C71-1065/401

C71-1065/401

VOTER-COMPARATOR-SWITCH OPTIMIZATION STUDY

FINAL REPORT

North American Rockwell
Electronics Group

NAS 9-11835

C71-1065/401

VOTER-COMPARATOR-SWITCH OPTIMIZATION STUDY

FINAL REPORT

3 January 1971

J. Jurison
Program Manager

Prepared for

National Aeronautics and Space Administration
Manned Spacecraft Center

North American Rockwell
Electronics Group

3370 Miraloma Avenue, Anaheim, California 92803

FOREWORD

This Final Report covers the work performed by the Autonetics Avionics and Sensors Division of North American Rockwell Corporation under a study contract entitled Voter-Comparator-Switch Optimization Study. This report is submitted to the National Aeronautics and Space Administration's Manned Spacecraft Center under the requirements of Contract NAS9-11835. The study covered the period from 4 May 1971 through 3 January 1972. Autonetics' Program Manager was J. Jurison. Key participants in this study were D. B. Brosius, R. K. Kibby, L. J. Koczela and J. Swetz. The NASA Technical Monitor was P. E. Sollock.

ABSTRACT

The report covers the design optimization and simulation effort of the Voter-Comparator Switch (VCS) element. The VCS is an adaptive voter element that performs failure detection and reconfiguration function in a computer system designed to tolerate any three single failures in a fail-operational-fail-operational-fail-safe manner. Detailed logic equations were developed and the design was evaluated and refined using a logic level fault simulator.

CONTENTS

ILLUSTRATIONS

TABLES

SECTION 1

SUMMARY

1.1 OBJECTIVES

Under the Reconfigurable G&C Computer Study Contract (NAS9-10416) Autonetics defined a redundant computer and data bus system which satisfied the requirement to operate through three failures in a fail-operational-fail-operational-fail-safe manner (Ref. 1). During the course of the study it became apparent that some type of adaptive voter element was necessary to accomplish certain types of failure detection and reconfiguration functions which were beyond the capabilities of autonomous, redundant computers. As the requirements for this voter were defined, a unique element, the VCS (Voter-comparator-switch), evolved. The purpose of the VCS Optimization Study was to pursue the design of the VCS device to a stage where hardware, usable with off-the-shelf computers, could be developed. The effort consisted of the logical design and computer simulation to perform detailed evaluation of VCS operation under different operating conditions.

1.2 PROGRAM OVERVIEW

The study covered the time interval from May 4, 1971 through January 3, 1972. It was divided into the following major tasks (Ref. 2):

1. VCS design optimization
2. Simulation program modification
3. Evaluation program definition
4. Evaluation program performance
5. Detailed hardware design

The VCS design optimization task consisted of modification of the basic VCS element that was described in the final report of the Reconfigurable G&C Computer Study (Ref. 1). The major modifications were (1) making the VCS physically separate from the computer and (2) defining an interface that would permit ease of mating with any general purpose aerospace computer. A parallel by bit and serial by byte interface using eight bit bytes and odd parity as a ninth bit was selected. The functional step by step operation of the VCS was defined in flow chart form and the boolean equation describing the logic necessary to implement these operations were written. The logic equations and flow charts are included in Appendix A and B of this report, repectively.

The objective of the simulation program modification task was to extend the computer system simulator developed during the Reconfigurable G&C Computer Study Contract to enable logic level simulation of the VCS device. Provision for component level fault simulation was included to allow simulation of logic gate failures. The simulator program was written in the Fortran IV language

and was designed for execution on IBM S360/85 and the XDS Sigma 5. Initial development and debugging took place on the IBM S360/85 at Autonetics. The final degugging and the design evaluation was performed on the XDS Sigma 5 at NASA MSC. Section 3.1 describes the operation and the capabilities of the simulation program.

The objective of the evaluation program definition was to develop and document a plan for conducting an evaluation of the VCS design with particular concern for the identification and elimination of single point failures. It consisted of two primary efforts. The first effort was to determine the overall approach to be used in the evaluation process. Essentially this required determining answers to two basic questions: (1) how is the VCS logic to be exercised while the logic faults are simulated? and (2) how is VCS operation to be monitored and assessed during the fault simulation? The second effort under this task was to document the plan and procedures required to perform the evaluation. This plan consisted of a description of the overall evaluation plan and the detailed procedures, in terms of card deck setup, computer operating procedures, etc for conducting the simulations. The plan was submitted to NASA MSC in the Monthly Progress Report No. 5 (Ref. 3).

The evaluation program performance task covered the actual simulation activity in accordance with the Evaluation Program Plan. The simulation runs were mostly made of the XDS Sigma 5 computer at NASA MSC. After initial debugging phase where Autonetics provided on site support at MSC, the data were mailed to Autonetics for analysis. Section 3.2 of this report contains a description of the evaluation program and provides a summary of the evaluation results, showing the number of cases run, number of faults detected, significant statistics and conclusions.

During the detailed hardware design phase the original logic equations and flow charts were updated and modified based upon the results of the evaluation program. Up to this point the design had been carried out to the functional and logic level and kept purposely independent of any specific semiconductor technology. The objective of this phase was to consider hardware design aspects of the VCS and prepare a model specification which could be used for preparation of an RFP for fabrications with state-of-the-art integrated circuit technology. MOS technology was selected by NASA as the most desirable technology for VCS implementation.

Section 2.4 describes the recommended mechanization and provides a discussion of the key design considerations. The model specification for the VCS element appears in Appendix C.

1.3 RESULTS, CONCLUSIONS, AND RECOMMENDATIONS

The VCS design defined during the Reconfigurable G&C Computer Study has been modified to operate as an add-on device to a set of redundant general purpose aerospace digital computers to achieve fault tolerance to three failures in a fail-operational-fail-operational-fail-safe manner. A complete set of logical equations and functional flow charts have been prepared and the complete logic has been simulated to (1) answer that the logic will perform as specified under no-fault

conditions and (2) that internal component failures in the VCS will not cause errors to propagate through the computers, data buses or other systems. The results have provided a high degree of confidence that these objectives have been met. The design has been carried to a point where hardware fabrication with MOS technology can be initiated. Such a system optimized for minimum number of components in the logic section will require a total of eight MOS/LSI devices of six types. It is recommended that Standard TTL bipolar circuits be used for mechanizing interface circuitry and clock generation. All components can be mounted on a plug in module with a minimum area of 40 square inches for component mounting.

An alternate design approach which can potentially reduce the VCS development cost appears to be microprogrammed control, using semiconductor read-only memories (ROMS). If the minimization of the development cost is the prime factor to be considered, then it is recommended that the microprogrammed design approach be investigated and traded off against fixed logic design optimized for minimum component count.

SECTION 2

VCS DESCRIPTION

2.1 OVERALL SYSTEM CONCEPT

The VCS is designed for application in redundant computer systems. It evolved out of the triple failure tolerant computer system designed during the Reconfigurable G&C Computer Study. This section presents the basic design concept of that computer system. However, it should be noted that the VCS device can be used in other fault tolerant systems to perform comparison and/or voting on redundant digital data from multiple sources and to prevent error propagation through the system. Section 2.3 describes the use of the VCS from a general system application standpoint.

The system uses quad redundant computers and VCS elements providing the ability to withstand failure of any three components in a fail-operational-fail-operational-fail-safe (FOOS) manner. The first two failures are tolerated in a fail operational manner, meaning that 100% error detection capability and 100% reconfiguration effectiveness are provided with no degradation in performance. The third failure is also detected in 100% cases; however, the fail safe requirement permits slower reconfiguration that can be met by a combination of self test software and hardware within the individual computers.

The computer system consists of four computers and four I/O busses that interface with local processors of external subsystems. These computers are interconnected to achieve a FOOS computer system as shown in Figure 2-1. Each computer contains a connection to one of the four external I/O busses in the system. In addition, each computer contains four interconnections: three receive channels, one from each of the other computers, and one output channel to all the other computers; it is via these connections that the voter-comparator-switch (VCS) concept is mechanized to implement adaptive voting.

The VCS concept enables the computer system to be operated in a variety of modes: four-way voting (all four computers doing the same job, with one or more of the VCSs voting on the I/O information), three-way voting with the other computer dormant or doing a different job, two-way comparison with the other two computers also in comparison or doing distinct jobs, and four non-redundant computers. The mode is under the control of the software executive system which is distributed among all the computers. The executive is redundant, in a distributed sense, to satisfy the FOOS requirement. The level of redundancy of the computations nay be changed under software control by changing the mode of the system.

The VCS is designed such that the condition of its operating state and the results of voting/comparison processes may be interrogated by the executive program.

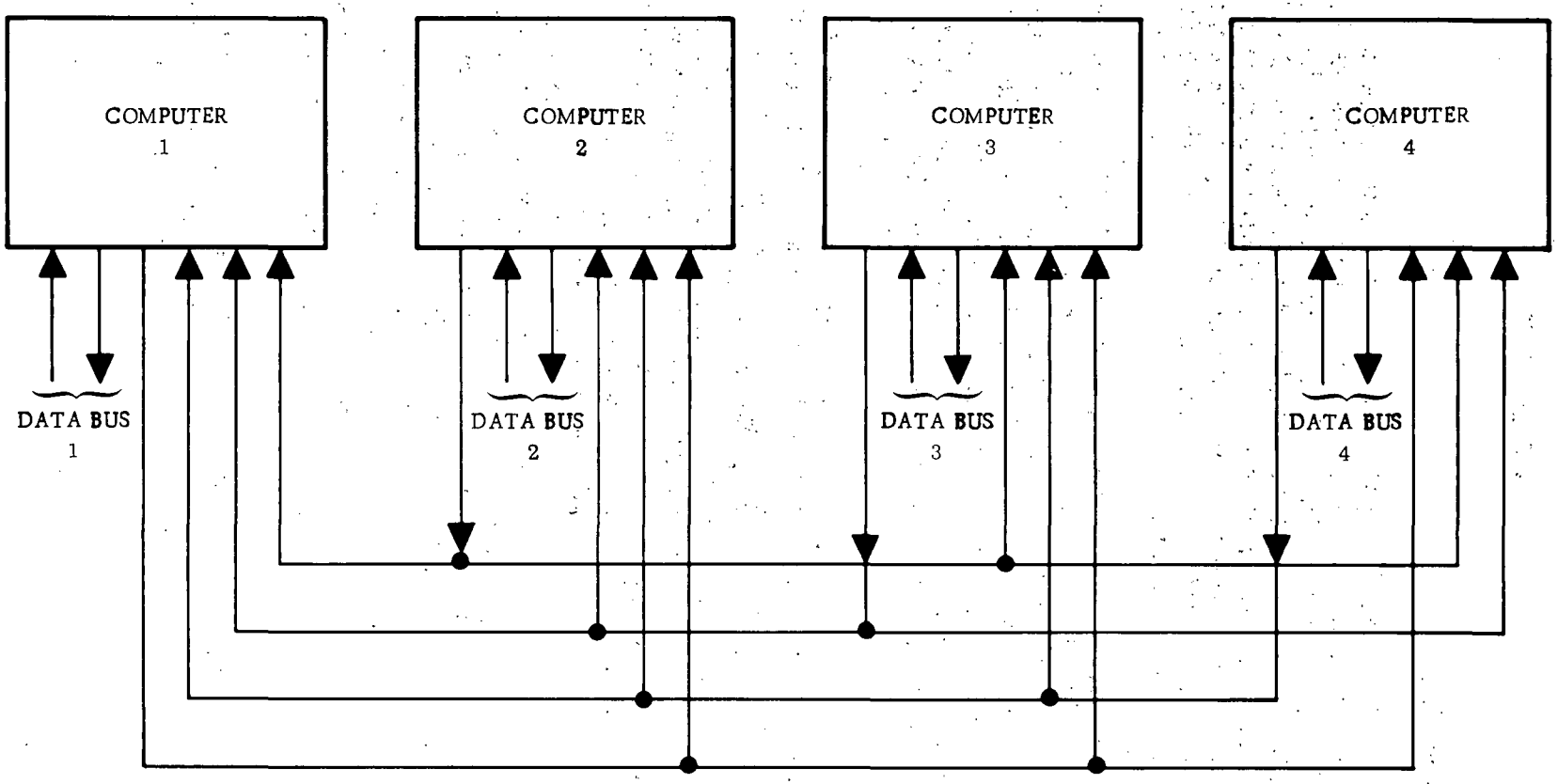


Figure 2-1. Computer System Interconnection Diagram

The four computers each operate on their own independent clock. Redundant data operated on by the VCS is required to be in synchronization within a specified tolerance; this data is not required to be in bit sync.

The VCS device is capable of operating on redundant data in a majority voting or a comparison mode, thereby performing a redundancy reduction of either 4:1, 3:1, or 2:1, or it may operate on non-redundant data. The device is adaptive in that it may be switched into different modes. The device is also adaptive to failures in the computer system by means of adaptive majority logic.

The diagram indicating the basic architecture of the VCS device and the interface between the VCS and computer I/O is shown in Figure 2-2. The VCS device communicates on the I/O bus to the external subsystems; it has as inputs, the outputs of the I/O sections of the four computers. As shown, these inputs to the VCS may be used by the voting, comparison, or selector logic. This logic is directed by the matrix section of the VCS, which operates on the principle of adaptive majority logic. Its function is to control the mode of the voter-comparator-selector logic block and to decide which computers are failed or non-failed. These functions are implemented by means of a P matrix and a R matrix.

The P matrix (4 x 4) contains each computer's opinion of the failure status of the other computers and the majority decision of the failure status of each computer. Essentially this matrix contains the failure status of the computer system. The failure status decision on an individual computer is derived from the other computers' failure opinions and the go/no-go self test results from the built-in test equipment in each computer. The decisions are arrived at on the basis of adaptive majority logic.

The R matrix represents the desired mode of operation. It operates under a majority decision rule as to the selection of the mode. Further, it is adaptive in that the P matrix is used to determine which computers are failed and should be ignored in the R matrix. The R matrix is also a 4 x 4 matrix and is set directly by the computers. The R matrix logic directs the connections to the voter-comparator-selector logic to implement the mode of operation.

The VCS device constitutes the primary means of failure detection in the computer system since an erroneous input to the voter may be readily detected. In addition, each computer contains self test capability that provides autonomous failure detection to a specified confidence level; this capability is used to aid the computer system failure reconfiguration operations.

A description of the I/O process for outputting data to the external subsystems when operating in a three-way voting mode is shown in Figure 2-3. Computers 1, 2, and 3 are shown as operating in a voting mode, computer 4 could be performing another task or in a failed state. The VCS receives three copies of the message and transmits the majority on the bus. Simultaneously, the IOPs of computers 1, 2, and 3 monitor the transmitted message by the "loop around" design of the bus. This monitoring process consists of comparing the information sent to the VCS with that transmitted on the bus, any discrepancies are noted, and "go" - "no-go" code is sent to the VCS by each IOP at the end of the monitoring process. Once again the majority "go/no-go" opinion would be transmitted on the

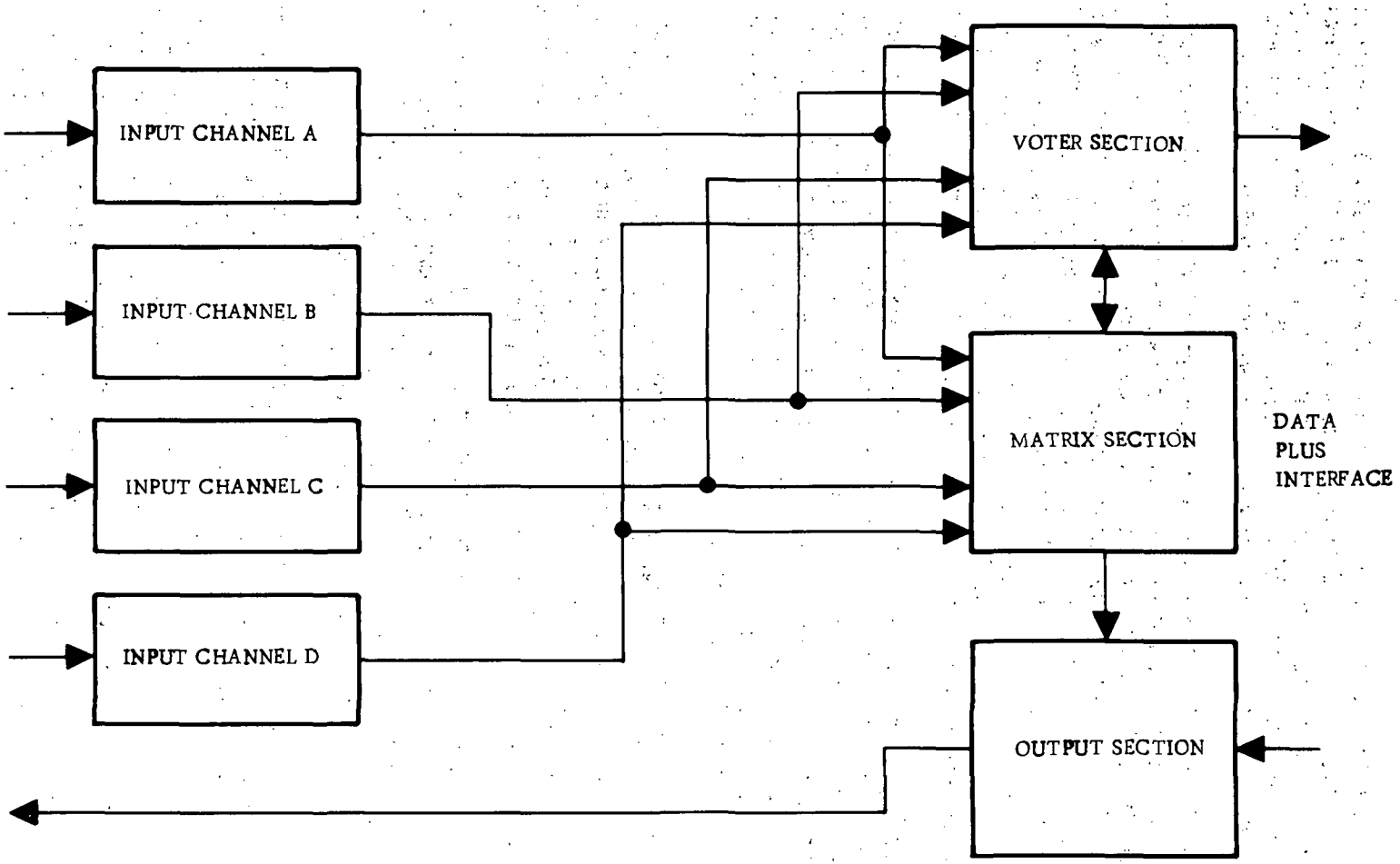


Figure 2-2. VCS Functional Block Diagram

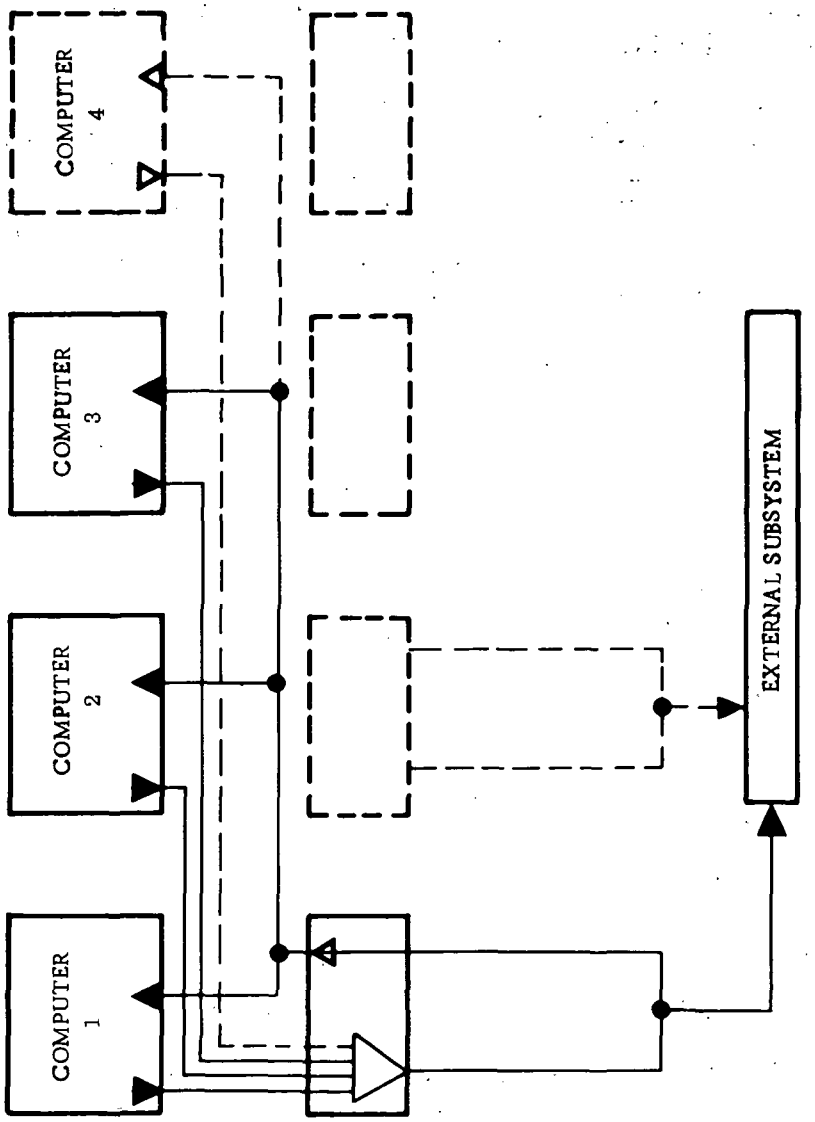


Figure 2-3. Data Output Process

bus. The IOPs also monitor the go/no-go code transmitted on the bus to determine if they are in agreement with the majority. The IOPs contain automatic message re-transmission capabilities in case of a majority no-go opinion. Based on the monitored go/no-go code, the IOPs either proceed to the next message or re-transmit the last message. This re-transmission is attempted twice with the same VCS, if three successive transmissions fail, the IOPs then switch over to a new VCS. Three successive transmission failures with the new VCS will result in the IOPs proceeding on to the next message in the IOP sequence. The IOP also stores status words at the end of each message which indicate the occurrence of any errors or retries in the transmission process.

The process of inputting data from external subsystems for the same mode of operation, namely, three-way voting is quite similar. The command to input data to the computer systems is sent to the external subsystem over one bus. However, the data is sent to the computer system over multiple busses. Further, the data received over a bus is sent to all the IOPs; therefore, each IOP will receive three copies of the data sent to the computer system. This redundant set of input data received by each IOP is used in a software voting process by the CPU.

2.2 FUNCTIONAL DESCRIPTION OF THE VCS

2.2.1 VCS Organization

The VCS provides a means of preventing bad data from being sent to subsystems from a multiple computer system. In addition, the VCS provides data to enhance the fault detection and isolation capability of the system.

The VCS is divided into five functional areas:

1. Input channels A through D
2. Voter Section
3. Matrix section
4. Output channel
5. Clock section

The functional organization of the VCS is shown in Figure 2-4.

Each of the functional areas will be described in detail in the following paragraphs. The clock section of the VCS will be described in detail in section 2.4 since the mechanization used for the VCS defines the type of clock section required.

2.2.2 Input Channels

2.2.2.1 General

The VCS contains four identical input channels that provide the interface with the computer system for data inputs to the VCS. All four input channels

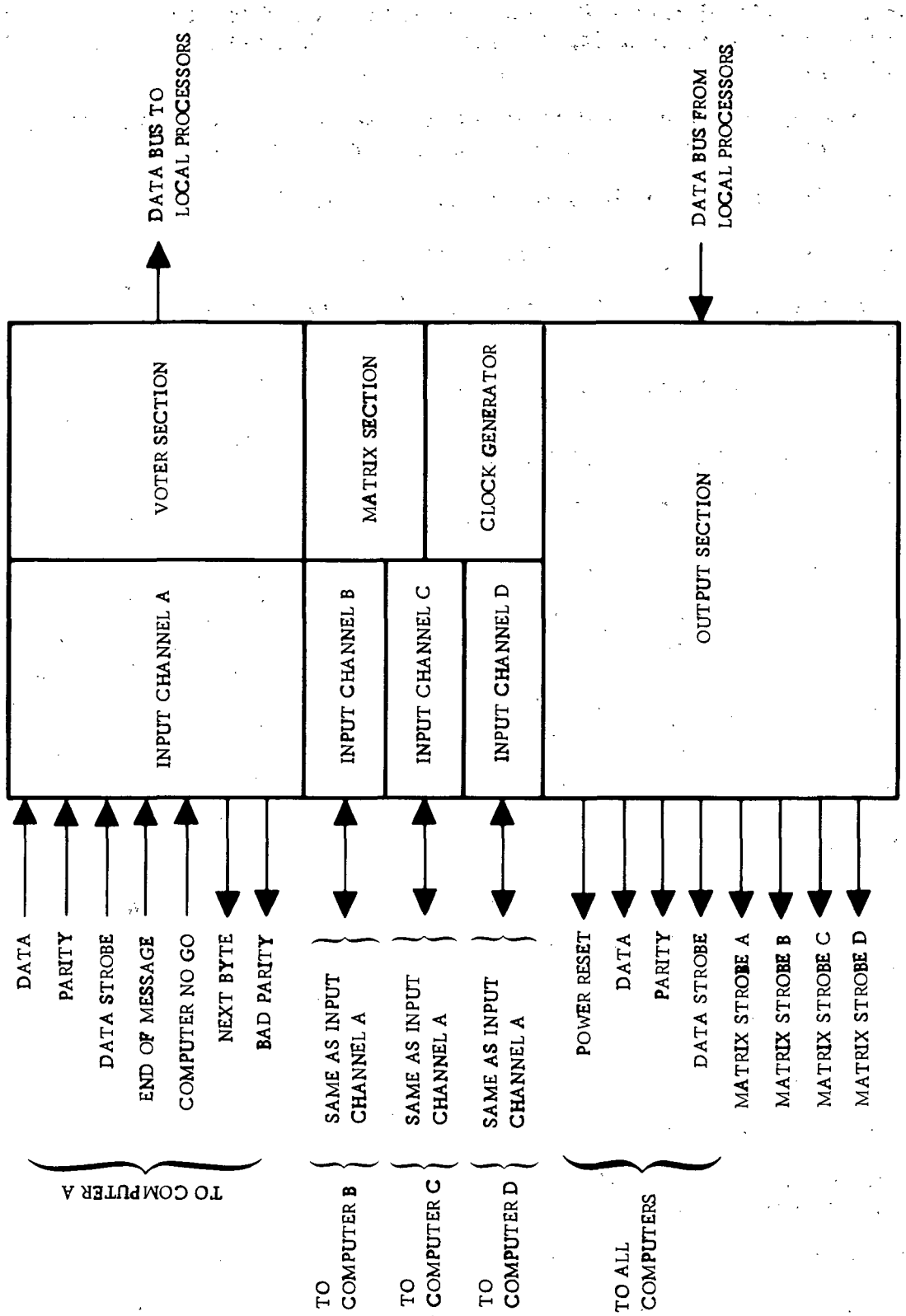


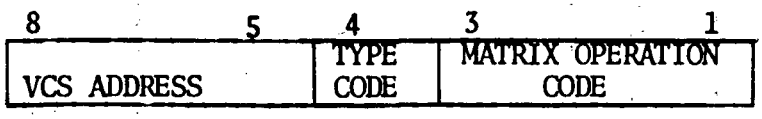
Figure 2-4. VCS Functional Diagram

operate independently of each other and are turned on and off by commands from the computer connected to that input channel. A block diagram of an input channel is shown in Figure 2-5.

Input channel operation is determined by control bytes received from the computer. There are three types of operations that are possible:

1. Transfer of data to the voter section of the VCS.
2. Transfer of data to the matrix section of the VCS.
3. Transfer of data requests to the matrix section of the VCS.

The control byte is made up of the following fields:



The VCS address field consists of four bits where each bit represents one of the four possible VCS units in the system. A binary one in the proper bit position will cause a particular VCS to operate according to the control byte. The use of individual bits for each VCS permits a computer to address up to four VCS units simultaneously. The type code field specifies whether the operation is to involve the voter or matrix section of the computer. A binary zero indicates the voter section and a binary one indicates the matrix section. The matrix operation code field is valid only if the type code field contains a binary one. The operation defined by the matrix operation code field are:

Operation	Bit Position 3	2	1
1. Set S matrix to zero and set R and P matrices to value	0	0	0
2. Set R and P matrix to value	0	0	1
3. Set S matrix to zero	0	1	1
4. Sample all matrices	1	0	0
5. Sample P matrix diagonal and the operating mode register	1	1	1
6. Sample S matrix	1	1	1

A detailed description of these operations will be given in Section 2.2.4.

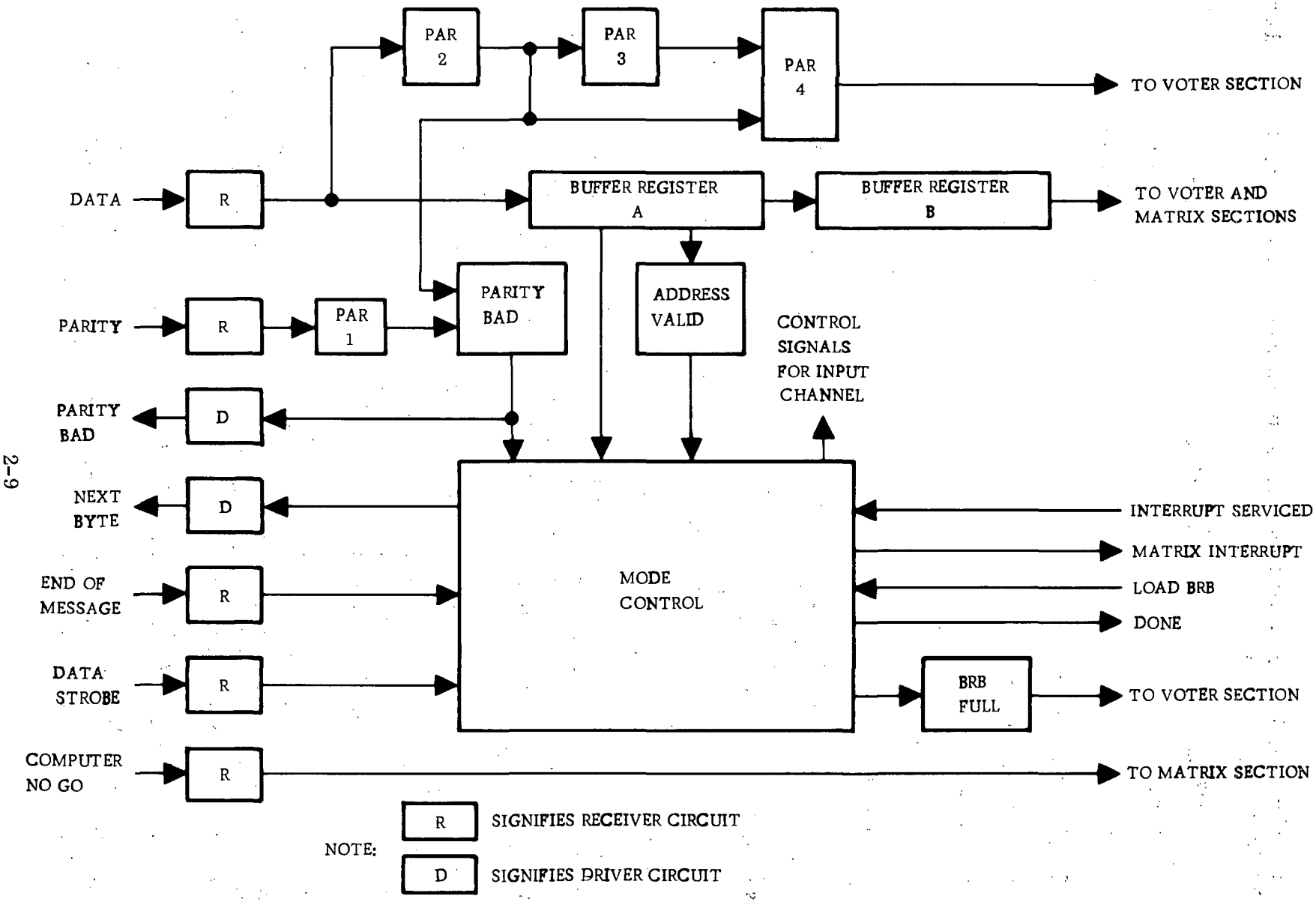


Figure 2-5. Input Channel Block Diagram

The physical interface between the input channel and the computer consists of the following signals:

Signal	Quantity	Source
1. Data	8	Computer
2. Parity	1	Computer
3. Computer No Go	1	Computer
4. End of Message	1	Computer
5. Data Strobe	1	Computer
6. Next Byte	1	VCS
7. Bad Parity	1	VCS

The signals from the computer are bussed to all VCS units in the system while the signals from the VCS are not. Thus, the computer sees six more signals in the interface than the VCS units do. Data are transferred in bit parallel, byte serial fashion with each byte having eight bits and an odd parity bit. A message contains a control byte and zero or more data bytes. The VCS imposes no limit on the number of data bytes contained in a message.

