IBM Customer Engineering

Instruction-Maintenance

7094 II Data Processing System–Volume 1

Introduction

Component Circuits

System and Functional Components

Timing

IBM® Customer Engineering

Instruction-Maintenance

7094 II Data Processing System—Volume 1

Introduction

Component Circuits

System and Functional Components

Timing

Preface

This manual contains four sections: IBM 7094 II Data Processing System, Component Circuits, System and Functional Components, and Timing.

The purpose of these sections is to:

1. Introduce the 7094 II system.
2. Describe the various circuit cards and circuit configurations.
3. Explain briefly the purpose and operation of the various system components (such as the multiplexor, core storage, data channels, etc.), and internal functional components (such as registers, adders, counters, etc.).
4. Explain the 7094 II timing and cyclic makeup.

Customer Engineers who are familiar with the 7094 system may easily skip over the first section by reading only the portion on Instruction Overlap.

Condensed logic diagrams used in this manual are as close to actual systems as possible. Most of these diagrams have been converted to positive logic by eliminating any references to + or − levels. In maintaining this positive logic, in-phase outputs are used to indicate an active (conditions met) state from the condensed logic block. Out-of-phase outputs are also used in some cases to simplify the diagrams by eliminating the cluttering effect of convert and invert blocks.

The material in this manual is written at engineering change level 253407; however, future engineering changes may change the logic and machine operations from the presentation in this manual.

The following manuals pertain to the 7094 II system:

FORM	TITLE
223-2721	7094 II DPS CEIM Manual Volume 1 (Introduction, Component Circuits, System and Functional Components, and Timing)
223-2722	7094 II DPS CEIM Manual Volume 2 (Arithmetic Instructions)
223-2723	7094 II DPS CEIM Manual Volume 3 (Non-Arithmetic Instructions, Overlap, Trapping, Compatibility, 7151-2 Console)
223-2724	7302-3 Core Storage CEIM Manual
223-6910	7607 Data Channel CE I-R Manual
223-2551	7909 Data Channel CE I-R Manual

Safety

The following safety practices should be observed:

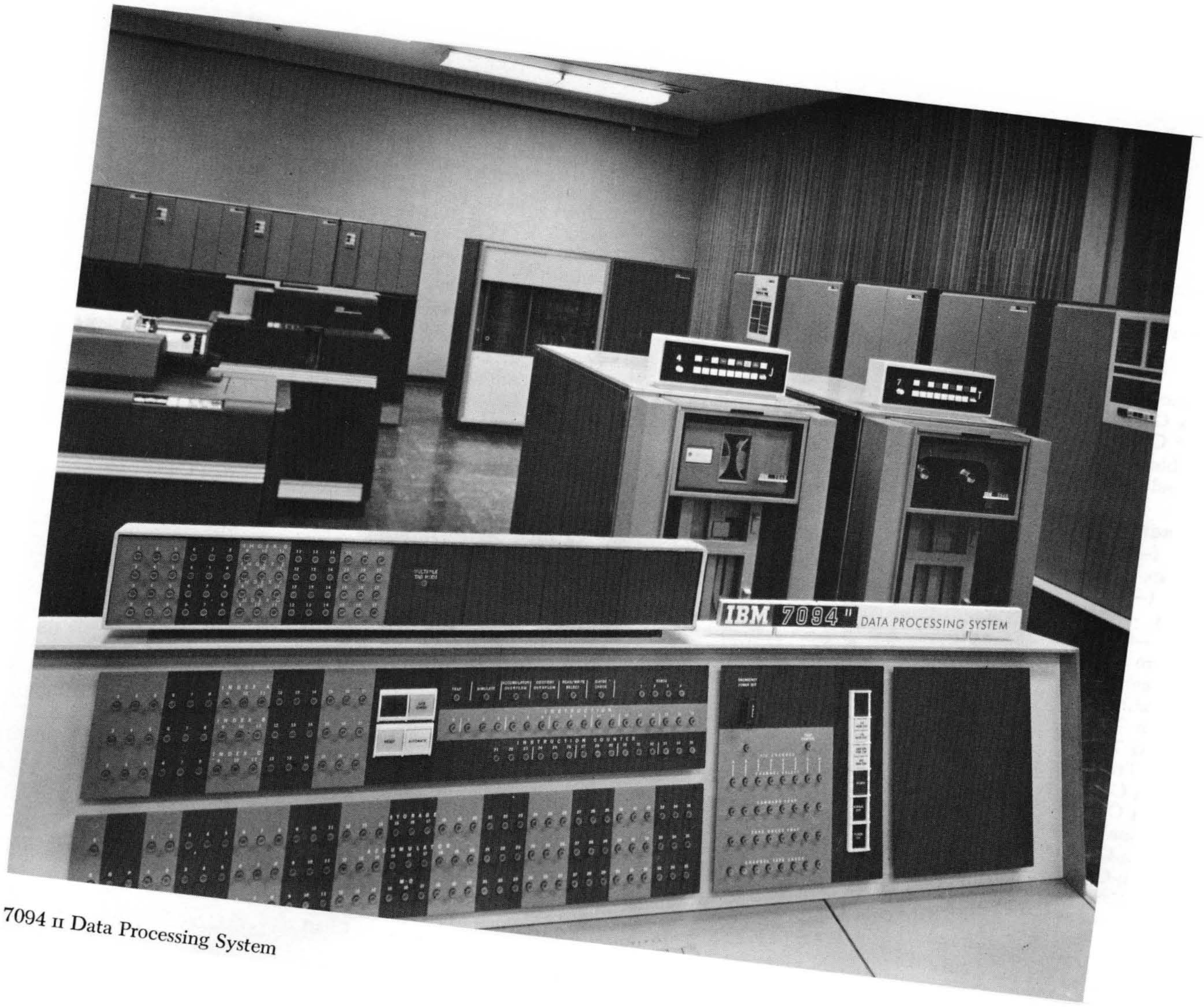
1. At least two men should be within sight of each other when working on a machine with power on.
2. Safety glasses must be worn when soldering or performing other operations which may endanger the eyes.
3. Use caution when lowering a tailgate. Keep fingers clear of gate slides when sliding a gate into a module. Avoid hitting laminar bus connections.
4. 120 volts, 60 cycles, and 48 vdc are still present inside SMS frame with frame power off and 7618 power on. If it is necessary to work near live power connectors, convenience outlets, or inside the MG unit or core storage control, disconnect power cables, or turn off wall circuit breakers.
5. Discharge capacitors before working on DC power supplies.
6. Always turn off power before replacing a fuse.
7. Replace safety covers that have been removed before proceeding to another operation.
8. Prior to servicing, note and check the following items:

   Master power switch location ________________
   Air conditioning switch location ________________
   Fire extinguishers (CO_2 type) ________________
   Emergency exit doors location ________________
   Fire control phone number ________________
   First aid phone number ________________
9. Remove metal jewelry before servicing the computer.

Copies of this and other publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Corporation, CE Manuals, Dept. B96, PO. Box 390, Poughkeepsie, N.Y. 12602.

Contents

Illustrations



7094 II Data Processing System

Modern methods of accounting, measuring, testing, research and design generate huge quantities of information that must be processed quickly and accurately. A vast amount of data constantly pours into such places as retail establishments, weather stations, insurance companies, and tax bureaus. In addition, our rapidly expanding scientific investigations into rocket and missile design, atomic research, and missile tracking need faster and faster methods for carrying out increasingly complex calculations. To meet these demands, machines have been developed which can compute, select and correlate data at electronic speeds.

Automating paper work is possible because the actions involved are sufficiently repetitive. The variety of steps necessary in processing business records or in computing scientific problems, for example, is small, indeed, compared with the number of times these steps must be taken. One of the first paper work machines was the ink stamp, possibly because applying a date or name was so obviously repetitive. As additional mechanization was applied to paper work, machines began to take over the long and painstaking task of accounting.

Although scientific applications of computers require a certain amount of arithmetic, such as accumulating partial results or totals, the problem is principally one of processing data. A large amount of information is fed into these machines (input) and a relatively small amount of information is produced (output). The machines are, therefore, called data processing machines.

The first data processing machines handled information in a series of individual operations. These included punching information into cards, sorting and classifying cards, producing totals and balances, and printing the results. Intermediate results from one machine were transferred to another; many human decisions and interventions were necessary for a complete accounting procedure.

With the application of electronics, the rate of calculation was vastly increased. But more important, a basic new technique was introduced which might be called intercommunication. Electronic devices were able to make decisions, and, on the basis of these decisions, to provide internal transporting of data and intermediate results from step to step. Information (data) is fed into one end of a data processing system and final results come out the other. This machine system, as we know it today, is the modern computer, or data processing system.

The five functional sections of a generalized computer are illustrated in Figure 1. The advantages to be realized by using a computer include greatly increased processing speeds, a high degree of automation, and great flexibility.

All information used by the computer must pass through the input section where the incoming information is interpreted and converted to the language that the computer understands. The input section includes such devices as card readers, magnetic tape units, disk files, etc.

From the input section, information is directed to the storage section. As their main storage unit, most computers use an information-holding device composed of magnetic cores. This magnetic core storage may serve as the source of all information to be used by the computer. Core storage has some very important advantages. Most important of these is the high speed at which information may be placed in, or removed from, core storage. The highest degree of performance from core storage and many other types of storage can be realized only when the information is arranged in specific order. Once the information is located in core storage, it may be called for instantly in any sequence.

The control section of a computer directs the operation of the entire computer, receiving its directions

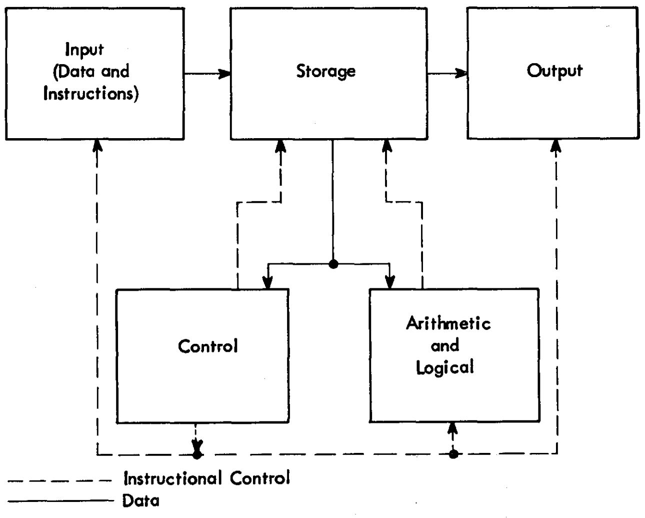


Figure 1. General Computer Functional Arrangement

from units of detailed information from core storage. These units of information, which tell the control section what operations are to be performed, are called instructions. That portion of the information in storage which is to be operated on is commonly referred to as data. A single piece of data used in an operation is often called an operand. From the above, it may be seen that instructions as well as data must be delivered to storage from the input section. Note in Figure 1 that the control section receives instructions from storage and then exhibits the necessary control over all other sections of the computer.

The actual operations are, for the most part, performed in the arithmetic and logical section. The control section directs exactly what operation is to be performed and what operand is to be involved in the operation. The instructions that may be executed by a given computer include such arithmetic operations as add, subtract, multiply, or divide. Some instructions may place the results of the arithmetic operation back in core storage. Subsequent instructions may tell the control section to deliver the information to the output section which may include printers, punches, magnetic tape units, or a variety of other I/O devices.

From the above description, it can be seen that a single instruction causes only a specific operation to be performed by the computer. If a complete problem is to be performed by the computer, a number of instructions are required to direct the computer. A group of such instructions, and any necessary constant data used to direct the computer to the accomplishment of a job, is called a program. Because, in modern computers the program is contained in storage, the computers are called stored program computers. When operating under stored program control, the computer executes one instruction at a time. After executing one instruction, the computer automatically proceeds to the next instruction. It is important to realize that every computer must be directed by some type of program during every step of its operation.

A generalized stored program computer (Figure 1) operates in the following manner: A program of instructions to direct the computer in every step of its operation is stored on magnetic tape. All data upon which the computer is to operate are also stored on tape. The tapes are readied, and a key on the computer console is pressed which tells the control section that the information located on tape is to be read into storage. The control section then starts the tapes and delivers the information to the proper locations in storage. Information contained in the last tape record tells the control section where to find the first instruction. The control section then calls for and decodes the instruction to determine what operation is to be performed and where the operand that is to participate is located. Next, the control section causes the operand, also located in storage, to be delivered to the arithmetic and logical section. The arithmetic section then performs the operations as called for by the control section. After the first instruction has been performed, the control section calls for the next instruction from storage. This instruction may very well be one that causes the results of the previous instruction to be stored. This process continues until such time that an instruction is encountered that causes the results (now in storage) to be delivered to the output section.

Computer Words

Before a computer can be told what to do, a common language is necessary between programmer and computer. The 7094 II is a binary machine, so all inputs and outputs, internal processing, and internal communication is in terms of 1-bits and 0-bits. These 1's and 0's are combined in a 36-bit word in such combinations that are meaningful to the computer.

For example, the combination of 000 100 000 000, properly placed in the computer word, instructs the computer to perform an addition. Another portion of this particular 36-bit word (address portion) tells the computer which word in core storage is to be added (the operand). The next instruction might contain the combination of 000 110 000 001, which, in the proper location within the word, instructs the computer to store the sum just obtained. Again, the address portion of this word will instruct the computer where the sum is to be placed in core storage.

From the foregoing, two types of words are apparent —instruction words and data words. A third type exists, the channel command. Channel commands control the data channel during a particular operation, such as read or write. These three types of words (data, instruction and command) are arranged by the programmer into a logical sequence that will result in desired problem-solving or data processing.

A 7094 II word may be a numeric quantity, an instruction to the computer, or a data channel command. In all cases, the word contains a full 36 positions (or bits). The contents of the word become significant according to the computer cycle of operation. Thus, a word coming into the computer during an instruction cycle is treated as an instruction. A word coming into the computer during an execution cycle is treated as a numeric quantity.

Data Word

When the 36 bits are expressing a numeric quantity, the word is referred to as a data word. Figure 2 shows a data word. Note that the numerical value is expressed in positions 1 through 35, and the sign of the value is expressed in the S position. When S is a 0, the value is positive. When S is a 1, the value is negative.

Many logical operations, on the other hand, operate on full 36-bit data words. In these cases, the sign bit loses its special meaning and becomes just another bit of information in the full data word.

Instruction Word

A computer instruction word is shown in Figure 3. Because this word is coming into the computer during an instruction cycle, it will, in effect, be segmented, and the various segments will be interpreted to determine what action is expected of the computer. The 36 bits of the computer word are now broken into significant sections—prefix, decrement, tag and address.

The sign position is always a part of the operation code. The function to be performed is dictated by the sign and either the remainder of the prefix field or the decrement. If positions 1 and 2 contain zeros, the sign and decrement determine the operation code. If either or both positions 1 and 2 contain ones, the prefix contains the entire operation code and the decrement is used for another purpose.

The address field usually contains the address of a data word in core storage. This data word is brought into the computer as a part of whatever arithmetic or logic function is called for by the operation code. Thus, the instruction not only dictates the operation to be performed, but also specifies the address of the data to be used. In some instances, the address field is a part of the operation code. When this is the case, the address field is not used to address the data in storage.

The tag field causes the computer to either operate on a specific index register or modify the address field of the instruction.

Figures 4 and 5 are variations of the computer instruction word.

Instruction Addressing

In the 7094 II, an instruction can make reference to three types of addresses: direct address, effective address, or indirect address. Several hypothetical examples are shown below to illustrate how each of the three forms of addressing would obtain the same data word from core storage.

As a review of symbolic instruction coding, consider the clear and add instruction: CLA* Y, T, D.

where:

CLA	is the mnemonic instruction coding.
*	is the indication of indirect addressing.
Y	indicates the address portion of the instruction.
T	indicates the indexing portion of the instruction (tag) and may be omitted if no tag is specified.
D	indicates the decrement portion of the instruction and may be omitted if not necessary.

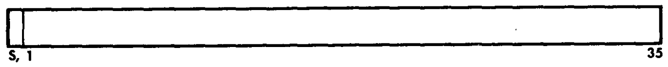


Figure 2. 7094 II Data Word

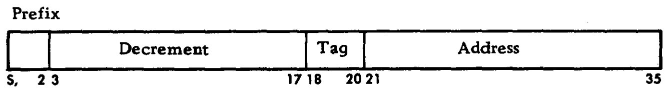


Figure 3. 7094 II Instruction Word

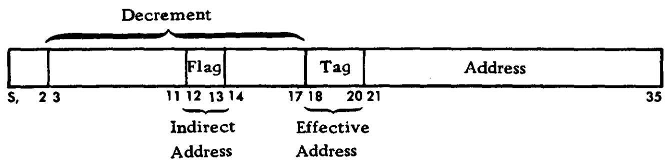


Figure 4. Instruction Word Address Modification Fields

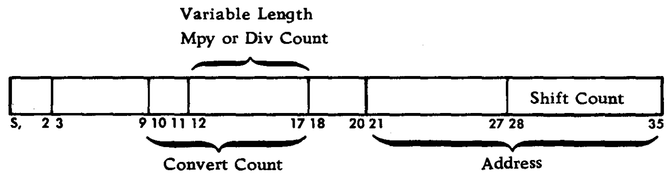


Figure 5. Instruction Word Count Fields

Direct Address

The direct address is the address specified by positions 21-35 of the instruction. Symbolic location DATA(Y) is the direct address shown in the example below. Execution of the CLA causes the contents of DATA (a numerical quantity in location 00002) to be moved from core storage and into the accumulator.

```
00000    START    CLA    DATA
00001    STOP     HTR    00000
00002    DATA     OCT    +000000001572
```

Effective Address

The effective address is the direct address modified by the contents of the specified index register (Y-T). In the example below, assume that index register 1 contains 00002. During execution of CLA DATA, 1 the content of XR1 is subtracted from the location indicated by DATA (00004−00002=00002). The computer makes reference to this new modified location (00002), reads the numerical quantity out of core storage and places it into the accumulator.

Assume XR1 = 00002

```
00000    START    CLA    DATA, 1
00001    STOP     HTR    00000
00002       −2    OCT    +000000001572
00003       −1    OCT    +000000000145
00004    DATA     PZE    0, 0, 0
```

Indirect Addressing

During indirect addressing, the effective address is determined (Y modified by T), and a core storage reference is made to this effective address. The address portion of this data word is then modified by its indexing tag (if specified); this second effective address represents the actual address that is to supply the instruction's data. Four situations can occur: no indexing, indexing at either address, and indexing at both addresses.

No Indexing: The CLA makes an indirect reference to location TABLE. In the address portion of location TABLE is the address represented by DATA. It is this latter address (0002) where the computer obtains the numerical quantity for the accumulator.

```
00000   START   CLA*   TABLE
00001   STOP    HTR    00000
00002   DATA    OCT    +000000001572
00003     +1    OCT    +000000000145
00004   TABLE   PZE    DATA
```

Indexing at the Instruction Address: The direct address of the CLA is modified to form an effective address of TABLE – 2 (00006 – 00002 = 00004). An indirect reference is then made to the address portion of TABLE – 2 (location 00004). The address portion of TABLE – 2, DATA (00002) indicates where the computer is to obtain the numerical quantity for the accumulator.

Assume XR1 = 00002

```
00000   START   CLA*   TABLE, 1
00001   STOP    HTR    00000
00002   DATA    OCT    +000000001572
00003     –3    OCT    +000000000145
00004     –2    PZE    DATA
00005     –1    PZE    0, 0, 0
00006   TABLE   PZE    0, 0, 0
```

Indexing at the Indirect Address: The CLA makes an indirect reference to TABLE (location 00010). The address portion of location TABLE is represented by symbolic location DATA (00005), and the tag specifies index register 2. Address modification is performed at this indirect address to produce the data address of DATA – 3 (00005 – 00003 = 00002). This new address of location 00002 indicates where the computer is to obtain the numerical quantity for the accumulator.

Assume XR2 = 00003

```
00000   START   CLA*   TABLE
00001   STOP    HTR    00000
00002     –3    OCT    +000000001572
00003     –2    OCT    +000000000145
00004     –1    OCT    +000000000256
00005   DATA    PZE    0, 0, 0
00006     –2    PZE    0, 0, 0
00007     –1    PZE    0, 0, 0
00010   TABLE   PZE    DATA, 2
```

Indexing at Both the Instruction and Indirect Address: The direct address of the CLA is modified to form an effective address of TABLE – 2 (00010 – 00002 = 00006). An indirect reference is then made to this new address. The address portion of location TABLE – 2 is represented by symbolic location DATA (00005) and the tag specifies index register 2. Address modification is performed at this indirect address to produce the data address of DATA – 3 (00005 – 00003 = 00002). This new effective address of location 00002 indicates where the computer is to obtain the numerical quantity for the accumulator.

Assume XR1 = 00002 and XR2 = 00003

```
00000   START   CLA*   TABLE, 1
00001   STOP    HTR    00000
00002     –3    OCT    +000000001572
00003     –2    OCT    +000000000145
00004     –1    OCT    +000000000256
00005   DATA    PZE    0, 0, 0
00006     –2    PZE    DATA, 2
00007     –1    PZE    0, 0, 0
00010   TABLE   PZE    0, 0, 0
```

7607 Data Channel Command Word

Similar in format and application to the computer instruction word, the 7607 data channel command word (Figure 6) gains its special significance by being called out of storage by the control function in the data channel. This, as in the computer, occurs when one operation has been completed and the data channel must be directed to perform the next operation.

After the channel operation is initiated by the computer, the 7607 data channel functions as an asynchronous unit under control of an I/O program. This I/O program, located in storage, is constructed from special instruction words for the channel called commands. These commands inform the channel as to how many data words to transmit, where to obtain the data words in core storage during writing operations or where to store the data words in storage during reading operations. Each command also includes control information which can indicate indirect addressing, non-transmission of data, and what to do upon completion of the command. The command word format is:

S, 1-2	Operations code (Informs the channel as to what operation is to be performed)
3-17	Word count (Maximum number of words this command is to transmit)
18	Indirect addressing flag
19	Non-transmission indicator (Read select operations only)
20	Not used
21-35	Starting address where data words are to be stored in core storage

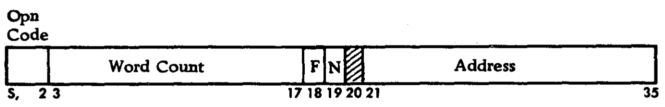


Figure 6. Data Channel Command Word

7909 Data Channel Command Word

The 7909 data channel commands are decoded in the channel's operation decoder. Five major bits define the command: S, 1, 2, 3, and 19. Note, however, that position 3 is located in the decrement portion of the word. Commands which do not require the decrement portion portion of the word can use this position for decoding purposes. Other channel commands which require either a full or partial decrement field cannot use this position 3-bit for operation decoding. Formats for these commands are shown in Figure 7 and use the following field nomenclature:

Y	Address
C	Count
M	Mask
F	Indirect Address Flag

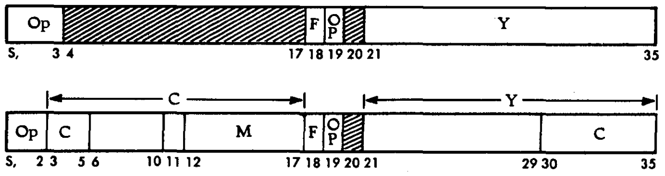


Figure 7. 7909 Command Word Formats

Instruction Overlap

The overall operational speed of the 7094 II is greatly increased by parallel execution (overlapping) of two sequential instructions.

While the "current" instruction is in the computer for execution, the next sequential instruction is also obtained from core storage. This second (overlapping) instruction is stored in the central processing unit, analyzed, modified, and partially (or completely) executed during the same E/L cycle that the current instruction is being executed. The 7094 II overlap capabilities are made possible by the new IBM 7302-3 Core Storage whose 32,768 storage locations are divided into two logically independent 16,384 sections.

Two types of overlap are performed in the 7094 II; "double instruction overlap," and "extended sequence overlap." The main difference between the two types is how and when the two instructions are received into the computer from core storage—both types, however, achieve the same basic objectives.

Overlap requires teamwork between the program register and the instruction backup register. Figure 8 shows two sequences of double instruction overlap. In Figure 8a, two instructions are obtained from core storage. The first instruction is placed in the program register for immediate execution; the next sequential instruction (overlapping instruction) is obtained from the other half of core storage and placed into the instruction backup register. As the current instruction is being executed, preliminary functions are performed on the overlapping instruction in the IBR. When the current instruction is completed, the overlapping instruction is transferred to the program register (and storage register) for completion (Figure 8b). This double instruction overlap occurs either initially in the program or immediately after an extended sequence series has been broken.

7094 II double instruction overlap is similar to 7094 overlap. The 7094 requires that the first instruction come from an even address; the 7094 II, because of the new split-memory, does not have this restriction.

After double instruction overlap has initially started the overlap operation, extended sequence overlap can take over. Figure 9a shows the two initial instructions being received from core storage. Figure 9b shows the second instruction being passed from the IBR to the program register (and storage register) as in double instruction overlap, but with one addition—the IBR is now also reaching for the third instruction. As long as conditions are favorable, the program register (and storage register) will continue to receive instructions from the IBR (Figures 9c and 9d). All I-time functions of the overlapping instruction are performed during II time (IBR I time) which is concurrent with E/L time of the overlapped instruction.

Conditions which can break the overlap sequence include: trapping, double-precision instructions, one-cycle instructions, and POD 76 instructions (except shifting instructions). The specific details concerning instruction overlap are covered in volume 3.

Binary Arithmetic

The binary system is used in computers because all present components are inherently binary. That is, a relay maintains its contacts either closed or open, magnetic materials are utilized by magnetizing them in one direction or the other, a vacuum tube is conveniently maintained either fully conducting or nonconducting, or the transmission of information along a wire may be accomplished by transmitting an electrical pulse at a certain time.

Although binary numbers in general have more terms than their decimal counterparts (about 3.3 times as many), computation in the binary system is quite simple.

The only convenient way to learn the operation of a computer is to learn the binary system. The octonary or octal system is a shorthand method of writing long binary numbers. Octal notation is used when discussing the computer but has no relation to the internal circuits.

Perhaps, as the first step, it would be well to see what is meant by the binary system of numbers. The binary, or base-two system, uses two symbols, 0 and 1,

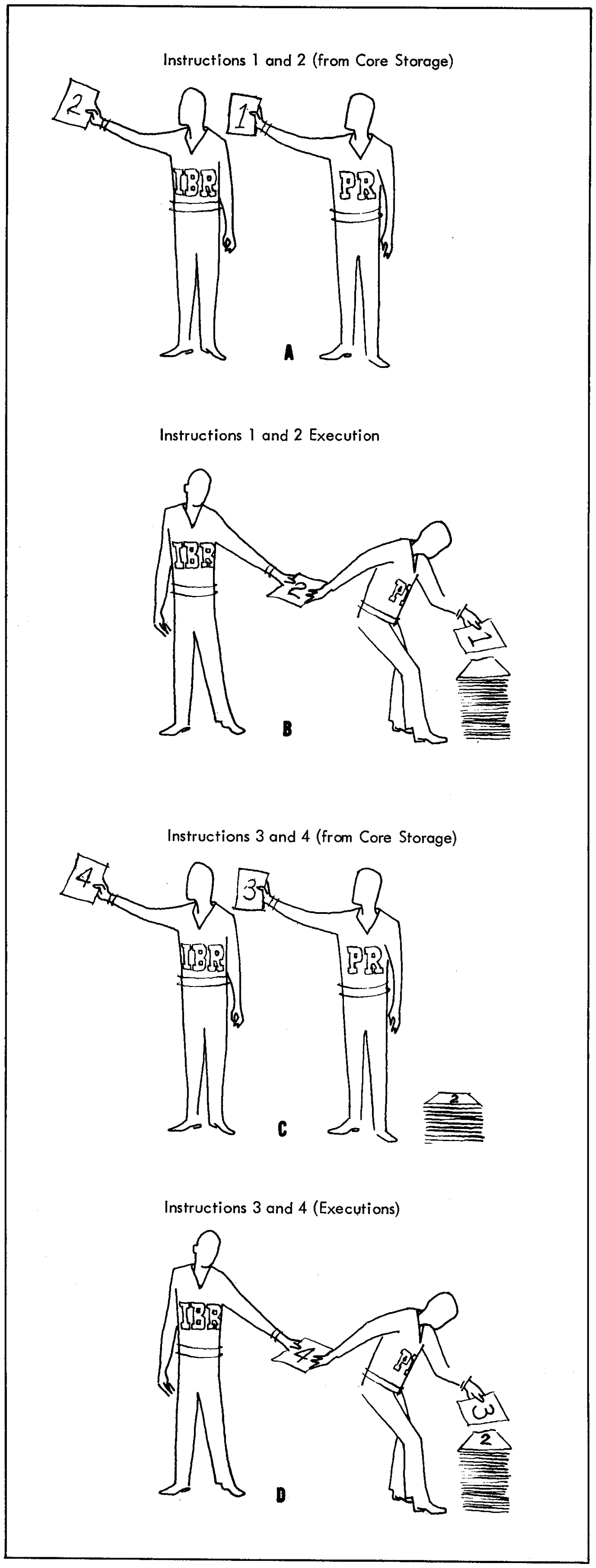

Instructions 1 and 2 (from Core Storage)
2
1
IBR
PR
A
Instructions 1 and 2 Execution
IBR
2
PR
1
B
Instructions 3 and 4 (from Core Storage)
4
3
IBR
PR
2
C
Instructions 3 and 4 (Executions)
IBR
4
PR
3
2
D

Figure 8. Double Instruction Overlap

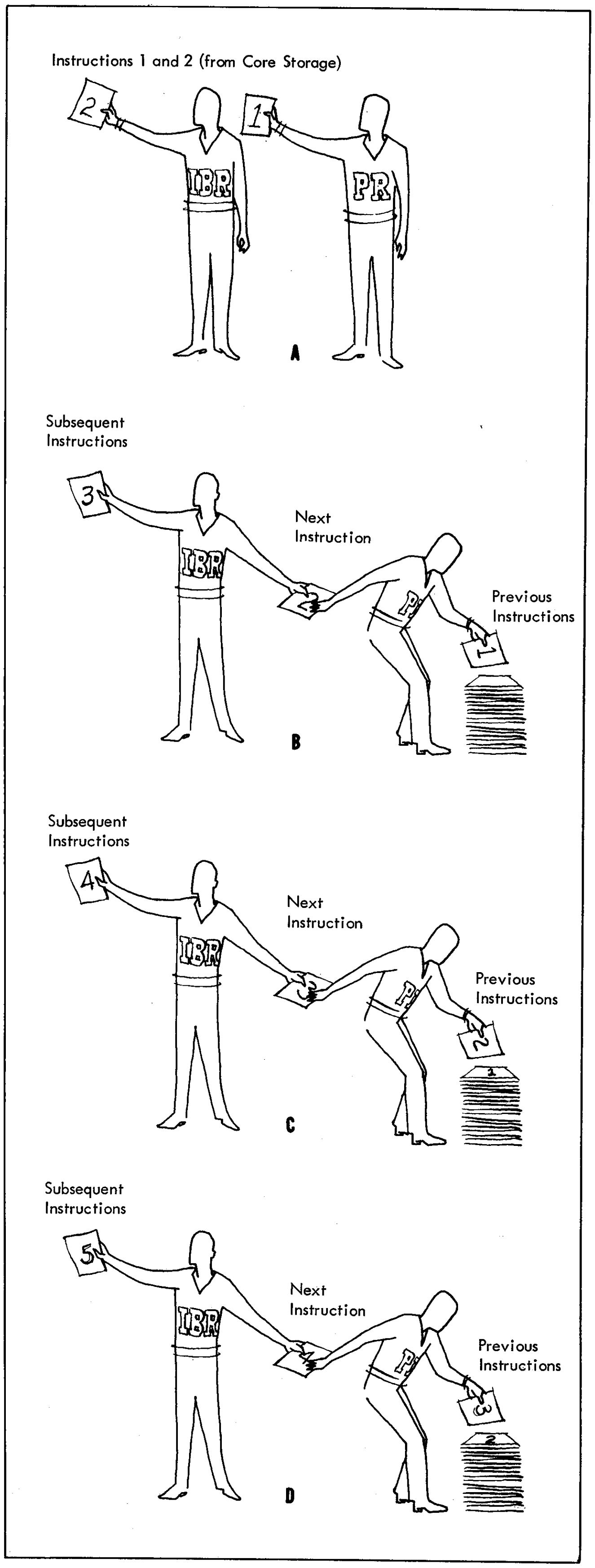


Figure 9. Extended Sequence Overlap

to represent all quantities. Counting is started in the binary system in the same manner as in the decimal system with 0 for zero and 1 for one. At two in the binary system it is found that there are no more symbols to use. It is therefore necessary to take the same move at two in the binary system that is taken at ten in the decimal system. This move is to place a 1 in the next position to the left and start again with a zero in the original position. A binary 10 is equivalent in this respect to a 2 in the decimal system. Counting is continued in a similar manner with a carry to the next higher order every time a two is reached instead of every time a ten is reached. Counting in the binary system is as follows:

BINARY	DECIMAL
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10

Octal Number System

It has already been pointed out that binary numbers require about three times as many positions as decimal numbers to express the equivalent number. This is not much of a problem to the computer itself. However, in talking and writing, these binary numbers are bulky. A long string of ones and zeros cannot be effectively transmitted from one individual to another. Some shorthand method is necessary. The octal number system fills this need. Because of its simple relationship to binary, numbers can be converted from one system to another by inspection. The base or radix of the octal system is 8. This means there are eight symbols: 0, 1, 2, 3, 4, 5, 6, and 7. There are no 8's or 9's in this number system. The important relationship to remember is that three binary positions are equivalent to one octal position.

A comparison of the binary, octal and decimal systems is as follows:

BINARY	OCTAL	DECIMAL
000	0	0
001	1	1
010	2	2
011	3	3
100	4	4
101	5	5
110	6	6
111	7	7
001 000	10	8
001 001	11	9
001 010	12	10
001 011	13	11
001 100	14	12

Remember that the computer's internal circuitry is concerned with only binary ones and zeros. The octal system is used to provide a shorthand method of reading and writing binary numbers.

The following are examples of whole numbers and fractions expressed in decimal, octal, and binary form. Octal-decimal conversion tables can be found in Appendix A, B, and C.

	DECIMAL	OCTAL	BINARY
Whole numbers	5	5	101
	85	125	1 010 101
	106	152	1 101 010
	127	177	1 111 111
	725	1325	1 011 010 101
	1125	2145	10 001 100 101
	3333	6405	110 100 000 101
	4095	7777	111 111 111 111
	4096	10000	1 000 000 000 000
Fractions	0.145	0.112	0.001 001 010
	0.250	0.200	0.010 000 000
	0.330	0.251	0.010 101 001
	0.500	0.400	0.100 000 000
	0.625	0.500	0.101 000 000
	0.656	0.520	0.101 010 000
	0.700	0.546	0.101 100 110
	0.734	0.570	0.101 111 000
	0.900	0.715	0.111 001 101
	0.915	0.724	0.111 010 100
Improper fractions	2.250	2.200	10.010 000 000
	3.375	3.300	11.011 000 000
	15.078	17.050	1 111.000 101 000
	17.050	21.031	10 001.000 011 001
	40.960	50.753	101 000.111 101 011
	63.984	77.770	111 111.111 111 000

Addition

Binary addition is simple. Its rules are as follows:

$0 + 0 = 0$
$1 + 0 = 1$
$0 + 1 = 1$
$1 + 1 = 0 + 1$ to carry

These rules operate in all cases of addition and apply to both addition of integers and of fractions. Binary numbers are added from right to left, and the carry is added to the adjacent bit on the left. The following examples illustrate the rules for binary addition. Note that the carry is placed in the column, to which it will be added, in parentheses.

$$\begin{array}{rrrrrr}
 & (1) & & (11) & & (1) \\
0 & 1 & 10 & 11 & 100 & 101 \\
+\ 1 & +\ 1 & +\ 1 & +\ 1 & +\ \ 1 & +\ \ 1 \\
\hline
1 & 10 & 11 & 100 & 101 & 110
\end{array}$$

The technical terms in addition are defined as the augend, addend, and the sum. The augend is the term that is to be increased; the addend is the term to be added to the augend; the sum is the result of the operation. For example:

101	Augend
+ 011	Addend
1000	Sum

In adding more than one number, the addition of the first set of numbers is performed and, to the sum, is added the third number. To the sum of the succeeding additions, add the next number until all the numbers have been totaled. For example, add:

```
  011
  111
+ 110
```

Addition of the first set of numbers	0111
	+ 111
First Sum	1010
Addition to the third number	+ 110
Final Sum	10000

```
  1101
  1001
  0010
+ 1111
```

Addition to the first set of numbers	1101
	+ 1001
First Sum	10110
Addition of the third number	+ 0010
Second Sum	11000
Addition of the fourth number	+ 1111
Final Sum	100111

Binary fractions are added in accordance with the rule that governs whole numbers. The binary point is fixed as in the decimal system. The carry from the addition of the binary fractions in the first position to the right of the binary point is an integer. For example, in addition of the following fractions:

DECIMAL	BINARY
1/8	.001
+ 3/8	+ .011
4/8 or 0.5	.100
4/8	.100
+ 6/8	+ .110
10/8 or 1.25	1.010
5 3/8	101.011
+ 6 7/8	+ 110.111
12 2/8 or 12.25	1100.010

Subtraction

The rules for binary subtraction are as follows:

$0 - 0 = 0$
$0 - 1 = 1$ (borrow 1 and make $0 = 10$)
$1 - 0 = 1$
$1 - 1 = 0$

The technical definitions of the terms used in subtraction are minuend, subtrahend, and difference. The minuend is the number to be decreased; the subtrahend is the quantity of the decrease; the difference is the result of the operation. Thus:

0110	Minuend
– 100	Subtrahend
010	Difference

The similarity which exists between decimal and binary arithmetic when a carry is involved is analogous to the similarity which exists when a borrow is involved. When subtracting a 1 from a 0, a 1 must be borrowed from the next higher order, diminishing that order by 1.

The following examples illustrate the rules for binary subtraction and the method of borrowing from the next higher order.

```
(A)  1101     (B)  1110     (C)  1100
    –0100         –0101         –1001
     1001          1001          0011
```

In the example A, the subtraction of 0 from 1, 0 from 0, and 1 from 1 produces the difference. In the example B, a 1 must be borrowed from the second order when attempting to subtract the 1 of the first order from 0. The 1 in the second order then diminishes to 0. In the example C, a slightly different borrow situation arises. The 1 to be borrowed must come from the third order of the minuend. That 1 then diminishes to 0. The 1 of the first order of the minuend can then be borrowed from the 10 which appears in the second order. Borrowing the 1 from 10 leaves a 1 in the second order of the minuend. Applying the rules of binary subtraction then produces the difference shown.

Complement Method

The preceding discussion delineated the methods of direct subtraction. The complement method of subtraction is a means of subtracting by addition. Design requirements of a processing unit do not allow for borrowing, so the complement method of subtraction fits in with processing unit design and capabilities.

A disadvantage of direct binary subtraction is that the direct subtraction of a number from a smaller number yields an incorrect result unless the subtraction is done by subtracting the smaller from the larger and then changing the sign of the difference. For example:

```
  5/16     0.0101
– 9/16   – 0.1001
– 4/16   – 0.0100
```

The difficulty encountered with negative results and the problem of providing for borrowing in circuit design are eliminated by changing the subtraction to an addition of negative numbers by means of the complement process.

The complement system of subtraction is possible because it is possible to limit the number of significant digits to be used in any one problem or machine. The problem is then said to have a modulus which is the count of the maximum number of numbers it would be possible to represent in this problem. For instance, suppose that a binary machine has facilities for handling four places, the machine could represent 16 different numbers from 0 to 1111. Such a machine has a modulus of 16 and is said to perform modulo 16 arithmetic.

The significance of the modulus of the machine is that each time an addition results in a number equal to or greater than the modulus of the machine, an integral multiple of the modulus is lost. An example of this action in everyday life is given by the automobile speedometer. When it reaches 100,000 miles, it resets to zero and starts over. The speedometer has lost 100,000 by resetting to zero. This property of machine-counting methods is important in the use of complements for subtraction by addition.

The complement method of subtraction may be derived from the following identity:

$$P - M + (M - N) = P - N$$

P = Minuend
N = Subtrahend
M = Modulus of the machine
$P - N$ = Difference sought

To derive the complement system of subtraction, let $(M - N)$ equal a number called the complement of N. Let C stand for this complement so $M - N = C$. Now substitute C in the identity:

$$P - M + C = P - N$$
$$\text{or } (P + C) - M = P - N$$

If M is moved to the other side of the identity, it becomes:

$$P + C = M + (P - N)$$

It is now evident that the minuend plus the complement of the subtrahend is equal to the difference of the minuend and subtrahend plus the modulus. It should now be recalled that when two numbers are added to obtain a sum greater than the modulus, the modulus is lost. Therefore, $P + C = P - N$ in any system with a fixed modulus, provided only the sum $P + C$ is greater than the modulus of the number system used.

The above is a derivation of what, in binary arithmetic, is called the 2's complement system. A similar derivation of a 1's complement system may be derived using $(M - 1)$ in place of M. In this case, the final equation is $P + C_1 - 1 = P - N$, which implies that the difference sought will be found by adding 1 to $P + C_1$. Note that C_1 is equal, in this case, to $(M - 1) - N$.

1's Complement

Every processing unit has a modulus which is one greater than the largest number the processing unit can register. For example, a six-place binary counter can express all the numbers from 0 to 111 111. The modulus of such a counter is 1 000 000.

To obtain the 1's complement of a number, it was shown in the derivation above that the number must be subtracted from $(M - 1)$. Therefore, to obtain the 1's complement of a number in a six-place machine, the number is subtracted from $(1\ 000\ 000 - 1)$; that is, from 111 111. As an example, find the 1's complement of the binary numbers 101 001 and 001 101.

111 111	Modulus −1
101 001	Number
010 110	1's complement of number
111 111	Modulus −1
001 101	Number
110 010	1's complement of number

A close examination of the numbers and their 1's complements shows that the 1's complement in binary arithmetic is nothing more than the original number with its bits reversed. That is, the original number's 0's are made 1's and the original number's 1's are made 0's. The way to get the 1's complement, then, is by inspection; just exchange 0's for 1's and 1's for 0's. For example:

100 101 = Number
011 010 = 1's complement

To perform subtraction by the 1's complement method, proceed as follows:

1. Find the complement of the subtrahend.
2. Add the complement to the minuend.
3. Perform *end-around carry* if there is a carry out of the highest position of the difference (explained below).

The result is the difference in complement form if it is negative and in true form if it is positive.

There are four possibilities, as shown by the examples below. All except the last are treated exactly the

EXAMPLES	DIRECT SUBTRACT		COMPLEMENT SUBTRACT	
Minuend < Subtrahend	+ 011 011	Minuend	011 011	Minuend
	− 101 010	Subtrahend	010 101	Complement
	− 001 111	Difference	110 000	Complement of difference
Minuend —Subtrahend	+ 011 011	Minuend	011 011	Minuend
	− 011 011	Subtrahend	100 100	Complement
	000 000	Difference	111 111	Complement of difference
—Minuend > Subtrahend	− 011 011	Minuend	100 100	Minuend complement
	− 010 011	Subtrahend	010 011	Subtrahend
	− 001 000	Difference	110 111	Complement of difference
Minuend > Subtrahend	+ 011 011	Minuend	011 011	Minuend
	− 010 101	Subtrahend	101 010	Complement
	+ 000 110	Difference	000 101	Difference — 1
			With a 1 end carry 1	
			000 110	True difference

same. The last requires the extra step of end-around carry, which is a carry from the highest order around to the lowest order. This carry is required because of the cyclical nature of the number system.

The only time it is required is when the minuend is larger than the subtrahend, that is, when the answer will come out a true positive answer. Fortunately, whenever it is required, there is a carry from the leftmost position, which serves as a reminder.

2's Complement

In the derivation of the complement system, it was shown that a 2's complement of a number is equal to the modulus minus the number, (M − N). Therefore, to obtain a 2's complement in a six-place machine, the number is subtracted from the modulus, 1 000 000. As an example, find the 2's complement of the numbers 101 001 and 001 101:

```
1 000  000 = Modulus           1  000  000 = Modulus
  101  001 = Number               001  101 = Number
  010  111 = 2's Complement       110  011 = 2's Complement
```

An examination of the numbers and their complements shows that the 2's complement of a number is the same as the 1's complement with a 1 added to it. The 2's complement is therefore formed by obtaining the 1's complement and adding 1 to it. For example, to form the 2's complement of 001 101:

```
001  101 = Number
110  010 = 1's Complement

110  010 = 1's Complement
      +1
110  011 = 2's Complement
```

To perform subtraction by the 2's complement method:

1. Find the 2's complement of the subtrahend.
2. Add this complement to the minuend.

The result is the difference in complement form if it is negative and in true form if it is positive. In the 2's complement system, there is no need to end-around carry.

Signed Numbers

How can negative numbers in complement form be distinguished from positive numbers in true form? In this regard, also, binary numbers offer an advantage with respect to representation. The sign of a number is binary in nature; that is, a number is either positive or negative. Thus, a bit representing the sign can be used in addition to the bits representing magnitude. A 0 in the sign bit position can be interpreted to mean that the number is positive. A 1 in the sign bit position can be interpreted to mean that the number is negative. By treating the signs separately from the magnitudes in each operation, the result sign can be predicted. Therefore, the rules of algebra apply in determining the result sign.

Multiplication

The rules for binary multiplication are similar to those of decimal multiplication. The rules for multiplying two single digits are the same in both systems. These rules are:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

The general procedure when multiplying two multiple digit binary numbers is the same as that in decimal arithmetic. That is, the multiplicand is multiplied by a digit of the multiplier, and the partial product obtained is placed so that the least significant digit is under the multiplier digit. When all the partial products have been found, they are added together to find the final product. The only difference between decimal and binary multiplication, therefore, is in the summing of the partial products. In binary, the binary addition table is used while in decimal, the decimal table is used.

As can be seen from the following examples, the method of obtaining partial products and then adding them to obtain the final product is identical to that of decimal arithmetic.

```
Multiplicand                 1010      10.11       1111
Multiplier                   1101      100.1       1111
First partial product        1010       1011       1111
Second partial product      0000       0000       1111
Third partial product      1010       0000       1111
Fourth partial product    1010       1011       1111
Final product           10000010   1100.011   11100001
```

Note the placement of the binary point in the second example. The same rules hold for its placement as hold for placement of the decimal point in decimal arithmetic.

The third example also illustrates an interesting point. This is the multiplication of the two largest possible 4-bit numbers. The product is 8 bits long. In other words the largest product that can result from the multiplication of two numbers will be no longer than the sum of the number of bits in the multiplier and multiplicand.

If a number is multiplied by the radix of the number system, this multiplication has the effect of shifting the number one place to the left with respect to the radix point. This is true in any number system. For example, multiply 12.51_{10} by 10 (the radix of the decimal system) and multiply the number 10.11_2 by 2 (the radix of the binary system):

Number	12.51	10.11
Number times radix	125.1	101.1

Binary multiplication, then, is nothing more than a series of add and shift operations. An example of such an operation is given under Fixed Point Arithmetic in Volume 2.

Division

Binary division is the process of counting the number of times a divisor goes into a dividend. The count of the number of times the divisor may be subtracted from the dividend before a negative remainder occurs is called the quotient.

