SERIES 200

201

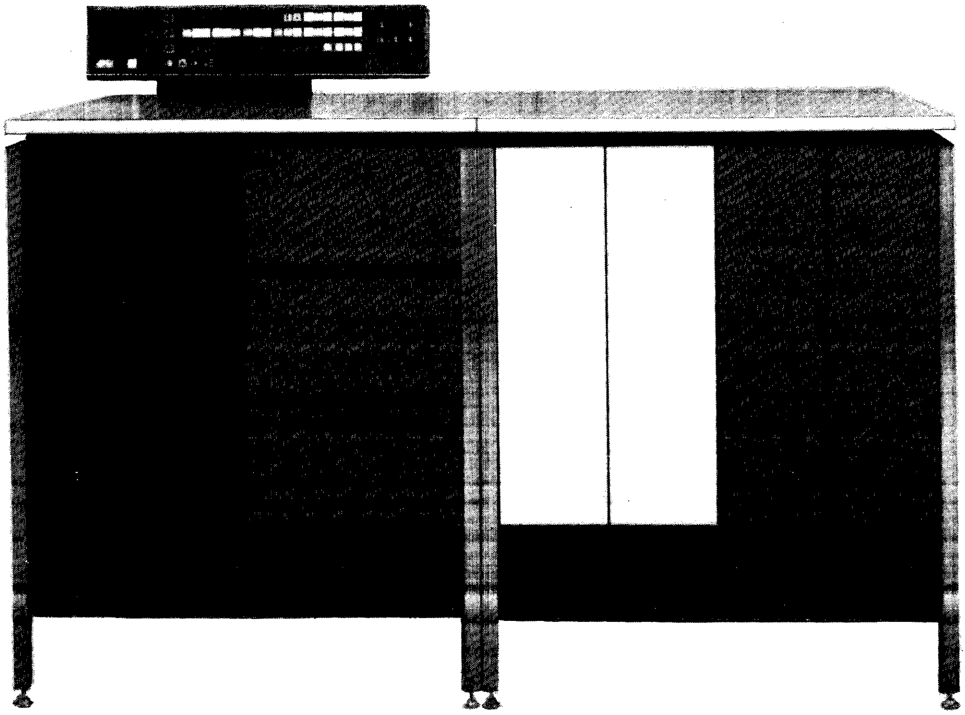
LOGIC TRAINING MANUAL

CP

EDUCATION
DEPARTMENT

HONEYWELL ELECTRONIC DATA PROCESSING

201 CENTRAL PROCESSOR



This manual is designed for Field Service training use. As such, no guarantee as to revision status can be made. The information contained in this publication is classified as Proprietary with Honeywell and is for the conduct of company business only. Accordingly, the contents thereof shall not be disclosed or reproduced in whole or in part for any purpose other than as explicitly authorized in writing by Honeywe ll, Inc.

by
Honeywell
ELECTRONIC DATA PROCESSING
EDUCATION DEPARTMENT

ETM/200-0
FIFTH PRINTING, AUG. 1965

TABLE OF CONTENTS

SECTION 1
INTRODUCTION

THE PURPOSE OF THIS MANUAL

However reliable and efficient a computer system is designed, an essential contributing factor to customer satisfaction and future sales is the efficiency and dependability of the service engineers who represent the manufacturer. The objective of this manual, therefore, is to present to the student the concepts and techniques used in the logical design of the 200 computer system so that he will be capable of performing maintenance in a logical and expedient manner.

SCOPE

This introductory chapter is intended to furnish the student with an insight into the basic computer organization and function of each of the main elements of the central processor. A detailed logic and circuit analysis of each of the elements will be discussed in the subsequent chapters with comprehensive explainations of the execution of all instructions.

It is assumed that the student is already familar with the instruction repertoire and data format as presented in the 200 programming course.

This publication is for educational purposes only and is therefore intended to present concepts rather than specifications. Field Service and Engineering department documents should be referred to for actual performance specifications.

ORDER EXTRACTION (FETCHING) CYCLES

- OP CODE EXTRACTION
- A ADDRESS EXTRACTION
- B ADDRESS EXTRACTION
- VARIANT EXTRACTION

NOTE: NUMBERS ARE KEYED TO PARAGRAPHS IN TEXT. Ⓧ

EXAMPLE: ADD/200/300
IN LOCATIONS: 100 ———— 106

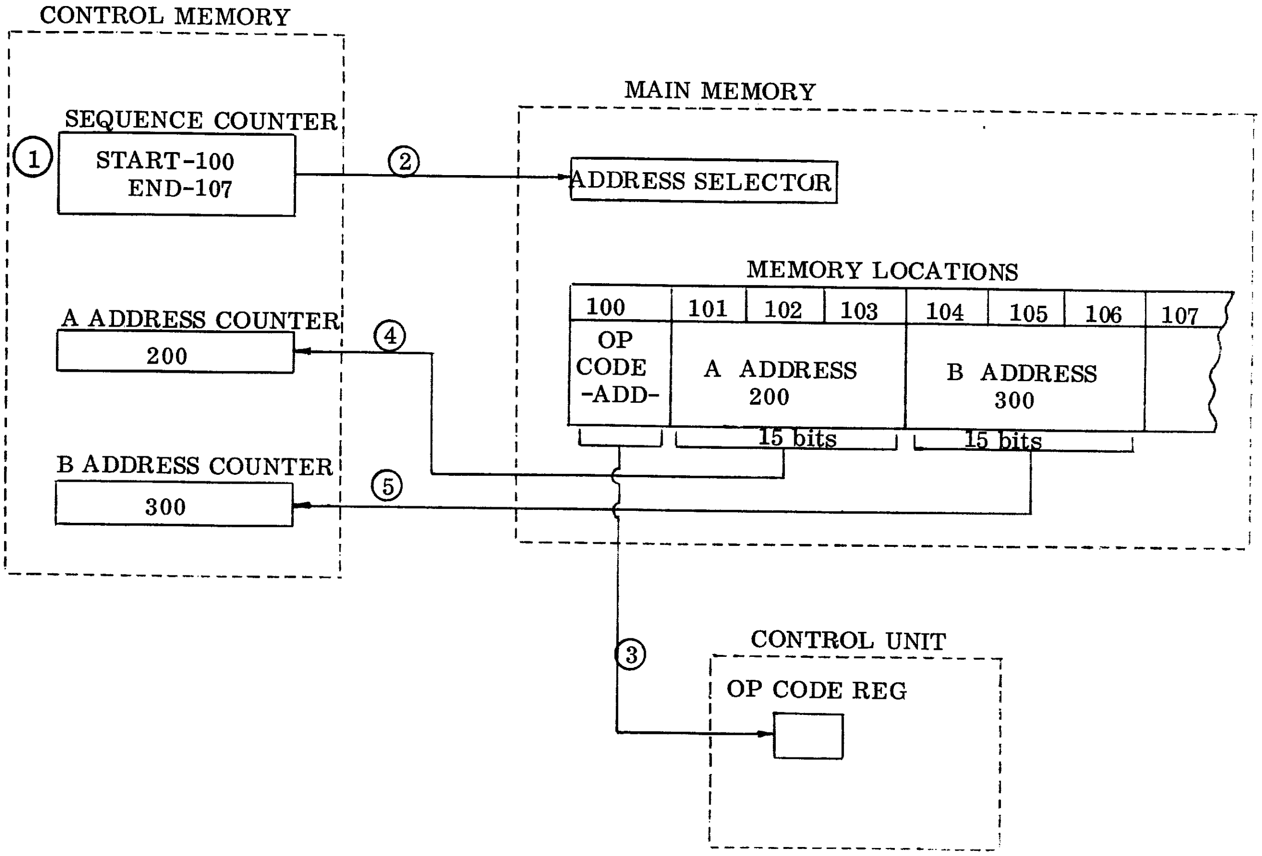


Figure 1-1 ORDER EXTRACTION

BLOCK DIAGRAM ANALYSIS

To understand the machine organization the student must first understand the sequence of events necessary to execute an instruction. Once this sequence is understood, then it is not difficult to determine the logical design necessary to implement the sequence.

The extraction phase ("fetching" cycles) of a typical instruction is represented in figure 1•1. The sequence is outlined below.

1. The sequence counter (in control memory) generally contains the address of the instruction to be executed. The address of the operation code of the first instruction of a program can be written into the control memory manually from the control panel. After the machine is started, however, the sequence counter is modified by the logic so as to contain the address of the succeeding instruction to be executed.

2. The address in the sequence counter is read from control memory and transferred to the main memory where it is used to address the location of operation code of the instruction.

   It must be remembered each main memory location can contain one character and that most instructions require several characters to define an operation code, A and B addresses, and variants.

3. The operation code is read from main memory and transferred to the control unit for storage and interpretation. It will remain there until the succeeding operation code is extracted.

The requirement of addressing several consecutive memory locations to extract the A and B address characters should make it apparent that the sequence counter will have to be incremented after each character is read from memory. It follows therefore, that the sequence counter will contain the main memory address of the operation code of the next instruction after the last character of the current instruction word is read from main memory.

4. As each character of the A address (two or three characters depending on the addressing mode) is read from main memory, it is transferred to the A address counter in control memory.

   After the subsequent indirect or indexed addressing cycles have been performed, the A address counter will always contain the effective A address to be used in the instruction.

5. The B-address is processed in exactly the same manner with the exception that the effective B address to be used in the instruction is written into the B address counter in control memory.

6. The termination of the extraction process is signaled by a word mark in the location of the next operation code. At this point the use of the sequence counter is discontinued until the start of the next instruction extraction.

The execution phase is represented in figure 1•2. Before the actual sequence is discussed there are several significant facts which should be emphasized. In accordance with the economy of design, the capacity for operand storage outside of main memory is limited to two 6-bit character buffers in the arithmetic unit. Because the arithmetic unit is used as a transfer path for all instructions (except peripheral) the two buffers are referred to as the A and B transfer registers. In view of the limited storage capacity, it is readily apparent that all operations which involve more than one character operands will have to be executed one character position (a pair of operand characters) at a time. The procedure therefore, will have to be as explained below.

7. In the case of an arithmetic operation, the technique will generally be that of extracting operand characters from the memory locations specified by the A address and B address counters.
8. The characters are transferred to the A and B transfer registers where they can be added or subtracted in the accumulator.
9. The result is then transferred to the location specified by the B address counter.

The aforementioned process is the procedure for one character operands. However, it is obvious that in most cases each operand will be several characters long. If it is considered how an addition or subtraction is performed it becomes fairly obvious how it must be implemented.

For example, it is apparent that the reason that the low order characters are specified in the instruction word is that the carries or borrows from one character position to the next must result from a previous addition or subtraction of a lower order character position. It can also be assumed that if the sequence counter can be incremented to extract characters from ascending locations in memory, then the A and B address counters can be decremented to extract operand characters from descending locations in memory using essentially the same logic circuits.

The implementation of the transfer of a character from the location specified by the A address counter to the location specified by the B address counter is not difficult to understand if it is considered that the A operand added to nothing (0) is unchanged and that the result from the accumulator is delivered to the memory location addressed by the B address counter.

ORDER EXECUTION
- OPERAND EXTRACTION
- RESULT DELIVERY

EXAMPLE: ADD/200/300

NOTE: NUMBERS ARE KEYED TO PARAGRAPH NUMBERS IN TEXT Ⓧ

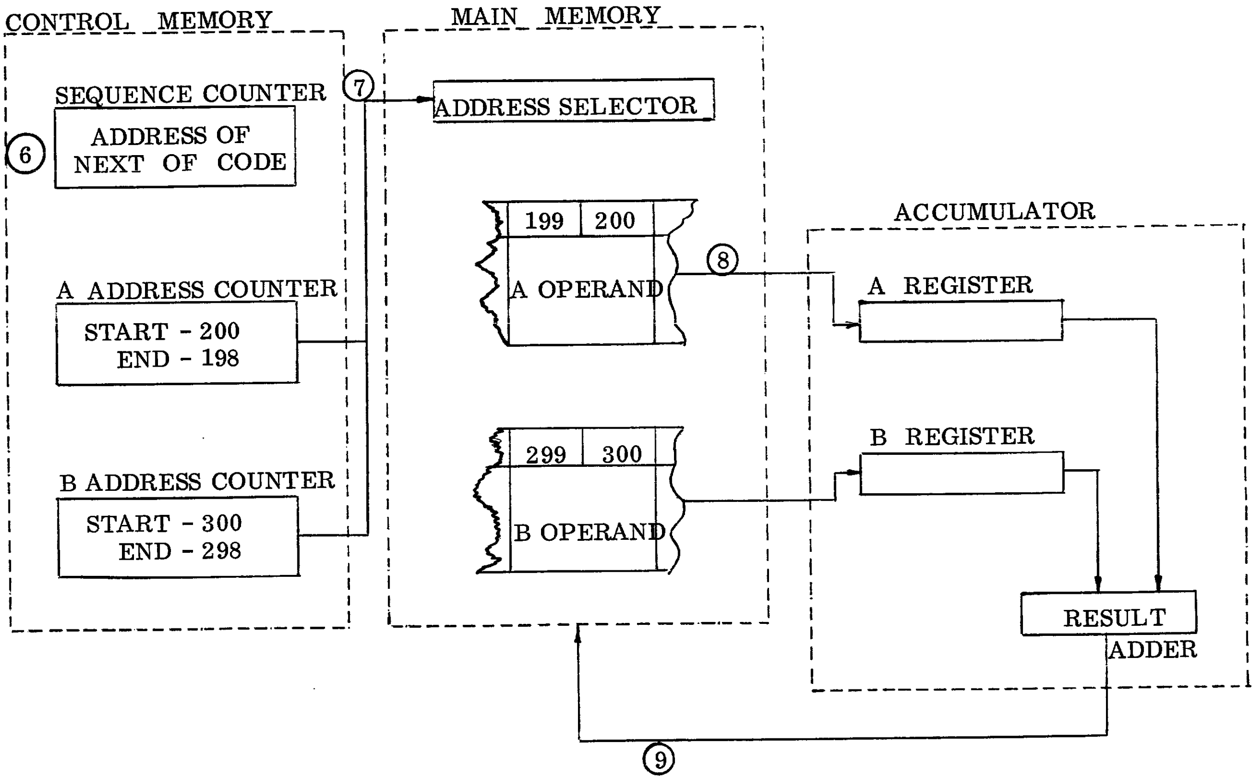


Figure 1-2 ORDER EXECUTION

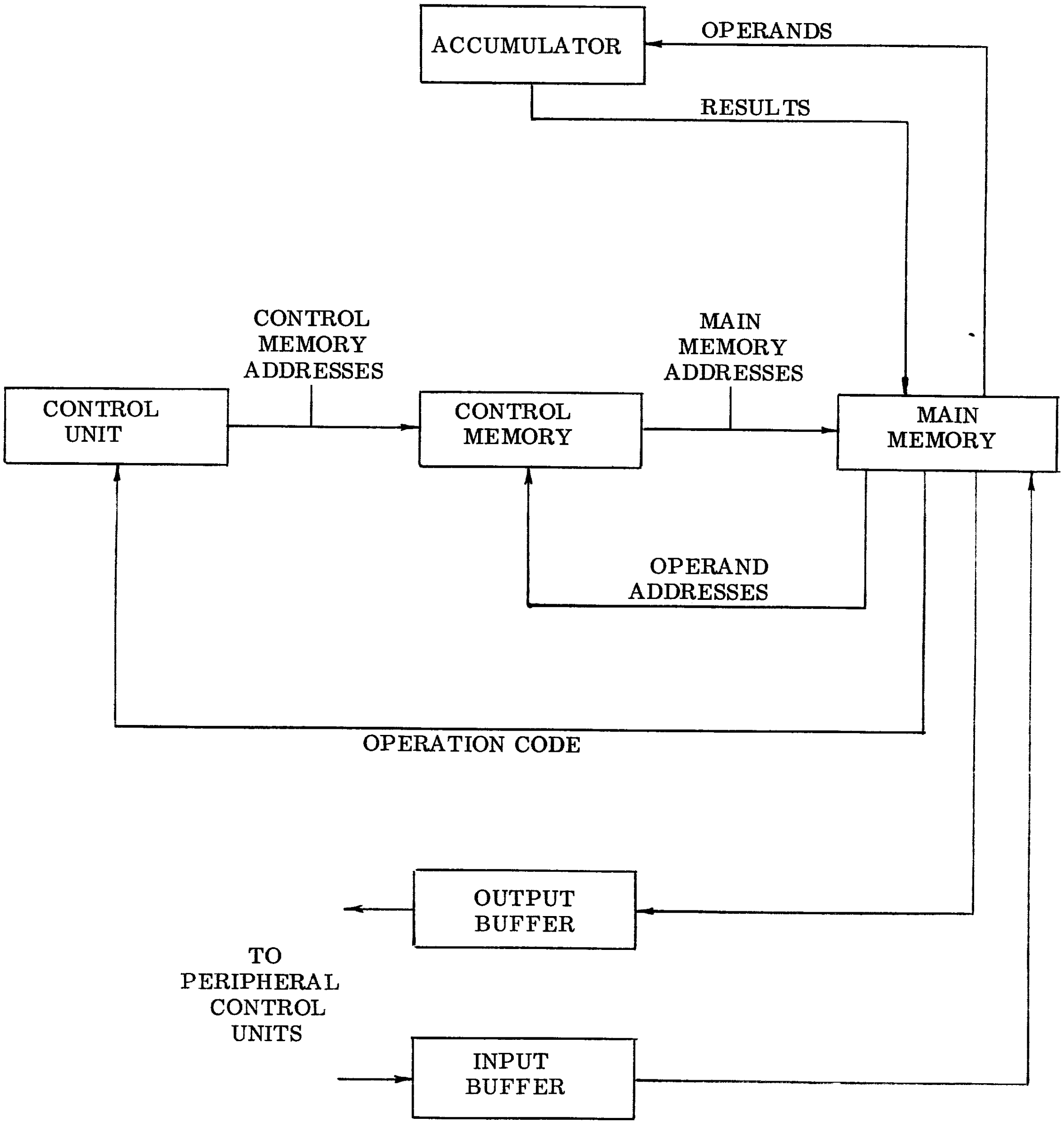


Figure 1-3, BLOCK DIAGRAM

In summary, the functions of each of the primary elements are:

Control Unit -

a. Store and interpret operation codes to

b. control sequencing of execution cycles so that

c. addresses of counters in control memory are generated in each cycle and

d. subcommands are generated to control all internal transfers of information.

Control Memory -

a. Consists of a series of counters and registers which contain 15 bit main memory addresses among which are;

b. sequence counter which is used for instruction extraction-ascending locations,

c. A and B address counters which are used for operand extraction-descending locations, and

d. associated logic for incrementing or decrementing any address which is being returned to control memory after addressing main memory.

Main Memory -

a. Stores instructions and operands.

b. Usually addressed by a 15 bit address from control memory to deliver operands one character at a time to the arithmetic unit or store a result character from the adder.

Accumulator -

a. Consists of the A and B transfer registers and an adder.

b. The adder is capable of delivering binary or decimal results and storing carries as a result of adding the contents of the A and B registers.

c. The result from the accumulator can be delivered to the main memory or to the control memory for address storage.

Input and Output Buffers -

a. The P and F buffers are used for input and output transfers respectively.

b. When a data transfer is requested by a peripheral control unit, the proper read write counter in control memory is used to address main memory so that the transfer can be performed.

GEOGRAPHY

Each pin in the 200 is defined by a seven character code. The example shown in figure 1-4 illustrates the relationship of one pin on one package to the rest of the 200 CP. The entire central processor is packaged in one cabinet containing four drawers. The control unit is in one drawer, the arithmetic unit and registers are in another drawer, 16K of main memory is in one drawer. If an addition 16K of main memory is included it would be the fourth drawer of the CP. A peripheral control unit usually is in one drawer by itself.

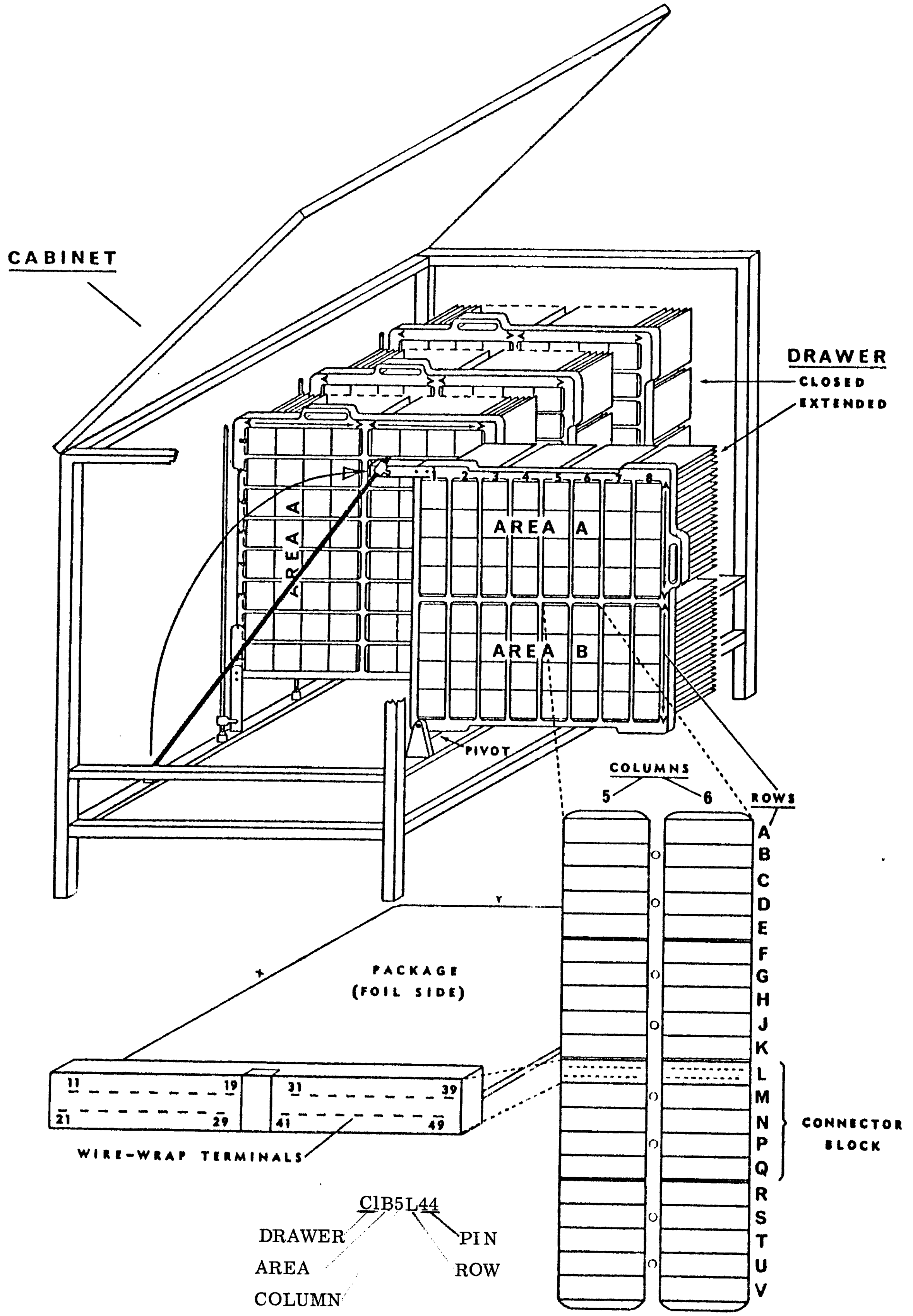


Figure 1-4, GEOGRAPHY

SECTION II
H-200 TIMING

Studying specialized computer logic is similar to solving puzzles involving the use of deductive rather than intuitive powers. A particular operation is described by the condition of a number of functions at some unique time. The condition of those functions of course depends on the conditions of still more functions at some unique time. These "times" then are used to control the sequence of machine operations. The ability to recognize these timing functions and when they occur is necessary to determine when one of the logical functions will be active.

There are more than 35 active assertion outputs from the clock (general term given to that section of the machine whose purpose is to generate "clocking" or timing signals). This doesn't really complicate recognition of timing functions, since the outputs are mnemonically classified to origin and use. Further "time one" of one classification occurs at the same absolute time as "time one" of all other classifications. And since there are only eight sequential "times" per clock cycle, recognition is no problem.

The primary purpose of this chapter is to describe the derivation of the clocking signals, but the text will detour now and then to discuss the reasons for the clock's configuration. An attempt will also be made to relate specific times to important machine functions and areas, if these attempts do not cloud this chapter's primary purpose.

CLOCK CYCLE

The H-200 clock must provide a cyclic series of eight pulses, each cycle recurring every two microseconds. The reason for this configuration and cycle time is graphically shown in Fig. 2-1, and explained in the following text.

PULSE PERIODS	1	2	3	4	5	6	7	8
MAIN MEMORY CYCLE-2us		READ				WRITE		
CONTROL MEMORY CYCLES	W	R	W	R	W	R	W	R
—	.5 usec	ONE CYCLE						

FIG. 2-1

The H-200 main memory is continuously being cycled, i.e. memory read-out, and write-in (one memory cycle) occurs all the time the machine is running. Because of this, all data transfers to and from memory to and from storage registers, and result delivery from the accumulator, must be kept in step with the main memory cycle. This fact then, dictates that the clock cycle must be the same duration as is required for one memory cycle, i.e. 2 usec. During one main memory cycle, the control memory (which is used to store Instruction and Operand Addresses) must cycle four times to address main memory and to update the pertinent stored addresses in preparation for the next main memory cycle. Each control memory cycle is divided into read and write times, thereby requiring the clock to generate eight timing pulses during each clock cycle. With all of this inter-relation it should be stressed that nothing happens without the clock. It is used to synchronize main memory, control memory, internal, and peripheral transfer, and processing operations.

Basic Block Diagram of the Clock (Fig. 2-2)

The time base used to develop the clocking signals is a crystal controlled sine wave oscillator. A pulse shaper squares the amplified sine wave, and feeds its output to an eight period ring counter to provide the required cyclic series of eight timing pulses. The machine actually has two ring counters, one in each central processor drawer, but they are inter-connected so that both remain in step.

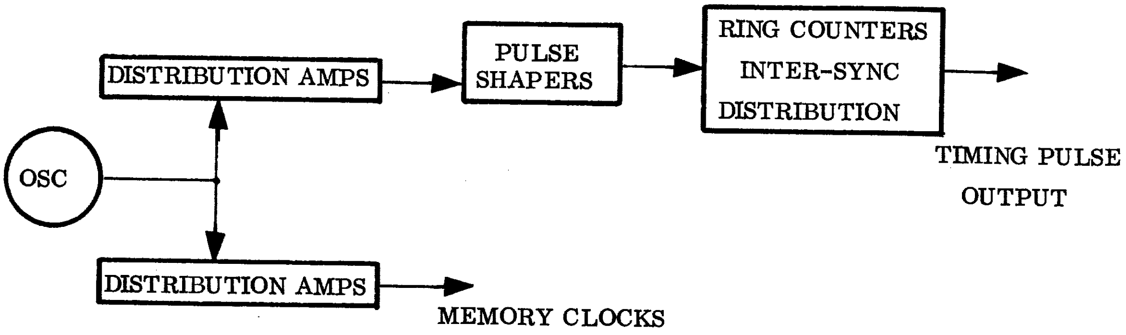


Figure 2-2

The ring counters which generate the timing signals are connected so that a "count" or pulse has two requirements. The first is a function of the previous count, and the second is the squared sine wave which determines the leading edge of the new count. This action continues until time "eight" is achieved. The "eight" is used as the previous count requirements for time "one", thus completing the ring. The result is a counter which develops eight sequential pulses, each pulse having a nominal width of 300 nsec. and recurring every 2 usec.

Fig. 2-3 is a graphic representation of both c.p. (central processor) ring counters, mutual interconnection, and the starting gate. Also shown is the connection to the main memory clock, which is not a ring counter but will have similar outputs (though not as many). The memory clock uses the outputs of the ring counter in drawer #1 for sync. Functions with a CA prefix are in the C1 drawer, while CB indicates C2 drawer, and MA the M1 drawer. Note that the function name CAT0 * 10 refers to a timing pulse derived by the ring counter in C1,drawer and * corresponds to the specific timing pulse. The 2mc square wave which will synch each "time" is not shown, but the "previous count" requirement is represented by the connecting arrows.

To start the clock, the control panel master clear button is punched forcing CPMCL10 high. CAT6I00 will be high if the clock is stoppped, or if it is running, during times "four" and "five" only. With both legs high the previous count condition for time 6 is satisfied and the clock is started. Note that CBT0710, and CAT0510 have their "previous time" requirement satisfied by a previous time in another drawer. This insures that both ring counters are in step. Further insurance that both counters remain in step is accomplished by allowing time 6 only during the time that CAT6I00 is high.

Sine Wave Generation and Distribution

The time base for the 200 is a 2MC, crystal controlled, amplitude stabilized oscillator. The sine

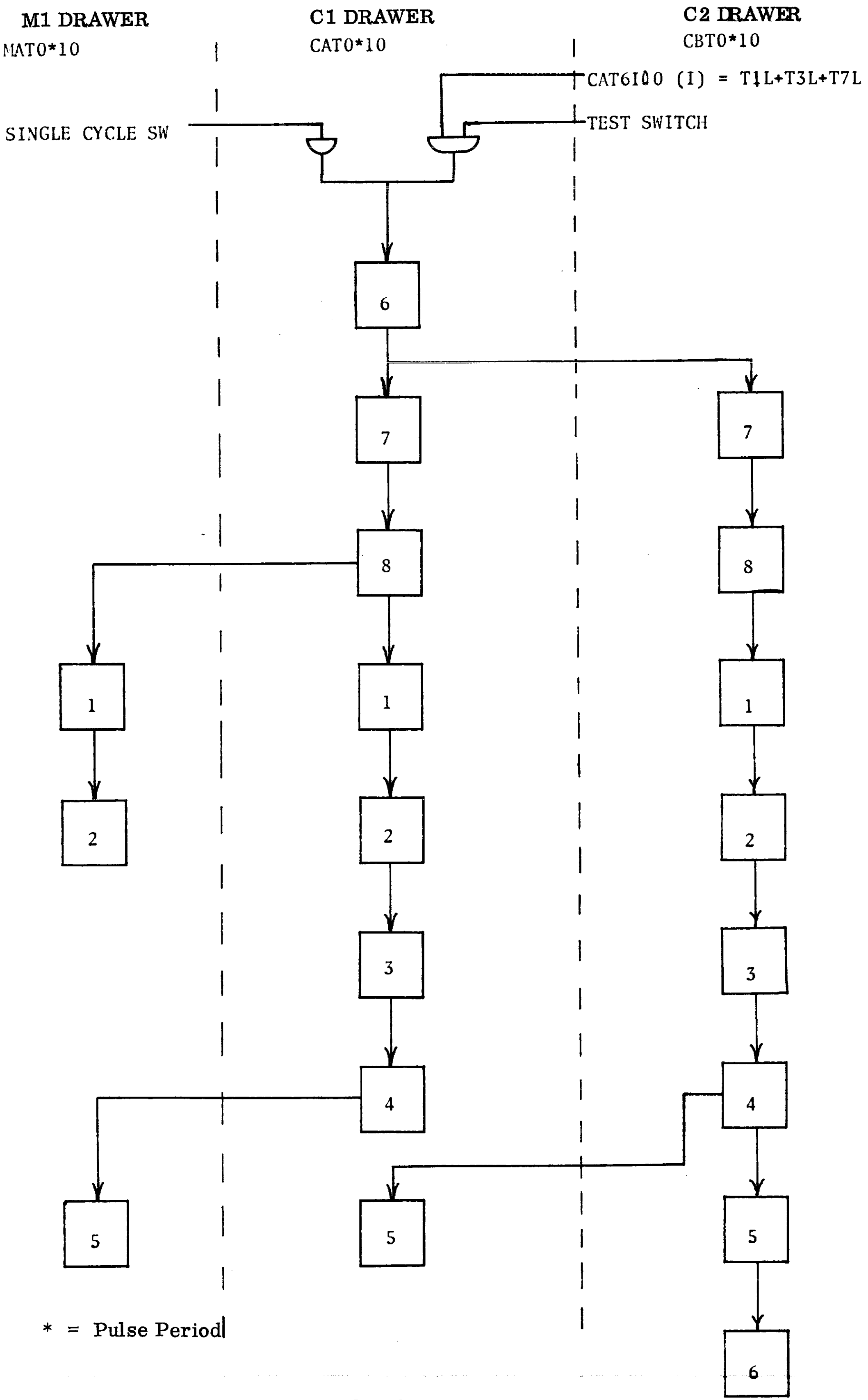

M1 DRAWER
MAT0*10
C1 DRAWER
CAT0*10
C2 DRAWER
CBT0*10
CAT6I00 (I) = T1L+T3L+T7L
SINGLE CYCLE SW
TEST SWITCH
6
7
7
8
8
1
1
1
2
2
2
3
3
4
4
5
5
5
6
* = Pulse Period

Fig. 2-3

wave is brought out through two emitter follower buffers. A special test point is brought out to a pin and is used as to reference all cp timing adjustments. See Fig. 2-6 for the oscillator package schematic. The sine wave outputs from the oscillator feed two KDA packages. Each KDA (clock distribution amplifier) consists of four class A emitter followers. See Fig. 2-8 for the schematic of the KDA package. The above distribution is shown on Fig. 2-4 in block diagram form.

CP Clock

Fig. 2-4 is a block diagram of the ring counter in drawer C1, and is the same as that represented in Fig. 2-3 albeit more detailed. The block labeled KPS (clock pulse shaper) is one of five circuits on the 2KTG5 package (see Fig. 2-7), and its purpose is to clip the incoming square wave. By varying the clipping level,the active duration of the output, and consequently the "times" which it synchronizes, can be adjusted. All timing functions in the c.p. are aligned so that the 2.5v level of the positive going leading edge coincides with the zero crossing point of the sine wave as monitored at the oscillator package test point. The maximum rise time for timing functions is 50 nsec,and the adjustment tolerance is ±20 nsec.

The KPS circuit incorporates an input transformer, and switching input connections provide an easy means of phase reversal. Refering to the circuit schematic (Fig. 2-7) will show that if pin 18 is grounded, the output at pin 24 will be a square wave of the same polarity as the sine wave input to pin 19. By grounding pin 19 the output polarity will be the inverse of the sine wave input at pin 18. In other words, a variable positive pulse is obtained from the same type of circuit for both halves of the input sine wave just by reversing the input connections. In general, the KPS-circuits with pin 18 grounded, will provide the "even" times. Conversly those KPS circuits with pin 19 grounded will provide the "odd" times.

Refer again to Fig. 2-4, noting the flow for the time "one". The source for all cp timing signals is the oscillator; distribution and amplification of the sine wave time base is the function of the KDA's; squaring the sine wave and positioning the output timing pulse is the job of the KPS circuit. The individual timing pulses are formed by a previous time (CAT0810 for time one) and synch from the oscillator (CAT1010 for time one). Each pulse period is defined by two outputs,one of which is present every 2 usec so long as the clock is running (CAT0110). The other is interrupted by a peripheral buffer cycle($\overline{\text{CACPC30)}}$ and is designated "cp times" (CACT110). The following discussion will develop the two outputs of time one.

CAT1000 = (KPS) = MA0SA1A
Pin 19 ground

CAT1L10 = CAT1000•CAT0810
+ CAT1010•CAT1L10

CAT1010 = (I) = CAT1000

CACT110 = CAT1010•CAT1L10•CACPC30

CAT0110 = CAT1010•CAT1L10

Function CAT1000 is the output of a KPS circuit, and pin 19 being grounded determines that the output polarity is opposite that of the sine wave input. Refer to timing diagram Fig. 2-5. The first two characters of the function indicate the area of the machine and the drawer (as previously mentioned). The first square wave shown is equivalent to all odd negation outputs of the KPS circuits. The second square wave shown is equivalent to all even negation outputs of the KPS circuits. But note that CAT1010 is the output of an inverter and therefore can be shown as the equivalent of T2E00. The reverse (concerning T2E10 and T1000) is also true. This is logical since, if time even not were at +5v (indicating truth) then time odd assertion would be true also.

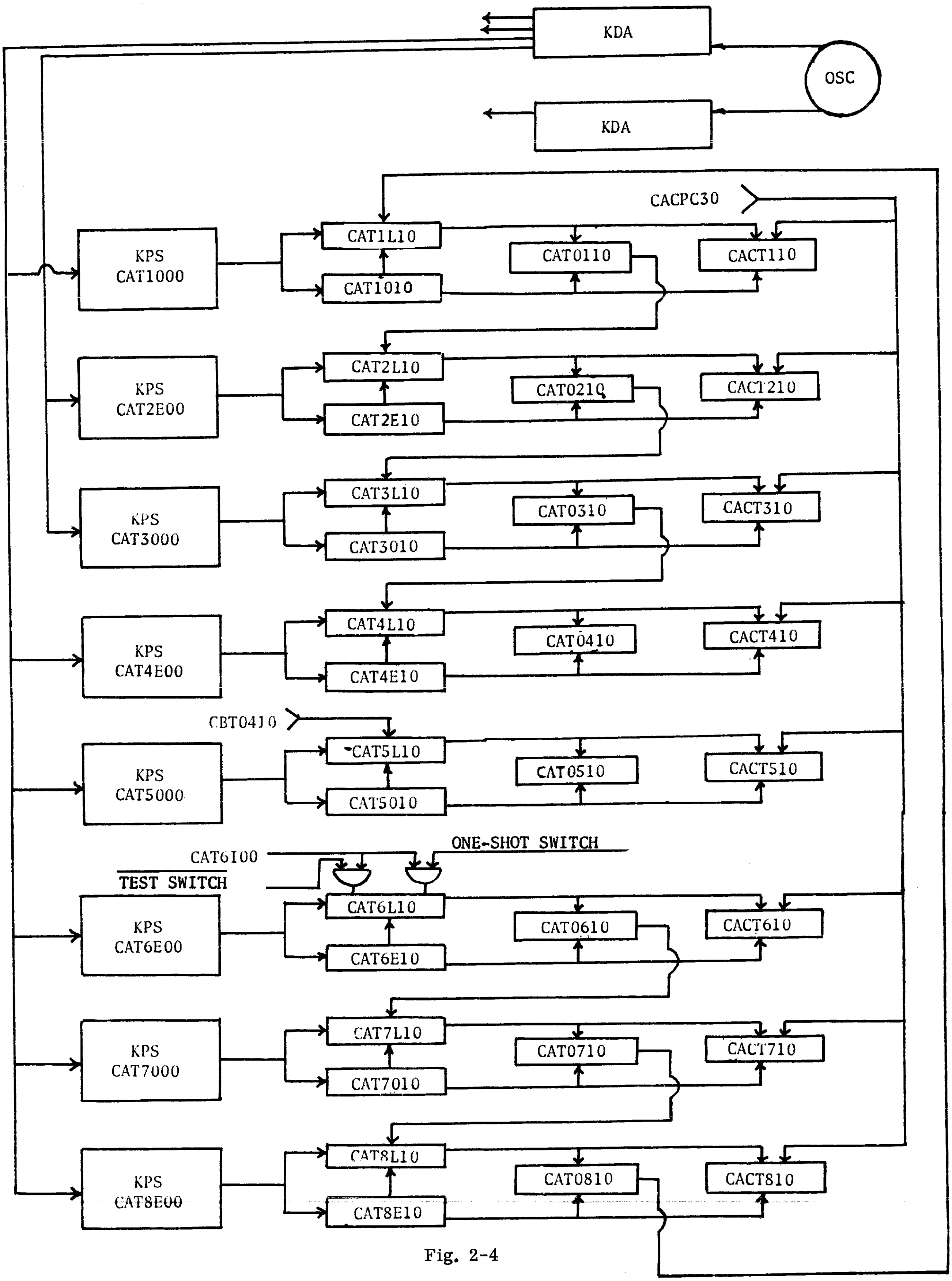

KDA
OSC
KDA
CACPC30
KPS CAT1000
CAT1L10
CAT1010
CAT0110
CACT110
KPS CAT2E00
CAT2L10
CAT2E10
CAT0210
CACT210
KPS CAT3000
CAT3L10
CAT3010
CAT0310
CACT310
KPS CAT4E00
CAT4L10
CAT4E10
CAT0410
CACT410
CBT0410
KPS CAT5000
CAT5L10
CAT5010
CAT0510
CACT510
CAT6I00
ONE-SHOT SWITCH
TEST SWITCH
KPS CAT6E00
CAT6L10
CAT6E10
CAT0610
CACT610
KPS CAT7000
CAT7L10
CAT7010
CAT0710
CACT710
KPS CAT8E00
CAT8L10
CAT8E10
CAT0810
CACT810

Fig. 2-4

CAT1L10, by the stated boolean expression, requires T1000 and CAT0810. Note from the timing chart that T08 and $\overline{\text{T10}}$ are coincidcnt with TIL and provide the "previous time" and "synch" requirements. TIL has another gate (recirculation gate) whose purpose is to hold TIL set after the normal "set gate" has gone low. But notice that the recirculation function (CAT1010) is the assertion of one leg of the set gate (CAT1000). The student might assume that this would cause a certain amount of instability, since $\overline{\text{T10}}$ must go low before T10 can come high. But what is not shown in a boolean expression is the physical consideration of rise and decay time. T10 is the output of the GIN circuit of the 2KTG5 (fast inverter) while TIL is the output of GBA circuit of the 2KTG5 (slow amplifier). This indicates that T10 will be high before TIL can go low, thus keeping a high input gate.

With TIL and T10 high the conditions for CAT0110 are satisfied. This is the "time one" output of the clock. TIL and T10 along with CACPC30 will satisfy the input gate for CACT110. This time is the same as T01, except for the additional requirement of CPC which dictates that CT1 (and all other CT* times) will be inhibited if CPC is low.

The function CACPC30 is forced low when there is a peripheral demand (buffer cycle). This occurs when a peripheral device is ready to communicate with main memory and traffic control allocates 2 usec to the read-write channel the device control unit is using. The inhibiting of the CT* times allows the interuption of a central processor instruction for two microseconds (while main memory is at the disposal of the peripheral control unit). This interuption is really just a suspension of certain central processor activities such as transferring or resetting certain registers. Naturally the gating of the peripheral data is by the T0* times.

TIL is recirculating on T10, and both TIL and T10 are necessary for T01, and CT1. So when T10 goes low the functions it feeds will also go low. It should be pointed out, however, that T01 provided the previous time input for T2L thus pepetuating the clock.

That in essence is how the ring counters, which make up the clock, function. But one assumption was made in the previous discussion : the fact that T08 was present to set TIL. If the clock was stopped it would be necessary to backtrack to find a way to artifically inject a previous time to start the clock. From Fig. 2-3, it can be seen that time six's previous time requirement can be satisfied by CPMCL10 and CAT6I00. Refering back to Fig. 2-5, MCL is shown coming high with the clear button, which in fact it does.

$$\text{CAT6I00} = (\text{I}) = \text{T1L} + \text{T3L} + \text{T7L}$$

$\overline{\text{T6I}}$ will be high if the clock is stopped, since no input gate can be satisfied. Thus time 6 and the clock can be initiated with the clear button. Note however, that T7L will force $\overline{\text{T6I}}$ low, and it will not come high again until the end of T3L thus insuring against spurious triggering of T6L.

Summary

There are ring counters in both central processor drawers, interlocked so that they remain in step.

The output of the clock is a train of eight pulses repeated every two microseconds. All times are synchronized by a 2 mc crystal controlled oscillator.

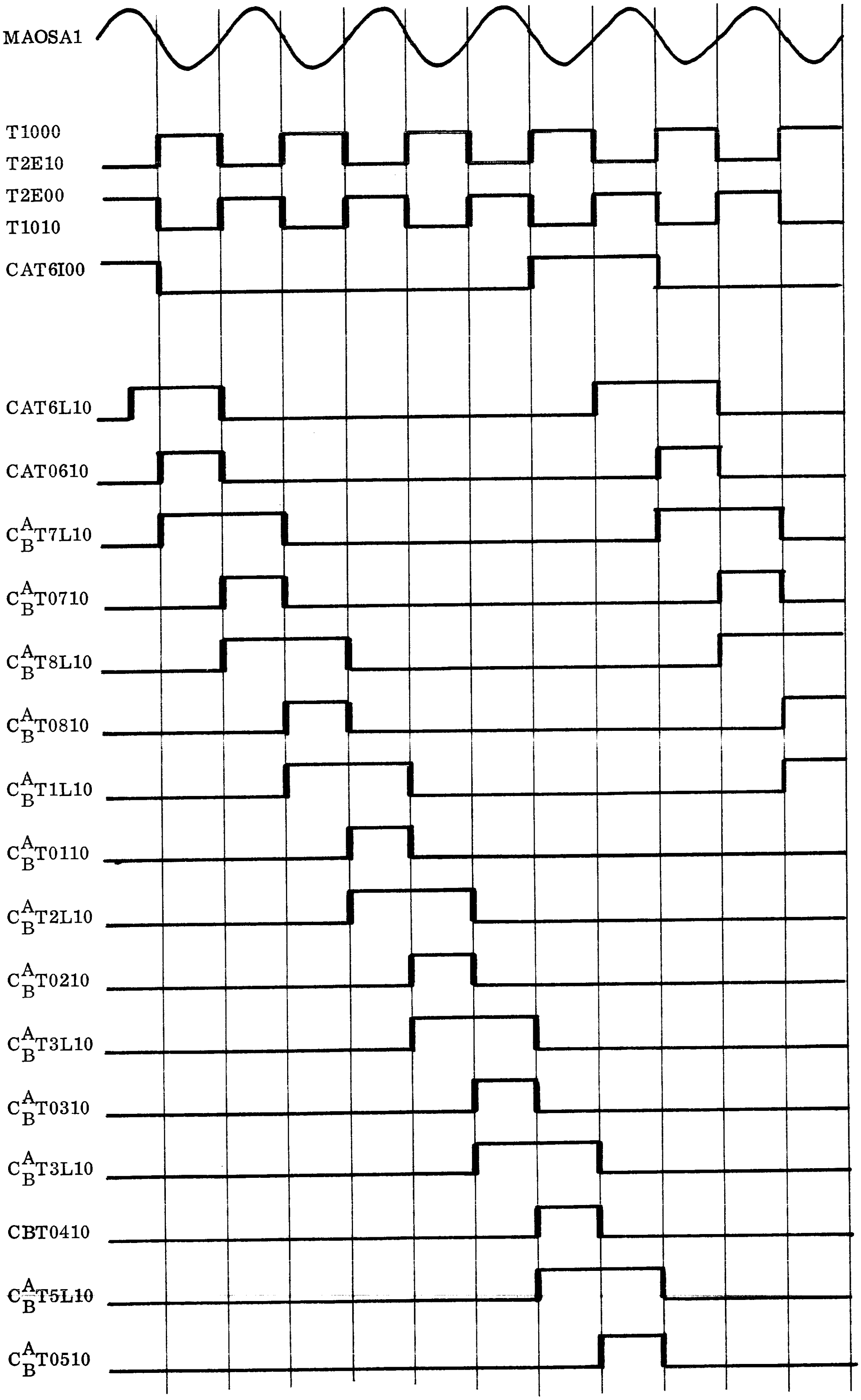
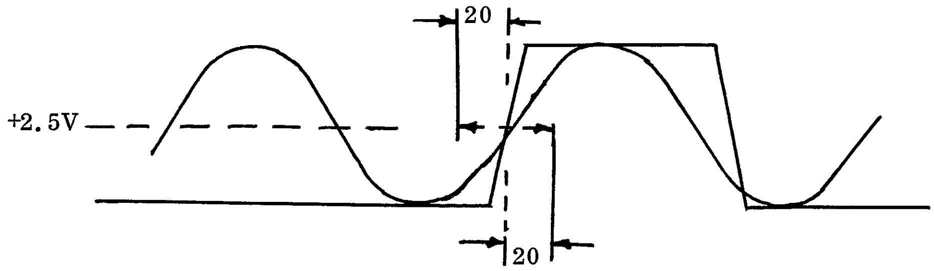

MAOSA1
T1000
T2E10
T2E00
T1010
CAT6I00
CAT6L10
CAT0610
$C^{A}_{B}T7L10$
$C^{A}_{B}T0710$
$C^{A}_{B}T8L10$
$C^{A}_{B}T0810$
$C^{A}_{B}T1L10$
$C^{A}_{B}T0110$
$C^{A}_{B}T2L10$
$C^{A}_{B}T0210$
$C^{A}_{B}T3L10$
$C^{A}_{B}T0310$
$C^{A}_{B}T3L10$
CBT0410
$C^{A}_{B}T5L10$
$C^{A}_{B}T0510$

Figure 2-5

Distribution of clock timing to certain areas of the machine is interupted during a peripheral buffer cycle for the purpose of suspending a central processor instruction for one memory cycle.



The clock should be adjusted so that at the 2.5 volt level, the leading edge of the KPS is within ± 20 nsec of the crossover point of the oscillator. This adjustment is made by syncing on the previous time and observing the output of the oscillator in the memory drawer and the KPS to be adjusted.

Clock outputs, as has already been mentioned, are indentified, as to drawer, by the second letter of the function and, as to use, by the three letter mnemonic. The two character suffix of the function has quite a bit to tell also. 10, 50, 1A, 5A, 1B, and 5B are assertions. 00, 40, 0A, 0B, and 4B are negations. The last character describes the derivation of the signal. 0 , A, and B represent three different pulse shapers for the same logical time.

Two special times are generated: time two not and time five not. These functions are separately adjustable because they have their own pulse shapers and are not just the inverted assertion. CAT0200 and CAT050A (see also CAT050B) are the names of these special times and they are adjusted the same way as the assertions.

Two additional clock outputs, CBTEV10 and CBTOD10, are used for control memory timing. They represent even times and odd times respectively.

LOGICAL SPECIFICATIONS 2KCA2

a. Output Capabilities - Output pins 34 and 37 are each capable of driving one KDA package or 350 ohms.

b. Output Signal - A 2 mc sine wave.

NOTE

The outputs must not be grounded in normal use. However, accidental grounding will not permanently damage the circuit.

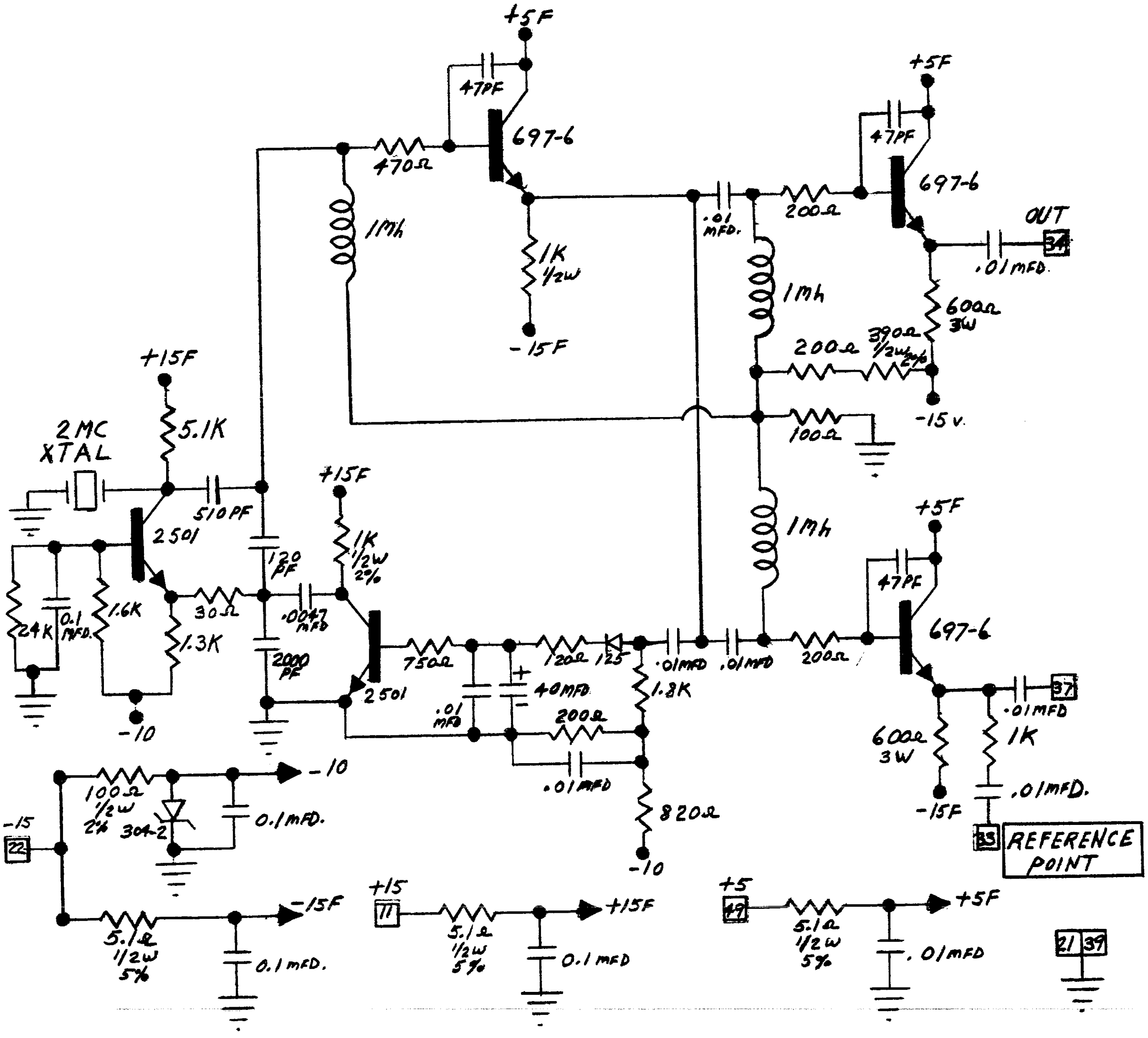
A reference point pin #33 is provided for use in alignment of the KTG package.

THEORY OF OPERATION 2KCA2

The KCA package consists of a 2 mc crystal controlled basic oscillator amplitude control circuitry and two output emitter followers.

The oscillator is the Clapp type, where the feedback is determined by a capacitive divider from collector to ground. The output of the oscillator is stepped down by the capacitive divider and fed into the buffer emitter follower which in turn drives the two output emitter followers and the amplitude stabilizing circuit.

In the amplitude stabilizing circuit, the amplitude of the positive half cycle of the sine wave is compared against a preset dc voltage; the difference between these two is integrated and fed to the base of the amplitude control transistor Q4. In normal operation Q4 is driven to near saturation presenting a low dynamic resistance between emitter and collector. Q4, in series with C14, is connected across C13, the oscillator feedback capacitor. Therefore, any variation in the oscillator output amplitude will cause a corresponding change in the dynamic resistance of Q4, affecting the oscillator feedback, thereby maintaining an output of constant amplitude.



2KCA2
FIG. 2-6

THEORY OF OPERATION 2KTG5

a. Pulse Shaper Circuit

The Pulse Shaper has a sine wave input of 500 nsec. This period is applied to input terminals 18 and 19. When the input signal becomes sufficiently positive, Q5 starts conducting which in turn cuts off Q2. The off period of Q2 is determined by the variable RC time constant. When the base of Q2 becomes sufficiently positive or the capacitor charges up towards +5 volts (Zener diode voltage), Q2 will conduct and the output pulse will be terminated. As a result, a positive output pulse of 300 nsec. variable duration is obtained with variable positioning of its trailing edge. Q5 is turned off when the input signal becomes sufficiently negative. The output pulse can be shifted by 180°, i.e., "even" or "odd" pulses can be obtained by interchanging the backboard wiring to the input terminals of the transformer (18 and 19).

b. Gate Inverter

The Gate Inverter is an inverter circuit with an AND gate at its input. When the AND gate is satisfied, (when both input terminals are positive) transistor Q1 conducts and ground level appears at the output terminal. When either of the inputs is at ground, Q1 is cut off, and the +5 voltage appears at the output terminal.

c. Gate Buffer Amplifier

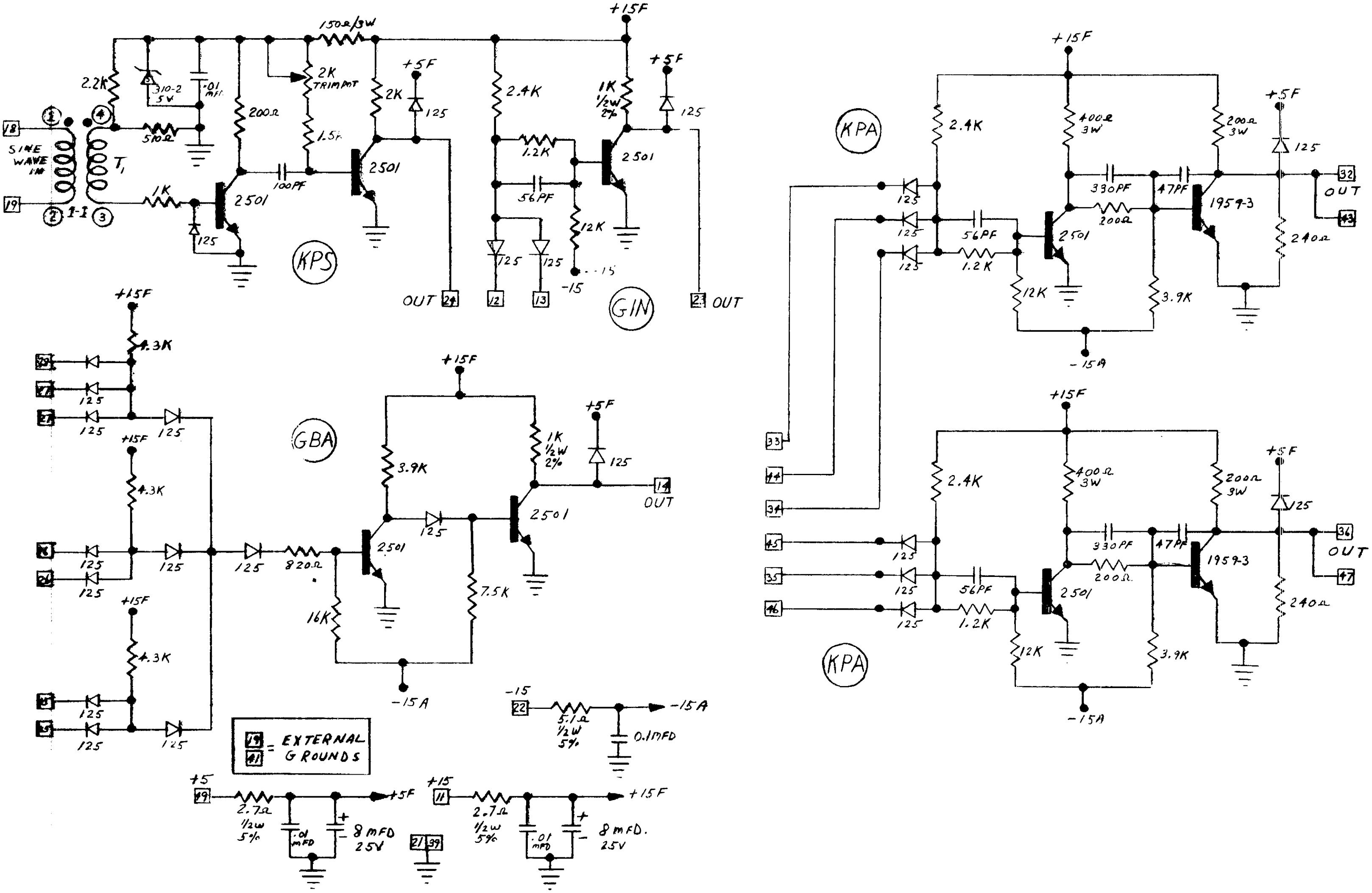
The Gate Buffer Amplifier is a two-stage amplifier preceded by three AND gates buffered together to form an OR gate. When any of the AND gates is satisfied, a positive voltage appears at the output; otherwise, the output is at ground level. If a gate is unused, one of its gate-legs must be grounded.

d. Clock Pulse Amplifier

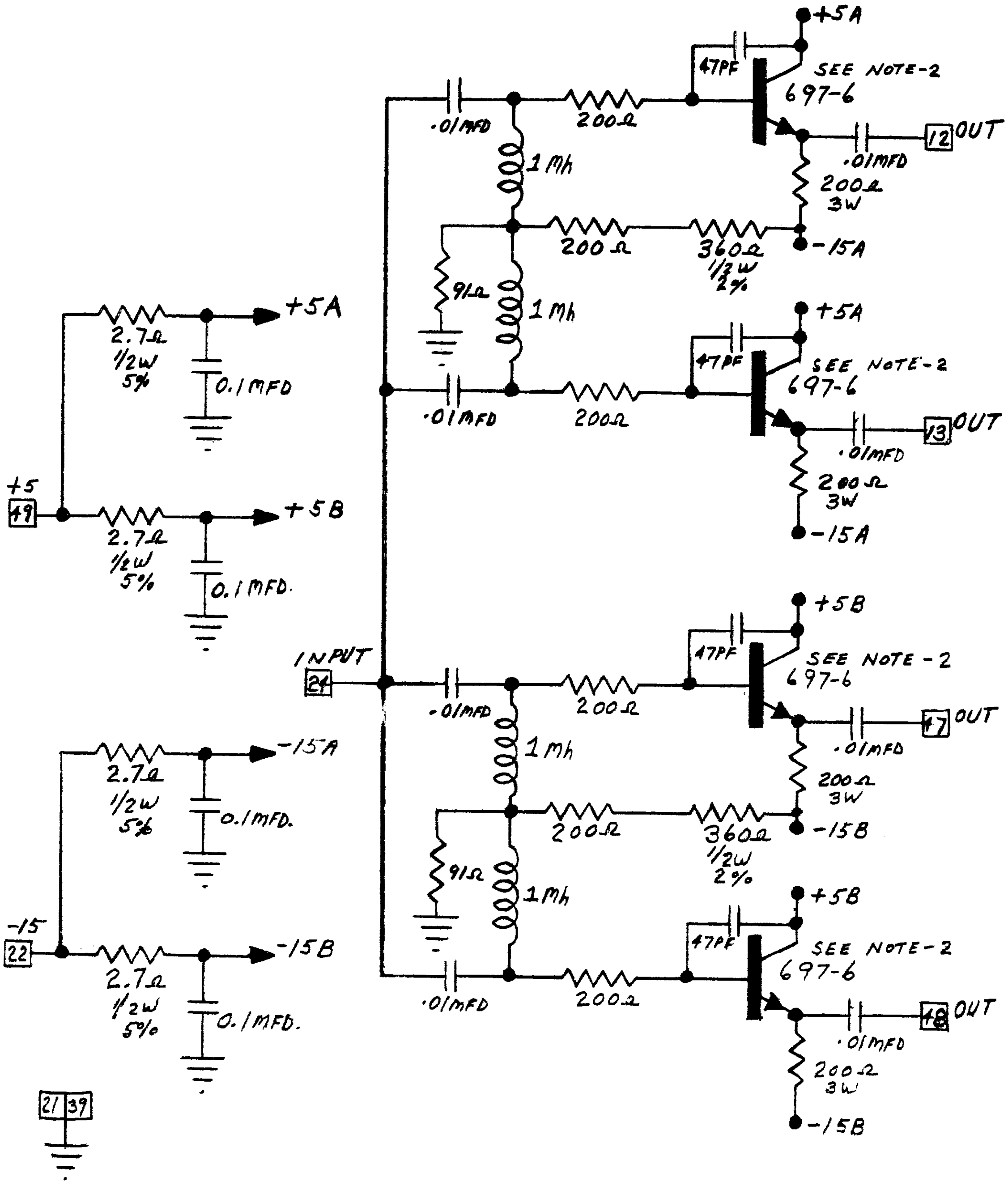
The Clock Pulse Amplifier is a two-stage power amplifier with a three-input AND gate at its input. If all three inputs are high, the output of the amplifier is high; otherwise, the output is zero.

e. Generation of the Timing Pulse

To generate the required eight timing pulses, eight GBA circuits are interconnected to form a ring counter whose outputs are gated with the even or odd pulses generated on the corresponding KTG cards. Consequently, the leading edge of each timing pulse is individually adjustable. One card is needed for the generation of each timing pulse.