2.2.2.2 Control Byte Transfer Operation

All operations of the input channel begin when the data strobe line from the computer is set true with the input channel in the idle mode. The data lines are read into a buffer register A (BRA) and the parity line read into the parity buffer flip flop (PAR1). Odd parity for the received byte is generated and stored in flip flop PAR2.

If PAR1 and PAR2 are not in the same state, this indicates bad parity on the received byte. The input channel sets the parity bad discrete line to the computer true for the two microseconds and returns to the idle mode. The decision to retransmit is made by the computer. If PAR1 and PAR2 are in the same state, the parity is correct for the byte and operation continues.

The VCS address field of the control byte is checked to see if a binary one is in the correct bit position. A binary zero causes the input channel to return to the idle mode. A binary one causes the address valid flip flop to be set and the operation proceeds to the testing of the type code field to determine whether a voter or matrix section operation is to be performed.

2.2.2.3 Data Transfer to Voter Section Operation

This operation is performed if the type code field holds a binary zero. The next byte line to the computer is set true and the DONE flip flop is zero set indicating to the voter section that an operation is to begin.

When the data strobe line from the computer goes true, the byte is read in and parity checked as for the control byte. If the parity is not correct, the parity bad line is set true and the input channel waits for the data strobe line to go true indicating the same byte is being retransmitted by the computer. This transmission is repeated until either the parity is received correctly or the end of message line from the computer goes true. The end of message line being true causes the input channel to terminate the operation by going to the idle mode.

Finding the parity to be good, the input channel transfers the contents of BRA into buffer register B (BRB) and sets the buffer register B full indicator (BRBF) true. The next byte line is also set true and the input channel waits for the data strobe line to go true.

As before, when the data strobe line goes true, the data lines are read into BRA and parity checked. Bad parity is treated as described previously. Good parity causes the input channel to generate a single odd parity for the two data bytes that have been received. This is done since the data word length on the VCS/Local Processor bus is 16 bits plus parity. If the parity on the two eight bit bytes are the same, then a binary one is placed in a parity buffer store (PAR4) to be used by the voter section at the appropriate time. If the two parity bits are not the same, a binary zero is placed in PAR4.

When the load BRB line from the voter section goes true, the contents of BRA are read into BRB and the BRBF flip flop is zero set. If the end of message signal from the computer has not been set true up to this point, the input channel sets the next byte line true and repeats the above actions. If the end of message line had been set true at sometime, the END flip flop is set true. At this point, END being true causes the DONE flip flop to be set true. This signifies the end of the operation for the input channel which goes to the idle mode.

2.2.2.4 Data Transfer to Matrix Section Operation

This operation is performed if the type code field holds a binary one and the matrix operation code field holds a code specifying a set R and P matrices or set all matrices operation. First, the contents of BRA are read into BRB and the next byte line set true. The input channel inputs a data byte as described in 2.2.2.3 when the data strobe line goes true.

Parity is checked with bad parity causing the input channel to request a retransmission by setting the parity bad line true. Good parity causes the input channel to set the matrix interrupt true to the matrix section. The detection of the interrupt serviced signal from the matrix section going true causes the contents of BRA to be read into BRB and the matrix interrupt set false. The next bit time the matrix interrupt is again set true. The operation is terminated when the interrupt serviced line goes true again. The input channel resets all controls and goes to the idle mode.

2.2.2.5 Data Request Transfer to Matrix Section Operation

This operation is performed if the type code field holds a binary one and the matrix operation code field holds a code specifying a set S matrix or sample

operation. The contents of BRA are read into BRB and the matrix interrupt set true. Detecting the interrupt serviced signal going true, the input channel resets all controls and goes to the idle mode.

2.2.3 Voter Section

2.2.3.1 General

The VCS contains one voter section where data from the computer system are voted upon according to a specified voting mode. The data that meets the voting requirements are transmitted serially to the local processor (LP). The data from each computer involved in the voting are compared with the data sent to the LP with the results of that comparison sent to the matrix section. A block diagram of the voter section is shown in Figure 2-6.

The voter section can operate in four voting modes:

1. Selector mode - Data from one computer and no voting.

2. Comparator mode - Data from any two computers with only that data that agrees sent to the LP.

3. Three way voting mode - Data from any three computers with only that data that has majority agreement set to the LP.

4. Four way voting mode - Data from four computers with only that data that has majority agreement set to the LP.

These modes are determined by the R and P matrices in the matrix section and indicated to the voter section by one of 15 logic terms being true. Depending upon which of these terms is true, the conditions for leaving the idle mode of the voter section are determined.

2.2.3.2 Selector Mode Operation

With this mode specified, the mode control requires either the new mode term to go true or the specified BRBF indicator to go true to move the mode control from idle. The new mode term going true means that the matrices in the matrix section have been changed and the voter section should reset and be ready to enter a new mode of operation.

When the BRBF indicator goes true, the most significant bit of BRB in the selected input channel is read into the voter buffer flip flop (VBR). The bit counter is incremented by one count. Each bit time one bit is shifted serially from BRB into VBR and the bit counter incremented. The output of VBR is fed into the transmitter and placed on the LP bus. This continues until the bit counter reaches a count of six when the load BRB line is set true.

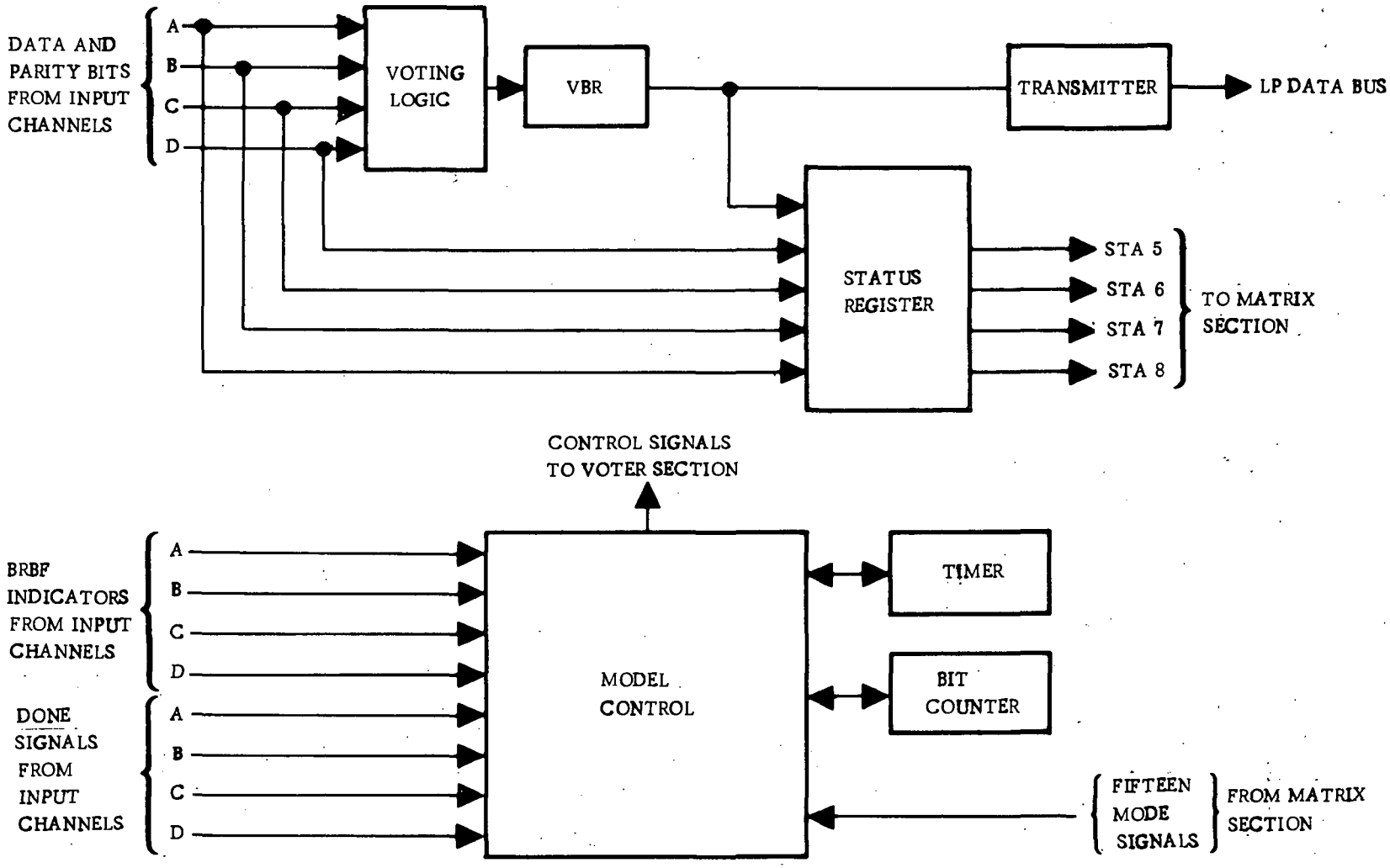


Figure 2-6. Voter Section Block Diagram

Shifting continues until a count of eight is reached in the bit counter. The byte counter flip flop (CYCLE) is caused to change state. If the new state is one set, the voter section has processed the first eight bits of the LP data word and continues shifting bits from BRB as described above. If the new state is zero set, the full sixteen bits of the LP data word have been processed and the parity bit held in PAR4 of the input channel is read into VBR as the seventeenth bit of the data word.

As the parity is shifted out, the DONE signal from the input channel is checked and, if false, the voter section repeats the previous actions. If the DONE signal is true, the voter section places the parity bit on the LP bus and returns to the idle mode.

2.2.3.3 Comparator Mode Operation

This mode is entered when any one of the six terms from the matrix section that define a comparator mode is true. The voter mode control waits for either of the BRBF indicators to go true. Detecting one true, a timer is started. If the timer reaches a count of 15 before the second BRBF indicator goes true, the mode control will reset to the idle mode. If the second BRBF indicator goes true before the timer reaches 15, the mode control proceeds with this type of voting. This timer operation allows time differences of up to 16 microseconds between the arrival of data from two computers and still have proper operation.

The most significant bits of each BRB of the two selected input channels are shifted into the voter logic and into the first stage of the status register. If the two bits agree, VBR is set to that state. If the two bits do not agree, VBR is not changed. The next bit time VBR is compared with the status of the first stage flip flops of the status register. Agreement or disagreement is then entered into the second stage of the status register. The bit counter is incremented once each time new bits are shifted into the voter logic.

The operation is the same as that in 2.2.3.2 except that there are two inputs to the voting logic. This continues until both DONE signals from the two input channels involved are found to be true. The voter section resets to the idle mode.

2.2.3.4 Three Way Voting Mode Operation

This mode is entered when any one of the four terms from the matrix section that define a three way voting mode is true. The mode control waits for a majority of the BRBF indicators of the input channels involved to go true. A timer is started when this happens. The voting begins when either all BRBF indicators go true or the timer reaches the count of 15.

The voting, taking of status and transmission of the data to the LP units is the same as described for the comparator mode except that three input channels are involved. The operation continues until the majority of the input channels involved set their DONE signals true. The mode control resets to the idle mode.

2.2.3.5 Four Way Voting Mode Operation

This mode is entered when the term defining this mode is set true by the matrix section. The operation in this mode is the same as that of 2.2.3.4 except that all four input channels are involved. This mode is terminated when a majority of the input channels set their DONE signals true.

2.2.4 Matrix Section

2.2.4.1 General

The matrix section contains the computer status (P) matrix, the VCS mode (R) matrix and the voting status (S) matrix. These matrices can be set or sampled by the computers in the system. The R matrix modified by the P matrix determines the mode of operation of the voter section. A block diagram of the matrix section is shown in Figure 2-7.

The P matrix contains the status of each computer in the system as a result of the computer's own self test and the evaluation of a computer by each of the other computers. Thus, a computer can be declared bad by its own self test results or by a majority of the non-failed computers declaring the computer to be bad. Binary ones in the P matrix indicate "good" and zeroes "bad".

The R matrix holds the data received from each computer that specifies the voting mode to be performed by the VCS. Binary zeroes in this matrix represent a "don't care" entry by a computer. The output of the R matrix is modified by the P matrix to form the mode signals for the voter section.

The S matrix holds the results of the comparison of the data received from each computer and the data sent over the LP bus. Binary ones in the matrix indicate a disagreement of that computer's data with the transmitted data.

Each of these matrices are sixteen bits in size. Four bits relate to each of the four computers as follows:

1. Computer A — Flip Flops 1 through 4 of each matrix (First Row)
2. Computer B — Flip Flops 5 through 8 of each matrix (Second Row)
3. Computer C — Flip Flops 9 through 12 of each matrix (Third Row)
4. Computer D — Flip Flops 13 through 16 of each matrix (Fourth Row)

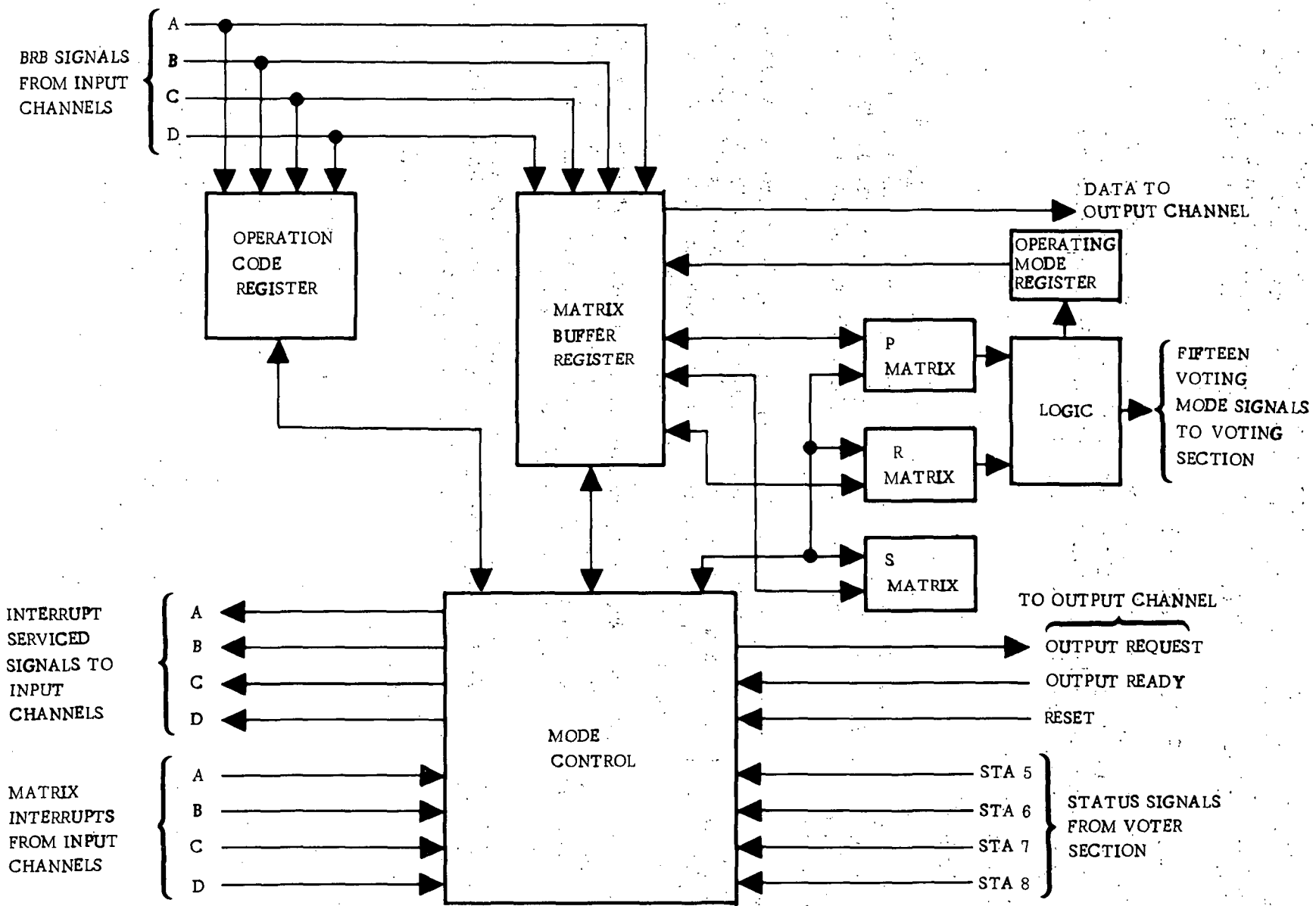


Figure 2-7. Matrix Section Block Diagram

Each computer can read the entire contents of each matrix but can set only those four bits in each matrix assigned to it. These matrices are volatile and their contents must be restored by the computers after loss of power for any reason.

The matrix section operates in one of three modes:

1. Matrix interrupt scanning mode

2. Set one or all of the matrices mode

3. Sample one or all of the matrices mode

The latter two modes are entered as a result of the matrix operation code received as part of the control byte. The matrix section responds to commands from one computer at a time. A sample operation is terminated automatically when signals appear on the LP data bus to the output channel of the VCS.

2.2.4.2 Matrix Interrupt Scanning Mode Operation

This mode is the predominant mode of operation for the matrix section. The matrix section is in this mode whenever power is applied and a set or sample operation has not been commanded. In this mode, the voting mode signals to the voting section are continuously developed and the S matrix is set according to the status signals from the voter section upon receipt of a read status signal from the voter section. The operating mode register is set according to the voting mode signals to indicate which computers are involved in the voting.

The matrix interrupts from each input channel are scanned sequentially by means of a four state counter. At each state of the counter one matrix interrupt is tested. If the interrupt is not true, the counter increments and another interrupt is tested. If the interrupt is true, the counter is held fixed to be used to identify which matrix interrupt and therefore which computer is addressing the matrix section.

Bits one through three of BRB in the input channel whose interrupt was set are read into the operation code register. The interrupt serviced line to the input channel is set true for one bit time. The operation code is deciphered and the proper set or sample mode is entered.

2.2.4.3 Sample Mode Operation

The sample all matrices operation starts with the states of flip flops R1 through R8 copied into the matrix buffer register (MBR). The output request (OREQ) signal to output channel is set true and the matrix section waits for the output channel to set the output ready (OR) signal true. When OR goes true, OREQ is set false and flip flops R9 through R16 are copied into MBR. OREQ is set true again and the matrix section tests the OR signal. When OR goes true, the matrix section copies the states of flip flops P1 through P8 into MBR and OREQ is set true. When OR goes true, OREQ is set false and the states of flip flops P9 through P15 are read into MBR and OREQ set true again. When OR goes true, the matrix section goes on to sample the S matrix. The contents of the S matrix are transferred to the output channel in the same way as that of the R

and P matrices. When OR goes true after sending the last bits of the S matrix, the matrix section goes to the matrix interrupt scanning mode.

The sample S matrix operation is a subset of the sample all matrices operation. The mode control goes to the state where the contents of the S matrix are copied into the MBR. The operation concludes as described in the previous paragraph.

The sample P matrix diagonal and operating mode operation is performed as a sample S matrix operation except that elements P1, P6, P11 and P16 of the P matrix and the four flip flops forming the operating mode register are copied into the MBR.

2.2.4.4 Set Mode Operation

The setting of the matrices consists of placing four bits in the R matrix, three bits in the P matrix or zero setting four bits in the S matrix. The data to go into the P and R matrices are received from the computer as an eight bit byte. The R matrix data is always in bits 5 through 8 and the P matrix data always in bits 1 through 4.

The set all matrices or set R and P matrices operations are essentially the same operation. After setting the interrupt serviced line false, the matrix section waits for the same matrix interrupt to go true again. When it does, the contents of BRB of the input channel are read into MBR and the interrupt serviced line set true for one bit time. The respective bits of MBR are read into the proper flip flops in the R and P Matrices according to the settings of the operation code register. The matrix section goes to the matrix interrupt scanning mode if the operation was a set R or set P matrix operation or goes on to zero the S matrix if the operation was set all matrices.

The set S matrix operation has the four flip flops of the S matrix identified with the state of the matrix interrupt counter zero set. The matrix section then goes to the matrix interrupt scanning mode.

2.2.5 Output Channel

2.2.5.1 General

The output channel provides the interface between the computer system and the VCS for data being sent to the computer and the interface between the LP data bus and the VCS for data sent from the LP. The output channel also forms the data path between the matrix section and the computer system. A block diagram of the output channel is shown in Figure 2-8.

The LP/output channel interface consists of a data bus that is used to send data serially between the LP and the VCS. Data words received over this bus are 16 bits plus an odd parity bit.

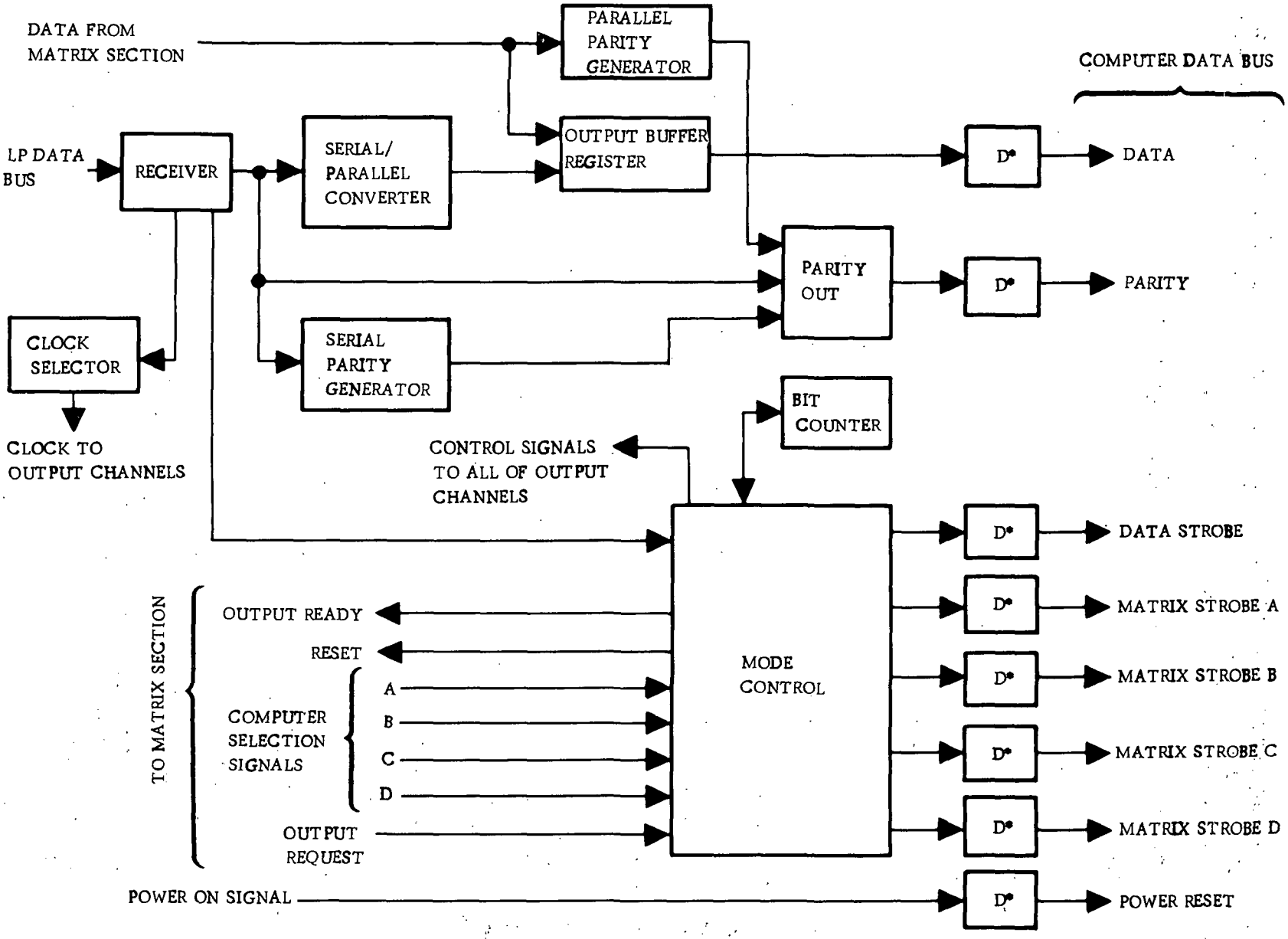


Figure 2-8. Output Channel Block Diagram

The computer/output channel interface consists of a parallel bus with data being handled in bit parallel, byte serial fashion in eight bit bytes. Odd parity is sent with each byte. The interface consists of:

	Source	Quantity
1.	Data	8
2.	Parity	1
3.	Data Strobe	1
4.	Matrix Strobe A	1
5.	Matrix Strobe B	1
6.	Matrix Strobe C	1
7.	Matrix Strobe D	1
8.	Power Reset	1

The data, parity, power reset and data strobe lines are connected to all computers in a party line fashion. The matrix strobe lines are assigned to a specific computer. LP data are sent to all computers using the data strobe signal to indicate valid data on the data lines. Matrix data are sent to one computer by using one of the matrix strobe lines to indicate valid data on the data lines.

The output channel converts the serial data into parallel binary data in eight bit bytes. Parity on the bytes sent to the computer is generated such that good or bad parity as received from the LP is sent on to the computer without any change. The computer makes the decision on good or bad data as shown by parity. The output channel operates in either of two modes:

1. Matrix data to computer mode
2. LP data to computer mode

The LP data to computer mode has absolute priority. The appearance of a signal on the LP data bus causes the output channel to terminate the matrix data to computer mode and to start the LP data to computer mode.

The power reset signal indicates that power to the VCS has dropped below operating levels sometime earlier and has just returned to normal. This loss of power can be from normal shutdown and re-application of power or due to some system transient on the power lines. This signal going true tells the computer system that the P and R matrices must be reset before the VCS can perform any operations. If a power transient were responsible for power reset being set true, the computer also knows that the last operation in progress was not completed successfully.

2.2.5.2 Matrix Data to Computer Mode Operation

This mode is entered if there is no signal on the LP data bus and the OREQ signal from the matrix section goes true. The contents of MBR of the matrix

section are read into the output buffer register (OBR) and parity generated for the byte. The appropriate matrix strobe line is set true according to the signals received from the matrix section. The OR signal is set true at the same time for one bit time. After two bit times, the matrix strobe line is set false. The OREQ line is tested and, if true, the above operation is repeated. If the OREQ signal is false, the output channel goes to the idle state.

2.2.5.3 LP Data to Computer Mode Operation

The mode is entered any time a signal is detected on the LP data bus to the output channel. Timing is derived from the data bus to be used as the clock in the output channel during this mode of operation.

The serial data are shifted one bit at a time from the receiver into a serial to parallel converter (SPC). Odd parity is generated as the SPC is filled with data. A bit counter is incremented one count each time a bit is read into the SPC. When the counter reaches a count of eight, the contents of the SPC are read into OBR and the parity read into the parity output flip flop. The data strobe line is set true for two microseconds to let the computers read the data lines.

The SPC continues to receive data bits and the bit counter continues to increment until the count of 16 is reached. The contents of the SPC are read into OBR as before. This time the parity bit for the byte is not derived from the data that is in OBR. The parity bit received from the LP is tested for value. If this parity bit is a binary one, the parity bit for this data byte is made the same as the parity bit for the previous byte. If the parity bit received from the LP is a binary zero, the parity bit for the second byte is made the opposite of that for the previous byte. In this way, the correctness of the parity received from the LP is passed to the computer system with no change by the VCS. The second byte is sent to the computer system as the first byte was sent.

The above actions are repeated until the signal on the LP data bus diasppears. The output channel generates the proper parity for this last byte, sends it to the computer system and returns to the idle state.

2.3 VCS/COMPUTER INTERFACE

This section describes the VCS operation from a system application standpoint.

The VCS device is designed for application in highly reliable, redundant computer systems employing redundant data buses as the primary communication paths between the computer system and other subsystems. The primary function of the VCS in such a system is to control the transmissions on the data bus through comparison/voting on redundant copies of data from the multiple (up to 4) input channels (from redundant computers). It therefore has a dual purpose: (1) prevent undetected transmission of erroneous data, and (2) detect computer failures through data comparison/voting.

Figure 2-9 is a block diagram of the application of the VCS in a quad redundant system. Note that the VCS design does not include internal redundancy and therefore the VCS is replicated in the system to the redundancy/reliability level dictated by the overall system requirements.

2.3.1 Computer Interface Requirements

The design of the VCS input and output channels is intended to provide an efficient, interface with a wide variety of computer system designs. An eight bit data byte was chosen because of its common usage and compatibility with the most popular computer word lengths (16, 24, & 32) as well as the 16 bit basic word length to be used on the data bus.

The 1MHz clock rate on the data bus requires a rate of 125K bytes/sec at the computer/VCS interface. This rate must be sustained in both directions simultaneously during a transmission over the data bus. With current CPU/memory speeds, this requirement precludes use of a CPU program controlled I/O channel to drive the VCS interface. An independently operating, but CPU-controlled, I/O channel having direct access to the computer operating memory would appear to be the most efficient design. Such an interface is common for magnetic tape drives on small and medium-size computer systems. A block diagram of a typical interface design is shown in Figure 2-10.

Application of the VCS in a redundant system requires a means of synchronizing I/O operations of the redundant computers. Since the data comparison/voting process performed on data bus transmissions is serial on a byte basis, multiple computers must be synchronous within 8-16 μ sec* in initiating an I/O operation. Once an operation is successfully initiated, all data is transmitted over the VCS/computer interface upon VCS request where each computer must respond within 8 μsec. The Input/Output process is described in detail in Section 2.3.3; the primary implication in terms of computer interface requirements is the need for a timing source to provide the necessary synchronization.

In order to utilize the voting logic in the VCS effectively, a self-test signal is required from each computer. This signal should be controlled by logic of the watchdog timer (dead man's switch) type which requires that the computer maintain a specific, time-critical sequence of operations in order to hold the self-test signal in its "Go" state.

In addition to the watchdog timer, the results of any other built-in-test circuitry and program-controlled self tests should be "ored" to form the Go/No go signal wired to the VCS(es).

---

*The precise requirement is that a given computer must transmit its control byte no later than 15 μsec after a majority of the control bytes have been received from the computers in the current operating mode. See Section 2.3.2.2 Input/Output Operations.