Direct binary division is performed by a series of subtractions of the divisor (actually a multiple of the divisor), just as it is in the decimal system. For example, divide 100 011 100 by 1 110:

```
                 (bd ehi  jk)
                  10 100.01
         1 110 )100 011 100.00
           (a)   11 10
           (c)       111 1
           (f)       111 0
           (g)           100 00
           (l)            11 10
           (m)               10
```

In the example, the first step is to place the divisor below the dividend in a position which is as far removed to the left as possible (a), but which will allow a positive difference to result when the divisor is subtracted from the dividend. Since the divisor will go into this many bits of the dividend once, a 1 is placed in the quotient at b in the same column as the lowest order digit of the divisor. The divisor is then multiplied by the quotient digit, and the resulting product is subtracted from the dividend to produce the positive difference (c), called the current remainder. The next digit in the dividend is brought down to the line c. Compare the divisor to line c; note that the divisor is larger than line c, or that the divisor goes into line c 0 times. Therefore, place a 0 in the quotient at the d position. The next digit of the dividend is then brought down to line c. Comparing the divisor to line c shows line c to be greater. Place a 1 in the quotient at the e position. Multiply the divisor by the last quotient bit to form line f. Subtract line f from line c to start line g. The next digit in the dividend is brought down to line g. Compare the divisor to line g; the divisor is greater, so place a 0 in the quotient at position h. Bring the next digit of the dividend down to line g; by comparison line g is still smaller than the divisor. Place a 0 in the quotient in position i, and place the next dividend digit on line g. Still, line g is smaller than the divisor, so a 0 is placed in the quotient at position j. Placing the next dividend digit on line g now makes line g greater than the divisor. Place a 1 in the quotient at position k, and multiply the divisor by this 1 to form line l. Subtract line l from line k to start line m. Assuming a quotient has been developed of sufficient length, terminate the operation. The quotient is 10 100.01 with a remainder of 10 (line m).

Since the quotients bit is always either 0 or 1, the division process can be reduced to a series of subtractions of the divisor, multiplied by the power of the quotient bit being sought from the dividend. Each time a subtraction results in a positive current remainder, a 1 is placed in the corresponding quotient bit position, and the process is immediately repeated for the next quotient bit. Each time the subtraction results in a negative remainder, a 0 is placed in the corresponding quotient bit. In this case, the current remainder is restored to a positive number by adding the divisor back to it. Following this, the next quotient bit is obtained by the subtraction of the divisor multiplied by the power of the next quotient bit.

Since the quotient bits are generated from left to right, the power of each quotient bit is one smaller than that of the last bit generated. This means that as the divisor is successively subtracted from the dividend (or current remainder), the divisor is shifted to the right in relation to the binary point. The division process can therefore be reduced to a process of successive subtract and shift steps. An example of such a process is given under Fixed Point Arithmetic in Volume 2.

Component Circuits

All logic in the A and B gates of CPU 1 (7111) is composed of DIF (DIode Feedback) circuitry. This new F-level circuitry is of the non-saturating type and provides the three basic logical functions of AND, OR, and invert. A collector-to-base diode feedback network prevents transistor saturation and, therefore, allows high-speed circuit operation. Voltage inversion always occurs between the input and output.

Basic DIF Circuit Operation

Three logical functions are performed by the DIF circuit block: AND'ing, OR'ing, and inverting. Other functions are also performed such as terminating, driving, and converting from one voltage level to another, but these are only "convenience" functions, not logical functions.

DIF circuitry allows both the AND and OR logical functions to be performed within the same logic block before output powering and inversion. Not all logic blocks contain both functions, however; some produce just AND'ing, others just OR'ing, while others are simply inverters.

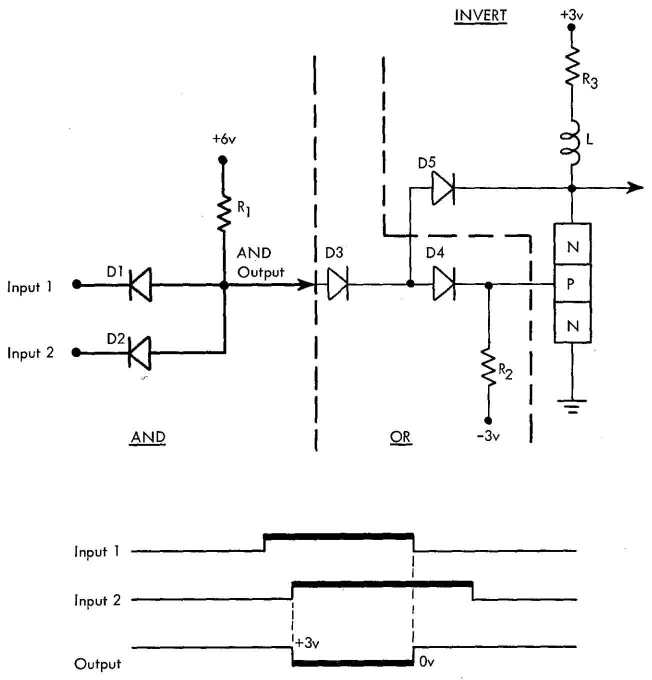


Figure 10. +AI (—OI) Circuit

AND-Invert

The two-legged circuit shown in Figure 10 represents either a +AI or —OI function. The circuit configuration for the +AND and —OR are identical. The polarity designations are adapted to work in negative logic; that is, the recognition of the absence as well as the presence of information. The +AI circuit requires that all inputs must be up (+3 volts) to obtain a down-level output (0 volts).

Note that even though this circuit represents only an AND function; an OR diode, D3, is also included. This one-legged OR circuit serves no logical function, but is always a part of the +AI configuration.

Figure 11a shows the junction point of the two input AND diodes. Only two inputs are shown in this case—three or more could also be used. The resistor (R1) limits the current flow and controls the rise time of the output.

If both inputs are at 0 volts (Figure 11b), the polarity is correct for both diodes to conduct. The resultant current flow through R1 causes a voltage drop across it to maintain a level of about 0 volts. (Consider the forward resistance of the diode to be insignificant.)

If input 1 rises to +3 volts (Figure 11c), D1 is cut off because the cathode is more positive than the plate. D2, with 0 volts on its cathode, maintains conduction and the output remains unchanged.

When input 2 changes to +3 volts (Figure 11d), D2 is cut off momentarily. The junction voltage starts rising towards +6 volts with a rise time effected by the stray capacitance of the associated circuitry. As the junction reaches 3 volts, both diodes return to conduction as a final steady state condition. If the input signals are not of the same voltage levels, the junction voltage assumes the lowest of the input sources.

When input 1 falls to 0 volts (Figure 11e), D1 conducts harder, D2 is cut off, and the output follows the input down to 0 volts. When input 2 falls to 0 volts, D2 goes back into conduction to help maintain the 0 volt output level.

At the output of every DIF logic circuit is a powering transistor which automatically causes a voltage inversion. When all inputs are down (0 volts), the base-to-emitter voltage keeps the transistor out of conduction. In this condition, the output signal level is essentially at that of the collector power source, +3 volts.

When both inputs 1 and 2 are at +3 volts, for example, diode D3 conducts and causes the voltage at the base of the transistor to go more positive. In this condition, the transistor conducts and the output signal is essentially that of the transistor emitter, 0 volts.

If the active logical input lines are represented as plus (+3 volts) levels, the active logical circuit output is a minus (0 volt) level (Figure 10). If this output is fed into a −AND or −OR circuit, however, the inversion was not a "logical" inversion.

Diode D4 in the OR circuit performs a resistance function only and does not affect the logic. D5 is the diode which clamps the collector at a more positive voltage than if the transistor were allowed to conduct fully. By preventing the transistor from going into saturation, better waveshapes are produced to allow faster computer operation.

OR-Invert

The two-legged OR circuit shown in Figure 12 represents either a +OI or −AI function; both differ in logical function but are identical in circuitry. The +OI circuit produces a down-level output (0 volts) if any one of the inputs is up (+3 volts). This inversion, as explained previously for the AND-invert, is caused by the output powering transistor.

Note that even though this circuit represents only an OR function, two corresponding AND diodes (D3 and D4) are included. These one-legged AND circuits serve no logical function, but are always a part of the +OI configuration.

Figure 13 shows the junction point of the two input OR diodes (D5 from Figure 12 has been eliminated at this time). If both inputs are at 0 volts (Figure 13b), the polarity is correct for both diodes to conduct. The voltage drop across the current limiting resistor (R2) sets the correct output level—for explanation purposes, 0 volts.

If either input rises to +3 volts (Figure 13c), that circuit leg conducts harder. The other diode cuts off and the output follows the input to +3 volts.

Normally only one input is active at any one time. The junction voltage, however, tends to follow the highest of the input signal levels.

AND-OR-Invert

Figure 14 shows the previously explained AND and OR functions combined into a typical DIF +AOI circuit. This circuit, too, can be considered as a −OAI depending on either the inputs available, or the output level desired. For example, if a minus (−) output level is desired, a +AOI designation would be used; if a plus (+) output level is desired, a −OAI would be applicable.

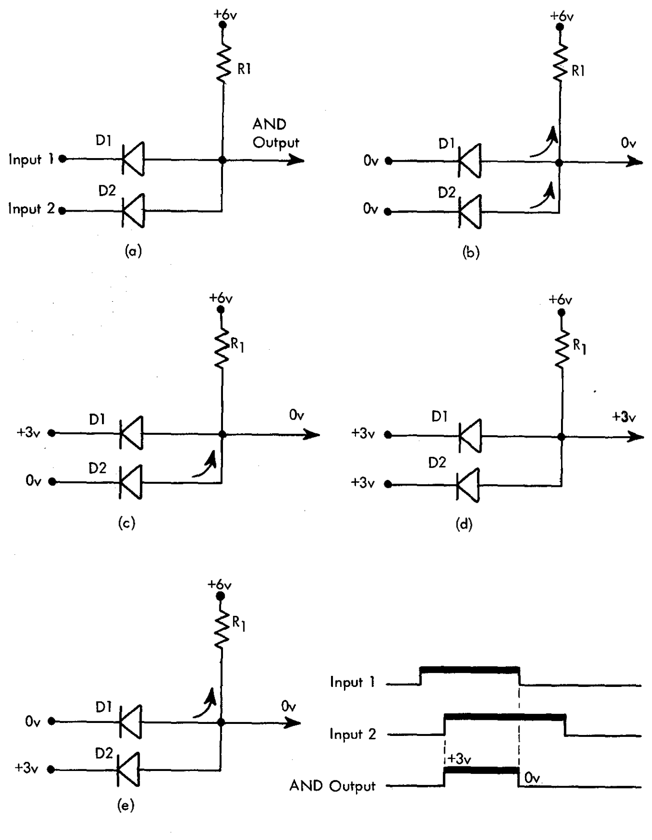


Figure 11. + AND, −OR Circuit

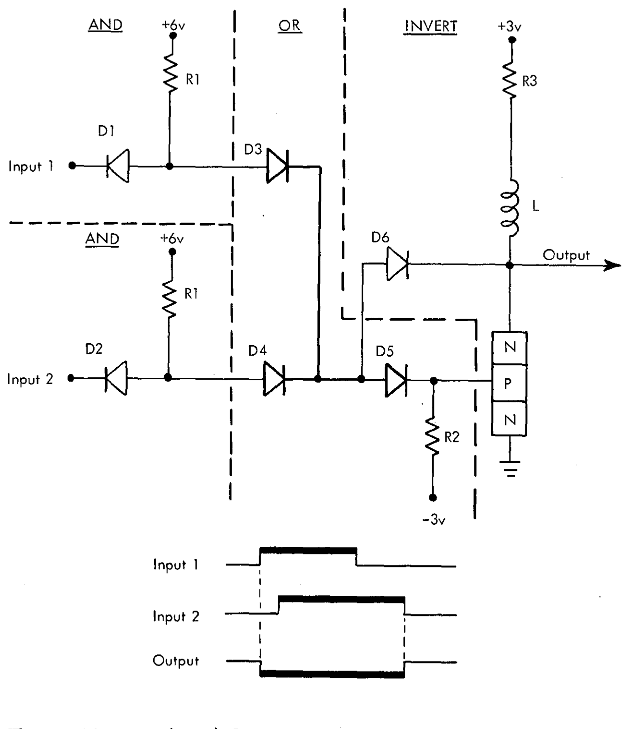


Figure 12. +OI (−AI) Circuit

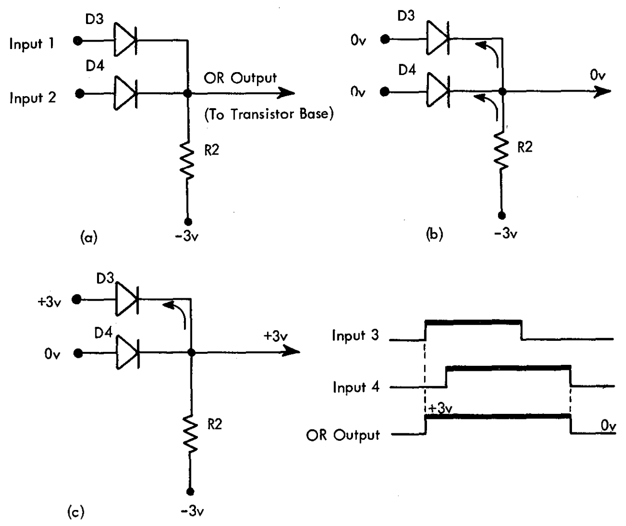


Figure 13. +OR, —AND Circuit

DIF Circuit Logic Blocks

Micro and Macro Blocks

Both "macro" and "micro" logic blocks are used on the 7094 II systems pages; the type of block used depends mainly on the availability of printing space. Macro blocks can incorporate two or more micro blocks, but only when there are no pin connections and back panel wiring between micro blocks.

As an example, consider the AND-OR-invert (AOI) circuit described in Figure 14. Figure 15a shows a macro block condensation of the corresponding three micro blocks at (b). Inputs to the macro block are "pinpointed" to specify input pins so that the AND functions are grouped and easy to separate visually. Of the eight possible inputs to the logic block, for example, the top AND function is pinpointed to input pins 2 and 3; the bottom AND function is pinpointed to input pins 6 and 7. Input pins 4 and 5 provide a visual separation of the two AND functions.

Note that the in-phase output is used from the two AND (+A) micro blocks. These two blocks represent diode circuitry only (see Figure 14). The out-of-phase output is used from the +OI because the transistor is located at this point.

AND-OR-Invert

The +AOI was described in a previous section. Because of the limited number of logic block input pins, only certain combinations of AND-OR functions can be shown without losing the visual spacing. Obviously, then, it is not possible to show two 4-way AND's. Combinations used include:

1. Two 2-way AND's,
2. Three 2-way AND's,
3. 3-way and 2-way AND,
4. 3-way and 1-way AND, etc.

In cases of negative logic (or because of the availability of signal lines), a +AOI can be represented by a —OAI. In either case, the internal circuitry is identical; only the final active output level will be changed.

The +AOI circuit is used in many cases to "gate" information into registers. In these cases the +AOI is replaced by a G within the circuit block.

Most circuit diagrams shown in this manual are condensed ALD systems with logic converted to positive (active) logic. Figure 16 shows at comparison of systems logic (a) and condensed positive logic (b). Note that the condensed logic is not concerned with voltage levels, only active logical levels. The out-of-phase (inverted) output of the systems gating circuit feeds a —SC; therefore, the inversion is not a logical inversion. Because there is no logic to the inversion, the condensed logic, therefore, uses an in-phase (active) output from the gating (AO) circuit.

DIF (F-level) Triggers

Triggers (also referred to as "latches") act as storage or remembering devices. Once they are turned on, they remain in that state until turned off. Triggers are used in some cases to form registers (tag register and address register, for example); in other cases they are

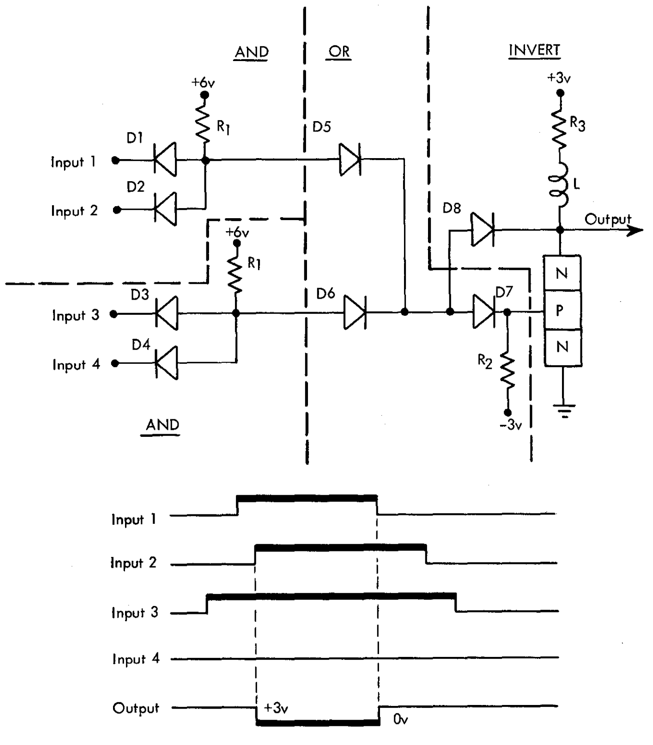


Figure 14. AND-OR-Invert Circuit

used singly to retain a specific condition (adder Q carry and MQ overflow, for example). Various trigger configurations exist in the 7094 II.

+ *AOI Trigger*

The +AOI is the most common type of trigger used in the new F-level circuitry of the 7094 II. This circuit, Figure 17, uses a +AOI followed by an inverter. One or more input AND conditions can exist, usually consisting of a data input and a gating input. The output from the inverter is fed back to the input circuit and AND'ed with the reset conditions to form a latch (or hold) circuit. Outputs from the +AOI and inverter blocks indicate the trigger ON and OFF conditions, respectively.

In Figure 17a, either data input AND being fully conditioned causes the output of the +AOI to go minus (−) and the output of the inverter to go plus (+). This inverter output is fed back to AND with the reset pulse line forming a hold circuit.

When the original data input AND circuit is deconditioned, the hold circuit keeps the trigger on. The trigger remains on until such time that the reset signal goes minus (−) and breaks the hold circuit.

Figure 17b shows the trigger in positive condensed logic. In-phase and out-of-phase trigger outputs indicate an ON and OFF state, respectively. Resets are shown at the bottom of the trigger block with an additional section added for each additional reset condition.

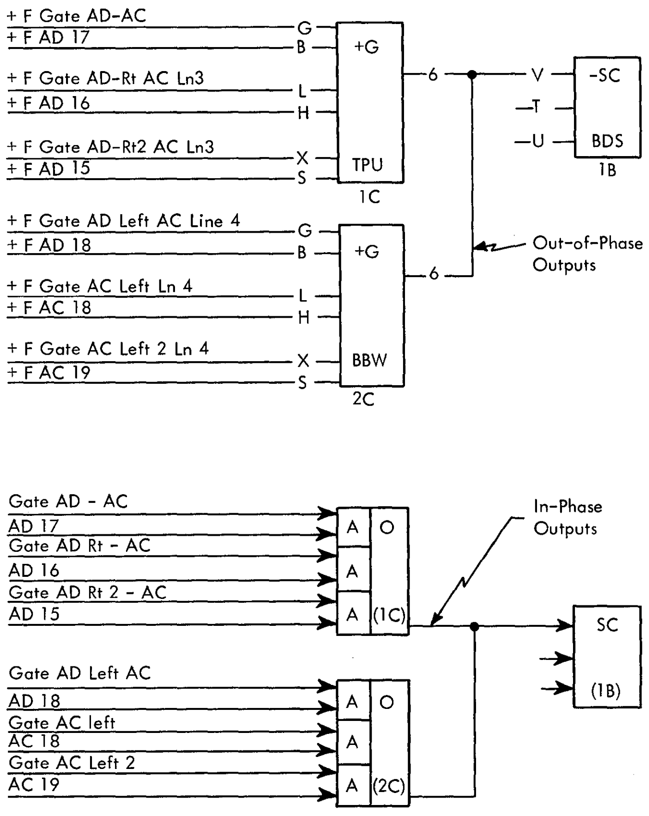


Figure 16. Systems Logic vs Condensed Positive Logic Gating Circuitry

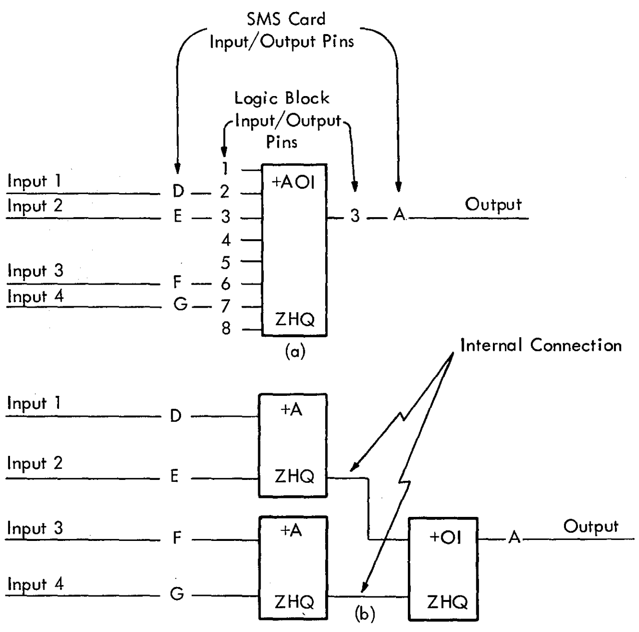


Figure 15. Macro and Micro Blocks

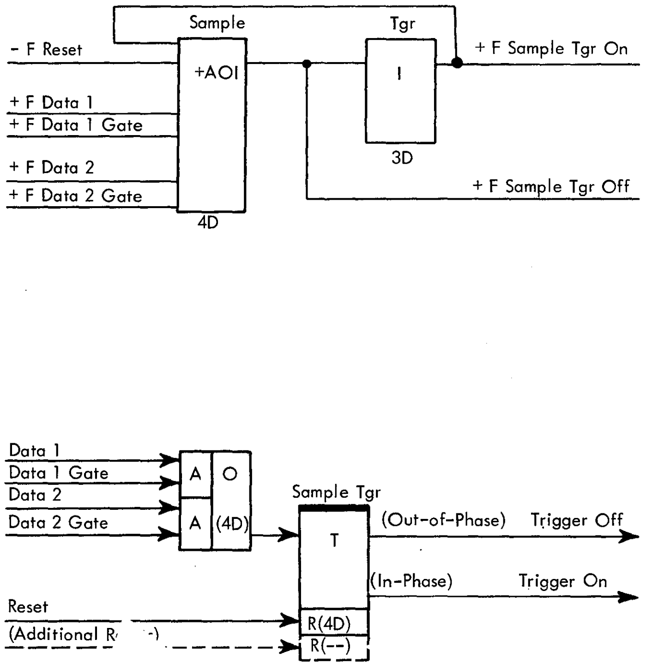


Figure 17. +AOI Trigger

— OAI Trigger

The −OAI trigger is sometimes used when +F input levels are not available and −F levels must be used. Figure 18 shows one example of a −OAI trigger that can be turned on from two sources and reset by one reset signal. Note that all of the active input levels are −F. The overall logic remains the same as the +AOI trigger discussed previously.

IBR Trigger

The IBR trigger circuit is packaged four to a twin card. On systems pages the trigger is shown as two +T blocks; two blocks are necessary because of the limited number of input pins available on an individual block. The trigger is also a macro block because the output inverter is not individually shown (Figure 19). The out-of-phase output of the +T block originates before the inverter; the in-phase output of the +T block originates after the inverter.

The address portion of the IBR (21-35) requires an additional input gate. The output of the +AI block is connected to the trigger output at a point in front of the output inverter; therefore, a hold circuit is established. The hold circuit is from pin 6 of the output inverter to pin 6 of the lower +T block. These two pin 6's are common points of internal card wiring; there is no back panel wiring as might be concluded from the systems page layout.

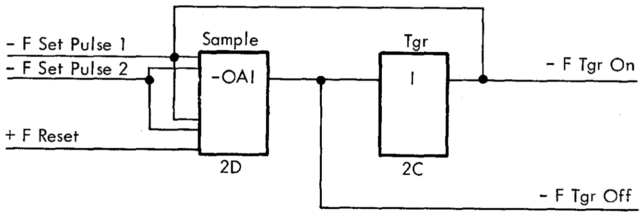


Figure 18. —OAI Trigger

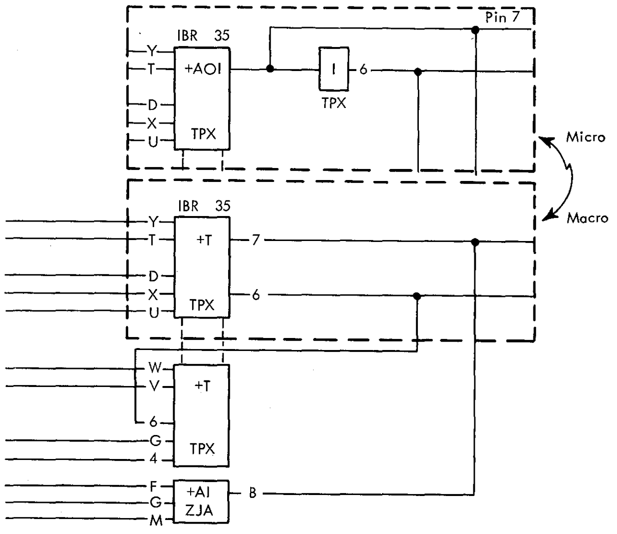


Figure 19. IBR Trigger

Shift Cell—SC

The shift cell is an element which can accept new data at its input while it simultaneously supplies old data at its output. Shift cells make up the storage register, accumulator, and multiplier-quotient (MQ) register in the 7094 II.

These shift cells are a "double-latch" type of circuit where two −OAI triggers are connected in series. Both "set" and "hold" pulses are generated by control circuitry and used to introduce new data. Figure 20 shows the shift cell configuration and sequence chart for setting in a 1 from an initial 0 condition. Note that the active levels of the input data, set, and hold pulse are −F. The appropriate data level must be present at the input to the shift cell prior to arrival of the set and hold pulses. The final output changes on the lagging edge of the pulse.

The set and hold pulses are approximately 60 nanoseconds in width and occur at the end of the 175 nanosecond clock pulse. The hold pulse is generated from the inverted output of the set pulse, skewed by one level of circuit delay.

If the shift cell initially contains a 0, point B (Figure 20) is plus (+) and O_2 is deconditioned. When a minus (−) data signal arrives at the shift cell, O_1 becomes con-

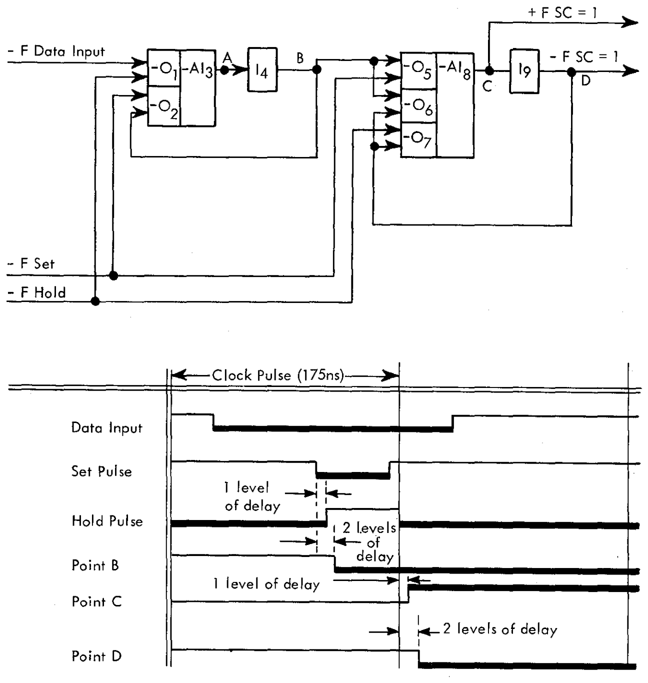


Figure 20. Shift Cell

ditioned. When the set pulse goes minus (−), both inputs are conditioned; point A goes plus (+), point B goes minus (−) and a data bit is set into the first half of the shift cell.

The minus (−) signal at point B feeds O_5 and O_6, but the final condition at O_7 is blocked because the hold pulse has gone plus (+). The incoming data bit is not allowed in the second half of the shift cell at this time because doing so would destroy the old data and defeat the purpose of the shift cell. The minus (−) output of the inverter at point B does, however, feed back to O_2 to act as a hold on the first half of the shift cell when the input data signal is removed.

At the end of the clock pulse, the hold pulse goes minus (−) and conditions O_7, the final input to the second half of the shift cell. At this time, point C goes plus (+), point D goes minus (−) and the shift cell indicates a 1 output. Point D is also returned to both O_6 and O_7 to form a hold circuit for the second half of the shift cell.

Figure 21 shows a sequence chart of a shift cell, AC(35) for example, under the following conditions:

1. Initially reset to a 0 state
2. The initial 0 replaced by a 1
3. The 1 replaced by a second 1
4. The second 1 replaced by a 0
5. The last 0 replaced by another 0

DOT-OR'ing and AND'ing

In many cases, the available circuits do not contain enough inputs to satisfy a logical function. To meet this requirement, two or more existing circuits can be DOT'ed together so that the function of one logical block is "extended" into the other. DOT'ing is accomplished by having the circuits share a common transistor load, and because of this, some circuit card types do not contain transistor collector loads.

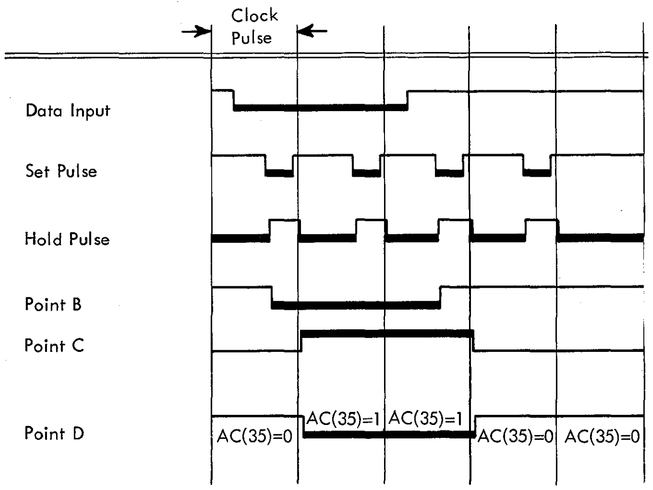


Figure 21. Shift Cell Timing Chart

Figure 22a shows the DOT'ing function with a three-legged OR and a two-legged AND. The top three-legged OR circuit supplies the collector load for both itself and the unloaded two-legged AND circuit below. Output connection between the two circuits is made by back panel wiring.

Note that the logical input and output levels desired (i.e., +F or −F) will determine whether the DOT'ing function is an AND or OR.

Figure 22b shows a DOT-OR'ed configuration. Any +F input to the +OI circuit causes transistor T1 to conduct and produce a −F output; also, both inputs being +F at the +AI circuit cause transistor T2 to conduct and produce a −F output. Therefore, if the logical output level is −F, either circuit block produces the required output.

Figure 22c shows the same identical circuit as (a) but as a DOT-AND configuration. All inputs being −F at the −AI circuit force transistor T1 out of conduction and produce a +F output; either input being −F at the −OI circuit forces transistor T2 out of conduction and also produces a +F output. If the input conditions are not as just described, either T1 or T2 will conduct and produce a −F output which is opposite to the logical output desired.

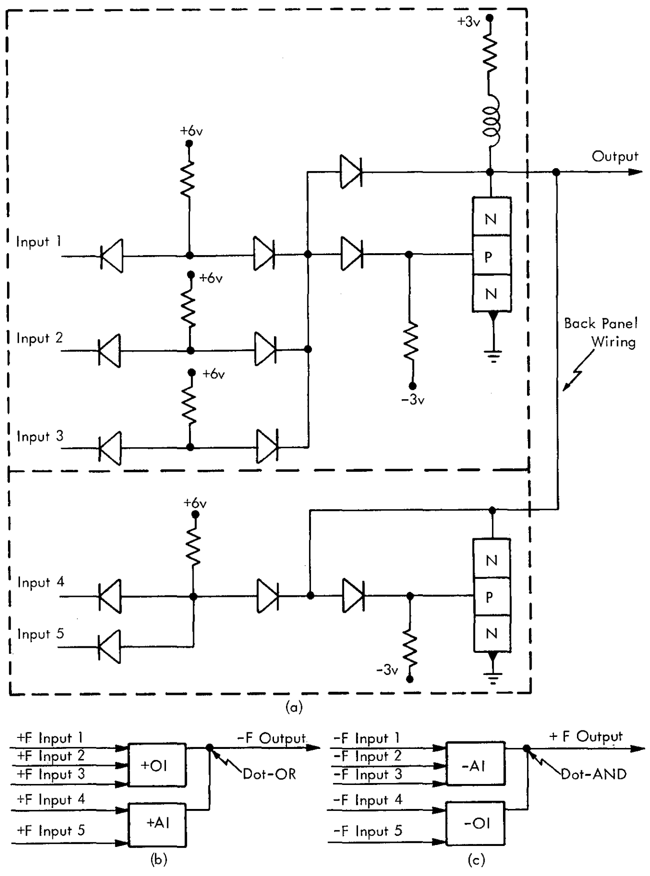


Figure 22. DOT-OR'ing and DOT-AND'ing

Component Circuits Card Types

Three types of SMS component circuit cards are used to support the DIF circuitry; single, twin, and STAN-PAC cards. In many cases, register positions or similar functions are combined on cards as both a packaging and trouble-shooting convenience.

SMS Single Card

All electronic components are mounted on the front side of the card and connections to the components are made on the back side by printed wiring patterns. The 16 contacts (labeled A through R) couple the signal and service voltages to the circuit components when the card is inserted into the SMS socket.

These single SMS cards form the bulk of the computer logic. They contain: basic circuit elements such as AND'ing, OR'ing, inverting, and terminating; and semi-specialized circuit functions such as adder look-ahead and gating.

SMS Twin Card

The twin SMS card is one physical card which requires the panel space of two single SMS cards. The use of twin cards provides more circuitry in a given space (compared to single cards) and is desirable in high speed circuitry because more operations can be performed before the resultant signal must be directed to other cards by way of connectors and back-panel wiring. The 32 contacts on the card (labeled A through Z and 1 through 8) couple the signal and service voltages to the circuit component when the card is inserted into the SMS sockets.

Circuitry using twin cards includes: input gating to the storage register, accumulator and MQ; instruction backup register; sense indicator register; index registers; and index adder positions.

Register positions and twin card locations are as shown in Figures 23 and 24.

SMS STAN-PAC Card

The STAN-PAC card is identified by its vertically mounted components. Resistors, diodes, chokes, and so forth have their top terminal welded to a component mounting strip which clamps to the body of the component for mechanical strength. The strip also provides an electrical path to the adjacent component. Both terminals of the components pass through a hole in the card and are soldered to a land pattern on the reverse side of the card. The 32 contacts on the card (labeled A through Z and 1 through 8) couple the signal and service voltages to the circuit components when the card is inserted into the SMS sockets.

Reg	Storage Register	Accumulator		MQ	
Pos	TPU	BBW	TPU	BBW	TPU
S	*	*	*	*	*
1	01B4F27-28	*	*	*	*
2	01B4F27-28	*	*	*	*
3	01B4F27-28	*	01B4A05	01B4A07	01B4A08
4	01B4F27-28	*	01B4A05	01B4A07	01B4A08
5	01B4F27-28	*	01B4A05	01B4A07	01B4A08
6	01B4F27-28	*	01B4A05	01B4A07	01B4A08
7	01B4F25-26	*	01B4A05	01B4A07	01B4A08
8	01B4F25-26	*	01B4A05	01B4A07	01B4A08
9	01B4F25-26	*	*	*	*
10	01B4F25-26	01B4A04	01B4A06	*	*
11	01B4F25-26	01B4A04	01B4A06	*	*
12	01B4F25-26	01B4A04	01B4A06	01B3A19	*
13	01B4F07-08	01B4A04	01B4A06	01B3A19	*
14	01B4F07-08	01B4A04	01B4A06	01B3A19	*
15	01B4F07-08	01B4A04	01B4A06	01B3A19	*
16	01B4F07-08	01B3A25	01B3A24	01B3A19	*
17	01B4F07-08	01B3A25	01B3A24	01B3A19	*
18	01B4F07-08	01B3A25	01B3A24	01B3A18	*
19	01B3F17-18	01B3A25	01B3A24	01B3A18	*
20	01B3F17-18	01B3A25	01B3A24	01B3A18	*
21	01B3F17-18	01B3A25	01B3A24	01B3A18	*
22	01B3F17-18	01B3A22	01B3A23	01B3A18	*
23	01B3F17-18	01B3A22	01B3A23	01B3A18	*
24	01B3F17-18	01B3A22	01B3A23	01B3A17	*
25	01B3H06-07	01B3A22	01B3A23	01B3A17	*
26	01B3H06-07	01B3A22	01B3A23	01B3A17	*
27	01B3H06-07	01B3A22	01B3A23	01B3A17	*
28	01B3H06-07	01B3A20	01B3A21	01B3A17	*
29	01B3H06-07	01B3A20	01B3A21	01B3A17	*
30	01B3H06-07	01B3A20	01B3A21	01B3A16	*
31	01B3H04-05	01B3A20	01B3A21	01B3A16	*
32	01B3H04-05	01B3A20	01B3A21	01B3A16	*
33	01B3H04-05	01B3A20	01B3A21	01B3A16	*
34	01B3H04-05	*	*	01B3A16	*
35	01B3H04-05	*	*	01B3A16	*

* Indicates the use of single cards.

Figure 23. Twin-Card Locations for SR, AC, and MQ Input Gating

The various BDS card register positions and corresponding machine locations are shown in Figure 25.

Figure 26 shows a STAN-PAC BDS card. To achieve the fast adder time of one clock pulse (175 nanoseconds), corresponding positions of the storage register, accumulator, MQ, and main adder have been combined on one STAN-PAC circuit card. Figure 26 shows position 35. This card position contains the shift cells for SR(35), MQ(35), and AC(35). One input gate (+G) circuit for SR(35) is also included; the remaining gates for the SR, MQ, and AC are on other cards and DOT-OR'ed at the shift cell as shown.

The corresponding main adder position also has its input gating circuitry and output lookahead functions (propagate, generate, and exclusive OR) on the same card. Note, however, that the actual adder sum output logic is on a separate card. Note, also, that the top circuit of the bottom adder gate is not used for AD(35).

(a)

Sense Indicator Register	Card Location (TPW)
S-3	01B2H28
4-7	01B2H27
8-11	01B2H26
12-15	01B2H25
16-19	01B2H24
20-23	01B2H23
24-27	01B2H22
28-31	01B2H21
32-35	01B2H20

(b)

Instruction Backup Register	Card Location (TPX)
S-3	01A3D12
4-7	01A3D11
8-11	01A3D10
12-15	01A3D09
16-19	01A3D08
20	*
21-23	01A3D07
24-27	01A3D06
28-31	01A3D05
32-35	01A3D04

*Indicates the use of single cards

(c)

Index Adder Position	Card Location (TPS)
3	01A3F18
4	01A3F17
5	01A3F16
6	01A3F15
7	01A3F14
8	01A3F13
9	01A3F12
10	01A3F11
11	01A3F10
12	01A3F09
13	01A3F08
14	01A3F07
15	01A3F06
16	01A3F05
17	01A3F04

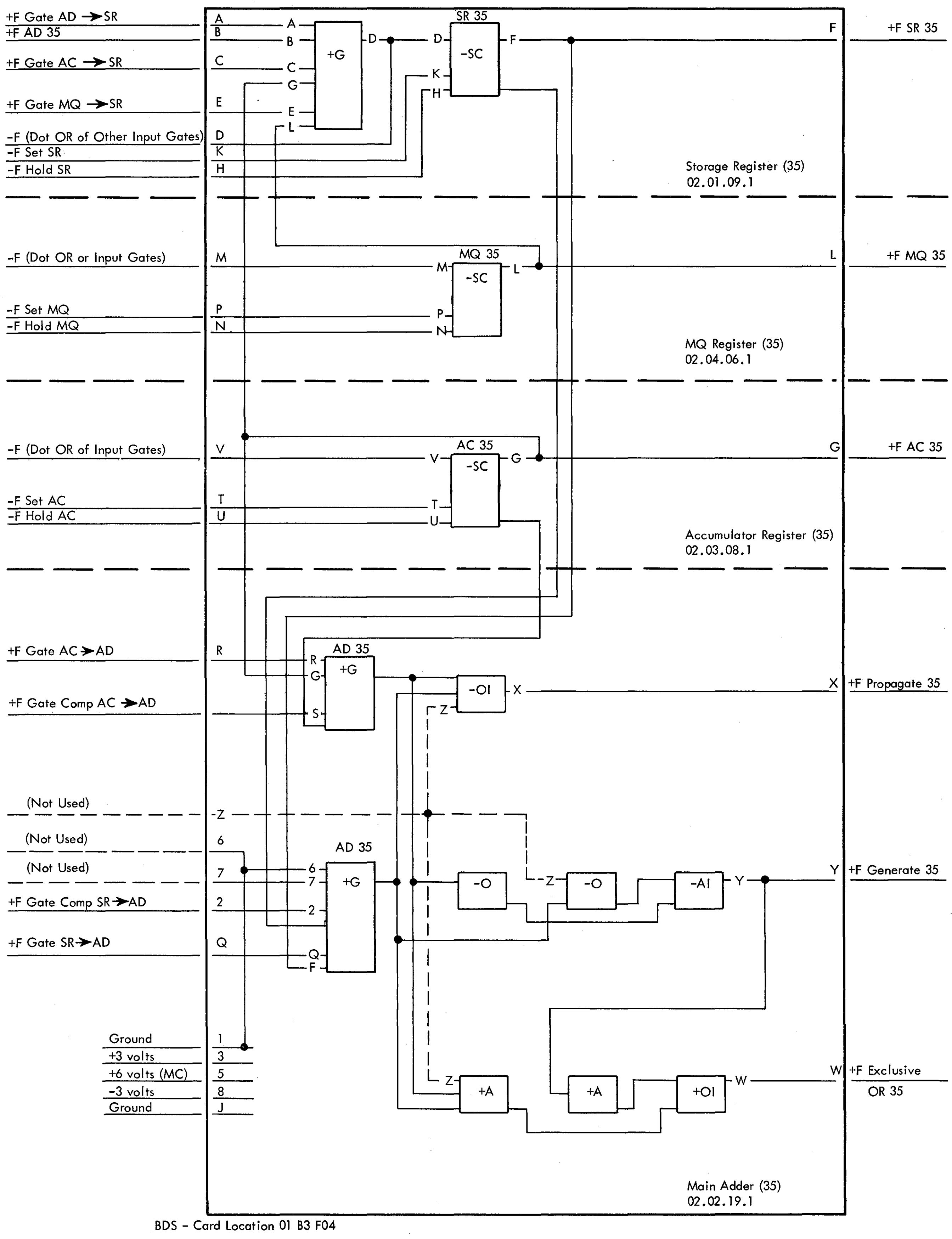
(d)

Index Registers Position	Card Location (TPV)
3	01A1G18
4	01A1G17
5	01A1G16
6	01A1G15
7	01A1G14
8	01A1G13
9	01A1G12
10	01A1G11
11	01A1G10
12	01A1G09
13	01A1G08
14	01A1G07
15	01A1G06
16	01A1G05
17	01A1G04

Figure 24. Twin-Card Locations for SI, IBR, XAD, and XR

SR,AC,MQ,AD Position	Card location (BDS)
SR(S),AC(S,P),AD(P)	01B4F22
1	01B4F21
2	01B4F20
3	01B4F19
4	01B4F18
5	01B4F17
6	01B4F16
7	01B4F15
8	01B4F14
9	01B4F13
10	01B4F12
11	01B4F11
12	01B4F10
13	01B4F09
14	01B4F06
15	01B4F05
16	01B4F04
17	01B3F25
18	01B3F24
19	01B3F23
20	01B3F22
21	01B3F21
22	01B3F20
23	01B3F19
24	01B3F16
25	01B3F14
26	01B3F13
27	01B3F12
28	01B3F11
29	01B3F10
30	01B3F09
31	01B3F08
32	01B3F07
33	01B3F06
34	01B3F05
35	01B3F04

Figure 25. BDS Card Locations for SR, AC, MQ and AD



BDS - Card Location 01 B3 F04

Figure 26. Typical BDS Card Layout

DIF Circuit Specifications

DIF Logic Block

Figure 27 shows a typical AND-OR-invert (AOI) DIF circuit. Diode D1 and the dotted circuitry below it indicates the AND function; diode D2 and the dotted circuitry below it indicates the OR function. Diode D3 produces an additional voltage drop in its circuit and is used because it provides better control than a resistor. Diode D4 provides the transistor collector-to-base feedback network to prevent the transistor from going into saturation. The inductor (L) in the transistor collector circuit is used to improve the output signal rise time by overcoming stray capacitance associated with the output circuit.

Inverter circuits are formed by using one-legged AOI circuits.

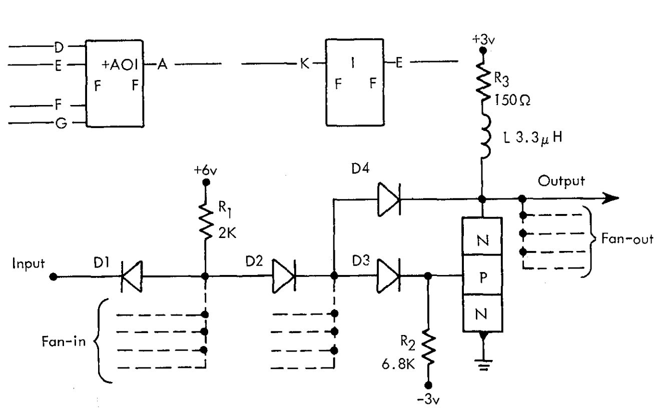
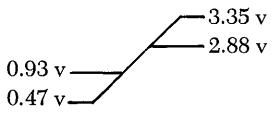


Figure 27. DIF Logic

Logic Block Input Specifications

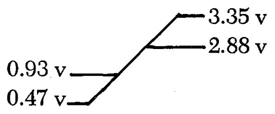
DC Voltage Levels and Limits:



Fan-In Capabilities: The logic block can have a fan-in of five on the AND circuitry, and five on the OR circuitry. Inputs can be expanded, however, as explained later.

Logic Block Output Specifications

DC Voltage Levels and Limits:



Fan-Out Capabilities: The logic block has a fan-out capability of ten of the following in any combination.

Logic blocks
Line drivers
Indicator drivers
Inverter (only one)—equivalent to three loads

Logic Block Circuit Delays

Circuit delays vary as a function of: the number of fan-in signals, the number of fan-out signals, and the total length of back panel wire. The total best-case to worst-case delays vary from approximately 5.0 to 20.0 nanoseconds. These delays apply to the logic block which is performing two logic functions (AND-OR).

Logic Block Power Supply Requirements

Nominal Voltages and Tolerances:

SUPPLY	TOLERANCE
+ 6 M	± 4 percent
+ 3 volts	± 4 percent
− 3 volts	± 4 percent

Overvoltage and Undervoltage Limits and Conditions:

1. The following overvoltages can be tolerated without causing component damage:

SUPPLY	OVERVOLTAGE LIMIT
+ 6 M	+ 9.0 volts
+ 3 volts	+ 3.6 volts
− 3 volts	− 6.0 volts

2. Any power supply can be open-circuited without causing component damage.
3. Any power supply can be shorted to ground without causing component damage.
4. Power supply sequencing is not required.
5. Logic block cards may be inserted into or removed from the computer with power on without causing component damage.

Logic Block Extended Capabilities

Eight-Way AND: An eight-way AND fan-in may be used with a maximum of one nanosecond delay added per block.

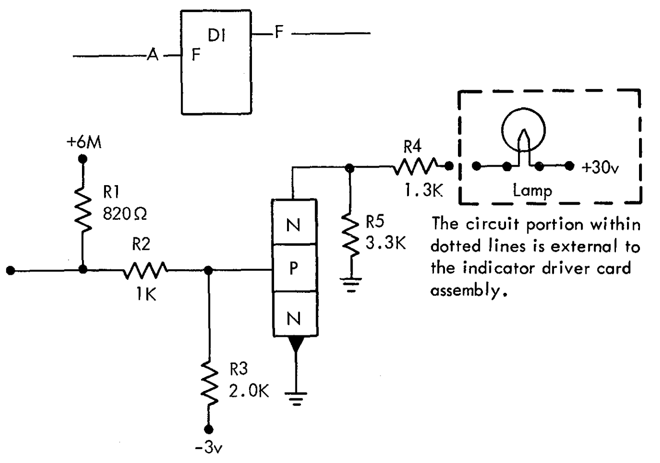
Dot-OR Connections: Logic block collectors may be DOT-OR'ed to increase the OR fan-in. Up to two additional collectors may be connected to a logic block output; the DOT-OR'ed collectors will have a single collector resistor and inductor. Each additional DOT-OR'ed collector adds 1.5 nanoseconds to the delay of the original circuit.