+15F
150Ω/3W
2.2K
310-2 5V
.01 MF
2K TRIMPOT
1.5K
200Ω
+5F
2K
125
2.4K
1K 1/2W 2%
SINE WAVE IN
18
19
T1
1
2
3
4
1:1
510Ω
1K
2501
100PF
125
KPS
1.2K
56PF
12K
-15
GIN
OUT 24
12
13
21 OUT
KPA
2.4K
400Ω 3W
200Ω 3W
330PF
47PF
200Ω
1959-3
240Ω
56PF
1.2K
12K
3.9K
-15A
32
43
OUT
4.3K
GBA
3.9K
1K 1/2W 2%
OUT
14
7.5K
820Ω
16K
33
44
34
45
35
46
36
47
EXTERNAL GROUNDS
-15
22
5.1Ω 1/2W 5%
0.1MFD
+5
2.7Ω 1/2W 5%
.01 MFD
8 MFD 25V
+15
11
8 MFD. 25V
21
39

2KTG5
Fig. 2-7

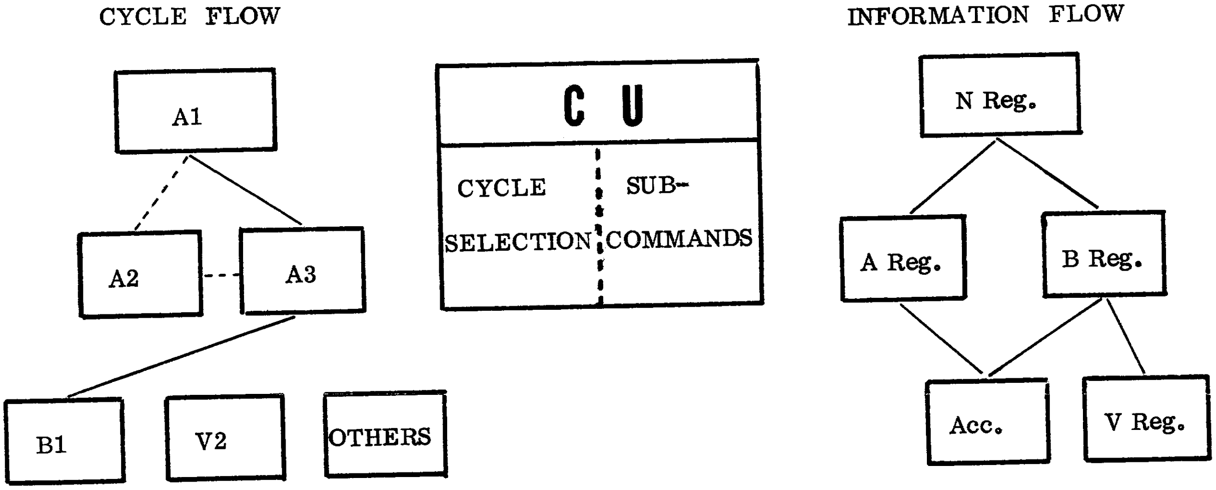

+5A
47PF
SEE NOTE-2
697-6
.01MFD
200Ω
12 OUT
.01MFD
200Ω
3W
1mh
200Ω
360Ω
1/2W
2%
-15A
91Ω
1mh
+5A
2.7Ω
1/2W
5%
0.1MFD
+5A
47PF
SEE NOTE-2
697-6
.01MFD
200Ω
13 OUT
.01MFD
200Ω
3W
-15A
+5
49
+5B
2.7Ω
1/2W
5%
0.1MFD.
+5B
47PF
SEE NOTE-2
697-6
INPUT
24
.01MFD
200Ω
47 OUT
.01MFD
200Ω
3W
1mh
-15A
2.7Ω
1/2W
5%
0.1MFD.
200Ω
360Ω
1/2W
2%
-15B
91Ω
1mh
+5B
47PF
SEE NOTE-2
697-6
-15
22
-15B
2.7Ω
1/2W
5%
0.1MFD.
.01MFD
200Ω
48 OUT
.01MFD
200Ω
3W
-15B
21
39

2KDA4
Fig. 2-8

SECTION III
CONTROL UNIT

INTRODUCTION

The control unit selects instructions, interprets them, and directs their execution as specified by the programmer. Even the simplest machine instructions cannot be performed all at once. They are divided into simpler steps, with each step performing a specific part of the overall job. The basic subdivision is a "memory cycle". The control unit selects the sequence of memory cycles and issues the subcommands that direct the flow of data within a memory cycle.



Representative Examples Of Cycle Flow And Information Gating

FIG. 3-1

MEMORY CYCLES

In the 200, a memory cycle is 2 usec in duration. This is the time required to read out a main memory location, store its contents in a group of GBA's (a register), and write the information back into the memory location where it originated. Of course, data stored in another register during some previous memory cycle may be manipulated simultaneously. During the course of a main memory cycle, control memory is cycled four times. The first control memory cycle is used to obtain the address of the main memory location to be cycled. The other three are usually used to store information that will be needed later on.

INFORMATION GATING

When it is necessary to transfer the contents of one register to another, subcommands are used to gate the information. For example, if the "N" register contained an octal 45 (CAN0610 and N03 and N01 at +5 volts, and N05, N04, and N02 at 0 volts), and we wanted to transfer the "N" register to the "I" register, we would use the subcommand CBN2I10 to deliver the information. CAI0110 would have on its input diodes a gate that has the assertion of N01 anded with the assertion of N2I. Now, every time N2I came high (+5) if N01 was a one (+5), the one bit would be set in I01. Similarly, I02 should have N02 anded with N2I:

CAI0210 = N2I•N02

Now, when N2I comes high, if N02 were a zero,I02 would not be able to set. In this manner a one will be set in the I register if the N register had a one bit in the corresponding bit position.

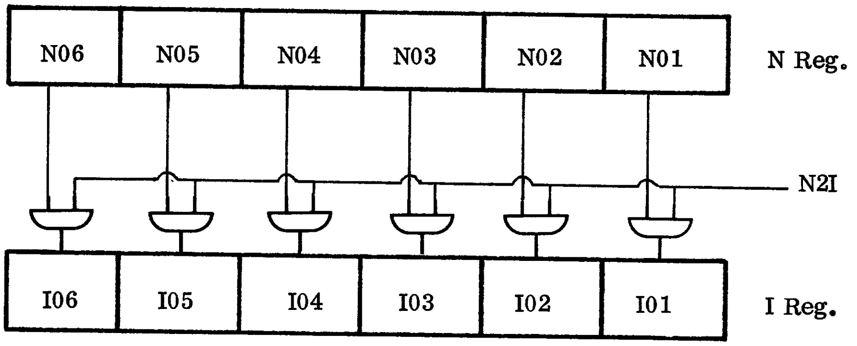


FIG. 3-2

Figure 3-3 below is a simpilified block diagram showing the relationships of the various parts of the control unit. The major inputs to the control unit are; the op code, which is stored in the I register (Instruction register), operand punctuation (usually is used to terminate the instruction), and the timing functions, which are inputs to practically everything else in the machine also.

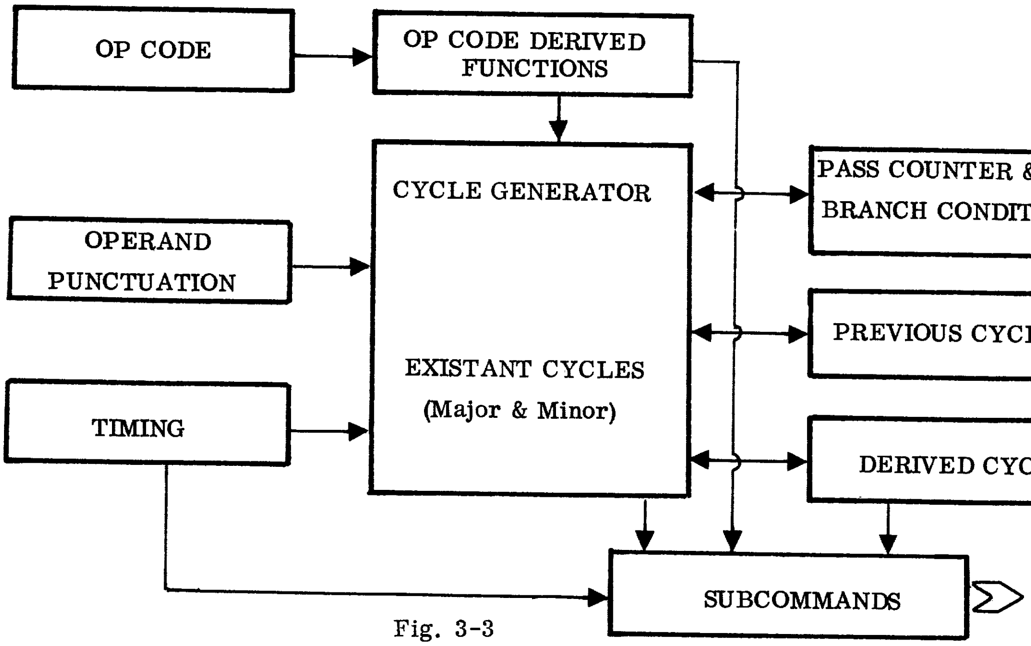

OP CODE
OP CODE DERIVED FUNCTIONS
CYCLE GENERATOR
PASS COUNTER & BRANCH CONDITIONS etc.
OPERAND PUNCTUATION
PREVIOUS CYCLES
EXISTANT CYCLES (Major & Minor)
TIMING
DERIVED CYCLES
SUBCOMMANDS

Fig. 3-3

The only thing that could properly be defined as an output of the C.U. is subcommand generation. However, cycle generation is also a major function of the C. U. It should be understood that this is a simplified diagram, and that in some cases, operand punctuation, op codes, and the cycles (previous, existant, and derived) will bypass the subcommands and be used directly.

EXTRACTION CYCLES

Memory cycles are divided into two main categories: extraction cycles and execution cycles. The extraction cycles are simply used to read the entire instruction out of memory one character at a time, using the sequence counter as a main memory address and incrementing it each time it's used. As the characters are read out, they are stored in appropriate places for use during order execution. The A and B addresses are stored in control memory and the op code is stored in the I register. The two "major cycles" used during the extraction cycles are EAC and EBC. The "minor cycles" E1C, E2C, and E3C are the basic cycles and are common to extraction and execution. During EAC the basic cycles E1C, E2C, and E3C extract the first, second, and third characters, respectively, of the A address. During EBC, the basic cycles E1C, E2C, and E3C are used to extract the three B address characters (and store them in the B address counter in control memory).

	EXTRACTION								EXECUTION						
MAJOR		EAC			EBC				EEC			EEC			
MINOR	E3C	E1C	E2C	E3C	E1C	E2C	E3C	E1C	E1C	E2C	E3C	E1C	E2C	E3C	E3C
MNEMONIC	V3	A1	A2	A3	B1	B2	B3	V1	E1	E2	E3	E1	E2	E3	V3

FIG. 3-4

EXECUTION CYCLES

The execution cycles are actually going to perform the operation requested by the programmer. For example, in an add instruction during E1C of EEC, the contents of the A address counter will be used to address main memory and the contents of that main memory location will be delivered to the A operand register and stored. In E2C of EEC the contents of the B address counter are used to address main memory, and the contents of that main memory location are sent to the B operand register. The accumulator will add the two operands. In E3C of EEC the B address counter will be used to address main memory. and deliver the result of the add. If there are more characters in the operands, repeat these cycles. Read out a second A operand character in E1C, a second B operand character in E2C, and deliver the second result character to the B address. These operations will be repeated until the defining word marks are found. The control unit will use the word marks as signals to start the extraction of the next instruction in sequence.

In general, E1C during EEC is used to read out an A operand. This cycle has been given the mnemonic name E1 (using the middle character of the major cycle and the middle character of the minor cycle). E[E]C plus E[1]C equals "E1". In other words, E1 is used to address the A operand. E2 (EEC•E2C) usually addresses the B operand, and E3 delivers the result. In some orders, such as the moves, there isn't a B operand, only an A operand and a result. In these cases, during EEC there would not be an E2C cycle. In that case the op code would cause E3C to be selected at the end of E1C, instead the usual E2C. If an address register needs to be incremented during execution cycles, the major cycle EMC will be used. By trying to use the same cycles for a similar purpose each time the C. U. can generate similar groups of subcommands based mainly on the cycle. Use of the op codes for generating subcommands provides the variations needed for particular instructions.

OP CODE DERIVED FUNCTIONS

When a group of ones and zeros in memory are to be used as an instruction, the six data bits in the low order character are used to designate the op code. For example, 100101 is the machine code for a halt

OP-CODE DERIVED FUNCTIONS

OP-CODE		OPTIONAL	SPECIFIC	GENERAL (I REG SET AT CTG)																**	
OCTAL	MNEMONIC		ONE INSTRUC. ONLY	U03	U46	BAB	BBC	BCH	CAS	DAS	DPA	LSR	MLA	MPN	POP	SPM	VAS	XOR	SAO		FLOW CHART PG. #
10	EXM	*	MUV																	C	63
14	MCW												*							C	60
15	LCA												*							C	
16	ZA	*							*		*									D	39
17	ZS	*							*		*									D	
20	SI										*			*		*				D	26
21	CI										*			*						D	
22	SW										*			*		*				D	
23	CW										*			*						D	
24	SCR		SCR				*					*								B_1	66
25	LCR	*					*					*								B_1	71
30	HA			*													*	*		C	42
31	EX		LCP	*													*			C	
32	SST		SBT	*																C	42
33	C		BCP	*															*	C	35
34	BA			*							*									D	31
35	BS			*							*								*	D	
36	A			*						*	*									D	
37	S			*						*	*									D	46
40	NOP				*	*														B_2	29
41	RNM	*	RNM		*															C	79
42	CAM	*	CAM		*	*														B_2	58
43	CSM	*	CSM		*															C	76
45	H		HLT		*															C	29
54	BCC							*												C	54
55	BCE	*						*										*		C	
60	MAT	*	MAT		*															C	82
64	PCB		PCB		*		*								*					B_1	87
65	BCT		BOI		*		*													B_1	52
66	PDT		PRW		*		*								*					B_1	90
74	MCE	*	EDT																	C	94

** B_1 = BYPASS B
B_2 = BYPASS A & B
C = CHAIN B
D = DUPLICATE A

Figure 3-5

instruction (octal 45). After these six bits are stored in the I register,they will be used to control the sequence of the execution cycles and the manipulation of data. The control unit has a group of functions that examine conditions in the I register and decode the six functions in the I register into one function that designates the op code. In many cases all six bits are not needed to identify a particular op code. Refer to figure 3-5 and note that if I05 and I04 are zeros and I03 is a one the instruction must be a halt order because there is no other legal op code that contains this configuration. The control unit has a function called CBHLT10 that will be high if this configuration of ones and zeros appear in the I register.

$$\text{CBHLT10} = \overline{\text{I05}} \bullet \overline{\text{I04}} \bullet \text{I03}$$

Many other op codes derived functions are set in a similar manner. These functions will be high as long as the op code is in the I register (the complete duration of the instruction). In some cases the same op code derived functions will be high for two or more similar instructions. But in most of these cases there will be other op code derived functions,examining different bit positions, high in sufficient combinations to distinguish each op code. These op code derived functions are the primary controlling factors in causing each order to be executed differently. As might be expected, these functions are most often inputs to the cycle selection functions and to subcommands.

PREVIOUS AND DERIVED CYCLE FUNCTIONS

The existant cycles go high at time six (CT6) and recirculate through CT4. No existant cycle, either major or minor, is high during CT5. Therefore,if cycle E2C should come high when E1C is done,the C. U. will have to store the fact that E1C was the previous cycle and use this function to set E2C. The control unit does in fact have a group of functions that are set during an existant cycle and recirculate well into the next cycle. Each major and minor cycle has a corresponding "previous" cycle. Most previous cycles set at CT3 and recirculate through CT1; some set at CT2 and recirculate through CT8. For example, at CT3 of every E3C the function CBP3C10 will set, and it will recirculate through CT1. The input gates to CBP3C10 look like this:

$$\begin{aligned}\text{CBP3C10} &= \text{E3C} \bullet \text{CT3}\\ &+ \text{P3C} \bullet \overline{\text{CT2}}\end{aligned}$$

The top gate is used to set P3C at CT3 of every E3C and due to the bottom gate, once P3C comes high it will stay high as long as $\overline{\text{CT2}}$ is high. The first time CT2 comes high P3C will go low. Similar gating is used on the other previous cycles.

During the course of a memory cycle, the control unit uses "derived cycle" functions to monitor the existant cycles and to aid in the generation of subcommands. All but three of the "derived cycles" are derived from the execution cycles. These monitoring functions usually are set at CT8 and recirculate through CT6. CADE110 is set during an E1 cycle (EEC•E1C) at CT8; however CBDE110 is set at CT6, and CBDE130 is set at CT3. But these last two should be considered exceptions to the normal case.

CYCLE SELECTION

Every major cycle and every minor cycle has a group of selection functions feeding it. For example, E1C can be set by any one of a group of selection functions, S1A + S1C + S1D + S1E + S1F, depending on certain conditions.

$$\begin{aligned} CBE1C10 &= \underline{N07} \bullet S1A \bullet CT6 \\ &+ N07 \bullet S1C \bullet CT6 \\ &+ CT6 \bullet S1D \\ &+ CT6 \bullet S1E \\ &+ CT6 \bullet S1F \\ &+ E1C \bullet HER \end{aligned}$$

The first thing that is apparent, is that every set gate has CT6 on it. The recirculation gate has HER which will go low at CT5. So once the selection function comes high, CBE1C10 will set at CT6 and recirculate through CT4. Each gate on the selection functions is examining a specific set of conditions that will require the next cycle to be an E1C. For example:

$$\begin{aligned} CBS1D10 &= P1C \bullet S1H \\ &+ PV1 \bullet SPM \\ &+ PV1 \bullet \underline{MAT} \\ &+ ITF \bullet PCC \\ &+ P3C \bullet HFC \end{aligned}$$

$$\begin{aligned} CBS1H10 &= PBC \bullet POP \\ &+ MAT \bullet PEC \end{aligned}$$

To aid in understanding these functions use the following definitions:

P1C	=	Previous cycle was E1C
PV1	=	Previous variant cycle and E1C
SPM	=	Set punctuation mark instruction, SI & SW
MAT	=	MAT op code derived function
ITF	=	Interrupt flop
PCC	=	Previous ECC
P3C	=	Previous E3C
HFC	=	Previous extraction cycle (fetching)
PBC	=	Previous EBC
POP	=	Peripheral Orders PCB + PDT
PEC	=	Previous EEC

The top gate on S1D will be high if the previous cycle was E1C and the selection function S1H is high. CBS1H10 will be high if the previous cycle was EBC during a peripheral order (top gate), or if the previous cycle was EEC during an MAT instruction (bottom gate). Now, using the bottom gate of S1H and the top gate of S1D, the control unit will select E1C as the next minor cycle of an MAT instruction, if the previous cycle was E1, (EEC•E1C). In this case ECC will set also and the new cycle will be C1. By the second gate on S1D, (and the third on E1C) E1C will be selected as the next minor cycle if this is a "set punctuation mark" instruction and the previous cycle was a variant cycle one. This is a sample of the type of logic used for cycle selection. Of course, in conjunction with the generation of ElC a major cycle is usually selected to completely define a memory cycle. If no major cycle is selected, a variant (fetching) cycle is assumed. Only one major cycle and one minor cycle can be high at a time.

SUBCOMMAND GENERATION

The N register is a collection and distribution center for data within the 200. When information is to be written into a memory location, it is loaded into the "N" register while main memory is addressed. The "N" register circuits will then be used as inputs to the memory circuitry that causes a one or zero to be written into a particular position. Likewise, when information is read out of a memory location, it is read into the N register for further distribution by the control unit. Data may be sent from the N register to four other registers; the "I" register (op code delivery), the "F" register (peripheral output bus), or to the A or B registers for arithmetic and logical operations or further distribution. The control unit will use the derived cycle functions and op code derived functions to set the subcommands that will transfer the information to another register. For example, it was stated earlier that E1 is generally used to read the A operand out of memory. Since the A operand is usually sent to the A register, it follows that the subcommand CAN2A10 should be generated during E1 (N2A gates information from the "N" register to the A register). If N2A had to be set every E1 cycle there would be a gate on N2A that had DE1 and the proper timing (CT6). However, the exceptions to the rule prevent this. Functions are needed to inhibit the delivery of N to A during E1 when it is damaging; i.e. during an MAT instruction, an SCR instruction, and while subtracting. To prevent N2A during E1 of an MAT instruction, the op code derived function $\overline{\text{MAT}}$ is anded with DE1. It is also desirable to inhibit the transfer of N2A in decimal orders when a blank (15_8) is in the "N" register. This will make a blank in the "N" register into a zero in the "A" register. For this reason $\overline{\text{BIZ}}$ (Blank is zero) is needed to set N2A.

CAN2A10 = DE1•$\overline{\text{MAT}}$•$\overline{\text{N4A}}$•CT6•$\overline{\text{BIZ}}$
+ DE2•BCH•CT5
+ HXC•CT5
+ N2A•CT6

To prevent the setting of N2A during an SCR instruction and during subtractions, a function that will go low in both these cases is anded with DE1 and $\overline{\text{MAT}}$.

CAN4A00(I) = LSR
+ TUB
+ SAO

The function $\overline{\text{N4A}}$ is an inverter (the output will go low if any input gates comes high), so $\overline{\text{N4A}}$ will go low for every SCR instruction (LSR is an op code derived function for SCR and LCR instructions). $\overline{\text{N4A}}$ will also go low for true subtractions (TUB) and binary subtracts (SAO). This all means that in every E1 cycle N will be delivered to A except when the conditions noted are present. There are two instructions which require N2A during E2. They are BCC and BCE. Since they have a common op code derived function, BCH, DE2 can be anded with BCH to get N2A as needed in these instructions. Another time N2A is needed is during the indexing cycles. The function HXC (held indexing cycle) does this. The bottom gate on N2A is to recirculate the function, so it will be high for times five and six.

The generation of the other subcommands is very similar. When reading the B operand out of memory the subcommand CAN2B10 is needed instead of N2A. E2 cycle is generally used to deliver the B operand to the B register. In fact, the only time N2B isn't generated during E2 is in a BCE or BCC instruction (note that N2A was generated if DE2•BCH are high), or if a blank is in the "N" reg. Function N2B has CT6•$\overline{\text{BIZ}}$•DE2•$\overline{\text{BCH}}$ on one gate; this means that in every E2 cycle, except when doing a BCE or BCC instruction we will transfer the information in N to B, provided BIZ is not high. N2B is also needed in some of the cases that N2A was inhibited during E1 cycles (MAT and LSR). The rest of the times that N2B is needed are during the extraction cycles.

CAN2B10	=	DE2•$\overline{\text{BCH}}$•$\overline{\text{BIZ}}$•CT6	(Normal Set Gate)
	+	IIN•N07•CT6	(Extraction only)
	+	N2B•T08•CEEDH	(Recirculation)
	+	N4B•CT5	(Auxiliary functions)

During the delivery cycle (almost always E3) information to be writter into memory is loaded into the N register. Simultaneously it's necessary to inhibit the normal read out of information into the N register to prevent the old contents of the memory location addressed from being superimposed on the information to be stored in that location. Therefore, to deliver information to the N register two subcommands are needed; CAINL10 inhibits the normal read out from memory, and A2N, or P2N, or G2N to load the new information into N. CAA2N10 gates information from the accumulator to the N register. The normal gate on INL is I2L•DE3•CT2. I2L is an auxiliary function that will be set for every instruction except; BCE, BCC, C, and under certain conditions, MCE and EXM. For all other instructions I2L will be high, so if the instruction uses an E3 cycle, INL will set at CT2. The normal gate on A2N is DE3•CT3•$\overline{\text{CEEDT}}$. This gate will set A2N at CT3 of every E3 cycle except during edit orders.

CAA2N10 = EWC•SCR•CT2•INA
+ ECC•E3C•CT3
+ DJ3•TAF•EDE (edit only)
+ DE3•CT3•$\overline{\text{EDT}}$ (normal gate)

The top gate on A2N defines the only condition for delivery to memory in some cycle other than E3C. EWC is set for the three execution cycles of the SCR instruction. The second gate takes care of a special delivery cycle. The third gate is for edit only.

SENSING OPERAND PUNCTUATION

Instructions that perform multi-character operations depend on operand punctuation to terminate the execution cycles. The functions that sense for word marks and item marks look complex at first glance because they are occasionally used for some other related purposes. CBACF10 is mainly used for sensing word marks with the help of the auxiliary function CBACG10.

CBACFl0 = N07•CT6•ACG
+ PMC•DUP•CT8
+ EEC•SPM•102
+ ACF•PVC•CLG

CBACGl0 = DEl•DAS•MPM
+ DEl•DAS•PPF
+ BCH•DE2
+ PCC•MAT

The top gate on ACG is high during every E1 cycle of any instruction except a decimal add or subtract, or a set or clear punctuation mark instruction. Since ACG is anded with N07 on the input to ACF, sensing a word mark in all of the other instructions will set ACF. The second gate on ACG will be high during E1 after the first pass is complete in a decimal add or subtract; therefore, if there is a word mark, ACF will set. The third gate down allows ACG to set ACF if there was a word mark sensed in a BCC instruction (it will set if there was a word mark in E2 of a BCE also, but in this case ACF will be ignored). The bottom gate on ACG is to sense a word mark within the translation table of a MAT instruction. The second gate down on ACF is used to allow a word mark to set NG7 during M2 of a PDT instruction. The third gate on ACF will simulate a word mark in every execution cycle of a SW instruction whether there was one or not. The bottom gate is for recirculation.

CBAFC10 serves the same general purposes for item marks as ACF did for word marks, plus a few miscellaneous odd jobs that will be discussed as needed in other chapters.

ITEM MARK TRAPPING MODE

In some cases a program may be run on a machine that doesn't recognize every op code in the program as legal. In these cases, the programmer may mark these op codes with item marks, and set up a subroutine which will analyze the op code and the A and B addresses of the item marked instruction and do a series of instructions which will simulate the non-existant op code. The programmer specifies the starting location of the subroutine in the co-sequence counter.To put the machine in the "item mark trapping mode" the programmer issues a CAM instruction with X4_8 as the variant (X depends on the addressing mode desired). To ignore item marks in the op code a programmer may issue a CAM with V03 a zero bit. In this manner the "item mark trapping mode" may be turned on and off at will.

When the machine is in the item mark trapping mode and an item mark is sensed in the op code the machine forces a CSM op code to replace the op code with the item mark and does a normal CSM instruction. The result of this is to put the program into the subroutine, instead of attempting to execute the instruction with the item mark.

CBIMT10 is the indicator for item mark trapping; it is set when a CAM instruction has a one in V03. As long as IMT00 is high N4I determines when N2I will come high (PV3•$\overline{\text{ITF}}$•RUN) for the normal loading of an op code into the I register.

$$\begin{aligned} \text{CBN2I10} &= \text{N4I}\bullet\text{T07}\bullet\overline{\text{IMT}} \\ &+ \text{N4I}\bullet\text{T07}\bullet\text{N08} \end{aligned}$$

However if $\overline{\text{IMT}}$ is at ground the op code is only loaded when N08 is a zero (no item mark). Therefore, when N08 is a one the op code isn't loaded.Instead CBCSD10 (change seq. demanded) comes high and sets those positions of the op code register needed to make a CSM op code $(43)_8$. CSD = N4I•IMT•N08.

It should be noted that IMT is an indicator very similar to CM2 and must also be stored during an interrupt. IMS is the storage flop for IMT during interrupt. This means that an interrupt routine is free to use IMT just as any other program can.

Flow Charts

Each logical memory cycle is represented on a block diagram that shows the data flow caused by the subcommands that are high during that memory cycle. Figure 3-5 is an example of a result delivery cycle. Every register in the 200 CP is represented on one sheet of paper. Because control memory cycles four times during one main memory cycle, all four control memory cycles are shown on the same page. At the top of the page this example is identified as E3 (result delivery) of a ZA or ZS instruction. The arrow from the control memory sense amps to S at T01 occurs in almost all memory cycles. This means that the first control memory location cycled at T07 nearly always contains the address of the main memory location to be cycled. The arrow from the accumulator to the Z register occurs every T02. In this example, as in most execution cycles, the Z register dows not feed into control memory during the second CM cycle. The address is decremented, and then during the fourth control memory cycle is written into the location it came from originally. This occurs in most EEC type execution cycles. The A2N transfer occurs during every E3 cycle. Because this is a delivery cycle, the contents of the N register are loaded from the accumulator and is written into main memory. This E3 cycle is nearly identical to every other E3 cycle, if some of the subcommands were not wanted another type of cycle would have been chosen by the designer.

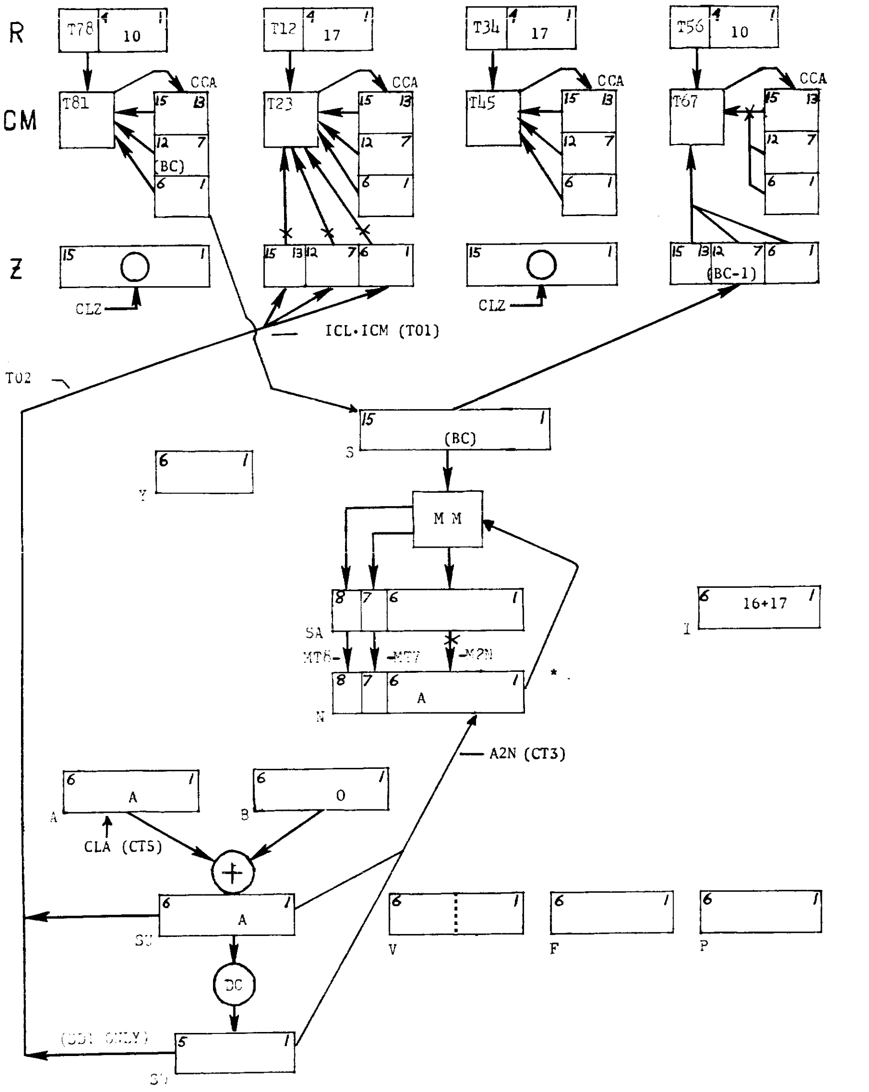
The right hand column lists the possible cycles that may be performed next and the conditions that determine which one will be selected. Since all possible next state conditions are listed,any legal example should be able to use one combination of the gates shown. There is room at the top of the page for remarks and timing relevant to this cycle

			6	7	8	1	2	3	4	5	6	7	8	ENTIRE CYCLE
INSTRUCTION	OP-CODE	DESIGNATOR												
ZA ZS	16 + 17	E3												
MAJOR	MINOR	EARLY												
EEC	E3C													

REMARKS

* If B field >A field, clear extra characters of B.

Set SGH = CAS$\cdot\overline{\text{HFC}}$
PPF = DE3$\cdot\overline{\text{DEC}}\cdot$CT6



NEXT STATE

EEC = SEB$\cdot\overline{\text{N07}}\cdot$CT6
SEB = DE3$\cdot\overline{\text{I06}}\cdot$I04$\cdot$I03
E1C = CT6$\cdot\overline{\text{N07}}\cdot$S1C
S1C = DE3$\cdot\overline{\text{ACF}}\cdot$S1B
S1B = $\overline{\text{I06}}\cdot$I04$\cdot\overline{\text{MUV}}\cdot\overline{\text{SBT}}$
$\cdot\overline{\text{BCP}}$

E3C = CT6$\cdot$S3B$\cdot\overline{\text{PST}}\cdot$N07
S3B = $\overline{\text{P1C}}\cdot$PEC$\cdot$TEC
TEC = $\overline{\text{TUB}}\cdot$P3C$\cdot\overline{\text{MUV}}\cdot\overline{\text{MAT}}$

E3C = S3A$\cdot\overline{\text{N07}}\cdot$CT6
S3A = DE3$\cdot$ACF$\cdot$CAS

REV 2

Fig. 3-6

SECTION 4
CONTROL MEMORY

INTRODUCTION

Control memory in the 200 consists of 16 registers. Each register is identical, in that each has the ability of storing up to 15 binary digits. All locations in control memory are directly accessible by the central processor and have the capacity of storing a single address of either an operand or instruction.

A linear select method is used to address control memory; i.e. current flows only through the cores in the location being addressed. With this type of selection method stray noise and interference are kept to a minimum to lessen the possibility of misinterpretation.

The circuits used in control memory are similar to those in main memory and perform almost identical tasks, although not simultaneously. Control memory cycles four times for every main memory cycle, thus making its contents available as an address for the main memory cycle. The following is a list of circuits, contained in control memory, that are similar to those in main memory.

1. Address Selector
2. Current Generators
3. Read/Write Drivers and Selector Switches
4. Sense Amplifiers

CONTROL MEMORY REGISTERS

The 16 registers, with their octal locations, mnemonic code and descriptions, are given in Figure 4-1.

Read/Write Channels

A read/write channel is used to complete the path between main memory and the peripheral device specified. Two locations are reserved for each read/write channel: one location contains the starting address of the data, the other keeps track of the current or present address. All peripheral transfer instructions must specify a read/write channel, and this channel will be used for this instruction alone until the data transfer is complete.

A and B Address Registers

The A and B address registers contain the main memory address of the operands that are specified by the instruction being executed.

Instruction Address Register #1

Instructions are performed, within the central processor in the order specified by Instruction Address Register #1 (usually referred to as the sequence counter). This register is initially set up to contain the address in main memory of the first instruction to be performed. As each portion of the instruction is extracted from memory the register is incremented by one and returned to control memory. Upon completion of the instruction extraction (when a word mark is sensed) the incrementing will stop, and the sequence counter will then contain the address of the next instruction to be performed. However, certain instructions specify the address of the next instruction to be executed and will supersede the address in the sequence counter.

LOC	MNEMONIC	DESCRIPTION
00	NOT USED	NOT USED
01	RWC1	READ/WRITE CHANNEL #1 CURRENT ADDRESS
02	RWC2	READ/WRITE CHANNEL #2, CURRENT ADDRESS
03	RWC3	READ/WRITE CHANNEL #3, CURRENT ADDRESS
04	IR #2	INSTRUCTION ADDRESS REGISTER #2 (OPTION)
05	RWC1'	AUXILIARY READ/WRITE CHANNEL, CURRENT ADDRESS (OPTION)
06	INTERRUPT	INTERRUPT REGISTER (OPTION)
07	WL2	WORKING LOCATION 2
10	bc	B ADDRESS COUNTER
11	RWS1	READ/WRITE CHANNEL #1, STARTING ADDRESS
12	RWS2	READ/WRITE CHANNEL #2, STARTING ADDRESS
13	RWS3	READ/WRITE CHANNEL #3, STARTING ADDRESS
14	ac	A ADDRESS COUNTER
15	RWS1'	AUXILIARY READ/WRITE CHANNEL, STARTING ADDRESS (OPTION)
16	WL1	WORKING LOCATION 1
17	IR #1	INSTRUCTION ADDRESS REGISTER #1 (SEQUENCE COUNTER)

Figure 4-1

Instruction Address Register #2 (optional)

This register is usually referred to as the co-sequence counter. With the option it is possible to interchange the contents of the sequence counter and the co-sequence counter by issuing a CSM (change sequence mode) instruction. Since the address of the next instruction to be performed is in the sequence counter, issuing a CSM causes the program to continue from the point specified by the co-sequence counter. Issuing a second CSM instruction again swaps the contents of the two registers, and the program will continue at the point where it left off.

Interrupt Register (optional)

The interrupt register is similar in concept to Instruction Address Register #2. The main difference between the two registers is that the Interrupt Register is interchanged with the sequence counter upon the receipt of an interrupt signal from a peripheral device without the need of an instruction. ARNM (resume normal mode) instruction will return the program to the point where it left off.

Working Locations

Working locations are used by the central processor during the execution of an instruction as a temporary storage area for an address. Occasionally the control unit will route unwanted information into a working location to avoid its use in another register.

Internal Transfers

When it becomes necessary for the machine to transfer the contents of one control memory register into another register during the execution of an instruction, an "internal transfer" is performed. The machine accomplishes this transfer by addressing the two locations concerned consecutively; the first read/write cycle is performed normally, but the information in the sense amps is not cleared at the end of the cycle. During the second read/write cycle, the information being read out is ignored and not allowed into the sense amps. Therefore, at the end of the second read cycle, the information in sense amps is the contents of the first location. This information will be written into the second location during the normal write cycle which follows (the last half of the second read/write cycle).

Timing

The control memory timing network is syncronized with the central processor through the function CBTEV50. Control memory will complete one cycle (read and write) for every cycle of the basic oscillator; i.e. every 500 ns.

The majority of the timing functions needed to control this memory are contained in two packages, a 2CTT4 and a 2CPT3.

The 2CTT4 package is used to derive the following functions:

CCATR10	-	Control Memory Address Transfer (high to address control memory)
CCRES10	-	Sense Amp Reset (when RES goes to ground, sense amps can no longer recirculate)
CCDDT10	-	Digit Timing

Figure 4-2 shows a logical block diagram, and a graph of the timing derived from each logical block, of the 2CTT4 package.

It can be seen from this figure, that the function CCATR10 will come high (+5v) at the beginning of each time even pulse, and that the duration of this function will be dependant upon the RC time constant of D1 (differentiator). CCRES10 is a 180° phase reversal of CCATR10, but can only reset the sense amps if CBIIT10 (inhibit internal transfer) is high.

The digit timing function, CCDDT10, which is a ground enable signal, will come true sometime around the trailing edge of time even. Both the leading and trailing edges of CCDDT10 are adjustable, the leading

A-2 Xfer the selected Reg. from
? into CCMA, which addresses the CM

2CCT4 PACKAGE to CM timing

C8A5K

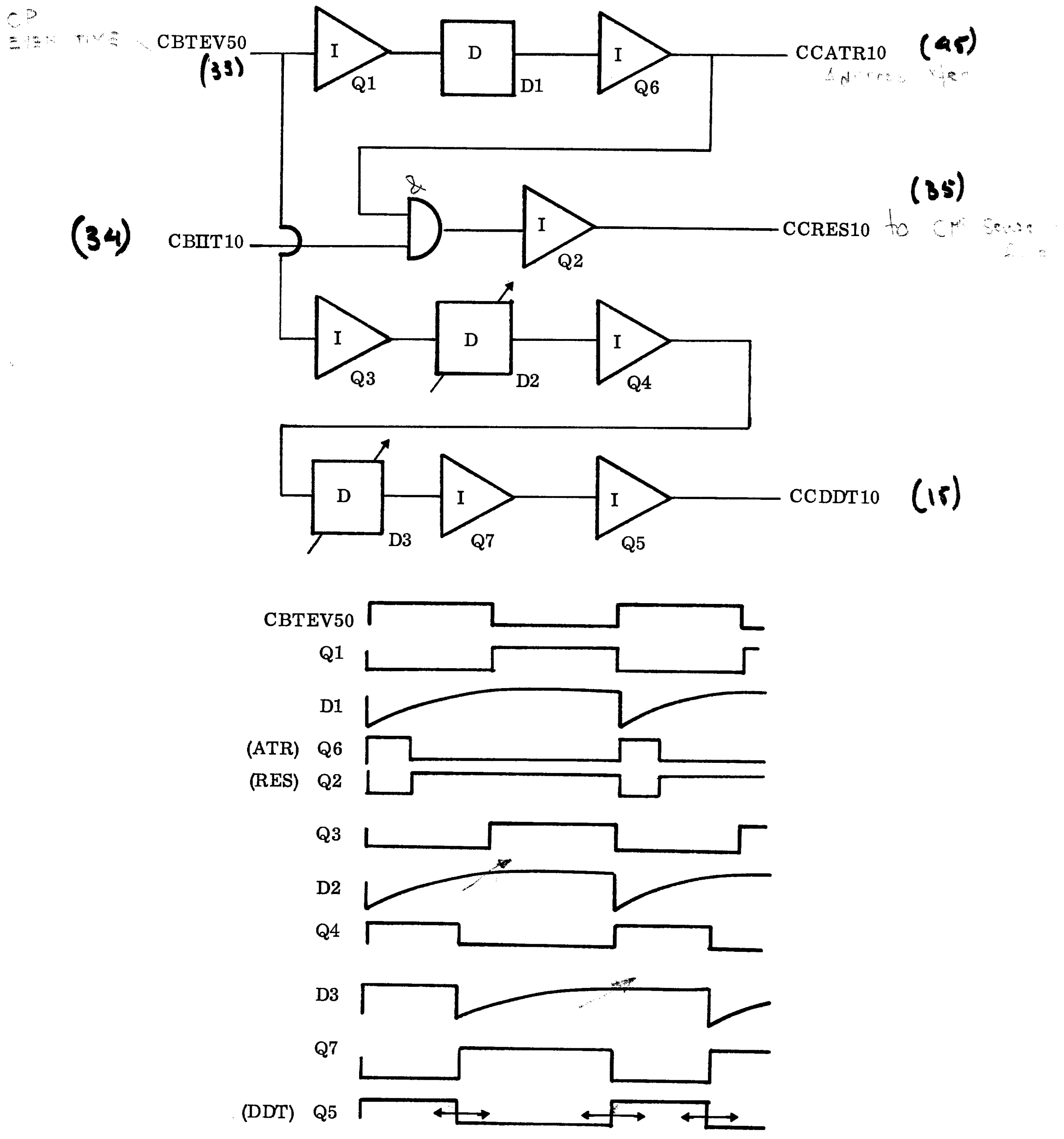


Figure 4-2

enables
writing into C.M.

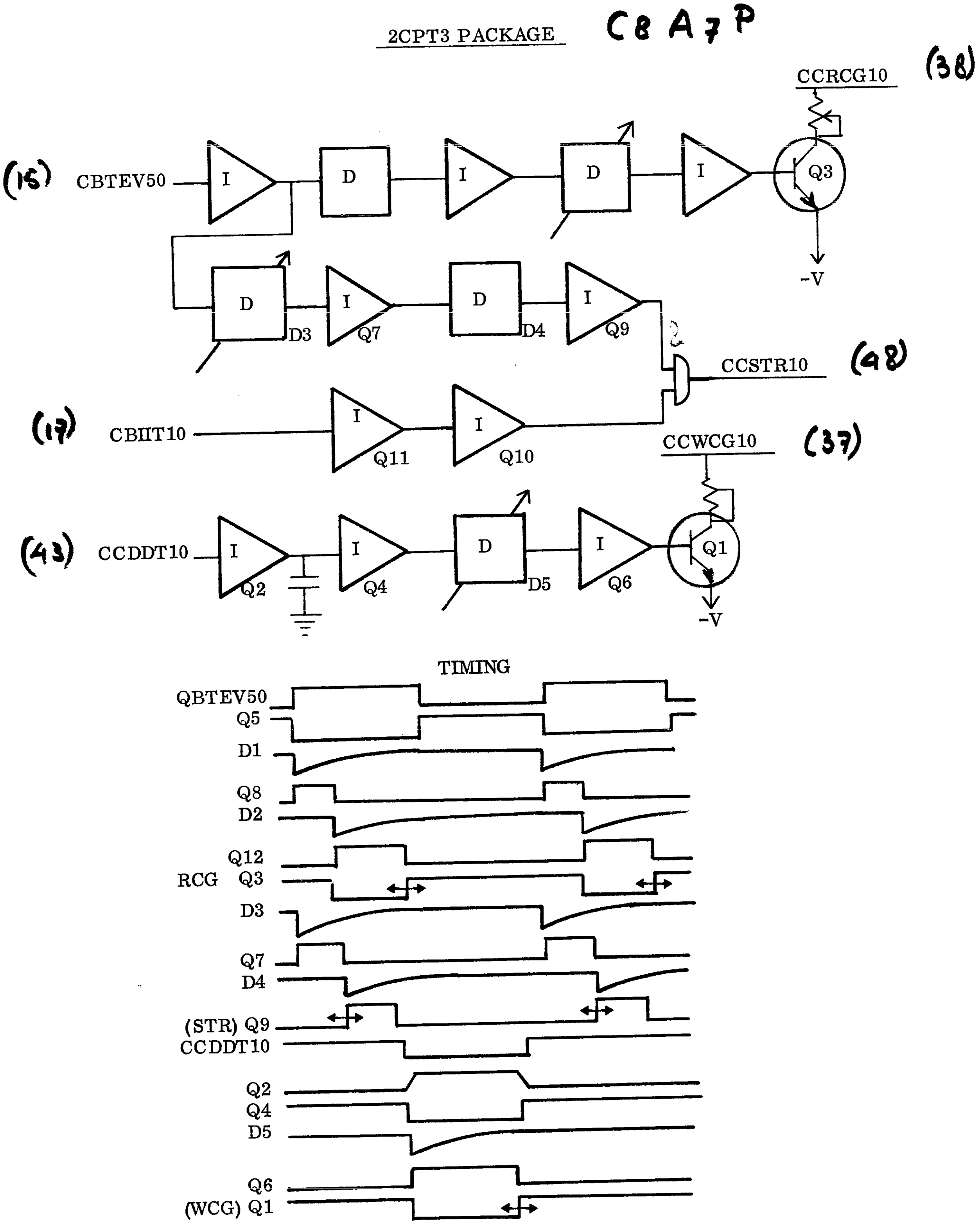

2CPT3 PACKAGE
C8 A7 P
(38)
CCRCG10
(15)
CBTEV50
I
D
I
D
I
Q3
-V
D
D3
I
Q7
D
D4
I
Q9
CCSTR10
(48)
(17)
CBIIT10
I
Q11
I
Q10
CCWCG10
(37)
(43)
CCDDT10
I
Q2
I
Q4
D
D5
I
Q6
Q1
-V
TIMING
QBTEV50
Q5
D1
Q8
D2
Q12
RCG Q3
D3
Q7
D4
(STR) Q9
CCDDT10
Q2
Q4
D5
Q6
(WCG) Q1

Figure 4-3

edge being variable by D2,the trailing edge by D3.

The 2CPT3 package generates the following functions:

CCRCG10	-	Read Current Generator
CCWCG10	-	Write Current Generator
CCSTR10	-	Sense Amp Strobe

Figure 4-3 shows a logical block diagram and a timing graph of these functions.

CBTEV50 and CCDDT10 are the main timing functions used to derive the three output functions.

The trailing edge of the function CCRCG10 is variable by D2. The current output of Q3 can be adjusted by varying the potentiometer in the collector circuit. CCSTR10, which is used to gate information from the cores into the sense amps , is a fixed length pulse, determined by D4. The positioning, or time at which this pulse will be true, is variable and controlled by D3. The function CCDDT10 is the controlling factor ofCCWCG10. Figure 4-3 depicts CCWCG10 occuring within the duration of CCDDT10, due to the capacitor between Q2 and Q4, with the trailing edge being variable depending on the setting of D5. The current output of Q1 is adjustable by the resistor in its collector circuit.

A timing chart, showing the relationships of the functions previously derived, is shown in Figure 4-4; this chart also takes transistor rise time into consideration.

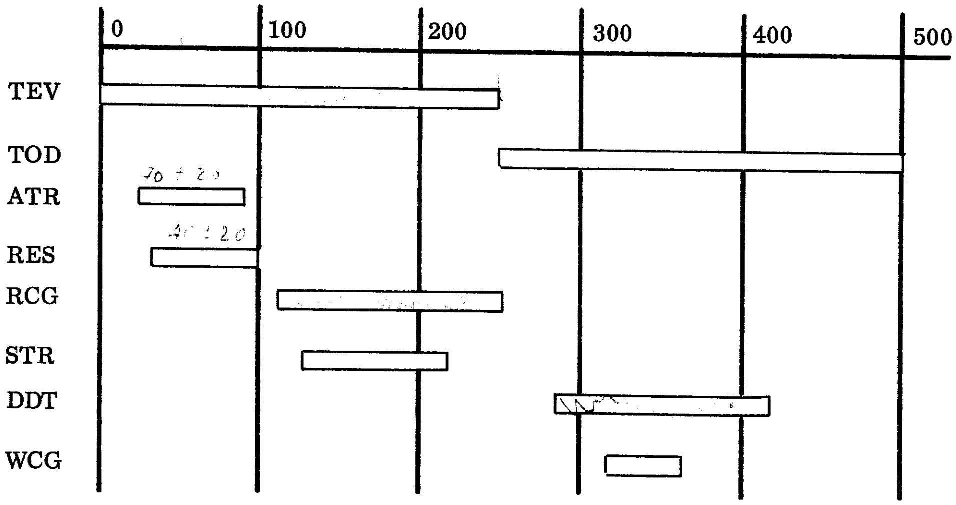


Figure 4-4

Control Memory Operations

The control memory stack is a linear select, two core per bit, storage element capable of storing sixteen, fifteen bit words. All cores for a single location are threaded by two common wires, one wire used for read the other for write. The four read/write drivers are logically located on one side of the stack with the four selector switches located on the other side. This arrangement allows the selection of a single location when only one read/write driver and one selection switch is activated. See figure 4-5.

The inner circumference of all cores is plated with a conductive material, and this material is joined to a

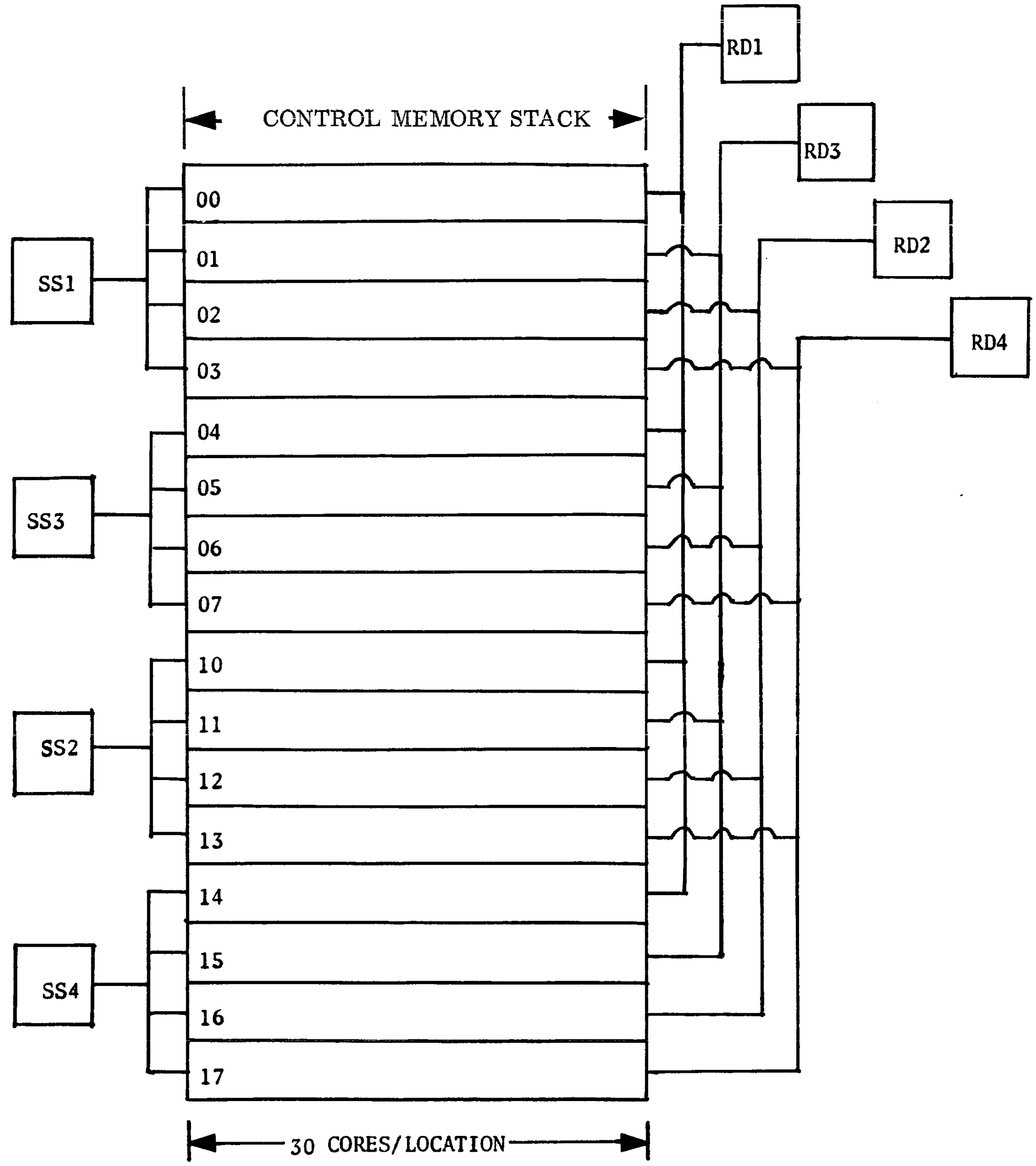

RD1
RD3
RD2
RD4
CONTROL MEMORY STACK
SS1
SS3
SS2
SS4
00
01
02
03
04
05
06
07
10
11
12
13
14
15
16
17
30 CORES/LOCATION

Figure 4-5

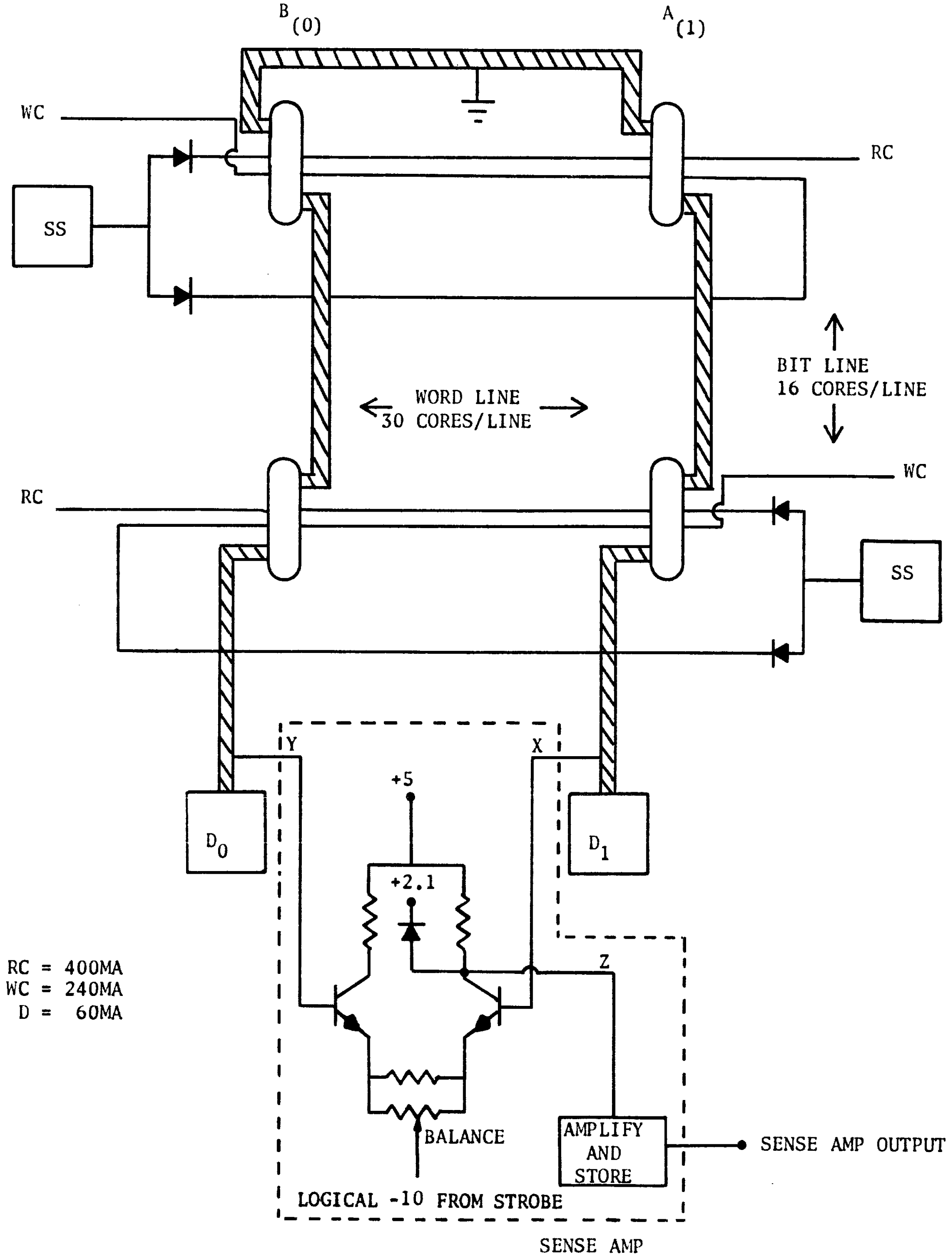

B (0)
A (1)
WC
RC
SS
BIT LINE
16 CORES/LINE
WORD LINE
30 CORES/LINE
RC
WC
SS
Y
X
+5
D_0
D_1
+2.1
Z
RC = 400MA
WC = 240MA
D = 60MA
BALANCE
AMPLIFY AND STORE
SENSE AMP OUTPUT
LOGICAL -10 FROM STROBE
SENSE AMP

Figure 4-6

printed conductive line which threads all cores for a specific bit position.

The above mentioned winding will serve a triple purpose. First,it will eliminate a possible core heating problem by acting as a heat sink. Second, during read time it will perform the job of a sense winding by sensing for a flux change in its particular bit position. Third, during write time it will serve as the digit drivers output line augmenting the write current in the selected core. Figure 4-6 shows the manner in which the cores are threaded. Only one bit position for two different locations is shown.

If a one is to be stored in memory, the write driver line will supply approximately 240 ma turns of magnetizing force to both the A and B core. Digit driver D_1 will activate and supply another 60 ma turns of magnetizing force to the A core, this current will augment the write current, giving a combined magnetizing force of 300 ma turns. D_0 depends on $\overline{D_1}$ and will activate only if D_1 does not, therefore the only magnetizing force supplied to the B core will be that of the write current.

