NASA CR-73262
Available to the Public

RESEARCH REPORT

MULTIPAC, A MULTIPLE POOL PROCESSOR AND COMPUTER FOR A SPACECRAFT CENTRAL DATA SYSTEM

By E. Cohler
T. Baker
G. Cummings
R. South

MARCH - 1969
[redacted]

Prepared under Contract No. NAS2-3255 by
APPLIED RESEARCH LABORATORY
SYLVANIA ELECTRONIC SYSTEMS
An Operating Group of Sylvania Electric Products, Inc.
40 Sylvan Road, Waltham, Massachusetts 02154

for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
AMES RESEARCH CENTER
MOFFETT FIELD, CALIFORNIA 94035

NASA CR-73262
Available to the Public

RESEARCH REPORT

MULTIPAC, A MULTIPLE POOL PROCESSOR AND COMPUTER FOR A SPACECRAFT CENTRAL DATA SYSTEM

By E. Cohler
T. Baker
G. Cummings
R. South

MARCH- 1969

[redacted]

Prepared under Contract No. NAS2-3255 by
APPLIED RESEARCH LABORATORY
SYLVANIA ELECTRONIC SYSTEMS
An Operating Group of Sylvania Electric Products, Inc.
40 Sylvan Road, Waltham, Massachusetts 02154

for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
AMES RESEARCH CENTER
MOFFETT FIELD, CALIFORNIA 94035

PRECEDING PAGE BLANK NOT FILMED.

TABLE OF CONTENTS

Section — Page

SUMMARY....1
1.0 INTRODUCTION....3
2.0 THE MULTIPAC CONCEPT AND ITS EVOLUTION....6
3.0 SYSTEM DESCRIPTION....12
- 3.1 Data Flow....12
  - 3.1.1 Address switching....15
  - 3.1.2 The I/O system....15
  - 3.1.3 Telemetry Output....22
  - 3.1.4 Command inputs....24
- 3.2 Control Functions....24
  - 3.2.1 Program selection....24
  - 3.2.2 Instruction list....24
  - 3.2.3 Command link override....31
  - 3.2.4 Interrupts....32

4.0 DESCRIPTION OF THE MULTIPAC INTEGRATED CIRCUIT LOGIC DESIGN....34
- 4.1 Circuitry....34
  - 4.1.1 Logic circuits....34
  - 4.1.2 Memory circuits....35
  - 4.1.3 Special circuits....36
- 4.2 Parts Count....36
- 4.3 Weight....36
- 4.4 Volume....37
- 4.5 Power....38
- 4.6 Speed....38

5.0 RELIABILITY....44
- 5.1 Individual Module Reliability....44
- 5.2 Reliability of Groups of Interchangeable Modules....51
- 5.3 Reliability of the I/O, Telemetry and Command Interfaces....54
- 5.4 System Reliability....55

6.0 PROGRAMMING....60
- 6.1 Timing....60
- 6.2 Multiplication....66
- 6.3 Subroutine Call....68
- 6.4 Program Modification....68
- 6.5 Addressable Registers....68
- 6.6 Ground Software....69

7.0 REPROGRAMMING AROUND FAILURES....70
- 7.1 Complete Failure of a Register....70
- 7.2 Complete Failure of a Logic Unit....71
- 7.3 Memory Failures....72
  - 7.3.1 Complete failure....72
  - 7.3.2 Partial failures....73

TABLE OF CONTENTS--- Continued

Section Page

7.4 Reprogramming Methods....74
7.4.1 Diagnostic tests....74
7.4.2 Timing....76
8.0 CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE IMPLEMENTATION....77
8.1 Conclusions....77
8.1.1 Speed and power....77
8.1.2 Weight and volume....77
8.1.3 Reliability....77
8.1.4 Programming....78
8.2 Recommended Changes....78
8.2.1 Use of macroinstructions....78
8.2.2 Elimination of base registers and two-level switching....80
8.2.3 System operation with recommended changes....81

Appendix

A DESCRIPTION OF LOGIC BY MODULE....85
B SYSTEM INTERCONNECTION OF MODULES....123
C SPECIFICATIONS FOR INTEGRATED CIRCUITS....157
REFERENCES....181

LIST OF ILLUSTRATIONS

Figure		Page
1	Simplified Block Diagram of MULTIPAC	4
2	Original MULTIPAC Concept	7
3	Data Flow in Original MULTIPAC Concept	8
4	MULTIPAC Block Diagram	13
5	MULTIPAC Address Allocations	16
6	MULTIPAC Telemetry and Command Interface	23
7	Instruction Format	29
8	Critical Propagation Paths	39
9	Six-Month System Reliability Models	45
10	Twelve-Month System Reliability Models	46
11	Twenty-Four Month System Reliability Models	47
12	Six-Month System Reliability	48
13	Twelve-Month System Reliability	49
14	Twenty-Four Month System Reliability	50
15	MULTIPAC Timing Diagram	86
16	Instruction Format and Karnaugh Mapping of Instruction Coding	87

LIST OF TABLES

Table		Page
1	SCIENCE INTERFACE LINES	17
2	ENGINEERING INTERFACE LINES	18
3	I/O CHANNELS REQUIRED	19
4	MULTIPAC INSTRUCTION LIST	25
5	MAXIMUM CIRCUIT PROPAGATION DELAYS	41
6	WORST CASE PROPAGATION DELAYS FOR CRITICAL PATHS	42
7	MODULE RELIABILITY	52
8	RELIABILITY FIGURES FOR GROUPS OF INTERCHANGEABLE MODULES	53
9	CRITERIA FOR SUCCESS OF THE I/O, TELEMETRY, AND COMMAND INTERFACES	56
10	MONTE CARLO RELIABILITY OF I/O, TELEMETRY, AND COMMAND INTERFACES (6 MONTHS)	57
11	MONTE CARLO RELIABILITY OF I/O, TELEMETRY, AND COMMAND INTERFACES (12 MONTHS)	58
12	MONTE CARLO RELIABILITY OF I/O, TELEMETRY, AND COMMAND INTERFACES (24 MONTHS)	59
13	A/D CONVERSION ROUTINE	61
14	INPUTTING ROUTINE	62
15	OUTPUTTING ROUTINE	64
16	LIST OF MACROINSTRUCTIONS	83
17	EQUATIONS OF LOGIC UNIT OUTPUT	90
18	TRUTH TABLE OF LOGIC UNIT CONTROL SIGNALS	92

LIST OF TABLES.-- Continued

Table		Page
19	JUMP AND INTERRUPT SEQUENCES	96
20	MULTIPAC MODULE INTERCONNECTIONS	124

LIST OF ABBkEVIATIONS

ANAMP	Analog amplifier
ANSW	Analog switch
BMIC	Bipolar-to-MOS interface circuit
BRA, BRB	Base register A and B
CDS	Central data system
CIR	Command input
CMD	Command module
DTL	Diode-transistor logic
IEN	Interrupt enable
INC	Incrementing carry (flip-flop)
LIR	Logic instruction register
LSI	Large-scale integration
LSR	Logic shifting register
LU (or LUR)	Logic unit (register)
MA	Memory address (register)
MBIC	MOS-to-bipolar interface circuit
MD	Memory data (register)
MOS	Metal oxide semiconductor
MR	Module reliability
OPC	Operation code
PC	Program counter
PSR	Program switch register
RD	Read (flip-flop)
SC	Shift clock
TCLK	Transfer clock
TFF	Test flip-flop
TM	Telemetry module
TOF	Telemetry output flip-flop

LIST OF ABBREVIATIONS.-- Continued

TOR	Telemetry output register
TTL	Triode-transistor logic
WR	Write (flip-flop)
WS	Word strobe

RESEARCH REPORT

MULTIPAC, A MULTIPLE POOL PROCESSOR AND COMPUTER FOR A SPACECRAFT CENTRAL DATA SYSTEM

By E. Cohler
T. Baker
G. Cummings
R. South

SUMMARY

A computer has been designed for use as a central data system on deep space probes. This computer has the unusual characteristic that it may be repaired during flight through the command and telemetry link by reprogramming around the failed unit. This computer organization uses pools of identical modules which the program organizes into one or more computers. In the event of a failure, new programs are entered which reorganize the central data system. The only effect of such reorganization is to reduce the total processing capability aboard the spacecraft, and therefore some low priority process may have to be eliminated.

This report contains a detailed description of this design including logic diagrams of all modules and interconnection lists for these modules. Further implementation of this design is now proceeding using large-scale integrated circuits (LSI). A section containing recommendations for improving the system organization when reimplementing this design is included in this report.

PRECEDING PAGE BLANK NOT FILMED.

1.0 INTRODUCTION

MULTIPAC is a spacecraft central processor whose modular organization permits reprogramming around failed modules. Machine reorganization may be accomplished by program changes to utilize surviving modules optimally, thus effecting a gradual degradation of processing capability as additional modules fail in the course of a long mission. Since there is an inverse relationship between the processing capacity of such reorganizations and their probability of survival, the probability that at least some minimal mode of operating the spacecraft can be sustained throughout very long missions is quite high.

The present report describes a conventional integrated circuit implementation of the system which indicates its feasibility and provides a base line for the evolution of the final LSI design.

The general block diagram of the present MULTIPAC system is shown in Figure 1. (This diagram is a simplification showing the data flow paths only.) The system consists basically of from 2 to 24 of each of seven different types of modules which are interconnected by data busses. The module types are:

(1) Logic Unit
(2) Address Switch
(3) Register
(4) D/A Register
(5) Memory
(6) Telemetry and Command Interface Unit
(7) Timing Generator

The control-arithmetic-logical module is called a logic unit. Each logic unit has two input and one output data busses and the data flow, which is a 13-bit serial transfer, is controlled by them. Each of the logic units can operate on the output from two other modules through its two input data busses and return the result to one of the two through its output data bus. The logic unit selects the connections on its own input and output busses.

The second type of module, the address switch, is an extension of the logic unit input and output bus structure and further expands their address switching by providing a second level of selection. These are not shown separately in Figure 1.

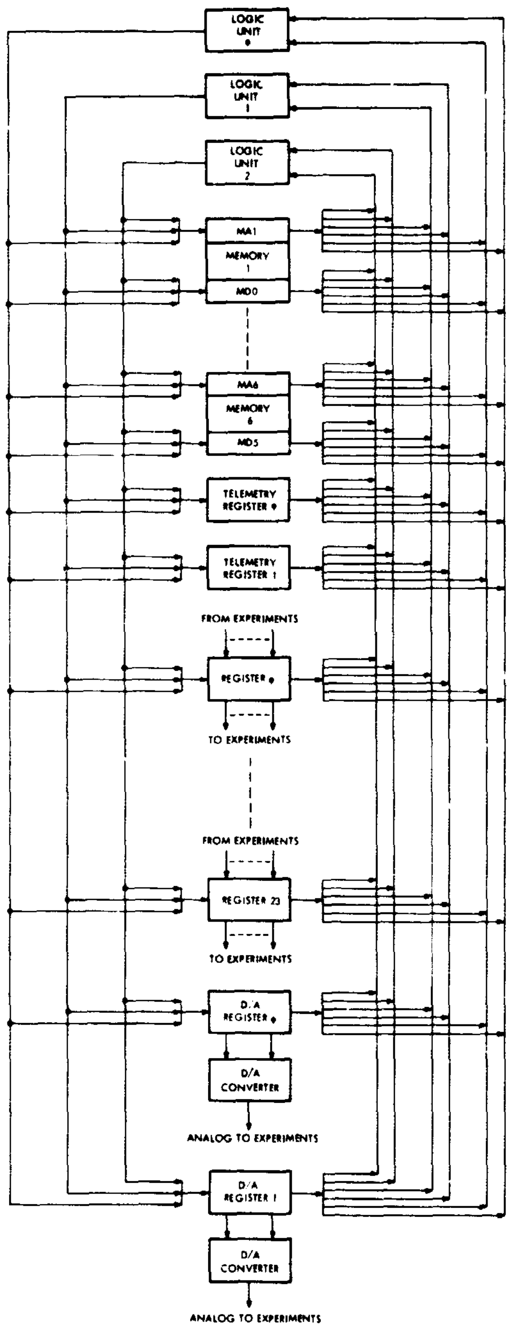


Figure 1. Simplified Block Diagram of MULTIPAC

The third type of module is the register. This module acts as a general purpose quick access storage register and also provides I/O interfaces to the rest of the spacecraft and the experiments. Each register may have 13 inputs and 12 outputs attached to it, and if an input or output command is given to this register, it may either take in 13 bits or put out 12 bits to the experiments. There are 24 of these registers, allowing up to 312 inputs and 288 outputs to the experiments and engineering subsystems.

In addition to the above 24 register modules, there are two more register modules with D/A converters attached to their outputs. The analog output is distributed through isolating amplifiers to the experiments to be used in conjunction with A/D conversion.

The fifth type of module is a 2048-word, 13-bit memory. There are six of these which are used for program store as well as data store. When used for program store, their address registers increment each machine cycle, thus serving as program counters.

The sixth type of module is for the telemetry and command interface. It is basically two register modules interfaced to the telemetry transmitter and command receiver with the additional capability of obtaining control over logic units via the command link. There are two of these modules for redundancy purposes.

The seventh type of module is a timing generator which supplies synchronizing clock pulses to the rest of the MULTIPAC system. It is not shown in Figure 1.

The three logic units, in conjunction with the six memories, provide for three possible processors. These would divide the processing load, such as one handling inputting and outputting of data to and from the experiments, one handling data reduction routines, and one doing the formatting and outputting to the telemetry. Data can be exchanged between such processors by simply exchanging their data memories, each processor updating certain tables. Communication between the processors is necessary to assure that the timing of such transferring is mutually agreeable. Under the graceful degradation concept, these three independent processors would have to absorb one another's tasks if equipment failures caused a reconfiguration to two processors or one. The processing load and telemetry rate would have to be adjusted as necessary to keep each such combined processor working within real time.

2.0 THE MULTIPAC CONCEPT AND ITS EVOLUTION

The MULTIPAC system was originally proposed as a computer organization which would make the versatility of a programable central processor available on long space flights without making the spacecraft dependent on the poor reliability of a conventional computer in which any failure normally makes the entire system useless. The solution then envisioned, shown in Figure 2, was a very simple processor organized from modules selected from pools of three basic module types (logic units, memories, and general-purpose registers) which would be assigned to their functional roles by software methods. Should a failure occur, the faulty module could be replaced by assigning another to fulfill its function. Moreover, spares would not have to be assigned as such but could be used insofar as possible to enlarge the initial capacities of the processor. Failures would simply cause a gradual degradation of processing capability so long as sufficient modules remained from which to construct the minimal processor.

As originally envisioned, a processor could be constructed from three logic units, two memories and several registers, plus a multiplexer to provide I/O to the spacecraft experiments and modules to interface with the command and telemetry links. The logic units perform all transfers in the machine as dictated by their individual instruction registers (LIR's). The memories automatically output to their data registers (MD) the contents of the location specified by their address registers (MA) or write in that location any data word transferred into their MD registers.

The data flow, which is programmed, is typically as shown in Figure 3. LIR1, the instruction register of Logic Unit 1, is initially loaded with an instruction causing the contents of a register (R_{pc}), used as the program counter, to be incremented and passed to the address register (MA1) of the memory containing the program.

A second logic unit (LU2) is also initially programmed causing it to continuously transfer the contents of the program memory data register into the instruction register of a third logic unit (LU3) which actually executes the program. It, in turn, operates on several registers which may be used as accumulators, index registers, counters, and scratch storage, and upon a second memory used for data storage. If the memory address is set in MA2 by the logic unit, one machine cycle later the contents of the location may be read from the memory data register (MD2).

The following table illustrates the overlapping timing with which the program counter is advanced, the instructions delivered into the instruction

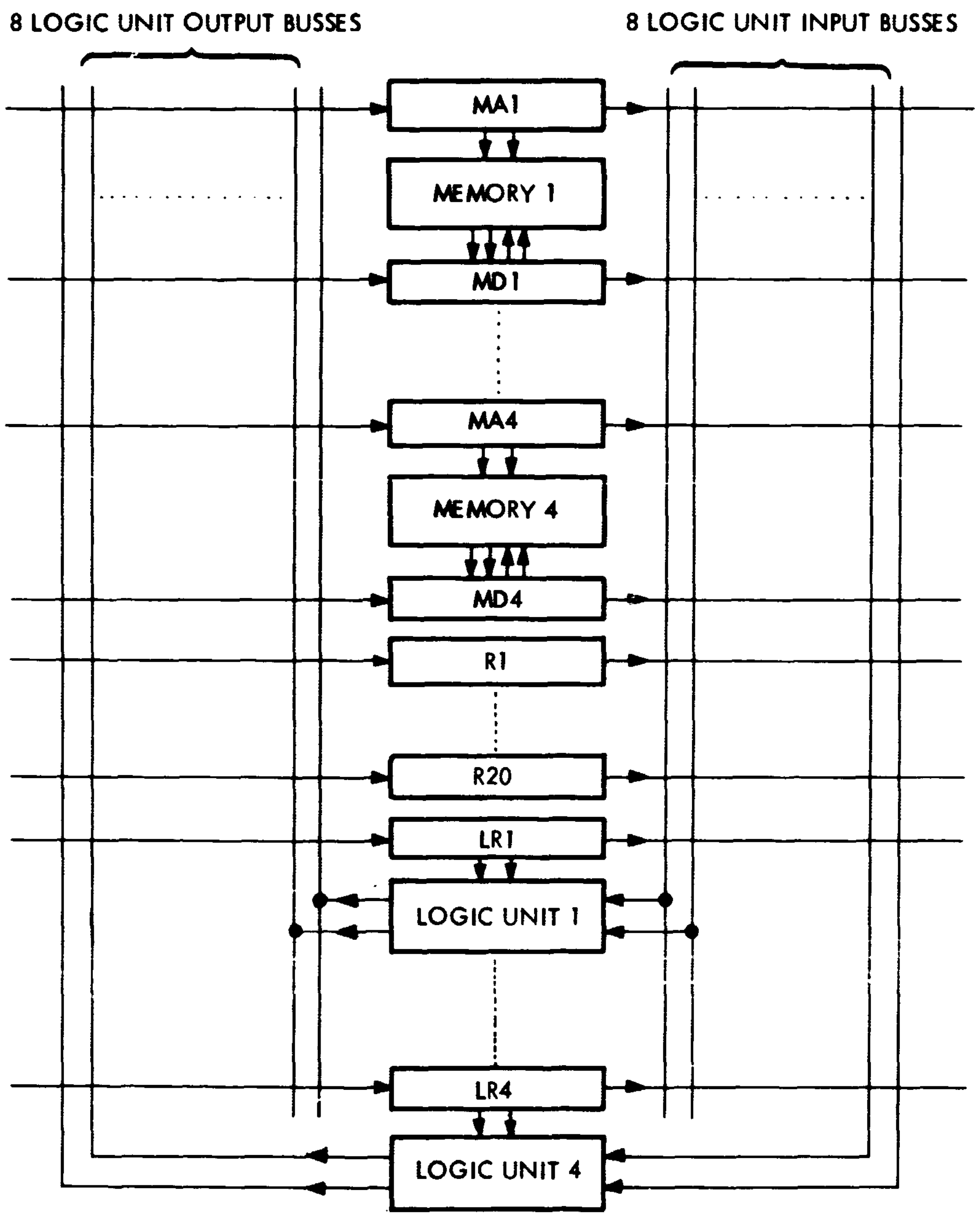


Figure 2. Original MULTIPAC Concept

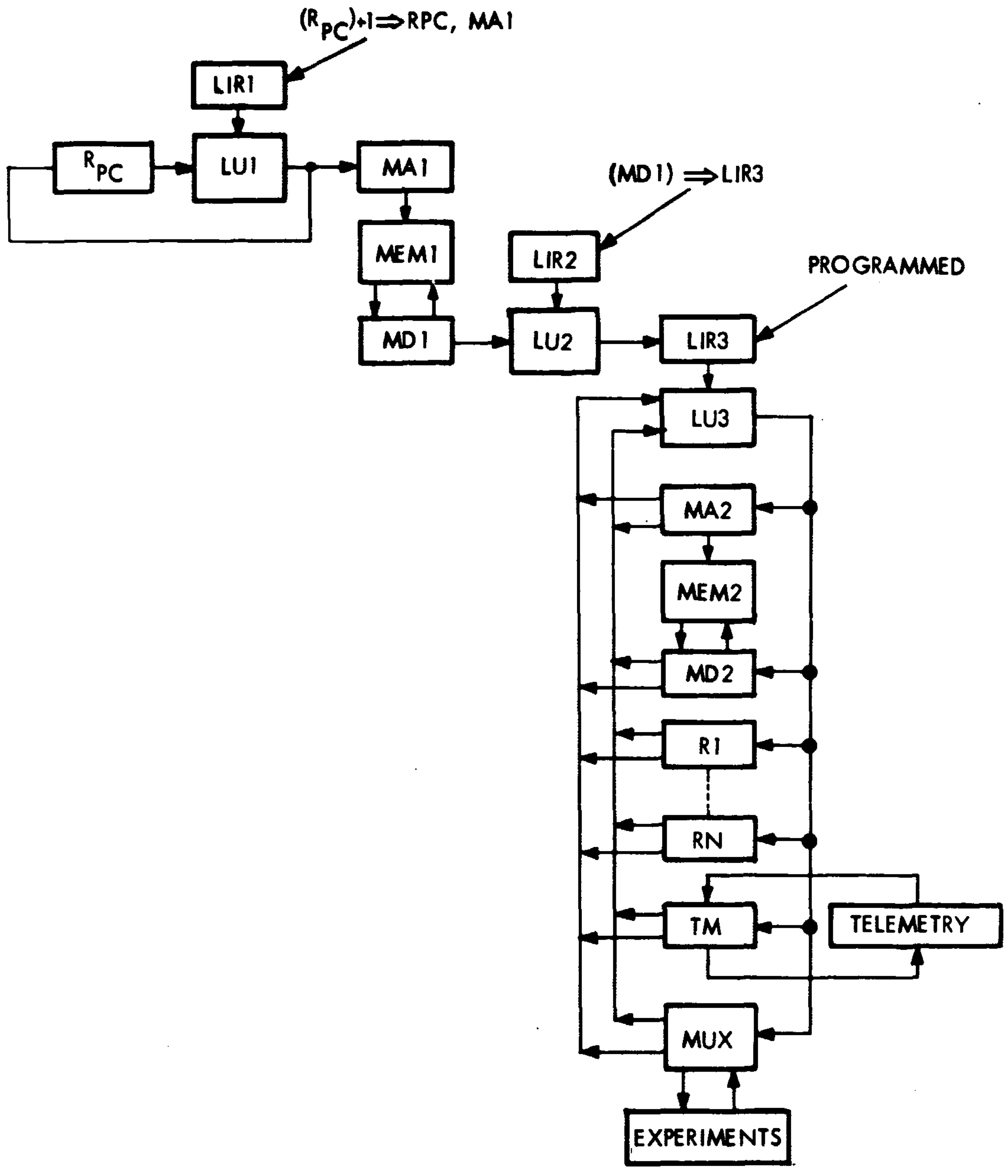


Figure 3. Data Flow in Original MULTIPAC Concept

register and, as an example, how an add from memory into the accumulator (R_{ACC}) is executed. Operation codes used are copy (COP), no operation (NOP), and add (ADD).

R_{pc}	MA1	MD1	LIR3	MA2	MD2	R_{ACC}
0	0	---	---	---	---	A
1	1	(0)	---	---	---	A
2	2	(1)	(0)=COP MD1, MA2	---	---	A
---	---	(2)	(1)=NOP ADDR	ADDR	---	A
---	---		(2)=ADD MD2, R_{ACC}	ADDR	(ADDR)	A
						A+(ADDR)

The original machine was to have used a 16-bit word having a 4-bit operation code and three 4-bit addresses. Addresses were 12 bits long and the operation code field contained all zeroes, defined as a no operation (NOP) instruction. Addresses were buried in the program stream in what were essentially two-word instructions and prevented from acting as instructions when they reached LIR3 by their NOP coding.

Three factors have proved troublesome in the practical design of the machine. First, there is a great deal of switching interconnecting all the modules in order that they may all be interchangeable. Second, the power limits set upon the design constrain the choice of circuitry to the lowest-powered (and lowest speed) logic families. Third, the real-time data processing requirement, initially assumed to consist of low-rate data formatting, has grown quite large, enough to tax a modern general-purpose computer, let alone the simple micro-ordered set of trivial modules originally envisioned.

These three factors have influenced the evolution of the design. The choice between a serial or parallel machine was resolved in favor of a serial one, largely in order to minimize the amount of switching logic between modules. Also involved in this decision was the question of speed versus power. Investigation indicated that a parallel machine would have been too large, considering the switching logic, and would exceed the power budget even with very low powered logic. The serial system was smaller and could stay within the power budget if constructed from very low powered logic. However, it would be an order of magnitude slower, which would have an adverse effect upon the ability of the machine to handle the processing load. This was alleviated to some extent by the use of a small percentage of higher powered, and thus faster, logic in the critical data paths.

The power budget was also responsible for the decision to reduce the machine word size to thirteen bits. Three-address instructions were eliminated, which not only disposed of one 4-bit address field, but also prohibited instructions designating two lccations in which the result should be stored. This permitted the simplification of the switching logic to include only one output data bus instead of two. One extra bit (the thirteenth) had to be added to distinguish data from instructions. The machine is now coded so that all instructions have the twelfth and thirteenth bit dissimilar, whereas for two's complement data the thirteenth bit acts as an extended sign bit, making it always the same as the sign, or twelfth bit.

Considering the speed that could be attained with such a serial machine (initially estimated at about 8-microsecond instruction times but later increased to 16 microseconds), it became necessary to consider a multiprocessing system having two or three independent processors in order to fulfill the real-time requirements. This, in turn, increased the number of modules required and the size of the switching matrix. At this point, it became necessary to depart from the generality of the Figure 2 arrangement, which had standard logic unit modules doing such simple tasks as incrementing the program counter and transferring the output of the program memory into the instruction registers. The logic units used for these simple tasks were eliminated. Now, self-incrementing logic is built into the memory address register and the program selection switch is built into the logic unit. The number of logic units was reduced by two thirds. Since logic units were no longer addressable devices, a secondary bus structure was created for the transfer of instructions. In essence, some of the simplicity and generality of the original concept had to be specialized to meet the demands of speed and efficiency.

In the course of designing the I/O devices, it was found that the multiplexer had switching problems very similar to those of the logic units. At least 200 I/O channels had to be provided, which called for something like four modules, each containing addressable 64-way switching of in, out, and control signals. This problem was solved by combining the multiplexer with the general-purpose registers and providing special instructions to use each bit of each register as an I/O channel. The number of channels required was essentially divided by 13 and the multiplex switching was moved back into the main address switching.

With the exclusion of logic units as addressable devices but the inclusion of 24 registers (which would provide 312 input and 288 output channels) and the number of processors set at three, the system has 40 addressable devices. This overtaxed the 4-bit address field, and therefore 3-bit base registers were incorporated in the logic unit addressing structure. Furthermore, to move a portion of the address switching outside of the logic unit, a second level of switching was incorporated which forms a common extension of the address switching of all the logic units. This is used to address the I/O registers and, in essence, constitutes the multiplexer switch, controlled by the logic unit base registers. Since this second level of address switching is common to all logic units, programming restrictions must be imposed to prevent different processors from interfering with each other.

Another step taken to increase the efficiency of the processing routines was the improvement of the A/D conversion method. The original concept called for analog-to-pulse-width converters at the experiments, the duration of whose output levels would be counted by the processor. This proved unworkable, since the processor would have had to devote itself to each conversion for about 50 milliseconds in a tight loop to achieve the required 8-bit accuracy. To solve the problem, a new module was created, a register which has a D/A ladder network on its outputs. This provides an analog reference signal to the experiments, and each of the latter now must have an analog comparator which returns a signal level to the processor I/O indicating whether their output signal is greater or less than the reference signal supplied. This makes possible the successive approximation method of conversion whose algorithm runs much faster, i.e., about 1.4 milliseconds for the maximum length (8-bit) conversion.

The instruction set was then enlarged from simple two-argument logical and arithmetic instructions and test commands to include the bit manipulating and I/O instructions necessary to make the I/O registers and the A/D algorithm operate efficiently.

Finally, the clock signals, the one set of signals which were common to all modules, were made less susceptible to failure by triplicating the buses and putting majority voting gates at each module interface.

At this point the external specifications and features of the MULTIPAC design were frozen in order that a detailed IC logic design could be done. The iteration of the design for LSI will offer opportunities for further improvements. Section 8 contains recommendations for such improvements as can presently be foreseen.

3.0 SYSTEM DESCRIPTION

The detailed block diagram of the MULTIPAC IC design is shown in Figure 4. This includes the control functions and details of data switching which were omitted from the simplified diagram of Figure 1.

3.1 Data Flow

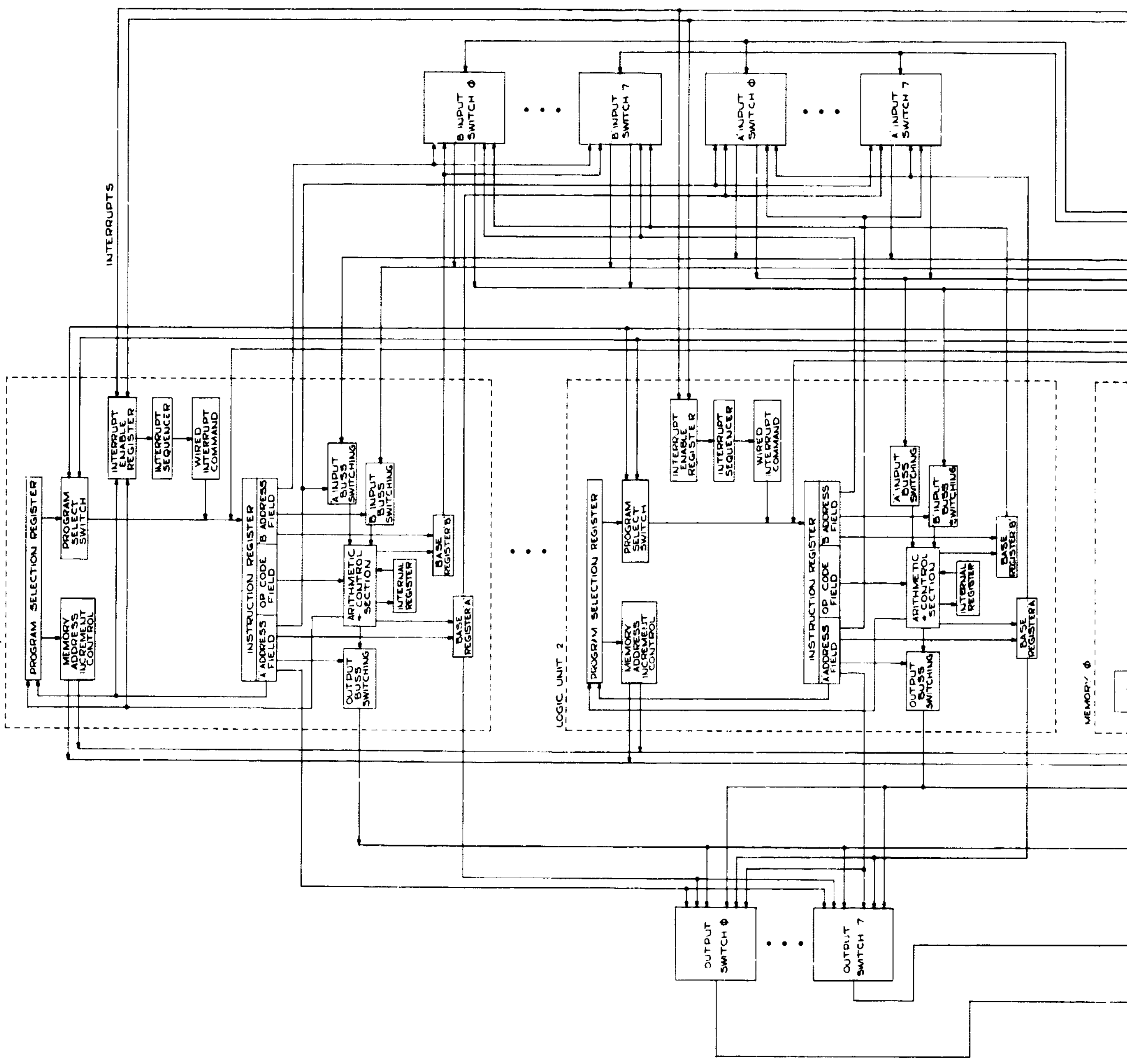
Data flow takes place only under the control of one of the three logic units. Each of these communicate directly with six memories, 24 general purpose registers also serving as I/O interfaces, two D/A registers, two command registers, and two telemetry registers.

The memories themselves each contain two addressable registers, the address register (MA) and the data register (MD). Except for the automatic incrementing of the MA, which will be considered under control functions, the memories operate entirely on the basis of transfers to and from these registers. When data is transferred into the MA, a read cycle is automatically initiated to read out the contents of the new location into the MD one cycle later. When data is transferred into the MD, a write cycle is initiated to store it in the location specified by the MA.

Each of the 24 general purpose registers has connections for 13 bilevel inputs and 12 bilevel (dc) outputs which may be accessed by a special I/O instruction. Thus, they provide 338 inputs (counting the D/A registers) and 288 outputs to the rest of the spacecraft and the instruments.

The D/A registers are very similar to the general purpose registers, having the same number of input channels, but the output channels are not present. Instead, a D/A ladder network is connected to provide an analog signal proportional to the arithmetic value of the register contents which is used by the experiments as a reference voltage in the successive approximation A/D conversion process.

The telemetry and command registers share a common address. Instructions operating on such an address will connect one of the command registers to its input bus and/or one of the telemetry registers to its output bus. Since these are normally addressed by simple copy instructions, they copy to the telemetry register or from the command register.


INTERRUPTS
B INPUT SWITCH Φ
B INPUT SWITCH 7
A INPUT SWITCH Φ
A INPUT SWITCH 7
LOGIC UNIT Φ
PROGRAM SELECTION REGISTER
PROGRAM SELECT SWITCH
MEMORY ADDRESS INCREMENT CONTROL
INTERRUPT ENABLE REGISTER
INTERRUPT SEQUENCER
WIRED INTERRUPT COMMAND
INSTRUCTION REGISTER
A ADDRESS FIELD
OP CODE FIELD
B ADDRESS FIELD
A INPUT BUSS SWITCHING
B INPUT BUSS SWITCHING
ARITHMETIC & CONTROL SECTION
INTERNAL REGISTER
BASE REGISTER B
BASE REGISTER A
OUTPUT BUSS SWITCHING
LOGIC UNIT 2
MEMORY Φ
OUTPUT SWITCH Φ
OUTPUT SWITCH 7

13A

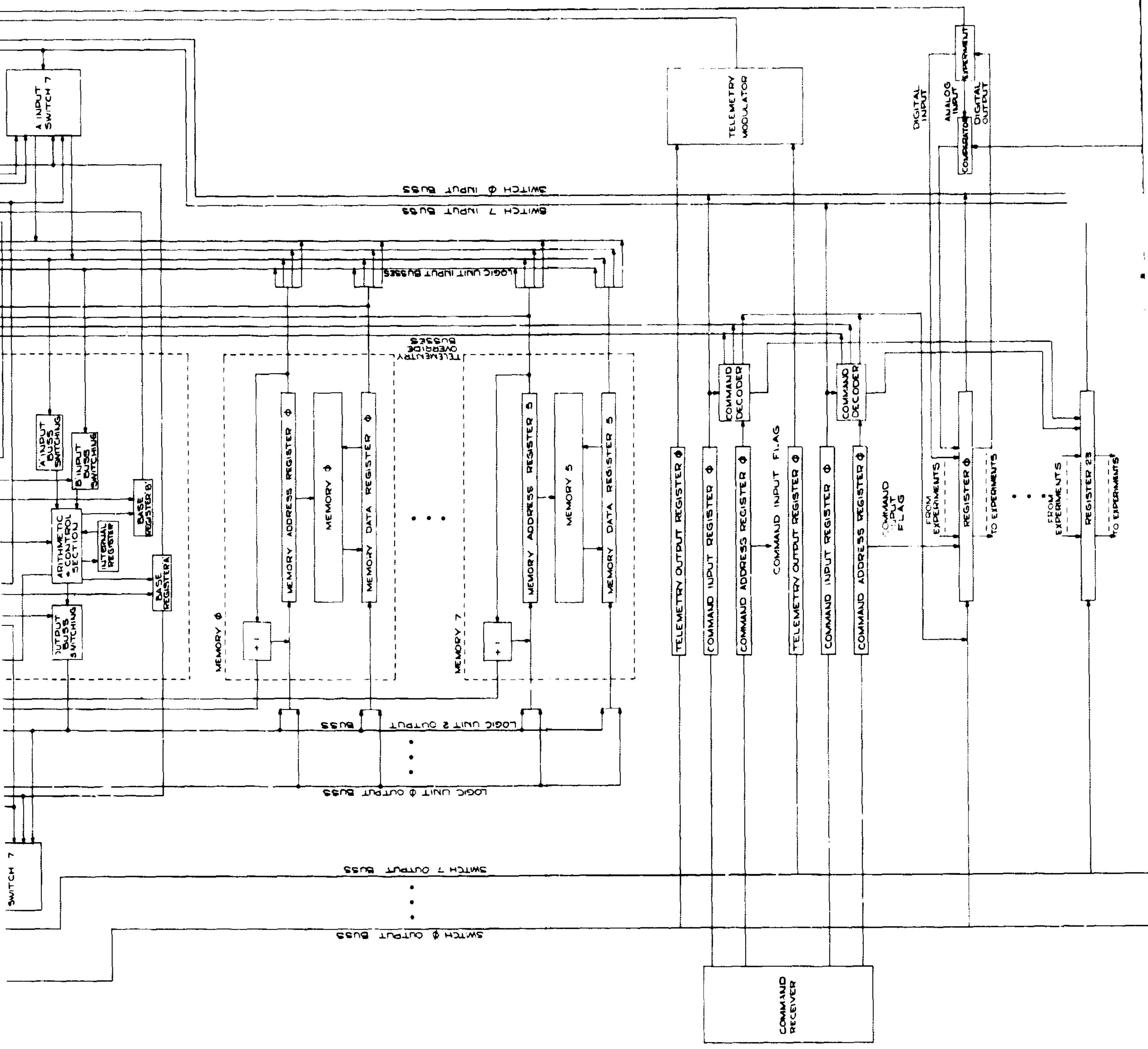

A INPUT SWITCH 7
TELEMETRY MODULATOR
DIGITAL INPUT
ANALOG INPUT
DIGITAL OUTPUT
EXPERIMENT
COMPARATOR
SWITCH Φ INPUT BUSS
SWITCH 7 INPUT BUSS
LOGIC UNIT INPUT BUSSES
TELEMETRY OVERRIDE BUSSES
COMMAND DECODER
COMMAND DECODER
'A' INPUT BUSS SWITCHING
'B' INPUT BUSS SWITCHING
BASE REGISTER B
ARITHMETIC + CONTROL SECTION
INTERNAL REGISTER
BASE REGISTER A
OUTPUT BUSS SWITCHING
MEMORY ADDRESS REGISTER Φ
MEMORY Φ
MEMORY DATA REGISTER Φ
MEMORY Φ
+1
MEMORY ADDRESS REGISTER 5
MEMORY 5
MEMORY DATA REGISTER 5
MEMORY 7
+1
TELEMETRY OUTPUT REGISTER Φ
COMMAND INPUT REGISTER Φ
COMMAND ADDRESS REGISTER Φ
COMMAND INPUT FLAG
TELEMETRY OUTPUT REGISTER Φ
COMMAND INPUT REGISTER Φ
COMMAND ADDRESS REGISTER Φ
COMMAND INPUT FLAG
FROM EXPERIMENTS
REGISTER Φ
TO EXPERIMENTS
FROM EXPERIMENTS
REGISTER 23
TO EXPERIMENTS
LOGIC UNIT 2 OUTPUT BUSS
LOGIC UNIT Φ OUTPUT BUSS
SWITCH 7
SWITCH 7 OUTPUT BUSS
SWITCH Φ OUTPUT BUSS
COMMAND RECEIVER

Figure 4.

13B

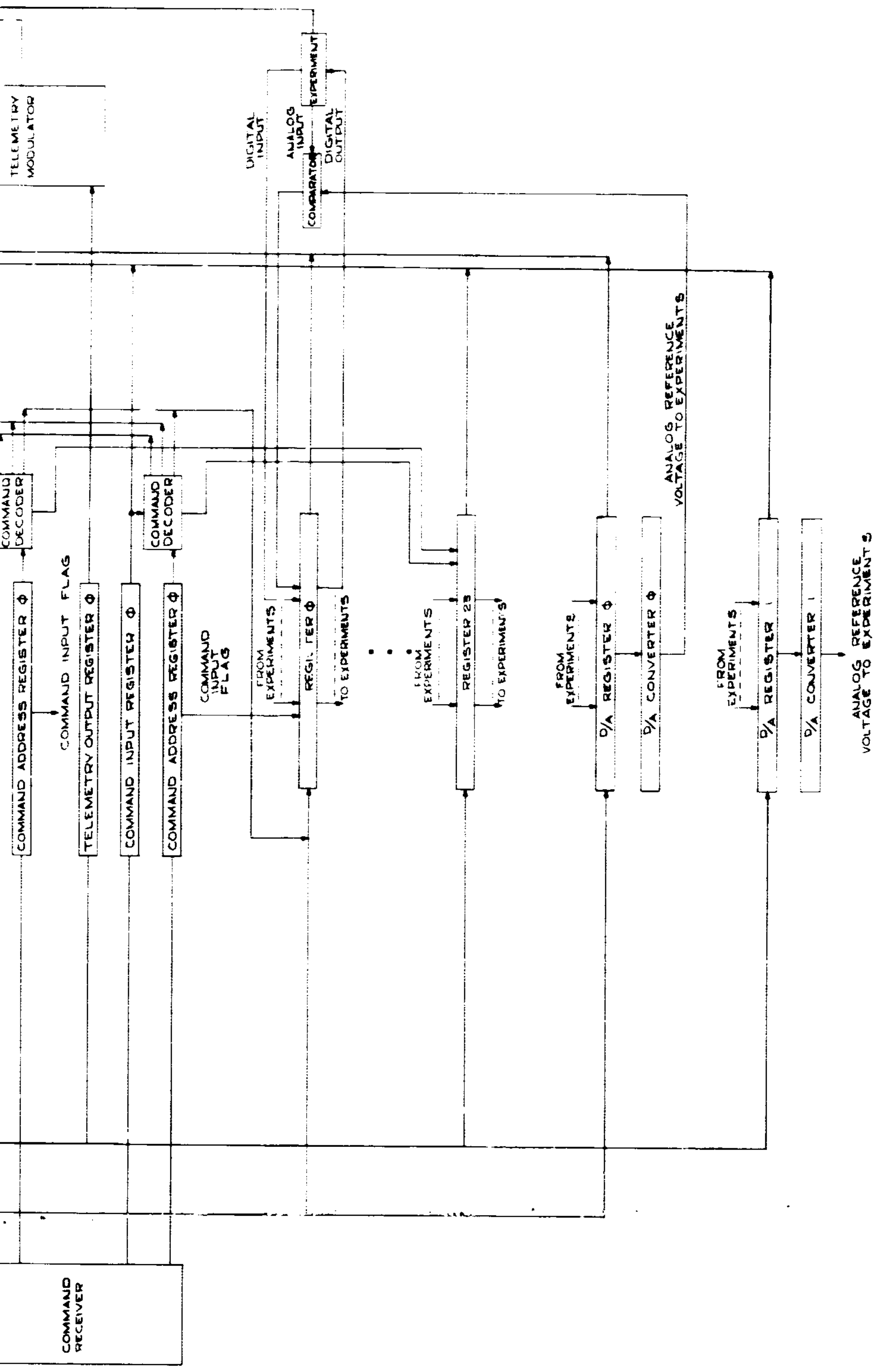


Figure 4. MULTIPAC Block Diagram

3.1.1 Address switching.-- Address selection is determined both by the address fields of the instruction and by the setting of base registers in the logic unit and is actually performed in two parts by the bus switching logic in the logic unit and by the external switching module operating on levels supplied to it from the logic unit. Each logic unit has 16 addresses directly addressable from each 4-bit field in the instruction word. Twelve of these directly address six memories. The other four addresses are passed on to the eight second-level switches where one of the four outputs of the switch is thus directly addressable by the instruction address field. The selection of a switch by a logic unit is controlled by the contents of a 3-bit base register which may be loaded by a special instruction. There are two such base registers per logic unit, one for each of the address fields in the instruction word. Each base register acts independently of the other register. There are three logic units, all connected similarly to the same switches. Figure 5 shows these address allocations. The 12 addresses of the six memories, which are common to all the logic units, plus the 32 outputs of the eight switches provide 44 total addresses.

The second-level switching arrangement introduces a programming restriction. Two logic units operating through the same second-level switch will have their input data busses ORed together, thus creating meaningless data. Therefore, separate programs running in separate logic units must not have their base registers set to the same value.

The intercommunication between processors, which is necessary to prevent their mutually interfering with one another, can be accomplished by connecting certain input and output channels located on different base register settings to one another, providing flags from one processor to the other.

3.1.2 The I/O system.-- The I/O interface of the central processor, as described in the Final Report of Phase 1 of this project, is summarized in Tables 1, 2, and 3.

There are 191 input channels and 126 output channels required. The science input lines are doubled in order to be connected through two different registers for path redundancy. Also, there is one output channel added for each serial digital input and each serial command output since these must have additional signals to control the serial transfer.