Frequently all of the circuit diodes are not used. In these cases:

1. AND diodes can "float"
2. OR diodes must be tied to ground

DIF Indicator Driver

The DIF indicator driver (Figure 28) is a saturating circuit designed to indicate an output level of the DIF logic block. The indicator light will be on when the logic block output is at the "up" (+3 volts) level, and off when the logic block output is at the "down" (0 volts) level. A small amount of "pre-energization" current continues to flow through the indicator lamp even when off and causes a faint glow at the indicator filament.

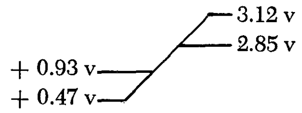


DI
F
A
F
+6M
R1
820Ω
R2
1K
R3
2.0K
-3v
N
P
N
R4
1.3K
R5
3.3K
Lamp
+30v
The circuit portion within dotted lines is external to the indicator driver card assembly.

Figure 28. DIF Indicator Driver

Indicator Driver Input Specifications

DC Voltage Levels and Limits:

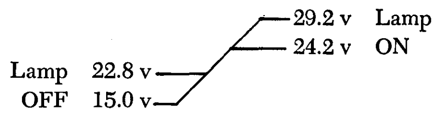


Loading: The indicator driver must be considered as at least three loads.

Indicator Driver Output Specifications

The output specifications for this circuit are determined by the particular indicator lamp being driven. These specifications are for driving indicator lamp P/N 550511 connected as shown in Figure 28.

DC Voltage Levels and Limits:



Indicator Driver Power Supply Requirements

Voltages and Tolerances:

+ 6 v	± 4 percent	
− 3 v	± 4 percent	
+ 30 v	± 4 percent	(required on the indicator panel—not on the circuit card)

Power Supply Limitations:

1. Power supply sequencing not required.
2. Duty cycle is 100 percent.
3. Card can be removed from the computer with power on without damage to the circuit.

N-Line to DIF Converter—Terminator

The N to F level converter provides proper termination for N-line (drift current mode) logic blocks (Figure 29). The converter also converts the N-line level to one capable of driving an F-level (voltage mode) logic block. The circuit is designed to be driven by 200 feet of coaxial cable from a B-type logic block.

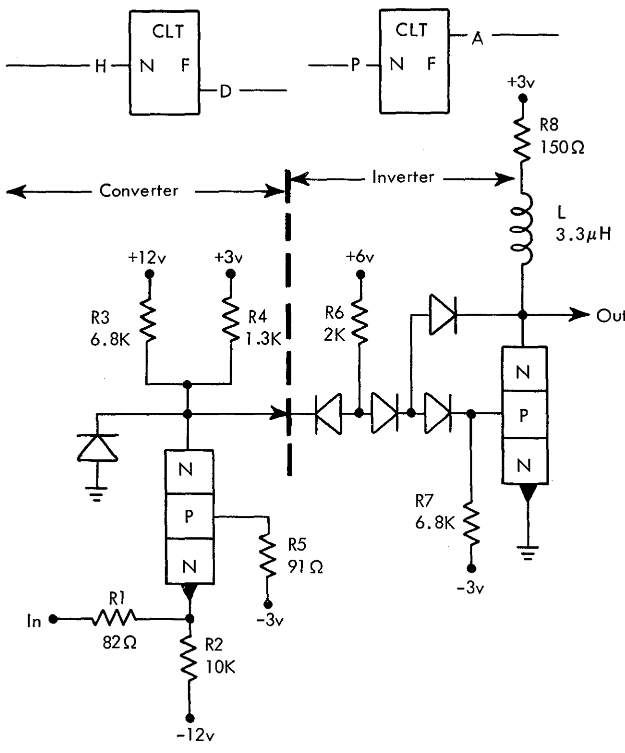


Figure 29. N-Line to F-Line Converter

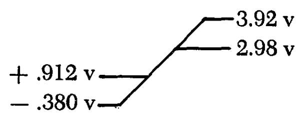
N to F Converter Input Specifications

Input voltage levels are a function of the input current which is set by the driving circuit, the 82 ohm input resistor, and the base-to-emitter drop of the transistor.

Fan-in Capabilities: The N to F converter-driver is designed to accept a single N-level input signal.

N to F Converter Output Specifications

DC Voltage Levels and Limits:



Fan-Out Capabilities: The N to F converter without the inverter output can drive only one logic block. When the converter has the inverter (out-of-phase) output, the capabilities are the same as for the AOI DIF logic block.

N to F Converter Delays

The best-case to worst-case circuit delays for the N to F converter vary from approximately 4 to 160 nanoseconds. Worst-case delays consider worst-case conditions of components, voltages, driver and output loading.

Power Supply Requirements

Power Supply Limitations:

1. All power supply tolerances are ±4 percent at the circuit.
2. Voltage sequencing not required.
3. Circuit card can be removed without damage to itself, its driver, or its load.

Over-Voltage Limits:

SUPPLY	OVER-VOLTAGE LIMITS
+ 12 volts	+ 20 volts
+ 3 volts	+ 9 volts
− 3 volts	− 9 volts
− 12 volts	− 20 volts

These limits assure that breakdown limits will not be exceeded; they do not assume proper delays and levels.

P-Line to DIF Converter—Terminator

This converter-terminator circuit is designed to terminate up to 200 feet of 93 ohm coaxial line driven by a P-line (current mode) logic block (Figure 30). The circuit also converts the incoming signal to an F-level (voltage mode) signal sufficient to drive ten DIF logic blocks.

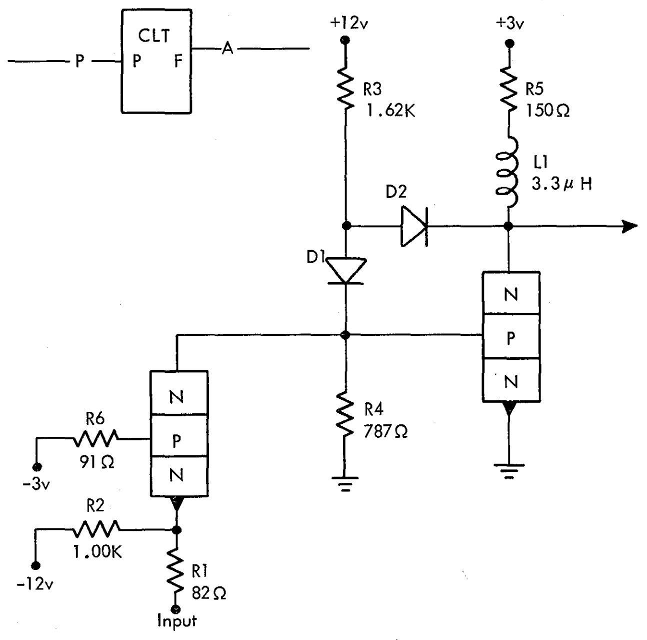
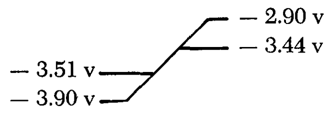


Figure 30. P-Line to F-Line Converter

P to F Converter Input Specifications

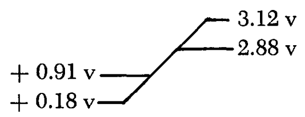
DC Voltage Levels and Limits:



Fan-in Capabilities: The P to F converter-driver is designed to accept a single P-level input signal.

P to F Converter Output Specifications

DC Voltages Levels and Limits:



Fan-Out Capabilities: The converter block has a fan-out capability of ten of the following in any combination.

1. Logic blocks
2. Line drivers
3. Indicator drivers
4. Inverter (only one)—equivalent to three loads

P to F Converter Delays

The best-case to worst-case circuit delays for the P to F converter vary from approximately 4 to 54 nanoseconds.

P to F Power Supply Requirements

Power supply requirements are the same as for the N-line to F converter-terminator.

DIF to N-Line Converter—Driver

The F to N-line converter is designed to accept an F-level (voltage mode) input signal and drive a single N-level (current mode) output (Figure 31). This output can feed up to 200 feet (maximum) of 95 ohm coaxial cable P/N 595997.

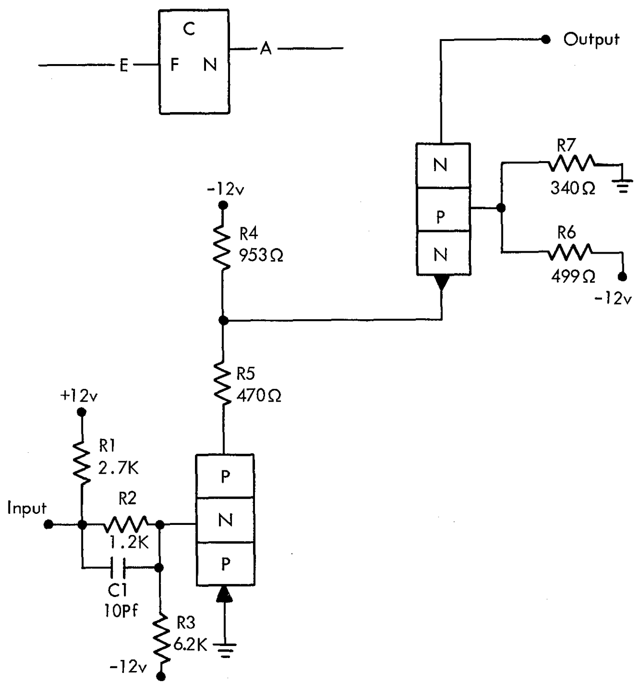
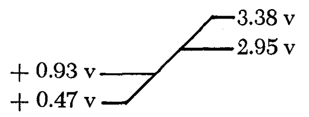


Figure 31. F-Line to N-Line Converter

F to N Converter Input Specifications

DC Voltage Levels and Limits:

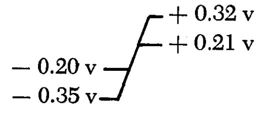


Fan-in Capabilities: The F to N converter-driver is designed to accept a single F-level input signal.

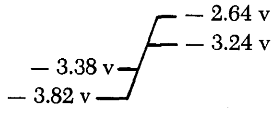
F to N Converter Output Specifications

DC Voltage Levels and Limits: The converter output voltage levels depend on whether the terminator is a translating (N to P-level) block, or a non-translating (N to N-level) block.

TRANSLATING LINE TERMINATOR



NON-TRANSLATING LINE TERMINATOR



Fan-Out Capabilities: The F to N converter can drive only one logic block. Output of the converter is an out-of-phase signal level.

F to N Converter Delays

The best-case to worst-case circuit delays for the F to N converter are approximately 6 to 76 nanoseconds.

F to N Power Supply Requirements

Voltages and Tolerances:

− 12 volts	± 4 percent
+ 12 volts	± 4 percent

A ±3 volt excursion is permissible on the +12 volt and −12 volt power supplies under fault conditions.

DIF to P-Line Converter—Driver

The F to P-line converter is designed to accept an F-level (voltage mode) input signal and drive a single P-level (current mode) output (Figure 32). The output can feed up to 200 feet (maximum) of 95 ohm coaxial cable P/N 595997.

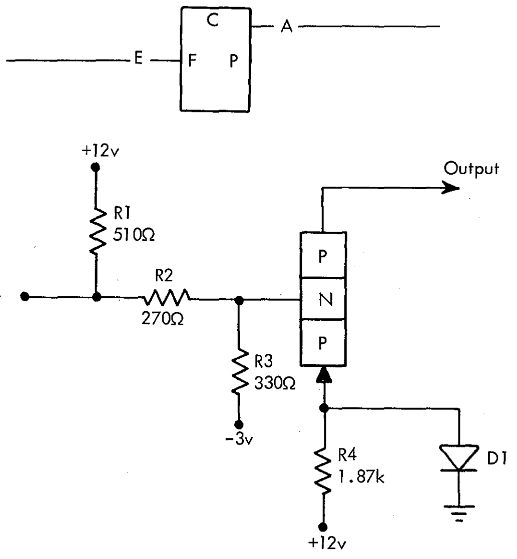
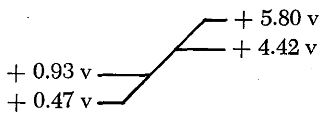


Figure 32. F-Line to P-Line Converter

F to P Converter Input Specifications

DC Voltage Levels and Limits:

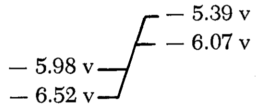


Fan-in Capabilities: The F to P converter is designed to accept a single F-level input signal. This converter circuit is to be driven by an unloaded DIF circuit. The maximum wire length between the DIF circuit output and converter input is not to exceed 24 inches. The DIF circuit driving this converter is to drive no other loads.

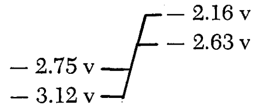
F to P Converter Output Specifications

DC Voltage Levels and Limits: The converter output voltage levels depend on whether the terminator is a translating (P to N-level) block, or a non-translating (P to P-level) block.

TRANSLATING
LINE TERMINATOR



NON-TRANSLATING
LINE TERMINATOR



Fan-Out Capabilities: The F to P converter can drive only one logic block. Output of the converter is an out-of-phase signal level.

F to P Converter Delays

The best-case to worst-case circuit delays for the F to P converter are approximately 11 to 62 nanoseconds. These delays are measured from the input of the DIF driving circuit to the output of the conversion circuit.

F to P Power Supply Requirements

Voltages and Tolerances:

— 3 volts	± 4 percent
+ 12 volts	± 4 percent

Power Supply Limitations:

1. The +12 volt supply cannot be more positive than +12.82 volts.
2. The −3 volt supply cannot be more positive than −1.5 volts.

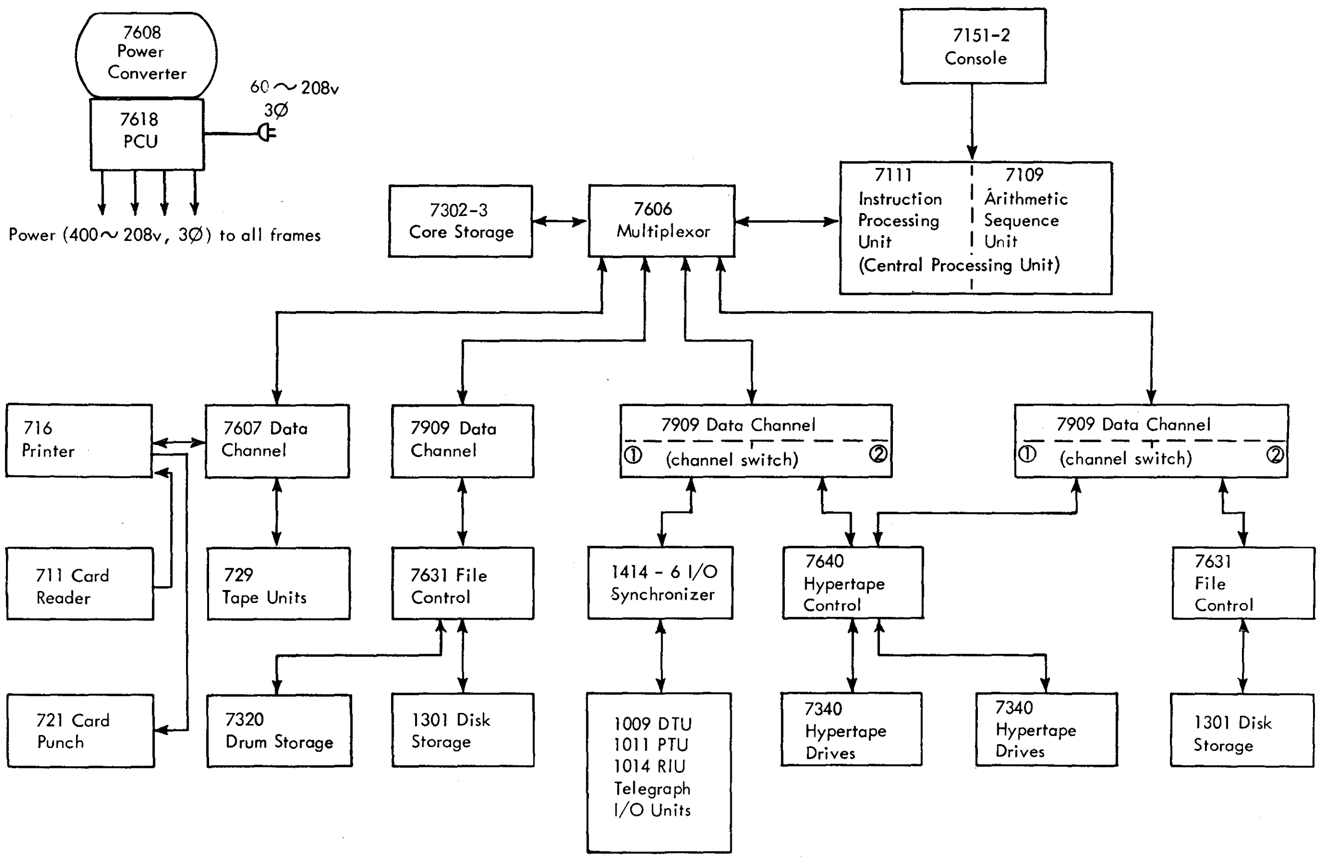


Figure 33. 7094 II System Configuration

System Components

Figure 33 is a block diagram of the 7094 II system configuration showing a possible combination of units. Of course, not all units are required in every installation. The final selection would be determined by customer need.

Figure 34 illustrates the basic functional organization of the 7094 II. Although there is a slight change in terminology, components and functions are essentially the same as previously described for a general computer (Figure 1). Note, however, that a multiplexor has been added. Arrows indicate the general flow of information. Although the sections can be neatly separated physically, there are many functional combinations not shown in this grouping. Storage is the only functional section that is a separate machine unit.

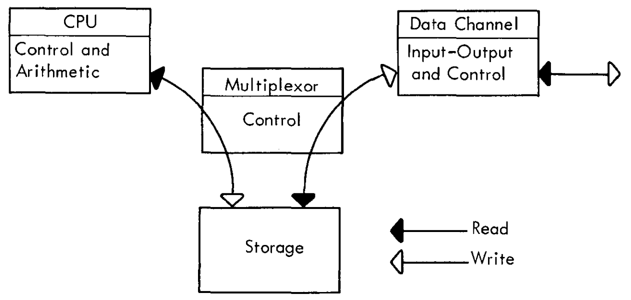


Figure 34. Basic Functional Organization

Briefly, computer information flow is from the input, through the multiplexor, and to core storage to set up the stored program. Instructions then come from core storage, through the multiplexor to the CPU for decoding, and a data reference is made back to core storage to the address specified in the instruction. Instructions return the answers to core storage where they, the answers, will eventually be transmitted to the output equipment.

The arrangement shown allows input-output (I/O) to operate somewhat independently, sharing storage with the computer. The highest order of controls is in the computer where control is delegated to the lower order controls in the data channel and multiplexor.

IBM 7111 and 7109 Central Processing Units

The central processing unit (CPU) of the 7094 II is actually made up of two sub-units—CPU-1 and CPU-2. The CPU-1 is the IBM 7111 Instruction Processing Unit and CPU-2 is the IBM 7109 Arithmetic Sequence Unit.

As these names imply, CPU-1 contains all arithmetic and control registers and accepts, decodes, and routes instructions to the rest of the computer. Because of the dense circuit packing, CPU-1 also performs a great deal of instruction execution. CPU-2 controls instruction execution for most POD 76 and sense indicator instructions, and I/O operations.

Many of the functions previous associated with CPU-2 are now contained in CPU-1. There is a great deal of interplay and overlap between these two units, and in some sequences it is difficult to determine an accurate functional boundary.

The arithmetic section is the calculating section of the computer system. Here, portions of information, either instructions or data, can be transformed, combined, or altered. This section also keeps account of the instruction it is using and the one it will use next.

The control section directs the other sections. It tells them what to do and when to do it. Instructions come into the control section from storage.

IBM 7606 Multiplexor

The IBM 7606 Multiplexor may best be described as a switching device which controls most of the data transfer and intercommunications within the system (Figure 35). It controls some of the core storage addressing and provides data paths to and from core storage for the computer and the data channels. The multiplexor also provides physical connections for the various cables to and from the data channels.

In addition to multiplexing data between core storage and the several sources of inputs and outputs, the multiplexor contains two master clocks for timings required by the main computer and data channels. These master clocks and their operation is described later in "Timing."

The multiplexor contains a buffer address register (BAR) used in data channel addressing and also look-ahead circuits that are used in conjunction with data channel operations.

Multiplexor Storage Busses

Data words are gated from the even/odd memory data registers and onto their respective memory data bus out (MDBO) lines. These two sets of 36-bit data lines arrive independently at the multiplexor (Figure 35).

Outputs from this multiplexor storage bus circuitry is sent to a variety of places. When sent to the CPU, both the even and odd busses remain independent and isolated from one another. Gating circuitry within the CPU (shown as Ⓖ in Figure 35) determines which bus/busses are gated into the storage register (SR), program register (PR) and instruction backup register (IBR).

For data channel operations only the appropriate memory is selected. Therefore, only one MDEO is active with data. Both sets of MDBO's are OR'ed together (as indicated by Ⓞ in Figure 35) and sent out unconditionally on banks 1 and 2. However, only the particular channel requesting the data gates the data word through the channel input switches. This gating occurs during B cycles or the E cycle of an appropriate channel instruction or command. The data word is treated as either data or a command word and set into appropriate registers.

Channel lookahead circuitry in the multiplexor tests the storage bus outputs to determine indirect addressing and TCH-type commands. If indirect addressing or a TCH-type command is detected, positions 21-35 (address portion) are immediately gated to the buffer address register so that a second core storage reference can be made for the data channel.

Multiplexor Storage Bus Input OR'ing

The multiplexor input OR'ing circuitry gates all data sent to core storage (Figure 35). Data can arrive from the storage register in the CPU or from banks 1 or 2 of the channel storage bus. Timing and priority circuits throughout the system, however, prevent more than one input from being active at any one time.

Outputs from the input OR'ing are sent over 36 lines to the memory data bus in (MDBI) of core storage. The data word is gated into either the even or odd memory data register as determined by the MAR selection and addressing circuitry.

Buffer Address Register

Memory address switching which existed in the multiplexor of the 7090/7094 computers has now been greatly expanded and relocated in CPU-1. Its place in the multiplexor has been taken over by the buffer address register (BAR) which is directly associated with data channel operations (Figure 35).

Inputs from the channels for memory selection come from the channel address switches in either bank 1 or 2. Inputs from positions 21-35 of the multiplexor storage bus occur for memory selection under indirect addressing or TCH-type channel commands. Logic, however, prevents more than one input from being active at any one time. During channel trap operations, various BAR positions are forced on to cause trapping to appropriate memory locations. For example, BAR (14 and 16) indicate the channel A trap address of 12_8.

Outputs from the buffer address register go to the MAR selection circuitry for memory addressing, and also to banks 1 and 2 as an input to the channel address input switches.

IBM 7302-3 Core Storage

All information in the system is at one time or another in storage; therefore, computer speed depends on storage speed. The storage scheme of most computer systems today is random access (any portion of information can be located directly without searching other locations).

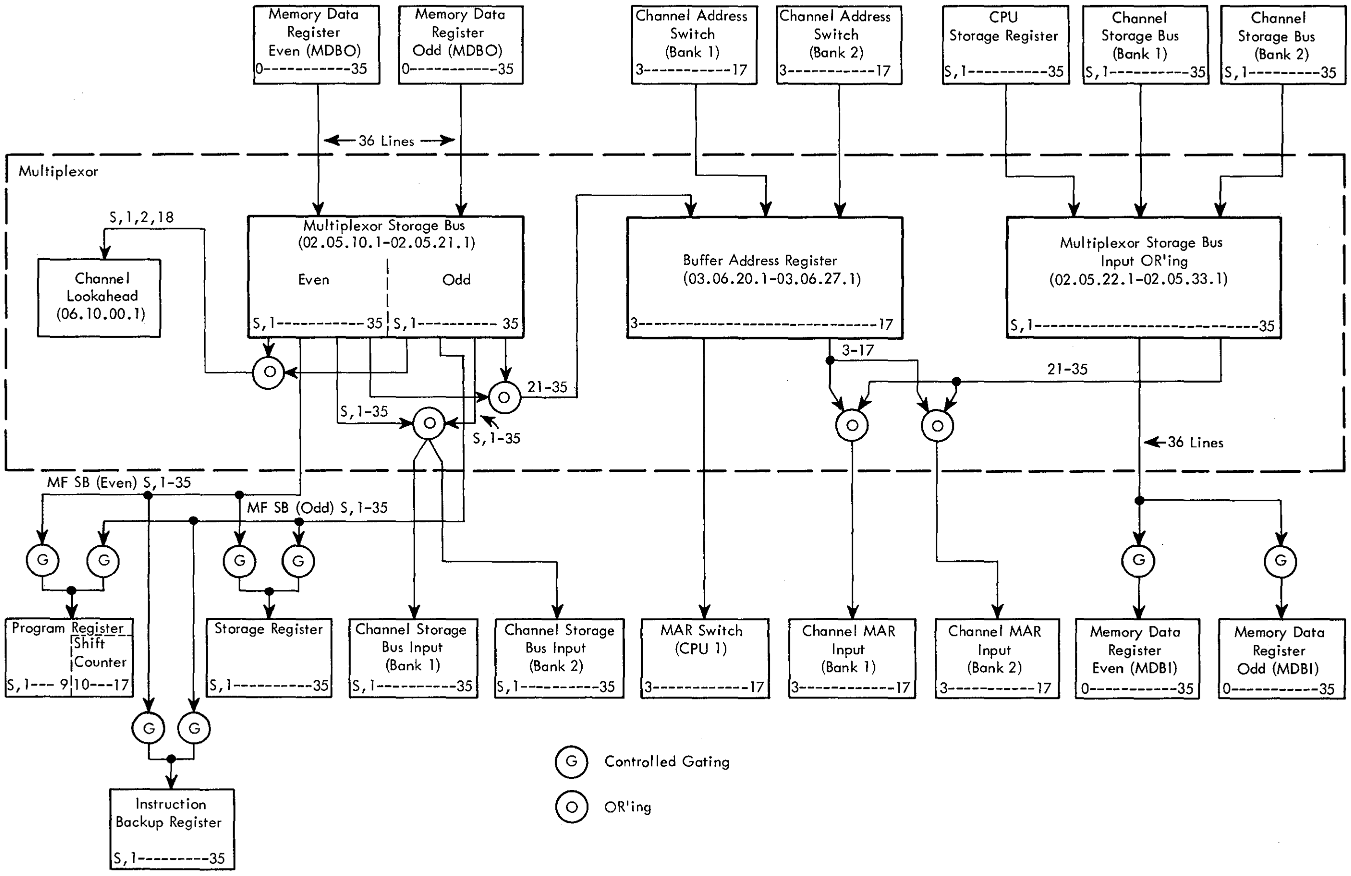
The new 1.4-microsecond air-cooled 7302-3 core storage provides two logically independent random access storage blocks of 16,384 words each. Both blocks (arrays) can perform simultaneous data fetches because each array has its own 14-bit memory address register (MAR), 36-bit memory data register (MDR) and 36-bit memory data bus out (MDBO). All of the following multiplexor controls have been duplicated:

Memory select
Memory read-out control
Store prefix control
Store decrement control
Store tag control
Store address control
OR to storage
Memory test time

One memory data bus in (MDBI) is located in the multiplexor and wired to both MDR's. Storing can not, therefore, be executed simultaneously to both the even and odd memory; only the memory selected by the store operation generates the data-in gate (DIG). A simultaneous store and fetch combination may occur provided there is no memory conflict.

The data channel and computer can not use core storage simultaneously. Channel B cycles can overlap CPU L cycles, however, because L cycles require no reference to core storage. Channel cycle requests are honored on the cycle following their receipt in the CPU, and continue to assume priority until such time that all requests from all channels have been satisfied.

For more information on core storage, refer to *IBM 7302-3 Customer Engineering Instruction—Maintenance Manual,* Form 223-2724.



Memory Data Register Even (MDBO) 0----35
Memory Data Register Odd (MDBO) 0----35
Channel Address Switch (Bank 1) 3----17
Channel Address Switch (Bank 2) 3----17
CPU Storage Register S,1----35
Channel Storage Bus (Bank 1) S,1----35
Channel Storage Bus (Bank 2) S,1----35
36 Lines
Multiplexor
S,1,2,18
Channel Lookahead (06.10.00.1)
Multiplexor Storage Bus (02.05.10.1-02.05.21.1)
Even
Odd
S,1----35 S,1----35
Buffer Address Register (03.06.20.1-03.06.27.1)
3----17
Multiplexor Storage Bus Input OR'ing (02.05.22.1-02.05.33.1)
S,1----35
3-17
21-35
21-35
S,1-35
S,1-35
36 Lines
MF SB (Even) S,1-35
MF SB (Odd) S,1-35
Program Register
Shift Counter
S,1---9 10---17
Storage Register S,1----35
Channel Storage Bus Input (Bank 1) S,1----35
Channel Storage Bus Input (Bank 2) S,1----35
MAR Switch (CPU 1) 3----17
Channel MAR Input (Bank 1) 3----17
Channel MAR Input (Bank 2) 3----17
Memory Data Register Even (MDBI) 0----35
Memory Data Register Odd (MDBI) 0----35
Instruction Backup Register S,1----35
G Controlled Gating
O OR'ing

Figure 35. Multiplexor Data Flow

7302-3 Addressing

Because each array consists of 16,384 positions, only 14 address bits are required to effect any location. The 15th bit is used for memory selection (even or odd). MAR (3 through 16) select an address; MAR (17), which no longer exists as such, controls the memory selection. Memory selection is effected by whether the computer is in normal or diagnostic mode of operation. In normal mode, bit (17) controls memory selection whereas in diagnostic mode bit (3) controls memory selection. This "switching" of bits (3 and 17) is explained later in this chapter under "MAR Bus Selection and Switching."

Because of the signal levels sent from the computer to core storage, 1's complement addressing is indicated in the MAR lights at the CE test panels; lights that are on in the program counter at the operator's console, for example, will be off at the 7302-3 and vice-versa. This address interpolation is best done visually by reading the address in the lights that are off. The following chart, however, shows a light correlation between the CPU and 7302-3 for four different address groups.

CPU	7302-3 CE PANEL MAR LIGHT INDICATIONS			
ADDRESS	NORMAL MODE		DIAGNOSTIC MODE	
INDICATION	EVEN MAR	ODD MAR	EVEN MAR	ODD MAR
00000	77776	—	77776	—
00001	—	77776	37776	—
00002	77774	—	77774	—
00003	—	77774	37774	—
37774	40002	—	40002	—
37775	—	40002	00002	—
37776	40000	—	40000	—
37777	—	40000	00000	—
40000	37776	—	—	77776
40001	—	37776	—	37776
40002	37774	—	—	77774
40003	—	37774	—	37774
77774	00002	—	—	40002
77775	—	00002	—	00002
77776	00000	—	—	40000
77777	—	00000	—	00000

Note: lights will show 77776_8 if this memory is not selected.

Input/Output

The input section of a computer system accepts information from an outside source and places it into the storage section. This information may come from punched cards, magnetic tape, manually operated keys or a variety of other devices. The information may be instructions, data (numbers for arithmetic calculations), or alphabetic characters for printing page headings, comments, etc.

The output section takes calculated information from storage and presents it to an outside user. Commonly used forms of outputs are: magnetic tape, magnetic disk, punched cards, printed reports, or indicator lights.

The devices and associated data channels applicable to 7094 II input/output (I/O) operations are variable and include the following:

7607 DATA CHANNEL
MODEL 1
729 II Magnetic Tape Unit
729 IV Magnetic Tape Unit
716 Printer
721 Card Punch
711 Card Reader

7607 DATA CHANNEL
MODEL 2
729 II Magnetic Tape Unit
729 IV Magnetic Tape Unit

7607 DATA CHANNEL
MODEL 3
729 II Magnetic Tape Unit
729 IV Magnetic Tape Unit
729 V Magnetic Tape Unit
729 VI Magnetic Tape Unit
716 Printer
721 Card Punch
711 Card Reader

7607 DATA CHANNEL
MODEL 4
729 II Magnetic Tape Unit
729 IV Magnetic Tape Unit
729 V Magnetic Tape Unit
729 VI Magnetic Tape Unit

7607 DATA CHANNEL
MODEL 5
716 Printer
721 Card Punch
711 Card Reader

7909 DATA CHANNEL
7631 File Control
7640 Hypertape Control
1414-6 I/O Synchronizer
7750 Programmed Transmission Control

IBM 7607 Data Channel

The IBM 7607 Data Channel controls the flow of information between I/O units and core storage. The Model 1 can control any combination of ten 729 II and 729 IV tape units and up to one each of reader, punch, and printer. The printer must be present if either a reader or a punch is to be used. A 7607-2 can control any combination of ten 729 II and IV tape units, but no card machines. The 7607-3 and -4 perform the same functions as Models 1 and 2 but with the added capacity for handling 729 V and VI tape units. The 7607-5 handles only the card equipment; one each of reader, printer, and punch.

The IBM 7094 II Data Processing System may include up to eight data channels. Each data channel can be regarded as a subsystem with its own manual control console and indicator panel (not shown in Figure 34). Once a data channel is set in operation by an instruction in the computer program, it can call its own instructions (called commands in channel operations). These commands make up what is known as an I/O program. This program controls the operation of the I/O unit. Information received from an I/O unit is placed in core storage, or information is taken from core storage to be supplied to a selected I/O unit.

The computer is responsible for selecting a particular data channel and supplying its first command. The first command can be the first of a series of commands (I/O program) that will sustain the selected channel and device in operation independently of the computer. When this I/O program has run its course, the selected device stops and the operation is complete.

It is possible for a 7094 II, using eight data channels, to have eight I/O programs and the computer program in operation simultaneously—each independent of the others and all sharing one common core storage.

The IBM 729 Magnetic Tape Units write information on magnetic tape or read information from magnetic tape. The higher model numbers indicate advanced versions of the basic unit—usually referring to an increased character rate.

The IBM 711 Card Reader reads information from punched cards at 250 cards per minute.

The IBM 716 Printer prints information from core storage at 150 lines per minute. The typewheel echo pulses are available to the computer where they may be used to check the accuracy of printing.

The IBM 721 Card Punch punches information from core storage at 100 cards per minute.

For a more detailed discussion of the IBM 7607 Data Channel and its associated I/O units refer to the *IBM 7607 Data Channel Customer Engineering Instruction-Reference Manual,* Form 223-6910 and the *IBM 7094 Data Processing System Reference Manual,* Form A22-6703.

IBM 7909 Data Channel

The IBM 7909 Data Channel is a stored program device designed to increase the capabilities of the IBM 7094 II Data Processing System. The data channel attaches to the IBM 7606 Multiplexor in the same manner as the IBM 7607 Data Channel and controls data flow between core storage and a variety of I/O devices. Communication and data flow between the channel and I/O adapter is through a Standard Interface.

The many commands at its disposal give the IBM 7909 Data Channel expanded capabilities for performing logical and testing operations as well as exercising normal control of data transmission. A variety of I/O devices and appropriate adapters can be attached to a 7909 oriented system.

The IBM 1301 Disk Storage and IBM 7631 File Control provide a capacity of more than 55 million characters of storage for each disk storage unit.

The IBM 7320 Drum Storage and IBM 7631 File Control provide random storage of 1,118,400 characters on 400 data tracks. In the normal six-bit mode, each data track provides the following capacities:

Data Bits	16,776
Character (6-bit)	2,796
Words (36-bit)	466

The IBM 7340 Hypertape Drive and IBM 7640 Hypertape Control introduce a new concept in magnetic tape devices. Character rates as high as 170,000 alphameric characters (28,330 words) a second are possible.

The IBM 7750 Programmed Transmission Control links a central computer with telecommunication equipment such as telegraph machines, IBM 65/66 Data Transceivers, and IBM 7701 Magnetic Tape Transmission Terminal.

The IBM 1414-6 Input/Output Synchronizer permits the attachment of communications and paper tape devices such as:

IBM 1009 Data Transmission Unit
IBM 1011 Paper Tape Reader
IBM 1014 Remote Inquiry Units
Telegraphic Input/Output Units

IBM 7151-2 Console Control Unit

The IBM 7151-2 Console Control Unit provides a manual means of controlling the system and displaying, by indicator lights, the contents of various registers, or any one of the storage locations. Several registers are continually displayed. The console also contains a CE test panel and a marginal voltage check panel. The various lights and switches, and their meaning or operation are discussed in more detail in Volume 3.

IBM 7608 and IBM 7618 Power Control Unit

The power supply used with transistorized circuits in standard modular systems (SMS) has four major parts:

1. The IBM 7608, a power converter or motor-generator set which converts incoming 60-cycle, three-phase, 208-volt power to regulated 400-cycle, three-phase, 208-volt power.
2. Power supplies in each modular frame which supply voltage to all circuits in that frame.
3. The IBM 7618, a power control unit (PCU) which contains system power control circuits, motor-generator (M-G) regulator and marginal check Variacs.*
4. Marginal check (M/C) controls located in the main console.

The output voltage of the motor-generator, a commercially available unit, is regulated, eliminating the need for voltage regulation in most of the modular power supplies. Using 400-cycle AC to rectifiers means that fewer filter capacitors are required in the modular power supplies, as well as smaller transformers.

*Trademark of General Radio Company

Figure 36 shows the logic of the power system. Two power supplies are located in each modular frame; one supply (power supply A) is for gates A and B; the other (power supply C) is for gate D. The power supplies are three-phase, full-wave rectifiers using the 400-cycle output of the generator. All rectifiers are silicon diodes which have large current carrying capacities. The power supplies are physically located at the rear of each slide in the modular frame. The power unit for gates A and B of CPU-I also provides ±3 volt supplies to accommodate the new DIF circuitry.

Marginal checking may be performed on part of the +6-volt supply, part of the −12-volt supply, and core storage driver collector voltages. Marginal check controls for all units (except −12 M for the air memory) are mounted on the console unit.

Each modular power supply has its own open-fuse detection circuit. In 7094 II systems, a blown fuse drops power to only its own frame. Interlock circuits drop power to the second half of the unit when a fuse blows.

The PCU is a separate frame containing circuits which control power to all modular units. Power sequence contactors, power-on and marginal check variacs, blower relays and their overload circuit breakers (CB), and system power control keys are the major circuits in the PCU.

Power for the tape drives is from the wall outlets through the channel module, where the power is interlocked with a data channel power-on relay.

A 55-volt supply and a 46-volt supply located in the printer furnish necessary DC voltages to the card machines.

Functional Components

Figure 37, CPU Data Flow, should be referred to in conjunction with this section. Figure numbers are shown within the various register blocks of Figure 37. These figure numbers refer to more detailed information concerning that particular functional component.

Storage Register—SR (Systems 02.01.00.1-02.01.09.1)

The storage register is a 37-position register. Positions S and 1 through 35 accommodate the standard 36-bit computer word and are comprised of shift cells (SC). Position Q is used as temporary storage when saving AC(Q) during certain logical and sense indicator instructions and is a trigger position instead of a shift cell. This change in circuitry is permissible because simultaneous read-in read-out is not necessary at the SR(Q) position.

The storage register acts as a buffer for words arriving from either the even or odd core storage and provides the only exit from the CPU for data going to core storage. The storage register also serves a variety of functions dependent on the particular instruction being executed.

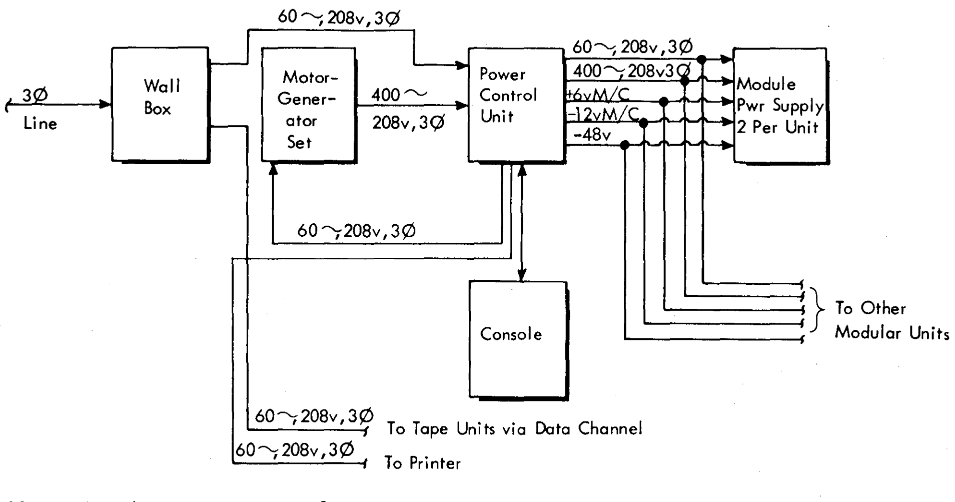


Figure 36. 7094 II Power Distribution

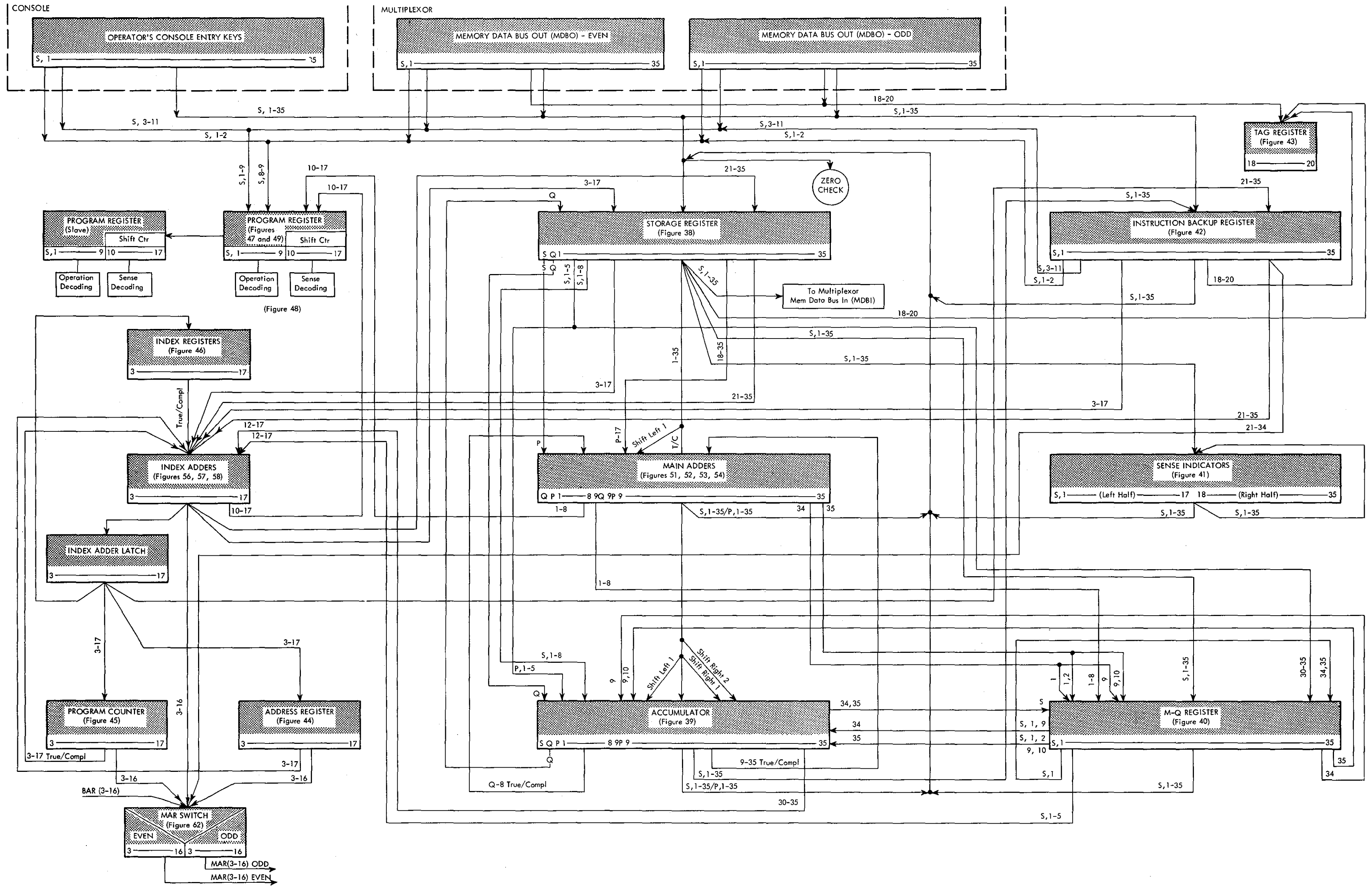

CONSOLE
OPERATOR'S CONSOLE ENTRY KEYS
MULTIPLEXOR
MEMORY DATA BUS OUT (MDBO) - EVEN
MEMORY DATA BUS OUT (MDBO) - ODD
TAG REGISTER (Figure 43)
PROGRAM REGISTER (Slave) Shift Ctr
PROGRAM REGISTER (Figures 47 and 49) Shift Ctr
Operation Decoding
Sense Decoding
(Figure 48)
ZERO CHECK
STORAGE REGISTER (Figure 38)
INSTRUCTION BACKUP REGISTER (Figure 42)
To Multiplexor Mem Data Bus In (MDBI)
INDEX REGISTERS (Figure 46)
True/Compl
INDEX ADDERS (Figures 56, 57, 58)
MAIN ADDERS (Figures 51, 52, 53, 54)
Shift Left 1
SENSE INDICATORS (Figure 41)
(Left Half)
(Right Half)
INDEX ADDER LATCH
ACCUMULATOR (Figure 39)
Shift Left 1
Shift Right 2
Shift Right 1
M-Q REGISTER (Figure 40)
PROGRAM COUNTER (Figure 45)
ADDRESS REGISTER (Figure 44)
3-17 True/Compl
9-35 True/Compl
Q-8 True/Compl
BAR (3-16)
MAR SWITCH (Figure 62)
EVEN
ODD
MAR(3-16) ODD
MAR(3-16) EVEN

Figure 37. 7094 II CPU Data Flow

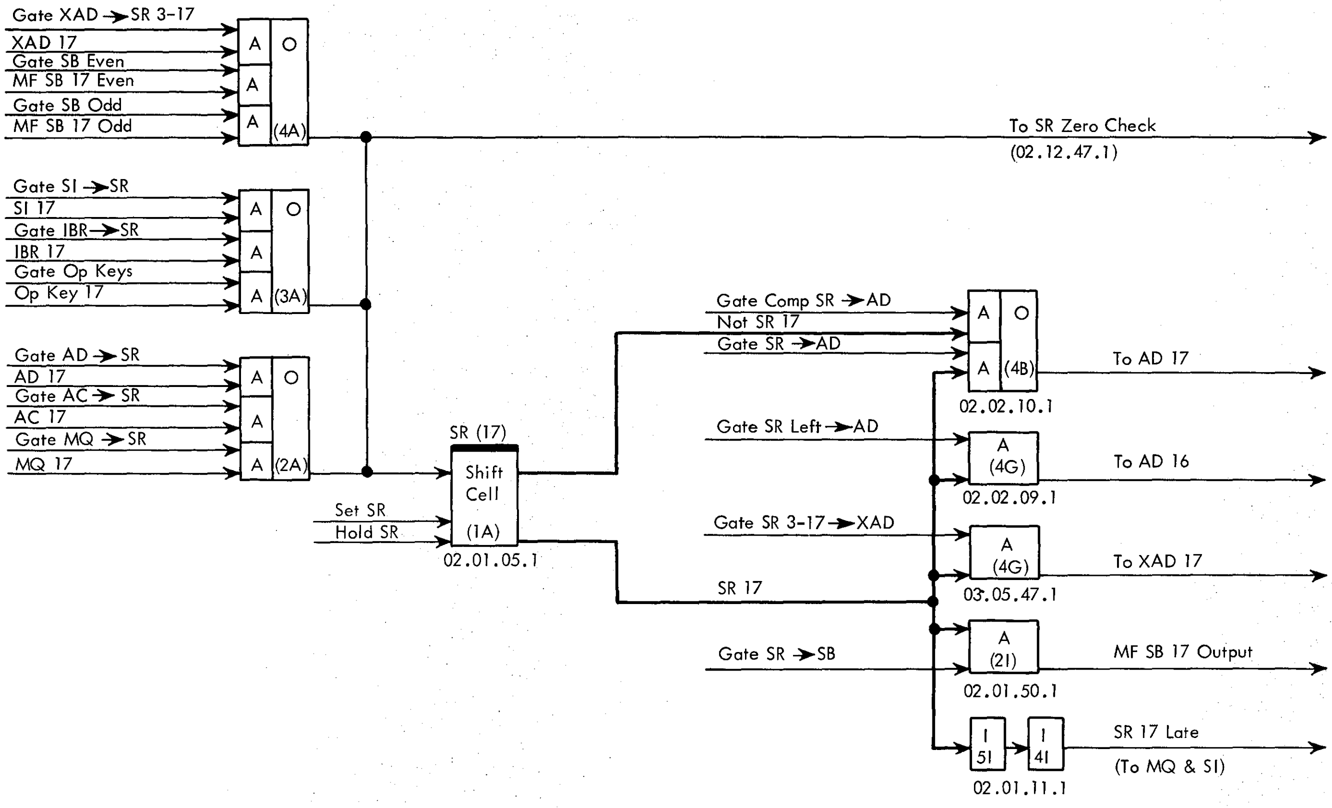


Figure 38. Storage Register Position 17

Figure 38 shows condensed logic of SR(17). Inputs to the storage register as a whole can come from: the op keys; instruction backup register (IBR); sense indicator register (SI); accumulator register (AC); MQ register; main adders (AD); index adders (XAD); and either the even or odd storage bus (even or odd memory).

