generator and its output are zero.

Except during a formatted read, the postamble bytes are not transferred to the CPU so the only result of reading the postamble is recording a CRC error as a status bit.

When the postamble counter advances to 4, reset read (RSTRD-) is generated. RSTRD- resets the enable read flip-flop which resets the read delay counter and removes the read gate from the Disc Drive Unit, resets the sync bit flip-flop which resets and disables the bit counter, and resets the read byte register so all outputs are zero. The disc control logic then waits for the next sector pulse. If all the bytes for the read operation have not been read (present address is less than the end address) the read-and-store operation will continue when the next DBRDY- is generated.

2.23 KEYSEARCH MODE

A keysearch is initiated in order to locate a record in a direct file. Direct file records are identified by a key. A key is usually some part of the data stored in each record such as a part number or a serial number. When a direct file is established, the length of key is defined in the direct statement. A key may be up to 49 characters long and the key as stored in the disc index is up to 55 bytes long. When a record in a direct file is to be located, the keylength, number of records (keys) in the file, and disc location of the file index is assembled by the user's program. A keysearch is then initiated and the program provides the controller with the following parameters for the keysearch.

1. The beginning sector of the file index (sector address).
2. The number of keys in the index (N).
3. The number of keys per sector (NS).
4. The key length or size (S).
5. The next higher key (argument key) than the one being sought.

The argument key is used for comparison with each key as it is read and so is designated as KC. KC is stored in the KC first-in-first-out (FIFO) memory. The rest of the parameters are stored in the scratch pad memory. As the keysearch is conducted, the number of keys searched in this sector (KS), the total number of keys searched (K), and the number of the best key found so far are also stored in the scratch pad memory. The best key found so far is stored in one of the best key (KB) first-in-first-out memories.

When the argument key is stored in the KC FIFO, the KB0 FIFO key is set to all zeroes. The first key read is compared to KC and 0 since the best key has to be less than KC and greater than zero. The first key read is stored in the KB1 FIFO. If the first key read was less than KC it becomes the trial best key. As soon as a key is found that becomes a trial best key, the best flag, (BFLAG+) flip-flop is toggled, its key number is stored in the scratch pad memory, and the next key is stored in the KBO FIFO. When a keysearch does not locate a key less than KC, no best key is found and the KBO FIFO is still all zeroes. As soon as one key is found that is less than KC and greater than zero it becomes the trial best key and subsequent keys are compared to KC and it. Each time a new trial best key is found, the range between it and KC is narrowed. A keysearch is continued until the number of keys defined by N have been searched even though the best key is the first key read.

2.24 KEYSEARCH PARAMETER TRANSFER (figure 2-5)

The start and end addresses stored in the A and B registers during the controller set-up sequence define the CPU memory locations containing the keysearch parameters.

The program counter is jumped from count 10 to count 23 when the present operation is a keysearch. The program ROM issues SPEN- and WRTSP- which transfers the end address from the B register to scratch pad location 1. The next clock pulse advances the program counter to 24 and the program ROM issues SPEN- and WRTSP- which writes zeroes into scratch pad location 2 (key size). The next CPU clock advances the program counter to 25 and the program ROM issues a BTEST+. If this were a keyresult mode the program counter would be advanced to 46. The next CPU clock advances the program counter to 26 and the program ROM issues RQSTR+, LD+, SPEN-, LAHB-, and LALB-. This transfers the next memory address through the DA multiplexer to the CPU memory address register and to the A register and causes DMAR- and DMAS- to be sent to the CPU. The program counter is stopped for two clock periods after the contents of that memory location are transferred to the G register. This byte is the high byte of the numbers of keys to be searched and is stored in the B register at the next program step. The next clock pulse advances the program counter to 28 and the program ROM issues SPEN- and WRTSP-. This transfers the present memory address plus one (the next memory address) from the arithmetic unit to scratch pad location 0.

The next clock pulse advances the program counter to 29 and the program ROM causes the next memory address to be stored in the CPU memory address register and the A register. The next step of the program counter causes the low byte of the number of keys to be searched (N) to be transferred from memory to the B register. The next two program steps transfer N to scratch pad location 3 and set the B register to zero. The next three program steps fetch the number of keys (NS) per sector from memory, store it in the B register, and then transfer NS in scratch pad location 6.

The controller now has obtained and stored all the parameters necessary to load the argument key in the KC FIFO memory. Program step 36 transfers the keysize (S) from scratch pad location 2 to the B register. S is initially zero and is incremented as each byte is stored. When KC is complete, S is returned to scratch pad location 2 for use during the actual search operation. As each byte of the argument key is loaded in the KC FIFO memory it ripples toward the output. It helps to visualize the FIFO memory as a vertical stack of memory cells with the input at the top and the output at the bottom. As a byte is entered at the input it falls from memory cell to memory cell until it reaches the lowest unoccupied cell or the output cell. Since key size is limited to 49 bytes, the FIFO memory is never full when a complete key is stored.

Program step 37 transfers the next CPU memory address from the A register to scratch pad location 0. The memory address is then transferred back to the A register and the memory address register during program step 38. Program step 39 is a no operation step since the next program step is to increment the B register and so the B register would be incremented each CPU clock during PAUSE-. Program step 40 issues WRTKEY+ to enter the KC byte into the KC FIFO and INCB+ to increment the B register thus keeping the keysize count correct. Program step 41 transfers keysize from the B register to scratch pad location 2 so program step 42 can transfer the CPU memory end address to the B register. Program step 43 sets the A = B latch if this was the last memory location containing KC bytes. If this is not the last KC byte, the program branches back to step 36 so the next byte can be transferred. If this is the last KC byte, the program branches to step 63 and issues a halt.

A halt causes the routine circuits to switch from the DMA routine to the keysearch routine. This enables the keysearch program ROM and disables the DMA program ROM. When the program counter resets to 0, the keysearch program sets the best key number in scratch pad location 7 to zero. Keysearch program step 1 sets the number of keys searched in scratch pad location 4 to zero and keysearch program step 2 stops the program counter until sector minus one is announced by SYNM1-. Program step 3 sets the operation latch and the enable read latch. A read check of sector-minus-one is performed as in the case of a read or write operation. The keysearch continues whether there is an error detected by the read check but the CPU operational program may have a status report routine to act on the keysearch operation result in a special way when an error has been detected. Program step 4 sets the number of keys searched in scratch pad location 8 to zero and step 5 stops the program counter until SYPLS- is received indicating the start of the first sector to be searched.

The controller now has established initial conditions for the keysearch to begin. KC is stored in the KC FIFO memory and table 2-1 lists the initial parameters/ results stored in the scratch pad.

Table 2-2. Scratch Pad Initial Conditions for Keysearch

Scratch Pad Location	Parameter/Result	Symbol
0	Last CPU memory address *	SA
1	End CPU memory address *	EA
2	Key size (number of bytes per key)	S
3	Number of keys to be searched	N
4	Number of keys searched = 0	K
6	Number of keys per sector	NS
7	Best key number = 0	BKN
8	Number of keys searched this sector = 0	KS

2.25 KEY COMPARISON (figure 2-5)

As each KC byte is stored a zero byte is stored in the KB0 FIFO so when the argument key is in the KC FIFO an equal number of zeroes is stored in the KB0 FIFO. As each byte is transferred out for the comparison process, it is reentered at the input of the the FIFO (recirculated). This way each key is restored for the next comparison even though it may not be used. In the case of the KB0 and KB1 memories, one KB memory is active supplying KB bytes for comparison with KD bytes from the disc and recirculating the bytes used in the comparison while the KD bytes are being stored in the other KB memory. After a byte is identified as a trial best key the two KB memories change roles for the next comparison.

Keys from the disc are read and processed the same way as other read data. Read clock and read data are received from the Disc Drive Unit by line buffers. Since the RDGT+ is active, RDCLK+ becomes CNTCLK-. CNTCLK- advances the bit counter and shifts RZDAT+ bits into the read byte register. Each time the bit count is 8 after the preamble has been read, a disc key byte is contained in the read byte register and both DBRDY- and BRDCLK- are active. BRDCLK- and ENRD- enters RBIT0+ through RBIT7+ from the F2 multiplexer into the F register.

Each time DBRDY- is generated the data counter in the control logic is advanced, the program counter is advanced to 8, and the keysearch program ROM issues RDKEY+ and KEYCMP+. RDKEY+ shifts a KC byte into the comparator and recirculates the byte to the input of the KC multiplexer. It also shifts a KB0 byte through the DA multiplexer, into the comparator and back to the input of the KB0 multiplexer. KEYCMP+ gates the CPU clock to the comparison latches. There are four (two pair) comparison latches. Except when KD and KC or KD and KB bytes are equal, two of the latches will be set by the first byte of each comparison. Once one latch of a pair is set, the other is disabled so successive byte comparisons for a key cannot result in the ambiguity of two decisions being made for a single key. Since the bytes of a key are (by definition) sequential with the first byte read being the most significant byte, a key will be greater than or less than the comparison byte as soon as the case occurs where both bytes are not equal. Since the first KD key is compared with zero from KB0, the comparison should yield a KD>KB input to the KD/KB within-bounds latch for some byte in the key. When the KD/KB within-bounds latch is set, the KD/KB out-of-bounds latch is disabled. Unless the first key read is equal to the argument key, either the KD/KC within-bounds latch or the KD/KC out-of-bounds latch will be set by the end of the comparison. When both within-bounds latches are set KEYMID- is active and the key has been identified as a trial best key. Keysearch program step 9 causes each KD byte to be stored in either the KB0 or the KB1 FIFO. The first key is stored in the KB1 FIFO. When a trial best key is identified, the next key read is stored in the KB0 FIFO.

After each byte is compared and stored the number of bytes compared (B register) is compared with the keysize (A register). When A = B, the whole key has been compared with KC and KB and the program counter is advanced to 12. Until A = B, the program counter is reset from 11 to 8 and the byte-for-byte comparison continues.

Program step 12 transfers N from scratch pad location 0 to the A register. Program step 13 transfers K from scratch pad location 4 to the B register. Program step 14 increments K in the B register to include the key just compared. Program step 15 transfers the new value of K back to scratch pad location 4. Program step 16 checks BSTKY- to determine if this key was a trial best key. If so K is stored in scratch pad location 7 as the best key number (BKN). If this key is not a trial best key, the program counter is branched to 18 and the comparison latches are reset. A and B are compared and if they are equal, the program counter is advanced to 31 and a HALT+ is issued by the keysearch program ROM.

Program step 20 transfers NS from scratch pad location 6 to the A register and program step 21 transfers KS from scratch pad location 8 to the B register. Program step 22 increments the B register so K includes the key just compared in the number of keys searched this sector. Program step 23 restores KS to scratch pad location 8 and program step 24 compares A with B. When A does not equal B the program counter is reset to 6 and waits for the next DBRDY-. When A = B the program counter is reset to 4, KS is set to 0, and the program waits for the next SYPLS-.

2.26 KEY RESULT MODE

After a keysearch is completed, the CPU issues a key result command. The controller responds by returning the best key number from the scratch pad and the best key from the KB FIFO memory where it was stored.

The key result mode does not require any information to be exchanged between the controller and the Disc Drive Unit so the disc control logic is only used to decode the IKRLT- (input key result) command and store KRLTMD+ in the mode register (figure 2-2) KRSLTMD+ generates the ENDMA+ signal which set the DMA routine flip-flop which results in an active ROUTEN+. ROUTEN+ starts program counter operation and the first nine program steps load the CPU memory start and end address as with the other modes of operation. The program ROM then branches the program counter to 23. Program step 23 transfers the end address from the B register to scratch pad location 1 and step 24 sets scratch pad location 2 to zero. Program step 25 branches the program counter to step 46 and the key result transfer begins with the significant information stored in the following locations.

1. The CPU memory start address is stored in scratch pad location 0.
2. The CPU memory end address in stored in scratch pad location 1.
3. The keysize in scratch pad location 2 is set to zero.
4. The best key number is stored in scratch pad location 7.
5. The best key is stored in either the KBO FIFI memory or the KB1 FIFO memory and its location is indicated by the state of BFLAG+.

2.27 KEY RESULT TRANSFER (figure 2-6)

Program step 46 transfers the starting CPU memory address from scratch pad 0 through the DA multiplexer to the CPU memory address register and to the A register. Step

47 issues a request to write (RQSTW+) and transfers the low byte of the best key number through DA multiplexer bits DA0+ - DA07+, the F1 and F2 multiplexers and to the F register. RQSTW+ causes the controller to send DMAR-, DMAW-, and DMAS- to the CPU. After the byte has been transferred to the CPU memory, program step 48 transfers the CPU memory start address through the arithmetic unit where it is incremented by one to become the next CPU memory address and stored in scratch pad location 0. Step 49 transfers the next memory location back to the A register and to the CPU memory address register. Step 50 issues a RQSTW+ and transfers the high byte of the best key number through DA8+ - DA15+, the F1 and F2 multiplexers, and to the F register.

After the high byte of the best key number has been entered in the CPU memory the present CPU memory address is transferred through the arithmetic to be incremented to the next CPU memory address and stored in scratch pad location 0. The next address is then transferred back to the A register and to the CPU memory address register. Program step 53 transfers key size (starts at 0) from scratch pad location 2 to the B register. Step 54 transfers the available byte from the KB FIFO memory selected by BFLAG+ through the DA multiplexer, the F1 and F2 multiplexers, and into the F register. Step 54 also issues a request to write and step 55 is a no-op so the B register will not be incremented during PAUSE-. Program step 56 increments the key size (number of best key bytes transferred). Step 57 returns key size to scratch pad location 2 and step 58 transfers the CPU memory end address from scratch pad location 1 to the B register. The contents of the A and B registers are compared and, if they are equal, the program halts. When A does not equal B, the program is reset to count 51 and the next byte of the best key is transferred. As in the other modes of operation, when the program halts, an interrupt is sent to the CPU so another operation may be commenced.

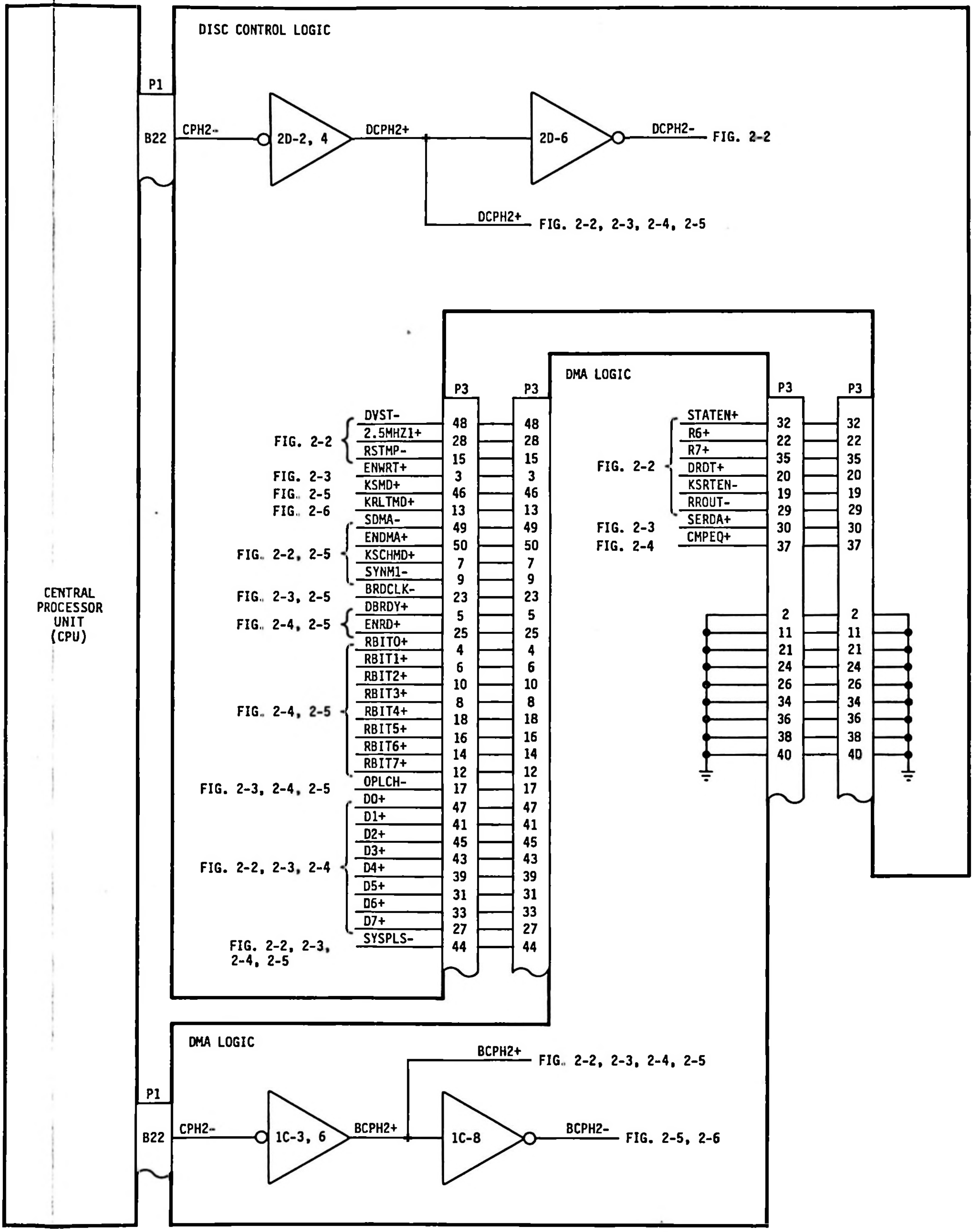
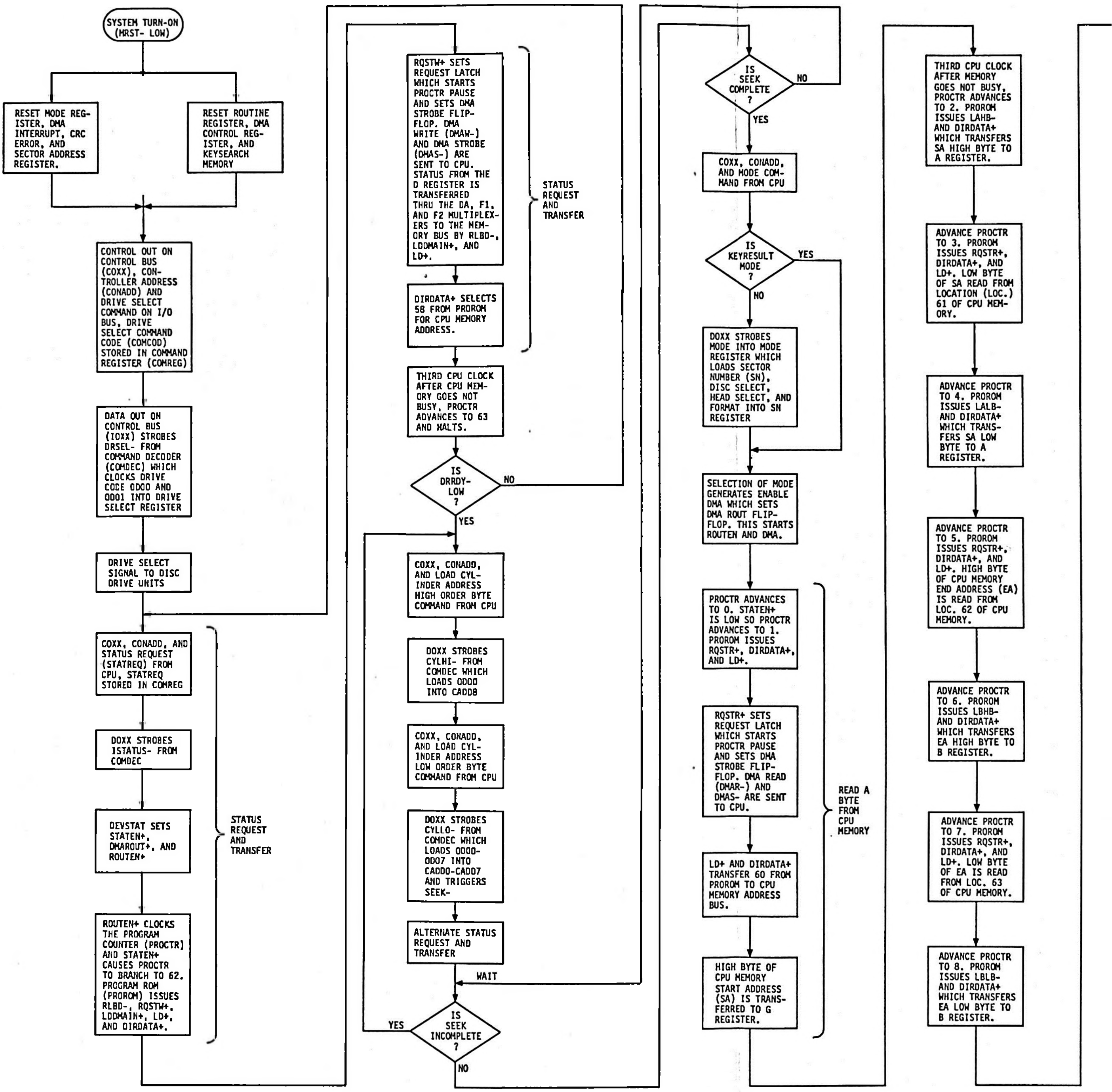