Read current drives all cores in the selected location to a pre-determined saturated state. If a one had been stored in this position the change in flux of core A will be greater than the change in flux of core B; therefore, the signal at point X (figure 4-6) will be greater in amplitude than the signal at point Y. The sense amp input circuit, being a difference amplifier (X-Y), will produce a negative going signal at point Z that will be amplified and stored, in the sense amp package, as a one.

Assume that during the next write cycle, a zero is to be returned to this bit position. The write driver line will again supply 240 ma turns of force to both cores A and B. Digit driver D_1, however, will not be turned on, but D_0 will activate, supplying an additional 60 ma turns of force to aid the write current in core B. Therefore, core B will be driven further from the saturation level than core A.

The 400 ma of read current supplied by the read winding will again drive all cores to the saturated state. This time, however, the signal sensed at point Y will be larger than the signal at point X. Through adjustment of the balance potentiometer,the signal at point Z for the read out of a zero can almost be eliminated. Therefore, a zero will be stored in the sense amp.

The read current, as previously stated supplies 400 ma turns of magnetizing force. During write, 240 ma of current, supplied by the write current generator, plus 60 ma supplied by the digit driver, produces a total of 300 ma turns of magnetizing force. This inequality is necessary to prevent an undesired event called creeping. Examine the hysterisis loop as applied to control memory.

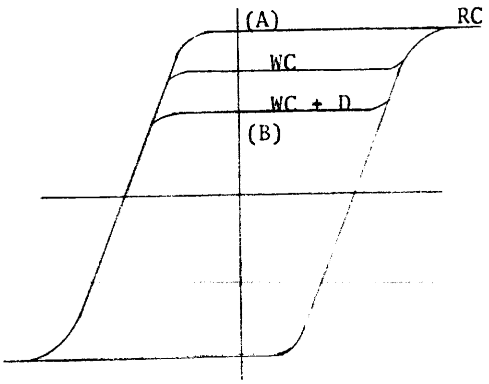


Figure 4-7

Note that only the top portion of the curve is actually used, the core never changes polarity. This buys speed. However, if read and write currents were the same, point B might tend to creep downward due to a very minor adjustment inaccuracy. In this system, by overdriving the cores and insuring total saturation on the read drive, Point A is established as a fixed reference point.

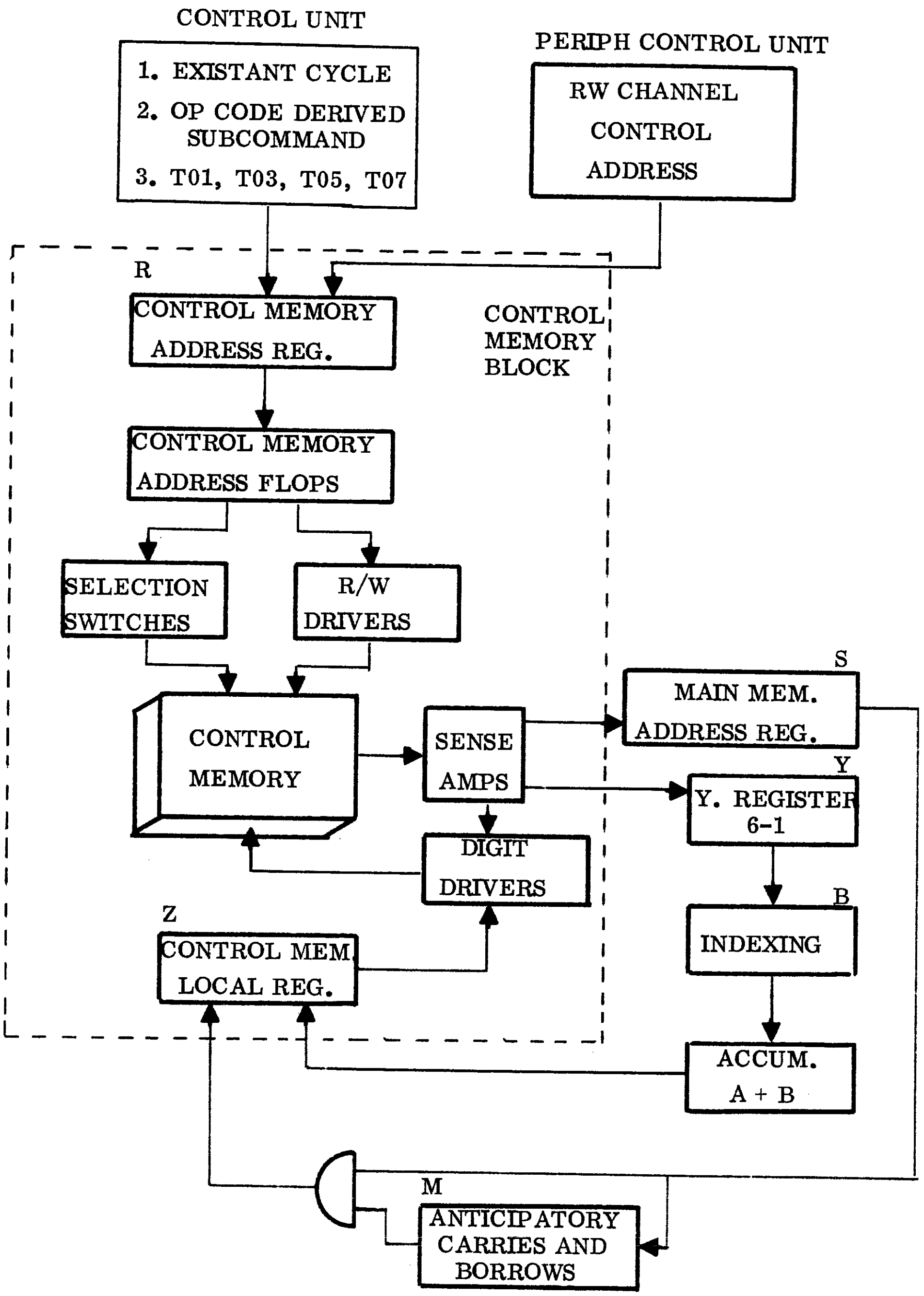


Figure 4-8

Control Memory Addressing

An over-all block of control memory is shown in figure 4-8. Addresses are generated in the R register during the odd times (T01, T03, T05, T07). The addresses generated will be dependent upon either the central processor control unit or a peripheral control unit. At CCATR10 time the address will be transferred from the R register to the control memory address flops; this address will determine which location is to be read into the sense amps. The address flops will gate on a specific read/write driver and a specific selector switch which will allow current to pass through a single location. With the aid of CCSTR10 the information previously stored in this location will be stored in the sense amps.

During the write cycle the address used to select the read-out location is still contained in the address flops. Information can now be returned to this location from either the sense amps, the Z register,or a combination of both. Write Z functions (WZU, WZM, and WZL), and write S functions (WSU, WSM and WSL) will determine whether the Z register or the sense amps will control the digit drivers. See figure 4-9 below.

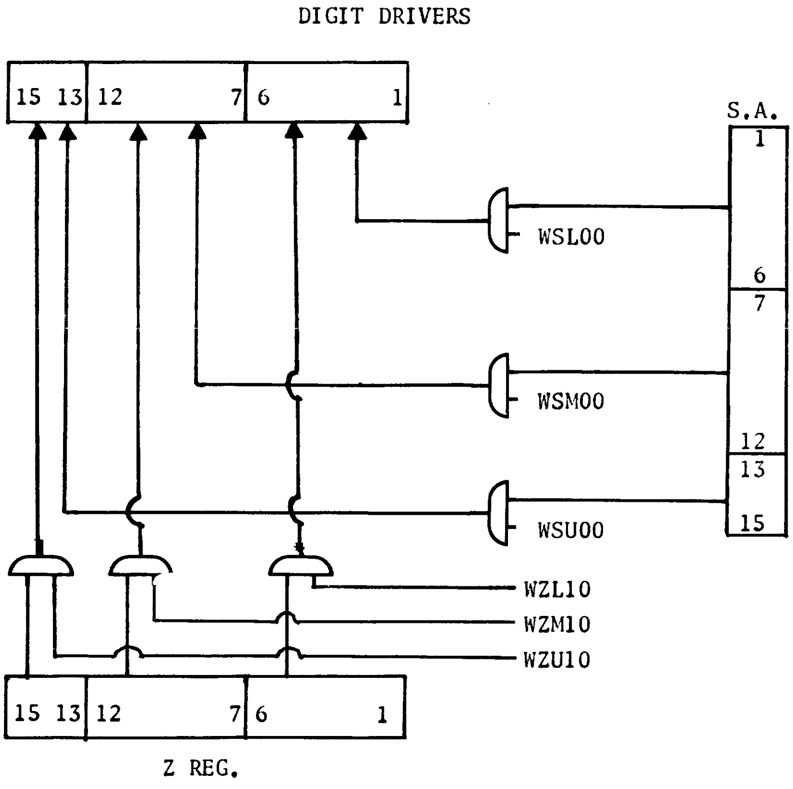


Figure 4-9

Normally, if new information is to be written into a control memory location from Z register, it is done at time 3 or time 7. At time 3, either Z upper, Z middle, or Z lower may be written into control memory (the remaining bits are entered from the sense amps). At time 7, if any portion of Z is to be entered the entire register will be used.

Note that the Write S functions are negations while the Write Z functions are assertions. The write S functions are in reality the negation of the Write Z functions; so that if WZL10 is low, WSL00 will be high. At no one time can WZL10 and WSL00 both be high.

Delivery of Information From Control Memory

If the subcommands CBICL10 and CBICM10 are high at time one, the contents of control memory will be delivered to the main memory S register. The subcommand CBICL10 will control the low order six bits, while CBICM10 controls the remaining nine bits.

The subcommand CBKYL10 will deliver bits one through six from the sense amps to the Y repeater register. CBKYM10 will deliver bits 7 through 12 and CBKYU10 will deliver bits 13 through 15. This path is used in the extraction cycles and the SCR instruction. The delivery to the Y register is made at time five and the information delivered will recirculate in Y until time three.

Control Memory Local Register

There are two sources of inputs to the memory local register (Z register): the accumulator and the main memory S register.

Delivery from the accumulator to the Z register is made, unconditionally, at time two. Figure 4-9 depicts the manner, in block form, in which the accumulator is fed to the Z register.

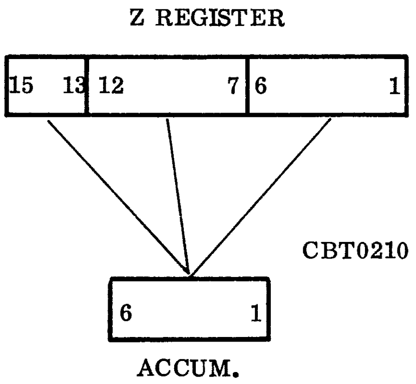


Figure 4-10

There are two input gates on each Z register storage unit for the S register to Z register transfer. The address contained in the S register is always incremented or decremented by unity during a CP cycle transfer, and these two gates are used in deriving the correct result. (The incrementing and decrementing will be discussed in detail later in this chapter.) The result is delivered to CBZ0110 and CBZ0910 at time five, and to the remaining Z functions at time six. This was done to eliminate a possible loading problem with time six.

Incrementing and Decrementing Control

The decision to increment or decrement is made by the central processor control unit through CBISR10 (Increment S register). CBISR10 is a GBA that looks at the central processor control functions and will come high at time eight if an input gate is satisfied. ISR recirculates on CBT0720; so if it is once set it will remain high for almost a complete CP cycle. ISR feeds CBDSR10 (decrement S register) which is a GBI; therefore, if ISR is high DSR will be low. Decrementing is inhibited during a periphial response cycle(CBCPC20 will be high),a stop condition,and a Bootstrap order (CPIDS10 is high for the latter two conditions). Decrementing can only occur when all the input gates to CBDSR10 are at ground.

Incrementation

Incrementation is an arithmetic operation. If the function CBISR10 is high the address contained in the S register will be incremented by unity when it is delivered to the control memory local register at times five and six.

This incrementation is accomplished with the modify bit register in conjunction with an exclusive "OR" circuit on the input gate to the Z register.

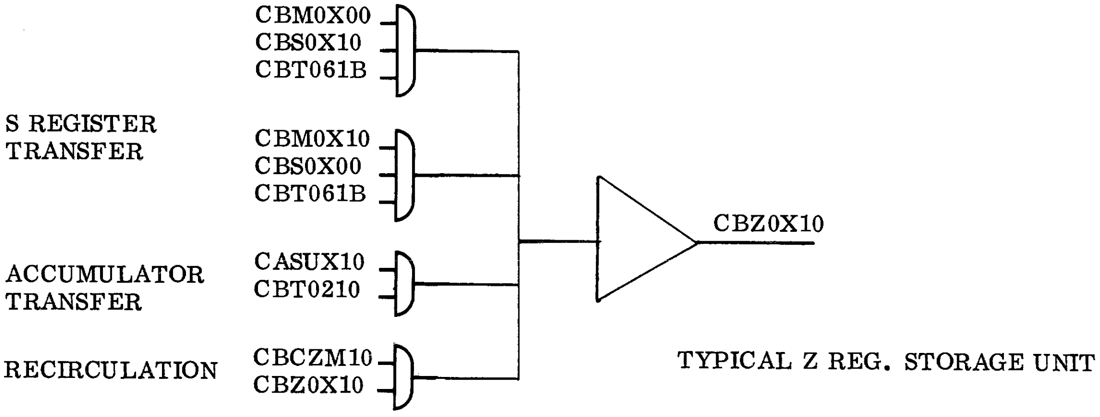


Figure 4-11

With the use of figure 4-1l the following truth table can be derived.

S Reg	M Reg	=	Z Reg
0	0		0
1	0		1
0	1		1
1	1		0

Thus it can be seen that if the M register bit is high, the S register bit will be modified (complimented) when it is stored in the Z register. If the M register bit is low the S register bit will be transferred directly to the Z register. When incrementing by unity there are two major facts to be considered. The low order bit is always modified, and the remaining bit positions will only be modified if there is a carry into that position. The truth table for the modify bit register can now be constructed.

M01	=	ISR
M02	=	S01•ISR
M03	=	S01•S02•ISR
M04	=	S01•S02•S03•ISR
M05	=	S01•S02•S03•S04•ISR

Fig. 4-12

Figure 4-12 shows the first 4 positions of the M register, this same principal could be used to derive the table for the entire register. The table, however, can be simplified by cascading the M register bits as follows.

$$\begin{aligned} M01 &= ISR \\ M02 &= S01 \bullet ISR \\ M03 &= M02 \bullet S02 \bullet ISR \\ M04 &= M03 \bullet S03 \bullet ISR \end{aligned}$$

The logical implementation of this cascading principal could be carried up to bit 15. However, due to a timing limitation this serial method cannot be used for all functions in the M register. M05 is the first function to use speed up logic. $M05 = S01 \bullet S02 \bullet S03 \bullet S04 \bullet ISR$. Cascading is used again up to bit nine. M09 is fed by a special carry hop function CBMBF10.

$$\begin{aligned} CBMBF10\ (I) &= S05 \bullet S06 \bullet S07 \bullet S08 \bullet ISR \\ CBM0910\ (I) &= MBF + \overline{M05} + \overline{M01} \end{aligned}$$

M09 can be broken down to its logical statement, by the use of boolean algebra, as follows:

$$\begin{aligned} M09\ \text{Primed} &= \overline{MBF} \bullet M05 \bullet M01 \\ &= \overline{MBF} \bullet M05 \bullet ISR \\ &= \overline{MBF} \bullet S01 \bullet S02 \bullet S03 \bullet S04 \bullet ISR \end{aligned}$$

Since MBF is an inverter, $\overline{MBF}$ is equal to the input gates, therefore the equation:

$$M09 = S05 \bullet S06 \bullet S07 \bullet S08 \bullet S01 \bullet S02 \bullet S03 \bullet S04 \bullet ISR$$

Bits 12 and 15 are the only remaining positions in the M register where speed up logic is used.

$$\begin{aligned} M12 &= S09 \bullet S10 \bullet S11 \bullet M09 \bullet ISR \\ M15 &= S12 \bullet S13 \bullet S14 \bullet M12 \bullet ISR \end{aligned}$$

Figure 4-13 shows the boolean equations for all 15 M register bits. The derivation of the boolean equations for the decrementing functions is left as an exercise for the student.

Notice that MHC (on M13) inhibits the modification of the module bits (S15 thru S13) if the machine is presently operating in the two character mode.

$\mathrm{M01} = \mathrm{ISR}$

$\mathrm{M02} = \mathrm{S01} \bullet \mathrm{ISR}$

$\mathrm{M03} = \mathrm{M02} \bullet \mathrm{S02} \bullet \mathrm{ISR}$

$\mathrm{M04} = \mathrm{M03} \bullet \mathrm{S03} \bullet \mathrm{ISR}$

$\mathrm{M05} = \mathrm{S01} \bullet \mathrm{S02} \bullet \mathrm{S03} \bullet \mathrm{S04} \bullet \mathrm{ISR}$

$\mathrm{M06} = \mathrm{M05} \bullet \mathrm{S05} \bullet \mathrm{ISR}$

$\mathrm{M07} = \mathrm{M06} \bullet \mathrm{S06} \bullet \mathrm{ISR}$

$\mathrm{M08} = \mathrm{M07} \bullet \mathrm{S07} \bullet \mathrm{ISR}$

$\mathrm{M09} = \overline{\mathrm{MBF}} \bullet \mathrm{M05} \bullet \mathrm{M01}$ *

$\mathrm{M10} = \mathrm{M09} \bullet \mathrm{S09} \bullet \mathrm{ISR}$

$\mathrm{M11} = \mathrm{M10} \bullet \mathrm{S10} \bullet \mathrm{ISR}$

$\mathrm{M12} = \mathrm{S09} \bullet \mathrm{S10} \bullet \mathrm{S11} \bullet \mathrm{M09} \bullet \mathrm{ISR}$

$\mathrm{M13} = \mathrm{MHC} \bullet \mathrm{S12} \bullet \mathrm{ISR}$

$\mathrm{M14} = \mathrm{M13} \bullet \mathrm{S13} \bullet \mathrm{ISR}$

$\mathrm{M15} = \mathrm{S12} \bullet \mathrm{S13} \bullet \mathrm{S14} \bullet \mathrm{M12} \bullet \mathrm{ISR}$

$\mathrm{MHC} = \overline{\mathrm{CM2}} \bullet \mathrm{M12}$

* Boolean equation for input gate

Figure 4-13

SECTION 5
200 MAIN MEMORY

INTRODUCTION

The main memory is a magnetic core random access device which provides storage for instructions and data during the performance of programs. The memory unit supplied as part of the basic central processor has a capacity of 2048 characters, each of which is stored in a separate, addressable, memory location. This capacity may be expanded in modular increments by adding one 2,048 - character module, and additional 4,096 - character modules.

There are no reserved input/output areas in main memory, but an indexed addressing option provides the facility of using six groups of three memory locations each as index registers (octal locations 2 through 30). When these locations are not being used for indexing operations they can function as normal storage locations. A location in main memory consists of nine binary bits. Planes one through six are information; planes seven and eight are punctuation (word and item marks respectively). Plane nine is parity for the first six information planes.

The student is referred to Fundamentals of Digital Systems magnetics section, for a basic discussion of coincident current memory principles. This chapter is presented with the assumption that the reader is familiar with the information presented there.

Addressing (Single Module)

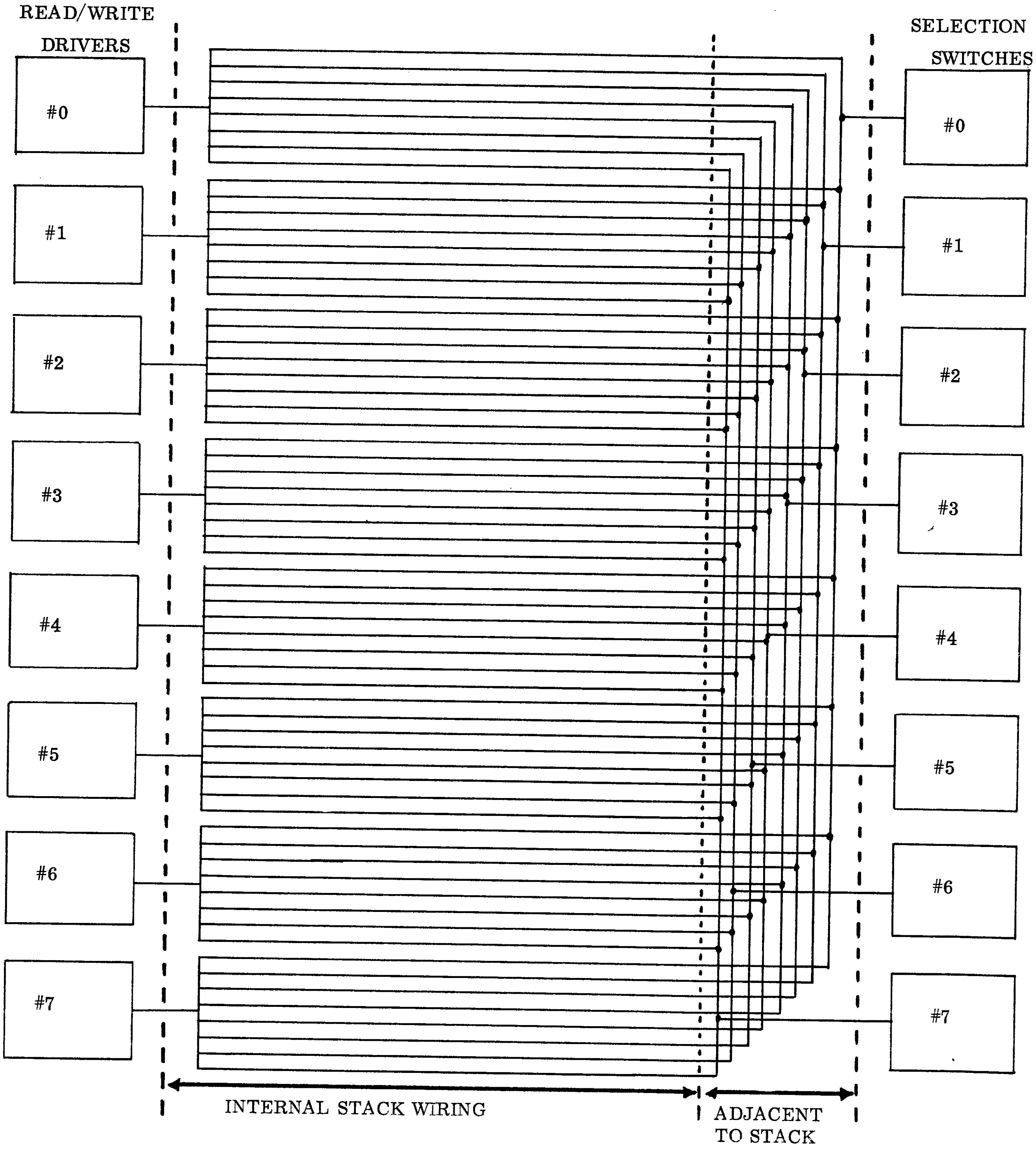
Each main memory module consists of 4096 locations, thus requiring a 64 x 64 matrix to define any one location. An axis of 64 ordinates is separated to eight groups of eight lines each. One group of eight lines is attached to one driver of this axis. (See Fig. 5-1). Choosing a driver then picks an eight line group, and it is up to the selection switches to choose one of those eight lines. Any one selection switch is tied to one line of each driver; so that when a switch is activated it,will allow current through one line of each driver. But since only one driver should be active at a time, the only line carrying current will be defined by the driver and the selection switch.

The above discussion is applicable to both the 'X' and 'Y' axis. Both have eight drivers and eight selection switches. Fig. 5-2 is a block diagram of the address selection for one main memory stack, showing the address register bit positions used to pick those elements that actively address the desired location. As an example,an individual 'X' selection switch is selected by a particular configuration of bits one, two, and three of the address register. An individual 'X' driver is selected by a particular configuration of bits four, five, and twelve of the address register,etc.

Addressing (All Stacks)

Refer to Fig. 5-3. As previously mentioned the selection of a driver picks a group of eight lines inside the stack, in the process of addressing a particular location. Note, however, in Fig. 5-3 each driver is shown to have four outputs, only one of which goes to the stack. The output labeled '0' can be traced back to a read and write generator (for stack "0") which is selected by a particular configuration of bits 13, 14, and 15 of the address register. If another read and write generator were chosen instead of the one shown there

STACK WIRING (X AXIS)



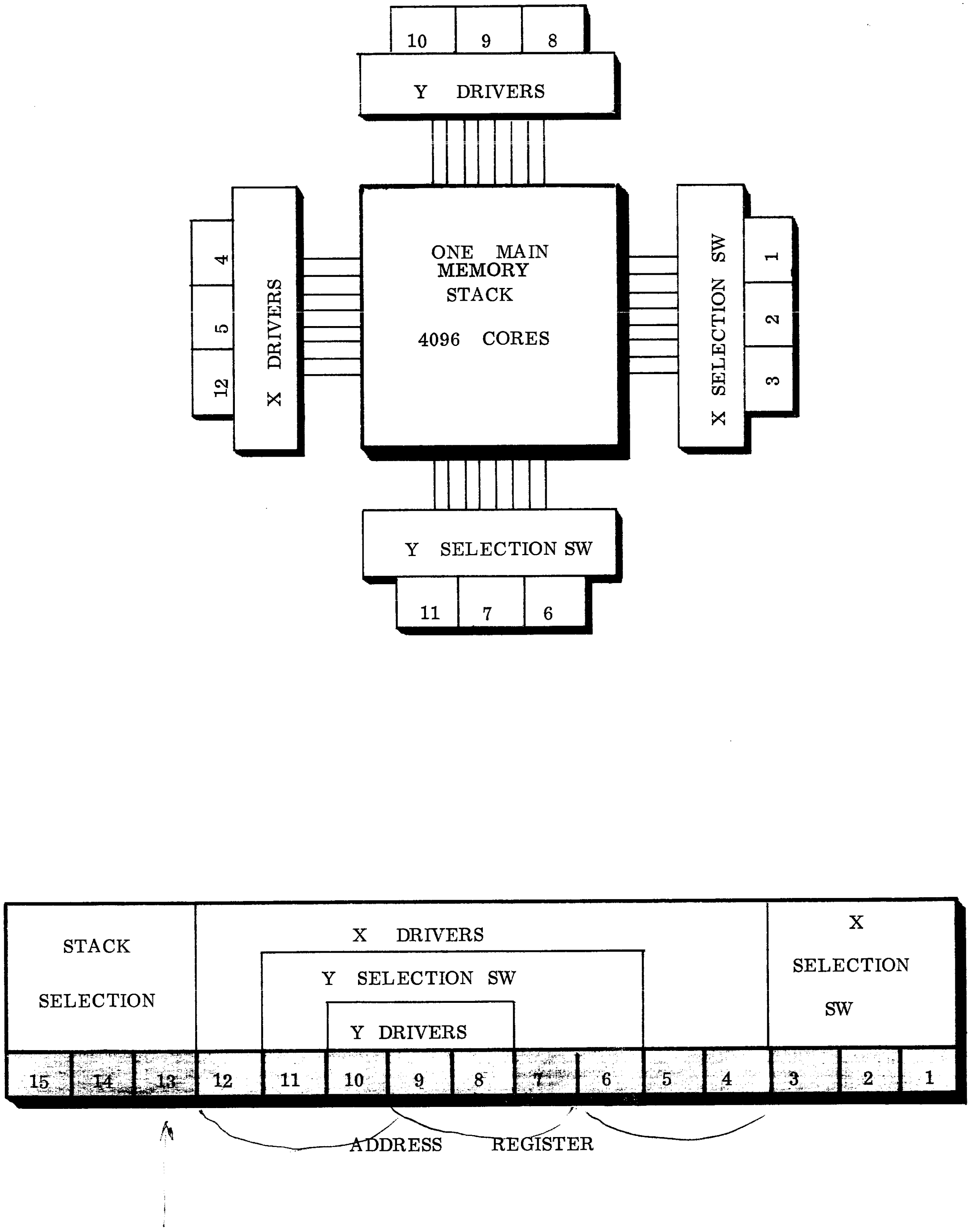

10
9
8
Y DRIVERS
4
5
12
X DRIVERS
ONE MAIN MEMORY STACK
4096 CORES
X SELECTION SW
1
2
3
Y SELECTION SW
11
7
6
STACK SELECTION
X DRIVERS
Y SELECTION SW
Y DRIVERS
X SELECTION SW
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
ADDRESS REGISTER

Fig. 5-2

would be no current through stack zero since the generators for that stack have been disabled. The generators chosen would provide current through the selected driver via another output to the appropriate stack. The selection switches are common to all stacks.

An important point to note is that the drivers do not originate read and write **currents.** Their function is to address 4096 locations. But the driver output to the stack is chosen by the active R/W current generators, which in turn are selected by the address register. So in effect the drivers can be thought to function as a switch (in 4096 loc. address), and also as a steering circuit (to choose the stack).

ADDRESSING COMPONENTS

The following discussion will refer to, and briefly explain, the schematics of the various component circuits used in addressing a particular location.

'S' Register

The 'S' **(address)** register determines which location in main memory is cycled by logically choosing the generators, drivers, and switches, which will be active. The S register is normally loaded from the control memory sense amps, but,depending on the instruction,can be loaded in part from the V register as well. Operand addresses are stored in control memory (see **section 4).**Another input to the S register is the control panel **which allows the** interrogation or alteration of a main memory location. This 15 bit register is identified by the code CBS**10, where ** represents bit positions 1 through 15. Refer to fig. 5-**12** for the package schematic of one register bit position. Note that the output of Q4 is "anded" with pin 32 so that once the flop is set it stays set until the pin 32 input goes low.

Memory Address Registers

It will be seen later that two memory addresses registers must be used because the 'S' register is cleared in preparation for a new address before the write cycle is completed. This requires that one memory address register be recirculated until the write cycle is complete.

MAS0100 = I = CBS011R

MAS0110 = CBS011R

This "read" memory address register is just a repetition of the S register, and will be cleared at the same time (T08).

MAS0130 = MAS0110•MAT0210
+ MAS0130•MAT0120

MAS0140 = I = MAS0130

The "write" memory address register is set at T02 with the corresponding bit position of the "read" memory address register and recirculates on itself until T01.

Bit positions 1 through 11 of both read and write memory address registers are identified with an MA prefix, in their function names, to designate their necessity in addressing the basic memory of 2048 locations. Bit position 12 has a prefix of MB to indicate that it is used in the first memory option (additional 2048 locations). Bit position 13 has a prefix of MC; bit 14 has a prefix of MD; and bit 15 has a

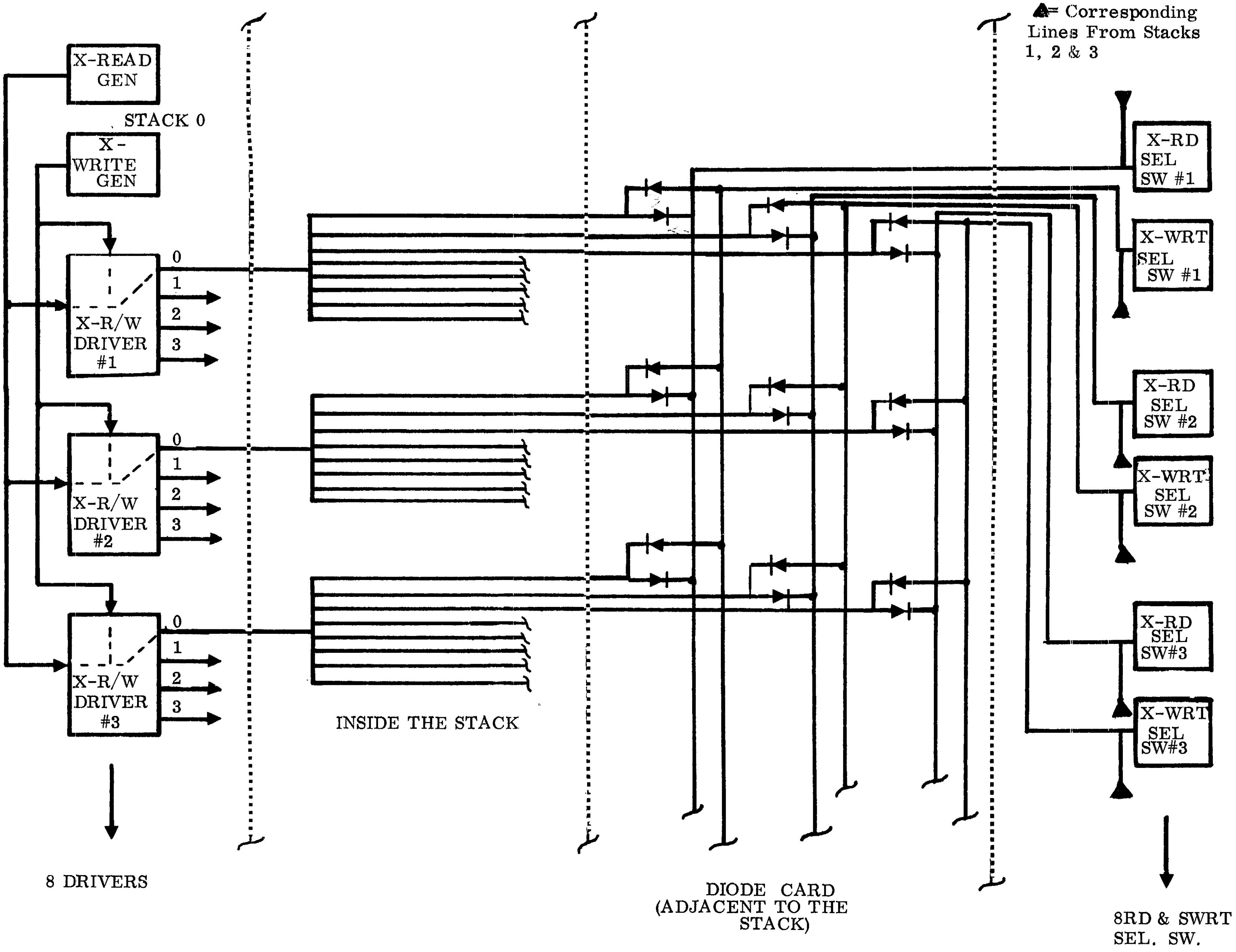

▲= Corresponding Lines From Stacks 1, 2 & 3
X-READ GEN
STACK 0
X-WRITE GEN
X-R/W DRIVER #1
X-R/W DRIVER #2
X-R/W DRIVER #3
0
1
2
3
8 DRIVERS
INSIDE THE STACK
DIODE CARD (ADJACENT TO THE STACK)
X-RD SEL SW #1
X-WRT SEL SW #1
X-RD SEL SW #2
X-WRT SEL SW #2
X-RD SEL SW#3
X-WRT SEL SW#3
8RD & SWRT SEL. SW.

Fig. 5-3

prefix of MF. In general MA and MB prefixes refer to stack '0', MC to stack '1', MD to stack '2', ME to stack '3'...etc.

Logic Board Diagram

Current Generators

Each axis requires a read and write current generator for each stack. Each package (2MGB2 - see fig 5-14) contains a read and a write generator. The generators are shown as having two outputs in the LBD's; a current output; and a voltage reference output. Although this discussion is concerned with the current output, the circuit is indexed in the LBD's by the voltage reference output; therefore,both are shown here.

'X' Axis Generators For Stack 0

(MAVXR10)	MARGX10	=	MFS1510	(MAVXW10)	MAWGX10	=	MAWG010
		+	MDS1410			+	MFS1530
		+	MCS1310			+	MDS1430
		+	MARG010			+	MCS1330

All four inputs must be low before a generator is active. RGO and WGO are timing functions. Note the use of the read (10) and write (30) memory address registers, on the input gates of the read and write generators.

Read/Write Drivers

There are eight 'X' R/W drivers and eight 'Y' R/W drivers. Each driver has four outputs. The drivers are indexed in the LBD's by MARX$_{3}^{0}$10 or MARY$_{7}^{0}$10 for the X and Y axis respectively. (These named drivers are used to drive the basic memory of 2048 locations. The additional 'X' drivers required for the first option of an additional 2048 locations are MBRX$_{7}^{4}$10.) All of the above named functions are the first output of the drivers and go to stack "0". The outputs feeding stack '1' are prefixed MC; those feeding stack '2' are prefixed MD; and those feeding stack '3' are prefixed ME.

See Fig 5-16 for the R/W driver schematic. Pins 32, 43, 33, and 44 are the read address register and read timing input. Pins 45, 47, 15 and 13 are the read current generator inputs. Output pins are 46, 36, 14, and 24. The write side is similar to the read except that the read selection is by an 'and' gate, and write selection by an 'or' gate. Thus for one all inputs must be high,and for the other all gates must be low.

Selection Switches

Refering to Fig. 5-3, it can be seen that a line from a driver goes through a diode card to both a read and a write selection switch. Thus each axis must have eight read and eight write selection switches.

'X' Axis Switch #1 Read			'Y' Axis Switch #2 Write		
MASX1RO	=	MARS010	MASYZW0	=	MAS1130
	+	MAS0310		+	MAS0740
	+	MAS0210		+	MAS0630
	+	MAS0100		+	MAWS010

Functions RS0, and WS0 are timing functions. Note that a read switch is designated with the suffix R0 while the write switch is designated by the suffix W0. The three letter mnemonic indicates; first letter - selection switch; second letter - axis; and third letter - switch number. Refer to Fig. 5-17 for the schematic of the 2MSS4 package. There are two read and two write switches for each package, and the only difference between the two types is the direction of current that the output transistor will allow.

INFORMATION HANDLING COMPONENTS

Memory read out will drive the cores of the selected location to zero. Those cores that were in a "one" state will change magnetic polarity inducing a voltage into the sense windings for their respective planes. Sense amplifiers detecting this voltage will be set and recirculate until a reset pulse forces all sense amps to zero in anticipation of the next read cycle.

Fig. 5-4 shows the data flow from core to central processor and back to core. One group of sense amps is fed by two stacks (the stacks serviced are represented by the prefix of the sense amp name). The two sense amp groups in drawer M1 feed a collector register (MAMA$^{1}_{9}$10) which will simply repeat the contents of the sense amp in use. When the subcommand CAM2N10 (memory to N register) is high, the contents of the collector register of either memory drawer is gated to the N register (CAN0$^{1}_{9}$10). This register is often referred to as the MLR or memory local register.

The old contents of the addressed memory location is now in both the sense amps and the MLR. The cores in the addressed location are all zero. During the write cycle, drive current will attempt to force all addressed cores to a one state, but will be prevented in those planes whose inhibit driver is active. The inhibit drivers are fed by the negations of the sense amps, and by the memory N register. Since the memory N register is a bank of inverters fed by the MLR assertions, they are in reality just the negation of the MLR. Whether writing into memory from the sense amps or the MLR, a high input (indicating a zero) will activate the inhibit driver.

The punctuation bits, by different sub-commands are handled independantly of the information and parity bits. Main memory is checked for accuracy by an odd parity check of the six information bits and plane 9. The parity checker is used, as the name implies, to determine if the character read from memory has correct parity. When writing in new information, the parity checker generates correct parity and sends it to the plane 9 inhibit driver.

Sense Amps

Fig. 5-20 is a schematic of a sense amplifier package. The sense amps are fed by corresponding sense windings from two stacks. The input pins are 47 and 37 and 36 and 46. Q8 is held reverse biased until a strobe pulse comes in at pin 17, and a "one" has been sensed, cutting off Q7. When Q8 is forward biased the flip flop formed by Q11 and Q12 is set (pin 16 = +5v) and will stay set until the reset pulse at pin 43 causes the flip flop to change state (pin 15 = +5v)

The LBD's represent the sense amps as follows:

$$\begin{aligned} \text{MASA110} &= \text{MARST10, MASTR10}\\ &+ \text{MAPL110}\bullet\text{MAPL100}\\ &+ \text{MCPL110}\bullet\text{MCPL100} \end{aligned}$$

A comma indicates a non-logical relation, and this is true as seen from the schematic. RST is the reset pulse, and STR is the strobe pulse. The sense winding input is not really "anded" either, but can be thought of as an input transformer secondary.

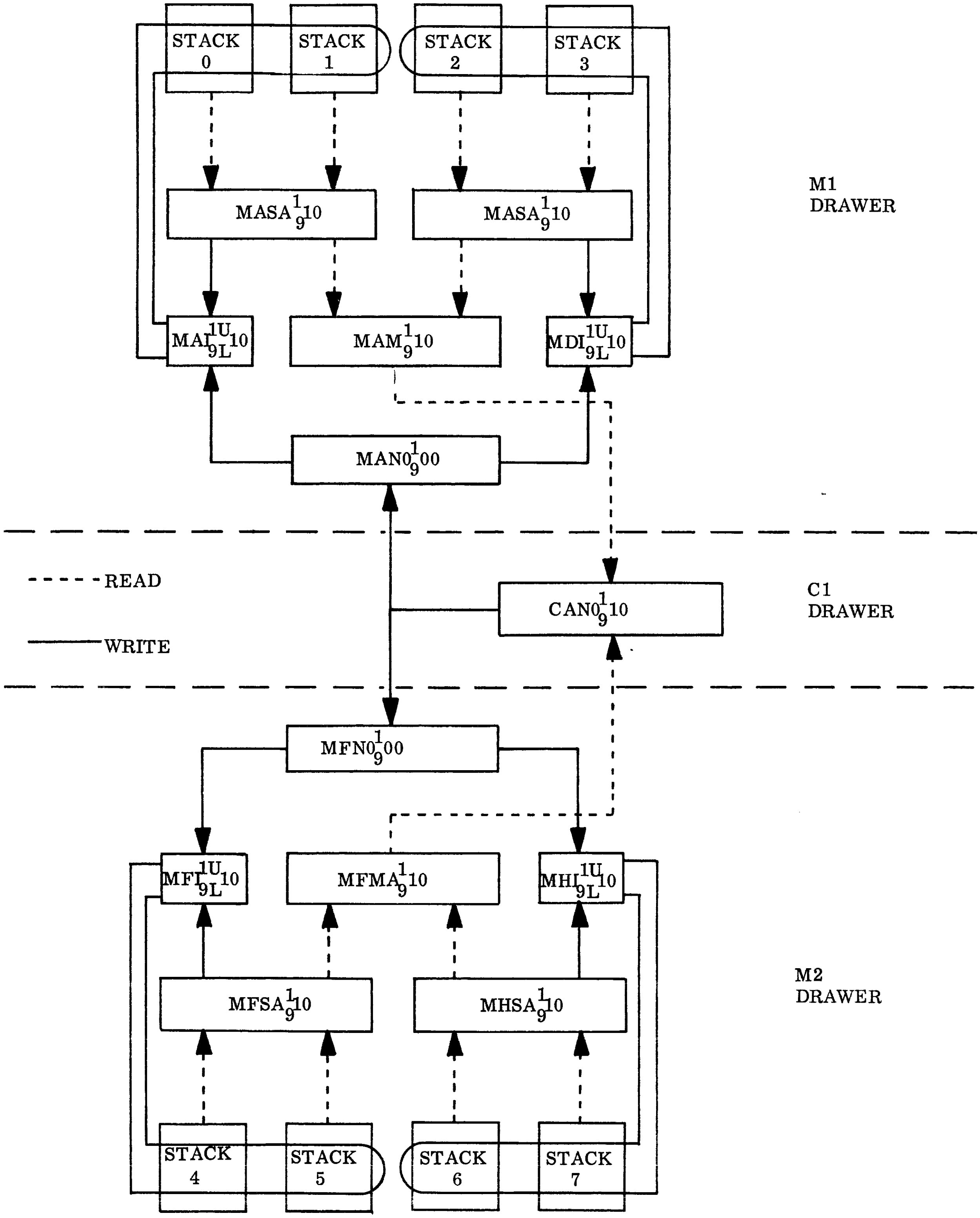

STACK 0
STACK 1
STACK 2
STACK 3
MASA$^1_9$10
MASA$^1_9$10
M1 DRAWER
MAI$^{1U}_{9L}$10
MAM$^1_9$10
MDI$^{1U}_{9L}$10
MAN0$^1_9$00
READ
WRITE
CAN0$^1_9$10
C1 DRAWER
MFN0$^1_9$00
MFI$^{1U}_{9L}$10
MFMA$^1_9$10
MHI$^{1U}_{9L}$10
MFSA$^1_9$10
MHSA$^1_9$10
M2 DRAWER
STACK 4
STACK 5
STACK 6
STACK 7

Fig. 5-4

'N' Register (MLR)

```
CAN0110 = CAHNR10, CAM2N10•MAMA110•CAT0510
        + CAM2N10•MFMA110•CAT0550
        + CAA2N10•CASD110
        + CAPTN10•CAP0110
        + CEG2N10•CACT350•CEG0110
```

The comma after CAHNR10 in the first gate indicates a non-logical relation. HNR is a recirculation function, which is really or'd with all the set gates. Refer to Fig. 5-15, the MLR package schematic. HNR comes in on pin 32.

Excluding HNR the first gate is the input to the MLR bit one, from plane one, of any stack in memory drawer one. The second gate is the input from drawer two (see Fig. 5-4). M2N is a sub-command, and T05 a timing function.

The third gate is the input from the arithmetic unit. A2N is a sub-command saying "accumulator to "N" register" SD1 is an accumulator output.

The fourth gate is the input from the P register, which is a buffer for all peripheral input data. P2N is the sub-command and P01 the first bit position of the P register.

The last gate is for the purpose of gating in the G register which is the edit register.

The student is invited to investigate the differences between the information bits and punctuation bits of the N register, noting especially the different sub-commands used to gate in the sense amps.

CAN0910, the parity bit, can only be set by the sense amps. The contents of the six information bits are checked for odd parity by the functions CAPGL10 and CAPGU10. PGL checks N01, N02, and N03, for an odd number of bits. PGU checks N04, N05, and N06 for an odd number of bits. By "exclusive or" gating, PGU and PGL will set CAP2M10 if the information has an odd number of bits. If P2M is high, indicating odd parity, then N09 must be low. If P2M were low, then N09 must be high to insure odd parity. This condition is checked on the input gates to CAPER10. PER is the parity error flop. It is set when the six information bits plus parity is an even number, and CAPCA10 is true. PCA (parity check allow) will inhibit setting a parity error when memory readout is inhibited ($\overline{\text{M2N}}$) and during a delivery cycle from N to memory (CADE310). When memory readout is inhibited to clear the information and parity bits to zero, a parity error would be set if not for PCA. The same is true for a delivery cycle. Parity is generated for the new information gated into N , but since N09 cannot be set by the parity generator, an error would be set except for PCA.

One point that is sometimes confusing to the new student, is how proper parity is written into memory. The following pertinant facts should clear up any uncertainty. The write cycle is trying to drive all cores to one's; a core is prevented from switching if the inhibit driver is active; the inhibit driver is active when its input is high; and CAP2M10,the input to the plane 9 inhibit driver, indicates an odd number of information bits.

Inhibit Drivers

The purpose and function of the inhibit drivers have been explained already, but there are some details yet to be cleared up. Refer to Fig. 5-**21** for the schematic of an inhibit package. Pins 24 and 48 are the "upper" inhibit winding output. Pins 14 and 47 are the "lower" inhibit winding output. Each inhibit winding threads one half of a plane in two separate stacks. The inhibit driver then, via both output windings, inhibits an entire plane in two main memory stacks. The input is two four legged "and" gates (pin 33 is not used).

S15 S14 S13

$MASS010 = \overline{MFS15} . \overline{MDS14}$ → STACK 0 1

$MDSS110 = \overline{MFS15 . MDS14}$

a zero in read no inhibit

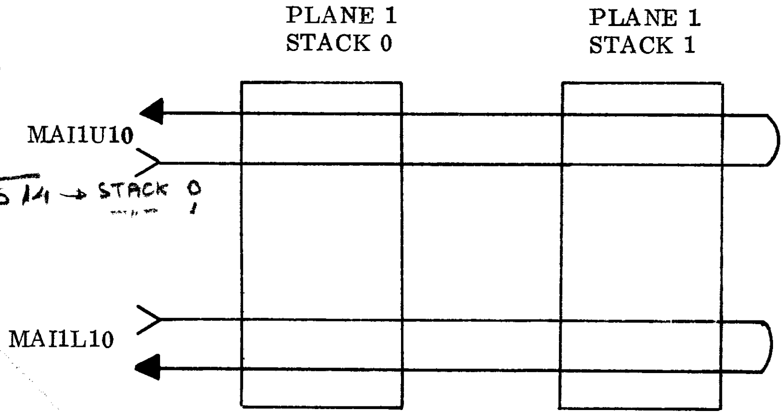


FIG. 5-5

MAI1U10

MAI1L10 = MAIML10•HAN0100•MAID010•MASS010
+ MAIML20•MASA100•MAID010•MASS010

The top gate is used when writing from the N register, while the bottom is used for writing from the sense amps. ID0 is a timing function, and SS0 is a function of the address register permitting only one set of inhibit drivers (one set for two stacks) to be active. $\overline{N01}$ and $\overline{SA1}$ are negations of the MLR and sense amps respectively.

Choice between the gates then depends on IML for the N register and $\overline{IML}$ for the sense amps.

MAIML10 = CANLD10

CANLD10 = CAINL10

CAINL10 is a sub-command generated during a delivery cycle, that is to say,when new information is being written into memory.

Main Memory Timing

Synchronization of memory to C.P., is the function of three times, MAT0110, MAT0210, and MAT0510. Refer to Fig. 2-3 to see the relationship between the C.P. and memory clocks. The read gates are synched by T01, write address register by T02, and write gates'by T05.

Refer to Fig. 5-6, for main memory timing waveforms. The memory clock outputs (T01, T02, and T05) are derived in the same manner as the C.P. ring counter times. For a review of the mechanics of timing generation, the student is referred to **section 2.**

Read and write timing and control gates are the outputs of 2MPG2 packages: The one exception being MASTR10 which is the output of a 2MSG2 package. However, the only difference between the two packages is the size of four components, C10, C14, R37, and R43. Because of this only the print for the 2MPG2 package is shown in Fig. **5-23.**

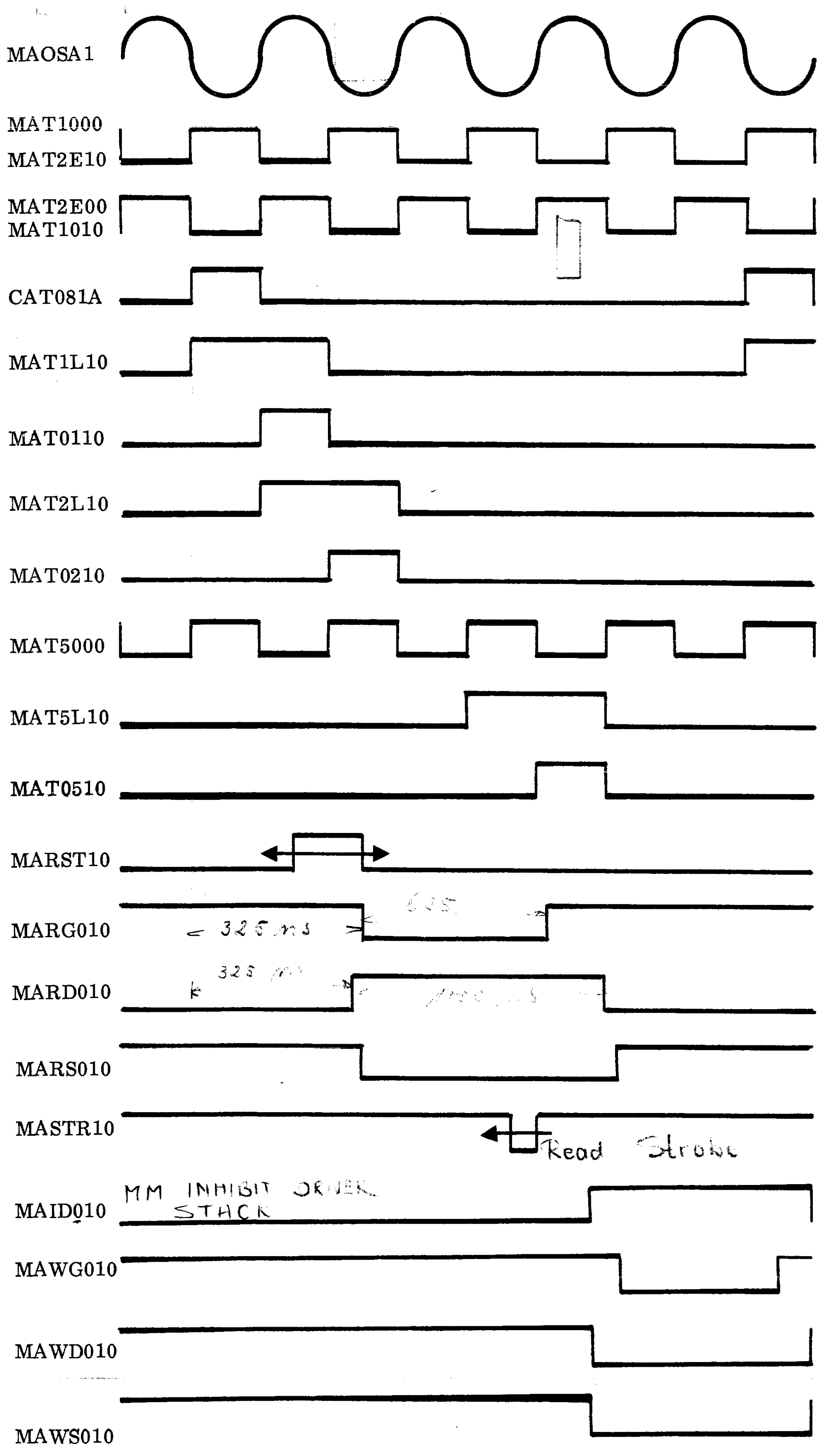
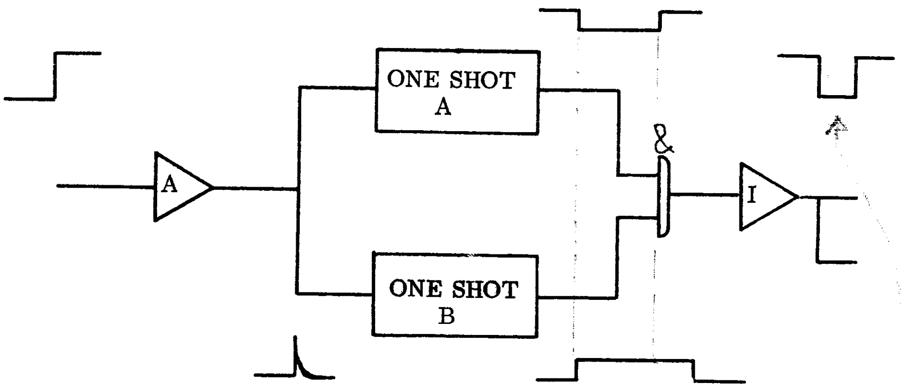

MAOSA1
MAT1000
MAT2E10
MAT2E00
MAT1010
CAT081A
MAT1L10
MAT0110
MAT2L10
MAT0210
MAT5000
MAT5L10
MAT0510
MARST10
MARG010
625
325 ns
325 ns
MARD010
400 ns
MARS010
MASTR10
Read Strobe
MM INHIBIT DRIVER STACK
MAID010
MAWG010
MAWD010
MAWS010

Fig. 5-6



2MPG2

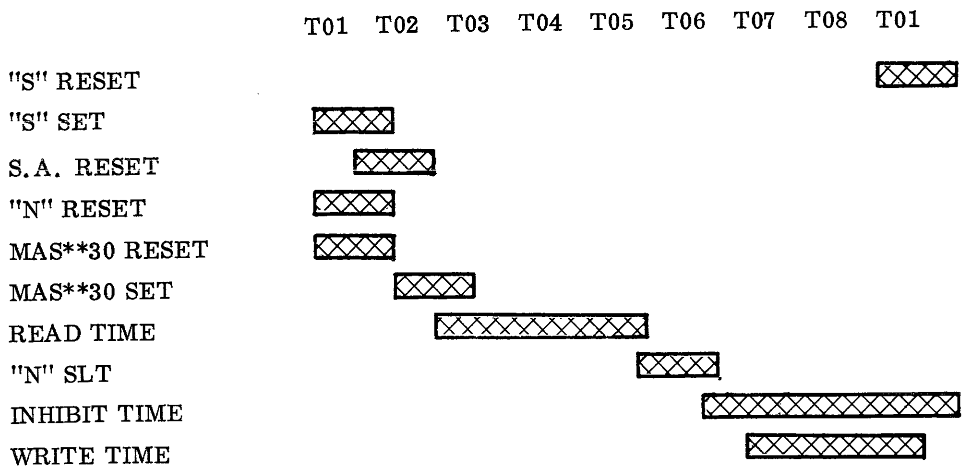
Fig. 5-7

Fig. 5-7 is a block diagram of the 2MPG2 package, showing pertinent waveforms. In brief, it requires a positive going input which is differentiated and triggers two one shots adjusted to different durations, and having opposite polarity outputs. When the 'and' gate inputs are both high there will be an output of the same duration but opposite polarity. The output leading edge is determined by the duration of one shot 'A', while the output duration is determined by the duration of one shot 'B'. Not shown in Fig. 5-7 is an inverter circuit which is separate from the pulse generator. This circuit is used to invert the pulse generator output in those cases where a positive going gate is required.

Derivation of the memory gates is left to the student. The following table describes their use and characteristics. For example, MARST10 begins 200 NS after the leading edge of T01 and has a duration (width) of 200 NS. Refer to Fig. 5-6 for a graphic representation of these gates.

FUNCTION	REFERENCE	POS.	WIDTH	USE
MARST10	T01	200	200	Sense Amp Reset
MARG010	T01	325	650	Read Current Gen. Timing
MARD010	T01	325	1000	R/W Driver Timing (Read)
MARS010	T01	325	1000	Read Sel. SW. Timing
MASTR10	RG0	≈ 450	120	Sense Amp Strobe
MAID010	T05	250	800	Inhibit Driver Timing
MAWG010	T05	280	550	Write Current Gen. Timing
MAWD010	T05	250	900	R/W Driver Timing (Write)
MAWS010	T05	250	900	Write Sel. SW. Timing

Fig. 5-8 shows the relative timing and sequence of events during one memory cycle. Points to be noted are; the S register reset time (during the write cycle) requiring another address register which will not be reset until the completion of the write; the N register must be set by time 6, since the write is beginning and the inhibit drivers must overlap the write gate to insure stable operation.



* MEMORY ADDRESS REGISTER BIT POSITION

Figure 5-8

Summary

The S register in the CP is transferred to a memory address register in the memory drawers. Specific bit positions of the address register are used to select a particular current generator, driver, and selection switch. Whether the chosen component is for read or write cycles, is determined by the memory clock 'gate' signals,which also provide the required synchronization to the rest of the central processor. The student is referred to the charts on Fig 5-10 and 5-11 for information on selection of specific memory components.

Fig. 5-9 shows the path of information read out of memory into the sense amps; the gating to the N' register; parity checker inputs and outputs; and the two 'write' paths with their respective subcommands. In general, if the sub-command gating the sense amps to N , is true, so also will be the sub-command gating the sense amp to the inhibit drivers. If sense amp to N register transfer is inhibited, the sub-command gating N to the inhibit drivers is generally high.

If a parity error occurs, the function CAPER10 will set CPSV310, which forces a V3 cycle by resetting the recirculation function of all machine cycles except E3C. EAC is inhibited by CPRUN00, keeping the machine stopped.

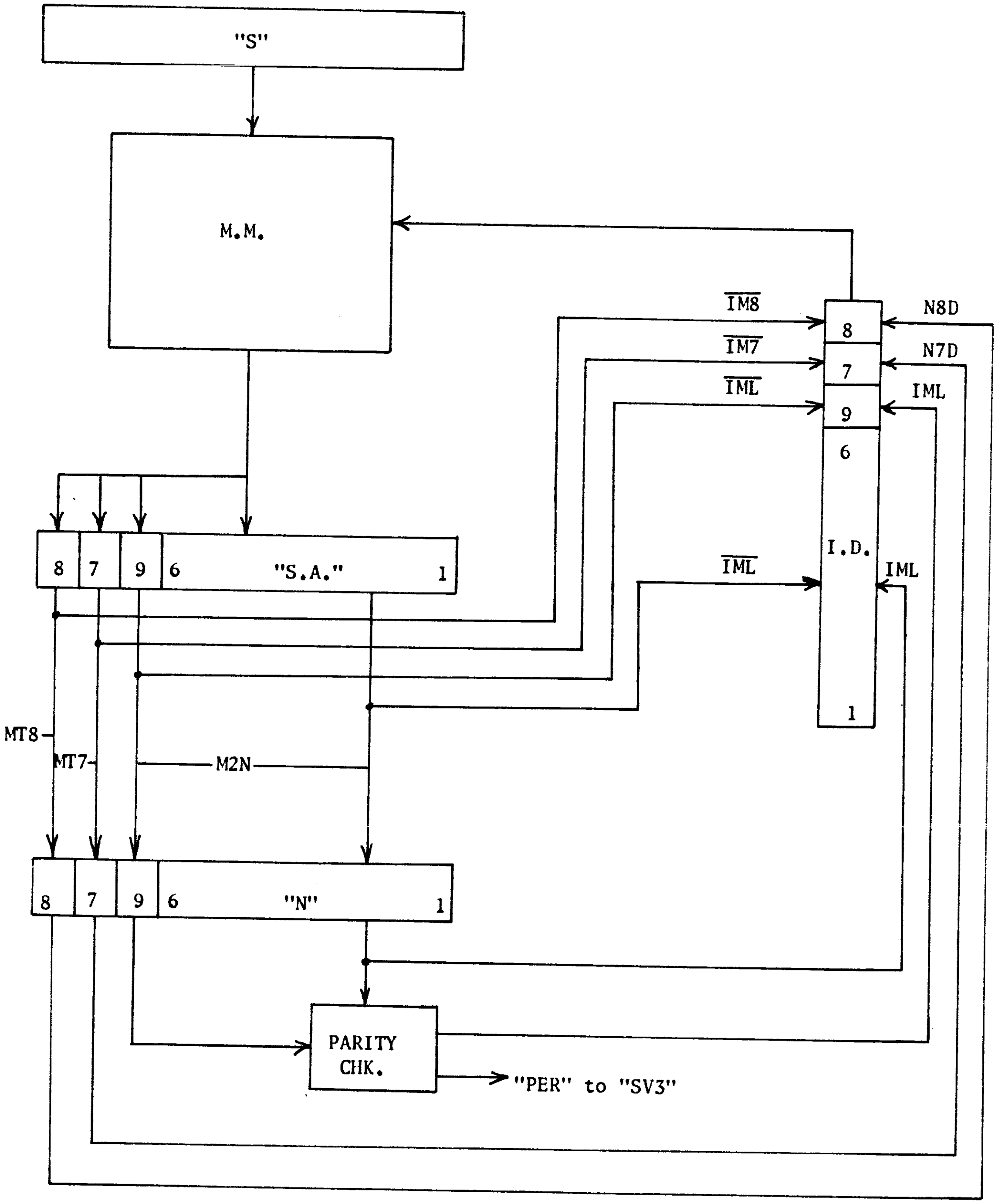

"S"
M.M.
$\overline{IM8}$
$\overline{IM7}$
$\overline{IML}$
N8D
N7D
IML
8
7
9
6
I.D.
1
$\overline{IML}$
IML
8
7
9
6
"S.A."
1
MT8
MT7
M2N
8
7
9
6
"N"
1
PARITY CHK.
"PER" to "SV3"

Figure 5-9

MEMORY ADJUSTMENTS

Memories are such sensitive devices that it as nearly impossible to manufacture them exactly identical. Even in one memory stack individual cores may have slightly different characteristics due to a wire not threading a core exactly through the center or due to the proximity of shielding,or due to a variety of physical imperfections. Rather than attempt to design memory circuits that would work pretty good with any memory stack, the memory circuits are usually adjustable to enable them to be set for optimum performance of the memory stack in use.