Note that the index adders can be gated into either SR(3-17) or SR(21-35). This choice in gating is helpful when routing the index registers to either decrement or address positions. SR(Q) also receives AC(Q) as temporary storage during certain instructions as mentioned previously.

Note that the storage register inputs are also gated to a zero check circuit (Systems 02.12.47.1). During many operations, information is gated to the storage register to make use of this circuit in checking for zero values. Because there is no set pulse accompanying the input data, the present storage register contents are not destroyed or affected.

The storage register is the focal point for information distribution and register swapping. Therefore, many outputs are gated from various sections of the storage register to various sections of other registers or adders.

Units receiving information from the storage register include: the storage bus (the storage register is the only exit from the CPU to memory), main adders, index adders, SI register, MQ, accumulator, and tag register (manual operations).

Note that either the true or complement SR values can be sent to the main adders. The true output of the SR can also be shifted left one place as it is sent to the main adders. This shift feature is used during multiplication to effectively multiply by two (2).

There is no full-word routing path from the storage register to the accumulator; data transfer of this type is accomplished by routing the storage register to the main adders and from there to the accumulator. SR(S, 1-8) and SR(S, 1-5) have a direct path to AC(S, 1-8) and AC(P, 1-5) respectively. These paths are used during floating-point operations for characteristic routing and convert instructions. SR(Q) can also be gated directly to AC(Q) when restoring the AC to its original value after certain instructions.

Either SR(3-17) or SR(21-35) can be gated to the index adders. SR(3-17) gating is used primarily during class A (TIX, TNX, etc.) and index transmission (LXD, LDC, PDX, and PDC) instructions. SR(21-35) gating is used primarily as: a routing path for index transmission instructions (LXA, LAC, etc.); POD 76 routing to the shift counter for class and unit decoding; and normal address modification with a specified index register.

Each position of the storage register is combined on the same STAN-PAC circuit card with a corresponding position of the AD, AC, and MQ.

Accumulator Register—AC (Systems 02.03.00.1-02.03.09.1)

The accumulator is a 39-position register, each position consisting of a shift cell (SC). The register positions are labeled S, Q, P, 1-8, 9P, and 9-35. Positions S and 1 through 35 accommodate the computer word in standard operations. Positions Q and P are used as overflow positions because the sum of two 35-position numbers can be greater than 35 positions. Position 9P is also an overflow position used during floating-point operations; it replaces the 9 overflow trigger which was used on previous systems.

The term accumulator is somewhat misleading because the register is not actually able to accumulate. The AC can contain only one word at a time. When adding, the AC and adders work as a unit to perform the addition, and the AC merely holds the result.

Figure 39 shows condensed logic of AC(17). Inputs to the accumulator can come from: the storage register, main adders, adjacent positions of the accumulator, and the MQ. SR(S, 1-8), SR(S, 1-5), and SR(Q) have a direct path to the accumulator. When the entire storage register contents is to be set into the accumulator, however, the main adders are used as a routing path.

Note that the main adders can be gated into the accumulator in a variety of ways: direct (true or complement form for positions Q, 1-8), shifted left one position, shifted right one position, or shifted right two positions. Advantage is taken of this shift feature during execution of arithmetic operations.

Right and left shift operations cause a particular AC position to receive data from adjacent right or left AC positions. This shifting process proceeds at two positions per clock pulse as long as the shift counter value is two or greater. With a shift count value of one, a single position shift is accomplished. Because of these shifts, routing circuits are available to either the first (adjacent) or second left/right position of the AC and MQ registers. Right shifts take advantage of the "shift right 1" and "shift right 2" circuitry from the main adders; for example, the shift is accomplished by routing the AC to the main adders, and from there back into the appropriate AC/MQ positions.

Routing paths are available to AC(9, 10) from MQ(34, 35) and used during the initial phase of DFAD instructions when aligning characteristics.

Outputs from the accumulator can go to a variety of places: the storage register; main adders (true or com-

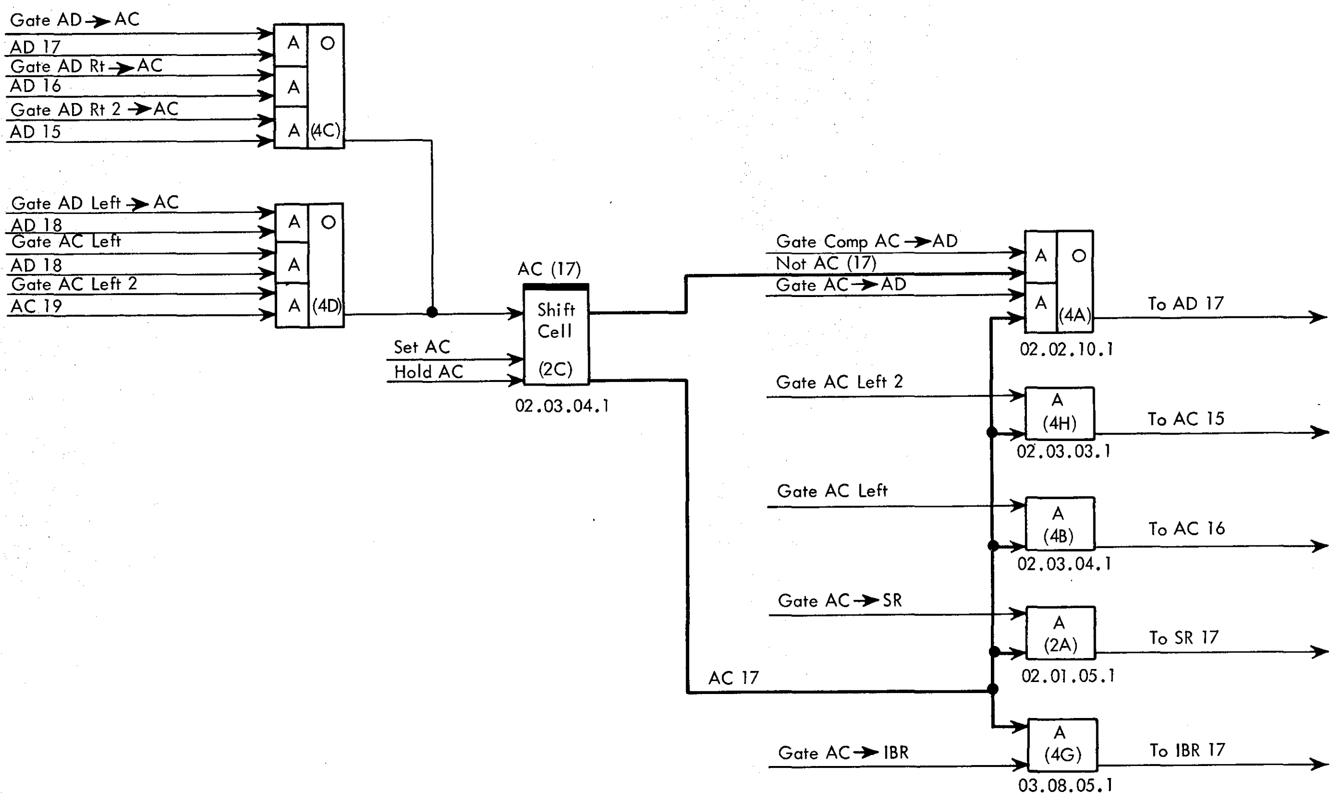


Figure 39. Accumulator Position 17

plement form); index adders (AC positions 30-35 for convert instructions); IBR; and adjacent positions of the AC or MQ during shift operations. The accumulator is sent to the input of the storage register in many cases (without a set pulse) to take advantage of the zero check circuitry.

Each position of the accumulator is combined on the same STAN-PAC circuit card with a corresponding position of the SR, AD, and MQ.

Multiplier-Quotient Register—MQ (Systems 02.04.00.1-02.04.06.1)

The MQ is a 36-position register, each position consisting of a shift cell (SC). The MQ receives its name from functions performed. At the start of a multiply operation it contains the multiplier; after the multiply it contains the least significant half of the product. At the beginning of a divide operation, it contains the least significant half of the dividend; after the divide it contains the quotient. Due to the double latch makeup of the shift cell, the MQ has the ability of shifting left or right.

Figure 40 shows condensed logic of MQ(17). Inputs to the MQ can come from the storage register, accumulator, or main adders. The SR outputs may come to the MQ as a full word, or SR(S, 1-5) may be gated to MQ(30-35) for convert instruction operations.

Inputs from the adders during multiplication are from AD(34-35) to either MQ(1-2) or MQ(9-10), depending on fixed-point or floating-point operations. The AD(34-35) routing paths are also used as inputs to MQ(1, 2) when performing right shift operations.

Outputs from the MQ go to a variety of places. Gatings to the storage register are used for MQ store operations (the SR is the only output register to the storage bus) or for register swapping during arithmetic operations. MQ(S, 1-5) are gated to the index adders to produce required core storage addresses during convert operations. MQ(34, 35) are gated into AC(9, 10) during DFAD operations when aligning the fractions prior to the actual addition. Gating is provided for normal right and left shifts within the MQ or between the MQ and AC. Outputs are also available from MQ(S, 1) to MQ(34, 35) for ring shift (RQL) operations.

Each position of the MQ is combined on the same STAN-PAC circuit card with a corresponding position of the SR, AD, and AC.

Sense Indicator Register—SI (Systems 02.06.00.1-02.06.11.1)

The sense indicator register is a 36-position register labeled S, 1-35. This register serves a variety of functions. It can be used as a set of switches which are set and tested by the program to check the progress or direction of the program. It may also be used to store words or parts of words temporarily; in this way the register is useful in altering and testing words.

During some logical or masking operations, SI(S) is referred to as SI(0); in these cases the first position has lost its identity as a sign and has become just another bit in the overall group of bits being operated on.

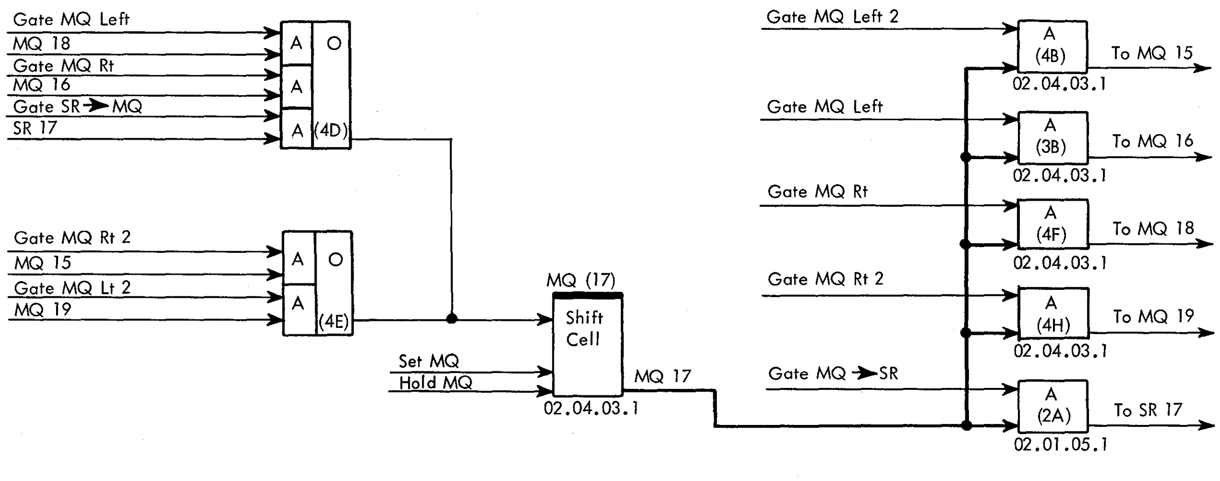


Figure 40. MQ Position 17

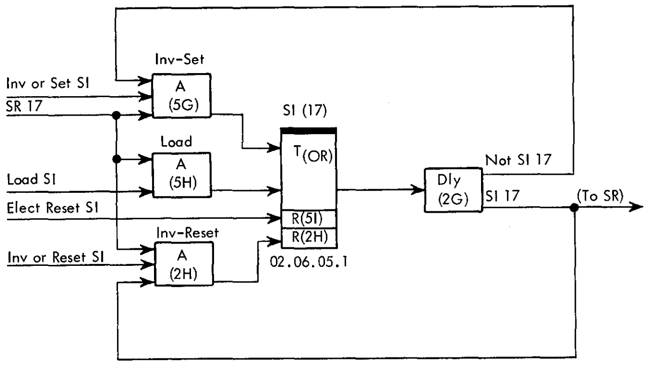


Figure 41. Sense Indicator Position 17

During double-precision floating-point operations the SI register is used extensively for temporary storage of intermediate results. In these cases the S position retains its identity as an arithmetic sign.

Each sense indicator position is composed of a single trigger-type position with a delayed output (Figure 41). The delayed output holds the trigger information long enough to allow proper input sampling during operations requiring SI register inversion. The only path for information into or out of the SI register is by way of the storage register.

Instruction Backup Register—IBR (Systems 03.08.00.1-03.08.11.1)

The IBR is a 36-position trigger register labeled S, 1-35 and is used primarily during instruction overlap operations. As long as the instruction sequence permits, the IBR contains the next sequential instruction to be executed. Because of the simultaneous readout feature from both the even and odd core storages, the IBR is able to obtain a next sequential instruction during the same cycle that the current instruction is being placed into the program register. Simultaneous memory readout also allows the IBR to obtain the next sequential instruction simultaneously with a data fetch (E) cycle of the current instruction. The first case occurs during "double-instruction" overlap and the second applies to "extended-sequence" overlap. The IBR is also used as a working register during double-precision floating-point and ERA instructions. Figure 42 shows condensed logic of IBR position 17.

Inputs to the IBR can come from: either the even or odd storage bus when receiving a new instruction; the accumulator during double-precision register swapping or ERA operations; XAD(3-17) to IBR(21-35). In the latter case, the index adder is either returning a modified IBR instruction address or sending the incremented program counter value to the IBR for temporary storage.

Input gating is such that the outputs from the IBR immediately follow the inputs from the storage bus. This feature (the same as for the program register to be discussed later) provides the earliest possible decoding of instructions as they are read for core storage, and is necessary for controlling overlap functions. This input gating feature does not apply, however, for data arriving from the accumulator.

Outputs from the IBR can go to a variety of places: IBR(S, 1-35) are gated to the SR during both overlap and nonoverlap operations; IBR(S, 1-2) or IBR(S, 3-11) are gated to the program register at a time when the current instruction has completed operation and the overlapping (IBR) instruction is to begin execution; IBR(18-20) is sent to the tag register for indexing indications; IBR(3-17) are gated to the index adders for index register testing or modification during overlapping class A instructions such as TIX, TXI, and TNX; IBR(21-35) are also gated to the index adders so that address modification can be performed by the specified index register and the new address returned to the IBR; IBR(21-34) are gated to the MAR switch for an appropriate instruction or data fetch memory reference.

Extensive decoding circuitry at the IBR analyzes the overlapping instruction to determine the various aspects of overlap possibilities and their operation.

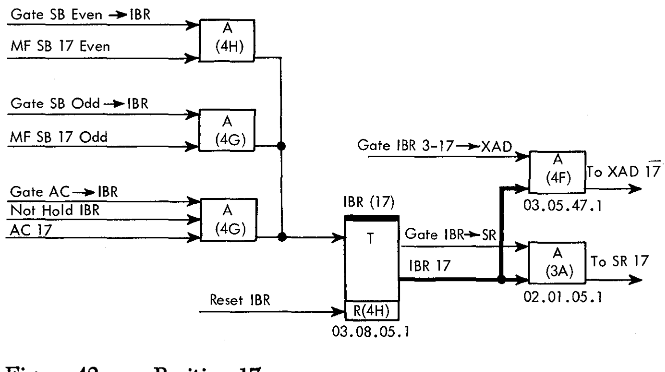


Figure 42. IBR Position 17

Tag Register—TR (Systems 03.05.22.1)

The tag register is a three-position trigger register labeled 18, 19, 20. This register is used to inform the system which index register (XR) or registers are concerned with a particular operation. The outputs of the tag register may be used singly or in combination to specify the index register/registers required.

When operating in seven-XR mode, the tag register uses all combinations of these tag bits to specify the particular index register. Therefore, the presence of more than one tag bit does not mean the OR'ing together of index register contents.

The 7094 II does provide compatibility with previous systems having only three index registers. The normal power-on status of the 7094 II is the three-XR mode. By using the instructions LMTM and EMTM, the programmer has the ability to place the computer in either seven-XR or three-XR mode, respectively. For a further explanation, refer to the section on "7090/7094/7094 II Compatibility" in Volume 3.

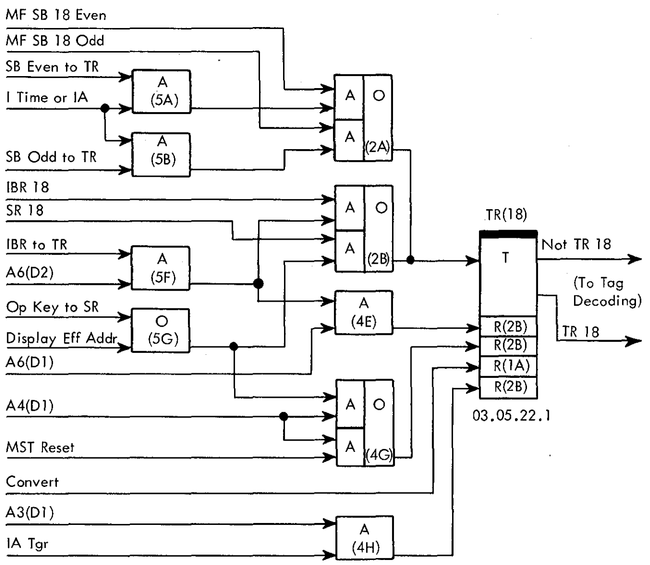


Figure 43. Tag Register Position 18

Figure 43 shows condensed logic for TR(18). Inputs to the tag register come from: either the even or odd storage bus during normal instruction loading from memory; the IBR during instruction overlap; or the storage register during manual operations.

Outputs go to the tag register decoder circuits which are conditioned by either three-XR or seven-XR mode of operation.

Address Register—AR (Systems 03.06.00.1-03.06.01.1)

The address register is a 15-position trigger register labeled 3-17.

The primary purpose of the register is to provide transfer or data reference addresses to core storage. These references may be: a direct address; an effective address as modified by an index register; an indirect address; or a skip 1/2 address as required by skip-type instructions.

The only gated input to the address register (Figure 44) is from the index adders which are used as either an incrementing, modifying, or direct routing path from other counters or registers. Positions AR(3, 16 and 17) can also be forced on as a direct result of control logic resulting from certain trapping operations.

The address register is not a counter; any address register incrementing is accomplished through use of the index adders.

Outputs from the address register go to the index adders when forming skip 1/2 addresses, or the MAR switch when making references to core storage. The output of AR(17) is sent to the MAR bus selection circuitry (Systems 03.06.28.9). This circuitry, depending on whether AR(17) is a 0/1, controls address register gating to either the even or odd memory during normal mode of operation. When in diagnostic mode, however, AR(17) becomes an integral part of the address; this latter condition, when applicable, forces a bit to be gated to MAR(3) of either the even or odd memory. AR(3 or 17) also effects the "AR odd trigger" which in turn controls gating circuitry for either the even or odd storage bus into the CPU.

Indicator lights for the address register are located in the CE test panel area of the operator's console.

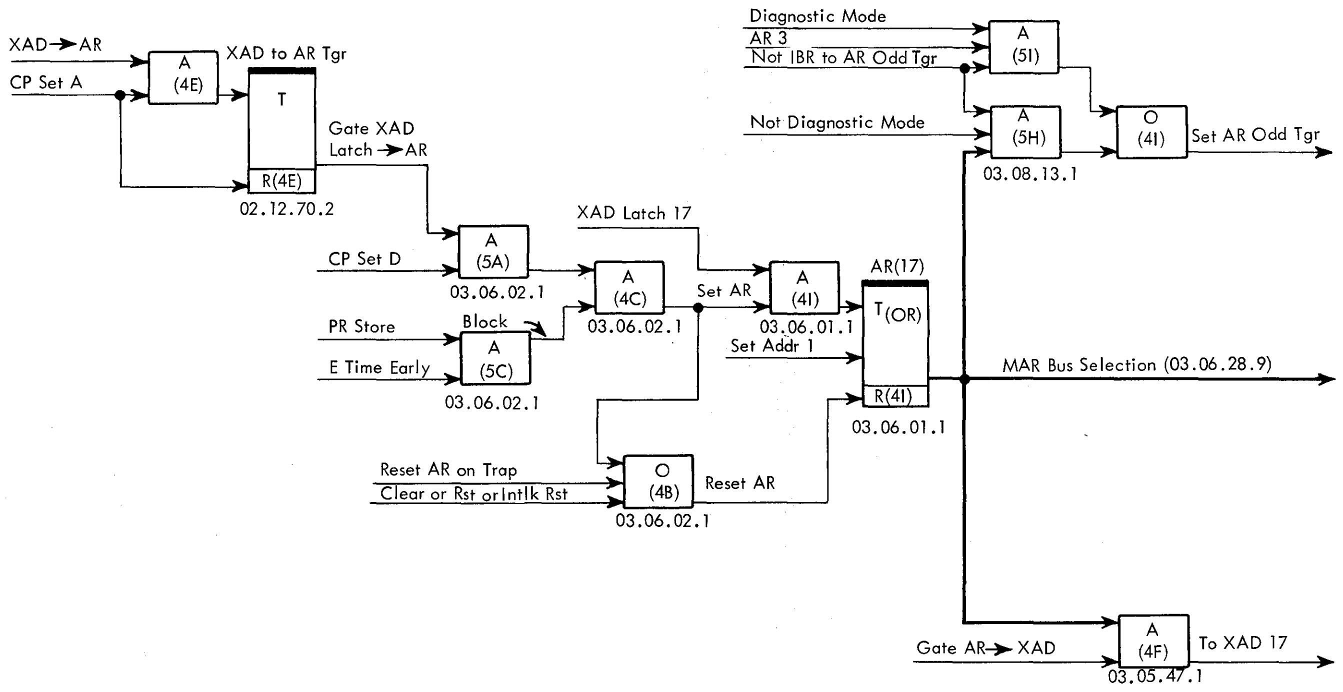


Figure 44. Address Register Position 17

Program Counter—PC (Systems 03.06.30.1-03.06.31.1)

The program counter (labeled "Instruction Counter" on the operator's console) is a 15-position trigger register labeled 3-17.

The primary purpose of the program counter is to keep account of the next sequential instruction in a running program.

The program counter is not a counter; any incrementing or decrementing is accomplished through use of the index adders. During normal sequential instruction execution, the new address associated with the program counter is generated at 6-time of the cycle when a reference is made to core storage. This new address is placed temporarily in either the address register or IBR until 1-time and then returned to the program counter via the index adders. Note that when updating the program counter at 1-time, the index adders are used as a routing path only, and do not affect the value being transmitted.

The only gated input to the program counter (Figure 45) is from the index adders. As previously explained for the address register, the index adders act as either an incrementing, modifying, or direct routing path from the other counters or registers. PC(3, 16, and 17) can also be forced on directly by control logic resulting from certain trapping operations.

Note that either true or complement outputs can be sent to the index adders. The true output is used during incrementing, decrementing, or routing of the program counter to other registers. The complement PC output is used only during overlap store operations to check if the store instruction (in the program register) is storing in the next sequential core storage location. If this is the case, overlap is interrupted because the overlapping instruction (now in the IBR) is being modified by the program and therefore invalid as it stands in the IBR. The program counter is also sent to the MAR switch when making references to core storage.

The output of PC(17) is sent to the MAR bus selection circuitry (Systems 03.06.29.2) for the same purpose as previously explained for the address register. This circuitry, depending on whether PC(17) is a 0/1, controls program counter gating to either the even or odd memory during normal mode of operation. In diagnostic mode, however, PC(17) becomes an integral part of the address, and when applicable (for example, when PC(17) equals 1) forces a bit to be gated to MAR(3) of either the even or odd memory.

PC(17), indicating an even or odd status, also controls gating circuitry from the even or odd storage busses into the program register, storage register and IBR. It must be remembered that the program counter is stepped +1 prior to the time that this gating circuitry is used. Because of this prior stepping, an odd PC indication, for example, gates the even storage bus to the SR and PR, and the odd storage bus to the IBR. Storage bus gating is explained later in the "Master I-Time" section.

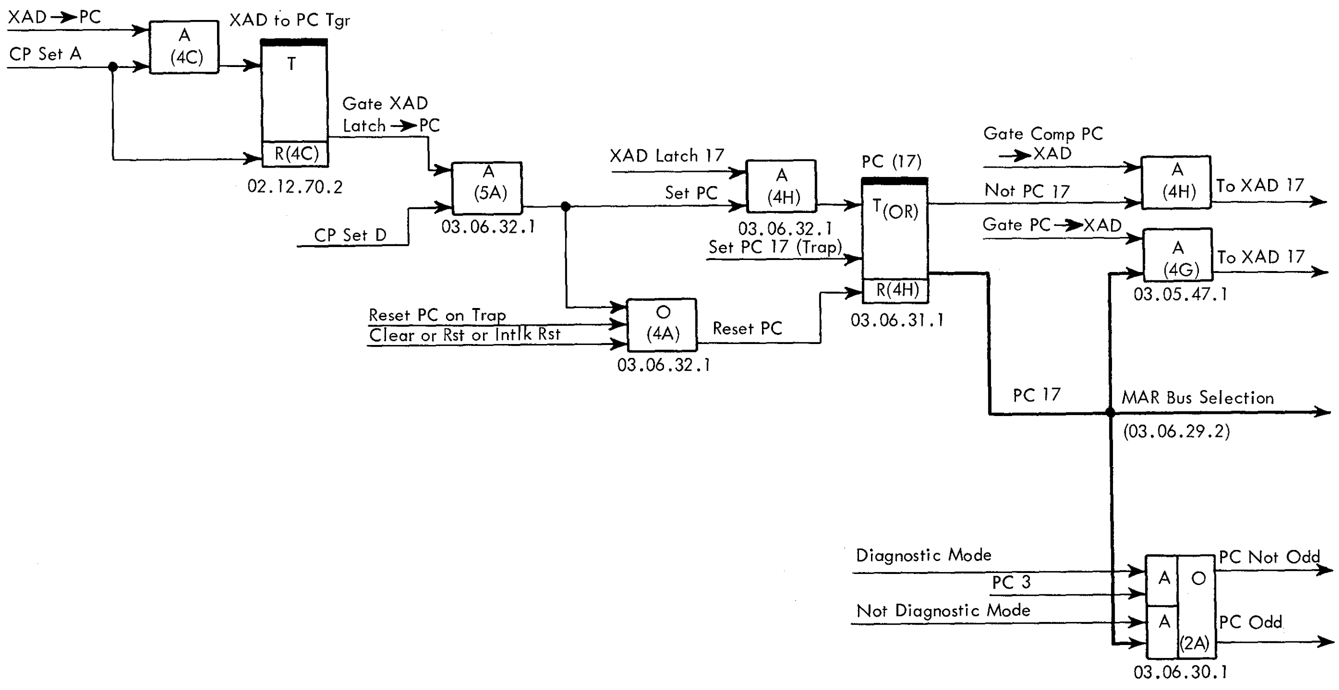


Figure 45. Program Counter Position 17

Index Registers—XR (Systems 03.05.00.1-03.05.14.1)

The 7094 II has seven index registers. Each is a 15-position trigger register labeled 3-17. All seven XR's are identical and used primarily for address modification. The 7094 II can be in either three-XR or seven-XR mode of operation. In three-XR mode (multiple tag mode), multiple tag bits cause an OR'ing of index registers.

Input to the index registers (Figure 46) is from the index adders which are used, in this case, as either a modifying or direct routing path from other registers within the CPU.

Individual positions of all seven XR's have a common output which is gated (under control of tag register decoding) to the index adders in either true or complement form. Because of the true XR outputs, it is possible to accomplish true addition in the index adders as is required during TXI operations. True outputs are also used to advantage during PXA, PXD, SXA, and SXD operations when true values are moved to the accumulator or core storage.

Tag register decoding gates the complement outputs of the specified index registers to the index adders for a variety of operations, the primary one being address modification. By addition of the complement XR value to an address, the address is effectively reduced by the contents of the XR.

There are many instructions which operate on the index registers, thereby making these registers also useful programming tools for counting, word alteration, and program loop control.

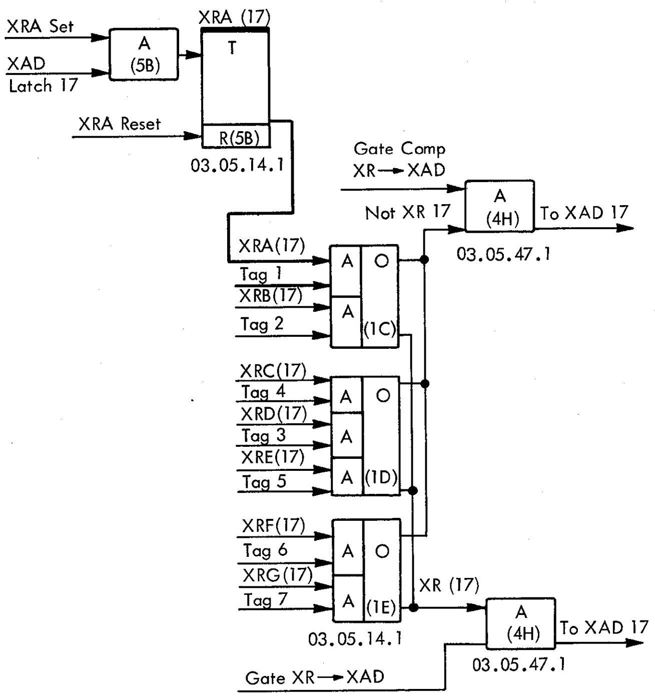


Figure 46. XRA Position 17

Program Register—PR (Systems 03.14.01.1-03.14.06.1)

The program register is a ten-position trigger register. The purpose of this register (labeled "Instruction Register" at the operator's console) is to receive and decode the operation portion of the instruction to be executed. Decoder outputs then initiate and control the computer operation until the instruction is completed.

The operation code depends on the type of instruction involved and consists of either positions S, 1-2, or S, 3-11 of the instruction word. Positions S, 1-2 are routed to PR(S, 8-9); positions S, 3-11 are routed to PR(S, 1-9).

Primary inputs to the program register (Figure 47) come from: either the even or odd storage bus; or the IBR depending on whether or not instruction overlap is involved. Outputs from the PR follow (rise with) the inputs from the storage bus. This feature of the input gating provides the earliest possible decoding of instructions as they are read from core storage. (During memory read-out, the S-bit is strobed before the 35-bit.) Inputs from the IBR or op keys are gated in with a clock pulse and, therefore, differ from SB input gating.

Certain trapping operations force specific PR triggers on directly—for example, PR(S and 9)—without having to rely on storage bus or IBR inputs.

During manual operations, the program register can be entered from the console op keys to allow execution of specific instructions set up by the operator or customer engineer.

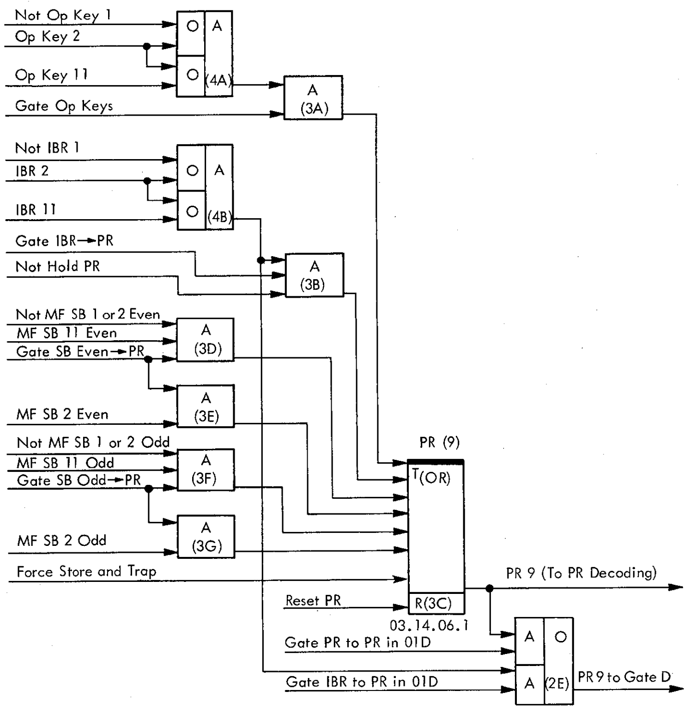


Figure 47. Program Register Position 9

Outputs of the program register feed into the operation decoder circuitry (Figure 48). PR(S, 1-5) provides the primary operation part and PR(6-9) provides the secondary operation part of the operation decoding. During some instructions, the shift counter becomes an extension of the program register and provides a class and unit address decoding.

Note that the 7094 II actually has two program registers: the "master" register, the one which receives inputs directly from the storage bus or IBR, is located in A-gate of CPU-1 (7111) and the second (slave) program register is the register that existed in the 7094 system. The "slave" register (Systems 03.04.00.1-03.04.06.1) is located in D-gate of CPU-1 and is not really a register as such; converter circuit blocks have replaced the triggers so that these outputs follow the outputs of the master register as gated to the D-gate (Figure 47).

Basically, the new program register provides new and faster F-level circuitry necessary in the 7094 II. The old register provides P/N level operation decoding and gating in CPU-2 for instructions that do not have such critical timing requirements. The functions of the two program registers and operation decoders cannot be cleanly separated; therefore, both must be considered when analyzing certain instruction operations.

Indicators at the operator's console are powered from the old program register.

Shift Counter—SC (Systems 03.14.14.1)

The shift counter is an eight-position count-down counter labeled 10-17. Each position consists of a shift cell (double latch) which allows simultaneous read-in and read-out when counting.

The shift counter is used to count the number of shifts or indicate the progress of operations such as MPY, DIV, convert, floating-point add, shifting instructions, etc. The shift counter also functions as a class and unit address decoder for operations that have a primary operation of 76 (Figure 48).

The primary entrance to the shift counter is by way of the index adders. The time at which the number is gated is controlled by the operation being performed. During POD 76 operations, the SC is set either during I5 time from PR decoding or at the beginning of L time when the IBR is transferred to the PR during overlap. During variable length multiply or divide operations, the count field is routed from the storage register via the index adders in E time; during convert instructions, the count is routed in the same manner in L time.

The shift counter is also used during single and double-precision floating-point add and subtract operations (POD 30). During these operations the shift counter is used in lining up characteristics. The characteristic difference between the numbers in the SR and AC is

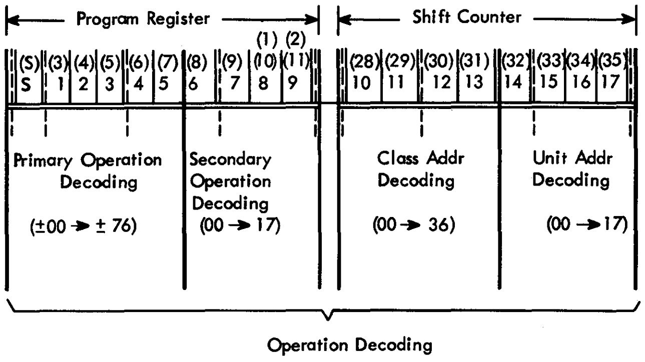


Figure 48. Operation Decoding

computed in the main adders, AD(1-8), and then gated directly to SC(10-17) to control the number of shifts necessary to align the fractions.

During multiply and divide operations (other than variable length), a constant is forced into the shift counter to control the number of iterations. In the case of a floating-point operation, 33_8 is set into the shift counter; for a fixed-point operation, 43_8 is set into the shift counter.

Note that the 7094 II actually has two shift counters: the "master" register, the one which receives the inputs just described, is located in A-gate of CPU-1 and the second "slave" shift counter is the counter that existed in the 7094 system. This "slave" register (Systems 03.04.14.1-03.04.17.1), located in D-gate of CPU-1, is not really a register as such and has no counting capabilities; converter circuit blocks have replaced the triggers so that these outputs follow the outputs of the master counter as gated to the D-gate.

As for the program register described previously, the new shift counter provides fast F-level control and counter circuitry necessary in the 7094 II. The old register, again, provides P/N-level controls for instruction operation in CPU-2; the main function is in class and unit decoding and POD 76 controls. The functions of the two counters overlap somewhat; therefore, both must be considered when analyzing certain instruction operations.

Indicators at the operator's console are powered from the old shift counter.

Stepping the Shift Counter

Figure 49 shows condensed logic for positions 10-17 of the shift counter. Remember that the shift counter is a self-contained true binary count-down counter and does not rely on the index adders for decrementing as is necessary with the address register and program counter.

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}
SC Value	33	31	27	25	23	21	17	15	13	11	7	5	3	1	0
SC (10)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
SC (11)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
SC (12)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
SC (13)	N	N	N	N	N	N	F	F	F	F	F	F	F	F	F
SC (14)	N	N	F	F	F	F	N	N	N	N	F	F	F	F	F
SC (15)	F	F	N	N	F	F	N	N	F	F	N	N	F	F	F
SC (16)	N	F	N	F	N	F	N	F	N	F	N	F	N	F	F
SC (17)	N	N	N	N	N	N	N	N	N	N	N	N	N	N	F

N = On
F = Off

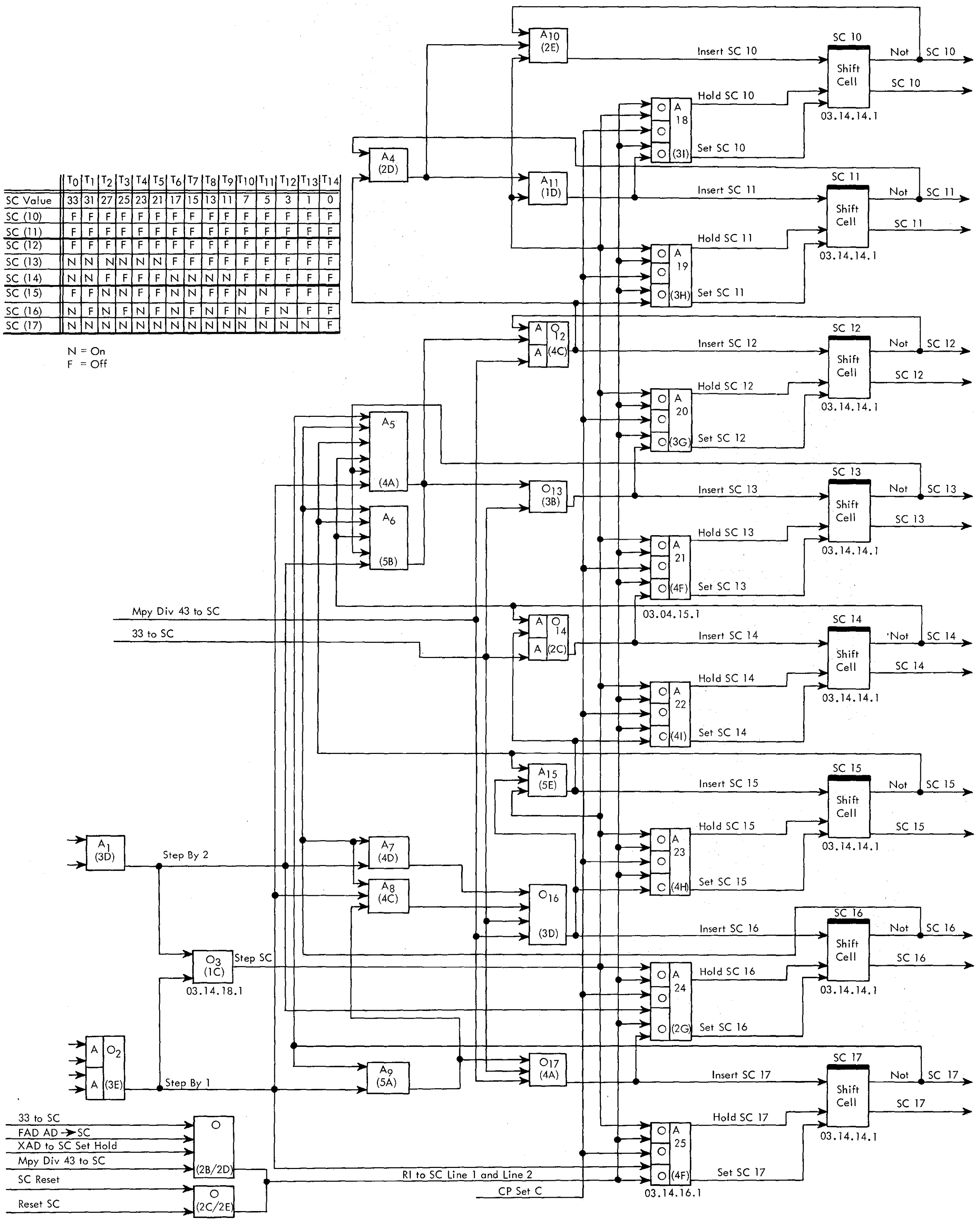


Figure 49. Shift Counter Stepping

Figure 50 shows a sequence chart of the shift counter stepping to zero from a count of seven.

Note that stepping the shift counter is conditioned by either "step by 1" or "step by 2" control circuitry. Whether the stepping is by one or by two depends on the particular instruction being executed. Stepping by two is allowed during multiplication, floating-point add/subtract, and shift-type instructions; stepping by one must be performed during divide and convert-type instructions.

Each position of the shift counter has control circuitry affecting three inputs to the shift cell ("insert SC 17," "set SC 17," and "hold SC 17"). The "insert" input is activated *only* when that particular position is to be turned on. The "set" and "hold" pulses are both CP set pulse inversions of one another, and generated from the same control circuitry (OA blocks) *only* when the status of that particular position is to be affected—turned off or on.

Whether a shift counter position is to be changed or not by a set/hold pulse combination is determined by the next lower position. If the next lower position shift cell is to be set on, the state of the next higher position will be reversed. This rule applies for both single and double stepping.

When stepping large numbers, the ripple effect from position to position can become too great for reliable operation. To reduce this ripple time, look-ahead circuitry has been inserted at the controlling circuitry of SC(13).

Note that SC(16) alternately changes state between ON and OFF during double stepping and SC(17) alternates state during single stepping. The primary difference between single and double step operations is where the step pulse control comes into the counter; for example, SC(17) for single step and SC(16) for double step. Double stepping is, therefore, the same as single stepping except that it occurs one rung higher on the ladder.

Assume that a count of 33_8 has been set into the shift counter and that counting is by 2. The initial status of the counter with 33_8 is: positions SC(17), SC(16), SC(14), and SC(13) are in the ON position; all remaining positions are OFF. Because the original number is odd, each stepping of the counter produces an odd result. If the original number was even (6_8, for example) each double step would, in turn, produce an even result. Note that all during double stepping, SC(17) remains unaffected.

The first step reduces the counter from 33_8 to 31_8. This reduction is accomplished by reversing the state of SC(16) from ON to OFF. SC(16) being initially on, blocks A_7 from activating "insert SC 16." Set and hold pulses, however, are produced at CP set time by OA_{24} with the overall effect of inserting a zero into SC(16).

The second step which reduces the counter from 31_8 to 27_8 initially finds positions SC(17), SC(14), and SC(13) on; and all remaining positions off. SC(16) being off conditions A_7 and O_{16} to change the state of SC(16) to ON. "Insert SC 16" conditions OA_{23} to produce set and hold pulses for SC(15), and also test the initial status of SC(15) at A_{15}. SC(15) being off, activates "insert SC 15" to flip SC(15) on. "Insert SC 15" also conditions SC(14) circuitry to produce only the set and hold pulses thereby turning SC(14) off. At the end of the clock pulse, positions SC(17), SC(16), SC(15), and SC(13) are on and all remaining positions are off.

This pattern of double-stepping continues until all shift counter positions except SC(17) are turned off. As the count is reduced from three to one, "step by 1" control circuitry replaces "step by 2." "Step by 1" tests the status of SC(17) at A_9, but SC(17) being on blocks "insert SC 17." A set and hold pulse is produced at OA_{25}, however, with the result of turning SC(17) off. At this point the shift counter equals zero and all stepping controls are dropped.

System Page	Test Point	Line Name	Level
02.11.79.1	02A4E13D	Shift Gate	-N
03.14.20.1	01A4G26C	SC 2 or More	-F
03.14.20.1	01A4G23F	SC Eq 1	-F
03.14.18.1	01B2B28B	SC Zero Slow	-F
03.14.18.1	01A4D24D	Step by 2	-F
03.14.18.1	01A4G25B	Step by 1	-F
03.14.18.1	01A4A25B	Step SC	-F
03.04.16.1	01A4F28D	CP Set C	-F
03.14.14.1	01A4E24B	SC 15	-F
03.14.16.1	01A4F27G	Insert SC 15	-F
03.14.16.1	01A4F27F	Set SC 15	-F
03.14.14.1	01A4F20C	SC 16	-F
03.14.16.1	01A4E27A	Insert SC 16	-F
03.04.16.1	01A4F27C	Set SC 16	-F
03.14.14.1	01A4E26B	Set SC 17	-F
03.14.16.1	01A4F25B	Insert SC 17	-F
03.04.16.1	01A4F27E	Set SC 17	-F

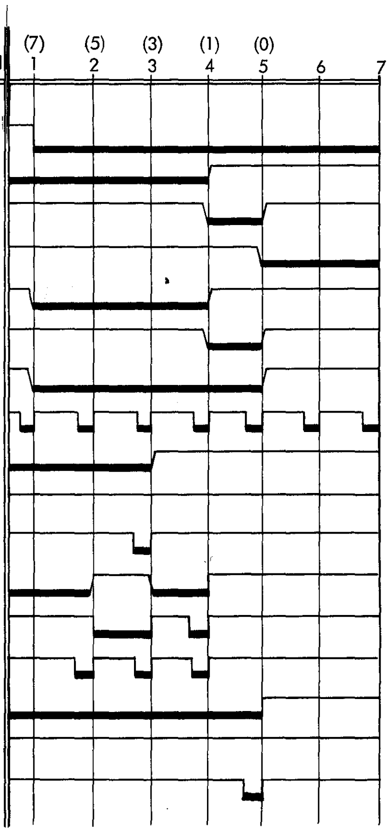


Figure 50. Shift Counter Timing Chart

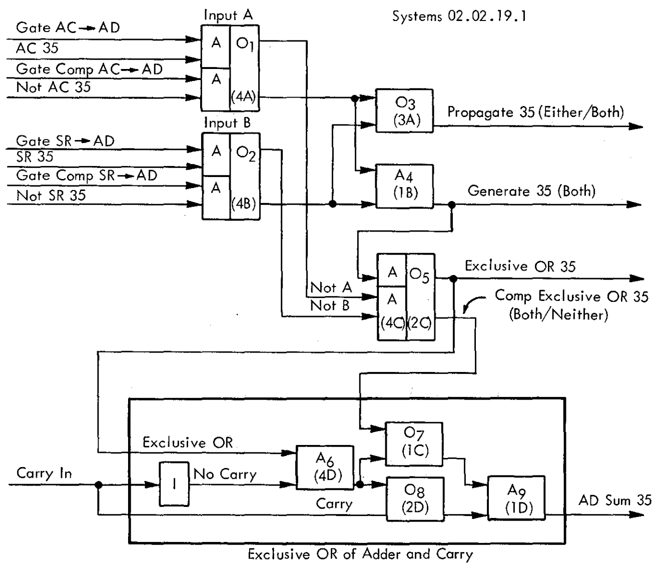
Main Adders—AD (Systems 02.02.00.1-02.02.19.1)

The adders in the 7094 II are made up of basic building blocks shown in Figure 51. The adders perform arithmetic functions and are involved in many internal data transfers within the computer. Inputs to the basic adder circuit may be from either the accumulator or storage register. Outputs from the adders go to the accumulator, MQ, storage register, and shift counter, depending on the operation involved.