Figure 2-1. Signal Distribution Diagram


SYSTEM TURN-ON (MRST- LOW)
RESET MODE REGISTER, DMA INTERRUPT, CRC ERROR, AND SECTOR ADDRESS REGISTER.
RESET ROUTINE REGISTER, DMA CONTROL REGISTER, AND KEYSEARCH MEMORY
CONTROL OUT ON CONTROL BUS (COXX), CONTROLLER ADDRESS (CONADD) AND DRIVE SELECT COMMAND ON I/O BUS, DRIVE SELECT COMMAND CODE (COMCOD) STORED IN COMMAND REGISTER (COMREG)
DATA OUT ON CONTROL BUS (IOXX) STROBES DRSEL- FROM COMMAND DECODER (COMDEC) WHICH CLOCKS DRIVE CODE OD00 AND OD01 INTO DRIVE SELECT REGISTER
DRIVE SELECT SIGNAL TO DISC DRIVE UNITS
COXX, CONADD, AND STATUS REQUEST (STATREQ) FROM CPU, STATREQ STORED IN COMREG
DOXX STROBES ISTATUS- FROM COMDEC
DEVSTAT SETS STATEN+, DMAROUT+, AND ROUTEN+
ROUTEN+ CLOCKS THE PROGRAM COUNTER (PROCTR) AND STATEN+ CAUSES PROCTR TO BRANCH TO 62. PROGRAM ROM (PROROM) ISSUES RLBD-, RQSTW+, LDDMAIN+, LD+, AND DIRDATA+.
STATUS REQUEST AND TRANSFER
RQSTW+ SETS REQUEST LATCH WHICH STARTS PROCTR PAUSE AND SETS DMA STROBE FLIP-FLOP. DMA WRITE (DMAW-) AND DMA STROBE (DMAS-) ARE SENT TO CPU. STATUS FROM THE D REGISTER IS TRANSFERRED THRU THE DA, F1, AND F2 MULTIPLEXERS TO THE MEMORY BUS BY RLBD-, LDDMAIN+, AND LD+.
DIRDATA+ SELECTS 58 FROM PROROM FOR CPU MEMORY ADDRESS.
STATUS REQUEST AND TRANSFER
THIRD CPU CLOCK AFTER CPU MEMORY GOES NOT BUSY, PROCTR ADVANCES TO 63 AND HALTS.
IS DRRDY- LOW ?
NO
YES
COXX, CONADD, AND LOAD CYLINDER ADDRESS HIGH ORDER BYTE COMMAND FROM CPU
DOXX STROBES CYLHI- FROM COMDEC WHICH LOADS OD00 INTO CADD8
COXX, CONADD, AND LOAD CYLINDER ADDRESS LOW ORDER BYTE COMMAND FROM CPU
DOXX STROBES CYLLO- FROM COMDEC WHICH LOADS OD00-OD07 INTO CADD0-CADD7 AND TRIGGERS SEEK-
ALTERNATE STATUS REQUEST AND TRANSFER
WAIT
IS SEEK INCOMPLETE ?
YES
NO
IS SEEK COMPLETE ?
NO
YES
COXX, CONADD, AND MODE COMMAND FROM CPU
IS KEYRESULT MODE ?
YES
NO
DOXX STROBES MODE INTO MODE REGISTER WHICH LOADS SECTOR NUMBER (SN), DISC SELECT, HEAD SELECT, AND FORMAT INTO SN REGISTER
SELECTION OF MODE GENERATES ENABLE DMA WHICH SETS DMA ROUT FLIP-FLOP. THIS STARTS ROUTEN AND DMA.
PROCTR ADVANCES TO 0. STATEN+ IS LOW SO PROCTR ADVANCES TO 1. PROROM ISSUES RQSTR+, DIRDATA+, AND LD+.
RQSTR+ SETS REQUEST LATCH WHICH STARTS PROCTR PAUSE AND SETS DMA STROBE FLIP-FLOP. DMA READ (DMAR-) AND DMAS- ARE SENT TO CPU.
LD+ AND DIRDATA+ TRANSFER 60 FROM PROROM TO CPU MEMORY ADDRESS BUS.
HIGH BYTE OF CPU MEMORY START ADDRESS (SA) IS TRANSFERRED TO G REGISTER.
READ A BYTE FROM CPU MEMORY
THIRD CPU CLOCK AFTER MEMORY GOES NOT BUSY, PROCTR ADVANCES TO 2. PROROM ISSUES LAHB- AND DIRDATA+ WHICH TRANSFERS SA HIGH BYTE TO A REGISTER.
ADVANCE PROCTR TO 3. PROROM ISSUES RQSTR+, DIRDATA+, AND LD+. LOW BYTE OF SA READ FROM LOCATION (LOC.) 61 OF CPU MEMORY.
ADVANCE PROCTR TO 4. PROROM ISSUES LALB- AND DIRDATA+ WHICH TRANSFERS SA LOW BYTE TO A REGISTER.
ADVANCE PROCTR TO 5. PROROM ISSUES RQSTR+, DIRDATA+, AND LD+. HIGH BYTE OF CPU MEMORY END ADDRESS (EA) IS READ FROM LOC. 62 OF CPU MEMORY.
ADVANCE PROCTR TO 6. PROROM ISSUES LBHB- AND DIRDATA+ WHICH TRANSFERS EA HIGH BYTE TO B REGISTER.
ADVANCE PROCTR TO 7. PROROM ISSUES RQSTR+, DIRDATA+, AND LD+. LOW BYTE OF EA IS READ FROM LOC. 63 OF CPU MEMORY.
ADVANCE PROCTR TO 8. PROROM ISSUES LBLB- AND DIRDATA+ WHICH TRANSFERS EA LOW BYTE TO B REGISTER.

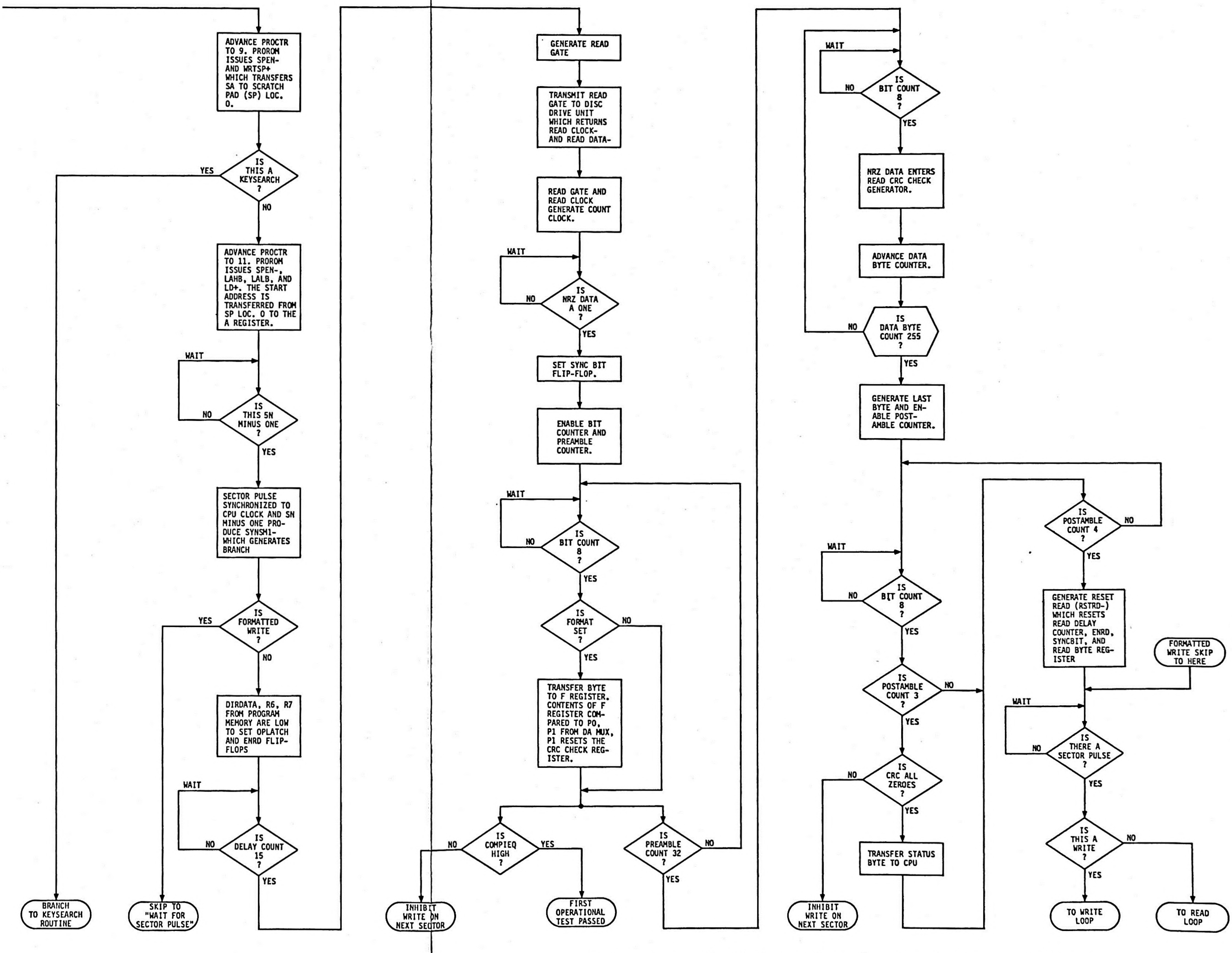

ADVANCE PROCTR TO 9. PROROM ISSUES SPEN- AND WRTSP+ WHICH TRANSFERS SA TO SCRATCH PAD (SP) LOC. 0.
IS THIS A KEYSEARCH ?
YES
NO
ADVANCE PROCTR TO 11. PROROM ISSUES SPEN-, LAHB, LALB, AND LD+. THE START ADDRESS IS TRANSFERRED FROM SP LOC. 0 TO THE A REGISTER.
WAIT
IS THIS SN MINUS ONE ?
NO
YES
SECTOR PULSE SYNCHRONIZED TO CPU CLOCK AND SN MINUS ONE PRODUCE SYNSM1- WHICH GENERATES BRANCH
IS FORMATTED WRITE ?
YES
NO
DIRDATA, R6, R7 FROM PROGRAM MEMORY ARE LOW TO SET OPLATCH AND ENRD FLIP-FLOPS
WAIT
IS DELAY COUNT 15 ?
NO
YES
BRANCH TO KEYSEARCH ROUTINE
SKIP TO "WAIT FOR SECTOR PULSE"
GENERATE READ GATE
TRANSMIT READ GATE TO DISC DRIVE UNIT WHICH RETURNS READ CLOCK- AND READ DATA-
READ GATE AND READ CLOCK GENERATE COUNT CLOCK.
WAIT
IS NRZ DATA A ONE ?
NO
YES
SET SYNC BIT FLIP-FLOP.
ENABLE BIT COUNTER AND PREAMBLE COUNTER.
WAIT
IS BIT COUNT 8 ?
NO
YES
IS FORMAT SET ?
NO
YES
TRANSFER BYTE TO F REGISTER. CONTENTS OF F REGISTER COMPARED TO PO, P1 FROM DA MUX, P1 RESETS THE CRC CHECK REGISTER.
IS COMPIEQ HIGH ?
NO
YES
IS PREAMBLE COUNT 32 ?
NO
YES
INHIBIT WRITE ON NEXT SECTOR
FIRST OPERATIONAL TEST PASSED
WAIT
IS BIT COUNT 8 ?
NO
YES
NRZ DATA ENTERS READ CRC CHECK GENERATOR.
ADVANCE DATA BYTE COUNTER.
IS DATA BYTE COUNT 255 ?
NO
YES
GENERATE LAST BYTE AND ENABLE POSTAMBLE COUNTER.
WAIT
IS BIT COUNT 8 ?
NO
YES
IS POSTAMBLE COUNT 3 ?
NO
YES
IS CRC ALL ZEROES ?
NO
YES
TRANSFER STATUS BYTE TO CPU
INHIBIT WRITE ON NEXT SECTOR
IS POSTAMBLE COUNT 4 ?
NO
YES
GENERATE RESET READ (RSTRD-) WHICH RESETS READ DELAY COUNTER, ENRD, SYNCBIT, AND READ BYTE REGISTER
FORMATTED WRITE SKIP TO HERE
WAIT
IS THERE A SECTOR PULSE ?
NO
YES
IS THIS A WRITE ?
NO
YES
TO WRITE LOOP
TO READ LOOP