The first adjustments that should be made are the ones that control the amounts of current to the stack (RCG, WCG and Inhibit drivers). The object here is to vary the resistance in this package to compensate for variations in the wire through the stack. All inhibit drivers should be set to the same amplitude of current.

Sense amplifiers have two adjustments, balance and threshold. The threshold adjustment is used to vary the sensitivity of the sense amp. The sense amplifier should accept the smallest possible ones without sensing the noise of a zero as a one. When making this adjustment remember that once a bit is sensed by a sense amplifier, the amplifier may feed to the inhibit driver in the normal fashion and cause legitimate ones to be sensed during successive memory cycles. The balance adjustment should make the sense amplifier equally sensitive to positive and negative pulses. The sense amplifier must be a differential amplifier because the sense winding threads half the cores in the opposite direction to reduce noise. Therefore, it is important for this adjustment to be made while information is cycling through as many different locations as possible. The distance of a core from the end of the sense winding may also affect the magnitude of the pulse at that end of the winding.

The strobe should be positioned (in time) at the center of its operable limits. This point is found by making the strobe occur earlier and noting when the first parity failure occurs . Then the strobe should be delayed until a failure occurs at this limit. The method in use to determine where the limits are is actually more complex than the method described here, but idea is the same. One of the complexities is caused by the fact that it is possible in some memory stacks (various manufacturers are used) to move the strobe to a point where the magnetic field induced by the read current pulse itself can be sensed as a one.

Specific values and procedures for the memory adjustments are given in the service manual

X AXIS RD/WRT CURRENT GENERATORS

STACK SEL.			LOCATION	RD GEN	WRT GEN
S15	S14	S13			
0	0	0	MIA5F	MAVXR10 MARGX10	MAVXW10 MAWGX10
0	0	1	MIA5G	MCVXR10 MCRGX10	MCVXW10 MCWGX10
0	1	0	MIB5L	MDVXR10 MDRGX10	MDVXW10 MDWGX10
0	1	1	MIB5M	MEVXR10 MERGX10	MEVXW10 MEWGX10

X AXIS RD/WRT DRIVERS

DRIVER SEL.			DRIVER	LOCATION
S12	S05	S04		
0	0	0	M*RX010	MIA5K
0	0	1	M*RX110	MIA5L
0	1	0	M*RX210	MIA5M
0	1	1	M*RX310	MIA5N
1	0	0	M*RX410	MIA5P
1	0	1	M*RX510	MIA5Q
1	1	1	M*RX610	MIA5R
1	1	1	M*RX710	MIA5S

X SELECTION SWITCHES

S03	S02	S01	READ SW	LOCATION-PIN	WRT SW.	LOCATION-PIN
0	0	0	MASXORO	MIA7S-23	MASXOWO	MIA7S-37
0	0	1	MASX1RO	MIA7S-12	MASX1WO	MIA7S-47
0	1	0	MASX2RO	MIA7T-23	MASX2WO	MIA7T-37
0	1	1	MASX3RO	MIA7T-12	MASX3WO	MIA7T-47
1	0	0	MASX4RO	MIA7U-23	MASX4WO	MIA7U-37
1	0	1	MASX5RO	MIA7U-12	MASX5WO	MIA7U-47
1	1	0	MASX6RO	MIA7V-23	MASX6WO	MIA7V-37
1	1	1	MASX7RO	MIA7V-12	MASX7WO	MIA7V-47

Y AXIS R/W CURRENT GENERATORS

STACK SEL.			LOCATION	RD GEN	WRT GEN
S15	S14	S13			
0	0	0	MIA5C	MAVYR10 MARGY10	MAVYW10 MAWGY10
0	0	1	MIA5D	MCVYR10 MCRGY10	MCVYW10 MCWGY10
0	1	0	MIB5H	MDVYR10 MDRGY10	MDVYW10 MDWGY10
0	1	1	MIB5J	MEVYR10 MERGY10	MEVYW10 MEWGY10

Y AXIS R/W DRIVER

DRIVER SEL.			DRIVER	LOCATION
S10	S09	S08		
0	0	0	M*RY010	MIA5T
0	0	1	M*RY110	MIA5U
0	1	0	M*RY210	MIA5V
0	1	1	M*RY310	MIB5A
1	0	0	M*RY410	MIB5B
1	0	1	M*RY510	MIB5C
1	1	0	M*RY610	MIB5D
1	1	1	M*RY710	MIB5E

Y SELECTION SWITCHES

S11	S07	S06	READ SW	LOCATION-PIN	WRT SW.	LOCATION-PIN
0	0	0	MASYORO	MIB7A-23	MASYOWO	MIB7A-37
0	0	1	MASY1RO	MIB7A-12	MASY1WO	MIB7A-47
0	1	0	MASY2RO	MIB7B-23	MASY2WO	MIB7B-37
0	1	1	MASY3RO	MIB7B-12	MASY3WO	MIB7B-47
1	0	0	MASY4RO	MIB7C-23	MASY4WO	MIB7C-37
1	0	1	MASY5RO	MIB7C-12	MASY5WO	MIB7C-47
1	1	0	MASY6RO	MIB7D-23	MASY6WO	MIB7D-37
1	1	1	MASY7RO	MIB7D-12	MASY7WO	MIB7D-47

*A + B	=	STACK #0
C	=	STACK #1
D	=	STACK #2
E	=	STACK #3

Fig. 5-11

INHIBIT DRIVERS

N REGISTER INPUT	SA INPUT	DRIVERS		LOCATION
MAN0100	MASA100	MAI1U$^{10}_{00}$	MAI1L$^{10}_{00}$	MIA7G 24/14 48/47
2	2	2	2	MIA7H
3	3	3	3	MIA7J
4	4	4	4	MIA7K
5	5	5	5	MIA7L
6	6	6	6	MIA7M
7	7	7	7	MIA7N
8	8	8	8	MIA7P
9	MASA900	MAI9U$^{10}_{00}$	MAI9L$^{10}_{00}$	MIA7Q
1	MDSA100	MDI1U$^{10}_{00}$	MDI1L$^{10}_{00}$	MIB7F
2	2	2	2	MIB7G
3	3	3	3	MIB7H
4	4	4	4	MIB7J
5	5	5	5	MIB7K
6	6	6	6	MIB7L
7	7	7	7	MIB7M
8	8	8	8	MIB7N
MAN0900	MDSA900	MDI9U$^{10}_{00}$	MDI9L$^{10}_{00}$	MIB7P

MAI1U10 - INHIBITS STACK #1
MAI1L10 - INHIBITS STACK #0
MDI1U10 - INHIBITS STACK #3
MDI1L10 - INHIBITS STACK #2
MASS010 - $\overline{\text{S15}}$ and $\overline{\text{S14}}$
MDSS110 - $\overline{\text{S15}}$ and $\overline{\text{S14}}$

MASS010

MDSS110

Fig. 5-11

SENSE AMPS

STACK INPUT	STACK INPUT	SENSE AMP	LOCATION PIN 15 = 00, PIN 16 = 10
A-Plane 1	C-Plane 1	MASA110	M1AIG
2	2	2	M1AIH
3	3	3	M1AIJ
4	4	4	M1AIK
5	5	5	M1AIP
6	6	6	M1AIQ
7	7	7	M1AIR
8	8	8	M1AIS
A-Plane 9	C-Plane 9	MASA910	M1AIT
D-Plane 1	E-Plane 1	MDSA110	MIBIB
2	2	2	MIBIC
3	3	3	MIBID
4	4	4	MIBIE
5	5	5	MIBIF
6	6	6	MIBIG
7	7	7	MIBIH
8	8	8	MIBIJ
D-Plane 9	E-Plane 9	MDSA910	MIBIK

STROBE - PIN 17------------MASTR10 & MDRST10
RESET - PIN 43------------MARST10 & MDRST10

THEORY OF OPERATION 2MMR4

Main Memory register

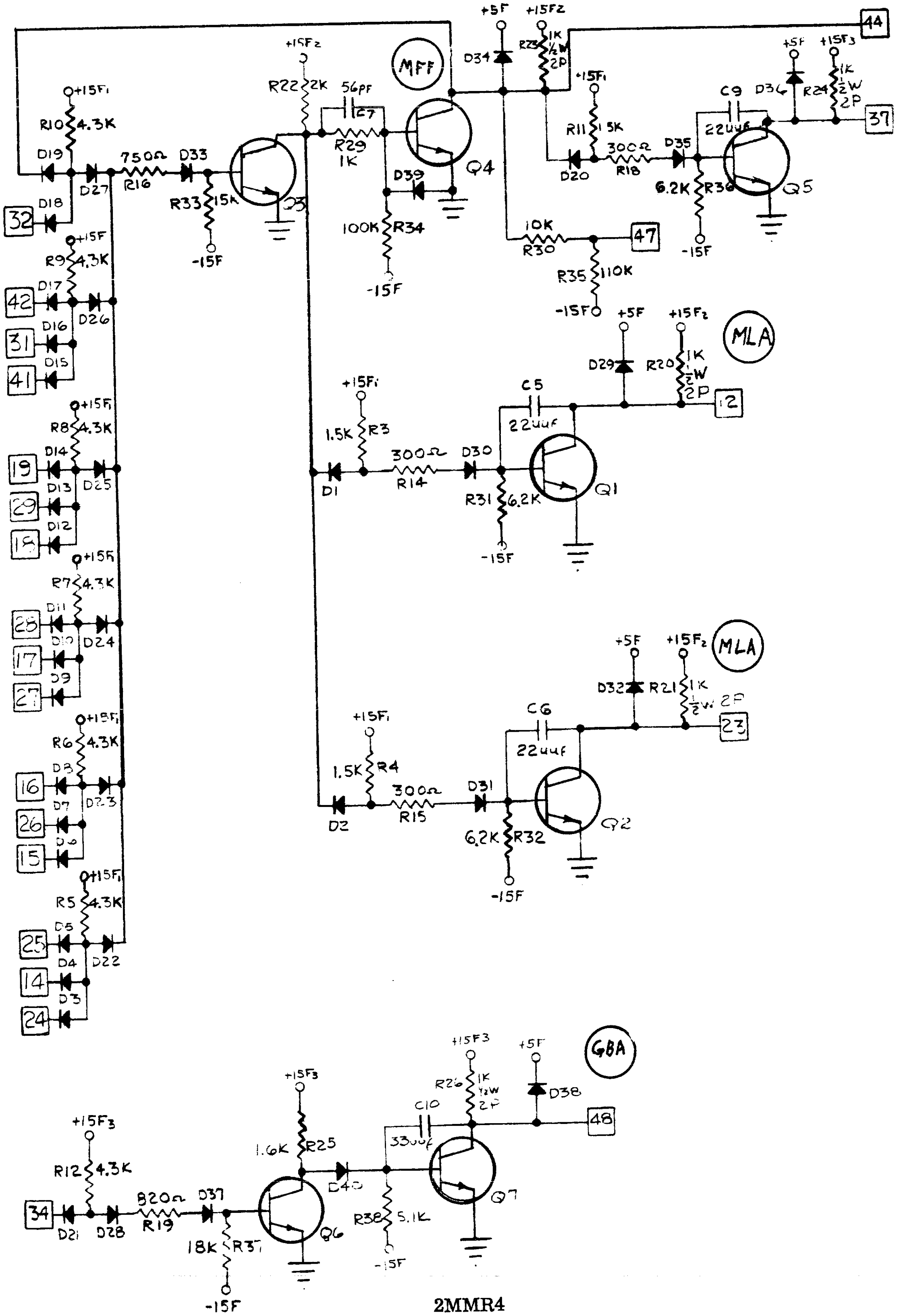
The input gating structure consists of five set gates and one recirculation gate (pin 32 is normally at +5 and will keep the output high once it is set). When Q3 is cut off, the output of the flop is ground, and Q4, Q1 and Q2 will be on. Q5 is off and the collector of Q5 is clamped to +5 volts.

When one input gate comes high, Q3 turns on and its collector goes to ground. This will turn off Q4. D39 prevents the base of Q4 from going more negative than -0.7 volts during switching. D20 is cut off and Q5 is on; the collector of Q5 will be at ground. Since the collector of Q3 is at ground, D1 is forward biased and the base of Q1 is at about -0.7 volts, through R14, D30 and R31 to -15 volts. Hence, Q1 is off. The stage associated with Q2 is identical to the Q1 stage.

GBA

When the input is at ground, D21 is forward biased and the base of Q6 stays at approximately -1.6 volts (determined by D28, R19, D37 and R37). Q6 is cut off and its collector circuit provides forward bias for Q7. With Q7 saturated the output at pin 48 is ground.

When the input switches to +5 volts, Q6 will be forward biased and Q7 will cut off allowing the output to go to +5 C10 helps to slow down the rise and fall time of Q7.



2MMR4

Fig. 5-12

THEORY OF OPERATION 2MGB2

The Read/Write Current Generator package contains one read generator and one write generator, each driving 8 read or write drivers. The R/W generator sends current through one R/W driver associated with the particular R/W line being selected at a given time.

The Read generator is controlled by a negative level "and" gate. The normally saturated transistor, Q2, is turned off when all of the input pins 17, 42, 43 and 33 are simultaneously at ground potential. With Q2 in the "off" state (momentarily), we have an effective constant current source of 15 ma flowing through R13 and the coupling capacitor C11. This current, minus the small current shunted by R29, is available to flow in the base of Q4, which is initially in the "off" condition. While Q4 is in its active region, the base current required is determined by the collector current in Q4 and the Beta of Q4. Therefore, during the entire rise time of Q4, a constant current of

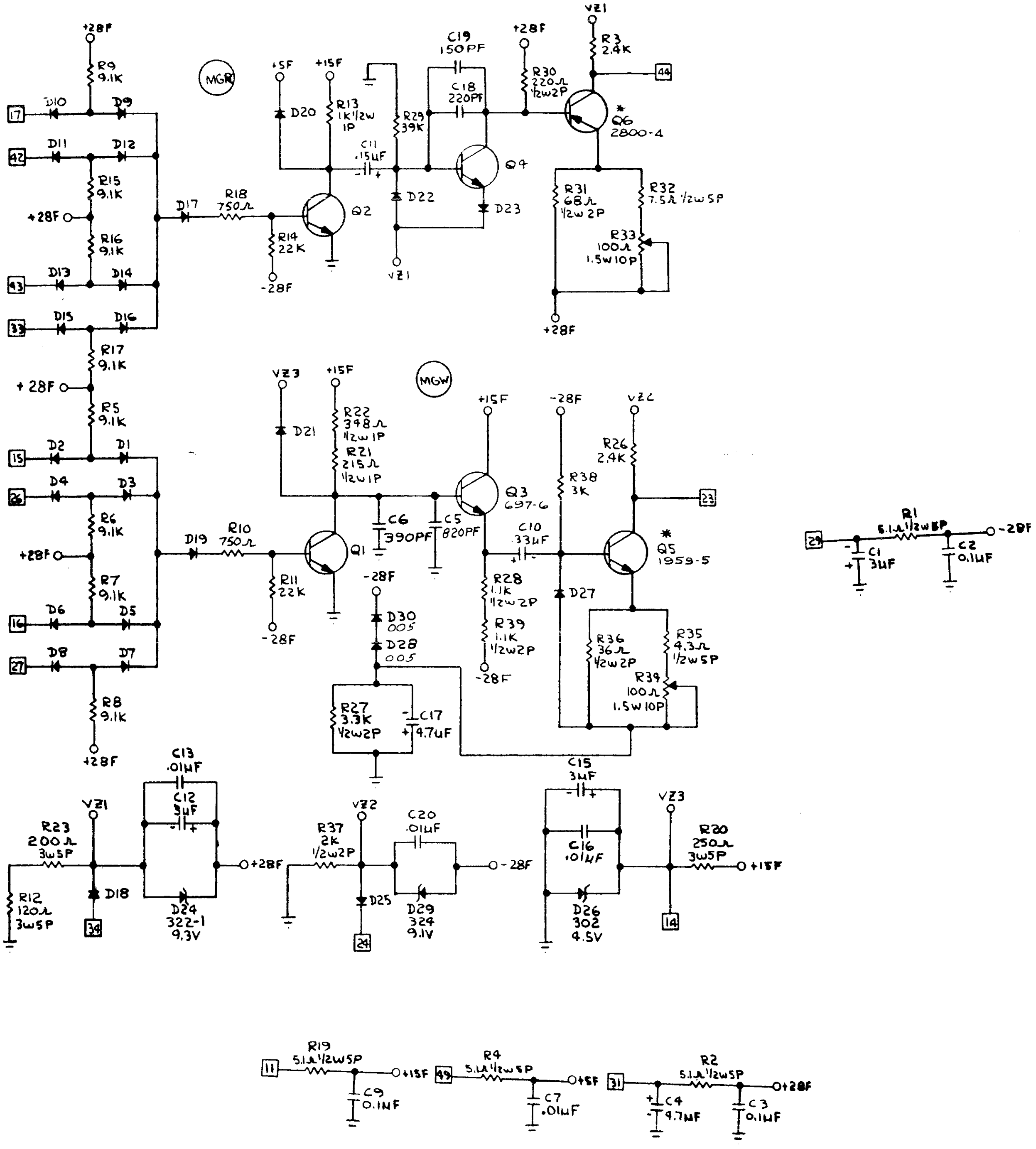
$$15 \text{ ma} - IR_{29} - \frac{IcQ4}{B\ Q4}$$

is available to charge up the Miller capacitors C18 and C19. This constant current produces a linear voltage sweep which continues until Q4 saturates. At this time C18 and C19 stop charging, and all of the current flows into the base of Q4. The voltage sweep at the collector of Q4 simultaneously appears at the emitter of the emitter follower output stage Q6. It produces a linearly rising current in Q6, since this voltage appears across a resistive emitter network. This current can be adjusted by varying R33. The rise time of the output is adjusted to 160 nsec by choice of a capacitor for installation in parallel with C18.

The Write Current Generator is similarly activated by a negative level "and" gate. The normally saturated Q1 is turned off when all of the input pins (15, 26, 16, 27) are simultaneously at ground potential. With Q1 "off", current through R22 and R21 is available to charge up capacitors C5 and C6 and also to supply base current to Q3. The current through C5 and C6 is changing due to the fact that as the capacitor charges, the collector voltage at Q1 is changing causing the current through R21 and R22 to change. The voltage at Q1 rises from ground potential to VZ3, at which time Q1 is shut off. The rate of this rise is determined by R22, R21 and capacitors C5 and C6. This voltage rise is coupled to the emitter of Q3, through the coupling capacitor C10 and then to the base and emitter of Q5. This sweep along with the emitter resistive network determines the current flow in the output. By varying R34, this output current magnitude can be varied. The current rise time down the selected line is adjusted to 170 nsec by choice of a capacitor for installation in parallel with C5.

In the event that temperature compensation of the current is required, the Zener reference voltages VZ_1 and VZ_3 are brought out to pins 34 and 14 respectively. These pins will be connected to a temperature compensation circuit which would effectively vary the voltage swing across the output stage emitter networks. This would cause the current to be changed with temperature. In this contingency, zener diodes D24 and D26 would be removed.

REVISION 0

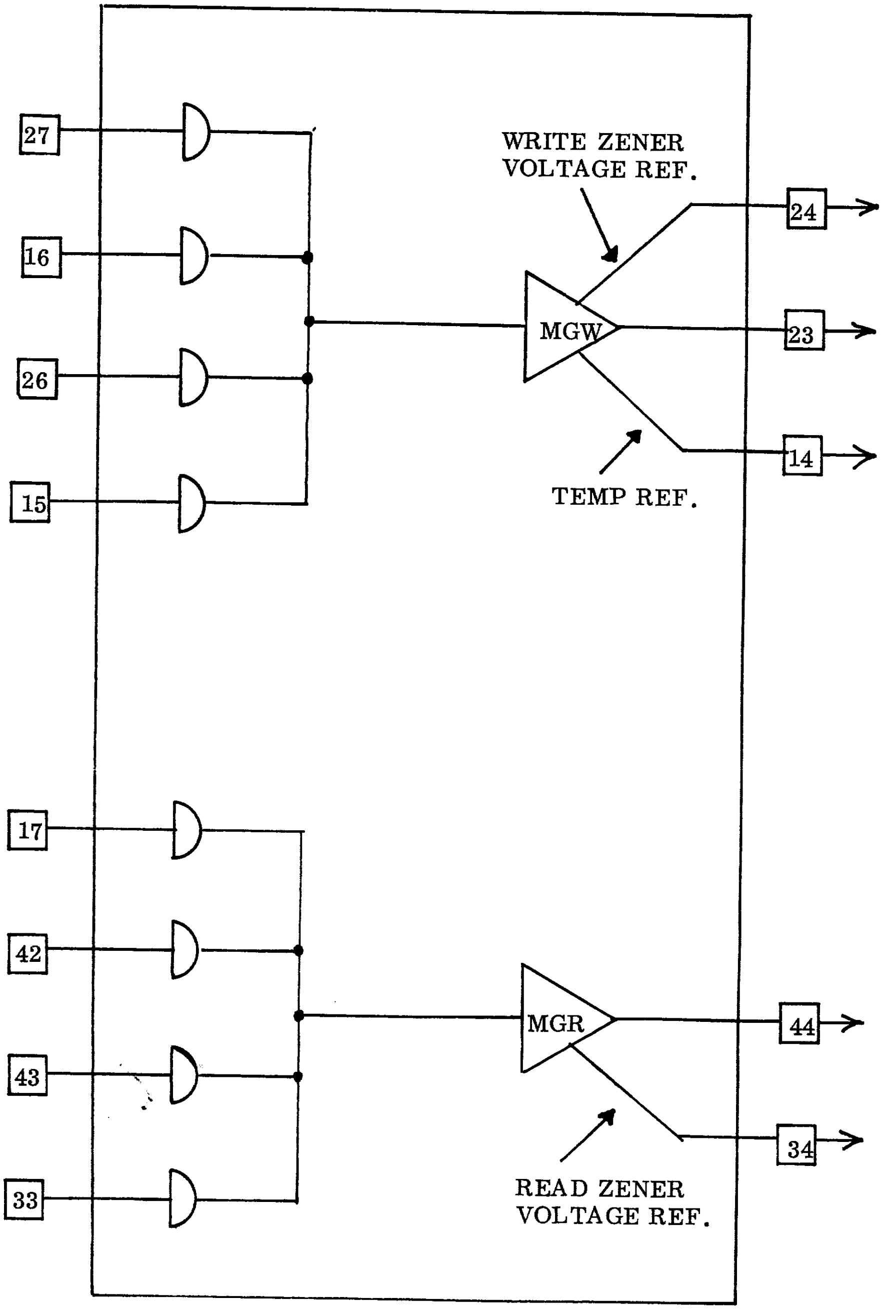


2MGB2

Fig. 5-13

MAIN MEMORY

R/W CURRENT GENERATOR PKG.



2MGB2
Fig. 5-14

MAIN MEMORY READ/WRITE DRIVER

2MDB4

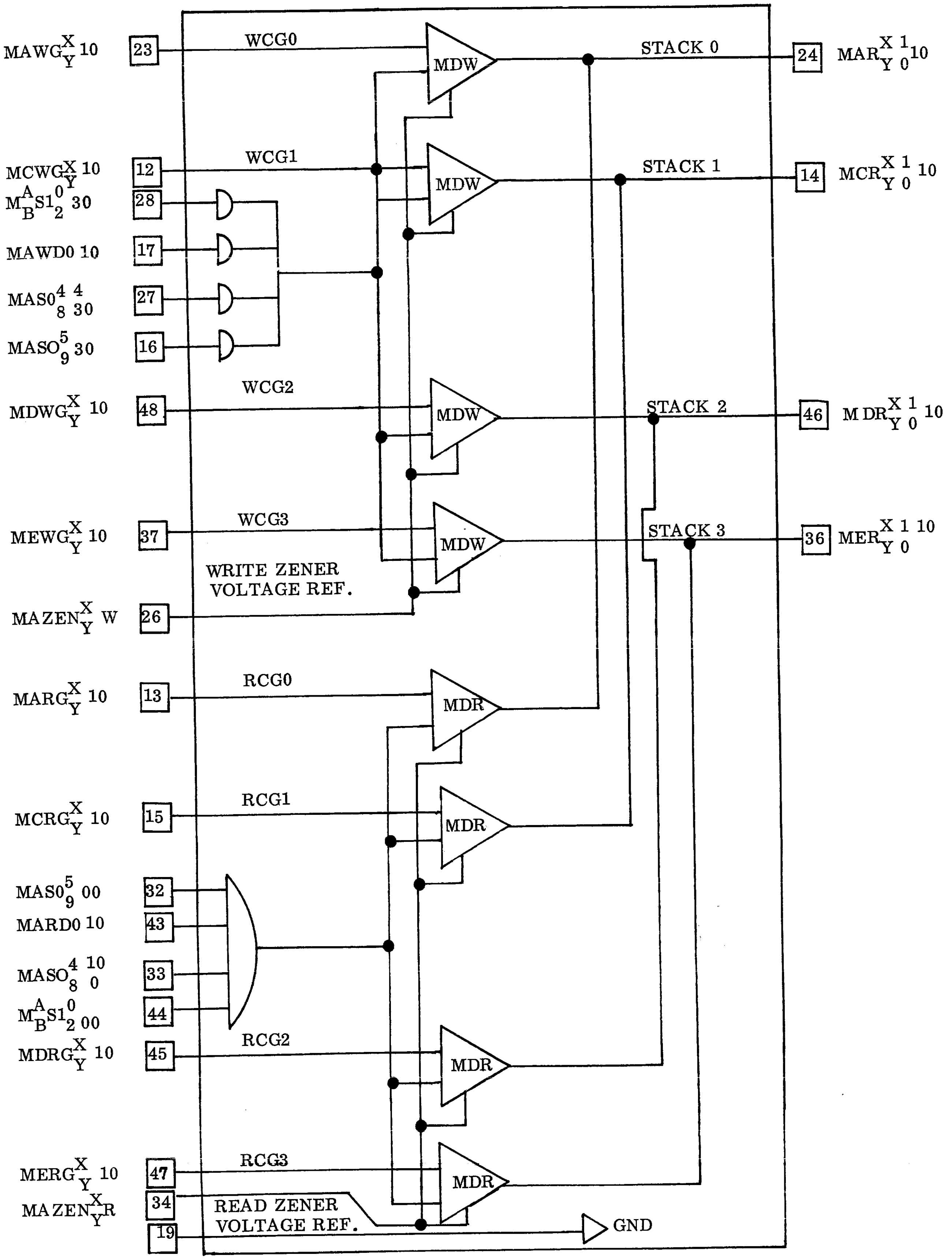


Fig. 5-15

THEORY OF OPERATION 2MDB4

The read and write driver circuits are 2-stage amplifiers driven from external timing pulses. The output stage contains four sets of two output transistors, each pair driving a given stack. At any one time, only one output transistor will be selected in accordance with which read/write generator is chosen. The output stage of the driver acts as a switch and effectively shunts the current produced by the current generators down the selected line with the aid of the selection switch.

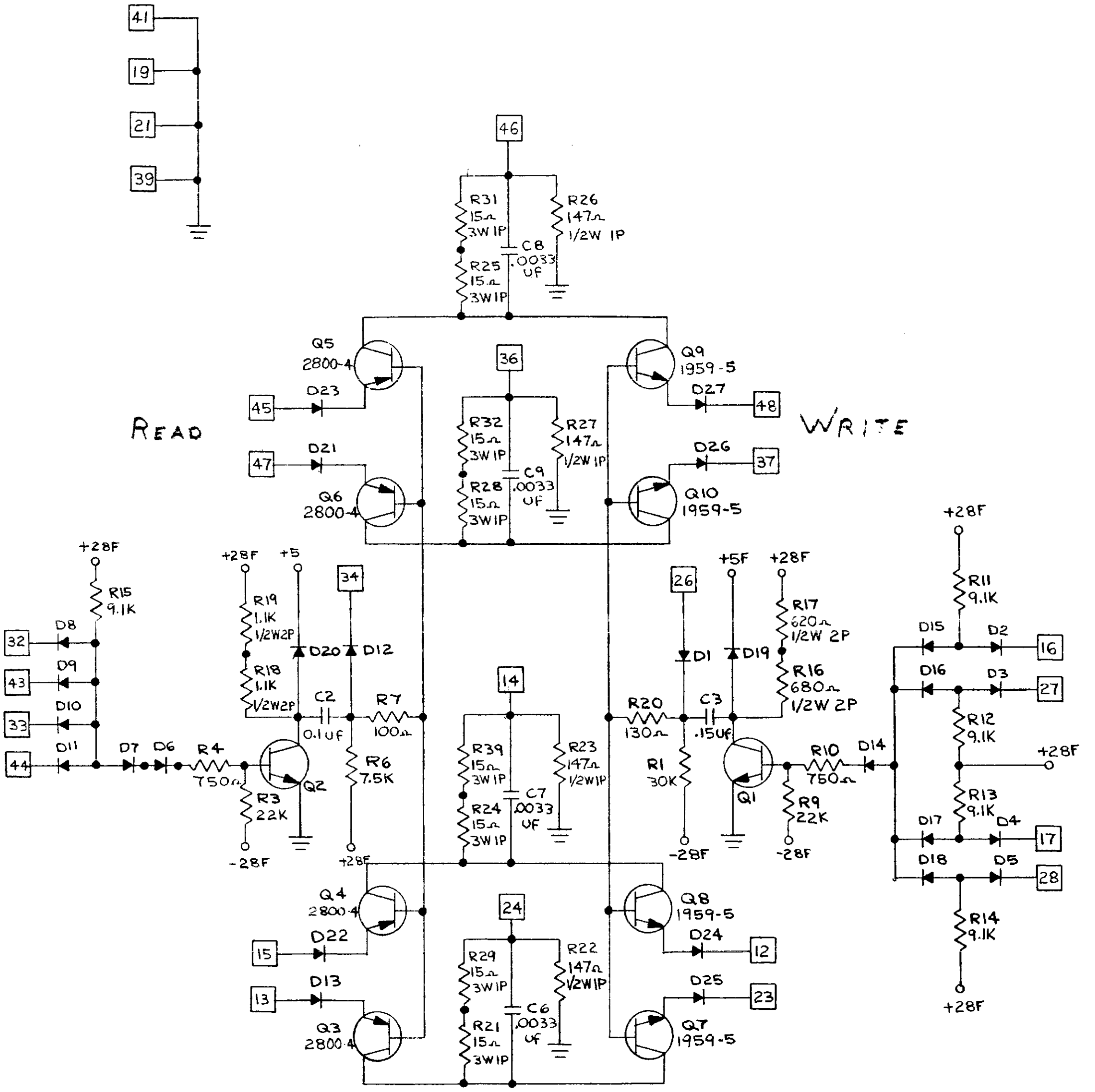
Read Circuit

In the "off" state, Q2 is held "off" with one or more of the four gate legs held at ground level. A 0 to +5 timing pulse is applied to pin 43. If the other 3 gate legs are at +5 volts, this causes Q2 to saturate producing a 5 volt swing on the collector (+5 to ground). This 5 volt swing is coupled through C2 to the bases of the output transistors Q3 through Q6. These bases were initially clamped to the zener supply which is located in the RCG. After the output transistors are on, the RCG corresponding to the selected stack is turned on and supplies 230 ma of current to the appropriate read driver output transistor, which in turn drives it down the selected line.

Write Circuit

In the "off" state, Q1 is in saturation and the parallel output transistors (Q7 through Q10) are held "off" by a diode clamped to a zener voltage which is derived in the write current generator circuit. Three of the four buffer diodes are held back biased by conducting gate diodes. A +5 to 0 timing pulse is applied to the fourth gate, which turns off" the normally "on" Q1, causing its collector to swing from ground to +5. This 5 volt swing is level shifted and coupled to the output stages by C3. The output transistors are turned "on" and one of the appropriate WCG circuits supplies 250 ma of collector current to one of the driver outputs and down the selected line.

REVISION 0



2MDB4

Fig. 5-16

THEORY OF OPERATION MSS4

There are two pairs of switches per Selection Switch card. A pair is defined as one READ switch and one WRITE switch. The inputs are arranged so that one READ timing input is required for both READ switches and one WRITE timing input is required for both WRITE switches. Besides the timing signal pins each of the four switches has three logic inputs. Thus each 2MSS4 package has twelve logic inputs and two timing inputs. Each switch has one output resulting in four outputs per package.

READ switch

This circuit is essentially a two stage current amplifier. For the purpose of this discussion we define +5 volts as a "1" and ground as a "0". When one or more of the inputs is in the "1" state, Q1 will be "ON" and in saturation. With Q1 saturated the base of Q2 is biased slightly positive (+0.3 volts), but not enough to turn it on. Thus pin 23 is at +15 volts through the resistor R44 (470). To cause a change in the output all inputs must be in the "0" state. Thus when the logic is such as to put pins 13, 24 and 14 at 0V, a negative going timing signal (+5 to 0V) will turn Q1 off. The base of Q2 will try to rise to a voltage determined by the voltage divider R23, R22, R21 and R39. This voltage is more positive than the voltage defined by Q2 being on and in saturation. Therefore, Q2 will turn on and saturate. The "ON" base current for Q2 is the current in resistors R21, R22 and R23 minus the current in R39. This base current is enough to ensure an I drive line of up to 300 ma in the worst case and at end of lift.

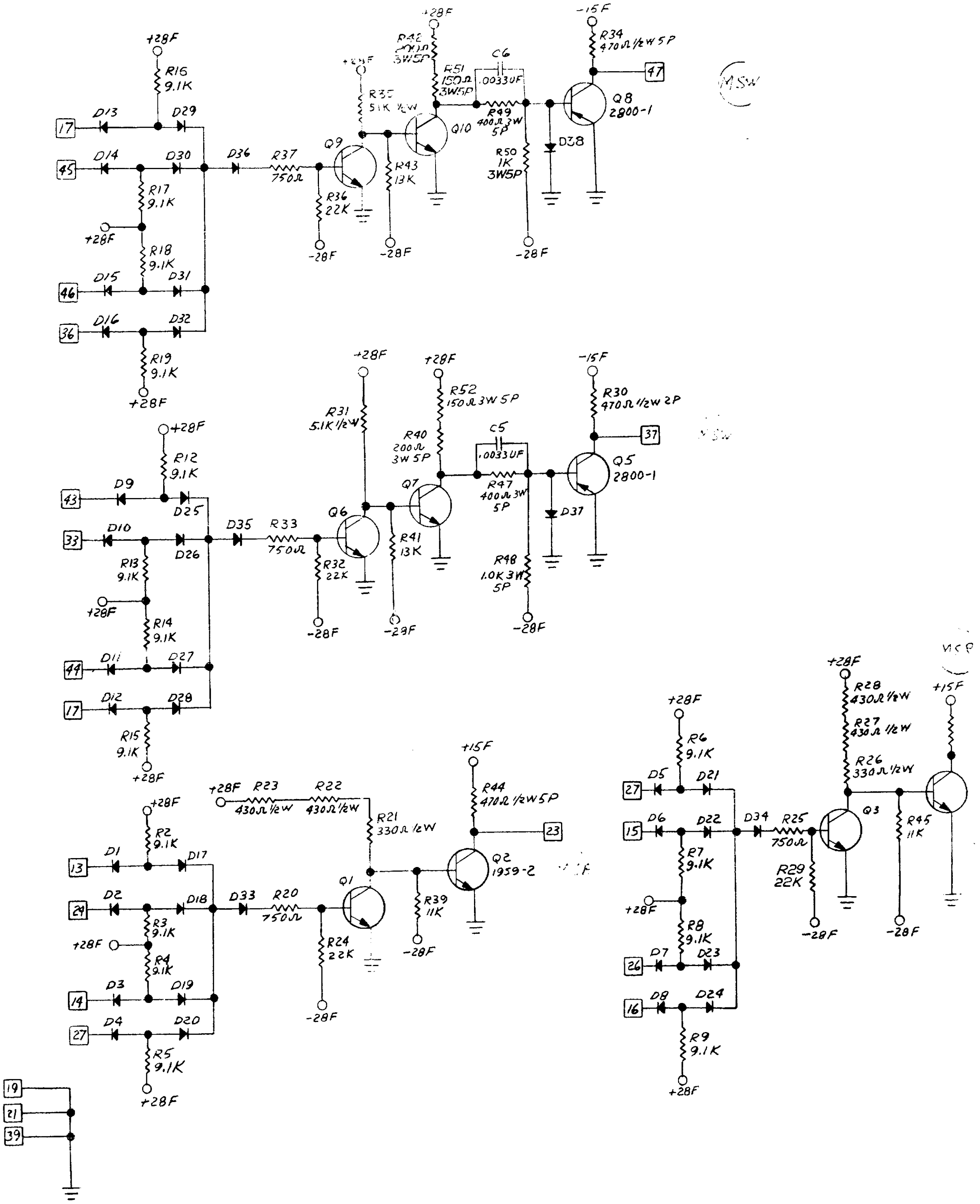
The capacitive load seen at pin 23 is quite large consisting of up to 48" of twisted wire and as many as 32 diodes (8 diodes per 4K of memory). When the voltage at pin 23 goes from +15 volts to ground it does so through the saturated transistor Q2. This results in a reasonably fast rise time. At the end of the READ timing pulse the voltage at pin 23 goes to +15 volts through R44 (470). This resistor and the load capacity control the fall time. Hence the large fall time. There is also storage time involved when Q2 is turned OFF.

WRITE switch

The conditions for Q6 being "ON" or "OFF" are the same as those for Q1 in the READ Switch. When Q6 is "ON" Q7 is held "OFF". The purpose of Q5 is to give the level shift needed to provide the proper polarity of output voltage swing. When Q7 is "OFF" the voltage divider of R52, R40, R47 and R48 hold the base of Q5 positive and therefore off. With the proper logic and the application of the WRITE timing pulse Q6 turns off turning Q7 on. This places the base of Q5 negative enough to ensure saturation. Enough base current is provided to Q5 to allow 300 ma of Collector Current (worst case end of life). Capacitor C5 is a "speed up" capacitor to turn on Q5 faster.

The capacitive loading at pin 37 is the same as at pin 23 and, incidentally, the same as at pins 12 and 47. This means that the voltage waveform at the WRITE outputs will have fast rise times (-15V to ground) and slow fall times (ground to -15V) as do the READ outputs.

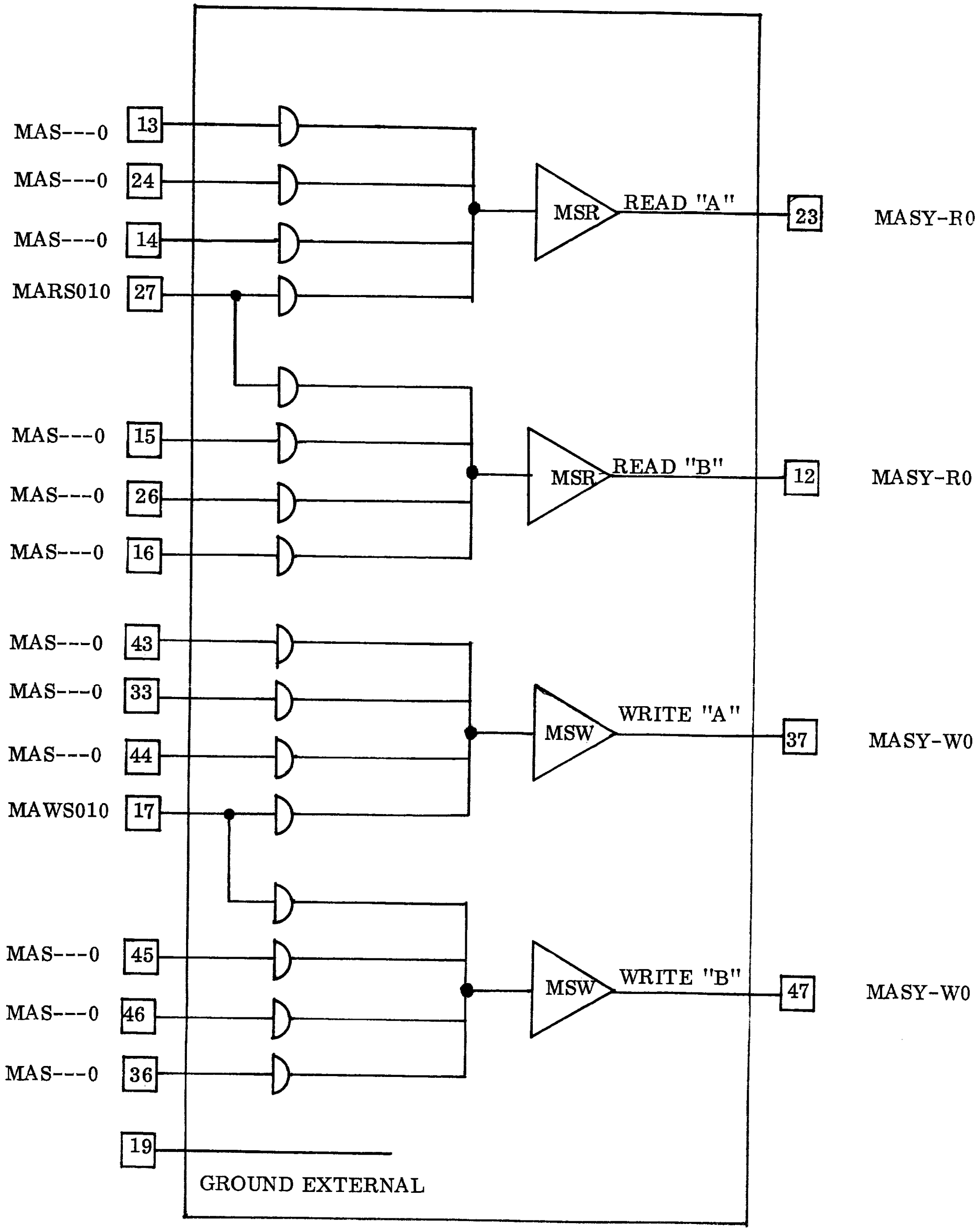
REVISION 0



2MSS4

Fig. 5-17

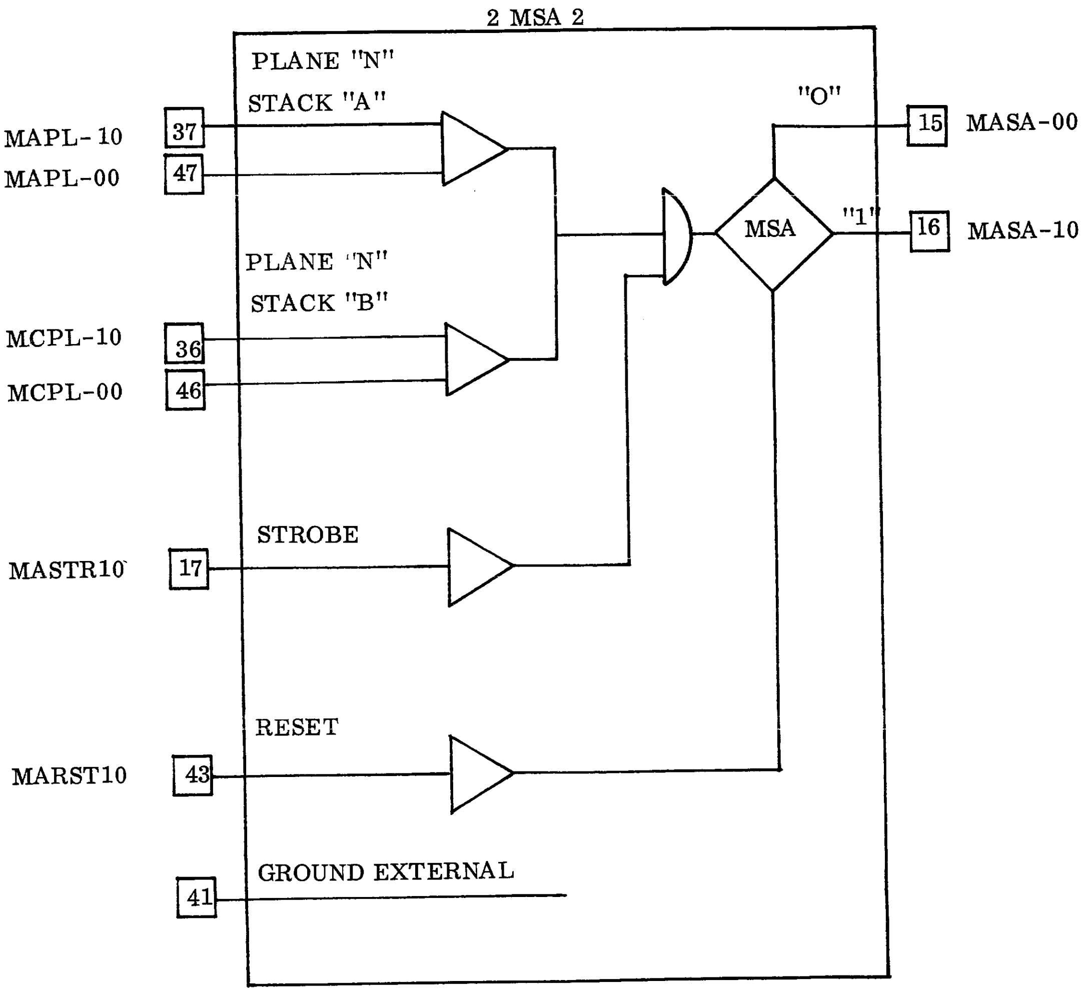
MAIN MEMORY SELECTION SWITCH



2MSS4

Fig. 5-18

MAIN MEMORY SENSE AMPLIFIER



2MSA2

Fig. 5-19

THEORY OF OPERATION 2MSA2

All and A12 are identical differential stages that feed into a common second stage amplifier which is also a differential stage. The "AND" circuit implies that the flop will "SET" only when the Strobe is concident with a "1" signal out of A_2. When the flop does "SET" the voltage on pin 16 goes to +5 volts and the voltage on pin 15 goes to ground. When the Reset signal occurs the flop again changes state; pin 16 goes to ground, and pin 15 goes to +5 volts.

The two amplifier stages A11 and A12 are identical except for the emitter impedance. A11 has an emitter impedance that includes a 100 ohm pot (R_{44}) which is used to balance the Sense Amplifier. These two amplifier stages share the same load resistors R24 and R26. Because of this arrangement,signals appearing on the inputs to A11 and A12 are amplified summed and then fed into A2 as a differential signal. However, the inputs to A11 and A12 are defined so that they come from two different stacks, only one of which is being read. Therefor e signals appear on only one input during a READ time. A2 is operated in the differential mode only to reduce the total dc voltage variation to the threshold detector level.

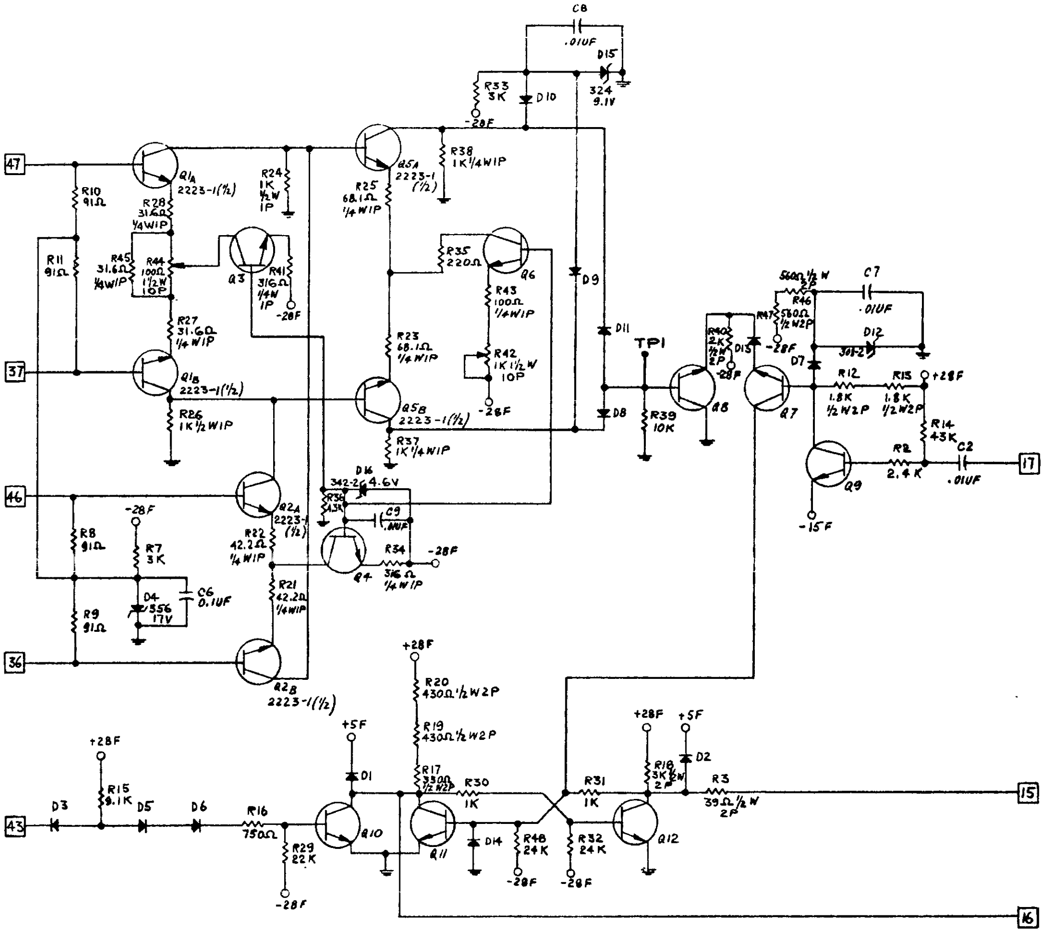
All three amplifier stages have constant current sources that are referenced to the same zener diode (D16). A 1K pot (R42) in the current source of A2 determines the quiescent voltage on the collectors of both its transistors (Q5A and Q5B). This voltage setting determines the signal threshold. Diodes D8 and D11 rectify the signals so that the waveform on the Test Point will always be negative going signals riding on a negative dc level.

Transistors Q7 and Q8 comprise a current mode switch used as a detector. One of the transistors will be "ON" when the other is "OFF". The one with the more negative base will be "OFF". When the strobe is not present Q9 will be on and saturated placing the base of Q7 at -15 volts. The threshold is set so that the base of Q8 is about -5 volts. When a "1" signal is read out of memory the voltage on Q8 will go as low as -9 volts. This is still not enough to switch the detector. It is possible that a noise signal could occur to lower the voltage on Q8 below -15 volts causing erroneous detection. This is avoided by placing a limit on the negative going signal allowed at the test point. Zener diode D15 and diodes D9 and D10 set this voltage at -9.1 volts.

When strobe occurs Q9 is turned off, and the base of Q7 rises towards +28 volts through 3.6 K (R12 and R13). Zener diode D12 along with D7 clamps the base at -5.8 volts. At this time a signal that lowers the test point voltage to -6.5 volts will switch the detector. The difference between -5.8 volts and -6.5 volts is due to diode D7.

When Q7 is turned on it draws a collector current on the order of 10 ma. This current causes the base of Q11 to go slightly negative. Q11 turning off causes Q12 to turn on. Thus there is a 5 volt excursion on pins 15 and 16. The flop will remain "SET" until the beginning of the next cycle when a reset pulse occurs on pin 43. This pulse turns Q10 on causing the flop to change back to its original state.

REVISION 0



2MSA2

Fig. 5-20

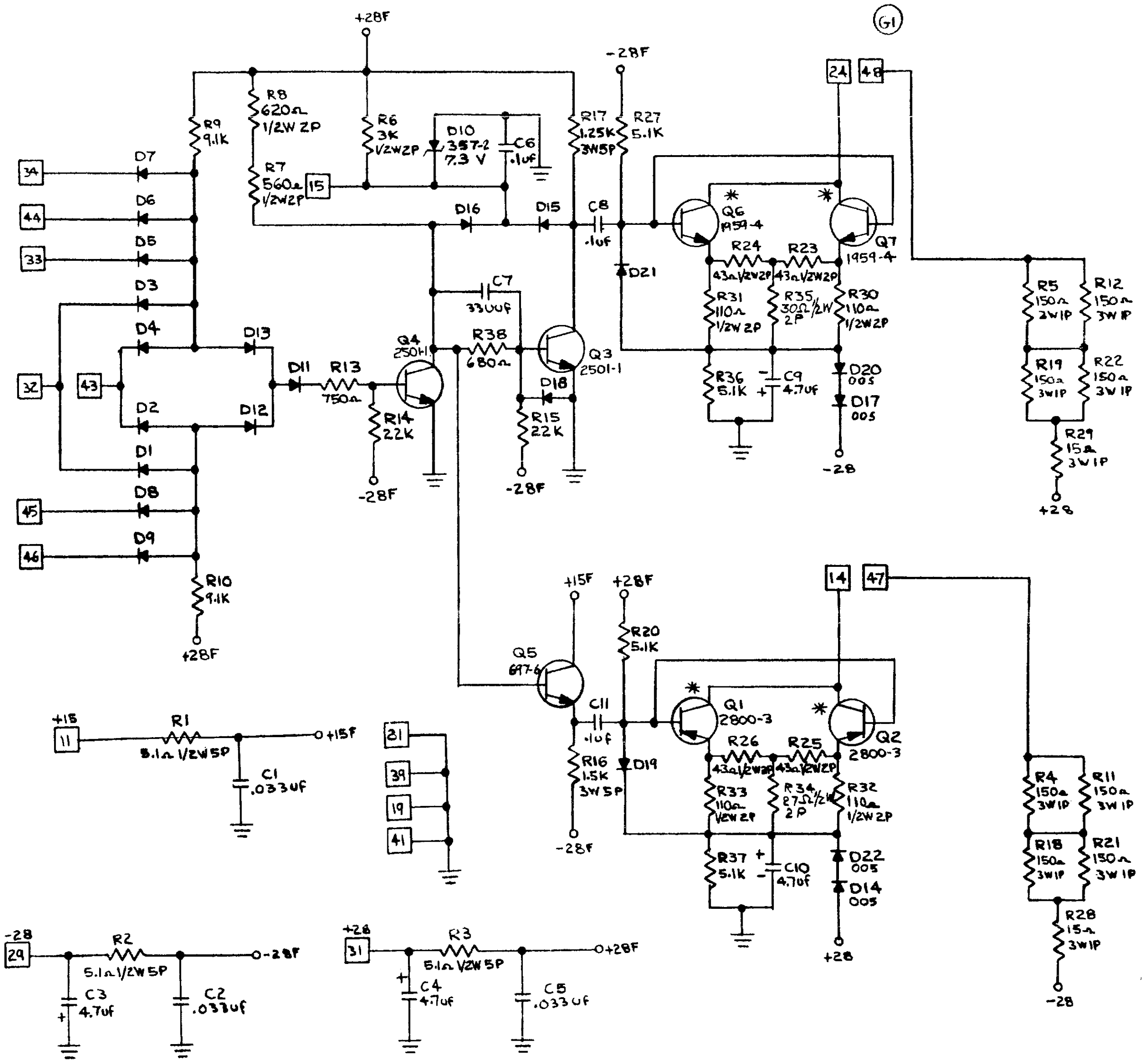
THEORY OF OPERATION 2MDN2

The 200 Memory Inhibit Driver is operated in conjunction with a place having a split inhibit line. Half of the inhibit line is pulsed positively and the other half negatively by the positive and negative outputs of the driver. The reason for doing this is to reduce the noise coupled into the sense line and current coupled into the Read/Write lines. The circuit is basically two current amplifiers (one positive and one negative) tied to a common input in order to insure that they are both pulsed at the same time. The rise and fall times and the delays of the outputs are about the same so that the output pulses will start and end at the same time. Capacitor C_7 is used as a speed-up capacitor, and its purpose to speed up the negative output, compensating for the difference in the delay between Q3 and Q5.

The input is a gate-buffer structure. There are two sources of information which tell the inhibit driver to fire. One source is the sense-amplifier which is sampled and tells the inhibit driver to write into memory what has been there before. The other source is the MLR which tells the inhibit driver to write in new information. Therefore, by appropriate input signals, the information written into memory at write time is determined by one sourse or the other. In its normal state, Q4 is "off", Q5 and Q3 are "on", and the output transistors Q2, Q1, Q6 and Q7 are "off". An input timing pulse of 0 to +5 volts is applied which saturates Q4 forcing its collector to swing from +8 volts to 0 volts. This swing is coupled to Q3 by the speed-up capacitor C_7 turning it off, causing the collector voltage to swing from ground to +8 volts. At this same time, the voltage swing on the collector of Q4 is coupled to the emitter follower stage Q5. The voltage pulses (0 to +8 volts and +8 volts to 0 volts) are coupled by the voltage level shifting capacitors C8 and C11 respectively, to the output transistor pairs. The output stages have parallel transistors to eliminate over-dissipation in the output transistors.

The voltage swing at the bases of the output transistors appears across the identical emitter networks of the output emitter follower stages. This allows for identical currents to flow in each half of the split inhibit line, with the current in each half being in opposite directions.