The multiplexing system which has been devised employs each bit in each of the registers of the machine as a bilevel input channel as well as a bilevel output channel. The use of a special I/O instruction at the time

INSTRUCTION ADDRESS																BASE REGISTER SETTING
00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17	
MAϕ	MDϕ	MA 1	MD 1	MA 2	MD 2	MA3	MD3	MA 4	MD4	MA5	MD5	DAϕ	Rϕ	R2	R2	0
												DAϕ	R 3	R 4	R 5	1
												DA1	R6	R7	R 8	2
												DA1	R 9	R10	R 11	3
												TMϕ	R12	R 13	R14	4
												TMϕ	R15	R16	R17	5
												TM1	R18	R19	R20	6
												TM2	R 21	R22	R 23	7

Figure 5. MULTIPAC Address Allocations

TABLE 1

SCIENCE INTERFACE LINES

Magnetometer:	3 Analog Inputs 2 Serial Commands 2 Bilevel Commands
Cosmic Ray Telescope:	4 Analog Inputs 5 Serial Inputs 1 Bilevel Input 4 Bilevel Commands
Plasma Probe:	11 Analog Inputs 1 Serial Command 1 Bilevel Command
Radio Propagation:	5 Analog Inputs 1 Serial Input 1 Bilevel Command
Neutron Detector:	19 Serial Inputs 2 Bilevel Commands
VLF Experiments:	6 Analog Inputs 1 Serial Command 1 Bilevel Command
Micrometeorite Detector:	2 Analog Inputs 3 Serial Inputs 3 Bilevel Inputs 4 Bilevel Commands
Totals:	31 Analog Inputs 28 Serial Digital Inputs 4 Bilevel Inputs 4 Serial Commands 15 Bilevel Commands

TABLE 2

ENGINEERING INTERFACE LINES

Orientation Subsystem:	5 Analog Inputs 5 Bilevel Inputs 10 Bilevel Commands
Power Subsystem:	14 Analog Inputs 1 Bilevel Input 11 Bilevel Commands
RF Subsystem:	15 Analog Inputs 9 Bilevel Inputs 10 Bilevel Commands
S/C Support Subsystem:	6 Analog Inputs 2 Bilevel Inputs 2 Bilevel Commands
Central Data Engr. Subsystem:	5 Analog Inputs 3 Bilevel Commands
Other Subsystems:	1 Serial Input 2 Bilevel Inputs 3 Serial Commands 4 Bilevel Commands
Totals:	45 Analog Inputs 1 Serial Digital Input 19 Bilevel Inputs 3 Serial Commands 40 Bilevel Commands

TABLE 3

I/O CHANNELS REQUIRED

Inputs (Science lines doubled):				
Analog	-	Science	62	
		Engineering	45	107
			107	
Serial	-	Science	56	
		Engineering	1	57
			57	
Bilevel	-	Science	8	
		Engineering	19	27
			27	191
Outputs				
Serial Commands	-	Science	4	
		Engineering	3	7
			7	
Bilevel Commands	-	Science	15	
		Engineering	40	55
			55	
		Control Lines for Serial Inputs	57	57
		Control Lines for Serial Commands	7	7
				126

that serial word transfer has ceased affects the data line differently from the other instructions. For all other instructions, the data line will be forced to zero after each data transfer. After an I/O instruction, the data line will be forced to a one after the data transfer and will gate the word strobe which either inputs data to a register or outputs data from it, depending on the contents of the data that was transferred. If the data word transferred to a register by an I/O instruction contains a ONE in the high order bit, the data will be dumped to an output register. If the data word contains all ZEROES, the I/O instruction will serve as an input command and the data from the input interface lines will be dumped into the address register to be read by the next instruction. I/O instructions will also cause interface shift pulses to be generated at the register, from which they are distributed to the peripheral devices connected to that register. These, in conjunction with enabling levels, are used to transfer serial data to and from the experiments.

The I/O interfaces fall into the following categories, and the methods for handling each of them is discussed in detail below.

Inputs:

Bilevel inputs:	Two-state signals which are sampled asychronously at the interface
Serial inputs:	2- to 17-bit words to be transferred serially into the machine
Analog inputs:	0- to 5-volt analog levels to be converted into digital words with up to 8-bit accuracy

Outputs:

Bilevel commands:	Single-bit commands which are held indefinitely as levels at the interface. Pulsed reset signals are also included here, but they are set, then cleared, by the program.
Serial commands:	Commands of two to five bits which are transferred serially to the peripheral device

3.1.2.1 Bilevel inputs:-- These are simply levels which must be read by the processor. They are present at inputs to individual stages of the I/O registers and are transferred into the registers by an input instruction addressed to the register. The operation is a clear-set in which the input instruction performs the clear by transferring a data word of all ZEROES into the register. The whole word is then read from the register by the processor executing a "test with mask" (TWM) instruction. This AND's the contents of the I/O register with a masking register which contains a ONE in the bit to be tested and sets the Test flip-flop if the result produces a ONE. The Test flip-flop in the logic unit is thus set to the state of the input bit, and the program can pack such bits into words as necessary. These inputs require only one input channel each, and no output channel.

3.1.2.2 Serial inputs:-- Serial data will appear as sequential bilevel inputs to one stage of an I/O register, but there must also be output signals to synchronize shifting of the data with reading it into the register. A channel will be allocated to each such input which will be used to gate the peripheral device into its outputting mode. It will then respond to shift pulses from the register which occur each time an I/O instruction is addressed to the register.

3.1.2.3 Analog inputs:-- A/D conversion is accomplished by the successive approximation method, where processor software is used for the customarily hard-wired conversion logic. Two addressable registers are equipped as D/A converters. Their outputs are fed into a ladder network and the resultant analog signals distributed through isolating amplifiers to all devices requiring A/D signal conversion. Each such analog signal will be connected to its own comparator, which will also receive the distributed reference signal from the D/A output. The output of the comparator will then be treated as a bilevel input to one stage of an I/O register.

To accomplish the conversion, the program sets a ONE in the high order end of theD/A converter, which is a digital number one-half the number range of the register, and produces an analog output equal to one-half of the analog signal range. The comparator response indicates by its output which signal is larger. This is detected by the program through testing the input channel, and the high order bit in the D/A register is left at ONE or set to ZERO according to whether the analog signal is greater or less than half the signal range. The next highest bit is then set to ONE, and the process is repeated to see whether the analog value is greater or less than 1/4 or 3/4, depending on what the first bit was, and so on. After all bits are thus determined, the converter value is read out of the D/A converter. The interface required, apart from the distribution of the D/A converter outputs, consists of one input channel per analog signal.

3.1.2.4 Bilevel command outputs:-- The register module includes an output buffer register which is loaded in parallel from the shifting register using the word clock timing. This is a typical structure throughout the machine. In this case, however, the buffer register will be loaded only upon receipt of an output command. The outputs of this buffer register constitute the bilevel output channels. This arrangement provides both an input and an output channel for each stage of the I/O register, and since the number of each is approximately equal, it effectively doubles the multiplexing.

Since there is no way of outputting data to one channel without affecting others, and no way of transferring the contents of the output buffer register back into the processor to regenerate the bits which should not be changed, it is intended that a command table of 24 locations be kept in memory. Command routines would operate on the appropriate word in memory to alter the appropriate bits, using masking instructions that leave the other bits unaffected. Then the updated word would be transferred to the register by an I/O instruction.

3.1.2.5 Serial command outputs:-- Two bilevel channels are required for serial commands, one to switch the peripheral device to its input mode and the second to provide data levels. A shift pulse from the register will be provided each time it receives an I/O instruction.

3.1.3 Telemetry output.-- Figure 6 is a block diagram of the telemetry and command interfaces. Data to be output to the telemetry modulator is transferred in maximum word lengths of twelve bits to the telemetry register. The next higher order bit, to the left of the data word, must always be a ONE because the output operation requires it. The telemetry register will shift right, with each pulse of an external telemetry clock delivered to it, outputting the contents of the register serially into a 1-bit buffer flip-flop whose output may be used to directly modulate the telemetry transmitter. The telemetry register is filled from the left with ZEROES as it shifts so that when the last bit of data has been shifted into the buffer, it contains a ONE in the low order bit and otherwise all ZEROES. This condition is detected by a decoder whose output is sensed by the processor which, in turn, causes the next output word to be transferred into the telemetry register. It must act within one bit time of the telemetry rate in order that the next data bit can be transferred into the output buffer when required. The decoder output may be connected to one of the I/O input channels to be sensed by the processor, but the response time required at a high telemetry rate could require checking the flag frequently enough to prevent the processor from doing other unrelated tasks efficiently. The normal mission configuration, therefore, should be to connect the decoder output onto one of the interrupt lines.

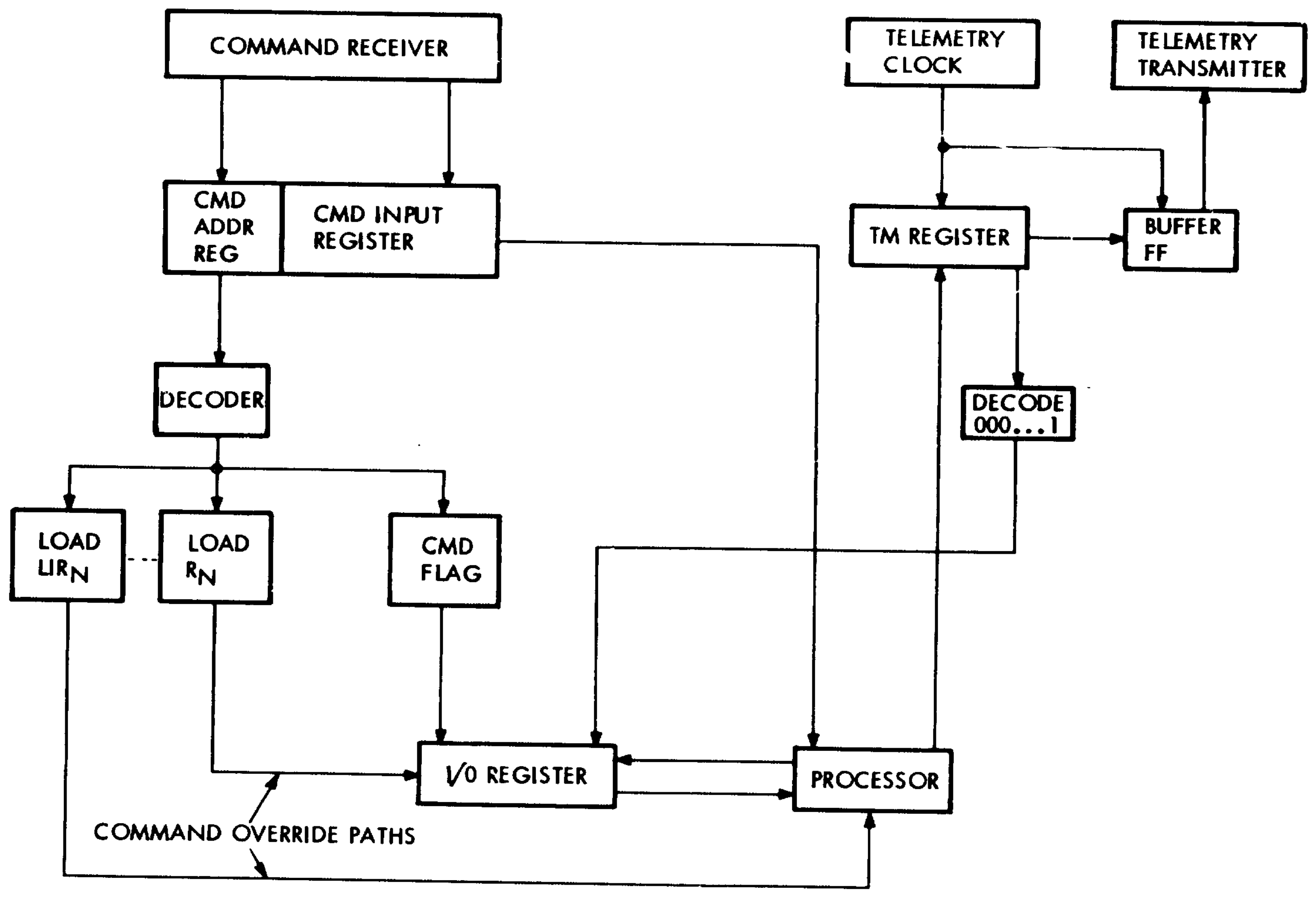


Figure 6. MULTIPAC Telemetry and Command Interface

3.1.4 Command inputs.-- Normal commands received from the command transmitter, i.e., those which do not attempt to override the processor, are loaded into the data register of the command input module and a flag sensed by the processor is set. The processor then reads out the contents of the command register under program control. Although this sensing could be accomplished by interrupt, as described above for the telemetry output control signal, it is anticipated that the up-link data rate will be slow enough to permit periodically sampling the flag level connected through one of the normal input channels.

The different operations caused by commands which override the program in order to acquire control of the processor are explained in paragraph 3.2.3 below.

3.2 Control Functions

The paths for program flow in the MULTIPAC are separate from the data flow and are determined by the program switch in each logic unit described below. In addition to this software control, there is provision for interrupting and an overriding input to each logic unit from the telemetry registers which permits the command link to acquire control of the system at any time.

3.2.1 Program selection.-- Program sources for each logic unit are selected by a switch in the logic unit itself. This switch, in turn, is controlled by the contents of the 3-bit program selection register. This register is addressable only by the logic unit of which it is a part, and its contents are entered by a special instruction. The switch addresses any one of the six memories. When a memory is thus selected, a control line from the logic unit causes the memory address to increment at each word time and the memory data register to shift out its contents into the instruction register (LIR) of the logic unit. From there, these instructions are dumped into the buffer register on each word pulse, where they are decoded to control the operation of the logic unit.

3.2.2 Instruction list.-- Each logic unit in the MULTIPAC system performs the instruction set listed in Table 4 using the instruction format shown in Figure 7.

Programming of MULTIPAC is accomplished through groups of machine microinstructions which actually constitute multiple word macroinstructions and must take into account the MULTIPAC machine timing for their

TABLE 4

MULTIPAC INSTRUCTION LIST

	Mnemonics	Operation Code	Operation
(1)	ADD, R1, R2	20	Add the contents of R1 to R2 and put the result in R2.
(2)	SUB, R1, R2	21	Subtract the contents of R1 from R2 and put the result in R2.
(3)	COP, R1, R2	13	Move the contents of R1 to R2.
(4)	JMP, R1, R2	22	Copy R1 into R2 and blank the second instruction which follows.
(5)	JCN, R1, R2	26	If the Test flip-flop is a ONE, copy R1 into R2 and blank the second instruction which follows; otherwise, do nothing.
(6)	XOR, R1, R2	10	Perform a bit-by-bit logical exclusive or between the contents of R1 and R2, putting the result in R2.
(7)	LOR, R1, R2	24	Perform a bit-by-bit logical OR of R1 and R2, putting the result in R2.
(8)	AND, R1, R2	14	Perform a bit-by-bit logical AND of the contents of R1 and R2, putting the result in R2.
(9)	SBR, N, M	27	Set Base Register 1 to literal M; set Base Register 2 to literal N.
(10)	SHR, R1, R2	12	Cycle R1 one bit to the right; shift R2 one bit to the right, replacing the high order bit with the bit cycled in R1.

TABLE 4.-- Continued

MULTIPAC INSTRUCTION LIST

	Mnemonics	Operation Code	Operation
(11)	CYC, R1, N	16	Cycle the contents of R1 N bits to the right, where $1 \leq N \leq 12$
(12)	TWM, R1, R2	15	Test the bit-by-bit logical AND function of R1 and R2 and set the Test flip-flop (TFF) to ONE if the result is nonzero. Otherwise, clear this flip-flop.
(13)	CWM, R1, R2	11	Copy the value of the Test flip-flop into the bits in R2 not masked by zeros in R1.
(14)	CIO, R1, R2	23	Copy the contents of R1 into R2. If the high order bit of R2 = 0 after the transfer, load the input interface signals into R2. If the high order bit of $R2 \neq 0$ after the transfer, load the contents of R2 into the output register of R2. Then transmit an interface transfer pulse (TCLK) from R2.
(15)	TST, R, X	17	Test the register R for conditions specified by X and set the Test flip-flop (TFF), if the condition is met: X = 01 equal to zero 00 not equal to zero 04 less than zero 14 greater than zero 15 less than or equal to zero 05 greater than or equal to zero 11 equal to one 10 not equal to one 07 Unconditionally set TFF to ONE.

TABLE 4.-- Continued

MULTIPAC INSTRUCTION LIST

	Mnemonics	Operation Code	Operation
(15)	TST, R, X (continued)	17	X = C6 Unconditionally set TFF equal to ZERO.
			02 Set TFF equal to the overflow flip-flop (OVF).
			03 Set TFF not equal to the overflow flip-flop (OVF).
(16)	OPR, X, R	25	Perform the operation specified by the X field on R as follows:
			X = 07 INC R - Increment the contents of R.
			06 DCR R - Decrement the contents of R.
			00 CMP R - Perform the one's complement of R, putting the result in R.
			02 SRT R - Shift right the contents of R, filling the high order bit with the contents of the Test flip-flop.
			03 SHL R - Shift left the contents of R, filling the low order bit with zero.
			01 XCH R - Exchange the contents of R with the contents of the LU register.

TABLE 4.-- Concluded

MULTIPAC INSTRUCTION LIST

	Mnemonics	Operation Code	Operation
(16)	OPR, X, R (continued)	25	05 MSO R - Copy a number in the LU register which is two larger than the most significant one in R.
			14 SPS N - Set the Logic Unit Program Switch to literal N.
			10 SIE N - Set the Interrupt Enable Register to literal N.

BITS 8 - 12	BITS 4 - 7	BITS 0 - 3
OPC	A	B

OPC (bits 8-12)	=	Operation Code (Bits 12 and 11 are different for instructions, identical for data)
A (bits 4-7)	=	Register Address Field 1
B (bits 0-3)	=	Register Address Field 2

Figure 7. Instruction Format

proper operation. When programming, care must be taken in: (1) copying addresses or literal data from the program stream, where they are imbedded in what are essentially the address fields of multiple word instructions, and (2) allowing for the timing of the memory read or write cycles initiated by transferring data into the address or data registers respectively.

The example below of adding to a register R, used as an accumulator, for memory shows both these aspects of machine timing. Memory No. 1 is used as the program memory and Memory No. 2 is used as the data memory. MA and MD refer to the memory address register and memory data register, respectively. The LIR is the instruction register from which instructions are decoded in the logic unit.

Starting in location 0, consider the following three instructions in Memory No. 1:

Location	Instruction
0	COP MD1, MA2
1	ADDR
2	ADD MD2, R

This leads to the following machine timing, where each entry in the list represents the execution time of one microinstruction.

Contents of MA1	Contents of MD1	Contents of LIR	Contents of MA2	Contents of MD2	Contents of R
0	--	--	--	--	(R)
1	(0)	--	--	--	(R)
2	(1) =ADDR	(0) =COP MD1, MA2	--	--	(R)
--	(2)	(1) =ADDR	ADDR	--	(R)
--	--	(2) =ADD MD2, R	ADDR	(ADDR)	(R)
--	--	--	ADDR	(ADDR)	(R) + (ADDR)

Note that the instruction sequencing permits: (1) the COP instruction to be executed in LIR while ADDR is in MD1, and (2) the ADD instruction is executed in LIR after the memory cycle has loaded (ADDR) in MD2.

ADDR also appears in LIR and is thus treated as an instruction. The thirteenth bit of the machine, however, is coded to distinguish instructions and data. The operation field of all executable instructions is coded with the 12th and 13th bits dissimilar. Data words and addresses are limited to 12 bits in length and have the 13th bit the same as the 12th bit. Thus, the arithmetic unit can treat the 13th bit as any other, but a difference between the 12th and 13th represents an overflowed condition. Data and addresses which are not overflowed can safely be imbedded in the program since they will be decoded by the LIR as non-executable as instructions.

Jumping is accomplished through the data flow paths (shown in Figures 4 and ~~5~~6) by the following instruction sequence.

Contents of MA1	Contents of MD1	Contents of LIR
0		
1	(0)	
2	(1) =JMPADR	(0) =COPY MD, MA
JMPADR	(2)	(1) =JMPADR
JMPADR+1	(JMPADR)	(2) =NOP
JMPADR+2	(JMPADR+1)	(JMPADR)

The data flow path has a hard-wired priority over the program control function which increments the address register and thus overrides it as it shifts in the jump address.

3.2.3 Command link override.-- The command module inputs to the program switch exercise hardwired priority over the source dictated by the register setting, and this overriding control is used to take over the MULTIPAC system for reprogramming. An instruction from the telemetry unit will replace any instructions from the normal program stream. The first reprogramming instruction to each logic unit causes the latter to reset its program switch to a blank input where it reads only zeroes.

This is interpreted as a series of no operation (NOP) instructions and the logic unit is effectively disabled. Since only the logic unit itself can address its own program switch, there is no danger of another processor which is still active interfering and restarting it. During the relatively long period while each instruction of the reprogramming bootstrap loader is being received on the command link, the logic units are disabled, but when an instruction has been assembled in the command receiver and transferred to the MULTIPAC command register, it is inserted in the stream of NOP's at the normal instruction rate.

Figure 6 shows the block diagram of the command interface. The command receiver must assemble a digital word of data, and load this word into one of the two CMD registers in the CDS.

The transfer signal dumps the CMD address and CMD data into the proper registers of the CMD module, the data going into the register which is addressable by the processor. The decoding logic addresses one of seven devices in the MULTIPAC system and transfers the data word into one of them, or on decoding the eighth address, sets a flag which is sensed by the program. The latter is the normal method of command input (see paragraph 3.1.4 above).

The seven devices which are addressed are the instruction registers of the three logic units and four of the general purpose registers. These are used for taking control of the MULTIPAC system for reprogramming. The signals from the command receiver to the logic units override those from the program selection switch contained in each logic unit, and the first three instructions in a reprogramming sequence direct each logic unit to set its program select switch to a blank position where only NOP's will be read. Thus, there will be no program action from the logic units except when a telemetry word is directed to one of them. Each word of a bootstrap loader is then read into one of the general purpose registers, followed by an instruction to one of the logic units to store it in one of the memories. Instructions setting and incrementing the memory address are also directed to the logic unit. Once the bootstrap loader has been stored in memory, an instruction is directed to a logic unit to set its program control switch to that memory. From then on the operational program is read in through the usual command input. The loader program detects each new word when it is ready in the CMD register, stores it, and increments memory address. Finally, the override must be employed again to set the program selection switch of each logic unit to the memory into which its program has been loaded.

3.2.4 Interrupts.-- Each logic unit has four interrupts which are connected in parallel with those from the other logic units. A 4-bit masking register in each logic unit, called the interrupt enable register, is loaded by special instruction and sensitizes that particular logic unit to the interrupts corresponding to the stages in which it contains ONES. Upon responding to one of the interrupts, the logic unit breaks off the program stream and executes a wired instruction to exchange the contents of the program memory address register with the contents of its own internal register. The latter must initially contain the address of the interrupt routine and afterward will hold the return address. Actually, the contents of the register will be greater by three than the correct return address owing to the overlapping timing and the inhibiting of instruction execution as soon as the interrupt begins. The return address must therefore be corrected before the return to the main program can be accomplished. The program sequence is given below.

Contents of MA1	Contents of MD1	Contents of LIR	Contents of INTR	Contents of R1	Interrupt in Process
100	--	--	INTADR	--	0
101	(100)	--	INTADR	--	0
102	(101)	(100) LAST EXECUTED INSTR	INTADR	--	0
103	(102)	(101)	INTADR	--	1
104	(103)	(102) XCH MA1		--	1
INTADR	(104)	(103)	104	--	1
INTADR+1	(INTADR)	(104)	104	--	1
	(INTADR+1)	(INTADR)	104	--	0
		(INTADR+1)	104	--	0
		INTERRUPT SUBROUTINE			

The interrupt lines must also appear as bilevel inputs to the I/O interface in order that they can be tested by the program to see which one occurred. The interrupt routine is also responsible for storing any data which it may destroy.

Since a second interrupt during the interrupt subroutine would destroy the return address, the interrupt enable register is cleared at the start of each interrupt and must be reloaded from a memory image to re-enable the proper interrupts immediately before returning to the main program.

4.0 DESCRIPTION OF THE MULTIPAC INTEGRATED CIRCUIT LOGIC DESIGN

In October 1967 a baseline system was established for the detailed logic design and documentation of MULTIPAC in integrated circuit form. It was initially intended to constitute the final MULTIPAC design. However, the increasing development of practical LSI technology together with system improvements discovered in the course of working out the present design, have made it desirable to consider the integrated circuit design reported here as a feasibility study to guide subsequent LSI implementation. This report is, therefore, written as a Research Report on the integrated circuit work rather than a Final Design Report.

4.1 Circuitry

4.1.1 Logic circuits.-- The chief constraint on the choice of integrated circuits for logic was the extremely low power consumption demanded. The Fairchild LPDTμL line was chosen as having the features best suited to the design among the logic families available in the required speed-power class at the time the choice was made.[1] The manufacturer's specifications for those circuits used in the design are included as Appendix C. Other logic families considered were Philco MEL, RCA LPDTL, Amelco 500 Series TTL, Signetics 400 Series TTL, and the several manufacturers' families deriving from Fairchild MWμL. Chief considerations in the selection were 1) low-power (approximately 1 mw per gate) operation; 2) circuit delay less than 100 nanoseconds; 3) availability of collector-ORed bus drivers; 4) availability of a compatible higher powered, higher speed logic family; and 5) maximum fan-in and fan-out. The Fairchild circuits required the least power of those considered except for the Philco MEL or the lowest powered of the Amelco circuits, and additionally met the remaining criteria better than the latter two. The Amelco circuits which were competitive in power levels with the Fairchild were significantly slower. The Philco family, while exhibiting the best speed/power ratio in the group, included only a very restricted choice of circuits with limited fan-in and fan-out capability. The Fairchild circuits were also in more widespread use, thus offering better statistical data. Two drawbacks of the Fairchild circuits for the MULTIPAC design were noted: 1) The flip-flops required a relatively long clock pulse width which takes in data on the rising edge and outputs on the falling edge. This pulse width must be added as delay into all propagation paths. 2) There is a fan-out of only one when the low-speed circuits drive the high-speed family.

The above matters should all be considered in the choice of LSI circuitry. However, they should not present the problems in tailored LSI circuitry that they do in working with a standard off-the-shelf family of integrated circuits.

4.1.2 Memory circuits.-- The memory storage medium postulated is a modification of the 256-bit complementary MOS memory chip under development for NASA by Westinghouse on contract No. NAS-5-10243.[2] As with the integrated circuits, the chief reason for this choice was the low power drain of this medium. In a flip-flop memory, one important consideration is the power required to maintain the memory contents under dc conditions. Standby power required for this MOS circuitry is reported as typically less than one nanowatt per cell, which would be approximately .026 milliwatts for a 2048-word, 13-bit memory. This is negligible compared to the operating power of the one chip in the memory selected by the addressing logic, which is reported as 30 milliwatts for the present device containing 16 words of 16 bits each when cycled at a 0.5-MHz rate. If this were scaled to a 13-bit word with a 16-microsecond cycle time, the indications are that the 2048-word memory would consume on the order of 3 milliwatts. This power is, in turn, negligible compared to that required by the bipolar portions of the memory module, especially the circuits interfacing the MOS chip. These would probably require special design to minimize their power consumption.

In addition to shortening the word length to 13 bits for the present system (or removing power from those bits not needed), two other changes in the Westinghouse chip are needed to make its use practical in MULTIPAC system. There are presently no clear provisions for the use of multiple 16-word chips in a larger memory. There is no method of expanding the addressing structure except for gating the "strobe" and "write" signals of each chip separately, and the "bit lines" are not presently such that they can be collector ORed to form the larger memory. If both these signals from each chip have to be separately ORed or gated after leaving the chip, it would require more interface circuits than memory chips and the power demands would be exhorbitant. Making these changes on the chip would seem relatively easy, however. The change to the output circuits to permit ORing them means gating out the active pull-up circuit as well as the pull-down circuit from non-selected chips in a similar manner to what is now done for writing. This leaves only the output circuits from the selected chip active to drive the bit lines. In write mode they too are disabled, leaving only the write drivers active on these lines. In addition, the addressing logic should be expanded to include two more inputs which would gate all address decoders and appear on outside pins to be used for x-y coordinate addressing of the chip. This would reduce the number of interface address drivers required from 128 to 24, with a corresponding saving in power. The logic diagram given here for the memory assumes these modifications will be made.

4.1.3 Special circuits.-- There are several non-logical circuits required which should be expecially designed for MULTIPAC, such as the memory interface circuits mentioned above. Such circuits are not properly a part of the logic design but they do influence the parts count, power and reliability estimates. Therefore, certain assumptions have been made concerning them as follows:

(1) Clock generator: This circuit must produce approximately 350 nsec wide, 5-volt pulses at 1.0 MHz. One circuit per package assumed, with 100-milliwatt power consumption.

(2) Bipolar-to-MOS interface circuit (BMIC) which interfaces the MULTIPAC logic levels to the memory. Two circuits per IC package assumed, with 10-milliwatt power consumption per circuit.

(3) MOS-to-bipolar interface circuit (MBIC) which interfaces memory to the MULTIPAC logic. Two circuits per package assumed, with 10-milliwatt power consumption per circuit.

(4) Analog switch (ANSW): This circuit switches D/A inputs to ladder network. One circuit per package assumed, with 25-milliwatt power consumption per circuit.

(5) Analog amplifier (ANAMP): These are isolation amplifiers which supply the analog output to the experiments. One circuit per package assumed, with 100-milliwatt power consumption.

4.2 Parts Count

The parts count for the integrated circuit design is given below. The numbers of integrated circuits required for the processor proper and for the memories are given separately in order to facilitate comparison with processors having non-integrated circuit memories. The entire system requires about 3000 IC's, almost half of which are found in the memories, with the remainder in the processors and I/O. Of the latter, about half the parts are devoted to the registers which, in a sense, constitute the I/O multiplexer, and about half are used in the three processors themselves.

Module Type	No. of IC's Per Module	No. of Modules	Total IC's
A. Processors and I/O			
Logic Unit	190	3	570
Address Switch	19	8	152
Register	30	24	720
D/A Register	26	2	52
Telemetry Unit	13	2	26
Command Unit	35	2	70
Timing Generator	31	1	31
		Total	1621
B. Memories			
Memory Storage	128	6	768
Memory Control	87	6	522
		Total	1290
C. System Total			2911

4.3 Weight

Assuming a packaging density of 500 flat-packs per ground and a 70-percent utilization of available mounting spaces, the expected weight of the system is:

$$\frac{2911 \text{ FP's}}{0.7} \times \frac{1 \text{ lb}}{500 \text{ FP's}} = 8.3 \text{ lbs}$$

4.4 Volume

Assuming a packaging volumetric density of 20 FP's per cubic inch and a 70-percent utilization of available mounting spaces, the expected volume of the system is:

$$\frac{2911 \text{ FP's}}{0.7} \times \frac{1 \text{ in}^3}{20 \text{ FP's}} = 208 \text{ in}^3$$

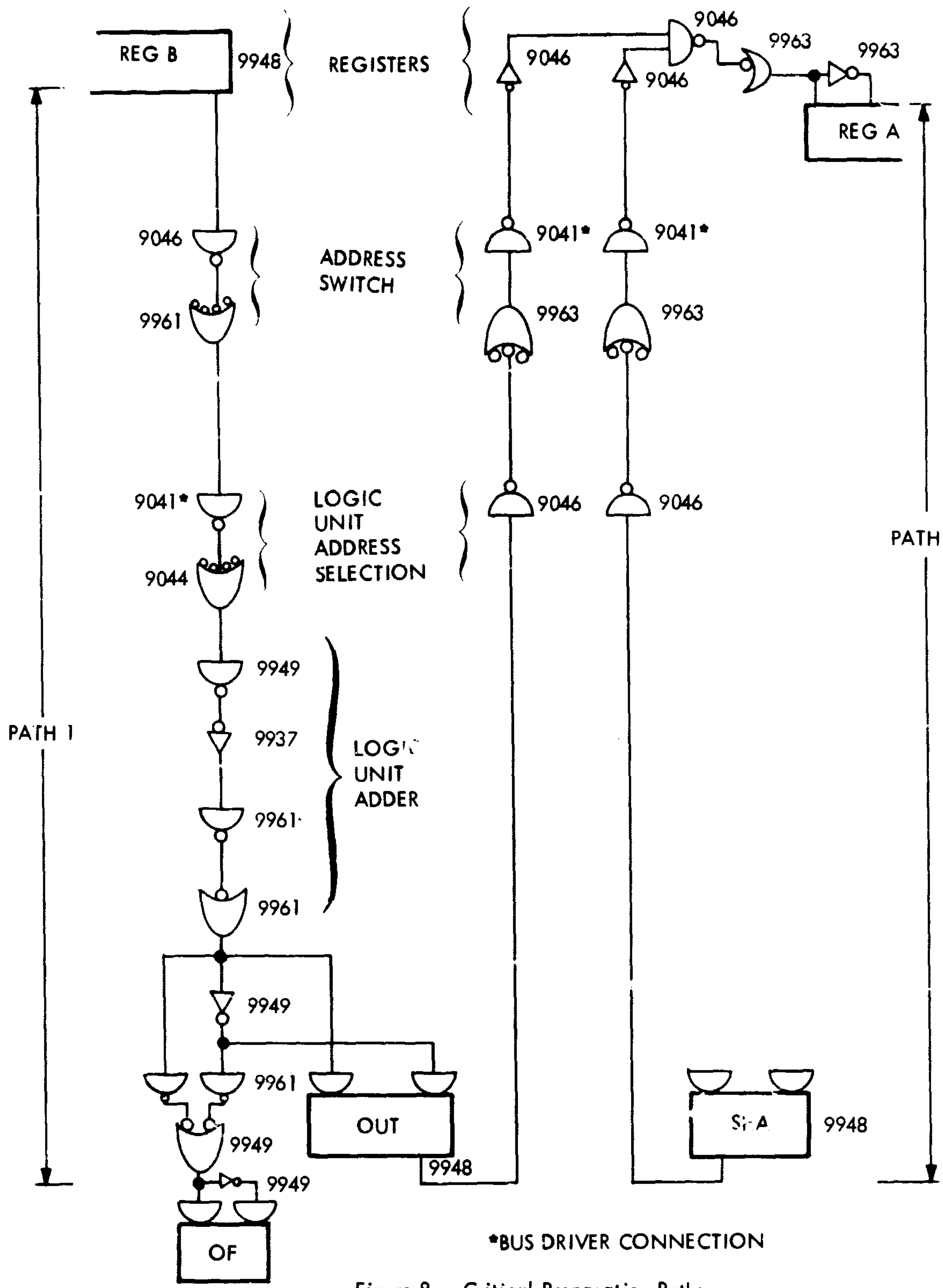
4.5 Power

The anticipated power consumption of the MULTIPAC System is about 18 watts, as indicated in the table below. The largest power consumers are the logic units at about 4 watts, the I/O registers at about 6 watts, and the memories drawing about 6 watts.

Module Type	Module per Module (ma)	No. of Modules	Total Power (mw)
A. Processor and I/O			
Logic Unit	1276	3	3828
Address Switch	112	8	896
Register	247	24	5928
D/A Register	467	2	934
Telemetry Unit	158	2	316
Command Unit	87	2	174
Timing Generator	384	1	84
			12.160 w
B. Memory			
Memory Storage	3	6	18
Memory Control	987	6	5922
			5.940 w
C. System Total			18.100 w

4.6 Speed

Figure 8 shows the critical propagation paths of the integrated circuit design with each circuit type indicated. There are two paths which are distinctly longer than all others, from the output of the source (B) register to the logic unit output flip-flop where it is reclocked, and from the output of that flip-flop to the recipient (A) register. Actually, the first path has a slightly longer branch as its end to the overflow flip-flop.



*BUS DRIVER CONNECTION

Figure 8. Critical Propagation Paths

Table 5 lists the worst case delays of the circuits involved for three temperatures: -55°C., +25°C., and +125°C. These unpublished specifications were obtained from the manufacturer by telephone. The output capacitance assumed in these specifications is 30 pf, permitting an average of four loads at 5 pf each plus about 6 inches of signal etch at 20 pf/foot. This seems a reasonable nominal value.

Table 6 tabulates the delays of the two critical paths for each of the three temperatures. In addition, the clock pulse width is added since the flip-flops used in the design sample data on the rising edge of the clock but do not transfer it to the output until the falling edge. Thus, the signals output on the falling edge of a clock pulse must complete their propagation through the combinational logic to the input of other flip-flops before the rising edge of the next clock pulse. The 350-nsec nominal clock pulse width is dictated by the minimum pulse width required by the slowest flip-flop at the -55°C temperature extreme. The 9040 flip-flop requires 340 nsec at -55°C., 240 nsec at 25°C., and 190 nsec at 125°C.

The figures of Table 6 indicate that for the worst case over the entire temperature range, with the load capacitance assumed, and with all circuits barely meeting their specifications, the machine will not run faster than about 700 kHz. These assumptions are all pessimistic, however, and operating speed can be increased by restricting any one of these variables.

The internal temperature of the spacecraft has not been specified as yet, and restricting it to less than -55°/+125°C. would increase the permissible clock rate. Related to this is the fact that if the lowest operating temperature were restricted to -30°C., the higher powered DTL circuits could be replaced with TTL circuits having the same power consumption and about twice the speed. However, the manufacturer specifies that interface between the low-powered DTL logic and the TTL logic is not satisfactory below this temperature.

The wiring capacitance assumed is probably excessive too, particularly for sophisticated packaging, but this could not be specified accurately without building a breadboard. Certainly, in LSI implementation the wiring capacitance will be greatly reduced at any rate.

It is also pessimistic, though good, conservative design practice, to assume that all circuits in a critical path will exhibit the maximum delay allowable under their specifications. If there are a large number of circuits in these paths, it is very unlikely that all of them will exhibit worst case delays. This will, in effect, happen when the system is converted to LSI since the speed of any path can only be specified from pin to pin of

TABLE 5

MAXIMUM CIRCUIT PROPAGATION DELAYS

(Load Capacitance = 30 pf)

		Tpd+(ns)	Tpd-(ns)
Fairchild 9040 flip-flop:	$-55^oC.$	190	310
	$+25^oC.$	150	250
	$+125^oC.$	170	300
Fairchild 9046 gates			9044/others
Fairchild 9047 gates	$-55^oC.$	110	110/80
Fairchild 9044 gates	$+25^oC.$	100	90/60
Fairchild 9041[a] gates	$+125^oC.$	85	118/75
Fairchild 9948 flip-flop:	$-55^oC.$	75	65
	$+25^oC.$	65	75
	$+125^oC.$	80	90
Fairchild 9937 gates	$-55^oC.$	50	42
Fairchild 9949 gates	$+25^oC.$	50	30
Fairchild 9963 gates	$+125^oC.$	80	35
Fairchild 9961 gates			

Tpd+ = propagation delay for output going positive

Tpd- = propagation delay for output going negative

[a]Active pull-up connected. For collector ORed operation with a pull-up resistor (Rc), add $\Delta t = (R_L - 1K)C_L \ \ell n \frac{V_{CC} - V_{OL}}{V_{CC} - V_{IH}}$ where

C_L = 30 pf (assumed)

V_{CC} = +5V

V_{OL} = 0.2V at $-55^oC.$ 0.2V at $+25^oC.$ 0.2V at $+125^oC.$

V_{IH} = 2.0V at $-55^oC.$ 1.7V at $+25^oC.$ 1.4V at $+125^oC.$

TABLE 6

WORST CASE PROPAGATION DELAYS FOR CRITICAL PATHS

	-55°C.		+25°C.		+125°C.	
Path 1	0/1	1/0	0/1	1/0	0/1	1/0
9948	75	65	65	75	80	90
9046	80	200	60	100	75	85
9961	50	42	50	30	80	35
[a]9041	80	256	60	145	75	119
9044	200	110	100	90	85	118
9949	42	50	30	50	35	80
9937	50	42	50	30	80	35
9961	42	50	30	50	35	80
9961	50	42	50	30	80	35
9949	42	50	30	50	35	80
9961	50	42	50	30	80	35
9949	42	50	30	50	35	80
9949	50	42	50	30	80	35
	853	1041	655	760	855	907
Pulse Width	350	350	350	350	350	350
	1203nsec	1391nsec	1005nsec	1110nsec	1205nsec	1257nsec
Path 2						
9948	75	65	65	75	80	90
9046	80	200	60	100	75	85
9963	50	42	50	30	80	35
[b]9041	80	292	60	174	75	140
9046	200	80	100	60	85	75
9046	80	200	60	100	75	85
9963	50	42	50	30	80	35
9963	42	50	30	50	35	80
	675	971	475	619	585	625
Pulse Width	350	350	350	350	350	350
	1007nsec	1321nsec	825nsec	969nsec	935nsec	975nsec

[a]Collector ORed bus driver connection with 5000-ohm load resistor.

[b]Collector ORed bus driver connection with 7500-ohm load resistor.

the LSI, and this will reflect an average of the actual circuits on the chip. In any event, it is possible, when ordering custom circuits, to narrow the range of the specification as long as yield and price remain reasonable.

In summary, the present design cannot be conservatively rated at over 700 kHz under the assumed conditions, which leads to a 23-microsecond instruction rate. However, there are enough factors as yet unspecified or unknown to assume that the final MULTIPAC can operate at about 1.0 MHz, with instruction times on the order of 16 microseconds. It is this figure upon which timing considerations throughout this report are based.

5.0 RELIABILITY

The reliability of the MULTIPAC system depends largely on its restructurability in the event of failure. Since the initial configuration makes use of all modules in the system, such restructuring to delete failed modules leads to a progressively simpler machine and consequently to a gradual degradation of the processing capability. Thus, to state the reliability of the system, or the probability that it will be operational after some specified period of time, one has to define the configuration under consideration. The initial configuration has a very low probability of lasting for the duration of a lengthy mission without requiring some restructuring for repair purposes. There is a high probability, on the other hand, that the minimal processing mode can still be implemented from the surviving modules after many months of system operation. Intermediate modes of operation have probabilities of survival between these extremes according to the module requirements, and hence, the processing capacities of such modes.

Figures 9 through 14 show the reliability models and graphically display the probabilities for the survival of each of three modes consisting essentially of three, two, and one processor(s) surviving for missions of 6, 12, and 24 months each. In addition to the number of processors which can be configured, a gradual degradation of the multiplexed connections to the experiments is also to be expected, and this is reflected in the probability of a certain percentage of the I/O lines remaining available in each of these modes. Each of the module failure rates reflects the conservatism that a single failure disables the entire module. For the memory modules, this is pessimistic.