Figure 2-2. Preliminary Transfer Requirements Functional Diagrams, Sheet 1 of 4

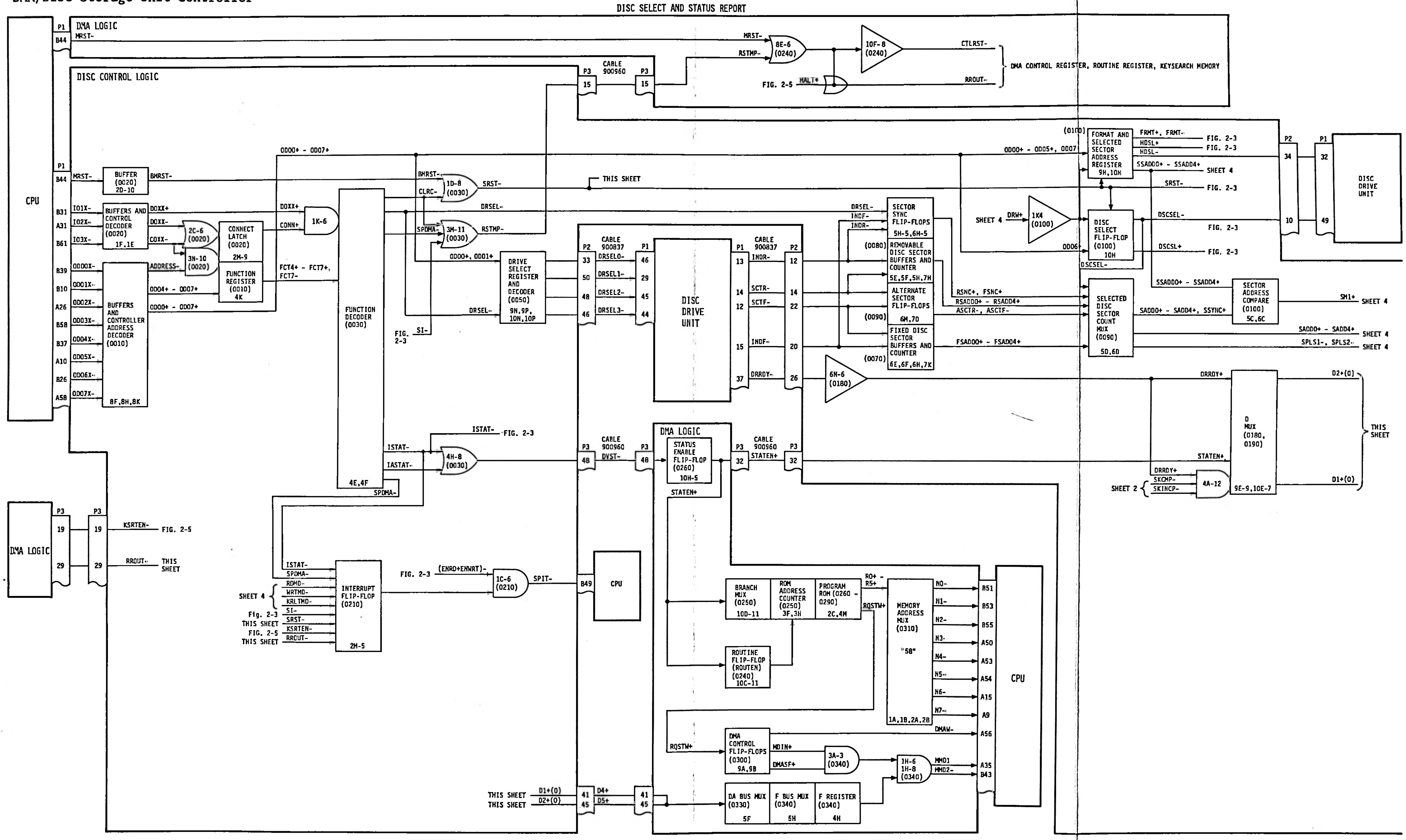


Figure 2-2. Preliminary Transfer Requirements Functional Diagrams, Sheet 2 of 4

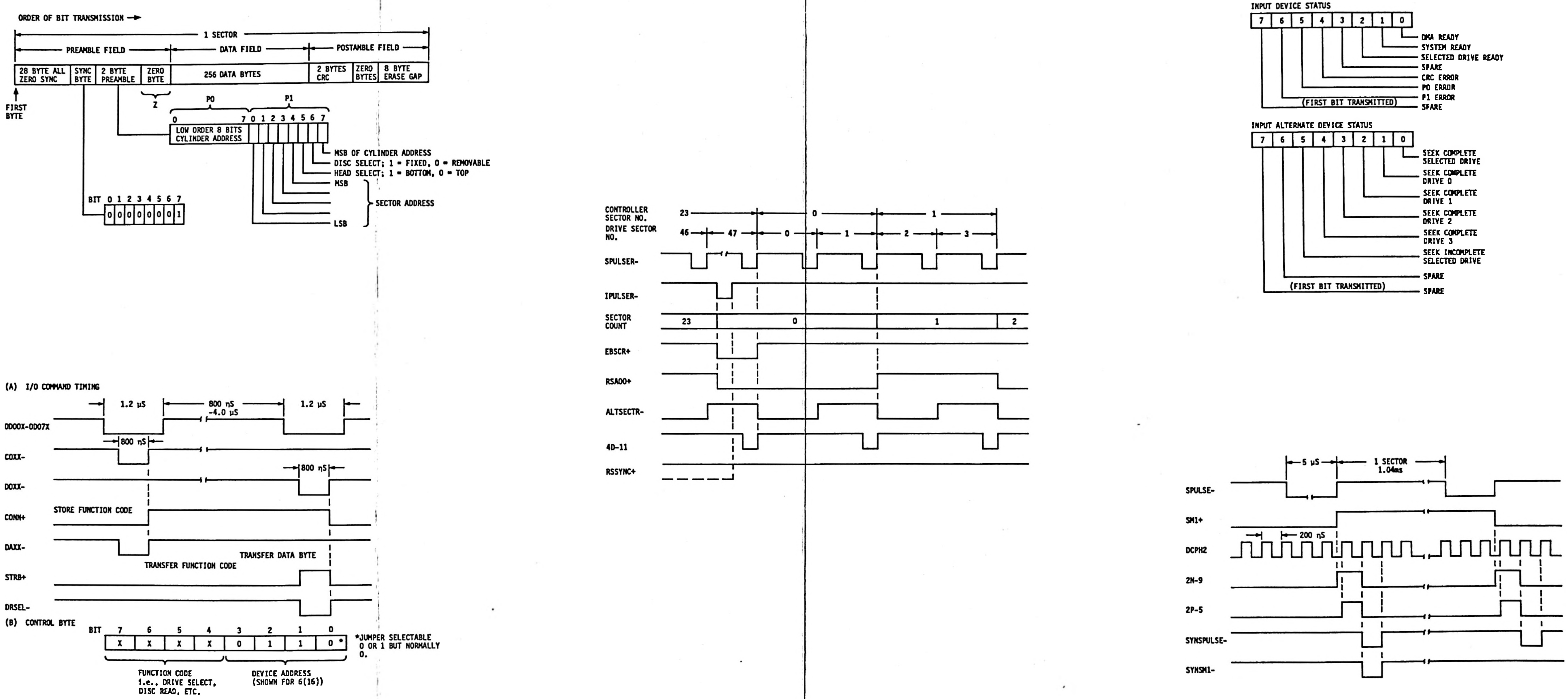


Figure 2-2. Preliminary Transfer Requirements Functional Diagrams, Sheet 3 of 4

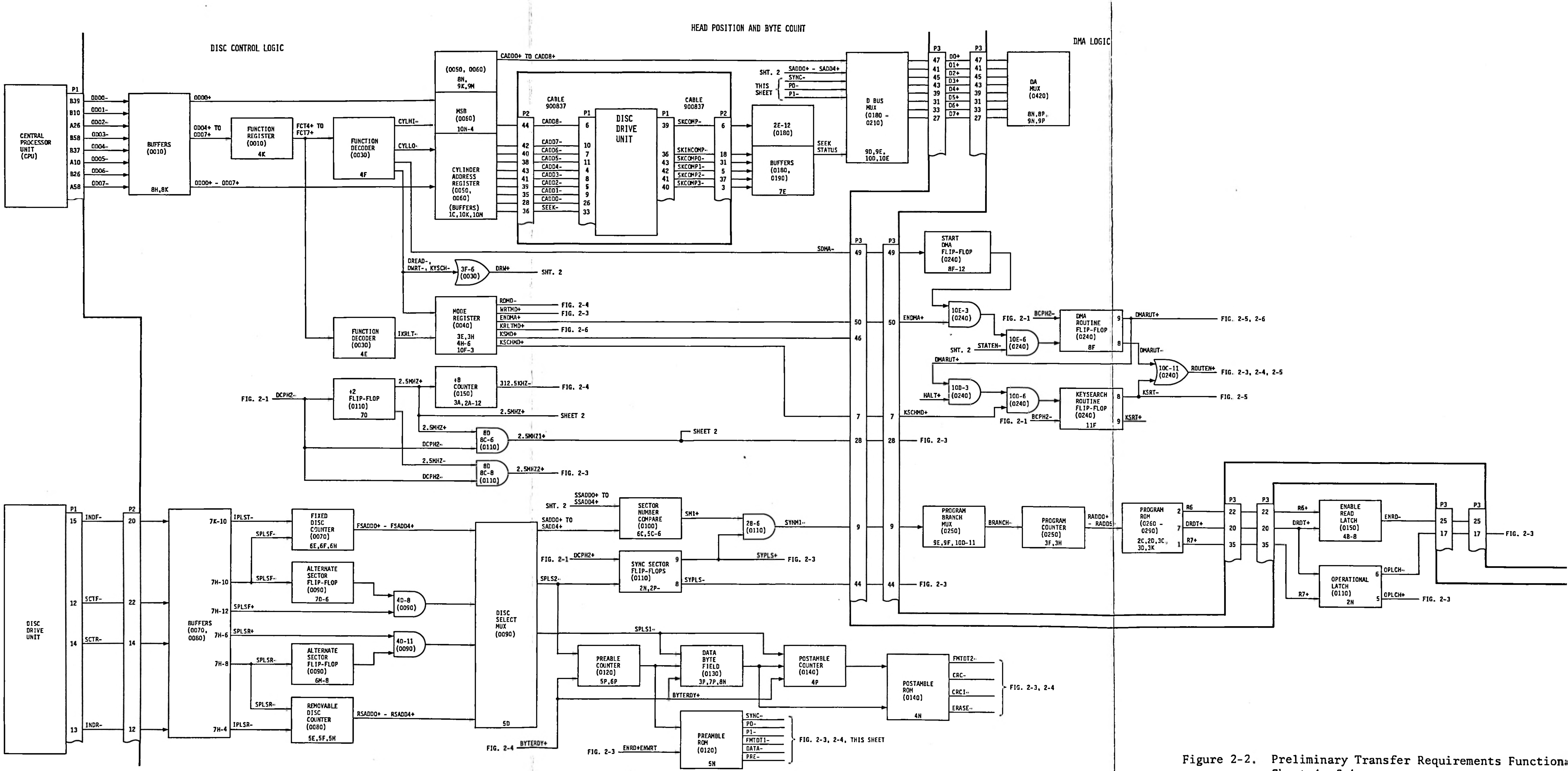


Figure 2-2. Preliminary Transfer Requirements Function
Sheet 4 of 4

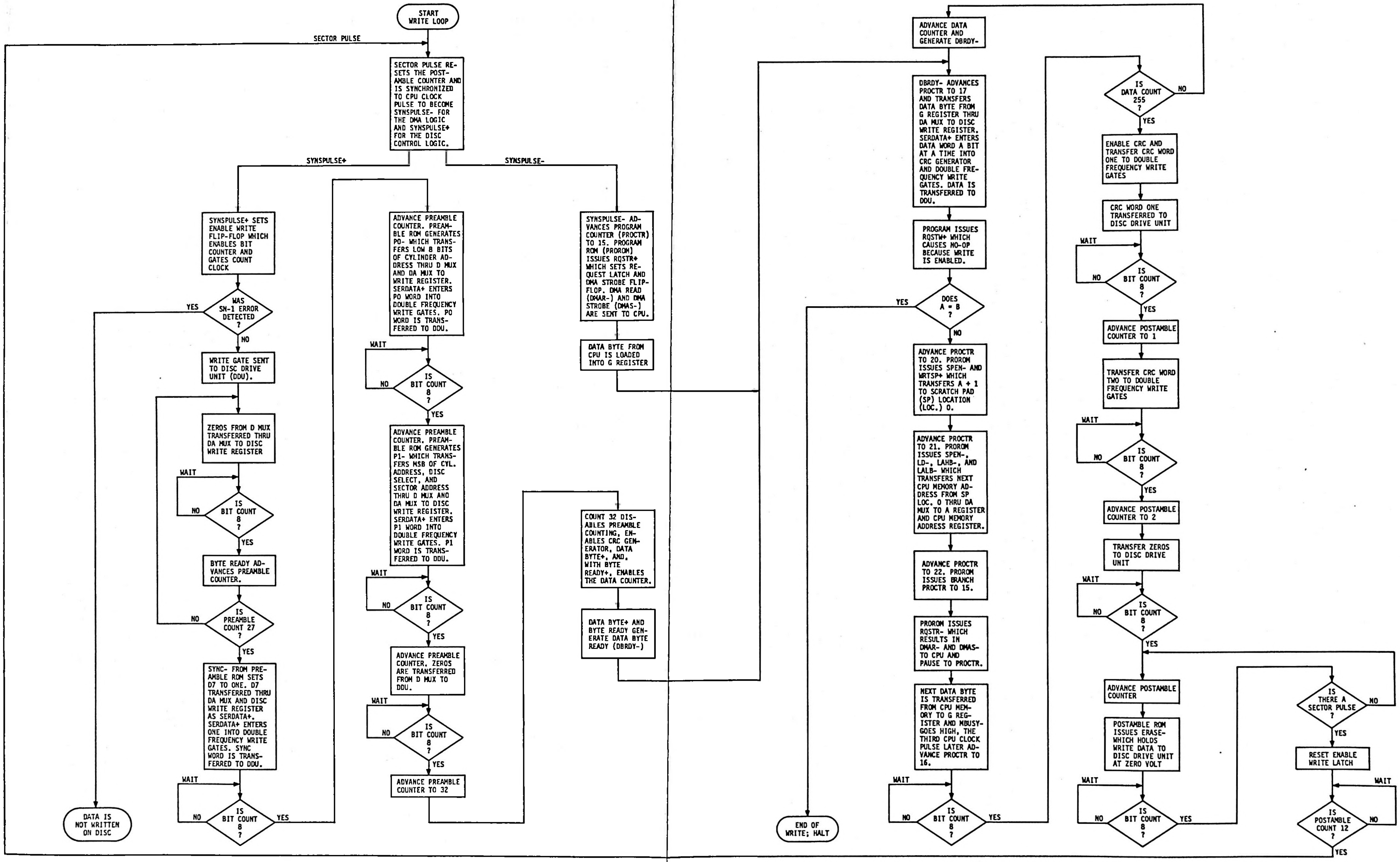
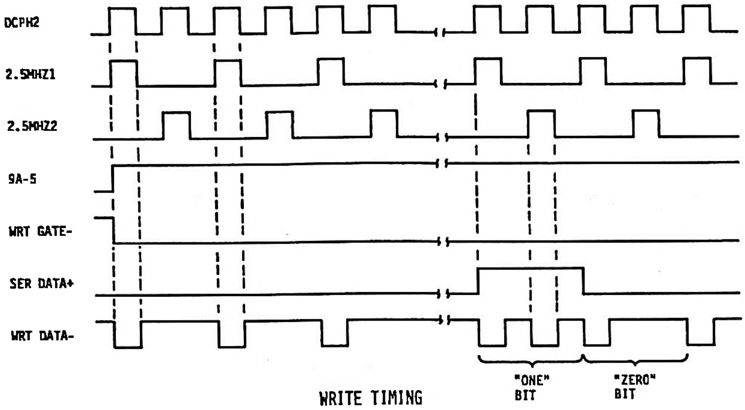
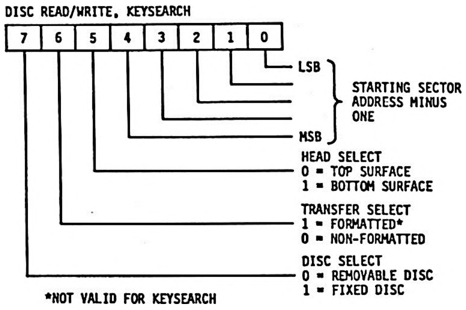

START WRITE LOOP
SECTOR PULSE
SECTOR PULSE RESETS THE POSTAMBLE COUNTER AND IS SYNCHRONIZED TO CPU CLOCK PULSE TO BECOME SYNSPULSE- FOR THE DMA LOGIC AND SYNSPULSE+ FOR THE DISC CONTROL LOGIC.
SYNSPULSE+
SYNSPULSE-
SYNSPULSE+ SETS ENABLE WRITE FLIP-FLOP WHICH ENABLES BIT COUNTER AND GATES COUNT CLOCK
WAS SN-1 ERROR DETECTED ?
YES
NO
DATA IS NOT WRITTEN ON DISC
WRITE GATE SENT TO DISC DRIVE UNIT (DDU).
ZEROS FROM D MUX TRANSFERRED THRU DA MUX TO DISC WRITE REGISTER
WAIT
NO
IS BIT COUNT 8 ?
YES
BYTE READY ADVANCES PREAMBLE COUNTER.
NO
IS PREAMBLE COUNT 27 ?
YES
SYNC- FROM PREAMBLE ROM SETS D7 TO ONE. D7 TRANSFERRED THRU DA MUX AND DISC WRITE REGISTER AS SERDATA+. SERDATA+ ENTERS ONE INTO DOUBLE FREQUENCY WRITE GATES. SYNC WORD IS TRANSFERRED TO DDU.
WAIT
NO
IS BIT COUNT 8 ?
YES
ADVANCE PREAMBLE COUNTER. PREAMBLE ROM GENERATES P0- WHICH TRANSFERS LOW 8 BITS OF CYLINDER ADDRESS THRU D MUX AND DA MUX TO WRITE REGISTER. SERDATA+ ENTERS P0 WORD INTO DOUBLE FREQUENCY WRITE GATES. P0 WORD IS TRANSFERRED TO DDU.
WAIT
NO
IS BIT COUNT 8 ?
YES
ADVANCE PREAMBLE COUNTER. PREAMBLE ROM GENERATES P1- WHICH TRANSFERS MSB OF CYL. ADDRESS, DISC SELECT, AND SECTOR ADDRESS THRU D MUX AND DA MUX TO DISC WRITE REGISTER. SERDATA+ ENTERS P1 WORD INTO DOUBLE FREQUENCY WRITE GATES. P1 WORD IS TRANSFERRED TO DDU.
WAIT
NO
IS BIT COUNT 8 ?
YES
ADVANCE PREAMBLE COUNTER. ZEROS ARE TRANSFERRED FROM D MUX TO DDU.
WAIT
NO
IS BIT COUNT 8 ?
YES
ADVANCE PREAMBLE COUNTER TO 32
SYNSPULSE- ADVANCES PROGRAM COUNTER (PROCTR) TO 15. PROGRAM ROM (PROROM) ISSUES RQSTR+ WHICH SETS REQUEST LATCH AND DMA STROBE FLIP-FLOP. DMA READ (DMAR-) AND DMA STROBE (DMAS-) ARE SENT TO CPU.
DATA BYTE FROM CPU IS LOADED INTO G REGISTER
COUNT 32 DISABLES PREAMBLE COUNTING, ENABLES CRC GENERATOR, DATA BYTE+, AND, WITH BYTE READY+, ENABLES THE DATA COUNTER.
DATA BYTE+ AND BYTE READY GENERATE DATA BYTE READY (DBRDY-)
ADVANCE DATA COUNTER AND GENERATE DBRDY-
DBRDY- ADVANCES PROCTR TO 17 AND TRANSFERS DATA BYTE FROM G REGISTER THRU DA MUX TO DISC WRITE REGISTER. SERDATA+ ENTERS DATA WORD A BIT AT A TIME INTO CRC GENERATOR AND DOUBLE FREQUENCY WRITE GATES. DATA IS TRANSFERRED TO DDU.
PROGRAM ISSUES RQSTW+ WHICH CAUSES NO-OP BECAUSE WRITE IS ENABLED.
DOES A = B ?
YES
NO
END OF WRITE; HALT
ADVANCE PROCTR TO 20. PROROM ISSUES SPEN- AND WRTSP+ WHICH TRANSFERS A + 1 TO SCRATCH PAD (SP) LOCATION (LOC.) 0.
ADVANCE PROCTR TO 21. PROROM ISSUES SPEN-, LD-, LAHB-, AND LALB- WHICH TRANSFERS NEXT CPU MEMORY ADDRESS FROM SP LOC. 0 THRU DA MUX TO A REGISTER AND CPU MEMORY ADDRESS REGISTER.
ADVANCE PROCTR TO 22. PROROM ISSUES BRANCH PROCTR TO 15.
PROROM ISSUES RQSTR- WHICH RESULTS IN DMAR- AND DMAS- TO CPU AND PAUSE TO PROCTR.
NEXT DATA BYTE IS TRANSFERRED FROM CPU MEMORY TO G REGISTER AND MBUSY- GOES HIGH, THE THIRD CPU CLOCK PULSE LATER ADVANCE PROCTR TO 16.
WAIT
NO
IS BIT COUNT 8 ?
YES
IS DATA COUNT 255 ?
NO
YES
ENABLE CRC AND TRANSFER CRC WORD ONE TO DOUBLE FREQUENCY WRITE GATES
CRC WORD ONE TRANSFERRED TO DISC DRIVE UNIT
WAIT
NO
IS BIT COUNT 8 ?
YES
ADVANCE POSTAMBLE COUNTER TO 1
TRANSFER CRC WORD TWO TO DOUBLE FREQUENCY WRITE GATES
WAIT
NO
IS BIT COUNT 8 ?
YES
ADVANCE POSTAMBLE COUNTER TO 2
TRANSFER ZEROS TO DISC DRIVE UNIT
WAIT
NO
IS BIT COUNT 8 ?
YES
ADVANCE POSTAMBLE COUNTER
POSTAMBLE ROM ISSUES ERASE- WHICH HOLDS WRITE DATA TO DISC DRIVE UNIT AT ZERO VOLT
WAIT
NO
IS BIT COUNT 8 ?
YES
IS THERE A SECTOR PULSE ?
NO
YES
RESET ENABLE WRITE LATCH
WAIT
IS POSTAMBLE COUNT 12 ?
NO
YES