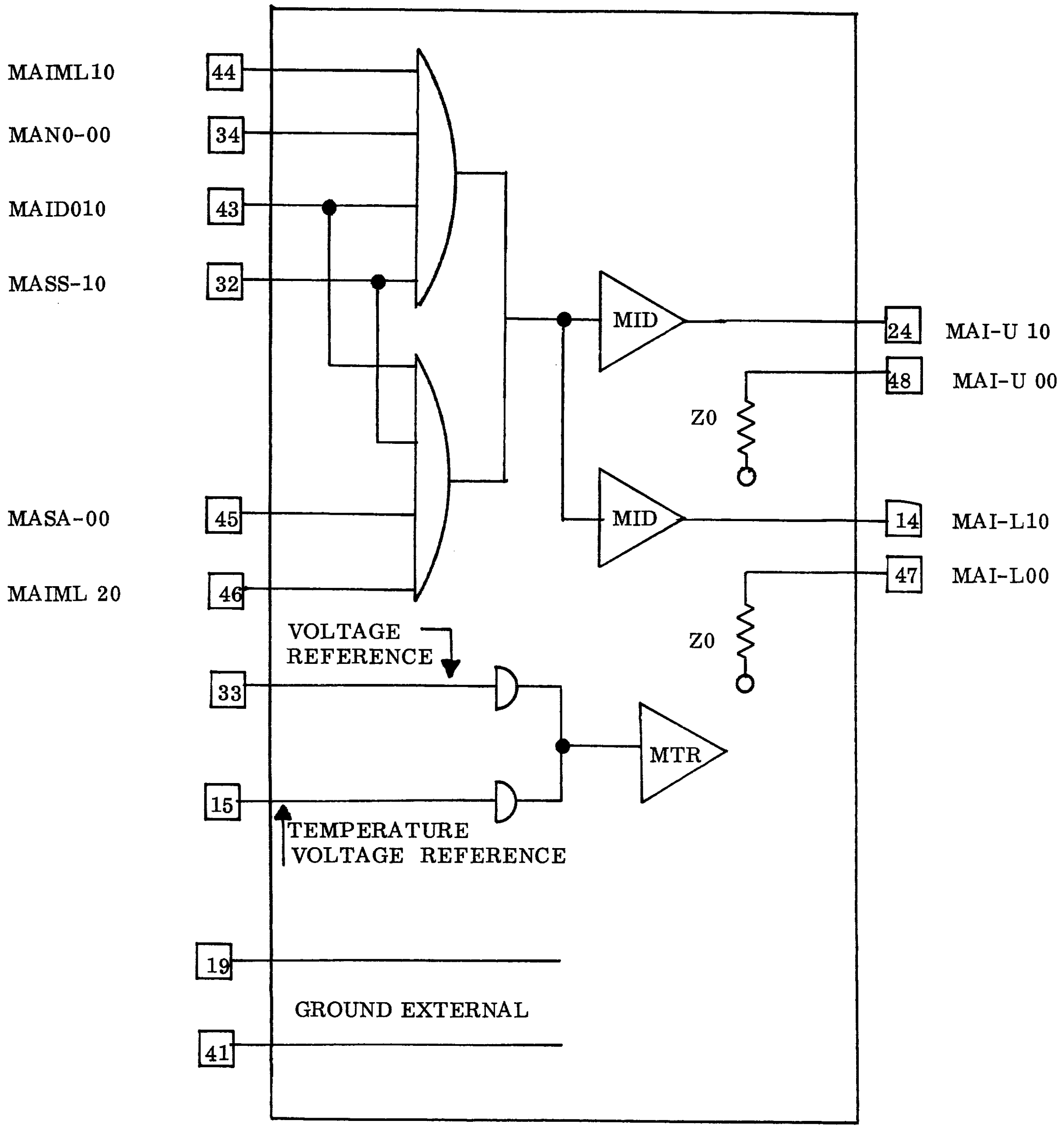
REVISION O



2MDN2

Fig. 5-21

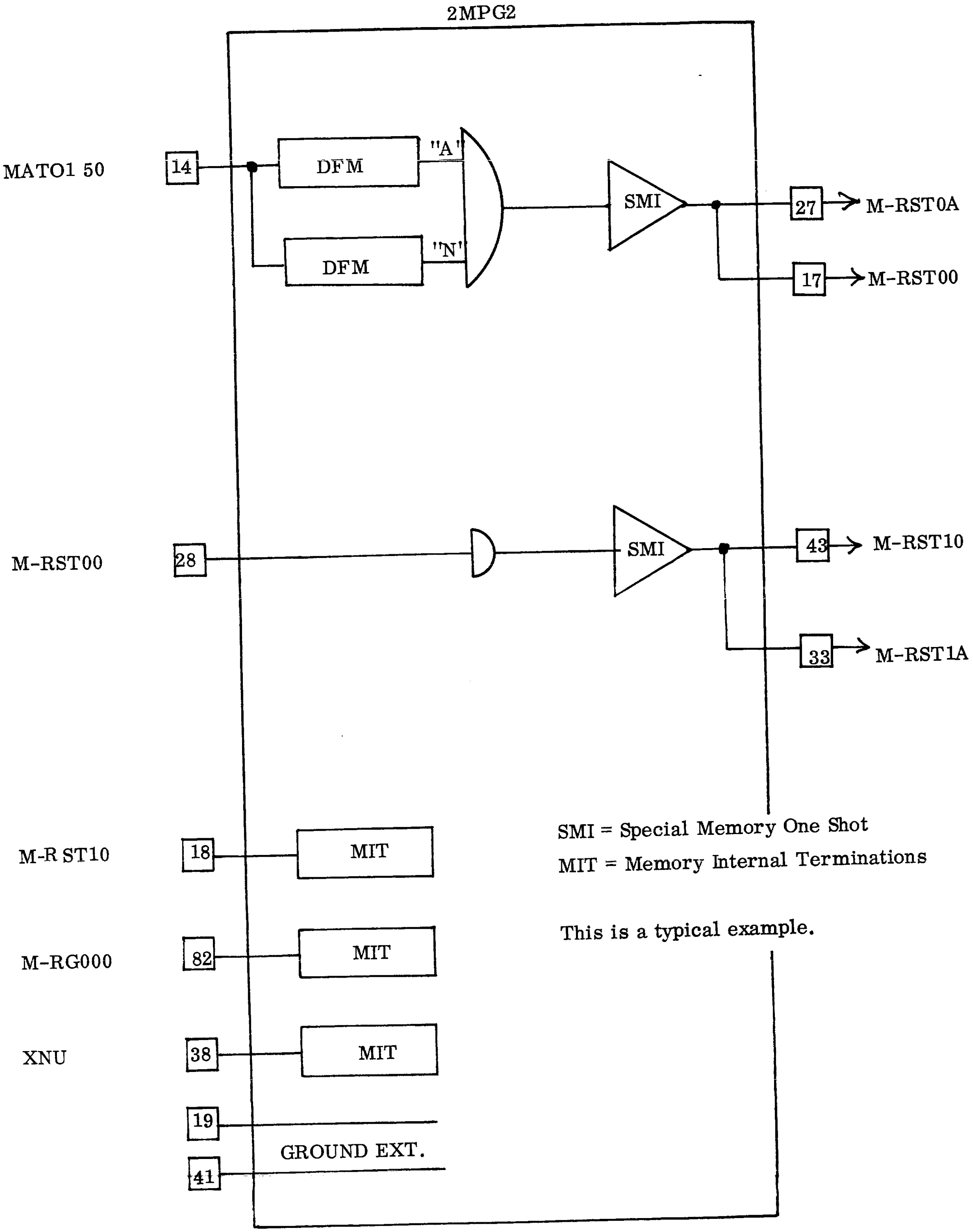
MAIN MEMORY INHIBIT DRIVER



2MDN2

Fig. 5-22

MEMORY PULSE GENERATOR



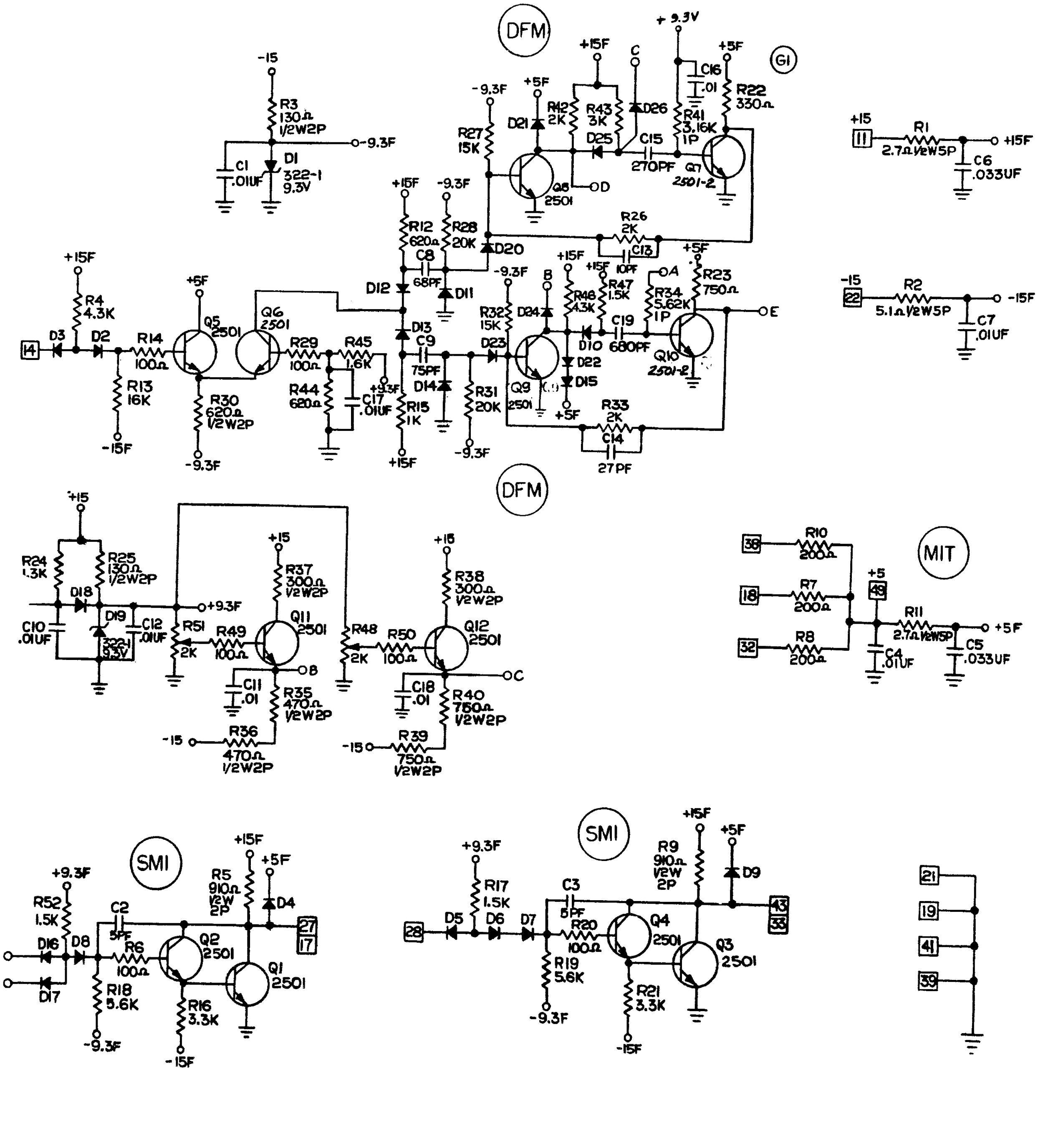
SMI = Special Memory One Shot

MIT = Memory Internal Terminations

This is a typical example.

Fig. 5-23

REVISION 0



2MPG2

Fig. 5-24

SECTION 6
EXTRACTION CYCLES

INTRODUCTION

There are two basic types of cycles used during the execution of a program, extraction cycles and execution cycles. Extraction cycles, referred to as fetching cycles in the LBD's, retrieve a complete instruction from main memory, modify the addresses, if indicated, and store each portion of the instruction in a designated storage area. The execution cycles perform the necessary steps to carry out the instruction. The extraction cycles will be discussed in detail in this chapter.

Instruction Format

Machine instructions are stored in main memory in a sequence determined by the programmer. A single instruction may be contained in one location, if op-code only, or in several locations. Word marks are set in the locations containing op-codes, thus giving the central processor a method of determining the instructions length. Figure 6-1 shows the six basic formats in which machine-language instructions may appear.

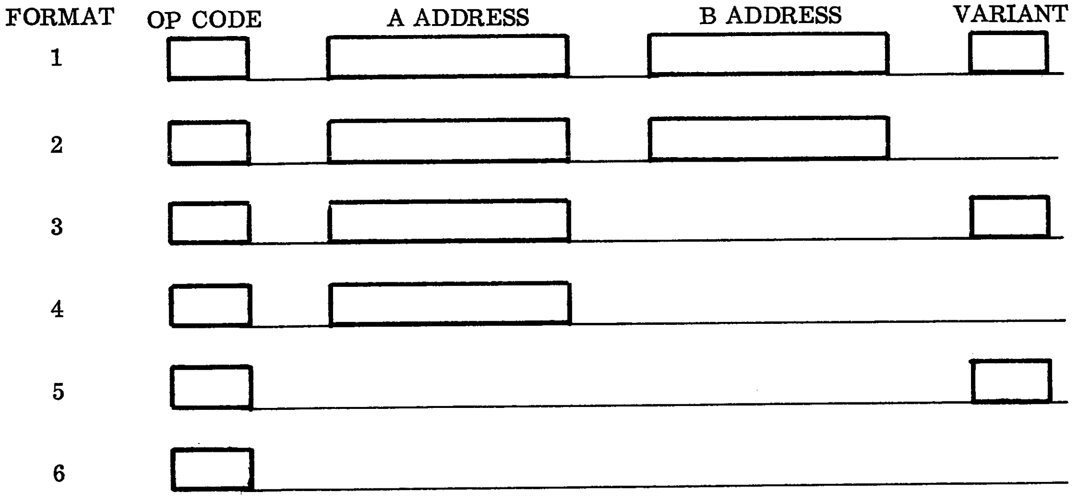


Figure 6-1

It should be noted that this figure represents only the basic instruction formats and that several variations are possible. Also that the op-code block is, in reality, six binary bits; the A and B address blocks are either 12 or 13 binary bits depending upon the character mode. The variant block is equal to six binary bits.

ADDRESS MODIFICATION

There are two types of address modification possible during the extraction cycles if the machine is in three character mode operation. The high order three bits of the operand address will determine what type of modification is to be done. If the high order three bits form an octal configuration of one through six the address will be augmented by the contents of the index register specified. If the high order three bits form an octal seven the address will be used to specify the location in main memory where the true operand address is stored. These two types of addressing are referred to as Indexing and Indirect. It should be noted that one indirect address can refer to another indirect address and so on through any number of indirect addresses. It is also possible to go from an indirect address to a indexed address, but not from an indexed to an indirect address ,or another indexed address.

Index registers are contained in the low order 24 locations of main memory. If index register #1 is specified as the address augmentor, by taking the binary configuration (001) and shifting it left two binary places (00100) the address of the low order character of index register #1 is obtained. The two locations preceeding this base location contain the middle and high order character. See Figure 6-2.

Index Register	Binary Indicator	Storage Field (in octal)	Octal Address
1	001	2 - 4	4
2	010	6 - 10	10
3	011	12 - 14	14
4	100	16 - 20	20
5	101	22 - 24	24
6	110	26 - 30	30

Figure 6-2

Machine Cycles

The extraction cycles are performed by three major machine cycles. "Derived Variant Cycle" (DVC), will be used to extract the op code and any variants that may be in the instruction." Extract A Cycle " (EAC) will extract the A address, and "Extract B Cycle" (EBC) will extract the B address.

The three minor cycles E1C, E2C,and E3C will be used to carry out a certain portion of the overall job performed by the major cycles. For example, during the EAC cycle E1C extracts the high order portion of the A address, E2C the middle,and E3C the low order.

There are also two early cycles that may come true during the extraction cycles. Existant Indexing Cycle (EXC), will be high during the augmenting process if indexing is indicated in the address field, and "Existant Indirect Cycle" (EDC) will be high during the extraction of the indirect address if indirect addressing is specified in the address field.

A mnemonic code is used to designate each memory cycle. This code is derived from the machine cycles that are high for that particular memory cycle. During the first cycle of the instruction extraction, the major cycle DVC is high and the minor cycle E3C is high. If the middle terms of these two cycles are combined the memory cycle V3 is formed. All memory cycles, both extraction and execution, are designated in this manner. The flow chart book should be used along with the following pages.

V3 Cycle

The sequence counter (location 17_8) is addressed during the first control memory cycle of V3. The sub-commands ICL and ICM transfer the sequence counter contents to the S register at time one.

$$\text{ICL (I)} = \text{EEC}\bullet\text{CP1}\bullet\text{MAT}+\overline{\text{IIN}}\bullet\text{CP1} + \text{BMF}$$

$$\text{ICM (I)} = \text{EXC}\bullet\text{CP1} + \text{BMF}$$

$$\text{BMF} = \text{HSR}+\overline{\text{ARS}}+\text{BMF}\bullet\text{T07}$$

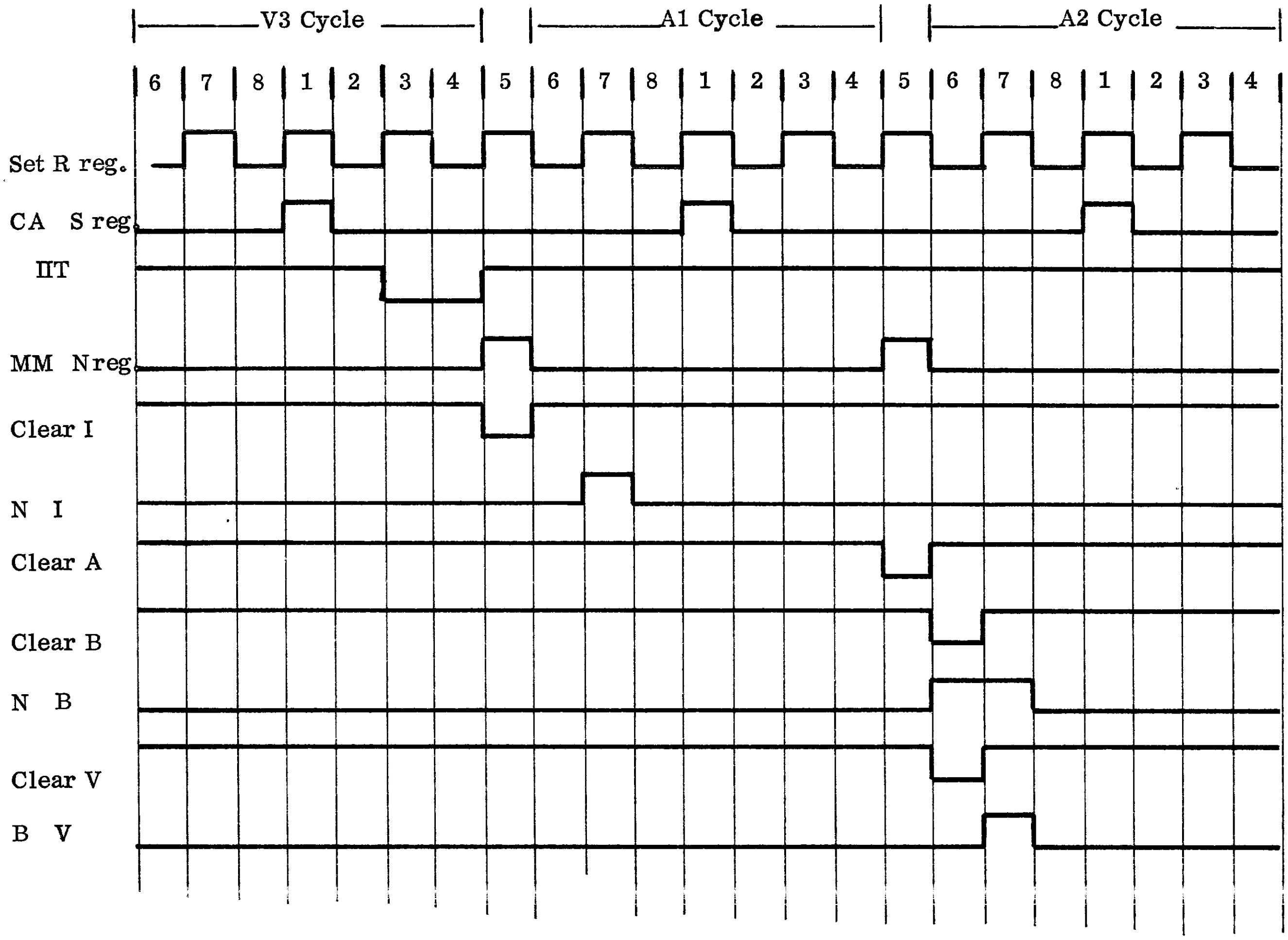


Figure 6-3

ICL and ICM, both being inverters, will be high only if all input gates are disabled. Examine the first gate, EEC•CP1•MAT; the only cycles high at this time are DVC and E3C, therefore, the functions EEC and CP1 will both be low disabling this gate. The second gate, $\overline{\text{IIN}}$•CP1, also will be low because of CP1. The third gate, BMF, is normally low and comes high only when the contents of the S register are to be recirculated for more than one cycle.

The main memory location specified by the sequence counter is read out, and its contents stored in the sense amps; at time five with the aid of the subcommand M2N this information is transferred to the N register. At time five the I register is cleared, CLI (I) = PVC•P3C•T05, and during time six the extracted information, which is the instruction op-code, is transferred from the N register to the I register, where it will be recirculated until the execution of the instruction is completed.

$$\text{N2I} = \text{N4I}\bullet\text{T07}\bullet\overline{\text{IMT}}$$

It should be noted at this time that information extracted from main memory during any cycle is not ~~transferred to a storage register until the following cycle. Therefore, the N2I transfer at time~~ seven will, in reality, occur during the next cycle, which will be an A1 cycle. See figure 6-3.

The second control memory cycle addresses the A address counter (14), and stores its contents in the control memory sense amps.

The third control memory cycle addresses working location two (07); at this time IIT will be low inhibiting sense amp reset and strobe .Therefore,at write time the contents of the A address counter will be stored in WL2. This is necessary to accommodate the SCR instruction which states that if the A or B address is specified by the variant character, the previous contents of that register are stored in the location specified by the A address of the instruction.This internal transfer saves the previous A address.

The sequence counter is addressed during CM cycle four,and at times five and six an incremented result (SC + 1) is delivered from the S register to the Z register.

ISR = DFC•$\overline{\text{EXC}}$•CT8•RUN

At CM write time of this cycle the WZ functions will be activated,and the incremented sequence counter is returned to CM.

WZU = IWZ•IYZ•T06

WZM = IWZ•IYZ•T06

WZL = IWZ•IY2•T06

Examination of the input gates for IWZ and IYZ for a V3 cycle will show no input gates satisfied, therefore, since both functions are GBI's, IWZ•IYZ will be high,allowing the WZ functions to come high at time six.

The next state condition for a V3 cycle, provided an interrupt signal has not been sensed, is an A1 cycle.

A1 Cycle

The sequence counter is addressed, and its contents sent to the S register, during the first control memory cycle. Locations 14 and 15 are addressed during CM cycles two and three. However, there is no information transfer,and the contents of both locations will remain unchanged.

Since the instruction being extracted contains a complete A and B address no word mark will be sensed in this cycle, therefore, $\overline{\text{N07}}$ will be high.

$\overline{\text{N07}}$ being high allows:

WZU = $\overline{\text{N07}}$•IYZ•T06

WZM = $\overline{\text{N07}}$•IYZ•T06

WZL = $\overline{\text{N07}}$•IYZ•T06

CLB = CLE•$\overline{\text{N07}}$•CT6 (CLE = IIN•H1C)

N2B = IIN•$\overline{\text{N07}}$•CT6

HVR (I) = HFC•$\overline{\text{PXC}}$•H1C•$\overline{\text{N07}}$•T06 (Clear V Reg)

B2V = HFC•$\overline{\text{PXC}}$•H1C•$\overline{\text{N07}}$•T07

CLA (I) = CAF•CT5 CAF = HFC•$\overline{\text{EDE}}$

The last function, CLA, although mentioned above is not dependent on $\overline{\text{N07}}$. It will, however, go low at time five of every HFC cycle and reset the A register. The only path from main memory to control memory is through the accumulator, which is continually adding the contents of the A and B registers, therefore with the A register set to zero the accumulator will be equal to the contents of the B register.

At the completion of the A1 cycle the information extracted from main memory will be contained in the B register, accumulator,and the V register. The sequence counter will contain its original contents plus two.

The next state condition for the A1 cycles, with $\overline{N07}$ and $\overline{BAB}$ is an A2 cycle.

A2 Cycle

The sequence counter is addressed during the first CM cycle,and its contents sent to the S register. At time two the accumulator is delivered to the Z register,and during the write cycle of CM cycle two the high order three bits of the A address will be written into the A address counter (14).

$WZU = BWA \bullet T02$

$BWA = DAB \bullet \overline{EXT}$

$DAB = E2C \bullet EAC$

At time four the B register is cleared,

$CLB\ (I) = CLC \bullet T04$

$CLC = HFC \bullet \overline{DVC} \bullet \overline{H1C}$

and at time five the A register is cleared.

$CLA\ (I) = CAF \bullet T05$

$CAF\ (I) = \overline{HFC} \bullet \overline{DE3} + EDE \qquad (CAF = HFC \bullet \overline{EDE} + DE3 \bullet \overline{EDE})$

The subcommand N2B will be high for times six and seven and will transfer the middle portion of the A address to the B register.

At the completion of the A2 cycle, the sequence counter will contain its original contents plus three, and the accumulator and B register will contain the middle portion of the A address.

The next state condition for the A2 cycle is the A3 cycle.

A3 Cycle

The sequence counter is addressed for the first CM cycle and at time one its contents delivered to the S register. At time two the middle portion of the A address is delivered to the Z register via the accumulator. The second CM cycle will address the A address counter: the function WZM will come high, and the middle six bits of the A address will be written into CM.

$WZM = BWC \bullet T02$

$BWC = DFC \bullet \overline{DVC} \bullet E3C$

The B register will be cleared at time four the the A register will be cleared at time five. At times six and seven the N register will be transferred to the B register.

At the completion of the A3 cycle the sequence counter will contain its original address plus four, and the accumulator and B register will contain the low order six bits of the A address. The V register still contains the high order six bits of the A address. If indirect addressing is specified (the high order three bits of the V register form an octal seven), the next state condition for an A3 cycle will be an AlD cycle. However, the normal next state is Bl.

AlD Cycle

At time seven the subcommand B2V will transfer the lower six bits of the A address to the V register.

$HVR\ (I) = EDC \bullet \overline{PAC} \bullet CT6$

$B2V = EDC \bullet \overline{PAC} \bullet T07$

The first CM cycle will address the A address counter and at time one the subcommand ICM will gate the high order nine bits of the A address to the S register. ICL will be low:

$ICL\ (I) = \overline{IIN} \bullet CP1$

$\overline{IIN} = EDC + HDX$

$HDX\ (I) = SV3+CT3 \qquad (HDX = \overline{SV3} \bullet \overline{CT3})$

The subcommand V2S will be high for the complete AID cycle and at time one will gate the V register to the low order six positions of the S register, thus completing the A address in the S register.

$V2S = EDC \bullet E1C$

The second CM cycle addresses the A address counter,and at time two the low order six bits of the A address are delivered to the Z register. The write Z functions,however, will all be disabled,and at the completion of the second CM cycle the A address counter will still contain only the high order nine bits of the A address.

The B register is cleared at time four and the A register is cleared at time five. The subcommand N2B will be high for times six and seven and gate the high order six bits of the indirect address to the B register.

Since it is specified that indirect addresses may be cascaded and also that it is possible to index an indirect address,it will be necessary to transfer the high order six bits of this indirect address to the V register to determine if further address modification has been specified. Therefore, B2V will again come high and transfer this information to the V register.

$B2V = EDC \bullet \overline{P2C} \bullet T07$

The fourth control memory cycle addresses WL #1 (16) and delivers the original A address plus one to control memory for storage.

At the completion of the A1D cycle the high order six bits of the indirect address are located in the accumulator, B register and V register. The original A address plus one is stored in WL #1 and will be called upon to extract the remaining portion of the indirect address.

The next state condition for the A1D cycle will be the A2D cycle.

A2D Cycle

Working location one is addressed for the first CM cycle,and its contents, at time one, are delivered to the S register. The accumulator is delivered to the Z register at time two. The A address counter is addressed during the second CM cycle, WZU will be high, and the high order three bits of the indirect address will be delivered to CM.

$WZU = BWA \bullet CT2$

$BWA = DAB \bullet \overline{EXC}$

$DAB = EAC \bullet E2C$

The B register is cleared at time four and the A register is cleared at time five. The subcommand N2B is high for times six and seven and will transfer the middle portion of the indirect address to the B register.

Upon completion of the A2D cycle the middle six bits of the indirect address will be contained in the accumulator and the B register. WL #1 will contain the original A address plus two, and the A address counter, location 14, will contain the high order three bits of the indirect address.

The next state condition for the A2D cycle is the A3D cycle.

$\mathrm{EDC} = \mathrm{CT4} \bullet \mathrm{PDC} \bullet \overline{\mathrm{P3C}}$

$\mathrm{EAC} = \mathrm{PAC} \bullet \mathrm{EDC} \bullet \mathrm{CT6}$

A3D Cycle

Working location one is addressed during the first CM cycle, and at time one its contents sent to the S register. At time two the accumulator is transferred to the Z register. The second CM cycle addresses the A address counter, WZM is activated, and the middle six bits of the indirect address will be delivered to CM.

$\mathrm{WZM} = \mathrm{BWC} \bullet \mathrm{CT2}$

$\mathrm{BWC} = \mathrm{DFC} \bullet \overline{\mathrm{DVC}} \bullet \mathrm{E3C}$

The B register is cleared at time four. The A register was cleared in the previous cycle and is already set to zero. The subcommand N2B will be true at times six and seven transferring the low order portion of the indirect address to the B register.

At the completion of the A1D cycle the low order six bits of the indirect address are contained in the B register and the accumulator. The variant register still holds the high order six bits of the indirect address.

Assume now that an octal two is specified in the high order three bits of the V register: i.e. VO4 = 0, VO5 = 1, and VO6 = 0. The next state condition for this cycle will be an A1X cycle.

$\mathrm{EAC} = \mathrm{PAC} \bullet \overline{\mathrm{IIN}} \bullet \mathrm{CT6}$

$\mathrm{EXC} = \mathrm{CT4} \bullet \mathrm{INX} \bullet \mathrm{SIT}$

$\mathrm{INX} = \overline{\mathrm{VEO}} \bullet \overline{\mathrm{VE1}} \bullet \overline{\mathrm{CM2}}$

$\mathrm{SIT} = \mathrm{PFC} \bullet \mathrm{PXC} \bullet \overline{\mathrm{DVC}} \bullet \overline{\mathrm{CM2}} \bullet \mathrm{E3C}$

A1X Cycle

Working location one is addressed during the first CM cycle. The subcommands ICL and ICM will be low inhibiting the normal sense amp to S register transfer.

$\mathrm{ICL}\ (\mathrm{I}) = \overline{\mathrm{IIN}} \bullet \mathrm{E1C}$

$\mathrm{ICM}\ (\mathrm{I}) = \mathrm{EXC} \bullet \mathrm{E1C}$

However, the subcommand VRS will be high, and at time one the high order three bits of the V register will be delivered to the S register. Figure 6-5 shows the manner in which this delivery is made.

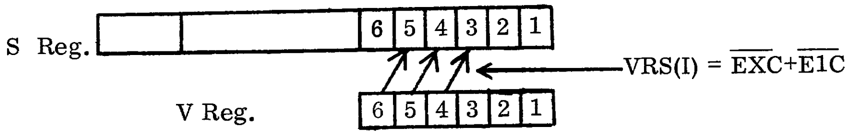

S Reg.
6
5
4
3
2
1
$\mathrm{VRS(I)} = \overline{\mathrm{EXC}} + \overline{\mathrm{E1C}}$
V Reg.
6
5
4
3
2
1

Figure 6-5

The S register will therefore address the low order character of IR #2 which is main memory location ten (octal). At time five the low order six bits of IR #2 will be delivered to the N register. The A register will be cleared at time five; the subcommand N2A will be high for times five and six transferring the contents of the N register to the A register. At the completion of the N2A delivery the accumulator, which is continually adding the A and B register, will contain the binary sum of the low order six bits of the indirect address and IR #2.

CLA (I) = CT5•CAF

N2A = HXC•CT5

The fourth CM cycle addresses WL #1. The Z register will contain IR #2's basic address minus one, and the Write Z functions will be high storing this address in WL #1.

The set gate for ISR (increment S register), DFC•$\overline{\text{EXC}}$•CT8•RUN, which is used for the fetching cycles will be low allowing DSR to come true, thus decrementing the IR #2 address on the S to Z register transfer.

The next state condition for the A1X cycle will be the A2X cycle.

A2X Cycle

The first CM cycle addresses WL #1,and at time one IR #2's address minus one will be delivered to the S register. At time two the binary sum of the low order characters of IR #2 and the indirect address is delivered to the Z register. The second CM cycle addresses the A address counter, WZL will be high, and this result will be stored in CM.

CM cycle three again addresses the A address counter; the subcommand KYM will be high, and at time five the middle character of the indirect address will be gated from the CM sense amps to the Y repeater register. The B register is cleared at time four,and the subcommand Y2B will gate the information in Y to the B register at times five and six.

The N register will contain the middle character of IR #2 at time five, N2A will gate this character to the A register at time six. At the completion of this delivery the accumulator will contain the binary added result of the A and B registers.

The fourth CM cycle addresses WL #1, and the address of IR #2 minus two will be written into memory. EXC being high will again inhibit ISR, allowing DSR.

The next state condition for the A2X cycle will be the A3X cycle.

A3X Cycle

Working location one is addressed for the first CM cycle and at time one its contents are delivered to the S register. At time two the accumulator is transferred to the Z register. WZM is high for the second CM cycle,and the middle character of the resultant A address is delivered to CM.

The third CM cycle again addresses the A address counter, KYU is high and at time five the high order three bits of the indirect A address are transferred to the Y register.At time four the B register is cleared allowing the Y to B register transfer at times five and six.

Y2B = KYU•CT5

The N register will contain the high order character of IR #2 at time five, the subcommand N2A will transfer this character to the A register. At this time the resultant binary sum (the high order character of IR #2 plus the specified indirect address) is contained in the accumulator.

The next state condition for the A3X cycle will be the B1 cycle.

B1 Cycle

The sequence counter is addressed during the first CM cycle and at time one its contents are delivered to the S register. At time two the high order three bits of the resultant A address are delivered to the Z register. The second CM cycle addresses the A address counter and WZU will deliver the remaining portion of the A address to memory. The modification of the A address is now complete.

$$\mathrm{WZU} = \mathrm{BWA} \bullet \mathrm{CT2}$$

$$\mathrm{BWA} = \mathrm{DFC} \bullet \mathrm{INX} \bullet \mathrm{BWB}$$

$$\mathrm{INX\ (I)} = \mathrm{VEO+VE1+CM2} \qquad (\mathrm{INX} = \overline{\mathrm{VEO}} \bullet \overline{\mathrm{VE1}} \bullet \overline{\mathrm{CM2}})$$

$$\mathrm{BWB} = \mathrm{IIN} \bullet \overline{\mathrm{EAC}} \bullet \mathrm{E1C}$$

Summary

The extraction of the B address is nearly identical. The major difference is that octal ten is used to address control memory instead of octal fourteen. By the time the B extraction cycles are complete, the op code has been stored and decoded, and the final addresses of the A and B operands have been stored in the control memory address registers. If no word mark is found in the location beyond the last B address character, all successive characters will be placed in the variant register until a word mark is found. The last character that is put in the variant register is the one that will be used. A word mark terminates the extraction cycles and begins the execution cycles. The extraction cycles make very few distinctions between op codes. However, the execution of each instruction requires slightly different types of cycles. The execution cycles of each type of instruction are explained in detail in later sections of this manual.

SECTION VII
ACCUMULATOR

The Accumulator plays an important role in all machine operations. All of the arithmetic operations (addition, subtraction, zero and add, zero and subtract) and the logical instructions (extract, half add, compare and substitute) are performed within the Accumulator. In addition, the Accumulator is used in the transfer of information to and from Control and Main Memory.

Since the Accumulator is primarily a 4 bit decimal or 6 bit binary adder, the general operation is quite simple. The add logic is pure binary with additional inputs to implement the logical instructions. Decimal operation is accomplished by converting the binary result to binary coded decimal with separate logic.

ADD LOGIC

The interpretation of the binary add logic for any given stage will become apparent by examining the possible input combinations.

	All possible combinations							
A	0	1	0	1	0	1	0	1
B	0	0	1	1	0	0	1	1
CARRY IN	0	0	0	0	1	1	1	1
SUM	0	1	1	0	1	0	0	1
CARRY OUT	0	0	0	1	0	1	1	1

Figure 7•1

From Figure 7•1 the following general statements can be made concerning sums and carries.

SUM = 1 if only one or all inputs are high
CARRY = 1 if two or more inputs are high

Figure 7•2 is a simplified block diagram of the carry propogation showing the add functions SU1-6, carry functions C01-5 and anticipate carry collectors CH3-5 and CF1,2 and 4. It illustrates the usage of anticipatory carry logic to minimize delay due to circuit cascades.

As illustrated, the first three stages of add logic do not make use of the anticipatory carry collectors; therefore, a detailed analysis of only one stage will suffice.

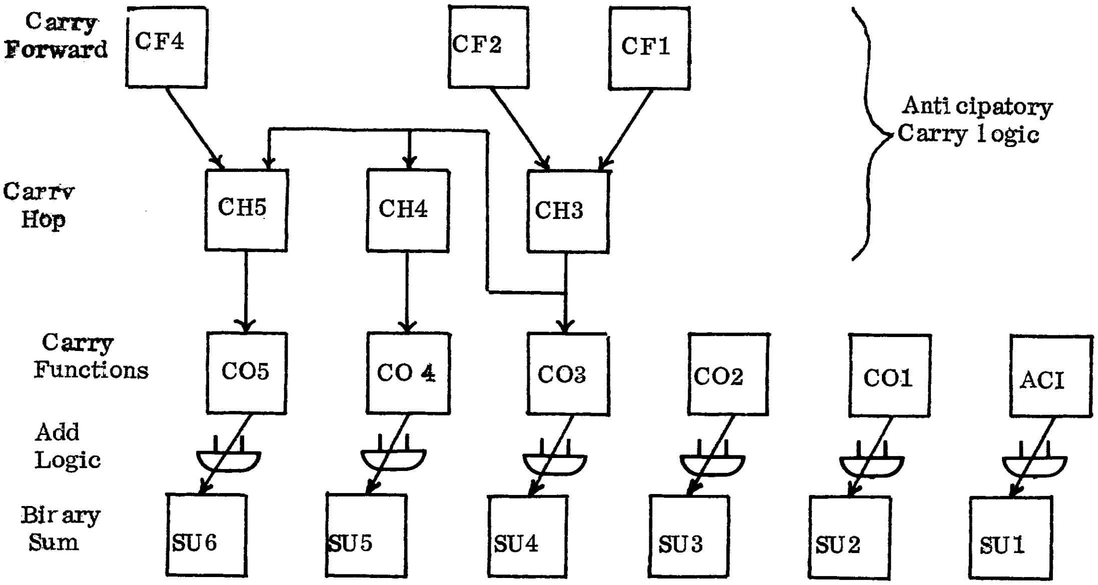


Figure 7•2

STAGE ONE: The first stage of add logic has the following input configuration:

CASU100 (I) = A01•B01•ACI (I) = Inverter
+ $\overline{\text{C01}}$•ACI
+ $\overline{\text{C01}}$•A01
+ $\overline{\text{C01}}$•B01

The first gate will be true when all three inputs are high.

NOTE: ACI is "ACCUMULATOR CARRY IN" and will be covered in detail when discussing order implementation.

The order gates make use of the carry signal C0100.

CACO100 (I) = A01•ACI CAC0100 will be low
+ B01•ACI when there is a carry
+ A01•B01 out of stage one.

This signal, being an inverter, will be at ground when at least two inputs are high satisfying stage one's carry requirements. When $\overline{\text{C01}}$ is high, it indicates that there is <u>no</u> carry.

The add logic using $\overline{\text{C01}}$ therefore insinuates that only one of the inputs are high. $\overline{\text{C01}}$ being at a +5v says that there are not two or more inputs high. This,gated with the individual inputs, will produce the desired output if only one input is true. Thus, the absence of a carry out of stage one may set stage one.

STAGE TWO: This stage makes use of the same input theory with the first gate specifying C0110 as an input carry. C0110, being an inverter, will be high (true) when no input gates are satisfied. The input gate SBS00 and CO100 will allow stage one carry conditions.

NOTE: SBS will only be true during a substitute instruction therefore CO100 is the only controlling signal for CO110 during arithmetic operations.

STAGE FOUR: This stage varies only in the fact that the input carry C0310 is anticipated by CH3 which derives its gating from CF1 and CF2.

CF1, you will note, has as its input conditions the compliment of the inputs to CO100. Therefore it will produce a high output for stage one carry conditions.

$$\begin{aligned} \text{CACF110 (I)} &= \overline{\text{A01}}\bullet\overline{\text{B01}} \\ &+ \overline{\text{A01}}\bullet\overline{\text{ACI}} \\ &+ \overline{\text{B01}}\bullet\overline{\text{ACI}} \end{aligned}$$

CF2 will produce a high output only if stages two and three both contain at least one "1" bit.

NOTE: Because CF2 is an inverter, to arrive at the input requirements for a high output, prime and simplify the boolean equation.

$$\begin{aligned} \text{CACF210 (I)} &= \overline{\text{A02}}\bullet\overline{\text{B02}} \\ &+ \overline{\text{A03}}\bullet\overline{\text{B03}} \end{aligned}$$

$$\begin{aligned} \text{PRIMED:} \quad \text{CF2} &= (\text{A02} + \text{B02})\bullet(\text{A03} + \text{B03}) \\ &= \text{A02}\bullet\text{A03} + \text{A02}\bullet\text{B03} + \text{B02}\bullet\text{A03} + \text{B02}\bullet\text{B03} \end{aligned}$$

The stage four input carry (CH300) can now be defined.

$$\begin{aligned} \text{CACH300 (I)} &= \text{A03}\bullet\text{B03} && (1) \\ &+ \text{A02}\bullet\text{B02}\bullet\text{CF2} && (2) \\ &+ \text{CF1}\bullet\text{CF2} && (3) \end{aligned}$$

(1) carry from stage three.

(2) carry from stage two (A02•B02) will propogate thru stage three because stage three has at least one high input (CF2), plus the carry in.

(3) carry from stage one (CF1) will go thru stages two and three, because they each contain at least one high input (CF2).

The last stages of add logic use similar anticipatory carry logic, with CF4 high indicating at least one "1" bit in stages four and five. The anticipatory carry logic is used to insure that all carries will be propagated by result delivery time.

HALF ADD

In the Half Add instruction, we wish to perform an add without carries. The normal logic will suffice for the summation, carries will be stopped by XOR being high.

$$\begin{aligned} \text{CAXOR10} &= \overline{\text{I01}}\bullet\overline{\text{HFC}}\bullet\text{VAS} \\ \text{CBVAS10} &= \text{U03}\bullet\overline{\text{I03}}\bullet\overline{\text{I02}} \end{aligned}$$

XOR being high will unconditionally cause the assertion carry signals to be off. (XOR is a single input to all the carry functions, and since they are inverters, the outputs will be forced to ground.) The negation carry signals are used on the add logic, and will operate in their normal manner. Due to the forcing off of the carry signals an add will take place but no carries will propogate.

EXTRACT

In the Extract instruction, the data in the A field is combined with data in the B field, and the result is stored in the B field. The execution of this instruction passes A information only if the corresponding B bit is a one. This leads to the result table shown in Fig. 7•3.

A	0	1	0	1
B	0	0	1	1
RESULT	0	0	0	1

Figure 7•3

The Accumulator operation for this instruction is implemented by killing the SU sum gates for a single input high and allowing only the gate specifing all inputs high. The signal LGP will force all negation carry logic (ACI00 and C0100 thru CO600) to ground which will in turn force on the assertion carry logic (ACI10 and CO110 thru CO610).

$$\text{CALGP10} = \text{VAS}\bullet\text{I01}\bullet\overline{\text{HFC}}$$
$$\text{CBVAS10} = \text{U03}\bullet\overline{\text{I03}}\bullet\overline{\text{I02}}$$

Hence, the add logic will produce a "1" bit result only if A and B are both high.

SUBSTITUTE

The Substitute instruction consists of passing the A or B information depending upon the condition of the corresponding Variant bit. If the Variant is a "zero" pass the B information bit, if a "one" pass A.

This involves some rather unusual manipulations of the accumulator logic,but again charting the operation will simplify the solution.

A	0	1	0	1	0	1	0	1
B	0	0	1	1	0	0	1	1
VARIANT	0	0	0	0	1	1	1	1
RESULT	0	0	1	1	0	1	0	1

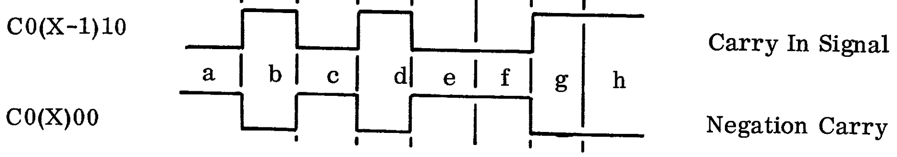


Figure 7•4

Figure 7•4 graphs the logical implementation of any stage. CO(X-1)10 illustrates the status of this stage's carry in signal, which is being forced to desired conditions. CO(X)00 is the normal negation carry out signal for this stage.

EXAMPLE: Examining the operation of stage one in a Substitute instruction, the gating on ACI will justify the graph of the input carry signal CO(X-1)10.

$$\begin{aligned} \text{CAACI10 (I)} \;(\text{b,d,g,h}) &= \text{SBS}\bullet\text{V01}\bullet\overline{\text{B01}} \quad (\text{e, f}) \\ &+ \text{SBS}\bullet\overline{\text{V01}}\bullet\overline{\text{A01}} \quad (\text{a, c}) \\ \text{CASBS10} &= \text{SBT}\bullet\overline{\text{HFC}} \\ \text{CASBT10} &= \text{U03}\bullet\overline{\text{I03}}\bullet\text{I02}\bullet\overline{\text{I01}} \quad (\text{op code } 32_8) \end{aligned}$$

The negation carry signal (CO(X)00) will be provided by the normal gating on CO100.

$$\begin{aligned} \text{CACO100 (I)} \;(\text{a,c,e,f}) &= \text{A01}\bullet\text{ACI} \;(\text{b,d,h}) \\ &+ \text{B01}\bullet\text{ACI} \;(\text{d,g,h}) \\ &+ \text{A01}\bullet\text{B01} \;(\text{d,n}) \end{aligned}$$

With the input carry forced and the stage carry operating in the normal manner the add gates on SU1 will now produce desired Substitute instruction results.

$$\begin{aligned} \text{CASU100 (I)} &= \text{A01}\bullet\text{B01}\bullet\text{ACI} \;(\text{d, h}) \\ &+ \overline{\text{C01}}\bullet\text{ACI} \quad (-) \\ &+ \overline{\text{C01}}\bullet\text{A01} \quad (\text{f}) \\ &+ \overline{\text{C01}}\bullet\text{B01} \quad (\text{c}) \end{aligned}$$

DECIMAL CORRECTION

When performing decimal instructions, the Accumulator will add binarily the two 4 bit characters. Decimal correction will be performed on the result by the SD functions.

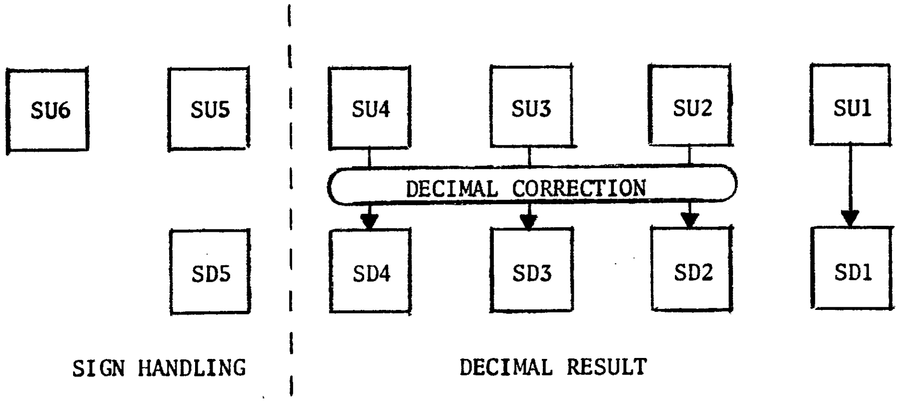


Figure 7•5

A result sign is produced by SD5 and SD6 for the first character. This sign will be a plus or minus depending upon the instruction and operand signs.

Decimal correction will be necessary only when the result is 10 or greater. Figure 7•6 lists the binary result and the corrected decimal character.

Notice that the highest uncorrected result is 19, 9 + 9 plus an input carry.

	BINARY RESULT					DECIMAL CORRECTION					
	SU4	SU3	SU2	SU1		SD4	SD3	SD2	SD1		
	1	0	0	1	9 = 9	1	0	0	1		
	1	0	1	0	10 = 0	0	0	0	0	ACD	4•2 or 4•3
	1	0	1	1	11 = 1	0	0	0	1		
	1	1	0	0	12 = 2	0	0	1	0		
	1	1	0	1	13 = 3	0	0	1	1		
	1	1	1	0	14 = 4	0	1	0	0		
	1	1	1	1	15 = 5	0	1	0	1		
c	0	0	0	0	16 = 6	0	1	1	0		CH4
c	0	0	0	1	17 = 7	0	1	1	1		
c	0	0	1	0	18 = 8	1	0	0	0		
c	0	0	1	1	19 = 9	1	0	0	1		

c = binary carry

Figure 7•6

To indicate the need for correction the function ACD is used. The conditions of SU4 and SU2 or SU4 and SU3 will identify a result of 10 thru 15. A carry from stage four (CH4) will satisfactorily identify a result of 16 or greater.

Since ACD is an inverter the prime of the correct conditions are found on the input gates plus DEC which is on for decimal instructions.

$$\begin{aligned} \text{CAACD10 (I)} &= \overline{\text{SU3}}\bullet\overline{\text{SU2}}\bullet\overline{\text{CH4}} \\ &+ \overline{\text{SU4}}\bullet\overline{\text{CH4}} \\ &+ \overline{\text{DEC}} \\ \text{PRIMED ACD} &= \text{DEC (CH4 + SU4}\bullet\text{SU3 + SU4}\bullet\text{SU2)} \end{aligned}$$

CORRECTION LOGIC

The correction logic (SD1-SD4) will contain normal drop thru logic for either binary results or decimal results below 10 identified by $\overline{\text{ACD}}$. The Decimal Correction Chart figure 7•6 will be used to analyze correction conditions (ACD true).

SD1 - There is no change in bit one for decimal correction therefore SU1 feeds SD1 directly

$$\text{CASD110 (I)} = \overline{\text{SU1}}$$

SD2 - During decimal correction bit two should always be complimented.

$$\begin{aligned} \text{CASD210 (I)} &= \overline{\text{SU2}}\bullet\overline{\text{ACD}} \\ &\quad \text{SU2}\bullet\text{ACD} \end{aligned}$$

SD3 - The correction chart, figure 7•6, illustrates that bit number three should be high for an uncorrected result of 14 thru 17. An input configuration of ACD SU3•SU2 will identify results of 14 and 15. ACD•CH4•$\overline{\text{SU2}}$ will suffice for a result of 16 or 17.

$$\begin{aligned} \text{CASD310 (I)} &= \overline{\text{SU3}}\bullet\overline{\text{ACD}} && \text{(Normal Result)} \\ &\left.\begin{aligned} &+ \overline{\text{SU2}}\bullet\overline{\text{C04}}\bullet\text{ACD} \\ &+ \overline{\text{SU3}}\bullet\text{SU2} \end{aligned}\right\} && \text{(Decimal Correction)} \end{aligned}$$

Logical analysis will be necessary since SD3 will produce a "1" output when all input gates are dead. For normal results ACD is reset and SU3 will be the controlling function. When SU3 not ($\overline{SU3}$) is at ground a one bit will be produced. During decimal correction ACD will be set. For a result of 14 or 15 $\overline{ACD}$ will stop the top gate, $\overline{SU2}$ at ground will kill the second gate and $\overline{SU3}$ at ground kills the bottom gate. For a result of 16 or 17 ACD again takes care of the top gate, $\overline{C04}$ at ground stops the second gate and if SU2 is at ground the last gate is killed and a "1" bit is produced.

SD4 - Here a one bit will be produced for uncorrected results of 18 or 19 and ACD•CH4•SU2 will identify these conditions.

CASD410 (I) = $\overline{SU4}$•$\overline{ACD}$ (Normal)
+ $\overline{SU2}$•ACD
+ $\overline{C04}$•ACD (Decimal Correction)

Here again ACD being high kills the top gate and causes SU2 and C04 to be the controlling functions. If both are high no gate is active and a one bit is produced.

DECIMAL CARRIES

During decimal arithmetic anytime a character result is over 9 a carry must be stored for the next character. This is accomplished by feeding ACD to the storage function for ACl (carry in).

ACI (I)= $\overline{OVF}$•CCT•CT5
OVF = ACD•DEC•CT3

INSTRUCTION FLOW

BINARY INSTRUCTION FLOW

In performing a Binary Add, A and B are added, one six bit character at a time, and the result is stored in B. No signs are present or sensed.

The Binary Subtract is absolute B minus A. This is accomplished by the complement and add theory $B + \overline{A} + 1$.

Figure 7•7 illustrates the cycle flow of the Binary Add or Subtract. A B word mark will terminate the instruction. If an A word mark is sensed first, indicating the end of A characters, "zeros" will be added to B for an Add instruction or subtracted from B for a Subtract instruction. The subtraction of "zeros" from B is actually "all ones" added to B and is performed to propagate any necessary carries.

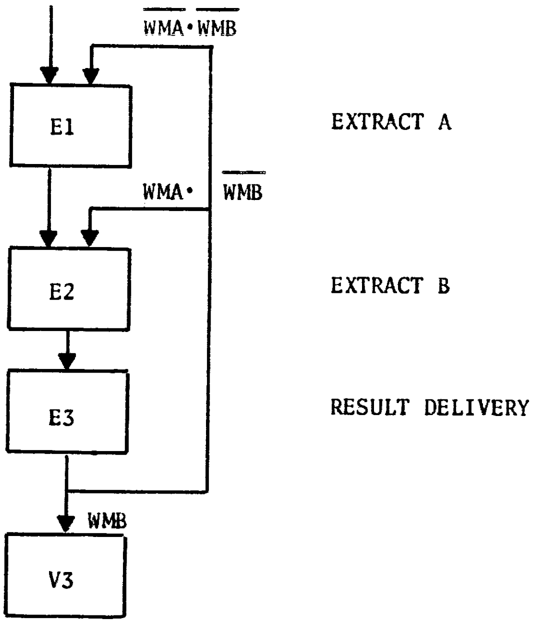


Figure 7·7

E1 Cycle - Extract "A" Character

The A counter is addressed,used, decremented,and returned to A. If a chain B format is used the first pass will cause the A counter to be duplicated in the B counter.

The "A" character, or its complement if a Subtract order, is delivered and stored in the "A" Register.

If a Subtract is being performed the first pass will also force ACI on to accomplish the +1 of $B+\bar{A}+1$.

If an "A" word mark is sensed ACF will be set.

E2 Cycle - Extract "B" Character

The B counter is addressed, used and not changed because B-1 is sent to an unused work location.

The B character is delivered and stored in the B Register. Addition will be automatically performed by the Accumulator.

If cycle E2 was entered from E3 ("A" word mark stored) there are no more valid A character for processing. In an add order the situation is correct since the A Register has been previously cleared, and B will be added to zero's. However, since the subtract order adds B to the complement of A,the A Register should contain all ones. This is accomplished by setting NNA at time 3 of the E2 cycle. The N Register is reset at this time and complemented zeros or all ones are delivered to the A Register.

E3 Cycle - Result Delivery

The B counter is addressed, used for result delivery and decremented.

The result of the arithmetic operation is delivered to the main memory location specified by the B counter.

If a B word mark is sensed it is returned to memory,and the instruction is terminated.

If no B word mark is sensed in this character the instruction will loop back to El if no A word mark had been sensed $\overline{(ACF)}$ or to E2 if there are no more A characters (ACF).
The first loop thru E3 sets PPF,$\overline{PPF}$ was used to identify the first pass thru execution cycles.
If an A word mark had been sensed in El (ACF) then AFC will be set in the E3 cycle (by A4A).

DECIMAL INSTRUCTION FLOW

In the Decimal instructions the four bit characters of A and B are arithmetically manipulated depending upon the Instruction type and operand signs.

Instruction	A Sign	B Sign	Operation	Result Sign
Add	+	+	Sum	+
Add	+	-	Difference	Sign of Larger
Add	-	+	Difference	Sign of Larger
Add	-	-	Sum	-
Subtract	+	+	Difference	Sign of Larger
Subtract	+	-	Sum	-
Subtract	-	+	Sum	+
Subtract	-	-	Difference	Sign of Larger

Figure 7•8

Figure 7•8 illustrates all variations of Instruction and signs. Experiments with the algebraic formula for Adds (B+A) and Subtracts (B-A) will prove this chart correct.

Since both signs must be examined before Instruction operation can be determined the first part of execution flow will concern sign delivery and examination.

First E1 - Extract A Sign

Fortunately the A sign is in the low character of the A operand. Therefore the A counter is addressed and used to extract the first A character. The contents of the A counter are NOT decremented in the 4th control memory cycle at this time (PPF is not true).
As in the Binary instruction, if there were no B address in the instruction format, the A counter contents will be duplicated in the B counter.
The entire first A character is delivered from Main memory to the A Register where the A sign is determined by ASP. ASP is normally set after the extraction cycles and will reset only if a negative A sign is sensed.

$$\begin{aligned} \text{CAASP10 (I)} &= \text{A06}\bullet\overline{\text{A05}} \\ &+ \text{HFC} \end{aligned}$$

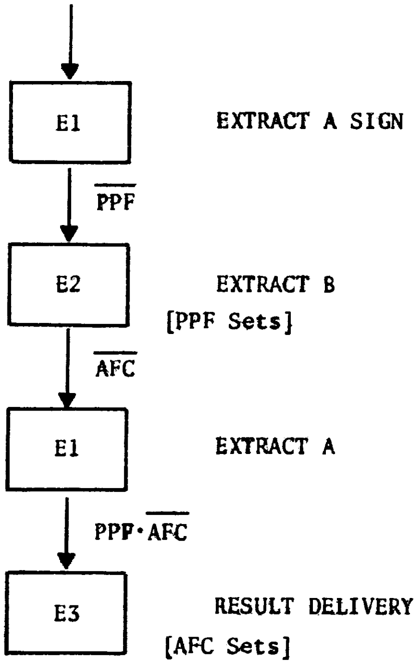


Figure 7'9

First E2 - Extract B Character

The B counter is addressed and used to extract the first B character. No increment or decrement of the B counter occurs.

The second Control Memory cycle duplicates the B counter contents in Work Location 2. This stores the starting point of the B field and will be used if correction is necessary.

The first B character is transferred to the B Register and stored.

The B sign is determined in the B Register by BSP.

$$\begin{aligned} CABSP\ (I) &= B06 \bullet \overline{B05} \\ &+ HFC \end{aligned}$$

ASP and BSP now indicate the signs of the respective operands. The functions are normally high and will go low only if a Negative sign is sensed.

To determine the actual Instruction operation, the A and B signs are compared with the Op Code on the subtract function TUB.

$$\begin{array}{lcll} CATUB10 & = & \overline{I01} \bullet \overline{ASP} \bullet BSP \bullet DDF & \left.\right\} \text{Decimal Add Order} \\ & + & \overline{I01} \bullet ASP \bullet \overline{BSP} \bullet DDF & \quad \text{Signs Unlike} \\ & + & I01 \bullet ASP \bullet BSP \bullet DDF & \left.\right\} \text{Decimal Subtract Order} \\ & + & I01 \bullet \overline{ASP} \bullet \overline{BSP} \bullet DDF & \quad \text{Signs Alike} \\ & + & TUB \bullet CLI & \text{Recirculation} \end{array}$$

Comparing the "True Subtract" input gating with Figure 7•8 verifies that a Difference (subtract) will be performed for a Add order with unlike signs or a Subtract Order with like signs. A Sum (Add) operation is defined by $\overline{\text{TUB}}$. DDF is the gating function to determine comparison time.

Now that the Operation has been determined an interesting situation arises. If a Sum is to be performed the A and B Registers contain the correct characters to proceed into the Execution loop. However if a Difference is to be performed the A character must be "Decimal" complemented. To simplify gating the E1, Extract A, cycle is unconditionally entered again to insure correct A character delivery. Function PPF setting in this first E2 cycle will identify the second E1 as the valid Extract A.

Second E1 - Extract A

The A counter is addressed, the contents used and decremented.

The first A character, or its Decimal complement, will be stored in the A Register.

Accumulator operation will automatically proceed. The B character is already valid. An input carry will be forced if a Subtract (TUB) is being performed.

Since the first character result is generated,the Result delivery cycle will be entered next.

E3 - Result Delivery

The B counter is addressed, the contents used for result delivery, then decremented and returned to the B counter.

The B sign is assumed at this time to be the result sign and is therefore forced on the gates of SD5 and SU6 for return to memory. If an Add is being performed the actual B sign is loaded. In a difference operation the B sign is the result sign only if the B operand is larger than the A operand . Therefore, in difference operations, if B is positive a normalized B sign is produced. This will be corrected later if necessary (A larger than B).

NOTE: A normalized plus sign is the configuration $\overline{6}$•5, if complimented it will produce a negative sign 6.$\overline{5}$.

$\text{SGG} = \text{TUB}\bullet\text{HEC}\bullet\overline{\text{BSP}}$ (Negative B sign in Subtract order)

$\text{SG6} = \text{SGG} + \text{B06}\bullet\overline{\text{TUB}}\bullet\text{DEC}$

$\text{SG5} = \text{SGG}$ (Negative sign) $+ \text{B05}\bullet\overline{\text{TUB}}\bullet\text{DEC}$ (Add-Result sign equals actual B sign)

$\text{SU6} = \text{SG6}\bullet\overline{\text{AFC}}$

$\text{SD5 (I)} = \text{SG5}$

Function AFC will set during the first E3 cycle to cause the normal Execution loop to be entered and prevent sign generation on further result characters.

A loop is entered to process the A and B characters. Figure 7•10 shows that, as in Binary instructions, if an A word mark is sensed the logic will bypass the E1 cycle. The loop for extracting B and result delivery is performed to propagate carries.

NOTE: The function BIZ (Blank is zero) senses an octal 15 being delivered from memory in decimal instructions and will cause a zero or its decimal compliment to be delivered to A if an E1 cycle, and a zero to B if the E2 cycle.