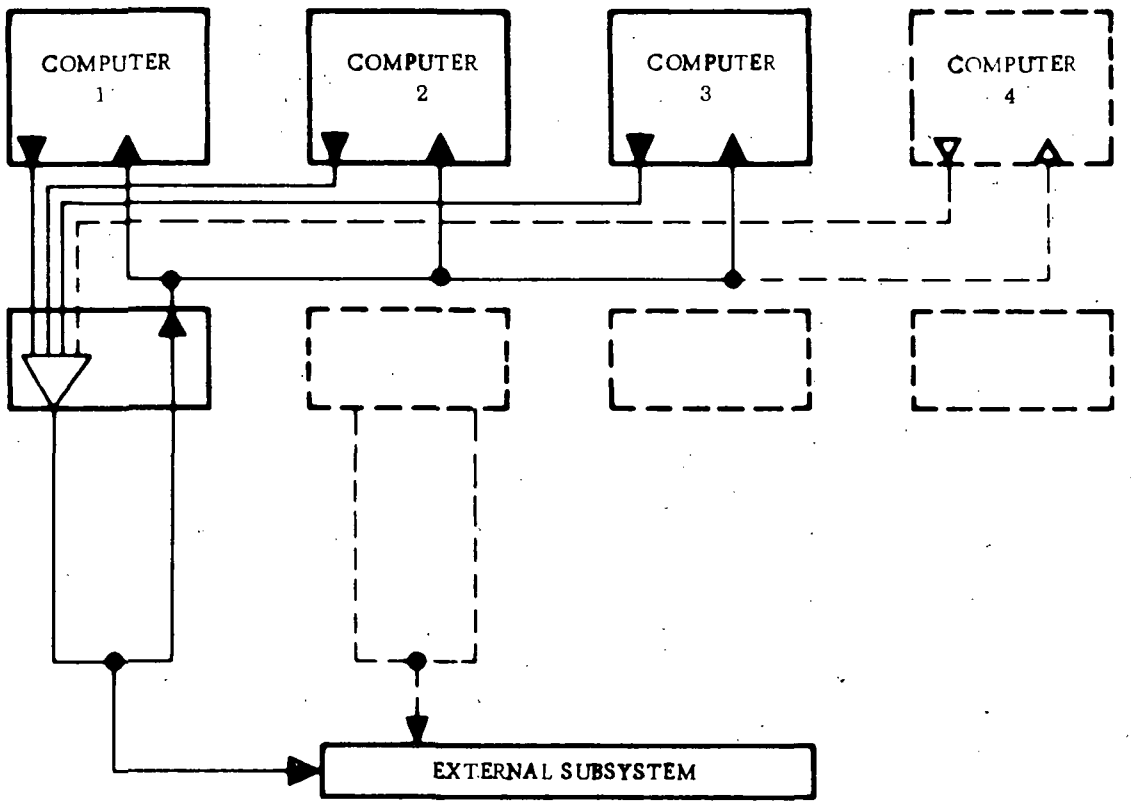


Figure 2-9. Quad Redundant VCS System

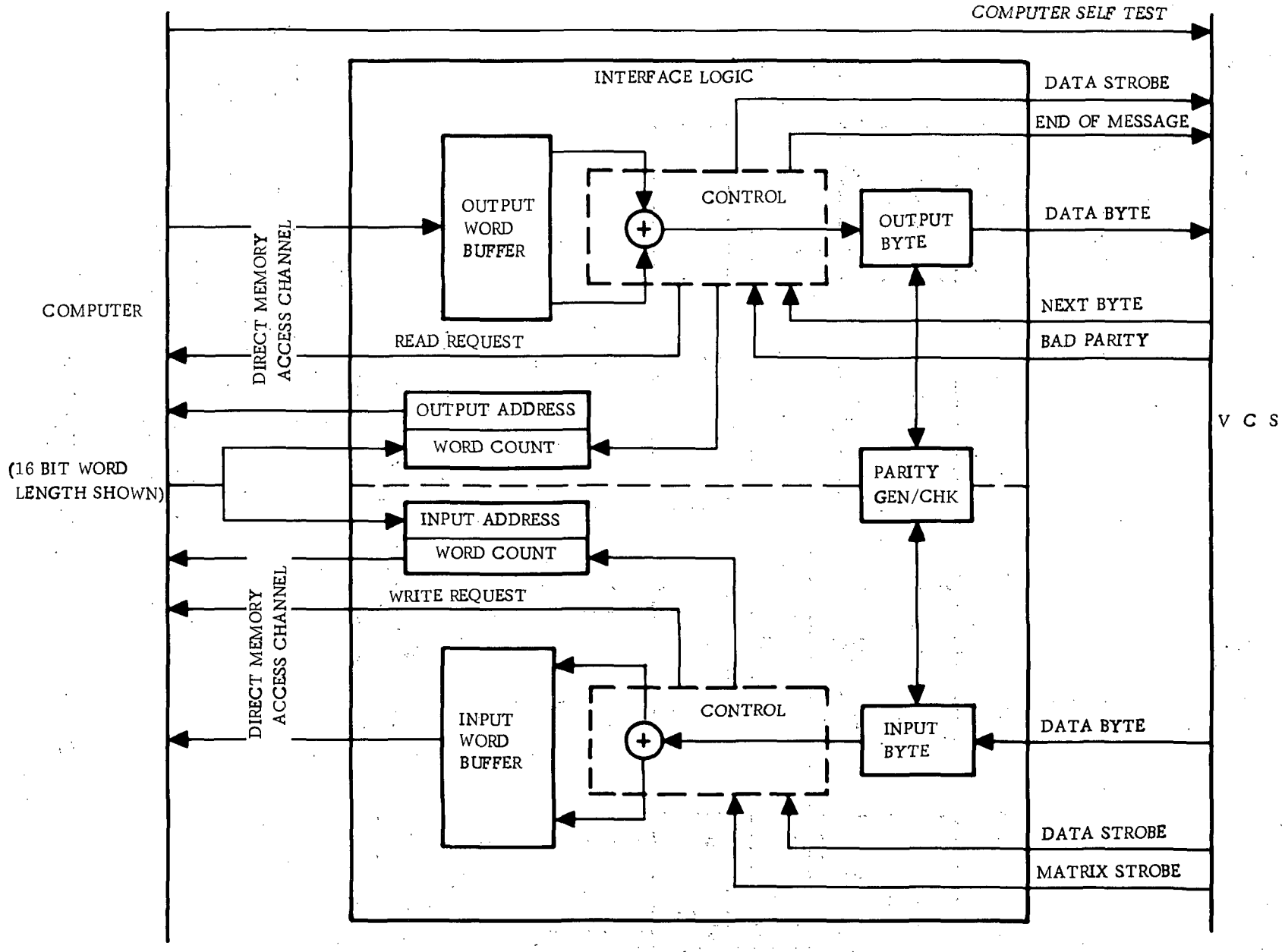


Figure 2-10. Typical Interface

2.3.2 VCS Operations

By transmitting one of the seven control byte configurations, a computer can initiate one of the two types of VCS operations in the particular VCS addressed by the control byte:

1) Matrix operations - used to modify or sample the VCS mode/status data.

2) Input/Output operations - used to transmit/receive data over the data bus between the VCS and the subsystems external to the computer system.

Note that matrix operations can be initiated by a single computer and proceed independently, whereas I/O operations can only be initiated through the "agreement" of a majority of the non-failed computers in the system.

2.3.2.2 Matrix Operations

The six control bytes associated with VCS matrix operations are described in Section 2.2. These are functionally quite simple and are used to modify/sample the mode/status information associated with the three matrices in the VCS. The matrices, designated P, R, & S, are all 4 x 4 where a row of each is associated with each of the four input channels (computers).

The function of the three matrices is described below:

P-Matrix - This matrix determines the status of each of the four computers. The diagonal elements of this matrix determine which of the computers is considered operational (i.e., non-failed). The other three elements of each row of the matrix can be set by a control byte from the computer corresponding to that row; these elements specify this computer's opinion of the status of the other three computers. The states of the diagonal elements are controlled by majority voting logic which uses these 12 "opinions" and the 4 computer self-test signals. The diagonal elements of the P-matrix are then used to determine which computers are to be used by the R-matrix in selecting the VCS operating mode.

A computer can load the three non-diagonal elements of its row, sample the four diagonal elements, or sample the entire 16 element array. In a system application, the load P operation would be performed during system initialization, recovery from a power transient, or during reconfiguration subsequent to a computer failure. Sampling the P-matrix diagonal would be performed on a periodic basis in order to detect computer (or VCS) failures. The frequency of the sampling period would be determined from the system requirements in terms of allowable reconfiguration time, etc. Sampling the entire P-matrix is useful in diagnosing a failure for isolation purposes and could be used to help resolve fault isolation anomalies or during VCS tests.

R-Matrix - This matrix determines the operating mode of the VCS. The VCS mode determines which computer(s) are to participate in subsequent I/O operations. Sixteen modes

are provided corresponding to all possible combinations of the four computers. Each computer selects the mode it wants by setting the four elements in its row (all zero's means "don't care"). The actual mode is determined by majority voting logic using only the mode selections of the computers which are considered operational (according to the P-matrix diagonal).

A computer can load the four elements of its row, sample the entire 16 element array, or sample a four bit register which contains the current VCS operating mode. Sampling the VCS operating mode register would be performed any time the mode selection is to be changed, and on a periodic basis for failure detection. Sampling the entire R-matrix is analogous to the corresponding P-matrix operation and would be used to help resolve fault isolation anomalies or during VCS tests. In a system application, the load R operation would be used to change VCS operating modes as a result of a system reconfiguration (due to computer failure) or to transmit data of varying degrees of criticality. Even in a highly redundant system some functions may not be sufficiently critical to warrant full parallel redundancy. After operating in a 3-way voting mode for highly critical data, the VCS might be switched to a selector or comparator mode for transmission of less critical data. An alternate operating procedure might use one bus/VCS for critical data (3 or 4 way voting) and another bus/VCS for less critical data (comparator or selector).

S-Matrix - This matrix stores the results of the voting process performed during I/O operations. The four rows of this matrix are identical but are duplicated to provide an independent sample for each computer. Each of the four elements of a row determine whether a computer disagreed, i.e., was "voted out", during an I/O operation in which the VCS mode said it was to participate.

A computer can clear (set to zero) the four elements of its row, sample the four elements of its row, or sample the entire 16 element array. In a system application, the S matrix sampling would be used to detect computer failures. The frequency of the sampling would depend on the system requirements in terms of allowable reconfiguration time, etc., however it would also be necessary/desirable to test the voting logic of the VCS periodically during system operation. The sampling operation would be used to obtain the results of the tests. Clearing a row of the S-matrix would generally be performed after a sample is taken though this is not necessary if no voting discrepancies had occurred since the last clearing operation.

Figure 2-11 is a flow diagram of the operation and timing of the computer/VCS interface during the matrix operations.

2.3.2.2 Input/Output Operations

The term Input/Output operations refers to transmission over the serial data bus which is the primary communication channel between the VCS/computer

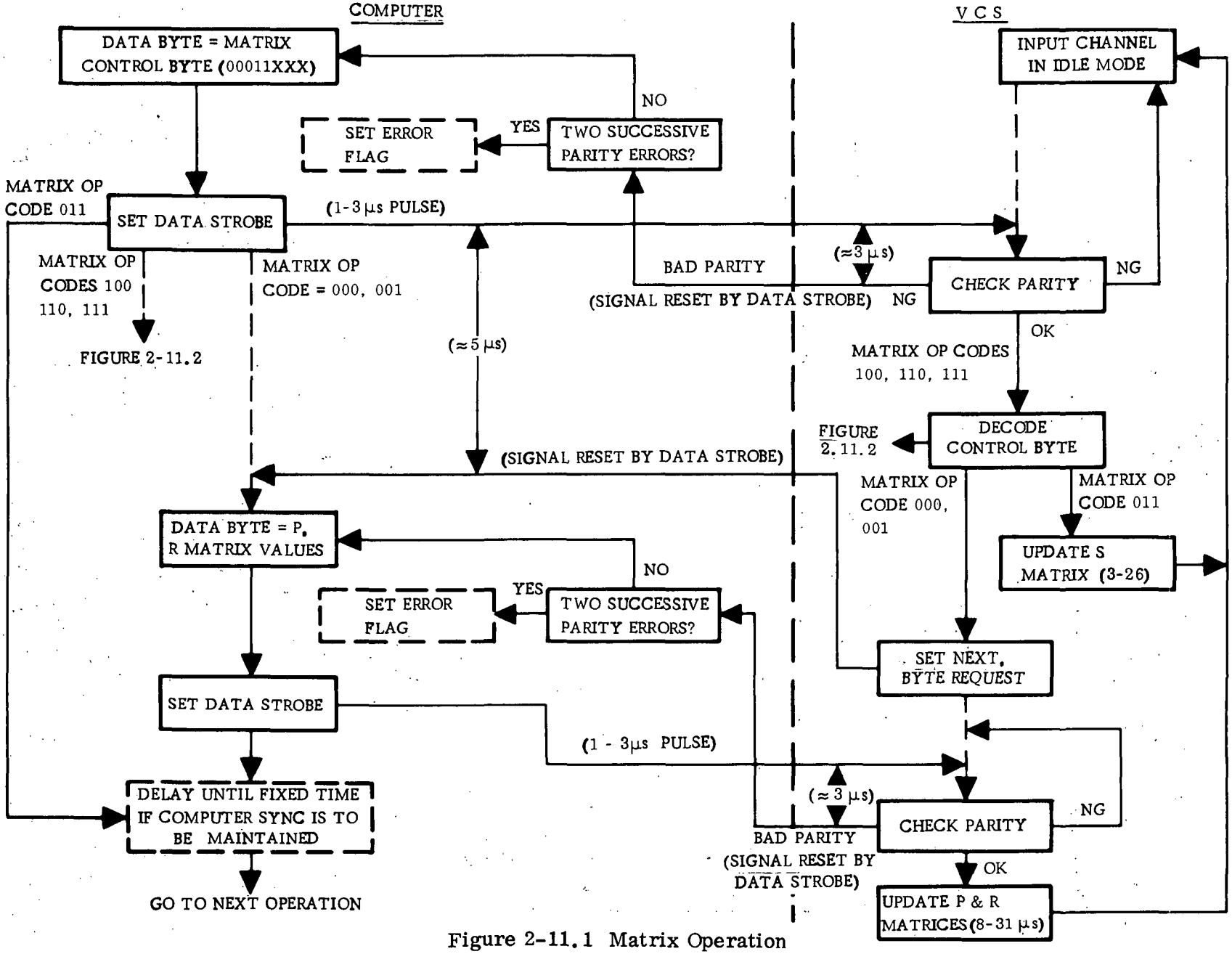


Figure 2-11.1 Matrix Operation

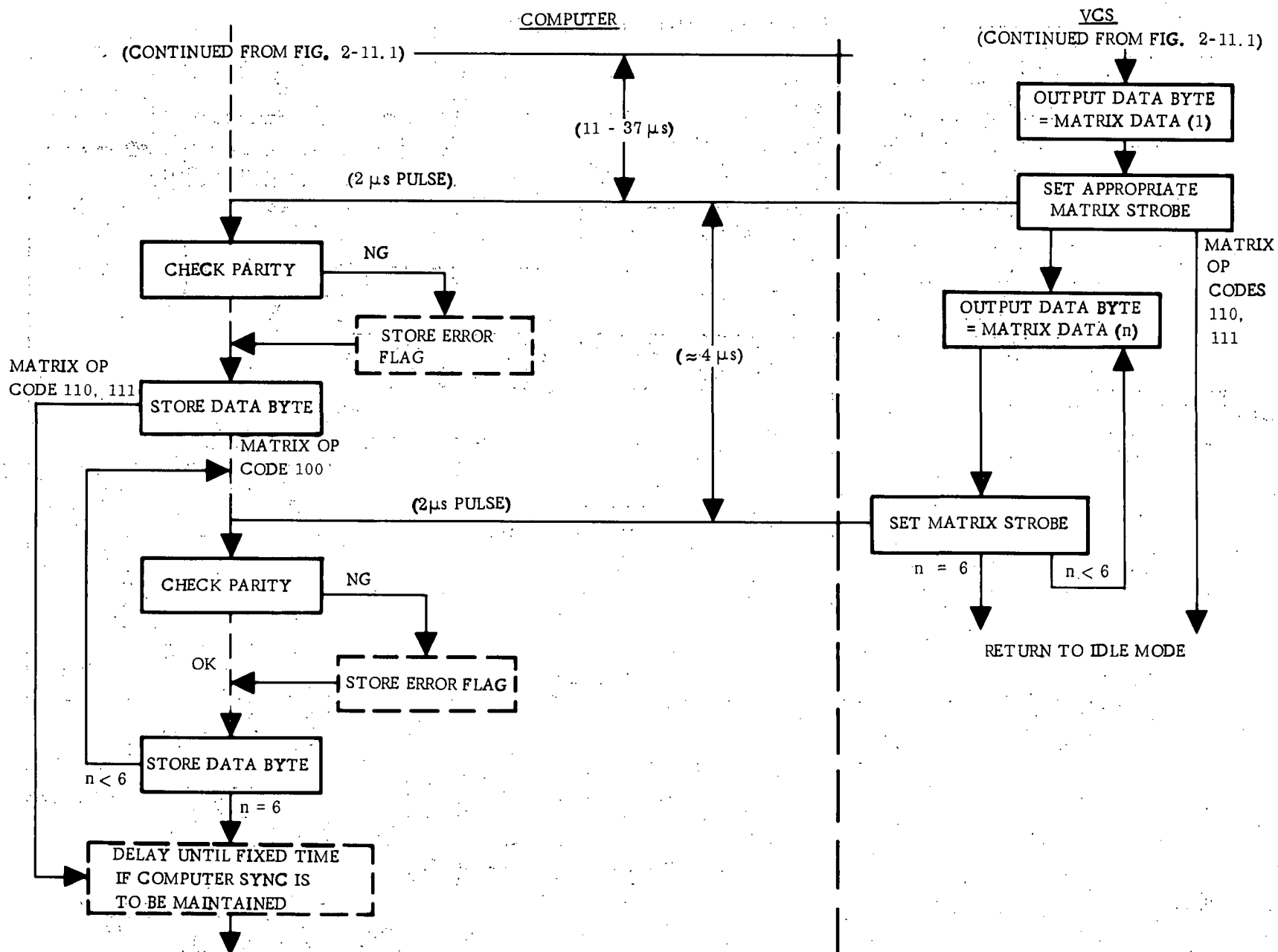


Figure 2-11.2 Matrix Operation (Continued)

system and external subsystems. The VCS does not distinguish between input and output since bidirectional transmission at the computer/VCS interface occurs in either case, hence, a single control byte is used to initiate input or output and the subsequent data transmitted is used at the subsystem interface to determine whether data is being transmitted to, or requested from, the subsystem being addressed.

Initiation of any VCS I/O transmission requires that the I/O control byte be received from a majority (1-1, 2-2, 2-3, 2-4) of the computers indicated by the current VCS operating mode. The actual transmission begins, 1) 16 μsec after the control byte majority is received, or 2) when the control byte is received from all computers indicated by the current operating mode. Once initiated, the VCS transmission continues until the "End of Message" signal is received from a majority of the participating computers. Note that the control byte is not transmitted on the data bus.

As can be seen in Figure 2-3, the VCS contains an independent bus receiver channel which detects data transmission on the serial bus, formats the data into 8 bit bytes, and presents the data on the computer system interface. In the case of an output operation the data on the bus is simply a feedback of the data transmitted by the VCS. This feedback can be used by the computer system to verify proper data transmission. In the case of an input operation, the data on the bus is the result of a transmission from an external subsystem and represents data to be stored by the computer system. Note that every 16 bits transmitted on the bus are followed immediately by a bit which represents odd parity on the preceeding 16 bits.

As previously mentioned, the VCS does not distinguish between input and output operations. Any transmission on the serial bus is monitored and presented to the computer interface. The computer system software makes the decision as to the functional significance of the data.

The VCS design does not constrain the format of data transmissions on the bus except for the 16 bit word parity format. However, the VCS was designed as part of a total system concept oriented toward highly reliable total system operation. The following paragraphs describe the recommended system I/O approach.

2.3.2.2.1 Output

Figure 2-12 shows the message format for an output transmission. The first word (two bytes) of the message (ignoring the control byte which is not transmitted) is used to address the particular subsystem(s) and to identify the message type. As this word and subsequent data words are transmitted by the participating computers, the VCS performs a serial voting function according to its current mode and transmits the majority opinion on the bus. The output channel of the VCS receives the transmitted data and transfers it in 8 bit bytes to the computer system. Each participating computer compares the received data with its own version of the data (i.e., the data that it sent to the VCS). The "feedback" of a data byte lags its transmission to the VCS by 8-15 μ sec. When the data block is completed, a block parity word is transmitted. During transmission of the parity word on the bus, the feedback of the last data word

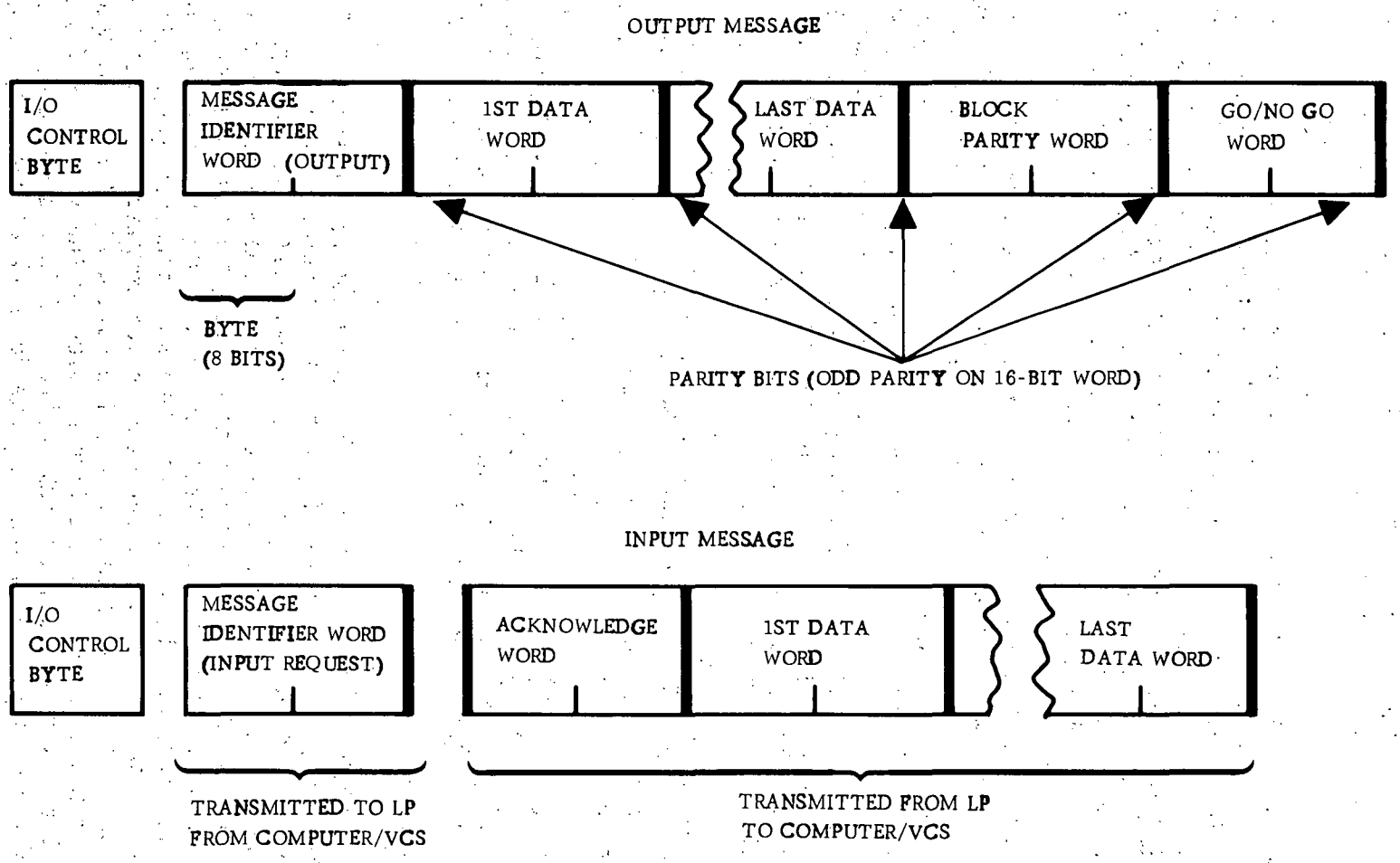


Figure 2-12. Message Formats

is received and, based on the results of the feedback comparison during the entire message, each participating computer selects either a "Go" or a "No Go" word to transmit as the final word of the message. The Go/No Go word passes through the VCS voting logic just as the rest of the message and can then be used by the subsystem as a validation of the message transmission. Moreover, the feedback of this word can be used by the participating computers to determine whether the message should be re-transmitted (either on the same bus, or another bus).

Note that the final event in the output process is the receipt of the Data Strobe signal from the VCS to the participating computers. Since this signal is common to all computers it provides a synchronizing point for subsequent I/O operations.

Figure 2-13 is a flow diagram of the operation and timing of the computer/VCS interface during an output operation.

2.3.2.2.2 Input

Figure 2-12 shows the message format for an input transmission. All communication between the computer system and external subsystems is presumed to be initiated by the computer system. Hence, an input transmission is initiated by an output transmission of an "input request" message. Transmission of the input request message is analogous to the output operation just described except that a Go/No Go word is not transmitted. However, the results of the feedback comparison of the input request are used to determine whether the resultant input transmission is valid.

The format of the subsequent transmission from the subsystem to the VCS is somewhat flexible but would generally be headed by an "acknowledge" word which is used to verify subsystem identification, message length, subsystem status, etc. For critical data, it is anticipated that the subsystem would transmit its data over multiple bus/VCS links in order to provide a means of verifying the transmission.

Figure 2-13 is a flow diagram of the operation and timing of the computer/VCS interface during an input operation.

2.3.3 Error Detection

Detection of errors/malfunctions is the primary function of the VCS device. The VCS is designed to be independent from the remainder of the computer system in terms of failure propagation. Therefore it is significant to consider the effect of two categories of failures, VCS failures and computer failures. The VCS has been specifically designed to detect the latter category. The design has been further refined to insure that failures in the first category do not propagate to other elements of the computer system and/or cause erroneous data transmission to go undetected.

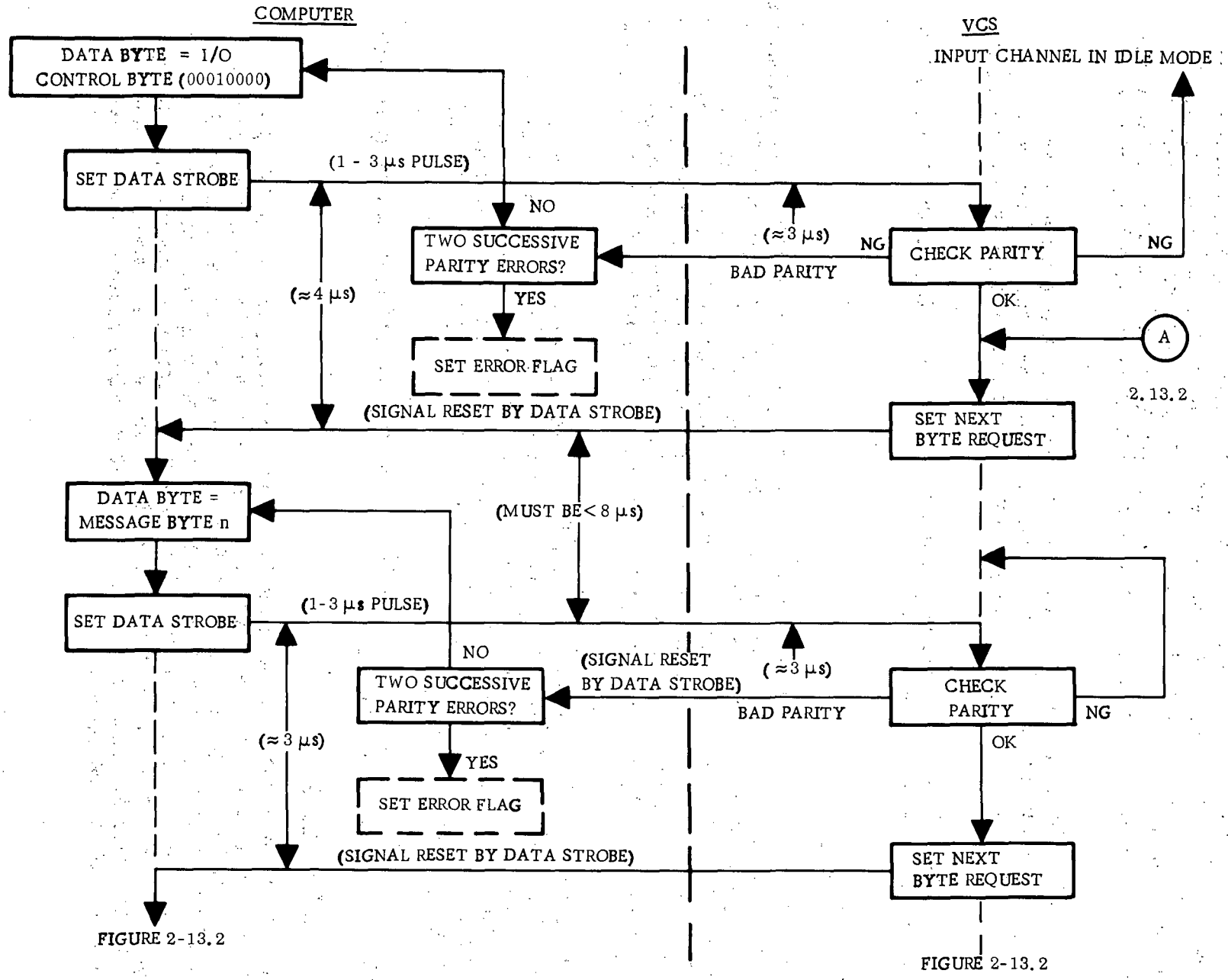


Figure 2-13.1 Input/Output Operation

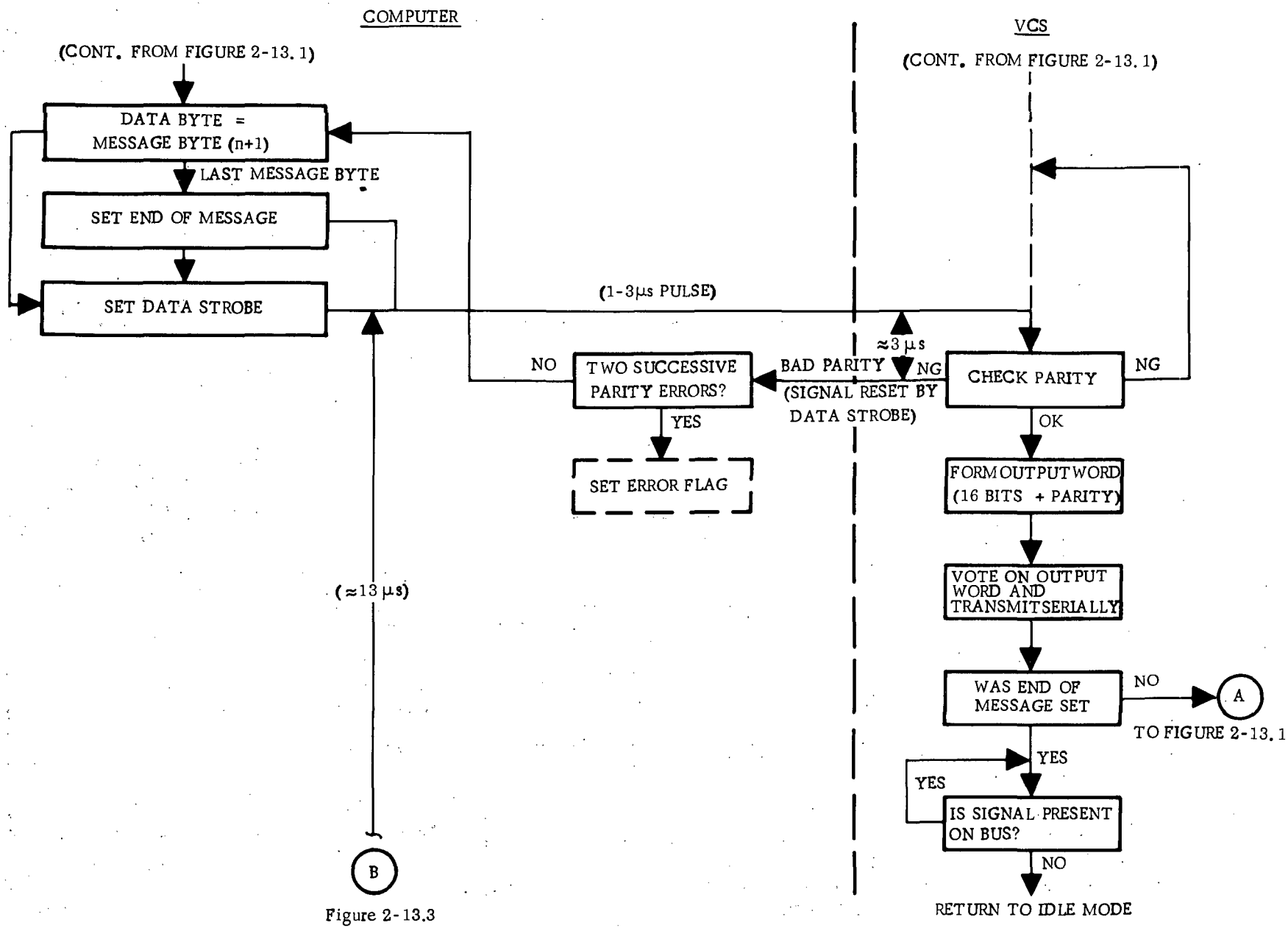


Figure 2-13.3

Figure 2-13.2 Input/Output Operation (Continued)