Figure 2-3. Write Mode Functional Diagrams, Sheet 1 of 2

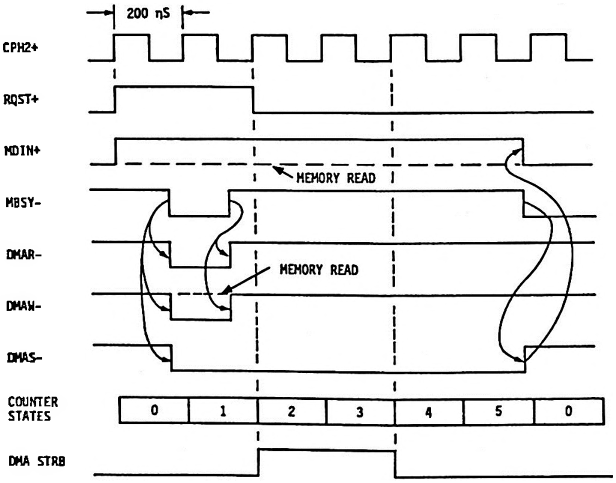


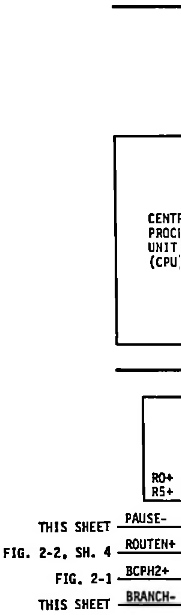
WRITE TIMING



COMMAND FORMAT

MEMORY READ CYCLE TIMING





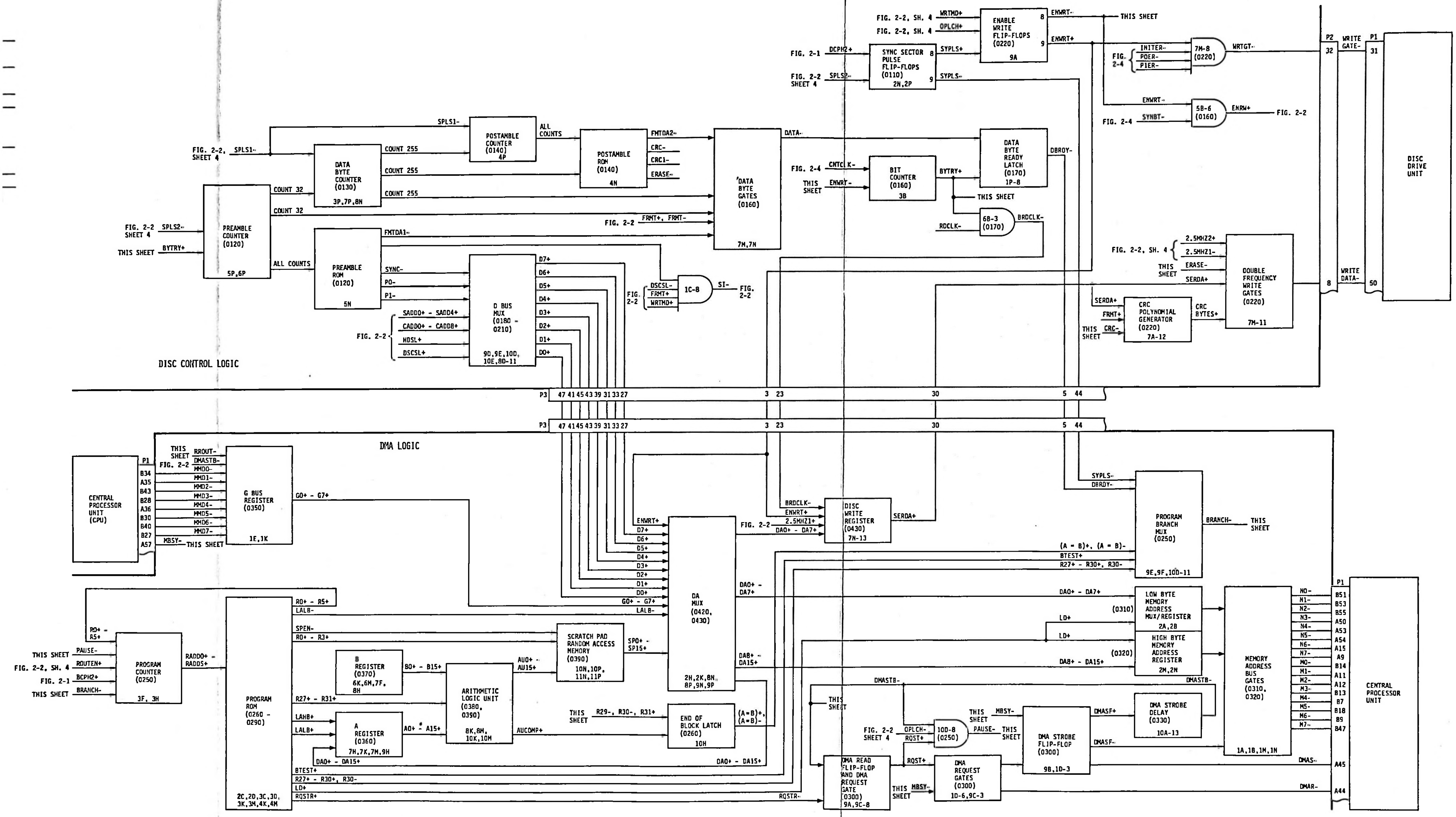


Figure 2-3. Write Mode Functional Diagrams, Sheet 2 of 2

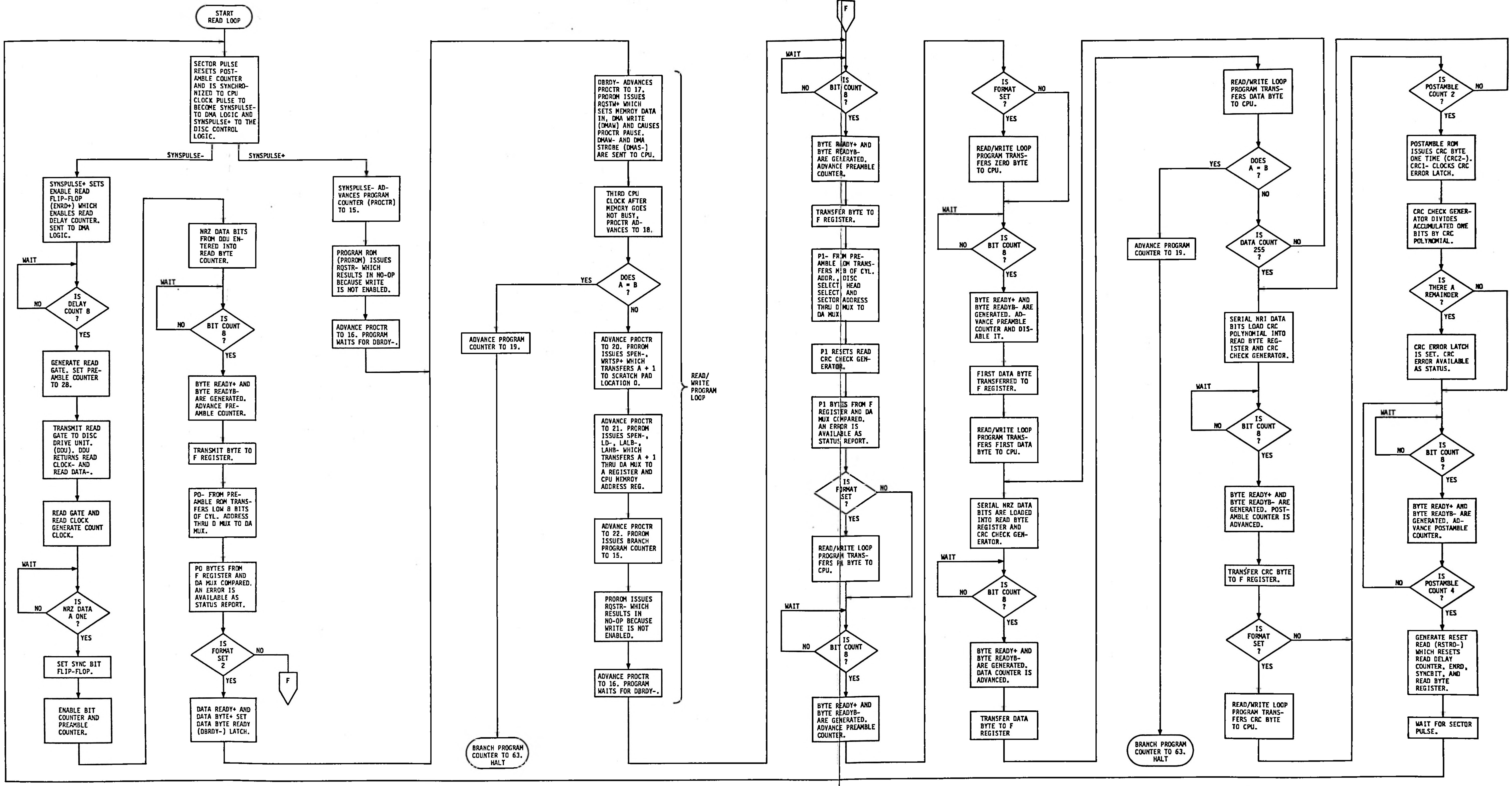
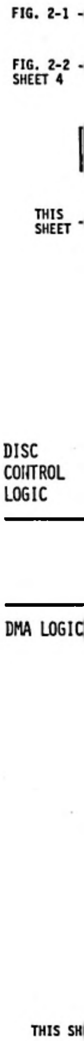
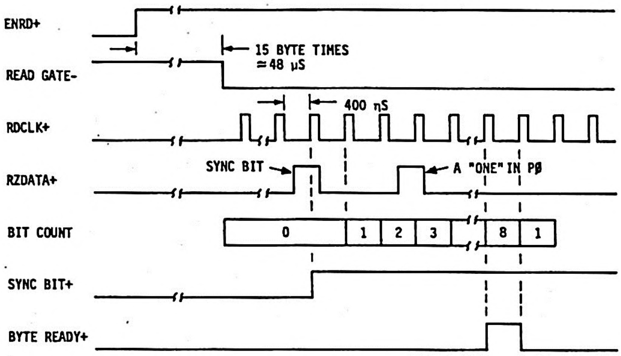

START READ LOOP
SECTOR PULSE RESETS POST-AMBLE COUNTER AND IS SYNCHRONIZED TO CPU CLOCK PULSE TO BECOME SYNSPULSE- TO DMA LOGIC AND SYNSPULSE+ TO THE DISC CONTROL LOGIC.
SYNSPULSE-
SYNSPULSE+
SYNSPULSE+ SETS ENABLE READ FLIP-FLOP (ENRD+) WHICH ENABLES READ DELAY COUNTER. SENT TO DMA LOGIC.
WAIT
NO
IS DELAY COUNT 8 ?
YES
GENERATE READ GATE. SET PREAMBLE COUNTER TO 28.
TRANSMIT READ GATE TO DISC DRIVE UNIT. (DDU). DDU RETURNS READ CLOCK- AND READ DATA-.
READ GATE AND READ CLOCK GENERATE COUNT CLOCK.
WAIT
NO
IS NRZ DATA A ONE ?
YES
SET SYNC BIT FLIP-FLOP.
ENABLE BIT COUNTER AND PREAMBLE COUNTER.
NRZ DATA BITS FROM DDU ENTERED INTO READ BYTE COUNTER.
WAIT
NO
IS BIT COUNT 8 ?
YES
BYTE READY+ AND BYTE READYB- ARE GENERATED. ADVANCE PREAMBLE COUNTER.
TRANSMIT BYTE TO F REGISTER.
PO- FROM PREAMBLE ROM TRANSFERS LOW 8 BITS OF CYL. ADDRESS THRU D MUX TO DA MUX.
PO BYTES FROM F REGISTER AND DA MUX COMPARED. AN ERROR IS AVAILABLE AS STATUS REPORT.
IS FORMAT SET 2
NO
F
YES
DATA READY+ AND DATA BYTE+ SET DATA BYTE READY (DBRDY-) LATCH.
SYNSPULSE- ADVANCES PROGRAM COUNTER (PROCTR) TO 15.
PROGRAM ROM (PROROM) ISSUES RQSTR- WHICH RESULTS IN NO-OP BECAUSE WRITE IS NOT ENABLED.
ADVANCE PROCTR TO 16. PROGRAM WAITS FOR DBRDY-.
DBRDY- ADVANCES PROCTR TO 17. PROROM ISSUES RQSTW+ WHICH SETS MEMROY DATA IN, DMA WRITE (DMAW) AND CAUSES PROCTR PAUSE. DMAW- AND DMA STROBE (DMAS-) ARE SENT TO CPU.
THIRD CPU CLOCK AFTER MEMORY GOES NOT BUSY, PROCTR ADVANCES TO 18.
DOES A = B ?
YES
NO
ADVANCE PROGRAM COUNTER TO 19.
BRANCH PROGRAM COUNTER TO 63. HALT
ADVANCE PROCTR TO 20. PROROM ISSUES SPEN-, WRTSP+ WHICH TRANSFERS A + 1 TO SCRATCH PAD LOCATION 0.
ADVANCE PROCTR TO 21. PROROM ISSUES SPEN-, LD-, LALB-, LAHB- WHICH TRANSFERS A + 1 THRU DA MUX TO A REGISTER AND CPU MEMROY ADDRESS REG.
ADVANCE PROCTR TO 22. PROROM ISSUES BRANCH PROGRAM COUNTER TO 15.
PROROM ISSUES RQSTR- WHICH RESULTS IN NO-OP BECAUSE WRITE IS NOT ENABLED.
ADVANCE PROCTR TO 16. PROGRAM WAITS FOR DBRDY-.
READ/WRITE PROGRAM LOOP
F
WAIT
NO
IS BIT COUNT 8 ?
YES
BYTE READY+ AND BYTE READYB- ARE GENERATED. ADVANCE PREAMBLE COUNTER.
TRANSFER BYTE TO F REGISTER.
P1- FROM PREAMBLE ROM TRANSFERS MSB OF CYL. ADDR., DISC SELECT, HEAD SELECT, AND SECTOR ADDRESS THRU D MUX TO DA MUX.
P1 RESETS READ CRC CHECK GENERATOR.
P1 BYTES FROM F REGISTER AND DA MUX COMPARED. AN ERROR IS AVAILABLE AS STATUS REPORT.
IS FORMAT SET ?
NO
YES
READ/WRITE LOOP PROGRAM TRANSFERS P1 BYTE TO CPU.
WAIT
NO
IS BIT COUNT 8 ?
YES
BYTE READY+ AND BYTE READYB- ARE GENERATED. ADVANCE PREAMBLE COUNTER.
IS FORMAT SET ?
NO
YES
READ/WRITE LOOP PROGRAM TRANSFERS ZERO BYTE TO CPU.
WAIT
NO
IS BIT COUNT 8 ?
YES
BYTE READY+ AND BYTE READYB- ARE GENERATED. ADVANCE PREAMBLE COUNTER AND DISABLE IT.
FIRST DATA BYTE TRANSFERRED TO F REGISTER.
READ/WRITE LOOP PROGRAM TRANSFERS FIRST DATA BYTE TO CPU.
SERIAL NRZ DATA BITS ARE LOADED INTO READ BYTE REGISTER AND CRC CHECK GENERATOR.
WAIT
NO
IS BIT COUNT 8 ?
YES
BYTE READY+ AND BYTE READYB- ARE GENERATED. DATA COUNTER IS ADVANCED.
TRANSFER DATA BYTE TO F REGISTER
READ/WRITE LOOP PROGRAM TRANSFERS DATA BYTE TO CPU.
DOES A = B ?
YES
NO
ADVANCE PROGRAM COUNTER TO 19.
BRANCH PROGRAM COUNTER TO 63. HALT
IS DATA COUNT 255 ?
NO
YES
SERIAL NRI DATA BITS LOAD CRC POLYNOMIAL INTO READ BYTE REGISTER AND CRC CHECK GENERATOR.
WAIT
NO
IS BIT COUNT 8 ?
YES
BYTE READY+ AND BYTE READYB- ARE GENERATED. POSTAMBLE COUNTER IS ADVANCED.
TRANSFER CRC BYTE TO F REGISTER.
IS FORMAT SET ?
NO
YES
READ/WRITE LOOP PROGRAM TRANSFERS CRC BYTE TO CPU.
IS POSTAMBLE COUNT 2 ?
NO
YES
POSTAMBLE ROM ISSUES CRC BYTE ONE TIME (CRC2-). CRC1- CLOCKS CRC ERROR LATCH.
CRC CHECK GENERATOR DIVIDES ACCUMULATED ONE BITS BY CRC POLYNOMIAL.
IS THERE A REMAINDER ?
NO
YES
CRC ERROR LATCH IS SET. CRC ERROR AVAILABLE AS STATUS.
WAIT
NO
IS BIT COUNT 8 ?
YES
BYTE READY+ AND BYTE READYB- ARE GENERATED. ADVANCE POSTAMBLE COUNTER.
NO
IS POSTAMBLE COUNT 4 ?
YES
GENERATE RESET READ (RSTRO-) WHICH RESETS READ DELAY COUNTER, ENRD, SYNCBIT, AND READ BYTE REGISTER.
WAIT FOR SECTOR PULSE.

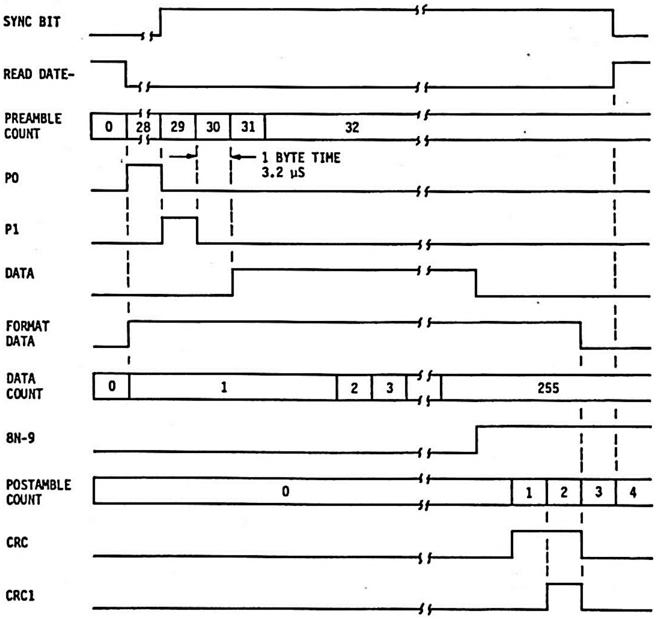
Figure 2-4. Read Mode Functional Diagrams, Sheet 1 of 2