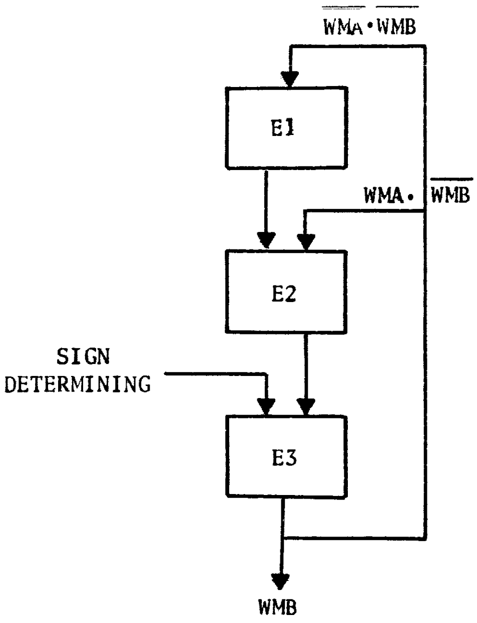


Figure 7·10

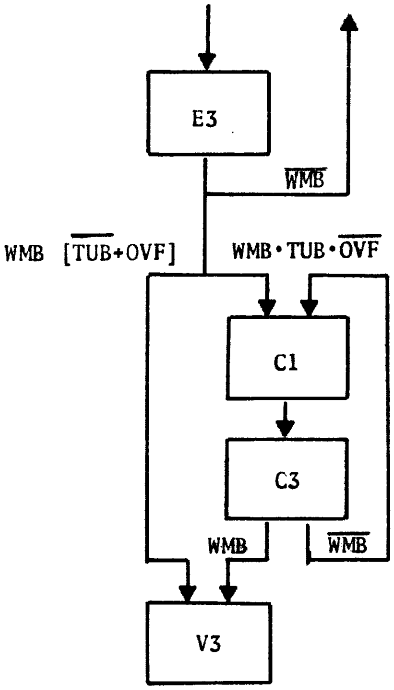


Figure 7·11

SUBTRACT CORRECTION

Upon sensing a B word mark the Add operation is complete. However, a subtract operation may have been performed incorrectly. We subtracted A from B (B-A) which was implemented as B+A+1. A subtract is actually the subtracting of the smaller number from the larger. If B was in fact larger than A we have the correct result and the correct sign (the B sign). However, if A were larger than B the result is not correct, but in fact the complement of the desired result and has the wrong sign.

Example: $B + \overline{A} + 1$ (B-A) where B = 4 and A = 5
$4 + \overline{5} + 1 = 4 + 4 + 1 = 9$ (5 complimented = 4)
+ 9 (incorrect result) complimented = -1

Due to the manner in which the Subtract was implemented, a carry out of the high character (OVF) will indicate correct subtraction, B was larger than A. Therefore, in Figure 7•11 the path from E3 to V3 states: Word mark in B, and Add instruction ($\overline{\text{TUB}}$) or carry out (OVF). The path from E3 to C1 is to enter a correction routine for a incorrect Subtract; Word mark in B, subtract (TUB) and no carry out ($\overline{\text{OVF}}$).

The correction cycles will correct the result sign and complement the result characters.

C1 - Extract and Complement

Work location 2 is addressed (it was loaded with the starting location of the B field in the first E2 cycle) and used to extract the B character.
This count is loaded into the B counter during the second Control Memory Cycle.
The result character is Decimal complemented upon delivery to the A register.
A carry in is forced for the first correction cycle.
The sign is also complemented and therefore corrected.

C3 - Result Delivery

The B counter is addressed for result delivery.
Since the B register was never loaded,A will be added to "zeros" in the accumulator and the result delivered to Main Memory.
Function AFC is set it was reset to indicate the first pass thru the correction loop.

The correction loop will correct the result field and will terminate when the B word mark is sensed.

SECTION VIII
LOGIC INSTRUCTIONS

- EXTRACT
- HALF ADD
- COMPARE
- SUBSTITUTE
- BRANCH IF CHARACTER EQUAL
- BRANCH
- BRANCH ON CHARACTER CONDITION

The "logic instructions" is a general heading for those machine instructions which require a logical decision, as opposed to an arithmetic operation, or transfers etc. The logic instructions can be further broken down into branch and non-branch instructions, but the separation is not as distinct as it may at first appear. After all, the decision to branch is really a logical operation. Further, one Branch uses the accumulator to make the branch decision, while another can branch on indicators set by the accumulator. The accumulator is used to perform all other logical instructions.

EXTRACT

The extract instruction can be thought of as acting like a sieve. The 'A' field is sifted through the 'B' field, and the result is determined by the bit in the 'B' field (aperture of the sieve). Since the result is placed right back in the B field, the 'mask' used as the sieve is destroyed as a result of the operation. A somewhat more sophisticated explanation is that of the "logical product". The product being a bit by bit multiplication. Note the following example;

A	Field	0	0	1	1
B	Field	0	1	0	1
Result	Field	0	0	0	1

The 'B' field is indeed acting as a sieve with the one bits being the apertures. But also note that the logical product holds true as well, since the only one bit in the result is the product of two ones. The product of a one and a zero, or two zeros is of course zero. This again leads to another conclusion, the fact that a one in the result field can only occur if the A field and 'B' field bits are both ones.

Referring to the extract order in the flow charts, one can see that in E1 cycle the first A character is extracted from memory and transferred to the A register. In E2 cycle the first B character is extracted from memory and transferred to the B register. In E3 cycle the accumulator result is gated to the N register, and written into memory (address of the B character presently in the B register). It's rather obvious then, that the 'extract' or 'logical product' must have occurred in the accumulator.

The function CALGP10 is OP code derived and will be high only for an "extract" instruction. LGP, logical product, is used to set all of the accumulator carry functions. This leaves the accumulator output (CASU$^{1}_{6}$00) with only one possible set gate, i.e. the gate describing the condition of a "carry in" (forced by LGP) and a one bit in both A and B fields. Thus in order for the result bit to be a one, both operand bits must be ones.

The instruction returns to E1 cycle to extract another A character and the whole thing is repeated until a word mark is detected in either the A field or the B field. When a word mark is sensed in the A field (read out in E1 cycle) the indicator CBACF10 is set and recirculated until the next op code extraction cycle. It is during E3 (result delivery) cycle that the decision to go back for another A character, or to exit the instruction, is made.

A requirement to set another EEC, or another E1C, is $\overline{\text{N07}}$. So if a 'B' word mark is sensed first, an E1 cycle is not allowed Instead an E3 cycle is set. Since no major cycle can be set, a V3 cycle exists and the instruction is terminated. Had an A word mark been sensed first, then CBACF10 will have been set at the end of the E1 cycle in which it was read out of memory. $\overline{\text{ACF}}$ is a requirement for the back-up functions of both EEC and E1C when exiting the E3 cycle. Since ACF is set, an E1 cycle <u>cannot</u> be set. In order to exit the instruction, however, an <u>E3C</u> must be set. A special gate is used for the purpose of exiting an "extract" or a "half add".

$$\begin{aligned}
\text{E3C} &= \text{CT6}\bullet\text{S3D}\bullet\overline{\text{ITF}}\\
\text{S3D} &= \text{S3E}\\
\text{S3E} &= \text{WTN}\bullet\overline{\text{EDT}}\bullet\overline{\text{MUV}}\bullet\overline{\text{MPM}}\\
\text{WTN} &= \text{ACF}\bullet\text{VAS}\bullet\text{DE3}\\
\text{VAS} &= \text{U03}\bullet\overline{\text{I03}}\bullet\overline{\text{I02}}
\end{aligned}$$

WTN is normally used to set N07, to write a word mark into memory. But because CAIN710 cannot be set, bit seven is written back from the sense amps and remains unchanged.

<u>HALF ADD</u>

The only difference between the "extract" and the half add occurs in the accumulator. Operand extraction, loading the accumulator, delivery of the result, and exiting the instruction is exactly the same for both. Gating LGP into the accumulator for the extract instruction effected the desired combination of bits for that instruction. For the "half add" instruction the sub-command CAX0R10 is generated to set the accumulator to such a state as to allow the following combination of the A and B field bits.

A field	0	0	1	1
B field	0	1	0	1
Result field	0	1	1	0

$\overline{\text{ACI}}$ due to XOR

Note that when one or the other of the input bits is a one the result is a one . If neither or both input bits are ones, the result is a zero. The result is the same as though the two operands had been added binarily but without propagating any carries. This is the method the accumulator uses to perform the half add , which is sometimes called an exclusive OR (XOR).

CAX0R10 is used to reset all of the accumulator carry functions to effect the non-propagation of carries. Resetting the carry functions (assertions) does not affect the negations since they are <u>not</u> cross coupled. The accumulator output ($\text{CASU}^{1}_{6}\text{00}$) has only two possible input gates as a result.

$$\begin{aligned} \mathrm{CASU200(I)} &= \overline{\mathrm{C02}}\bullet \mathrm{A02} \\ &+ \overline{\mathrm{C02}}\bullet \mathrm{B02} \\ \overline{\mathrm{C02}}\ \mathrm{(I)} &= \mathrm{A02}\bullet \mathrm{B02} \end{aligned}$$

If both A02 and B02 are ones , $\overline{\text{C02}}$ is forced low, preventing $\overline{\text{SU2}}$ from going low, and forcing $\overline{\text{SU2}}$ high. Thus if both A02 and B02 are true the result is a zero. If $\overline{\text{C02}}$ is high, SU2 will be set if either A02 or B02 is high.

The single addition of X0R has changed the accumulator parameters so that the operands are added without carries. The affect of X0R on the "carry in" function ACI differs slightly from the "carry out" functions. Rather than resetting the assertion directly (as on all other carry functions) $\overline{\text{X0R}}$ is used to recirculate ACI. So when X0R is true, ACI cannot be recirculated therefore it is held low during a half add.

SUBSTITUTE

The "substitute" instruction has the format F/A/V, and its function is to replace the B character bits with bits from the A character when corresponding bits of the variant are one's. Where the variant bits are zero, the corresponding bits of the B character are not altered. The substitute rule is as follows:

A Character Bit	0	0	0	0	1	1	1	1
B Character Bit	0	0	1	1	0	0	1	1
V Character Bit	0	1	0	1	0	1	0	1
Result Character Bit	0	0	1	0	0	1	1	1

The substitute is a one character instruction. Exit from the fetching cycles is normally from V2 cycle, when the op code of the next instruction is read out and recognized because of its word mark. The variant character had been placed in the V register in V1 cycle and recirculated. The instruction can be chained, allowing entry into the execution cycles from A1 cycle (OP code of the next instruction read out in A1). These are the only legal instruction formats. Entrance into the execution cycles can be accomplished from B1 cycle for the case of OP code and A address only, but the contents of the variant register and consequently the results of the substitute are unspecified for this format.

During execution, the A character is extracted from main memory in E1 and placed in the 'A' register. The B character is extracted in E2 and transferred to the B register. The variant character is still recirculated in the V register. The A and B registers are gated into the accumulator normally. The V register is gated into the accumulator carry functions with the subcommand CASBS10, which comes high only for the substitute instruction. The carry functions (both assertions and negations) and the inputs to the accumulator sum bits ($\mathrm{CASU}^{1}_{6}00$) can be simplified to the following expression using boolean algebra.

$$\begin{aligned} \mathrm{CASUx10} &= \mathrm{AOx}\bullet \mathrm{BOx} \\ &+ \mathrm{AOx}\bullet \overline{\mathrm{BOx}}\bullet \mathrm{VOx} \\ &+ \overline{\mathrm{AOx}}\bullet \mathrm{BOx}\bullet \overline{\mathrm{VOx}} \end{aligned}$$

Note that this expression will comply with all requirements of the substitute rule. The reader is invited to trace several conditions through the accumulator, as well as simplifying the gates to arrive at the expression shown above.

Since the substitute is a single character instruction, it must be terminated after the result of the substitute is delivered in E3 cycle. Since no major cycle can be set, a DVC comes high. An E3C minor cycle is set, as can be seen by the following expression.

$$\begin{aligned} \mathrm{CBE3C10} &= \overline{\mathrm{N07}}\bullet \mathrm{S3A}\bullet \mathrm{CT6} \\ &+ \mathrm{N07}\bullet \mathrm{S3B}\bullet \mathrm{CT6}\bullet \overline{\mathrm{PST}} \\ \mathrm{S3A} &= \mathrm{SBT}\bullet \mathrm{DE3} \\ \mathrm{S3B} &= \overline{\mathrm{P1C}}\bullet \mathrm{PEC}\bullet \mathrm{TEC} \\ \mathrm{TEC} &= \overline{\mathrm{TUB}}\bullet \mathrm{P3C}\bullet \overline{\mathrm{MUV}}\bullet \overline{\mathrm{MAT}} \end{aligned}$$

Notice that E3C will be set after E3 regardless of punctuation. The gate on E3C with N07 as an input is used for those instructions which normally exit from E3 when a word mark is sensed. The gate including $\overline{\mathrm{N07}}$ requires the backup function S3A, which requires CBSBT10 (high only for substitute) and CBDE310. So this gate is specifically used to allow exiting the substitute after one pass.

COMPARE

The compare instruction is, in reality, a binary subtract. When the instruction is first issued,the two compare indicators are initialized (CAALB10 = Reset; and CAAEB10 = Set). If the A and B operands are equal, then all during the compares the accumulator output will be zero. If this is true, then AEB will remain set. If the operands were not equal, the accumulator output would not be zeros, and this would reset the equality indicator, AEB.

The instruction is terminated by a word mark in the B field. If at this time accumulator overflow is sensed, the function ALB (A less than B) is set. Overflow is a valid indicator of relative magnitude because the A operand is subtracted from the B operand by complimenting A and adding. If the A operand were larger than B , there could not be any overflow, thus ALB would remain reset.

If the A operand is shorter than the B operand, no A characters are extracted beyond the A word mark. The instruction is completed by subtracting zeros from the remaining characters of the B field. This of course implies that a true comparison AEB can exist, even though two operands are of unequal lengths, if the B operand characters extracted after the A word mark are all zeros.

Note from the flow charts that entrance into the execution cycles is possible from A1 for an instruction format of OP code only, B1 for OP code and A address and V1 for OP code A and B address. The order of the execution cycles and their functions (briefly) are as follows:

In the first E1 cycle the compare indicators are preloaded; the first A character is complemented and sent to the A register, and the "end around carry" is forced into the accumulator. The first is due to the op code derived sub-command CABCP10, and $\overline{\mathrm{PPF}}$. The second is due to BCP having set CASA010 (binary subtract). And the third is a result of SA0 and $\overline{\mathrm{PPF}}$. It is during this and succeeding E1 cycles that the A character is checked for punctuation, and CBACF10 is set if a word mark is detected.

COMPARE (F/A/B)

CONDITIONS: A = B

A = 0 ⓪ 0 0

B = ⓪ 0 0 0 0

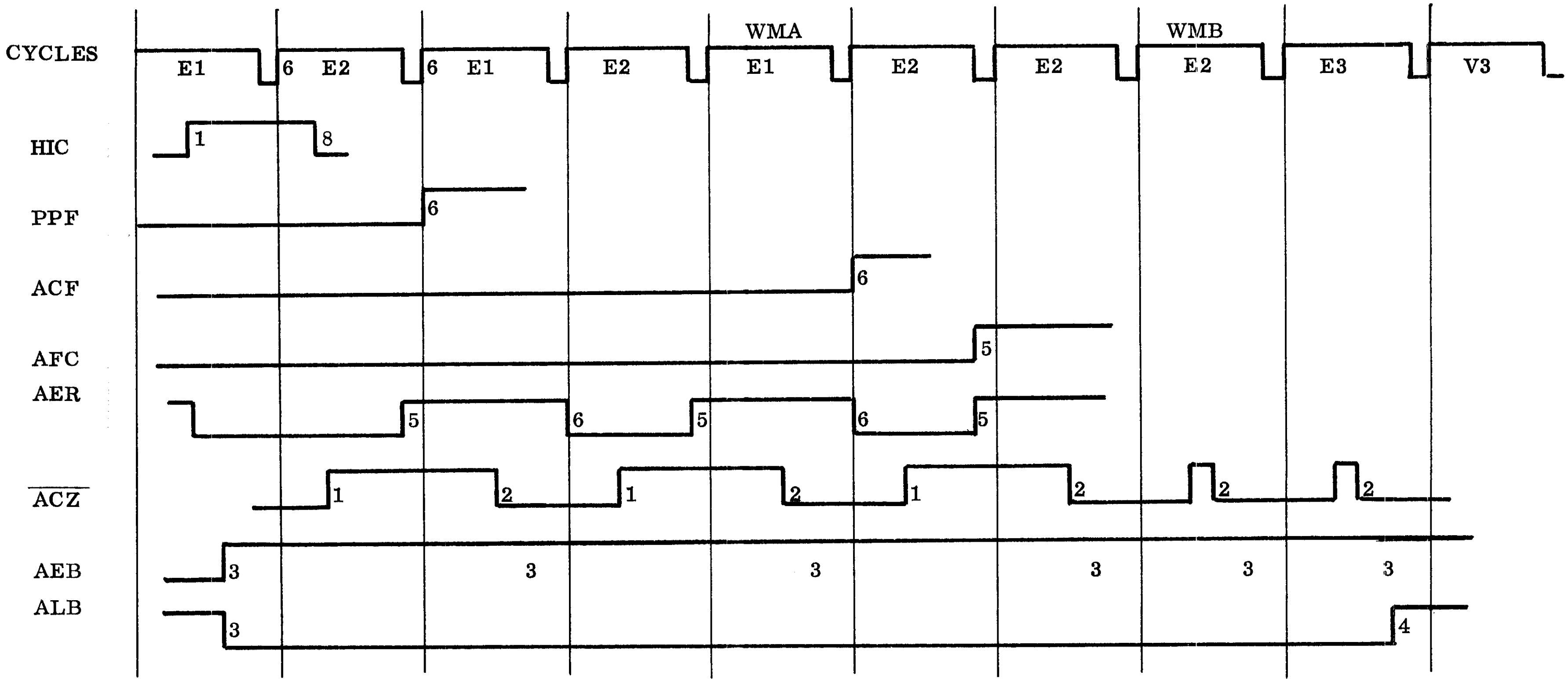


FIGURE 8-1

Because the indicator AEB is set and allowed to remain set only if the accumulator result is zero, the accumulator must only be looked at when its output is valid. Otherwise AEB would be reset when the first character was in the accumulator but the other operand had not been extracted from memory yet. The function that allows this is CAAER10.

Fig. 8-1 shows the pertinent waveforms for a compare with the conditions shown. Note the state of the functions shown for the first E1 cycle.

In E2 cycle the B character is extracted from main memory, transferred to the B register, and from there gated with the complemented A character into the accumulator. At the end of the first E2 PPF is set to indicate the end of the first pass. If a word mark was read from memory in this cycle, the next would be an E3 cycle to allow exit. If no word mark is read out, the next cycle is E1 to extract another A character (if a word mark had not already been sensed (ACF) in the A field). If the A field had been terminated (ACF), the machine would stay in E2 extracting B characters until a word mark was found. For the latter case, the A register is loaded with the complement of the N register which, at time three,is still reset. Therefore, in effect, zero's are subtracted from the remaining B field characters after the A field has been terminated.

Refer again to Fig 8-1. Notice that the first time AEB could be reset is at time three following E2. Since the B character read out of memory in E2 is transferred through the B register into the accumulator at time 6, and CBACZ10 does not check the accumulator for zeros until time 2; then four pulse periods (1 microsecond) are allowed for the adder to settle out and the zero indicators to come high. Note also that until the A field is complete(ACF and AFC) the function AER will kill the reset gate for AEB during E2 cycle.

$$\mathrm{AEB\ (I)} = \mathrm{HEC} \bullet \mathrm{BCP} \bullet \overline{\mathrm{ACZ}} \bullet \mathrm{AER} \bullet \mathrm{CT3}$$

After the A field terminates AER is always high. So, every time three AEB will be allowed to reset if the accumulator result is not zero. 1 us is still allowed for the accumulator to settle.

A word mark detected in the B field causes the next cycle to be an E3 cycle. Although E3 is a normal delivery cycle (adder to N to memory), this is inhibited in a compare instruction, and the contents of the sense amps are written back into memory.

$$\begin{aligned} \mathrm{MAIML10} &= \mathrm{NLD} \\ \mathrm{NLD} &= \mathrm{INL} \\ \mathrm{INL} &= \mathrm{I2L} \bullet \mathrm{DE3} \bullet \mathrm{CT2} \\ \mathrm{I2L} &= \overline{\mathrm{MUV}} \bullet \overline{\mathrm{BCP}} \bullet \overline{\mathrm{EDT}} \bullet \overline{\mathrm{BCH}} \end{aligned}$$

At time three the accumulator is checked for zeros one last time. If set, AEB cannot be changed except by another compare. An 'interupt' will store the compare indicators in the auxiliaries and reset the indicators, but it will restore the old contents after a "resume normal mode" instruction is issued.

$$\mathrm{ALB} = \mathrm{DE3} \bullet \mathrm{BCP} \bullet \mathrm{OVF} \bullet \mathrm{CT4}$$

At time four of the E3 cycle the magnitude indicator (ALB) is set, if overflow is high. For the conditions shown in Fig. 8-1, ALB is set because the two operands are equal. There will be an accumulator overflow if A is less than or equal to B. Once set,ALB can only be reset by another compare.

BRANCH (GENERAL)

The actual mechanics of any branch is exactly the same. The decision to branch is logically determined by various functions and indicators, but the actual branch is accomplished by loading the sequence register in control memory with the address of the op code of the next instruction to be performed. Since all branch instructions use the A address to store the address to which the sequence counter is to be set for a branch, all that is required is to transfer location 14 to location 17 of control memory. But another specification of the branch is that the contents of the sequence register at the time of the branch is to be stored in the B address counter of control memory (location 10). This is accomplished by transferring the sequence counter into the B address counter, and then transferring the A address counter into the sequence counter.

The actual transfer of the contents of any control memory location into another control memory location is rather simple and ingenious. It is accomplished by reading out the information (of the location to be transferred) into the sense amps normally. The location to which this information is to be transferred is forced into the R register in the following control memory cycle. But the sense amp reset and strobe pulse are inhibited by not allowing CBIIT10 to be set. Killing the reset pulse allows the sense amps to recirculate the information to be transferred for an additional control memory cycle. Killing the strobe prevents the destruction of that information during the read time of this second control memory cycle. So during write time when the sense amps are written back into control memory, the information written in was really read out in the previous cycle from a different location. Thus an internal transfer is accomplished.

The various branch instructions differ only in how the decision to branch is made. An unconditional branch will occur for the "branch on indicators" instruction (mnemonic = branch; op code = 65), if the instruction format is F/A instead of F/A/V.

BRANCH IF CHARACTER EQUAL (BCE)

The normal format of this instruction is F/A/B/V, and on execution will cause a branch to the A address if the character specified by the 'B' address is equal (bit by bit comparison) to the variant character. After the discussion of the compare instruction, the student might be willing to wager that the equality of B and V would be determined by subtracting one from the other. But this conclusion would only show that he had been thinking and was not necessarily a gambling man. Actually the equality is determined by half adding the two characters. This half adding is really much easier to accomplish than subtraction. If the result is zero a branch is performed; in either case, the result is ignored.

The BCE instruction on extraction will have placed the variant character into the B register in V1 cycle. In V2, since the op code of the following instruction was read out, N07 will prevent the B register from being reset. Therefore in E2, the first execution cycle, the variant character is still present in the B register. The character specified by the B address is read out of memory now and transferred to the A register.

$$\text{CAX0R10} = \text{BCH}\bullet\text{I01}\bullet\overline{\text{HFC}}$$
$$\text{CABCH10} = \text{I06}\bullet\overline{\text{I05}}\bullet\text{I04}$$

Since the OP code of a BCE is 55 (octal) X0R is set, allowing the accumulator to half add the contents of the A and B registers. The following cycle (E3) allows the accumulator time to settle out, and at time 2, if the two characters were equal, CAACZ10 is set. There is a transfer from accumulator to N , but since the sense amps and not the N register is written back into memory,it doesn't matter. The next cycle is M1.

It is in M1 cycle that the actual branch mechanics occur, and since CBIIT10 must not be allowed to be set for a branch, this is a good spot to begin back tracking.

IIT = II1•T01
II1(I) = BI1•E1C•CPC
BT1 = EMC•PST
PST = P2P•T07
P2P = ACZ•BCH•EMC•I01•E1C

IIT = II5•T05
II5 (I) = DM1•AFC•CPC

$\overline{\text{IIT}}$ will allow transfer of location 17 to location 10 if B was equal to V. Note that ACZ (accumulator is zero) is used to set P2P, which sets PST, which sets BI1 which forces II1 low, therefore not allowing IIT to come high at T01.

Since PST, once set, will recirculate until a previous V3 cycle, IIT will not be allowed to come high at T05. As a result, the A counter (location 14) is transferred to the sequence counter (location 17).

The next cycle will be a V3 cycle, and if a branch occurred the sequence counter will extract the op code indicated by the 'A' address of the branch. If the characters were not equal the sequence counter would be left unaltered, and the next instruction in sequence would be extracted.

BRANCH ON CHARACTER CONDITION (BCC)

This instruction has the format F/A/B/V. The variant character is placed in the V register in V1 cycle. In the first execution cycle (E2) the character specified by the B address is read out of memory and transferred to the A register. The punctuation bits set two indicators: CBACF10 for a word mark, CBAFC10 for an item mark, and both of them for a record mark.

In E3 cycle the sub-command A2N is generated, and even though a transfer from the adder to 'N' does occur, the sense-amps are written back into memory. A2N is used here to time the function CACOK10.

$$\text{CACOK10} = \text{A2N}\bullet\text{C0H}\bullet\text{C0JAP}$$

CAC0H10 is high if the condition specified by the bits of the variant register is true. C0H can be used to test for the following conditions if the machine is not equipped with the option (function C0J).

10 - word or record mark
20 - item or record mark

06 - negative sign
02 - high order bit of the B character is a one

The variants may be combined. For example,a 36 (octal) would allow C0H to be high only for a character with a record mark and a negative sign.

$$\begin{aligned} \text{CAC0H10 (1)} &= \text{V03}\bullet\overline{\text{V01}}\bullet\text{A05} \\ &+ \text{V05}\bullet\overline{\text{AFC}} \\ &+ \text{V04}\bullet\overline{\text{ACF}} \\ &+ \text{V02}\bullet\overline{\text{A06}} \end{aligned}$$

$$\begin{aligned} \text{CAC0J10} &= \text{V03}\bullet\overline{\text{V02}}\bullet\text{A06} \\ &+ \text{V06}\bullet\overline{\text{V05}}\bullet\text{AFC} \\ &+ \text{V06}\bullet\overline{\text{V04}}\bullet\text{ACF} \\ &+ \text{V01}\bullet\overline{\text{A05}} \end{aligned}$$

An octal 36 will cause $\overline{\text{V06}}$, V05, V04, V03, V02, $\overline{\text{V01}}$. With these V register bits true, the top or bottom gate will drive C0H low for anything but a negative sign (B character in the A register). The middle two gates can only be low if a record mark had set AFC and ACF. The concept used here is to gate the checking bits with the negation of the condition to be checked. This way if the condition checked for does not exist, C0H is driven low not allowing a branch. Without the option CAC0JAP is tied to +5.

For the option, the function CAC0J10 is used to further define the character presently in the A register. The chart at Fig. 8-2 shows the variant configuration and the character condition which will allow both C0H and C0J to be high, thus allowing a branch. Note that V06, V05, and V04, only check punctuation while V03, V02, and V01 check the sign bits of the character in the A register.If 00$_8$ is the variant, no combination of conditions can ground C0H or C0J to stop C0K from causing a branch.

CONDITIONS FOR BRANCH

V06	V05	V04	N08	N07	V03	V02	V01	N06	N05
0	0	0	x	x	0	0	0	x	x
0	0	1	x	1	0	0	1	x	1
0	1	0	1	x	0	1	0	1	x
0	1	1	1	1	0	1	1	1	1
1	0	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	1	0	1
1	1	0	1	0	1	1	0	1	0
1	1	1	1	1	1	1	1	1	1

x = Don't care

Fig. 8-2

In M1, the following cycle, the control memory transfers which effect the branch actually occur. The only difference in the M1 cycle for the BCE and BCC instruction is that a different gate is used to set CBP2P10 and cause the branch.

$$\text{CBP2P10} = \text{BCH}\bullet\text{COK}\bullet\overline{\text{I01}}\bullet\text{EMC}\bullet\text{E1C}$$

Where the BCC used ACZ (accumulator zero) to indicate a branch condition, the BCE instruction uses C0K (conditions OK) to indicate the branch condition.

BRANCH (BRANCH ON INDICATORS)
This branch instruction has two formats:The first (F/A) will cause an unconditional branch to the A address, and the second (F/A/V) will cause a branch if the indicator specified by the variant character is set. The indicators this instruction can check are listed in the following chart.

		ARITHMETIC		COMPARE
CSI	=	Decimal overflow	AEB	A is equal to B
Ƶ	=	Zero balance indicator	ALB	A is less than B

The variant can also check the setting of the four control panel "sense" or "break point" switches. The branch conditions that can be checked by the variant character are shown in Fig. 8-3.

OCTAL VARIANT	BRANCH ON	OCTAL VARIANT	BRANCH ON
00	Unconditional	50	Overflow indicator on
01	Switch 1 on	51	Overflow or (B) <(A)
02	Switch 2 on	52	Overflow or (B) = (A)
03	Switch 1•2 on	53	Overflow or (B) ≤(A)
04	Switch 3 on	54	Overflow or (B) >(A)
05	Switches 1•3 on	55	Overflow or (B) ≠ (A)
06	Switches 2•3 on	56	Overflow or (B) ≥(A)
07	Switches 1•2•3 on	57	Unconditional
10	Switch 4 on	60	Zero Balance indicator on
11	Switches 1•4 on	61	Zero Balance or (B) <(A)
12	Switches 2•4 on	62	Zero Balance or (B) =(A)
13	Switches 1•2•4 on	63	Zero Balance or (B) ≤(A)
14	Switches 3•4 on	64	Zero Balance or (B) >(A)
15	Switches 1•3•4 on	65	Zero Balance or (B) ≠(A)
16	Switches 2•3•4 on	66	Zero Balance or (B) ≥(A)
17	Switches 1•2•3•4 on	67	Unconditional
20-37	Undefined	70	Overflow or zero balance
40	Do not branch	71	O'flow or zero bal or (B)< (A)
41	(B) < (A) (low compare)	72	O'flow or zero bal or (B)= (A)
42	(B) =(A)	73	O'flow or zero bal or (B)≤ (A)
43	(B) ≤(A)	74	O'flow or zero bal or (B)> (A)
44	(B) >(A) (high compare)	75	O'flow or zero bal or (B)≠ (A)
45	(B) ≠(A)	76	O'flow or zero bal or (B)≥ (A)
46	(B) ≥(A)	77	Unconditional
47	Unconditional		

Figure 8-3

Exit from the fetching cycles for the three formats are A1 cycle for OP code only; B1 cycle for F/A (unconditional branch) and V2 cycle for the F/A/V. For the last format, the variant character is placed in the V register in the first V2 cycle. The second V2 cycle reads out the op code of the next instruction. N07 being true allows the V register to recirculate and also allows exit into the execution cycle, M1.

There is only one execution cycle, and that is an M1 cycle. This is the same cycle used in the BCC and BCE instruction to effect the branch. The only real difference is another gate is used to set CBP2P10.

$IIT = II1 \bullet T01$

$II1 = E1C \bullet BI1 \bullet CPC$

$BI1 = EMC \bullet PST$

$PST = P2P \bullet T07$

$IIT = II5 \bullet T05$

$II5\ (I) = DM1 \bullet AFC \bullet CPC$

$AFC = PST \bullet DM1 \bullet CT1$

$PST = P2P \bullet T07$

$$\begin{aligned} P2P &= IIN \bullet N07 \bullet PBC \bullet B0I \\ &+ B0K \bullet EMC \bullet B0I \bullet E1C \end{aligned}$$

* For the unconditional branch (format F/A): a word mark is sensed in B1 of the fetching cycles when the OP code of the next instruction is read out. CBB0I10 is an op code derived subcommand, and CBIIN10 is high during the fetching cycles if an indirect or indexing cycle is not being performed. Thus the requirements for an unconditional branch is determined before the first and only execution cycle (M1) is set up.

The other gate to set P2P requires an M1 cycle as well as the op code derived subcommand B0I. But the function that determines if a branch is to occur is CAB0K10 (Branch OK).

The arithmetic indicators are gated with V06 and their respective V register checking bits right on the input gates of B0K.

The compare indicators with their checking V register bits set the collector function CAAIS00, and this function is gated with V06 to set B0K.

The negations of the breakpoint switches are gated with their respective V register checking bits. So if a switch is checked and it is not set, the gate is high forcing CABPS00 (output of an inverter) low. CABPS00 is gated with $\overline{V06}$ to set B0K.

Any high gate on B0K indicates that a condition specified by the variant character has been met, and a branch is desired. B0K then comes high and sets P2P which at time 7 sets PST, the branch indicator.

The overflow indicator CACSI10 is reset as a result of being checked by the branch instruction. This is the only indicator reset in this manner. All other indicators checked by the branch are not changed as a result of being checked.

SECTION IX
CONTROL INSTRUCTIONS

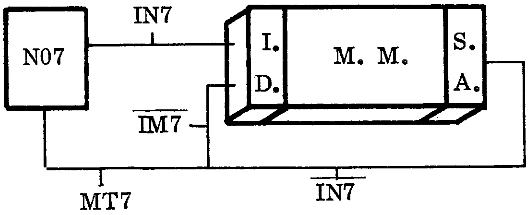
SI, CI, SW, CW

The flow and implementation of these four orders is all about the same. The description that follows will be specifically about the SW instructions, with the variations necessary for the other three noted as they occur. Three acceptable formats are given in the "Programmers Reference Manual". However, after the extraction cycles, only one distinction between formats is made. If the instruction is going to duplicate A, in E1 the contents of the A address counter are transferred to the B address counter. The SW instruction will set a word mark in the location specified by each address, whether that location already had a word mark or not. A good understanding of the handling of word marks going in and out of memory is necessary to grasp the significance of some functions used in this instruction. Therefore review of word mark manipulation is in order.

Word Mark Handling

Normally, when memory is cycled,bit seven is read into the sense amps. From there it is gated to the N register. In the write half of the memory cycle, the inhibit driver is fed by sense amp seven. Even though sense amp seven is gated to the N register, N07 will not have any control over the contents of bit position seven in memory. In other words,bit seven is available for examination but cannot be changed. On the other hand, when changing the word mark, bit seven does not go from the sense amps to the N register, and the N register is used to drive the inhibit drivers.

Three subcommands do all the gating. MAIM720 gates the sense amps to the inhibit drivers. CAMT710 gates the sense amps to the N register. CAN7D10 gates the N register to the inhibit drivers. All three of these functions are controlled by CAIN710. The normal cycling of word marks will be from the sense amps to the inhibit drivers, as long as IN7 is at ground. With IN7 at ground, MT7 and $\overline{IM7}$ will be high, and N7D will be low. Therefore, to change bit position seven in memory it is necessary to set IN7 and put the new bit in N07. Every instruction that can change word marks has an op code derived function on the input of IN7.



The Normal Recirculation Is S. A. to I. D. by $\overline{IN7}$.

$\overline{IN7}$	IN7
MT7	N7D
$\overline{IM7}$	

Figure 9-1

In a set word mark instruction IN7 is set,and a bit is forced into N07. Nothing more is required to set a word mark. To clear a word mark, it is only necessary to set IN7 without forcing a bit into N07. The logic for setting and clearing item marks is almost indentical (only the names have been changed).

E1

Normal entry into the execution cycles will be from V1. The first execution cycle will be E1 in all cases. In E1 the A address counter will be used to address memory. IN7 will be set by HEC•MPM•I02•CT2. CAWTN10 will be set by ACF•DEM•I02•$\overline{I01}$. The op code and EEC will set CBACF10 (EEC•SPM•I02).

As long as WTN is high,N07 will be high. Because IN7 is high, the inhibit driver for bit seven will be fed by N07. A word mark will be written into memory during E1 because both IN7 and WTN will be high. The difference between SW and CW is slight. On WTN,the gate that says ACF•DEM•I02•$\overline{\text{I01}}$ will not be high because I01 will be high for CW instructions. Therefore, during CW instructions IN7 comes high allowing N07 to write into memory, but no gate on WTN will come high, so position seven will be a zero. The CI and SI instructions work the same way except that IN8,ITN,and AFC are used instead of IN7, WTN, and ACF. After E1 the next cycle will be E2.

E2

E2C will be selected by S2F•P1C at CT6. CBS2F10 is set by PEC and S2E. CBS2E10 is an inverter that will be reset during MAT instructions, decimal add and subtracts under certain conditions, and any op code from 10 to 17 (octal). Since the SW instruction meets none of these conditions, S2E will be high and indirectly set E2C. During E2 the B address counter will be addressed,and the same subcommands will be generated as in E1. ACF requires only the op code and EEC; IN7 and WTN do not make any distinction between E1 and E2 either. The next cycle will be V3. E3C will be set by CT6•PEC•S3C. S3C will be set by P2C•$\overline{\text{DAS}}$•$\overline{\text{BCP}}$.

HALT INSTRUCTION

The halt instruction may take any one of three formats. All three forms will cause the machine to stop processing orders. The format of op code only will halt the machine and do nothing else. The format of op code and an A address will stop the machine and load the sequence counter with the A address. If the instruction had both an A and B address, the machine will stop after loading the A and B address counters, but it will not sequence change. A halt instruction only requires one execution cycle, M1. In this cycle the halt op code derived function, HLT, and DM1 will set CPBAC10. Immediately, BAC will set CPACL10 (allow clear). If the instruction format was op code only, this is all that happens in the halt order. The machine will not really stop until it's in the next cycle (V3). In V3, ACL will set CPSV310 at T03.

```
CPSV310  =  ACL•DVC•E3C•T03
         +  T02•SV3
         +  PER
         +  CCL
```

SV3 controls several functions. CPRUN10 has been recirculating on $\overline{\text{SV3}}$. With RUN low the normal path from V3 to A1 is blocked, because EAC will not set without RUN high if the last cycle was V3. The thing that really stops the machine is the fact that EAC won't be able to set. Therefore, the extraction cycles will never progress beyond extracting the op code. SV3 also sets CBS3D10 which will set E3C every time six as long as ITF is low. So, SV3 keeps the machine cycling in V3 and prevents the setting of the next cycle. SV3 will also set CPSTP10, if no peripheral devices are active (PNA). Until STP is set the machine will continue to process buffer cycles (the data transfers as a result of a peripheral order that was initiated before the halt order was performed).

As long as the machine is stopped ACL will stay high. SV3 will go low every time two, but ACL will set it again at T03. To start the machine again ACL must be reset. The go button will generate a series of functions that will kill the recirculation of ACL. Once ACL is reset SV3 will not be able to set at the next T03. The go button will also set RUN, indirectly, and RUN will allow EAC to set, and the normal flow of cycles will start again.

A parity error (CAPER10) at any time will set SV3 and bring the machine to a stop in the next cycle. PER also sets BAC to keep setting SV3 in the same manner as the halt instruction. So, the machine will stay stopped after PER is cleared, until the go button is pushed. The stop button will set ACL directly, with CPSTB1C.

If the instruction word had an op code and an A address only, the machine would stop in the same way as described earlier. In addition, a sequence change must be done. Every time the 200 sequence changes, the contents of the sequence counter are transferred to the B address counter before the sequence counter is loaded. All four control memory cycles (of M1) are needed to transfer location 17 to 10 and location 14 to 17. If a sequence change is to be done, two functions (CBPST10 and CBAFC10) will be set. PST will be used to ground IIT for the transfer of loc. 17 to 10 at times one and two. AFC will be used to ground IIT at times five and six to transfer loc. 14 to the sequence counter. To determine if a sequence change is specified, a function called CBP2P10 examines N07 for a word mark at the end of B1. The gate that senses this is: IIN• N07•PBC•HLT•P1C. The reasoning behind this gate is: while extracting a halt instruction, if the extraction cycle flow gets as far along as B1, there must have been a whole A address. Furthermore, if a word mark is read out in B1, there must not be a B address. This is the format that specifies a sequence change in addition to halting. P2P will set CBPST10 at T07, PST will set CBBII10, and BII will reset CBII110. At T01, because II1 is at ground, CBIIT10 will not be able to set. With IIT at ground, the control memory sense amps will recirculate and will not accept new inputs. Since loc. 17 will be addressed at T07 and loc. 10 addressed at T01, having IIT at ground will transfer the contents of the sequence counter to the B address counter. At CT1, CBAFC10 will be set by PST•DM1•CT1. Now AFC will be used to ground IIT at times five and six using auxiliary functions in the same manner as PST did to reset IIT at time one.

The preceding description of the sequence change operation shows that the halt operation is not affected in any way by the sequence change. A halt op code will always stop the machine. If, in the execution cycle of the halt order (M1), it is found that the last cycle was B1, then the machine will load the sequence counter with the A address before halting. It should be apparent that if the instruction had a B address, then the last cycle before M1 would have been a variant cycle, and no sequence change would occur.

NOP INSTRUCTION

The NOP instruction has one execution cycle. During this memory cycle (M1), nothing of any significance takes place. The total effect of a NOP instruction is that the sequence counter gets incremented until a word mark is found (indicating a new op code) during the extraction cycles. Once the word mark is found the next cycle will be M1, which allows the machine to get back to V3 via a normal route. Incidentally, the variant register is loaded in A1 if no word mark is found.

CHANGE SEQUENCE MODE

The execution cycles for the CSM instruction completely ignore the information read out of memory. The real work in this instruction is the manipulation of control memory information. Because the instruction must interchange two control memory locations, an intermediate storage area is needed. In M1, loc. 17 is temporarily stored in loc. 07, and loc. 04 (the co-sequence counter) is transferred to the sequence counter. Now the co-sequence counter has to be loaded with the old contents of the sequence counter. C1 is used to transfer loc. 07 to 04. It may look like C1 would not be necessary if the S register had been used as an intermediate storage area instead of loc. 07 during M1 (using the second and third C. M. cycles to transfer loc. 04 to 17), but by setting ITF an interrupt signal is simulated and the logic used to swap the sequence counter with the interrupt register can be used, with only slight modifications.

M1

Derived M1 cycle and CBCSM10 will set ITF at time six. ITF will set BII, and BII •E1C•CPC will reset II1. With II1 reset at T01 the control memory sense amps will recirculate after the sequence counter is read out. And when loc 07 is addressed, at T01, the strobe will be inhibited thereby loading the contents of the sequence counter into loc. 07 during the write half of the control memory cycle. At time five ITF will reset II5 with the help of CBPCP10 (P1C•CPC). This will recirculate the contents of loc. 04 in the C. M. sense amps while loc. 17 is addressed. There isn't a choice for the selection of the next cycle; it will always be C1, mainly due to ITF and a previous M cycle.

C1

This cycle has the sole purpose of transferring the old contents of the sequence counter, which were stored in loc. 07, into loc. 04. This is accomplished by using ITF to reset IIT at time three. The R register will be loaded with 07 at time one, and 04 at time three. At time four ECC will reset ITF.

CHANGE ADDRESSING MODE

The CAM instruction does not disturb either main memory or control memory information. Some locations are addressed, but only because it would have required additional logic to inhibit the normal cycling of memory. Any information read out is ignored. The only thing that is required is to set or reset CBCM210 depending on whether or not the variant register has a one bit in position five. If V05 is high CM2 will be set, otherwise reset CM2 and set CBCM200 .

$$\begin{array}{lcl}
\mathrm{CBCM210\ (I)} & = & \mathrm{DM1{\bullet}CAM{\bullet}\overline{V05}} \\
 & + & \mathrm{COS{\bullet}CAB{\bullet}AMS} \\
 & + & \mathrm{\overline{MCL}{\bullet}\overline{CM2}} \\
 & + & \mathrm{RRI}
\end{array}
\qquad
\begin{array}{lcl}
\mathrm{CBCM200\ (I)} & = & \mathrm{DM1{\bullet}CAM{\bullet}V05} \\
 & + & \mathrm{COS{\bullet}CAB{\bullet}\overline{AMS}} \\
 & + & \mathrm{CM2} \\
 & + & \mathrm{XR1{\bullet}CMF}
\end{array}$$

The top gate is used during CAM instructions. The second gate is used to change the addressing mode from the control panel. The third gate is recirculation, and the bottom gate is used during interrupt to change the addressing mode or to restore the original mode. An additional use of the CAM instruction is to set the "item mark trapping" indicator (IMT) when V03 is a one (or to reset IMT if $\overline{\mathrm{V03}}$). A description of item mark trapping is on page 9 of section 3.

RESUME NORMAL MODE

The RNM instruction does five things: (1) It turns off the interrupt indicator (CBITM10) (2) It interchanges the sequence counter and the interrupt register. (3) The arithmetic and comparison indicators are restored, (4) The addressing mode that was in use before the interrupt signal was received will be restored, (5) The A and B address counters will be loaded if a word mark does not follow the op code.

The RNM instruction takes two execution cycles, M1 and C1. In M1 the interrupt flop is set, (ITF). This function will be used to interchange the control memory locations; it is not an interrupt indicator. At T01 of M1, ITF will reset IIT to transfer the sequence counter to loc. 07. At T05 of M1, ITF will reset 11T again to transfer the interrupt register to the sequence counter. During M1, a function called "reset regular indicators" is set (CBRR110). This function will reset HR1, which has been recirculating the regular indicators (ALB, CS1, etc.). It should be noted that the regular indicators will be reset whether changing to or from the interrupt mode. However, it is only when returning to the normal mode that the indicators will be restored. The function that makes the difference is CBXR110. XR1 will come high as soon as ITM is reset. ITM is reset by: RNM•PCC•CT5. Therefore, at the tail end of C1, as ITM is reset, XR1 will come high and transfer the auxiliary indicators back into the regular indicators. Loc. 07 is transferred to loc. 06 during the second and third control memory cycles of C1. Finally, ITF is reset.

MOVE CHARACTERS TO WORD MARK

There are two execution cycles for a MCW instruction; they are repeated as many times as needed. E1 reads out the A character, and E3 writes it into the B location. Word marks are not altered in any way. The first word mark that is sensed, from either the A or B field, will terminate the instruction.

E1

In E1, as the A character is read out it is examined for punctuation. Timing-wise, the information in the location addressed in E1 is not used until CT6 of the next cycle, E3, but it will be discussed now to avoid confusing this information with the B character, which will be read out in E3. If there is a word mark in the A character, CBACF10 will set. ACF will set N07 in E3 to terminate the instruction.

$$\text{ACF} = \text{N07}\bullet\text{CT6}\bullet\text{ACG}$$
$$\text{ACG} = \text{DE1}\bullet\overline{\text{DAS}}\bullet\overline{\text{MPM}}$$

An item mark in the A character will set CBAFC10. AFC will set N08 in E3 to write an item mark in B.

$$\text{ACF} = \text{A2A}\bullet\text{N08}\bullet\text{CT6}$$
$$\text{A2A} = \text{MLA}\bullet\text{DE1}$$

CBMLA10 is an op code derived function that will be high for MCW and LCA instructions. At CT5, between E1 and E3, both the A and B registers are cleared. At times five and six N2A is high;hence, the A character will be loaded into the A register at CT6. Since theA register can only store six bits, ACF and AFC are used to store the punctuation bits.

Note that the A address counter is addressed in the first and fourth control memory cycles. At time one the contents of the A address counter are used to address main memory. At time seven, the decremented S register is written into the A address counter. The next cycle will unconditionally be E3.

E3

In E3 the B address counter will be used to address main memory. In this cycle, the data bits read out of memory will have no significance. INL will be set, this will inhibit reading memory into the N register. Instead, the contents of the accumulator are transferred to the N register. (A2N = DE3•CT3•EDT) at time three. The A character will be in the accumulator, because in the last cycle, as the A operand was put in the A register, the B register was cleared to zero's. Hence, when the accumulator adds the contents of the two registers the result will be the A character. Since INL is set, whatever is put in the N register will be written into memory.

The punctuation bits are handled individually (as always). If AFC was set (by an item mark) in E1 it will still be high in E3. When reading out memory in E3, CAIN810 will be set. This will inhibit the read out of the sense amp to the N register. IN8 will also let N08 feed the inhibit driver for plane eight instead of sense amp eightfeeding the inhibit driver N08 will be set by CAITN10 if the A character had an item mark, $ITN = DLA \bullet AFC \bullet \overline{IBP}$. CADLA10 will be high for MCW and LCA instructions. CPIBP00 is a signal from the control panel that will be normally high. If the A character did not have an item mark, then AFC would not have been set during E1. Consequently, even though IN8 will be high to allow an item mark to be inserted, ITN will not be set and N08 will be at ground, causing a zero to be written into plane eight. A more detailed description of punctuation bit handling may be found at the beginning of this section.

Bit seven of the B character will not be altered in any way, regardless of whether or not the A character had a word mark. This is guaranteed by not setting IN7. If there was a word mark in the A character, ACF would be set in E1. Now, ACF will set WTN and WTN will force N07 high ($WTN = DLA \bullet \overline{IBP} \bullet ACF$). Since IN7 is at ground, the normal read out of sense amp seven to N07 is not inhibited. Therefore, a word mark in <u>either</u> the A <u>or</u> B character will result in a one bit in N07 at the end of E3. This one bit will determine the next cycle. If N07 is high V3 will be the next cycle:

$$\begin{aligned} E3C &= CT6 \bullet S3B \bullet \overline{PST} \bullet N07 \\ S3B &= \overline{P1C} \bullet PEC \bullet TEC \\ TEC &= \overline{TUB} \bullet P3C \bullet \overline{MUV} \bullet \overline{MAT} \end{aligned}$$

If N07 is low, EEC•E1C will be set, and the machine will cycle from E1 to E3 to E1 etc. until a word mark in either A or B sets N07.

<u>LCA INSTRUCTION</u>

The LCA instruction is very similar to the MCW. The data and item marks at A are transferred to B. The difference is that the LCA instruction must erase B word marks and continue transferring data until an <u>A</u> word mark is sensed. This difference may be implemented very simply. By setting IN7 in every E3 cycle bit seven from the B field will not be gated into N07 (this means B word marks cannot be sensed), and a one bit will not be written into plane seven unless ACF senses an A word mark and sets WTN and N07. The difference between MCW and LCA is one gate on IN7 that will be high during LCA instructions but not during MCW's ($IN7 = DLA \bullet I01 \bullet T02$). All other functions needed for the LCA will be high for MCW's also.

<u>EXTENDED MOVE</u>

The EXM instruction reads out an A character in E1 and writes it into the B field in E3. The distinguishing feature of the EXM instruction is its ability to move data or punctuation bits independently of each other. The configuration of the variant character determines what will be transferred, and what will terminate the instruction. V01 transfers data bits, V02 transfers word marks, and V03 transfers item marks. V04 indicates a left to right transfer. Combinations of V06 and V05 determine the punctuation that will terminate the instruction.

<u>E1</u>

In E1 the A character is read out of memory and transferred to the A register while the B register is cleared to zeros. When A and B are added, the result in the accumulator will be the A character. After the A address counter is used to address main memory it will either be incremented or decremented depending on V04, which will be set if the transfer is left to right. CB1SR10 (increment S) will be set by CB1SA10 if V04 is a one bit ($ISA = MUV \bullet V04 \bullet EEC$). If the A character had a word mark, ACF will be set at the next time six.

$$\begin{aligned} \text{ACF} &= \text{ACG}\bullet\text{N07}\bullet\text{CT6} \\ \text{ACG} &= \text{DE1}\bullet\overline{\text{MPM}}\bullet\overline{\text{DAS}} \end{aligned}$$

If the A character had an item mark, AFC will be set at time six.

$$\begin{aligned} \text{AFC} &= \text{A2A}\bullet\text{N08}\bullet\text{CT6} \\ \text{A2A} &= \text{MUV}\bullet\text{DE1} \end{aligned}$$

By time seven of the next cycle, the data bits will be in the accumulator, the word mark (or lack of it) will be stored in ACF, any item mark will be stored in AFC, and the contents of the A address counter will have been incremented or decremented, depending on V04.

E3

In E3, data will be transferred to the N register from the accumulator ($\text{A2N} = \text{DE3}\bullet\text{CT3}\bullet\overline{\text{EDT}}$). If V01 if high, the N register will be written into memory.

$$\begin{aligned} \text{INL} &= \text{I2L}\bullet\text{DE3}\bullet\text{CT2} \\ \text{I2L} &= \text{MUV}\bullet\text{V01} \end{aligned}$$

Also in E3, ACF will be sent to N07, and AFC sent to N08

$$\begin{aligned} \text{WTN} &= \text{DE3}\bullet\text{MUV}\bullet\text{ACF} \\ \text{ITN} &= \text{DE3}\bullet\text{MUV}\bullet\text{AFC} \end{aligned}$$

If V02 is high, N07 will be written into memory, and if V03 is high, N08 will be written into plane eight.

$$\begin{aligned} \text{IN7} &= \text{DE3}\bullet\text{MUV}\bullet\text{V02}\bullet\text{CT2} \\ \text{IN8} &= \text{DE3}\bullet\text{MUV}\bullet\text{V03}\bullet\text{CT2} \end{aligned}$$

Note that while the handling of data, word marks, and item marks is similar, each is completely independent of the other. Variant bits five and six are also free of influence from the other four bits. However, V05 and V06 are used together to examine the punctuation bits to terminate the instruction. At the end of E3, the cycle selection function CBS3F10 will select V3 as the next cycle if the terminating conditions called for in the variant are present (S3F can only set E3C if PEC•CT6•DE3 are high)

$$\begin{aligned} \text{CBS3F10} &= \overline{\text{V06}}\bullet\overline{\text{V05}} \\ &+ \text{ACF}\bullet\text{AFC} \\ &+ \overline{\text{V05}}\bullet\text{AFC} \\ &+ \overline{\text{V06}}\bullet\text{ACF} \end{aligned}$$

Due to the top gate on S3F if V05 and V06 are zero's, only one character will be transferred. The second gate on S3F will be high if a record mark is sensed in the A character, and the instruction will terminate regardless of which one (or both) of the punctuation bits was specified in the variant. The third gate on S3F assumes that if the programmer didn't want to stop on a word mark, he must want to terminate on an item mark. The bottom gate makes a similar assumption concerning $\overline{\text{V06}}$. Simplified, it would be the same as this: $\text{S3F} = (\overline{\text{V06}}\bullet\overline{\text{V05}}) + (\text{V05}\bullet\text{ACF}) + (\text{V06}\bullet\text{AFC})$. Of course, the opposite conditions should appear on the gates of EED•E1C in order to transfer another character if the specified punctuation is not present. The B address counter will be incremented or decremented by the same logic as the A address counter.

MOVE AND TRANSLATE

The MAT instruction uses the character found at the A address to select a character from a table in memory and places the selected character in the B address. The two variant characters are used to form the base address of the table. If you are not clear how this is done refer to the "Programmers' Reference Manual".

M1

The first execution cycle is M1: all it really does is load the upper three bits of control memory working loc. 07 with the three low order bits of the first variant character and read the second variant character out of memory. This is the only time than an execution cycle is used to extract part of the instruction.

E1

In E1 the A character is read out of memory and transferred to the V register by way of the B register. Meanwhile, the second variant character, which was read out of memory in M1, is transferred to the Z register and is written into the middle of control memory working location 07. Now, the high order nine bits of location 07 contain the base address of the translation table. The character to be translated (in the V register) will constitute the low order six bits of the complete main memory address. Note that the A address counter is decremented in anticipation of another character to be translated in case the instruction is not terminated on the first pass.

C1

Working location 07 is addressed, and ICM will load the high order nine bits of the S register. The low order six bits of the S register will not be loaded from the control memory sense amps because ICL is used by ECC•E1C•MAT. Instead, CBV2S10 will deliver the V register to S (V2S = ECC•MAT). The address in the S register will be the address of the translated equivalent of the A character (provided the programmer set up the table correctly and specified the correct base address in the variants). The character read out of memory is the translated character. It will be stored in the accumulator by N2B and CLA.

E3

In E3 the B address counter addresses main memory, and A2N in conjunction with INL will deliver the translated character to the B address. In addition to transferring data from the table, if an item mark is found when reading the character from the table in C1, AFC will be set and will set ITN and N08 in E3. The MAT instruction sets IN8 in E3 to allow N08 to be written into memory. Bit seven in the B field cannot be altered in any way since IN7 cannot come high during the MAT instruction. If ACF is high at the end of E3 no more characters will be translated. ACF will be set by a word mark from either the character to be translated or from the table.

$$\begin{aligned} \mathrm{ACG} &= \mathrm{DE1} \bullet \overline{\mathrm{DAS}} \bullet \overline{\mathrm{MPM}} \\ &+ \mathrm{PCC} \bullet \mathrm{MAT} \end{aligned}$$

If ACF is not high, the next cycle will be E1, and the next character to be translated will be sent to V. The machine will repeat the cycle flow of E1 to C1 to E3 to E1 until a word mark is sensed.

STORE CONTROL REGISTER

The SCR instruction will store the contents of the control memory location specified by the variant in three consecutive locations. The A address designates the location that is to receive the low order six bits of the control memory location, A minus one will receive the next six bits, and A minus two the last three bits.

E1

The variant is used to address control memory at time three. Subcommand KYL will be high, and at time five KYL will gate the low order six control memory sense amps into the Y register. At times five and six Y2B will be high and gate the Y register to the B register. The A address counter is decremented by one and stored in working location 16.

W2

The A address counter is used to address main memory at time one. The low order six bits of the variant specified control register are delivered from the accumulator to the N register for storage in the A address. At time three the variant specified control register is addressed,and sense amps 7 thru 12 are gated to the Y register by KYM at time five. From there they are gated to the B register and will end up in the accumulator (formerly occupied by the low order six bits).

W3

This cycle is identical to the last with three exceptions: (1) KYU is generated instead of KYM. (2) Writing from the Z register is not inhibited at time seven. (3) Working location 16 is used to address main memory instead of 14. The reason for the first difference is obvious. The other differences are due to one, not so obvious, reason. Way back in E1 the decremented contents of the A address counter were written into loc 16. Therefore, in W2 location 14 contained the correct main memory address. Also in W2, the decremented A address was already in location 16 and did not need to be written. In W3 location 16 is used to store AC-2.

W1

Location 16 is used to address main memory,and location 14 receives the PREVIOUS A address from location 07. The high order portion of the variant specified control register is delivered to the N register but is written into memory only if the machine is in three character mode.

$$CBINA10 = E1C \cdot \overline{CM2} \cdot SCR$$

LOAD CONTROL REGISTER

The control register specified by the variant is loaded with three characters of data from A, A-1, and A-2 address.

E1

In E1 the A address counter addresses main memory and is decremented. The first A character is read out of memory and stored in the accumulator.

W2

In W2 location 14 is again used to address main memory,but it is written into location 16 after it is decremented this time. At time one the variant register is used to address control memory,and the first A character is written from the low order six bit of the Z register. The second A character is stored in the accumulator.

W3

In this cycle the second A character is written into the specified control memory location, and the third character is read out of memory. Location 16 is decremented after use.

W1

In this cycle the third character is written into the high order portion of the control register,and the A address counter is restored to its previous contents by an internal transfer from location 07. Location 07 always receives the A address counter during V3.

Interrupt Operation

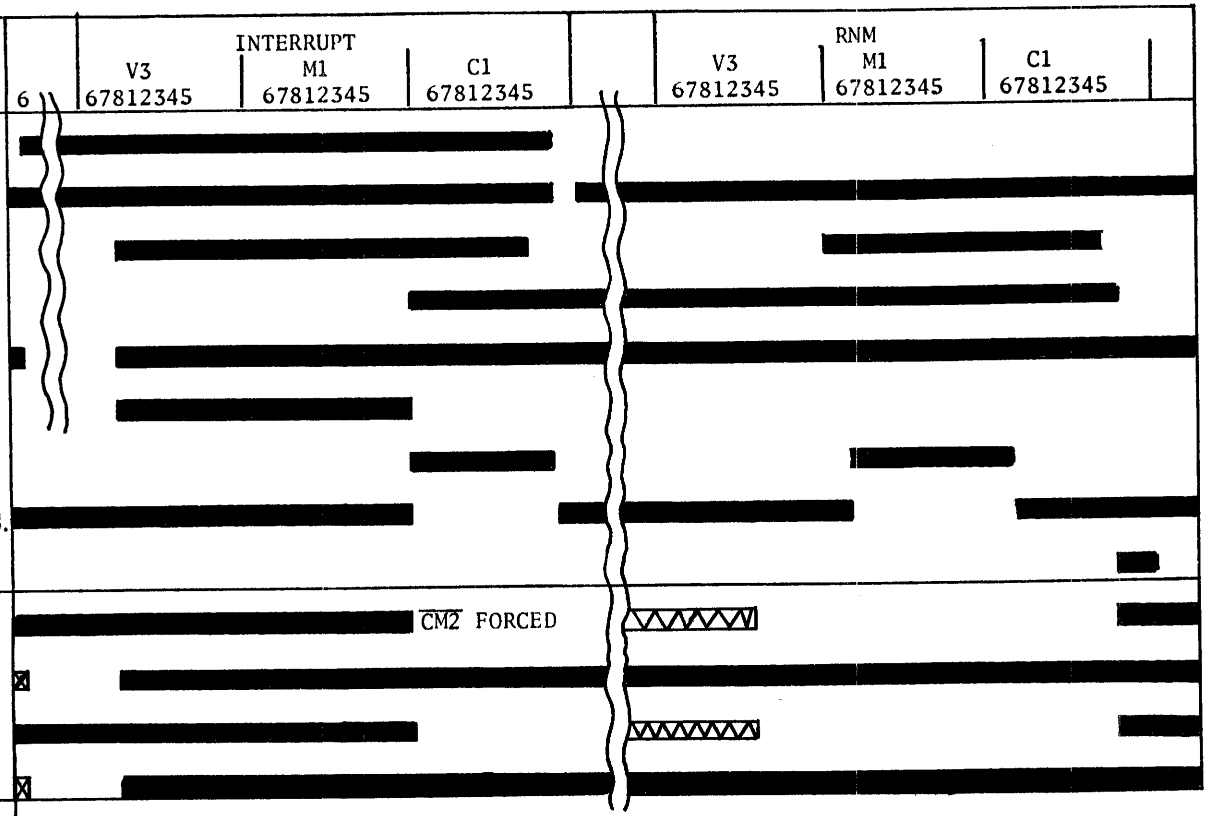
The interrupt feature makes it possible for a running program to be temporarily interrupted so that a predetermined subroutine may be executed, then the normal sequencing of instructions may be resumed. One I/0 control unit capable of generating an interrupt signal is the data communication device. This control unit is usually used to accept data over a phone line whenever something at the other end of the line wants to insert some data into the 200. A typical interrupt subroutine might interrogate this data and send a response (through the data communication device) then return control of the machine to the original program at the point where it left off.