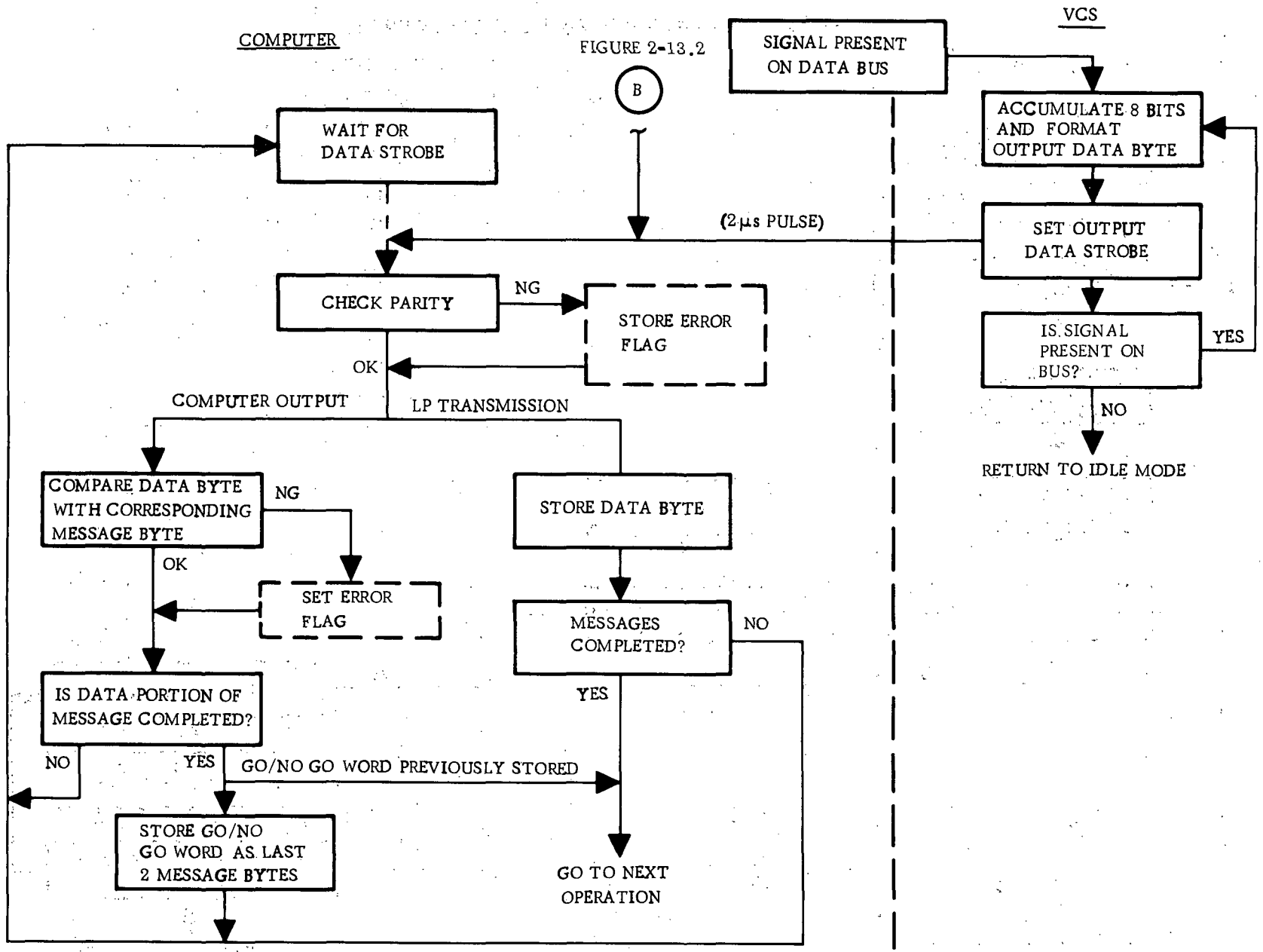


Figure 2-13.3 Input/Output Operation (Continued)

2.3.3.1 Computer Failures

It is intended that the P-matrices in redundant VCSes be used as the primary status for the redundant computers in the system. By sampling the P-matrix diagonals, any computer in the system can immediately determine the state of the entire system as reflected by the majority consensus. This allows the reconfiguration logic of the system to be readily distributed among the computers (presumably in the software) while avoiding the need for some sort of central, "hard-core", control. Note, however, that multiple VCSes must be utilized in determining this status in order to protect against VCS failures.

Computer failures will be detected in one of two ways, 1) the self-test signal goes false due either to loss of CPU program control, built-in-test circuitry, or programmed self test or, 2) the voting status, S matrix, indicates a disagreement. In the first case, the appropriate P-matrix diagonal will be zero set immediately. In the second case, the "good" computers, upon noting the S-matrix discrepancy, will modify their P-matrix rows to "vote out" the indicated computer.

It is interesting to note that previous studies have indicated that a high percentage (90%+) of computer failure modes can be detected by a combination of programmed self-test and watchdog timer circuitry. This type of detection also provides the most straightforward detection/reconfiguration procedure in the VCS system.

Another area of significant concern is potential propagation of computer failures to/through the VCS. The majority voting logic employed throughout the VCS design is specifically intended to prevent failures from propagating through the VCS. Moreover, in all but one area, the data channels between the VCS and the computers are independent, dedicated wires and logic. The single exception is the VCS output channel which is a byte-parallel bus to all computers (with dedicated strobes for matrix data, however). It is intended that isolation circuitry be provided at the computer interface to this bus to prevent a computer failure from destroying operation on the bus.

2.3.3.2 VCS Failures

One of the primary goals of the simulation activity performed during this study was to investigate the operation of the proposed VCS design in the presence of logic gate failures. Of particular concern was the identification and elimination of any "single point" failure modes in the design, where a single point failure refers to a VCS failure which would propagate to some other element of the computer system. This activity is described in Section 3 of this report.

In general, two techniques are employed in detecting and isolating VCS failures. The primary means of fault isolation is the use of redundant VCSes to allow comparison of results to provide isolation between computer and VCS failures. The second technique is a periodic test of the voting logic in the VCS to uncover malfunctions which, if not detected, could cause subsequent computer errors to go undetected also.

When combined with the VCS testing, the normal operation of the computer/VCS system provides a rather, comprehensive verification of the VCS device. When an error is detected during system operation, a redundant VCS is used to determine whether the error should be attributed to the VCS or to some other element of the computer system.

2.4 RECOMMENDED MECHANIZATION

The preceding paragraphs of Section 2 of this report have described the functional requirements of the VCS with respect to its functional place in the operational system. Translation of this functional design (as described by the logical equations of Appendix A) into hardware must take into account other aspects of the system and the state of the art in available technologies. At present, there are two major technologies that can be used to mechanize the VCS with a low risk. These are the TTL bipolar and four phase P channel MOS processes. To provide a subsystem with the required performance and greatest flexibility, each process must be looked at in terms of VCS needs and the decision made to use either process exclusively or to use both to get the benefit of the unique advantages of each process.

The VCS is to be used in systems where data are received from and transmitted to many devices by the VCS. These devices and systems may be present or future designs so the VCS should be made to be capable of interfacing with the greatest number of systems. In the near future (five to ten years) the technology most used for system interfaces within a package will be TTL. Thus, the VCS interface will have the greatest flexibility if it is mechanized in TTL circuits.

A second way of looking at the role of the VCS in a system is from the physical aspect. The VCS is to be an "add on" unit placed inside of some host device. The design of the VCS should be such that the host is impacted the least. The host is impacted least if the VCS could be packaged as a component capable of being mounted on a module in the host. The complexity of the VCS precludes this in the near future. The next level of low impact is to have the VCS on one plug in module. Since the hosts and, therefore, the module size can vary with the system, the VCS design should be made such that the physical layout can be made at the same time as the host or when the host has been decided upon. Relatively simple module layouts can be achieved by mechanizing a unit with large scale integrated (LSI) devices with much of the logical interconnections in the device instead of on the module. This gives a fairly simple module that can be quickly designed for the VCS to match the host.

A third way of looking at the role of the VCS in the system is from the power and heat aspect. To minimize the impact on the host, the power required and, as a result, the heat dissipated by the VCS should be as low as possible. This will either allow operation of the VCS using some of the reserve capacity of the host or, if the host must increase its power capability, lessen the loss of efficiency when the host is operated without the VCS. It is a general fact that the more functions that are placed on an LSI chip, the less power per function that is required. This is due to the need for higher drive for signals that go between two devices than for signals that stay on the chip. The MOS process, however, permits smaller transistors on the chip so that most functions per chip can be achieved for equal chip areas than in TTL and these smaller devices use less power. In addition to this, the use of four phase clock logic eliminates direct DC paths to ground on the chip. This means that power is dissipated only when the devices on the chip change state and require the charging of a junction capacitance.

The above discussion gives reasons to use four phase MOS devices for the logic mechanization of the VCS. Now, the decision to use existing standard building block devices or to generate new custom design devices must be made. Standard building block devices do not require the design expense of custom design but, since these blocks are designed to satisfy a wide variety of applications, they carry functions that may not be useful. Also, usually more than one device type is needed to obtain the performance desired. Finally, since each application is rather unique, some type of control must be custom designed to control the standard devices. This is usually some custom MOS devices or a micro-program control unit. The use of a large number of devices is in opposition to the requirements for the physical and electrical aspects of the VCS. Since even for the standard device design some unique design is required, it was felt that the VCS would be more desirable if the most efficient design was made by custom designing the necessary devices. This decision is further helped by the fact that the VCS can be separated into functional areas that can be readily placed on a single chip.

The four phase MOS mechanization requires the generation of a four phase clock for the MOS circuits. A square wave signal having a frequency nine times that represented by the VCS bit time is required from the host. The signal is converted into a four phase clock by some TTL integrated circuits and a standard hybrid thin film clock driver to provide the proper power and interface to the MOS devices. The VCS bit time for this study has been characterized as one microsecond but the VCS design is such that the actual bit time can be selected anywhere between 0.8 and 3 microseconds without any change in VCS performance.

In summary then, it is recommended that the VCS be mechanized with TTL integrated circuits, custom design four phase P channel MOS/LSI devices and a hybrid thin film clock driver. The TTL circuits are to be used to mechanize the following functions:

1. VCS/Host Interface

2. MOS/Bipolar and Bipolar/MOS translation

3. Four Phase Clock Generation

The custom MOS/LSI devices are to be used to mechanize the logical equations of Appendix A. Each functional area of the VCS (as described in Section 2.2) will require one MOS/LSI device except the matrix section which will need two MOS devices due to the large number of control signals to the other areas of the VCS. The entire VCS will be placed on a single plug in module with a minimum of 40 square inches of area for component mounting. A single connector having a minimum of 90 pins is sufficient to meet the VCS interface requirements.

SECTION 3

SIMULATION

A software simulation system was developed in order to evaluate the detailed logic design of the VCS; in particular, the failure modes of the design.

3.1 SIMULATION SYSTEM DESCRIPTION

The software system developed to evaluate the detailed design of the VCS device is an extension of the computer system simulation developed during the Reconfigurable G&C Computer Study Contract. The current system consists of two programs, the Logic Preprocessor Program and the Simulator Program. These programs are written in the Fortran IV language and are designed for execution on the IBM S360/85 at Autonetics or the XDS Sigma 5 at NASA MSC.

Essentially, the software is designed to provide a logic-level simulation of the VCS device together with sufficient simulation of the VCS interfaces to exercise and evaluate the proposed VCS design. Additionally, the simulation software provides the capability to selectively introduce logic gate failures into the simulated system.

Figure 3-1 is a block diagram of the system being simulated. Only the VCS device is simulated at a logic element level. The necessary interface functions (computers and local processors) are simulated accurately on a bit-by-bit basis at the VCS logic clock rate; however, internal operation of the computer(s)/local processor(s) is simulated at a functional level and only to the extent necessary to provide an accurate VCS interface and to monitor operation of the VCS.

Figure 3-2 is a block diagram of the simulation system operation.

The VCS logic equations have been prepared in the form of a punched card deck and this is the primary input to the simulation. The equations are written in standard and/or notation using (+) for logical "or", (.) for logical "and", and (-) for logical inversion. The number, 1, preceding a term name indicates the one-set side of a flip-flop and 0, the zero-set side. All flip-flops are assumed to be JK type; i.e., if both one-set and zero-set inputs are true, the flip-flop changes state and if neither input is true, the flip-flop retains its current state. All logic is clocked at a 1 MH rate implicit in the simulator.

Prior to executing the Simulator program, the VCS logic equation card deck must be processed by a separate program called the Logic Preprocessor. This program checks the equations for syntax errors, generates various listings of the logic equations, and converts the logic into two separate formats:

1. Logic Equation Subroutines - This is a direct translation of the logic equations into a form compatible with the Fortran IV language. These subroutines are then compiled by the Fortran compiler to form executable program modules which will simulate the VCS logic.

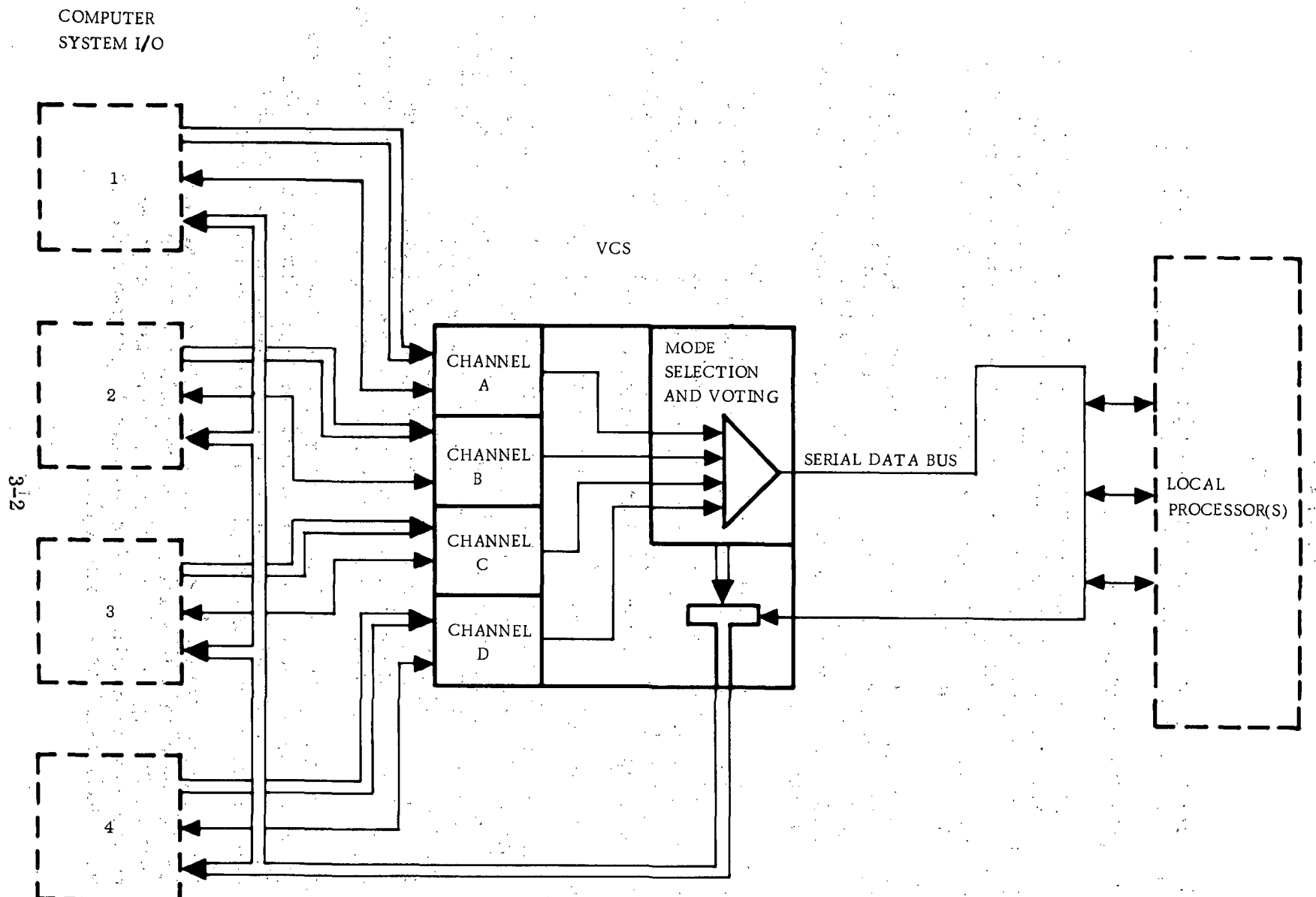


Figure 3-1. Simulation System

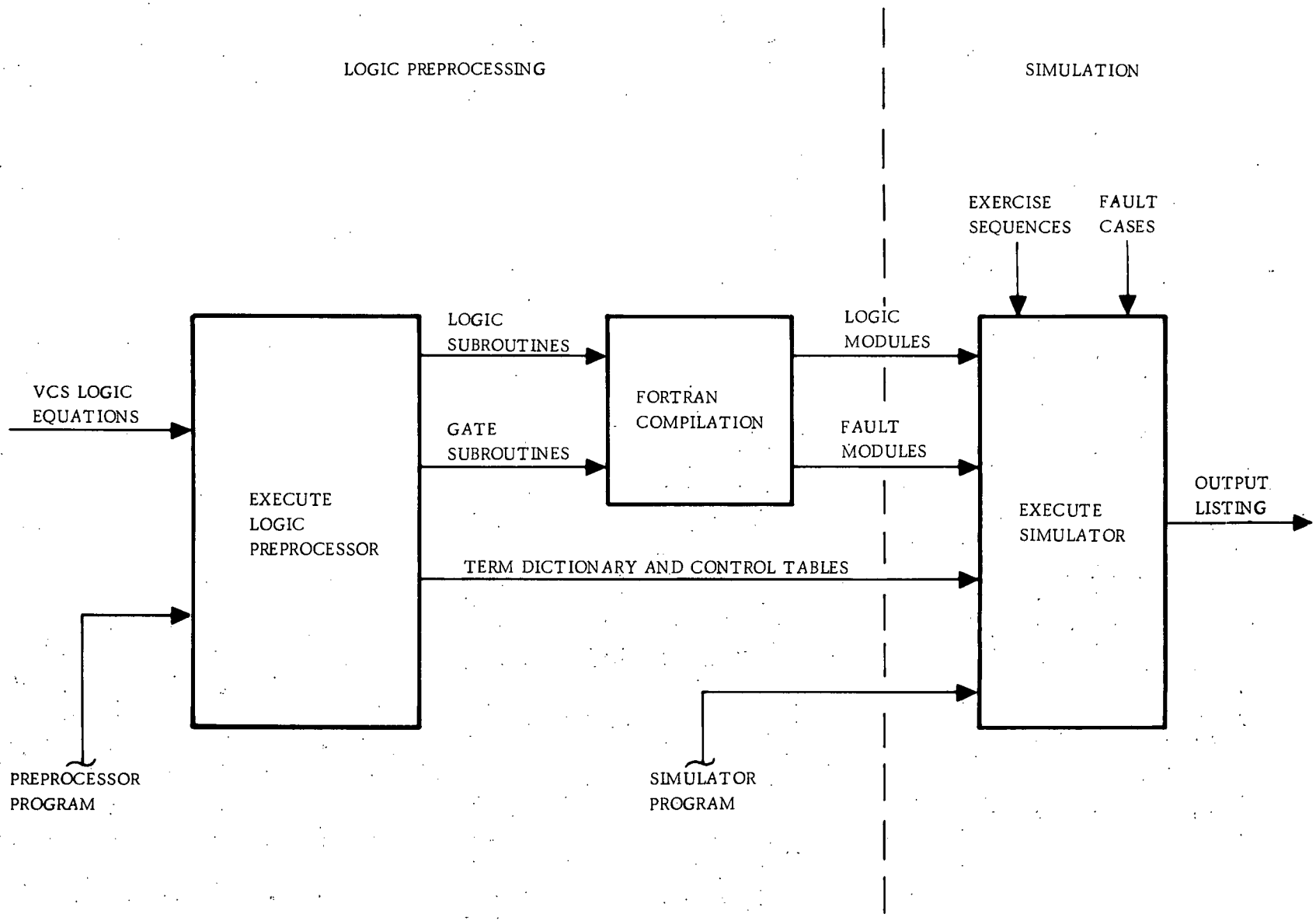


Figure 3-2. Simulation System

2. Gate Equation Subroutines - This is an equivalent representation of the logic except that the equations have been expanded and intermediate terms defined such that each logic gate is represented by a separate logic equation. These subroutines are also in the Fortran language and are subsequently compiled to produce executable program modules (called fault modules) which will be used to simulate logic gate failures during the subsequent simulations. Note that there is a one-for-one correspondence between logic equation modules and fault modules.

Additionally, the Preprocessor Program generates a dictionary of the logic terms and various control tables which are required by the Simulator Program.

The Simulator Program consists primarily of three program modules: (1) Control Program, (2) Fault Generator, and (3) Interface Simulator.

The Control Program performs the executive functions in the simulation.

The Fault Generator controls the simulation of logic failures. This is accomplished by dynamically replacing a Logic Equation module with the corresponding fault module. Any gate within that module can then be forced to a steady 1-state or 0-state. Selection of gates to be failed, the type of failure, and the sequence of failures is determined by fault control cards which are prepared manually and supplied as input during the simulation.

The Interface Simulator dynamically controls/monitors the interface terms to the VCS logic during the simulation. This interface is described in detail in Section 2.3.

The four input channels to the VCS are controlled by a computer I/O simulation which operates from prestored Exercise Sequences which are prepared manually. The Exercise Sequences consist of combinations of the following operations:

1. Output to LP
2. Input from LP
3. Reset S Matrix
4. Set P & R matrices
5. Sample P & R modes
6. Sample all matrices
7. Sample S matrix
8. Compute Status

The first seven operations correspond directly to VCS commands. The "compute status" operation consists of examining the results of previous VCS operations

to determine if any VCS errors were detected.

The Local Processor (LP) interface is simulated by monitoring the data bus output from the VCS logic, interpreting the commands, and verifying the data transmitted. In this manner, an erroneous data transmission from the VCS can be detected.

At the completion of an exercise sequence, one line of print is generated which identifies the fault (if any) which was simulated and summarizes the status computed by the computer I/O simulation and the LP interface simulation. A given exercise sequence can be repeated as many times as desired while different logic faults are simulated.

3.2 EVALUATION PROGRAM

3.2.1 Description

The purpose of the VCS design evaluation was twofold: (1) debug the logic design and verify its conformance to the functional performance requirements, and (2) investigate the failure modes of the VCS design. Of particular concern in the failure mode investigation was identification and elimination of failures which exhibit either of the following characteristics:

(1) Allow erroneous data to be transmitted on the data bus with a "Go" code.

(2) Induce a failure (real or apparent) in some other element of the system.

Failures which exhibit either of these characteristics potentially represent "single point" failures in the system. The term single point failure generally refers to a malfunction or failure mode in a given unit which renders that unit nonoperable or which removes other subsystems (or the total system) from operation. The first type is the more common use of the term single point failure. This type is of interest where some form of redundancy is being employed internal to a given unit in order to allow for successful operation of that unit after some number of internal malfunctions. It is generally very difficult, if not impossible, to design a subsystem of any functional complexity which does not contain some failure modes of this type.

As previously indicated, internal redundancy is not being used in the VCS design. Rather, multiple VCSes would be employed in a system to allow the total system to tolerate VCS malfunctions. Therefore, the second type of single point failure is of concern in the VCS design; i.e., the design should preclude failure modes which result in functional loss of any subsystem other than the VCS which suffered the malfunction.

The Simulation System described in Section 3.1 was the primary tool used in the design evaluation. As previously indicated, the VCS design has been developed in the form of logic equations. These equations were prepared and maintained in the form of a punched card deck. A listing of these equations in their final form is included in Appendix A of this report.

The first phase of the evaluation procedure consisted of executing the simulation system using exercise sequences which exercise all the VCS control byte commands, modes, etc., to verify that the VCS design operates as expected in a no-fault situation.

During simulation, the status of the VCS operation is recorded by the Interface Simulator module. This status represents data that would be observed by computer and LP hardware/software in an actual VCS system application and is the data that would be used to detect and diagnose VCS malfunctions. The primary status information is summarized in Figure 3-3. The print out of the data at the conclusion of an Exercise sequence simulation is used to analyze the results of a given run. Note that four sets of computer interface status are recorded corresponding to the information available to each of the four computers in the system.

The failure mode investigation was conducted using the simulator's capability to induce faults into the VCS logic during simulation. The type of failures which were simulated are single logic element (gate) malfunctions of two types; the gate fails to a constant true state or to a constant false state. The VCS design contains approximately 2600 logic gates. Approximately 1400 gates are used in the four input channels (350 gates/channel). Since these are identical it is sufficient to simulate faults in only one channel. Therefore, approximately 1550 gates or 3100 specific fault cases were of interest in the failure mode investigation.

The failure mode investigation consisted of executing various Exercise sequences while inducing the simulated faults. The entire Exercise sequence must be executed for each individual fault. Due to the amout of time required to execute the simulation, the Exercise sequences had to be kept relatively short and therefore each Exercise sequence would exercise only a portion of the logic. For a given sequence, only faults in that portion of the logic being exercised were simulated. Once a particular fault case was "eliminated" it was not simulated with subsequent Exercise sequences. "Eliminating" a fault required that: (1) it be detected by the Interface Simulator as a VCS failure (equivalent to detection by computer system software in an actual VCS system application), and (2) no erroneous data transmissions went undetected (erroneous data transmitted with a "Go" code, as determined by the LP Interface Simulator).

Fault cases which did not cause any detectable change in VCS performance during a given Exercise sequence were retained in the list of faults to be simulated with subsequent exercise sequences since this situation meant that either the logic gate was redundant, i.e., unnecessary in the design, or the particular Exercise sequence did not stimulate the precise conditions necessary to exercise the gate. The latter situation is by far the more common.

Fault cases which caused either an undetected erroneous data transmission or any degradation in performance without detection as a VCS failure were analyzed individually to determine how to correct this situation. In this manner, a refined VCS design was evolved together with an assurance of acceptable operation over a range of possible malfunctions.

	FLAG ID	CONDITION FOR SETTING FLAG
Computer Status	1	P-matrix diagonal or R-matrix mode incorrect
	3	S-matrix (voting status) non-zero
	5	"Next Byte" signal fails to come true
	6	No feedback on data bus
	7	Parity error(s) during input transmission
	8	Unsuccessful byte transmission due to parity errors
	9	Go/No Go word or input request word transmitted incorrectly
	10	Error in transmission of LP acknowledge word
	11	Parity error on matrix sample
	12	Unsuccessful output transmission (parity error or No Go word transmitted)
Local Processor Status	I1	Bus transmission interrupted (loss of bus timing)
	I2	Parity error on bus data
	I3	Bus busy when LP transmission desired
	N1	No Go word received
	B1	Output message incorrect but Go word received
	L1	Sequence of bus messages incorrect

Figure 3 - 3. COMPUTER/LP STATUS FLAGS

3.2.2 Summary of Results

The first phase of the evaluation, debugging the logic design, proceeded in a manner common to most design verification processes. A considerable number of problems were uncovered and corrected but no changes of major significance were required to the basic design concept/approach.

Some observations concerning the computer/VCS interface were made during the simulation activity and are worthy of note. As described in Section 2.3, the VCS interface was designed to be generally adaptable to a wide range of computers. The generality is necessary at this stage of the VCS system design since optimization for a specific computer system would be premature. However, the timing relationships necessary at this interface are rather critical due to the need to maintain synchronization between computers when using the VCS in voting modes.

In particular, the timing during matrix operations in the VCS is variable depending on the amount of activity in the other input channels. The computers must account for this variable timing if subsequent voting operations are to occur prior to a resynchronization of the computers. One way to operate the system and avoid this problem is to perform all input/output voting operations at the beginning of a time slot (some sort of interval timing is presumed to be used as a primary synchronization source) followed by any desired matrix operations. In this way computer synchronization need not be maintained during the matrix operations. Moreover, the specific design of the computer I/O interface will also affect the efficiency of the operation of the total computer/VCS system. It is therefore suggested that in future application of the VCS device, care be taken in adapting the VCS interface to the specific characteristics of the computer I/O being used.

At the time of this writing, the failure mode investigation is still in progress. The results to date substantiate the performance goals of the VCS design. Some 1500 failure cases have been simulated and as yet no failure modes of the type previously described have been identified. Logic design changes were indeed made during the investigation but these were not required to overcome undesirable failure modes but were simply a fallout of the additional simulation that was performed.

The results are not particularly startling or unexpected. The key to the success of the VCS design in terms of failure management is the simplicity of the failure detection process. Observing the results of the system operation at the "end" of the chain, i.e., the data bus, provides very conclusive evidence as to the operability of the system and, in particular, the VCS. Use of a majority controlled validation code (the Go/No-Go word) to confirm/cancel each message on the data bus appears to provide a foolproof means of "controlling" VCS failures since the VCS is, by design, incapable of internally generating the Go code. (One can of course, always postulate some intricate sequence of failures and conditions which could cause a precise duplication of the serial bit pattern of the code, but from a practical standpoint such as exercise is rather meaningless).

Moreover, using multiple copies of the VCS in the system provides the most straightforward means of isolating VCS failures; i.e., comparison of the results of performing identical operations with two units.

Therefore, the overall conclusion from the results of the failure mode investigation to date is that the proposed VCS design is indeed free of the failure modes of the type described. This conclusion may seem optimistic or premature but from observation of the system performance during the investigation to date the writer feels confident that the conclusion is valid.

Table 3-1 summarizes the fault simulation results.

Table 3-1. Fault Simulation Summary

	No. of Logic Gates	No. of Fault Cases Run	No. of Logic Gate Faults "Eliminated"	% of Faults "Eliminated"
Input Channel	337	425	305	45.3%
Matrix Section	603	234 (700)*	230	19%
Voter Section	420	(800)*	--	--
Output Channel	243	(280)*	--	--
TOTALS	1,603	659 (1780)*	535	16.7%

( )* The results of these runs were not available at the time of this printing.