5.1 Individual Module Reliability

The figures given were arrived at as follows: Underlying all calculations is an assumed uniform failure rate for all integrated circuits used in the design of 100×10^{-9} failures per integrated circuit-hour. This figure was chosen based upon the range (30×10^{-9} to 300×10^{-9}) encountered upon investigation and inquiry. The usual exponential failure distribution formula was used to determine the module reliability (MR).

$$MR = e^{-(PC)(FR)(730)(MO)}$$

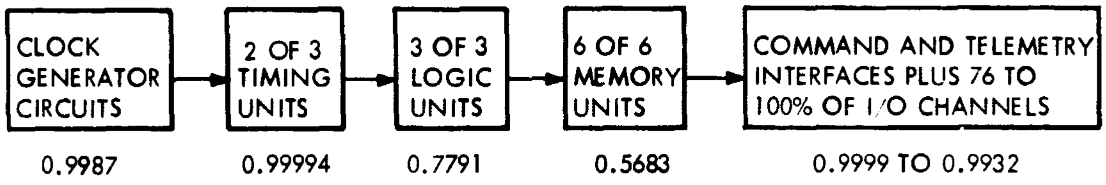
where

PC = parts count of integrated circuits

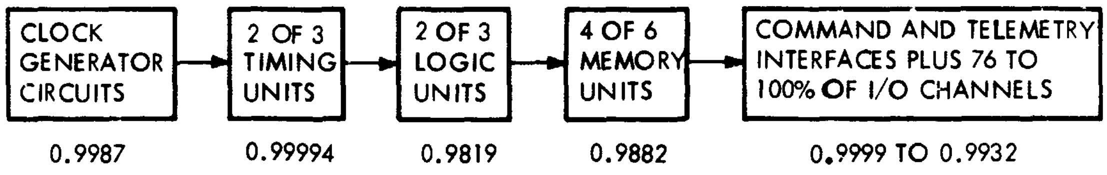
FR = failure rate = 10^{-7}

MO = mission duration in months

MODE 1: RELIABILITY = 0.44



MODE 2: RELIABILITY = 0.97 TO 0.96



MODE 3: RELIABILITY = 0. [illegible] TO 0.99

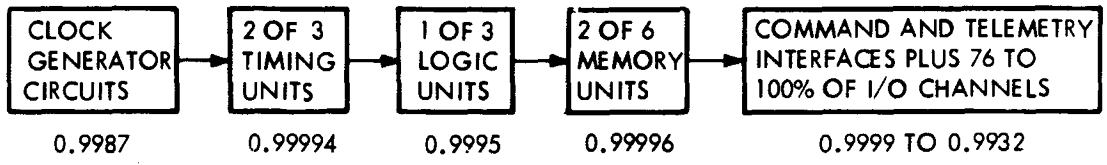


Figure 9. Six-Month System Reliability Models

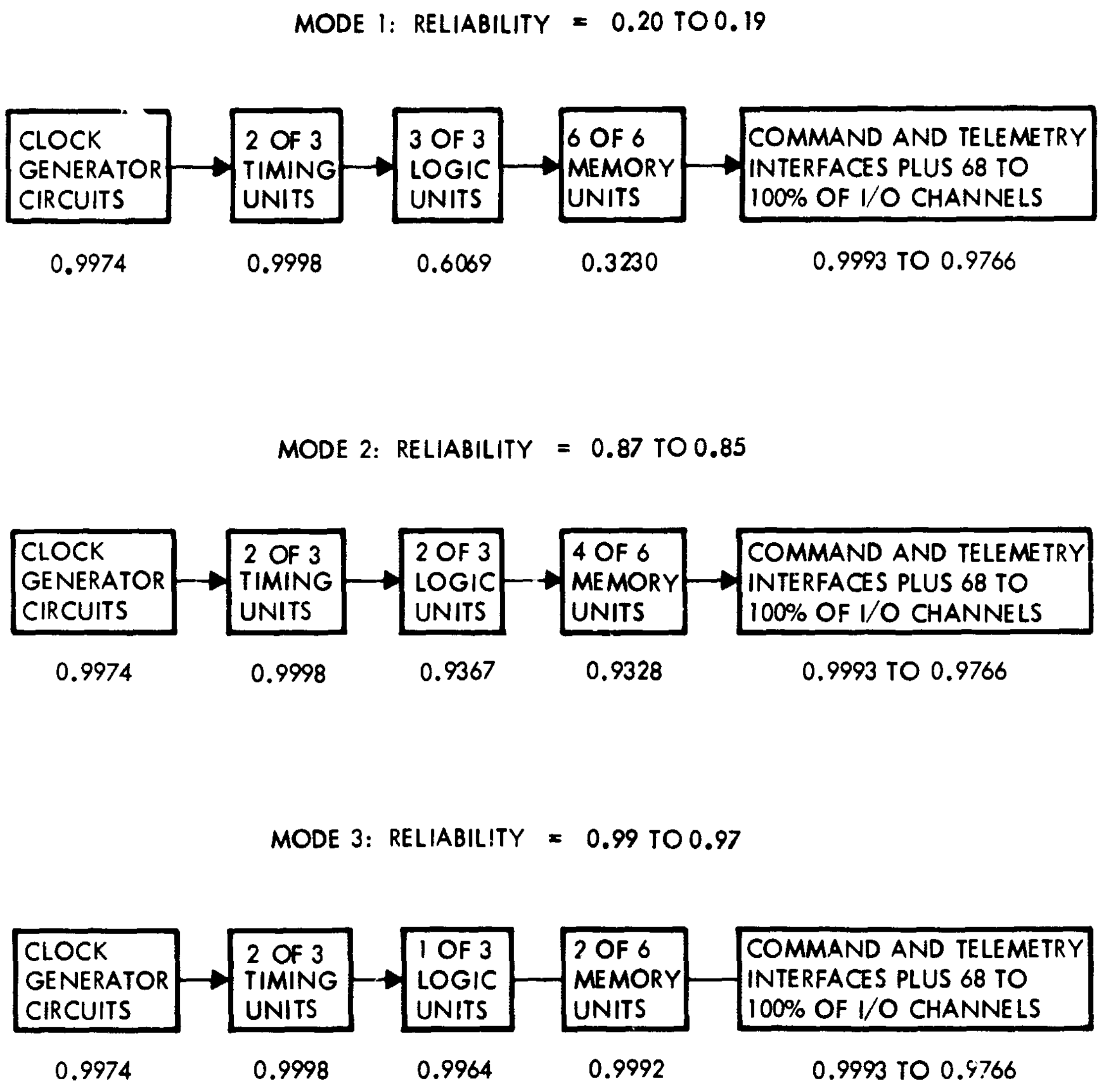


Figure 10. Twelve-Month System Reliability Models

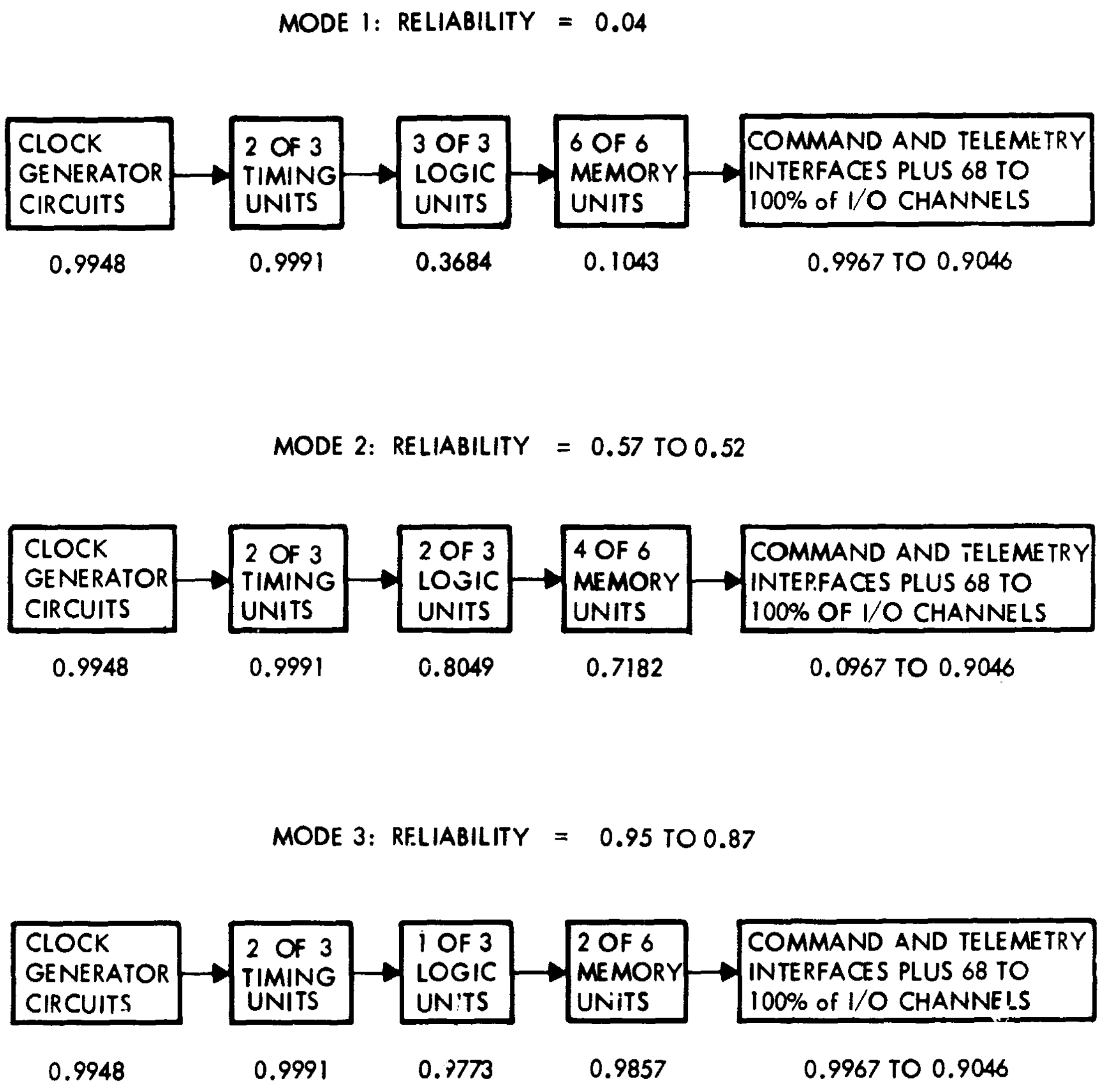


Figure 11. Twenty-Four Month System Reliability Models

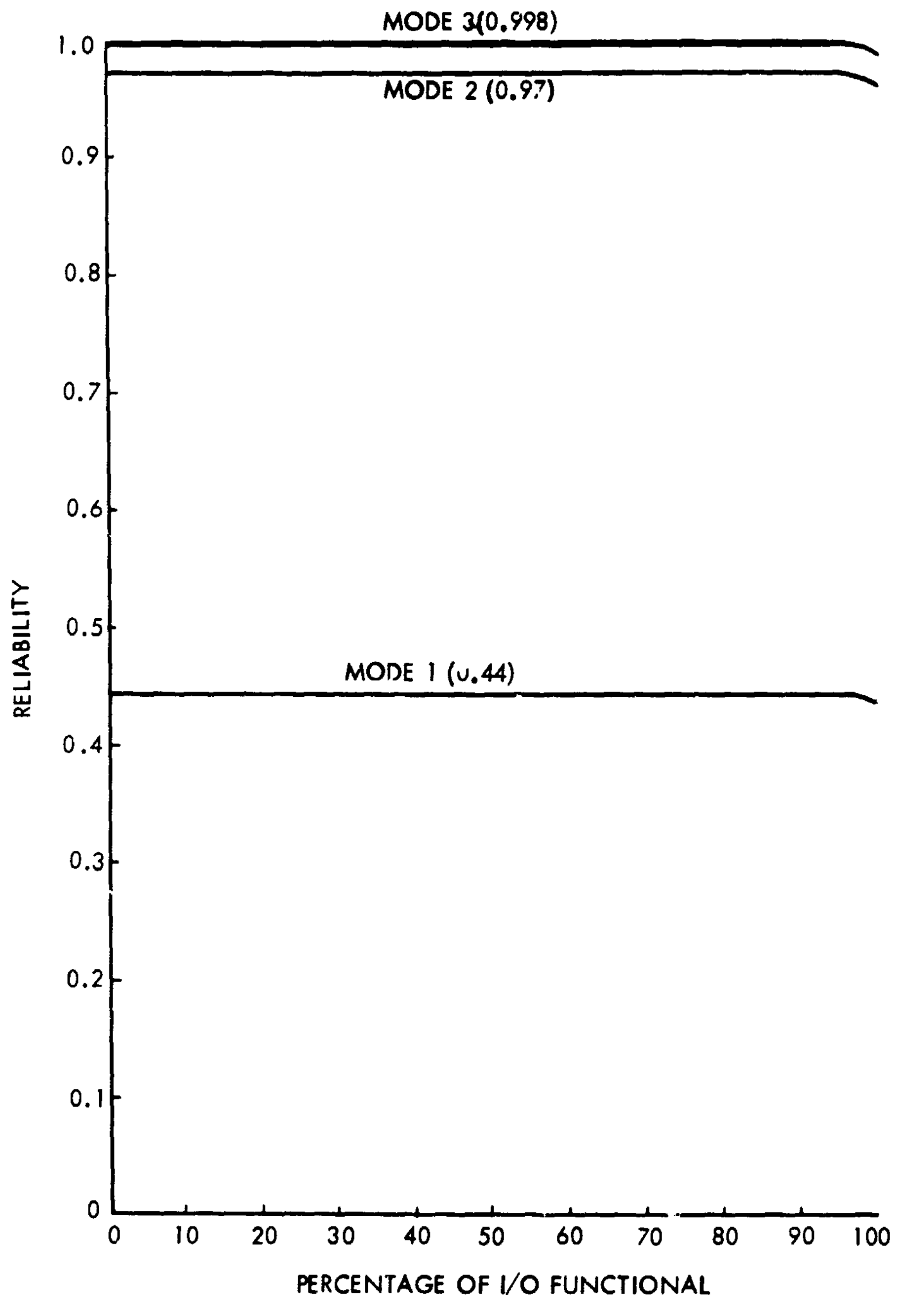


Figure 12. Six-Month System Reliability

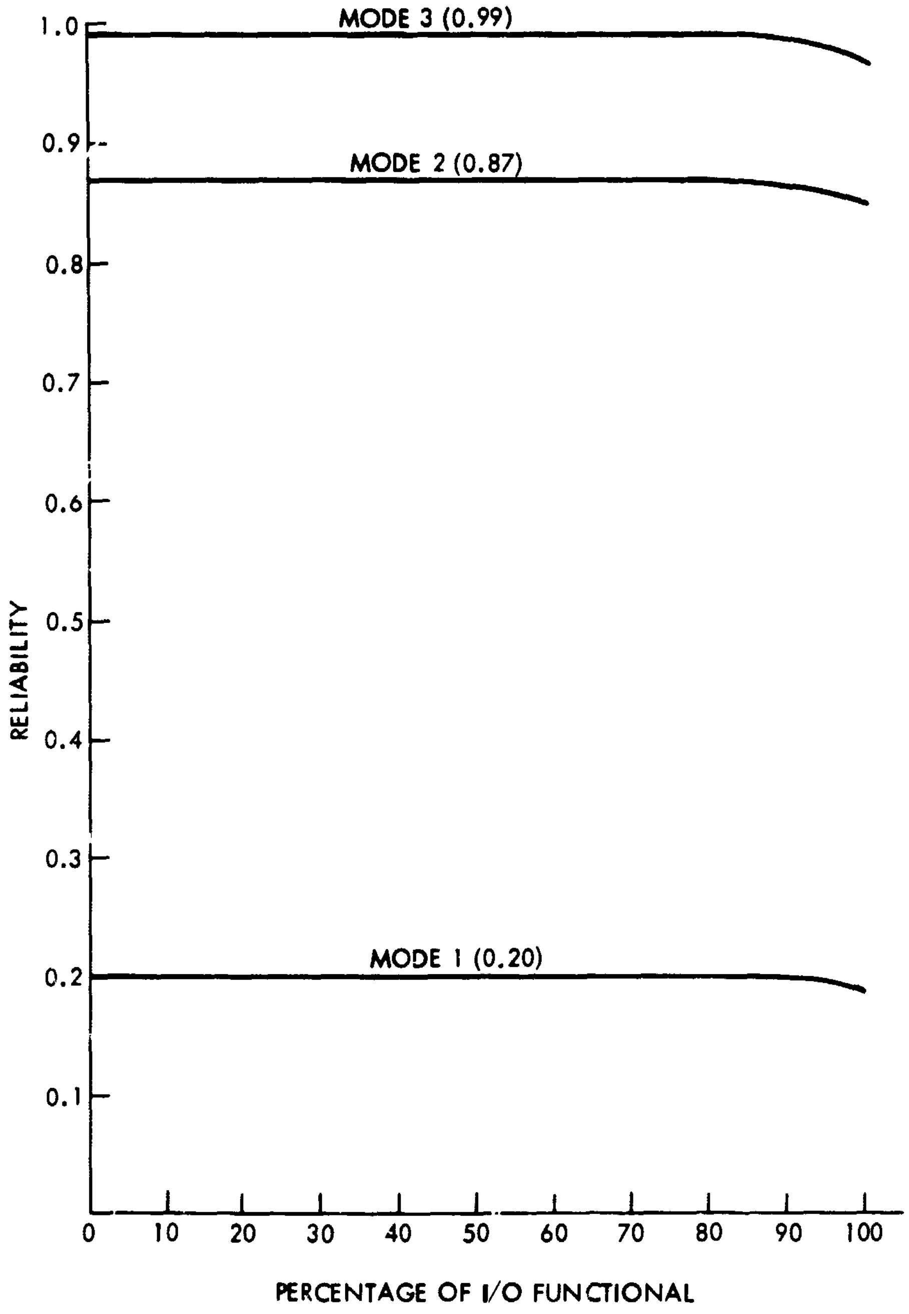


Figure 13. Twelve-Month System Reliability

Table 7 gives the resulting reliability figures for each module for mission durations of 6, 12, and 24 months. The reliability of the Timing Generator Unit is calculated on a somewhat different basis since it is itself modular and requires only two of its three timing units to be operational in order to function properly. Calculation of such probabilities is covered in detail in paragraph 5.2.

All IC failures were considered to cause complete failures at the modular level, which assumes that the modules fail completely upon failure of one IC within them and that they fail in such a way as not to affect the operation of any other modules. The detailed consideration of either of these exceptional cases would lead to a vastly more complicated analysis at the individual IC level and does not represent sufficient error to warrant such treatment at this time.

5.2 Reliability of Groups of Interchangeable Modules

Where modules are connected so that each of a certain type module is completely interchangeable with any other of that type, the only requirement for implementing a particular mode of operation is that some minimum number of modules required (MNR) of the original number (N) should survive. This is calculated as follows:

$$P = \sum_{NS=MNR}^{N} \frac{N!}{NS!(N-NS)!} (MR)^{NS} (1-MR)^{N-NS}$$

where

P = Probability of survival of $\geq$ MNR modules

N = Total number of modules

NS = Number of modules surviving

MNR = Minimum number of modules required

MR = Reliability of the module

The probability of survival for MNR or more of N of those modules which are completely interchangeable are given in Table 8. It should be noted that the memory reliability constitutes an especially conservative worst case estimate. Since the organization of the memory is such that 16 consecutive words are located on a chip, if the chip should fail internally without

TABLE 7

MODULE RELIABILITY

Module Type	Parts Count	Reliability		
		6-month	12-month	24-month
Logic Unit	190	0.9201	0.8467	0.7169
Address Switch	19	0.9917	0.9835	0.9673
Memory Unit	215	0.9101	0.8283	0.6861
Register	30	0.9869	0.9741	0.9488
D/A Register	26	0.9887	0.9775	0.9555
Command Unit	35	0.9848	0.9698	0.9405
Telemetry Unit	13	0.9943	0.9887	0.9775
Timing Generator:				
Clock Generator	3	0.9987	0.9974	0.9948
Timing Unit	10	0.9956	0.9913	0.9826

TABLE 8

RELIABILITY FIGURES FOR GROUPS OF INTERCHANGEABLE MODULES

Mission Time	Module Type	No. of Modules (n)	Minimum No. of Modules Required (MNR)	Probability of $\geq$MNR Surviving
6 months	Logic Unit	3	3	0.7791
			2	0.9819
			1	0.9995
12 months			3	0.6069
			2	0.9367
			1	0.9964
24 Months			3	0.3684
			2	0.8049
			1	0.9773
6 months	Memory Unit[a]	6	6	0.5683
			4	0.9882
			2	0.9999
12 months			6	0.3230
			4	0.9328
			2	0.9992
24 months			6	0.1043
			4	0.7182
			2	0.9857
6 months	Timing Unit (part of Timing Generator Module)	3	2	0.99994
12 months				0.99977
24 months				0.9991

[a]The assumption that a single device will cause complete failure is used here, which is unduly pessimistic.

affecting others, it would not appreciably affect use of that memory since 2032 of the 2048 words would remain intact and the 16 consecutive bad locations could be avoided by reprogramming. There is a very small probability of more than two or three of the 128 chips failing, and thus there is a very high probability that sufficient locations will survive to make the memory usable. Owing to the nature of the memory interconnections, however, special consideration must be given to failures at the point of interconnection which might disable a larger number of locations, and for this reason the memory will be treated for the worst case situation as serially dependent on each IC in it.

5.3 Reliability of the I/O, Telemetry and Command Interfaces

For modules which are uniquely connected in such a way that they are not completely interchangeable, a different analysis must be used. This is true for any modules connected through the address switch modules to the logic units. Their usability is conditional upon the survival of the specific switch or switches through which they are connected. Furthermore, it is assumed that all I/O connections will have parallel, redundant connections to more than one I/O channel on different registers. The question of their being operational, therefore, resolves itself as whether or not some specific device remains in contact with the processor(s) through one or more of its I/O connections, one or more of the registers and one or more of the address switches. The number of possible paths for the successful functioning of any group of I/O lines must include all workable combinations of the modules involved. Calculation of the probability of the occurrence of one or more successful combinations requires that they be reduced to mutually exclusive sample spaces, similar to Boolean minterms, before their individual probabilities can be calculated and summed to a probability for the total sample space representing success. This becomes a difficult matter involving a very large number of terms. A general approach would be to calculate the probability of occurrence of all possible combinations of modules failed and not failed, determine which terms represent success and which failure, and sum the probabilities of the success terms. This requires evaluating 2^N terms, however, where N is the number of modules. This amounts to 2^{38} terms in the present case, which would result in excessive computer running time.

For this reason a Monte Carlo sampling technique was used to evaluate the reliability of the I/O. In this technique a large number (e.g. - 10,000) of imaginary missions are flown using random numbers to determine if parts survived or failed. For each mission the program compares the reliability figure for each module with a different random number between zero and one and assigns the module a condition of survival or failure according to whether its reliability is greater or less than the random number. Thus, the frequency of occurrence of each such state of the machine (which modules have failed and which have not) in the samples is directly related to the reliability of the modules. Next, each sample state is tested against the set of equations, given in Table 9, which defines the criteria for successful functioning of the system. A specific set of connections at the I/O must be postulated here, so it is assumed that the inputs to R1 through R12 are analog comparator connections with the rest of the inputs digital. Each of the inputs and outputs are assumed connected to the same channel on two different registers for redundancy. The criteria consists of 19 equations. each giving the requirements for continued functioning of a specific set of I/O lines to the experiments and spacecraft, and two equations giving the requirements for the command and telemetry interfaces. Loss of either of the latter functions would mean complete failure, of course, but the frequency of occurrence of states in which a certain number of I/O lines are inoperative permits giving reliability figures not only for the entire I/O but also for portions of it. For example, the 24-month reliability of the entire I/O is 0.9046, but the probability of losing 24 or less lines, and thus retaining about 92 percent of the 288 lines, is 0.9790. The probability of losing 48 or less (retaining about 83 percent) is 0.9900. It should be noted that these losses proceed in increments of 11, 12, 23, 24, etc., because of the assumption that an entire register fails as a whole.

Tables 10, 11, and 12 give the results of such a program run for 10,000 samples. The Kolmogorov Theory may be applied to show that with 99 percent confidence, the results of cumulative sampling will be within

$$162.76 \times N^{1/2} = 162.76 \times \left(10^{[illegible]}\right)^{1/2} = 1.63\% \text{ of full scale}$$

of the theoretical value.

5.4 System Reliability

The series reliability models of Figures 9 through 11 show the reliability treatment of the system as a whole for the three modes of operation and Figures 12, 13, and 14 show these results in graphic form. The high order of reliability deriving from the restructurability of the MULTIPAC system is clearly demonstrated here since the minimal processing mode, Mode 3, demonstrates a 24-month reliability as high as 0.95, even under the consistently pessimistic terms of the analysis.

TABLE 9

CRITERIA FOR SUCCESS OF THE I/O, TELEMETRY, AND COMMAND INTERFACES

Number of I/O Channels Affected	Boolean Equation for Success
12 analog inputs	(DAØ)(RØ)(SWØ) + (DA1)(R6)(SW2)
	(DAØ)(R1)(SWØ) + (DA1)(R7)(SW2)
↓	(DAØ)(R2)(SWØ) + (DA1)(R7)(SW2)
	(DAØ)(R3)(SW1) + (DA1)(R9)(SW3)
	(DAØ)(R4)(SW1) + (DA1)(R1Ø)(SW3)
	(DAØ)(R5)(SW1) + (DA1)(R11)(SW3)
11 outputs	(RØ)(SWØ) + (R6)(SW2)
	(R1)(SWØ) + (R7)(SW2)
↓	(R2)(SWØ) + (R8)(SW2)
	(R3)(SW1) + (R9)(SW3)
	(R4)(SW1) + (R1Ø)(SW3)
	(R5)(SW1) + (R11)(SW3)
12 digital inputs	(DAØ)(SWØ+SW1) + (DA1)(S2+SW3)
12 digital inputs plus 11 outputs	(R12)(SW4) + (R18)(SW6)
	(R13)(SW4) + (R19)(SW6)
↓	(R14)(SW4) + (R2Ø)(SW6)
	(R15)(SW5) + (R21)(SW7)
	(R16)(SW5) + (R22)(SW7)
	(R17)(SW5) + (R23)(SW7)
Telemetry Interface	(TMØ)(SW4+SW5) + (TM1)(SW6+Sw7)
Command Interface	(CMØ)(SW4+SW5) + (CM1)(SW6+SW7)

Notes:

a. Failed module = 0, Operational Module = 1

b. Arithmetic operators are used in the normal boolean sense.

TABLE 10

MONTE CARLO RELIABILITY OF I/O, TELEMETRY, AND COMMAND INTERFACES (6 MONTHS)

Total Missions = 10,000

Total I/O Channels = 288

Number of I/O Channels Lost	Times Occurred	Cumulative Reliability	for	Percentage of I/O Channels Surviving
0	9932	0.9932 ± 0.0163		100.0
12	19	0.9951 ± 0.0163		95.8 or more
23	41	0.9992 ± 0.0163		92.0 or more
36	3	0.9995 ± 0.0163		87.5 or more
69	4	0.9999 ± 0.0163		76.0 or more
Failure of Telemetry or Command	$\frac{1}{10,000}$			

TABLE 11

MONTE CARLO RELIABILITY OF I/O, TELEMETRY, AND COMMAND INTERFACES (12 MONTHS)

Total Missions = 10,000

Total I/O Channels = 288

Number of I/O Channels Lost	Times Occurred	Cumulative Reliability	for	Percentage of I/O Channels Surviving
0	9766	0.9766 ± 0.0163		100.0
12	51	0.9817 ± 0.0163		95.8 or more
23	143	0.9960 ± 0.0163		92.0 or more
24	3	0.9963 ± 0.0163		91.7 or more
35	3	0.9966 ± 0.0163		87.8 or more
36	9	0.9975 ± 0.0163		87.5 or more
46	1	0.9976 ± 0.0163		84.0 or more
47	1	0.9977 ± 0.0163		83.7 or more
48	2	0.9979 ± 0.0163		83.3 or more
69	8	0.9887 ± 0.0163		76.0 or more
82	1	0.9988 ± 0.0163		71.5 or more
84	4	0.9992 ± 0.0163		70.8 or more
92	1	0.9993 ± 0.0163		68.0 or more
Failure of Telemetry or Command	$\frac{7}{10{,}000}$			

TABLE 12

MONTE CARLO RELIABILITY OF I/O, TELEMETRY,
AND COMMAND INTERFACES (24 MONTHS)

Total Missions = 10,000
Total I/O Channels = 288

Number of I/O Channels Lost	Times Occurred	Cumulative Reliability	for	Percentage of I/O Channels Surviving
0	9046	0.9046 $\pm$ 0.0163		100.0
12	189	0.9235 $\pm$ 0.0163		95.8 or more
23	533	0.9768 $\pm$ 0.0163		92.0 or more
24	22	0.9790 $\pm$ 0.0163		91.7 or more
35	13	0.9803 $\pm$ 0.0163		87.8 or more
36	39	0.9842 $\pm$ 0.0163		87.5 or more
46	36	0.9878 $\pm$ 0.0163		84.0 or more
47	11	0.9889 $\pm$ 0.0163		83.7 or more
48	11	0.9900 $\pm$ 0.0163		83.3 or more
58	1	0.9901 $\pm$ 0.0163		79.8 or more
59	1	0.9902 $\pm$ 0.0163		79.5 or more
69	42	0.9944 $\pm$ 0.0163		76.0 or more
70	1	0.9945 $\pm$ 0.0163		75.7 or more
81	1	0.9946 $\pm$ 0.0163		71.9 or more
82	1	0.9947 $\pm$ 0.0163		71.5 or more
84	18	0.9965 $\pm$ 0.0163		70.8 or more
92	2	0.9967 $\pm$ 0.0163		68.0 or more
Failure of Telemetry or Command	$\frac{33}{10,000}$			

6.0 PROGRAMMING

Several pieces of the required operating system have been coded (see Tables 13, 14 and 15) in order to achieve three goals: (1) get an idea of the timing involved, (2) discover difficulties in programming imposed by the design, and (3) develop useful techniques for circumventing various design limitations.

The results in this section point out a number of improvements for a future design which will be incorporated in the LSI (large-scale integrated circuit) version of MULTIPAC. These recommendations are discussed in Section 8.

For the purposes of this discussion, the memory was assumed to have a 16-microsecond cycle time. (See paragraph 4.6.)

6.1 Timing

The routines chosen represent the most important and most time consuming of the primary tasks of the CDS. Additional time must be allotted for the executive routine and the remaining tasks. The results for these tasks are summarized here.

	A/D Conversion (μsec)	Input (μsec)	Telemetry Buffering (μsec)
Time Equation	128n + 96	48n + 1680	256n + 496
n= 5 bits	736	1920	1776
n= 8 bits	1120	2064	2554
n=12 bits	1632	2256	3792

If the above tasks of A/D conversion, input, and telemetry buffering are the only tasks desired, then the basic time, not including interconnecting overhead for each device, for an average 8 bits per device would be approximately 8 msec/dev (assuming 500 used in interconnection overhead). At 6 msec/dev, 160 devices per second is saturation. That results in 1280 bits/second to the telemetry. To gain more speed, the process would have to be divided over several logic units. The rate can be at most doubled since over 40 percent of the time is soaked up in the telemetry buffering. Buffering time could be cut to a small overhead by simply

TABLE 13

A/D CONVERSION ROUTINE

k = mask for device in reg. R4

b = base reg. setting for register Rl, R2, R3, R4

```
          SBR     b,b          ; Set bases.
          COP     MD1,R2       ;
          k                    ; Get device mark.
          COP     MD1,R3
          0200                 ; Initialize single bit mask.
          XOR     R1,R1        ; Set R1 = 0
          IOR     R3,R1        ; Add to sum.
AD1:      SHR     R3,R3        ; Shift bit mask and wait for comparator.
          CIO     R1,R4        ; Read in comparator.
          TWM     R2,R4        ; Extract Result.
          CWM     R3,R4        ; Replace result in sum.
          TST     R2,11        ; Test for done. (R2 = 1?)
          JCN     MD1,MA1      ; Continue.
          AD1                  ; Done, R1 = sum.
```

Timing: overhead = 6 cycles = 96 μsec
loop = 8n cycles, = 8m μsec, where n = number of converted bits.
variable significance timing = 6 + 8n cycles
= 96 + 128n μsec

$n = 5 \Longrightarrow t = 736\ \mu\text{sec}$
$n = 8 \Longrightarrow t = 1120\ \mu\text{sec}$
$n = 12 \Longrightarrow t = 1632\ \mu\text{sec}$

TABLE 14

INPUTTING ROUTINE

```
READ:       COP       R,MA2        ; R contains device numbers.
            ADD       MD1,MA2      ; Start-up, find right prog. block
            DEVINP
            JMP       MD2,MA1      ; Jump to proper subroutine
            COP       R,MA2        ; Save dev. no. in MA2.
             .                       Routines are broken up by reg contain-
             .                       ing dev input line.
             .
DEVN:       SBR       B,B          ; B = base for R1, R2, R3, R4, = dev reg
            ADD       Md1,MA2
            MASK
            COP       MD2,MA2
            NBITS -   MASK
            COP       MD2,R3       ; R3/no. bits to read in
            XOR       R1,R1        ; R1/0
DEVN 1:     CIO       R3,R4        ; R3 has 0 in sign bit so input is done.
            TWM       R2,R4        ; Extract response.
            SRT       R1           ; Shift bit into cur. word.
            DCR       R3
            TST       R3,1
            JCN       MD1,MA1      ; Check if done.
            DEVN1
            TST       R1,4
            SRT       R1
            COP       MD1,R3       ; Done but word is left adj. in R1.
            12
            SBR       MD2,R3       ; R3/no. places to shift right
DEVN 2:     TST       R3,15        ; ≤ 0
            JCN       MD1,MA1      ; Get out if right adj.
            DEVNX
            CYC       R1,1
            DCR       R3
            JMP       MDi,MA1
            DEVN2
DEVNX:      COP       R1,MA2       ; Transmit result in MA2.
            JMP       MD1,MA1
            READX
             .
             .
             .
```

TABLE 14.-- Continued

INPUTTING ROUTINE

Timing:

overhead	=	21 cycles
read loop	=	10*n cycles ; n = no. bits to read
adj. loop	=	7*(20n) cycles; $n \leq 12$
Total Time	=	21 + 10n + 84 cycles
	=	105 + 3n cycles
	=	1680 + 48n μsec

n = 5	t = 1920 μsec
n = 8	t = 2064 μsec
n = 12	t = 2256 μsec

Range: $1748 \leq t \leq 2256$

TABLE 15

OUTPUTTING ROUTINE

```
; telemetry buffer, 11 bits + parity
; R1/word, R2/no. bits

TYO:        COP      MD1,MA2
            CPAR
            COP      MD2,R4         ; Cur. parity
            COP      MD1,MA2
            BCNT
            COP      MD2,R3         ; Bit count for cur. word
            COP      MD1,MA2
            OBUFI                   ; OBUFI/index into TBUF for cur. word.
            COP      MD2,MA2
            ADD      MD1,MA2
            TBUF
TYO1:       SHR      R1,MD2         ; Shift next bit in.
            XOR      MD1,R4         ; Sign of R4/parity.
            DCR      R3
            TST      R3,14          ; ≤ 0
            JCN      MD1,MA1
            TYO2
            TST      R4,4           ; Word full (even parity, x=5=odd parity)
            SRT      MD2            ; Shift parity bit into word.
            TST      ,7
            SRT      MD2            ; Put 1 in sign for telemetry.
TYO3:       COP      MD1,MA2
            TYOBS
            COP      MD2,R4
            SUB      MD1,R4
            TBUFL
            TST      R4,5           ; ≥ 0
            JCN      MD1,MA1
            TYO3                    ; Wait for word to be outputted.
            COP      MD1,R3         ; Set new bit count.
            COP      MD1,MA2        ; DCR OBUFI
            OBUFI
            DCR      MD2
            TST      R4,4           ; ≤ 0
            JCN      MD1,MA1
            .+3
            COP      MD1,MA2        ; Wraparound
            TBUFL-1
            COP      MD2,R4
            ADD      MD1,R4
            TBUF     :              ; Save new word ptr.
            COP      MD1,MA2        ; Only now can TYOBS be inc.
```

TABLE 15.-- Continued

OUTPUTTING ROUTINE

```
TYO3:       TYOBS
            INC        MD2
            COP        R4,MA2
            XOR        R4,R4           ; Parity = 0.
TYO2:       DCR        R2
            TST        R2,14           ; > 0
            JCN        MD1,MA1
            TYO1
            COP        MD1,MA2
            CPAR
            COP        R4,MD2          ; Save cur. parity.
            COP        MD1,MA2
            BCNT
            COP        R3,MD2          ; Save cur. bit count.

Time:       overhead      = 17  cycles
            basic loop    = 16n cycles, n = no. bits to shift in
            overflow loop = 28  cycles
            If overflow happens, every other call then.

            Total Time =  17 + 14 + 16n cycles
                       =  31 + 16n cycles
                  or t = 496 + 256n µsec

            n =  5 ⟹ t = 1776 µsec
            n =  8 ⟹ t = 2544 µsec
       but  n = 12 ⟹ t = 3792 µsec because overflow is guaranteed.
```

buffering one byte/word. However, this necessarily requires very large amounts of memory for buffering in order to avoid the overflow condition. In addition, the telemetry would be interrupting much more frequently because of the small bytes going out. This further degrades the effective speed of the logic unit being interrupted.

The CDS will be configured to operate three processors simultaneously. A logic unit, two memories and 1 to 4 registers will be configured to form a processor. There must be some form of communication among these processors for various reasons. The most obvious is, of course, communicating data. This could be done by a system consisting of a memory and a flag register which tells whether the memory is busy. Better and faster would be the ability of one processor to halt another long enough to transfer data into the other's data memory (or even the program memory). The latter is not possible and the former system is quite complicated if it is to be unambiguous. The system chosen is to use a bit in one of the processor's registers as a flag and tie this bit through the register in/out system to a register of another processor.

However, the worst problem due to simultaneous processing is the register (or base register setting) use conflict. The fact that I/O lines are tied to general purpose registers means that great care must be taken to separate the address space of the various processors. The input and output processors, of course, use a wide range of registers on a cyclical basis. The exact extent of the conflict can only be determined through an extensive programming effort.

6.2 Multiplication

Multiplication can be accomplished in reasonable time by implementing a 6-bit multiplier subroutine. This routine uses one operand as an index to jump indirectly through a table of addresses into a sequence of shifts and adds. This technique results in a multiply time of approximately 500 μsec. (To handle cases where the product is longer than twelve bits takes very much longer by a factor of three or four). Multiplying two 12-bit numbers together would require several milliseconds at least. The 6-bit multiplier routine is illustrated below. The initial four instructions jump through a table, MTAB (not shown), to one at 64 locations given by M000000 through M111111, where the 6-bit number following the M consists of the multiplier digits. The 44 of the 64 entries not shown are derived in a like manner.

```
;  RL/multiplicand
;  R2/multiplier, assumed to be 6-bits or less
;  R3/result
     COP  R2,MA2
     ADD  MD1,MA2 ;  get jump address from MTAB
      MTAB
     JMP  MD2,MA1 ;  jump to proper multiplier
     XOR  R3, R3  ;  R3=0
       .
       .
       .
M111111:            ADD      R1,R3
M111110:            SHL      R1
M011111:            ADD      R1,R3
M011110:            SHL      R1
M001111:            ADD      R1,R3
M001110:            SHL      R1
M000111:            ADD      R1,R3
M000110:            SHL      R1
M000011:            ADD      R1,R3
M000010:            SHL      R1
M000001:            ADD      R1,R3
M000000:            JMP      MD1,MA1
                    EXIT

Mi01111:            ADD      R1,R3
M101110:            SHL      R1
M010111:            ADD      R1,R3
M010110:            SHL      R1
M001011:            ADD      K1,R3
M001010:            SHL      R1
M000101:            ADD      R1,R3
M000100:            SHL      R1
                    JMP      MD1,MA1
                    M000010
                    .
                    .
                    .
                    etc.
```

6.3 Subroutine Call

Transmission of arguments to subroutines may be accomplished in several ways, depending primarily on the number to be sent. For one argument, the best way seems to be in the data memory MA register because this is independent of base register settings. For two to five arguments, the registers available at a fixed base register setting plus the MA should be used. After that point, arguments have to be stored away in memory. The largest problem of subroutine calling resides in the use of base registers. In general, a subroutine wants to use a fixed base register setting for both operands (i.e., B1 and B2) while it may be called from various places where different base register settings are in effect. The interfacing between processors which desire to use different sets of registers will prove to be painful and time consuming.

Subroutine calls are fairly easy for single level calls. One simply transmits the return address as an argument which is stored away in some appropriate place in the data memory by the subroutine. Recursive functions are certainly possible to implement but quickly become infeasible as the number of parameters grows. This is due primarily to the very cumbersome mechanism for indexing into tables (or lists).

6.4 Program Modification

Program modification can be done by remoting the action through a function residing in data memory. Briefly, the technique is to place the address of the instruction to be modified and the new instruction word in two registers and then switch the program stream to a program residing in data memory. The program in data memory saves the main program MA, loads the address argument into the main program MA, writes the word containing the instruction into the program MD, restores the old value of the program MA, and switches the logic unit back to the program memory. It thus wastes a lot of time in modifying a program and, in general, it makes more sense to keep program parameters in data memory as flags or switches (variable jump).

6.5 Addressable Registers

The largest single bind in efficient programming is due to lack of addressable registers. Nearly all routines use their registers in both fields of the instructions. This means that both base registers must be the same for the sake of time. However, this artificially limits the number of available registers to four. This can be alleviated partially in some cases by using the data MA as a general purpose register. One useful technique evolves naturally and easily. If a routine wants to continually update a word in data memory but still have it available for computation, the MA is set to the appropriate address and all references to the variable become references to the MD register. This is used in the telemetry buffering routine to great advantage.

6.6 Ground Software

Ground-based software must emphasize the ability to diagnose and reprogram in the shortest possible time. This primarily implies a large collection of preassembled routines using all the combinations of available hardware. This is, of course, impractical. What really is needed is: 1) a diagnostic generator which uses failure history to reduce its output, and 2) various organizations determined by sets of available units of the running software which are abstract in the actual units utilized. This latter means assuming some subset of units being available and writing the most efficient code for the situation. Parameters to each such encoding would be the actual unit numbers (switch addresses). This ability implies an assembler of only moderate complexity with relatively simple macro features.

No time should be wasted on any kind of compiler. Code simply must be as efficient as possible, which means only machine language coding.

A computer must be available at all times for reassembling programs. Some on-board problems, such as failures, will be solved only by having a programmer generate more code in real time. If possible, the computer should take care of transmission to the on-board system. Extremely desirable would be a diagnostic generator (as above) which checks out the system automatically when needed. There is probably no need of human analysis most of the time. This is more true of diagnostics than reprogramming.

7.0 REPROGRAMMING AROUND FAILURES

The reliability of the MULTIPAC system is achieved through its ability to reprogram around failures. This section describes some of the techniques used to accomplish this reprogramming.

It does not seem feasible at this time to fly enough memory to perform all the diagnostics and automatic reprogramming. A more realistic approach is to diagnose the error through the command and telemetry links; to re-assemble the program on a ground-based computer; and then transmit the new program to the spacecraft. In general, it will not be necessary to reprogram all of the memory.

A typical system will have three logic units and enough memories to have three complete processors operating simultaneously. A processor is defined here to mean enough programmable units to program one or more of the CDS tasks. The most likely division of work into these three processors will be: one responsible for inputting and outputting data; another responsible for telemetry buffering; and the third responsible for data reduction processing.

The discussion will be divided into failures of various units (e.g., registers, logic units, etc.).

7.1 Complete Failure of a Register

Most failures in a register will cause complete register failure. Data is moved in and out of the register serially. A failure in a flip-flop, input gating or output gating will cause all bits of the word to fail.

The major consequence of a register failure is the loss of the 12 input lines and 11 output lines. If these lines are not redundantly connected to another register, then this represents a permanent loss of data. In general, it is expected that enough registers exist so that science lines can be redundantly connected to two registers. Thus, the loss of a single register will not cause the loss of any science data.

In the case of those lines connected redundantly to another register, the I/O table in the I/O processor are simply changed to reflect the alternate register for the connection of the devices concerned. This programming task could be included in the I/O processing routines so that only a simple command to update these tables is necessary. However, since it takes very little time to update the tables directly from the command link, this latter approach is preferable in order to keep the CDS programs as simple as possible.

The more serious aspect of a loss of register is the reduction of available registers for a particular base register setting in the logic unit. The necessity to introduce base register switching in some of the tight programming loops will cause a very significant increase in running time for many processes. A second register failure is no worse than a single register failure, since the overhead in switching registers will be roughly the same as with the loss of one register. For example, if a process uses four registers efficiently, one register failure will introduce base register switching. This base register switch now gives the process a pool of seven registers. Thus, if two registers fail, the same base register switching will give the process six registers, which is still more than the four registers it initially needed. When the number of register failures has reached the point that two base register settings will not supply enough registers, then the process task should probably be dropped or the entire system structure should be reorganized. Extra program switching to three base register settings will sometimes preclude reasonable running times for the process.

Frequently, there exists a reprogramming solution to accomplish a process without relying so heavily upon available registers. Keeping process parameters in registers is usually the most efficient in time. Most often a loop which keeps constants in registers can be reoriented to retrieve from program (not data) memory each time they are needed and the result is faster than attempting to keep the constants in a register of a different base setting.

In our recommendations for future implementation of MULTIPAC, we recommend the elimination of base registers. The above problem associated with base register switching will then disappear. In this case register failures will generally only reduce the interface of the spacecraft as explained above and will not significantly change processing speed until a large number have failed.

7.2 Complete Failure of a Logic Unit

Any failure of a logic unit is likely to cause associative failure (due to failed logic), and some diagnostics should be attempted on a logic unit to determine to what use, if any, this unit can be assigned. In this discussion we will consider only the complete failure of the logic unit.

The failure of a single logic unit out of the three will primarily affect throughput. Two logic units should be able to supply more than the minimum required processing load of the central data system. It is the extra tasks, such as data reduction, which will be affected. Since initially all logic units will be working, it is reasonable to assume that as many data reduction tasks as possible will be added to the CDS program requirements to use up the MULTIPAC processing capability. In the event of a logic unit failure, a cut-back would then be made in the amount of data reduction, particularly at high telemetry rates. At low telemetry rates it is likely that no degradation in performance will occur.

7.3 Memory Failures

There are very few single failures in a memory which will cause a complete memory failure. However, multiple failures, particularly three or more, will tend to make it very difficult to use the memory. For example, programming around loss of every other word and one of the middle bits in every word may be more trouble than it is worth. In this case one would consider the memory totally failed. Complete failures and some of the more likely partial failures will be discussed in following paragraphs.

7.3.1 Complete failure.-- If more than six memory units are initially available, there will be very little overall affect, except for those data reduction algorithms which need large data stores. The computer on the ground will have to go over all programs which reference this memory and reallocate the storage If this memory contains programs, these programs will now have to be read into another memory and the program selection switch changed.

If the failure brings the total number of memories below two per logic unit, then it will not be possible to run three independent processors. The present organization of MULTIPAC requires a program memory and a data memory for each processor since it is difficult, if not impossible, to have data and programs in the same memory. This is further discussed in our recommendations for future implementation of MULTIPAC.

Enough memory failures will have to occur to bring the CDS below four operating memories before memory failures will prevent operating two processors simultaneously. However, once the system is reduced to four working memories, the amount of data reduction processing capability will be seriously limited since most of these algorithms tend to use large amounts of memory space.

7.3.2 Partial failures.--

7.3.2.1 Loss of the automatically incrementing address register:-- Any failure in the incrementing logic which prevents the automatic stepping through memory should preclude the use of this memory as a program memory. It would be much better to use this memory as a data memory than to try to program around the incrementing logic if it increments but not properly.

7.3.2.2 Complete loss of the memory address register:-- If this address register is completely failed, then the memory unit is completely failed. The memory data register can be used as a universal register.

7.3.2.3 Complete loss of memory data register:-- This is the same effect as loss of memory address register. The MA register can be used as a universal register.

7.3.2.4 Partial loss of addressable words:-- This can occur due to bits of the memory address register failing or an x or y decoder failing. If conveniently addressable segments remain, such as halves or fourths of memory, then the unit really behaves as a smaller memory. If the useful words are scattered throughout memory, then the memory can only be used as random temporary storage or for constants. In this latter case, its effect is very similar to the complete memory failures described above except that we can use the MA and the MD as universal registers.

7.3.2.5. Loss of a bit:-- The memory cannot be used for programs since there is no way to mask the effect. The use as data memory is limited, especially if the failure is intermittent or in the low order bits.

If the failure is in one of the high order bits, the unit could be used to store small data with some extra programming to mask off this bit. If the bit fails to a zero, only mask negative numbers are needed and vice versa for a failure to a one. If the failure is in the low order bits, it is necessary to shift the data word on every access to memory, which probably precludes widespread use.

7.3.2.6 Loss of a single word:-- This is a trivial problem for either program or data storage and can be easibly taken care of by the assembler in the ground-based computer.