READ LOGIC TIMING



BYTE COUNTER TIMING



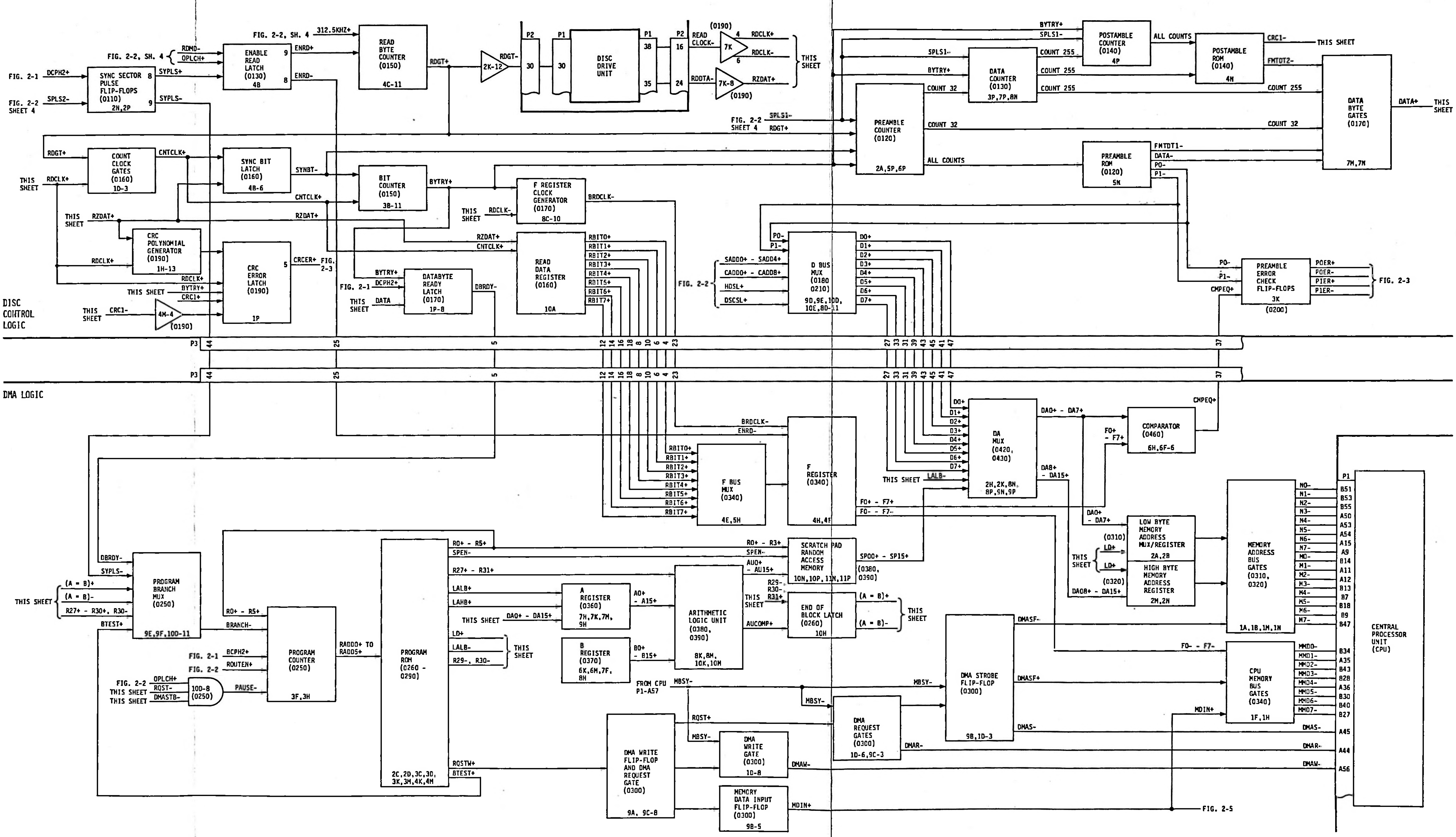

DISC CONTROL LOGIC
DMA LOGIC
SYNC SECTOR PULSE FLIP-FLOPS (0110) 2N,2P
FIG. 2-1 DCPH2+
FIG. 2-2 SHEET 4 SPLS2-
SYPLS+
SYPLS-
FIG. 2-2, SH. 4 RDMO- OPLCH+
ENABLE READ LATCH (0130) 4B
ENRD+
ENRD-
FIG. 2-2, SH. 4 312.5KHZ+
READ BYTE COUNTER (0150) 4C-11
RDGT+
2K-12
RDGT-
P2 P1 DISC DRIVE UNIT P1 P2
READ CLOCK-
RDDTA-
(0190) 7K RDCLK+ RDCLK-
7K-8 RZDAT+ (0190)
THIS SHEET
COUNT CLOCK GATES (0160) 1D-3
CNTCLK+
RDCLK+
THIS SHEET
SYNC BIT LATCH (0160) 4B-6
SYNBT-
BIT COUNTER (0150) 3B-11
BYTRY+
CNTCLK+
F REGISTER CLOCK GENERATOR (0170) 8C-10
BRDCLK-
THIS SHEET RDCLK-
RZDAT+
CRC POLYNOMIAL GENERATOR (0190) 1H-13
CRC ERROR LATCH (0190) 1P
CRCER+ FIG. 2-3
RDCLK+
BYTRY+
CRC1+
THIS SHEET CRC1-
4M-4 (0190)
DATABYTE READY LATCH (0170) 1P-8
FIG. 2-1 DCPH2+
THIS SHEET DATA
DBRDY-
READ DATA REGISTER (0160) 10A
RZDAT+
CNTCLK+
RBIT0+ RBIT1+ RBIT2+ RBIT3+ RBIT4+ RBIT5+ RBIT6+ RBIT7+
POSTAMBLE COUNTER (0140) 4P
ALL COUNTS
POSTAMBLE ROM (0140) 4N
CRC1-
FMTDT2-
THIS SHEET
SPLS1-
BYTRY+
DATA COUNTER (0130) 3P,7P,8N
COUNT 255
COUNT 32
PREAMBLE COUNTER (0120) 2A,5P,6P
FIG. 2-2 SHEET 4 SPLS1- RDGT+
PREAMBLE ROM (0120) 5N
FMTDT1-
DATA-
P0-
P1-
DATA BYTE GATES (0170) 7M,7N
DATA+
THIS SHEET
D BUS MUX (0180 0210) 9D,9E,10D, 10E,8D-11
FIG. 2-2 SADD0+ - SADD4+ CADD0+ - CADD8+ HDSL+ DSCSL+
D0+ D1+ D2+ D3+ D4+ D5+ D6+ D7+
PREAMBLE ERROR CHECK FLIP-FLOPS 3K (0200)
CMPEQ+
P0ER+ P0ER- P1ER+ P1ER-
FIG. 2-3
P3
44 25 5 12 14 16 18 8 10 6 4 23 27 33 31 39 43 45 41 47 37
BRDCLK-
ENRD-
F REGISTER (0340) 4H,4F
F BUS MUX (0340) 4E,5H
DA MUX (0420, 0430) 2H,2K,8N, 8P,9N,9P
THIS SHEET LALB-
DA0+ - DA7+
DA8+ - DA15+
COMPARATOR (0460) 6H,6F-6
F0+ - F7+
F0- - F7-
DA0+ - DA7+
LOW BYTE MEMORY ADDRESS MUX/REGISTER 2A,2B
(0310) LD+
THIS SHEET
(0320) LD+
HIGH BYTE MEMORY ADDRESS REGISTER 2M,2N
DA08+ - DA15+
MEMORY ADDRESS BUS GATES (0310, 0320) 1A,1B,1M,1N
M0- M1- M2- M3- M4- M5- M6- M7- M0- M1- M2- M3- M4- M5- M6- M7-
P1
B51 B53 B55 A50 A53 A54 A15 A9 B14 A11 A12 B13 B7 B18 B9 B47 B34 A35 B43 B28 A36 B30 B40 B27 A45 A44 A56
CENTRAL PROCESSOR UNIT (CPU)
DBRDY-
SYPLS-
THIS SHEET (A = B)+ (A = B)- R27+ - R30+, R30-
BTEST+
PROGRAM BRANCH MUX (0250) 9E,9F,10D-11
BRANCH-
R0+ - R5+
PROGRAM COUNTER (0250) 3F,3H
FIG. 2-1 BCPH2+
FIG. 2-2 ROUTEN+
FIG. 2-2 OPLCH+
THIS SHEET RQST-
THIS SHEET DMASTB-
10D-8 (0250)
PAUSE-
RADD0+ TO RADD5+
PROGRAM ROM (0260 - 0290) 2C,2D,3C,3D, 3K,3H,4K,4M
R0+ - R5+
SPEN-
R27+ - R31+
LALB+
LAHB+
THIS SHEET DA0+ - DA15+
LD+
LALB-
R29-, R30-
THIS SHEET
RQSTW+
BTEST+
SCRATCH PAD RANDOM ACCESS MEMORY 10N,10P,11N,11P
R0+ - R3+
SPEN-
AU0+ - AU15+
SP00+ - SP15+
(0380, 0390)
A REGISTER (0360) 7H,7K,7M, 9H
A0+ - A15+
B REGISTER (0370) 6K,6M,7F, 8H
B0+ - B15+
ARITHMETIC LOGIC UNIT (0380, 0390) 8K,8M, 10K,10M
THIS SHEET R29- R30- R31+
END OF BLOCK LATCH (0260) 10H
AUCOMP+
(A = B)+ (A = B)-
THIS SHEET
FROM CPU P1-A57 MBSY-
MBSY-
DMA STROBE FLIP-FLOP (0300) 9B,1D-3
DMASF-
DMASF+
DMAS-
DMA REQUEST GATES (0300) 1D-6,9C-3
RQST+
DMAR-
DMA WRITE FLIP-FLOP AND DMA REQUEST GATE (0300) 9A, 9C-8
MBSY-
DMA WRITE GATE (0300) 1D-8
DMAW-
MEMORY DATA INPUT FLIP-FLOP (0300) 9B-5
MDIN+
FIG. 2-5
CPU MEMORY BUS GATES (0340) 1F,1H
MMD0- MMD1- MMD2- MMD3- MMD4- MMD5- MMD6- MMD7-
MDIN+
DMAS-
DMAR-
DMAW-

Figure 2-4. Read Mode Functional Diagrams, Sheet 2 of 2

KEYSEARCH PARAMETER TRANSFER

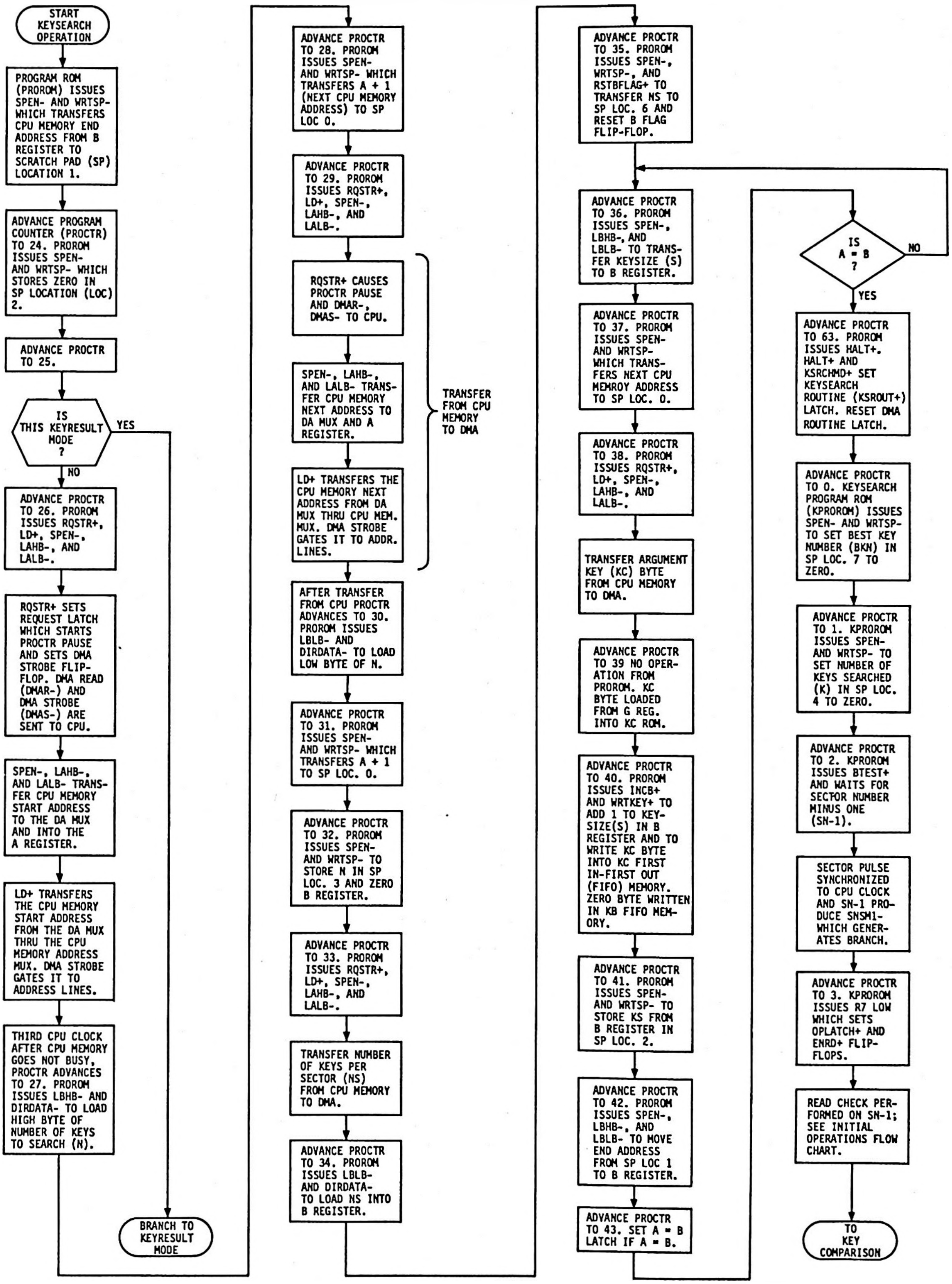

START KEYSEARCH OPERATION
PROGRAM ROM (PROROM) ISSUES SPEN- AND WRTSP- WHICH TRANSFERS CPU MEMORY END ADDRESS FROM B REGISTER TO SCRATCH PAD (SP) LOCATION 1.
ADVANCE PROGRAM COUNTER (PROCTR) TO 24. PROROM ISSUES SPEN- AND WRTSP- WHICH STORES ZERO IN SP LOCATION (LOC) 2.
ADVANCE PROCTR TO 25.
IS THIS KEYRESULT MODE ?
YES
NO
BRANCH TO KEYRESULT MODE
ADVANCE PROCTR TO 26. PROROM ISSUES RQSTR+, LD+, SPEN-, LAHB-, AND LALB-.
RQSTR+ SETS REQUEST LATCH WHICH STARTS PROCTR PAUSE AND SETS DMA STROBE FLIP-FLOP. DMA READ (DMAR-) AND DMA STROBE (DMAS-) ARE SENT TO CPU.
SPEN-, LAHB-, AND LALB- TRANSFER CPU MEMORY START ADDRESS TO THE DA MUX AND INTO THE A REGISTER.
LD+ TRANSFERS THE CPU MEMORY START ADDRESS FROM THE DA MUX THRU THE CPU MEMORY ADDRESS MUX. DMA STROBE GATES IT TO ADDRESS LINES.
THIRD CPU CLOCK AFTER CPU MEMORY GOES NOT BUSY, PROCTR ADVANCES TO 27. PROROM ISSUES LBHB- AND DIRDATA- TO LOAD HIGH BYTE OF NUMBER OF KEYS TO SEARCH (N).
ADVANCE PROCTR TO 28. PROROM ISSUES SPEN- AND WRTSP- WHICH TRANSFERS A + 1 (NEXT CPU MEMORY ADDRESS) TO SP LOC 0.
ADVANCE PROCTR TO 29. PROROM ISSUES RQSTR+, LD+, SPEN-, LAHB-, AND LALB-.
RQSTR+ CAUSES PROCTR PAUSE AND DMAR-, DMAS- TO CPU.
SPEN-, LAHB-, AND LALB- TRANSFER CPU MEMORY NEXT ADDRESS TO DA MUX AND A REGISTER.
LD+ TRANSFERS THE CPU MEMORY NEXT ADDRESS FROM DA MUX THRU CPU MEM. MUX. DMA STROBE GATES IT TO ADDR. LINES.
TRANSFER FROM CPU MEMORY TO DMA
AFTER TRANSFER FROM CPU PROCTR ADVANCES TO 30. PROROM ISSUES LBLB- AND DIRDATA- TO LOAD LOW BYTE OF N.
ADVANCE PROCTR TO 31. PROROM ISSUES SPEN- AND WRTSP- WHICH TRANSFERS A + 1 TO SP LOC. 0.
ADVANCE PROCTR TO 32. PROROM ISSUES SPEN- AND WRTSP- TO STORE N IN SP LOC. 3 AND ZERO B REGISTER.
ADVANCE PROCTR TO 33. PROROM ISSUES RQSTR+, LD+, SPEN-, LAHB-, AND LALB-.
TRANSFER NUMBER OF KEYS PER SECTOR (NS) FROM CPU MEMORY TO DMA.
ADVANCE PROCTR TO 34. PROROM ISSUES LBLB- AND DIRDATA- TO LOAD NS INTO B REGISTER.
ADVANCE PROCTR TO 35. PROROM ISSUES SPEN-, WRTSP-, AND RSTBFLAG+ TO TRANSFER NS TO SP LOC. 6 AND RESET B FLAG FLIP-FLOP.
ADVANCE PROCTR TO 36. PROROM ISSUES SPEN-, LBHB-, AND LBLB- TO TRANSFER KEYSIZE (S) TO B REGISTER.
ADVANCE PROCTR TO 37. PROROM ISSUES SPEN- AND WRTSP- WHICH TRANSFERS NEXT CPU MEMROY ADDRESS TO SP LOC. 0.
ADVANCE PROCTR TO 38. PROROM ISSUES RQSTR+, LD+, SPEN-, LAHB-, AND LALB-.
TRANSFER ARGUMENT KEY (KC) BYTE FROM CPU MEMORY TO DMA.
ADVANCE PROCTR TO 39 NO OPERATION FROM PROROM. KC BYTE LOADED FROM G REG. INTO KC ROM.
ADVANCE PROCTR TO 40. PROROM ISSUES INCB+ AND WRTKEY+ TO ADD 1 TO KEYSIZE(S) IN B REGISTER AND TO WRITE KC BYTE INTO KC FIRST IN-FIRST OUT (FIFO) MEMORY. ZERO BYTE WRITTEN IN KB FIFO MEMORY.
ADVANCE PROCTR TO 41. PROROM ISSUES SPEN- AND WRTSP- TO STORE KS FROM B REGISTER IN SP LOC. 2.
ADVANCE PROCTR TO 42. PROROM ISSUES SPEN-, LBHB-, AND LBLB- TO MOVE END ADDRESS FROM SP LOC 1 TO B REGISTER.
ADVANCE PROCTR TO 43. SET A = B LATCH IF A = B.
IS A = B ?
NO
YES
ADVANCE PROCTR TO 63. PROROM ISSUES HALT+. HALT+ AND KSRCHMD+ SET KEYSEARCH ROUTINE (KSROUT+) LATCH. RESET DMA ROUTINE LATCH.
ADVANCE PROCTR TO 0. KEYSEARCH PROGRAM ROM (KPROROM) ISSUES SPEN- AND WRTSP- TO SET BEST KEY NUMBER (BKN) IN SP LOC. 7 TO ZERO.
ADVANCE PROCTR TO 1. KPROROM ISSUES SPEN- AND WRTSP- TO SET NUMBER OF KEYS SEARCHED (K) IN SP LOC. 4 TO ZERO.
ADVANCE PROCTR TO 2. KPROROM ISSUES BTEST+ AND WAITS FOR SECTOR NUMBER MINUS ONE (SN-1).
SECTOR PULSE SYNCHRONIZED TO CPU CLOCK AND SN-1 PRODUCE SNSM1- WHICH GENERATES BRANCH.
ADVANCE PROCTR TO 3. KPROROM ISSUES R7 LOW WHICH SETS OPLATCH+ AND ENRD+ FLIP-FLOPS.
READ CHECK PERFORMED ON SN-1; SEE INITIAL OPERATIONS FLOW CHART.
TO KEY COMPARISON

Figure 2-5. Key Search Mode Functional Diagrams, Sheet 1 of 6

]