The adders are comprised of 39 individual adder units, or bit positions. These positions include AD (Q, P, 1-8, 9Q, 9P, 9-35); AD (Q and P) are used for overflows which might occur during certain arithmetic or logical operations. AD (9Q and 9P) are used during floating-point arithmetic.

Individual Main Adder Position

Figure 51 shows condensed logic for position 35 of the main adders. The primary objective of an adder position is to determine whether or not there is a sum output. To determine the output, three values must be analyzed; two adder inputs and one carry input.



Adder Inputs	0	1	0	1	0	1	0	1	SR
	0	0	1	1	0	0	1	1	AC
	0	0	0	0	1	1	1	1	Carry
Adder Outputs	-	+	+	+	-	+	+	+	Propagate 35
	-	-	-	+	-	-	-	+	Generate 35
	-	+	+	-	-	+	+	-	Exclusive OR 35
	-	+	+	-	+	-	-	+	AD (Sum) 35

+ Indicates an active logical state
- Indicates an inactive logical state

Figure 51. Main Adder Position 35

True or complement inputs from the storage register and accumulator are gated into the adder at gate circuits AO$_1$ and AO$_2$. Either/both adder inputs being active produces "propagate 35" at O$_3$; both inputs being active produces "generate 35" at A$_4$. Block AO$_5$ performs an exclusive-OR test of the adder inputs. An input exclusive-OR condition occurs only when either one but not both of the inputs contains a 1. AO$_5$ performs the test by looking for a "both" (generate 35) or "neither" condition which is actually a complement exclusive-OR function.

The in-phase (complement exclusive-OR) and out-of-phase (exclusive-OR) outputs of AO$_5$ are tested along with input carry conditions in a second exclusive-OR circuit configuration to determine the actual adder sum output. One adder output condition occurs at A$_6$ with "no carry" and "EX OR 35" (case 1 below); the other adder output condition occurs at O$_7$ and O$_8$ with "Comp EX OR" and "Carry" conditions, respectively (Case 2).

CASE 1	(a)	(b)
Input A	1	0
Input B	0	1
Input Carry	0	0
Adder Output	1	1

CASE 2	(a)	(b)
Input A	0	1
Input B	0	1
Input Carry	1	1
Adder Output	1	(1)1

Taking the other four possible combinations of carry and adder input conditions will prove that an adder sum output is not produced. All combinations are summarized in the chart on Figure 51.

Main Adder Bit Carry Lookahead

Each adder position, besides being able to determine a sum condition, must be able to provide necessary carries to adjacent positions. For determining possible carries, three additional lines are generated from each adder position: propagate (P), generate (G), and exclusive-OR (EX OR)—Figure 51.

The output of O$_3$, "propagate 35," is active when either one or both of the adder inputs are a 1. With these input conditions, an input carry must logically be passed on, propagated, to adder position 34. This carry requirement is proven below by taking the input combinations of 0-1, 1-0, and 1-1 and adding a carry; in each case, a carry results. (Also, see chart on Figure 51.)

	(a)	(b)	(c)
Input A	0	1	1
Input B	1	0	1
Input Carry	1	1	1
	10	10	11

The output of A$_4$, "generate 35," is active whenever both inputs to the adder position contain a 1. With both inputs active, the adder position has the ability to generate a carry from its inputs alone independent of whether or not an input carry is furnished from an outside source or a lower adder position in the group. Proof of this carry is easily obtained below and summarized in the chart on Figure 51. Note in the chart that "generate 35" is not dependent on the input carry status.

	(a)	(b)
Input A	1	1
Input B	1	1
Input Carry	0	1
	10	11

The output of AO_5, "EX OR 35," represents an adder input exclusive-OR condition; either one but not both inputs are a 1.

With the propagate, generate, and EX OR conditions just explained, individual adder bit carry lookahead can be developed. This lookahead is shown by the solid lines and logic in Figure 52.

Using adder 35 as an example, a "carry in 34" will be generated if: adder 35 can generate a carry; or adder 35 can propagate and there is a carry into position 35 (K in). Note that the lookahead circuitry becomes more involved as it progresses from the lowest-order to the highest-order adder position within the adder group.

The "carry in 31" circuitry uses exclusive-OR inputs instead of propagate as used in the lower-order positions of the adder group. This change was made because of circuit loading; the basic circuit logic, however, remains the same.

Note in Figure 52 that there is no individual adder lookahead circuitry for carry generation into adder 30. Adder 30, which is the low-order position of the next group, receives its carry indication from the group lookahead circuitry.

An important point to remember in the carry lookahead concept is that each input carry generation is dependent solely on inputs to individual adder positions; there is no "ripple" effect from one adder position to another as is characteristic of slower type adders.

Main Adder Group Carry Lookahead

If the lookahead were continued as explained in the previous section, the circuitry would soon become too large, unwieldly, and expensive. Therefore, to speed up adder operation and facilitate lookahead, the adder has been divided into seven groups as follows:

GROUP NUMBER	ADDER BIT POSITIONS
1	35, 34, 33, 32, 31
2	30, 29, 28, 27, 26
3	25, 24, 23, 22, 21, 20
4	19, 18, 17, 16, 15, 14
5	13, 12, 11, 10, 9, 9P
6	9Q, 8, 7, 6, 5
7	4, 3, 2, 1, P, Q

Each group contains either four, five, or six adder positions. Adder position 9Q is shown in group six but actually plays no part in the lookahead function.

The primary objective of the group lookahead is to determine whether or not carries should be introduced into the low-order adder positions of higher adder groups. Lookahead circuitry for group 1 is shown as dotted lines on Figure 52, lookahead circuitry for group 2 is shown in Figure 53. Overall lookahead for all seven adder groups is shown as solid lines on Figure 54.

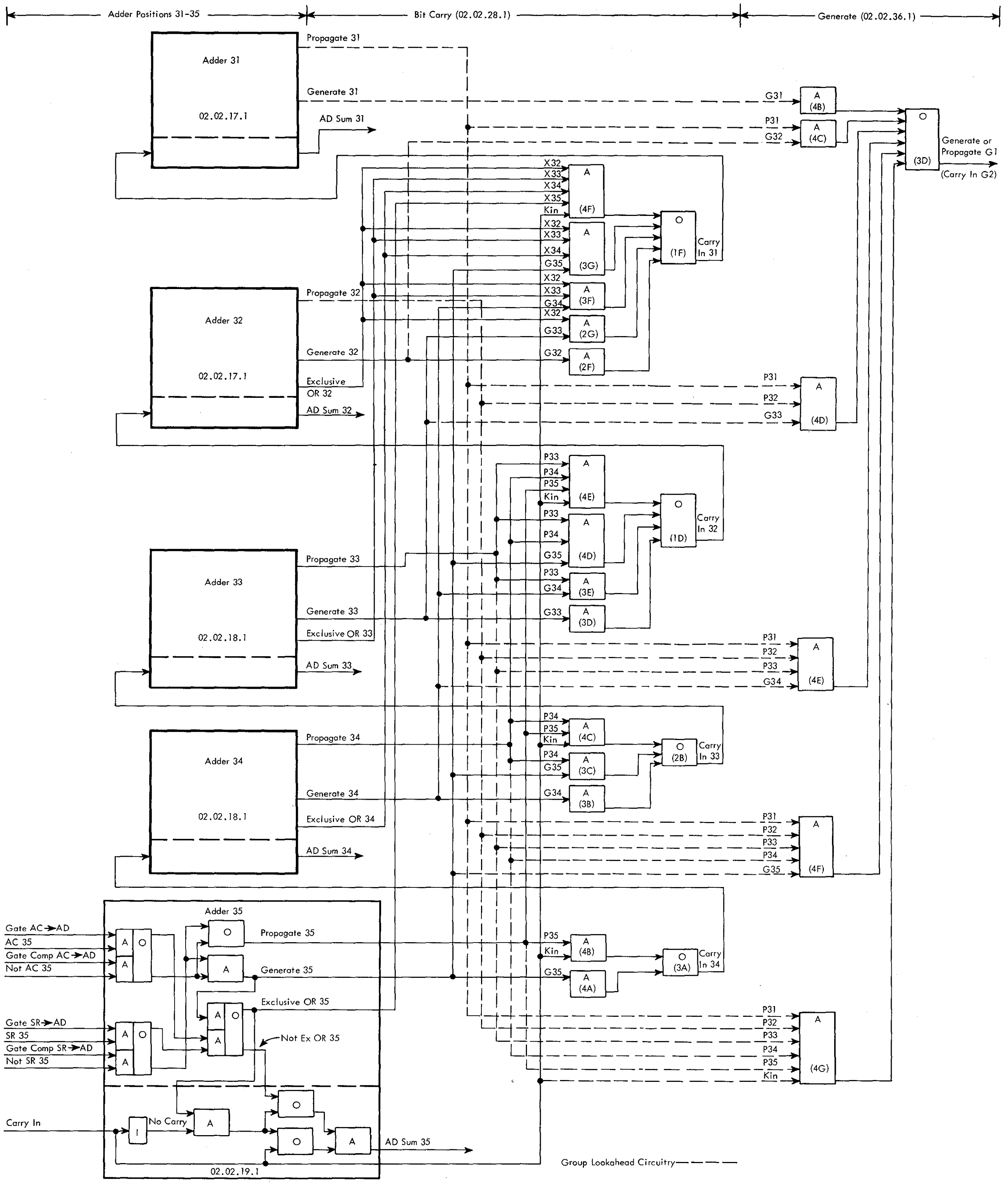


Figure 52. Group 1 Adder Carry, Generate and Propagate

Lookahead for both groups 1 and 2 is shown because of their difference in logic. The dotted lines in Figure 52 show group 1 lookahead. Six sets of conditions can produce an output from the group: first, AD(31) can generate a bit alone; second, AD(32) can generate a bit and AD(31) can propagate the bit; etc. Note that the logic becomes more involved from top to bottom; the last AND circuit indicates that an input carry (Kin) will produce a group carry if AD(31-35) have the ability to propagate.

Group 2, as well as the remaining groups 3-7, produce two outputs; "any generate G2" and "generate or propagate G2." Both conditions *must* be active to produce a carry into the next higher group. Design of the adder lookahead is the result of Boolean algebra, and may not, therefore, seem immediately logical and straightforward. This design, however, performs the required task of adder lookahead in a more efficient and economical method than other "more straightforward" configurations.

The adder group output, "any generate G2" results at O_1 (Figure 53), from any or all adder positions being able to generate a bit (both adder inputs are a 1).

The "generate or propagate G2" output from O_7 results from the following conditions: first, all adder positions are able to propagate at A_6; second, each position, except the low-order adder position, has the ability to generate and have this bit propagated through the remaining higher-order adder positions at A_5, A_4, A_3, and A_2. The ability of the low-order adder position to generate and have its bit propagated by the remaining adder positions is not defined specifically by the function "generate or propagate G2." This latter case is accounted for, however, because if AD(30) can generate, it can also propagate. P30 activates A_6 and G30 activates O_1; therefore, both of the group outputs are active and a carry is passed into group 3.

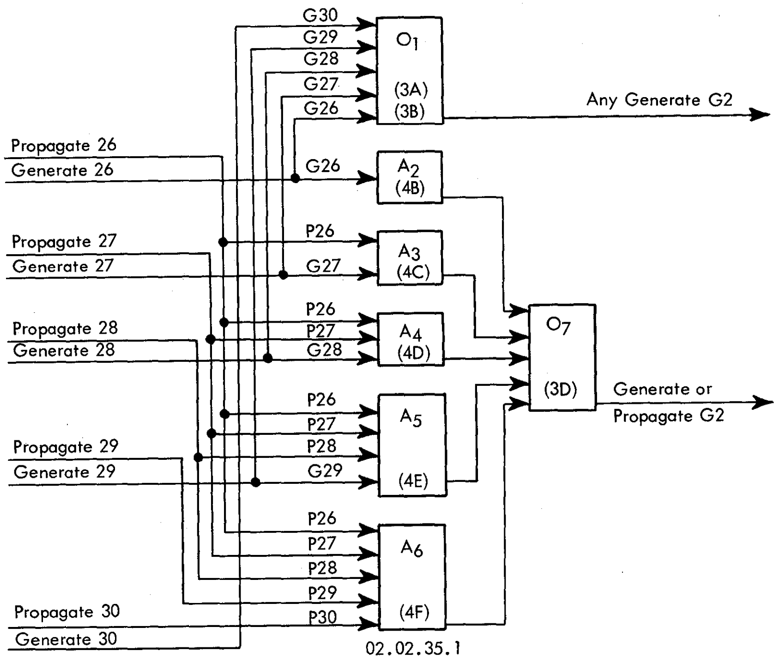


Figure 53. Group 2 Generate and Propagate Lookahead

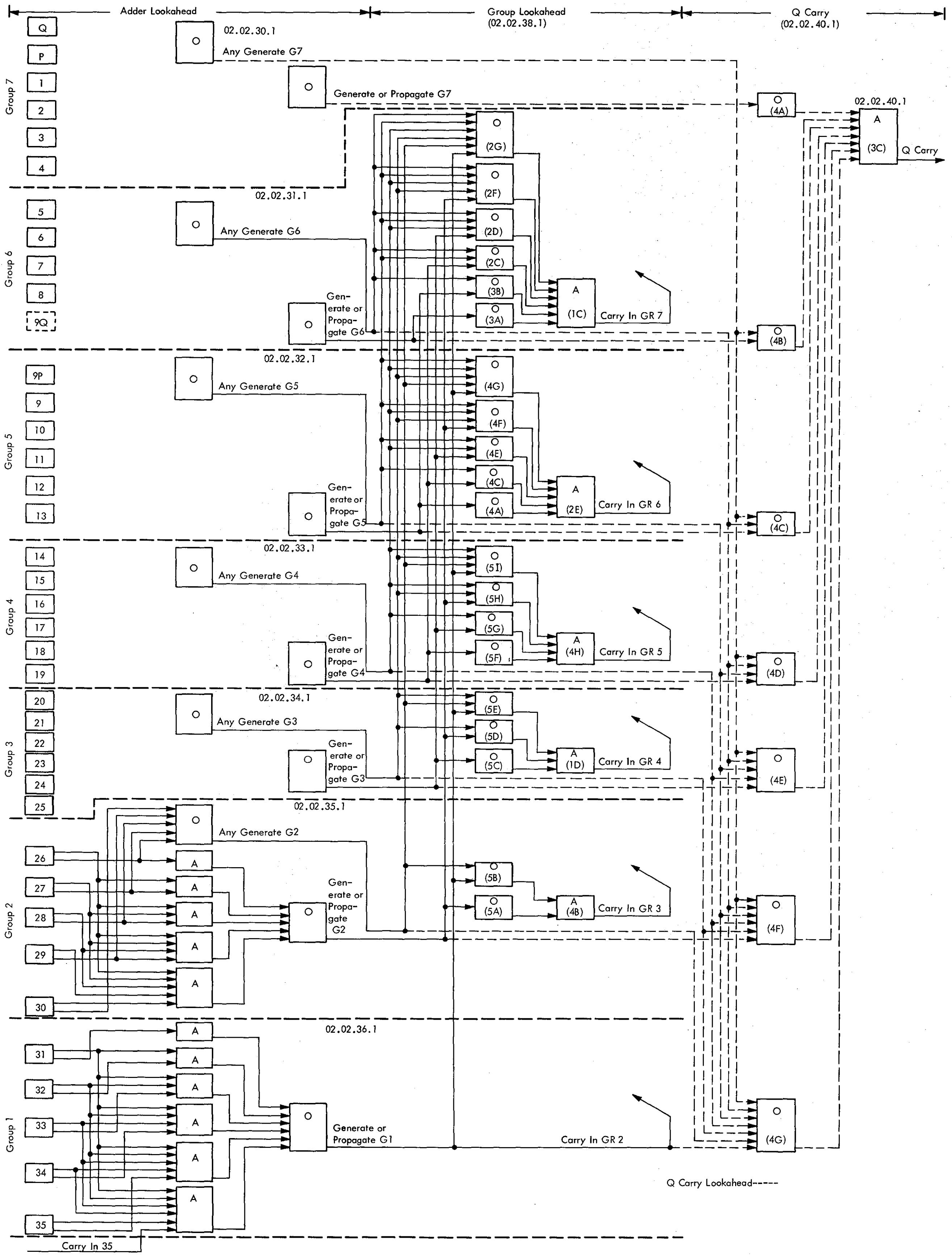


Figure 54. Main Adder Group Lookahead and Q Carry

Figure 54 shows all adder group carries. Group 1 having the ability to either generate a bit internally or to propagate a Kin produces a "carry in GR 2." Note that this same carry line also conditions the top OR circuit in each of the higher group lookaheads. Depending on the inputs to the remaining adder positions, this carry from group 1 may be propagated through the rest of the main adder.

A carry into group 3 can result from two conditions: group 2 can generate a carry internally, or group 2 can propagate a carry from group 1. Taking the first case, both "any generate G2" and "generate or propagate G2" are active and condition the top leg of O_{5B} and O_{5A}, respectively. In the second case, group 2 produces only the "generate or propagate G2" line as input to O_{5A}. However, the carry from group 1 conditions the bottom leg of O_{5B}; A_{4B} is satisfied and a carry is introduced into group 3.

Producing a carry into higher-order groups becomes progressively more involved but the basic concept just described remains the same. Note that each high-order group is conditioned by the outputs of each lower-order group. In this manner simultaneous decisions can be made at each group lookahead; no "ripple" is involved from one group to another.

Q Carry Lookahead

The carry from the main adder, Q carry, is used in testing and logical operations as well as during arithmetic instructions. The speed at which a Q carry can be obtained becomes important in many operations. Therefore, adder group outputs are also used in Q carry lookahead circuitry (shown as dotted lines in Figure 54) to obtain the earliest possible Q carry indication. This lookahead follows the same philosophy described in the preceding sections.

Index Adders—XAD (Systems 03.05.40.1-03.05.47.1)

The index adders (XAD) in the 7094 II perform a variety of functions such as:

1. Address modification on the instruction in either the storage register or IBR. In either case modification is from an index register as specified by the tag portion of the instruction concerned.
2. Tests on or modification of index registers as required by TIX, TNX, TXH, TXL, TXI (class A) instructions.
3. Incrementing the program counter, address register or address portion of the IBR.
4. Providing an address reference to MAR during overlap operations.
5. Producing the required data reference addresses during convert instructions.
6. Providing direct data flow paths to or from the program counter, address register, index register, storage register (positions 3-17 or 21-35), instruction backup register (positions 3-17 or 21-35) and shift counter.

Two data input gates are provided to the index adders; input A and input B (Figure 55). These inputs are divided such that one half of the input data always arrives at input A and the other half arrives at input B. During address modification, for example, the address arrives at input A and the modifying index register value (complement) arrives at input B; during incrementing, the PC, AR, or IBR arrive at input A and a "hot 1" is inserted at input B of XAD(17).

It is possible to insert "hot 1's" into either all positions of the index adder or just XAD(17); the first case causes decrementing of the input value (−1); the latter case causes incrementing of the input value (+1).

When the index adders are being used as a data path, outputs go directly to SR(3-17), SR(21-35) or the shift counter. During instruction overlap, the index adders supply a second memory address to the MAR even/odd switch.

Note that during address modification, if no tag (tag of 0) is specified, 1's are gated into all index adder positions along with a carry into XAD(17). These ones effectively add a zero to the incoming value; therefore, the output value is the same as that supplied by the storage register or IBR.

An index adder latch (XAD LTH) is used in conjunction with the index adders. The latch acts as a common delay (second half of a shift cell) in the circuit for modifying the index registers, program counter, address register and instruction backup register. These registers consist of triggers (not shift cells); the delay therefore, allows proper operation on successive clock pulses.

The XAD latch is "free running" and always copies the contents of the index adders at the end of each clock pulse. The receiving register receives the new information immediately thereafter which is slightly into the following clock period.

When gating from the index adder latch to the XR, PC, AR, or IBR, four additional triggers are required as a short-time memory device to remember which register is to receive the output from the XAD latch. This remembering is necessary because the gating line to the index adders (XAD to AR, for example) will have already dropped before the new output can be gated from the XAD latch to the address register. Figure 44 shows condensed logic of the gating sequence for data flow from the index adder latch to the address register.

Index Adder Position

Figure 56 shows condensed logic for index adder positions 3, 4, and 17. As can be seen, the index adder is almost identical to the main adders discussed previously. The main difference between the XAD and AD is the number of inputs and outputs used. Also, because of the type of circuit cards used, some logic blocks appear as micro blocks whereas others appear as macro blocks.

The basic function remains the same—that of determining whether or not there is an adder sum output.

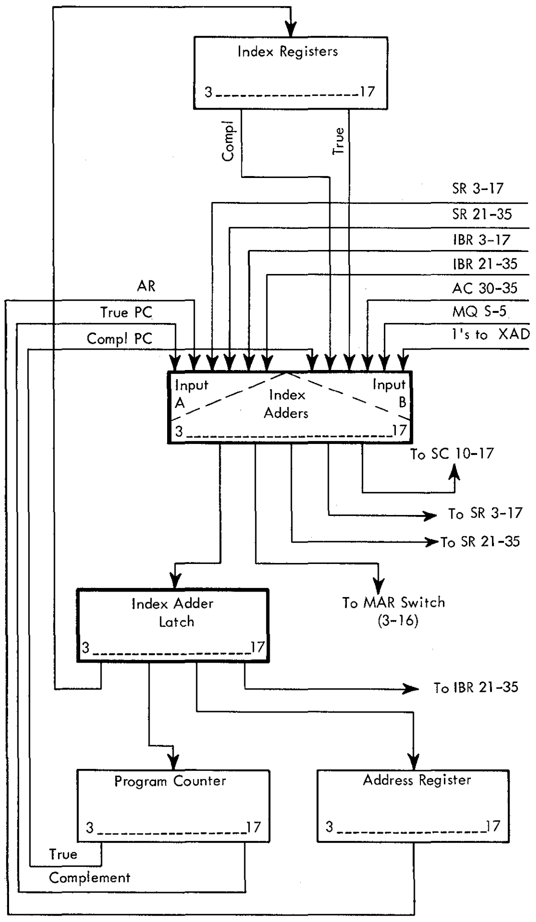


Figure 55. Index Adder Routing

The three basic inputs to the index adder are: input A, input B, and an input carry. As already discussed, the XAD inputs are divided so that the two values to be added arrive at inputs A and B, respectively. Logically, there never should be more than one input active at either input A or B at any given time.

Using XAD(17) in Figure 56 as an example, there are two A-input gates (AO_{11} and AO_{12}) tied together to form a 6-way AND-OR function. Two B-input gates also exist (AO_{13} and AO_{14}) but are not OR'ed together directly. The difference between the A- and B-inputs is due to circuit card usage and not because of logic.

Any data input being active ($A_1/A_2/B_1/B_2$) produces "propagate 17" at O_{19}. Either A_1/A_2 being active at O_{15}, and either B_1/B_2 being active at O_{16} produces "generate 17" at A_{20}. These propagate and generate lines are used in the adder lookahead circuitry to be discussed later.

The circuit combination of A_{17}, A_{18}, and O_{21} performs an exclusive OR test of the two data inputs. The in-phase (active) output of O_{21} indicates a "not exclusive OR" state (not $A \forall B$); the A and B inputs are either both 0's or both 1's. The out-of-phase (inactive) output of O_{21} indicate an "exclusive OR" state ($A \forall B$); either A or B is a 1, but not both. In the main adders, loading conditions required that this exclusive OR function be used in adder lookahead. In the index adders, however, loading is not as great and the exclusive OR function is not required outside of the immediate adder positions.

The carry into XAD(17) affects only the sum output. The input carry is, therefore, combined with the output conditions of O_{21} in a second exclusive OR network consisting of A_{22} and OA_{23}. Taking various examples of A, B, and carry inputs should quickly prove the circuit validity.

Index Adder Bit Carry Lookahead

As was explained with the main adders, each index adder position must be able to provide necessary carries into adjacent positions. For determining possible carries, two lines are produced from each adder position: propagate (P) and generate (G).

The output of O_{19}, "propagate 17" (Figure 56), is active if any A/B input is a 1. The output of A_{20}, "generate 17," is active if both A and B inputs are 1's. With the propagate and generate lines, individual adder bit carry lookahead is developed as shown by the solid lines in Figure 57. This lookahead circuitry, again, becomes progressively more involved as it progresses from the lowest-order to the highest-order adder position within the adder group.

No adder bit carry lookahead circuitry is provided for XAD(12). XAD(12), which is the low-order position of the next group, receives its carry indication from the group lookahead circuitry to be discussed next.

Index Adder Group Carry Lookahead

The 15 index adder positions are divided into three groups of 5 positions each as follows:

GROUP NUMBER	XAD POSITIONS
1	17, 16, 15, 14, 13
2	12, 11, 10, 9, 8
3	7, 6, 5, 4, 3

The object of dividing the index adders into three groups is to speed up adder operation and simplify lookahead circuitry. Group lookahead circuitry, as shown by solid lines in Figure 58, determines whether or not carries should be introduced into the low-order positions of groups 2 and 3. The lookahead circuitry for group 1 is also shown by dashed lines in Figure 57. Lookahead circuitry for groups 2 and 3 are different than group 1 but identical to that of the main adder as shown in Figure 53. Because of the similarity, see "Main Adder Group Carry Lookahead" section for a more detailed explanation.

Index Adder 3 Carry

A carry from the index adder, "XAD 3 carry," is used primarily as a test for successful transfers when executing class A instructions (TIX, TNX, etc.) from either the program register or IBR.

Direct outputs from all three index adder groups are used in detecting an XAD(3) carry. The circuitry is shown as dashed lines in Figure 58 and is simply an extension of the lookahead circuitry used for groups 1 and 2. No XAD(3) carry trigger is necessary; the circuit output is active long enough to allow all tests to be made.

Again, carry lookahead is dependent solely on inputs to the individual adder positions; no individual adder or group carries are involved.

Index Adder Compatibility

When executing 704 or 709 programs under compatibility mode, XAD(3) and XAD(4) must be effectively bypassed. This bypassing can be accomplished by forcing the XAD position into a propagate condition (Figure 56).

When the memory nullify trigger is on, "memory null" is applied through O_6 to activate "propagate 3." Any resulting carry up through XAD(4) will, therefore, logically be "passed through" XAD(3) and give an XAD(3) carry. Because XAD(3) does not theoretically exist at this point, any possible sum output must be blocked. This blocking is accomplished at the output of OA_{24} by dot AND'ing with a "not memory null" condition.

If the 16K/24K switch on the CE panel is in the 24K position, 24K of memory is available to the I/O compatibility program and 8K available to the executing program. Under these conditions, XAD(4) must also be bypassed and the XAD(4) sum blocked. The methods used are the same as those described above for XAD(3).

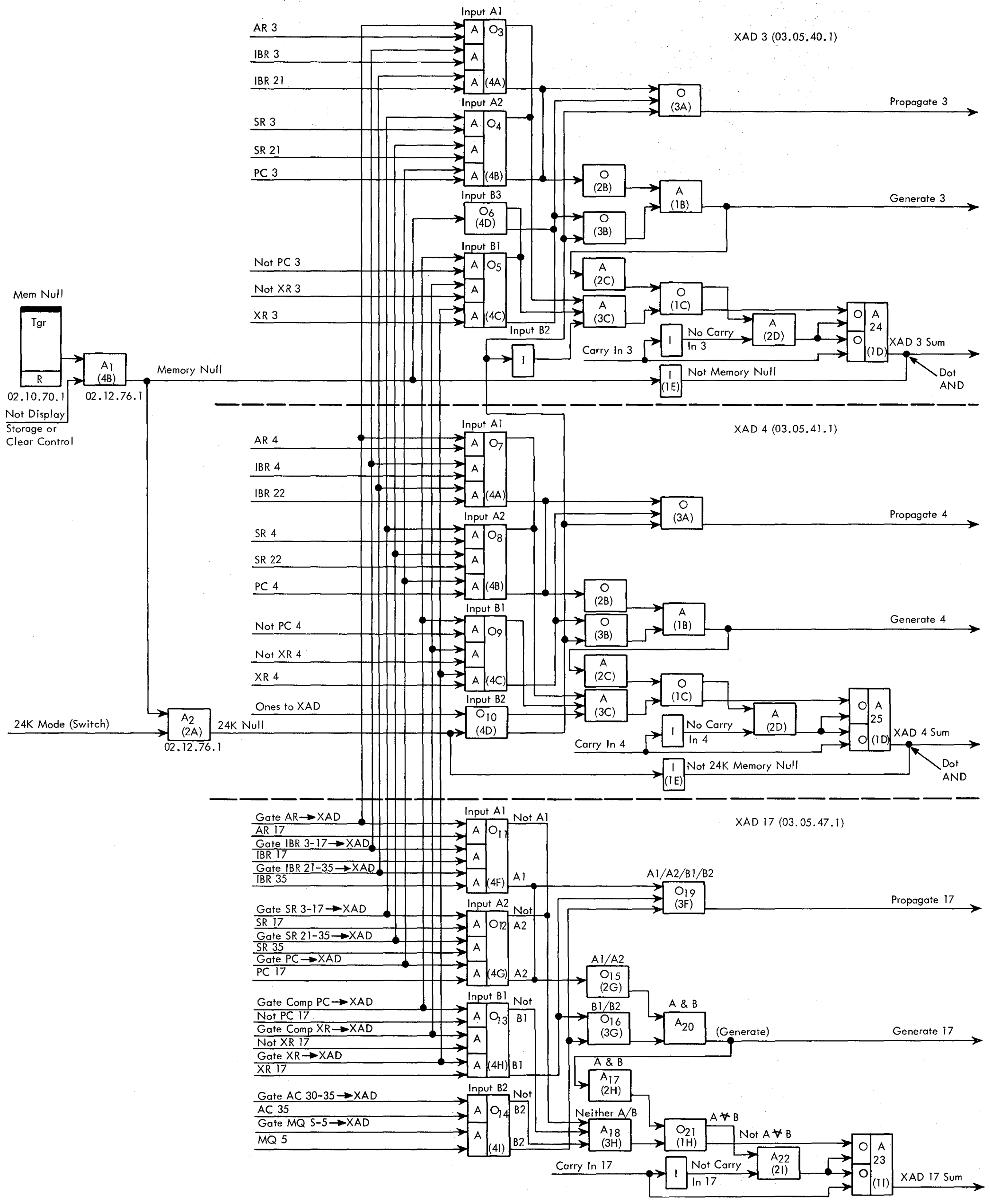


Figure 56. Index Adder Positions

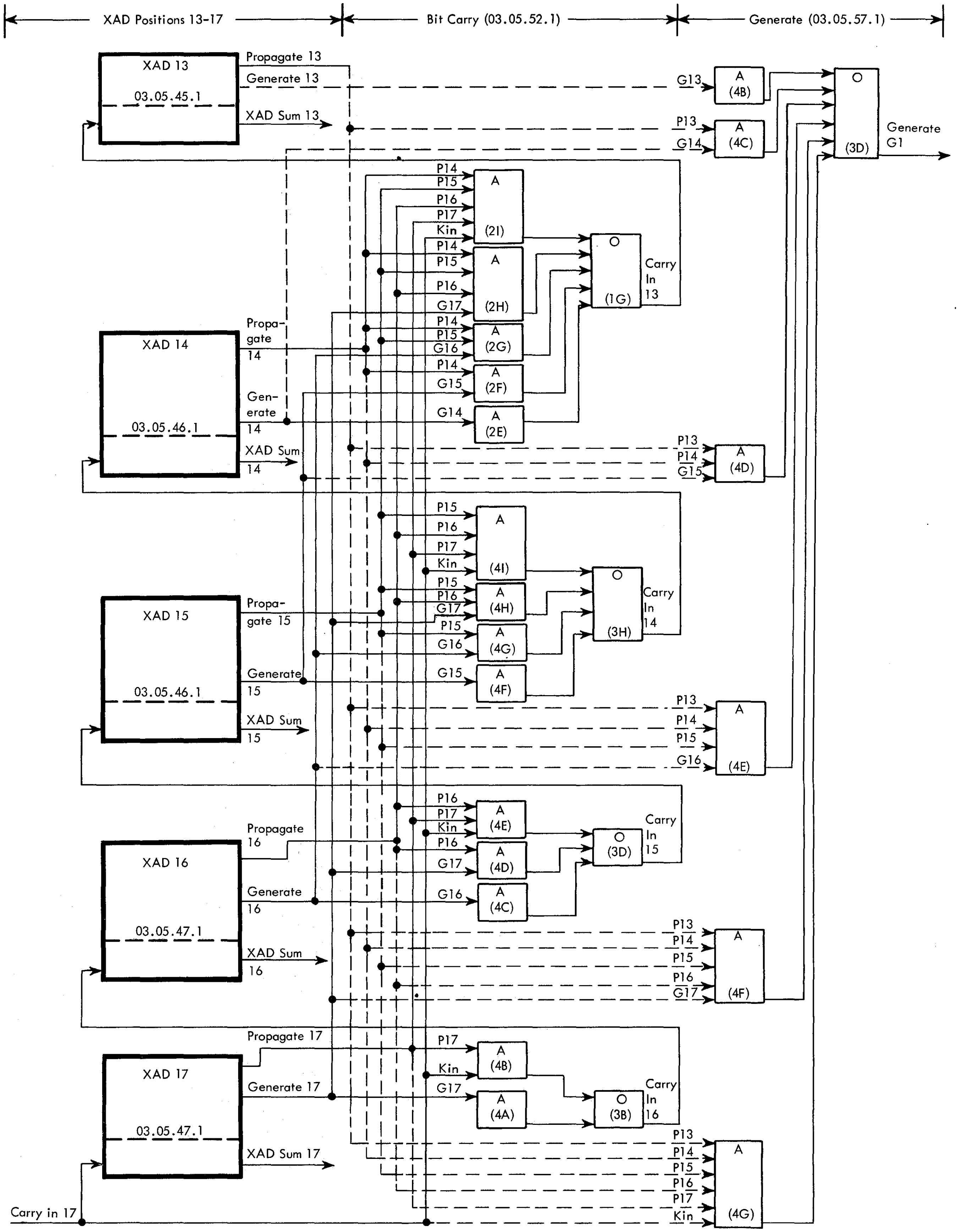


Figure 57. Group 1 Index Adder Bit Carry and Lookahead

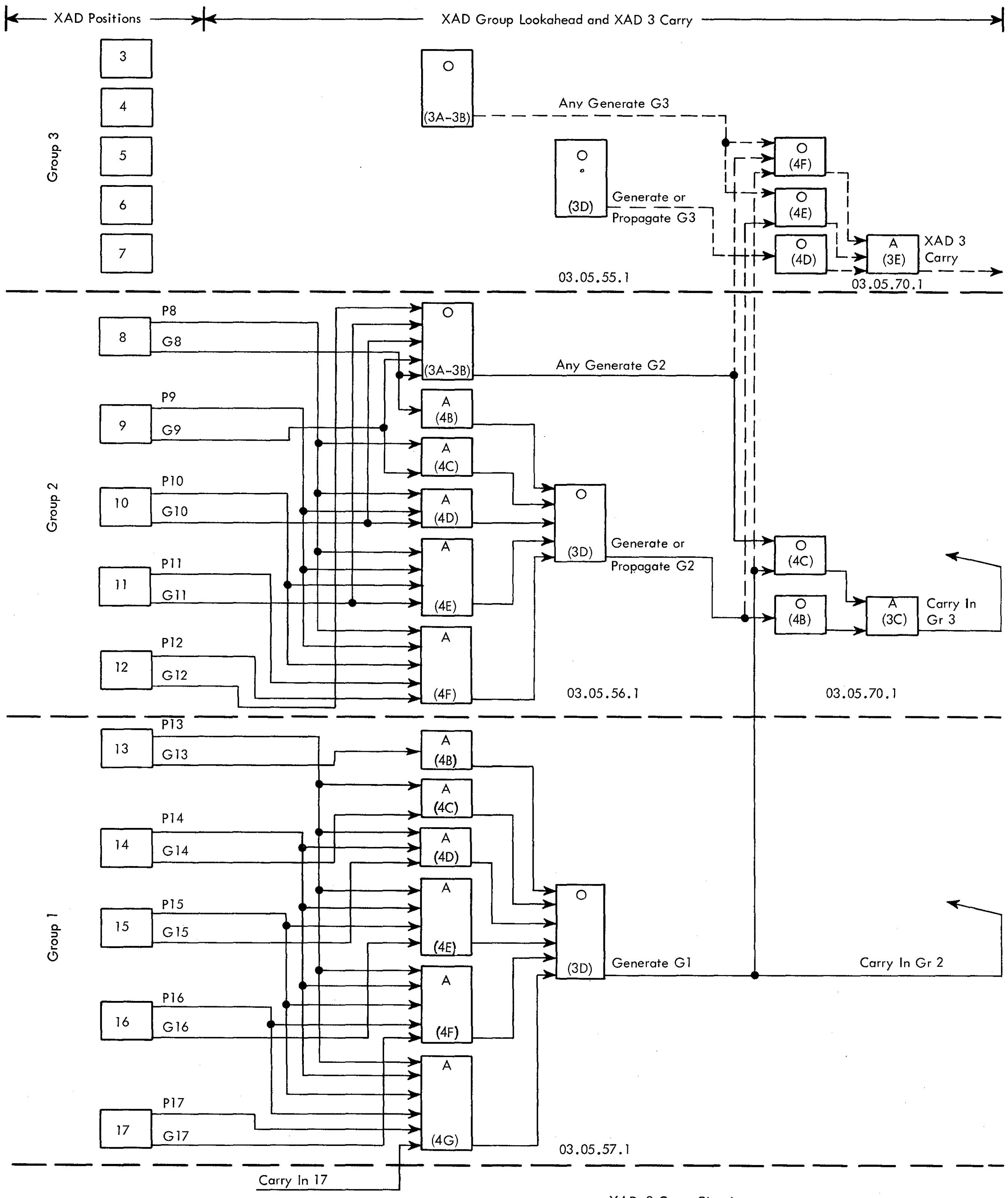


Figure 58. Index Adder Group Lookahead and XAD(3) Carry

SR Zero Check (Systems 02.12.47.1)

Outputs from storage register input gating (1-35) are fed to a zero check circuit as shown in Figure 59. Note that the test is made from data introduced at the storage register input gating and *not* from data presently in the storage register. In most cases, the data to be tested will not actually be set into the storage register.

Many instructions gate the contents of the various CPU registers and adders to this zero check circuit as a check for either zero or equal conditions. Arithmetic operations make extensive use of this check circuit for detecting initial zero quantities (i.e., multiplier or multiplicand) or zero final results. When tests are made on the fraction of floating-point quantities, only positions 9-35 are gated from the register concerned to the check circuit.

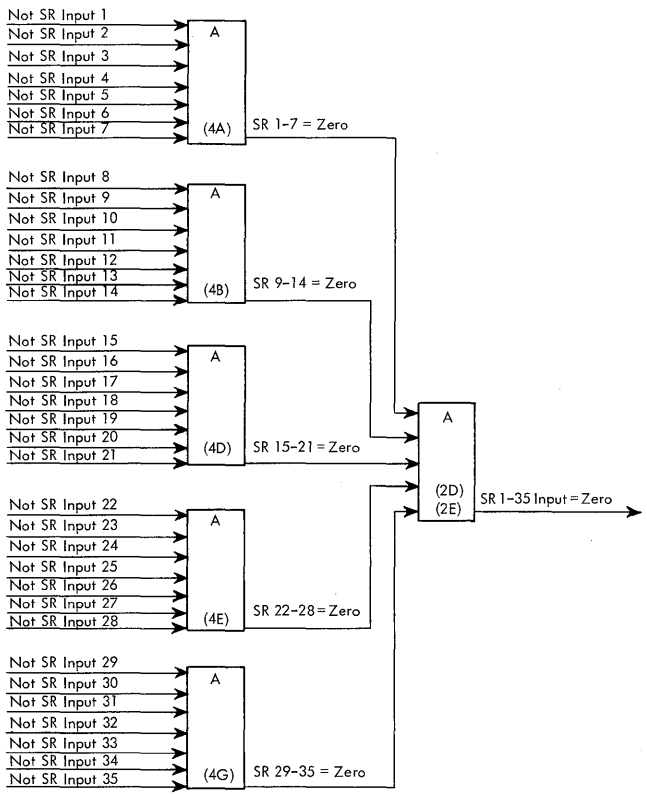


Figure 59. SR Input Zero Check

Memory Selection Circuits (Systems 03.06.29.4)

The 7302-3 core storage is divided into two logically independent units of 16,384 positions each. To initiate these two units, separate "memory select" pulses are required. When computer circuitry requires a reference to the even memory, a "memory select even" pulse is generated; when computer circuitry requires a reference to the odd memory, a "memory select odd" pulse is generated—if both references are required, both select pulses are generated.

In Figure 60, O_1 and O_2 receive the circuit indications as to which memory/memories are required. These gating pulses occur at the end of the machine cycle, 6(D2) time, and also cause gating of MAR(3-16) addressing lines to the 7302-3. Just how these even or odd gating lines are developed will be discussed in the following section.

The memory select pulses can occur at two different times: CPU operations at 7(D1) time; and channel operations at A2(D1) time. Data channel operations select memory at a later time because of the longer 12-point cycle. The memory, operating at a faster rate, provides the necessary data at the proper channel time. The reverse also applies when storing data from the channel.

Note from AO_3 that channel operations assume priority—CPU memory selection is blocked until all channels are completed with either B or E time.

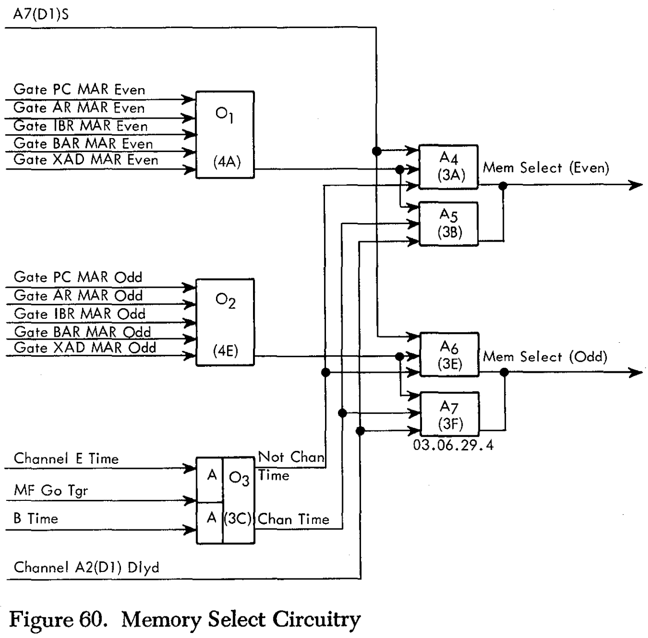


Figure 60. Memory Select Circuitry

MAR Switching and Address Controls

Only 14 of the 15 address bits are required to select any one of the 16,384 even or odd memory positions. The remaining 15th address bit determines selection of the appropriate memory.

Memory selection and MAR gating is effected by whether the computer is operating in the normal mode or diagnostic mode.

In normal mode, the even memory contains all of the even addresses (0, 2, 4, 6, 10 through $77{,}776_8$); the odd memory contains all of the odd addresses (1, 3, 5, 7, 11 through $77{,}777_8$). With this normal case, memory selection is under control of address bit (17)—if bit (17) = 0, select even memory; if bit (17) = 1, select odd memory. This selection circuitry for program counter gating is shown in Figure 61 at A_2 and A_4, respectively.

In diagnostic mode, the "even" memory contains the lower half of all memory addresses (0 through $37{,}777_8$); the "odd" memory contains the upper half of all memory addresses ($40{,}000_8$ through $77{,}777_8$). With this diagnostic case, memory selection is under control of address bit (3)—if bit (3) = 0, select even memory; if bit (3) = 1, select odd memory. This selection circuitry is shown at A_3 and A_5.

MAR selection and gating is under control of five sources: the program counter as just described, the address register, IBR, buffer address register for channel operations, and index adders. Figure 62 shows the overall logic for all five controlling sources; the program counter controls as shown in Figure 61 are repeated at the top of Figure 62. Note the similarity of the controls and also that they control only MAR(4-16).

Controls for MAR(3) are handled separately because of the MAR(3 and 17) "switching" for normal and diagnostic modes of operation. Circuit tests are made to determine whether or not a bit should be sent on the MAR(3) bus. Four combinations of address bits (3 and 17) can exist:

DIAGNOSTIC MODE	3	17	NORMAL MODE
1. Select even memory	0	0	Select even memory
2. Select odd memory	1	0	Select even memory
3. Select even memory	0	1	Select odd memory
4. Select odd memory	1	1	Select odd memory

Selection of MAR(3)—even memory requires item 2 for normal mode and item 3 for diagnostic mode. These two cases are shown for the program counter at A_6 and A_7 in Figures 61 and 62.

Selection of MAR(3)—odd memory requires only item 4. Note that this item is independent of either normal or diagnostic mode, and is shown at A_8 in Figures 61 and 62.

When operating in diagnostic mode, two sequential instructions cannot be fetched from the same memory on the same cycle. Because of the memory arrangement, "gate XAD to MAR" circuitry has no logic, and is, therefore, blocked at A_{4B}, Figure 62.