SECTION 4

RECOMMENDATIONS FOR FUTURE EFFORT

The VCS design optimized for minimum parts count can be mechanized with six different MOS/LSI circuit types. Of these six circuits, only one device, an eight bit multiplexer circuit, is presently available as an off-the-shelf standard MOS/LSI circuit. The remaining five circuits are all unique to VCS and would have to be developed. This represents a substantial investment in device development alone.

An alternate design approach which could reduce the development cost and still result in relatively efficient use of MOS/LSI circuits is to mechanize the control logic of the VCS with a microprogrammed control unit. Microprogramming is a feature which allows the control function of the VCS to be mechanized with a read only memory (ROM). High density ROMs are readily available from a number of LSI manufacturers. The process of fabricating a ROM device with the desired bit pattern is simple and relatively inexpensive. Although this design approach may not result in an optimum design from the standpoint of component count, it appears to offer two significant advantages: (1) reduced development cost, and (2) flexibility to adapt the VCS to different system requirements. Therefore, it is recommended that the microprogrammed VCS design be initiated and carried to a point where the two approaches can be traded off.

SECTION 5

REFERENCES

1. Reconfigurable G&C Computer Study for Space Station Use, Final Report, Autonetics Division of North American Rockwell Corp., C70-171/301. January 31, 1971

2. Voter/Comparator/Switch Optimization Study, Study Program Plan, Autonetics Division of North American Rockwell Corp., C71-484/401, May 17, 1971

3. Voter/Comparator/Switch Optimization Study, Monthly Progress Report No. 5, Autonetics Division of North American Rockwell Corp., C71-512.5/401

4. The Integrated Circuits Catalog for Design Engineering, Texas Instruments Electronics Group Engineering Staff, 1971

5. Total MOS Capability, Motorola Inc., Semiconductor Products Division, 1971

6. MOS Technology Notes, North American Rockwell Microelectronics Co., Training Department, 1971

SECTION 6

DEFINITIONS

CPU	-	Central Processor Unit
FOOS	-	Fail-operational-fail-operation-fail-safe
IOP	-	Input Output Processor
LP	-	Local Processor
ROM	-	Read Only Memory
VCS	-	Voter-Comparator Switch

APPENDIX A

VCS LOGIC EQUATIONS

*** VCS LOGIC EQUATIONS ***

CLOCK AND POWER ON

PWRON =

INCLKA=

*** VCS LOGIC EQUATIONS ***

COMPUTER A INPUTS

IN1A =
IN2A =
IN3A =
IN4A =
IN5A =
IN6A =
IN7A =
IN8A =
IN9A =
DATSTA=
COMNDA=
ENDMSA=

*** VCS LOGIC EQUATIONS ***

COMPUTER B INPUTS

IN1B =
IN2B =
IN3B =
IN4B =
IN5B =
IN6B =
IN7B =
IN8B =
IN9B =
DATSTB=
COMNOB=
ENDMSB=

*** VCS LOGIC EQUATIONS ***

COMPUTER C INPUTS

IN1C =
IN2C =
IN3C =
IN4C =
IN5C =
IN6C =
IN7C =
IN8C =
IN9C =
DATSTC=
COMNDC=
ENDMSC=

*** VCS LOGIC EQUATIONS ***

COMPUTER D INPUTS

IN1D =
IN2D =
IN3D =
IN4D =
IN5D =
IN6D =
IN7D =
IN8D =
IN9D =
DATSTD=
COMNDD=
ENDMSD=

*** VCS LOGIC EQUATIONS ***

LOCAL PROCESSOR INPUTS

PRES =

LPIN =

*** VCS LOGIC EQUATIONS ***

INPUT CHANNEL A

```
 EOM    =ENDMSA.-(-MCIC3.-MCIC4.MCIC5.(-MCIC2.-MCIC6+-MCIC1.MCIC2.MCIC6)
         +-MCIC6.MCIC4.-MCIC3.(-MCIC5.-MCIC2+MCIC5.MCIC2.-MCIC1))
 PARBA  =PAR1.-PAR2+-PAR1.PAR2
1MCIC1 =-PWRON.(-MCIC2.-MCIC6.DATSTA
                                        .(-MCIC5+-MCIC3.-MCIC4)+MCIC2.MCIC5.D
         ATSTA
             .(MCIC4.-MCIC6+MCIC3.-MCIC6+-MCIC3.-MCIC4.MCIC6)+MCIC2.-MCIC
         3.-MCIC4.-MCIC5.-MCIC6.ADDVAL.(PAR1.PAR2+-PAR1.-PAR2)+-MCIC2. MC
         IC3.-MCIC4.MCIC5.-MCIC6.LDBRB+-MCIC2.MCIC3.-MCIC4.-MCIC5.MCIC6.-
         BRA3.(-BRA2+-BRA1)+-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6.INSRVA+MCIC2
         .-MCIC4.-MCIC5.                MCIC3.MCIC6
         +MCIC2.-MCIC6.(MCIC4.-MCIC5
         +MCIC3.-MCIC5+-MCIC3.-MCIC4.MCIC5).-PARBA)
OMCIC1 =-MCIC4.-MCIC6.(-MCIC2+-MCIC5+MCIC2.-MCIC3.MCIC5.-END.LDBRB)
         +-MCIC2.MCIC4.-MCIC5.(-MCIC3+-MCIC6)
         +MCIC2.MCIC4.-MCIC5.-MCIC6.(MCIC3.LDBRB+-MCIC3.(LDBRB+END))
                                              +MCIC3.-MCIC4.MCIC6.DATSTA
         .(-MCI
         C2.-MCIC5+MCIC2.MCIC5)+MCIC2.MCIC3.-MCIC4.-MCIC5.MCIC6.(PAR1.PAR
         2+-PAR1.-PAR2)+EOM+PWRON
         +MCIC2.-MCIC3.-MCIC4.MCIC5.-MCIC6.END
1MCIC2 =-PWRON.(MCIC1.-MCIC6.(-MCIC5+-MCIC3.-MCIC4)+MCIC1.-MCIC3.MCIC4.-
         MCIC5+MCIC1.MCIC3.-MCIC4.-MCIC5.MCIC6.DATSTA
         +-MCIC1.MCIC3.-MCIC4.-MCIC5.MCIC6.(BRA3+BRA1.BRA2))
OMCIC2 =MCIC1.-MCIC4.-MCIC5.(-MCIC6+MCIC3)+-MCIC1.-MCIC3.-MCIC5.MCIC6
         .(-MCIC4+INSRVA)+-M
         CIC1.MCIC5.(MCIC3.-MCIC6+MCIC4.-MCIC6+-MCIC3.-MCIC4.MCIC6).DATST
         A+LDBRB.-END.MCIC1.-MCIC3.-MCIC4.MCIC5.-MCIC6
         +MCIC1.MCIC4.-MCIC5.-MCIC6.(MCIC3.LDBRB+-MCIC3.(LDBRB+END))
                                              +-MCIC1.-MCIC3.-MCIC4.-MCIC5.-MC
         IC6.-ADDVAL+PWRON+EOM
         +MCIC1.-MCIC3.-MCIC4.MCIC5.-MCIC6.END
1MCIC3 =-PWRON.(MCIC1.MCIC2.-MCIC6.(-MCIC4.(-MCIC5+MCIC5.(LDBRB+END))
         +MCIC4.-MCIC5.(LDBRB+END)))
OMCIC3 =MCIC1.-MCIC4.(MCIC2.-MCIC5+-MCIC2.MCIC5.-MCIC6)+MCIC1.MCIC2.MCIC
         4.-MCIC5.-MCIC6.LDBRB+PWRON+EOM
```

```
                    *** VCS LOGIC EQUATIONS ***
    INPUT CHANNEL A

         +-MCIC1.-MCIC2.-MCIC4.-MCIC5.MCIC6.(BRA3+BRA1.BRA2)
1MCIC4 =-PWRON.(MCIC1.MCIC2.MCIC3.-MCIC5.(-MCIC6+PAR1.PAR2+-PAR1.-PAR2)
         +-MCIC1.-MCIC2.MCIC3.-MCIC5.MCIC6. (BRA3+BRA1.BRA2)
         +LDBRB.-END.MCIC1.MCIC2.-MCIC3.MCIC5.-MCIC6)
OMCIC4 =MCIC2.-MCIC5.(MCIC1.MCIC3.-MCIC6.LDBRB+-MCIC1.-MCIC3.MCIC6
         .INSRVA+MCIC1.-MCIC3.-MCIC6.END)+EOM+PWRON
1MCIC5 =-PWRON.(MCIC1.MCIC2.MCIC3.MCIC4.-MCIC6.LDBRB
         +PARBA.MCIC2.(-MCIC1.-MCIC6.(MCIC3+MCIC4)+MCIC1.MCIC3.-MCIC4.
         MCIC6)+MCIC1.MCIC2.-MCIC3.MCIC4.-MCIC6.END)
OMCIC5 =-MCIC1.MCIC2.-MCIC6.DATSTA
                                    .(MCIC4+MCIC3)+MCIC1.MCIC3.-MCIC4.(-MCI
         C2.-MCIC6+MCIC2.MCIC6.DATSTA)+PWRON+EOM
         +LDBRB.-END.MCIC1.MCIC2.-MCIC3.-MCIC4.-MCIC6
1MCIC6 =-PWRON.(MCIC2.-MCIC3.(-MCIC1.PARBA+MCIC1.-MCIC4.-MCIC5.BRA4))
OMCIC6 =-MCIC1.-MCIC3.(-MCIC2.-MCIC4.-MCIC5+MCIC2.MCIC4.-MCIC5.INSRVA
         +MCIC2.-MCIC4.MCIC5.DATSTA)+PWRON+EOM
1NXBYTA=-MCIC4.-MCIC5.-MCIC6.MCIC2.(MCIC1.-MCIC3.-BRA4+-MCIC1.MCIC3.(PAR
         1.PAR2+-PAR1.-PAR2))+LDBRB.-END+-MCIC1.-MCIC2.MCIC3.-MCIC4.-MCIC
         5.MCIC6.-BRA3.(-BRA2+-BRA1)
ONXBYTA=DATSTA+PWRON
 INPR1 =IN1A.-IN2A+-IN1A.IN2A
 INPR2 =IN3A.-IN4A+-IN3A.IN4A
 INPR3 =IN5A.-IN6A+-IN5A.IN6A
 INPR4 =IN7A.-IN8A+-IN7A.IN8A
 INPR5 =INPR1.-INPR2+-INPR1.INPR2
 INPR6 =INPR3.-INPR4+-INPR3.INPR4
1PAR2  =DATSTA
0PAR2  =(INPR5.-INPR6+-INPR5.INPR6).(-MCIC1.MCIC2.MCIC3.-MCIC4.-MCIC5.MC
         IC6+-MCIC2.-MCIC6.MCIC1.(-MCIC5+-MCIC3.-MCIC4))
1BRA1  =IN1A.DATSTA
0BRA1  =DATSTA.-IN1A
1BRA2  =DATSTA.IN2A
0BRA2  =DATSTA.-IN2A
1BRA3  =DATSTA.IN3A
0BRA3  =DATSTA.-IN3A
1BRA4  =DATSTA.IN4A
```

*** VCS LOGIC EQUATIONS ***

INPUT CHANNEL A

```
OBRA4  =DATSTA.-IN4A
1BRA5  =DATSTA.IN5A
OBRA5  =DATSTA.-IN5A
1BRA6  =DATSTA.IN6A
OBRA6  =DATSTA.-IN6A
1BRA7  =DATSTA.IN7A
OBRA7  =DATSTA.-IN7A
1BRA8  =DATSTA.IN8A
OBRA8  =DATSTA.-IN8A
 BOTH  =COMPA.BRBFA.BRBFB+COMPB.BRBFA.BRBFC+COMPC.BRBFA.BRBFD
        +COMPD.BRBFB.BRBFC+COMPE.BRBFB.BRBFD+COMPF.BRBFC.BRBFD
 SHFTA =OM1.(CYCLE.-MCVS2.MCVS4.(MCVS1.MCVS3.MCVS5+-MCVS1.-MCVS3.-MCVS5)
        +-DONE.MCVS4.(MCVS1.-MCVS2.-MCVS3.-MCVS5+-MCVS1.MCVS2.MCVS3
        .MCVS5)+MCVS3.-MCVS4.-MCVS5.(-MCVS1.MCVS2+MCVS1.-MCVS2
        +-MCVS1.-MCVS2.CDNMET)+MCVS4.MCVS5.(MCVS2.-MCVS3+MCVS1.-MCVS2
        .-MCVS3.BRBFA)+-MCVS1.-MCVS2.MCVS3.-MCVS4.MCVS5.BOTH)
 COBRB =LDBRB+-MCIC1.-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6.INSRVA+MCIC2.-MCIC
        4.-MCIC5.-MCIC6.(-MCIC1.MCIC3.(PAR1.PAR2+-PAR1.-PAR2)+MCIC1.-MCI
        C3.BRA4)
1BRB1A =COBRB.BRA1
OBRB1A =COBRB.-BRA1+SHFTA
1BRB2A =COBRB.BRA2+SHFTA. BRB1A
OBRB2A =COBRB.-BRA2+SHFTA.-BRB1A
1BRB3A =COBRB.BRA3+SHFTA. BRB2A
OBRB3A =COBRB.-BRA3+SHFTA.-BRB2A
1BRB4A =COBRB.BRA4+SHFTA. BRB3A
OBRB4A =COBRB.-BRA4+SHFTA.-BRB3A
1BRB5A =COBRB.BRA5 +SHFTA. BRB4A
OBRB5A =COBRB.-BRA5+SHFTA.-BRB4A
1BRB6A =COBRB.BRA6 +SHFTA. BRB5A
OBRB6A =COBRB.-BRA6+SHFTA.-BRB5A
1BRB7A =COBRB.BRA7 +SHFTA. BRB6A
OBRB7A =COBRB.-BRA7+SHFTA.-BRB6A
1BRB8A =COBRB.BRA8 +SHFTA. BRB7A
OBRB8A =COBRB.-BRA8+SHFTA.-BRB7A
1PAR4A =MCIC1.MCIC2.-MCIC3.-MCIC6.(-MCIC4.MCIC5+MCIC4.-MCIC5).(PAR1.PAR3
```

*** VCS LOGIC EQUATIONS ***

INPUT CHANNEL A

```
        +-PAR1.-PAR3)
0PAR4A =MCIC1.MCIC2.-MCIC3.-MCIC6.(-MCIC4.MCIC5+MCIC4.-MCIC5).(PAR1.-PAR
        3+-PAR1.PAR3)
1PRBADA=(PAR1.-PAR2+-PAR1.PAR2).(-MCIC1.MCIC2.-MCIC6.(-MCIC5+-MCIC3.-MCI
        C4)+MCIC1.-MCIC2.-MCIC3.MCIC4.MCIC5.MCIC6)
0PRBADA=DATSTA
1BRBFA =-MCIC1.MCIC2.MCIC3.-MCIC4.-MCIC5.-MCIC6.(PAR1.PAR2+-PAR1.-PAR2)
0BRBFA =-MCIC1       .-MCIC3.-MCIC4.-MCIC5.-MCIC6+PWRON
1ADDVAL=MCIC1.-MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6.IN5A
0ADDVAL=-MCIC1.-MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6+PWRON
1PAR1  =DATSTA.IN9A
0PAR1  =DATSTA.-IN9A
1PAR3  =-MCIC1.MCIC2.MCIC3.-MCIC5.-MCIC6.PAR1
0PAR3  =-MCIC1.MCIC2.MCIC3.-MCIC5.-MCIC6.-PAR1
1END   =ENDMSA
0END   =DONEA+PWRON
1DONEA =END.LDBRB.-MCIC1.-MCIC2.MCIC3.-MCIC4.MCIC5.-MCIC6+PWRON
0DONEA =MCIC1.MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6.-BRA4
1MIA   =MCIC3.-MCIC4.-MCIC5.MCIC6.(-MCIC1.-MCIC2.(BRA3+BRA1.BRA2)+MCIC1.
        MCIC2.(PAR1.PAR2+-PAR1.-PAR2))
        +MCIC1.-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6
0MIA   =INSRVA+PWRON
```

*** VCS LOGIC EQUATIONS ***

INPUT CHANNEL B

```
 EOM    =ENDMSB.-(-MCIC3.-MCIC4.MCIC5.(-MCIC2.-MCIC6+-MCIC1.MCIC2.MCIC6)
         +-MCIC6.MCIC4.-MCIC3.(-MCIC5.-MCIC2+MCIC5.MCIC2.-MCIC1))
 PARBA  =PAR1.-PAR2+-PAR1.PAR2
1MCIC1  =-PWRON.(-MCIC2.-MCIC6.DATSTB
                                        .(-MCIC5+-MCIC3.-MCIC4)+MCIC2.MCIC5.D
         ATSTB
              .(MCIC4.-MCIC6+MCIC3.-MCIC6+-MCIC3.-MCIC4.MCIC6)+MCIC2.-MCIC
         3.-MCIC4.-MCIC5.-MCIC6.ADDVAL.(PAR1.PAR2+-PAR1.-PAR2)+-MCIC2. MC
         IC3.-MCIC4.MCIC5.-MCIC6.LDBRB+-MCIC2.MCIC3.-MCIC4.-MCIC5.MCIC6.-
         BRA3.(-BRA2+-BRA1)+-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6.INSRVB+MCIC2
         .-MCIC4.-MCIC5.                  MCIC3.MCIC6
         +MCIC2.-MCIC6.(MCIC4.-MCIC5
         +MCIC3.-MCIC5+-MCIC3.-MCIC4.MCIC5).-PARBA)
0MCIC1  =-MCIC4.-MCIC6.(-MCIC2+-MCIC5+MCIC2.-MCIC3.MCIC5.-END.LDBRB)
         +-MCIC2.MCIC4.-MCIC5.(-MCIC3+-MCIC6)
         +MCIC2.MCIC4.-MCIC5.-MCIC6.(MCIC3.LDBRB+-MCIC3.(LDBRB+END))
                                              +MCIC3.-MCIC4.MCIC6.DATSTB
         .(-MCI
         C2.-MCIC5+MCIC2.MCIC5)+MCIC2.MCIC3.-MCIC4.-MCIC5.MCIC6.(PAR1.PAR
         2+-PAR1.-PAR2)+EOM+PWRON
         +MCIC2.-MCIC3.-MCIC4.MCIC5.-MCIC6.END
1MCIC2  =-PWRON.(MCIC1.-MCIC6.(-MCIC5+-MCIC3.-MCIC4)+MCIC1.-MCIC3.MCIC4.-
         MCIC5+MCIC1.MCIC3.-MCIC4.-MCIC5.MCIC6.DATSTA
         +-MCIC1.MCIC3.-MCIC4.-MCIC5.MCIC6.(BRA3+BRA1.BRA2))
0MCIC2  =MCIC1.-MCIC4.-MCIC5.(-MCIC6+MCIC3)+-MCIC1.-MCIC3.-MCIC5.MCIC6
         .(-MCIC4+INSRVB)+-M
         CIC1.MCIC5.(MCIC3.-MCIC6+MCIC4.-MCIC6+-MCIC3.-MCIC4.MCIC6).DATST
         B+LDBRB.-END.MCIC1.-MCIC3.-MCIC4.MCIC5.-MCIC6
         +MCIC1.MCIC4.-MCIC5.-MCIC6.(MCIC3.LDBRB+-MCIC3.(LDBRB+END))
                                                +-MCIC1.-MCIC3.-MCIC4.-MCIC5.-MC
         IC6.-ADDVAL+PWRON+EOM
         +MCIC1.-MCIC3.-MCIC4.MCIC5.-MCIC6.END
1MCIC3  =-PWRON.(MCIC1.MCIC2.-MCIC6.(-MCIC4.(-MCIC5+MCIC5.(LDBRB+END))
         +MCIC4.-MCIC5.(LDBRB+END)))
0MCIC3  =MCIC1.-MCIC4.(MCIC2.-MCIC5+-MCIC2.MCIC5.-MCIC6)+MCIC1.MCIC2.MCIC
         4.-MCIC5.-MCIC6.LDBRB+PWRON+EOM
```

INPUT CHANNEL B

```
        +-MCIC1.-MCIC2.-MCIC4.-MCIC5.MCIC6.(BRA3+BRA1.BRA2)
1MCIC4 =-PWRON.(MCIC1.MCIC2.MCIC3.-MCIC5.(-MCIC6+PAR1.PAR2+-PAR1.-PAR2)
        +-MCIC1.-MCIC2.MCIC3.-MCIC5.MCIC6. (BRA3+BRA1.BRA2)
        +LDBRB.-END.MCIC1.MCIC2.-MCIC3.MCIC5.-MCIC6)
OMCIC4 =MCIC2.-MCIC5.(MCIC1.MCIC3.-MCIC6.LDBRB+-MCIC1.-MCIC3.MCIC6
        .INSRVB+MCIC1.-MCIC3.-MCIC6.END)+EOM+PWRON
1MCIC5 =-PWRON.(MCIC1.MCIC2.MCIC3.MCIC4.-MCIC6.LDBRB
        +PARBA.MCIC2.(-MCIC1.-MCIC6.(MCIC3+MCIC4)+MCIC1.MCIC3.-MCIC4.
        MCIC6)+MCIC1.MCIC2.-MCIC3.MCIC4.-MCIC6.END)
OMCIC5 =-MCIC1.MCIC2.-MCIC6.DATSTB
                                .(MCIC4+MCIC3)+MCIC1.MCIC3.-MCIC4.(-MCI
        C2.-MCIC6+MCIC2.MCIC6.DATSTA)+PWRON+EOM
        +LDBRB.-END.MCIC1.MCIC2.-MCIC3.-MCIC4.-MCIC6
1MCIC6 =-PWRON.(MCIC2.-MCIC3.(-MCIC1.PARBA+MCIC1.-MCIC4.-MCIC5.BRA4))
OMCIC6 =-MCIC1.-MCIC3.(-MCIC2.-MCIC4.-MCIC5+MCIC2.MCIC4.-MCIC5.INSRVB
        +MCIC2.-MCIC4.MCIC5.DATSTB)+PWRON+EOM
1NXBYTB=-MCIC4.-MCIC5.-MCIC6.MCIC2.(MCIC1.-MCIC3.-BRA4+-MCIC1.MCIC3.(PAR
        1.PAR2+-PAR1.-PAR2))+LDBRB.-END+-MCIC1.-MCIC2.MCIC3.-MCIC4.-MCIC
        5.MCIC6.-BRA3.(-BRA2+-BRA1)
ONXBYTB=DATSTB+PWRON
 INPR1  =IN1B.-IN2B+-IN1B.IN2B
 INPR2  =IN3B.-IN4B+-IN3B.IN4B
 INPR3  =IN5B.-IN6B+-IN5B.IN6B
 INPR4  =IN7B.-IN8B+-IN7B.IN8B
 INPR5  =INPR1.-INPR2+-INPR1.INPR2
 INPR6  =INPR3.-INPR4+-INPR3.INPR4
1PAR2   =DATSTB
OPAR2   =(INPR5.-INPR6+-INPR5.INPR6).(-MCIC1.MCIC2.MCIC3.-MCIC4.-MCIC5.MC
        IC6+-MCIC2.-MCIC6.MCIC1.(-MCIC5+-MCIC3.-MCIC4))
1BRA1   =IN1B.DATSTB
OBRA1   =DATSTB.-IN1B
1BRA2   =DATSTB.IN2B
OBRA2   =DATSTB.-IN2B
1BRA3   =DATSTB.IN3B
OBRA3   =DATSTB.-IN3B
1BRA4   =DATSTB.IN4B
```

*** VCS LOGIC EQUATIONS ***

INPUT CHANNEL B

```
0BRA4   =DATSTB.-IN4B
1BRA5   =DATSTB.IN5B
0BRA5   =DATSTB.-IN5B
1BRA6   =DATSTB.IN6B
0BRA6   =DATSTB.-IN6B
1BRA7   =DATSTB.IN7B
0BRA7   =DATSTB.-IN7B
1BRA8   =DATSTB.IN8B
0BRA8   =DATSTB.-IN8B
 SHFTB  =OM2.(CYCLE.-MCVS2.MCVS4.(MCVS1.MCVS3.MCVS5+-MCVS1.-MCVS3.-MCVS5)
         +-DONE.MCVS4.(MCVS1.-MCVS2.-MCVS3.-MCVS5+-MCVS1.MCVS2.MCVS3
         .MCVS5)+MCVS3.-MCVS4.-MCVS5.(-MCVS1.MCVS2+MCVS1.-MCVS2
         +-MCVS1.-MCVS2.CONMET)+MCVS4.MCVS5.(MCVS2.-MCVS3+MCVS1.-MCVS2
         .-MCVS3.BRBFB)+-MCVS1.-MCVS2.MCVS3.-MCVS4.MCVS5.BOTH)
 COBRB  =LDBRB+-MCIC1.-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6.INSRVB+MCIC2.-MCIC
         4.-MCIC5.-MCIC6.(-MCIC1.MCIC3.(PAR1.PAR2+-PAR1.-PAR2)+MCIC1.-MCI
         C3.BRA4)
1BRB1B  =COBRB.BRA1
0BRB1B  =COBRB.-BRA1+SHFTB
1BRB2B  =COBRB.BRA2+SHFTB. BRB1B
0BRB2B  =COBRB.-BRA2+SHFTB.-BRB1B
1BRB3B  =COBRB.BRA3+SHFTB. BRB2B
0BRB3B  =COBRB.-BRA3+SHFTB.-BRB2B
1BRB4B  =COBRB.BRA4+SHFTB. BRB3B
0BRB4B  =COBRB.-BRA4+SHFTB.-BRB3B
1BRB5B  =COBRB.BRA5 +SHFTB. BRB4B
0BRB5B  =COBRB.-BRA5+SHFTB.-BRB4B
1BRB6B  =COBRB.BRA6 +SHFTB. BRB5B
0BRB6B  =COBRB.-BRA6+SHFTB.-BRB5B
1BRB7B  =COBRB.BRA7 +SHFTB. BRB6B
0BRB7B  =COBRB.-BRA7+SHFTB.-BRB6B
1BRB8B  =COBRB.BRA8 +SHFTB. BRB7B
0BRB8B  =COBRB.-BRA8+SHFTB.-BRB7B
1PAR4B  =MCIC1.MCIC2.-MCIC3.-MCIC6.(-MCIC4.MCIC5+MCIC4.-MCIC5).(PAR1.PAR3
         +-PAR1.-PAR3)
0PAR4B  =MCIC1.MCIC2.-MCIC3.-MCIC6.(-MCIC4.MCIC5+MCIC4.-MCIC5).(PAR1.-PAR
```

```
    INPUT CHANNEL B

          3+-PAR1.PAR3)
1PRBADB=(PAR1.-PAR2+-PAR1.PAR2).(-MCIC1.MCIC2.-MCIC6.(-MCIC5+-MCIC3.-MCI
          C4)+MCIC1.-MCIC2.-MCIC3.MCIC4.MCIC5.MCIC6)
0PRBADB=DATSTB
1BRBFB =-MCIC1.MCIC2.MCIC3.-MCIC4.-MCIC5.-MCIC6.(PAR1.PAR2+-PAR1.-PAR2)
0BRBFB =-MCIC1      .-MCIC3.-MCIC4.-MCIC5.-MCIC6+PWRON
1ADDVAL=MCIC1.-MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6.IN5B
0ADDVAL=-MCIC1.-MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6+PWRON
1PAR1  =DATSTB.IN9B
0PAR1  =DATSTB.-IN9B
1PAR3  =-MCIC1.MCIC2.MCIC3.-MCIC5.-MCIC6.PAR1
0PAR3  =-MCIC1.MCIC2.MCIC3.-MCIC5.-MCIC6.-PAR1
1END   =ENDMSB
0END   =DONEB+PWRON
1DONEB =END.LDBRB.-MCIC1.-MCIC2.MCIC3.-MCIC4.MCIC5.-MCIC6+PWRON
0DONEB =MCIC1.MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6.-BRA4
1MIB   =MCIC3.-MCIC4.-MCIC5.MCIC6.(-MCIC1.-MCIC2.(BRA3+BRA1.BRA2)+MCIC1.
         MCIC2.(PAR1.PAR2+-PAR1.-PAR2))
         +MCIC1.-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6
0MIB   =INSRVB+PWRON
```

*** VCS LOGIC EQUATIONS ***

INPUT CHANNEL C

```
 EOM    =ENDMSC.-(-MCIC3.-MCIC4.MCIC5.(-MCIC2.-MCIC6+-MCIC1.MCIC2.MCIC6)
         +-MCIC6.MCIC4.-MCIC3.(-MCIC5.-MCIC2+MCIC5.MCIC2.-MCIC1))
 PARBA =PAR1.-PAR2+-PAR1.PAR2
1MCIC1 =-PWRON.(-MCIC2.-MCIC6.DATSTC
                                    .(-MCIC5+-MCIC3.-MCIC4)+MCIC2.MCIC5.D
         ATSTC
             .(MCIC4.-MCIC6+MCIC3.-MCIC6+-MCIC3.-MCIC4.MCIC6)+MCIC2.-MCIC
         3.-MCIC4.-MCIC5.-MCIC6.ADDVAL.(PAR1.PAR2+-PAR1.-PAR2)+-MCIC2. MC
         IC3.-MCIC4.MCIC5.-MCIC6.LDBRB+-MCIC2.MCIC3.-MCIC4.-MCIC5.MCIC6.-
         BRA3.(-BRA2+-BRA1)+-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6.INSRVC+MCIC2
         .-MCIC4.-MCIC5.                MCIC3.MCIC6
         +MCIC2.-MCIC6.(MCIC4.-MCIC5
         +MCIC3.-MCIC5+-MCIC3.-MCIC4.MCIC5).-PARBA)
0MCIC1 =-MCIC4.-MCIC6.(-MCIC2+-MCIC5+MCIC2.-MCIC3.MCIC5.-END.LDBRB)
         +-MCIC2.MCIC4.-MCIC5.(-MCIC3+-MCIC6)
         +MCIC2.MCIC4.-MCIC5.-MCIC6.(MCIC3.LDBRB+-MCIC3.(LDBRB+END))
                                          +MCIC3.-MCIC4.MCIC6.DATSTC
         .(-MCI
         C2.-MCIC5+MCIC2.MCIC5)+MCIC2.MCIC3.-MCIC4.-MCIC5.MCIC6.(PAR1.PAR
         2+-PAR1.-PAR2)+EOM+PWRON
         +MCIC2.-MCIC3.-MCIC4.MCIC5.-MCIC6.END
1MCIC2 =-PWRON.(MCIC1.-MCIC6.(-MCIC5+-MCIC3.-MCIC4)+MCIC1.-MCIC3.MCIC4.-
         MCIC5+MCIC1.MCIC3.-MCIC4.-MCIC5.MCIC6.DATSTA
         +-MCIC1.MCIC3.-MCIC4.-MCIC5.MCIC6.(BRA3+BRA1.BRA2))
0MCIC2 =MCIC1.-MCIC4.-MCIC5.(-MCIC6+MCIC3)+-MCIC1.-MCIC3.-MCIC5.MCIC6
         .(-MCIC4+INSRVC)+-M
         CIC1.MCIC5.(MCIC3.-MCIC6+MCIC4.-MCIC6+-MCIC3.-MCIC4.MCIC6).DATST
         C+LDBRB.-END.MCIC1.-MCIC3.-MCIC4.MCIC5.-MCIC6
         +MCIC1.MCIC4.-MCIC5.-MCIC6.(MCIC3.LDBRB+-MCIC3.(LDBRB+END))
                                          +-MCIC1.-MCIC3.-MCIC4.-MCIC5.-MC
         IC6.-ADDVAL+PWRON+EOM
         +MCIC1.-MCIC3.-MCIC4.MCIC5.-MCIC6.END
1MCIC3 =-PWRON.(MCIC1.MCIC2.-MCIC6.(-MCIC4.(-MCIC5+MCIC5.(LDBRB+END))
         +MCIC4.-MCIC5.(LDBRB+END)))
0MCIC3 =MCIC1.-MCIC4.(MCIC2.-MCIC5+-MCIC2.MCIC5.-MCIC6)+MCIC1.MCIC2.MCIC
         4.-MCIC5.-MCIC6.LDBRB+PWRON+EOM
```