7.4 REPROGRAMMING METHODS

Reprogramming may be accomplished from the ground by simply sending up a new section of code. Given any failure, the next most efficient program usually involves a complete reorganization. This entails approximately 1000 words being sent up. At 10 bits per second, this requires less than one-half hour to transmit. Total reprogramming (~6000 words) would take only 3 hours to accomplish. On the other hand, reprogramming from within simply cannot cope with this kind of reorganization. Specifically, any reprogramming requires changing code, hence a reassembly.

A relabeling process is not possible, primarily due to the use of base registers. Again we see another reason for eliminating base registers from this design. In addition, any failure other than a register will require a total reorganization of some routine in order to prevent inefficiency of running time and memory storage utilization. There seems to be no real use for reprogramming which can be accomplished on-board so that the command link transmission speed is the factor which determines the amount of reprogramming done in a given time.

A very critical area is the problem of determining "what" has failed whenever it becomes apparent that "something" has failed. A great deal of the failure detection is going to occur on the ground. It does not seem practical to put sophisticated detection programs on-board, since these generally take a great amount of running time and memroy. During the period when the spacecraft is not transmitting to ground, a reasonable amount of failure detecting can be run, but this does not detect failures that occur during the transmission period.

There seems to be no useful diagnostics which could be run on-board, since these are far more complex than failure detection. Consequently, the discussion in this section is concerned with how to diagnose from the ground once the failure is apparent on the ground. In most cases all activity will have to be stopped and all modules cleared so they will not conflict with the diagnosing process.

7.4.1 Diagnostic tests.-- These diagnostics should not be very extensive. Rather, they should be a short sequence of tests which are optimal for the length of the sequence. The main purpose of these diagnostics is to quickly get some confidence in most of the units which are not definitely known to be bad. If these tests fail to reveal trouble, then some more extensive tests may be run. There is a reasonable chance that the bootstrap and register tests could be included in every memory unit. The remaining tests will probably be too large. However, they might be stored on a tape. The time estimates involve transmitting everything from the ground, but tape could be used for such things. It is encouraging that even the worst case of numerous duplicate transmissions from the ground is not exorbitant in time. As units fail, the search is reduced because the unit no longer has to be considered. Looking at the problem in this light, it seems clear that the bootstrap test can always be done in quite reasonable time unless a large number of failures occur at once.

7.4.1.1 The bootstrap:-- The simplest possible program which generates feedback is necessary. This would probably consist of a sequence of words being sent back via telemetry, for example,

```
TEST      COP     MD1, TELI
          WORD1
          TST     TEL1, 11
          JCN     MD1, MA1
          . - 2
          COP     MD1, TEL1
          WORD2
          .
          .
          .
          JMP     MD1, MA1
          TEST
```

where the code is repeated for each word to be sent back. This program uses one memory, one logic unit and one telemetry register. If the desired sequence comes back some specified number of times, then a basic processor has been located. Otherwise, the program must be reassembled and retransmitted using some other combination of the three units. This can be done in an optimal manner with some analysis.

7.4.1.2 Register test:-- The next step is check out all registers. A basic processor exists but cannot be very useful unless a reasonable number of registers are alive. In addition, checking registers is at least logically simple. The program would be very similar to that above except that the words to be transmitted would be first copied into a register, then into the telemetry. This is a very basic test and can test all registers at once (i.e., with one simple program).

7.4.1.3 Creating a full processor:-- The next two steps are to perform an instruction test for all the modules used up to now and then to search for a good data memory. A full processor needs two memories, a logic unit, and one to four registers. The search for a good data memory can be performed on-board, since we have already determined through the instruction test that the selected modules work properly. This program should interact often with the ground so that unforeseen problems may be detected. At this time it will be more efficient to test all memories rather than just to look for a good one.

7.4.1.4 Test of remaining logic units:-- When the above is completed, the system is known to have at least one full processor plus a number of available memories and registers. The next test should be the complete checkout of the other logic units. This can be done completely on-board by the good processor monitoring the output of a standard program residing in a good memory. As soon as one unit checks out, it is turned off and the other unit is set to execute from the same test program (same memory unit).

7.4.1.5 I/O test:-- The final test is to check out all the I/O devices and generate a device number-to-register map for use by the I/O programs. Absolutely failed units should be reported to ground, although reprogramming would not usually be required. The other telemetry unit should be tested at this time if not tested previously.

7.4.2 Timing.-- Overall time for a complete set of diagnostics where everything is transmitted from the ground is on the order of 1-2 hours. This assumes an unlikely 2000 words of transmission and one hour of analysis. Running diagnostics from an on-board tape unit might appreciably reduce this time by reducing transmission time and allowing more lengthy self-checking routines to be run. Thus, it looks like the bad failure situation might take two hours of diagnosis and three hours of reprogramming. If very definitive diagnostics are desired, then the time increases, simply because they must be run for long periods of time. It is clear that intermittent failures will cause either long periods of diagnosis or living with intermittently bad data.

8.0 CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE IMPLEMENTATION

The MULTIPAC system will provide a small and low-powered spacecraft central processor with exceptional long-term survivability through graceful degradation of its processing capacity. Its feasibility has been established by a detailed logical design, using off-the-shelf integrated circuits, which provides a reasonably accurate estimate of the characteristics of the LSI model and a definite baseline for its design.

8.1 Conclusions

The conclusions which may be drawn from the MULTIPAC integrated circuit design are described in the following paragraphs.

8.1.1 Speed and power.-- It has proven feasible to design a machine based on the MULTIPAC concepts of restructurability and to keep within the maximum power budget of 20 watts utilizing standard off-the-shelf integrated circuits. The resultant machines is not as fast as was originally envisioned, however, since the amount of switching between modules and the relatively low speed of the low-powered circuits lead to long propagation delays in the serial data-flow paths. The basic serial clock rate at which such a machine could be operated is on the order of 1.0 MHz, resulting in a 16-microsecond instruction time. If similar bipolar circuits are used on the LSI chip, both speed and power should remain of the same order of magnitude in LSI implementation. Some gains can be expected owing to less wiring capacitance on the chip and the more accurate tailoring of circuits to meet but not exceed their drive requirements, but the degree of such improvement is difficult to predict without detailed investigation.

8.1.2 Weight and volume.-- The projected weight and volume of the IC packaging is well within the design goals, and the LSI version should, of course, exceed this by an order of magnitude.

8.1.3 Reliability.-- The survivability through reconfiguration principle has proved practical in design and does enable the MULTIPAC system to realize a high proability of survival as a functioning, though degraded, system over missions as long as two years. Probability of the system surviving with full capacity for two years is about 0.04, and the probability for the survival of two of the three initial processors is about 0.57, but the probability that a minimal functional processor can still be configured after two years is quite high, about 0.95.

8.1.4 Programming.-- Cursory programming estimates indicate that the proposed system of three processors can handle the processing load envisioned by the preceding study with some excess capacity. In degraded modes of operation caused by restructuring around failures, reduction of the maximum telemetry rate of 4096 bps will almost certainly be necessary. Further programming studies are necessary, however, to determine accurately the capacity of the system in different modes of operation.

8.2 Recommended Changes

Design and programming studies of the IC version of MULTIPAC has suggested improvements of an architectural nature that were too far reaching to introduce into the design once significant portions of it had been done in detail. Changes of this nature which are considered desirable in LSI model are described below.

8.2.1 Use of macroinstructions.-- MULTIPAC evolved from a processor in which each microinstruction was dealt with separately and was independent, so far as the hardware was concerned, from those preceding it and following it. Thus, its individual treatment by the instruction register had to be coded into it, and sequences of microinstructions were solely a matter of programming. As a more specialized system evolved, however, sequences of instructions had to be anticipated in the jump and interrupt hardware. This specialization could be extended to realize further efficiencies by recognizing other similar sequences of microinstructions, or macroinstructions, in the hardware. A number of these are listed in Table 16 as they will be implemented if all recommendations made herein are carried out.

8.2.1.1 Memory addressing:-- First among these would be multiple word instructions containing memory addresses. The sequencing hardware need only recognize these addresses and prevent their being treated as instructions. This would eliminate any need for the thirteenth bit of the present machine word.

8.2.1.1 Use of the program memory for data storage:-- It would be useful to be able to use the same memory for instructions and data, thus permitting more flexibility in the modes of MULTIPAC operation, including a single memory mode with a higher probability of survival than any presently workable configuration. This could be accomplished through macroinstructions if a program counter (PC) separate from the memory address register were incorporated. It should increment its own contents, delivering the result back to itself and at the same time transferring it to the memory address register. Jump instructions would require a control signal to alter the PC as well as the MA. Other addressing of the MA would override the input from the PC to the MA but would not affect the actual PC contents.

8.2.1.3 Interrupt:-- The mechanism of interrupting is cumbersome and needs improvement. This could be accomplished through a macroinstruction type sequence in which the input signal to the PC and MA would be forced to zero and a hard-wired control signal would cause the MD to copy the contents of the MA. This would initiate a normal write cycle to write the return address into location zero, and the PC would continue incrementing to location one where the first instruction of the interrupt subroutine would be located. Since macroinstructions cannot, in general, be interrupted in the middle, all such sequence counters should inhibit interrupt, and as soon as interrupt is allowed to take effect, the execution of any further instructions must be inhibited in order that no further macroinstructions are allowed to begin. The next instruction which the interrupt sequence counter should allow to reach the instruction register would be the first instruction of the interrupt subroutine. As in the present design, the return address stored would be too high because of the instructions which must be inhibited from execution. A correction of minus two would have to be made to the address before the return jump could be executed. Note that this method depends on being able to read data from the program memory, a change already suggested in the previous paragraph, since the return address is stored as data in location zero of the program memory.

8.2.1.4 Incorporation of a separate control module:-- The notion of macroinstruction control operating by shipping only microinstructions into the instruction register or inhibiting their transfer leads to a further possible partitioning of the logic unit itself. The latter is the largest module (excluding the special case of memory) and thus the one whose subdivision would result in the greatest increase in system reliability. Since there would be a minimal interface between the macroinstruction control hardware (including the interrupt control, the program selection switch, and the new program counter) and the rest of the logic unit, it is recommended that it be a separate module. One control unit, one logic unit, one or more memories, and several or more registers would thus function as a processor. The control unit would have an addressable register containing two fields specifying which program memory and which logic unit it would unite. This is actually a reversion to the original MULTIPAC concept in which a second logic unit performed this program transfer function.

The control unit should also have its own instruction register which would copy each instruction from the program memory at the same time it is being transferred to the logic unit instruction register. The control unit would thus independently decode those microinstructions which cue macroinstruction sequences and would itself then execute the macrosequences. These would be limited to transferring microinstructions from the program memory into the LIR or inhibiting such transfers, and to inhibiting the incrementing and transfer of the program counter. The control unit should also decode and execute those microinstructions acting entirely within it, such as loading the memory logic unit selection register and the interrupt masking register. The logic unit, on the other hand, would act only to execute individual microinstructions, as in the original MULTIPAC concept.

8.2.2 Elimination of base registers and two-level switching.-- The present address switching system has two distinct drawbacks which argue against its incorporation as is in the final LSI design. First, the two series bussing systems contribute toward limiting the operating speed of the machine since they are both necessarily composed of relatively slow circuits and lie in series with one another in the critical propagation path which limits the machine clock rate. Secondly, the use of base registers has proved undesirable from a programming point of view since there are not sufficient scratch storage registers available for programming without changing the base registers, and the latter operation costs heavily in the programming overhead.

8.2.2.1 Use of single address instructions:-- It is, therefore, desirable to increase the address fields from four to six bits in order to address all devices directly. The present machine requires 40 addresses, and the inclusion of three control modules would raise it to 43. Six-bit fields would provide an expansion capability of 21 addresses, a feature which is highly desirable in the system. Additionally, the use of macroinstructions suggested above would require at least five or six bits in the operation code field. It is therefore apparent that, if three fields are retained in the instruction, the machine word would have to be lengthened to 18 bits as follows:

OPC	B	A
6 bits	6 bits	6 bits

The above, of course, increases the power requirements and the instruction execution time by approximately 50 percent. The machine word could be reduced again to 12 bits, however, by reducing the number of address fields to one. The reason for the two-address instruction, if traced back, lies largely with the earlier use of one logic unit to transfer instructions into another logic unit. Normal data operations could function with only one address field if the logic unit contained two accumulators and a 1-bit field in the instruction referencing one or the other. The reason is found when one considers indexed or indirect addressing. These operations require a temporary storage and addition facility in the logic unit whose use will not destroy the contents of the regular accumulator, i.e., a second accumulator. A 1-bit field in the instruction word would specify the accumulator to be used:

Operation Code	Accumulator Number	Address
5 bits	1 bit	6 bits

8.2.3 <u>System operation with recommended changes</u>.-- Table 16 describes the operation of the MULTIPAC system if the changes suggested above were to be incorporated, i.e., it shows the use of single argument microinstructions to accomplish eight standard macroinstruction type operations. Execution time is on the whole faster than for a two-argument, 18-bit machine, especially for individual macroinstructions, which run faster by a factor of 2/3 owing to the shorter word length. The macroinstructions also run somewhat faster since there are never as many as 50 percent more microinstructions required. Relative execution times for a 12-bit machine compared to an 18-bit machine are given below. (This comparison ignores any extra overhead bits which may be required to accomplish machine transfers.)

	Number of 12-Bit Machine Cycles	Number of 18-Bit Machine Cycles	Relative Execution Time of 12-Bit Machine
(1) Single microinstruction	1	1	2/3
(2) Copy from Data Memory	3	3	2/3
(3) Copy Indexed from Data Memory	4	3	8/9

	Number of 12-Bit Machine Cycles	Number of 18-Bit Machine Cycles	Relative Execution Time of 12-Bit Machine
(4) Copy Indirect from Data Memory	6	5	4/5
(5) Copy Indirect and Indexed from Data Memory	7	6	7/9
(6) Copy from Program Memory	4	4	2/3
(7) Write in Program Memory	6	6	2/3
(8) Jump	3	3	2/3
(9) Interrupt	4	4	2/3

The specific macroinstructions listed in Table 16 may be changed or added to, or course. However, each requires a separate operation code and, to some extent, separate hardware in the control unit for its execution. Their number, therefore, should be limited. An attempt was made to modularize the macroinstructions using common sequences, but about half of those listed still required unique sequences and modularizing a sequence necessarily removed the interrupt protection over the whole. It is, therefore, recommended that the microinstruction order code contain a unique sensing instruction for each macrosequence as has been done here.

TABLE 16

LIST OF MACROINSTRUCTIONS

Macroinstruction Function	Contents of Program Counter	Contents of MA1	Contents of MD1	Contents of LIR	Inhibit Instruction Transfer	Inhibit Incrementing and Transfer of Program Counter	Inhibit Interrupt	Interrupt
(1) ADDRESS	0	0						
DATA	1	1	(0) = ADM[a],-,MA2					
MEMORY (ADM)	2	2	(1) = ADDR	CPG,-,MA2	1	0	1	0
			(2) = COP[b],ϕ,MD2	-NOP-	0	0	1	0
				COP,Φ,MD2	0	0	0	0
(2) ADDRESS	0	0						
DATA MEMORY	1	1	(0) = COP,1,R_{10x}					
INDEXED (IDX)	2	2	(1) = IDX[a],1,MA2	COP,1,R_{IDX1}				
	3	3	(2) = ADDR	APG,1,MA2	1	0	1	0
			(3) = COP[b],ϕ,MD2	-NOP-	0	0	1	0
				COP,-,MD2	0	0	0	0
(3) ADDRESS	0	0						
DATA MEMORY	1	1	(0) = IDR[a],-,MA2					
INDIRECT (IDR)	2	2	(1) = INDR ADDR	CPG,-,MA2	1	0	1	0
	3	3	(2) = COP,1,MD2	-NOP-	0	1	1	0
	3	3	(3) = OUT,1,MA2	COP,1,MD2	0	0	1	0
	4	4	(3) = OUT,1,MA2	OUT,1,MA2	1	0	1	0
			(4) = COP[b],ϕ,MD2	-NOP-	0	0	1	0
				COP,-,MD2	0	0	0	0
(4) ADDRESS	0	0						
DATA MEMORY	1	1	(0) = IID[a],-,MA2					
INDIRECT	2	2	(1) = INDR ADDR	CPG,-,MA2	1	0	1	0
AND INDEXED	3	3	(2) = COP,1 MD2	-NOP-	0	0	1	0
(IID)	4	4	(3) = ADD,1,R_{IDX}	COP,1,MD2	0	1	1	0
	4	4	(4) = OUT,1,MA2	ADD,1,R_{IND}	0	0	1	0
	5	5	(4) = OUT,1,MA2	OUT,1,MA2	1	0	1	0
			(5) = COP[b],ϕ,MD2	-NOP-	1	0	1	0
				COP,ϕ,MD2	0	0	0	0

a These instructions are decoded in the LIR as the microinstructions appearing there on the next line, but are also decoded as macroinstructions by the control unit to initiate the sequences of control levels which also commence on the next line. New microinstructions used are "Copy program stream to R." (CPG,-,R) and "Add contents of program stream to contents of accumulator N and transfer result to R." (APG,N,R).

b May also be any arithmetic or logical microinstruction (ADD, SUB, AND, etc.) or output (OUT). The latter will cause the accumulator contents to be written into the selected location.

c May also be any arithmetic or logical microinstruction but this sequence may not be used to write in the program memory. See macroinstruction No. 6.

TABLE 16.-- Continued

LIST OF MACROINSTRUCTIONS

Macroinstruction Function	Contents of Program Counted	Contents of MA1	Contents of MD1	Contents of LIR	Inhibit Instruction Transfer	Inhibit Incrementing and Transfer of Program Counter	Inhibit Interrupt	Interrupt
(5) ADDRESS	0	0						
PROGRAM	1	1	(0) = ADM[a],-,MA1					
MEMORY (APM)	2	2	(1) = ADDR	CPG,-,MA1	1	1	1	0
	2	ADDR	(2) = COP[c], ,MD1	-NOP-	0	0	1	0
	3	3	(ADDR) = DATA	COP,φ,MD1	1	0	1	0
				-NOP-	0	0	0	0
(6) WRITE	0	0						
PROGRAM	1	1	(0) = WPM[a],-,MA1					
MEMORY	2	2	(1) = ADDR	CPG,-,MA1	1	1	1	0
(WPM)	2	ADDR	(2) = OUT,φ,MD1	-NOP-	0	1	1	0
	2	ADDR	(ADDR) = PREV CONT	OUT,φ,MD1	1	1	1	0
	2	ADDR	(ACCφ)	-NOP-	1	0	1	0
	3	3	(ACCφ)	-NOP-	1	0	1	0
			(3) = NEXT INSTR	-NOP NEXT INSTR	0	0	0	0
(7) JUMP(JMP)	0	0						
	1	1	(0) JMP[a],-,MA1					
	2	2	(1) JADR	CPG,-,MA1	1	0	1	0
	JADR	JADR	(2) = DON'T CARE	-NOP-	1	0	1	0
			(JADR) = NEXT INSTR	-NOP- NEXT INSTR	0	0	0	0
(8) INTERRUPT	100	100						
	101	101	(100) = MICRO/END OF MACRO					
	102	102	(101) = DON'T CARE	MICRO/END OF MACRO	1	0	0	1
	0	0	102	-NOP-	1	0	0	0
	1	1	(0) = 102	-NOP-	1	0	0	0
			(1) = NEXT INSTR	-NOP- NEXT INSTR	0	0	0	0

a These instructions are decoded in the LIR as the microinstructions appearing there on the next line, but are also decoded as macroinstructions by the control unit to initiate the sequences of control levels which also commence on the next line. New microinstructions used are "Copy program stream to R." (CPG,-,R) and "Add contents of program stream to contents of accumulator N and transfer result to R." (APG,N,R).

b May also be any arithmetic or logical microinstruction (ADD, SUB, AND, etc.) or output (OUT). The latter will cause the accumulator contents to be written into the selected location.

c May also be any arithmetic or logical microinstruction but this sequence may not be used to write in the program memory. See macroinstruction No. 6.

APPENDIX A

DESCRIPTION OF LOGIC BY MODULE

A.1 The Timing Unit and Clock Signals

Clock signals required by the MULTIPAC system are a shift clock (SC) and a word strobe (WS), both deriving from a clock generator operating at a 1-MHz rate with a 350-nsec pulse width. (See Figure 15.) The timing and distribution system is triplicated and operates throughout the system through majority voting gates so that a single fault anywhere in the clock system can be absorbed without causing system failure.

This triplication is first evident in the timing generator itself. Three identical and independent counters are used to count out the number of SC pulses (15) before one is gated out and a WS pulse substituted. The three are kept in synchronism since they all hang up at the count of 15 and must be reset by a majority (two of three) coincidence of the WS signals. As soon as the second counter reaches ZERO, there will be such a coincidence, and all three will be reset to run in synchronism thereafter.

Throughout the MULTIPAC system, these two clock signals are distributed in triplicate (designated SC1, SC2, SC3, and WS1, WS2, WS3) and are decoded in each module by majority logic gates which require only two of the three to transmit the clock pulses in order to function properly. The A, B or C appended to the signal (for example, WS1A) has no logical significance and merely indicates different driver outputs for the same signal.

A.2 The Logic Unit and Switch Modules

The chief sections of the logic unit are the instruction register, the address-controlled switching and bussing structure, and the arithmetic section. In addition, there are two other small sections, one to select the source of program instructions for the logic unit and the other to control the special instruction sequences involving jumps or interrupts. The switch module is an extension of the logic unit switching structure which is shared by all logic units.

The logic instruction register (LIR) is decoded in three fields designated operation code (OPC), address A, and address B (refer to Figure 16, for the instruction format). During each word time the OPC field is decoded throughout the arithmetic section to provide the proper operations on the two serial data streams selected through the switching structure by signals

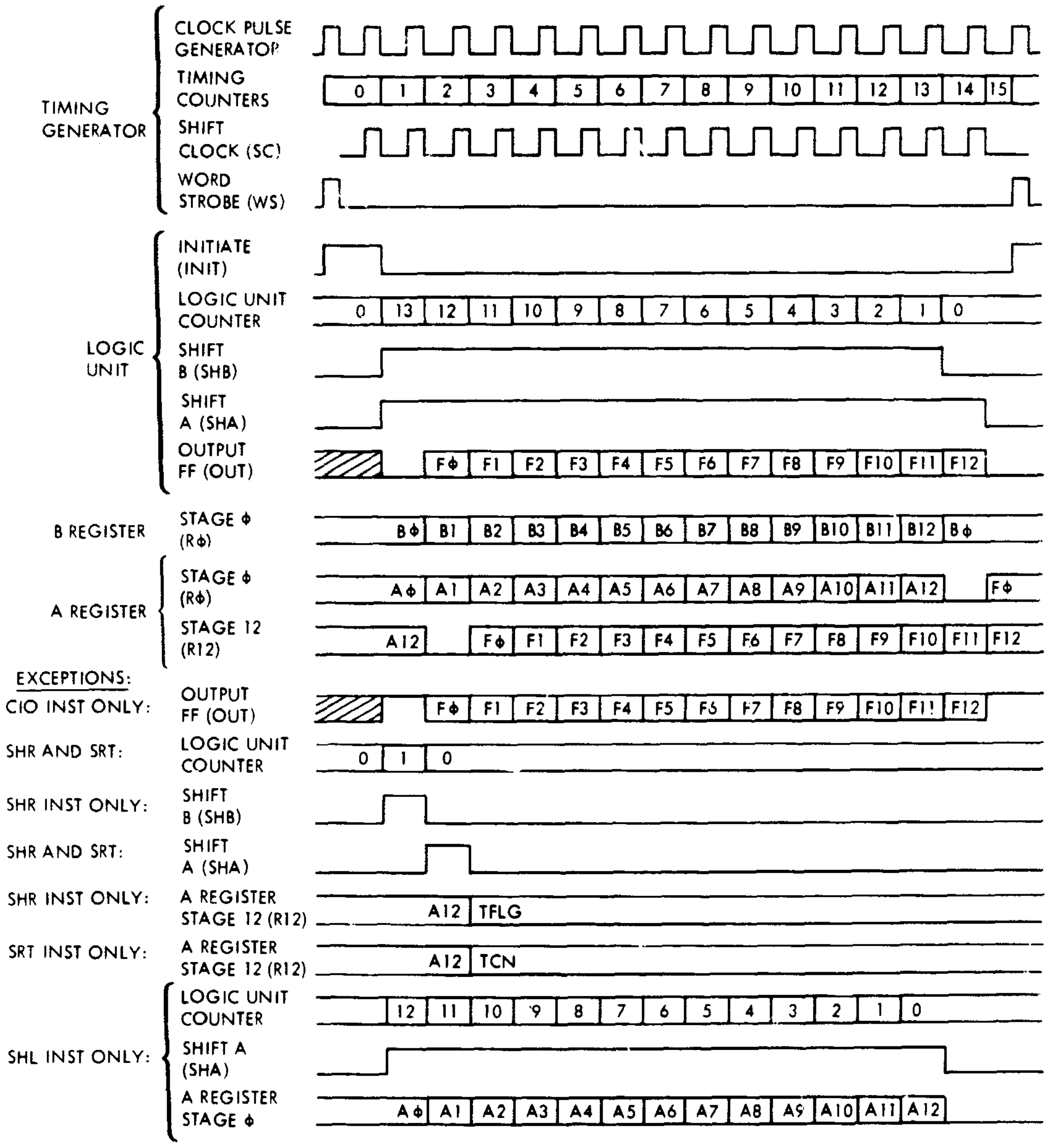


Figure 15. MULTIPAC Timing Diagram

INSTRUCTION FORMAT

LIR 12	LIR 11	LIR 10	LIR 9	LIR 8	LIR 7	LIR 6	LIR 5	LIR 4	LIR 3	LIR 2	LIR 1	LIR 0
OPC					B				A			

OPC =

	00	01	11	10
100	ADD	SUB	CIO	JMP
101	LOR	OPR	SBR	JCN
011	AND	TWM	TST	CYC
010	XOR	CTM	COP	SHR

OPC = 1111 (TST)
A =

	00	01		10
00	$\neq 0$	$= 0$	$\neq 0F$	$=0F$
01	<0	≥ 0	SET	RESET
11	>0	≤ 0		
10	$\neq 1$	$=1$		

OPC = 0101(OPR)
B =

	00	01	11	10
00	CMP	XCH	SHL	SRT
01		MSO	INC	DCR
11	SPS			
10	SIE			

Figure 16. Instruction Format and Karnaugh Mapping of Instruction Coding

decoded from fields A and B. In general, some function of the A and B signals is formed and returned to the register from which the A signal was obtained. Table 4, MULTIPAC Instruction List, lists the instructions executable by a logic unit.

A.2.1 Switching structure.-- The first 12 addresses (addresses 0 through 11) are addressed directly by the A and B fields. Both of these 4-bit fields are initially decoded two bits at a time, and a coincidence of a decoded output from one pair with one from the other pair determines the actual gates addressed in the switching. Two gates are selected by the B and three by the A field. The B field selects one gate which provides a logical zero level to the output control ($\overline{\text{OUTC}}$) of the register being addressed. This causes the register to cycle upon itself, thus retaining its contents but providing them as a serial data stream at its output. The second gate activated by the B field gates this serial data onto the B bus where it appears as one of the two inputs to the arithmetic section. One of the three gates selected by the A field performs a similar function by gating the output of the register to which it is connected onto the A bus and thus into the other input of the arithmetic section. The second gate provides a logic zero to activate the input control ($\overline{\text{INPC}}$) of the selected register. This causes the register to shift, thus providing its contents as a serial signal at the register output for input to the arithmetic section and at the same time shifting in data present at the register data input terminal ($\overline{\text{DIN}}$) to replace that previously contained in the register. The third gate selected by the A field connects the arithmetic section output signal (OUT) to the $\overline{\text{DIN}}$ input to provide the output data which is shifted into the register.

A.2.2 The switch modules.-- The highest four addresses (12 through 15) operate differently from the lower 12 described above. The signals and their effect on the register which is addressed are the same, but the register to be addressed is not determined by the A and B fields alone. Selection of one of the higher addresses extends all of these control and data signals through the second level switches which are common to all three logic units. Those selected are determined by the contents of the two base registers, BRA and BRB, which select second level switches for the A and B signals, respectively. (The A and B halves of each switch act independently so that the switches selected by BRA and BRB may be either the same or different.) The second level switches are merely extensions of the same selection gates and busses used in the first level of switching. They each select one of four devices based upon the two lowest order bits of the A and B fields of the logic unit using the switch. The two higher order bits of A and B must be ONES in order for the switches determined by BRA and BRB to be enabled and used so the switches, in fact, select addresses 12 through 15. Since each switch may be used by any logic unit, the lines decoding the two low order bits of A and B of each logic unit (DAϕ - DA3 and DBϕ - DB3) must be present at each switch. Which of the three sets of signals the switch will utilize is determined by connections from each logic unit called the select signals (ASEL and BSEL). These are activated only when a logic unit addresses that particular switch with its base register. Programming must avoid addressing the same switch from different logic units since two different sets of gates in a given switch structure might be enabled and the unrelated control and data signals would OR together on the switch busses to produce meaningless results.

A.2.3 Arithmetic section.-- The arithmetic section consists of a serial adder/subtractor/logic function generator, an overflow flip-flop for detecting a test condition, logic for controlling the shift and timing, and a register for holding one data word within the logic unit.

A.2.3.1 Arithmetic and logical functions:-- The output of the arithmetic section comes from the output flip-flop (OUT) into which some function of the two input variables (A and B) and the carry/borrow (C/B) is clocked. Arithmetic functions use the two's complement representation of negative numbers. Table 17 gives the logic equations for OUT as a function of A, B, and C/B for each instruction. Other variables would actually enter the mechanization equations of these signals but they become constants (ONE or ZERO) for any given instruction. Thus, by specifying the instruction, they are eliminated, along with any terms composed of A, B, and C which they gate out, to give the expressions which actually represent the signal transfers. A truth table of such enabling/inhibiting signals is given for the instruction set in Table 18. Figure 16, Instruction Format and Karnaugh Mapping of Instruction Coding, is useful in relating the decoders in the logic diagram to the truth table.

One such signal is LOG, which alters $\underline{C}$ for the logical instructions. In general, $C = C/B$ and the signals C and $\overline{C/B}$ are used as the carry and carry inverted, respectively, in the adder circuit. For logical instructions, however, C/B remains ZERO, keeping $\overline{C/B}$ a ONE while LOG forces C to a ONE. This effectively eliminates these signals from consideration in the adder and enables the decoding and ORing of all the minterms of A and B so that any logical function may be generated by selectively disabling the proper terms. This is done by INHD, INHE, and INHF.

A.2.3.2 Control and modification of transfer timing:-- The timing of these transfers is accomplished by the shift A (SHA) and shift B (SHB) flip-flops whose outputs are sent through the address switching to the two registers addressed by the A and B fields, respectively, where they control the shifting done by each. They, in turn, are controlled by the initiate (INIT) flip-flop and counter (Cϕ - C3). INIT is set by each word strobe (WS) and is reset again by the first shift clock (SC) pulse. The INIT signal is inhibited by a gate at the output if an invalid operation code is

TABLE 17

EQUATIONS OF LOGIC UNIT OUTPUT

ADD, INC: OUT $= (A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC)SC$

$C = C/B = (\overline{A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C}})SC$

$= (\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC)SC$

SUB, DCR: OUT $= (A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC)SC$

$C = C/B = (\overline{A\bar{B}\bar{C} + AB\bar{C} + A\bar{B}C + \bar{A}\bar{B}\bar{C}})SC$

$= (\bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC)SC$

XOR: OUT $= (A\bar{B} + \bar{A}B)SC$

AND, TWM: OUT $= (AB)SC$

C/B $= 0$

C $= 1$

LOR: OUT $= (A\bar{B} + \bar{A}B + AB)SC$

C/R $= 0$

C $= 1$

CMP: OUT $= (\bar{A})SC$

C/B $= 0$

C $= 1$

CTM: OUT $= (A\bar{B} + \text{TFLG}\cdot B)SC$

C $=$ C/B $= 0$

XCH: OUT $= (\text{LUR}\phi)SC$

C $=$ C/B $= 0$

TABLE 17.-- Continued

SRT:	OUT	=	(TFLG)SC
	C	=	C/B = 0
SHL,MSO:	OUT	=	(A)SC
	C	=	C/B = 0
COP,JCN, CIO,SHR, TST,JMP:	OUT	=	(B)SC
	C	=	C/B = 0
CYC,SBR, SPS,SIE:	OUT	=	DON'T CARE
	C,C/B	=	DON'T CARE

TABLE 18

TRUTH TABLE OF LOGIC UNIT CONTROL SIGNALS

				INHA1	INHA2	INHB(=OPR)	INHSA	INHSB	CAR	BOR	EOF	ID	TST1	INHD	INHE	INHF	LOG	TCL	TEN	RPS	RNS	TOF	CMPTST	ENTNV
0	ADD	B A		0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0		0
1	SUB	B A		0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0		0
2	XOR	B A		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
3	LOR	B A		0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0		0
4	AND	B A		0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0		0
5	COP	B A		0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
6	JCN	B A		0	1	0	1*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
7	CIO	B A		0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
8	SHR	B A		0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
9	CTM	B A		0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0		0
10	CYC	B N		0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
11	TST	B Ø	= 0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1
		B 1	≠ 0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
		B 2	< 0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1
		B 3	≥ 0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1
		B 4	> 0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	1
		B 5	≤ 0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1
		B 6	= 1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1
		B 7	≠ 1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	1
		B 8	= OF	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1
		B 9	≠ OF	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1
		B10	Set	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1
		B11	Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
12	OPR	Ø A	INC	0	0	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0		0
		1 A	DCR	0	0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0		0
		2 A	CMP	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0		0
		3 A	SRT	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
		4 A	SHL	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
		5 A	XCH	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0		0
		6 A	MSO	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
		7 A	SPS	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
		8 A	SIE	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
13	TWM	A B		0	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	1	1
14	SBR	N N		0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
15	JMP	A B		0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0

detected in the LIR. (Data words are transferred into the LIR together with the rest of the program stream but are distinguished from instructions since all the latter have dissimilar bits in the two highest order positions. This is an overflow condition for data.) In the general case, the INIT signal will set SHA and SHB and will dump the number 13 into the counter. It will then count down on each SC pulse, which also serve as the shift pulses at the registers, until on the counts of one and zero it resets SHB and SHA, respectively. This timing may be followed from Figure 15.

The control signals of Table 18 also modify SHA and SHB to alter the timing of transfers or inhibit the shifting of one or the other register altogether. INHSA and INHSB inhibit the setting of the SHA and SHB flip-flops respectively for those instructions indicated in the truth table. Other signals such as SHR, SHL, SRT, and CYC, corresponding to the same named instructions, alter the timing of SHA and SHB. CYC causes the contents of the A field (N) to be loaded into the counter instead of the number 13 and inhibits SHA. Thus, the B register cycles upon itself, shifting N times, while there is no A register defined. SHL causes the number 12 to be loaded into the counter instead of 13 and the contents of A are returned to itself. The resulting number of A shift pulses is 13, one short of the number necessary to reposition the contents of A, so these are left shifted to the left one bit with the original contents of OUT, namely ZERO, filling the low order bit as is required for the two's complement shift left. SHR alters the duration of SHA and SHB to one bit each, timed to transfer the low order bit of the B register into the high order bit of the A register as indicated in Figure 15. SRT treats the A register similarly but does not set SHB at all. Instead, it gates the test flag (TFLG) into OUT and hence into the high order bit of the A register. SHR is thus a shift right bit packing instruction from B to A, while SRT permits an arithmetic two's complement shift right by first testing the sign of A and then shifting the result (the sign bit) into the high order bit.

A.2.3.3 Initial and final states of output and carry/borrow:-- Before and after each transfer all inputs to OUT and CAR are disabled except for OUTCONT and CARCONT, which are enabled. These signals thus establish the state that these flip-flops will start at and return to. They serve to preset the carry for INC, DCR, TST = 1, and TST ≠ 1 instructions, and to switch OUT high on a CIO instruction.(See Figure 15 MULTIPAC Timing Diagram.) The timing level used to perform this switching is the OR of INIT and a decoding of zero from the counter. The decoding of zero alone would apparently serve the purpose, but it should be noted that it goes away on the leading edge of the first SC pulse, whereas the INIT signal remains until the trailing edge as required for proper timing of the switching level.

A.2.3.4 The test instructions:-- The test instruction (TST) is intended to ascertain whether some stated test condition, usually pertaining to the contents of the register addressed by the B field, is true or not and set the test flag (TFLG) accordingly. The test conditions utilize the A field for an extended operation code and they are arranged into complementary pairs which require only the inversion of the TFLG signal to, in effect, test for the complementary condition. This is accomplished by a second flip-flop (TNV) which inverts the output signal of the first for the complementary conditions when it is enabled by the ENTNV signal and set to a one by CMPTST.

The TCL signal clears the test flip-flop (TFF) at the beginning of each test instruction and the latter then scans the data under test during the timing level TEN. In testing for zero, the A signal is blocked and TFF is set by any ONE in the data, which then consists of only the B signal. For "equal to or greater than zero," TFF is strobed at a time when the sign bit is available ($\overline{\text{SHB}} \cdot \text{SHA}$) and is reset by RPS if the sign is positive. Likewise, for "equal to or less than zero" TFF is reset by RNS if the sign is negative. Testing for the number one is similarly accomplished by setting the carry initially to ONE with the TST 1 signal. With B a zero, the function transferred to OUT is an exclusive or of the carry and A. Initially set to ONE, the carry generates the number one and the exclusive or function will set TFF if there is not a comparison between CAR and A. The overflow test simply sets TFF with WS if OVF is a one; otherwise TFF remains cleared. The unconditional set and reset simply allow TFF to remain cleared. The set is treated as a complementary condition and makes the TFLG signal unconditionally a ONE.

The TWM instruction is intended to permit testing with a mask furnished by some other register. It thus operates just as the test for zero or one described above; however, the function generated in the adder is the same as the AND instruction, a logical AND between the contents of A and B. If any of the bits selected by ONES in A are also ONE in B, the flip-flop will be set.

A.2.3.5 The operate instructions:-- The operate instructions (OPR) either operate on the A register or load control registers in the logic unit itself, and they utilize the B field for an extended operation code.

Two of these instructions make use of the internal register of the logic unit (LUR 0-12). Since the bussing structure allows only one address to receive the logic unit output, the one addressed by the A field, an exchange between the data of A and B is not possible without enlarging the switching structure. Such an instruction was desirable, however, in order to implement interrupt, which must retain the original program address while inserting a new address. By putting an internal register in the logic unit, this is possible. XCH causes the contents of this internal register to be exchanged with the contents of the register specified by the A field. When executed, it blocks the adder, ORing the output of the internal register into the output signal as it shifts in the A signal at its input.

Another use found for the register was in locating the most significant ONE in the data for use in floating point operations. While executing this instruction, each ONE encountered in the A stream (the B input is blocked) causes the inverted contents of the timing counter (C0 through C3) to be dumped in the low order end of the register. The inversion is used since it is a down counter and an increasing count is desirable. The last ONE encountered (which is the most significant ONE in the data) is responsible for the last count dumped in the register, and it will be two larger than the bit position of the most significant ONE. This offset of two is unavoidable without additional hardware but can be compensated for in programming.

Three of the OPR instructions are used to set the contents of logic unit control registers. These include SPS and SIE, which set the program switch register (PSR) and the interrupt enable register (IEN) to the contents of the A field, and SBR, which sets the two base registers, BRA and BRB, to the contents of the A and B fields, respectively. All of these are accomplished by simple jam transfers using the word strobe (WS), and other actions are inhibited by inhibiting SHA and SHB from being set.

A.2.4 The program switch.-- The program switch selects one of six memories as the program source for the logic unit. It is controlled by a three-bit register which is loaded by program instruction. The numbers 0 through 5 are decoded from the register and the decoded levels each control two gates. One provides a level, timed by SHB, which causes the contents of the data register of the selected memory to cycle each word time. This signal also causes the memory address register to cycle and increment itself. The second gate connects the output of the data register onto the input bus of the LIR, thus providing it as a program stream to the logic unit. Two other inputs, from the telemetry units, have the power to override the first six. When either of these is activated, it immediately blocks all others, forcing them off the input bus, and the gate connecting the output data from that telemetry unit to the input bus is enabled.

A.2.5 The jump and interrupt sequencers.-- The action of the jump and interrupt instruction sequencers must be considered as one. This circuitry is necessary to control the unusual and potentially conflicting sequences of transfers that occur with these two instructions. Table 19 shows these transfers, starting with the jump.

TABLE 19

JUMP AND INTERRUPT SEQUENCES

MA1	MD1	LIR Input Bus	LIR	ISQ1/ISQØ	JSQ	Interrupt Inhibited	Jump Inhibited
0	--	--	--	--	--	--	--
1	(0)	(0)	--	0 0	0	0	0
2	(1)	(1)	(0)=JMP MD1,MA1	0 0	0	1	0
JADR	(2)	--	(1)=NOP JADR	0 0	1	1	0
JADR+1	(JADR)	(JADR)	--	0 0	0	0	0
JADR+2	(JADR+1)	--	decoder inhibited	1 0	0	0	1
JADR+3	(JADR+2)	--	XCH MA1	1 1	0	0	1
INT	(JADR+3)	--	--	0 1	0	0	1
INT+1	(INT)	(INT)	--	0 0	0	0	1
INT+2	(INT+1)	(INT+1)	(INT)1st interrupt inst	0 0	0	0	0
			(INTERRUPT SUBROUTINE)				
N							
N+1	(N)	(N)					
N+2	(N+1)	(N+1)	(N)=XCH R1	0 0	0	0	0
N+3	(N+2)	(N+2)	(N+1)=SUB MA1,R1	0 0	0	0	0
N+4	(N+3)	(N+3)	(N+2)=NOP 3	0 0	0	0	0
N+5	(N+4)	(N+4)	(N+3)=COP R1,MA1	0 0	0	0	0
JADR	(N+5)	(N+5)	(N+4)=NOP	0 0	0	0	0
--	(JADR)	(JADR)	(N+5)=SIE M	0 0	0	0	0
--	--	--	(JADR)=NEXT INST IN MAIN PROGRAM	0 0	0	0	0

A.2.5.1 The jump sequence:-- When a jump instruction occurs, either a JMP or a JCN in which the condition for jumping is met, interrupts are inhibited while the jump is being decoded in the LIR and the JSQ flip-flop is also set. JSQ remains set for one word-time after the actual jump instruction and inhibits interrupts during this time as well. Since interrupts are strobed by WS, which occurs at the end of each period, the sequence of Table 19 shows the shortest sequence in which an interrupt can follow the disabling action of a jump. Note also that during the time JSQ is set, the input bus to the LIR is forced to zero. This prevents the instruction in the next location following the jump address (JADR) from being executed.

A.2.5.2 The interrupt sequence:-- There are four interrupt lines to each LIR. Those from different LIR's should be paralleled so that any LIR may assume any task requiring any one of them. When assigned such a task, the LIR should be enabled to respond to a particular interrupt by placing a ONE in the mask loaded in the interrupt enable register (IEN). When thus enabled, and not otherwise inhibed by the jump sequencer, the interrupt signal will be strobed by WS to start the interrupt sequencer. This immediately disables the instruction decoder, thus eliminating the possibility of another jump sequence getting started, and it then proceeds to step through the sequence during which it blanks three more instructions on the LIR input bus in order to allow the interrupt sequence itself to time out. The state 10 of the sequencer is decoded to inhibit the normal dump of the shifting register (LSR) into LIR, and instead to dump in a wired XCH instruction. The A address field of the XCH is wired in from the program switch register (PSR), and is the address of the memory address register of the program memory, MA1 in Table 19. Thus, an exchange takes place between the contents of MA1 and the contents of the logic unit register (LUR). The latter must contain the starting address of the interrupt subroutine (INT). The contents of the location appear in the LIR just as the interrupt sequencer finishes its inhibiting action on the program stream.

The interrupt sequence also clears the IEN register in order that no further interrupts may occur during the interrupt subroutine, since this would lose the address for returning to the main program and substitute some address from within the interrupt subroutine itself, resulting in an endless loop.

The sequence for returning to the main program is also shown in Table 19. The return address held in LUR is larger by three than the correct return address, so it must be brought out into another register (R1) where three can be subtracted from it. Then, a normal COP instruction is used to set this address in MA1 since if a JMP were used the instruction which follows would be blanked. This last instruction is used to load the IEN register with the mask M, thus permitting further interrupts to take place.

A.3 The Memory Unit

The memory module consists of a storage medium (a complementary MOS LSI storage array) interfaced to the rest of the system by two registers somewhat similar to those found throughout the MULTIPAC system. These are the memory address register (MA) and the memory data register (MD). Both are directly addressable by the logic unit(s). When the MA is addressed in order to change it, a read cycle is automatically initiated which takes place during the next word time, and at the end of this period the data contained in the new address is dumped into the MD. Thereafter it can be obtained by addressing the MD from the B field, i.e., as the source of data. If the MD is addressed by the A field, i.e., to store new data in it, a write cycle is automatically initiated to take place during the following word time.

A special control signal ($\overline{\text{PCON}}$) coming from the program switch(es) of the logic unit(s) causes the MA to cycle upon itself through incrementing logic and a read cycle to be initiated each word time. The MD is also affected as if it were being addressed from the B field each time and its data is cycled to provide output of the contents of each memory location. This constitutes the program stream which is copied by the LIR. This mechanism is overridden if the MA is addressed by an LIR in the normal manner. This permits jumps to take place.

A.3.1 The storage medium.-- The storage device proposed is a modification of the 256-bit complementary MOS memory chip developed by Westinghouse under NASA contract number NAS-5-10243. The modification required consists of those measures necessary to permit use of the 16-word, 16-bit memory, which has been designed on a single chip, as a constituent part of a larger multiple-chip memory. Such expansion requires a method of addressing a selected chip from among others, preferably a coordinate-select method, and a method of ORing the bit lines from each chip without extensive external circuitry. Either of these functions could be carried out in external circuitry but would require a larger number of extra gates and interface circuits to do so, whereas a slight modification of the chip would allow them to be accomplished by simple bussing interconnections of the memory chips themselves.