The suspension of the normal sequence of instructions isn't too difficult. This is done by using the interrupt flop (ITF) to inhibit the normal path of extraction cycles from V3 → A1 and to inhibit loading the I register. Then ITF is used like an op code derived function to cause the machine to do two execution cycles, M1 and C1. The primary purpose of these two cycles is to swap the contents of two control memory locations 06_8 and 17_8. The net result of this is to cause subsequent instructions to be obtained from consecutive locations, starting with the location that was in the interrupt register prior to receiving the interrupt signal.

In addition, during the execution cycles, the arithmetic and compare indicators must be stored in auxiliary flops so that they may be restored when the normal mode is resumed. CBXAI10 is the function that loads the auxiliary indicators. CBXAI10 = $\text{INT}\bullet\text{ITF}\bullet\overline{\text{ITM}}$ (see fig. 9-2). The actual interrupt signal is FIT. CAFIT10 sets INT provided the machine is not already in the interrupt mode. CAINT10 = FIT•ITM•T07. If INT is high in V3, then ITF will be set(CBITF10 = DVC•E3C•INT•CT1• ITM). An Ml cycle is forced by P3C•ITF. In M1 the sequence counter is stored in loc. 07 and the interrupt register is loaded into the seq. counter. Refer to RNM in flow charts since the action is similiar. A C1 cycle is forced by ITF•PMC (previous M cycle). In C1, loc. 07 is loaded into loc. 06 (the interrupt register). As soon as the machine enters C1 the interrupt mode indicator is set (ITM) and this sets RRI which resets the regular indicators. RRI also forces off CM2 and IMT (item mark trapping indicator). The interrupt routine will now be able to make full use of all regular indicators.

INTERRUPT OPERATION

FUNCTION	DEFINITION	PAGE
INT	INTERRUPT REQ	260
HNT	HOLD INT	260
ITF	INTERRUPT FLOP	260
ITM	INTERRUPT MODE	260
HAI	HOLD AUX IND	163
XAI	XFER TO AUX IND	163
RRI	RESET REG IND	163
HRI	HOLD REG IND	YPGS.
XRI	XFER TO REG IND	163
CM2	2 CHAR MODE	261
CMF	CM2 STORAGE	260
IND*	INDICATOR	
IDS**	INDICATOR STOR.	



IND*	PAGE	IDS**	PAGE
AEB	162	AEC	163
ALB	162	ALC	163
CSI	162	CSJ	163
ZBI	160	ZBJ	163

Fig. 9-2

SECTION X
EDIT (74)

Standard format: F/A/B

Extraction path: Chains the B address

Synopsis: The data-contents of the A field are edited into the B field under control of the data-contents of the B field. One or more passes are required, depending upon the B field contents. The direction of successive passes alternate, starting with a right-to-left pass. Transfer from the A field takes place during the first pass only.

Details:

a. First pass (right to left) - The sign of (A) is tested and remembered. Successive characters in the B field are examined.

(1) Any location in the B field that contains octal 37 has its contents replaced by octal 15 (blank).

(2) If the sign of (A) is positive, then until a zero or an octal 15 is encountered any location in the B field that contains C, R, C_R, or - (i.e., octal 23, 51, 75, or 40) has its contents replaced by octal 15 (blank); this function (2) is suspended as soon as a zero or octal 15 is found in the B field, but it is resumed after transfer of the high-order character of the A field.

(3) A B-field location is "available" for transfer of a character, provided that it contains zero or octal 15, or that it is immediately to the left of the first (rightmost) zero encountered in the B field and contains * (octal 54) or $(octal 53). Each time an available B-field location is found, a character is transferred to that location from the A field; the first character transferred is (A), but with its two sign bits replaced in the result location by 00; subsequent transfers are of the unmodified six-bit characters, and successive transfers are from successive locations (right to left) in the A field.

(4) Transfer of data from the A field terminates with the transfer of the high-order character of the A field; after this the scan of the B field continues (octal 37 always being replaced by blank, and conditional replacement of C, R, C_R, is resumed), and now any commas (octal 73) found in the B field are replaced by blanks. This pass terminates when the high-order location of the B field has been examined and appropriately treated, and its field mark is cleared.

If during the first pass a zero has been found in the B field, a call for zero suppression is registered and the location of the first (rightmost) zero found is marked. The next location to the left is inspected to determine whether it contains * or $; in the former case, a call for *-protection is registered, and in the latter a call for floating $ is performed.

If no zero was found in the B field, the end of the first pass is the end of the operation.

b. Second pass (left to right) - Zero suppression: Successive characters (left to right) are examined, starting with the high-order location of the B field. When zero suppression is on, all zeros, commas, and blanks are replaced either by blank (if no call for *-protection has been registered) or by * (if a call for *-protection has been registered). When a location in the B field is found to contain a dot (octal 33) (decimal point) or a numeric ≠ 0, zero suppression is turned off; but the scan continues. This pass terminates when the marked location of the right-most zero originally in the B field has been scanned and processed; the mark previously set in this location is cleared.

If no call for floating $ has been registered, the end of the second pass is the end of the operation.

c. Third pass (right to left) - Floating the dollar sign: After the processing of the terminating location of zero-suppression, this location and successive locations to the left are examined until one is found that contains a blank. The character $ is stored in this location, and the operation terminates.

During the edit instruction, the A character is transferred to the A register, and the B character is transferred to the V register via the B register which is cleared after the ricochet. The B character is then investigated to see if it is allowable to transfer the A character into it or to replace the B character with a blank. If neither of these two events are allowed, then the B character remains the same, and the next one is investigated.

To help in the decoding of the B character that is in the variant register, there are a number of decoder functions` which should be investigated.

			H	L
CEDDA10 (I)	=	$V06+V05+V04$	0	x
CEDDB1X	=	$\overline{V01}\bullet\overline{V02}\bullet\overline{V03}$	x	0
CEZR010	=	$DDA\bullet DDB$	0	0
CEDDC10 (I)	=	$\overline{V06}+V05+\overline{V04}$	5	x
CEDDD10 (I)	=	$V03+\overline{V02}+\overline{V01}$	x	3
CEDDE10 (I)	=	$\overline{V03}+V02+V01$	x	4
CEDDH10 (I)	=	$V06+V05+\overline{V04}$	1	x
CEDDM10 (I)	=	$\overline{V03}+V02+\overline{V01}$	x	5
CEBLK10(I)(I)	=	$DDH\bullet DDM$	1	5
CEDDJ10	=	$\overline{V06}\bullet V05\bullet V04$	3	x
CEDDK10 (I)	=	$V03+V02$	x	0 + 1

Those functions plus other positions of the variant register can be combined to give a certain character configuration. ZRO and BLK show how two of the decoder functions can be combined, and one of the gates of CEARB00 will show the other combination.

CEARB00 (I) = $V01\bullet V02\bullet V03\bullet DDJ$ = 37 = Non Replaceable Blank (b̸)

The conditions necessary to transfer an A character are found on the inputs to one function, CETAF10-Transfer from A Field. If the B character is zero (00) or blank (15), the transfer is unconditional. If the character to the left of the first rightmost zero is an asterisk (54) or a dollar sign (53), an A character may also be transferred into it. The transfer of the A characters continues until the A field is terminated. After the A field is terminated, the B character is only investigated for replacement by blanks.

CETAF10	=	DDA•DDB•$\overline{\text{DKL}}$•$\overline{\text{AFT}}$	(00)
	+	DDH•DDM•$\overline{\text{DKL}}$•$\overline{\text{AFT}}$	(15)
	+	DDC•DDD•CAZ•$\overline{\text{DKL}}$•$\overline{\text{AFT}}$	(53)
	+	DDC•DDE•CAZ•$\overline{\text{DKL}}$•$\overline{\text{AFT}}$	(54)
	+	TAF•$\overline{\text{CT7}}$	
CECAZ00 (I)	=	ZRO•ZFL•DJ3•CT5	(1st zero sets CAZ)
	+	CAZ•$\overline{\text{PVC}}$	
CECAZ10 (I)	=	$\overline{\text{ZFL}}$•DE3•CT4	
	+	$\overline{\text{CAZ}}$	
CEZFL00 (I)	=	ZFL+PVC	(zero first left is set at beginning of order;
CEZFL10 (I)	=	ZRO•DE2•PPF•CT2	reset by first zero).
	+	$\overline{\text{ZFL}}$	

The conditions to replace the B character by a blank are found on the inputs to two functions, which are fed into the common function CERBB10 - Replace by Blank. Four of the inputs not only require the B character to meet a certain condition, but also that the A operand be positive, and that a zero not to have had appeared in the B operand as yet. This is controlled by CEPNZ10. Two inputs are used to replace an asterisk or dollar sign if the A field has already been terminated. One gate replaces commas if the A field has been terminated, and the last gate will unconditionally replace octal 37, which is the decoded configuration for a non replaceable blank (b̸).

CEARB00 (I)	=	V06•$\overline{\text{V05}}$•$\overline{\text{V04}}$•DDB•PNZ	(40 = $\overline{0}$)
	+	V01•V02•V03•DDJ	(37 = non replaceable blank b̸)
	+	DDZ•DDD•AFT	(53 = $)
CEBRB00 (I)	=	$\overline{\text{V06}}$•V05•$\overline{\text{V04}}$•DDD•PNZ	(23 = C)
	+	$\overline{\text{V03}}$•$\overline{\text{V02}}$•V01•DDC•PNZ	(51 = R)
	+	V03•$\overline{\text{V02}}$•V01•VE1•PNZ	(75 = CR)
	+	VE1•DDD•AFT•$\overline{\text{PN1}}$	(73 = ,)
	+	DDZ•DDE•AFT	(54 = *)
CERBB10 (I)	=	$\overline{\text{ARB}}$•$\overline{\text{BRB}}$	
CEPNZ10	=	AFP•$\overline{\text{ZFT}}$•$\overline{\text{DKL}}$	
CEZFT00 (I)	=	DDH•DDM•CT8•PPF•E2C	(15)
	+	DDA•DDB•CT8•PPF•E2C	(00)
	+	$\overline{\text{AFT}}$•ZFT•EDE	(Set by first zero or blank. Reset by
CEZFT10 (I)	=	$\overline{\text{ZFT}}$	AFM. Can be set again by another zero or blank.

If the A operand is positive, then the characters that can be replaced include C, R, and CR. A $\overline{0}$ (octal 40) will also be replaced by a blank. If the A operand was negative, the characters C, R, and CR will remain the same, and the octal 40 will be printed as a negative sign.

As the character in B is being investigated on the first pass, there are three things which can happen. An A character can be transferred into it, or it can be replaced by a blank. Otherwise, it is written back into memory unchanged.

The first character of the A operand will decide if the operand is negative or positive. Its sign is gated into CEAFP10 (A field plus) which recirculates until the next instruction, if it is set. With the sign stored in this manner, the sign bits are stripped from the first A character through the action of the accumulator. Before the first transfer actually takes place, the two high order positions of the accumulator are low. The fifth position (CASD510 an inverter) is forced low, and the sixth position (CASU610 an amplifier) cannot be set through the action of CASGH00.

CASD510 (I) = EDE•$\overline{\text{TFT}}$
CASGH00 (I) = EDT•$\overline{\text{TFT}}$•$\overline{\text{HFC}}$
CETFT00 (I) = TAF•DJ3•CT6
+ TFT•$\overline{\text{HFC}}$

Before anything else is discussed,let's do an example using only what has been learned so far.

A = 3 4 $\overset{+}{3}$; B = b b . b b C_R

The plus three is stored in the A register, and the C_R is stored in the V register. The plus will cause PNZ to be set. If TAF is investigated, no input gate will be satisfied. If RBB is investigated, the third gate on BRB will be satisfied, and C_R will be replaced by a blank. The actual replacement will be covered later. No A character is transferred, and it remains in the A register. Another B character is investigated, and this time a blank is found. Now this is not the blank from the C_R symbol. It is the blank in the next position. TAF is satisfied, and the 3 is transferred into the position with the zone bits stripped off through the method discussed previously.

The 4 is stored in the A register, and the next B location is investigated. The conditions are again favorable for transfer.

The three with the word mark is sensed. The three is stored in the A register, and the work mark sets CEAFM10.

CEAFM00 (I) = N07•CT6•DE1
+ AFM•CLI
+ DK3•CT2
CEAFM10 (I) = DE3•BFM•CT6
+ $\overline{\text{AFM}}$

Since there is an A field mark, A Field Terminated should become true. But before this is finally settled, let's investigate the B character - a period. A period does not satisfy either TAF or RBB; therefore, the period remains unchanged. Now if AFT becomes set, TAF could not ever be satisfied again, and there is still a character in the A register. Taking a look at AFT shows that it cannot be set until the A character is transferred.

CEAFT10 = DE3•AFM•CT4•TAF
+ AFT•CLI
+ $\overline{\text{DKL}}$•BFM•CT4

The next B character is investigated, a blank is found, and the A character is transferred into it. Since the A field is terminated, only the B character is investigated. It will either remain unchanged or become a blank. The next B character is a blank, and it contains a word mark. A blank is returned, and for this example, the instruction ends. The reason it ends will be discussed later. The result of the example is: B = b3.43b or 3.43

Now that a simple example has been covered, lets see how the B character is replaced by a blank. If the character is replaceable, then the function CERBB10 will be high. In some way, RBB must affect both the transfer of data from the sense amps to the N register, and the transfer or formation of the blank configuration in the N register.

The transfer of sense amps to N register is handled by CAINL10. If INL becomes high, we inhibit the transfer from the sense amps and automatically write back into memory from the N register.

CAINL10 = G2N•DE3•CT2•EDE
CEG2N10 = RBB•DJ3

So, this shows how we inhibit the sense amps; now the formation of the blank configuration must be shown.

There is a four position G register, which consists of positions 1, 2, 3, and 6. Upon getting the subcommand, CEG2N10, the G register is transferred to corresponding positions of the N register. The four G functions are outputs of inverters; therefore, their output is normally high unless we drive it low with a high input. If we look at the 2nd and 6th position, we see that these can be driven low.

CEG0210 (I) = RBB•DE3 ; CEG0610 (I) = RBB•DE3

The 1st and 3rd positions cannot be touched, therefore they remain high. This now gives a configuration of 0--101. Nothing has yet been said of position four and five. For these two, the N register must be investigated. On the 4th position if we get G2N, it is unconditionally high. G2N is no where to be found on position 5; therefore, it must be unconditionally low. The other 4 positions which have G2N on them all require a position of the G register. As a result of all this, the N register ends up with 001101 - a blank.

It was also shown that an A character could be transferred into a B character position through TAF. To do this, TAF must also affect the transfer of data from the sense amps to the N register, and then cause something to be transferred to the N register. If CAINL10 becomes high, we inhibit transfer of SA to N as before, and then write into memory from the N register.

CAINL10 = I2L•DE3•CT2
CAI2L10 = TAF•EDE

This proves that the sense amps are inhibited; now how is the A character transferred. In E1 cycle, the A character is transferred to and stored in the A register. In E2J cycle, the B character is ricocheted off the B register into the V register, and the B register is then cleared. The contents of the A and B registers are constantly being added in the accumulator; therefore, the A character is in the accumulator. If the B character allows an A character transfer, TAF is high and this results in the accumulator being transffered into the N register.

CAA2N10 = DJ3•TAF•EDE•CT3

In general, for TAF the read out of the sense amps is inhibited, and the A character is transferred into the N register from the accumulator.

Only one part of the instruction has been discussed. It can do much more. What about suppressing leading zeros and floating dollar signs or inserting asterisks. It is possible to suppress leading zeros only; but to float dollar signs or insert asterisks leading zeros must also be suppressed. Exactly how is this all accomplished? First of all, it is the first zero that does the controlling. Upon sensing this zero, CEJDS10 (Justify Dollar Sign) and CEAPT10 (Asterisk Protection) are set. If the next character to the left is an asterisk, this means that insertion of the asterisk sign is wanted;so, the asterisk configuration automatically resets the dollar sign function. If the next character had been a dollar sign, then the reverse would have been true.

$$\begin{array}{lcll} \text{CEJDS00 (I)} & = & \text{ZRO}\bullet\text{ZFL}\bullet\text{DJ3}.\text{CT2} & (00) \\ & + & \text{JDS}\bullet\overline{\text{PVC}} & \\ \text{CEJDS10 (I)} & = & \text{DJ3}\bullet\text{DDZ}\bullet\text{DDE}\bullet\text{CT2} & (54) \\ & + & \overline{\text{JDS}} & \\ \text{CEAPT00 (I)} & = & \text{ZRO}\bullet\text{ZFL}\bullet\text{DJ3}\bullet\text{CT2} & (00) \\ & + & \text{APT}\bullet\overline{\text{PVC}} & \\ \text{CEAPT10 (I)} & = & \text{DDZ}\bullet\text{DDD}\bullet\text{DJ3}\bullet\text{CT2} & (53) \\ & + & \overline{\text{APT}} & \end{array}$$

Let's extend the example used before, so that our new knowledge can be put to use.

A = ⓑ0 0 0 1 3 4 $\overset{+}{3}$; B = ⓑb b b b $\$$ 0 b . b b b̸ c_r

The plus three is stored in the A register as before, and the C_R is found to be replaceable by a blank. The three was not transferred. The b̸ (37_8) is brought out, and this is always replaceable by a blank. The three is finally transferred to the first blank actually found in B. Remember that the zone bits of the first transfer are 00. The next two A characters are transferred as before. Now as the one is stored in the A register, the zero is sent to the variant register, and is recognized as a zero. JDS and APT are set, and the one is transferred into the zero position. The first zero from A is stored in the A register. The next character of B is sent to the variant register and is recognized as a dollar sign. This causes APT to be reset, and since this is the character after zero, CECAZ10 will be high which allows the dollar sign to be transferred into. The next two zeros are transferred into the blanks. At this point, there is one character remaining in A, and B =ⓑb b 0 0 0 1 3.4 3 b b. As the next character in A is read out, a word mark is detected, which gives A field terminated.

$$\begin{array}{lcl} \text{CEAFT10} & = & \text{AFM}\bullet\text{DE3}\bullet\text{CT4}\bullet\text{TAF} \\ & + & \text{AFT}\bullet\text{CLI} \\ & + & \overline{\text{DKL}}\bullet\text{BFM}\bullet\text{CT4} \end{array}$$

Notice from the equation, that AFT cannot become true unless an A character is actually able to be transferred.

$$\begin{array}{lcl} \text{CEAFM00 (I)} & = & \text{N07}\bullet\text{CT6}\bullet\text{DE1} \\ & + & \text{AFM}\bullet\text{CLI} \\ & + & \text{DK3}\bullet\text{CT2} \\ \text{CEAFM10 (I)} & = & \text{DE3}\bullet\text{BFM}\bullet\text{CT6} \\ & + & \overline{\text{AFM}} \end{array}$$

After this A character is transferred, only the B character is investigated to see if it is replaceable by a blank. In this example, the only characters allowable are (37_8) or commas. Since the A operand was positive and the A field is terminated, the replacement of the characters controlled by PNZ is again resumed.

The B operand is investigated and satisfied until the B word mark is sensed. Then at this point, a check is made to see if the instruction terminates or not. In other words, is zero suppression wanted? If it is, then the negation of either JDS or APT or both will be low, which allows CEJAP10 to be high. In the first example, there was no zero suppression; therefore, JAP was low. Also, the negation of both JDS and APT were high.

$$\text{CES3G10} \quad = \quad \overline{\text{JDS}}\bullet\overline{\text{APT}}\bullet\text{BFM}$$

This allows another E3C and stops EEC; therefore, it must be DV3, so the next instruction is now in progress. In the second example, the instruction will not be terminated at this point.

CES2H10 = BFM•$\overline{\text{DKL}}$•JAP

Since S3G was not satisfied, there is still existing E cycle. Also, since zero suppression is desired, some type of pass counter should be set, for now a pass from left to right must be made to suppress leading zeros. If the character after the zero had been an asterisk, asterisks would be inserted on this pass, but in this example, leading zeros will be replaced with blanks until either the first decimal digit (the one) is reached, or the character position where the control zero was sensed is found.

If the digit comes first, the pass will still continue until the control zero position is sensed. How is it known when this position is actually sensed? As the control zero was sensed, a word mark was forced into that position. On this second pass, this word mark is looked for, and when found, the third pass is started to float the dollar sign.

CAWTN10	=	ZRO•DE3•EDE•ZFL	Forces word mark
CAIN710	=	CEIN710•EDE•CT2	into the control
CEIN710	=	DJ3•ZFL•ZRO	zero position.

As the second pass continues, this forced word mark will be sensed, and the third pass will be started in the opposite direction.

CEPN210 (I) (I)	=	DE3•BFM•PN1•CT6•DK3
CEDR310 (I) (I)	=	DE3•$\underline{\text{PN1}}$•$\overline{\text{PN2}}$•CT8
	+	DK3•$\overline{\text{CT7}}$

Before the third pass is started, the second pass and its zero suppression had better be covered. As it is started, B = b b b b 0 0 0$\textcircled{1}$3 . 4 3 b b. The first blank is brought out and causes CEZSP10 (zero suppression) to be set. ZSP would also set for zero or a comma.

CEZSP10	=	DDA•DDB	(00-zero)
	+	VE1•DDD	(73-comma)
	+	DDH•DDM	(15-blank)

Zero suppression along with CETZS10 (which is high unless you have a period or decimal digit) is used to give CETZP10. This now allows CEB2G10.

CEB2G10 = JDS•TZP•DK3

B2G forces the 2nd and 6th positions of the G register low, and CEG2N10 high. The action of the G register and its subcommand have already been explained. This time an octal 15 or blank is forced into the N register and written into memory. This will continue until the word mark position is reached or a decimal digit or period is reached. Once the digit or period is reached, TZS is forced low and remains low for the rest of the order.

CETZS10 (I)	=	DDJ•DDD	(33-period)
	+	DDH•DDK	(10+11 = 8+9)
	+	DDA•V01	(1 through 7)
	+	DDA•V02	
	+	DDA•V03	

At the end of the second pass, B = b b b b b b b 1 3 . 4 3 b b

As the third pass starts, the "1" will again be brought out and investigated. After it is seen that it is not the desired character, the next one is investigated. A blank sets CEBLK10. This along with CEDL310 will drive the 3rd position of the G register low, and set CEC2N10. The result of which is an octal 53 or dollar sign in the N register. G2N caused the N register to be delivered to memory. At this point B = b b b b b b $ 1 3 . 4 3 b b, which on the printer would look like $13. 43. The instruction should be ending now, so there should be some way of proving this. CEBLK10 once more comes into play; it forces CES3G10, which says we want another three cycle. Once S3G is set, existing F Cycle is no longer true; therefore, the next cycle must be PV3.

SECTION 11
INPUT/OUTPUT

INTRODUCTION

Input/Output operations are controlled by two general purpose instructions: Peripheral Data Transfer (PDT) and Peripheral Control and Branch (PCB). These instructions may be used with any 200 peripheral device; however, only the general use of the instructions will be discussed, with reference made to specific control units.

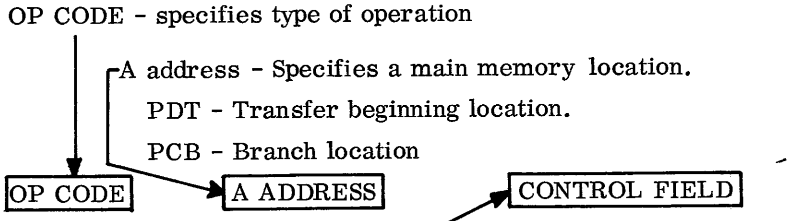

OP CODE - specifies type of operation
A address - Specifies a main memory location.
PDT - Transfer beginning location.
PCB - Branch location
OP CODE
A ADDRESS
CONTROL FIELD

Figure 11•1

CONTROL FIELD - Contains one or more input/output control characters which specify such parameters as data path, direction of data flow, control unit designation, test conditions, etc.

Peripheral Data Transfer - General Description

The Peripheral Data Transfer instruction, by the use of specific bit configurations in each control character, can initiate the transfer of data between the main memory and any of the peripheral devices in the H-200 system.

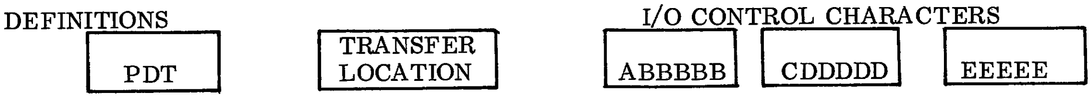


Figure 11•2

Interlock Bit (A)

The status of this bit specifies whether or not the auxiliary read/write channel will be granted access to main memory by the traffic control. If the bit is a zero, then both channels will be granted access. If the bit is a one, then only RWC1 is granted access. This bit is effective only when RWC1 is specified to transfer the data. Naturally RWC1 cannot be interlocked while the auxiliary is busy.

Read/Write Channel Designation (B)

These bits specify the read/write channel selected to complete the data path between the main memory and the peripheral device. Read/write channels 1, 2, and 3 are specified as 01001, 01010, and 01011 respectively. RWC1' is specified as 01101.

Direction Indicator Bit (C)

The status of this bit indicates whether the transfer is an input operation (1) or an output operation (0).

These are established individually at each 200 installation. Whenever possible, they should be as follows:

Control Unit	Octal Address
Magnetic Tape (1/2")	00 for output 40 for input
Card or paper tape reader	41
Card or paper tape punch	01
Printer	02
OLA	42
Console	07 for output 47 for input

Figure 11•3

Additional Parameters (E)

Printer Line Spacing Information

Magnetic Tape - Direction of tape motion, odd or even parity, tape unit, and presence or absence of data transfer.

Peripheral Control and Branch - General Description

The Peripheral Control and Branch instruction can initiate two types of operations. It can initiate control operations such as error card rejection on the card reader or card punch, and rewind tape on a magnetic tape unit; or it can test peripheral status indicators such as error indicators, control unit activity indicators, and read/write channel indicators.

DEFINITIONS

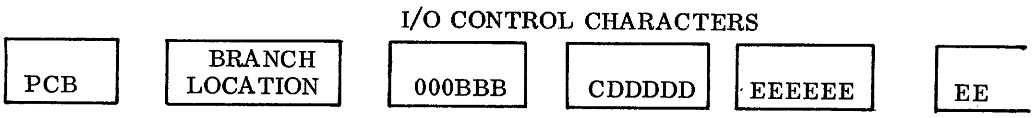


Figure 11•4

B, C, and D are identical to the PDT. E can be one or more control characters used to specify status tests and control activities. See figures 11•5 and 11•6 for summaries of the PCB and PDT variants.

PERIPHERAL CONTROL AND BRANCH (See flow chart pages 87 thru 89 and figure 11•6)

If only the channel is to be investigated, then only one variant is needed. This variant would be extracted in B1 cycle, and the channel condition would be stored by CBFTD10.

CAFTD10 (I) = $FTB \bullet FTC \bullet \overline{BST}$

CAFTB10 (I) = $B01 \bullet FTA \bullet B06 \bullet \overline{F1A} \bullet \overline{F5A}$ RWC1 Interlock
+ $B01 \bullet \overline{B02} \bullet B03 \bullet \overline{F5A} \bullet \overline{F1I}$ RWC1'
+ $B01 \bullet FTA \bullet \overline{B06} \bullet \overline{F1A}$ RWC1

CAFTC10 (I) = $\overline{B01} \bullet B02 \bullet \overline{B03} \bullet \overline{F2A}$ RWC2
+ $\overline{B01} \bullet FTA \bullet \overline{B04} \bullet \overline{B05} \bullet \overline{B06}$ Forces Ready
+ $B01 \bullet B02 \bullet \overline{B03} \bullet \overline{F3A}$ RWC3

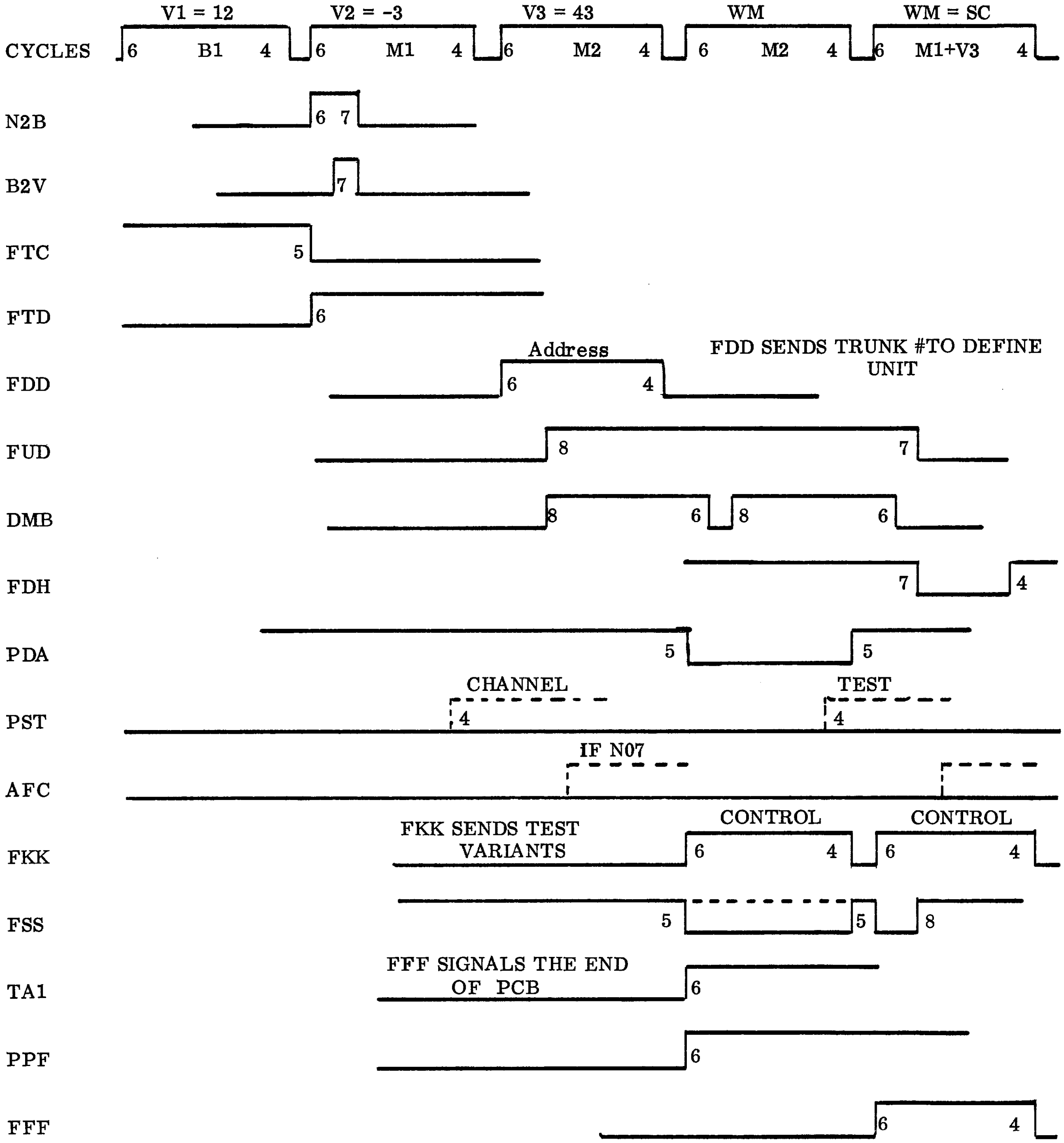
CAFTA10 (I) = B02 + B03

PCB VARIANTS

C = R/W CHANNEL	P = PRINTER
D = TAPE DRIVE	R = READER
H = PUNCH	T = TAPE CONT. UNIT
O = OLA	U = PERIPH. CONT. UNIT

V1		V2		V3			BRANCH ON:	SET CONDITION:	REMARKS:
RWC	NO	-	-	-	-	ALL	RWC BUSY		
0	0	4/0	U	0	-	R,H,P,O	DEVICE BUSY		
		↕	↑	0	D	T	TAPE DRIVE BUSY		
		4/0		1	-	ALL	PCU BUSY		
		0		2	D	T	BUSY	REWIND	
		4		2	D	T	BUSY	REWIND INTERLOCK	
		4/0		2	0	R,H,P,O	↑	NONE	
		↑		2	1	R, H		EJECT CARD W/H,C,E_R	(H) EJECT 2 CARDS
				2	2	R, H		EJECT CARD W/I,P,E_R	(H) PFR MODE
				2	3	R, H		BUSY ON H,C,E_R	MUST BE MANUALLY RESET BY HAND (H) PFR MODE
				2	4	R, H		BUSY ON I,P,E_R	
				2	5	R,H		SELECT D.T. MODE	
				2	6	R,H	↓	SELECT MODE 2	
				2	7	R,H	BUSY	SELECT MODE 1	CLEAR OTHER COND.
				3	-	O	UNCOND.	ISSUE DRP	(DEVICE READY PULSE)
				4	-	O,P	ERROR		
				4	1	R,H	H.C. ERROR		
				4	2	R,H	I.P. ERROR		(H) PFR MODE
				4	D	T	READ PARITY ERROR OR WRITE CURRENT ERR.		MODE 6 CHECK ON 3/4" READ
				5	-	O	RESPONSE FAILURE FROM H-800		PARITY ERROR
				5	D	T	TAPE LONG CHECK ERF		
		↓ 4/0		6	-	O	DATA TRANSFER NOT COMPLETE		
		0		6	D	T	TAPE DRIVE AT EOT		
		4		6	D	T	TAPE DRIVE AT BOT		
		4/0		3	1	R,H	BUSY	SELECT POCKET 3	
		↑		7	0	ALL	DON'T BRANCH	TURN ALLOW INT. OFF	
				7	1	ALL	DON'T BRANCH	TURN ALLOW INT. ON	
		↓	↓	7	4	ALL	DON'T BRANCH	TURN FIT OFF	
		4/O	U	7	5	ALL	FIT		

Figure 11-5



PCB TIMING

Figure 11-6

If the channel is ready, then the next instruction is performed. If the channel is busy, then another M1 cycle is performed to do the sequence change to the A address.

Some forms of the PCB test more than channel busy. In these cases, there would be additional variants to either set some type of control or to check for a condition. All of these variants are shown in Figure 11•5. If the variant is used for checking, and the condition is found, then the sequence change routine follows. This routine should be familiar by now; the contents of location 14 are transferred to location 17 by $\overline{\text{IIT}}$ in the 3rd control memory cycle.

PCB - 1st M1 cycle - (channel check only)

This cycle would normally be bringing out the second variant character; but in this case, it will be bringing out the op ocde of the next instruction with its word mark. If the channel is busy, this will be indicated by CBPST10 to the next state conditions. PST being set will keep the system in both EMC and E1C; therefore, another M1 cycle will take place to accomplish the sequence change. If the channel is ready, PST will be low. This inhibits the previously described conditions, but allows E3C instead. No major cycle is set; therefore, V3 cycle will be derived.

The sequence change is caused or happens by going into the second M1 cycle. As was stated before, PST will indicate that the channel is busy. PST will cause AFC to become set, and during the next M1 cycle, AFC causes II5 to be driven low. II5 being low at time 5, does not inhibit the internal transfer.

CBAFC10 = PST•DM1•CT1
CBII510 (I) = DM1•AFC•CPC
CBIIT10 = II5•T05 (Since II5 is low, so is IIT)

In the cases where more than just the channel is being tested, the system goes into a series of M2 cycles until a word mark is sensed. After this, it checks to see if any of the conditions have been met. As an example, let us test the card reader control unit for an error using RWC2; the unit is on trunk 1.

PCB/TEST/12/41/41/42.

B1•M1 Cycles

V1 will check for the RWC ready condition as was explained previously. CAFTD10 being high will indicate a ready condition. If it was low, then CBPST10 would be set to store this condition.

CBPST10 = CT4•FTD•DM1•PCB

Note: CBDM1$\underline{30}$ is used throughout the PCB. It is delayed, 8 to 6.

PST being set causes AFC to become set at the next M1 cycle. It cannot set during the first M1 because of timing. PST is set at CT4 of M1 and CBAFC10 = PST•DM1•CT1.

M1•M2 Cycles

V2 will address the particular peripheral unit through CAFDD1Y which gates in the trunk number to FUD.

$$\text{CAFDD1Y} = \text{CT6}\bullet\overline{\text{N07}}\bullet\text{PCB}\bullet\text{DM1}$$

(FDD from the CP will turn on FUD in the control unit with the matching trunk #).

M2 Cycles

Once a unit is addressed, CAFKK1Y will gate the condition being tested into BBFSS1Z through functions in the card reader control unit.

SUMMARY OF PCB

Checking the cycle flow of the PCB in the flow charts, it can be seen that the instruction will cause a sequence change to the A address if PST gets set at anytime during the execution of the PCB. During the first M2, the trunk number that was read out of memory at the end of M1 is sent to the peripheral control unit. During the second M2, the first test variant is sent to the peripheral control unit; if the condition tested for is present, the peripheral control unit function BBFSS10 will remain high. In the CP if FSS is high during M2 of a PCB, the PST will set. PST, you remember, will recirculate and cause a branch at the end of the PCB. If there are more variants, M2 will keep repeating itself. In each M2, successive variants are sent to the peripheral control unit; if any one of the conditions tested for is present, FSS will be high during that M2 cycle. Therefore, PST will be set and a branch will result. As the PCB keeps sending successive characters from memory, it will eventually send the op code of the next instruction. However, the word mark in the op code will send FFF to the control unit; FFF will cause the control unit to ignore the character sent and all future characters (until the next PCB). The word mark will also cause M1 to be the next cycle if a branch is required; if no branch is required, the next cycle will be V3.

Peripheral Status Signal (FSS)

$$\begin{aligned}\text{RAFSS10} = {} & \overline{\text{F06}}\bullet\text{STA} \\ & + \text{HCS}\bullet\text{F06} \\ & + \text{IPS}\bullet\text{F06}\bullet\overline{\text{DTM}} \\ & + \overline{\text{FUD}} \\ & + \overline{\text{FKK}}\end{aligned}$$

FUD (unit defined) is high from the time the trunk # is sent with FDD until FFF indicates no more variants are left. Since $\overline{\text{FUD}}$ is on the input to FSS, FSS will be high continuously if this control unit is not addressed. Even during the time that $\overline{\text{FUD}}$ is at ground, FSS may be high. FKK comes high every time a test variant is sent to the unit; so, if the unit is not being tested, FSS will remain high due to the bottom gate. Only if this unit is defined and is currently being tested can the other gates affect FSS. The top gate on FSS examines the busy STAtus. If the control unit is busy, STA will be high and will set FSS if bit six in the testing variant is a zero. However, in our example, FSS should depend on error conditions, so F06 is a one bit in our case. HCS (hole count status) will only come high if a hole count check is stored, and the low order half of the testing variant is 1_8. Similar logic tests for illegal punches.

$$\text{RA HCS10 (I)} = \overline{\text{HCC}} + \text{F03} + \text{F02} + \overline{\text{F01}}$$

$$\text{RD IPS00 (I)} = \text{IPC}\bullet\overline{\text{F03}}\bullet\text{F02}\bullet\overline{\text{F01}}$$

Meanwhile in the CP, if FSS is high it should set PST so that a branch will be performed.

$$\text{CBPST10} = \text{T04}\bullet\text{FSS}\bullet\overline{\text{PDA}}$$

$$\text{CBPDA10 (I)} = \text{PCB}\bullet\text{DM2}\bullet\text{CT6}\bullet\overline{\text{N07}}$$

PDA negation gets turned off every T05 because of a requirement of the PDT instruction; therefore, every M2 of a PCB instruction PDA assertion is turned off to allow the sensing of FSS on the input to PST.

Testing is only one purpose of the PCB instruction. It is also used to precondition peripheral control units. Exactly how each control unit reacts to the various PCB's is covered in detail in the seperate manuals that deal with each device.

PDT VARIANT CHART

			V1						V2						V3					
															(TAPE)					
			1 = INTERLOCK	RWC #					0 = OUT 1 = IN	PCU #					1 = FWD	1 = MOVE DATA	1 = EVEN PAR	DRIVE #		
		V	6	5	4	3	2	1	6	5	4	3	2	1	6	5	4	3	2	1
PRINT/SPACE			1	X	X	X	X	X	0	X	X	X	X	X	HOF	EOF	LINE COUNT			
CARD		READ	1	X	X	X	X	X	1	X	X	X	X	X	NOT USED					
		PUNCH	0	X	X	X	X	X	0	X	X	X	X	X	NOT USED					
MAG. TAPE	1" 2 AND 3" 4	READ FWD	*	X	X	X	X	X	1	X	X	X	X	X	1	1	X	X	X	X
		READ BACK	*	X	X	X	X	X	1	X	X	X	X	X	0	1	X	X	X	X
		BACK-SPACE	0	X	X	X	X	X	1	X	X	X	X	X	0	0	X	X	X	X
		WRITE FWD	*	X	X	X	X	X	0	X	X	X	X	X	N	1	X	X	X	X
	1" 2	ERASE FWD	0	X	X	X	X	X	0	X	X	X	X	X	N	0	X	X	X	X
	3" 4	SKIP WRITE	0	X	X	X	X	X	0	X	X	X	X	X	N	1				

X = REQUIRED DATA
N = NOT USED

* MUST BE "ONE" FOR TAPE TRANSFER RATES ABOVE 41 KC.

Fig. 11-7

PERIPHERAL DATA TRANSFER (See flow chart pages 90 thru 94 and figure 11•8)

Peripheral data transfer is under control of the RWC assigned to the control unit at order initiation. Should RWC2 or 3 be assigned, that PCU will be assured of a data transfer period every 6 us. Should RWC1 or the auxiliary be assigned, data transfer periods will occur at 12 us intervals, unless RWC1 is interlocked. In this case, it would receive a data transfer period every 6 us, and the auxiliary would not be allowed any. These two are usually assigned to the units with the slower data transfer rates. Whenever a RWC controlling a particular PCU becomes active, its number is sent to that Peripheral control unit on a special bus (FC1-5).

B1 Cycle

In this cycle, V1 is stored in both the B and V registers. The two low order bits of the V register are gated into three functions (CAFV110-FV3) for channel storage. The bits of the B registers are being gated into CAFTB10 and CAFTC10 to check for channel ready, which shows up on CAFTD10 as already explained. FTD has no affect on PST during the PDT, but it does affect the internal transfer of the starting address to the two control registers. If the channel is not ready, then that means that the registers are being used in some other instruction and should not be changed until it has finished. If FTD is not set, the transfers will be allowed later.

M1 Cycle

The second variant is gated into the F register for the trunk address. CAFDD1Y is generated to take this address over to the peripheral unit, and to identify either INPUT or OUTPUT. Both the assertion and negation of the F register positions are gated into five jumper functions, so that the different trunk configurations can be obtained. These configurations are obtained by cutting either the assertion or negation, depending upon which one is needed. Position 6 is normally used to indicate whether a PCU is input or output. Now, in the case of a TCU, the PCU is both; therefore, it can be addressed with position 6 either set or reset. Which one is cut? Both are cut, for into this position of the jumpers, there is a third choice namely, plus five. This third choice allows the TCU's interface to react for either input or output operations.

TAFJ110 = TAF0130 + TAF0140
TAFJ210 = TAF0230 + TAF0240
TAFJ310 = TAF0330 + TAF0340
TAFJ410 = TAF0430 + TAF0440
TAFJ610 = TAF0630 + TAF0640 + X11

$$\begin{aligned} \text{TAFUD10 (I) (I)} &= \text{T8S•FDD•FJ1•FJ2•FJ3•FJ4•FJ6} \\ &+ \overline{\text{FKK}}\text{•FUD} \\ &+ \text{FDH•FUD} \end{aligned}$$

CAFDD1Z = CT6•PRW•DM1

If the channel is ready, CBDUP10 will become set to indicate this.

CBDUP10 = EMC•PRW•FTD•CT3

Also during M1, if the channel is ready, the starting address is stored in the RWS location. CBDUP10 is set to store the fact that the channel was ready.

$$\begin{aligned}
&\text{CBIIT10} = \text{II3}\bullet\text{T03}\\
&\text{CBII310 (I)} = \text{EMC}\bullet\text{PRW}\bullet\text{FTD}\bullet\text{CPC}\\
&\text{CBDUP10} = \text{EMC}\bullet\text{PRW}\bullet\text{FTD}\bullet\text{CT3}
\end{aligned}$$

Another function which must be looked at is CBPDA10, which is normally high, but is driven low at certain times of the PDT and PCB instructions.

$$\begin{aligned}
\text{CBPDA10 (I)} &= \text{PRW}\bullet\text{DM1}\\
&+ \text{PST}\bullet\text{PRW}\bullet\text{PMC}\bullet\text{T06}\\
&+ \text{PCB}\bullet\text{DM2}\bullet\text{CT6}\bullet\overline{\text{N07}}\\
&+ \overline{\text{PDA}}\\
\\
\text{CBPDA00 (I)} &= \text{T05}+\text{PDA}+\text{BST}
\end{aligned}$$

M2a CYCLE

Taking a look at Figure 11•8, shows that PDA jumps up and down a few times storing the previous condition of PST. PDA is driven reset the first time by PRW•DM1 to allow the first check and PST should become set since FSS is normally high. PST being set prohibits the incremented sequence register from being delivered. PST being set also allows PDA to be driven low again, to recheck FSS on PST which is reset each cycle by CBCPS10. During the first M2 cycle, instead of a variant, a forced busy test is sent to the PCU. Shortly afterwards, CAFKK1Y checks to see if the CU is busy. During FKK, FSS will be allowed to go low, if the CU is not busy.

If the control unit is busy, FSS will not be allowed to go low. This causes PST to be set again, which still inhibits the incremented sequence register from being written into control memory. Therefore, the system will stay in the first M2 cycle until the CU becomes not busy. Meanwhile CP operations are stalled. If the channel becomes ready, FTD would set ACF to allow the starting address to be loaded into the RWC.

M2b-1 CYCLE

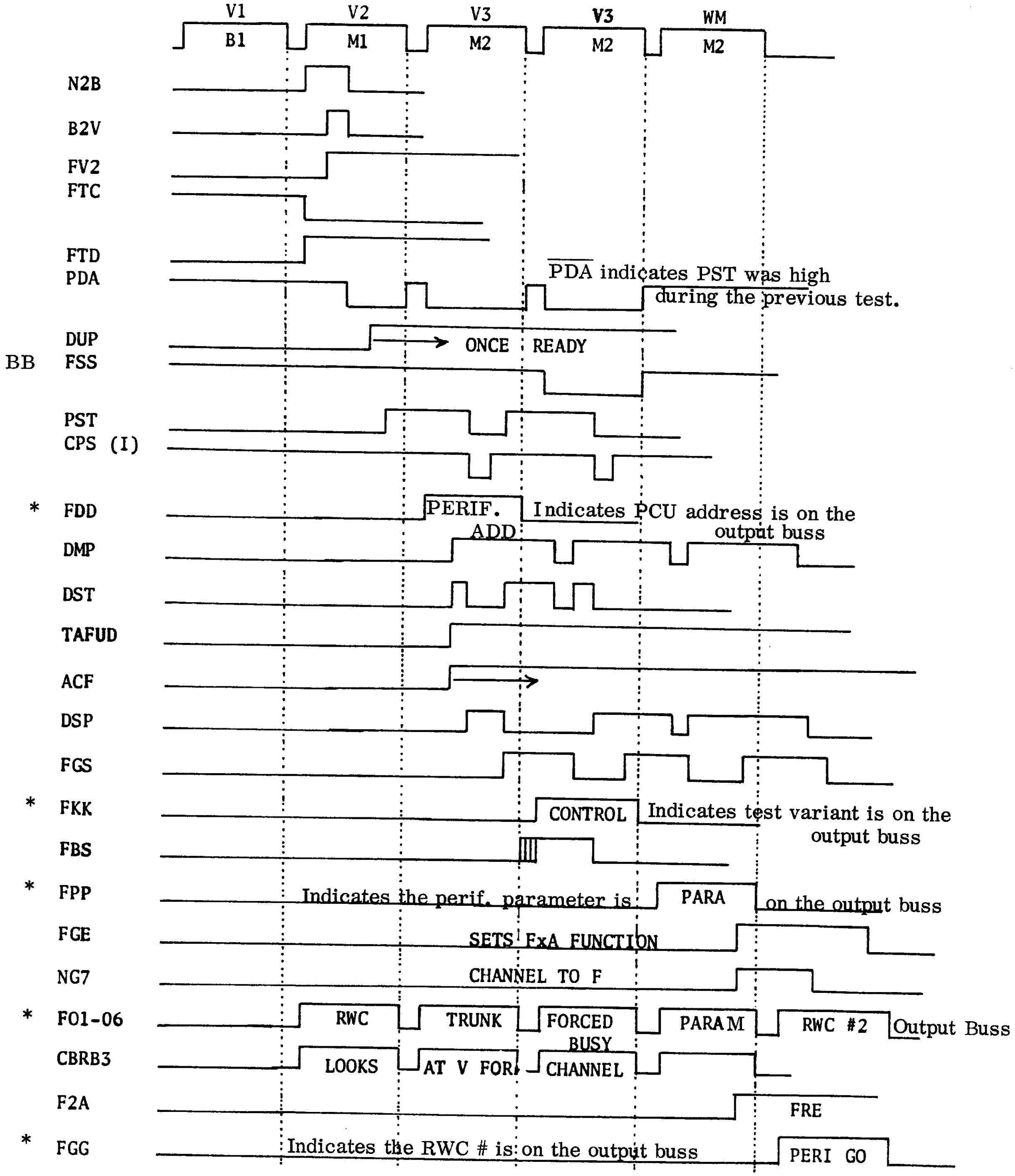
If the control unit was ready, FSS would go low, PST would not be set, and the next variant will be extracted. For the TCU, this would contain the drive number, and other needed information. V3 is gated to the appropriate TCU functions through CAFPP1Y.

$$\text{CAFPP1Y} = \text{CT6}\bullet\overline{\text{N07}}\bullet\text{DMP}\bullet\overline{\text{PST}}$$

As can be seen, FPP cannot come high if the control unit is busy. If there are more variants, the CU must make sure that only the one it needs is allowed. In the 1/2 inch TCU, only a V3 is needed. Of V3, bit 1, 2 and 3 are used for the drive number; bits 4 and 5 are the remainder of the operation code-V2 identified input/output; and bit 6 is direction. After the PCU receives the variant(s) it desires, successive ones are ignored. The card reader and card punch control units, require only two variants; therefore, the M2 cycle for this case is described in the M2 with word mark sensed.

M2 - Word Mark Sensed

When the word mark is sensed, if the channel is ready, the RWC will be transferred from the V register to F so that the PCU can store the channel it is assigned. This transfer is handled by NG7.



* Signals to PCU from CP

PDT TIMING

Figure 11•8

CANG710 = FGS•MA7
CAFGS10 = CT4•DMP•ACF

This is not a straight transfer of V to F; in fact, only one position of F is affected by V. Back in B1 cycle, the first two positions of the first variant were stored in FV1-3. These three functions are now gated in F by NG7 to the first three positions. The third position of the first variant is transferred to the fourth position of the F register.

CAF0110 = NG7•FV1
CAF0210 = NG7•FV2
CAF0310 = NG7•FV3
CAF0410 = NG7•V03

The fifth and sixth positions are not affected. An example will leave no doubts whatsoever.
Utilizing RWC #2 will cause V1 = 12

CAFV210 = $\overline{\text{V01}}$•V02

Then this will be gated into F02 as was shown. V03 would be set if the auxiliary channel is being used. During FGG time these four positions of the F register are gated into the FS's functions.

CAFGG1Y = CT6•N07•DSP•ACF
CADSP10 = $\overline{\text{PST}}$•DMP
TDFS100 (I) = FT8•FGG•F01
TDFS200 (I) = FT8•FGG•F02
TDFS300 (I) = FT8•FGG•F03
TDSF500 (I) = FT8•FGA•F04
FDFS400 (I) = $\overline{\text{FS5}}$

RWC storage functions in the TCU

As far as the CP is concerned, the execution cycles of the PDT are complete. However, as the CP continues to execute other instructions, its processing will be suspended occasionally while a memory cycle is used to transfer a character in or out of the CP.

Functions that originate in the CP and feed to the PCU'S:

F01-6	Output buss	FFF	Flag signal
FC1-5	Channel buss	FRR	Master reset
FUD	Defines trunk	FCC	Central clear
FKK	Control or test	F22	CP time 2
FPP	Perif. parameter	F66	CP time 6
FGG	Go signal		

Functions that originate in PCU's and feed to the CP:

F51-F56	Input buss
FR1-FR3	Response lines
FSS	Perif. status signal
FIT	Interrupt

PERIPHERAL RESPONSE CYCLES (See flow chart pages 105 thru 110)

Cycle Counter (Figure 11•9)

To insure that each PCU has time to transfer its data to the CP and/or to receive data from the CP, a cycle counter is set up so that each channel is allowed access time at regular intervals, (if needed). Normally, each channel is allowed access time to the CP every 6 us. With the option, the time that is normally granted to RWC1, is shared with the auxiliary so that these two would only be granted access time every 12 us.

The actual count is decoded by CAFD110-FD3, with inputs of FA1 and FA2 which use backup functions FB1 and FB2. FA1 and FA2 are set at TØ8 and their backup functions follow at TØ5.

$$\text{CAFA1ØØ(I)} = \text{TØ8}\bullet\overline{\text{FB1}}\bullet\text{FB2} + \text{FA1}$$

$$\text{CAFA11Ø(I)} = \underline{\text{TØ8}}\bullet\text{FB1}\bullet\overline{\text{FB2}} + \text{FA1}$$

$$\text{CAFA2ØØ(I)} = \text{TØ8}\bullet\overline{\text{FB2}} + \text{FA2}$$

$$\text{CAFA21Ø(I)} = \underline{\text{TØ8}}\bullet\text{FB1}\bullet\text{FB2} + \text{FA2}$$

As can be seen in Figure 11 • 9 , FA1 is high for two cycles and low for one cycle, as is FA2. With this arrangement, the three distinct configurations can be formed.

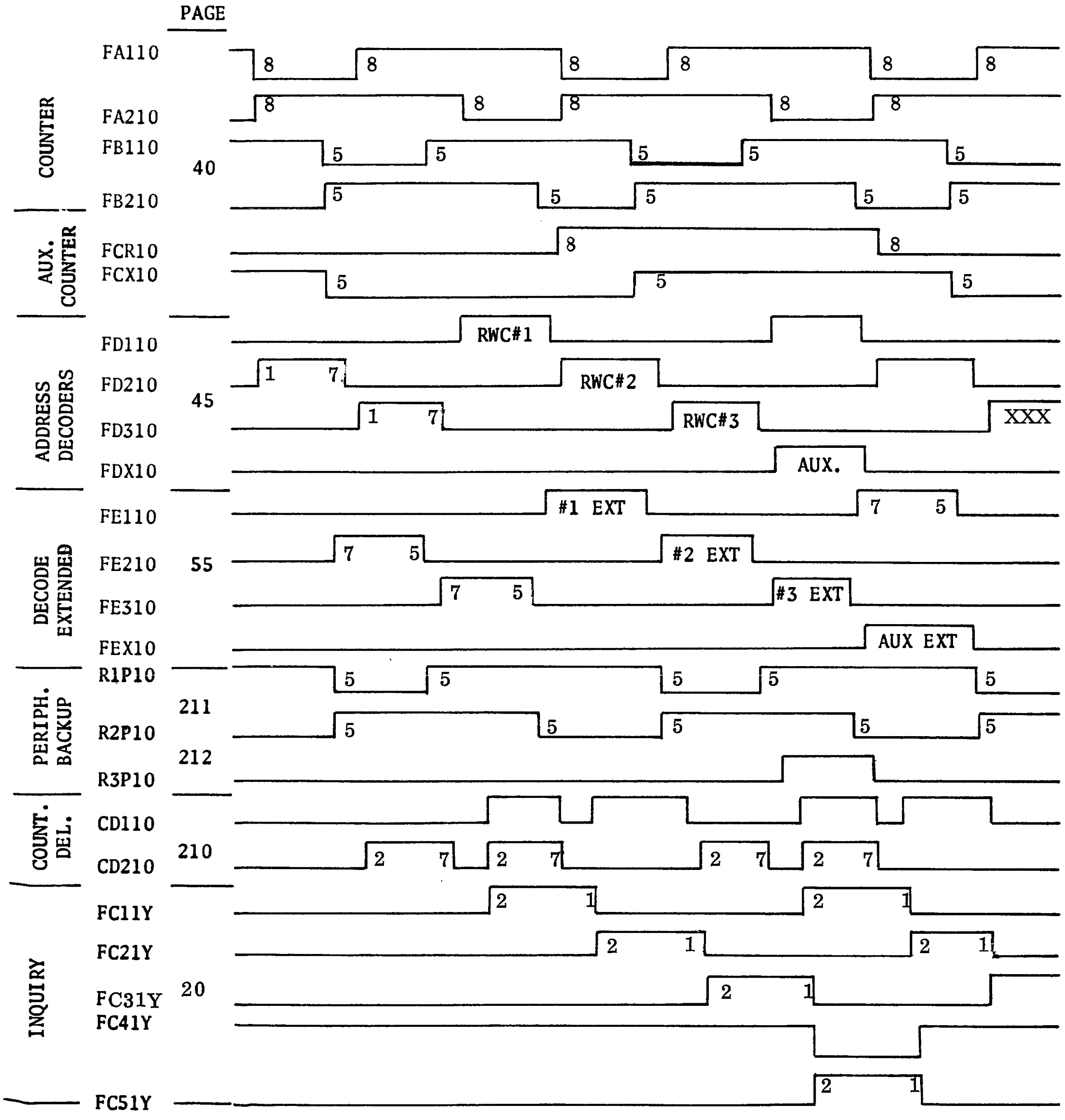
$$\text{CAFD11Ø} = \text{TØ1}\bullet\text{FA1}\bullet\overline{\text{FA2}} + \overline{\text{TØ8}}\bullet\text{FD1}$$
$$\text{CAFD21Ø} = \text{TØ1}\bullet\overline{\text{FA1}}\bullet\text{FA2} + \overline{\text{TØ8}}\bullet\text{FD2}$$
$$\text{CAFD31Ø} = \text{TØ1}\bullet\text{FA1}\bullet\text{FA2} + \overline{\text{TØ8}}\bullet\text{FD3}$$

For the optional RWC, auxiliary functions are needed so that every other time that RWC1 should be granted access time, the RWC1' is granted the time instead. This is handled by CAFCR1Ø and FCX. FD1 comes up every time, but the auxiliary counter comes up every other time. These are not the functions which ask the PCU's if they want a response cycle. The responses are handeld by FC1-5. FC1-3 will follow FD1-3 at TØ2. FC4 will follow FDX's negation, which allows it to go low once every 12 us. FC5 follows the assertion of FDX, so it is high once every 12 us.

$$\text{CAFC11Y} = \text{TØ2}\bullet\text{FD1} + \text{TØ3}\bullet\text{FD1} + \overline{\text{TØ2}}\bullet\text{FC1}$$
$$\text{CAFC21Y} = \text{TØ2}\bullet\text{FD2} + \text{TØ3}\bullet\text{FD2} + \overline{\text{TØ2}}\bullet\text{FC2}$$
$$\text{CAFC31Y} = \text{TØ2}\bullet\text{FD3} + \text{TØ3}\bullet\text{FD3} + \overline{\text{TØ2}}\bullet\text{FC3}$$
$$\text{CAFC41Y} = \text{TØ2}\bullet\overline{\text{FDX}} + \text{TØ3}\bullet\overline{\text{FDX}} + \overline{\text{TØ2}}\bullet\text{FC4}$$
$$\text{CAFC51Y} = \text{TØ2}\bullet\text{FDX} + \text{TØ3}\bullet\text{FDX} + \overline{\text{TØ2}}\bullet\text{FC5}$$

The inquiry functions are gated into each PCU to grant the PCU access time if it matches the inquiry configuration and needs the time.

Another job of the cycle counter is to generate the functions necessary to address the starting and current address counters during buffer cycles. The current address counter is addressed during



CYCLE COUNTER

Figure 11•9

the first CM cycle, mainly by R1P and R2P which follow FB1 and FB2 respectively. During the fourth CM cycle, CD1 and CD2 are used as a count delayed since the counter is changing.

$\text{CAR1P10} = \text{FB1}\bullet\overline{\text{KRF}} + \text{R01}\bullet\text{KRF}$

$\text{CAR2P10} = \text{FB2}\bullet\overline{\text{KRF}} + \text{R02}\bullet\text{KRF}$

$\text{CBCD100 (I)} = \text{R1P}\bullet\text{T02} + \text{CD1}$ $\qquad$ $\text{CBCD110 (I)} = \overline{\text{CD1}} + \text{T08}$

$\text{CBCD200 (I)} = \text{R2P}\bullet\text{T02} + \text{CD2}$ $\qquad$ $\text{CBCD210 (I)} = \overline{\text{CD2}} + \text{T08}$

During the PDT instruction, the channel that is to be used is gated into FS1-5 for storage by FGG. The cycle counter functions are gated against the storage functions for a match.

$\text{TDFS100 (I)} = \text{FT8}\bullet\text{FGG}\bullet\text{F01} + \overline{\text{FS2}}\bullet\overline{\text{FS3}}$

$\text{TDFS200 (I)} = \text{FT8}\bullet\text{FGG}\bullet\text{F02} + \overline{\text{FS1}}\bullet\overline{\text{FS3}}$

$\text{TDFS300 (I)} = \text{FT8}\bullet\text{FGG}\bullet\text{F03} + \overline{\text{FS1}}\bullet\overline{\text{FS2}}$

$\text{TDFS400 (I)} = \text{FT4}\bullet\overline{\text{FOS}} + \overline{\text{FS5}}$

$\text{TDFS500 (I)} = \text{FT8}\bullet\text{FGA}\bullet\text{F04} + \overline{\text{FS4}}$

All peripheral control units store the channel assigned to it in functions called FS1 - FS5.

$$\begin{aligned}\text{TDFCN10} = {} & \overline{\text{FS1}}\bullet\text{FC1}\\ & + \overline{\text{FS2}}\bullet\text{FC2}\\ & + \overline{\text{FS3}}\bullet\text{FC3}\\ & + \overline{\text{FS4}}\bullet\text{FC4}\\ & + \overline{\text{FS5}}\bullet\text{FC5}\end{aligned}$$

FCN = Channel compare not.