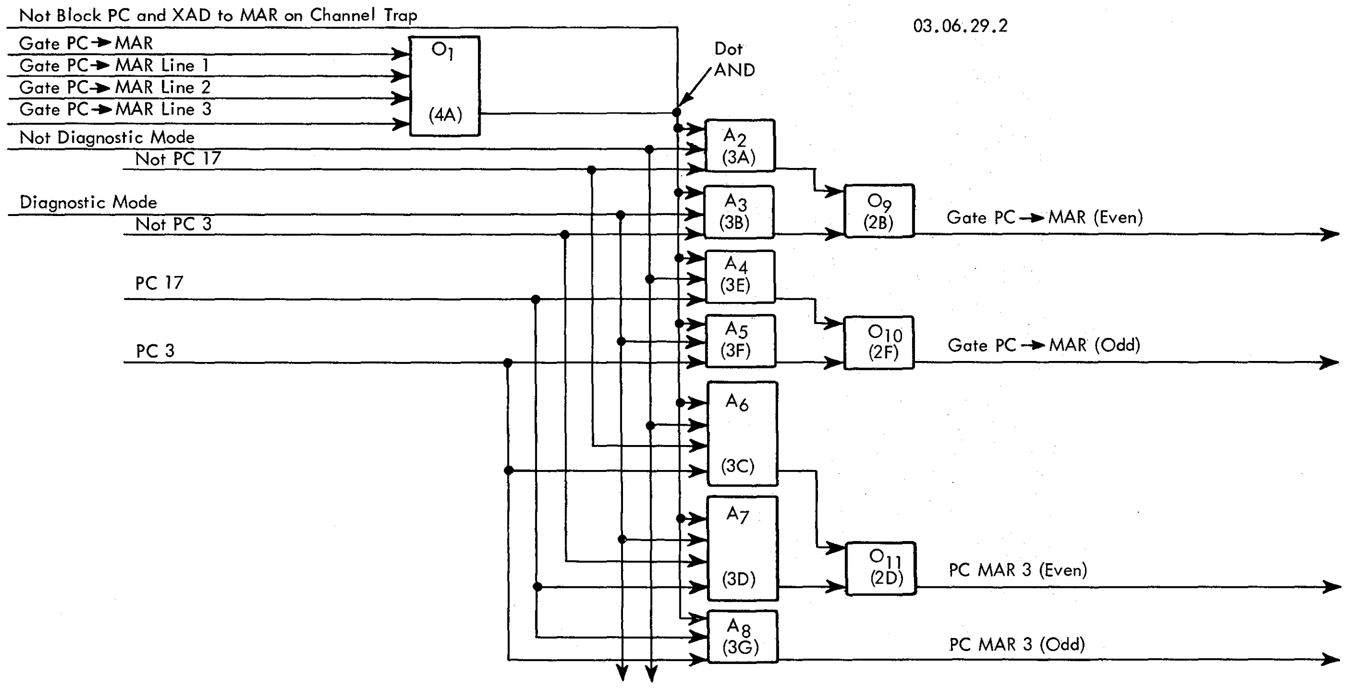


Figure 61. Program Counter MAR Bus Selection

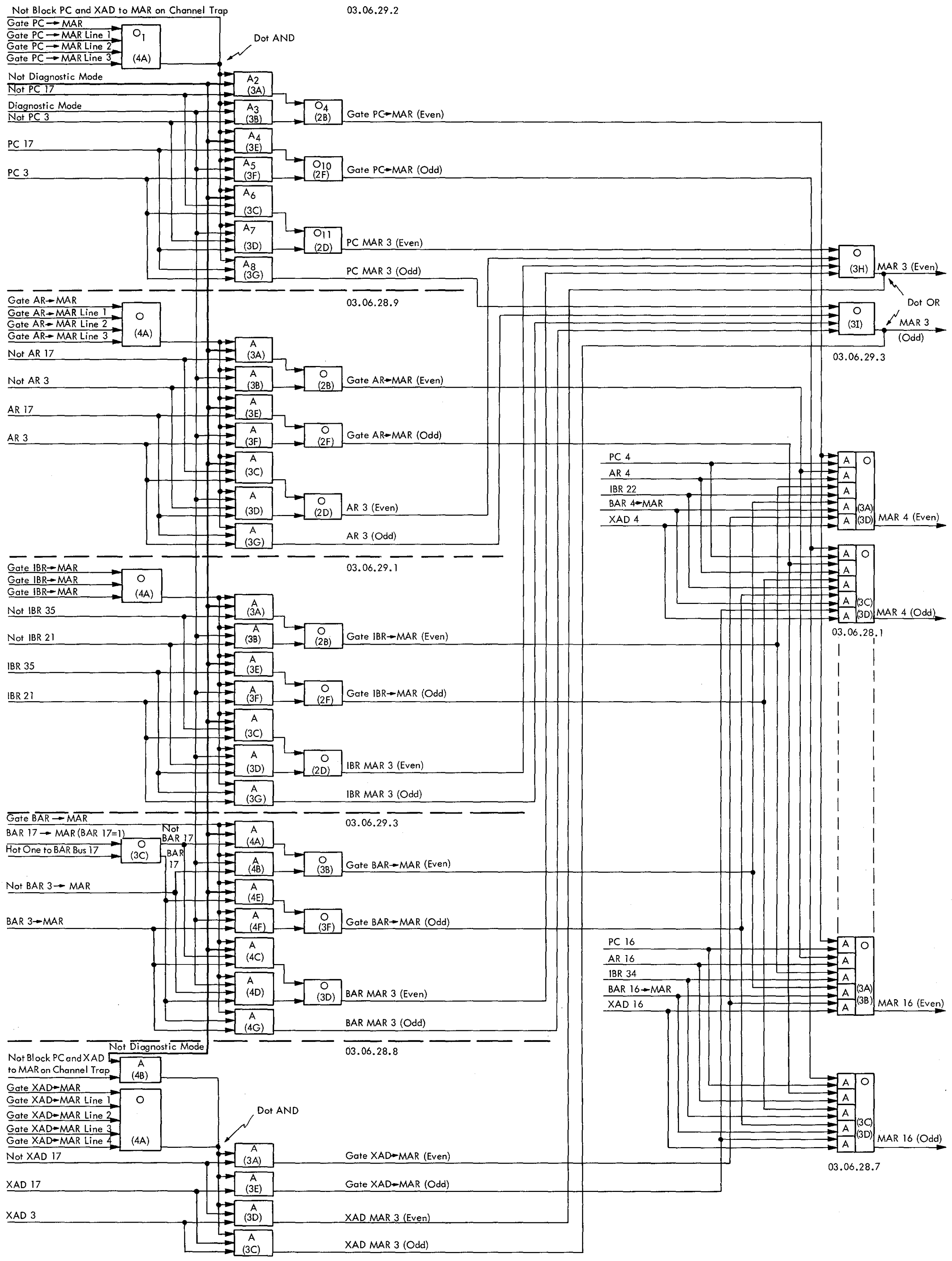


Figure 62. MAR Bus Selection and Switching

Timing

Master Clocks and Pulses

All of the computer functions are directly related to two master clocks shown in Figure 63. These clocks located in the multiplexor, provide the basic pulses necessary for CPU and channel operation.

The heart of these clocks is a 5.71 megacycle oscillator which produces a complete output cycle once every .175 microseconds (175 nanoseconds). Each positive and negative oscillator output pulse is, then, approximately 87 nanoseconds long (Figure 64).

The CPU clock is an 8 cycle-point clock composed of eight triggers (CPU clock triggers 0-7); the channel clock is a 12 cycle-point clock composed of 12 triggers (channel clock triggers 0-11). In either case, the clock forms a closed ring—each trigger is turned on in sequence and provides a gated output of 175 nanoseconds. The combined outputs produce the basic cycle times as follows:

CPU Cycle: 8 x .175 = 1.40 microseconds
Channel Cycle: 12 x .175 = 2.10 microseconds

Clock drive pulses needed to sequentially step the clock triggers are produced by the clock drive trigger (Figure 63). Oscillator outputs gate two controlling AND circuits to the clock drive trigger; the top AND circuit is used to set, and the lower AND circuit is used to reset the trigger. Delayed outputs from the clock drive trigger condition the input AND circuits such that the trigger changes state with each positive oscillator pulse. Outputs from the clock drive trigger ("even clock drive" and "odd clock drive"), Figure 65, are a series of pulses 175 nanoseconds in duration. The clock drive trigger has acted to halve the output rate of the oscillator.

Both the CPU and channel master clocks are reset under power-on conditions or by depression of the console clear key (Figure 63). At the end of the reset pulse, the DLY-AND circuit combination produces a 100 nanosecond "start clock" pulse which turns on both the CPU clock 0 and channel clock 0 triggers.

Using the CPU clock as an example in Figure 63, the next "even clock drive" gates the CPU clock 0 trigger output to produce an A0(D1) pulse of 175 nanoseconds duration. This same A0(D1) pulse, besides being gated to other circuitry, also turns on the CPU clock 1 trigger. Note that even though the CPU clock 1 trigger is turned on at 0-time, the output of its AND circuit does not become active until "odd clock drive," 175 nanoseconds later. The main logical output of the trigger is, therefore, A1(D1).

As a result of the preceding logic, the A1(D1) pulse turns on CPU clock 2 trigger. This trigger output when gated with "even clock drive" produces an A2(D1) pulse. Referring to Figure 63, CPU clock 2 trigger turning on at 1-time acts as a reset to CPU clock 0 trigger, and the A2(D1) gated output acts as a set pulse for CPU clock 3 trigger. This sequence continues through CPU clock 7 trigger. Rise of the A7(D1) pulse turns the CPU clock 0 trigger back on and the clock continues to run in a closed ring.

I/O operations on the 7094 II are bascially the same as for the 7090/7094. Timing conditions, however, are such that the 1.4 microseconds cycle is too fast to support channel operations without considerable rework within the channel itself. Therefore, to accommodate the channels and simplify the 7094 to 7094 II conversions, the 12 cycle-point clock has been retained strictly for use by the channels. Looking at the channel clock in Figure 63 shows that operation is identical to the CPU clock except that the cycle duration is extended beyond clock 7 time to include clock 11 time.

Note that even though the CPU and channel clocks are separate circuits, *both* are reset, started, and under control of common clock logic. From Figure 66 it can be seen that two channel cycles occur for every three CPU cycles. Therefore, at every third CPU cycle both clocks align themselves with respect to 0-time. More information concerning channel operation and timings will be found later in this section and in volume 3.

Figure 67 shows outputs and controls significant to each stage of the CPU and channel clock rings. Figures 68 and 69 shows sequence charts for the CPU and channel clocks respectively. Note that each clock trigger is on for 350 nanoseconds, twice the time duration of an individual clock pulse. A particular clock trigger is turned on one clock pulse early and only the last half of the output is gated by the clock drive pulse. This slower switching of the clock triggers provides increased reliability in the clock operation as well as additional pulses usable in the computer.

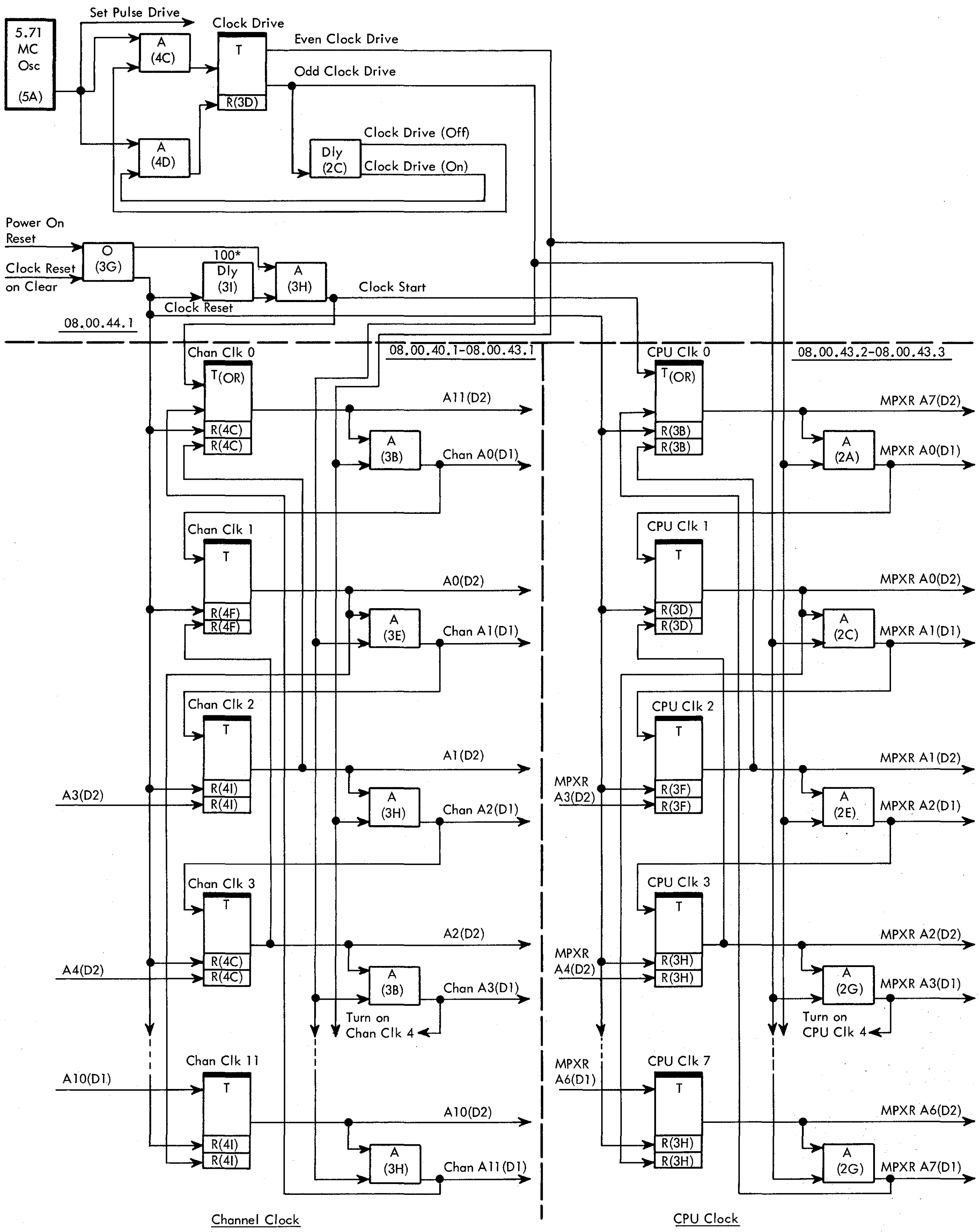


Figure 63. Master Clock Logic

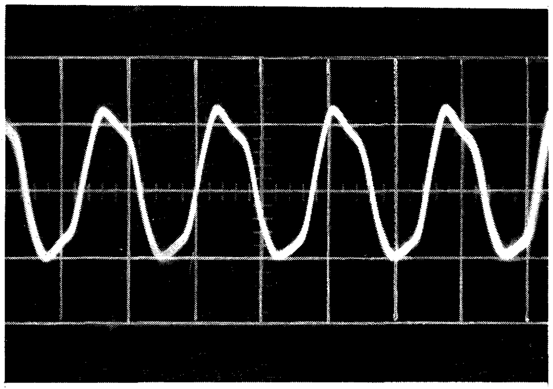


Figure 64. 5.71 MC Oscillator Output

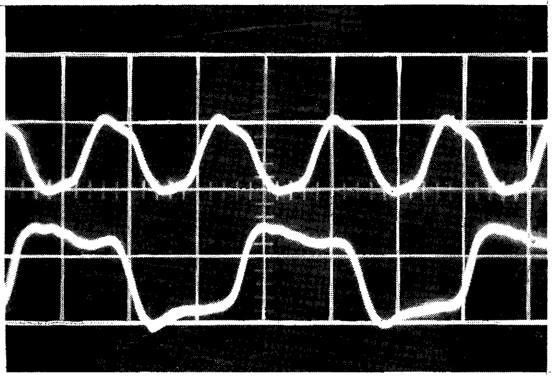


Figure 65. Oscillator Output and Even Clock Drive

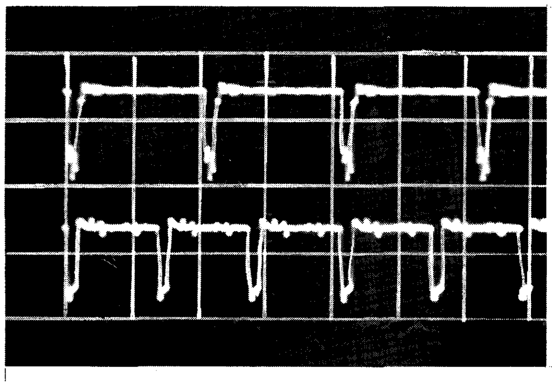


Figure 66. CPU and Channel Cycle Relationship

CPU Clock Tgr.	Turned On by	Turned Off by	Tgr Duration and Output	Gated Output
0	A7 (D1)	Clock 2 Tgr	A7 (D2)	A0 (D1)
1	A0 (D1)	Clock 3 Tgr	A0 (D2)	A1 (D1)
2	A1 (D1)	Clock 4 Tgr	A1 (D2)	A2 (D1)
3	A2 (D1)	Clock 5 Tgr	A2 (D2)	A3 (D1)
4	A3 (D1)	Clock 6 Tgr	A3 (D2)	A4 (D1)
5	A4 (D1)	Clock 7 Tgr	A4 (D2)	A5 (D1)
6	A5 (D1)	Clock 0 Tgr	A5 (D2)	A6 (D1)
7	A6 (D1)	Clock 1 Tgr	A6 (D2)	A7 (D1)

Channel Clock Tgr.	Turned On by	Turned Off by	Tgr Duration and Output	Gated Output
0	A11(D1)	Clock 2 Tgr	A11 (D2)	A0 (D1)
1	A0 (D1)	Clock 3 Tgr	A0 (D2)	A1 (D1)
2	A1 (D1)	Clock 4 Tgr	A1 (D2)	A2 (D1)
3	A2 (D1)	Clock 5 Tgr	A2 (D2)	A3 (D1)
4	A3 (D1)	Clock 6 Tgr	A3 (D2)	A4 (D1)
5	A4 (D1)	Clock 7 Tgr	A4 (D2)	A5 (D1)
6	A5 (D1)	Clock 8 Tgr	A5 (D2)	A6 (D1)
7	A6 (D1)	Clock 9 Tgr	A6 (D2)	A7 (D1)
8	A7 (D1)	Clock 10 Tgr	A7 (D2)	A8 (D1)
9	A8 (D1)	Clock 11 Tgr	A8 (D2)	A9 (D1)
10	A9 (D1)	Clock 0 Tgr	A9 (D2)	A10 (D1)
11	A10 (D1)	Clock 1 Tgr	A10 (D2)	A11 (D1)

Figure 67. CPU and Channel Clock Output and Controls

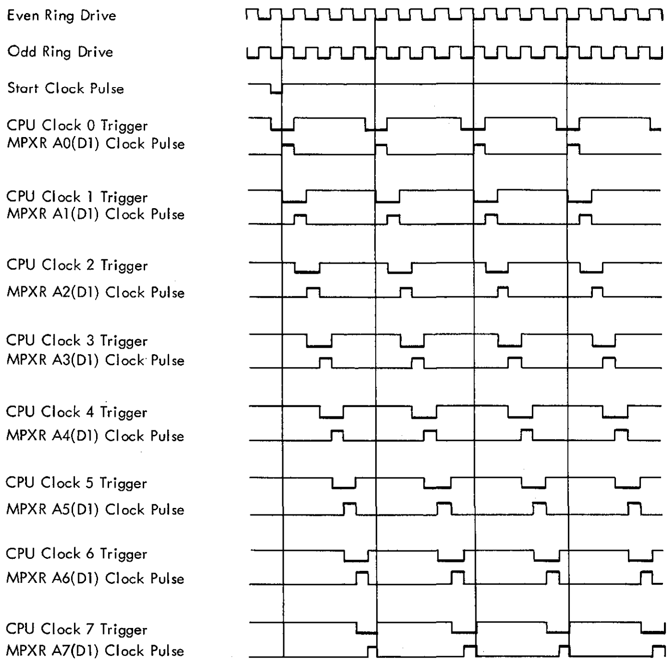


Figure 68. CPU Clock Sequence Chart

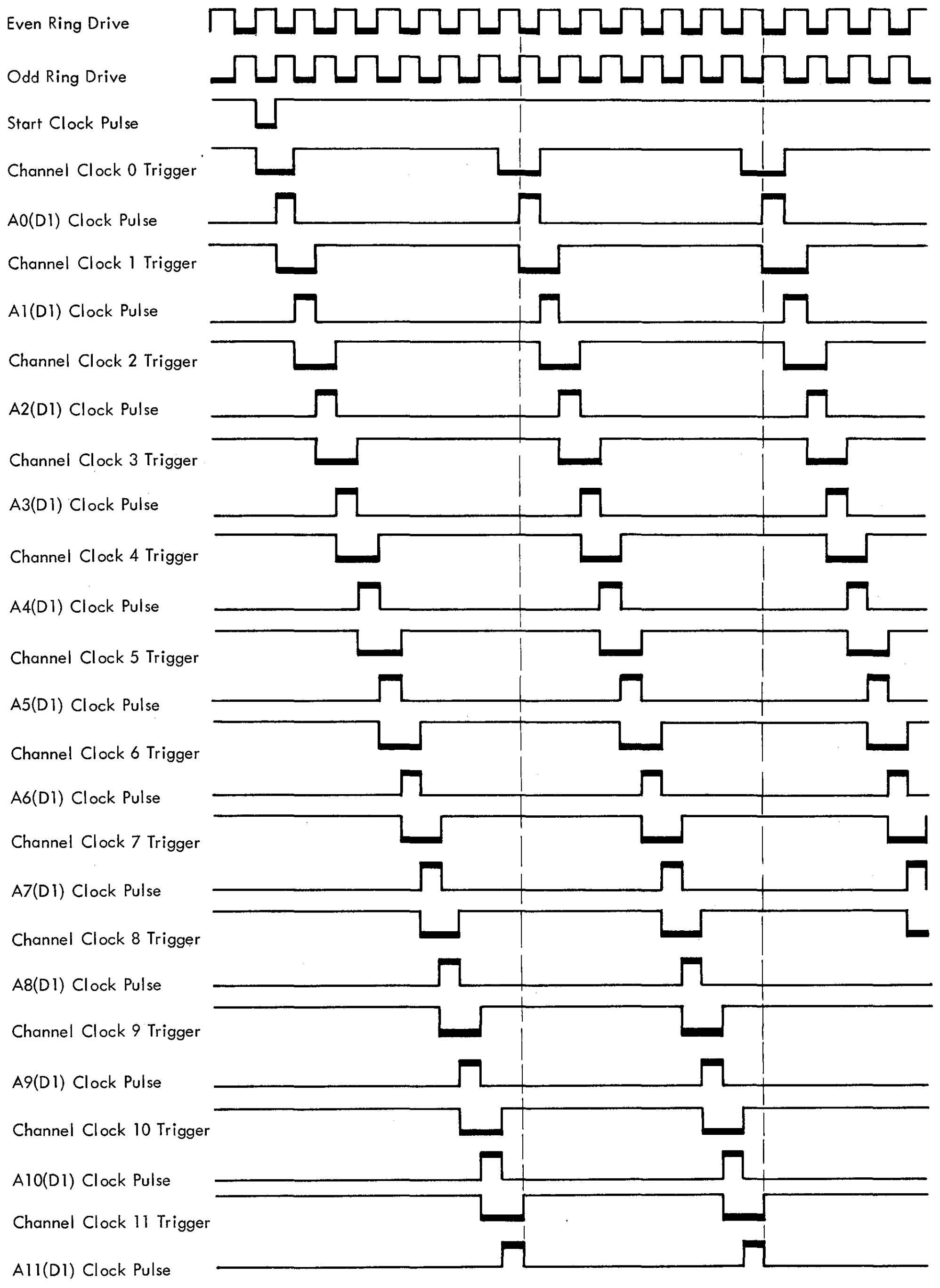


Figure 69. Channel Clock Sequence Chart

CPU Clock Pulse Distribution

The CPU clock is located in the multiplexor and the output pulses are distributed to both CPU-1 and CPU-2 for usage. Because of inherent delays in logic blocks and cable transmission lines, clock pulses arrive at the various CPU gates after finite delays.

Figure 70 shows the typical distribution pattern of a CPU pulse. This MPXR A3(D1) pulse is generated by the CPU clock 3 trigger (Figure 63) and sent to both CPU-1 and CPU-2 for usage. By the time the MPXR A3(D1) pulse arrives at CPU-2, however, the MPXR A4(D1) pulse is being formed. Therefore, to provide alignment of pulses between the multiplexor and CPU-2, the original MPXR A3(D1) pulse is relabeled as MF A4(D1) upon entering CPU-2.

As the original MPXR A3(D1) pulse continues on its way to CPU-1, additional delays are encountered similar to the ones just described. With the help of a DLY block, the incoming pulse to CPU-1 is delayed a small amount and again relabeled as a MF A A5(D1).

There is, therefore, a skew of one clock pulse between the CPU-2 and multiplexor, and a skew of two clock pulses between gates A and B of CPU-1 and the multiplexor. A5(D1) pulses throughout the computer, for example, are the result of three different CPU clock trigger outputs in the multiplexor. Except for trouble-shooting purposes, these various outputs need not concern the reader. The computer circuitry doesn't care where the pulses come from as long as they represent the correct timing or logic.

Although the A0(D1) through A7(D1) pulses are the prime outputs of the CPU clock, the D2 pulses are also distributed and used throughout the computer. Pulses of other durations such as A3(D3), A0(D6), etc., are produced throughout the systems as needed by using either triggers or combinations of AND and OR circuits.

In the computer, clock pulses are gated during cyclic operation and then labeled I6(D1) or E5(D1), depending on the particular cycle of operation. Whenever an A pulse is encountered in studying the computer, for example A6(D1), it means this pulse is used directly from the clock and is independent of the computer's cycle of operation. This A pulse always occurs at 6 time and is always available.

CPU 1 A and B Gate Clock Pulse Designation

Timing pulse alignment becomes increasingly more important as the internal computer operations are compressed into smaller and smaller intervals of time. In the A and B gates of CPU-1 where a major portion of the computer operations are performed, the clock pulses have been systematically delayed, aligned and labeled accordingly.

In Figure 70 the clock pulse coming into the A gate of CPU-1 is directed into two similar groups of circuitry. The top group of pulses labeled LN B (Line B) are sent to the B gate for usage; the bottom group (LN A) remains in the A gate for usage.

Pulses used in both A and B gates of CPU-1 also have an additional letter and number designation—A5(D1) G3, for example. In Figure 70 "−F A5(D1) G5 LNA" is the earliest A5(D1) pulse used in the A gate. This pulse delayed through an inverter produces a "+F A5(D1) G4." Additional inverter delays produce "−F A5(D1) G3," "+F A5(D1) G2," and "±F A5(D1) R" pulses. Note that the higher the G number, the earlier the pulse. By picking the appropriate pulse, compensation can be made for the number of levels of logic delay preceding a trigger or register gating circuitry.

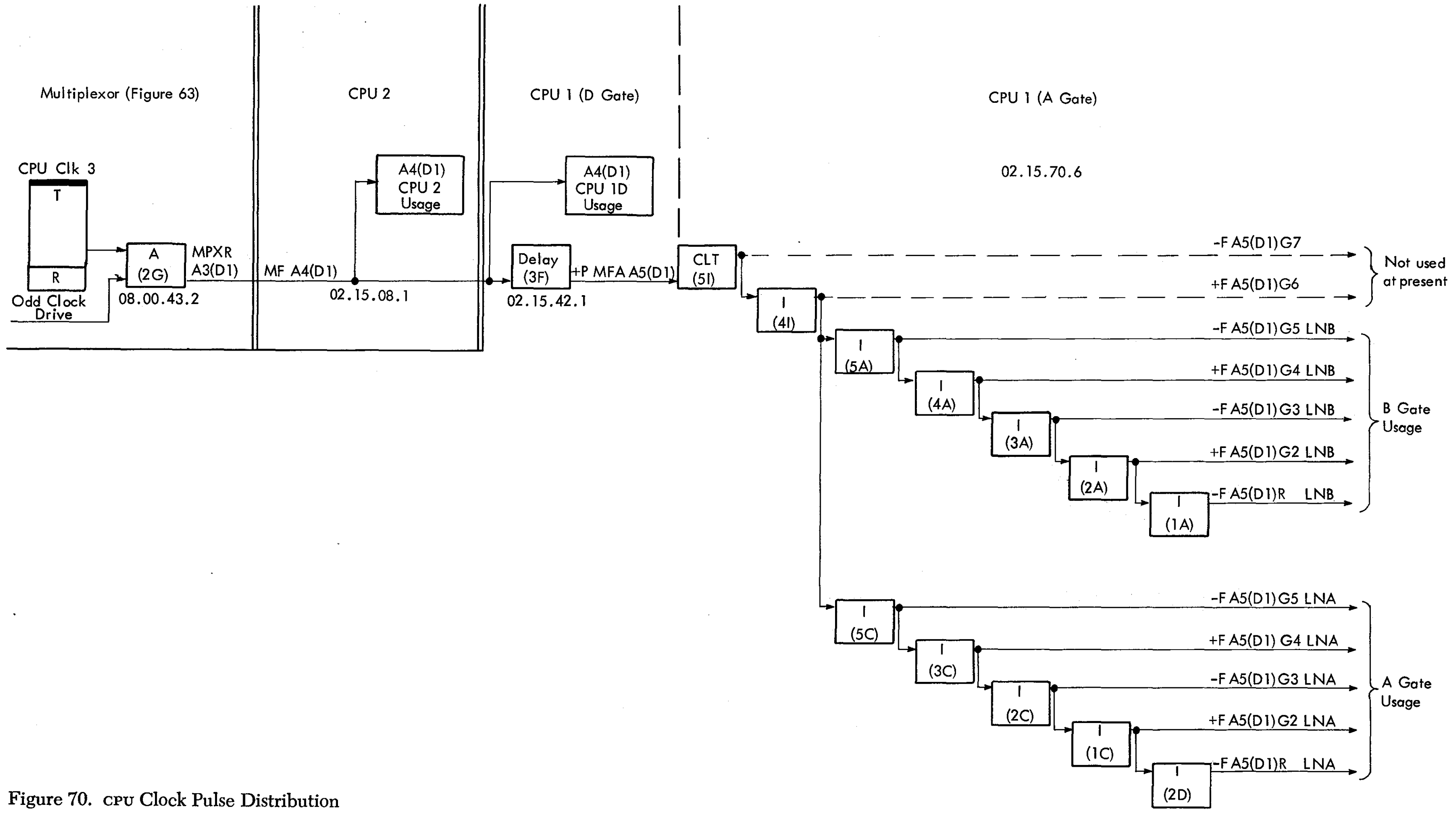


Figure 70. CPU Clock Pulse Distribution

Figure 71 shows four typical examples of pulse usage. At circuit (a), the A5(D1) G4 pulse is used because of the four levels of logic in front of the register gate. At circuits (b) and (c) the G3 and G2 pulses are used because of the three and two levels of logic in front of the register gating. The R (register) pulse, circuit (d), is the latest pulse and is usually used as a direct input to a trigger. There are, of course, exceptions to the pattern of usage just discussed.

Note that the G-pulse levels alternate between +F and −F at each step because of the natural inversion from each DIF logic block. The majority of the triggers and register inputs use + level logic (i.e. +G or +AOI).

CP Set Pulse Generation and Distribution

Computer set pulses (CP set pulses) are developed directly from the master oscillator. These pulses are precisely generated and timed to set triggers and registers, and control much of the gating and shifting of data within the computer. Width and timing of the CP set pulses, as related to the clock pulses, are extremely important to successful machine operation.

The input set pulse drive pulses are received directly from the oscillator. A variable delay is used to position the CP set pulse when aligning it with the clock pulses in the computer. Actual CP set pulse settings are made at the A and B gates in CPU-1.

Figure 72 shows condensed logic of how the CP set pulses are shaped, delayed, and distributed for specific usage in the computer. The inverters in Figure 72 have been included, not because they perform logic, but because they contribute to the timing delays.

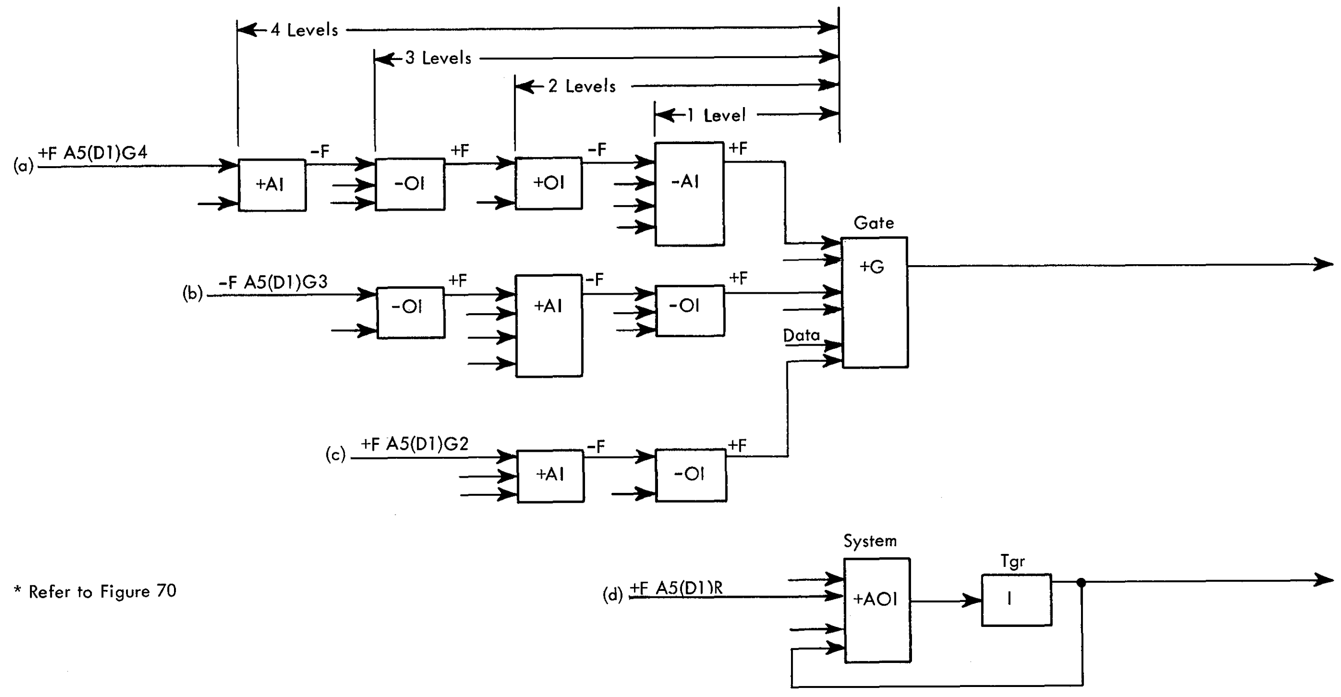


Figure 71. CPU Clock Pulse Designations and Usage

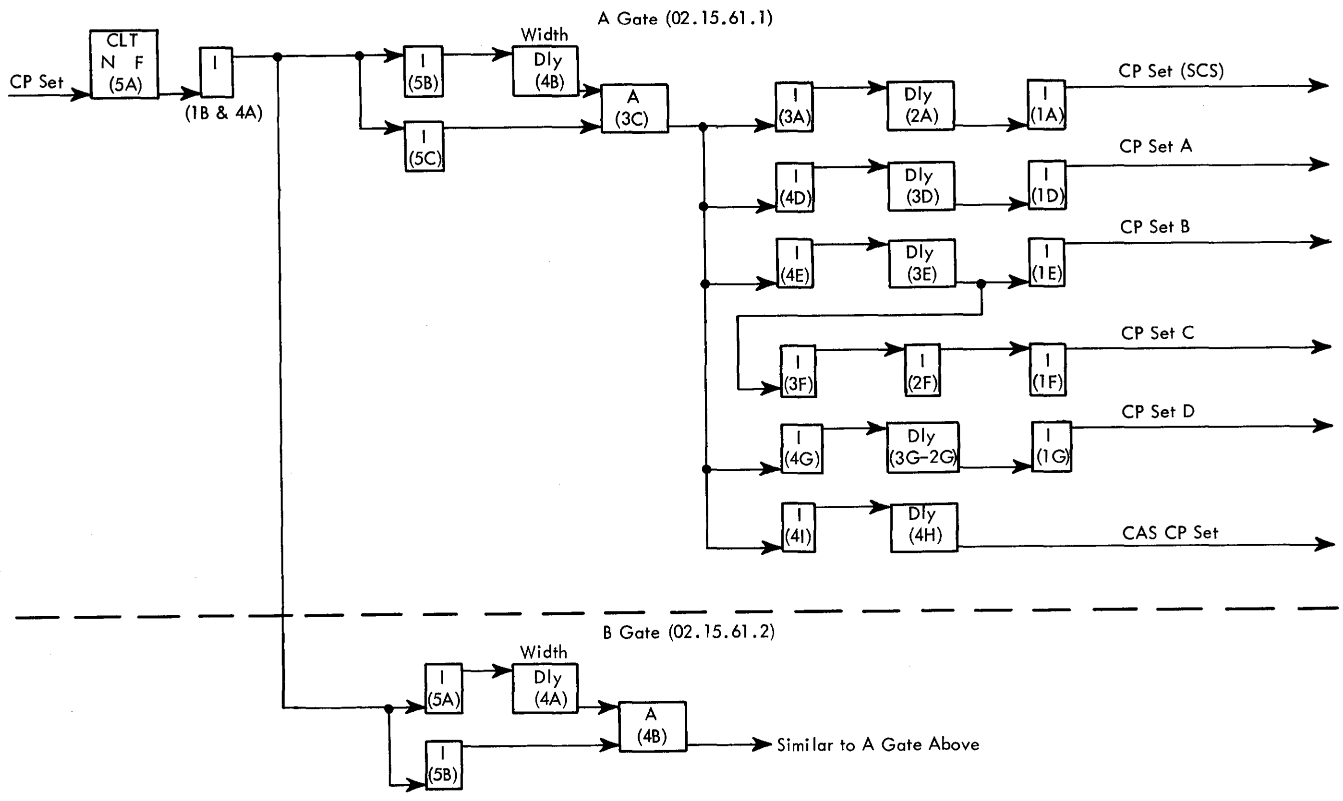


Figure 72. CP Set Pulse Distribution

Machine Timing Cycles

7904 II computer operation consists of several types of machine cycles concerned with both CPU and channel operations.

CPU CYCLES		CHANNEL CYCLES	
I	Instruction	B	Buffer
II	IBR Instruction	Chan I	Channel Instruction
E	Execution	Chan E	Channel Execution
L	Logical	Chan L	Channel Logical

The CPU cycles are directly concerned with CPU operations and the 8 cycle-point 1.4 microsecond clock. The channel cycles are directly concerned with channel operation and the 12 cycle-point 2.1 microsecond clock.

The cyclic sequence of a computer instruction is fixed and, depending on overlap conditions, begins with either an I or II cycle. The total number and types of machine cycles required for each instruction is determined by the number of steps to be performed before the operation is completed. This number varies depending on the conditions set forth by the particular instruction.

The various cycles essentially perform the following functions. Each will be discussed in greater detail in later sections.

I Cycle: The I cycle occurs because of a break in the overlap sequence. References are made to core storage and two sequential instructions are received into the program register and IBR for execution.

II Cycle: The II cycle occurs simultaneously with an E or L cycle and is used to perform functions concerned with the overlapping instruction in the IBR.

E Cycle: The E cycle is used for data or IA (indirect address) cycle references to core storage. This E cycle can be initiated by an instruction in either the program register or IBR depending on the conditions of overlap.

L Cycle: During the L cycle the computer performs logical or arithmetic functions without reference to core storage. This L cycle can be initiated by an instruction in either the program register or IBR depending on the conditions of overlap.

B Cycle: During the B cycle, a data channel uses core storage for either accepting or delivering data (or a data channel command word) in connection with an input-output operation. This B cycle occurs simultaneously with L cycles but never occurs simultaneously with I, E, or II cycles.

Chan I Cycle: The channel I cycle supplies 1.05 microsecond I time gatings to channel banks 1 and 2 and is used primarily for reset functions.

Chan E Cycle: The channel E cycle supplies 2.1 microsecond E times gatings to the CPU circuits as well as channel banks 1 and 2. This gating is used primarily during channel operations requiring references to core storage (i.e., POD's 54/64 or ENB).

Chan L Cycle: The channel L cycle supplies 2.1 microsecond L time gatings to banks 1 and 2 for channel circuit controls.

The CPU I and E cycle timings are such that they each gate their own instruction and data memory references. To accomplish this, I or E time for addressing comes up early (6 time of the previous cycle). There is, therefore, considerable overlapping of cycle times as shown in Figure 73.

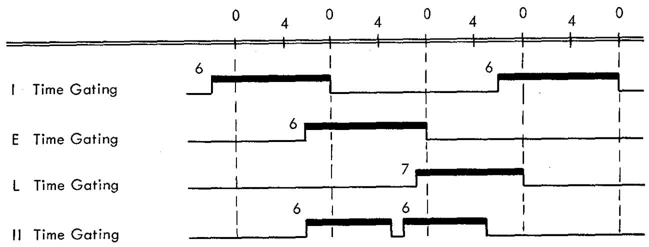


Figure 73. Cycle Time Relationship

Master I Time

I time is not the steadily recurring type of cycle that existed on previous systems; it results only because of a break in the overlap sequence of instructions. For example, this break in sequence might occur because of a successful transfer or skip condition resulting from the instruction in the program register or IBR.

Figure 74 shows condensed logic of I time. Note that even though the I time trigger might be turned on, its outputs may be blocked until such time that channel B cycle demands are satisfied. I time outputs are also gated by the master stop trigger or machine cycle gate (manual operations).

The basic objectives of the I cycle are to:

1. Select the proper memory/memories and gate the corresponding address(es) out to MAR(s).

2. Update the program counter to the proper address.

3. Gate the proper instruction(s) from the even/odd storage bus into the program register, storage register, tag register and IBR.

4. Perform address modification when applicable.

5. Determine and initiate the next type of machine cycle.

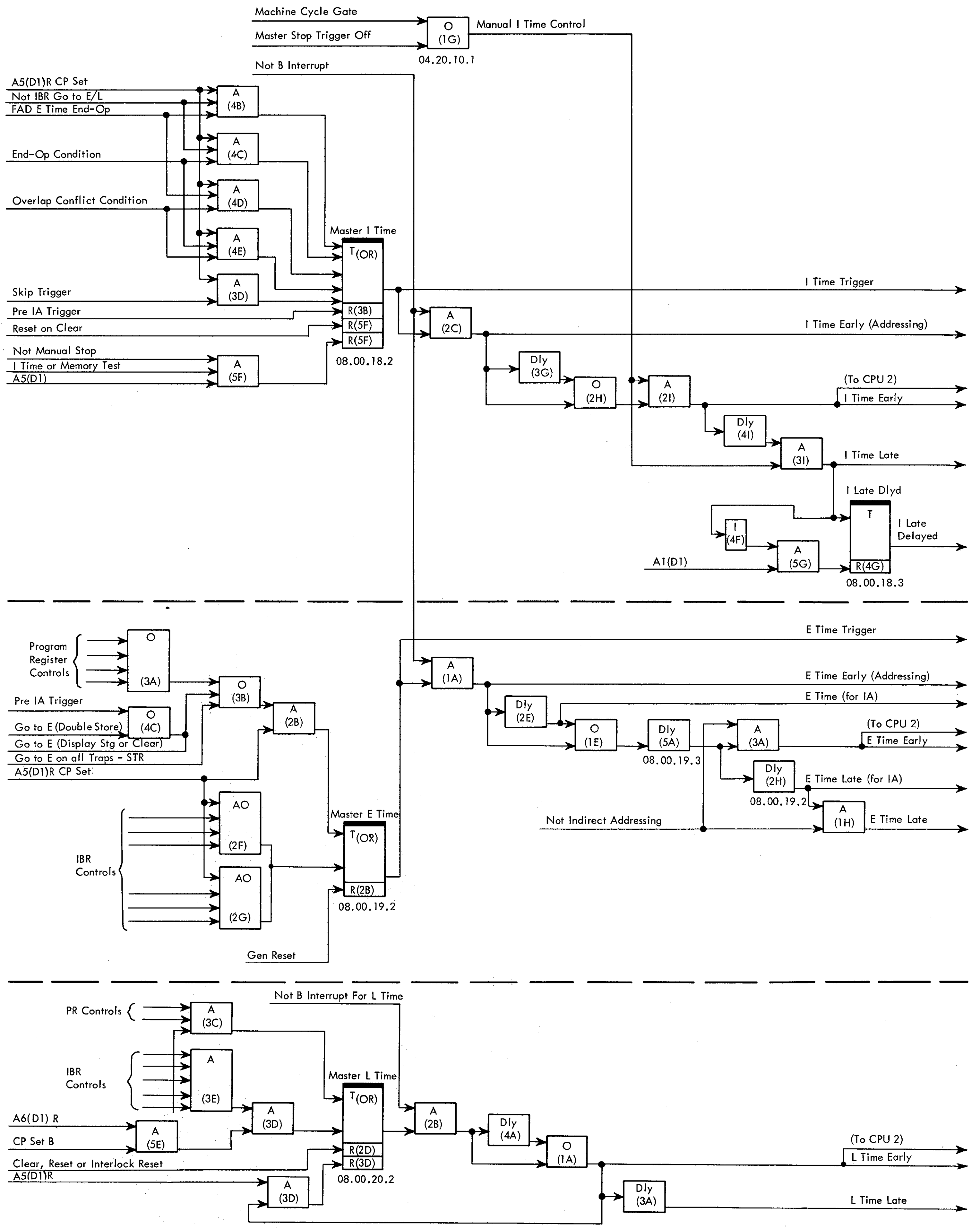


Figure 74. I, E, and L Time Cycle Logic

Address Gating

Objective 1 is initiated in the upper area of Figure 75 (sheet 1). This area shows the decisions required for initiating an I cycle. The skip trigger coming on as the result of an instruction in the program register nullifies overlap and forces an I cycle because the instruction in the IBR is going to be bypassed.

If the skip trigger does not come on but there is an overlap conflict condition, overlap will, again, be inhibited. In this latter case, the computer waits until the instruction in the program register has ended operation before initiating an I cycle. Special FAD E time end-op circuitry has been added because of timing considerations.

If the skip trigger is off and there is no overlap conflict condition (Figure 75), a test is made to see if the instruction being executed from the IBR will send the computer into either an E or L cycle. If this E/L test is successful, the computer proceeds to E/L time to complete execution of the overlapping instruction. If the test fails (the IBR contains a 1-cycle instruction, for example) the computer waits until the instruction in the program register completes operation and then initiates an I cycle.

Note that the I time trigger is turned on at 5 time (three clock pulses before the end) of the previous cycle. Turning the trigger on early allows its outputs to provide addressing gates starting at 6 time. Machine cycle design is such that the upcoming cycle provides its own memory address gating.

With the exception of MAR addressing controls, the I time does not logically start until the following 0 time.

Any B times requested by a data channel will be serviced before the I cycle is allowed to continue. This blocking of I time is accomplished by degating the output of the I time trigger with "B interrupt."

I time outputs can be grouped as follows:

I time early for addressing: These outputs are available starting at 6 time of the preceding cycle and are used primarily for gating the proper instruction addresses to memory.

I time early: These outputs are delayed outputs of the I time trigger and are available primarily for gating functions at the beginning or during early portions of the I cycle.

I time late: These outputs are delayed longer than the early pulses and are, therefore, available for gating functions during the latter portion of the I cycle. The I time trigger is turned off at I5(D1) time so it is the delayed outputs that maintain the required gating until the end of the cycle.

I time late delayed: This output is used to decrement the program counter under HTR control so that the counter will indicate the actual address of the HTR instruction.

Going into I time is like getting a fresh start in the program sequence and because of this, two sequential instructions will be referenced from core storage and brought into the CPU. Except for certain operations, the address of one of the next two instructions to be executed will always be found in either the program counter, address register, or the address portion of the IBR. The address of the other instruction can be obtained by incrementing or decrementing the first address. Both of these addresses when gated to the even and odd memories will retrieve the next two desired sequential instructions.

Note that the addressing blocks in the lower area of Figure 75 are divided from left to right into distinct groups according to function.

Group 1—Address Gating to MAR: One of these three gates will be active to provide one instruction address to MAR. This same address is gated to the index adders for either incrementing or decrementing.

Group 2—XAD Gating to MAR: This block provides the other instruction address to MAR. In the majority of cases this second address is an incremented (+1) value of the first address. In some cases, however, this second address is a decremented (−1) value of the first address. Note that the XAD to MAR path is only active when the computer is in overlap mode and not diagnostic mode.

Group 3—1's to XAD (3-17): These two blocks cause incrementing or decrementing in the index adders. Gating a 1 to XAD(17) is an unconditional I6(D2) function and causes address incrementing (+1). Gating 1's to XAD(3-16) along with the 1 to XAD(17) causes address decrementing (−1) by performing 1's complement addition.

Group 4—Address Updating Paths: This fourth functional group takes the incremented (or decremented) XAD address output and routes it to either the address register or IBR for temporary storage. This address will update the program counter at I1(D1) time.

Group 5—BAR Address Gating to MAR: This last functional group is used during channel trap operations when addressing the specific data channel trap location.