The logic diagram of the memory unit shows one possible such modification requiring only four gates and two additional external connections per chip. These modifications would permit x-y selection of the chip, using only 24 interfact circuits instead of the 128 otherwise required, and wired-OR operation of the bit lines. The latter would be high impedance from all but the selected chip during read, and high impedance from all chips during write. In this condition the write drivers are enabled to drive the bit line

A.3.2 The memory address register.-- The memory address register operates very much like any other register in the machine except for the incrementing logic. As in any other register, there is an input control ($\overline{\mathrm{AINPC}}$) and an output control ($\overline{\mathrm{AOUTC}}$) which are operated by the LIR(s). When the latter is pulled to ZERO level, the MA shifts and its output is fed back into the input so that the same data remains in the register after the operation. When $\overline{\mathrm{AINPC}}$ is pulled low, however, the data at the input (AIN) is fed into the register. In addition to the two gates which feed these signals into the input, there are two others, however. They are enabled by the program control signal ($\overline{\mathrm{PCON}}$) and together they form the exclusive-or of the MA output and the incrementing carry flip-flop (INC). The latter is set to a ONE at the beginning of each word time by WS and continues to be set as long as the carry condition (MA0)(INC) = 1 persists.

When either $\overline{\mathrm{AINPC}}$ or $\overline{\mathrm{PCON}}$ are pulled low, it indicates the address is being changed. These conditions are used to gate the dumping of the new address from the shifting register (MAS) into the holding register (MA) and to set the read flip-flop (RD). The latter, in turn, first enables the select line in the MOS memory and, after allowing one word time for the gating to settle, enables the next WS pulse to strobe the memory output into the MD register. The holding register is necessary on the MA since while the read cycle is taking place the address in MA must not be altered. However, a new address may be shifting into MAS at this time.

A.3.3 The memory data register.-- The MD operates as any other register in regard to the input ($\overline{\mathrm{DINPC}}$) and output ($\overline{\mathrm{DOUTC}}$) control lines except that $\overline{\mathrm{DINPC}}$ also sets the write flip-flop (WR) for the word time following that during which data is shifted in. This enables both the select and write lines in the MOS memory and results in the data being stored in the location addressed by the contents of the MA. There are also jam gates for loading the output word of the MOS memory into the MD under control of the read signal as described above. It should be noted that this action is inhibited if the MD is addressed immediately after the MA has been addressed. This is necessary to prevent the read cycle from overwriting the data which has just been shifted into MD and is intended to be written into the location.

A.4 The Register and D/A Register

The MULTIPAC Register serves the multiple purposes of accumulator, index register, temporary storage, etc., and provides the I/O interface for the machine as well.

As a serial register, it may be addressed by either the A field or the B field of an LIR. The latter pulls $\overline{\mathrm{OUTC}}$ to logical zero and simply causes the register to cycle, shifting its output back into its input but at the same time providing this serial data on an output bus (DOUT). When the register is addressed by the A field, however, it is the $\overline{\text{INPC}}$ signal which is pulled low. This gates the register input (DIN) bus into the shift input and causes the serial stream gated onto this bus by the LIR to be shifted into the register and stored there.

The register provides the machine I/O interface by means of its response to a copy I/O (CIO) instruction. Normally, the DIN signal is forced to logical zero after each transfer by the logic unit. (See Figure 15.) After a CIO instruction, however, this signal is forced to logic one. Following each transfer, there is a word strobe (WS) which will either dump the input interface (ICHØ - ICH12) signals into the register or will dump the contents of the register into an associated output register if DIN is a ONE at that time, depending on whether the high order bit (R12) is a ONE or a ZERO. A CIO instruction transferring data into the machine should transfer all ZEROES into the register. This will clear the register, after which, gated by a ZERO in R12, WS will set to a ONE all stages for which the input interface signal is a ONE. An output transfer should come from a memory location in which the output image is stored and in which R12 is a ONE. The image is modified by masking instructions to alter the appropriate bit, then transferred to the register by a CIO instruction. Gated by the ONE in R12, the WS pulse following the transfer will dump the updated output word into the output register.

Following each CIO instruction, an interface transfer clock pulse (TCLK) is created by two flip-flops (TCLK1 and TCLK). For each serial input or output channel there must be a separate transfer enabling channel assigned to gate the shift register in the peripheral equipment into its sending or receiving mode, as the case may be. Then upon receipt of each TCLK pulse, it will shift one bit, either shifting the new bit from an output CIO into its register or, for an input CIO, shifting the next bit to its output stage to be sampled by the next input CIO.

The MULTIPAC D/A Register is simply a register as described above except lacking the output register. Instead, a conventional D/A ladder circuit is connected directly to the register outputs.

A.5 The Command and Telemetry Units

The Command and Telemetry Units are two separate modules; however, they share the same addressable connections at the logic unit. The Command Unit is connected to the input of logic unit and must be addressed by the B field. The Telemetry Unit is connected to the logic unit output and must be addressed from the A field. The latter cannot serve as an input, even on instructions which use an input addressed by the A field. These units provide interfaces for receiving up-link commands from the command receiver and for outputting down link data to the telemetry modulator, respectively.

A.5.1 Command unit.-- The Command Unit has two functions, a passive one for receiving normal command signals and an active one for taking over the processor(s) for reprogramming. Both commence with signals received from the command receiver. The latter is required to assemble a 16-bit word, 13 bits of data to be transferred into the MULTIPAC system plus a 3-bit address determining its destination within the processor. It must also decode from the command format such information as which one of the two command input registers it will utilize, and might also address other parts of the spacecraft directly through the same addressing scheme if desired. Within the processor, however, the three bits address the three logic units and four of the registers. The eighth address is used to set a flag indicating that the data is not to be transferred immediately but is to be held in the command input register (CIR) until read out by the program. This is the method for transferring normal commands.

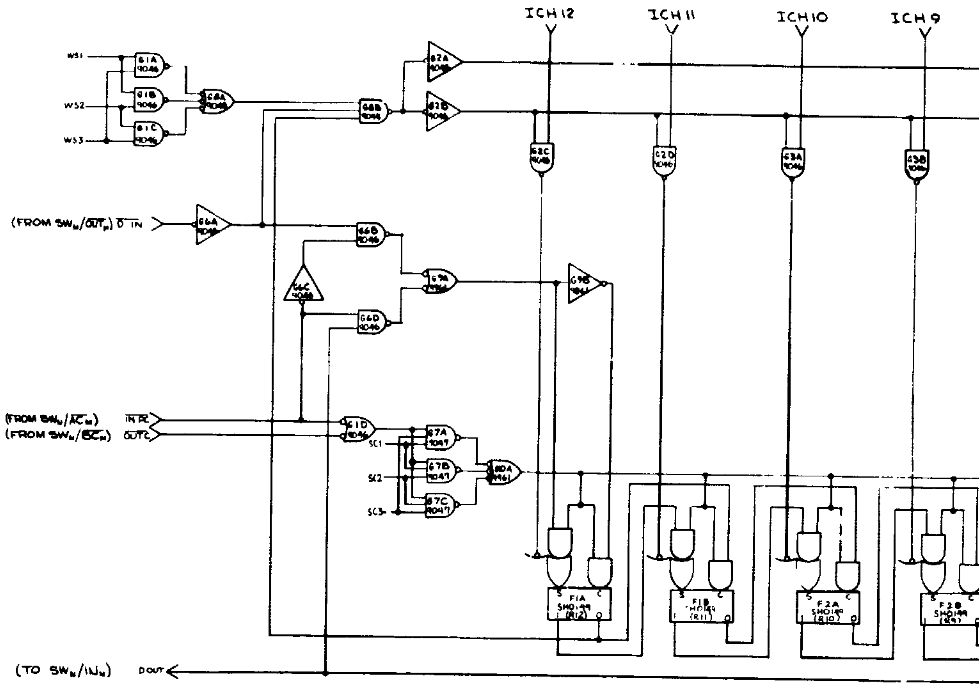
When the command receiver has assembled the 16-bit word, it sets a level called transfer (XFER). This starts the command sync sequencer (flip-flops CSYNC1 and CSYNC2), which steps through the states 10, 11, 01 and returns to its rest state 00. At the count of 10, it enables WS to dump the 16-bit word into CIRØ through CIR12 and CADØ through CAD2. The latter holds the three-bit command address which is decoded as explained above. The actual transfer gates are selected by the decoders and enabled by the next state of the sequencer, 11, for 13 SC pulses which shift the contents of CIR into the addressed register. At this point, a signal is also returned to the telemetry unit to indicate that the transfer is complete so that XFER can be reset and the interface data altered as necessary. The sequencer then steps to 01 and waits for the removal of XFER. As soon as this signal has been reset, the sequencer steps to 00 and stops.

The first seven TAD addresses are each used to activate a pair of gates called command control ($\overline{\text{CCN}}$) and command data ($\overline{\text{CDAT}}$). They are connected to $\overline{\text{PCON}}$ and $\overline{\text{PIN}}$ of the logic units and shift the receiving register and gate CDAT into its input. In order to reprogram the MULTIPAC system, each LU is first addressed with an instruction causing it to set its own program select switch to 7_8. This will cause it to see a program of all zeroes, interpreted as NOP instructions, except when the Command Input Unit switched onto the program bus. By alternately loading words into an available register directly from the command receiver, and then sending instructions to an LIR which cause the LIR to transfer it elsewhere, any location of the processor becomes accessible. By manipulating the address register and the data register of a memory, a bootstrap program can be loaded into it. A logic unit is then given an instruction switching it to that memory for a program source, and the normal method of receiving commands under program control is then employed to read in the rest of the program.

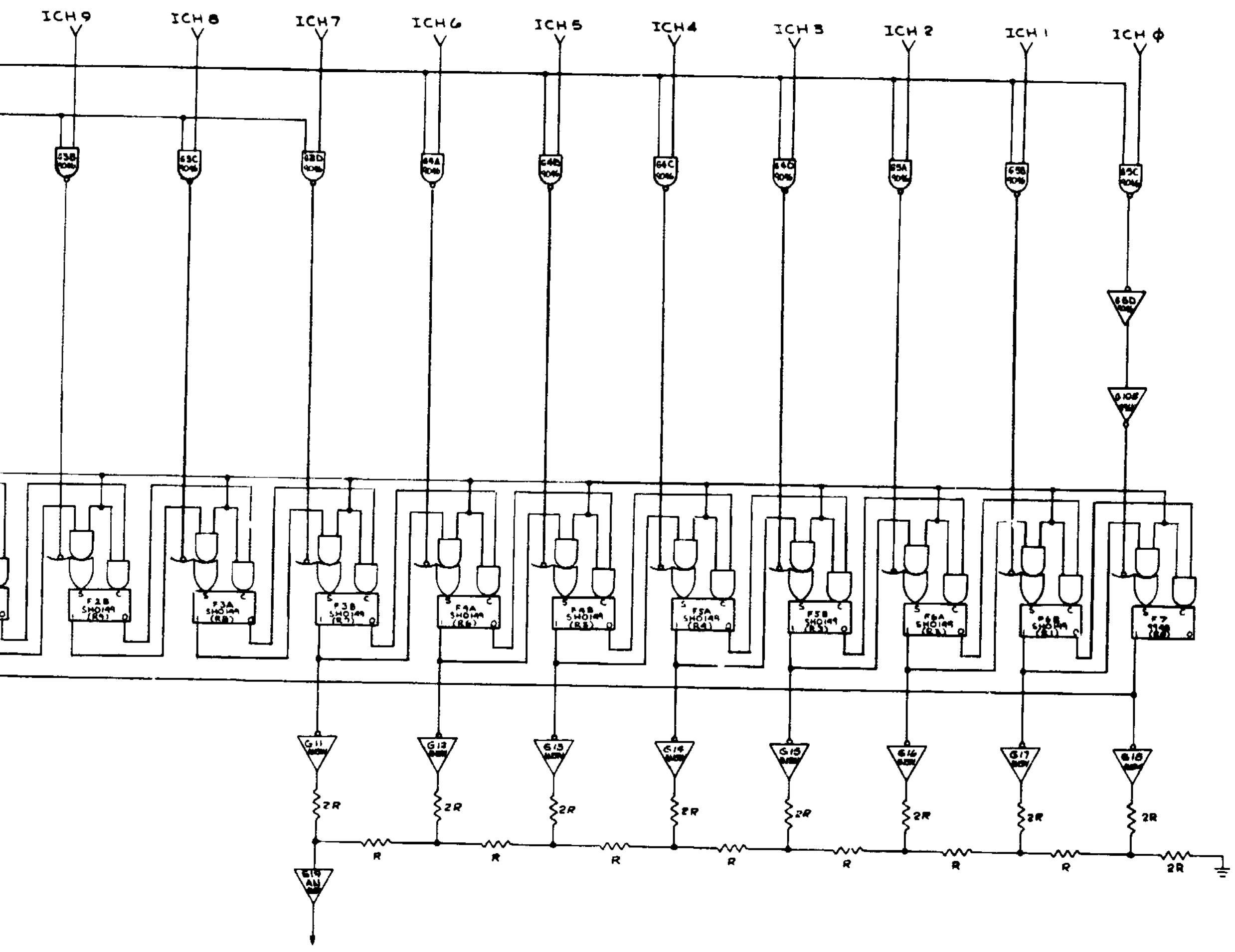
A.5.2 *The telemetry*.-- The telemetry output register (TOR) is addressable by its $\overline{\text{INPC}}$ and $\overline{\text{DATA IN}}$ signals. When INPC is pulled to zero, it will shift data in from the DATA IN line of each SC pulse. It also has a one-stage extension called the telemetry output flip-flop (TOF), but the latter does not shift in response to SC pulses. Both the register and the extension may be shifted by externally received telemetry clock pulses.

Operation of the register is as follows: The register is loaded by the program with a telemetry output word which contains a ONE in the high order bit. The telemetry clock pulse shifts this word into TOF at the telemetry rate, and the TOUT signal modulates the telemetry transmitter. When the last significant bit of the telemetry word is located in TOF, then the ONE is contained in TOR∅ and all ZEROES have been shifted in following it. This condition is detected by the decoder attached to TOR. This signal is used to interrupt the processor handling the data formatting and outputting routine and causes it to load the next telemetry output word into TOR. This must be accomplished before the next telemetry clock pulse in order for the telemetry output stream to be uninterrupted.

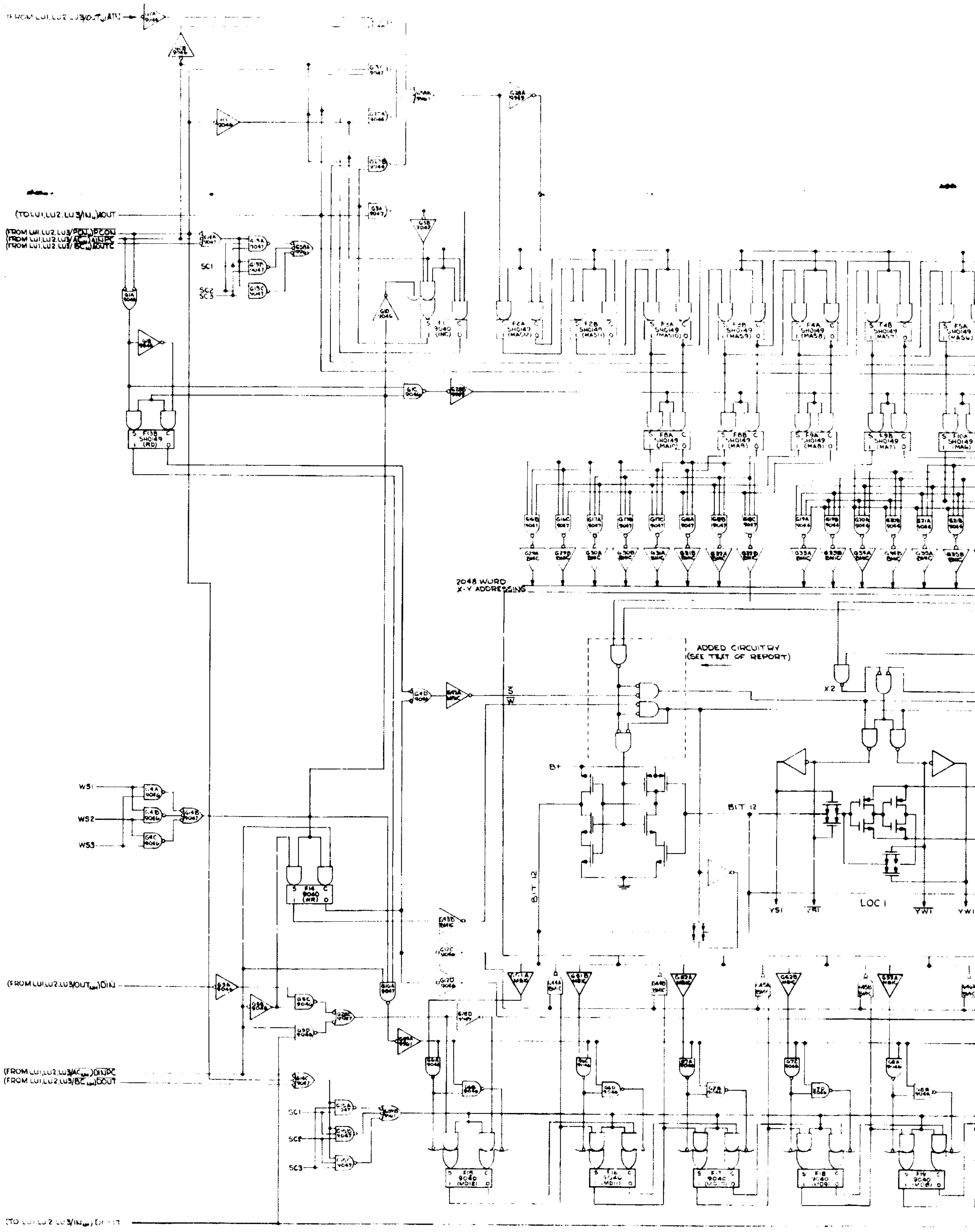
FOLDOUT FRAME |



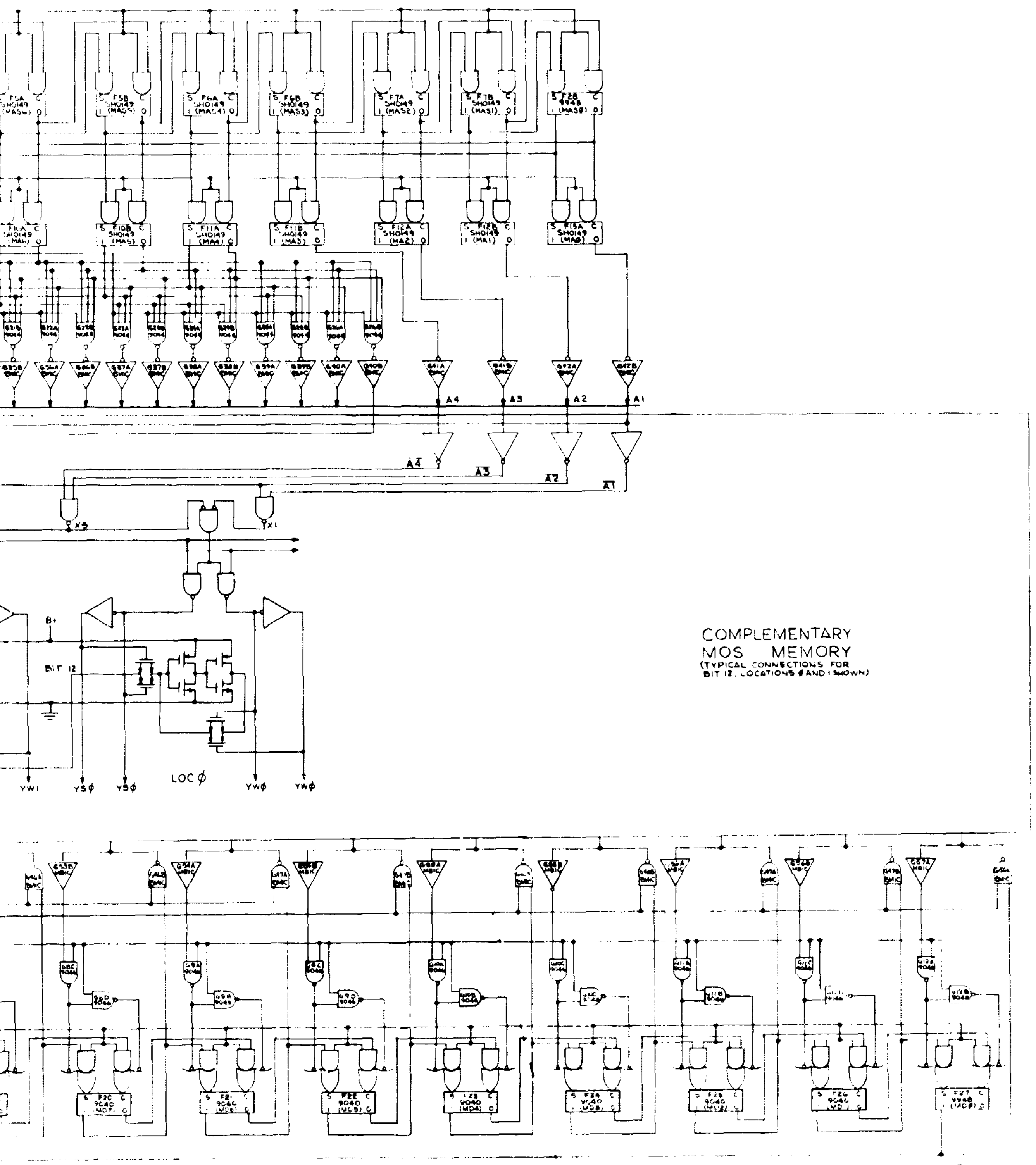
FOLDOUT FRAME 2



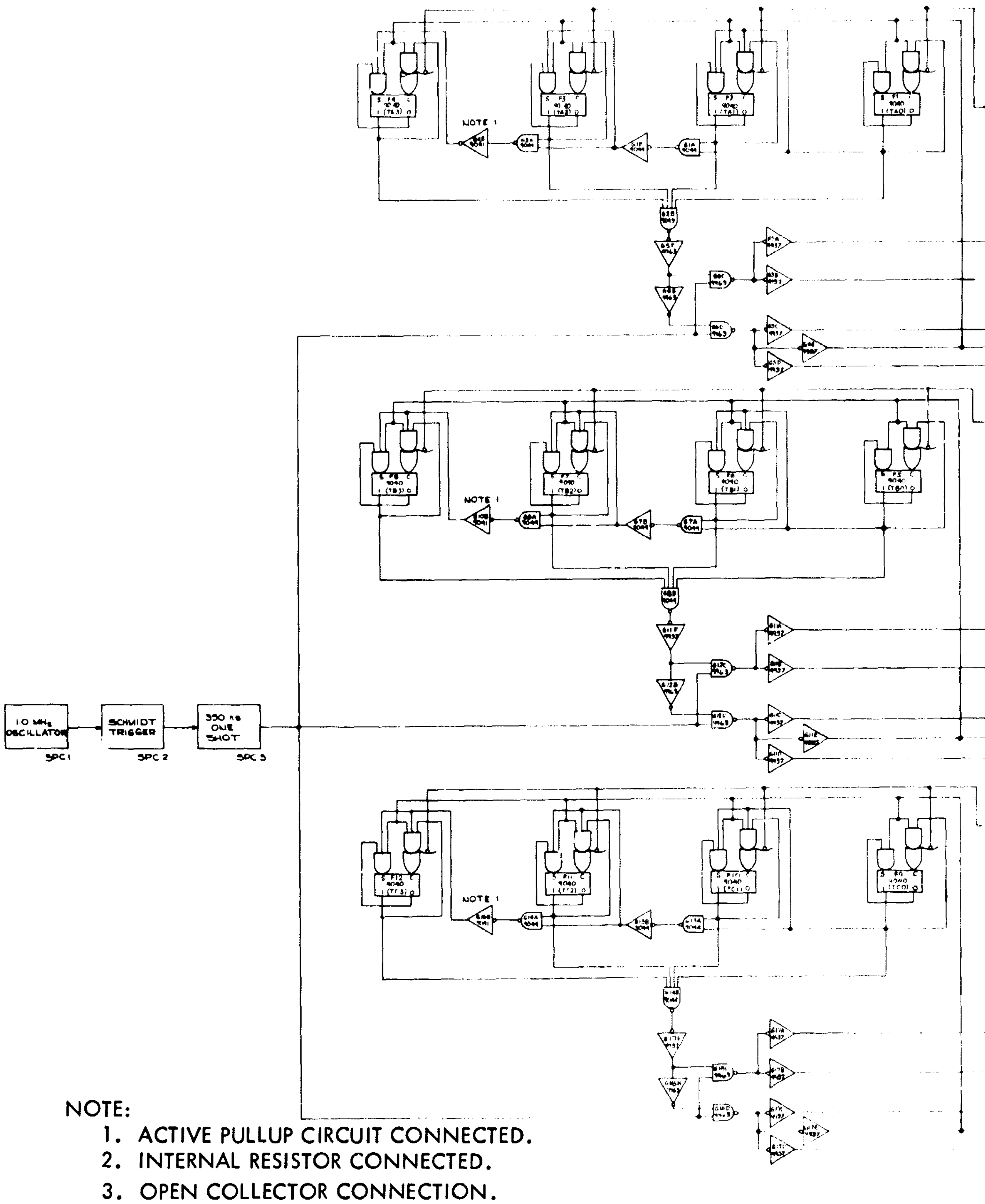
MULTIPAC D/A Register



FOLDOUT FRAME /



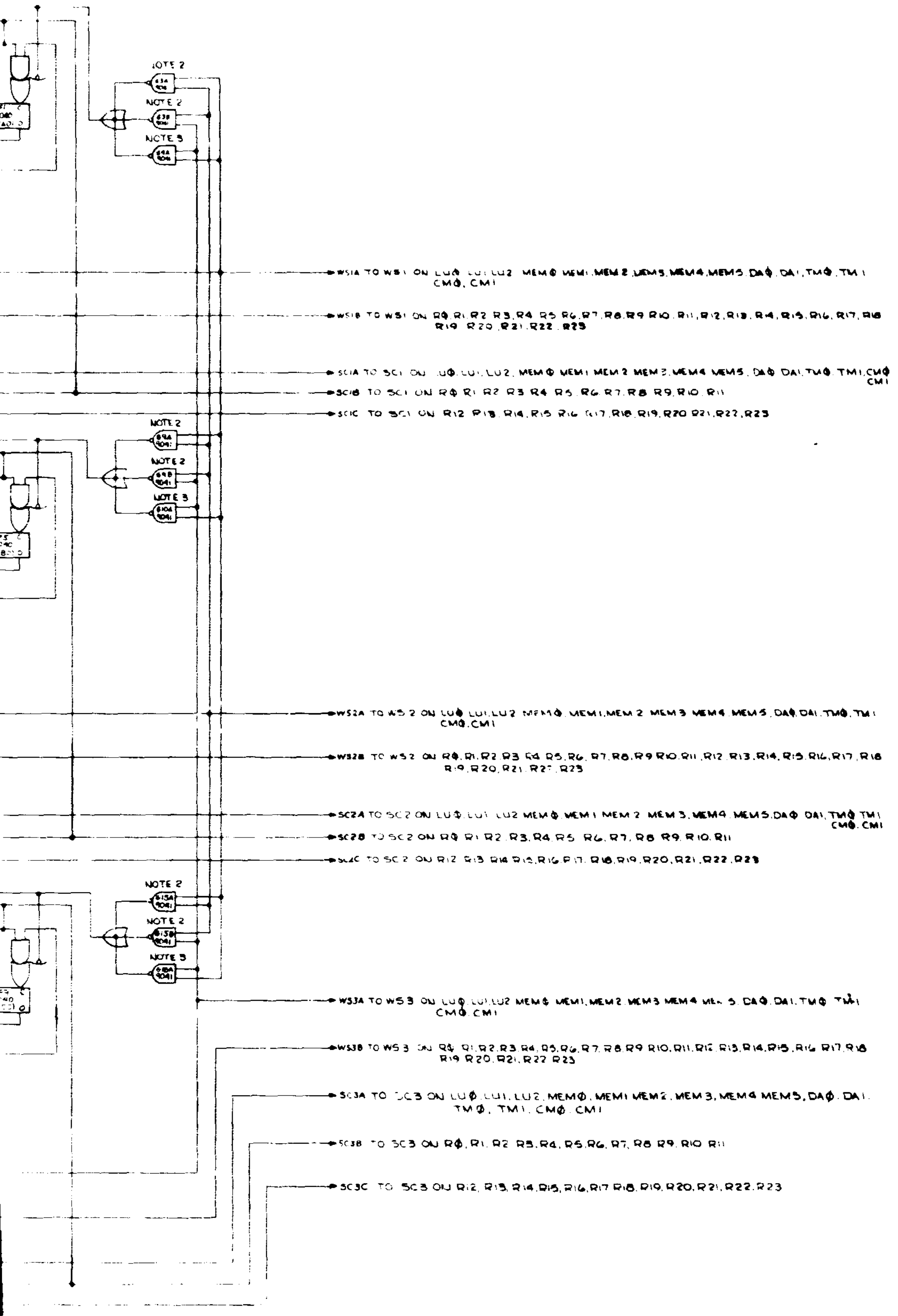
FOLDOUT FRAME 2



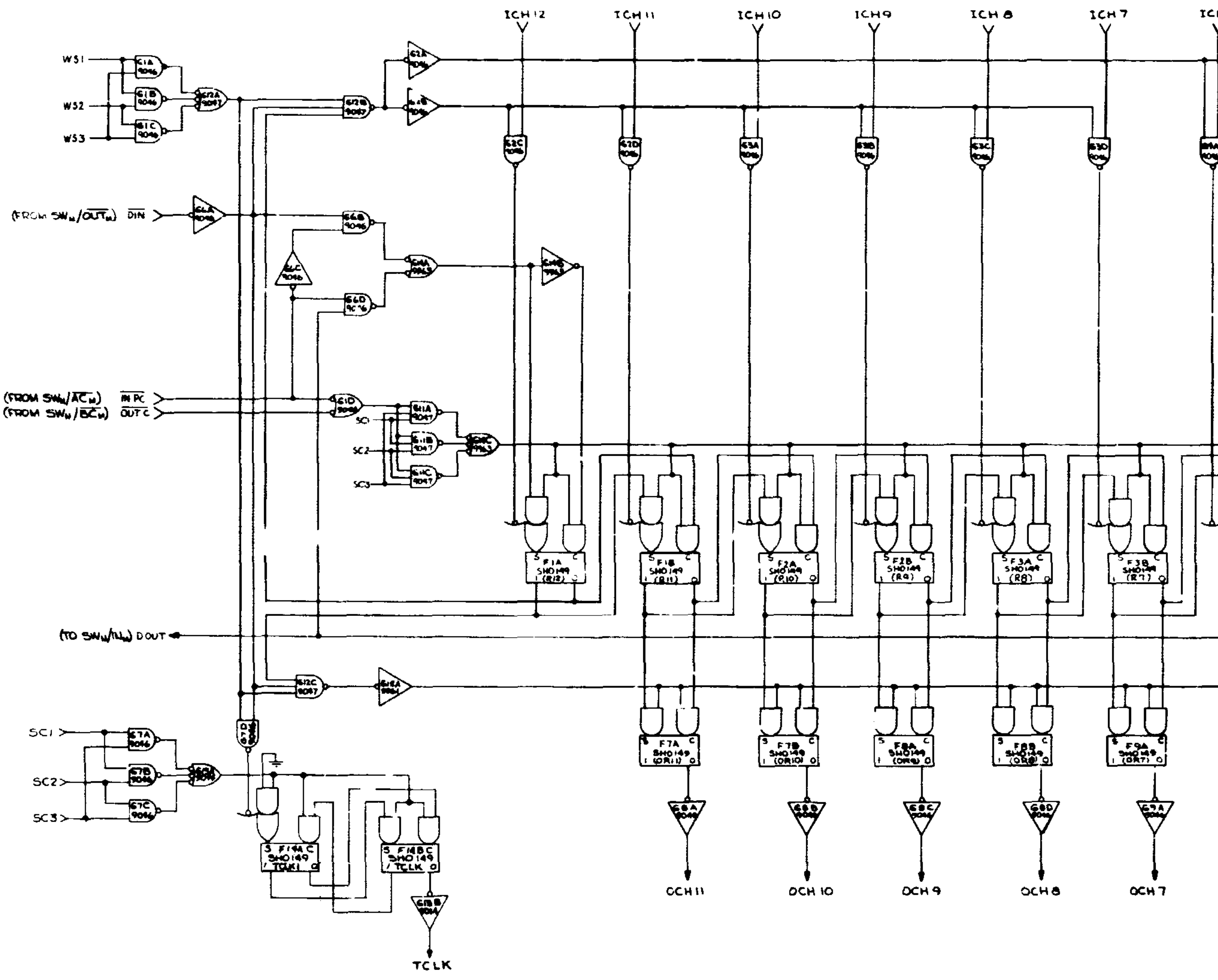
NOTE:

1. ACTIVE PULLUP CIRCUIT CONNECTED.
2. INTERNAL RESISTOR CONNECTED.
3. OPEN COLLECTOR CONNECTION.

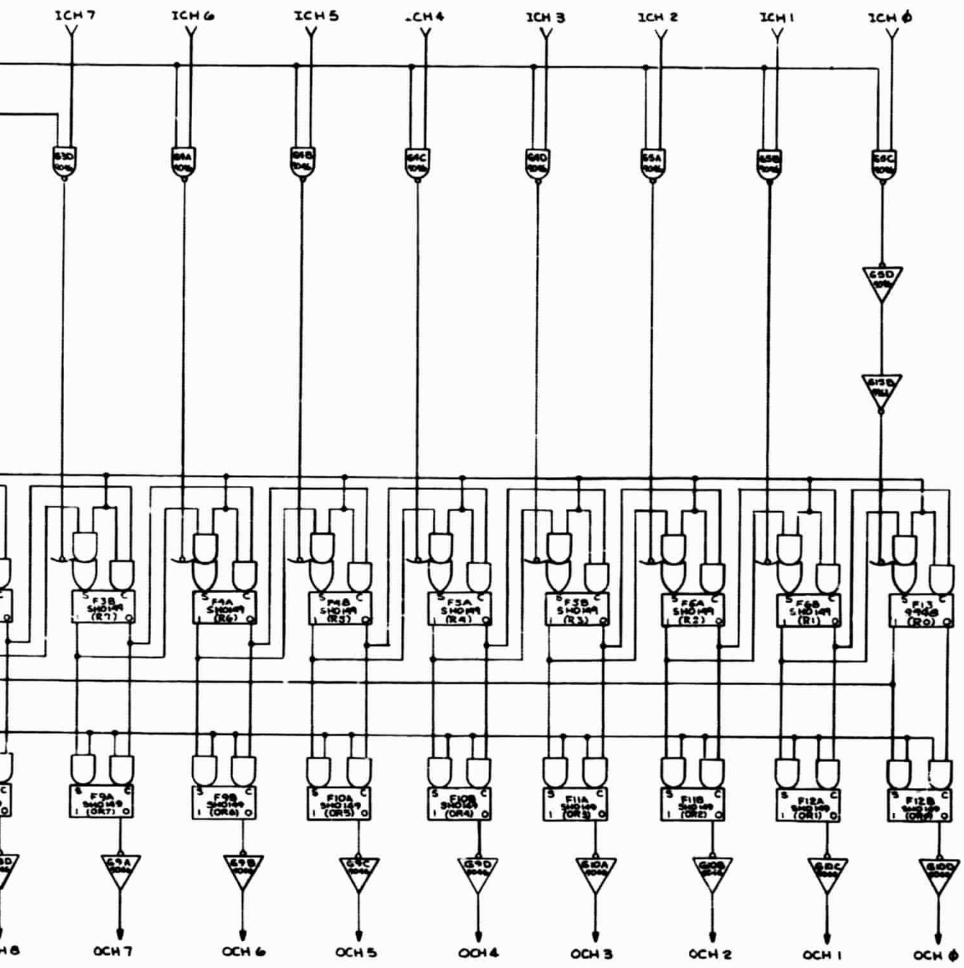
[illegible] FRAME 1


NOTE 2
NOTE 2
NOTE 3
WS1A TO WS1 ON LUΦ LU1 LU2 MEMΦ MEM1 MEM2 MEM3 MEM4 MEM5 DAΦ DA1 TMΦ TM1 CMΦ CM1
WS1B TO WS1 ON RΦ R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23
SC1A TO SC1 ON LUΦ LU1 LU2 MEMΦ MEM1 MEM2 MEM3 MEM4 MEM5 DAΦ DA1 TMΦ TM1 CMΦ CM1
SC1B TO SC1 ON RΦ R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11
SC1C TO SC1 ON R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23
NOTE 2
NOTE 2
NOTE 3
WS2A TO WS2 ON LUΦ LU1 LU2 MEMΦ MEM1 MEM2 MEM3 MEM4 MEM5 DAΦ DA1 TMΦ TM1 CMΦ CM1
WS2B TO WS2 ON RΦ R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23
SC2A TO SC2 ON LUΦ LU1 LU2 MEMΦ MEM1 MEM2 MEM3 MEM4 MEM5 DAΦ DA1 TMΦ TM1 CMΦ CM1
SC2B TO SC2 ON RΦ R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11
SC2C TO SC2 ON R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23
NOTE 2
NOTE 2
NOTE 3
WS3A TO WS3 ON LUΦ LU1 LU2 MEMΦ MEM1 MEM2 MEM3 MEM4 MEM5 DAΦ DA1 TMΦ TM1 CMΦ CM1
WS3B TO WS3 ON RΦ R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23
SC3A TO SC3 ON LUΦ LU1 LU2 MEMΦ MEM1 MEM2 MEM3 MEM4 MEM5 DAΦ DA1 TMΦ TM1 CMΦ CM1
SC3B TO SC3 ON RΦ R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11
SC3C TO SC3 ON R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23

MULTIPAC Timing Generator

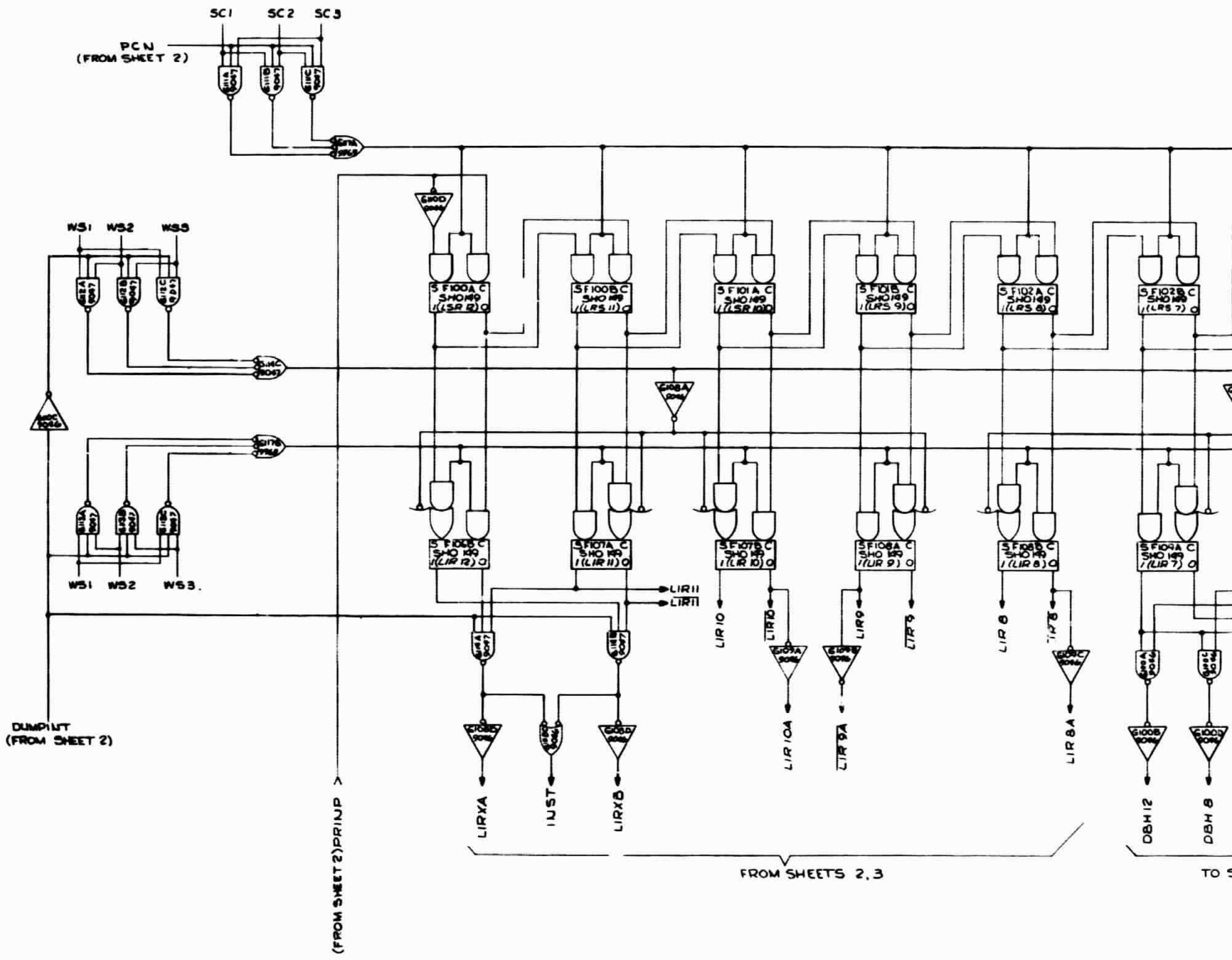


FOLDOUT FRAME /



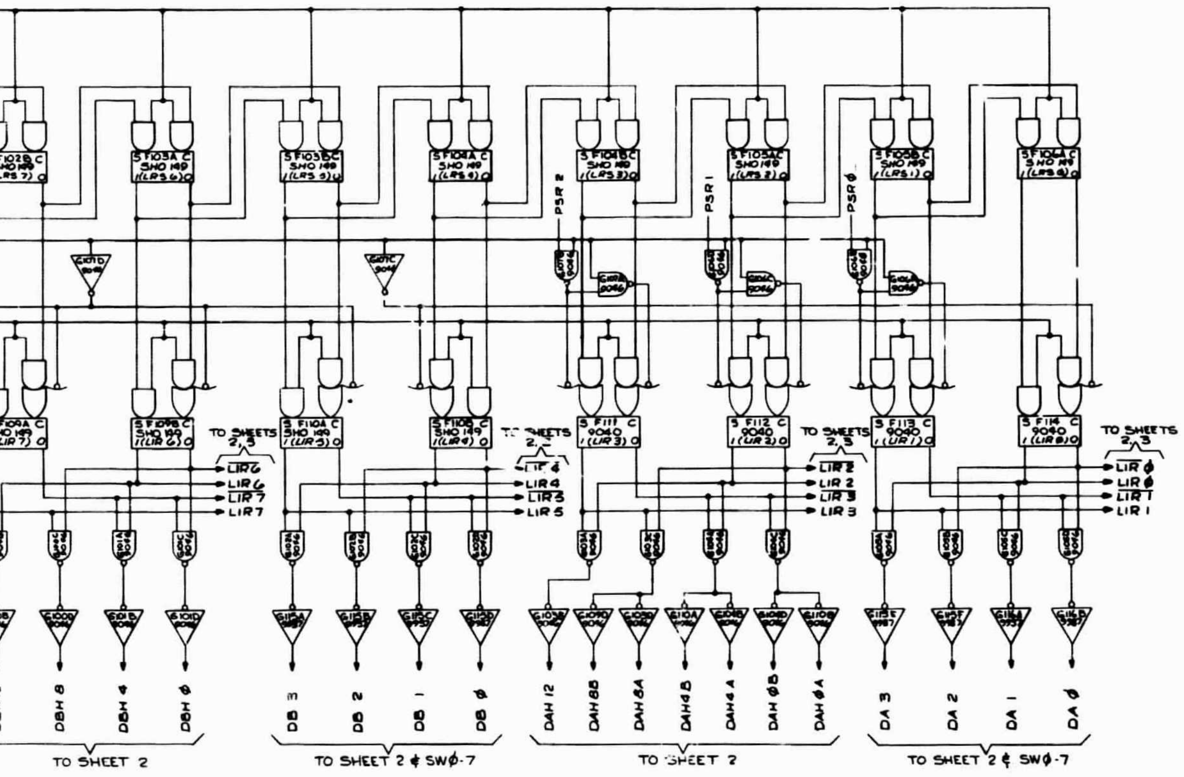
MULTIPAC Register

FOLDOUT FRAME 2

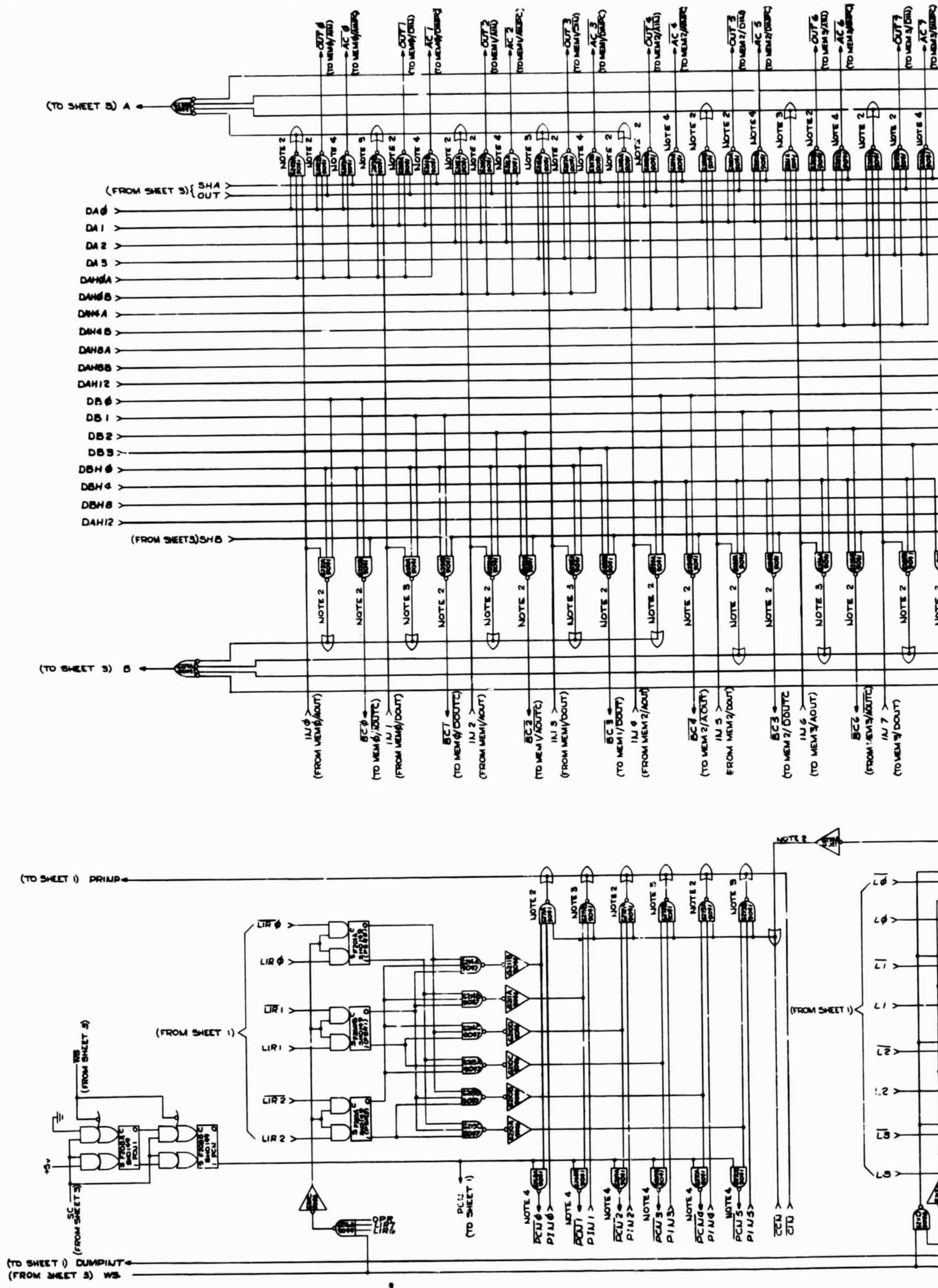


SC1
SC2
SC3
PCN
(FROM SHEET 2)
WS1
WS2
WS3
SF100A C SHO149 1(LSR 12) 0
SF100B C SHO149 1(LRS 11) 0
SF101A C SHO149 1(LSR 10) 0
SF101B C SHO149 1(LRS 9) 0
SF102A C SHO149 1(LRS 8) 0
SF102B C SHO149 1(LRS 7) 0
SF106B C SHO 149 1(LIR 12) 0
SF107A C SHO 149 1(LIR 11) 0
SF107B C SHO 149 1(LIR 10) 0
SF108A C SHO 149 1(LIR 9) 0
SF108B C SHO 149 1(LIR 8) 0
SF109A C SHO 149 1(LIR 7) 0
LIR11
LIR11
LIR10
LIR10
LIR9
LIR9
LIR8
LIR8
LIR10A
LIR9A
LIR8A
DUMPINT
(FROM SHEET 2)
(FROM SHEET 2) PRIJP
LIRXA
IUST
LIRXB
DBH12
DBH8
FROM SHEETS 2, 3
TO S