*** VCS LOGIC EQUATIONS ***

INPUT CHANNEL C

```
        +-MCIC1.-MCIC2.-MCIC4.-MCIC5.MCIC6.(BRA3+BRA1.BRA2)
1MCIC4 =-PWRON.(MCIC1.MCIC2.MCIC3.-MCIC5.(-MCIC6+PAR1.PAR2+-PAR1.-PAR2)
        +-MCIC1.-MCIC2.MCIC3.-MCIC5.MCIC6. (BRA3+BRA1.BRA2)
        +LDBRB.-END.MCIC1.MCIC2.-MCIC3.MCIC5.-MCIC6)
0MCIC4 =MCIC2.-MCIC5.(MCIC1.MCIC3.-MCIC6.LDBRB+-MCIC1.-MCIC3.MCIC6
        .INSRVC+MCIC1.-MCIC3.-MCIC6.END)+EOM+PWRON
1MCIC5 =-PWRON.(MCIC1.MCIC2.MCIC3.MCIC4.-MCIC6.LDBRB
        +PARBA.MCIC2.(-MCIC1.-MCIC6.(MCIC3+MCIC4)+MCIC1.MCIC3.-MCIC4.
        MCIC6)+MCIC1.MCIC2.-MCIC3.MCIC4.-MCIC6.END)
0MCIC5 =-MCIC1.MCIC2.-MCIC6.DATSTC
                                    .(MCIC4+MCIC3)+MCIC1.MCIC3.-MCIC4.(-MCI
        C2.-MCIC6+MCIC2.MCIC6.DATSTA)+PWRON+EOM
        +LDBRB.-END.MCIC1.MCIC2.-MCIC3.-MCIC4.-MCIC6
1MCIC6 =-PWRON.(MCIC2.-MCIC3.(-MCIC1.PARBA+MCIC1.-MCIC4.-MCIC5.BRA4))
0MCIC6 =-MCIC1.-MCIC3.(-MCIC2.-MCIC4.-MCIC5+MCIC2.MCIC4.-MCIC5.INSRVC
        +MCIC2.-MCIC4.MCIC5.DATSTC)+PWRON+EOM
1NXBYTC=-MCIC4.-MCIC5.-MCIC6.MCIC2.(MCIC1.-MCIC3.-BRA4+-MCIC1.MCIC3.(PAR
        1.PAR2+-PAR1.-PAR2))+LDBRB.-END+-MCIC1.-MCIC2.MCIC3.-MCIC4.-MCIC
        5.MCIC6.-BRA3.(-BRA2+-BRA1)
0NXBYTC=DATSTC+PWRON
 INPR1 =IN1C.-IN2C+-IN1C.IN2C
 INPR2 =IN3C.-IN4C+-IN3C.IN4C
 INPR3 =IN5C.-IN6C+-IN5C.IN6C
 INPR4 =IN7C.-IN8C+-IN7C.IN8C
 INPR5 =INPR1.-INPR2+-INPR1.INPR2
 INPR6 =INPR3.-INPR4+-INPR3.INPR4
1PAR2  =DATSTC
0PAR2  =(INPR5.-INPR6+-INPR5.INPR6).(-MCIC1.MCIC2.MCIC3.-MCIC4.-MCIC5.MC
        IC6+-MCIC2.-MCIC6.MCIC1.(-MCIC5+-MCIC3.-MCIC4))
1BRA1  =IN1C.DATSTC
0BRA1  =DATSTC.-IN1C
1BRA2  =DATSTC.IN2C
0BRA2  =DATSTC.-IN2C
1BRA3  =DATSTC.IN3C
0BRA3  =DATSTC.-IN3C
1BRA4  =DATSTC.IN4C
```

*** VCS LOGIC EQUATIONS ***

INPUT CHANNEL C

```
0BRA4  =DATSTC.-IN4C
1BRA5  =DATSTC.IN5C
0BRA5  =DATSTC.-IN5C
1BRA6  =DATSTC.IN6C
0BRA6  =DATSTC.-IN6C
1BRA7  =DATSTC.IN7C
0BRA7  =DATSTC.-IN7C
1BRA8  =DATSTC.IN8C
0BRA8  =DATSTC.-IN8C
 SHFTC =QM3.(CYCLE.-MCVS2.MCVS4.(MCVS1.MCVS3.MCVS5+-MCVS1.-MCVS3.-MCVS5)
        +-DONE.MCVS4.(MCVS1.-MCVS2.-MCVS3.-MCVS5+-MCVS1.MCVS2.MCVS3
        .MCVS5)+MCVS3.-MCVS4.-MCVS5.(-MCVS1.MCVS2+MCVS1.-MCVS2
        +-MCVS1.-MCVS2.CDNMET)+MCVS4.MCVS5.(MCVS2.-MCVS3+MCVS1.-MCVS2
        .-MCVS3.BRBFC)+-MCVS1.-MCVS2.MCVS3.-MCVS4.MCVS5.BOTH)
 COBRC =LDBRB+-MCIC1.-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6.INSRVC+MCIC2.-MCIC
        4.-MCIC5.-MCIC6.(-MCIC1.MCIC3.(PAR1.PAR2+-PAR1.-PAR2)+MCIC1.-MCI
        C3.BRA4)
1BRB1C =COBRC.BRA1
0BRB1C =COBRC.-BRA1+SHFTC
1BRB2C =COBRC.BRA2+SHFTC. BRB1C
0BRB2C =COBRC.-BRA2+SHFTC.-BRB1C
1BRB3C =COBRC.BRA3+SHFTC. BRB2C
0BRB3C =COBRC.-BRA3+SHFTC.-BRB2C
1BRB4C =COBRC.BRA4+SHFTC. BRB3C
0BRB4C =COBRC.-BRA4+SHFTC.-BRB3C
1BRB5C =COBRC.BRA5 +SHFTC. BRB4C
0BRB5C =COBRC.-BRA5+SHFTC.-BRB4C
1BRB6C =COBRC.BRA6 +SHFTC. BRB5C
0BRB6C =COBRC.-BRA6+SHFTC.-BRB5C
1BRB7C =COBRC.BRA7 +SHFTC. BRB6C
0BRB7C =COBRC.-BRA7+SHFTC.-BRB6C
1BRB8C =COBRC.BRA8 +SHFTC. BRB7C
0BRB8C =COBRC.-BRA8+SHFTC.-BRB7C
1PAR4C =MCIC1.MCIC2.-MCIC3.-MCIC6.(-MCIC4.MCIC5+MCIC4.-MCIC5).(PAR1.PAR3
        +-PAR1.-PAR3)
0PAR4C =MCIC1.MCIC2.-MCIC3.-MCIC6.(-MCIC4.MCIC5+MCIC4.-MCIC5).(PAR1.-PAR
```

*** VCS LOGIC EQUATIONS ***

```
    INPUT CHANNEL C

         3+-PAR1.PAR3)
1PRBADC=(PAR1.-PAR2+-PAR1.PAR2).(-MCIC1.MCIC2.-MCIC6.(-MCIC5+-MCIC3.-MCI
        C4)+MCIC1.-MCIC2.-MCIC3.MCIC4.MCIC5.MCIC6)
0PRBADC=DATSTC
1BRBFC =-MCIC1.MCIC2.MCIC3.-MCIC4.-MCIC5.-MCIC6.(PAR1.PAR2+-PAR1.-PAR2)
0BRBFC =-MCIC1      .-MCIC3.-MCIC4.-MCIC5.-MCIC6+PWRON
1ADDVAL=MCIC1.-MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6.IN5C
0ADDVAL=-MCIC1.-MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6+PWRON
1PAR1  =DATSTC.IN9C
0PAR1  =DATSTC.-IN9C
1PAR3  =-MCIC1.MCIC2.MCIC3.-MCIC5.-MCIC6.PAR1
0PAR3  =-MCIC1.MCIC2.MCIC3.-MCIC5.-MCIC6.-PAR1
1END   =ENDMSC
0END   =DONEC+PWRON
1DONEC =END.LDBRB.-MCIC1.-MCIC2.MCIC3.-MCIC4.MCIC5.-MCIC6+PWRON
0DONEC =MCIC1.MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6.-BRA4
1MIC   =MCIC3.-MCIC4.-MCIC5.MCIC6.(-MCIC1.-MCIC2.(BRA3+BRA1.BRA2)+MCIC1.
        MCIC2.(PAR1.PAR2+-PAR1.-PAR2))
        +MCIC1.-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6
0MIC   =INSRVC+PWRON
```

```
                        ***  VCS LOGIC EQUATIONS  ***
    INPUT CHANNEL D

 EOM    =ENDMSD.-(-MCIC3.-MCIC4.MCIC5.(-MCIC2.-MCIC6+-MCIC1.MCIC2.MCIC6)
         +-MCIC6.MCIC4.-MCIC3.(-MCIC5.-MCIC2+MCIC5.MCIC2.-MCIC1))
 PARBA =PAR1.-PAR2+-PAR1.PAR2
1MCIC1 =-PWRON.(-MCIC2.-MCIC6.DATSTD
                                        .(-MCIC5+-MCIC3.-MCIC4)+MCIC2.MCIC5.D
         ATSTD
              .(MCIC4.-MCIC6+MCIC3.-MCIC6+-MCIC3.-MCIC4.MCIC6)+MCIC2.-MCIC
         3.-MCIC4.-MCIC5.-MCIC6.ADDVAL.(PAR1.PAR2+-PAR1.-PAR2)+-MCIC2. MC
         IC3.-MCIC4.MCIC5.-MCIC6.LDBRB+-MCIC2.MCIC3.-MCIC4.-MCIC5.MCIC6.-
         BRA3.(-BRA2+-BRA1)+-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6.INSRVD+MCIC2
         .-MCIC4.-MCIC5.                  MCIC3.MCIC6
         +MCIC2.-MCIC6.(MCIC4.-MCIC5
         +MCIC3.-MCIC5+-MCIC3.-MCIC4.MCIC5).-PARBA)
0MCIC1 =-MCIC4.-MCIC6.(-MCIC2+-MCIC5+MCIC2.-MCIC3.MCIC5.-END.LDBRB)
         +-MCIC2.MCIC4.-MCIC5.(-MCIC3+-MCIC6)
         +MCIC2.MCIC4.-MCIC5.-MCIC6.(MCIC3.LDBRB+-MCIC3.(LDBRB+END))
                                                +MCIC3.-MCIC4.MCIC6.DATSTD
         .(-MCI
         C2.-MCIC5+MCIC2.MCIC5)+MCIC2.MCIC3.-MCIC4.-MCIC5.MCIC6.(PAR1.PAR
         2+-PAR1.-PAR2)+EOM+PWRON
         +MCIC2.-MCIC3.-MCIC4.MCIC5.-MCIC6.END
1MCIC2 =-PWRON.(MCIC1.-MCIC6.(-MCIC5+-MCIC3.-MCIC4)+MCIC1.-MCIC3.MCIC4.-
         MCIC5+MCIC1.MCIC3.-MCIC4.-MCIC5.MCIC6.DATSTA
         +-MCIC1.MCIC3.-MCIC4.-MCIC5.MCIC6.(BRA3+BRA1.BRA2))
0MCIC2 =MCIC1.-MCIC4.-MCIC5.(-MCIC6+MCIC3)+-MCIC1.-MCIC3.-MCIC5.MCIC6
         .(-MCIC4+INSRVD)+-M
         CIC1.MCIC5.(MCIC3.-MCIC6+MCIC4.-MCIC6+-MCIC3.-MCIC4.MCIC6).DATST
         D+LDBRB.-END.MCIC1.-MCIC3.-MCIC4.MCIC5.-MCIC6
         +MCIC1.MCIC4.-MCIC5.-MCIC6.(MCIC3.LDBRB+-MCIC3.(LDBRB+END))
                                                +-MCIC1.-MCIC3.-MCIC4.-MCIC5.-MC
         IC6.-ADDVAL+PWRON+EOM
         +MCIC1.-MCIC3.-MCIC4.MCIC5.-MCIC6.END
1MCIC3 =-PWRON.(MCIC1.MCIC2.-MCIC6.(-MCIC4.(-MCIC5+MCIC5.(LDBRB+END))
         +MCIC4.-MCIC5.(LDBRB+END)))
0MCIC3 =MCIC1.-MCIC4.(MCIC2.-MCIC5+-MCIC2.MCIC5.-MCIC6)+MCIC1.MCIC2.MCIC
         4.-MCIC5.-MCIC6.LDBRB+PWRON+EOM
```

```
                              ***  VCS LOGIC EQUATIONS  ***
    INPUT CHANNEL D

          +-MCIC1.-MCIC2.-MCIC4.-MCIC5.MCIC6.(BRA3+BRA1.BRA2)
1MCIC4 =-PWRON.(MCIC1.MCIC2.MCIC3.-MCIC5.(-MCIC6+PAR1.PAR2+-PAR1.-PAR2)
          +-MCIC1.-MCIC2.MCIC3.-MCIC5.MCIC6. (BRA3+BRA1.BRA2)
          +LDBRB.-END.MCIC1.MCIC2.-MCIC3.MCIC5.-MCIC6)
0MCIC4 =MCIC2.-MCIC5.(MCIC1.MCIC3.-MCIC6.LDBRB+-MCIC1.-MCIC3.MCIC6
          .INSRVD+MCIC1.-MCIC3.-MCIC6.END)+EOM+PWRON
1MCIC5 =-PWRON.(MCIC1.MCIC2.MCIC3.MCIC4.-MCIC6.LDBRB
          +PARBA.MCIC2.(-MCIC1.-MCIC6.(MCIC3+MCIC4)+MCIC1.MCIC3.-MCIC4.
          MCIC6)+MCIC1.MCIC2.-MCIC3.MCIC4.-MCIC6.END)
0MCIC5 =-MCIC1.MCIC2.-MCIC6.DATSTD
                                        .(MCIC4+MCIC3)+MCIC1.MCIC3.-MCIC4.(-MCI
          C2.-MCIC6+MCIC2.MCIC6.DATSTA)+PWRON+EOM
          +LDBRB.-END.MCIC1.MCIC2.-MCIC3.-MCIC4.-MCIC6
1MCIC6 =-PWRON.(MCIC2.-MCIC3.(-MCIC1.PARBA+MCIC1.-MCIC4.-MCIC5.BRA4))
0MCIC6 =-MCIC1.-MCIC3.(-MCIC2.-MCIC4.-MCIC5+MCIC2.MCIC4.-MCIC5.INSRVD
          +MCIC2.-MCIC4.MCIC5.DATSTD)+PWRON+EOM
1NXBYTD=-MCIC4.-MCIC5.-MCIC6.MCIC2.(MCIC1.-MCIC3.-BRA4+-MCIC1.MCIC3.(PAR
          1.PAR2+-PAR1.-PAR2))+LDBRB.-END+-MCIC1.-MCIC2.MCIC3.-MCIC4.-MCIC
          5.MCIC6.-BRA3.(-BRA2+-BRA1)
0NXBYTD=DATSTD+PWRON
 INPR1 =IN1D.-IN2D+-IN1D.IN2D
 INPR2 =IN3D.-IN4D+-IN3D.IN4D
 INPR3 =IN5D.-IN6D+-IN5D.IN6D
 INPR4 =IN7D.-IN8D+-IN7D.IN8D
 INPR5 =INPR1.-INPR2+-INPR1.INPR2
 INPR6 =INPR3.-INPR4+-INPR3.INPR4
1PAR2  =DATSTD
0PAR2  =(INPR5.-INPR6+-INPR5.INPR6).(-MCIC1.MCIC2.MCIC3.-MCIC4.-MCIC5.MC
          IC6+-MCIC2.-MCIC6.MCIC1.(-MCIC5+-MCIC3.-MCIC4))
1BRA1  =IN1D.DATSTD
0BRA1  =DATSTD.-IN1D
1BRA2  =DATSTD.IN2D
0BRA2  =DATSTD.-IN2D
1BRA3  =DATSTD.IN3D
0BRA3  =DATSTD.-IN3D
1BRA4  =DATSTD.IN4D
```

*** VCS LOGIC EQUATIONS ***

INPUT CHANNEL D

```
0BRA4   =DATSTD.-IN4D
1BRA5   =DATSTD.IN5D
0BRA5   =DATSTD.-IN5D
1BRA6   =DATSTD.IN6D
0BRA6   =DATSTD.-IN6D
1BRA7   =DATSTD.IN7D
0BRA7   =DATSTD.-IN7D
1BRA8   =DATSTD.IN8D
0BRA8   =DATSTD.-IN8D
 SHFTD  =0M4.(CYCLE.-MCVS2.MCVS4.(MCVS1.MCVS3.MCVS5+-MCVS1.-MCVS3.-MCVS5)
         +-DONE.MCVS4.(MCVS1.-MCVS2.-MCVS3.-MCVS5+-MCVS1.MCVS2.MCVS3
         .MCVS5)+MCVS3.-MCVS4.-MCVS5.(-MCVS1.MCVS2+MCVS1.-MCVS2
         +-MCVS1.-MCVS2.CONMET)+MCVS4.MCVS5.(MCVS2.-MCVS3+MCVS1.-MCVS2
         .-MCVS3.BRBFD)+-MCVS1.-MCVS2.MCVS3.-MCVS4.MCVS5.BOTH)
 COBRD  =LDBRB+-MCIC1.-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6.INSRVD+MCIC2.-MCIC
         4.-MCIC5.-MCIC6.(-MCIC1.MCIC3.(PAR1.PAR2+-PAR1.-PAR2)+MCIC1.-MCI
         C3.BRA4)
1BRB1D  =COBRD.BRA1
0BRB1D  =COBRD.-BRA1+SHFTD
1BRB2D  =COBRD.BRA2+SHFTD. BRB1D
0BRB2D  =COBRD.-BRA2+SHFTD.-BRB1D
1BRB3D  =COBRD.BRA3+SHFTD. BRB2D
0BRB3D  =COBRD.-BRA3+SHFTD.-BRB2D
1BRB4D  =COBRD.BRA4+SHFTD. BRB3D
0BRB4D  =COBRD.-BRA4+SHFTD.-BRB3D
1BRB5D  =COBRD.BRA5 +SHFTD. BRB4D
0BRB5D  =COBRD.-BRA5+SHFTD.-BRB4D
1BRB6D  =COBRD.BRA6 +SHFTD. BRB5D
0BRB6D  =COBRD.-BRA6+SHFTD.-BRB5D
1BRB7D  =COBRD.BRA7 +SHFTD. BRB6D
0BRB7D  =COBRD.-BRA7+SHFTD.-BRB6D
1BRB8D  =COBRD.BRA8 +SHFTD. BRB7D
0BRB8D  =COBRD.-BRA8+SHFTD.-BRB7D
1PAR4D  =MCIC1.MCIC2.-MCIC3.-MCIC6.(-MCIC4.MCIC5+MCIC4.-MCIC5).(PAR1.PAR3
         +-PAR1.-PAR3)
0PAR4D  =MCIC1.MCIC2.-MCIC3.-MCIC6.(-MCIC4.MCIC5+MCIC4.-MCIC5).(PAR1.-PAR
```

*** VCS LOGIC EQUATIONS ***

```
    INPUT CHANNEL D

          3+-PAR1.PAR3)
1PRBADD=(PAR1.-PAR2+-PAR1.PAR2).(-MCIC1.MCIC2.-MCIC6.(-MCIC5+-MCIC3.-MCI
          C4)+MCIC1.-MCIC2.-MCIC3.MCIC4.MCIC5.MCIC6)
OPRBADD=DATSTD
1BRBFD =-MCIC1.MCIC2.MCIC3.-MCIC4.-MCIC5.-MCIC6.(PAR1.PAR2+-PAR1.-PAR2)
OBRBFD =-MCIC1       .-MCIC3.-MCIC4.-MCIC5.-MCIC6+PWRON
1ADDVAL=MCIC1.-MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6.IN5D
OADDVAL=-MCIC1.-MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6+PWRON
1PAR1  =DATSTD.IN9D
OPAR1  =DATSTD.-IN9D
1PAR3  =-MCIC1.MCIC2.MCIC3.-MCIC5.-MCIC6.PAR1
OPAR3  =-MCIC1.MCIC2.MCIC3.-MCIC5.-MCIC6.-PAR1
1END   =ENDMSD
OEND   =DONED+PWRON
1DONED =END.LDBRB.-MCIC1.-MCIC2.MCIC3.-MCIC4.MCIC5.-MCIC6+PWRON
ODONED =MCIC1.MCIC2.-MCIC3.-MCIC4.-MCIC5.-MCIC6.-BRA4
1MID   =MCIC3.-MCIC4.-MCIC5.MCIC6.(-MCIC1.-MCIC2.(BRA3+BRA1.BRA2)+MCIC1.
         MCIC2.(PAR1.PAR2+-PAR1.-PAR2))
          +MCIC1.-MCIC2.-MCIC3.MCIC4.-MCIC5.MCIC6
OMID   =INSRVD+PWRON
```

*** VCS LOGIC EQUATIONS ***

VOTEP SECTION

```
MCVSA =-MCVS1.MCVS2.-MCVS3.MCVS4.-MCVS5
RESET =-MCVS1.-MCVS2.-MCVS3.-MCVS4.-MCVS5
DONE  =DONEA.DONEB.COMPA+DONEA.DONEC.COMPB+DONEA.DONED.COMPC+DONEB.DONE
       C.COMPD+DONEB.DONED.COMPE+DONEC.DONED.COMPF+V3WAYA.(DONEA.DONEB+
       DONEA.DONEC+DONEB.DONEC)+V3WAYB.(DONEA.DONEB+DONEA.DONED+DONEB.D
       ONED)+V3WAYC.(DONEA.DONEC+DONEA.DONED+DONEC.DONED)+V3WAYD.(DONEB
       .DONEC+DONEB.DONED+DONEC.DONED)+V4WAY.(DONEA.DONEB.DONEC+DONEA.D
       ONEB.DONED+DONEA.DONEC.DONED+DONEB.DONEC.DONED)+SLCTA.DONEA+SLCT
       B.DONEB+SLCTC.DONEC+SLCTD.DONED
MAJSET=BRBFA.BRBFB+BRBFB.BRBFC+BRBFB.BRBFD+BRBFA.BRBFC+BRBFA.BRBFD+BRBF
       C.BRBFD+BRBFA.BRBFB.BRBFC+BRBFA.BRBFC.BRBFD+BRBFA.BRBFB.BRBFD+BR
       BFB.BRBFC.BRBFD
CONMET=TIMOUT+BRBFA.BRBFB.BRBFC+BRBFA.BRBFB.BRBFD+BRBFA.BRBFC.BRBFD+BRB
       FB.BRBFC.BRBFD+BRBFA.BRBFB.BRBFC.BRBFD
VOTOUT=MCVS1.-MCVS2.MCVS3.-MCVS4.-MCVS5+-MCVS1.MCVS2.-MCVS3.MCVS4.MCVS5
EITSET=BRBFA.(COMPA+COMPB+COMPC)+BRBFB.(COMPA+COMPD+COMPE)+BRBFC.(COMPB
       +COMPD+COMPF)+BRBFD.(COMPC+COMPE+COMPF)
MSVA  =-MCVS1.MCVS2.-MCVS3.MCVS4.-MCVS5
VSMD2 =MCVS1.-MCVS2.MCVS3.-CYCLE
VSMD3 =MCVS3.-MCVS4.-MCVS5.(MCVS1+MCVS2+-MCVS1.-MCVS2.CONMET)+-MCVS2.-M
       CVS3.MCVS4.-MCVS5.(-MCVS1.CYCLE+MCVS1.-DONE)
      +-MCVS2.MCVS3.-MCVS4.MCVS5.BOTH
VSMD4 =-MCVS3+MCVS2.-DONE+-MCVS2.(-MCVS1+CYCLE)
TKVOTE=MCVS3.-MCVS4+-MCVS2.-MCVS3.MCVS4.-MCVS5.(-MCVS1+-DONE)
OUSLCT=MCVS4.MCVS5
TIMOUT=TIM1.TIM2.TIM3.TIM4
LDBRB =-BCVS1.BCVS2.BCVS3.MCVS2.(MCVS1.-MCVS3.MCVS4.MCVS5
       +-MCVS1.MCVS3.-MCVS4.-MCVS5)
MCVSB =MCVS3.(MCVS1.MCVS2.-MCVS4.-MCVS5+-MCVS1.-MCVS2.MCVS4.MCVS5)
TKSTAT=-MCVS2.-MCVS3.MCVS4.-MCVS5+MCVS3.-MCVS4.-MCVS5.(MCVS1+MCVS2)
VSMD1 =-MCVS1.-MCVS2.-MCVS3.MCVS4.-MCVS5.-CYCLE
1BCVS1 =-RESET.(SHFTA+SHFTB+SHFTC+SHFTD+MCVS3.(MCVS1.MCVS2.-MCVS4
       .-MCVS5+-MCVS1.-MCVS2.MCVS4.MCVS5))
OBCVS1 =RESET+ (SHFTA+SHFTB+SHFTC+SHFTD+MCVS3.(MCVS1.MCVS2.-MCVS4
       .-MCVS5+-MCVS1.-MCVS2.MCVS4.MCVS5))
1BCVS2 =BCVS1.-RESET
```

*** VCS LOGIC EQUATIONS ***

VOTER SECTION

```
0BCVS2 =BCVS1+RESET
1BCVS3 =BCVS1.BCVS2.-RESET
0BCVS3 =BCVS1.BCVS2+RESET
1BCVS4 =BCVS1.BCVS2.BCVS3.-RESET
0BCVS4 =BCVS1.BCVS2.BCVS3+RESET
1TIM1  =INCLKA.-MCVS2.MCVS3.-MCVS4.-RESET
0TIM1  =INCLKA.-MCVS2.MCVS3.-MCVS4+RESET
1TIM2  =TIM1.-RESET
0TIM2  =TIM1+RESET
1TIM3  =TIM1.TIM2.-RESET
0TIM3  =TIM1.TIM2+RESET
1TIM4  =TIM1.TIM2.TIM3.-RESET
0TIM4  =TIM1.TIM2.TIM3+RESET
 S1VBR =(COMPA+          V3WAYA+V3WAYB).(BRB8A.BRB8B.VSMD3+PAR4A.PAR4B.VSM
        D1)+(COMPB+V3WAYA+V3WAYC).(BRB8A.BRB8C.VSMD3+PAR4A.PAR4C.VSMD1)+
        (COMPC+V3WAYC+V3WAYB).(BRB8A.BRB8D.VSMD3+PAR4A.PAR4D.VSMD1)+(COM
        PD+V3WAYA+V3WAYD).(BRB8B.BRB8C.VSMD3+PAR4B.PAR4C.VSMD1)+(COMPE+V
        3WAYB+V3WAYD).(BRB8B.BRB8D.VSMD3+PAR4B.PAR4D.VSMD1)+(COMPF+V3WAY
        D+V3WAYC).(BRB8C.BRB8D.VSMD3+PAR4C.PAR4D.VSMD1)+V4WAY.VSMD3.(BRB
        8A.BRB8B.BRB8C+BRB8A.BRB8B.BRB8D+BRB8A.BRB8C.BRB8D+BRB8B.BRB8C.B
        RB8D)+V4WAY.VSMD1.(PAR4A.PAR4B.PAR4C+PAR4A.PAR4B.PAR4D+PAR4A.PAR
        4A.PAR4C.PAR4D+PAR4B.PAR4C.PAR4D)
 S0VBR =(COMPA+          V3WAYA+V3WAYB).(-BRB8A.-BRB8B.VSMD3+-PAR4A.-PAR4B
        .VSMD1)+(COMPB+V3WAYA+V3WAYC).(-BRB8A.-BRB8C.VSMD3+-PAR4A.-PAR4C
        .VSMD1)+(COMPC+V3WAYC+V3WAYB).(-BRB8A.-BRB8D.VSMD3+-PAR4A.-PAR4D
        .VSMD1)+(COMPD+V3WAYA+V3WAYD).(-BRB8B.-BRB8C.VSMD3+-PAR4B.-PAR4C
        .VSMD1)+(COMPE+V3WAYB+V3WAYD).(-BRB8B.-BRB8D.VSMD3+-PAR4B.-PAR4D
        .VSMD1)+(COMPF+V3WAYD+V3WAYC).(-BRB8C.-BRB8D.VSMD3+-PAR4C.-PAR4D
        .VSMD1)+V4WAY.VSMD3.(-BRB8A.-BRB8B.-BRB8C+-BRB8A.-BRB8B.-BRB8D+-
        BRB8A.-BRB8C.-BRB8D+-BRB8B.-BRB8C.-BRB8D)+V4WAY.VSMD1.(-PAR4A.-P
        AR4B.-PAR4C+-PAR4A.-PAR4B.-PAR4D+-PAR4A.-PAR4C.-PAR4D+-PAR4B.-PA
        R4C.-PAR4D)
1VBR   =TKVOTE.S1VBR                          +OUSLCT.(SLCTA.(BRB8A.VSMD4+PAR
        4A.VSMD2)+SLCTB.(BRB8B.VSMD4+PAR4B.VSMD2)+SLCTC.(BRB8C.VSMD4+PAR
        4C.VSMD2)+SLCTD.(BRB8D.VSMD4+PAR4D.VSMD2))
0VBR   =TKVOTE. S0VBR
```

VOTER SECTION

```
                    +OUSLCT.(SLCTA.(-BRB8A.VSMD4+-PAR4A.VSMD2)+SLCTB.(-B
      RB8B.VSMD4+-PAR4B.VSMD2)+SLCTC.(-BRB8C.VSMD4+-PAR4C.VSMD2)+SLCTD
      .(-BRB8D.VSMD4+-PAR4D.VSMD2))
1STA1  =BRB8A
0STA1  =-BRB8A
1STA2  =BRB8B
0STA2  =-BRB8B
1STA3  =BRB8C
0STA3  =-BRB8C
1STA4  =BRB8D
0STA4  =-BRB8D
1STA5  =TKSTAT.(STA1.-VBR+-STA1.VBR)
0STA5  =TKSTAT.(STA1.VBR+-STA1.-VBR)+RESET
1STA6  =TKSTAT.(STA2.-VBR+-STA2.VBR)
0STA6  =TKSTAT.(STA2.VBR+-STA2.-VBR)+RESET
1STA7  =TKSTAT.(STA3.-VBR+-STA3.VBR)
0STA7  =TKSTAT.(STA3.VBR+-STA3.-VBR)+RESET
1STA8  =TKSTAT.(STA4.-VBR+-STA4.VBR)
0STA8  =TKSTAT.(STA4.VBR+-STA4.-VBR)+RESET
1MCVS1 =-PWRON.(                    -MCVS4.-MCVS5. -MCVS2.-MCVS3
           +-MCVS3.(MCVS2.MCVS4.MCVS5+-MCVS2.-MCVS4.-MCVS5)+-MCVS2.-MCV
      S3.MCVS4.-MCVS5.-CYCLE+MCVS2.MCVS3.(MCVS4.DONE+-MCVS4.-MCVS5.-BC
      VS1.BCVS2.BCVS3) +-DONE.MCVS2.MCVS3.MCVS4.MCVS5
      +-MCVS2.MCVS3.(-MCVS4.-MCVS5.CONMET+MCVS5.(BOTH+MCVS4)))
0MCVS1 =-MCVS2.-MCVS3.-MCVS4.-MCVS5.(V3WAYA+V3WAYB+V3WAYC+V3WAYD+V4WAY)
      +-BCVS1.BCVS2.BCVS3      .MCVS2.-MCVS3.MCVS4.MCVS5+-MCVS2.MCVS3.M
      CVS4.MCVS5.-CYCLE+-MCVS4.MCVS5.MCVS2.-MCVS3.EITSET+-MCVS2.-MCVS3
      .MCVS4.MCVS5.(BRBFA.SLCTA+BRBFB.SLCTB+BRBFC.SLCTC+BRBFD.SLCTD
      +-MCVS3.MCVS5.NEWMOD.(-MCVS2.MCVS4+MCVS2.-MCVS4)+MCVS4.
                  (MCVS2.MCVS3.MCVS5+-MCVS2.-MCVS3.-MCVS5)+MCVS3.-MCVS4
      .-MCVS5+PWRON
1MCVS2 =-PWRON.(MCVS1.MCVS4.(-MCVS3.-MCVS5+MCVS3.MCVS5)+MCVS1.-MCVS5.-MC
      VS4.(MCVS3+ V3WAYA+V3WAYB+V3WAYC+V3WAYD+V4WAY+COMPA+COMPB+COMPC+
      COMPD+COMPE+COMPF)+-MCVS3.MCVS4.(-MCVS1.-MCVS5.CYCLE+MCVS1.MCVS5
      .(BRBFA.SLCTA+BRBFB.SLCTB+BRBFC.SLCTC+BRBFD.SLCTD)+MCVS1.-MCVS5)
      )
```