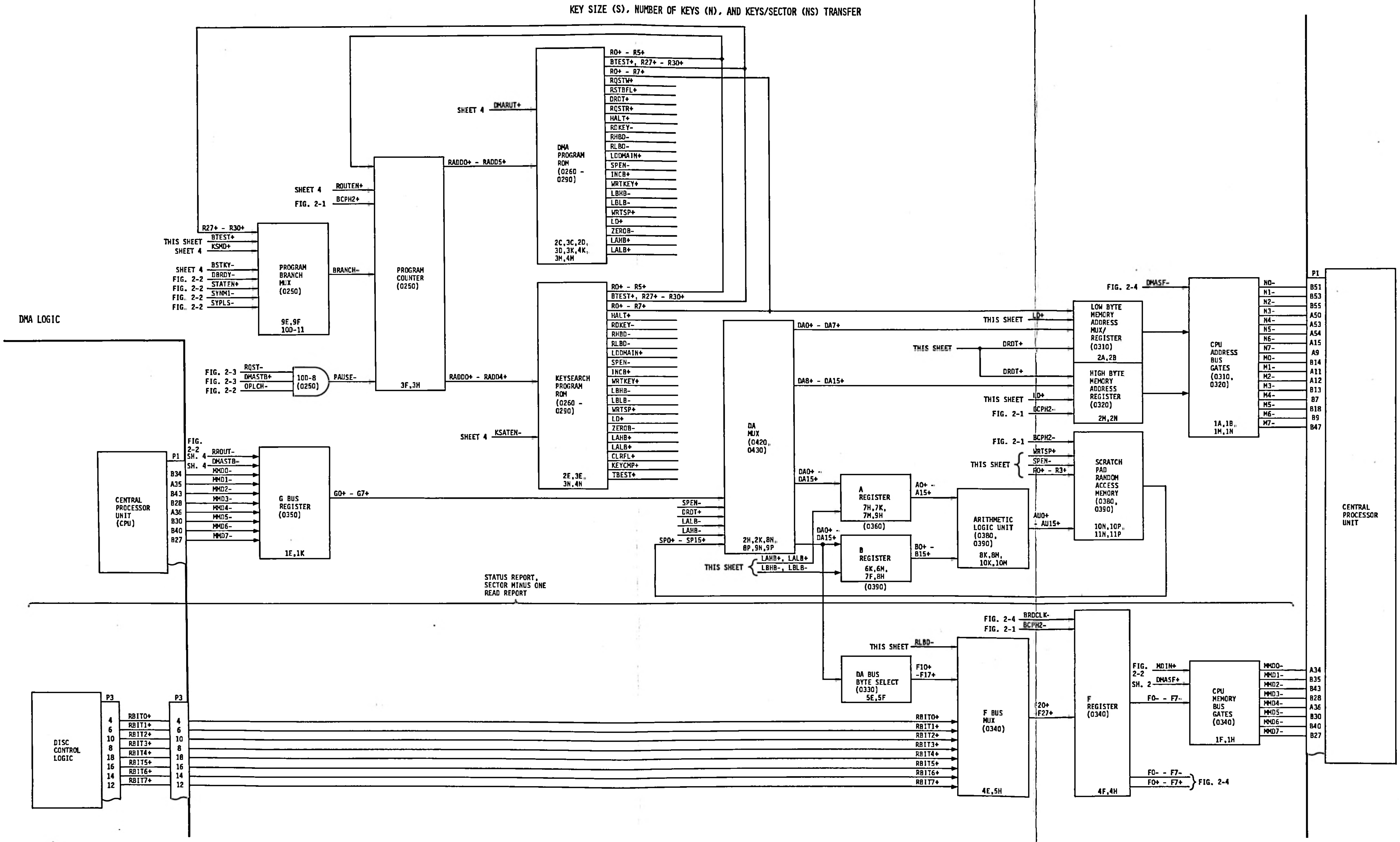

KEY SIZE (S), NUMBER OF KEYS (N), AND KEYS/SECTOR (NS) TRANSFER
DMA LOGIC
DMA PROGRAM ROM (0260 - 0290) 2C,3C,2D, 3D,3K,4K, 3M,4M
SHEET 4 DMARUT+
R0+ - R5+
BTEST+, R27+ - R30+
R0+ - R7+
RQSTW+
RSTBFL+
DRDT+
RQSTR+
HALT+
RDKEY-
RHBD-
RLBD-
LDDMAIN+
SPEN-
INCB+
WRTKEY+
LBHB-
LBLB-
WRTSP+
LD+
ZEROB-
LAHB+
LALB+
RADDO+ - RADD5+
PROGRAM COUNTER (0250) 3F,3H
SHEET 4 ROUTEN+
FIG. 2-1 BCPH2+
PROGRAM BRANCH MUX (0250) 9E,9F 10D-11
BRANCH-
R27+ - R30+
THIS SHEET BTEST+
SHEET 4 KSMD+
SHEET 4 BSTKY-
FIG. 2-2 DBRDY-
FIG. 2-2 STATEN+
FIG. 2-2 SYNMI-
FIG. 2-2 SYPLS-
FIG. 2-3 RQST-
FIG. 2-3 DMASTB+
FIG. 2-2 OPLCH-
10D-8 (0250)
PAUSE-
RADDO+ - RADD4+
KEYSEARCH PROGRAM ROM (0260 - 0290) 2E,3E, 3N,4N
SHEET 4 KSATEN-
CLRFL+
KEYCMP+
TBEST+
CENTRAL PROCESSOR UNIT (CPU)
P1
B34 A35 B43 B28 A36 B30 B40 B27
FIG. 2-2 SH. 4 RROUT-
SH. 4 DMASTB-
MMD0- MMD1- MMD2- MMD3- MMD4- MMD5- MMD6- MMD7-
G BUS REGISTER (0350) 1E,1K
G0+ - G7+
DA MUX (0420, 0430) 2H,2K,8N, 8P,9N,9P
DA0+ - DA7+
DA8+ - DA15+
SPEN- DRDT+ LALB- LAHB-
SP0+ - SP15+
THIS SHEET LAHB+, LALB+ LBHB-, LBLB-
DA0+ - DA15+
A REGISTER 7H,7K, 7M,9H (0360)
A0+ - A15+
B REGISTER 6K,6M, 7F,8H (0390)
B0+ - B15+
ARITHMETIC LOGIC UNIT (0380, 0390) 8K,8M, 10K,10M
AU0+ - AU15+
SCRATCH PAD RANDOM ACCESS MEMORY (0380, 0390) 10N,10P, 11N,11P
FIG. 2-1 BCPH2-
THIS SHEET WRTSP+ SPEN- R0+ - R3+
LOW BYTE MEMORY ADDRESS MUX/ REGISTER (0310) 2A,2B
HIGH BYTE MEMORY ADDRESS REGISTER (0320) 2M,2N
THIS SHEET LD+
THIS SHEET DRDT+
DRDT+
THIS SHEET LD+
FIG. 2-1 BCPH2-
FIG. 2-4 DMASF-
CPU ADDRESS BUS GATES (0310, 0320) 1A,1B, 1M,1N
N0- N1- N2- N3- N4- N5- N6- N7- M0- M1- M2- M3- M4- M5- M6- M7-
P1
B51 B53 B55 A50 A53 A54 A15 A9 B14 A11 A12 B13 B7 B18 B9 B47
CENTRAL PROCESSOR UNIT
STATUS REPORT, SECTOR MINUS ONE READ REPORT
DISC CONTROL LOGIC
P3
4 6 10 8 18 16 14 12
RBIT0+ RBIT1+ RBIT2+ RBIT3+ RBIT4+ RBIT5+ RBIT6+ RBIT7+
DA BUS BYTE SELECT (0330) 5E,5F
F10+ -F17+
THIS SHEET RLBD-
F BUS MUX (0340) 4E,5H
F20+ F27+
FIG. 2-4 BRDCLK-
FIG. 2-1 BCPH2-
F REGISTER (0340) 4F,4H
FIG. 2-2 MDIN+
SH. 2 DMASF+
F0- - F7-
CPU MEMORY BUS GATES (0340) 1F,1H
MMD0- MMD1- MMD2- MMD3- MMD4- MMD5- MMD6- MMD7-
A34 B35 B43 B28 A36 B30 B40 B27
F0- - F7-
F0+ - F7+
FIG. 2-4

Figure 2-5. Key Search Mode Functional Diagrams, Sheet 2 of 6

PROGRAM ROM OUTPUTS

MNEMONIC	ORIGIN REFERENCE DESIGNATORS	PROGRAM FUNCTIONS		KEYSEARCH FUNCTION
R0	2C-9, 2D-9, 2E-9	PROGRAM BRANCH ADDRESS	CPU MEMORY ADDRESS	SAME AS PROGRAM
R1	2C-7, 2D-7, 2E-7			
R2	2C-6, 2D-6, 2E-6			
R3	2C-5, 2D-5, 2E-5			
R4	2C-4, 2D-4, 2E-4			
R5	2C-3, 2D-3, 2E-3			
R6	2C-2, 2D-2, 2E-2	START READ		
R7	2C-1, 2E-1	START OPERATION		
ZEROB	2D-1, 3E-6	ZERO B REGISTER		ZERO B REGISTER
LD	3C-9, 3D-9, 3E-9	LOAD CPU MEMORY ADDRESS REGISTER		SAME
DIRDATA	3C-7, 3D-7	REQUEST READ FROM CPU MEMORY		
SETBEST	3E-7			SWITCH BEST FLAG FLIP-FLOP
RQSTR	3C-6, 3D-6	DATA FROM CONTROLLER SELECTED MEMORY LOCATIONS		
LALB	3C-5, 3D-5, 3E-5	LOAD A REGISTER LOW BYTE		SAME
LAHB	3C-4, 3D-4, 3E-4	LOAD A REGISTER HIGH BYTE		SAME
WRTSP	3C-3, 3D-3, 3E-3	WRITE IN SCRATCH PAD MEMORY		SAME
LBLB	3C-2, 3D-2, 3E-2	LOAD B REGISTER LOW BYTE		SAME
LBHB	3C-1, 3D-1, 3E-1	LOAD B REGISTER HIGH BYTE		SAME
WRTKEY	4K-9, 4M-9, 4N-9	WRITE INTO KEY FIFO MEMORY		SAME
RSTBFLAG	4K-7, 4M-7	RESET BEST FLAG FLIP-FLOP		
EKYCOMPARE	4N-7			COMPARE KD WITH KC AND KB
INCB	4K-6, 4M-6, 4N-6	INCREMENT B REGISTER		SAME
SPEN	4K-5, 4M-5, 4N-5	ENABLE SCRATCH PAD MEMORY		SAME
LDDMAIN	4K-4, 4M-4, 4N-4	LOAD DMA INPUT REGISTER		SAME
RQSTW	4K-3, 4M-3	REQUEST WRITE INTO CPU MEMORY		
CLEARFLAGS	4N-3			CLEAR KEY COMPARE LATCHES
RLBD	4K-2, 4M-2, 4N-2	ENABLE DA LOW BYTE TO F MUX		SAME
RHBD	4K-1, 4M-1, 4N-1	ENABLE DA HIGH BYTE TO F MUX		SAME
RDKEY	3K-9, 3M-9, 3N-9	READ FROM KEY FIFO MEMORY		SAME
BTEST	3K-7, 3M-7, 3N-7	TEST FOR BRANCH CONDITION		SAME
HALT	3K-6, 3M-6, 3N-7	HALT DMA PROGRAM		SAME
R27	3K-5, 3M-5, 3N-5	ARITHMETIC UNIT FUNCTION CODE	PROGRAM BRANCH CONDITION CODE	SAME
R28	3K-4, 3M-4, 3N-4			
R29	3K-3, 3M-3, 3N-3			
R30	3K-2, 3M-2, 3N-2			
R31	3K-1, 3M-1, 3N-1			

ARITHMETIC UNIT FUNCTIONS

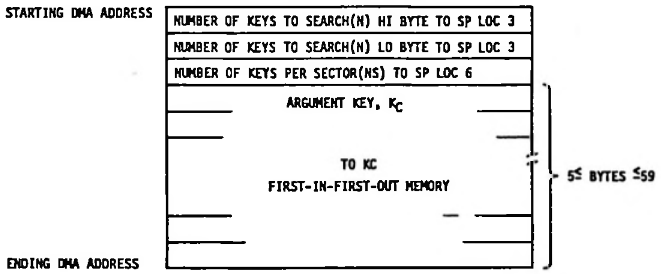
ARITHMETIC UNIT OUTPUT	ARITHMETIC UNIT FUNCTION CODE				
	R31	R30	R29	R28	R27
A REGISTER +1	0	0	0	0	0
ZERO	0	0	1	1	1
COMPARE A = B	1	0	0	1	1
B REGISTER	1	0	1	0	1
A REGISTER	1	1	1	1	1

NOTE: ALL CODES PERFORM OPERATIONS; ONLY THE ABOVE ARE USED.

PROGRAM BRANCH CONDITIONS

CONDITION TESTED FOR	BRANCH CODE			
	R30	R29	R28	R27
KEYSEARCH MODE	0	0	0	0
KEY RESULT MODE	0	0	0	1
A REG = B REG	0	0	1	0
SPARE	0	0	1	1
SPARE	0	1	0	0
BEST KEY NOT LOCATED	0	1	0	0
A REG = B REG	0	1	1	0
BRANCH UNCONDITIONALLY	0	1	1	1
STARTING SECTOR MINUS ONE	1	0	0	0
SECTOR START	1	0	0	1
DATA BYTE TIME	1	0	1	0
STATUS CHECK	1	0	1	1
SPARE	1	1	0	0
SPARE	1	1	0	1
SPARE	1	1	1	0
SPARE	1	1	1	1

KEYSEARCH PARAMETERS


STARTING DMA ADDRESS
NUMBER OF KEYS TO SEARCH(N) HI BYTE TO SP LOC 3
NUMBER OF KEYS TO SEARCH(N) LO BYTE TO SP LOC 3
NUMBER OF KEYS PER SECTOR(NS) TO SP LOC 6
ARGUMENT KEY, K_C
TO KC
FIRST-IN-FIRST-OUT MEMORY
5≤ BYTES ≤59
ENDING DMA ADDRESS

KEYSEARCH RESULTS

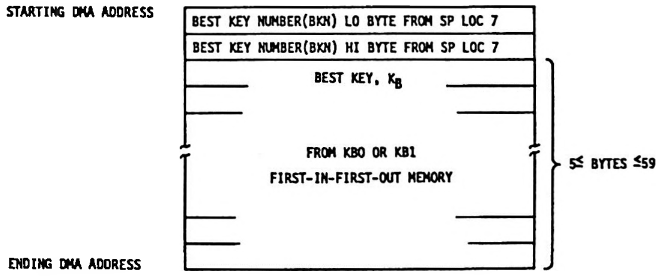

STARTING DMA ADDRESS
BEST KEY NUMBER(BKN) LO BYTE FROM SP LOC 7
BEST KEY NUMBER(BKN) HI BYTE FROM SP LOC 7
BEST KEY, K_B
FROM KB0 OR KB1
FIRST-IN-FIRST-OUT MEMORY
5≤ BYTES ≤59
ENDING DMA ADDRESS