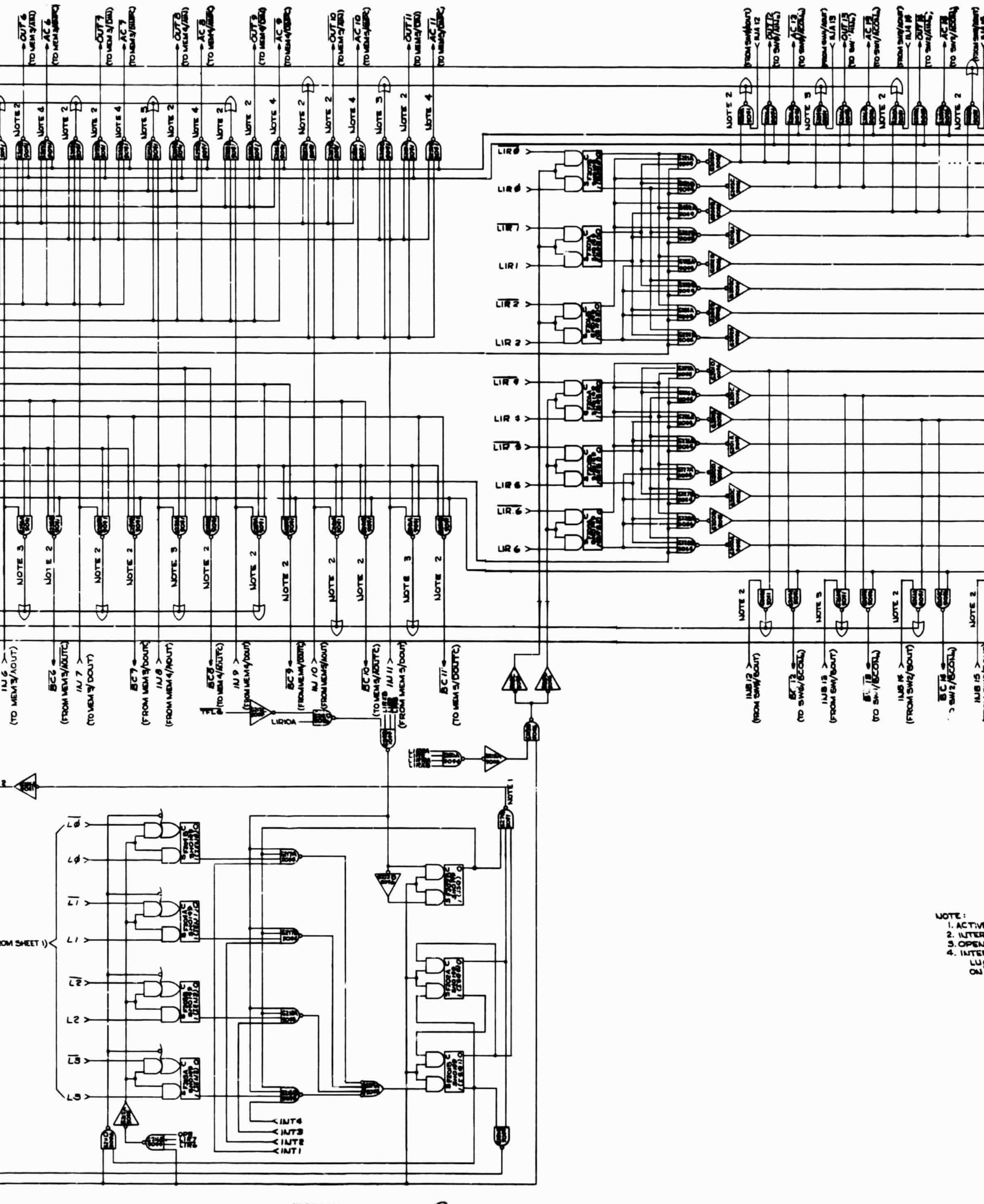
FOLDOUT FRAME 1



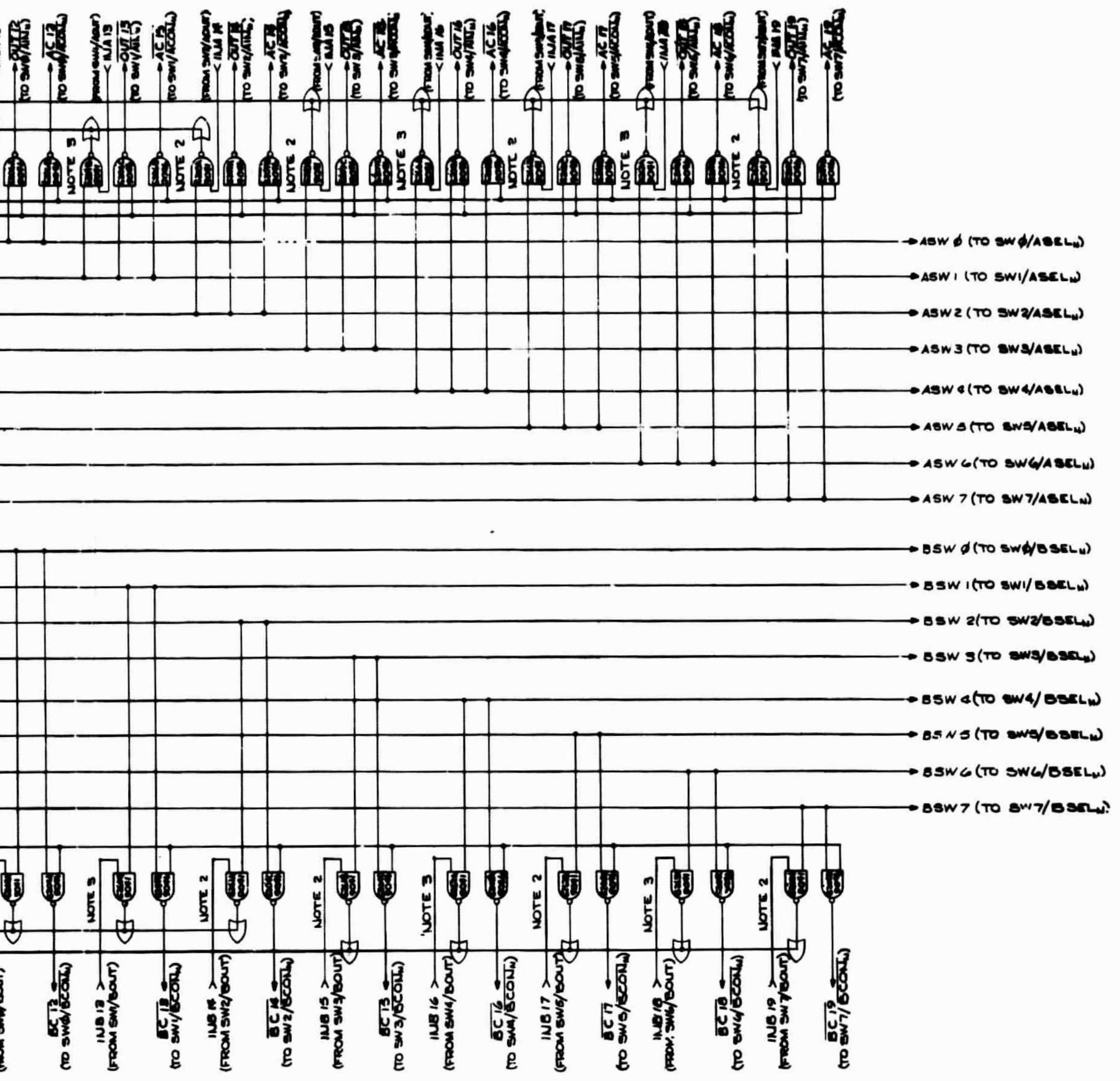
MUTIPAC Logic Unit



FOLDOUT FRAME 1

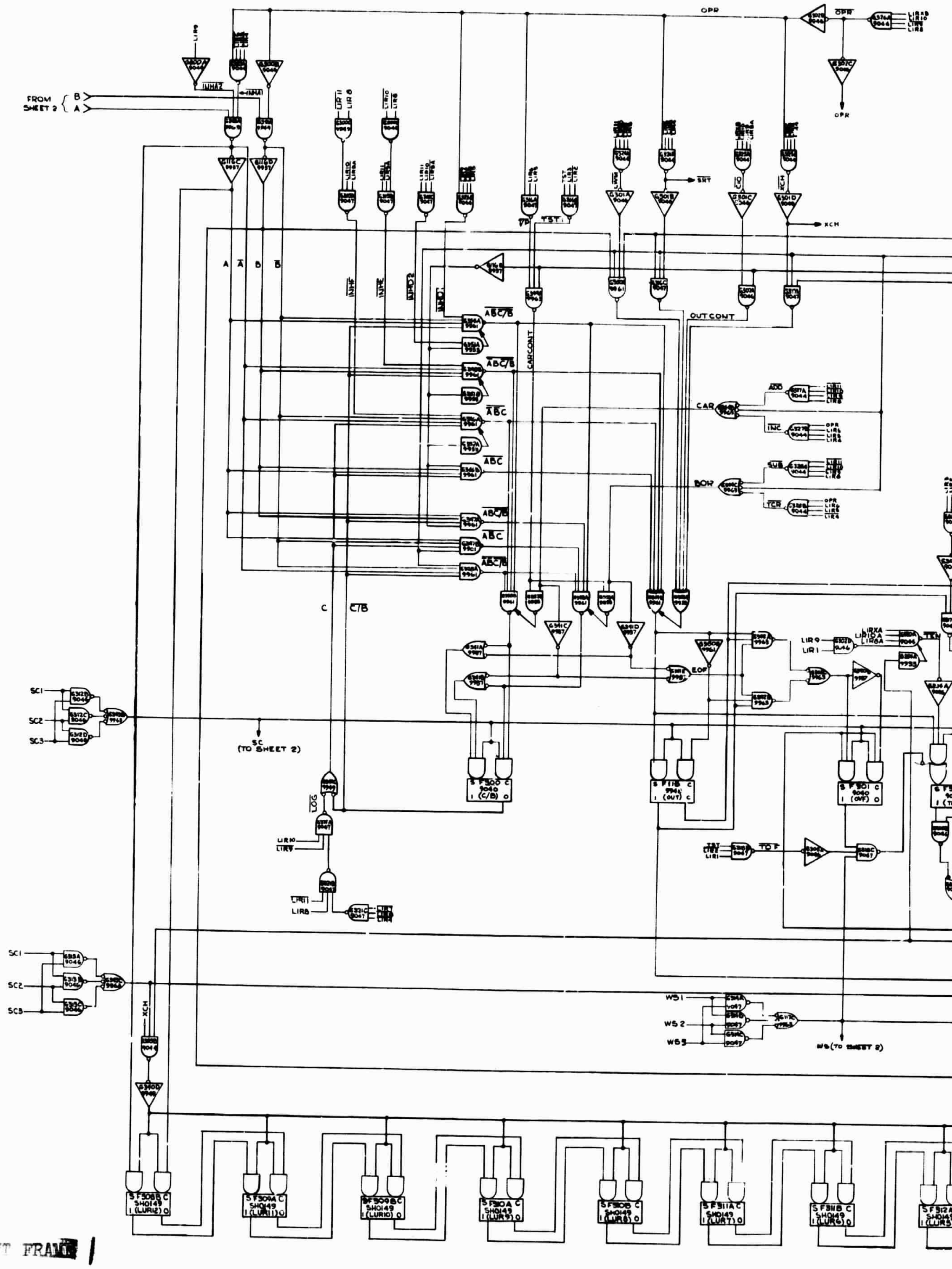


FOLDOUT FRAME 2

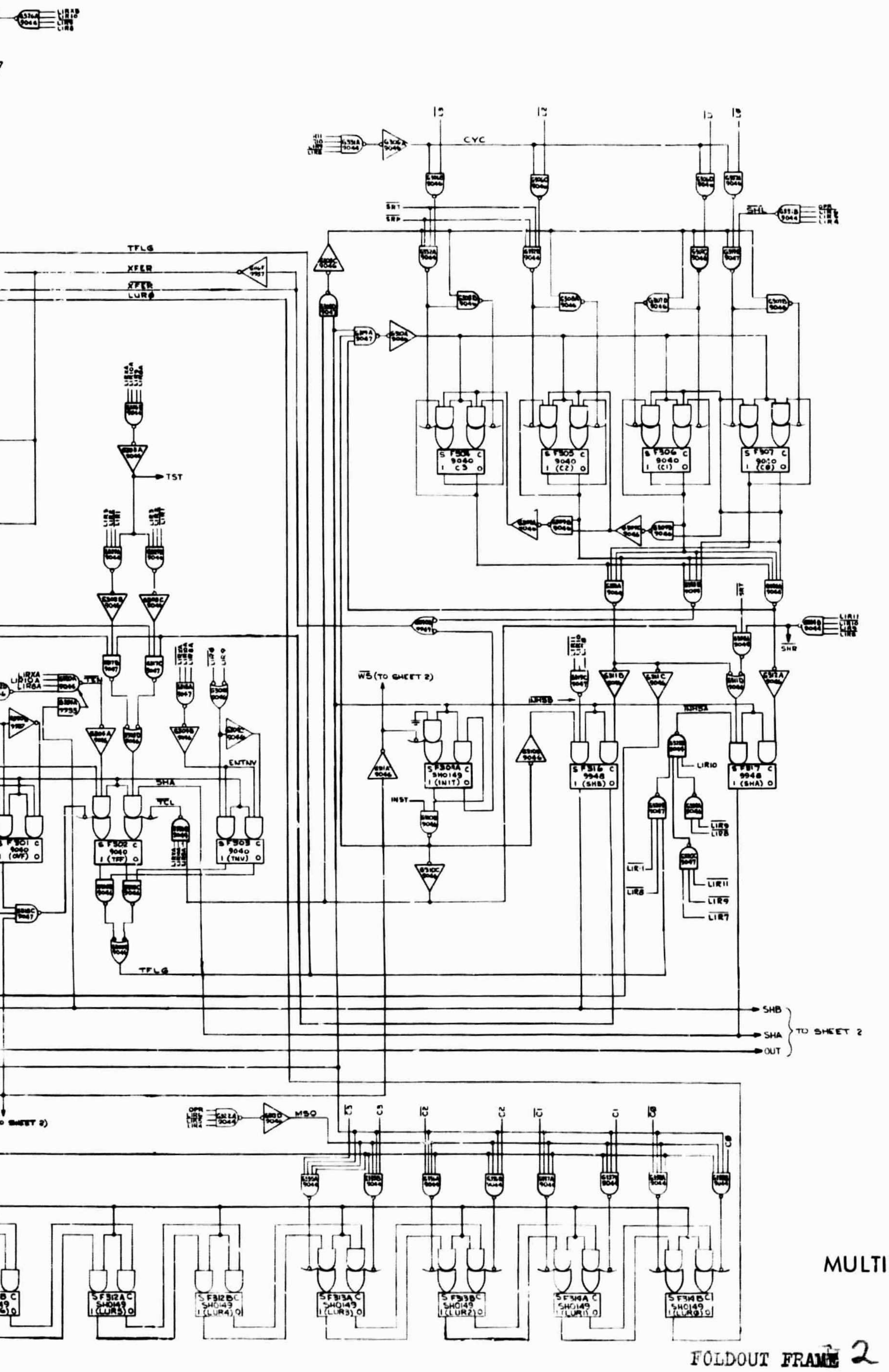


NOTE:
1. ACTIVE PULLUP CIRCUIT CONNECTED
2. INTERNAL RESISTOR CONNECTED
3. OPEN COLLECTOR CONNECTION
4. INTERNAL RESISTOR CONNECTED ON LU∅, OPEN COLLECTOR CONNECTIONS ON LU1, LU2

MULTIPAC Logic Unit

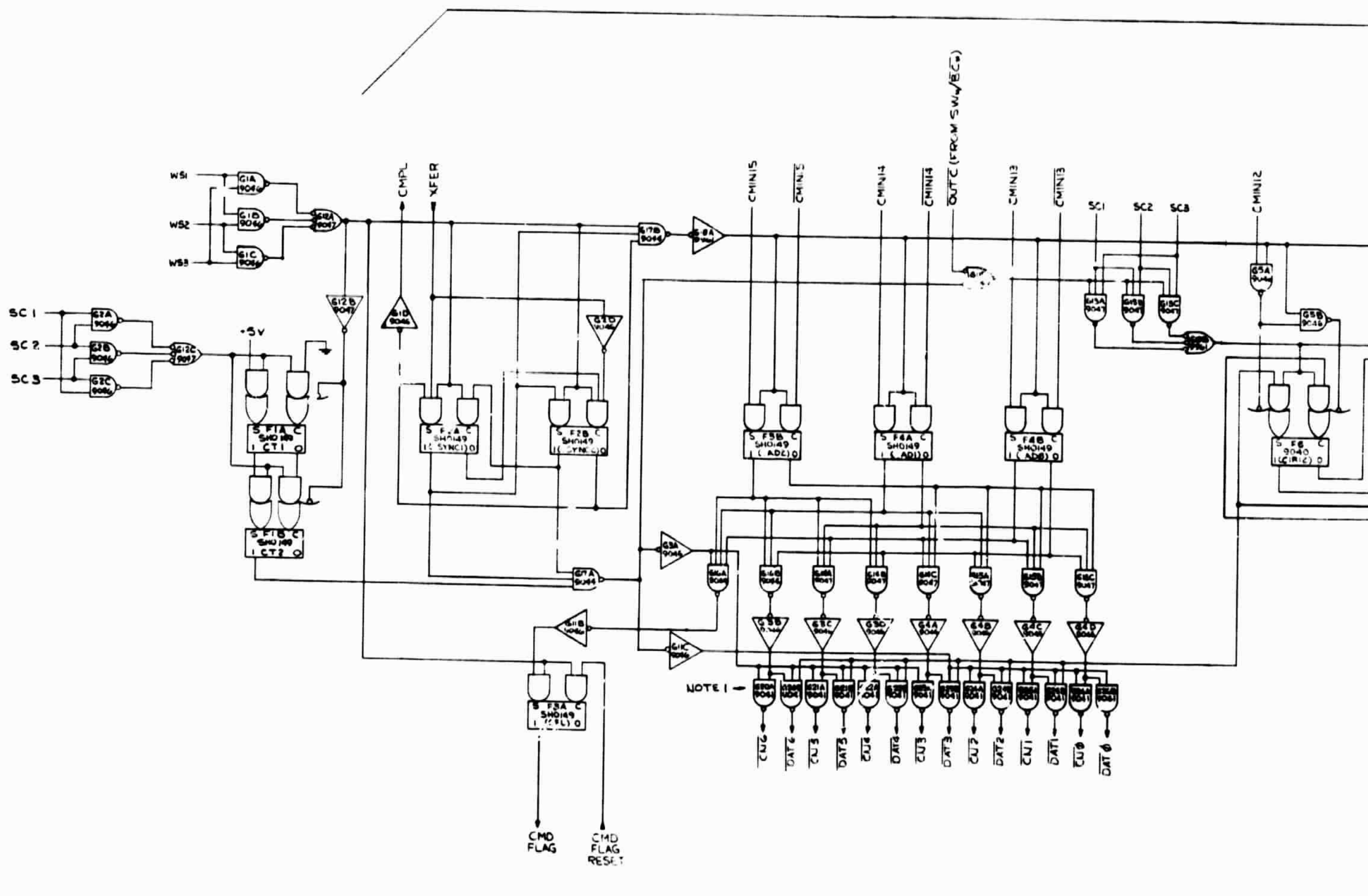


FOLDOUT FRAME 1

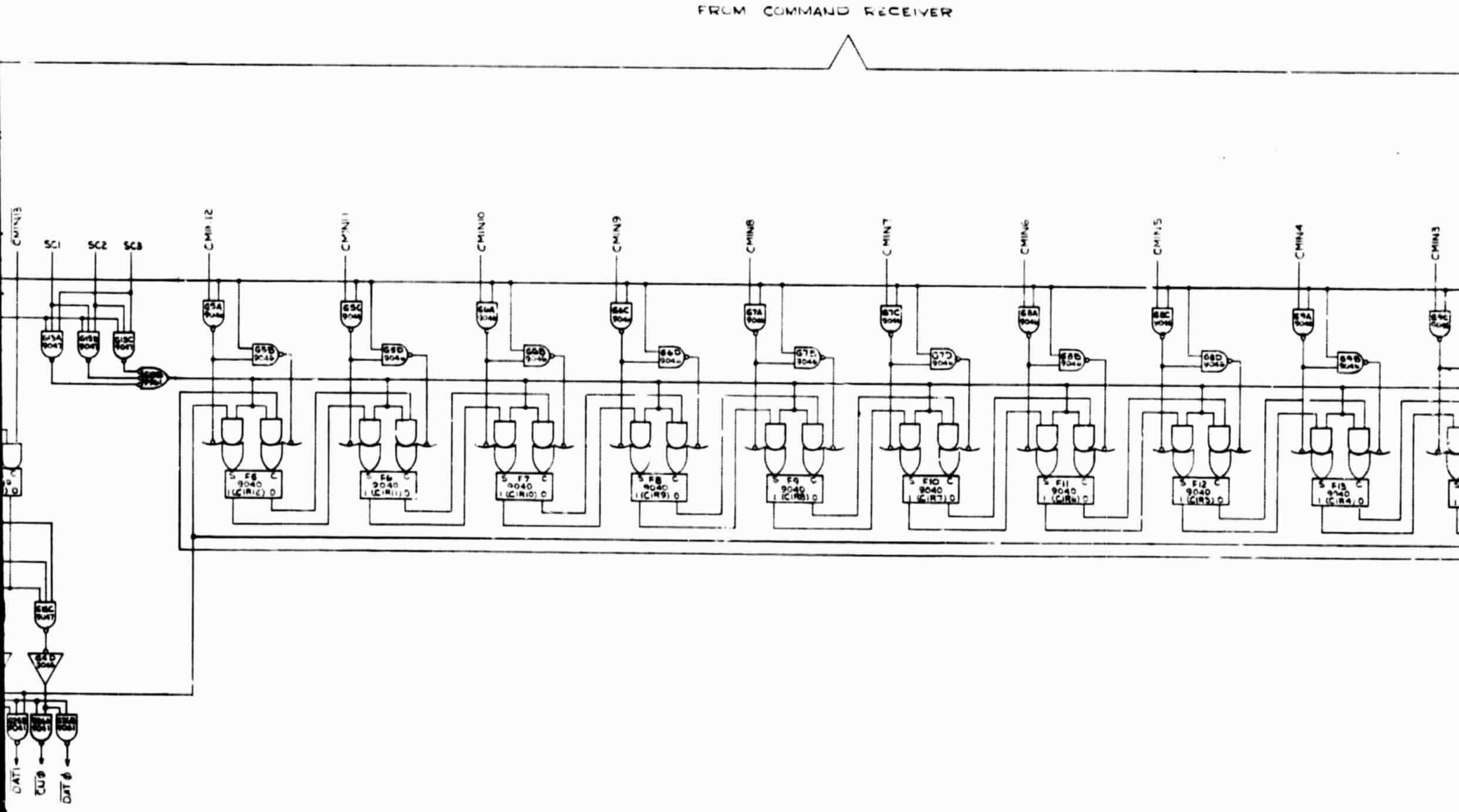


MULTIPAC Logic Unit

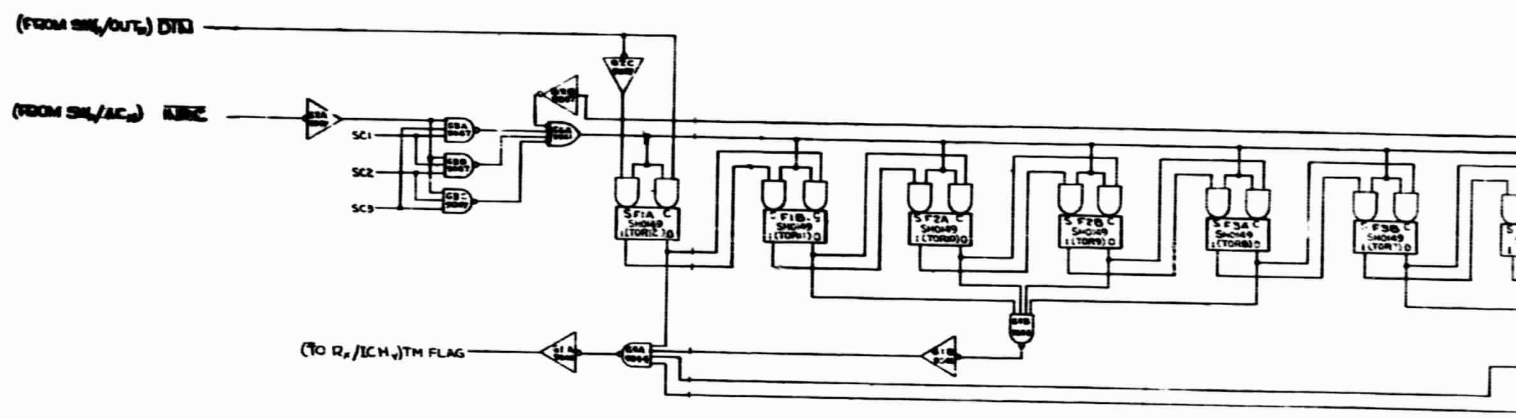
FOLDOUT FRAME 2



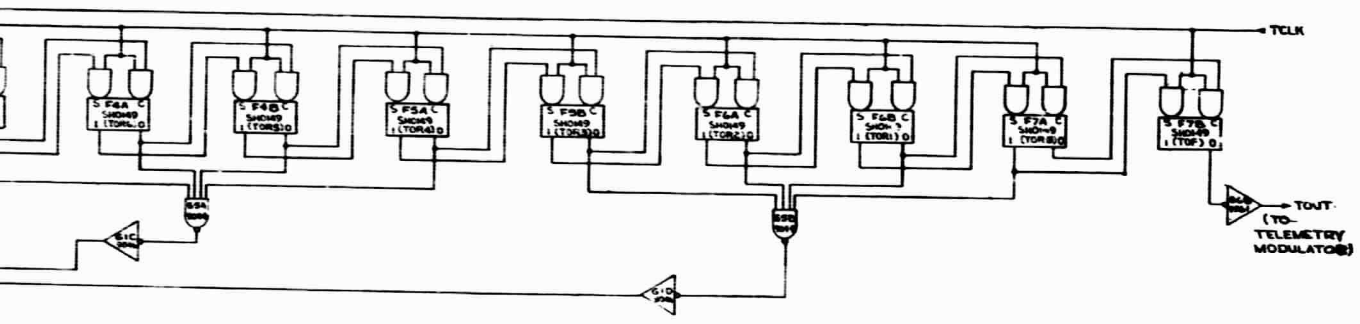
FOLDOUT FRAME 1



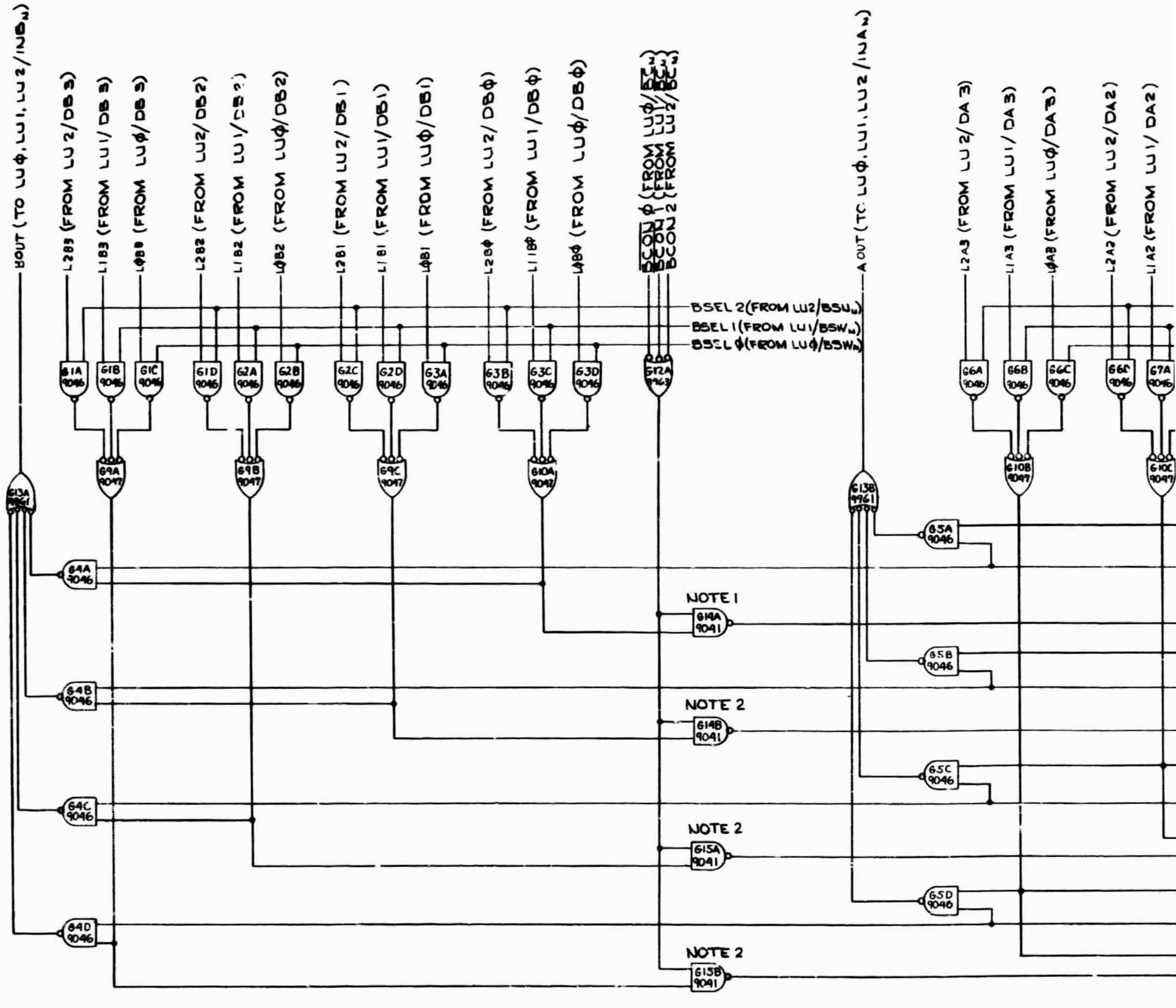
FOLDOUT FRAME 2



FOLDOUT FRAME 1

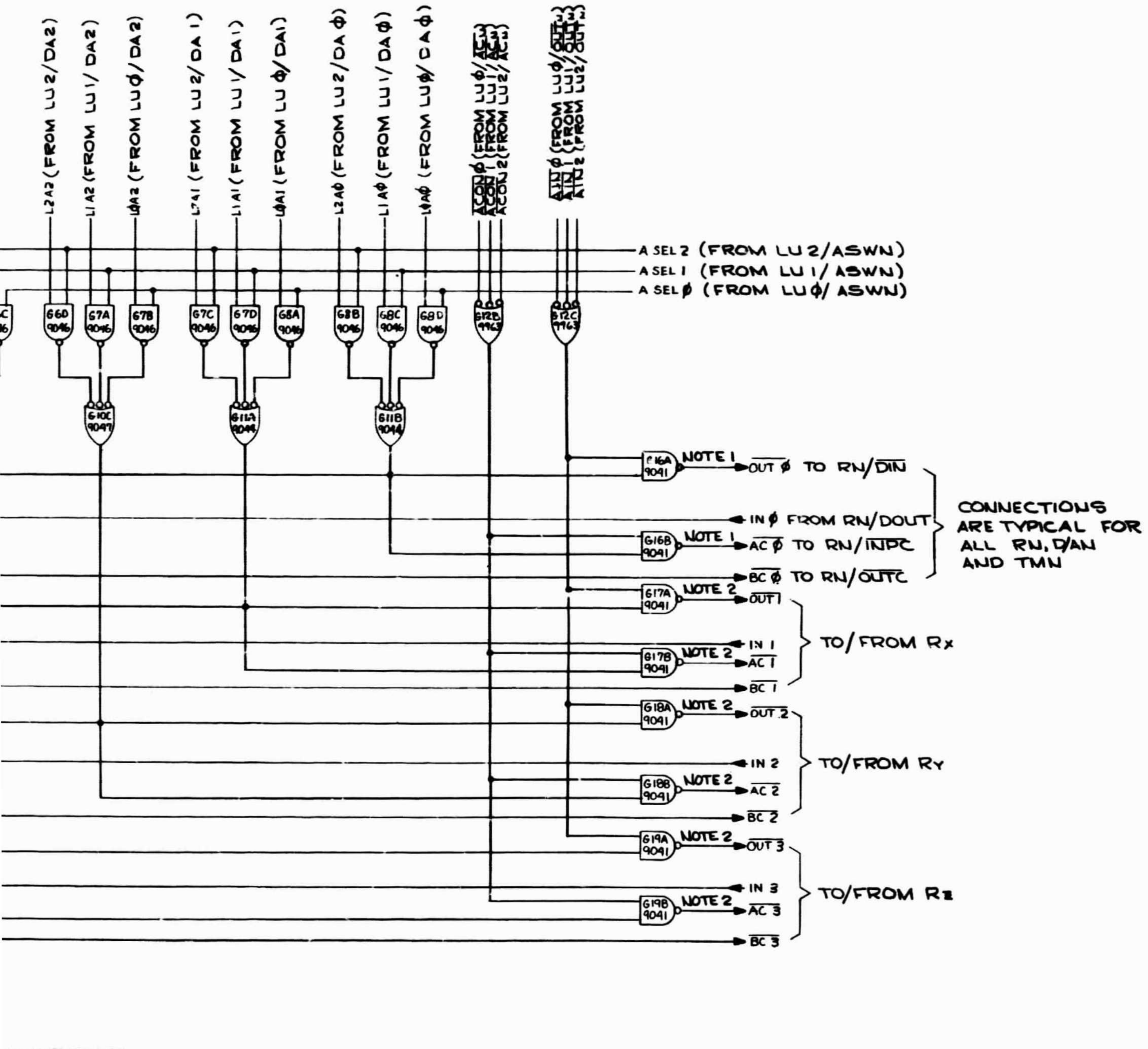


MULTIPAC Telemetry Output Unit



NOTE:
1. INTERNAL RESISTOR CONNECTE
2. OPEN COLLECTOR WITH EXTERNAL
7500 OHM LOAD RESISTOR
CONNECTED TO +5V.

FOLDOUT FRAME 1



ONNECTED
XTERNAL
OR

MULTIPAC Address Switch

FOLDOUT FRAME 2

APPENDIX B

SYSTEM INTERCONNECTION OF MODULES

The interconnection of all modules to form the system is given in Table 20. It is considered wiser to provide this detailed documentation as a wire list rather than a diagram since, owing to the large number of modules and the number of interconnections between them, a diagram would be very crowded and complex.

TABLE 20

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/INØ	LU1/INØ	LU2/INØ	MEMØ/AOUT
$\overline{\text{OUTØ}}$	$\overline{\text{OUTØ}}$	$\overline{\text{OUTØ}}$	$\overline{\text{AIN}}$
$\overline{\text{ACØ}}$	$\overline{\text{ACØ}}$	$\overline{\text{ACØ}}$	$\overline{\text{AINPC}}$
$\overline{\text{BCØ}}$	$\overline{\text{BCØ}}$	$\overline{\text{BCØ}}$	$\overline{\text{AOUTC}}$
IN1, PINØ	IN1, PINØ	IN1, PINØ	DOUT
$\overline{\text{OUT1}}$	$\overline{\text{OUT1}}$	$\overline{\text{OUT1}}$	$\overline{\text{DIN}}$
$\overline{\text{AC1}}$	$\overline{\text{AC1}}$	$\overline{\text{AC1}}$	$\overline{\text{AC1}}$
$\overline{\text{BC1}}$	$\overline{\text{BC1}}$	$\overline{\text{BC1}}$	$\overline{\text{BC1}}$
$\overline{\text{PCNØ}}$	$\overline{\text{PCNØ}}$	$\overline{\text{PCNØ}}$	$\overline{\text{PCON}}$
IN2	IN2	IN2	MEM1/AOUT
$\overline{\text{OUT2}}$	$\overline{\text{OUT2}}$	$\overline{\text{OUT2}}$	$\overline{\text{AIN}}$
$\overline{\text{AC2}}$	$\overline{\text{AC2}}$	$\overline{\text{AC2}}$	$\overline{\text{AINPC}}$
$\overline{\text{BC2}}$	$\overline{\text{BC2}}$	$\overline{\text{BC2}}$	$\overline{\text{AOUTC}}$
IN3, PIN1	IN3, PIN1	IN3, PIN 1	DOUT
$\overline{\text{OUT3}}$	$\overline{\text{OUT3}}$	$\overline{\text{OUT3}}$	$\overline{\text{DIN}}$
$\overline{\text{AC3}}$	$\overline{\text{AC3}}$	$\overline{\text{AC3}}$	$\overline{\text{AC1}}$
$\overline{\text{BC3}}$	$\overline{\text{BC3}}$	$\overline{\text{BC3}}$	$\overline{\text{BC1}}$
$\overline{\text{PCN1}}$	$\overline{\text{PCN1}}$	$\overline{\text{PCN1}}$	$\overline{\text{PCON}}$

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/IN4	LU1/IN4	LU2/IN4	MEM2/AOUT
$\overline{\text{OUT4}}$	$\overline{\text{OUT4}}$	$\overline{\text{OUT4}}$	$\overline{\text{AIN}}$
$\overline{\text{AC4}}$	$\overline{\text{AC4}}$	$\overline{\text{AC4}}$	$\overline{\text{AINPC}}$
$\overline{\text{BC4}}$	$\overline{\text{BC4}}$	$\overline{\text{BC4}}$	$\overline{\text{AOUTC}}$
IN5, PIN2	IN5, PIN2	IN5, PIN2	DOUT
$\overline{\text{OUT5}}$	$\overline{\text{OUT5}}$	$\overline{\text{OUT5}}$	$\overline{\text{AIN}}$
$\overline{\text{AC5}}$	$\overline{\text{AC5}}$	$\overline{\text{AC5}}$	$\overline{\text{AINPC}}$
$\overline{\text{BC5}}$	$\overline{\text{BC5}}$	$\overline{\text{BC5}}$	$\overline{\text{AOUTC}}$
$\overline{\text{PCN2}}$	$\overline{\text{PCN2}}$	$\overline{\text{PCN2}}$	$\overline{\text{PCON}}$
IN6	IN6	IN6	MEM3/AOUT
$\overline{\text{OUT6}}$	$\overline{\text{OUT6}}$	$\overline{\text{OUT6}}$	$\overline{\text{AIN}}$
$\overline{\text{AC6}}$	$\overline{\text{AC6}}$	$\overline{\text{AC6}}$	$\overline{\text{AINPC}}$
$\overline{\text{BC6}}$	$\overline{\text{BC6}}$	$\overline{\text{BC6}}$	$\overline{\text{AOUTC}}$
IN7, PIN3	IN7, PIN3	IN7, PIN3	DOUT
$\overline{\text{OUT7}}$	$\overline{\text{OUT7}}$	$\overline{\text{OUT7}}$	$\overline{\text{DIN}}$
$\overline{\text{AC7}}$	$\overline{\text{AC7}}$	$\overline{\text{AC7}}$	$\overline{\text{DINPC}}$
$\overline{\text{BC7}}$	$\overline{\text{BC7}}$	$\overline{\text{BC7}}$	$\overline{\text{DOUTC}}$
$\overline{\text{PCN3}}$	$\overline{\text{PCN3}}$	$\overline{\text{PC3}}$	$\overline{\text{PCON}}$

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/$\overline{\text{IN8}}$	LU1/$\overline{\text{IN8}}$	LU2/$\overline{\text{IN8}}$	MEM4/$\overline{\text{AOUT}}$
$\overline{\text{OUT8}}$	$\overline{\text{OUT8}}$	$\overline{\text{OUT8}}$	$\overline{\text{AIN}}$
$\overline{\text{AC8}}$	$\overline{\text{AC8}}$	$\overline{\text{AC8}}$	$\overline{\text{AINPC}}$
$\overline{\text{BC8}}$	$\overline{\text{BC8}}$	$\overline{\text{BC8}}$	$\overline{\text{AOUTC}}$
$\overline{\text{IN9}}$, PIN4	$\overline{\text{IN9}}$, PIN4	$\overline{\text{IN9}}$, PIN4	$\overline{\text{DOUT}}$
$\overline{\text{OUT9}}$	$\overline{\text{OUT9}}$	$\overline{\text{OUT9}}$	$\overline{\text{DIN}}$
$\overline{\text{AC9}}$	$\overline{\text{AC9}}$	$\overline{\text{AC9}}$	$\overline{\text{DINPC}}$
$\overline{\text{BC9}}$	$\overline{\text{BC9}}$	$\overline{\text{BC9}}$	$\overline{\text{DOUTC}}$
$\overline{\text{PCN4}}$	$\overline{\text{PCN4}}$	$\overline{\text{PCN4}}$	$\overline{\text{PCON}}$
$\overline{\text{IN1Ø}}$	$\overline{\text{IN1Ø}}$	$\overline{\text{IN1Ø}}$	MEM5/$\overline{\text{AOUT}}$
$\overline{\text{OUT1Ø}}$	$\overline{\text{OUT1Ø}}$	$\overline{\text{OUT1Ø}}$	$\overline{\text{AIN}}$
$\overline{\text{AC1Ø}}$	$\overline{\text{AC1Ø}}$	$\overline{\text{AC1Ø}}$	$\overline{\text{AINPC}}$
$\overline{\text{BC1Ø}}$	$\overline{\text{BC1Ø}}$	$\overline{\text{BC1Ø}}$	$\overline{\text{AOUTC}}$
$\overline{\text{IN11}}$, PIN5	$\overline{\text{IN11}}$, PIN5	$\overline{\text{IN11}}$, PIN5	$\overline{\text{DOUT}}$
$\overline{\text{OUT11}}$	$\overline{\text{OUT11}}$	$\overline{\text{OUT11}}$	$\overline{\text{DIN}}$
$\overline{\text{AC11}}$	$\overline{\text{AC11}}$	$\overline{\text{AC11}}$	$\overline{\text{DINPC}}$
$\overline{\text{BC11}}$	$\overline{\text{BC11}}$	$\overline{\text{BC11}}$	$\overline{\text{DOUTC}}$
$\overline{\text{PCN5}}$	$\overline{\text{PCN5}}$	$\overline{\text{PCN5}}$	$\overline{\text{PCON}}$