*** VCS LOGIC EQUATIONS ***

VOTER SECTION

```
OMCVS2 =MCVS1.-MCVS3.MCVS5.(MCVS4.-BCVS1.BCVS2.BCVS3      +-MCVS4.EITSE
        T)+-MCVS1.-MCVS3.-MCVS4.-MCVS5.MAJSET+MCVS1.MCVS3.(MCVS4.MCVS5+-
        MCVS4.-MCVS5)+-MCVS1.-MCVS3.MCVS4.-MCVS5+PWRON
        +-MCVS3.-MCVS4.NEWMOD.(-MCVS1.-MCVS5+MCVS1.MCVS5)
1MCVS3 =-PWRON.(MCVS1.MCVS2.(-MCVS4.MCVS5.EITSET+MCVS4.MCVS5.-BCVS1.BCVS
        2.BCVS3      )+-MCVS2.MCVS4.-MCVS5.(-MCVS1.CYCLE+MCVS1.-DONE)+-
        MCVS1.MCVS2.-MCVS4.-MCVS5.MAJSET)
OMCVS3 =MCVS1.MCVS2.(-MCVS4.-MCVS5+MCVS4.MCVS5)+-MCVS2.MCVS5.(-MCVS1.-MC
        VS4.TIMOUT+MCVS1.MCVS4.CYCLE)+-MCVS1.MCVS2.MCVS4.MCVS5.-DONE+PWR
        ON
1MCVS4 =-PWRON.MCVS1.-MCVS5.(MCVS2.MCVS3+-MCVS2.-MCVS3.(SLCTA+SLCTB+SLCT
        C+SLCTD))
OMCVS4 =-MCVS2.-MCVS3.-MCVS5.(-MCVS1.CYCLE+MCVS1.-DONE)+MCVS1.MCVS5.(-MC
        VS2.-MCVS3.NEWMOD+MCVS2.MCVS3)+-MCVS1.MCVS2.-MCVS3.-MCVS5+PWRON
1MCVS5 =-PWRON.MCVS1.-MCVS2.-MCVS3.-MCVS4.(SLCTA+SLCTB+SLCTC+SLCTD+COMPA
        +COMPB+COMPC+COMPD+COMPE+COMPF)
OMCVS5 =-MCVS1.-MCVS2.MCVS3.-MCVS4.(TIMOUT+COMPA.BRBFA.BRBFB+COMPB.BRBFA
        .BRBFC+COMPC.BRBFA.BRBFD+COMPD.BRBFB.BRBFC+COMPE.BRBFB.BRBFD+COM
        PF.BRBFC.BRBFD)+MCVS1.MCVS4.(MCVS2.MCVS3
        +MCVS1.-MCVS3.NEWMOD.(-MCVS2.MCVS4+MCVS2.-MCVS4)+PWRON
1CYCLE =-PWRON.MCVSB
OCYCLE =MCVSB+PWRON
```

*** VCS LOGIC EQUATIONS ***

OUTPUT CHANNEL

```
 MADEC2=LPIN
 RDATA =-BCOC1.-BCOC2.BCOC3.-BCOC4.-MCOC1.MCOC2.-MCOC3.MCOC4.-MCOC5
 MSA   =MS.-MS1.-MS2
 MSB   =MS.MS1.-MS2
 MSC   =MS.-MS1.MS2
 MSD   =MS.MS1.MS2
 RESP  =-BCOC1.-BCOC2.-BCOC3.MCOC5.(BCOC4+-MCOC1.-MCOC2.-MCOC3.-MCOC4)
 RMBR  =OREQ.MCOC1.-MCOC2.-MCOC4.-MCOC5
 RESETA=-MCOC1.-MCOC2.-MCOC3.-MCOC4.-MCOC5
 RESETC=MCOC4+MCOC5
1MCOC1 =-PWRON.(MCOC3.MCOC5.BCOC4.(MCOC2.-MCOC4+-MCOC2.MCOC4)+-MCOC2.MCO
         C3.(-MCOC4+-MCOC5)+-MCOC3.-MCOC4.(-MCOC5+MCOC2.PRES)+MCOC2.MCOC4
         .(MCOC3.MCOC5+-MCOC3.-MCOC5.BCOC3)+-MCOC4.-MCOC5.PRES)
0MCOC1 =OREQ.-MCOC4.-MCOC5                          +MCOC3.(MCOC4.MCOC5+-MCOC
         2.-MCOC4)+MCOC2.(MCOC4.MCOC5+-MCOC3.-MCOC4+MCOC3.MCOC4)+PWRON
1MCOC2 =-PWRON.(      -MCOC1.-MCOC3.-MCOC4.MCOC5.-BCOC1.-BCOC2.-BCOC3.-BC
         OC4+-MCOC4.-MCOC5.(PRES+MCOC1.OREQ)   +MCOC1.MCOC3.(MCOC5+MCOC4.B
         COC2))
0MCOC2 =MCOC1.(MCOC4.MCOC5+-MCOC3.-MCOC4+MCOC3.MCOC4.-MCOC5)
         +-MCOC1.-MCOC4.MCOC5.(-MCOC3.-PRES+MCOC3.BCOC4)+PWRON
1MCOC3 =-PWRON.(MCOC1.MCOC2.(-MCOC4+MCOC5)+-MCOC1.MCOC2.-MCOC4.MCOC5.-PR
         ES)
0MCOC3 =MCOC1.(MCOC2.MCOC4+-MCOC2.-MCOC4.-MCOC5)+-MCOC4.-MCOC5.PRES+PWRO
         N
1MCOC4 =-PWRON.(                                    -MCOC1.MCOC2.MCOC5.(MC
         OC3.BCOC4+-MCOC3.-PRES)+PRES. -MCOC4.-MCOC5)
0MCOC4 =MCOC1.MCOC2.MCOC3+PWRON
1MCOC5 =-PWRON.(MCOC1.-MCOC2.-MCOC3.MCOC4.PRES+-MCOC4.-MCOC5.PRES)
0MCOC5 =-MCOC1.MCOC2.-MCOC3.-MCOC4.-PRES+PWRON
1ODTST =-MCOC2.MCOC3.MCOC4.(MCOC1.MCOC5+-MCOC1.-MCOC5)
         +MCOC1.MCOC2.-MCOC3.-MCOC4.MCOC5
0ODTST =MCOC1.MCOC3.MCOC5.(MCOC2.MCOC4+-MCOC2.-MCOC4)
         +MCOC1.-MCOC2.MCOC3.MCOC4.-MCOC5.-BCOC1.BCOC2.-BCOC3.-BCOC4
1MS    =-MCOC1.MCOC2.-MCOC3.-MCOC4.-MCOC5
0MS    =-MCOC1.-MCOC2.MCOC3.-MCOC4.-MCOC5+PWRON
1OR    =-PWRON.(OREQ+-MCOC4.-MCOC5.PRES)
```

```
                              ***  VCS LOGIC EQUATIONS  ***
    OUTPUT CHANNEL

00R     =0R+PWR0N
1SP1    =MADEC2
0SP1    =-MADEC2
1SP2    =SP1.MC0C5
0SP2    =-SP1.MC0C5
1SP3    =SP2.MC0C5
0SP3    =-SP2.MC0C5
1SP4    =MC0C5.SP3
0SP4    =MC0C5.-SP3
1SP5    =MC0C5.SP4
0SP5    =MC0C5.-SP4
1SP6    =MC0C5.SP5
0SP6    =MC0C5.-SP5
1SP7    =MC0C5.SP6
0SP7    =MC0C5.-SP6
1SP8    =MC0C5.SP7
0SP8    =MC0C5.-SP7
10BR1   =MBR1.RMBR+SP1.RESP
00BR1   =-MBR1.RMBR+-SP1.RESP
10BR2   =MBR2.RMBR+SP2.RESP
00BR2   =-MBR2.RMBR+-SP2.RESP
10BR3   =MBR3.RMBR+SP3.RESP
00BR3   =-MBR3.RMBR+-SP3.RESP
10BR4   =MBR4.RMBR+SP4.RESP
00BR4   =-MBR4.RMBR+-SP4.RESP
10BR5   =MBR5.RMBR+SP5.RESP
00BR5   =-MBR5.RMBR+-SP5.RESP
10BR6   =MBR6.RMBR+SP6.RESP
00BR6   =-MBR6.RMBR+-SP6.RESP
10BR7   =MBR7.RMBR+SP7.RESP
00BR7   =-MBR7.RMBR+-SP7.RESP
10BR8   =MBR8.RMBR+SP8.RESP
00BR8   =-MBR8.RMBR+-SP8.RESP
10CP1   =MADEC2+-BC0C1.-BC0C2.-BC0C3.-BC0C4
00CP1   =MADEC2
1PAROUT=0CP1.-BC0C1.-BC0C2.-BC0C3.BC0C4+-BC0C1.-BC0C2.-BC0C3.-BC0C4.(0CP
```

*** VCS LOGIC EQUATIONS ***

OUTPUT CHANNEL

```
        1.MADEC2+-OCP1.-MADEC2)+OCP2.-MCOC5
OPAROUT=-OCP1.-BCOC1.-BCOC2.-BCOC3.BCOC4+-BCOC1.-BCOC2.-BCOC3.-BCOC4.(OC
        P1.-MADEC2+-OCP1.MADEC2)+-OCP2.-MCOC5
1BCOC1 =-RESETA.(-MCOC3.-MCOC4.PRES.(MCOC1.-MCOC2.-MCOC5
        +-MCOC1.MCOC2.MCOC5)+-MCOC1.-MCOC2.-MCOC3.-MCOC4.MCOC5
        .(BCOC1+BCOC2+BCOC3+BCOC4)+MCOC1.MCOC2.MCOC5
        +MCOC3.MCOC5+-MCOC2.MCOC3.MCOC4)
0BCOC1 =RESETA+ (-MCOC3.-MCOC4.PRES.(MCOC1.-MCOC2.-MCOC5
        +-MCOC1.MCOC2.MCOC5)+-MCOC1.-MCOC2.-MCOC3.-MCOC4.MCOC5
        .(BCOC1+BCOC2+BCOC3+BCOC4)+MCOC1.MCOC2.MCOC5
        +MCOC3.MCOC5+-MCOC2.MCOC3.MCOC4)
1BCOC2 =-RESETA.BCOC1
0BCOC2 =BCOC1+RESETA
1BCOC3 =-RESETA.BCOC1.BCOC2
0BCOC3 =BCOC1.BCOC2+RESETA+RDATA
1BCOC4 =-RESETA.BCOC1.BCOC2.BCOC3
0BCOC4 =BCOC1.BCOC2.BCOC3+RESETA
 OUTPR3=MBR1.-MBR2+-MBR1.MBR2
 OUTPR4=MBR3.-MBR4+-MBR3.MBR4
 OUTPR5=MBR5.-MBR6+-MBR5.MBR6
 OUTPR6=MBR7.-MBR8+-MBR7.MBR8
 OUTPR1=OUTPR3.-OUTPR4+-OUTPR3.OUTPR4
 OUTPR2=OUTPR5.-OUTPR6+-OUTPR5.OUTPR6
1OCP2  =-MCOC1.-MCOC2.MCOC3.-MCOC4.-MCOC5+RESETA
0OCP2  =OREQ.(OUTPR1.-OUTPR2+-OUTPR1.OUTPR2)
1MS1   =RMBR.(DESTB+DESTD)
0MS1   =RMBR.(DESTA+DESTC)
1MS2   =RMBR.(DESTC+DESTD)
0MS2   =RMBR.(DESTA+DESTB)
```

*** VCS LOGIC EQUATIONS ***

MATRIX SECTION

```
V4WAY =P1.R1.R2.R3.R4.(P6.R5.R6.R7.R8.(P11.R9.R10.R11.R12
      +P16.R13.R14.R15.R16)+P11.R9.R10.R11.R12.P16.R13.R14.R15.R16)
      +P6.R5.R6.R7.R8.P11.R9.R10.R11.R12.P16.R13.R14.R15.R16
V3WAYA=P1.R1.R2.R3.-R4.P6.R5.R6.R7.-R8.P11.R9.R10.R11.-R12
      +POR2                          .(P1.R1.R2.R3.-R4.(P6.R5.R6.R7.-R8
      +P11.R9.R10.R11.-R12)+P6.R5.R6.R7.-R8.P11.R9.R10.R11.-R12)
V3WAYB=P1.R1.R2.-R3.R4.P6.R5.R6.-R7.R8.P16.R13.R14.-R15.R16
      +POR1                          .(P1.R1.R2.-R3.R4.(P6.R5.R6.-R7.R8
      +P16.R13.R14.-R15.R16)+P6.R5.R6.-R7.R8.P16.R13.R14.-R15.R16)
V3WAYC=P1.R1.-R2.R3.R4.P11.R9.-R10.R11.R12.P16.R13.-R14.R15.R16
      +POR3                       .(P1.R1.-R2.R3.R4.(P11.R9.-R10.R11.R12
      +P16.R13.-R14.R15.R16)+P11.R9.-R10.R11.R12.P16.R13.-R14.R15.R16)
V3WAYD=P6.-R5.R6.R7.R8.P11.-R9.R10.R11.R12.P16.-R13.R14.R15.R16
      +POR4                       .(P6.-R5.R6.R7.R8.(P11.-R9.R10.R11.R12
      +P16.-R13.R14.R15.R16)+P11.-R9.R10.R11.R12.P16.-R13.R14.R15.R16)
COMPA =P1.R1.R2.-R3.-R4.P6.R5.R6.-R7.-R8.(POR1+POR2)
COMPB =P1.R1.-R2.R3.-R4.P11.R9.-R10.R11.R12.(POR3+POR2)
COMPC =P1.R1.-R2.-R3.R4.P16.R13.-R14.-R15.R16.(POR3+POR1)
COMPD =P6.-R5.R6.R7.-R8.P11.-R9.R10.R11.-R12.(POR2+POR4)
COMPE =P6.-R5.R6.-R7.R8.P16.-R13.R14.-R15.R16.(POR1+POR4)
COMPF =P11.-R9.-R10.R11.R12.P16.-R13.-R14.R15.R16.(POR3+POR4)
SLCTA =P1.R1.-R2.-R3.-R4.POR3.POR1.POR2
SLCTB =P6.-R5.R6.-R7.-R8.POR1.POR2.POR4
SLCTC =P11.-R9.-R10.R11.-R12.POR2.POR3.POR4
SLCTD =P16.-R13.-R14.-R15.R16.POR1.POR3.POR4
POR1  =-P11+-R9.-R10.-R11.-R12
POR2  =-P16+-R13.-R14.-R15.-R16
POR3  =-P6+-R5.-R6.-R7.-R8
POR4  =-P1+-R1.-R2.-R3.-R4
AUSED =V4WAY+V3WAYA+V3WAYB+V3WAYC+COMPA+COMPB+COMPC+SLCTA
BUSED =V4WAY+V3WAYA+V3WAYB+V3WAYD+COMPA+COMPD+COMPE+SLCTB
CUSED =V4WAY+V3WAYA+V3WAYC+V3WAYD+COMPB+COMPD+COMPF+SLCTC
DUSED =V4WAY+V3WAYB+V3WAYC+V3WAYD+COMPC+COMPE+COMPF+SLCTD
MI0   =-MIC1.-MIC2
MI1   =MIC1.-MIC2
MI2   =-MIC1.MIC2
```

*** VCS LOGIC EQUATIONS ***

MATRIX SECTION

```
MI3    =MIC1.MIC2
MAMO   =-MCMS1.-MCMS2.-MCMS3.-MCMS4.-MCMS5
MAM3   =-OPC3.MCMS1.MCMS2.-MCMS3.-MCMS4.-MCMS5
MAM5   =MCMS4+MCMS5
READA  =MAM3.MIA.MI1
READB  =MAM3.MIB.MI2
READC  =MAM3.MIC.MI3
READD  =MAM3.MID.MI0
OUTRA  =OPC3.-OPC2.-MCMS1.MCMS2.-MCMS3.-MCMS4.-MCMS5
OUTRB  =-MCMS1.-MCMS2.-MCMS3.MCMS4.-MCMS5
OUTPA  =MCMS2.-MCMS3.-MCMS5.(MCMS1.MCMS4+-MCMS1.-MCMS4.-OPC2      .OPC3)
OUTPB  =MCMS1.-MCMS2.MCMS3.MCMS4.-MCMS5
OUTSA  =-MCMS1.-MCMS3.-MCMS4.(-MCMS2.MCMS5+MCMS2.-MCMS5.OPC1.OPC2.OPC3)
OUTSB  =-MCMS1.MCMS2.-MCMS3.-MCMS4.MCMS5
DESTA  =MAM5.MI1
DESTB  =MAM5.MI2
DESTC  =MAM5.MI3
DESTD  =MAM5.MI0
MI     =MIA.MI1+MIB.MI2+MIC.MI3+MID.MI0
CPYOC1=MAMO.MI0
CPYOC2=MAMO.MI1
CPYOC3=MAMO.MI2
CPYOC4=MAMO.MI3
INSRVA=INSRV.MI1
INSRVB=INSRV.MI2
INSRVC=INSRV.MI3
INSRVD=INSRV.MI0
MODEA  =MAMO.MCVSA.(V4WAY+V3WAYA+V3WAYB+V3WAYC+COMPA+COMPB+COMPC)
MODEB  =MAMO.MCVSA.(V4WAY+V3WAYA+V3WAYB+V3WAYD+COMPA+COMPD+COMPE)
MODEC  =MAMO.MCVSA.(V4WAY+V3WAYA+V3WAYC+V3WAYD+COMPB+COMPD+COMPF)
MODED  =MAMO.MCVSA.(V4WAY+V3WAYB+V3WAYC+V3WAYD+COMPC+COMPE+COMPF)
SI1    =STA5.MODEA
SI2    =STA6.MODEB
SI3    =STA7.MODEC
SI4    =STA8.MODED
COPYP  =-OPC2      .-OPC3.MCMS1.-MCMS2.MCMS3.-MCMS4.-MCMS5
```

*** VCS LOGIC EQUATIONS ***

MATRIX SECTION

```
CPYP0 =COPYP.MI1
CPYP1 =COPYP.MI2
CPYP2 =COPYP.MI3
CPYP3 =COPYP.MI0
COPYR =-OPC2      .-OPC3.MCMS1.-MCMS2.MCMS3.-MCMS4.-MCMS5
CPYR0 =COPYR.MI1
CPYR1 =COPYR.MI2
CPYR2 =COPYR.MI3
CPYR3 =COPYR.MI0
COPYS =-OPC3.(-OPC1.-OPC2+OPC1.OPC2).-MCMS1.MCMS2.MCMS3.-MCMS4.-MCMS5
CPYS0 =COPYS.MI1
CPYS1 =COPYS.MI2
CPYS2 =COPYS.MI3
CPYS3 =COPYS.MI0
SAMP  =-MCMS1.MCMS2.-MCMS3.-MCMS4.-MCMS5.-OPC1.OPC2.OPC3
1OM1  =AUSED
0OM1  =-AUSED
1OM2  =BUSED
0OM2  =-BUSED
1OM3  =CUSED
0OM3  =-CUSED
1OM4  =DUSED
0OM4  =-DUSED
1MCMS1 =-PWRON.(MCMS2.-MCMS3.-MCMS4.-MCMS5.(-OPC2.-OPC3
                                              +OPC1.OPC2.OPC3+-OPC1.-
       OPC2)+-MCMS3.-MCMS4.MCMS5+-MCMS2.-MCMS5.(MCMS3.-MCMS4+-MCMS3.MCM
       S4)+MCMS4.-MCMS5.OR.(MCMS2.-MCMS3+-MCMS2.MCMS3)+MCMS2.MCMS3.MCMS
       4.-MCMS5+                    -MCMS2.
                                          -MCMS3.-MCMS4.-MCMS5.(MI0.MIA+MI
       1.MIB+MI2.MIC+MI3.MID))
0MCMS1 =-MCMS3.-MCMS4.(-MCMS2.-MCMS5+MCMS2.MCMS5)+MCMS4.-MCMS5.(-MCMS3+-
       MCMS2)+OR.(MCMS2.MCMS3.(-MCMS5.MCMS4+-MCMS4.MCMS5)+-MCMS2.-MCMS3
       .-MCMS4.MCMS5)+-MCMS2.-MCMS5.MCMS3.-MCMS4.-OPC3.-OPC2      +MCMS2
       .-MCMS3.-MCMS4.-MCMS5.MI                                  +RESETC.(
       MCMS4+MCMS5)+PWRON
1MCMS2 =-PWRON.(MCMS1.-MCMS5.(-MCMS3.-MCMS4+MCMS3.MCMS4+-MCMS3.MCMS4.-OP
```

*** VCS LOGIC EQUATIONS ***

MATRIX SECTION

```
          C1.-OPC2.OPC3+MCMS3.-MCMS4.-OPC1.-OPC2.-OPC3)+MCMS1.-MCMS3.-MCMS
          4.MCMS5.OR)
0MCMS2  =MCMS1.MCMS3.OR.(-MCMS4.MCMS5+MCMS4.-MCMS5)+MCMS1.-MCMS3.(MCMS4.-
          MCMS5+-MCMS4.MCMS5)+-MCMS3.-MCMS4.-MCMS5.(-MCMS1.OPC2.OPC3.OPC1+
          MCMS1.MI)+-MCMS1.MCMS3.-MCMS4.-MCMS5+RESETC.(MCMS4+MCMS5)+PW
          RON+-MCMS1.-MCMS3.MCMS4.MCMS5.OR
1MCMS3  =-PWRON.(MCMS2.-MCMS4.-MCMS5.(-MCMS1.OPC3.-OPC2.-OPC1+-MCMS1.OPC2
                .-OPC3.OPC1 +MCMS1.MI                                    )+MCMS
          1.MCMS2.(MCMS4.-MCMS5+-MCMS4.MCMS5))
0MCMS3  =OR.(MCMS1.MCMS2.(-MCMS5.MCMS4+-MCMS4.MCMS5)+-MCMS1.-MCMS2.-MCMS4
          .MCMS5)+-MCMS4.-MCMS5.(-MCMS1.MCMS2+MCMS1.-MCMS2.-OPC3. OPC1.-OP
          C2)              +RESETC.(MCMS4+MCMS5)+PWRON
1MCMS4  =-PWRON.(MCMS2.(MCMS1.MCMS3.MCMS5.OR+-MCMS1.-MCMS3.-MCMS5.-OPC1.O
          PC2.OPC3))
0MCMS4  =MCMS2.OR.(MCMS1.MCMS3.-MCMS5+-MCMS1.-MCMS3.MCMS5)+RESETC.(MCMS4+
          MCMS5)+PWRON
1MCMS5  =-PWRON.(MCMS2.MCMS3.MCMS4. MCMS1.OR                          +-MCM
          S3.(-MCMS1.MCMS2.-MCMS4.OPC3.(-OPC1+OPC2)))
0MCMS5  =OR.MCMS3.-MCMS4.(-MCMS1.-MCMS2+MCMS1.MCMS2)+RESETC.(MCMS4+MCMS5)
          +PWRON+-MCMS1.MCMS2.-MCMS3.MCMS4.OR
1R1     =MBR5.CPYRO
0R1     =-MBR5.CPYRO+PWRON
1R2     =CPYRO.MBR6
0R2     =CPYRO.-MBR6+PWRON
1R3     =CPYRO.MBR7
0R3     =CPYRO.-MBR7+PWRON
1R4     =CPYRO.MBR8
0R4     =CPYRO.-MBR8+PWRON
1R5     =CPYR1.MBR5
0R5     =CPYR1.-MBR5+PWRON
1R6     =CPYR1.MBR6
0R6     =CPYR1.-MBR6+PWRON
1R7     =CPYR1.MBR7
0R7     =CPYR1.-MBR7+PWRON
1R8     =CPYR1.MBR8
0R8     =CPYR1.-MBR8+PWRON
```

*** VCS LOGIC EQUATIONS ***

MATRIX SECTION

```
1R9     =CPYR2.MBR5
0R9     =CPYR2.-MBR5+PWRON
1R10    =CPYR2.MBR6
0R10    =CPYR2.-MBR6+PWRON
1R11    =CPYR2.MBR7
0R11    =CPYR2.-MBR7+PWRON
1R12    =CPYR2.MBR8
0R12    =CPYR2.-MBR8+PWRON
1R13    =CPYR3.MBR5
0R13    =CPYR3.-MBR5+PWRON
1R14    =CPYR3.MBR6
0R14    =CPYR3.-MBR6+PWRON
1R15    =CPYR3.MBR7
0R15    =CPYR3.-MBR7+PWRON
1R16    =CPYR3.MBR8
0R16    =CPYR3.-MBR8+PWRON
1P1     =PWRON+COMNOA.(P6.P5.P11.P9+P6.P5.P16.P13+P11.P9.P16.P13
         +-P6.-P11.-P16)
0P1     =-PWRON.(-COMNOA+P6.-P5.P11.-P9+P6.-P5.P16.-P13+P11.-P9.P16.-P13)
1P2     =CPYP0.MBR2+PWRON
0P2     =-PWRON.CPYP0.-MBR2
1P3     =CPYP0.MBR3+PWRON
0P3     =CPYP0.-MBR3.-PWRON
1P4     =CPYP0.MBR4+PWRON
0P4     =CPYP0.-MBR4.-PWRON
1P5     =CPYP1.MBR1+PWRON
0P5     =CPYP1.-MBR1.-PWRON
1P6     =PWRON+COMNOB.(P1.P2.P11.P10+P1.P2.P16.P14+P11.P10.P16.P14
         +-P1.-P11.-P16)
0P6     =-PWRON.(-COMNOB+P1.-P2.P11.-P10+P1.-P2.P16.-P14+P11.-P10.P16.
         -P14)
1P7     =CPYP1.MBR3+PWRON
0P7     =CPYP1.-MBR3.-PWRON
1P8     =CPYP1.MBR4+PWRON
0P8     =CPYP1.-MBR4.-PWRON
1P9     =CPYP2.MBR1+PWRON
```

*** VCS LOGIC EQUATIONS ***

MATRIX SECTION

```
0P9     =CPYP2.-MBR1.-PWRON
1P10    =CPYP2.MBR2+PWRON
0P10    =CPYP2.-MBR2.-PWRON
1P11    =PWRON+COMNOC.(P1.P3.P6.P7+P1.P3.P16.P15+P6.P7.P16.P15
          +-P1.-P6.-P16)
0P11    =-PWRON.(-COMNOC+P1.-P3.P6.-P7+P1.-P3.P16.-P15+P6.-P7.P16.-P15)
1P12    =CPYP2.MBR4+PWRON
0P12    =CPYP2.-MBR4.-PWRON
1P13    =CPYP3.MBR1+PWRON
0P13    =CPYP3.-MBR1.-PWRON
1P14    =CPYP3.MBR2+PWRON
0P14    =CPYP3.-MBR2.-PWRON
1P15    =CPYP3.MBR3+PWRON
0P15    =CPYP3.-MBR3.-PWRON
1P16    =PWRON+COMNOD.(P1.P4.P6.P8+P1.P4.P11.P12+P6.P8.P11.P12
          +-P1.-P6.-P11)
0P16    =-PWRON.(-COMNOD+P1.-P4.P6.-P8+P1.-P4.P11.-P12+P6.-P8.P11.-P12)
1S1     =SI1.-PWRON
0S1     =CPYS0+PWRON
1S2     =SI2.-PWRON
0S2     =CPYS0+PWRON
1S3     =SI3.-PWRON
0S3     =CPYS0+PWRON
1S4     =SI4.-PWRON
0S4     =CPYS0+PWRON
1S5     =SI1.-PWRON
0S5     =CPYS1+PWRON
1S6     =SI2.-PWRON
0S6     =CPYS1+PWRON
1S7     =SI3.-PWRON
0S7     =CPYS1+PWRON
1S8     =SI4.-PWRON
0S8     =CPYS1+PWRON
1S9     =SI1.-PWRON
0S9     =CPYS2+PWRON
1S10    =SI2.-PWRON
```

*** VCS LOGIC EQUATIONS ***

MATRIX SECTION

```
0S10    =CPYS2+PWRON
1S11    =SI3.-PWRON
0S11    =CPYS2+PWRON
1S12    =SI4.-PWRON
0S12    =CPYS2+PWRON
1S13    =SI1.-PWRON
0S13    =CPYS3+PWRON
1S14    =SI2.-PWRON
0S14    =CPYS3+PWRON
1S15    =SI3.-PWRON
0S15    =CPYS3+PWRON
1S16    =SI4.-PWRON
0S16    =CPYS3+PWRON
 CLOPC  =PWRON+MCMS1.-MCMS2.MCMS3.-MCMS4+-MCMS1.MCMS2
         .(MCMS3.-MCMS4.-MCMS5+-MCMS3.MCMS4.MCMS5)
1OREQ   =MCMS4.-MCMS5.(-MCMS1.-MCMS2.-MCMS3+MCMS1.MCMS2.-MCMS3+MCMS1.-MCM
         S2.MCMS3)+-MCMS1.-MCMS3.-MCMS4.(MCMS5+MCMS2.-MCMS5.OPC3
         .(-OPC1+OPC2))
0OREQ   =OR+PWRON
1INSRV  =(MI0.MIA+MI1.MIB+MI2.MIC+MI3.MID).MAMO
         +MCMS1.MCMS2.-MCMS3.-MCMS4.-MCMS5.(MI1.MIA+MI2.MIB+MI3.MIC
         +MI0.MID)
0INSRV  =INSRV
1NEWMOD=MCMS1.-MCMS2.MCMS3.-MCMS4.-MCMS5.-OPC2.-OPC3
0NEWMOD=NEWMOD
1MIC1   =INCLKA.MAMO.-PWRON
0MIC1   =INCLKA.MAMO+PWRON
1MIC2   =MIC1.MAMO.-PWRON
0MIC2   =MIC1.MAMO+ PWRON
1OPC1   =CPYOC1.BRB1A+CPYOC2.BRB1B+CPYOC3.BRB1C+CPYOC4.BRB1D
0OPC1   =CLOPC
1OPC2   =CPYOC1.BRB2A+CPYOC2.BRB2B+CPYOC3.BRB2C+CPYOC4.BRB2D
0OPC2   =CLOPC
1OPC3   =CPYOC1.BRB3A+CPYOC2.BRB3B+CPYOC3.BRB3C+CPYOC4.BRB3D
0OPC3   =CLOPC
1MBR1   =BRB1A.READA+BRB1B.READB+BRB1C.READC+BRB1D.READD+P1.
```

MATRIX SECTION

```
          (OUTPA+SAMP)+P9.OUTP
          B+R1.OUTRA+R9.OUTRB+S1.OUTSA+S9.OUTSB
0MBR1   =OR+PWRON+MAMO
1MBR2   =BRB2A.READA+BRB2B.READB+BRB2C.READC+BRB2D.READD+P2.OUTPA+P10.OUT
          PB+R2.OUTRA+R10.OUTRB+S2.OUTSA+S10.OUTSB+P6.SAMP
0MBR2   =OR+PWRON+MAMO
1MBR3   =BRB3A.READA+BRB3B.READB+BRB3C.READC+BRB3D.READD+P3.OUTPA+P11.
          (OUTPB+SAMP)
            +R3.OUTRA+R11.OUTRB+S3.OUTSA+S11.OUTSB
0MBR3   =OR+PWRON+MAMO
1MBR4   =BRB4A.READA+BRB4B.READB+BRB4C.READC+BRB4D.READD+P4.OUTPA+P12.OUT
          PB+R4.OUTRA+R12.OUTRB+S4.OUTSA+S12.OUTSB+P16.SAMP
0MBR4   =OR+PWRON+MAMO
1MBR5   =BRB5A.READA+BRB5B.READB+BRB5C.READC+BRB5D.READD+P5.OUTPA+P13.OUT
          PB+R5.OUTRA+R13.OUTRB+S5.OUTSA+S13.OUTSB+OM1.SAMP
0MBR5   =OR+PWRON+MAMO
1MBR6   =BRB6A.READA+BRB6B.READB+BRB6C.READC+BRB6D.READD+P6.OUTPA+P14.OUT
          PB+R6.OUTRA+R14.OUTRB+S6.OUTSA+S14.OUTSB+OM2.SAMP
0MBR6   =OR+PWRON+MAMO
1MBR7   =BRB7A.READA+BRB7B.READB+BRB7C.READC+BRB7D.READD+P7.OUTPA+P15.OUT
          PB+R7.OUTRA+R15.OUTRB+S7.OUTSA+S15.OUTSB+OM3.SAMP
0MBR7   =OR+PWRON+MAMO
1MBR8   =BRB8A.READA+BRB8B.READB+BRB8C.READC+BRB8D.READD+P8.OUTPA+P16.OUT
          PB+R8.OUTRA+R16.OUTRB+S8.OUTSA+S16.OUTSB+OM4.SAMP
0MBR8   =OR+PWRON+MAMO
```

TABLE I
Input Channel Term Definition

TERM	FLIP FLOP	DEFINITION
MCIC1	Y	Mode Control
MCIC2	Y	Mode Control
MCIC3	Y	Mode Control
MCIC4	Y	Mode Control
MCIC5	Y	Mode Control
MCIC6	Y	Mode Control
NEXBYT	Y	Signal to computer requesting the next byte of data
PARBA	N	Logic signal indicating bad parity on byte received from conputer
PAR2	Y	Stores parity generated on input byte
INPR1	N	Intermediate results in the generation of odd parity for the incoming byte
INPR2	N	
INPR3	N	
INPR4	N	
INPR5	N	
INPR6	N	
BRA1 through BRA8	Y	Register to store the data received from the computer
IN1 through IN8	N	The input data lines from the computer
IN9	N	Parity input line from the computer
DATST	N	Data strobe line from the computer
CØBRB	N	Indicates when the contents of BRA are to be loaded into BRB
BRB1 through BRB8	Y	Buffer register used to transfer data from the input channel to the voter or matrix sections
PAR4	Y	Holds the parity bit for use by the voter section
PRBAD	Y	Signal to the computer indicating bad parity on last transmission and requests repeat transmission.
BRBF	Y	Indicates when the BRB register holds valid data for the voter section use.
ADDVAL	Y	Indicates whether the address of the control byte received is the correct one.
PAR1	Y	Stores the parity bit sent over by the computer
PAR3	Y	Temporary store for parity received on even numbered bytes from the computer

TABLE I (cont.)