Figure 2-5. Key Search Mode Functional Diagrams, Sheet 3 of 6

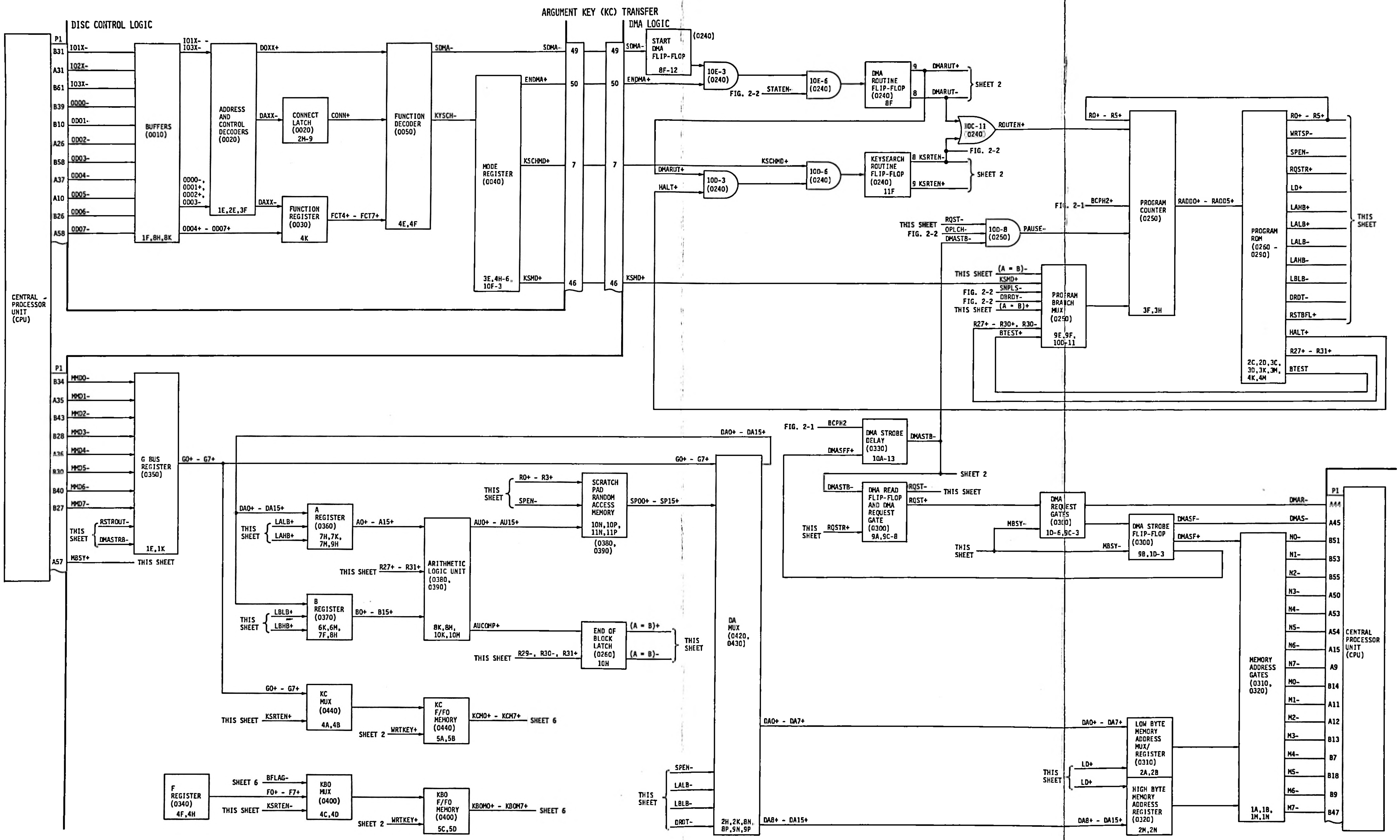


Figure 2-5. Key Search Mode Functional Diagrams, Sheet 4 of 6

KEY COMPARISON

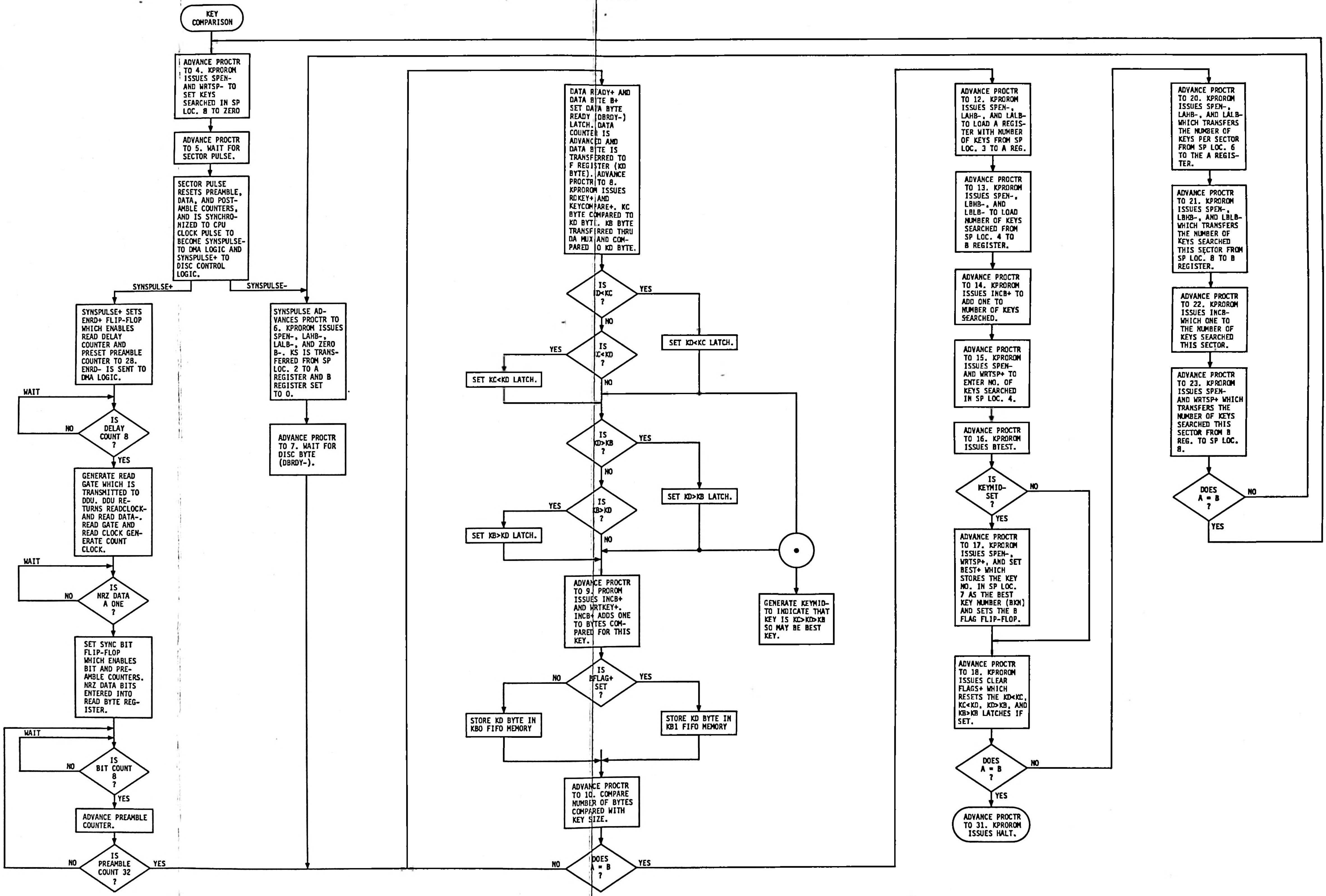

KEY COMPARISON
ADVANCE PROCTR TO 4. KPROROM ISSUES SPEN- AND WRTSP- TO SET KEYS SEARCHED IN SP LOC. 8 TO ZERO
ADVANCE PROCTR TO 5. WAIT FOR SECTOR PULSE.
SECTOR PULSE RESETS PREAMBLE, DATA, AND POSTAMBLE COUNTERS, AND IS SYNCHRONIZED TO CPU CLOCK PULSE TO BECOME SYNSPULSE- TO DMA LOGIC AND SYNSPULSE+ TO DISC CONTROL LOGIC.
SYNSPULSE+
SYNSPULSE-
SYNSPULSE+ SETS ENRD+ FLIP-FLOP WHICH ENABLES READ DELAY COUNTER AND PRESET PREAMBLE COUNTER TO 28. ENRD- IS SENT TO DMA LOGIC.
WAIT
NO
IS DELAY COUNT 8 ?
YES
GENERATE READ GATE WHICH IS TRANSMITTED TO DDU. DDU RETURNS READCLOCK- AND READ DATA-. READ GATE AND READ CLOCK GENERATE COUNT CLOCK.
WAIT
NO
IS NRZ DATA A ONE ?
SET SYNC BIT FLIP-FLOP WHICH ENABLES BIT AND PREAMBLE COUNTERS. NRZ DATA BITS ENTERED INTO READ BYTE REGISTER.
WAIT
NO
IS BIT COUNT 8 ?
YES
ADVANCE PREAMBLE COUNTER.
NO
IS PREAMBLE COUNT 32 ?
YES
SYNSPULSE ADVANCES PROCTR TO 6. KPROROM ISSUES SPEN-, LAHB-, LALB-, AND ZERO B-. KS IS TRANSFERRED FROM SP LOC. 2 TO A REGISTER AND B REGISTER SET TO 0.
ADVANCE PROCTR TO 7. WAIT FOR DISC BYTE (DBRDY-).
DATA READY+ AND DATA BYTE B+ SET DATA BYTE READY (DBRDY-) LATCH. DATA COUNTER IS ADVANCED AND DATA BYTE IS TRANSFERRED TO F REGISTER (KD BYTE). ADVANCE PROCTR TO 8. KPROROM ISSUES RDKEY+ AND KEYCOMPARE+. KC BYTE COMPARED TO KD BYTE. KB BYTE TRANSFERRED THRU DA MUX AND COMPARED TO KD BYTE.
IS KD<KC ?
YES
NO
SET KD<KC LATCH.
YES
IS KC<KD ?
NO
SET KC<KD LATCH.
IS KD>KB ?
YES
NO
SET KD>KB LATCH.
YES
IS KB>KD ?
NO
SET KB>KD LATCH.
GENERATE KEYMID- TO INDICATE THAT KEY IS KC>KD>KB SO MAY BE BEST KEY.
ADVANCE PROCTR TO 9. PROROM ISSUES INCB+ AND WRTKEY+. INCB+ ADDS ONE TO BYTES COMPARED FOR THIS KEY.
NO
IS BFLAG+ SET ?
YES
STORE KD BYTE IN KBO FIFO MEMORY
STORE KD BYTE IN KB1 FIFO MEMORY
ADVANCE PROCTR TO 10. COMPARE NUMBER OF BYTES COMPARED WITH KEY SIZE.
NO
DOES A = B ?
YES
ADVANCE PROCTR TO 12. KPROROM ISSUES SPEN-, LAHB-, AND LALB- TO LOAD A REGISTER WITH NUMBER OF KEYS FROM SP LOC. 3 TO A REG.
ADVANCE PROCTR TO 13. KPROROM ISSUES SPEN-, LBHB-, AND LBLB- TO LOAD NUMBER OF KEYS SEARCHED FROM SP LOC. 4 TO B REGISTER.
ADVANCE PROCTR TO 14. KPROROM ISSUES INCB+ TO ADD ONE TO NUMBER OF KEYS SEARCHED.
ADVANCE PROCTR TO 15. KPROROM ISSUES SPEN- AND WRTSP+ TO ENTER NO. OF KEYS SEARCHED IN SP LOC. 4.
ADVANCE PROCTR TO 16. KPROROM ISSUES BTEST.
IS KEYMID- SET ?
NO
YES
ADVANCE PROCTR TO 17. KPROROM ISSUES SPEN-, WRTSP+, AND SET BEST+ WHICH STORES THE KEY NO. IN SP LOC. 7 AS THE BEST KEY NUMBER (BKN) AND SETS THE B FLAG FLIP-FLOP.
ADVANCE PROCTR TO 18. KPROROM ISSUES CLEAR FLAGS+ WHICH RESETS THE KD<KC, KC<KD, KD>KB, AND KB>KB LATCHES IF SET.
DOES A = B ?
NO
YES
ADVANCE PROCTR TO 31. KPROROM ISSUES HALT.
ADVANCE PROCTR TO 20. KPROROM ISSUES SPEN-, LAHB-, AND LALB- WHICH TRANSFERS THE NUMBER OF KEYS PER SECTOR FROM SP LOC. 6 TO THE A REGISTER.
ADVANCE PROCTR TO 21. KPROROM ISSUES SPEN-, LBHB-, AND LBLB- WHICH TRANSFERS THE NUMBER OF KEYS SEARCHED THIS SECTOR FROM SP LOC. 8 TO B REGISTER.
ADVANCE PROCTR TO 22. KPROROM ISSUES INCB- WHICH ONE TO THE NUMBER OF KEYS SEARCHED THIS SECTOR.
ADVANCE PROCTR TO 23. KPROROM ISSUES SPEN- AND WRTSP+ WHICH TRANSFERS THE NUMBER OF KEYS SEARCHED THIS SECTOR FROM B REG. TO SP LOC. 8.
DOES A = B ?
NO
YES

Figure 2-5. Key Search Mode Functional Diagrams, Sheet 5 of 6

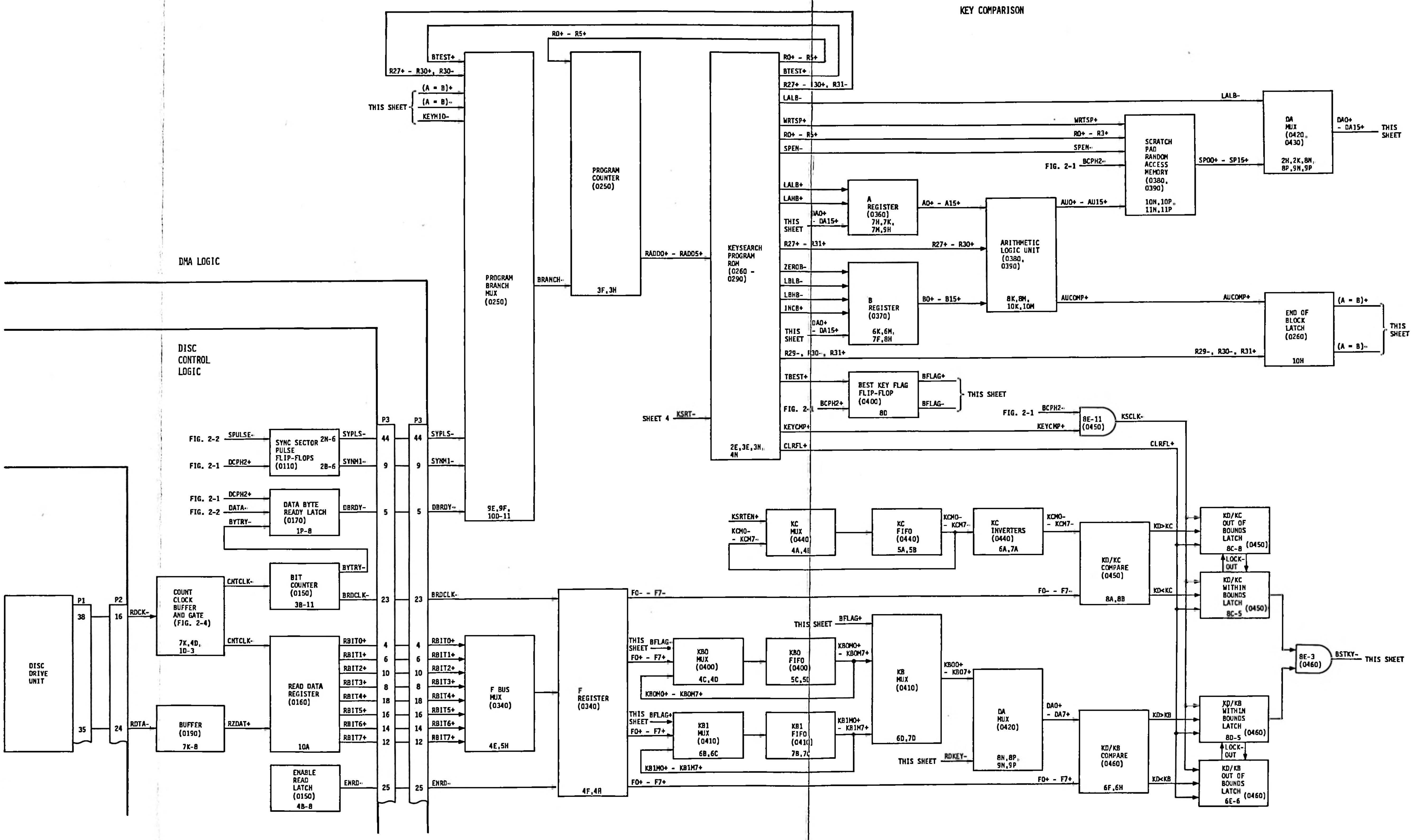


Figure 2-5. Key Search Mode Functional Diagrams, Sheet 6 of 6

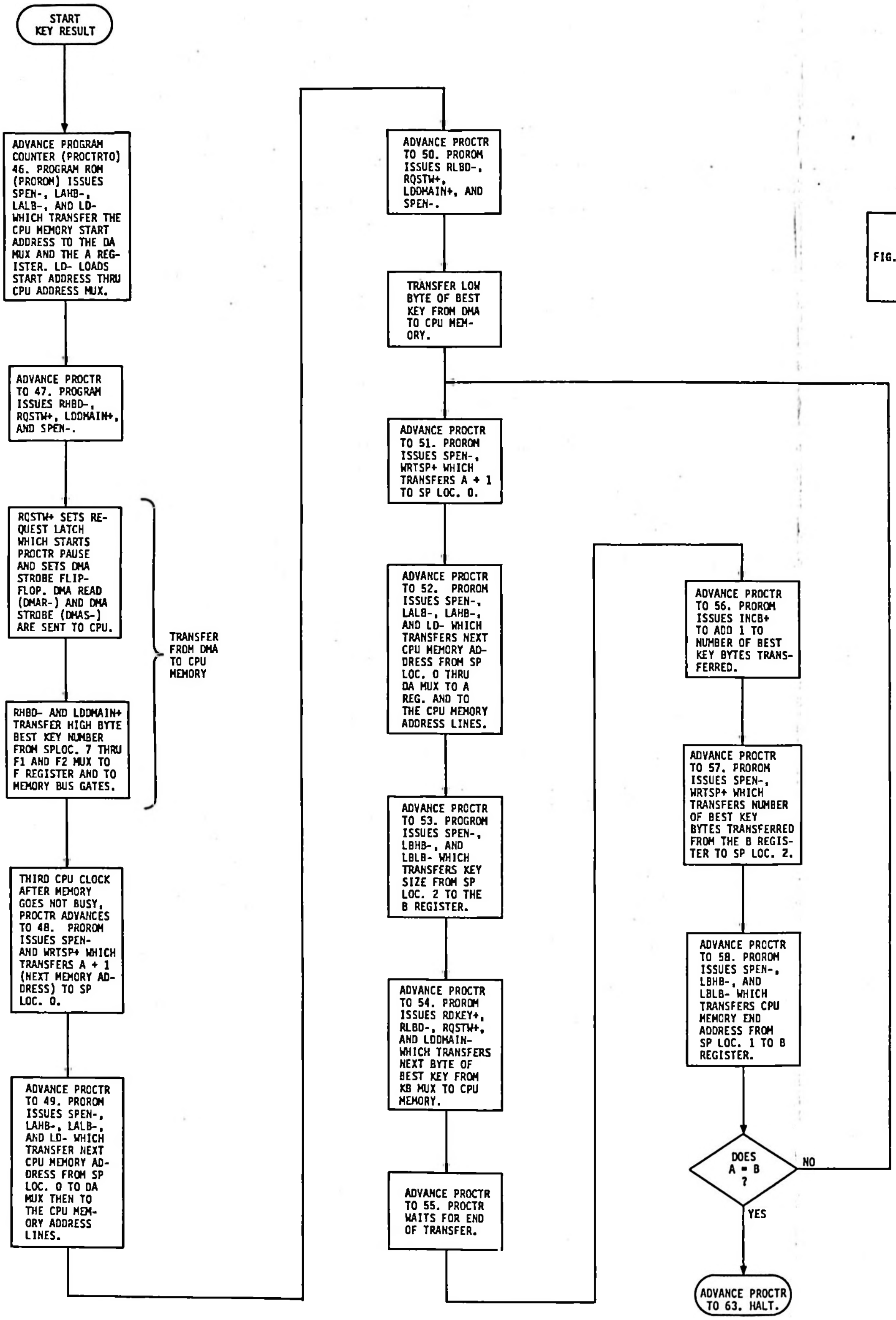

START KEY RESULT
ADVANCE PROGRAM COUNTER (PROCTRTO) 46. PROGRAM ROM (PROROM) ISSUES SPEN-, LAHB-, LALB-, AND LD- WHICH TRANSFER THE CPU MEMORY START ADDRESS TO THE DA MUX AND THE A REGISTER. LD- LOADS START ADDRESS THRU CPU ADDRESS MUX.
ADVANCE PROCTR TO 47. PROGRAM ISSUES RHBD-, RQSTW+, LDDMAIN+, AND SPEN-.
RQSTW+ SETS REQUEST LATCH WHICH STARTS PROCTR PAUSE AND SETS DMA STROBE FLIP-FLOP. DMA READ (DMAR-) AND DMA STROBE (DMAS-) ARE SENT TO CPU.
RHBD- AND LDDMAIN+ TRANSFER HIGH BYTE BEST KEY NUMBER FROM SPLOC. 7 THRU F1 AND F2 MUX TO F REGISTER AND TO MEMORY BUS GATES.
TRANSFER FROM DMA TO CPU MEMORY
THIRD CPU CLOCK AFTER MEMORY GOES NOT BUSY, PROCTR ADVANCES TO 48. PROROM ISSUES SPEN- AND WRTSP+ WHICH TRANSFERS A + 1 (NEXT MEMORY ADDRESS) TO SP LOC. 0.
ADVANCE PROCTR TO 49. PROROM ISSUES SPEN-, LAHB-, LALB-, AND LD- WHICH TRANSFER NEXT CPU MEMORY ADDRESS FROM SP LOC. 0 TO DA MUX THEN TO THE CPU MEMORY ADDRESS LINES.
ADVANCE PROCTR TO 50. PROROM ISSUES RLBD-, RQSTW+, LDDMAIN+, AND SPEN-.
TRANSFER LOW BYTE OF BEST KEY FROM DMA TO CPU MEMORY.
ADVANCE PROCTR TO 51. PROROM ISSUES SPEN-, WRTSP+ WHICH TRANSFERS A + 1 TO SP LOC. 0.
ADVANCE PROCTR TO 52. PROROM ISSUES SPEN-, LALB-, LAHB-, AND LD- WHICH TRANSFERS NEXT CPU MEMORY ADDRESS FROM SP LOC. 0 THRU DA MUX TO A REG. AND TO THE CPU MEMORY ADDRESS LINES.
ADVANCE PROCTR TO 53. PROGROM ISSUES SPEN-, LBHB-, AND LBLB- WHICH TRANSFERS KEY SIZE FROM SP LOC. 2 TO THE B REGISTER.
ADVANCE PROCTR TO 54. PROROM ISSUES RDKEY+, RLBD-, RQSTW+, AND LDDMAIN- WHICH TRANSFERS NEXT BYTE OF BEST KEY FROM KB MUX TO CPU MEMORY.
ADVANCE PROCTR TO 55. PROCTR WAITS FOR END OF TRANSFER.
ADVANCE PROCTR TO 56. PROROM ISSUES INCB+ TO ADD 1 TO NUMBER OF BEST KEY BYTES TRANSFERRED.
ADVANCE PROCTR TO 57. PROROM ISSUES SPEN-, WRTSP+ WHICH TRANSFERS NUMBER OF BEST KEY BYTES TRANSFERRED FROM THE B REGISTER TO SP LOC. 2.
ADVANCE PROCTR TO 58. PROROM ISSUES SPEN-, LBHB-, AND LBLB- WHICH TRANSFERS CPU MEMORY END ADDRESS FROM SP LOC. 1 TO B REGISTER.
DOES A = B ?
NO
YES
ADVANCE PROCTR TO 63. HALT.