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/INA12	LU1/INA12	LU2/INA12	SWØ/AOUT
INB12	INB12	INB12	BOUT
$\overline{\text{OUT12}}$			$\overline{\text{AINØ}}$
$\overline{\text{AC12}}$			$\overline{\text{ACONØ}}$
$\overline{\text{BC12}}$			$\overline{\text{BCONØ}}$
$\overline{\text{ASWØ}}$			ASELØ
BSWØ			BSELØ
LU1/$\overline{\text{OUT12}}$			AIN1
$\overline{\text{AC12}}$			ACON1
$\overline{\text{BC12}}$			BCON1
ASWØ			ASEL1
BSWØ			BSEL1
LU2/$\overline{\text{OUT12}}$			$\overline{\text{AIN2}}$
$\overline{\text{AC12}}$			$\overline{\text{ACON2}}$
$\overline{\text{BC12}}$			$\overline{\text{BCON2}}$
ASWØ			ASEL2
BSWØ			BSEL2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/INA13	LU1/INA13	LU2/INA13	SW1/AOUT
INB13	INB13	INB13	BOUT
OUT13			AINØ
AC13			ACONØ
BC13			BCONØ
ASW1			ASELØ
BSW1			BSELØ
LU1/OUT13			AIN1
AC13			ACON1
BC13			BCON1
ASW1			ASEL1
BSW1			BSEL1
LU2/OUT13			AIN1
AC13			ACON1
BC13			BCON1
ASW1			ASEL2
BSW1			BSEL2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/INA14	LU1/INA14	LU2/INA14	SW2/AOUT
INB14	INB14	INB14	BOUT
$\overline{\text{OUT14}}$			$\overline{\text{AINØ}}$
$\overline{\text{AC14}}$			$\overline{\text{ACONØ}}$
$\overline{\text{BC14}}$			$\overline{\text{BCONØ}}$
ASW2			ASELØ
BSW2			BSELØ
LU1/$\overline{\text{OUT14}}$			$\overline{\text{AIN1}}$
$\overline{\text{AC14}}$			$\overline{\text{ACON1}}$
BC14			$\overline{\text{BCON1}}$
ASW2			ASEL1
BSW2			BSEL1
LU2/$\overline{\text{OUT14}}$			$\overline{\text{AIN2}}$
$\overline{\text{AC14}}$			$\overline{\text{ACON2}}$
BC14			$\overline{\text{BCON2}}$
ASW2			ASEL2
BSW2			BSEL2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/INA15	LU1/INA15	LU2/INA15	SW3/AOUT
INB15	INB15	INB15	BOUT
$\overline{\text{OUT15}}$			AINØ
$\overline{\text{AC15}}$			ACONØ
$\overline{\text{BC15}}$			BCONØ
ASW3			ASELØ
BSW3			BSELØ
LU1/$\overline{\text{OUT15}}$			$\overline{\text{AIN1}}$
$\overline{\text{AC15}}$			$\overline{\text{ACON1}}$
$\overline{\text{BC15}}$			$\overline{\text{BCON1}}$
ASW3			ASEL1
BSW3			BSEL1
LU2/$\overline{\text{OUT15}}$			$\overline{\text{AIN2}}$
$\overline{\text{AC15}}$			$\overline{\text{ACON2}}$
$\overline{\text{BC15}}$			$\overline{\text{BCON2}}$
ASW3			ASEL2
BSW3			ASEL2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/INA16	LU1/INA16	LU2/INA16	SW4/AOUT
INB16	INB16	INB16	BOUT
$\overline{\text{OUT16}}$			$\overline{\text{AINØ}}$
$\overline{\text{AC16}}$			$\overline{\text{ACONØ}}$
BC16			$\overline{\text{BCONØ}}$
ASW4			ASELØ
BSW4			BSELØ
LU1/$\overline{\text{OUT16}}$			$\overline{\text{AIN1}}$
$\overline{\text{AC16}}$			$\overline{\text{ACON1}}$
$\overline{\text{BC16}}$			$\overline{\text{BCON1}}$
ASW4			ASEL1
BSW4			BSEL1
LU2/$\overline{\text{OUT16}}$			$\overline{\text{AIN2}}$
$\overline{\text{AC16}}$			$\overline{\text{ACON2}}$
$\overline{\text{BC16}}$			$\overline{\text{BCON2}}$
ASW4			ASEL2
BSW4			BSEL2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/INA17	LU1/INA17	LU2/INA17	SW5/AOUT
INB17	INB17	INB17	BOUT
$\overline{\text{OUT17}}$			$\overline{\text{AINØ}}$
$\overline{\text{AC17}}$			$\overline{\text{ACONØ}}$
$\overline{\text{BC17}}$			$\overline{\text{BCONØ}}$
ASW5			ASELØ
BSW5			BSELØ
LU1/$\overline{\text{OUT17}}$			$\overline{\text{AIN1}}$
$\overline{\text{AC17}}$			$\overline{\text{ACON1}}$
$\overline{\text{BC17}}$			$\overline{\text{BCON1}}$
ASW5			ASEL1
BSW5			BSEL1
LU2/$\overline{\text{OUT17}}$			$\overline{\text{AIN2}}$
$\overline{\text{AC17}}$			$\overline{\text{ACON2}}$
$\overline{\text{BC17}}$			$\overline{\text{BCON2}}$
ASW5			ASEL2
BSW5			BSEL2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/INA18	LU1/INA18	LU2/INA18	SW6/AOUT
INB18	INB18	INB18	BOUT
$\overline{\text{OUT18}}$			$\overline{\text{AINØ}}$
$\overline{\text{AC18}}$			$\overline{\text{ACONØ}}$
$\overline{\text{BC18}}$			$\overline{\text{BCONØ}}$
ASW6			ASELØ
BSW6			BSELØ
LU1/$\overline{\text{OUT18}}$			$\overline{\text{AIN1}}$
$\overline{\text{AC18}}$			$\overline{\text{ACON1}}$
$\overline{\text{BC18}}$			$\overline{\text{BCON1}}$
ASW6			ASEL1
BSW6			BSEL1
LU2/$\overline{\text{OUT18}}$			$\overline{\text{AIN2}}$
$\overline{\text{AC18}}$			$\overline{\text{ACON2}}$
$\overline{\text{BC18}}$			$\overline{\text{BCON2}}$
ASW6			ASEL2
BSW6			BSEL2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal	To Module/Signal
LUØ/INA19	LU1/INA19	LU2/INA19	SW7/AOUT
INB19	INB19	INB19	BOUT
$\overline{\mathrm{OUT19}}$			$\overline{\mathrm{AIN}Ø}$
$\overline{\mathrm{AC19}}$			$\overline{\mathrm{ACON}Ø}$
$\overline{\mathrm{BC19}}$			$\overline{\mathrm{BCON}Ø}$
ASW7			ASELØ
BSW7			BSELØ
LU1/$\overline{\mathrm{OUT19}}$			$\overline{\mathrm{AIN1}}$
$\overline{\mathrm{AC19}}$			$\overline{\mathrm{ACON1}}$
$\overline{\mathrm{BC19}}$			$\overline{\mathrm{BCON1}}$
ASW7			ASEL1
BSW7			BSEL1
LU2/$\overline{\mathrm{OUT19}}$			$\overline{\mathrm{AIN1}}$
$\overline{\mathrm{AC19}}$			$\overline{\mathrm{ACON1}}$
$\overline{\mathrm{BC19}}$			$\overline{\mathrm{BCON1}}$
ASW7			ASEL2
BSW7			BSEL2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/ Signal	To Module/ Signal	To Module/ Signal	To Module/ Signal	Tc Module/ Signal	To Module/ Signal	To Module/ Signal	To Module/ Signal	To Module/ Signal
LUØ/DAØ	SWØ/LØAØ	SW1/LØAØ	SW2/LØAØ	SW3/LØAØ	SW4/LØAØ	SW5/LØAØ	SW6/LØAØ	SW7/LØAØ
DA1	LØA1	LØA1	LØA1	LØA1	LØA1	LØA1	LØA1	LØA1
DA2	LØA2	LØA2	LØA2	LØA2	LØA2	LØA2	LØA2	LØA2
DA3	LØA3	LØA3	LØA3	LØA3	LØA3	LØA3	LØA3	LØA3
DBØ	LØBØ	LØBØ	LØBØ	LØBØ	LØBØ	LØBØ	LØBØ	LØBØ
DB1	LØB1	LØB1	LØB1	LØB1	LØB1	LØB1	LØB1	LØB1
DB2	LØB2	LØB2	LØB2	LØB2	LØB2	LØB2	LØB2	LØB2
DB3	LØB3	LØB3	LØB3	LØB3	LØB3	LØB3	LØB3	LØB3
LU1/DAØ	L1AØ	L1AØ	L1AØ	L1AØ	L1AØ	L1AØ	L1AØ	L1AØ
DA1	L1A1	L1A1	L1A1	L1A1	L1A1	L1A1	L1A1	L1A1
DA2	L1A2	L1A2	L1A2	L1A2	L1A2	L1A2	L1A2	L1A2
DA3	L1A3	L1A3	L1A3	L1A3	L1A3	L1A3	L1A3	L1A3
DBØ	L1BØ	L1BØ	L1BØ	L1BØ	L1BØ	L1BØ	L1BØ	L1BØ
DB1	L1B1	L1B1	L1B1	L1B1	L1B1	L1B1	L1B1	L1B1
DB2	L1B2	L1B2	L1B2	L1B2	L1B2	L1B2	L1B2	L1B2
DB3	L1B3	L1B3	L1B3	L1B3	L1B3	L1B3	L1B3	L1B3
LU2/DAØ	L2AØ	L2AØ	L2AØ	L2AØ	L2AØ	L2AØ	L2AØ	L2AØ
DA1	L2A1	L2A1	L2A1	L2A1	L2A1	L2A1	L2A1	L2A1
DA2	L2A2	L2A2	L2A2	L2A2	L2A2	L2A2	L2A2	L2A2
DA3	L2A3	L2A3	L2A3	L2A3	L2A3	L2A3	L2A3	L2A3
DBØ	L2BØ	L2BØ	L2BØ	L2BØ	L2BØ	L2BØ	L2BØ	L2BØ
DB1	L2B1	L2B1	L2B1	L2B1	L2B1	L2B1	L2B1	L2B1
DB2	L2B2	L2B2	L2B2	L2B2	L2B2	L2B2	L2B2	L2B2
DB3	L2B3	L2B3	L2B3	L2B3	L2B3	L2B3	L2B3	L2B3

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal
SWØ/$\overline{\text{INØ}}$	SW1/$\overline{\text{INØ}}$	DAØ/$\overline{\text{DOUT}}$
$\overline{\text{OUTØ}}$	$\overline{\text{OUTØ}}$	$\overline{\text{DIN}}$
$\overline{\text{ACØ}}$	$\overline{\text{ACØ}}$	$\overline{\text{INPC}}$
$\overline{\text{BCØ}}$	$\overline{\text{BCØ}}$	$\overline{\text{OUTC}}$
$\overline{\text{IN1}}$	CMØ/$\overline{\text{CDAT3}}$	RØ/$\overline{\text{DOUT}}$
$\overline{\text{OUT1}}$	$\overline{\text{CCN3}}$	$\overline{\text{DIN}}$
$\overline{\text{AC1}}$		$\overline{\text{INPC}}$
$\overline{\text{BC1}}$		$\overline{\text{OUTC}}$
$\overline{\text{IN2}}$		R1/$\overline{\text{DOUT}}$
$\overline{\text{OUT2}}$		$\overline{\text{DIN}}$
$\overline{\text{AC2}}$		$\overline{\text{INPC}}$
$\overline{\text{BC2}}$		$\overline{\text{OUTC}}$
$\overline{\text{IN3}}$		R2/$\overline{\text{DOUT}}$
$\overline{\text{OUT3}}$		$\overline{\text{DIN}}$
$\overline{\text{AC3}}$		$\overline{\text{INPC}}$
$\overline{\text{BC3}}$		$\overline{\text{OUTC}}$

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal
SW1/IN1	R3/DOUT	
$\overline{\text{OUT1}}$	$\overline{\text{DIN}}$	CM1/$\overline{\text{CDAT3}}$
$\overline{\text{AC1}}$	$\overline{\text{INPC}}$	$\overline{\text{CCN3}}$
$\overline{\text{BC1}}$	$\overline{\text{OUTC}}$	
IN2	R4/DOUT	
$\overline{\text{OUT2}}$	$\overline{\text{DIN}}$	
$\overline{\text{AC2}}$	$\overline{\text{INPC}}$	
$\overline{\text{BC2}}$	$\overline{\text{OUTC}}$	
IN3	R5/DOUT	
$\overline{\text{OUT3}}$	$\overline{\text{DIN}}$	
$\overline{\text{AC3}}$	$\overline{\text{INPC}}$	
$\overline{\text{BC3}}$	$\overline{\text{OUTC}}$	

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal
SW2/INØ	SW3/INØ	DA1/DOUT
OUTØ	OUTØ	DIN
ACØ	ACØ	INPC
BCØ	BCØ	OUTC
IN1		R6/DOUT
OUT1	CMØ/CDAT4	DIN
AC1	CCN4	INPC
BC1		OUTC
IN2		R7/DOUT
OUT2		DIN
AC2		INPC
BC2		OUTC
IN3		R8/DOUT
OUT3		DIN
AC3		INPC
BC3		OUTC

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal
SW3/IN1	R9/DOUT	
$\overline{\text{OUT1}}$	$\overline{\text{DIN}}$	CM1/$\overline{\text{CDAT4}}$
$\overline{\text{AC1}}$	$\overline{\text{INPC}}$	$\overline{\text{CCN4}}$
$\overline{\text{BC1}}$	$\overline{\text{OUTC}}$	
IN2	R10/DOUT	
$\overline{\text{OUT2}}$	$\overline{\text{DIN}}$	
$\overline{\text{AC2}}$	$\overline{\text{INPC}}$	
$\overline{\text{BC2}}$	$\overline{\text{OUTC}}$	
IN3	R11/DOUT	
$\overline{\text{OUT3}}$	$\overline{\text{DIN}}$	
$\overline{\text{AC3}}$	$\overline{\text{INPC}}$	
$\overline{\text{BC3}}$	$\overline{\text{OUTC}}$	

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal
SW4/$\overline{\text{IN}\emptyset}$	SW5/$\overline{\text{IN}\emptyset}$	TMØ/$\overline{\text{DATAOUT}}$
$\overline{\text{OUT}\emptyset}$	$\overline{\text{OUT}\emptyset}$	$\overline{\text{DATAIN}}$
$\overline{\text{AC}\emptyset}$	$\overline{\text{AC}\emptyset}$	$\overline{\text{INPC}}$
$\overline{\text{BC}\emptyset}$	$\overline{\text{BC}\emptyset}$	$\overline{\text{OUTC}}$
$\overline{\text{IN1}}$		R12/$\overline{\text{DOUT}}$
$\overline{\text{OUT1}}$	CMØ/$\overline{\text{CDAT5}}$	$\overline{\text{DIN}}$
$\overline{\text{AC1}}$	$\overline{\text{CCN5}}$	$\overline{\text{INPC}}$
$\overline{\text{BC1}}$		$\overline{\text{OUTC}}$
$\overline{\text{IN2}}$		R13/$\overline{\text{DOUT}}$
$\overline{\text{OL}}$?		$\overline{\text{DIN}}$
$\overline{\text{AC2}}$		$\overline{\text{INPC}}$
$\overline{\text{BC2}}$		$\overline{\text{OUTC}}$
$\overline{\text{IN3}}$		R14/$\overline{\text{DOUT}}$
$\overline{\text{OUT3}}$		$\overline{\text{DIN}}$
$\overline{\text{AC3}}$		$\overline{\text{INPC}}$
$\overline{\text{BC3}}$		$\overline{\text{OUTC}}$

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal
SW5/IN1	R15/DOUT	CM1/$\overline{\text{CDAT5}}$
OUT1	DIN	$\overline{\text{CCN5}}$
AC1	INPC	
BC1	OUTC	
IN2	R16/DOUT	
OUT2	DIN	
AC2	INPC	
BC2	OUTC	
IN3	R17/DOUT	
OUT3	DIN	
AC3	INPC	
BC3	OUTC	

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal
SW6/INØ	SW7/INØ	TM1/DATAOUT
OUTØ	OUTØ	DATAIN
ACØ	ACØ	INPC
BCØ	BCØ	OUTC
IN1		R18/DOUT
OUT1	CMØ/CDAT6	DIN
AC1	CCN6	INPC
BC1		OUTC
IN2		R19/DOUT
OUT2		DIN
AC2		INPC
BC2		OUTC
IN3		R20/DOUT
OUT3		DIN
AC3		INPC
BC3		OUTC

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal
SW7/IN1	R21/DOUT	
OUT1	DIN	CM1/$\overline{\text{CDAT6}}$
AC1	INPC	$\overline{\text{CCN6}}$
BC1	OUTC	
IN2	R22/DOUT	
OUT2	DIN	
AC2	INPC	
BC2	OUTC	
IN3	R23/DOUT	
OUT3	DIN	
AC3	INPC	
BC3	OUTC	

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal	To Module/Signal
CMØ/$\overline{\mathrm{CCN}\emptyset}$	CM1/$\overline{\mathrm{CCN}\emptyset}$	LUØ/$\overline{\mathrm{CCN}}$
$\overline{\mathrm{CDAT}\emptyset}$	$\overline{\mathrm{CDAT}\emptyset}$	$\overline{\mathrm{CIN}}$
$\overline{\mathrm{CCN1}}$	$\overline{\mathrm{CCN1}}$	LU1/$\overline{\mathrm{CCN}}$
$\overline{\mathrm{CDAT1}}$	$\overline{\mathrm{CDAT1}}$	$\overline{\mathrm{CIN}}$
$\overline{\mathrm{CCN2}}$	$\overline{\mathrm{CCN2}}$	LU2/$\overline{\mathrm{CCN}}$
$\overline{\mathrm{CDAT2}}$	$\overline{\mathrm{CDAT2}}$	$\overline{\mathrm{CIN}}$

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/SC1A	LUØ/SC1
	LU1/SC1
	LU2/SC1
	MEMØ/SC1
	MEM1/SC1
	MEM2/SC1
	MEM3/SC1
	MEM4/SC1
	MEM5/SC1
	DAØ/SC1
	DA1/SC1
	TMØ/SC1
	TM1/SC1
	CMØ/SC1
	CM1/SC1

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/SC2A	LUØ/SC2
	LU1/SC2
	LU2/SC2
	MEMØ/SC2
	MEM1/SC2
	MEM2/SC2
	MEM3/SC2
	MEM4/SC2
	MEM5/SC2
	DAØ/SC2
	DA1/SC2
	TMØ/SC2
	TM1/SC2
	CMØ/SC2
	CM1/SC2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/SC3A	LUØ/SC3
	LU1/SC3
	LU2/SC3
	MEMØ/SC3
	MEM1/SC3
	MEM2/SC3
	MEM3/SC3
	MEM4/SC3
	MEM5/SC3
	DAØ/SC3
	DA1/SC3
	TMØ/SC3
	TM1/SC3
	CMØ/SC3
	CM1/SC3

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/SC1B	RØ/SC1
	R1/SC1
	R2/SC1
	R3/SC1
	R4/SC1
	R5/SC1
	R6/SC1
	R7/SC1
	R8/SC1
	R9/SC1
	R10/SC1
	R11/SC1
TG/SC2B	RØ/SC2
	R1/SC2
	R2/SC2
	R3/SC2
	R4/SC2
	R5/SC2
	R6/SC2
	R7/SC2
	R8/SC2
	R9/SC2
	R10/SC2
	R11/SC2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/SC3B	RØ/SC3
	R1/SC3
	R2/SC3
	R3/SC3
	R4/SC3
	R5/SC3
	R6/SC3
	R7/SC3
	R8/SC3
	R9/SC3
	R10/SC3
	R11/SC3
TG/SC1C	R12/SC1
	R13/SC1
	R14/SC1
	R15/SC1
	R16/SC1
	R17/SC1
	R18/SC1
	R19/SC1
	R20/SC1
	R21/SC1
	R22/SC1
	R23/SC1

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/SC2C	R12/SC2
	R13/SC2
	R14/SC2
	R15/SC2
	R16/SC2
	R17/SC2
	R18/SC2
	R19/SC2
	R20/SC2
	R21/SC2
	R22/SC2
	R23/SC2
TG/SC3C	R12/SC3
	R13/SC3
	R14/SC3
	R15/SC3
	R16/SC3
	R17/SC3
	R18/SC3
	R19/SC3
	R20/SC3
	R21/SC3
	R22/SC3
	R23/SC3

TABLE 20.-- Continued

MUTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/WS1A	LUØ/WS1
	LU1/WS1
	LU2/WS1
	MEMØ/WS1
	MEM1/WS1
	MEM2/WS1
	MEM3/WS1
	MEM4/WS1
	MEM5/WS1
	DAØ/WS1
	DA1/WS1
	TMØ/WS1
	TM1/WS1
	CMØ/SW1
	CM1/SW1
TG/WS2A	LUØ/WS2
	LU1/WS2
	LU2/WS2
	MEMØ/WS2
	MEM1/WS2
	MEM2/WS2
	MEM3/WS2
	MEM4/WS2
	MEM5/WS2
	DAØ/WS2
	DA1/WS2
	TMØ/WS2
	TM1/WS2
	CMØ/WS2
	CM1/WS2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/WS3A	LUØ/WS3
	LU1/WS3
	LU2/WS3
	MEMØ/WS3
	MEM1/WS3
	MEM2/WS3
	MEM3/WS3
	MEM4/WS3
	MEM5/WS3
	DAØ/WS3
	DA1/WS3
	TMØ/WS3
	TM1/WS3
	CMØ/WS3
	CM1/WS3

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/WS1B	RØ/WS1
	R1/WS1
	R2/WS1
	R3/WS1
	R4/WS1
	R5/WS1
	R6/WS1
	R7/WS1
	R8/WS1
	R9/WS1
	R10/WS1
	R11/WS1
	R12/WS1
	R13/WS1
	R14/WS1
	R15/WS1
	R16/WS1
	R17/WS1
	R18/WS1
	R19/WS1
	R20/WS1
	R21/WS1
	R22/WS1
	R23/WS1

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/WS2B	RØ/WS2
	R1/WS2
	R2/WS2
	R3/WS2
	R4/WS2
	R5/WS2
	R6/WS2
	R7/WS2
	R8/WS2
	R9/WS2
	R10/WS2
	R11/WS2
	R12/WS2
	R13/WS2
	R14/WS2
	R15/WS2
	R16/WS2
	R17/WS2
	R18/WS2
	R19/WS2
	R20/WS2
	R21/WS2
	R22/WS2
	R23/WS2

TABLE 20.-- Continued

MULTIPAC MODULE INTERCONNECTIONS

From Module/Signal	To Module/Signal
TG/WS3B	RØ/WS3
	R1/WS3
	R2/WS3
	R3/WS3
	R4/WS3
	R5/WS3
	R6/WS3
	R7/WS3
	R8/WS3
	R9/WS3
	R10/WS3
	R11/WS3
	R12/WS3
	R13/WS3
	R14/WS3
	R15/WS3
	R16/WS3
	R17/WS3
	R18/WS3
	R19/WS3
	R20/WS3
	R21/WS3
	R22/WS3
	R23/WS3

APPENDIX C

SPECIFICATIONS FOR INTEGRATED CIRCUITS

"Reproduced by permission of Fairchild Semiconductor, a Division of Fairchild Camera & Instrument Corporation".

JANUARY 1966 ■

LPDTµL 9040, 9041 AND 9042
LOW POWER DIODE TRANSISTOR MICROLOGIC®
INTEGRATED CIRCUITS

GENERAL DESCRIPTION

The Fairchild LPDTµL Micrologic® Integrated Circuit Family consists of a set of compatible, integrated logic circuits specifically designed for low power, medium speed applications.

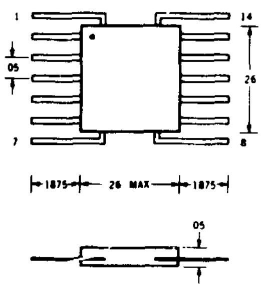
The circuits are fabricated with a silicon monolithic substrate using standard Fairchild Planar* epitaxial processes.

Packaging options include the Flat package and the Dual In-Line package.

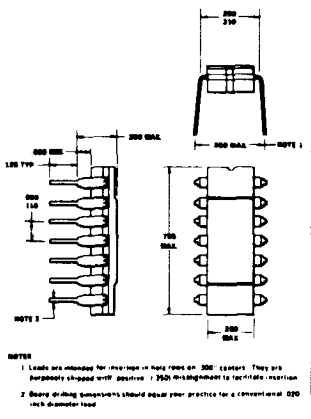
Important features of the LPDTµL Micrologic® integrated circuits include the following:

- Reliable operation over the full military temperature range of -55°C to +125°C
- Typical power drains of less than 1 mW per gate (50% duty cycle) for the logic gate elements and less than 4 mW for the clocked flip-flop.
- Single power supply requirement—5 volts optimum, 4.5 to 5.5 volts range.
- Guaranteed fan-out of 10 LPDTµL unit loads or 1 standard Fairchild DTµL unit load, over the full temperature and supply voltage range.
- Guaranteed minimum of 450 mV noise immunity at the temperature extremes.
- Typical logic gate propagation delays of 60 ns and binary clock rate of 2.5 MHz.
- Emitter follower outputs providing good capacitive drive capability.

TYPICAL FLAT PACKAGE
TOP VIEW



TYPICAL DUAL IN-LINE PACKAGE


.290
.310
.200 MAX
.120 TYP
.750 MAX
.350 MAX
NOTE 1
NOTE 2
.260 MAX
NOTES
1 Leads are intended for insertion in hole rows on .300" centers. They are purposely shipped with positive (.350) misalignment to facilitate insertion
2 Board drilling dimensions should equal your practice for a conventional .020 inch diameter lead

*Planar is a patented Fairchild process.

ORDER INFORMATION

To order Low Power Diode Transistor Micrologic® integrated circuit elements specify U31XXXX51X for flat package and U6AXXXX51X for Dual In-Line package where XXXX is 9040, 9041 or 9042.

FAIRCHILD
SEMICONDUCTOR
A DIVISION OF FAIRCHILD CAMERA AND INSTRUMENT CORPORATION

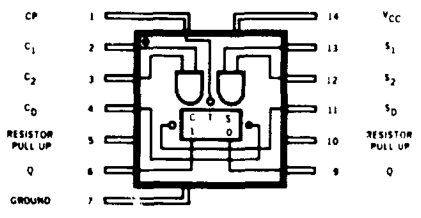
313 FAIRCHILD DRIVE, MOUNTAIN VIEW, CALIFORNIA, (415) 962-5011, TWX: 910-379-6435

LPDTμL 9040 CLOCKED FLIP-FLOP

DESCRIPTION

The LPDTμL 9040 element is a directly coupled, dual-rank flip-flop suitable for use in counters, shift registers and other storage applications. Either R-S or J-K mode operation is possible. Direct set and clear inputs are provided which override all other data inputs.

LOGIC DIAGRAM SHOWING FLAT OR DUAL-IN LINE PACKAGE PIN ASSIGNMENT

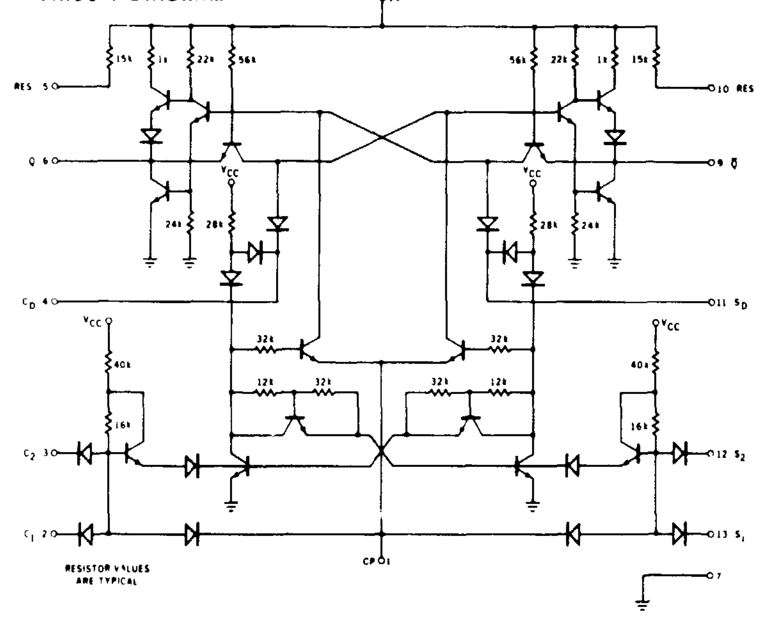


SYNCHRONOUS ENTRY TRUTH TABLES										ASYCHRONOUS ENTRY TRUTH TABLE			
R-S MODE OPERATION						J-K MODE OPERATION							
INPUTS @ t_n				OUTPUTS @ t_{n+1}		INPUTS @ t_n		OUTPUTS @ t_{n+1}		INPUTS		OUTPUTS	
S_1	S_2	C_1	C_2	Q	$\overline{Q}$	S_1	C_1	Q	$\overline{Q}$	S_D	C_D	Q	$\overline{Q}$
13	12	2	3	6	9	13	2	6	9	11	4	6	9
L	X	L	X	NC	NC	L	L	NC	NC	H	H	NC	NC
L	X	X	L	NC	NC	L	H	L	H	H	L	L	H
X	L	L	X	NC	NC	H	L	H	L	L	H	H	L
X	L	X	L	NC	NC	H	H	TOGGLED		L	L	H	H
L	X	H	H	L	H								
X	L	H	H	L	H								
H	H	L	X	H	L								
H	H	X	L	H	L								
H	H	H	H	AMBIGUOUS									

Symbols
- H - Most positive logic level
- L - Most negative logic level
- X - Either H or L can be present
- NC - No change in state

NOTES:
1. For J-K mode operation connect Pin 6 to Pin 3 and Pin 9 to Pin 12.
2. Asynchronous entries override all synchronous entries.

CIRCUIT DIAGRAM



LOADING RULES

INPUT	*NORMALIZED UNIT LOADS (U.L.)
S_1 S_2 / C_1 C_2	0.75 U.L.
S_D C_D	2.5 U.L.
CP	2.5 U.L.
OUTPUT	FAN-OUT
Q, $\overline{Q}$	10 U.L. 7 U.L. WITH RESISTOR PULL-UP CONNECTED

*1UNIT LOAD EQUALS 1-LPDTμL 9041 OR 9042 INPUT LOAD

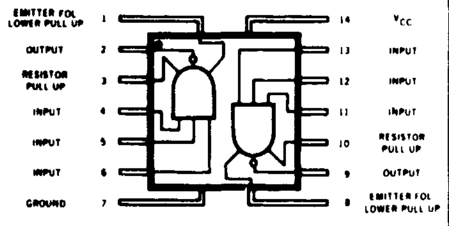
FAIRCHILD MICROLOGIC® LOW POWER DIODE TRANSISTOR INTEGRATED CIRCUITS

LPDTµL 9041 – DUAL 3 INPUT NAND GATE

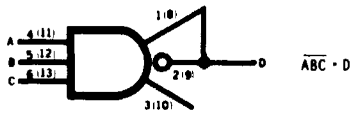
DESCRIPTION

The LPDTµL 9041 element consists of two, 3-input positive logic NAND gates suitable for general logic gate and inverter applications. The unique feature of this gate is that the output transistor collector and the emitter follower pull-up are not internally connected. This allows the user to tie collectors to a common node for the wired "OR" logic function.

LOGIC DIAGRAM SHOWING FLAT OR DUAL-IN-LINE PACKAGE PIN ASSIGNMENT



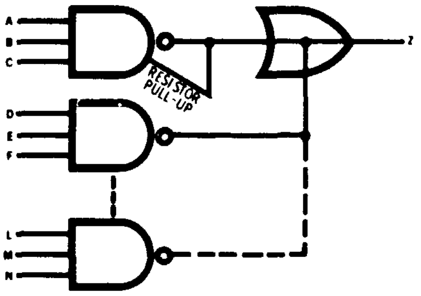
POSITIVE LOGIC NAND GATE



EACH INPUT = 1 UNIT LOAD
OUTPUT FAN-OUT = 10 UNIT LOADS
= 7 U.L. WITH RESISTOR PULL-UP CONNECTED

EITHER THE EMITTER FOLLOWER OR RESISTOR PULL-UP MUST BE CONNECTED TO THE OUTPUT TO ESTABLISH THE HIGH LEVEL.

WIRED 'OR' APPLICATION

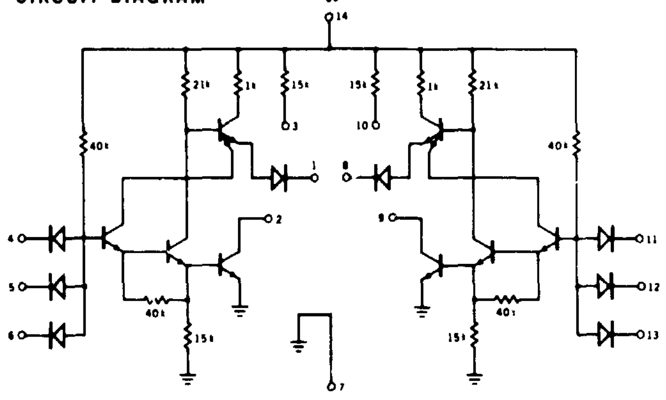


$\overline{ABC} + \overline{DEF} + \cdots + \overline{LMN} = Z$

OUTPUT FAN-OUT = 10 - 3 (NO. OF RESISTOR PULL-UPS)

ONE PULL-UP RESISTOR IS REQUIRED FOR EVERY 8 GATES CONNECTED TO THE COMMON "OR" NODE.

CIRCUIT DIAGRAM



RESISTOR VALUES ARE TYPICAL

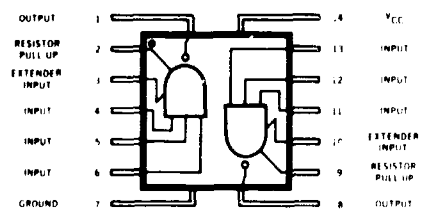
LPDTμL 9042 – DUAL 3 INPUT NAND GATE WITH EXTENDER INPUTS

DESCRIPTION

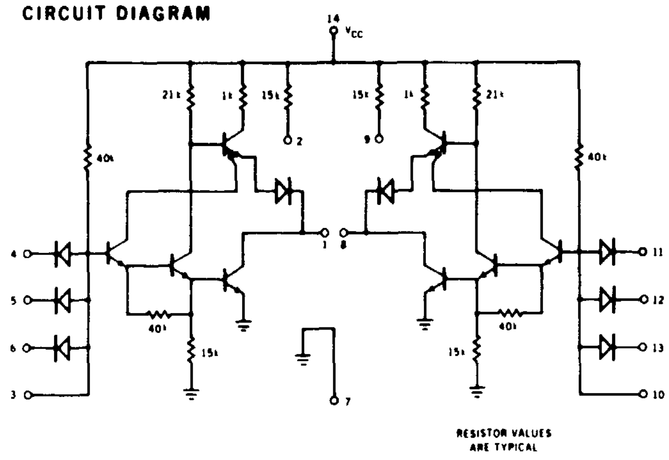
The LPDTμL 9042 element consists of two 3-input positive logic NAND gates with extender inputs. This element in the family allows the user to implement logic applications requiring a gate fan-in exceeding three.

The DTμL 9933 4-input extender element or equivalent—may be used to provide additional diode inputs. Any capacitance added to the extender input will increase the turn-on delay of the LPDTμL 9042 gate. Typically, the increase is 10 ns/picofarad. Turn-off delay is not affected.

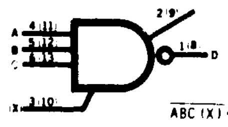
LOGIC DIAGRAM SHOWING FLAT OR DUAL-IN-LINE PACKAGE PIN ASSIGNMENT



CIRCUIT DIAGRAM



POSITIVE LOGIC NAND GATE



$\overline{ABC\,(X)} = D$

EACH INPUT = 1 UNIT LOAD

OUTPUT FAN-OUT = 10 UNIT LOADS

= 7 UNIT LOADS WITH RESISTOR PULL-UP CONNECTED

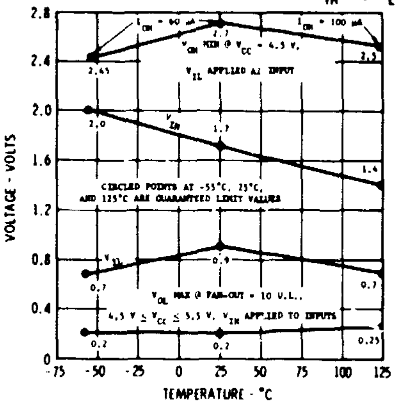
BUFFER ELEMENT

For applications requiring a fan-out exceeding ten, the Fairchild DTμL 9930 Dual 4-Input Gate may be used. The DTμL 9930 will drive 44 LPDTμL unit loads, while maintaining the same output logic levels as the low power circuits.

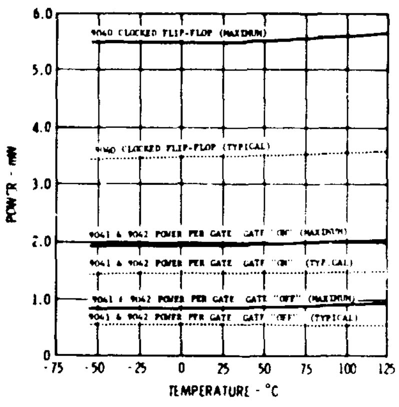
The input of a DTμL 9930 requires the equivalent of 10 LPDTμL unit loads. Therefore, a low power circuit can drive only one DTμL 9930 input.

OPERATING VOLTAGE CHARACTERISTICS

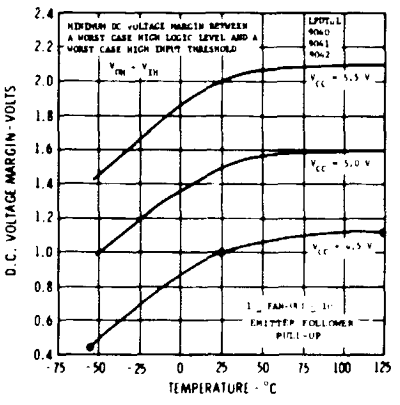
WORST CASE **OUTPUT LOGIC LEVELS - V_{OH} AND V_{OL}**
INPUT THRESHOLD LEVELS - V_{IH} AND V_{IL}



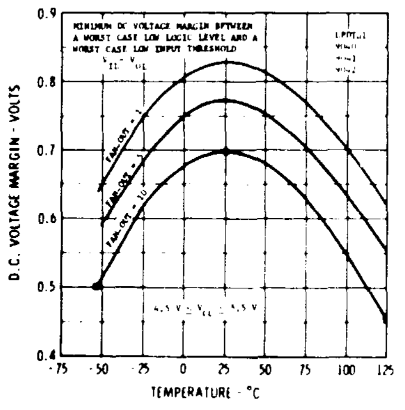
POWER CHARACTERISTICS
VCC = 5V
EMITTER FOLLOWER PULL UP



HIGH LEVEL NOISE IMMUNITY

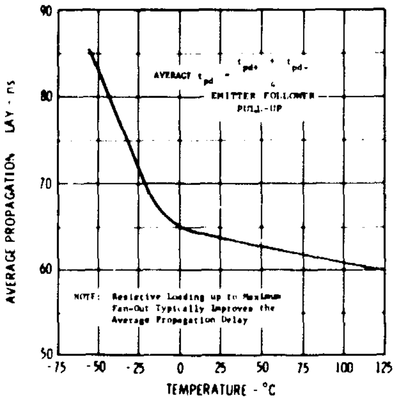


LOW LEVEL NOISE IMMUNITY

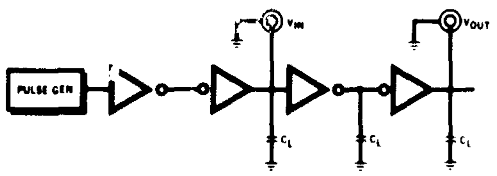


FAIRCHILD MICROLOGIC® LOW POWER DIODE TRANSISTOR INTEGRATED CIRCUITS

TYPICAL AVERAGE PROPAGATION DELAY LPDTμL 9041 • 9042



TEST CIRCUIT

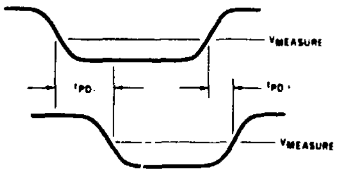


CONDITIONS

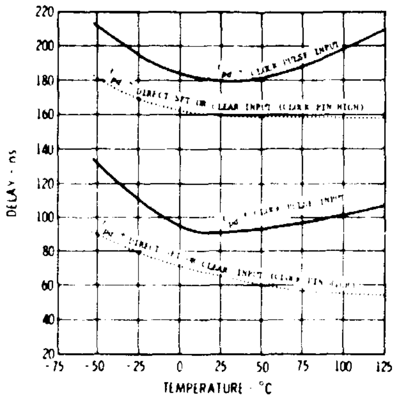
V_{CC} = 5.0V C_L = 50pf (INCLUDING PROBE AND JIG CAPACITANCE)

$V_{MEASURE}$ = 1.6V @ -55°C
(GND REF) 1.3V @ 25°C
0.9V @ 125°C

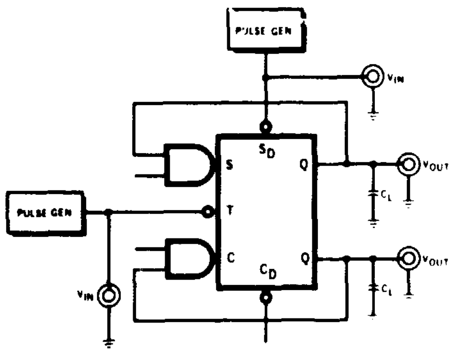
WAVE FORMS



TYPICAL DELAY CHARACTERISTICS LPDTμL 9040



TEST CIRCUIT

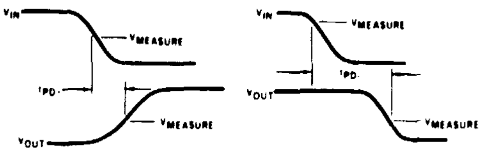


CONDITIONS

V_{CC} = 5.0V C_L = 50pf (INCLUDING PROBE AND JIG CAPACITY)

$V_{MEASURE}$ 1.6V @ -55°C
(GND REF) 1.3V @ 25°C
0.9V @ 125°C

WAVE FORMS



Fairchild cannot assume responsibility for use of any circuitry described other than circuitry entirely embodied in a Fairchild product. No other circuit patent licenses are implied.

CCSL COMPOSITE DATA SHEET

COMPATIBLE CURRENT SINKING LOGIC

-55°C TO +125°C TEMPERATURE RANGE

CCSL LOADING RULES

The first step towards realization of a compatible logic family is to establish optimized input-output logic levels. These levels determine the noise immunity for all the elements, as well as the basis for system interfacing.

Fairchild CCSL loading rules guarantee the optimum logic levels over the full military temperature range of -55°C to +125 C with V_{CC} supply within the range of 5V ± 0.5V. These same logic levels, as well as the input load and output drive factors are also guaranteed over the temperature range of -20°C to +100°C, for all CCSL elements. These guaranteed levels are:

Low level output voltage (V_{OL}) = 0.4V
High level output voltage (V_{OH}) = 2.5V
Low level input voltage (V_{IL}) = 0.7V
High level input voltage (V_{IH}) = 2.1V

Noise immunity is derived from the above numbers according to the following equations.

1. High level noise immunity = $V_{OH} - V_{IH}$
2. Low level noise immunity = $V_{IL} - V_{OL}$

Once the logic levels and DC noise margins are established, interfacing rules can be resolved. To simplify input loading and output drive capability, load factors and drive factors were assigned to each element.

These factors are written as a ratio, but are not defined as an arithmetic ratio. The numerator can be added or subtracted independent of the denominator and vice versa. This ratio form was chosen for convenient loading rule analysis.

Load Factor = $\frac{\text{High Level Load Factor}}{\text{Low Level Load Factor}}$ (Shown as ratio on inputs to circuits)

Drive Factor = $\frac{\text{High Level Drive Factor}}{\text{Low Level Drive Factor}}$ (Shown as ratio on outputs of circuits)

Where:

High Level Load Factor = Input current drawn into the inputs, during the High Input Level State.
Low Level Load Factor = Input current drawn out of the input during the Low Input Level State.
High Level Drive Factor = Ability of the output to supply current out of the output during the High Output Level State.
Low Level Drive Factor = Ability of the output to sink current into the output during the Low Output Level State.

A necessary condition is that the High Level Drive Factor must be equal to or greater than the sum of the Driven High Level Load Factors and the Low Level Drive Factor must be equal to or greater than the sum of the Driven Low Level Load Factors. Both High Level Drive and Load factors and Low Level Drive and Load factors must be considered if efficient interfacing is to be accomplished.

FAIRCHILD
SEMICONDUCTOR
A DIVISION OF FAIRCHILD CAMERA AND INSTRUMENT CORPORATION

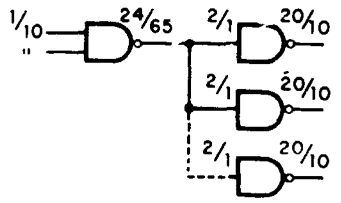
313 FAIRCHILD DRIVE, MOUNTAIN VIEW, CALIFORNIA, (415) 962-5011, TWX: 910-379-6435

CCSL COMPOSITE DATA SHEET

The load factors given are based on worst case conditions at both -55°C and +125°C. Input Low Level Load Factors are guaranteed with the Low Level Output Voltage (V_{OL}) applied to the inputs. Input High Level Load Factors are tested with arbitrarily selected voltages much higher than the V_{CH} value.