In each peripheral control unit, when the FCN20 function in that control unit comes high, it means that the channel assigned to that unit now has access to the CP. Because of the simplified logic used, it is sometimes hard to figure out how FCN works. The best way is to look at FC1, FC2, and FC3 as a scanner. One of these functions will be high at all times; therefore, half of one of the top three gates on FCN should always be high. When the channel count (FC1-3) gets to the channel storage function that is low (the negation of the channel stored), FCN will go low; unless the auxiliary count (FC5) is high, and the auxiliary channel was not assigned ($\overline{\text{FS5}}$ is high). Notice that $\overline{\text{FS4}}$ is really the assertion of $\overline{\text{FS5}}$, and remember that FC4 is really the negation of FC5.

Of course $\overline{\text{FCN}}$ does not cause a buffer cycle all by itself. Each control unit has a function called FAC that comes high every time a buffer cycle is needed. Also, there are seven different types of buffer cycles, and several functions are used to indicate the type of buffer cycle. For example, FAC•FON indicate a normal output response. Because the various peripheral control units all use identical functions for communication with the CP, it pays to learn the meaning and usage of these interface functions as soon as possible.

For the PCU to tell the CP what type of response cycle it desires, three response lines are used. With three lines, eight difference configurations can be high, and for action to be taken by the CP, one or more of them have to go low.

DEMAND TYPE	FR3	FR2	FR1	EXPLANATION
INPUT FRAME DEMAND (SPECIAL)	0	0	0	LOAD THIS CHARACTER INTO THE LOCATION SPECIFIED BY THE RWC AND DECREMENT THE RWC BY ONE.
SPECIAL OUTPUT FRAME DEMAND FOR	0	0	1	SEND THE CHARACTER IN THE LOCATION SPECIFIED BY THE RWC.
SPECIAL ROW DEMAND FRS	0	1	0	TRANSFER THE RWC INTO THE RWS (STARTING) ADDRESS STORAGE
NORMAL ROW DEMAND FRN	0	1	1	TRANSFER THE STARTING ADDRESS INTO THE RWC.
INPUT FRAME DEMAND FIN	1	0	0	LOAD THIS CHARACTER INTO THE LOCATION SPECIFIED BY THE RWC. INCREMENT THE RWC BY ONE.
NORMAL OUTPUT FRAME DEMAND FON	1	0	1	SEND THE CHARACTER IN THE LOCATION SPECIFIED BY THE RWC. INCREMENT THE RWC BY ONE.
END OF ORDER FEO	1	1	0	RELEASE THE RWC FROM THIS C.U.
NO ACTION	1	1	1	THE C.U. REQUIRES NO ACTION. USE THIS C.P. CYCLE FOR C.P. PROCESSING.

Figure 11•10

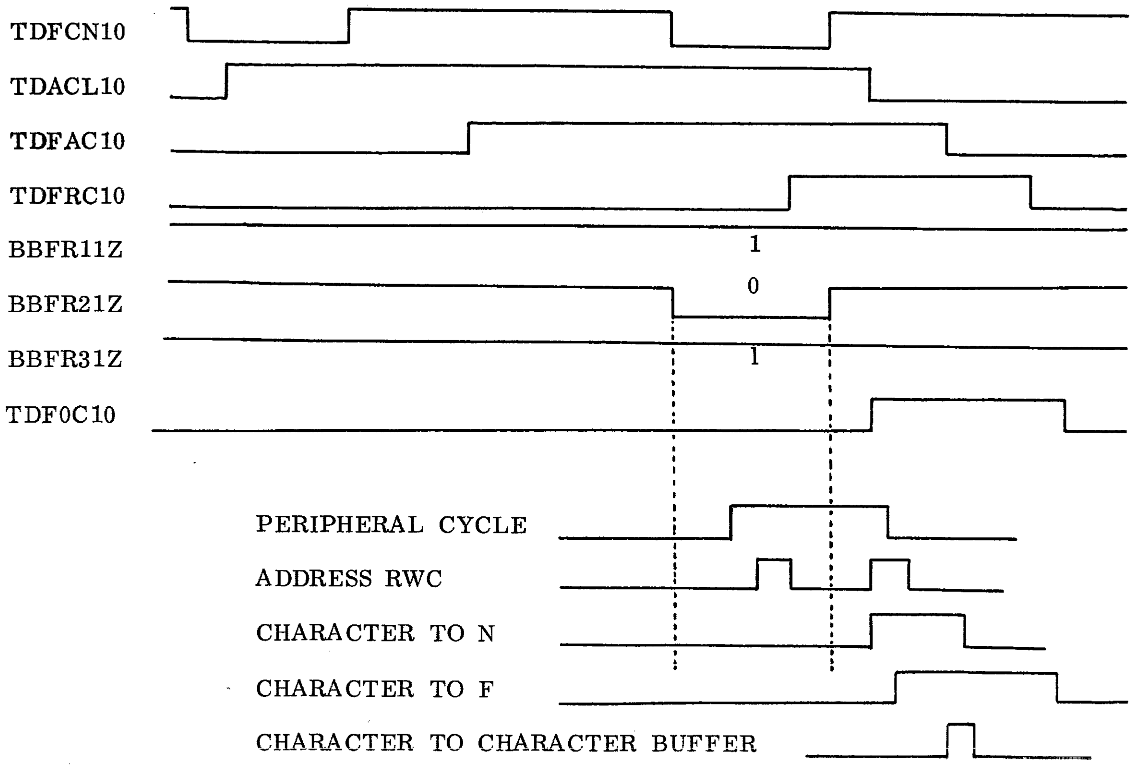
FON RESPONSE IN 3/4 INCH TCU

For the C.P. to send the TCU a character, it must be writing, so the function TAWFD1Ø will be set to indicate a write instruction. A look at the response chart shows that FR2 must go low. In the TCU, only FR1 and FR2 are present since the condition represented by FR3 being low can never exist. The response lines are common to all PCU's, and if one of them does not appear in a PCU, then the bus terminators will keep it high.

$$\begin{aligned} BBFR11Z &= \overline{FEO}\bullet EOR \\ &+ \overline{FEO}\bullet WFD \\ &+ \overline{FAC} \\ &+ FCN \end{aligned}$$

$$\begin{aligned} BBFR21Z &= EOR \\ &+ FEO \\ &+ \overline{FAC} \\ &+ FCN \end{aligned}$$

As can be seen, if FAC is not set, then both FR1 and FR2 are high. Once FAC is set, then FR1 and FR2 are still high as long as FCN is high. The only time FCN goes low, is when it is this PCU's time to request. Assuming that an output transfer (relative to the central processor) is required, a look at the top gate of FR1 shows that EOR (end of record) is needed. The second gate needs a write instruction which is present; therefore, FR1 will remain high. FR2 needs either end of record or peripheral end of order to remain high. Neither is present; therefore, it goes low. The peripheral response is now 1Ø1.



TIMING OF TYPICAL OUTPUT FRAME DEMAND
Figure 11•11

The peripheral response functions are always being monitored in the CP, so that it knows whether to use the next cycle or to grant that cycle to a PCU. This is accomplished by gating FR2•FR3 into the CPC functions. If either FR2 + FR3 goes low, then a PCU will be granted the cycle. Why doesn't FR1 affect what type of cycle it is? A look at the response chart shows that the only time FR2•FR3 are both set is 111 (no action) and 11Ø (end of order). These two condtions do not require use of memory. End of order action can take place at the same time as other CP operations.

There are several CPC functions, but all result from the same logic (except for timing).

CACPCØB(I) = FR2•FR3•$\overline{\text{KRF}}$
CACPC1A = FR2•FR3•$\overline{\text{KRF}}$

As can be seen, while FR2•FR3 are both high, these will be CP cycles. Either one going low, indicates that a peripheral cycle is needed. All of the CPC functions are not affected at one time as figure 11•12 will show. The reason for this is that different parts of the CP need to know there is a buffer cycle at different times.

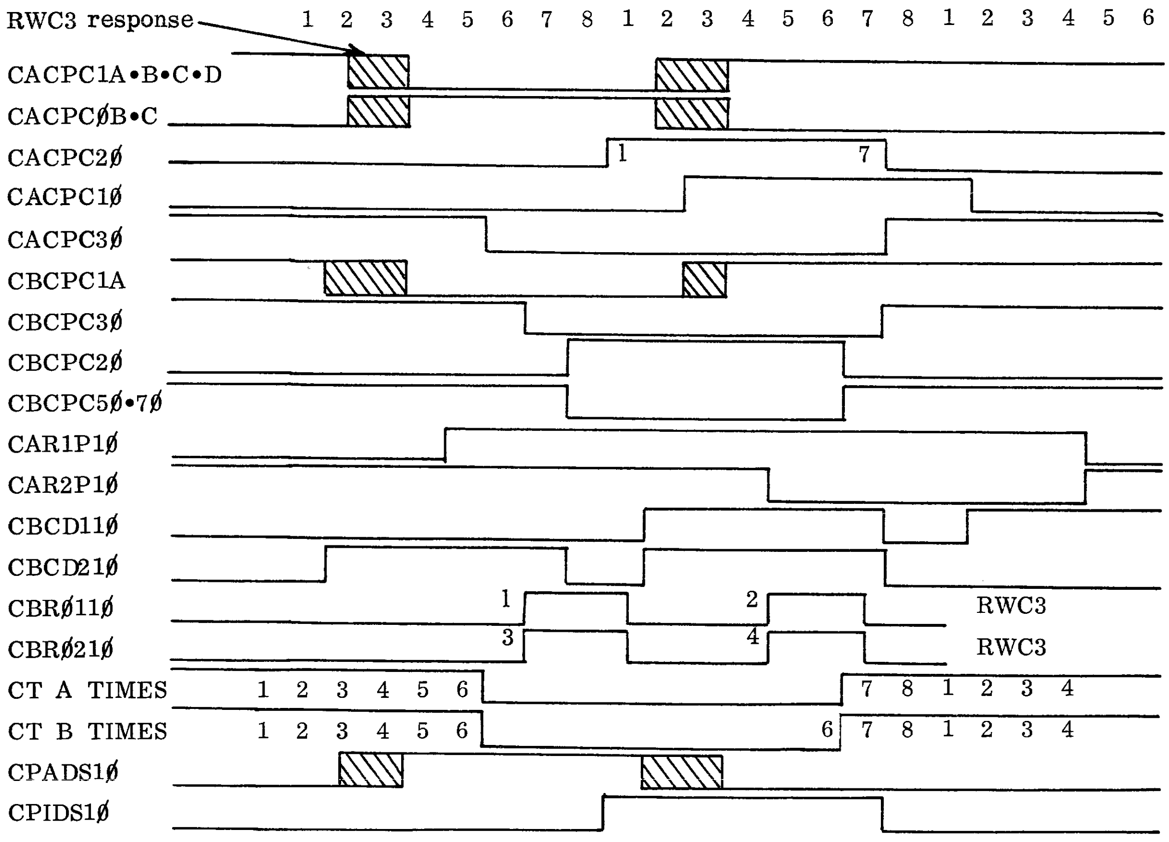


Figure 11-12

During the first CM cycle, R1P•R2P will generate the address for the channel being used. During the fourth CM cycle, CD1•CD2 will generate the address. The second and third CM cycles play no part in the peripheral cycles. There is no transfer taking place. Whenever one register is transferred to another one, it is done through the S register. In a normal input or output peripheral cycle, only the current counter is addressed. There are two cases in which the starting counter must be addressed. When reading or punching DTM at the half way point, the current address should become the starting address. Therefore, the current counter is transferred to the starting. This is handled by CAFRC10. If the student does not understand the term DTM, refer back to this section when studying the card reader and punch. FRC sets R4P which feeds CBR4510. At T05, R04 will be set to address the starting location. The channel counter supplies the low order bit to complete the address.

When reading or punching, at the end of each row the starting address is needed once more in the current counter. The row pulse signal causes a row demand, 022, which gives This is gated directly into R04 at T07 to give the starting counter's address.

The S register is normally incremented, but during the transfer of starting to current and current to starting, neither incrementing or decrementing is desired. No gate can be satisfied on CB1SR10 so no incrementing takes place. Since there is no CP cycle, CPADS10 becomes true. With ADS set at T01, CPIDS10 becomes true to drive CBDSR1A low. Since there is no incrementing or decrementing, the address in the register remains the same and is simply transferred.

Other Responses

While doing the read instruction, the flow of data is in the opposite direction from the character buffer to CP. Since the write function WFD is not up during the read instruction, FR1 will also go low along with FR2, making the response 100.

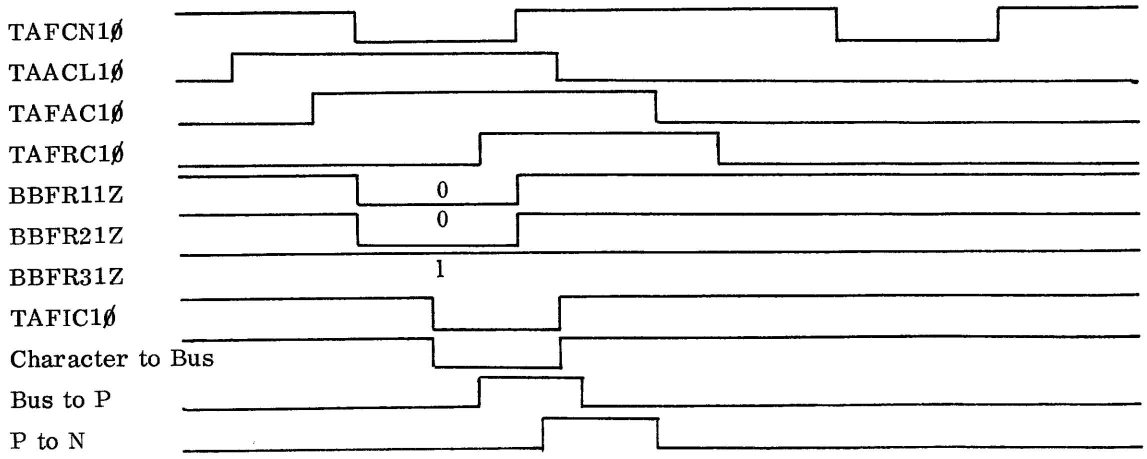


Figure 11•13

At the end of the instruction, FEO (peripheral end of order) will come high. A response cycle will be requested, and the response 110 will be generated so that the CP can release the channel the PCU was using. The response sets CAFRE10, which resets the activity function of the channel that has the use of this cycle.

SECTION 12
CENTRAL PROCESSOR CONTROL PANEL

I. GENERAL DESCRIPTION

The functions of the H-200 Central Processor Control Panel are: to enable the operator to enter binary information manually into the control memory and the main memory; to permit display of those areas; and to provide the controls necessary to operate the central processor; such as AC ON, DC ON, INITIALIZE, BOOTSTRAP, RUN, etc. The control panel also contains SENSE switches (which may be altered by the operator to change program flow) and check indicators (e.g., Parity).

Section II deals with "Operator Instructions." A description of what occurs when each button is hit, and methods of interrogation, instruction, and control are given.

Section III deals with the machine logic for the Control Panel Functions.

II. OPERATOR CONTROLS (See figure 12•1)

A. ADDRESS Push Buttons and Indicators

These indicators display the address of a charactor in memory. The contents of the Memory Address Register (MAR) are displayed in an array of illuminated push buttons arranged in four or five octal groups. The fifth (high-order) octal group is present only if the central processor has more than 4096 characters of storage. Room for a sixth octal group is provided at the extreme high-order end of the array. A memory address is displayed as a binary pattern of lights. If a button is illuminated, the corresponding bit is "one". If a button is not illuminated, the bit is "zero".

Any bit in the Memory Address Register can be changed from zero to one by depressing the push button, which sets the bit and illuminates the light. To change a bit from one to zero, however, it is necessary to reset all the MAR bits to zeros by depressing the ADDRESS CLEAR button (or the INITIALIZE button). The desired ones may then be set by depressing the appropriate push buttons.

The contents of the Memory Address Register can be changed manually only in the Stop mode, because the ADDRESS buttons are interlocked to be effective only if the Stop light is on. The lights behind the buttons are effective at all times, however, and display the instantaneous contents of MAR even while the program is running.

B. CONTENTS Push Buttons and Indicators

These push button indicators normally display a character of memory.

The contents of the Memory Local Register (MLR) are displayed as a binary pattern of lights. The MLR bits are set and reset and the push buttons are illuminated according to the same rules as for the Memory Address Register, except that CONTENTS CLEAR (or INITIALIZE) resets the register rather than ADDRESS CLEAR.

The stop button and the sense switches are the only buttons that have any effect while a program is running.

All signals [illegible] in Control Panel have Prefix CP Example CPCNB

All Buttons can be pushed during GO, but nothing is effected. [illegible] in Stop Mode the buttons are operational

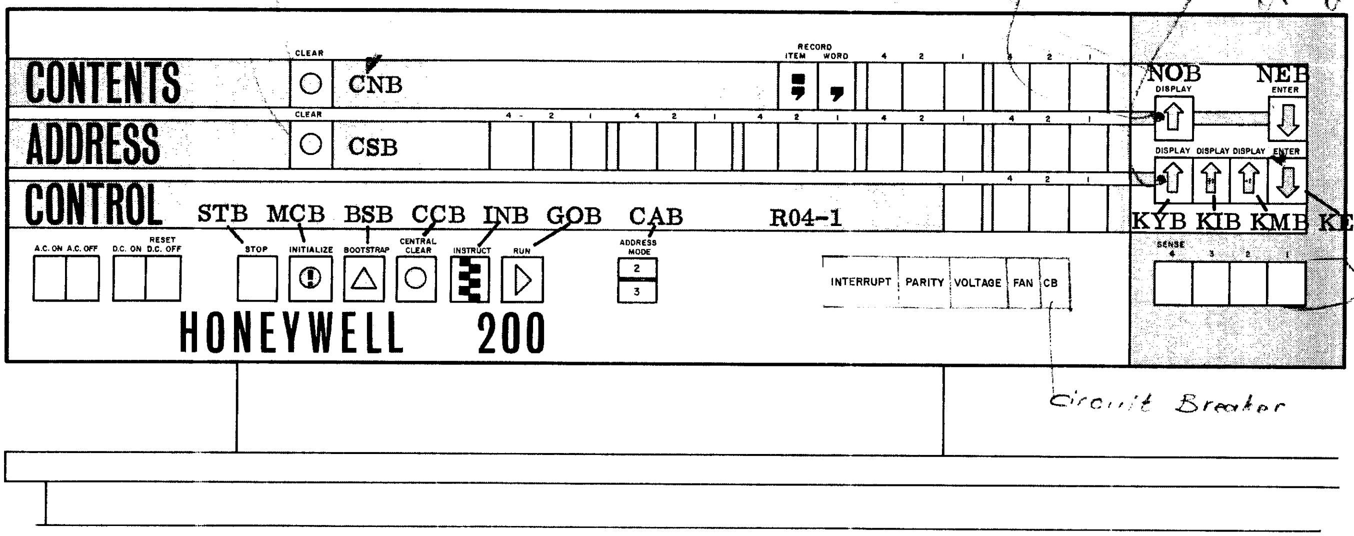
only to [illegible] in stop [illegible]

to load a specific Address Reg

Xfer of N Contents into B Reg

In stop mode indicates N the contents of the by B indicated address

Gives display of R-Reg selected address



for B conditions

Circuit Breaker

Figure 12•1

Pressing initialize clears M. S Parity

Bootstrap short [illegible]

Instruct Only one instr. is performed automatically, To proceed to next instr. this button must be hit

Honeywell 200 Operator's Control Panel

Two push buttons, DISPLAY and ENTER, are associated with the CONTENTS button.

1. DISPLAY
   This button is effective only in the Stop mode. When DISPLAY is depressed, the character in memory whose address is in the Memory Address Register is displayed with punctuation (item bit and word bit) in CONTENTS.

   If the character has bad parity, the Parity indicator will go on. The parity indicator may be reset prior to performing further manual operations (see Section II, E), but all manual operations except BOOTSTRAP and INSTRUCT can be performed while the Parity indicator is on.

2. ENTER
   This button is effective only in the Stop mode. When ENTER is depressed, the character and punctuation displayed in CONTENTS are written into the memory location whose address is displayed in the ADDRESS indicators. The previous contents of that memory location are destroyed. The character is always written into memory with correct parity. If that location contained bad parity before the ENTER button was depressed, it will contain correct parity after the button has been depressed. However, use of the ENTER button never causes the Parity indicator to go on or off.

C. CONTROL Push Buttons and Indicators

These push-button indicators display the address of a control memory location as a binary pattern of lights. The display is permanent and changes only when the push buttons are pressed. The setting of the CONTROL buttons can be altered at any time by pressing the push buttons. Alternate depressions cause a push button to be illuminated and darkened.

It should be noted that the CONTROL buttons play no role while the program is running, nor is their display affected by a running program. They are used only in conjunction with the following associated push buttons: DISPLAY, DISPLAY + 1, DISPLAY -1, & ENTER.

1. DISPLAY
   This button is effective only in the Stop mode. When DISPLAY is depressed, the following conditions result:

   a. The contents of the control memory location designated by the CONTROL indicators are displayed in the ADDRESS indicators.

   b. The character in memory stored at the address displayed in "a" above is displayed in the CONTENTS indicators.

2. DISPLAY + 1
   This button is effective only in the Stop mode. When DISPLAY + 1 is depressed, conditions "a" and "b" of paragraph 1 above result. In addition, the address displayed in "a" is incremented by one and returned to the control memory location designated by the CONTROL indicators.

For example, if the CONTROL indicators designate the sequence counter, whose contents are 3147 (octal), depression of the DISPLAY + 1 button causes: 3147_8 to be displayed in the ADDRESS indicators; the character at 3147_8 to be displayed in the CONTENTS indicators; and the contents of the sequence counter to be changed to 3150_8.

a. The DISPLAY -1 button will display and decrement by one, rather than incrementing by one as the DISPLAY + 1 button does.

3. ENTER

This button is effective only in the Stop mode. When ENTER is depressed, the octal configuration displayed in the ADDRESS indicators is written into the control memory location designated by the CONTROL indicators. The previous contents of that control memory location are destroyed. The character in memory stored at the new address is displayed in the CONTENTS indicators.

D. STOP Button

This push button is lighted when the system is in the Stop mode. It is not lighted when in the Run mode. In the Stop mode, the system executes no program instructions and actuates no peripheral equipment, except by manual "stepping". In the Run mode the system operates automatically.

The Stop mode can be entered in one of five ways:

1. A program halt, or halt and branch instruction is executed (normal).

2. The STOP button is depressed (normal).

3. A parity check occurs. This is not a normal stop. The instruction is stopped at the end of the current memory cycle, but all peripheral operations are allowed to go to completion. (In the interim, if any, neither the Stop light nor the Run light is lit.)

4. The CENTRAL CLEAR button is depressed after the STOP button has been pressed. This is not a normal stop (see Section II, E below).

5. The INITIALIZE button is depressed after the STOP button has been pressed. This is not a normal stop (see Section II, F, page 7).

In a normal stop, the instruction in extraction or execution at the time of the stop is fully executed and is the last one performed. The Stop mode is not entered and the Stop light is not lit, however, until all periperal operations that are in progress have come to completion. (In the interim, if any, neither the Stop light nor the Run light is lit.)

After an abnormal stop, the program generally cannot be resumed successfully by simply depressing the RUN button.

E. CENTRAL CLEAR Button

This button is effective only after the STOP button has been depressed. When CENTRAL CLEAR is depressed, central processor operations terminate at the end of the current memory cycle (and the RUN light goes off); the central processor cycle register is reset; and the Parity indicator is turned off.

Peripheral operations are allowed to go to completion. If the Stop light does not go on after CENTRAL CLEAR is actuated, then a stored order condition must exist in some peripheral device.

F. INITIALIZE Button

The INITIALIZE button is effective only after the STOP button has been depressed. When held down, this switch causes all control panel lamps except AC ON, AC OFF, DC ON, DC OFF, POWER, FAN and CB to light, thus providing a filament test for those lamps. When the button is released, the indicators resume their original state except for those indicators directly altered by the action of the INITIALIZE switch. Depressing the INITIALIZE button has the following effects:

1. The Stop light goes on and central processor operations terminate at once, with loss of information in main and control memories unless the RUN light is off and no peripheral transfers are in progress.

2. Central processor stored orders and subcommands are reset.

3. The central processor cycle register is reset.

4. The central processor clock is initialized.

5. Central processor parity, overflow, and zero balance indicators are cleared

6. The Memory Address and Memory Local Registers, together with their associated displays (ADDRESS and CONTENTS), are cleared to zeros.

7. The addressing mode is set to 2-character addressing.

8. The interrupt interlock is reset.

9. An "initialize" signal is sent to all peripheral control units, without exception, causing them to release RWCs and clear stored orders, to clear error indicators, to inhibit interrupts until enabled, and to enter a "standard" mode. Specifically:

   a. Printer Control:
   Same response as to local "clear", plus error flop reset.

   b. Tape Control:
   Error flops reset

c. Card Reader-Punch Control:
Stored order cleared; error flops reset; "alphanumeric" mode entered, error handling flops reset.

G. RUN Button

This push button is lighted when the system is in the Run mode. It is not lighted when in the Stop mode. The Run mode can be entered by depressing the RUN button, unless a parity check is stored. A stored parity check must be cleared before the RUN button will be effective.

In the Run mode, the system executes program instructions stored in memory, under control of the sequencing counter, with no manual intervention.

H. INSTRUCT Button

Depression of this button, which is effective in the Stop mode only, causes the program to execute the next instruction only. While that instruction is being executed, the Stop light is dark and the Run light is illuminated. If the instruction is a peripheral data transfer instruction, the Run light will go out when the data transfer begins, so that both Run and Stop lights will be dark until data transfer terminates, at which time the Stop light will go on.

I. BOOTSTRAP Button

Depression of this button, which is effective only in the Stop mode, has the following results:

1. The address indicated by the setting of the ADDRESS buttons is placed into the sequence counter, the A-address counter the B-address counter, and both counters of RWC1.

2. A simulated peripheral read instruction specifying RWC1 (with interlock) is generated. The control unit addressed is designated by the setting of the low-order five CONTENTS buttons. If a Tape Control is so designated, then Tape Unit number zero is the device that is read.

3. Each character read into memory during a Bootstrap operation has its punctuation bits automatically cleared to zeros.

4. A Bootstrap operation is not terminated by a record mark in memory; instead, the information flow is terminated by the peripheral control.

The Memory Address Register, the Memory Local Register, the sequence counter, the A-address counter, the B-address counter, and both counters of RWC1 are altered by the Bootstrap operation. The ADDRESS MODE button has no effect on the Bootstrap itself but is used when loading a program written for 3-character addressing (see Section II,M,5).

J. SENSE Switches

These four illuminated push buttons are used to control program flow, since they can each be tested by program instructions which branch conditionally on the setting of the switch. The system should be in the Stop mode while the switches are altered; otherwise, the results of the conditional branch cannot be specified if the conditional branch instruction is executed within 50 milliseconds after the SENSE switch has been altered.

The SENSE switches are numbered from left to right. Wrong!

K. AC ON, AC OFF, DC ON, DC OFF Indicating Push-Button Switches

1. AC ON, depressed first in a normal power-up, applies ac voltage, lights the AC ON indicator and extinguishes the AC OFF indicator. The AC ON indicator is not affected by the power checks -- Power, Fan or CB (see Section III).

2. The AC OFF switch removes both ac and dc voltages and lights the AC OFF indicator. AC OFF is interlocked with the STOP and RUN buttons. To remove ac power, therefore, the STOP button must be depressed prior to depressing AC OFF. If the RUN button has been depressed, the AC OFF switch is not effective until the STOP button is depressed.

3. DC ON, the second button depressed in a normal power-up, is effective only after AC ON has been depressed. DC ON applies dc voltages, lights the DC ON indicator and extinguishes the DC OFF indicator. The DC ON indicator is extinguished by the power checks--Power, Fan or CB.

4. The DC OFF switch removes dc voltages and lights the DC OFF indicator. DC OFF is interlocked with the STOP and RUN buttons. To remove dc power, therefore, the STOP button must be depressed prior to depressing DC OFF. If the RUN button has been depressed, the DC OFF switch is not effective until the STOP button is depressed. The DC OFF indicator is lit by the power checks--Power, Fan or CB--and is extinguished by the DC ON Switch. When lighted by a "Power" power check, the DC OFF button glows red; in other cases, it is white. Following a Power power check, depressing the DC OFF button resets only the Power check indicator. DC OFF has no control over AC power.

L. ADDRESS MODE Button

This split indicator push button appears on the Control Panel for only those systems with the Advanced Programming Instruction option. Its upper or lower half is illuminated if the central processor is in the 2-character mode or 3-character mode, respectivly. The indicator changes while the program is running as the program executes Change Addressing Mode instructions. The push button is effective in the Stop mode only, and successive depressions cause the central processor to switch from one mode to another, alternately.

Depression of the INITIALIZE button sets addressing to the 2-character mode, regardless of the previous mode.

The ADDRESS MODE button is used in the Bootstrap operation to load a program written for 3-character addressing.

M. CONTROL PANEL BASIC PROCEDURES

The basic operating procedures outlined below can be performed only if the Stop light is lit. If depressing the STOP button does not cause the Stop light to go on, then the central processor must be cleared or initialized before proceeding.

1. To display a control memory location and the character in memory it references:

   a. Depress the STOP button (if not already in Stop mode).
   b. Depress the CONTROL buttons (to designate the desired control memory location, if different from the one indicated).
   c. Depress CONTROL DISPLAY.

2. To display characters from successive memory locations:

   a. Depress the STOP button (if not already in Stop mode).
   b. Enter the starting address in MAR manually (by depressing the appropriate ADDRESS buttons) or from control memory.
   c. Depress CONTROL buttons (to designate control memory working location, if different).
   d. Depress CONTROL ENTER (to set control memory location to starting address).
   e. Depress CONTROL DISPLAY + 1 repeatedly.

3. To write characters into successive memory locations:

   a. Depress the STOP button (if not already in Stop mode).
   b. Enter starting address in MAR manually, or from control memory.
   c. Depress CONTROL buttons (to designate control memory working location, if different).
   d. Depress CONTROL ENTER (to set control memory location to starting address).
   e. Iterate the following sequence:
      Depress CONTROL DISPLAY + 1.
      Depress CONTENTS CLEAR (if a different character is to be entered into this location).
      Depress CONTENTS buttons (to creat the desired character, if different).
      Depress CONTENTS ENTER.

4. To trace program flow manually:

   a. Depress the STOP button (if not already in Stop mode).

Set CONTROL buttons to designate sequence counter.

b. Iterate the following sequence:

Depress CONTROL DISPLAY (to see the contents of the sequence counter and the op-code).

Depress INSTRUCT (to execute the instruction).

5. To perform a Bootstrap operation and run:

a. Depress the STOP button.

b. Depress the INITIALIZE button.

c. Depress ADDRESS buttons (to designate starting location, if different from zero).

d. Depress CONTENTS buttons (to designate peripheral control, if different from zero).

e. Prepare the designated peripheral device to accept a read instruction which will read in the loading program.

f. Depress BOOTSTRAP button.

g. Depress ADDRESS MODE button (only if loading program is written for 3-character addressing).

h. Depress RUN button.

III. PANEL INDICATORS

A. Parity Indicator

If a parity check is stored, the Parity indicator will be lighted.

B. Power Indicator

If the H-200 has ceased operation because of overvoltage or undervoltage of a dc power supply, the Power indicator will be lit and dc voltages will have been removed. The DC OFF indicator will be lit and the DC ON extinguished.

The Power indicator may be reset by depressing the DC OFF button.

C. Fan Indicator

If the H-200 has ceased operation because of a temperature check, the Fan indicator will be lighted and dc voltages will have been removed. The DC OFF indicator will be lit and the DC ON extinguished. This indicator remains lighted even when dc and ac power are off.

D. CB Indicator

If the H-200 has ceased operation because of a circuit breaker actuation, the CB indicator will be lit and dc voltages will have been removed. The DC OFF indicator will be lit and the DC ON extinguished.

E. Interrupt Indicator

This indicator is present only if the particular central processor includes the Interrupt option. It is illuminated when the central processor is in the interrupt mode, and not otherwise. It can be turned on only by an external interrupt. It can be turned off by execution of a Resume Normal Mode instruction, or by depressing the INITIALIZE button.

IV. CONTROL PANEL LOGIC

Refer to control panel flow charts. (Pages 111 thru 116)

A. Entering New Information into Control Memory

1. Enter C. M. Address into CONTROL switches.

   a. The control switches are alternate action switches. Depress to set, and depress to reset.

2. Depress ADDRESS CLEAR

   a. Depressing ADDRESS CLEAR (CPCSB1C) while in the stop mode forces CPARS10 low. ARS going low kills the 'S' register recirculation function CAHSR10. HSR is normally high in stop mode.

   b. The transfer from control memory to 'S' is inhibited by CBBMF10, which forces low the gating functions CBICM10 and CBICL10.

$$\begin{aligned} \text{CBBMF10} &= \text{HSR} \\ &+ \overline{\text{ARS}} \\ &+ \text{BMF}\bullet\overline{\text{T07}} \end{aligned}$$

   $\overline{\text{ARS}}$ causes HSR to go low (to reset S) but while HSR is low, $\overline{\text{ARS}}$ keeps BMF high keeping ICL and ICM low.

3. Enter the new information into the ADDRESS switches.

   a. The ADDRESS switches are of the momentary contact type. They are gated with CBLSC1A directly into the S register. LSC is high every time 6 during the stop mode. Since HSR is high the S register will recirculate.

4. Depress the ENTER button.

   a. Refer to Page 112 of the flow charts, and the timing chart at Fig. 12-2.

   b. While in stop mode COP comes high at time four, and synchs the 'ENTER' button function CPKEB1C on the input to CPKRF10, which recirculates until time three. KRF will set every time four (for this operation) so long as CPKEB1C is high. CPKRS10 is just a delayed version of CPKRF20.

   When CPKRF40 goes low a peripheral response is forced (see inputs to C.P. cycle flops), allowing the R register to be set to the R switch setting.

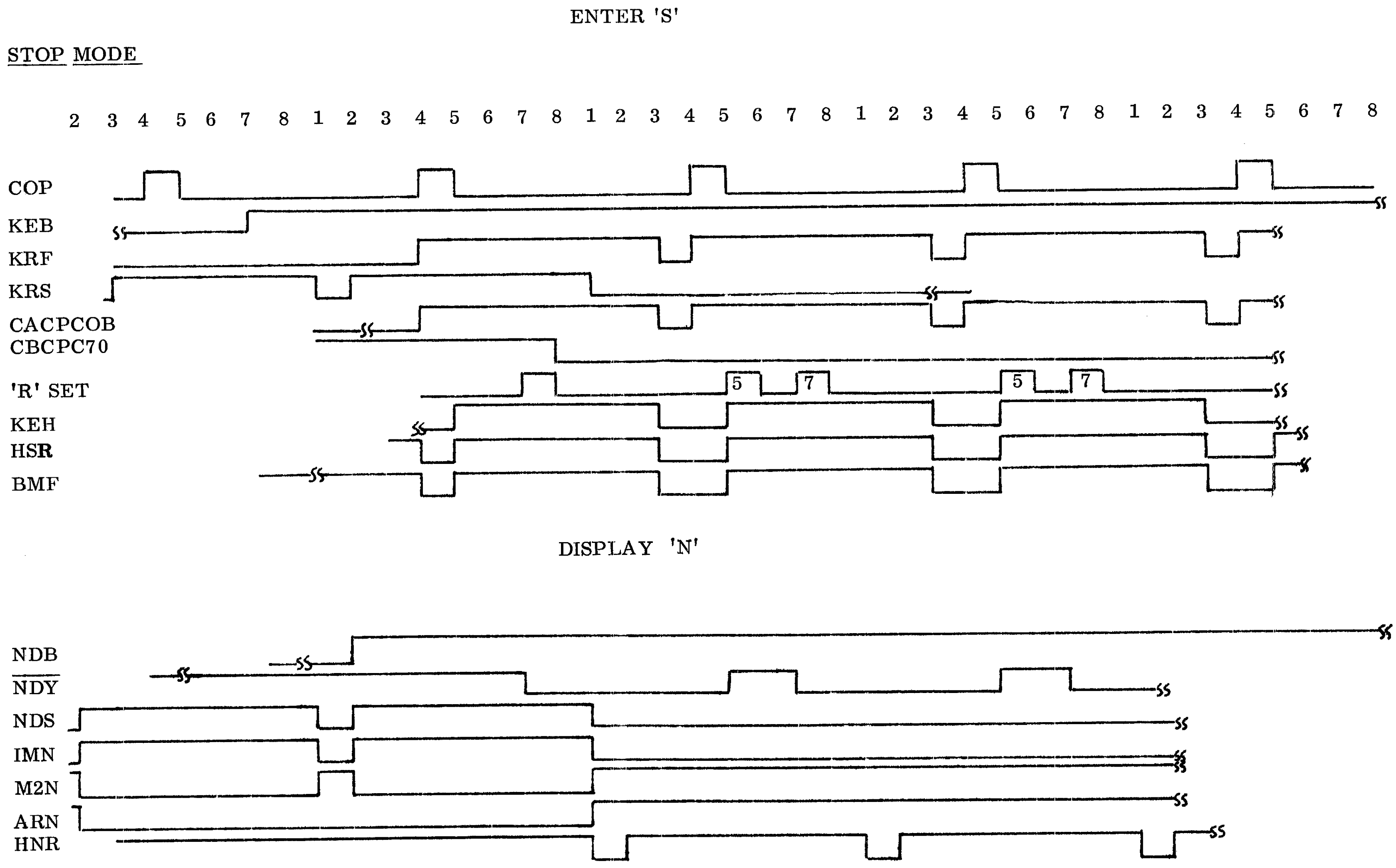

ENTER 'S'
STOP MODE
2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8
COP
KEB
KRF
KRS
CACPCOB
CBCPC70
'R' SET
5
7
5
7
KEH
HSR
BMF
DISPLAY 'N'
NDB
NDY
NDS
IMN
M2N
ARN
HNR

Figure 12-2

R01 = T07•CPC•R1P
R1P = (S) R01•KRF

R01 = $\underline{\text{T05}}$•$\underline{\text{R15}}$
R15 = CPC•CD1
CD1 = R1P•T02

R02 = T07•CPC•R2P
R2P = (S) R02•KRF

R02 = $\underline{\text{T05}}$•$\underline{\text{R25}}$
R25 = CPC•CD2
CD2 = R2P•T02

R03 = T07•CPC•R3P
R3P = (S) R03•KRF

R03 = $\underline{\text{T05}}$•$\underline{\text{R35}}$
R35 = FEX•CPC

R04 = T07•CPC•FRS
FRS = (S) R04•KRF

R04 = $\underline{\text{T05}}$•R45
R45 = CPC•R4P
$\underline{\text{R4P}}$ = T01•FRC
FRC = (S) R04•KRF

d. At T05 and T06, S is gated to Z. Since both ISR and DSR are low at this time, the M register is reset, and the S register is placed directly into Z. Writing from Z at T06 had been inhibited because of IYB on CBIYZ10. But IYB is forced low because of CBCPC70. Therefore at T06 "write from Z is allowed and the new information is written into the desired C.M. location.

e. At T07 the R register is again set to the switch settings, but C.M. sense amp transfer to S is inhibited due to ICL and ICM being held low by BMF, which is forced by HSR. Note, however, that HSR is now held high only by CPKEH10, since $\overline{\text{KRF}}$ is low. KEH is simply the "enter button," KEB, synched to the C.P. times. When HSR goes low, S recirculation is maintained by $\overline{\text{T08}}$ at pin 32.

f. The CONTENTS indicators will show the main memory contents for this new address. The logic discussion is at IV-C-1B.

B. Displaying the contents of a location in control memory

1. Enter C. M. address into control switches.

   a. See paragraph IV-A-1 and IV-A-4

2. Hit DISPLAY button.

   a. There are three display buttons for control memory.

      CPKYB1C - Display
      CPK1B1C - Display + 1 (Display address then increment)
      CPKMB1C - Display - 1 (Display address then decrement)

   b. For display only, KYB sets KRF and, so long as the button is held, KRF keeps coming high. The action is similar to that discussed in IV-A-4. The main difference is that HSR goes low with KRF coming high at T04. This allows BMF to go low, and this then allows ICL and ICM to come high and gate the control memory sense amps into S at T01 (normal S clear at T08).

      The M register is zero's because there is no gate to set the increment function ISR, and the decrement function is held low by CPIDS10. IDS is set by CPADS10 (high during stop mode) and CPKM100 which is forced low only by KMB the "display-1" button. Therefore the transfer from S to Z is done with no modification and the content of the displayed control memory location is not altered.

   c. If the display +1 (K1B,)or display -1 (KMB) buttons were hit, the display of the addressed control memory location would occur as presented in the previous discussion, but now the S register will be incremented or decremented before being written back into control memory. There is one small change in the display logic when K1B or KMB is punched, and that is that KRF is allowed to come high only once instead of being allowed all the time that an input (display or enter buttons) is held high. This must be done because if KRF were allowed more than once, more than one "buffer" cycle would be allowed and the S register would be augmented again for every "buffer" cycle.

d. The function CPKDM10 is set by KMB or K1B. KDM is anded with CPC0P10 on the input to KRF.

CPKRF10 = KDM•C0P
CPC0P10 = T04•$\overline{\text{C0S}}$•STP
CPC0S00 = I = KDM•KRF
CPC0S10 = I = $\overline{\text{C0S}}$ + T08•ACH•S0R

$\overline{\text{C0S}}$ will be high initially, allowing C0P at T04. C0P will then allow KRF to be set and recirculate until T03. But KDM and KRF will set C0S, which cannot be reset until S0R comes high. CPS0R10 is held low by KDM, and KDM is high all the time KMB or K1B is held depressed. This then allows only one KRF every time K1B or KMB is depressed.

e. K1B (display +1) is gated with KRF on the input to CAISB10, a backup function for ISR. Thus the display +1 button directly forces an increment.

KMB (display -1) forces CPKM100 low. KM1 had been used to keep CPIDS10 high. It is the function of IDS to keep CBDSR10 low during stop mode. So if KM1 is forced low, preventing IDS from being set, and thereby allowing DSR to come high, a decrement is forced.

f. Note that HSR is low only during KRF time (T04 through T03). At T08 the S register is reset and then loaded from control memory at T01. Normal recirculation ($\overline{\text{T08}}$) holds this information until HSR comes high again at the following T04. If at this time one of the display buttons is hit again the result of the previous agument will be seen on the S indicators. If the button hit had been one of the augmenting display buttons, the S register would be augmented again and the result of this second augment could be seen by hitting a display button again.

g. When the ADDRESS DISPLAY buttons are depressed the CONTENTS indicators will show the contents of the main memory address displayed in the ADDRESS indicators. The logic discussion for this is at IV C-1B.

C. Displaying A Main Memory Location

1. Depress the CONTENTS DISPLAY buttons.

a

M2N = I = CPIMN10
IMN = STP•KRS•CPNDS10
NDS = CPNDY00•T02 (Recirculated through T01)
NDY = (I) = STP•T07•CPNDB1C (Recirculated through T05)
NDB = 'N' Display Button

Refer to Fig. 12-1. Normally during stop mode N is recirculated, but NDB causing $\overline{NDY}$ to go low will allow CPARN10 to come high, and at T01 the N register is reset. Main memory is continously cycling so at T05 the N register is set from the M. M. sense amps with M2N. The main memory location interrogated is specified by the S register. This same location is cycled so long as the S register is not changed. $\overline{NDY}$ is released. When $\overline{NDY}$ does come high (just after the N register was set from main memory sense amps), CPARN10 is forced low, thereby keeping CAHNR10 high and recirculating the N register. $\overline{NDY}$ will also kill M2N to prevent clobbering thc N register if the S register is changed (addressing another location).

b. When displaying an address stored in control memory, or entering a new address into control memory, the contents of main memory for that address is displayed in the CONTENTS indicators. Entering information, or displaying a control memory location will cause CPKRF10 to come high. KRF coming high will force CPKRS10 low. $\overline{KRF}$ will allow CPARN10 to come high to allow clearing the N register at T01. KRS being low causes CPIMN10 to go low, which allows CAM2N10 to come high. M2N will gate the sense amps into the N register at T05. Thus any control panel display button and the control enter button will cause the contents of the main memory address displayed in ADDRESS indicators to be displayed in the CONTENTS indicators.

D. Entering New Information Into Main Memory

1. Depress CONTENTS CLEAR

   a. The CLEAR button CPCNB0C will allow CPARN10, thus allowing the N register to be cleared by HNR at T01. Information is set into the CONTENTS switches on the control panel, and the switches are gated into the P register all during the stop mode.

2. Enter new information into CONTENTS switches.

   a. The switches for the two punctuation bits are gated directly into N07, and N08. During the stop mode the sub-command CAPTN10 is always high, so the information bits from the P register are gated directly into N .

3. Depress CONTENTS ENTER button.

   a. Until now M. M. has been continually cycling and writing from the sense amps. But when the CONTENTS ENTER button (CPNEB1C) is hit, CAFRN10 is forced high, forcing CAINL10 high. During stop mode INL coming high forces NLD, N7D, and N8D. These last three functions allow the entire N register (as set by the control panel switches) to be written into memory.

E. Change Address Mode

1. Depress ADDRESS MODE button (CPCAB1C).

   a. As long as the ADDRESS MODE button is depressed, while in the stop mode, CACAB10 will be high.

      $\overline{\text{C0S}}$ = I = T07•CAB
      C0S = I = T08•ACH•S0R
      S0R = I = CAB

      When CAB is depressed $\overline{\text{C0S}}$ goes low at T07 and sets C0S, which cannot be reset until CAB is released because CAB is holding the reset function S0R low. The flop CPAMS10 is storing the present address mode, and at T07 when C0S is set, and CAB is true, it will force the character mode flop, CPCM210, to change state.

   b. When CAB is released S0R is allowed to come high to reset C0S at T08. (This kills the gates on CM2. C0S at T04 will now allow CPAMS10 to be set to the same state as CM2. Thus the character mode is changed, and the new state of CM2 is stored in AMS. Note that $\overline{\text{C0S}}$ is normally high during run mode, so that if the change address mode instruction is issued and changes CM2, then this change is also stored in AMS.

F. INSTRUCT Button

1. Stepping through a program one instruction at a time is accomplished by depressing the INSTRUCT button (CPINB1C). When in the stop mode, depressing the INSTRUCT button will start the machine just as though a normal start had been executed. In addition, the first EAC cycle will set CPACL10, which will set CPSV310 in the next V3 cycle. SV3 will cause the machine to stop after executing one instruction.

2. CPACH10 (I) = CPINB1C
   CPACR10 (I) = CPCOP10

   ACR will kill recirculation on ACL, allowing ACL to come high. The timing chart on page 12-18 of the flow charts shows the sequence of events for a start - stop. (For a detailed explanation of start-stop see page 12-18. The only real differences are that run is set by another gate;

   CPRUN10 = INB•$\overline{\text{ACH}}$•C0P

   And after run allows an EAC cycle, another gate is used to set ACL.

   CPACL10 = INS•EAC•T04
   CPINS10 = INB•C0P•$\overline{\text{ACH}}$
   + INS•HFC•CPCCL00

3. ACL will allow SV3 to set when the instruction has been completed.

   CPSV310 = ACL•DVC•E3C•T03

   SV3 will kill CPRUN10, preventing another instruction from being executed.

4. If the single instruction initiated had been a peripheral order, the machine would still go into a V3 cycle stall. But CPSTP10 could not be set until the peripheral

device releases the R/W channel to which it is tied. $\overline{\text{STP}}$ allows the data transfer to be completed, and $\overline{\text{RUN}}$ prevents the extraction of another instruction from main memory.

G. Bootstrap

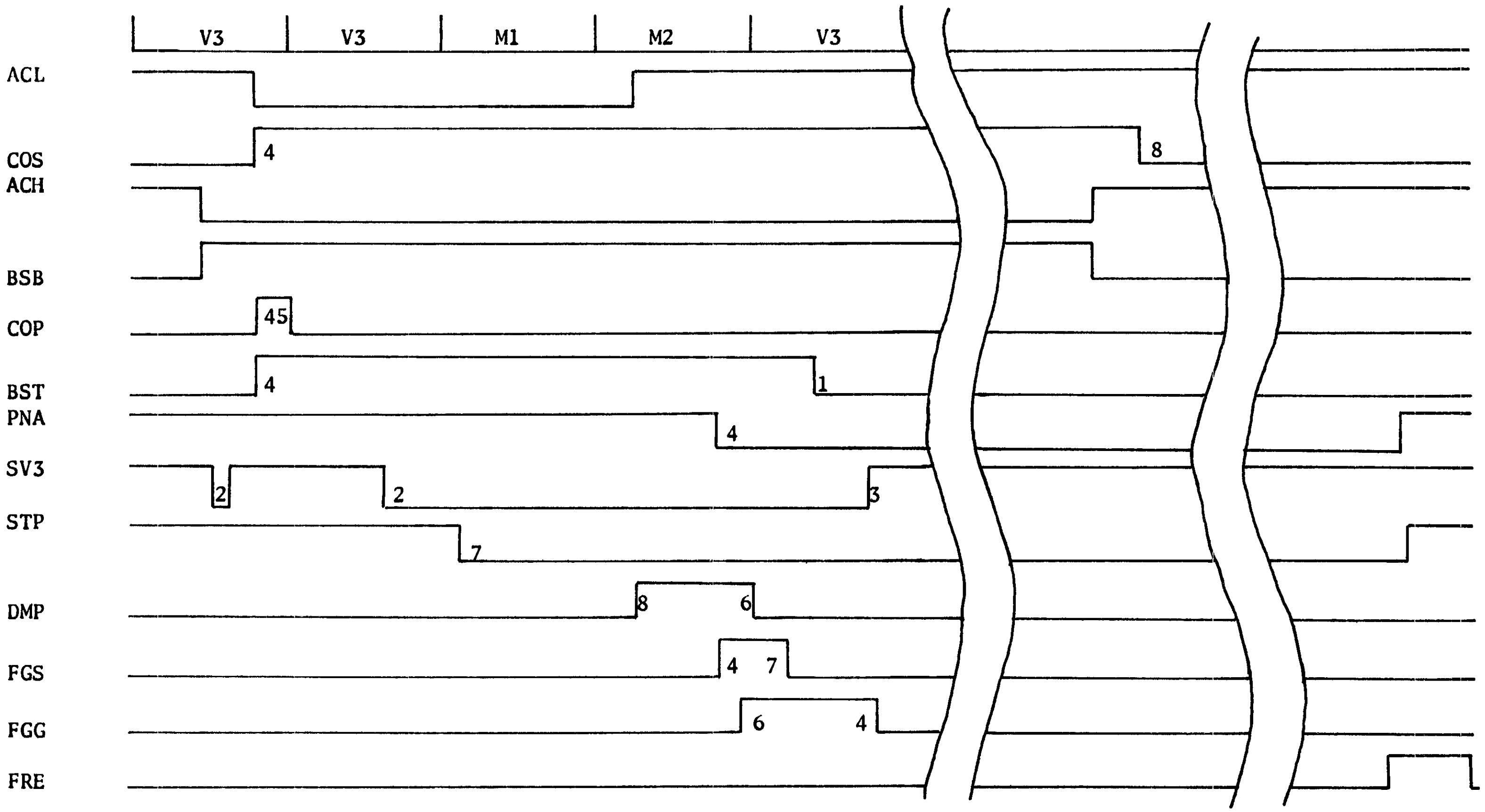
1. Depress BOOTSTRAP Button.

a. To enter a program into memory requires a small loading routine. But how is this routine entered into memory? Punching in the instructions on the control panel would be time consuming and tedious. A much better solution would be to force a peripheral read instruction. This, in fact, is exactly what is done. The operation is called a "bootstrap", because it allows reading one record from a peripheral input device with no external intervention other than pushing the bootstrap button. Of course, if the operator wants to know where the record will be entered in memory and what input device will be used, he had better specify this information. The bootstrap routine assumes that it will be loaded into a particular area of memory, so the starting location of the input area must be specified by entering this address into the ADDRESS switches. While the starting address is in S, the trunk number of the input device to be bootstrapped is entered into the N register by means of the CONTENTS switched.

b. Refer to the bootstrap timing chart (Figure 12-3) and flow charts. In general, when the BOOTSTRAP button is hit, the M1 cycle of a peripheral data transfer is forced. But during the last C.M. cycle of the V3 cycle, previous to the M1 cycle, the contents of the S register (unmodified because of $\overline{\text{ISR}}$ and $\overline{\text{DSR}}$) is written into the sequence counter. The bootstrap function CPBST10 is used to force most of the conditions to allow this modified PDT instruction. A complete load list for BST appears in Figure 12-3. Refer to page 114 of the flow charts. Note the S to Z transfer, and writing of Z into C.M. location 17. The N switches are shown feeding N07, N08 and the P register. In the stop mode there is an unconditional transfer of P to N every time 6, but in stop mode and during BST, N cannot be reset, and the sense amps write into memory. IBP (inhibit bootstrap punctuation) forces IN7 and IN8, insuring that punctuation is written from N during bootstrap. Since N07 and N08 are cleared during information transfer, all punctuation in the input area is also cleared. IBP will be reset after the RWC is released.

c. The RWC #1 starting location counter is forced into the V register by BST. This is to simulate the first variant character. BST will allow CPSTP10 (stop flop) to go low, but CPRUN10 is low all during the operation. $\overline{\text{STP}}$ is necessary for a peripheral transfer, but since only M1 and M2 cycles are required, CPRUN10 is not necessary.

BOOTSTRAP



BST SETS		BST RESETS
N07	PRW	HST → STP
BAC	SMA	HSR
COS	R3S	
ARN	PDA	V03
IMN	NG7	V02
V06	I01 ←	
V04	DLA ←	II1
V01	IBP	S3H
FGE →	FGG	DSR

Figure 12·3

d. On page 115 of the flow charts the M1 cycle is represented. FDD is generated to gate the trunk address that had been transferred from N to F, to the peripheral control units. Memory read out to N is inhibited all during the time of BST, so the trunk address is not destroyed. Location ten in the second C.M. cycle is used as temporary storage for the starting address before writing it into the location eleven in the third C.M. cycle. In the last C.M. cycle the sense amps are written back in, even though the contents of Z have not been modified $(\overline{ISR}\bullet\overline{DSR})$.

e. M2 cycle on page 116 shows how the PDT is terminated by forcing N07 to a one with BST and DM2. N07 will set FGG the peripheral go signal. NG7 (set with BST) gates V (RWS #1) to the peripheral control unit. NG7 also sets FGE, which forces the channel no. 1 active flop (because of contents of V). The channel active flop (F1A) resets the peripheral active collector, (PNA) allowing BST to be reset, and preventing CPSTP10 from being set when CPSV310 is set in the next cycle.ACL, the allow clear function, was set by CPBAC10 which in turn came high with DM2 (derived EMC and E2C).

f. Because C0S cannot be reset until ACH (held low by the bootstrap button) comes high, and $\overline{C0S}$ is required to set C0P, then BST can come high only once each time the bootstrap button is depressed.

g. The next cycle is V3. It cannot be followed by EAC since CPRUN10 never was set. CPACL10 was set and recirculated however, allowing CPSV310 to be set during this V3 cycle. SV3 will insure a V3 stall, and will allow CPSTP10 to be set at the end of data transfer.

h. When the bootstrap terminates (stop light will come on when the R/W channel is released), the starting location that was punched into the ADDRESS switches will be stored in control memory locations 17, 10, 14, and 11 (sequence counter, B address counter, A address counter, and R/W channel #1 starting counter). Loading the A and B counters allows bootstrapping into any module of memory, while still using two char. addressing in the loading routine.

Stop Mode

Figure 12-4 shows the timing of the functions necessary to stop the central processor from the operator's control panel. When the stop button is depressed (CPSTB1C) the central processor will complete the instruction being currently processed and, if no peripheral channels are active, stall in a V3 cycle. The stop button sets ACL, which will recirculate on ACR. ACL will allow SV3 to come true at time three of the next V3 cycle, SV3 will reset RUN and inhibit the recirculation on all major and minor cycles except DVC and E3C. RUN being low will not allow the setting of EAC which is the normal next state condition for a V3 cycle. The central processor is now considered to be in the stop mode. The subcommands N21 (N register to I register transfer) and ISR (Increment S register) which normally are high during a V3 cycle are inhibited during the stop mode. N2I is inhibited by RUN.

N2I = N4I•T07•IMT   N4I = PU3•ITF•CT6•RUN

ISR will be low, also because of RUN.

ISR = DFC•$\overline{\text{EXC}}$•T08•RUN

However, ISR feeds DSR (Decrement S register) which is an invertor, so it is necessary to provide another gate on the input to DSR to inhibit decrementing S. CPIDS10 (inhibit decrementing S) will serve this purpose.

IDS = T01•$\overline{\text{KM1}}$•ADS

KM1 (I) = KMB (Console Display -1 button)

ADS (I) = $\overline{\text{STP}}$•$\overline{\text{FR1}}$•$\overline{\text{FR2}}$•$\overline{\text{FR3}}$

+ $\overline{\text{STP}}$•CPC

It is sometimes said that during the stop mode the CP is performing "dummy V3" cycles, since the clock still runs and memory is cycled but nothing is accomplished.

200 STOP - START

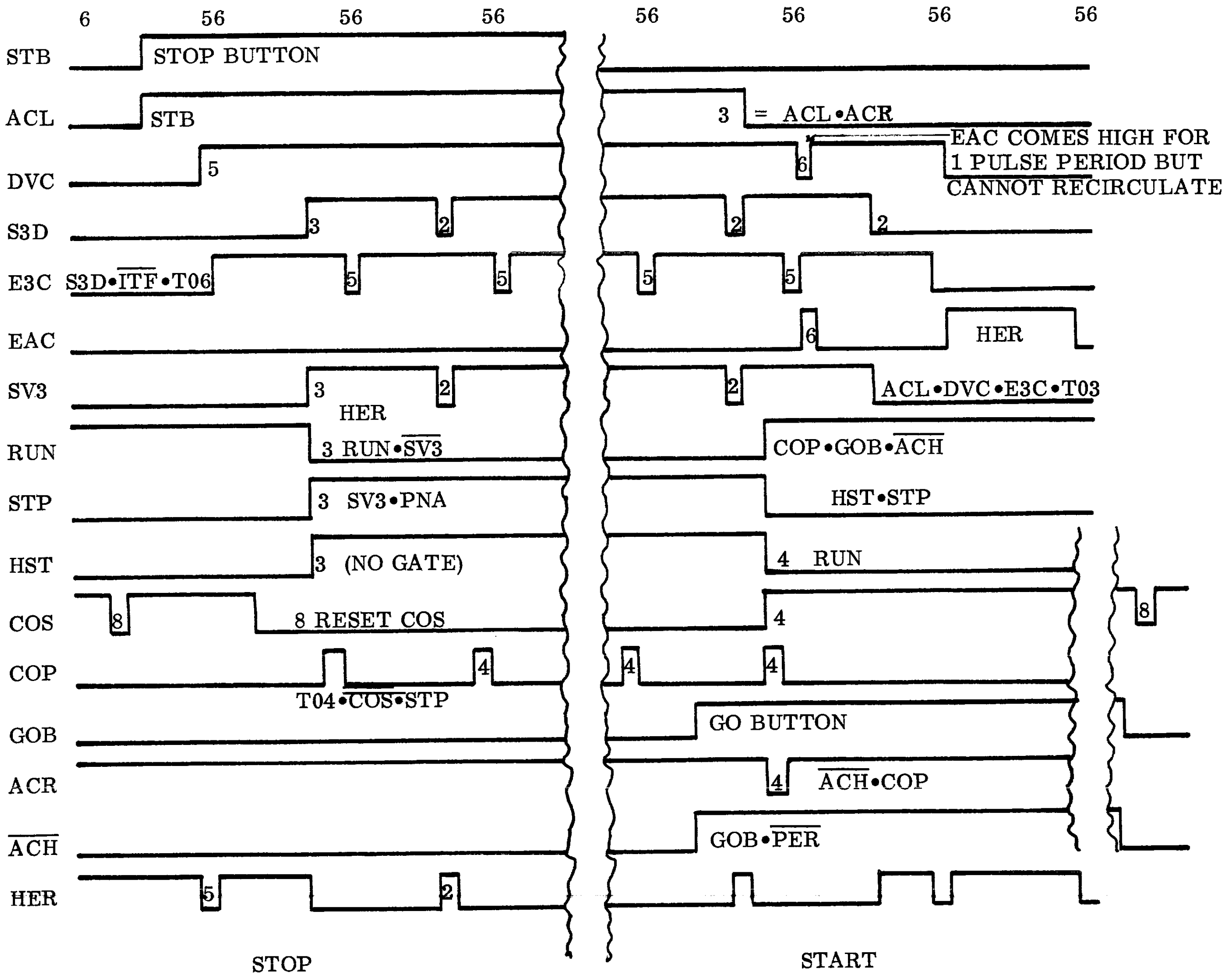


Fig. 12-4

HONEYWELL
ELECTRONIC
DATA
PROCESSING
WELLESLEY HILLS,
MASSACHUSETTS 02181