FIG. 2-3

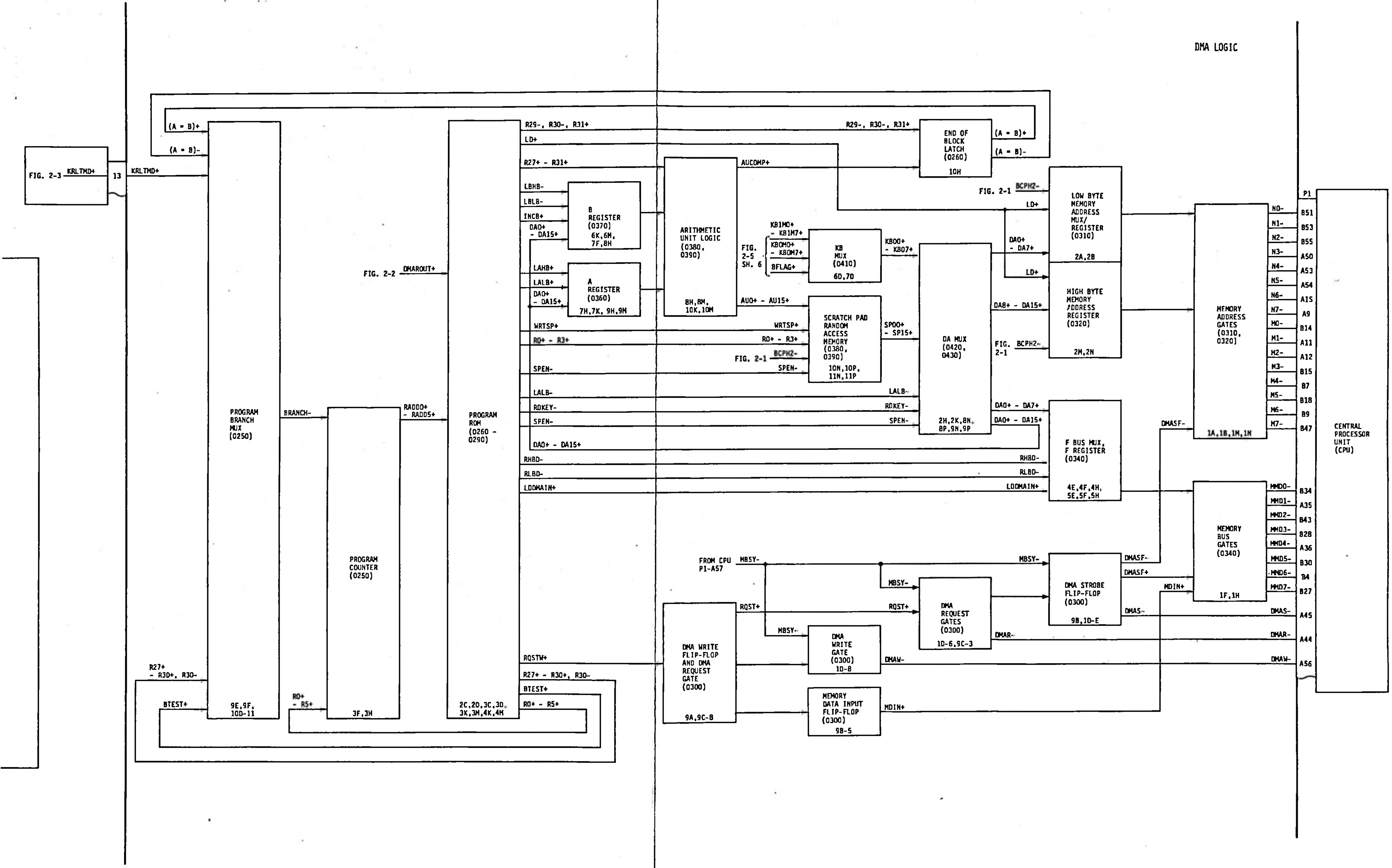


Figure 2-6. Key Result Mode Functional Diagram

Section 3

PARTS LIST

This section contains the parts list for the DMA/Disc Storage Unit Controller. A DMA/Disc Storage Unit Controller consists of a Disc Control Logic Board, Part Number 900850, a DMA Logic Microprocessor Board, Part Number 900860 or 901601, and an interconnecting cable, Part Number 900960. The parts are listed in reference designator order. Figures 3-1 and 3-2 are the parts location diagrams for the Disc Control Logic Board and the DMA Logic Microprocessor Board respectively.

Parts List for Disc Control Logic Board, figure 3-1

Reference Designator	Part No.	Part Name	Quantity
	900850	Disc Board	Ref
C1 thru C10, C37 thru C46	101101	Capacitor, 4.7µf (196D), 50V	20
C11 thru C14, C16 thru C20, C22 thru C34,C36	101100	Capacitor, 0.01µf ±20%, 100V	23
C15	101119	Capacitor, 680pf ±5%, 500V	1
C21,C35	101129	Capacitor, 330pf ±10%, 100V	2
P2,P3	100125	Connector	2
R1,R2,R3,R7,R9	101003	Resistor, 1K ±5%, 1/4w	5
R4	101016	Resistor, 680 ohm ±5%, 1/4w	1
R5,R8	101018	Resistor, 470 ohm ±5%, 1/4w	2
R6	101045	Resistor, 4.7K ±2%, 1/4w	1
1C,7M	101514	Integrated Circuit, SN7438	2
1D	101532	Integrated Circuit, SN7437	1
1E,4E,4F,9P	101513	Integrated Circuit, SN7442	4
1F,1K,2D,2E,3C,4A,4M,6N, 7E,7H,7K,8C,8H,8K,10P	101509	Integrated Circuit, SN7404	15
1H,7A	101619	Integrated Circuit, 9401	2
1M,2B,4D,4H,5B,5K,7N,8D	101507	Integrated Circuit, SN7400	8
1N,2A,3F,5A,7B	101506	Integrated Circuit, SN7410	5
1P,2M,2N,2P,3D,3E,3H,3K, 4B,5H,6H,6M,8N,9A,9M,9N, 10H	101505	Integrated Circuit, SN7474	17
2C	101599	Integrated Circuit, SN74H08	1
2F,3N	101502	Integrated Circuit, SN7402	2
2H,6A,8F	101508	Integrated Circuit, SN7420	3
2K,10K,10M,10N	101316	Integrated Circuit, SN7406	4
3A,3B,3P,4C,4P,5E,5F,5P, 6E,6F,6P,7P	101512	Integrated Circuit, SN74161	12
3M,6B,6K,10F	101500	Integrated Circuit, SN7408	4
4K	101549	Integrated Circuit, SN74175N	1
4N	101595	Integrated Circuit, 8223-4ND	1
5C,6C	101515	Integrated Circuit, SN7485	2
5D,6D	101318	Integrated Circuit, SN74157	2
5M	101052	Resistor Pack, 470 ohm	1
5N	101594	Integrated Circuit, 8223-5ND	1
7D	101504	Integrated Circuit, SN74107N	1
7F	101051	Resistor Pack, 220/330 ohm	1
8P	101516	Integrated Circuit, SN74121	1
9D,9E,10D,10E	101536	Integrated Circuit, SN74153	4
9H,9K	101548	Integrated Circuit, SN74174N	2
10A	101551	Integrated Circuit, SN74164N	1
	200032	Rivet	2
	200033	Extractor	2

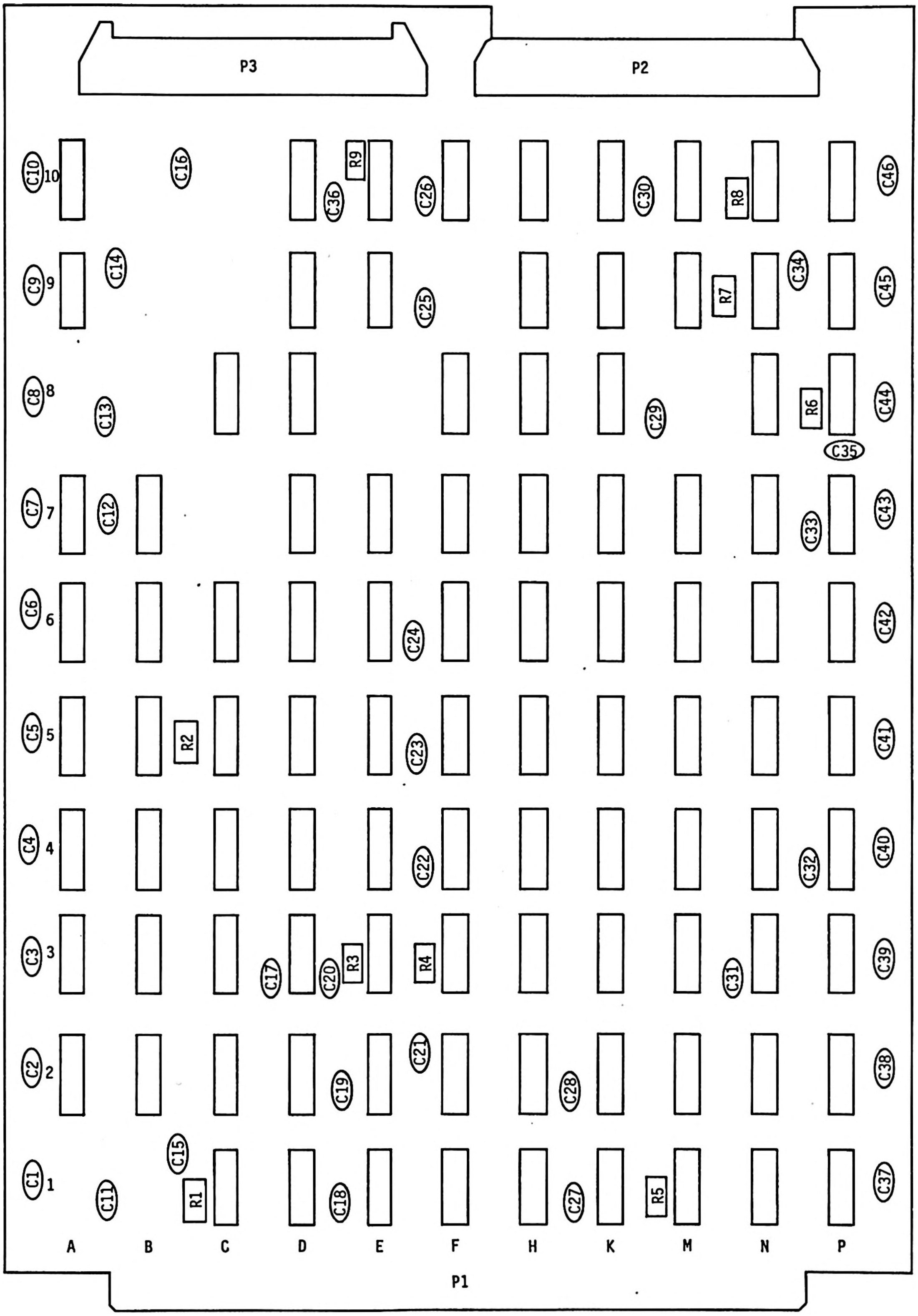


Figure 3-1. Disc Control Logic Board

Parts List for DMA Logic Microprocessor Board, figure 3-2

Reference Designator	Part No.	Part Name	Quantity
	900860	Microprocessor Board or (P/N 901601)	Ref
C1 thru C10,C26 thru C36	101101	Capacitor, 4.7μf (196D), 50V	21
C11,C12,C14 thru C17, C19,C21 thru C25	101100	Capacitor, 0.01μf ±20%, 100V	12
C13,C20	101112	Capacitor, 4.7μf ±10% (150D), 10V	2
P3	100125	Connector	1
Q1	101308	Integrated Circuit, LM320H-12	1
R3,R9,R10	101018	Resistor, 470 ohm, ±5%, 1/4w	3
R2,R4,R7,R8,R11,R12	101003	Resistor, 1K, ±5%, 1/4w	6
1A,1B,1D,1F,1H,1M,1N, 10F	101514	Integrated Circuit, SN7438	8
1C	101532	Integrated Circuit, SN7437	1
1E,1K,2M,2N,4F,4H,7H, 7K,7M,9H	101549	Integrated Circuit, SN74175N	10
1P,2P	101510	Integrated Circuit, SN7440	2
2A,2B	101597	Integrated Circuit, SN74298	2
2C	101603	Integrated Circuit, IM5610-2C	1
2D	101607	Integrated Circuit, IM5610-2D	1
2E	101611	Integrated Circuit, IM5610-2E	1
2H,2K,4E,5E,5F,5H	101318	Integrated Circuit, SN74157	6
3A,9D,7E,10D	101500	Integrated Circuit, SN7408	4
3B,5N,5P,11H,11K,11M	101541	Integrated Circuit, SN74S04N	6
3C	101602	Integrated Circuit, IM5610-3C	1
3D	101606	Integrated Circuit, IM5610-3D	1
3E	101610	Integrated Circuit, IM5610-3E	1
3F,3H	101653	Integrated Circuit, F93S16	2
3K	101600	Integrated Circuit, IM5610-3K	1
3M	101604	Integrated Circuit, IM5610-3M	1
3N	101608	Integrated Circuit, IM5610-3N	1
3P	101655	Integrated Circuit, SN74S02	1
4A,4B,4C,4D,6B,6C,6D,7D	101657	Integrated Circuit, SN74LS157	8
4K	101601	Integrated Circuit, IM5610-4K	1
4M	101605	Integrated Circuit, IM5610-4M	1
4N	101609	Integrated Circuit, IM5610-4N	1
5A,5B,5C,5D,7B,7C	101580	Integrated Circuit, 2841	6
5K,6P,8E,10E	101507	Integrated Circuit, SN7400	4
5M	101615	Integrated Circuit, SN74S08	1
6A,7A	101656	Integrated Circuit, SN74LS04	2
6E,8C,8D,10H	101505	Integrated Circuit, SN7474	4
6F,6H,8A,8B	101515	Integrated Circuit, SN7485	4
6K,6M,7F,8H	101317	Integrated Circuit, SN74163	4
6N	101506	Integrated Circuit, SN7410	1
7N	101537	Integrated Circuit, SN74166	1
7P,10B	101509	Integrated Circuit, SN7404	2
8F,11F	101581	Integrated Circuit, SN74H103	2
8K,8M,10K,10M	101616	Integrated Circuit, SN74S181	4
8N,8P,9N,9P	101536	Integrated Circuit, SN74153	4
9A,9B	101629	Integrated Circuit, SN74S74	2
9C,10C	101315	Integrated Circuit, SN74S00	2
9E,9F	101520	Integrated Circuit, SN74151	2
10A	101512	Integrated Circuit, SN74161	1
10N,10P,11N,11P	101598	Integrated Circuit, SN74S189	4
	200032	Rivet	2
	200033	Extractor	2

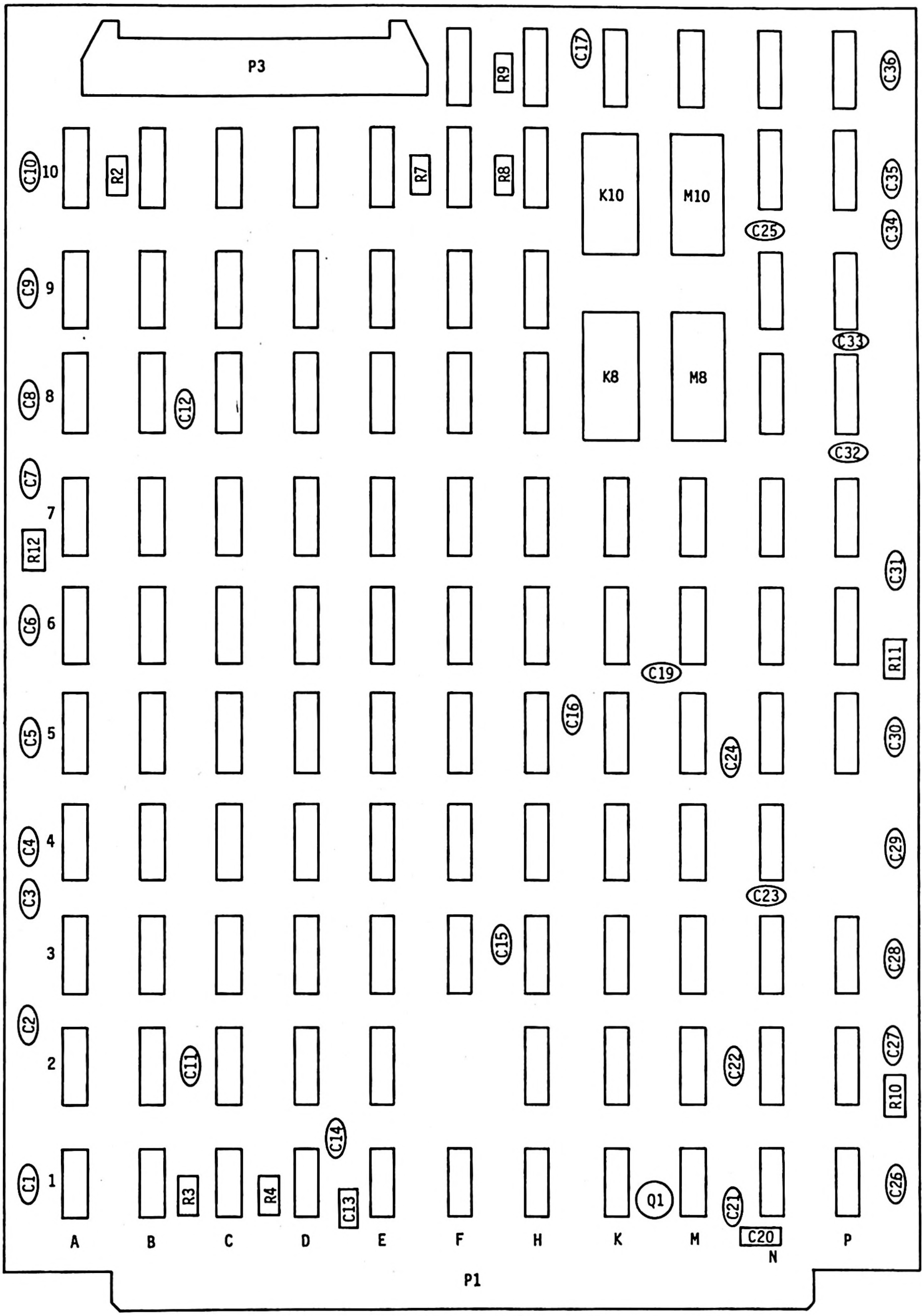


Figure 3-2. DMA Logic Microprocessor Board