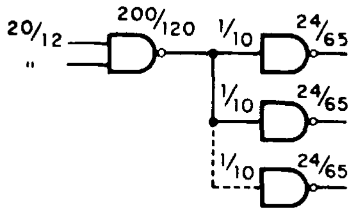
If the temperature range is restricted to -40°C to +110°C or the V_{CC} range to 4.75 to 5.25V, a 10% increase in drive factors may be used.

Examples: A.



(1) DTuL Gate (6K pullup) driving LPDTL Gates

B.



(2) TTL gate driving DTL Gates

Example A

Maximum load = 12 LPDTL gates. Limited by the high level drive capabilities. In this case the Low Level Load is only 12, and the drive capability is 65. Reference to the curves shown in Figure 1, will show this node to have a maximum V_{OL} = 100mV and therefore a worst case Low level noise immunity of 0.6 volts.

Example B

Maximum load - 12 DTL gates, limited by low level drive capabilities. Here the high level load = 12 and the drive capability is 200.

LIMITED LOADING APPLICATIONS

The curves shown in Figures 1, and 2 show improvements in the low level noise margin for TTL and DTμL. DTμL is shown with 6K or 2K pullup resistors.

Example: A TTL gate type output can drive a maximum of 120 low level loads and meet a guaranteed V_{OL} of 0.40 volts. If, however, this same output is only driving 50 low level loads, the worst case V_{OL} at that output would be 0.20 volts and the worst case low level noise immunity would be 0.50 volts.

Figure 2 shows similar low level curves for the TTL and DTμL Buffers.

Figure 3 shows the increase of V_{OH} of a DTμL output high level if the high level loads are less than the maximum specified in the CCSL loading rules. This drive is determined by using a -30% tolerence resistor and is guaranteed by D.C. testing.

Figure 4 shows changes in V_{OH} for DTμL devices which have 2K pullup resistor. The test point guarantees a V_{OH} 4.0 volts with 24 high level loads being driven. The increased number of loads which can be driven and still meet the CCSL V_{OH} of 2.5 volts is based on a worst case maximum tolerance pullup resistor as determined by D.C. testing. If less than 24 high level loads are driven, the increase in V_{OH} is determined by the worst case minimum tolerance pullup resistor.

TTL and LPDTμL devices having active pullups are relatively immune to changes in the high level loads. Their V_{OH} level is primarily set by being $2V_{BE}$'s below V_{CC}.

CCSL COMPOSITE DATA SHEET

SPECIAL APPLICATIONS

When using the DTμL 9944 element, the following currents represent unit loads, and must be considered when choosing different external collector resistors.

		-55 C	-25 C	-125 C
V_{CC} = 4.5V	High Level	$3\mu A$	$3\mu A$	$5\mu A$
	Low Level	.121mA	Not worst case	Not worst case
V_{CC} = 5.5V	High Level	Not worst case	Not worst case	Not worst case
	Lower Level	.151mA	.154mA	.146mA

"WIRED OR" APPLICATIONS

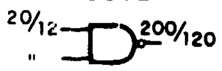
For elements 9930, 9962, 9946 and 9936 add 24 to the High Level Drive Factor and subtract 9 from the Low Level Drive Factor for each added gate.

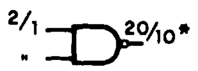
For elements 9041, 9043, 9046 and 9048 using one internal 15Ω pull-up resistor the High Level Drive Factor becomes 14 and the Low Level Drive Factor becomes 7. Subtract 2 from the High Level Drive Factor for each added gate.

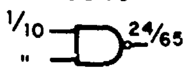
Note 1
If the minimum temperature is limited to -30°C, the Low Level Drive Factor is 12.

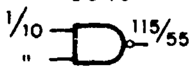
CCSL INPUT LOAD & DRIVE FACTORS

QUAD 2-INPUT NAND GATES

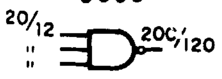
9002


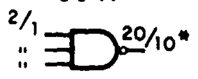
9046


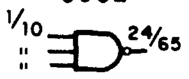
9946


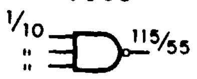
9949


TRIPLE 3-INPUT NAND GATES

9003


9047


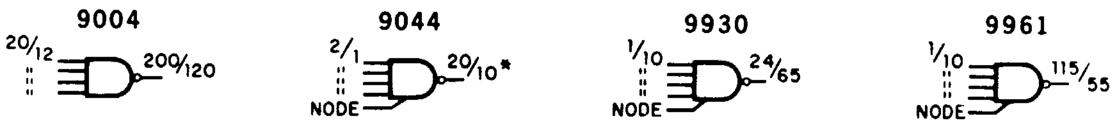
9962


9963


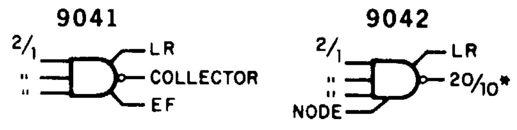
*See Note 1

CCSL INPUT LOAD & DRIVE FACTORS

DUAL 4-INPUT NAND GATES

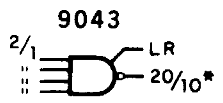


DUAL 3-INPUT NAND GATES



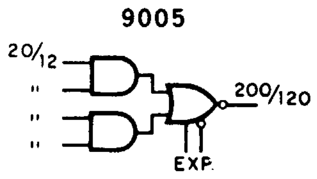
LR=LOAD RESISTOR OUTPUT, EF=EMITTER FOLLOWER OUTPUT

3 & 4-INPUT NAND GATES

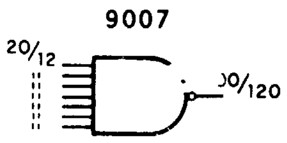


LR= LOAD RESISTOR OUTPUT

DUAL AND/NOR FUNCTION



EIGHT-INPUT NAND GATE

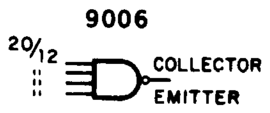


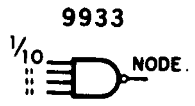
*See Note 1

CCSL COMPOSITE DATA SHEET

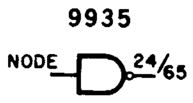
CCSL INPUT LOAD & DRIVE FACTORS

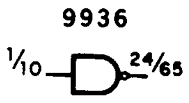
DUAL 4-INPUT EXPANDERS

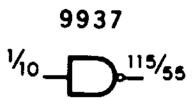




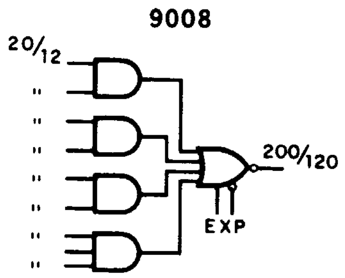
HEX INVERTERS



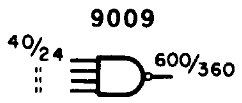


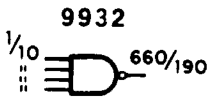


2-2-2-3-INPUT AND/NOR FUNCTION

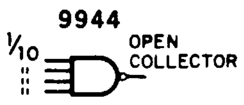


DUAL 4-INPUT BUFFERS





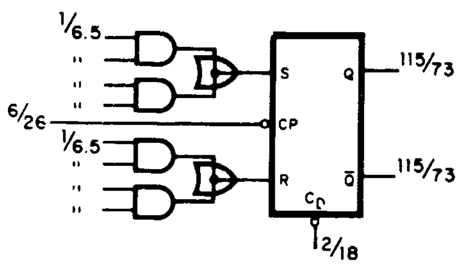
DUAL 4-INPUT DRIVER



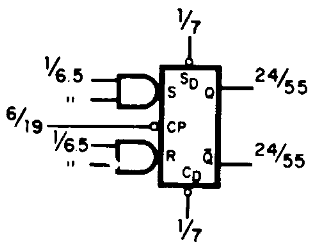
CCSL INPUT LOAD & DRIVE FACTORS

R-S FLIP-FLOPS

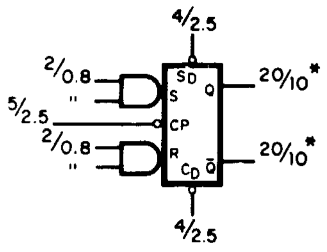
9111



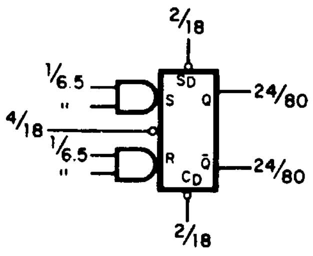
9931



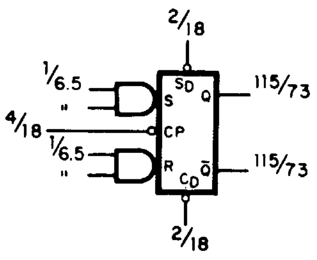
9040



9945

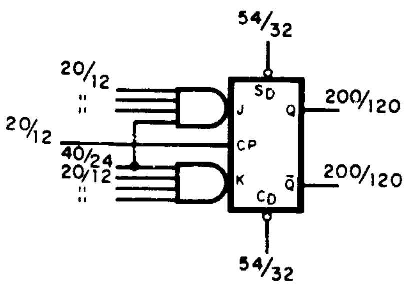


9948

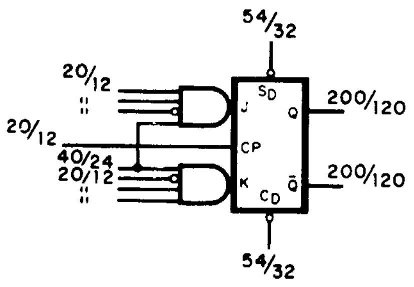


J-K FLIP-FLOPS

9000



9001



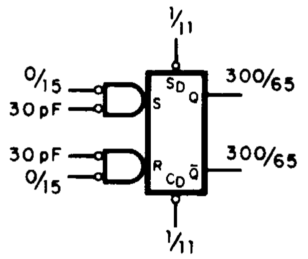
* See Note 1

CCSL COMPOSITE DATA SHEET

CCSL INPUT LOAD & DRIVE FACTORS

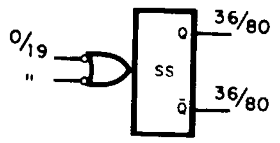
BINARY

9950

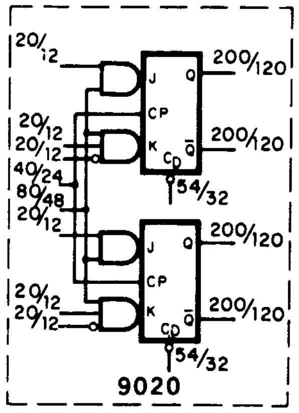


ONE-SHOTS (MONOSTABLE)

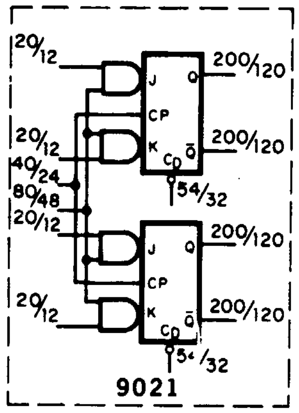
9941 & 9951



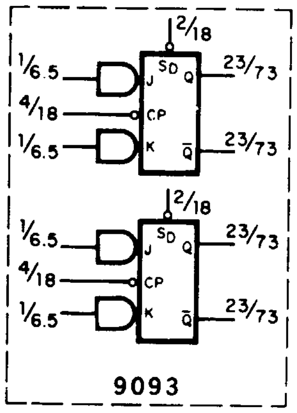
DUAL FLIP-FLOPS



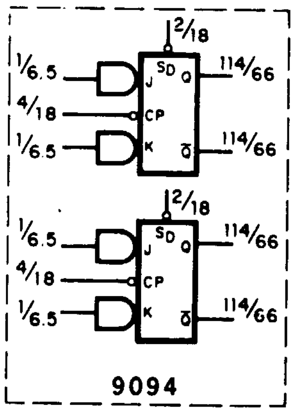
9020



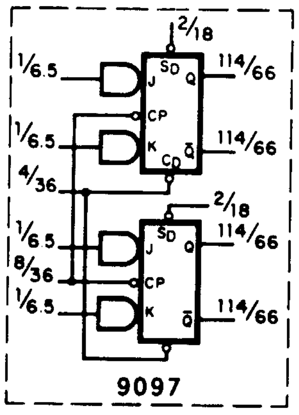
9021



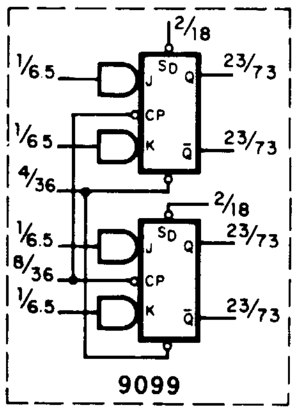
9093



9094



9097

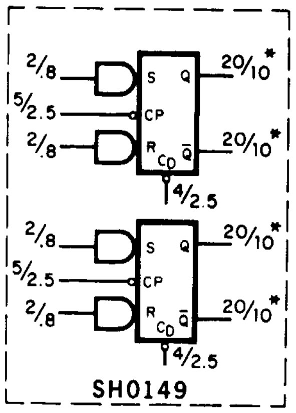


9099

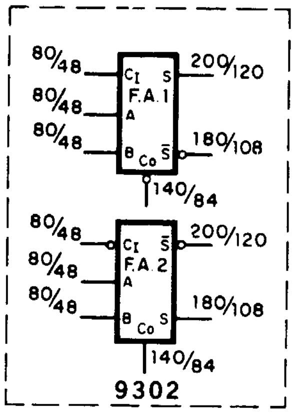
CCSL COMPOSITE DATA SHEET

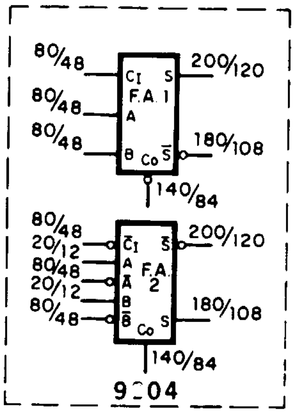
CCSL INPUT LOAD & DRIVE FACTORS

DUAL FLIP-FLOPS (continued)



DUAL FULL-ADDERS





OUTPUT LEVELS VERSUS LOADING

GATE TYPE ELEMENTS

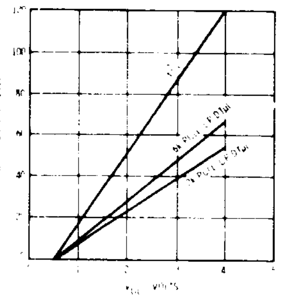


FIG. 1

BUFFER ELEMENTS

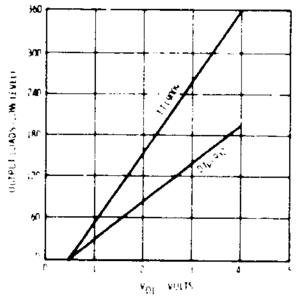


FIG. 2

6k PULL-UP

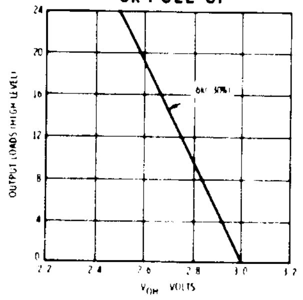


FIG. 3

2k PULL-UP

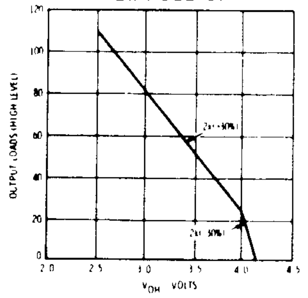


FIG. 4

* See Note 1

FAIRCHILD DIODE-TRANSISTOR MICROLOGIC®

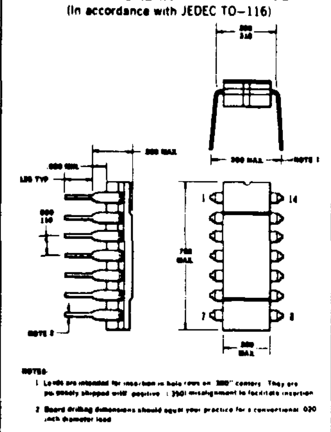
INTEGRATED CIRCUITS COMPOSITE DATA SHEET

A FAIRCHILD *COMPATIBLE CURRENT SINKING LOGIC* PRODUCT
0°C TO 75°C TEMPERATURE RANGE

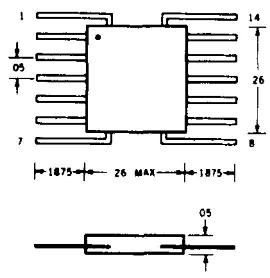
GENERAL DESCRIPTION — Fairchild Diode Transistor Micrologic® (DTμL) Integrated Circuits family uses diode-transistor logic and is designed specfically for integrated circuit technology. The design of these circuits offers distinctly superior performance. Some of the advantages follow:

- HIGH PERFORMANCE WITH A SINGLE POWER SUPPLY - - 5.0 V
- HIGH NOISE IMMUNITY - - 1.0 V
- HIGH FAN-OUT CAPABILITY - - 8-25
- GATES WITH 6 k OR 2 k PULL-UP RESISTORS FOR OPTIMUM SPEED
- FAN-OUT AND NOISE IMMUNITY TRADE-OFF
- LOW POWER DISSIPATION - - 8.5 mW/GATE
- GATE OUTPUTS CAN BE TIED TOGETHER FOR THE "WIRED OR" FUNCTION

TYPICAL DUAL IN-LINE PACKAGE
(In accordance with JEDEC TO-116)

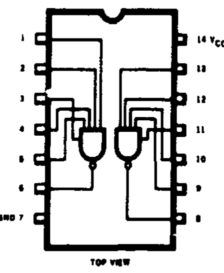
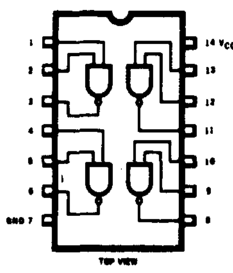
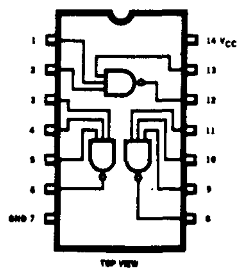
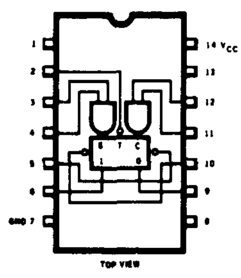


TYPICAL FLAT PACKAGE
TOP VIEW



ORDER INFORMATION — To order Diode Transistor Micrologic® Integrated Circuits elements specify U31XXXX59X for Flat package and U6AXXXX59X for Dual In-Line* package where XXXX is 9930, 9932 etc.

PIN CONFIGURATION: IDENTICAL FOR DUAL-IN-LINE AND FLAT PACKAGES

DTμL 9930 • DTμL 9961 DTμL 9932 • DTμL 9944	DTμL 9946 • DTμL 9949	DTμL 9962 • DTμL 9963	DTμL 9945 • DTμL 9948
			

*Fairchild Patent Pending

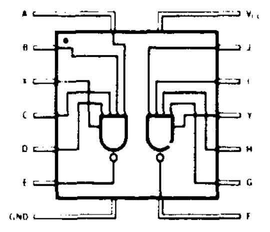
FAIRCHILD
SEMICONDUCTOR
A DIVISION OF FAIRCHILD CAMERA AND INSTRUMENT CORPORATION

313 FAIRCHILD DRIVE, MOUNTAIN VIEW, CALIFORNIA, (415) 962-5011, TWX: 910-379-6435

DTμL GATES

All DTμL gates are positive logic NAND gates or negative logic NOR gates. A variety of gate combinations is available which provides the system designer the utmost in logic flexibility and reduces package requirements to a minimum. Gate outputs may be paralleled to perform OR (collector) logic. In addition, gates may be cross-connected to form flip-flops, exclusive OR, etc. Gates with 2 kΩ pull-up resistors offer improved propagation delay times.

LOGIC DIAGRAM



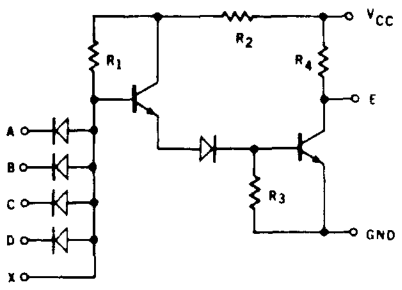
POSITIVE (NAND) LOGIC

$E = \overline{A \cdot B \cdot C \cdot D \cdot (X)}$

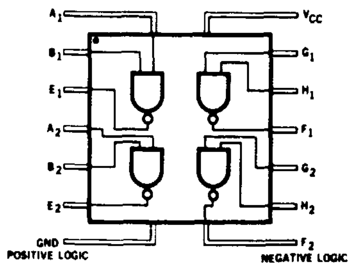
$F = \overline{G \cdot H \cdot I \cdot J \cdot (Y)}$

SCHEMATIC DIAGRAM — ONE GATE ONLY
DTμL 9930 • DTμL 9961

TYPICAL RESISTOR VALUES	
R_1	2.00k Ω
R_2	1.75k Ω
R_3	5.00k Ω
R_4	6.00k Ω (9930)
R_4	2.00k Ω (9961)



LOGIC DIAGRAM



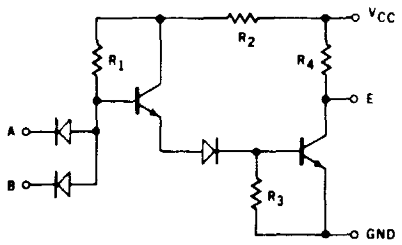
POSITIVE (NAND) LOGIC

$E = \overline{A \cdot B}$

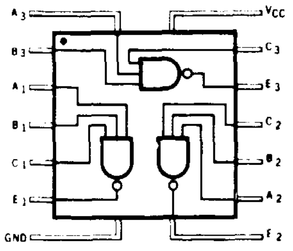
$F = \overline{G \cdot H}$

SCHEMATIC DIAGRAM — ONE GATE ONLY
DTμL 9946 • DTμL 9949

TYPICAL RESISTOR VALUES	
R_1	= 2.00k Ω
R_2	= 1.75k Ω
R_3	= 5.00k Ω
R_4	= 6.00k Ω (9946)
R_4	= 2.00k Ω (9949)



LOGIC DIAGRAM

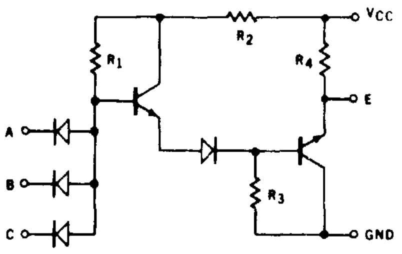


POSITIVE (NAND) LOGIC

$E = \overline{A \cdot B \cdot C}$

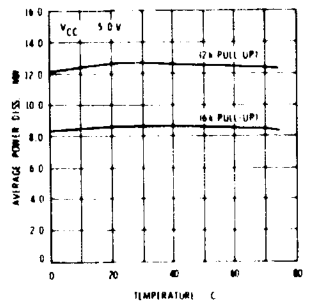
SCHEMATIC DIAGRAM — ONE GATE ONLY
DTμL 9962 • DTμL 9963

TYPICAL RESISTOR VALUES	
R_1	= 2.00k Ω
R_2	= 1.75k Ω
R_3	= 5.00k Ω
R_4	= 6.00k Ω (9962)
R_4	= 2.00k Ω (9963)

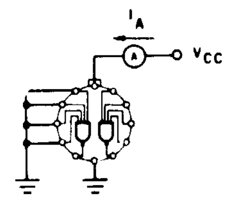


FAIRCHILD DIODE-TRANSISTOR MICROLOGIC®I.C.

AVERAGE POWER DISSIPATION VERSUS TEMPERATURE (TYPICAL EACH GATE)

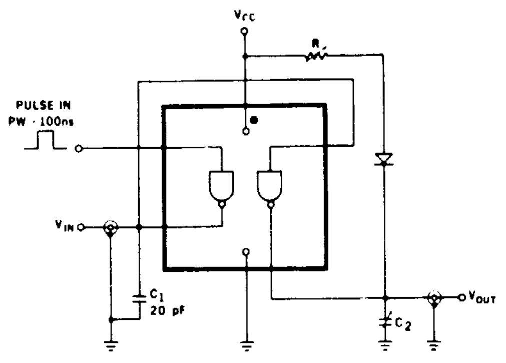


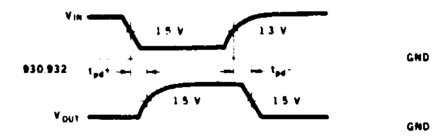
TEST CONDITIONS



AV POWER DRAIN $= \frac{V_{CC} \cdot I_A}{2}$

TYPICAL Tpd TEST CIRCUIT DTμL GATES





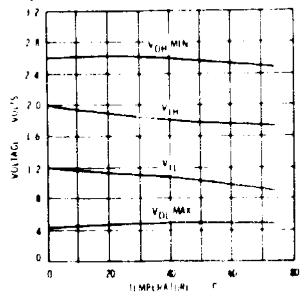
T_{pd} – will be read from input at 1.3 V

(V_{CC} = 5 V, T = 25°C)

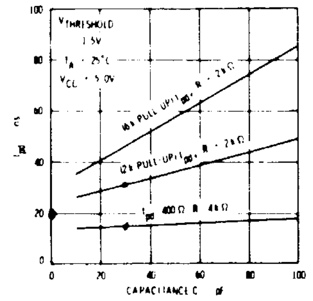
		R	C_2	Min.	Max.
(6 k Pull-up)	t_{pd+}	3.9kΩ	30 pF	25 ns	80 ns
(6 k & 2 k Pull-up)	t_{pd}	400 Ω	50 pF	10 ns	30 ns
(2 k Pull-up)	t_{pd+}	3.9kΩ	30 pF	15 ns	50 ns

OPERATING VOLTAGE CHARACTERISTICS

WORST CASE (OUTPUT LOGIC LEVEL V_{OH} AND V_{OL} / INPUT THRESHOLD LEVELS – V_{IH} AND V_{IL})



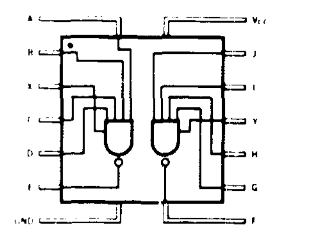
TIME DELAY VERSUS CAPACITIVE LOADS



DTμL 9932 BUFFER ELEMENT
DTμL 9944 POWER GATE

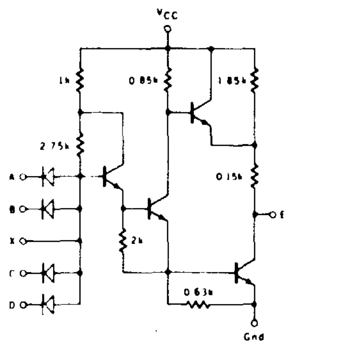
The DTμL 9932 is a dual 4-input inverting driver. It features an emitter-follower pull-up which provides a high fan-out device with superior capacitance-driving capability. The DTμL 9944 has an output with no internal pull-up. This provides a high fan-out device whose outputs may be tied together to perform the "wired OR" function. The 9944 is useful as an interface driver or as a low-power lamp driver. The fan-in of either element may be extended with the use of the DTμL 9933.

LOGIC DIAGRAM

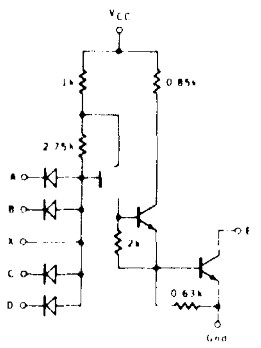


POSITIVE (NAND) LOGIC
E = $\overline{A \cdot B \cdot C \cdot D \cdot (X)}$
F = $\overline{G \cdot H \cdot I \cdot J \cdot (Y)}$

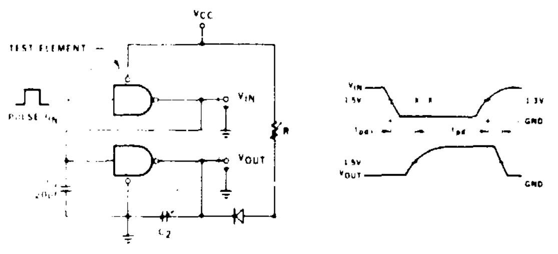
SCHEMATIC DIAGRAM OF THE DTμL 9932 ELEMENT (ONE SIDE ONLY)



SCHEMATIC DIAGRAM OF THE DTμL 9944 ELEMENT (ONE SIDE ONLY)



tpd TEST CIRCUIT FOR DTμL 9932 ELEMENT

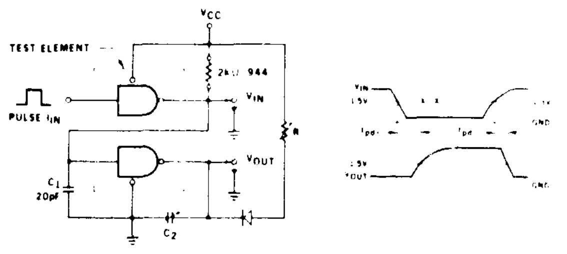


A Diodes are FD600 or Equivalent at 25 C
C1 and C2 includes Probe and Jig Capacitance

(V_{CC} = 5.0 V, T_A = 25°C)

		R	C	Min.	Max.
t_{pd+}	9932	510 Ω	500 pF	25 ns	80 ns
t_{pd-}	9932	150 Ω	500 pF	15 ns	40 ns

tpd TEST CIRCUIT FOR DTμL 9944 ELEMENT

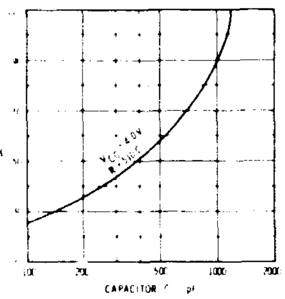


C_1 and C_2 includes Probe and Jig Capacitance

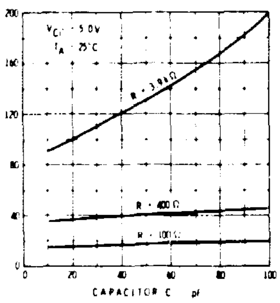
(V_{CC} = 5.0 V, T_A = 25°C)

		R	C	Min.	Max.
t_{pd+}	9944	510 Ω	20 pF	15 ns	50 ns
t_{pd-}	9944	150 Ω	100 pF	10 ns	35 ns

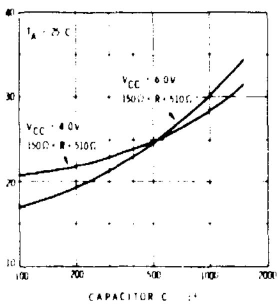
TYPICAL tpd + VERSUS CAPACITY (9932)



TYPICAL tpd + VERSUS CAPACITY (9944)



TYPICAL tpd − VERSUS CAPACITY (9932, 9944)



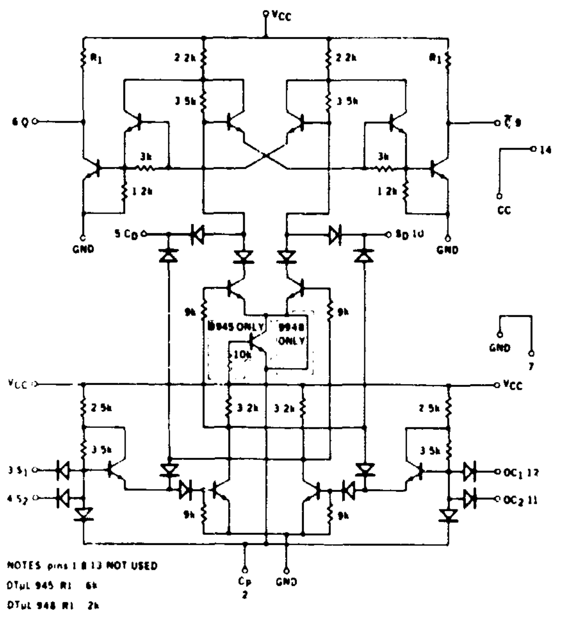
DTμL 9945 • DTμL 9948 - CLOCKED FLIP FLOP

The DTμL 9945 and DTμL 9948 Clocked Flip-Flops are directly-coupled units operating on the "master-slave" principle. Information enters the "master" while the Trigger input voltage is high and transfers to the "slave" when the Trigger input voltage goes low. Since operation depends only on voltage levels, any sort of waveshape having the proper voltage levels may be used as a trigger signal. Rise and fall times are irrelevant.

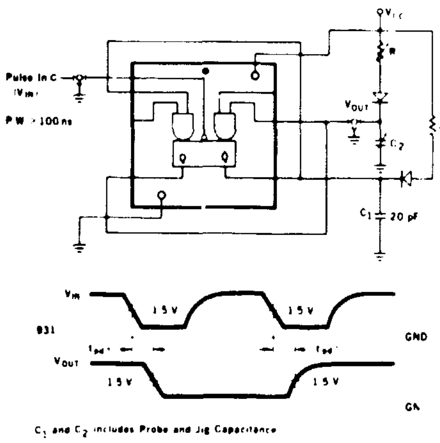
The DTμL 9945 and DTμL 9948 have an improved direct Set and Clear design which allows unhampered asynchronous entry irrespective of signals applied to any other inputs. The direct inputs always take precedence, thus simplifying the design of arbitrarily preset ripple counters and other minimum hardware applications.

Output buffers provide isolation between the "slave" and the output load, thereby enhancing immunity to signal line noise. The DTμL 9945 incorporates the standard 6 kΩ output pull-up resistor, while the DTμL 9948 features a 2 kΩ output pull-up resistor for improved rise times, and matched delay between rising and falling outputs for capacitive loading up to 100 pF.

SCHEMATIC DIAGRAM



tpd TEST CIRCUIT

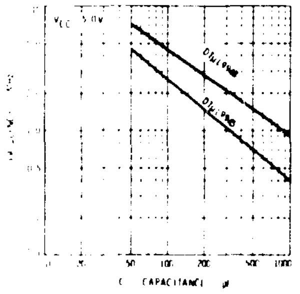


C1 and C2 includes Probe and Jig Capacitance

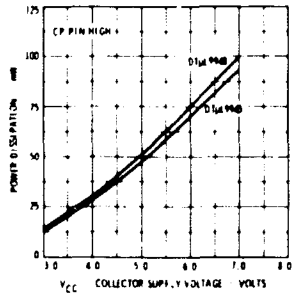
(Vcc = 5 V, T = 25°C)

		R	C_2	Min.	Max.
t_{pd+}	9945	2.00 k	30 pF	35 ns	75 ns
t_{pd-}	9945	330 Ω	30 pF	35 ns	75 ns
t_{pd+}	9948	2.00 k	30 pF	20 ns	65 ns
t_{pd-}	9948	330 Ω	30 pF	30 ns	75 ns

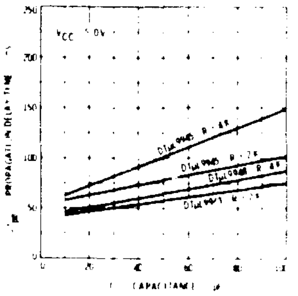
TYPICAL MAXIMUM BINARY COUNTING RATE VERSUS CAPACITY



TYPICAL POWER DISSIPATION VERSUS Vcc



TYPICAL tpd VERSUS CAPACITANCE

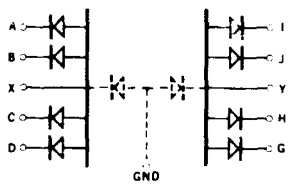


DTμL 9933 EXTENDER

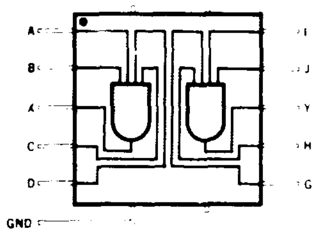
The DTμL 9933 is a Dual Input Extender consisting of two independent diode arrays identical in every respect to the input diodes of the DTμL Gate and Buffer elements. Good practice dictates that extension interconnection paths be as short as possible to minimize the effects of distributed capacitance on circuit performance.

Typical input capacitance of DTμL 9933 is 2 pF, output capacitance is 5 pF.

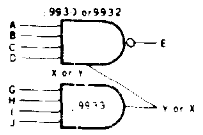
SCHEMATIC DIAGRAM



FLAT PACKAGE LAYOUT

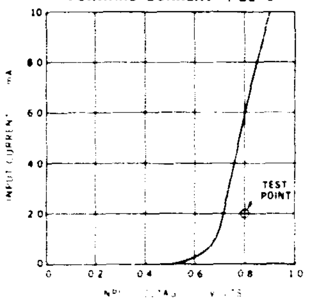


LOGIC EXAMPLE



POSITIVE LOGIC E ABCDGHIJ
NEGATIVE LOGIC E A + B + C + D + G + H + I + J

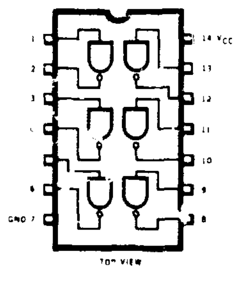
FORWARD VOLTAGE VERSUS FORWARD CURRENT +25°C



DTμL 9936 • DTμL 9937 - HEX INVERTER

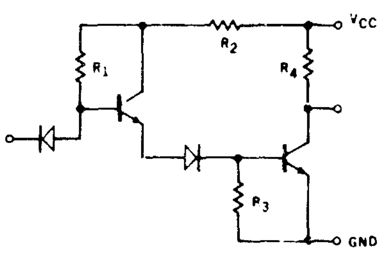
The DTμL 9936 hex inverter has input-output characteristics identical to the other DTμL gates. Inverters can be cross-connected to form flip-flops or the outputs can be paralleled to perform the "wired OR" function.

DTμL 9936 • DTμL 9937



SCHEMATIC DIAGRAM — ONE INVERTER ONLY
DTμL 9936 • DTμL 9937

TYPICAL RESISTOR VALUES	R_1 = 2.00k Ω
	R_2 = 1.75k Ω
	R_3 = 5.00k Ω
	R_4 = 6.00k Ω (9936)
	R_4 = 2.00k Ω (9937)



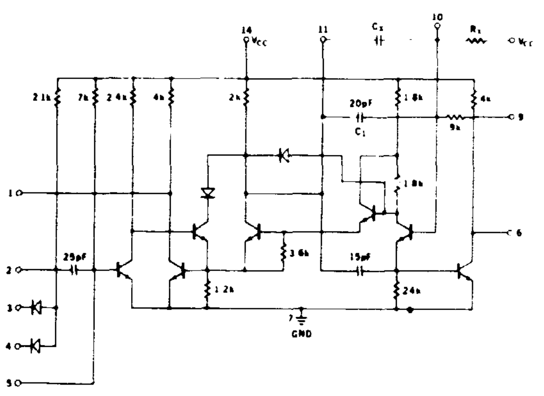
DTμL 9951 - MONOSTABLE MULTIVIBRATOR

The DTμL 9951 is an integrated monostable multivibrator designed for use with other members of the DTμL family. It provides complementary output pulses which are typically 100 ns wide. This pulse width is adjustable by the addition of external components.

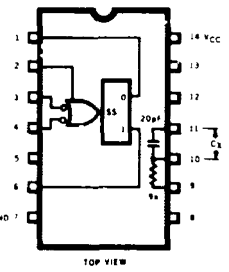
ABSOLUTE MAXIMUM RATINGS
(above which useful life may be impaired)

Supply Voltage (V_{CC}), −55°C to +125°C, continuous:	+8.0 Volts
Supply Voltage (V_{CC}), pulsed, <1 second:	+12 Volts
Output Current, into outputs:	50 mA
Current into Pin 10	5.0 mA
Input Forward Current	−10 mA
Input Reverse Current	1.0 mA

SCHEMATIC DIAGRAM



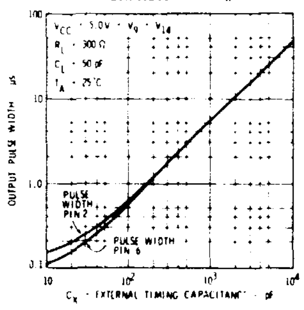
DTμL 9951



INPUT-OUTPUT LOAD FACTORS TO DTμL FAMILY

Each DTμL 9951 input should be rated at 2 loads.
Each DTμL 9951 output may drive 10 DTμL loads.

OUTPUT PULSE WIDTH VERSUS EXTERNAL TIMING CAPACITANCE C_x



RULES FOR USE OF DTμL 9951

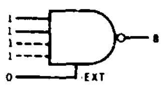
1. With Pin 9 connected to V_{CC} and no external capacitor (C_x), the output pulse width is approximately 100 ns.
2. With Pin 9 connected to V_{CC} and an external capacitor (C_x) connected between Pins 10 and 11, the output pulse width (T) is:
   $T \approx 4.5\ (C_x + 20)$ with C_x in pF and T in ns
3. For improved pulse width control, Pin 9 is left open and a stable external resistor (R_x) of 9 kΩ minimum to 15 kΩ maximum is connected from Pin 10 to V_{CC}. The output pulse width is given by the expression: $T \approx 0.5\ R_x\ (C_x + 20)$ with R_x in kΩ, C_x in pF and T in ns.
4. The output duty cycle (pulse width/period) should not exceed 40%. It may be increased to 50% by adding a 2 kΩ resistor between Pin 11 and V_{CC}. Higher duty cycles are obtainable but the output pulse width and performance are less predictable.
5. The maximum input fall time to trigger: 25 ns for a 1.0-volt swing; 50 ns for a 2.0 volt swing; 100 ns for a 4.0-volt swing.
6. The AC sensitivity of the inputs may be decreased by connecting a capacitor between Pin 5 and ground.
7. The minimum pulse width at output Pin 1 is approximately 100 ns. This pulse width may be decreased to 50 ns by connecting a 10 kΩ resistor between Pin 5 and V_{CC}.

FAIRCHILD DIODE-TRANSISTOR MICROLOGIC®I.C.

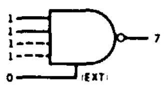
ABSOLUTE MAXIMUM RATINGS (above which useful life may be impaired)

Supply Voltage (V_{CC}), −55°C to 125°C, continuous	+8.0 Volts	Input Forward Current	−10 mA
Supply Voltage (V_{CC}), pulsed, <1 second	+12 Volts	Input Reverse Current	1.0 mA
Output Current, into outputs		Operating Temperature	0°C to +75°C
DTμL 9932 & 9944	100 mA	Storage Temperature	−65°C to +150°C
DTμL, except 9932 & 9944	30 mA		

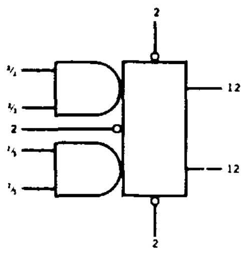
INPUT-OUTPUT LOADING FACTORS



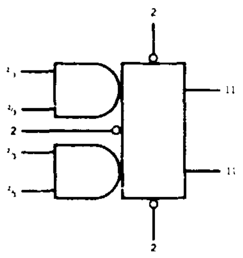
1/2 DTμL 9930,
1/4 9946,
1/3 9962,
1/6 9936



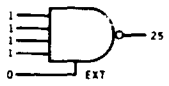
1/2 DTμL 9961,
1/4 9949,
1/3 9963,
1/6 9937



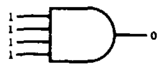
DTμL 9945



DTμL 9948



1/2 DTμL 9932
1/2 DTμL 9944

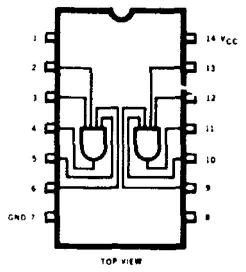


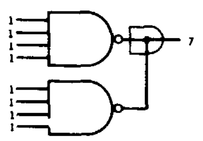
1/2 DTμL 9933

The number of elements that may be driven by an output terminal may consist of any combination of elements whose summation of input loading does not exceed the output terminal driving capability.

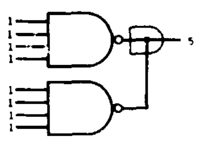
RULES FOR INPUT EXPANSION AND "WIRED OR" CONNECTION

DTμL 9933 EXPANDER

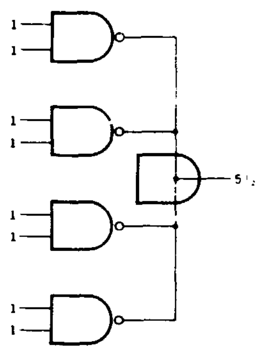




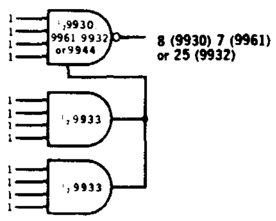
DTμL 9930 "wired OR"
ALSO 9946, 9962



DTμL 9961 "wired OR"
ALSO 9949, 9963



4 x 1/4 DTμL 9946



EXTENSION OF INPUTS

RULES

1. Outputs of DTμL gates with 6 kΩ pull-up resistors, 9930, 9946, and 9962 may be tied together for the "wired OR" function. Subtract 1 unit fan-out for each added gate. Subtract 5 fan-outs for 6 added gates.
2. Outputs of DTμL gates with 2 kΩ pull-up resistors, 9949, 9961, and 9963 may be tied together for the "wired OR" function. Subtract 2 units of fan-out for each added gate.
3. Outputs of DTμL 9932 may not be tied together for the "wired OR" function.

REFERENCES

1. Speed Power Chart for Digital IC's. The Electronics Engineers, March 1967, p. 148.

2. Cricchi, J. R.; Lancaster, E.; and Strull, G.: A Large Scale Complementary MOS Memory. Supplement to IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-3, No. 6, November 1967.