When progressing through the series of I time addressing decisions, a channel trap is concerned with group 5 and directs the routing of the buffer address register (BAR) to MAR.

For all other I time addressing decisions, groups 1 through 4 are used according to the following rules (Figure 75):

1. One block (and only one block) is always used from group 1. Either the PC, AR, or IBR will contain the correct address of one of the next two sequential instructions. This same address is also gated to the index adders.

2. An entry will always be made into group 2. Whether or not the index adders are gated to MAR depends on both overlap mode and diagnostic mode.

3. In group 3, a 1 is always gated to XAD(17) for incrementing purposes. During the two cases where decrementing is involved, the 1's to XAD(3-16) are also used.

4. One block (and only one block) is always used from group 4. These updating paths route the index adder value to either the address register or IBR as temporary storage until the program counter can be updated at the next I1(D1) time. Note that the value set into the address register or IBR at this time normally corresponds to the second instruction selected from memory (i.e., the instruction which will be set into the IBR at I4 time of the I cycle).

Using the above rules and making a systematic progression through the maze of addressing decisions should prove the logic of the I time addressing.

Program Counter Update

Objective 2 updates the program counter to the proper address. The program counter is always one step ahead of the current instruction being executed; therefore, the address to be set into the program counter will correspond to the instruction destined for the IBR.

In the previous section, addresses were sent to MAR during I6(D2). At the same time, the incremented (or decremented) value from the index adders was also routed and set into either the address register or IBR (Figure 75, sheet 1). These group 4 routing paths place the updating program counter address in "temporary storage." During I1(D1) time, this address is routed unmodified to the program counter via the index adders (Figure 75, sheet 2). Note that the pattern of decision making for the updating circuitry is similar to that for address gating discussed previously.

If the address register or IBR had received an incremented (+1) address from the index adders, that corresponding register contains the correct updating address for the program counter. If, however, the address register or IBR had received a decremented (−1) address from the index adders, the register originating the address contains the correct updating address for the program counter.

In some cases the program counter already contains the correct address. When these cases occur, the updating paths are blocked and the program counter value remains unchanged. One case, a successful class A transfer in trap mode, requires a carry to XAD(17) during update to put the value 00001 into the program counter.

Storage Bus Gating

The third objective is to gate the storage buses into the computer. The instructions which were addressed at the previous 6 time will be available by 4 time on the even and odd storage buses.

In Figure 75 (sheet 2) an external trapping condition will nullify the two new instructions arriving from core storage and, instead, force an STR operation by turning on PR (S, 9).

Excluding the trap condition, five routing and setting functions are available for the even and odd storage buses:

SB(S, 1-35) → SR(S, 1-35) and TR (18-20)	
SB(S, 1-2) → PR(S, 8-9)	
SB(S, 3-11) → PR(S, 1-9)	
SB(S, 1-35) → IBR(S, 1-35)	I4(D1) Set Pulse
SB(S, 1-35) → IBR(S, 1-35)	I4(D2) Set Pulse

Excluding the XEC instruction and channel traps for the present, the program counter is the deciding factor as to which storage bus is routed into the program register (PR) and IBR. Remember that the program counter was already updated (incremented) at I1 time and is +1 ahead of the current instruction. Because of this, if the program counter is at an odd value the even storage bus is gated to the program register and the odd bus to the IBR. If the program counter is at an even value the odd storage bus is gated to the program register and the even bus to the IBR.

A class A instruction coming into the program register is "pre-sensed" from the storage bus. If the instruction is a one cycle class A instruction (TIX, TNX, TXH, TXL, TXI) circuits are immediately set up to also route this same instruction into the IBR with an I4(D2) pulse. Note that this longer I4(D2) pulse overrides the instruction placed in the IBR with the normal I4(D1) pulse. The reason for this special gating into the IBR is because of addressing considerations which will be discussed in Volume 3.

Tracing through the flow chart (Figure 75—sheet 2) will also show the storage bus gatings for channel trap and XEC conditions. In some of these latter conditions the program counter value (even/odd) may cause some unnecessary gatings into the IBR. These gatings cause no problems, however, because overlapping is prevented at this time.

Figures 75 (sheet 2) and 76 provide a summary of the various SB gating conditions.

After the program register has been set with the new instruction, a check is made to determine if overlapping is possible. If so, the IBR loaded trigger is turned on.

Address Modification

Objective number 4 is address modification. This modification is performed during the I5(D1) clock pulse so that the modified address (if an index register was specified) will be available for MAR gating during the next two clock pulses; i.e., E6(D2) early.

Class A instructions (TIX, TNX, TXH, TXL, TXI) block address modification. Instead, the I4(D2) period of time is used to perform the specific test or operation on the index register (Figure 75, sheet 2). If the conditions of the test are successful, a PR condition-met-trigger is turned on; if the test fails, the trigger remains off.

If the instruction in the program register is indexable, SR(21-35) and the complement of the specified index register(s) are gated to the index adders with a carry to XAD(17). This 2's complement addition effectively performs a subtraction of the index register value from the address portion of the instruction. If no index register is specified, all 1's are gated from the index register along with the carry to XAD(17) to effectively subtract zero from the address in SR(21-35).

The modified address resulting in the index adders is gated to the address register for MAR gating at E6(D2) time. If the instruction is a POD 76, the index adders are also routed to the shift counter for further decoding.

Next Machine Cycle

At the same time that the address modification or class A testing is being performed, other circuitry is determining the next machine cycle.

If the PR instruction contains bits in positions 12 and 13, SR(12, 13), and meets the other conditions shown in Figure 75 (sheet 2), the pre-IA trigger is turned on and an E cycle follows.

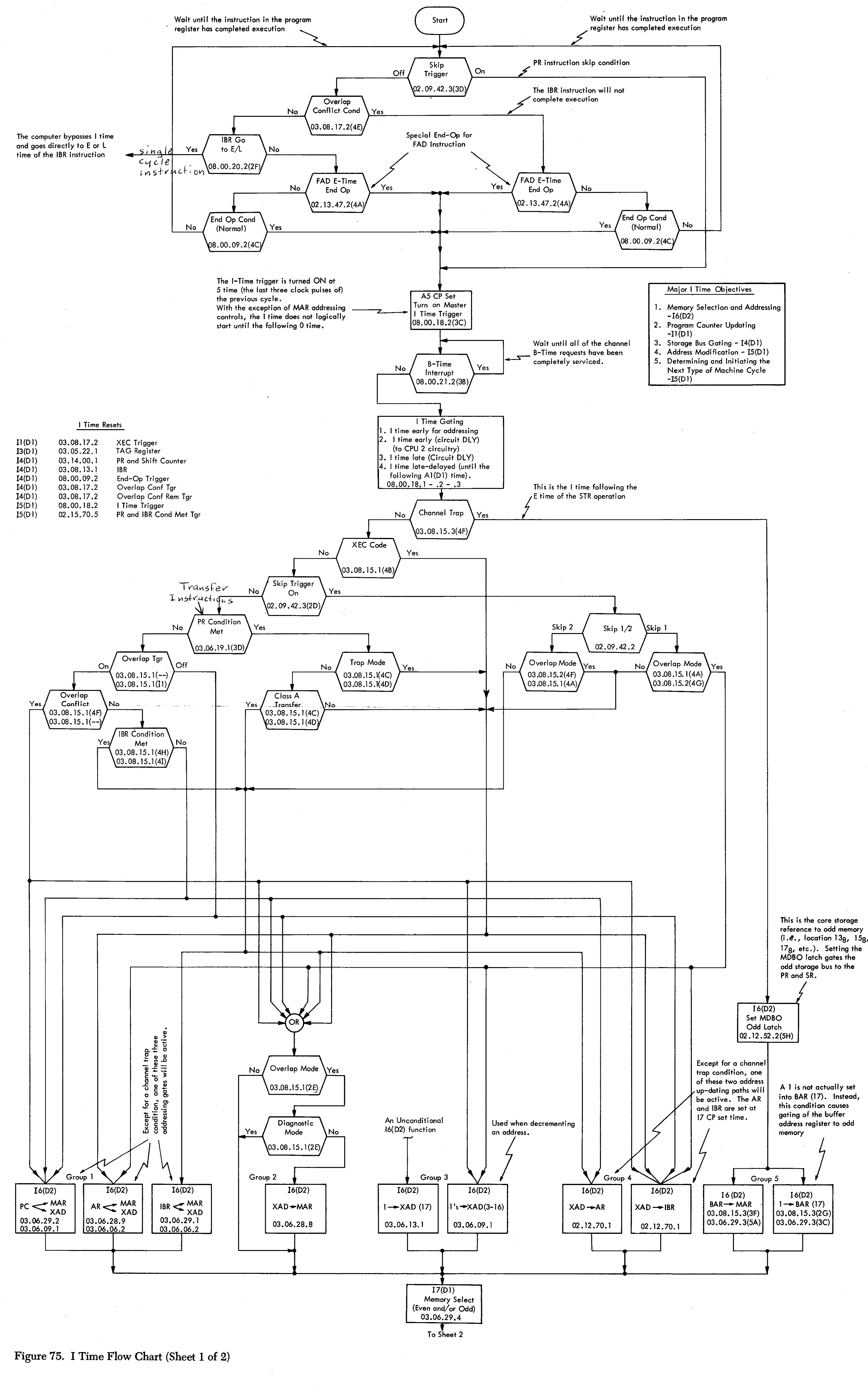
If no E(IA) cycle is called for, a test is made to see if a 1-cycle instruction is in the program register. If so, the end-op trigger is turned on and another I cycle follows.

If the instruction in the PR does not call for ending operation, tests are made to see if an E cycle should follow. If the test is successful, the master E time trigger is turned on and an E cycle follows. If, however, neither an I or E cycle is requested, the master L time trigger is turned on and an L cycle follows.

SB Gating	SB Gating Decision Blocks															
	XEC Operation								Chan Trap				Normal			
	A		B		C		D		E		F		G		H	
EVEN SB	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	N	Y	Y	N
SB→SR and TR	✓	✓	✓	✓											✓	✓
SB(S,1-2)→PR(S,8-9)	✓		✓												✓	
SB(S,3-11)→PR(S,1-9)		✓		✓												✓
SB→IBR I4(D1)			✓	✓							✓	✓	✓	✓		
SB→IBR I4(D2)	✓		✓				✓	✓							✓	
ODD SB																
SB→SR and TR					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
SB(S,1-2)→PR(S,8-9)					✓		✓		✓		✓			✓		
SB(S,3-11)→PR(S,1-9)						✓		✓		✓		✓	✓			
SB→IBR I4(D1)	✓	✓			✓	✓	✓		✓	✓					✓	✓
SB→IBR I4(D2)					✓				✓		✓			✓		

Y indicates Yes   N indicates No

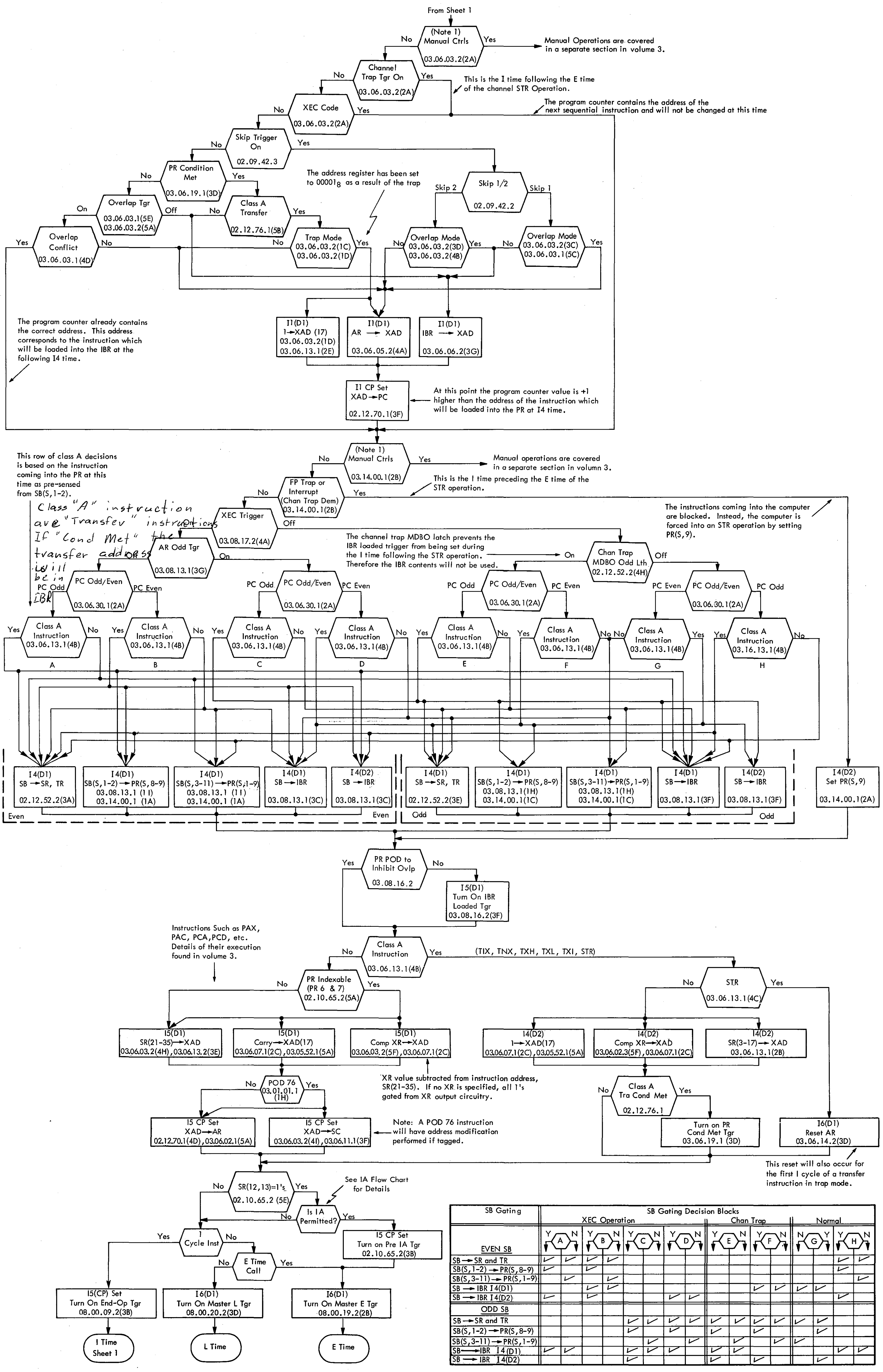
Figure 76. SB Gating Decision Chart


Start
Wait until the instruction in the program register has completed execution
Wait until the instruction in the program register has completed execution
Skip Trigger 02.09.42.3(3D)
Off
On
PR instruction skip condition
Overlap Conflict Cond 03.08.17.2(4E)
No
Yes
The IBR instruction will not complete execution
The computer bypasses I time and goes directly to E or L time of the IBR instruction
single cycle instruction
IBR Go to E/L 08.00.20.2(2F)
Special End-Op for FAD Instruction
FAD E-Time End Op 02.13.47.2(4A)
FAD E-Time End Op 02.13.47.2(4A)
End Op Cond (Normal) 08.00.09.2(4C)
End Op Cond (Normal) 08.00.09.2(4C)
The I-Time trigger is turned ON at 5 time (the last three clock pulses of) the previous cycle. With the exception of MAR addressing controls, the I time does not logically start until the following 0 time.
A5 CP Set Turn on Master I Time Trigger 08.00.18.2(3C)
Wait until all of the channel B-Time requests have been completely serviced.
B-Time Interrupt 08.00.21.2(3B)
Major I Time Objectives
1. Memory Selection and Addressing -I6(D2)
2. Program Counter Updating -I1(D1)
3. Storage Bus Gating - I4(D1)
4. Address Modification - I5(D1)
5. Determining and Initiating the Next Type of Machine Cycle -I5(D1)
I Time Gating
1. I time early for addressing
2. I time early (circuit DLY) (to CPU 2 circuitry)
3. I time late (Circuit DLY)
4. I time late-delayed (until the following A1(D1) time).
08.00.18.1 - .2 - .3
This is the I time following the E time of the STR operation
Channel Trap 03.08.15.3(4F)
XEC Code 03.08.15.1(4B)
Skip Trigger On 02.09.42.3(2D)
Transfer Instructions
PR Condition Met 03.06.19.1(3D)
Skip 1/2 02.09.42.2
Skip 2
Skip 1
Overlap Tgr 03.08.15.1(--) 03.08.15.1(I1)
Trap Mode 03.08.15.1(4C) 03.08.15.1(4D)
Overlap Mode 03.08.15.2(4F) 03.08.15.1(4A)
Overlap Mode 03.08.15.1(4A) 03.08.15.2(4G)
Overlap Conflict 03.08.15.1(4F) 03.08.15.1(--)
Class A Transfer 03.08.15.1(4C) 03.08.15.1(4D)
IBR Condition Met 03.08.15.1(4H) 03.08.15.1(4I)
This is the core storage reference to odd memory (i.e., location 13₈, 15₈, 17₈, etc.). Setting the MDBO latch gates the odd storage bus to the PR and SR.
I6(D2) Set MDBO Odd Latch 02.12.52.2(5H)
OR
Overlap Mode 03.08.15.1(2E)
Diagnostic Mode 03.08.15.1(2E)
Except for a channel trap condition, one of these three addressing gates will be active.
An Unconditional I6(D2) function
Used when decrementing an address.
Except for a channel trap condition, one of these two address up-dating paths will be active. The AR and IBR are set at I7 CP set time.
A 1 is not actually set into BAR (17). Instead, this condition causes gating of the buffer address register to odd memory
Group 1
Group 2
Group 3
Group 4
Group 5
I6(D2) PC ← MAR / XAD 03.06.29.2 03.06.09.1
I6(D2) AR ← MAR / XAD 03.06.28.9 03.06.06.2
I6(D2) IBR ← MAR / XAD 03.06.29.1 03.06.06.2
I6(D2) XAD→MAR 03.06.28.8
I6(D2) 1→XAD (17) 03.06.13.1
I6(D2) 1's→XAD(3-16) 03.06.09.1
I6(D2) XAD→AR 02.12.70.1
I6(D2) XAD→IBR 02.12.70.1
I6(D2) BAR→MAR 03.08.15.3(3F) 03.06.29.3(5A)
I6(D2) 1→BAR (17) 03.08.15.3(2G) 03.06.29.3(3C)
I7(D1) Memory Select (Even and/or Odd) 03.06.29.4
To Sheet 2
Yes
No

I Time Resets

I1(D1)	03.08.17.2	XEC Trigger
I3(D1)	03.05.22.1	TAG Register
I4(D1)	03.14.00.1	PR and Shift Counter
I4(D1)	03.08.13.1	IBR
I4(D1)	08.00.09.2	End-Op Trigger
I4(D1)	03.08.17.2	Overlap Conf Tgr
I4(D1)	03.08.17.2	Overlap Conf Rem Tgr
I5(D1)	08.00.18.2	I Time Trigger
I5(D1)	02.15.70.5	PR and IBR Cond Met Tgr

Figure 75. I Time Flow Chart (Sheet 1 of 2)



From Sheet 1
(Note 1) Manual Ctrls 03.06.03.2(2A)
No
Yes
Manual Operations are covered in a separate section in volume 3.
Channel Trap Tgr On 03.06.03.2(2A)
This is the I time following the E time of the channel STR Operation.
The program counter contains the address of the next sequential instruction and will not be changed at this time
XEC Code 03.06.03.2(2A)
Skip Trigger On 02.09.42.3
PR Condition Met 03.06.19.1(3D)
The address register has been set to 00001₈ as a result of the trap
Skip 1/2 02.09.42.2
Skip 2
Skip 1
On
Overlap Tgr 03.06.03.1(5E) 03.06.03.2(5A)
Off
Class A Transfer 02.12.76.1(5B)
Overlap Conflict 03.06.03.1(4D)
Trap Mode 03.06.03.2(1C) 03.06.03.2(1D)
Overlap Mode 03.06.03.2(3D) 03.06.03.2(4B)
Overlap Mode 03.06.03.2(3C) 03.06.03.1(5C)
The program counter already contains the correct address. This address corresponds to the instruction which will be loaded into the IBR at the following I4 time.
I1(D1) 1→XAD (17) 03.06.03.2(1D) 03.06.13.1(2E)
I1(D1) AR → XAD 03.06.05.2(4A)
I1(D1) IBR → XAD 03.06.06.2(3G)
I1 CP Set XAD→PC 02.12.70.1(3F)
At this point the program counter value is +1 higher than the address of the instruction which will be loaded into the PR at I4 time.
This row of class A decisions is based on the instruction coming into the PR at this time as pre-sensed from SB(S,1-2).
Class "A" instruction are "Transfer" instructions If "Cond Met" the transfer address will be in IBR
(Note 1) Manual Ctrls 03.14.00.1(2B)
Manual operations are covered in a separate section in volumn 3.
This is the I time preceding the E time of the STR operation.
FP Trap or Interrupt (Chan Trap Dem) 03.14.00.1(2B)
XEC Trigger 03.08.17.2(4A)
The instructions coming into the computer are blocked. Instead, the computer is forced into an STR operation by setting PR(S,9).
The channel trap MDBO latch prevents the IBR loaded trigger from being set during the I time following the STR operation. Therefore the IBR contents will not be used.
AR Odd Tgr 03.08.13.1(3G)
Chan Trap MDBO Odd Lth 02.12.52.2(4H)
PC Odd/Even 03.06.30.1(2A)
PC Odd
PC Even
Class A Instruction 03.06.13.1(4B)
Class A Instruction 03.16.13.1(4B)
A
B
C
D
E
F
G
H
I4(D1) SB→SR, TR 02.12.52.2(3A)
I4(D1) SB(S,1-2)→PR(S,8-9) 03.08.13.1 (1I) 03.14.00.1 (1A)
I4(D1) SB(S,3-11)→PR(S,1-9) 03.08.13.1 (1I) 03.14.00.1 (1A)
I4(D1) SB→IBR 03.08.13.1(3C)
I4(D2) SB→IBR 03.08.13.1(3C)
I4(D1) SB→SR, TR 02.12.52.2(3E)
I4(D1) SB(S,1-2)→PR(S,8-9) 03.08.13.1(1H) 03.14.00.1(1C)
I4(D1) SB(S,3-11)→PR(S,1-9) 03.08.13.1(1H) 03.14.00.1(1C)
I4(D1) SB→IBR 03.08.13.1(3F)
I4(D2) SB→IBR 03.08.13.1(3F)
I4(D2) Set PR(S,9) 03.14.00.1(2A)
Even
Odd
PR POD to Inhibit Ovlp 03.08.16.2
I5(D1) Turn On IBR Loaded Tgr 03.08.16.2(3F)
Instructions Such as PAX, PAC, PCA, PCD, etc. Details of their execution found in volume 3.
Class A Instruction 03.06.13.1(4B)
(TIX, TNX, TXH, TXL, TXI, STR)
PR Indexable (PR 6 & 7) 02.10.65.2(5A)
STR 03.06.13.1(4C)
I5(D1) SR(21-35)→XAD 03.06.03.2(4H), 03.06.13.2(3E)
I5(D1) Carry→XAD(17) 03.06.07.1(2C), 03.05.52.1(5A)
I5(D1) Comp XR→XAD 03.06.03.2(5F), 03.06.07.1(2C)
I4(D2) 1→XAD(17) 03.06.07.1(2C), 03.05.52.1(5A)
I4(D2) Comp XR→XAD 03.06.02.3(5F), 03.06.07.1(2C)
I4(D2) SR(3-17)→XAD 03.06.13.1(2B)
XR value subtracted from instruction address, SR(21-35). If no XR is specified, all 1's gated from XR output circuitry.
POD 76 03.01.01.1 (1H)
Class A Tra Cond Met 02.12.76.1
I5 CP Set XAD→AR 02.12.70.1(4D), 03.06.02.1(5A)
I5 CP Set XAD→SC 03.06.03.2(4I), 03.06.11.1(3F)
Note: A POD 76 instruction will have address modification performed if tagged.
Turn on PR Cond Met Tgr 03.06.19.1 (3D)
I6(D1) Reset AR 03.06.14.2(3D)
This reset will also occur for the first I cycle of a transfer instruction in trap mode.
SR(12,13)=1's 02.10.65.2 (5E)
See IA Flow Chart for Details
Is IA Permitted?
I5 CP Set Turn on Pre IA Tgr 02.10.65.2(3B)
1 Cycle Inst
E Time Call
I5(CP) Set Turn On End-Op Tgr 08.00.09.2(3B)
I6(D1) Turn On Master L Tgr 08.00.20.2(3D)
I6(D1) Turn On Master E Tgr 08.00.19.2(2B)
I Time Sheet 1
L Time
E Time

SB Gating	SB Gating Decision Blocks															
	XEC Operation								Chan Trap				Normal			
EVEN SB	A Y	A N	B Y	B N	C Y	C N	D Y	D N	E Y	E N	F Y	F N	G N	G Y	H Y	H N
SB→SR and TR	✓	✓	✓	✓											✓	✓
SB(S,1-2)→PR(S,8-9)	✓		✓												✓	
SB(S,3-11)→PR(S,1-9)		✓		✓												✓
SB→IBR I4(D1)			✓	✓							✓	✓	✓	✓		
SB→IBR I4(D2)	✓		✓				✓	✓							✓	
ODD SB																
SB→SR and TR					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
SB(S,1-2)→PR(S,8-9)					✓		✓		✓		✓			✓		
SB(S,3-11)→PR(S,1-9)						✓		✓		✓		✓	✓			
SB→IBR I4(D1)	✓	✓			✓	✓	✓		✓	✓					✓	✓
SB→IBR I4(D2)					✓				✓		✓			✓		

Figure 75. I Time Flow Chart (Sheet 2 of 2)

Master E Time

Figure 74 shows condensed logic for the E cycle. Note that this E cycle can be initiated by an instruction in either the program register or IBR depending on overlap conditions. In either case, the E time trigger is turned on by 6 time of the preceding cycle to allow memory addressing. In this manner, E time provides memory addressing for its own data reference.

Outputs from the E time trigger may be blocked until channel B time requests are satisfied. This same situation exists for I time, which was described previously. After all of the B times are satisfied, the blocking is removed and the computer proceeds with the instruction.

The master E time trigger is reset at E4 time; however, by use of delays, E time gating circuits are extended until the end of the cycle. E time outputs include:

1. E time early for addressing (not deconditioned by IA)
2. E time early
3. E time early (for IA)
4. E time late
5. E time late (for IA)

Note that, unless specifically labeled "for IA," E time gatings (with exception of E time for addressing) are blocked during an IA (indirect address) cycle. In this manner the instruction execution is blocked until the normal E cycle (not IA cycle) occurs. If an instruction requires several consecutive E cycles (CVR, for example) the master E time trigger is set and reset for each cycle.

IA E Cycle

Indirect addressing requires making a second reference to core storage. This second reference either obtains the actual data word in cases of data handling instructions (CLA, ADD, etc.), or the new transfer address in cases of transfer instructions (TRA, TQO, etc.).

During the first memory reference (IA E cycle), a new word is brought into the storage register from a location specified by positions 21-35 of the original instruction word. Positions 21-35 of the IA word contain, in effect, a new memory reference. This new address when sent to core storage retrieves the desired data (or transfer address).

Several examples of indirect addressing are shown in an earlier section of this manual, "Instruction Addressing."

The basic objectives of an IA cycle are:

1. Make proper reference to memory for the IA word.
2. Gate the storage bus even/odd into the storage register.
3. Perform address modification according to the tag specified in the IA word.
4. Determine and initiate the next type of machine cycle.
5. Select the proper memory(s) and gate the corresponding address(es) to MAR(s).

An IA cycle can be initiated by an instruction in either the program register (I time) or the IBR (II time). See Figure 77. Test conditions for the IA cycle are similar in either case.

Indirect addressing has the following restrictions:

1. The instruction must be indexable. Non-indexable instructions such as the class A type (TIX, TNX, etc.) make use of the entire decrement portion of the instruction word for indexing purposes. Therefore, positions 12 and 13 lose their meaning for indirect addressing. Other non-indexable instructions such as PAX, PXA, SXA, SXD, etc., actually operate on the index register and are, therefore, not subject to address modification.

2. Except for transfer instructions, indirect addressing is only possible on instructions of 2X, 3X, 4X, 5X, and 6X operation codes. Codes below 2X and above 6X require L cycles for their execution. Without references to core storage, indirect addressing has no purpose.

3. Transfer instructions can normally be indirectly addressed. However, this feature is nullified (except for TTR/ESNT) if the machine is in the trap mode of operation. When in the trap mode, the address of the transfer instruction is stored in location 00000_8 and if the transfer conditions are met, the computer traps to location 00001_8. Because a transfer is never actually made to the transfer address, indirect addressing in the trap mode accomplishes no useful purpose. The programmer's trap subroutine must test to determine if indirect addressing existed in the transfer instruction.

If the IA cycle is initiated by the program register instruction, a pre IA trigger is turned on. If the IA cycle is initiated by the IBR instruction a pre IIA trigger is turned on. In either case, a common IA trigger is turned on at the next E0 time.

II times are blocked from occurring simultaneously with the E(IA) cycle because of a conflict in usage of the index adders at 5 time.

Note that during an II cycle the instruction in the program register may detect a skip condition which will bypass the overlapping instruction. A trap condition may also demand recognition ahead of the overlapping instruction. In either of these two cases, overlapping is nullified and the E(IA) cycle is blocked from occurring.

If the IA cycle is requested by the overlapping instruction (i.e., II time), the IA cycle must wait for an end-op signal from the instruction in the program register. This condition would occur, for example, if the overlapping instruction is preceded by a multiply instruction. See example on Figure 77.

IA Memory Reference: The address to be gated to MAR depends on whether the IA cycle was initiated by a preceding I or II cycle. This fact is determined by the OFF and ON states of the end-op trigger, respectively, and causes gating of either the address register or IBR to MAR (Figure 77). Note that only one address is sent and only one memory is selected.

A check is made to determine if the value in the address register or IBR is even/odd. If odd, the AR odd trigger is turned on for later references. The even/odd conditions are also dependent on whether the computer is in normal or diagnostic mode of operation.

Storage Bus Gating: The data word arriving at E4 time is gated into the storage register. The proper bus is determined by the status of the AR odd trigger (Figure 77).

Address Modification: At E(IA)5(D1) time, Figure 77, SR(21-35) are routed to the index adders together with the complement of the specified index register. A carry to XAD(17) causes 2's complement addition and the index register value is effectively subtracted from SR(21-35).

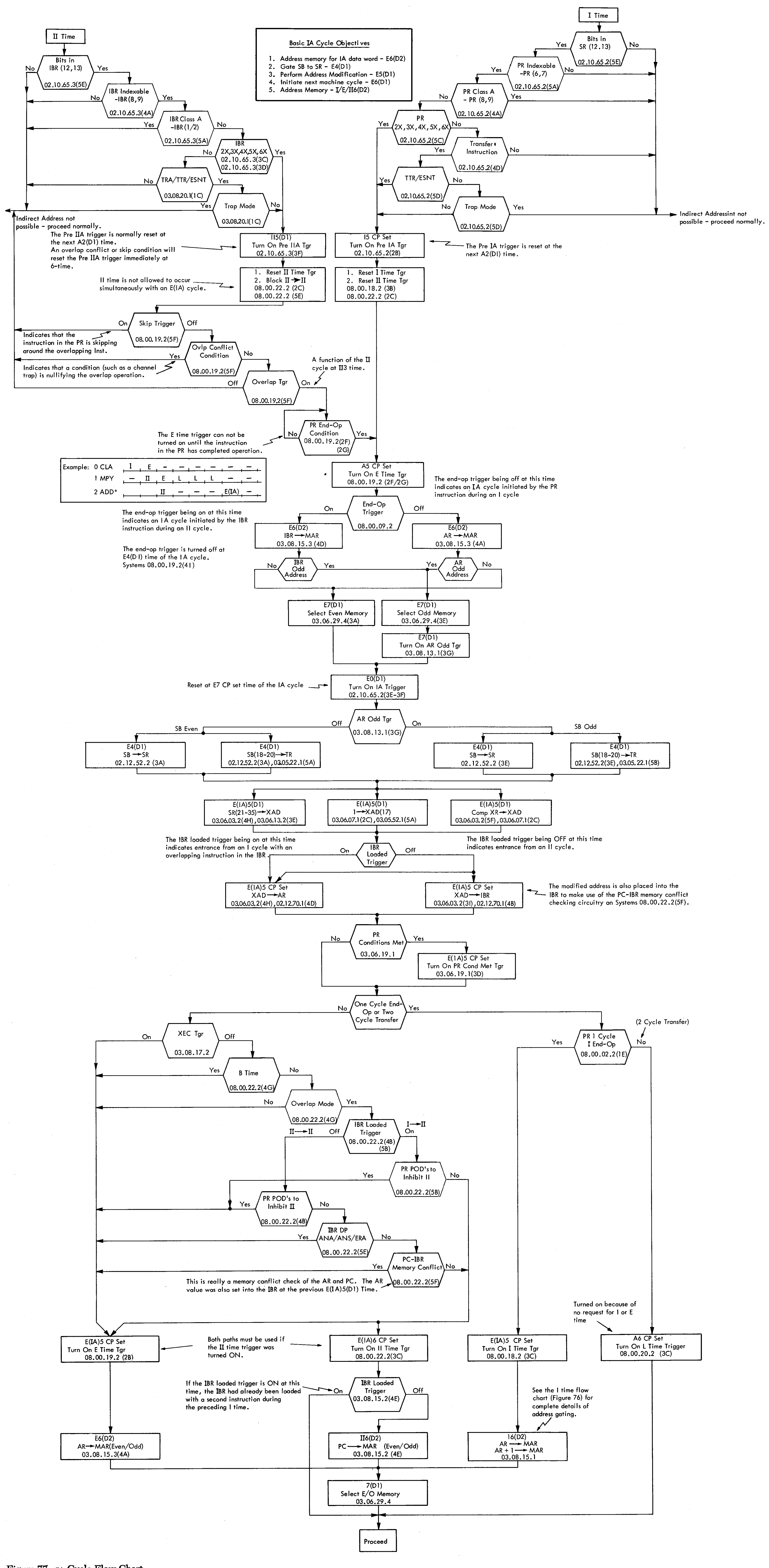
This new modified memory address is set into the address register for gating to MAR at E6(D2) time. This modified address is also gated to IBR(21-35) if the IBR loaded trigger is off. If the IBR loaded trigger is off, the IA cycle was initiated from an II cycle. Under these conditions, an II cycle may follow the IA cycle. Routing the modified address to the IBR allows use of memory conflict detection circuits on Systems 08.00.22.2 (5F).

Next Machine Cycle: One cycle transfer instructions end-op during the IA cycle; therefore, the next cycle will be an I cycle. Two cycle transfer instructions require an L cycle to complete operation. Because of this, an L cycle will follow the IA cycle.

If a 1-cycle or 2-cycle transfer condition does not exist, the next cycle must be an E cycle (except for possible B cycle interrupts). If conditions allow, an II cycle may also occur simultaneously with this next E cycle. Note that the path which turns on the II time trigger (Figure 77) also turns on the E time trigger. The reverse condition, however, is not true.

MAR Gating and Selection: A 1-cycle E(IA) transfer end-op causes two sequential addresses to be sent to MAR from the address register and index adders. This condition was covered in a previous section on I time and in Figure 75.

All other conditions force a data E cycle with the modified data address sent to MAR from the address register. If an II cycle is also allowed to occur, the program counter is sent to MAR to fetch a new overlapping instruction. There is one condition on the PC→MAR gating, however. The IBR loaded trigger being on indicates an II time without a reference to memory because the IBR had already been loaded with an instruction during the previous I time.



Basic IA Cycle Objectives
1. Address memory for IA data word - E6(D2)
2. Gate SB to SR - E4(D1)
3. Perform Address Modification - E5(D1)
4. Initiate next machine cycle - E6(D1)
5. Address Memory - I/E/II6(D2)
II Time
Bits in IBR (12,13) 02.10.65.3(5E)
IBR Indexable -IBR(8,9) 02.10.65.3(4A)
IBR Class A -IBR(1/2) 02.10.65.3(5A)
IBR 2X,3X,4X,5X,6X 02.10.65.3(3C) 02.10.65.3(3D)
TRA/TTR/ESNT 03.08.20.1(1C)
Trap Mode 03.08.20.1(1C)
Indirect Address not possible - proceed normally.
The Pre IIA trigger is normally reset at the next A2(D1) time. An overlap conflict or skip condition will reset the Pre IIA trigger immediately at 6-time.
II5(D1) Turn On Pre IIA Tgr 02.10.65.3(3F)
II time is not allowed to occur simultaneously with an E(IA) cycle.
1. Reset II Time Tgr 2. Block II→II 08.00.22.2 (2C) 08.00.22.2 (5E)
I Time
Bits in SR (12,13) 02.10.65.2(5E)
PR Indexable -PR (6,7) 02.10.65.2(5A)
PR Class A - PR (8,9) 02.10.65.2(4A)
PR 2X,3X,4X,5X,6X 02.10.65.2(5C)
Transfer Instruction 02.10.65.2(4D)
TTR/ESNT 02.10.65.2(5D)
Trap Mode 02.10.65.2(5D)
Indirect Addressint not possible - proceed normally.
I5 CP Set Turn On Pre IA Tgr 02.10.65.2(2B)
The Pre IA trigger is reset at the next A2(D1) time.
1. Reset I Time Tgr 2. Reset II Time Tgr 08.00.18.2 (3B) 08.00.22.2 (2C)
Skip Trigger 08.00.19.2(5F)
Indicates that the instruction in the PR is skipping around the overlapping Inst.
Ovlp Conflict Condition 08.00.19.2(5F)
Indicates that a condition (such as a channel trap) is nullifying the overlap operation.
Overlap Tgr 08.00.19.2(5F)
A function of the II cycle at II3 time.
The E time trigger can not be turned on until the instruction in the PR has completed operation.
PR End-Op Condition 08.00.19.2(2F) (2G)
Example: 0 CLA I E - - - - - -
1 MPY - II E L L L - -
2 ADD* II - - - E(IA) -
A5 CP Set Turn On E Time Tgr 08.00.19.2 (2F/2G)
The end-op trigger being off at this time indicates an IA cycle initiated by the PR instruction during an I cycle
End-Op Trigger 08.00.09.2
The end-op trigger being on at this time indicates an IA cycle initiated by the IBR instruction during an II cycle.
The end-op trigger is turned off at E4(D1) time of the IA cycle. Systems 08.00.19.2(4I)
E6(D2) IBR→MAR 03.08.15.3 (4D)
E6(D2) AR→MAR 03.08.15.3 (4A)
IBR Odd Address
AR Odd Address
E7(D1) Select Even Memory 03.06.29.4(3A)
E7(D1) Select Odd Memory 03.06.29.4(3E)
E7(D1) Turn On AR Odd Tgr 03.08.13.1(3G)
Reset at E7 CP set time of the IA cycle
E0(D1) Turn On IA Trigger 02.10.65.2(3E-3F)
AR Odd Tgr 03.08.13.1(3G)
SB Even
SB Odd
E4(D1) SB→SR 02.12.52.2 (3A)
E4(D1) SB(18-20)→TR 02.12.52.2(3A),03.05.22.1(5A)
E4(D1) SB→SR 02.12.52.2 (3E)
E4(D1) SB(18-20)→TR 02.12.52.2(3E),03.05.22.1(5B)
E(IA)5(D1) SR(21-35)→XAD 03.06.03.2(4H),03.06.13.2(3E)
E(IA)5(D1) 1→XAD(17) 03.06.07.1(2C),03.05.52.1(5A)
E(IA)5(D1) Comp XR→XAD 03.06.03.2(5F),03.06.07.1(2C)
The IBR loaded trigger being on at this time indicates entrance from an I cycle with an overlapping instruction in the IBR.
IBR Loaded Trigger
The IBR loaded trigger being OFF at this time indicates entrance from an II cycle.
E(IA)5 CP Set XAD→AR 03.06.03.2(4H),02.12.70.1(4D)
E(IA)5 CP Set XAD→IBR 03.06.03.2(3I),02.12.70.1(4B)
The modified address is also placed into the IBR to make use of the PC-IBR memory conflict checking circuitry on Systems 08.00.22.2(5F).
PR Conditions Met 03.06.19.1
E(IA)5 CP Set Turn On PR Cond Met Tgr 03.06.19.1(3D)
One Cycle End-Op or Two Cycle Transfer
XEC Tgr 03.08.17.2
B Time 08.00.22.2(4G)
Overlap Mode 08.00.22.2(4G)
IBR Loaded Trigger 08.00.22.2(4B) (5B)
II→II
I→II
PR POD's to Inhibit II 08.00.22.2(5B)
PR POD's to Inhibit II 08.00.22.2(4B)
IBR DP ANA/ANS/ERA 08.00.22.2(5E)
PC-IBR Memory Conflict 08.00.22.2(5F)
This is really a memory conflict check of the AR and PC. The AR value was also set into the IBR at the previous E(IA)5(D1) Time.
(2 Cycle Transfer)
PR 1 Cycle I End-Op 08.00.02.2(1E)
Turned on because of no request for I or E time
E(IA)5 CP Set Turn On E Time Tgr 08.00.19.2 (2B)
Both paths must be used if the II time trigger was turned ON.
E(IA)6 CP Set Turn On II Time Tgr 08.00.22.2(3C)
E(IA)5 CP Set Turn On I Time Tgr 08.00.18.2 (3C)
A6 CP Set Turn On L Time Trigger 08.00.20.2 (3C)
If the IBR loaded trigger is ON at this time, the IBR had already been loaded with a second instruction during the preceding I time.
IBR Loaded Trigger 03.08.15.2(4E)
See the I time flow chart (Figure 76) for complete details of address gating.
E6(D2) AR→MAR(Even/Odd) 03.08.15.3(4A)
II6(D2) PC→MAR (Even/Odd) 03.08.15.2 (4E)
I6(D2) AR→MAR AR+1→MAR 03.08.15.1
7(D1) Select E/O Memory 03.06.29.4
Proceed
Yes
No
On
Off

Figure 77. IA Cycle Flow Chart

Master L Time

L time provides a logic cycle of operation during which the computer performs functions not related to core storage.

Control circuitry (Figure 74) is similar in some respects to E time because it can be initiated by an instruction in either the program register or IBR depending on overlap conditions (only IBR shift instructions can force an L cycle).

The turn-on logic is such that an L cycle occurs if the instruction does not specifically call for an I or E cycle. The I or E turn-on is at A5 CP set; if these I or E controls are not present, the master L time trigger is turned on one clock pulse later at A6 CP set.

As long as the input PR or IBR controls are active, the L time trigger is turned on again each cycle. In this manner, the L time trigger remains on as long as needed. Use of delay circuitry produces both "L time early" and "L time late" outputs for circuit controls in CPU-1 and CPU-2.

When the master stop trigger is turned on, the computer remains at whatever cycle time it was proceeding to when the master stop trigger came on. An inhibit L trigger is turned on by the master stop trigger (Figure 78) to block both the master L time and II time outputs (Figure 74).

The inhibit L trigger is also turned on by the HTR and HPR instructions. L time of these two halt instructions is blocked until the start key is depressed. At this time, the inhibit L trigger is turned off and the computer either transfers or proceeds to the next instruction.

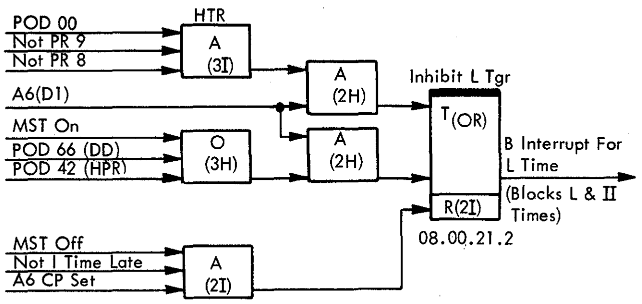


Figure 78. Inhibit L Trigger Logic

During channel operations, the CPU remains in L time until the next instruction is begun.

Master II Time

II time (IBR I time) is the instruction cycle for the overlapping instruction in the IBR. During this cycle address modification and preliminary tests are performed by the overlapping instruction. When not prohibited, II time occurs simultaneously with either an E or L cycle of the preceding instruction.

Figure 79 shows basic condensed logic of the II time trigger circuitry. Note that II time can be initiated by instruction controls from either the program register or IBR. The II time trigger is turned on at 6 time of the preceding cycle to allow memory addressing. In this manner, II time performs memory addressing for its own data reference.

Details of II time together with an II time flow chart are found in Volume 3.

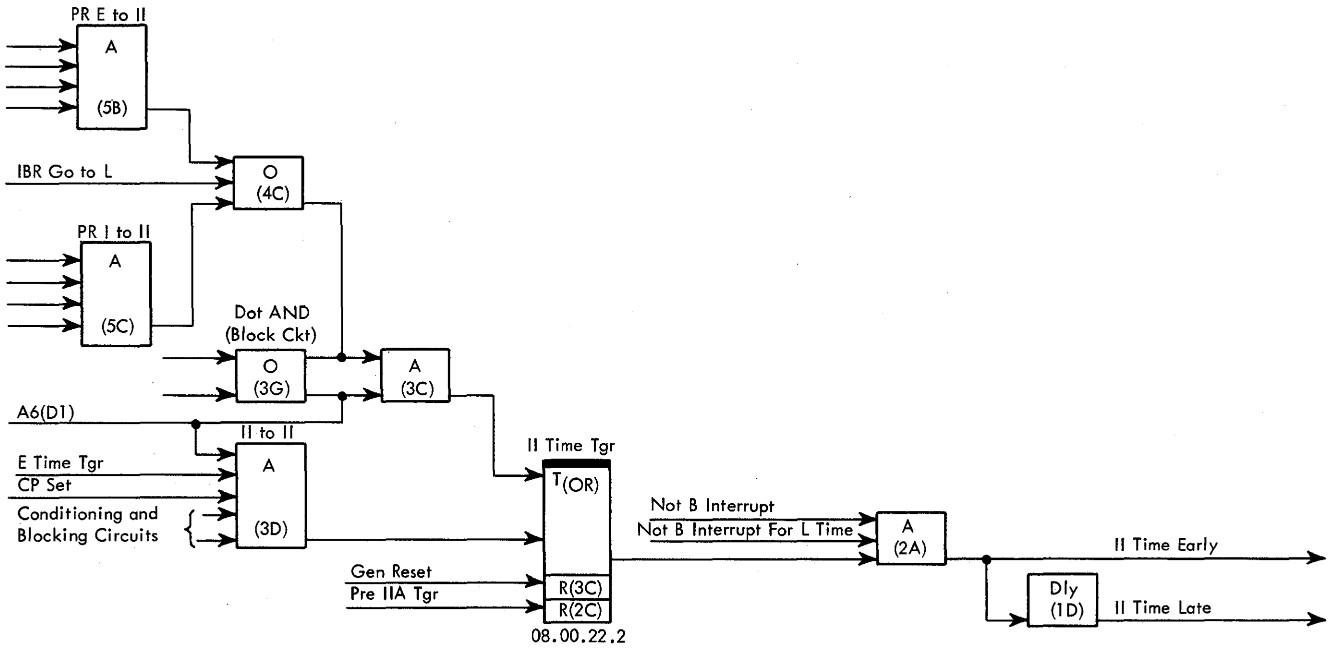


Figure 79. II Time Condensed Logic

Master B Time

B (buffer) time is the cycle during which one of the data channels makes reference to core storage. Most I/O devices move at a fixed rate and therefore request or transmit data at specific intervals. When data word requests are not serviced in time, information is lost and I/O checks result. Because of this timing requirement, channel B cycle demands are serviced during the cycle immediately following the request. If the next CPU cycle is to be an I, E or II cycle, the output of the master trigger is blocked. If the next CPU cycle is to be a normal L cycle, both the B and L cycles occur simultaneously.

"B cycle demand" is sent out by the channel early in the cycle such that the signal arrives at the multiplexor before channel 9 time. Provided the computer is not servicing a channel trap or a POD 64 instruction, the master B time trigger is turned on with the next channel A9(D1) pulse (Figure 80).

The POD 64 (store channel instruction) restriction prevents a possible alteration in the channel registers which would cause false error indications while running CE diagnostic programs. B time requests are blocked during channel trap operations to prevent destroying the trap address in the buffer address register.

The B time request occurs at channel 7 time. Channels requesting additional B cycles cause the B time trigger to be turned on again in time to prevent the CPU from regaining cycle control. "Retain priority" conditions (TCH or IA) recognized by the multiplexor look-ahead circuitry bypasses the normal turn-on controls.

The B interrupt trigger (Figure 80) blocks outputs from the master I, E, and II time triggers either during B times or when the master stop trigger is on. Note that the B time trigger is under control of channel clock pulses whereas the B interrupt trigger is under control of CPU clock pulses. The B time trigger turning on causes the B interrupt trigger to turn on at CPU-5 time which is early enough to block the next cyclic output from the master I, E, and II triggers.

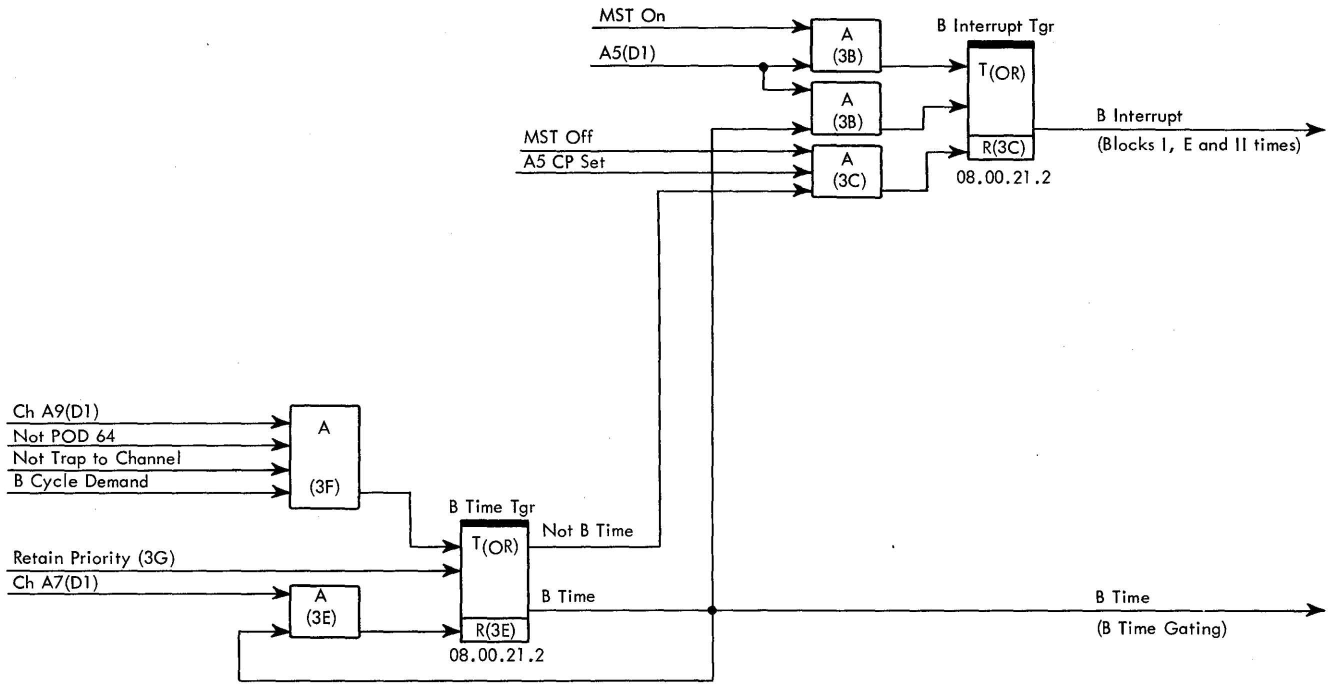


Figure 80. B Time Condensed Logic

Because of timing relationships between the channel and CPU clocks, the B interrupt trigger can be turned on and off for two cases (Figure 81). Case 1 shows the channel 9-CPU-1 timing relationship where the B time trigger turns on at channel 9 time and the B interrupt trigger turns on four clock pulses later at CPU-5 time. Considering only one channel B cycle request, the B interrupt trigger only blocks one CPU I, E or II cycle.

Case 2 shows the channel 9-CPU-5 timing relationship where both the B time and B interrupt triggers turn on during the same clock pulse. In this later case, however, the timings are such that the B interrupt trigger remains on for one extra cycle and two CPU cycles are blocked.

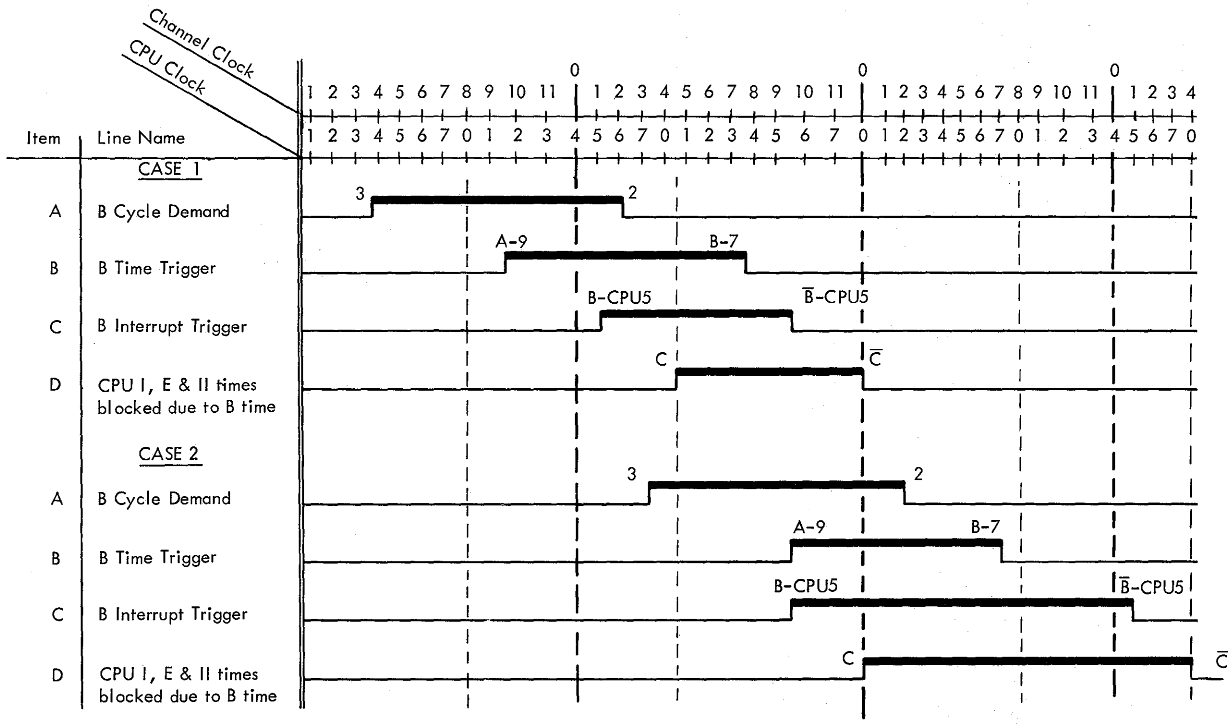


Figure 81. B Time Sequence Chart

B Cycle

The main system objectives associated with B cycles are to:

1. Accept B time requests from the data channel
2. Set BAR to the proper address
3. Select the proper even/odd memory
4. Route data and commands to the proper system areas
5. Test and perform TCH and IA functions
6. Initiate subsequent B cycles where applicable

B cycle requests are initiated by circuitry from within the attached data channels and are based on the immediate channel demands. In most cases, the "B cycle demand" is honored during the cycle following receipt into the CPU. Refer to Figure 82.

A "B cycle demand" arriving during the execution of a channel trap operation or POD 64 (store channel) instruction is delayed until after their completion. These delays are necessary because of conflicts in usage with the buffer address register, and for diagnostic programming reasons, respectively.

"B cycle demands" are initiated at approximately channel 3 time and arrive at the CPU circuitry in time to turn on the B time trigger with a channel A9(D1) pulse. The B time interrupt trigger is turned on at CPU 5 time to block the master I, E, and II time outputs. This blocking remains in effect until CPU A5 time following the turn off of the B time trigger (Figure 81).

Instruction overlap is suspended during all B cycles. Only one memory (even/odd) is addressed, and only one memory select pulse is generated. Therefore, only one data word is placed on the storage bus.

BDW Cycle

The first function performed (Figure 82) is setting the buffer address register (BAR) to the proper memory data reference as specified by the channel address switches (CAS). BAR was reset by a previous channel A11 pulse and is now set at B0(D2) time. For a read BDW cycle, the channel store trigger is turned on to provide the necessary memory "read-out" and "store" controls.

Selection of the proper memory and BAR gating is based on BAR(17). The memory select pulse is initiated at approximately B2 time. For a write BDW cycle, the data word is available on the storage bus at approximately B7 time for sampling into the channel's data register. For a read BDW cycle, the channel has the data word on the storage bus at the proper time to be sampled into the memory buffer register (Figure 35).

BCW Cycle

Consider a normal BCW cycle with no indirect addressing or TCH command (Figure 82). Accepting the "B cycle demand," turning on the B time trigger and selecting memory are as explained previously. The channel determines the memory location by routing an address from its location counter/command counter into the multiplexor buffer address register.

Note that all BCW cycles have the characteristics of a write operation where the data word is taken from memory and sent to the channel.

The data command read from memory is placed on the storage bus and sampled into the channel's operation register, word counter and address register. The channel's location counter (7607) or command counter (7909) is stepped +1 to indicate the next sequential command's location.

If the channel is performing a write operation at the time of the BCW cycle, a BDW cycle is immediately requested. This BDW cycle may not be initiated in time, however, to prevent the CPU from regaining program control. This channel must also seek priority with other channels requesting B cycles on the system.

Indirectly Addressed BCW Cycle

Consider IA commands other than TCH (Figure 82). Accepting the "B cycle demand," turning on the B time trigger, and selecting memory are the same as explained previously. The initial pass through the flow chart finds the IA address control trigger off.

Position 18 of both the even and odd storage bus are OR'ed together, MB(18), as a test for indirect addressing. OR'ing both storage buses is valid because only one of the buses will contain information.

Ignore for the moment the special case of a proceed type command with a word count equal zero. The command word on the storage bus is sampled into the channel's operation register, word counter, and address register. The operation register and word counter contain valid data; the address register contents will be replaced with a new value during the next cycle.

Because of SB(18), an IA address control trigger is turned on which generates and sends a "retain priority" signal to all channels (Figure 82). In the multiplexor, "retain priority" produces an immediate "B cycle demand" and prevents the B time trigger from being reset. In this manner a second B cycle is initiated and the CPU is prevented from regaining control.

The IA address control trigger being on allows an IND 18 trigger to be turned on and its "IND 18" signal sent out on banks 1 and 2 to all data channels.

The address portion of the initial BCW data word represents a new memory reference. Because of this, multiplexor storage bus positions 21-35 are routed directly into the buffer address register (Figure 35). The data word resulting from the second BCW cycle is placed on the storage bus as before. This time, however, the data channel only accepts the address portion (21-35) into its address register; the original operation coding and word count remain unaltered. Following this second

BCW cycle, the channel's location counter/command counter is stepped +1.

The IA address control trigger is turned off at channel A3(D2) time and the IND 18 trigger is turned off at the following channel A10(D2). Timing is such that only one level of indirect addressing is permitted. A bit in SB(18) of the second BCW word is ignored and the IND 18 trigger is not turned on again.

A special case of a proceed type command with a word count equal zero was ignored in the previous discussion. An IOCP, IORP, or IOSP command with a word count equal zero must be bypassed. Indirect addressing, therefore, performs no logic and is ignored (Figure 82). The command is routed to the channel and set into the operation register, word counter and address register as usual. Channel circuitry recognizes the zero word count condition and initiates another "B cycle demand." Multiplexor lookahead circuitry does not retain priority in this case.

TCH Command

Consider a TCH command without indirect addressing. The TCH acts as a channel transfer instruction to alter the sequence of I/O commands being executed. The transfer is accomplished by altering the value in the channel's location counter (7607) or command counter (7909). The channel determines the memory location by routing an address from its location counter/command counter into the multiplexor's buffer address register (BAR). Accepting the initial "B cycle demand," turning on the B time trigger and selecting memory are as explained previously.

The TCH command is detected (Figure 82) by testing the storage bus for not S, not 1, 2. The 7607 data channel's operation register, word counter and address register are set as usual; the 7909 sets only the address register and a TCH trigger. A TCH address control trigger is turned on in the multiplexor and a second B cycle is immediately initiated by "retain priority."

The address portion of the TCH command contains the transfer-to address. This address is immediately set into the buffer address register for the next memory reference. The address is also set into the channel location counter/command counter to indicate the new I/O command sequence.

The second BCW cycle places the new I/O command on the storage bus and the operation continues as explained previously. This new command (Figure 82) may be a TCH command, another I/O command, or an indirectly addressed I/O command.

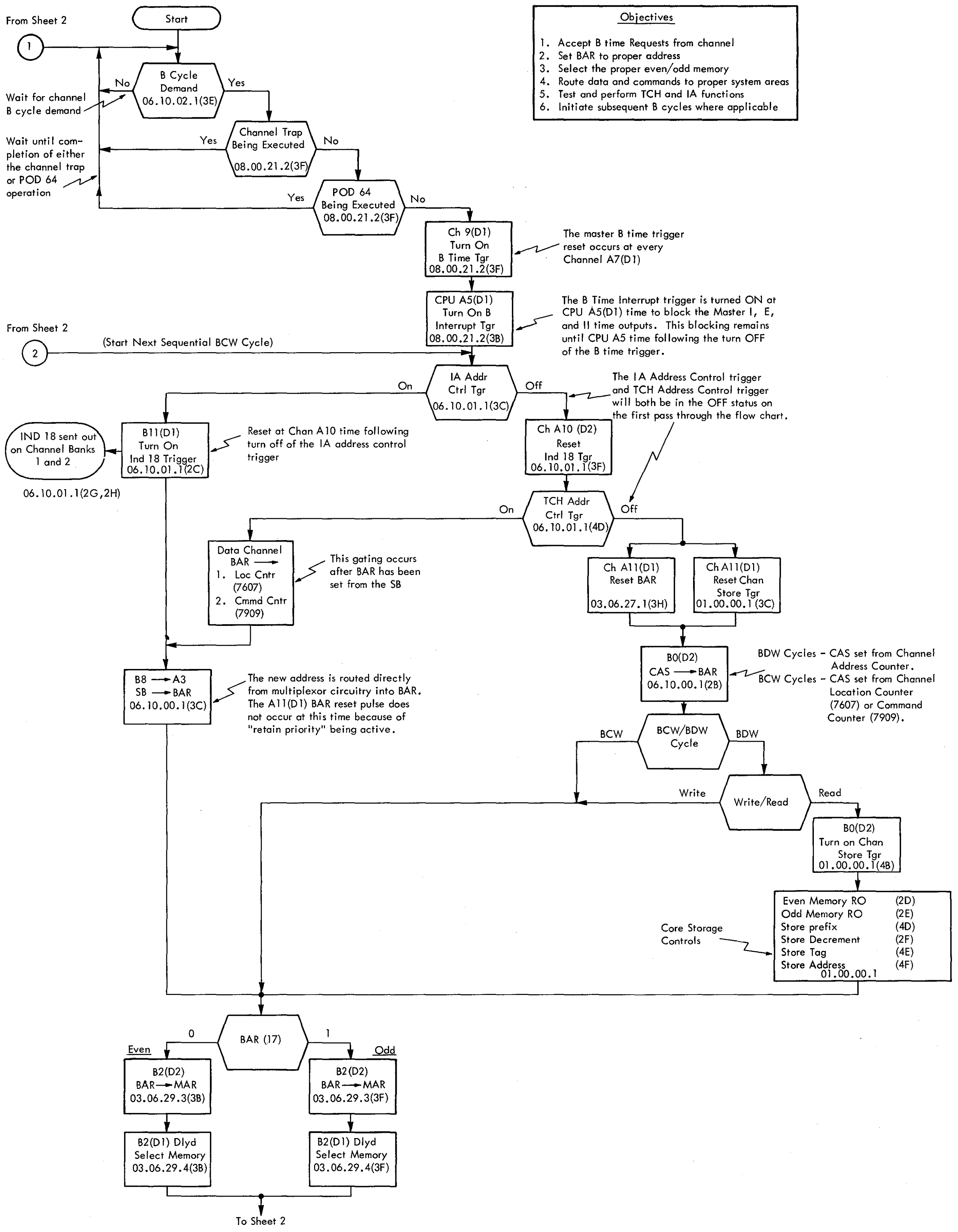
Indirectly Addressed TCH Command

An IA TCH command operates similar in most respects to the normal TCH (Figure 82). During the first BCW cycle, the storage bus indicates not S, not 1, 2, and 18. As a result, both the "TCH address control" and "IA address control" triggers are turned on. "Retain priority" circuitry in the multiplexor immediately initiates a second BCW cycle and turns on the IND 18 trigger. The new (IA) memory reference is routed from SB(21-35) and set into the buffer address register. Note that "IND 18" control circuitry in the channel prevents setting of the location counter/command counter during the first BCW cycle.

At channel A3 time, the IA address control trigger is turned off but the reset to the TCH address control trigger is blocked. "Retain priority" is maintained because of the TCH address control trigger and the multiplexor immediately requests the third BCW cycle.

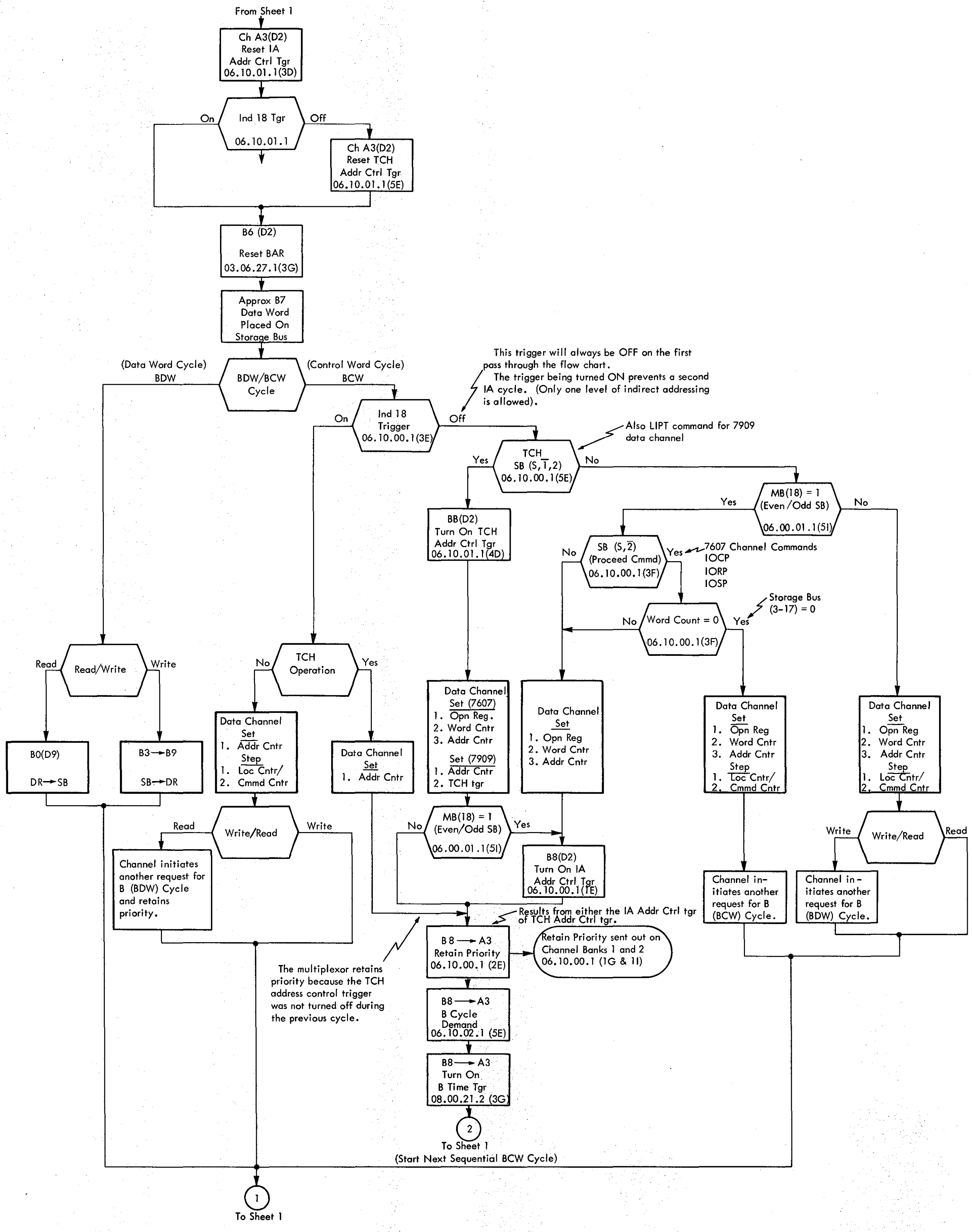
During the second BCW cycle, the address portion of this IA data word contains the actual TCH transfer-to address. This address is immediately set into the buffer address register for the third memory reference, and also set into the channel location counter/command counter to indicate the new I/O command sequence. The IND 18 trigger is turned off during the second BCW cycle which, in turn, allows the address control trigger to be turned off at channel A3 time.

The third BCW cycle places the new I/O command on the storage bus and the operation continues as explained previously.



From Sheet 2
1
Start
Wait for channel B cycle demand
No
B Cycle Demand 06.10.02.1(3E)
Yes
Wait until completion of either the channel trap or POD 64 operation
Yes
Channel Trap Being Executed 08.00.21.2(3F)
No
Yes
POD 64 Being Executed 08.00.21.2(3F)
No
Objectives
1. Accept B time Requests from channel
2. Set BAR to proper address
3. Select the proper even/odd memory
4. Route data and commands to proper system areas
5. Test and perform TCH and IA functions
6. Initiate subsequent B cycles where applicable
Ch 9(D1) Turn On B Time Tgr 08.00.21.2(3F)
The master B time trigger reset occurs at every Channel A7(D1)
CPU A5(D1) Turn On B Interrupt Tgr 08.00.21.2(3B)
The B Time Interrupt trigger is turned ON at CPU A5(D1) time to block the Master I, E, and II time outputs. This blocking remains until CPU A5 time following the turn OFF of the B time trigger.
From Sheet 2
2
(Start Next Sequential BCW Cycle)
On
IA Addr Ctrl Tgr 06.10.01.1(3C)
Off
The IA Address Control trigger and TCH Address Control trigger will both be in the OFF status on the first pass through the flow chart.
IND 18 sent out on Channel Banks 1 and 2
06.10.01.1(2G,2H)
B11(D1) Turn On Ind 18 Trigger 06.10.01.1(2C)
Reset at Chan A10 time following turn off of the IA address control trigger
Ch A10 (D2) Reset Ind 18 Tgr 06.10.01.1(3F)
On
TCH Addr Ctrl Tgr 06.10.01.1(4D)
Off
Data Channel BAR → 1. Loc Cntr (7607) 2. Cmmd Cntr (7909)
This gating occurs after BAR has been set from the SB
Ch A11(D1) Reset BAR 03.06.27.1(3H)
Ch A11(D1) Reset Chan Store Tgr 01.00.00.1(3C)
B0(D2) CAS → BAR 06.10.00.1(2B)
BDW Cycles - CAS set from Channel Address Counter.
BCW Cycles - CAS set from Channel Location Counter (7607) or Command Counter (7909).
B8 → A3 SB → BAR 06.10.00.1(3C)
The new address is routed directly from multiplexor circuitry into BAR. The A11(D1) BAR reset pulse does not occur at this time because of "retain priority" being active.
BCW
BCW/BDW Cycle
BDW
Write
Write/Read
Read
B0(D2) Turn on Chan Store Tgr 01.00.00.1(4B)
Core Storage Controls
Even Memory RO (2D)
Odd Memory RO (2E)
Store prefix (4D)
Store Decrement (2F)
Store Tag (4E)
Store Address (4F)
01.00.00.1
0
Even
BAR (17)
1
Odd
B2(D2) BAR → MAR 03.06.29.3(3B)
B2(D2) BAR → MAR 03.06.29.3(3F)
B2(D1) Dlyd Select Memory 03.06.29.4(3B)
B2(D1) Dlyd Select Memory 03.06.29.4(3F)
To Sheet 2

Figure 82. B Time Flow Chart (Sheet 1 of 2)



From Sheet 1
Ch A3(D2) Reset IA Addr Ctrl Tgr 06.10.01.1(3D)
On
Ind 18 Tgr 06.10.01.1
Off
Ch A3(D2) Reset TCH Addr Ctrl Tgr 06.10.01.1(5E)
B6 (D2) Reset BAR 03.06.27.1(3G)
Approx B7 Data Word Placed On Storage Bus
(Data Word Cycle) BDW
BDW/BCW Cycle
(Control Word Cycle) BCW
This trigger will always be OFF on the first pass through the flow chart. The trigger being turned ON prevents a second IA cycle. (Only one level of indirect addressing is allowed).
On
Ind 18 Trigger 06.10.00.1(3E)
Off
Also LIPT command for 7909 data channel
Yes
TCH SB (S,1,2) 06.10.00.1(5E)
No
Yes
MB(18) = 1 (Even/Odd SB) 06.00.01.1(5I)
No
BB(D2) Turn On TCH Addr Ctrl Tgr 06.10.01.1(4D)
No
SB (S,2) (Proceed Cmmd) 06.10.00.1(3F)
Yes
7607 Channel Commands IOCP IORP IOSP
Storage Bus (3-17) = 0
No
Word Count = 0 06.10.00.1(3F)
Yes
Read
Read/Write
Write
No
TCH Operation
Yes
B0(D9) DR→SB
B3→B9 SB→DR
Data Channel Set 1. Addr Cntr Step 1. Loc Cntr/ 2. Cmmd Cntr
Data Channel Set 1. Addr Cntr
Data Channel Set (7607) 1. Opn Reg. 2. Word Cntr 3. Addr Cntr Set (7909) 1. Addr Cntr 2. TCH tgr
Data Channel Set 1. Opn Reg 2. Word Cntr 3. Addr Cntr
Data Channel Set 1. Opn Reg 2. Word Cntr 3. Addr Cntr Step 1. Loc Cntr/ 2. Cmmd Cntr
Data Channel Set 1. Opn Reg 2. Word Cntr 3. Addr Cntr Step 1. Loc Cntr/ 2. Cmmd Cntr
Read
Write/Read
Write
No
MB(18) = 1 (Even/Odd SB) 06.00.01.1(5I)
Yes
Write
Write/Read
Read
Channel initiates another request for B (BDW) Cycle and retains priority.
B8(D2) Turn On IA Addr Ctrl Tgr 06.10.00.1(1E)
Channel initiates another request for B (BCW) Cycle.
Channel initiates another request for B (BDW) Cycle.
Results from either the IA Addr Ctrl tgr of TCH Addr Ctrl tgr.
B8 → A3 Retain Priority 06.10.00.1 (2E)
Retain Priority sent out on Channel Banks 1 and 2 06.10.00.1 (1G & 1I)
The multiplexor retains priority because the TCH address control trigger was not turned off during the previous cycle.
B8 → A3 B Cycle Demand 06.10.02.1 (5E)
B8 → A3 Turn On B Time Tgr 08.00.21.2 (3G)
2
To Sheet 1 (Start Next Sequential BCW Cycle)
1
To Sheet 1

Figure 82. B Time Flow Chart (Sheet 2 of 2)

Channel Cycle Times

Three channel cycle time triggers (Figure 83) have been added in CPU-2 to supply 2.1 microsecond cycle times for data channel usage. While these three cycle time triggers are controlling the channel, the CPU waits in CPU L time until the channel operation is completed. The end of the channel operation is indicated by the "MF go" trigger which turns on the end-op trigger and sends the CPU into its next cycle time.

Channel I Time

The channel I time trigger (Figure 83) is not the first channel cycle trigger to be turned on as might be thought. Instead, it is the last channel cycle time available (I time next) and performs mostly housekeeping functions (I time resets) in the channel.

Channel E Time

The channel E time trigger (Figure 83) provides a 2.1 microsecond cycle for channel operations requiring a reference to core storage. These operations would include: enable, store channel, POD 54(RCH/LCH/RSC/STC), and channel traps.

Circuit controls and timings are such that the channel E0 time is aligned with the CPU 4 time. This alignment is necessary for POD 64 and enable instructions.

The channel E time indicates the end of a channel operation. Because of this, the MF go trigger is turned on late in the channel E cycle to force an end-op condition and allow the CPU program to continue with the next instruction.

Channel L Time

The channel L time trigger (Figure 83) provides 2.1 microsecond cycles necessary for channel operations. These operations include all of the I/O select, sense, and test instructions. POD 54 instructions (RCH/LCH/RSC/STC) also require L cycles prior to the time that the channel may signal a "proceed to E."

Note that the channel L time trigger is turned on at channel 11 time which may be aligned with either a CPU 3 or 7 pulse. Synchronism is not performed as with the preceding channel E time trigger.

Channel—CPU Cycle Time Controls

Two triggers are used to control cycle times. The channel "L-E end" trigger controls channel cycle times, and the "MF go" trigger controls the CPU time.

The channel L-E end trigger coming on signals the end of the channel operation. In some cases, the channel I time trigger is then turned on to accomplish housekeeping resets to the channel.

The MF go trigger is normally on. At the start of the channel operation, the MF go trigger is turned off which, in turn, turns the "L end-op sync" trigger on (Figure 83). When the channel operation is completed, the MF go trigger is turned on again and the output of the "L end-op sync" trigger is gated to produce an "E or L end-op." This latter signal turns on the end-op trigger and allows the CPU to continue with the next instruction.

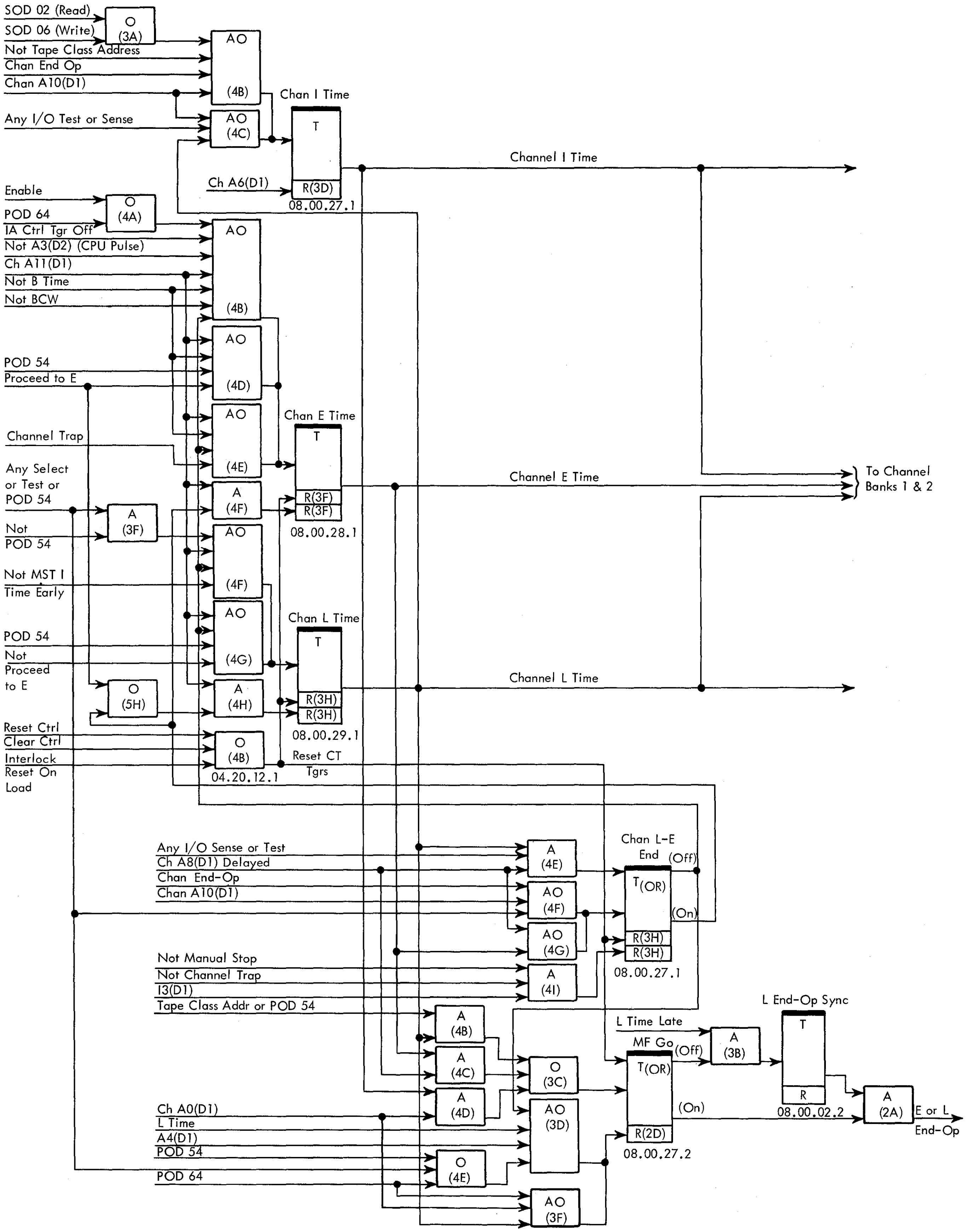


Figure 83. Channel Cycle Time Logic

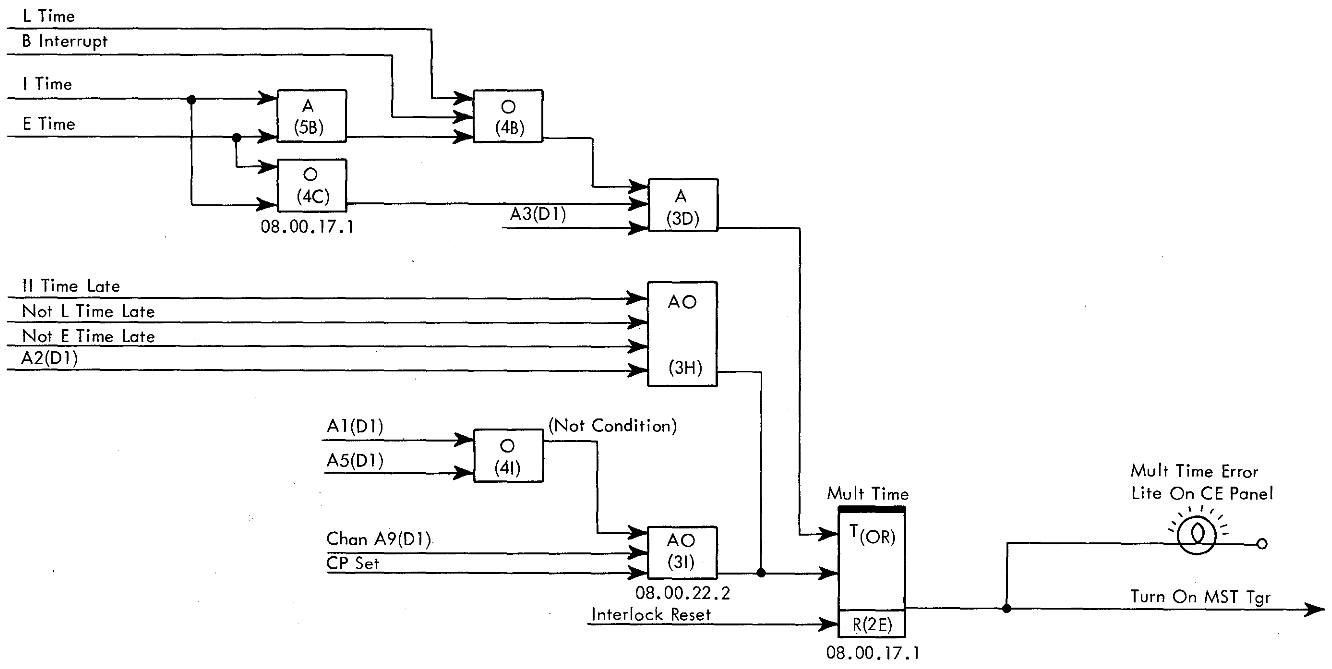


Figure 84. Multiple Time Check Circuitry

Multiple Cycle Time Error Detection

Eight different cycle times and two separate clocks exist in the 7094 II. Only certain cycles can occur simultaneously without causing machine malfunctions. Illegal cycle combinations turn on a multiple time trigger which immediately stops the computer and lights a mult time light on the console CE panel.

Test circuitry (Figure 84) monitors the various CPU cycle times once every cycle. The top group of circuits checks for the following illegal combinations: I and E; L and I; L and E; B and I; and B and E.

Note that B and L times are allowed to occur simultaneously.

II time is allowed to occur simultaneously with an E or L cycle. II time occurring without either an E or L time is detected at AO_{3H}.

Clock alignment is checked at AO_{3I}. Every channel A9(D1) pulse should occur simultaneously with either a CPU A1(D1) or A5(D1) pulse. An absence of these conditions turns on the multiple time trigger.

There is no check on multiple channel cycle times.

Waveforms and Variable Delay Adjustments

The oscilloscope being used must contain probes of the same length. Test the oscilloscope by placing both probes on the same point and noting if there is any difference in the time relationship between the A and B sweeps.

The following adjustments should be made in the order presented because in some cases a later adjustment is based on the proper earlier setting. It must be remembered, however, that many of these settings are nominal (starting point) settings and may vary slightly with later adjustments or from system to system.

Initial Delay Settings

All variable delays shall be set to the nominal values specified on Systems 00.92.01.0 (sheets 1 and 2) before proceeding with the following adjustments. Timings for F lines should be measured at the 1.5 volt level.

Odd and Even Clock Drive Pulses

Odd and even clock drive pulses should meet all requirements of the waveform shown in Figures 85 and 86 when observed at 03A4C15F (Systems 08.00.44.1, 1F).

CP Set Pulses

CP set pulses should meet all requirements of the waveform shown in Figure 87 when observed at 01A1C12C (Systems 02.15.61.1,4D) and 01B1C20B (Systems 02.15.61.2, 5E).

CPU-1 CP Set Pulse Width Adjustment

Connect the scope probe to the test points indicated below and adjust the CP set pulse width by means of the corresponding V7 delay card. The first two adjustments determine the CP set pulse widths at the A and B gates, respectively. The last width adjustment is for setting the FACT triggers during arithmetic operations and is more critical than most other set pulses.

TEST POINT	LEVEL	WIDTH	DLY LOCATION	SYSTEMS
01A1E13E	−F	50 ± 3 ns	01A1E14 (A-H)	02.15.61.1, 4B
01B1E24B	−F	60 ± 3 ns	01B1E25 (B-G)	02.15.61.2, 4A
01B1F21C	+F	40 ± 3 ns	01B1E12 (D-E)	02.15.61.2, 2B

CP Set Pulse—Clock Pulse Alignment

In this section the multiplexor CP set pulse is adjusted with respect to the A7(D1) CPU-1 pulse. This A7(D1) pulse is already properly aligned to the CPU-2 pulse because of the VB delay 01D2E21 (Systems 02.15.42.1, 3H).

1. Synchronize the scope on "−F A6(D2)" at 01A1-B14B (Systems 02.15.71.1, 4H).
2. Connect scope probe A to "−F A7(D1)" at 01B1-C05E (Systems 02.15.70.8, 4I).
3. Connect scope probe B to "+F CP set" at 01B1-E24C (Systems 02.15.61.2, 5A).
4. Adjust the variable delay control at 03B3D03 (Systems 08.00.47.1, 4C) so that the fall of the set pulse occurs 10 nanoseconds before the fall of the "−F A7(D1)" pulse. This 10 nanosecond timing is measured at the point where both pulses cross the F reference level (Figure 88).

The optimum multiplexor delay line operation point is determined by running 9M81. The setting is the midpoint of the error free operating range of 9M81 running at normal voltage as the delay line is varied. After determining the optimum operating point and without disturbing the delay line adjustment, remove and reinstall the delay line knob at 03B3D03 to read zero.

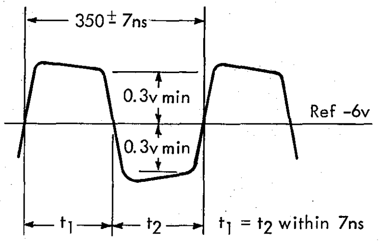


Figure 85. Clock Drive Pulse

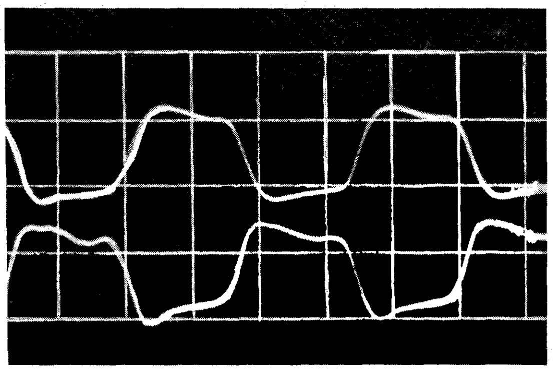


Figure 86. Even and Odd Clock Drive Pulses

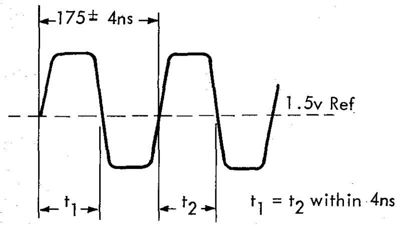


Figure 87. CP Set Pulse

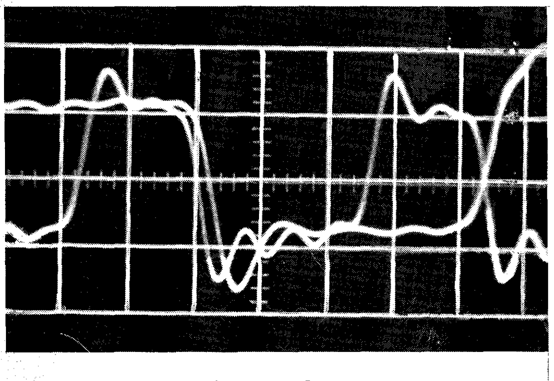


Figure 88. CP Set Pulse—Clock Pulse Alignment

CPU and Channel Memory Select Alignment

Memory selection occurs with different clock pulses (and from different clocks) when initiated during either CPU or channel operations. These two memory selects must be aligned to insure proper operation when memory is being alternately used by both the CPU and channel.

1. Synchronize the scope on "+F A6(D1)" at 01A1-B05G (Systems 02.15.70.7, 4I).
2. Connect scope probe A to "+F A2(D1)" at 01A4-J24E (Systems 03.06.29.4, 3B).
3. Connect scope probe B to "+F A7(D1)" at 01A4-J24G (Systems 03.06.29.4, 3A).
4. Adjust the V7 delay card at 03A4J17 (A-H) (Systems 08.00.40.1, 1G) so that the rise of the channel A2(D1) dlyd pulse crosses the reference line at the same time as the A7(D1)S pulse, ±3 nanoseconds as shown in Figure 89.

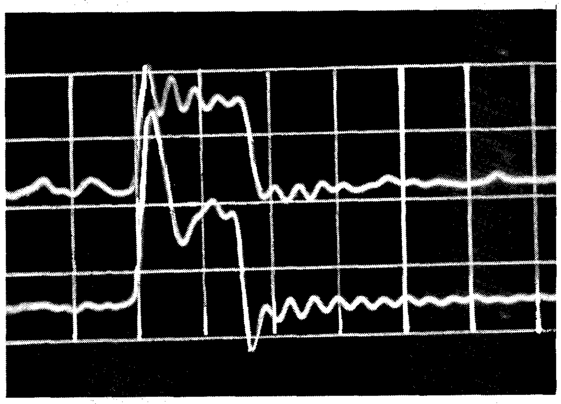


Figure 89. Memory Select Pulse Alignment

Memory Select and MAR Bus Alignment

This section checks that the MAR bus pulses precede the MAR set pulse by at least 20 nanoseconds. This check is made at both the even and odd memories. Figure 90 shows a representative pulse. Note that all of the test points are at the 7302-3 panels.

1. Execute either a TRA (+0020) 77776, 0 or TRA 77777, 0 as indicated below in continuous enter instruction.
2. Synchronize the scope on "−F E time" at 01A2E12P (Systems 08.00.19.3, 3A). This test point is in the CPU.
3. Connect scope probe A to the MAR set point indicated below.
4. Connect scope probe B to the MAR bus points indicated below.
5. All of the MAR bus lines should precede their respective MAR set pulses by at least 20 nanoseconds.

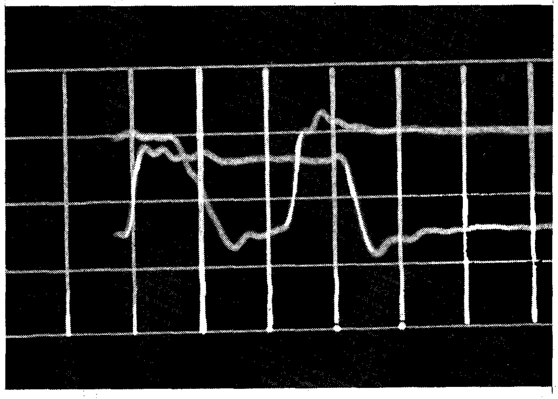


Figure 90. Memory Select—MAR Bus Alignment

CONDITIONS	EVEN MEMORY	ODD MEMORY
Instruction	TRA 77776, 0	TRA 77777, 0
MAR set	01B3C16D (01.11.01.1 —3B) Probe A	01C3C16D (01.41.01.1 —3B) Probe A
MAR 4	01B3C16E (01.11.01.1 —3B) Probe B	01C3C16E (01.41.01.1 —3B) Probe B
MAR 8	01B3C18E (01.11.01.1 —3F) Probe B	01C3C18E (01.41.01.1 —3F) Probe B
MAR 12	01B3C20V (01.11.02.1 —3C) Probe B	01C3C20V (01.41.02.1 —3C) Probe B
MAR 16	01B3C22V (01.11.02.1 —3G) Probe B	01C3C22V (01.41.02.1 —3G) Probe B

Appendix A: Octal-Decimal Integer Conversion Table

0000 to 0777 (Octal)	0000 to 0511 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

Octal-Decimal Integer Conversion Table (Continued)

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000 to 2777 (Octal)	1024 to 1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

3000 to 3777 (Octal)	1536 to 2047 (Decimal)

Octal-Decimal Integer Conversion Table (Continued)

4000 to 4777 (Octal)	2048 to 2559 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 to 5777 (Octal)	2560 to 3071 (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

Octal-Decimal Integer Conversion Table (Continued)

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000 to 6777 (Octal)	3072 to 3583 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000 to 7777 (Octal)	3584 to 4095 (Decimal)

Appendix B: Octal-Decimal Fraction Conversion Table

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.426781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

Octal-Decimal Fraction Conversion Table (Continued)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

Octal-Decimal Fraction Conversion Table (Continued)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

Appendix C: Table of Powers of Two

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

COMMENT SHEET

IBM 7094 II DATA PROCESSING SYSTEM – VOLUME I

CUSTOMER ENGINEERING INSTRUCTION–MAINTENANCE. FORM 223-2721-0

FROM

NAME ______________________________

OFFICE NO. ______________________________

FOLD FOLD

CHECK ONE OF THE COMMENTS AND EXPLAIN IN THE SPACE PROVIDED

☐ SUGGESTED ADDITION (PAGE , TIMING CHART, DRAWING, PROCEDURE, ETC.)

☐ SUGGESTED DELETION (PAGE)

☐ ERROR (PAGE)

EXPLANATION

CUT ALONG LINE

FOLD FOLD

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
FOLD ON TWO LINES, STAPLE, AND MAIL

STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 81

POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N.Y. 12602

ATTN: CE MANUALS, DEPARTMENT B96

CUT ALONG LINE

LD FOLD

10/64:1.5M-VO-112

STAPLE STAPLE

223-2721-0

IBM 7094 II Printed in U.S.A. 223-2721-0

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601