TERM	FLIP FLOP	DEFINITION
END	Y	Stores the end of message signal from the computer
DONE	Y	Generates the end of operation signal to the voter section
ENDMS	N	End of message signal from the computer
MI	Y	Generates the matrix interrupt to the matrix section
COMNO	N	Identifies the status of the interfacing computer as defined by the computer self test capability
EOM	N	Identifies valid end of message signal

TABLE II
Voter Section Term Definition

TERM	FLIP FLOP	DEFINITION
BCVS1 through BCVS4	Y	Bit Counter
RESET	N	Identifies the reset mode for the voter section.
TIM1 through TIM4	Y	Counter to time start of voting
VBR	Y	Stores the results of the voting that are to be sent out to the LP.
VSMD1	N	Identifies when the data word is being voted upon.
MCVS1 through MCVS5	Y	Mode Control
STA1 through STA8	Y	Generates and stores the status of the data from each input channel with respect to the voter output (VBR).
DONE	N	Identifies when the voting operation is ended.
MAJSET	N	Identifies when the majority of the BRBF signals from the input channels involved in the voting are set true.
CDNMET	N	Indicates start of voting for three-way or four-way modes.
VOTOUT	N	Indicates to the output channel when a voting mode is to begin.
EITSET	N	Starts timer in comparator mode.
MSVA, MCV5A	N	Identifies when the status register can be read into S matrix.
VSMD2	N	Identifies when voting on parity is done during comparator three-way or four-way voting modes.
VSMD3	N	Identifies when votingon data words is done during a selector mode.
VSMD4	N	Identifies when voting on parity is done during a selector mode.
TKVOTE	N	Identifies when voting is done for comparator, three-way and four-way voting modes.
OUSLCT	N	Identifies when voting is being done for a selector voting mode.
TIMOUT	N	Identifies when the counter timing the start of voting has overflowed.
LDBRB	N	Identifies when new data are to be placed in buffer register B.

TABLE II (Cont.)

TERM	FLIP FLOP	DEFINITION
BOTH	N	Identifies when the condition for voting during a comparator mode has been met.
SHFTA, SHFTB SHFTC, SHFTD	N	Identifies when the data in buffer register B of the input channel is to be shifted.
CYCLE	Y	Counts data bytes that have been processed by the voter.
MCVSB	N	Identifies when voting has been finished on an eight bit byte.
TKSTAT	N	Identifies when status due to the voting process is valid.
SIVBR	N	Identifies conditions to be met for agreement during voting.
SOVBR	N	Identifies conditions to be met for agreement during voting.

TABLE III

Output Channel Term Definition

TERM	FLIP FLOP	DEFINITION
MCOC1 through MCOC5	Y	Mode Control
RDATA	N	Sets bit counter to one after receiving sync code.
ODTST	Y	Data strobe for sending LP data to all computers
MS	Y	Matrix strobe for sending matrix data to the computer
MSA	N	Individual strobe line to each computer used for sending matrix data to a single computer
MSB	N	
MSC	N	
MSD	N	
OR	Y	Indicates readiness of output channel to the voter and matrix section for receiving data.
SP1 through SP8	Y	Serial to parallel converter
OBR1 through OBR8		Buffer register to store data being sent to the computers
RESP	N	Indicates when the contents of the serial to parallel converter are to be read into the output buffer register.
RMBR	N	Indicates when the contents of the matrix buffer register are to be read into the output buffer register.
MADEC1	Y	Data from the LP data bus receiver
MADEC2	Y	
OCP1	Y	Parity generator for serial data
PAROUT	Y	Parity signal to the computer
BCOC1 through BCOC4	Y	Bit counter for serial to parallel conversion
RESETA	N	Indicates the reset mode
OCP2	Y	Parity generator for parallel data
OUTPR1 through OUTPR6	N	Intermediate results for parallel parity generation.
RESETC	N	Indicates when a matrix operation must be terminated due to an LP operation using the output channel.
PRES	N	Indicates when a signal is present on the data line from the LP

TABLE III (Cont.)

TERM	FLIP FLOP	DEFINITION
MS1, MS2	Y	Register to generate the proper matrix strobe to the desired computer

TABLE IV
Matrix Section Term Definition

TERM	FLIP FLOP	DEFINITION
MCMS1 through MCMS5	Y	Mode Control
R1 through R16	Y	R Matrix
COPYR	N	Indicates when the contents of the matrix buffer register are to be copied into the R Matrix.
CPYR0	N	Indicates which flip flops of the R Matrix are to receive the data from the MBR.
CPYR1	N	
CPYR2	N	
CPYR3	N	
MI0	N	Indicate the four states of the matrix interrupt scanning counter.
MI1	N	
MI2	N	
MI3	N	
P1 through P16	Y	P Matrix
COPYP	N	Indicates when the contents of the MBR are to be read into the P Matrix.
CPYP0	N	Indicates which flip flops in the P Matrix are to receive the data from the MBR.
CPYP1	N	
CPYP2	N	
CPYP3	N	
S1 through S16	Y	S Matrix
SI1	N	Identifies the particular status bit being read from the voter section.
SI2	N	
SI3	N	
SI4	N	
MODEA	N	Identifies the voting mode during which the status bits were determined.
MODEB	N	
MODEC	N	
MODED	N	

TABLE IV (Cont.)

TERM	FLIP FLOP	DEFINITION
MAMO	N	Identifies the reset mode
COPYS	N	Identifies when the S Matrix flip flops are to be zero set.
CPYSO	N	Identify which flip flops of the S Matrix are to be zero set
CPYS1	N	
CPYS2	N	
CPYS3	N	
V4WAY	N	Identifies the four-way voting mode
V3WAYA	N	Identifies the three-way voting mode using input channels A, B and C.
V3WAYB	N	Identifies the three-way voting mode using input channels A, B and D.
V3WAYC	N	Identifies the three-way voting mode using input channels A, C and D.
V3WAYD	N	Identifies the three-way voting mode using input channels B, C and D.
COMPA	N	Identifies the comparator voting mode using input channels A and B.
COMPB	N	Identifies the comparator voting mode using input channels A and C.
COMPC	N	Identifies the comparator voting mode using input channels A and D.
COMPD	N	Identifies the comparator voting mode using input channels B and C.
COMPE	N	Identifies the comparator voting mode using input channels B and D.
COMPF	N	Identifies the comparator voting mode using input channels C and D.
SLCTA	N	Identifies selector voting mode using input channel A.
SLCTB	N	Identifies selector voting mode using input channel B.
SLCTC	N	Identifies selector voting mode using input channel C.
SLCTD	N	Identifies selector voting mode using input channel D.

TABLE IV (Cont.)

TERM	FLIP FLOP	DEFINITION
POR1	N	Identifies failed or don't care settings of the P and R Matrices.
POR2	N	
POR3	N	
POR4	N	
OREQ	Y	Indicates that valid data are in the MBR ready for use by the output channel.
INSRV	Y	Indicates that the matrix interrupt has been honored.
INSRVA	N	Identifies the particular matrix interrupt being honored.
INSRVB	N	
INSRVC	N	
INSRVD	N	
NEWMOD	Y	Indicates a change of mode due to new data being placed in the R Matrix.
MIC1 through MIC4	Y	Matrix interrupt scanning counter
OPC1 through OPC3	Y	Register for storing matrix operation codes
CPYOC1	N	Identifies the input channel that is to supply the matrix operation code.
CPYOC2	N	
CPYOC3	N	
CPYOC4	N	
MBR1 through MBR8	Y	Matrix buffer register for data coming into or going out of matrix section
READA	N	Identifies which input channel BRB is to be read into the MBR.
READB	N	
READC	N	
READD	N	
MAM3	N	Identifies when a BRB of an input channel is to be read into the MBR.
OUTRA	N	Identifies the two eight bit bytes that represent the contents of the R Matrix.
OUTRB		
OUTPA	N	Identifies the two eight bit bytes that represent the contents of the P Matrix.
OUTPB		
OUTSA	N	Identifies the two eight bit bytes that represent the contents of the S Matrix.
OUTSB		

TABLE IV (Cont.)

TERM	FLIP FLOP	DEFINITION
DESTA	N	Identifies the computer requesting the matrix data.
DESTB	N	
DESTC	N	
DESTD	N	
MAM5	N	Identifies when the computer identification terms are valid.
AUSED	N	Identifies which computer is being used in the voting mode being performed
BUSED	N	
CUSED	N	
DUSED	N	
CLOPC	N	Identifies when the operation code register is to be cleared to all zeros.
MI	N	Identifies the arrival of a matrix interrupt signal from an input channel
SAMP	N	Identifies when the P matrix diagonal and operating mode register states are to be copied into the matrix buffer register.
OM1 through OM4	Y	Operating mode register to store the information as to which computers are defined by the P and R matrices to be involved in the boting being performed.

APPENDIX B

VCS FLOW DIAGRAMS

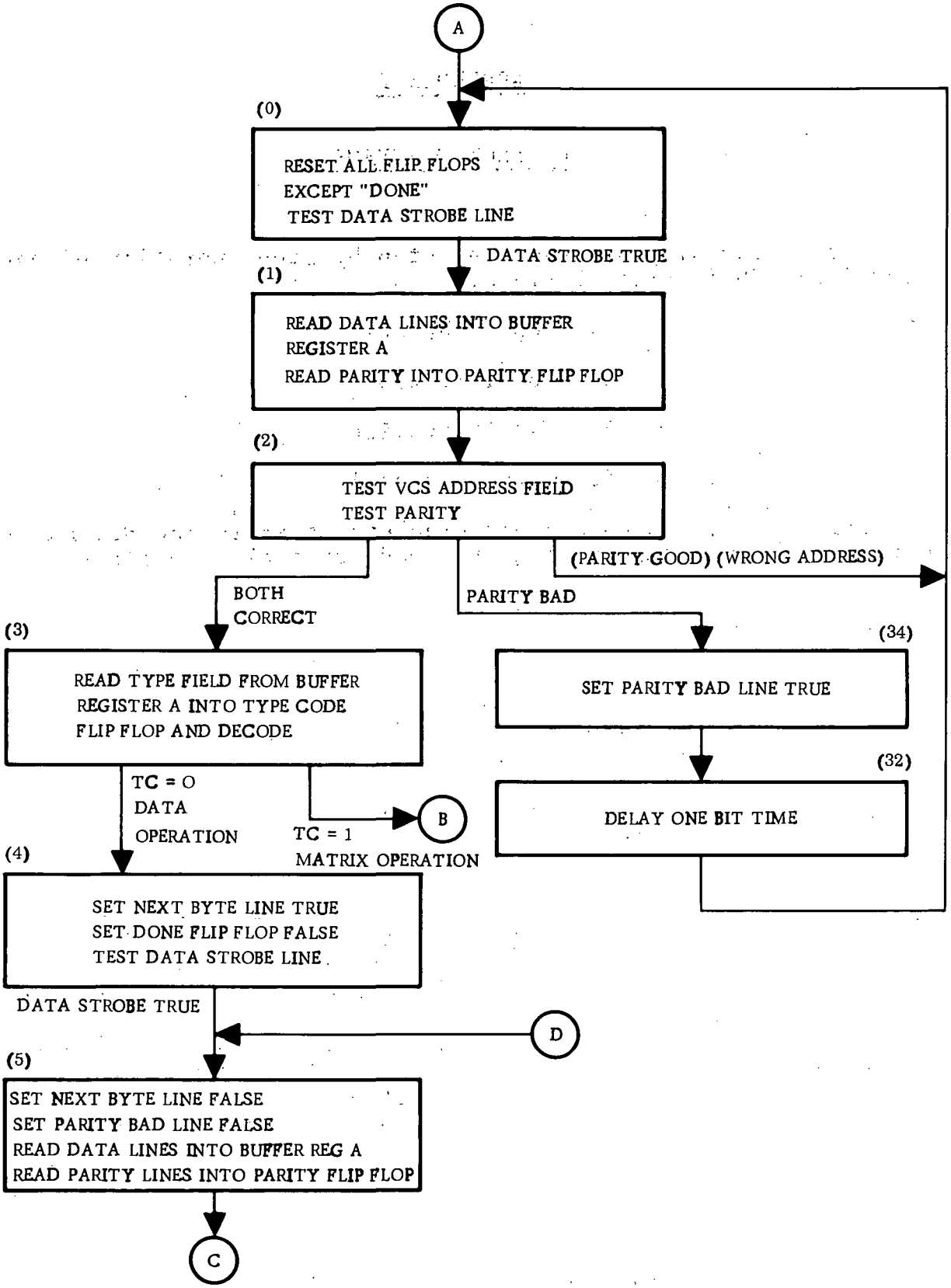
APPENDIX B

VCS FLOW DIAGRAMS

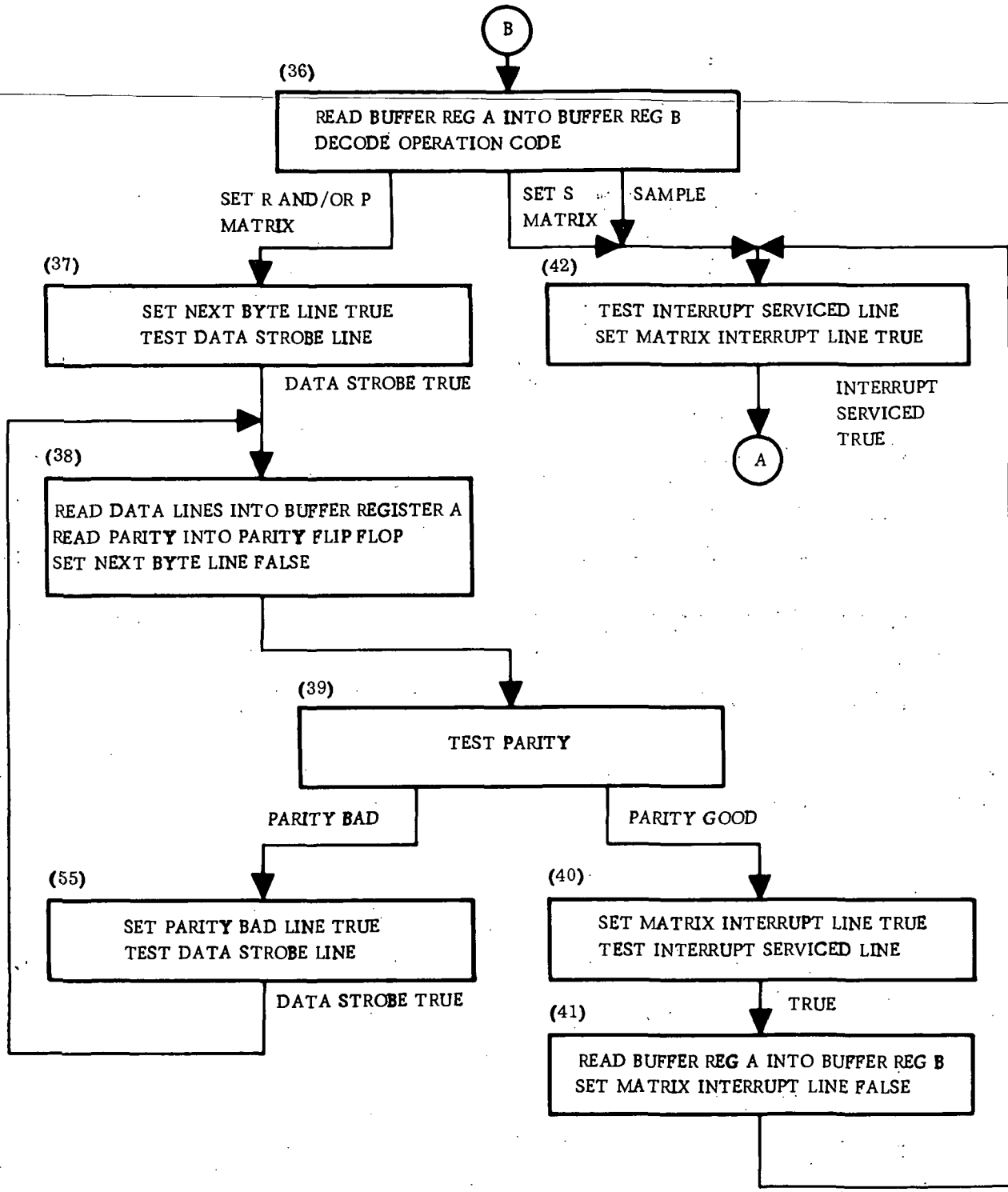
The following diagrams show the step by step operation of each of the four VCS functional areas:

1. Input Channel
2. Voter Section
3. Matrix Section
4. Output Channel

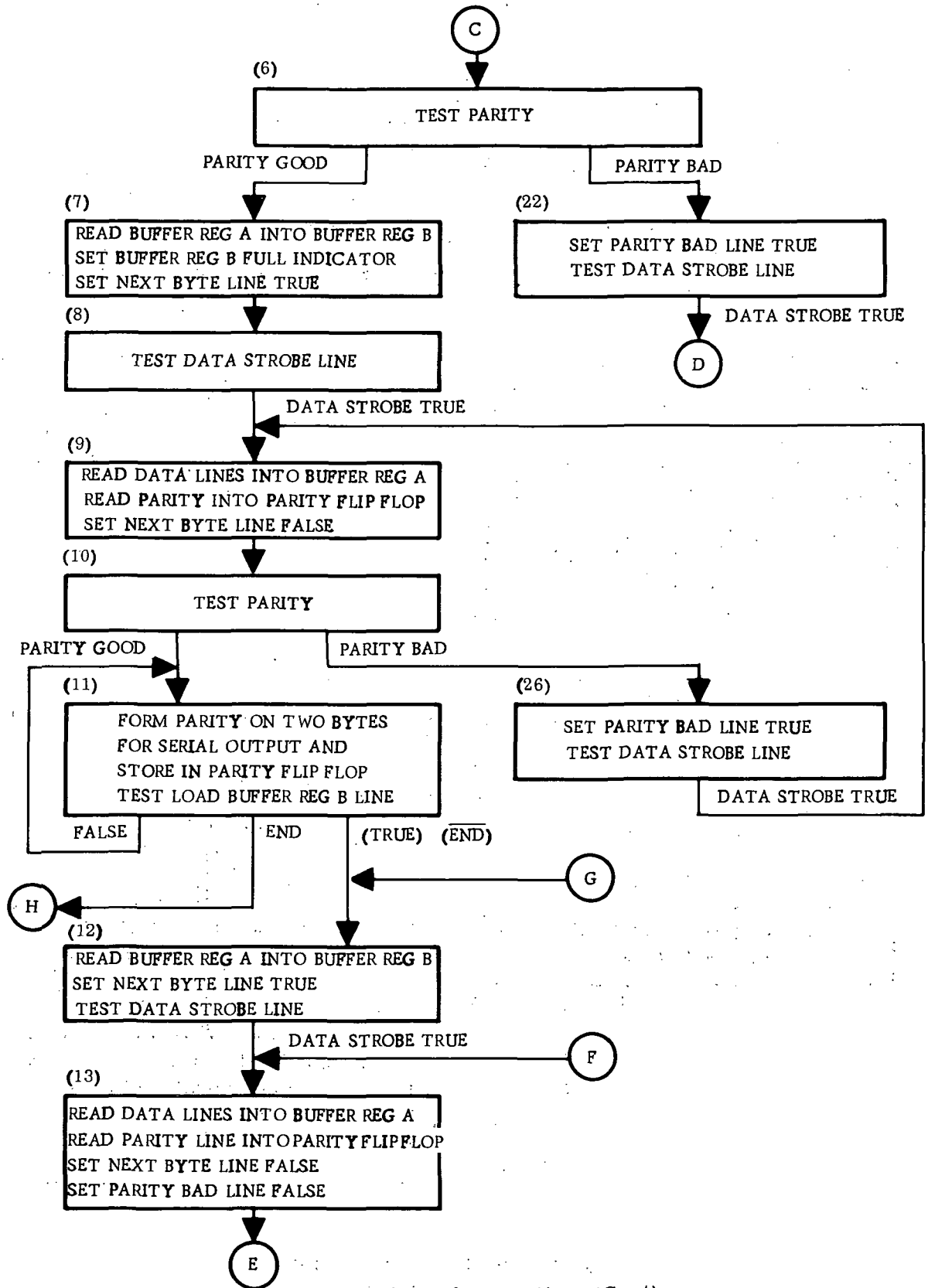
On each chart, the numbers in brackets refer to the decimal equivalent of the binary states of the mode control flip flops.



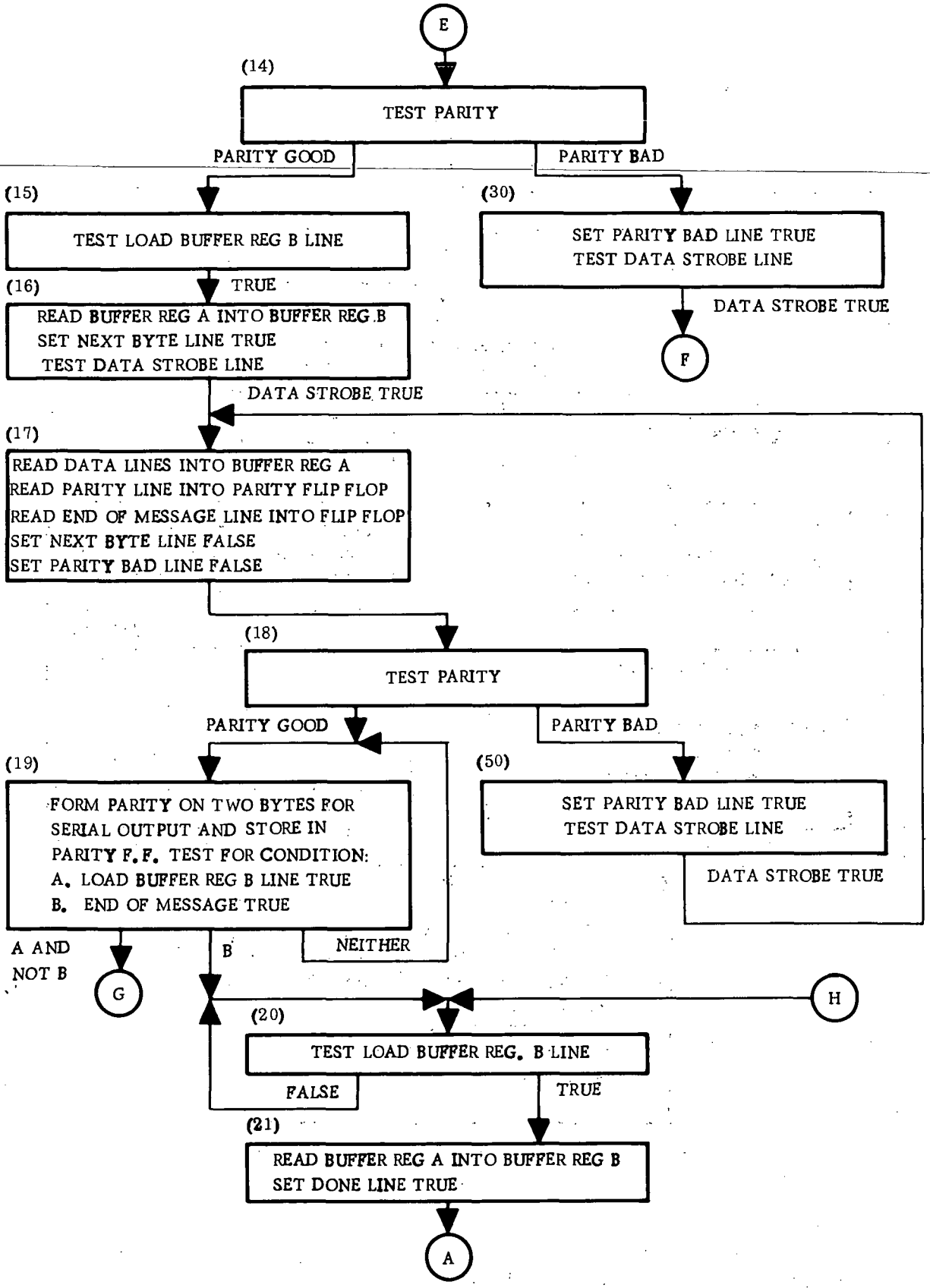
Input Channel Operation



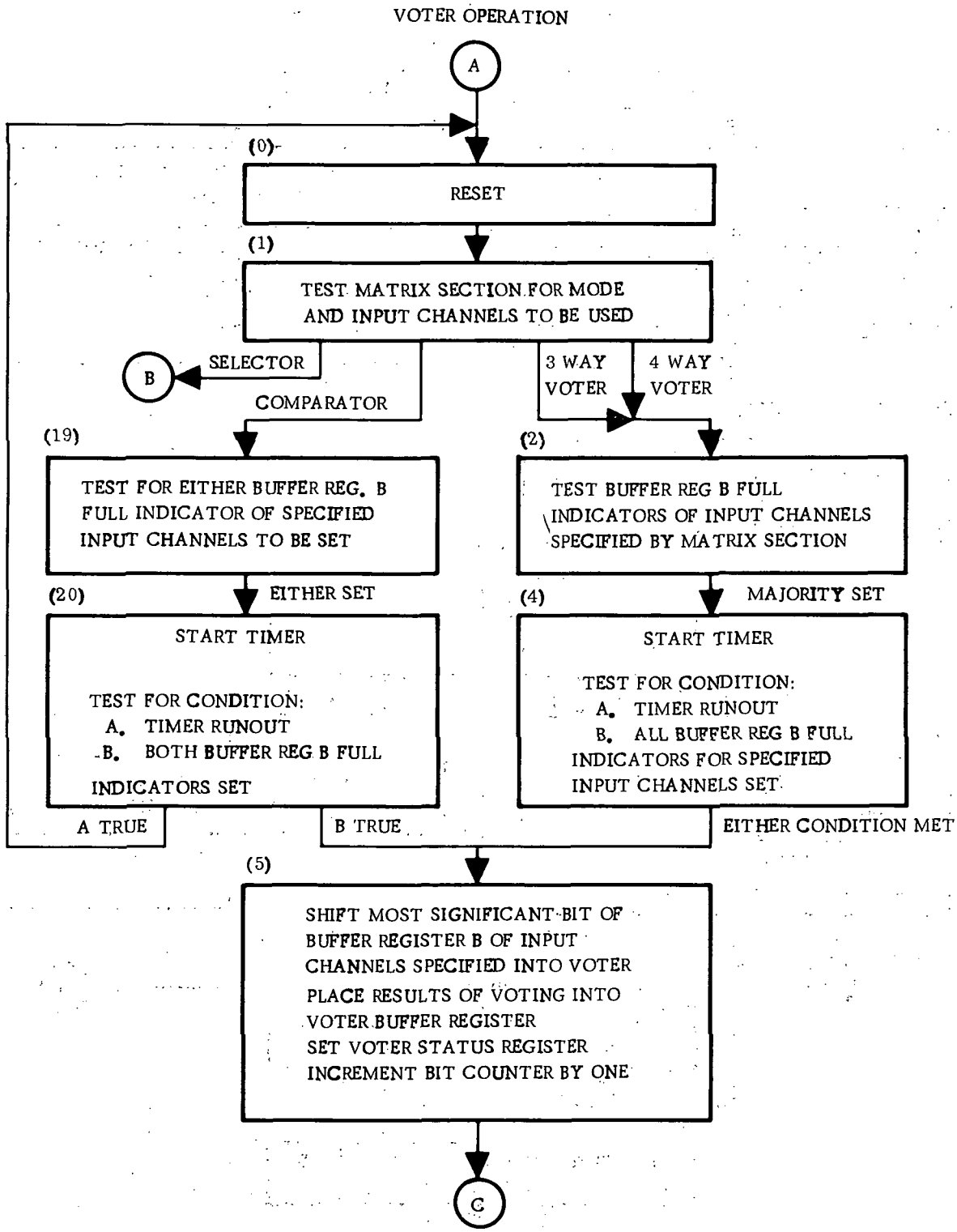
Input Channel Operation (Cont)



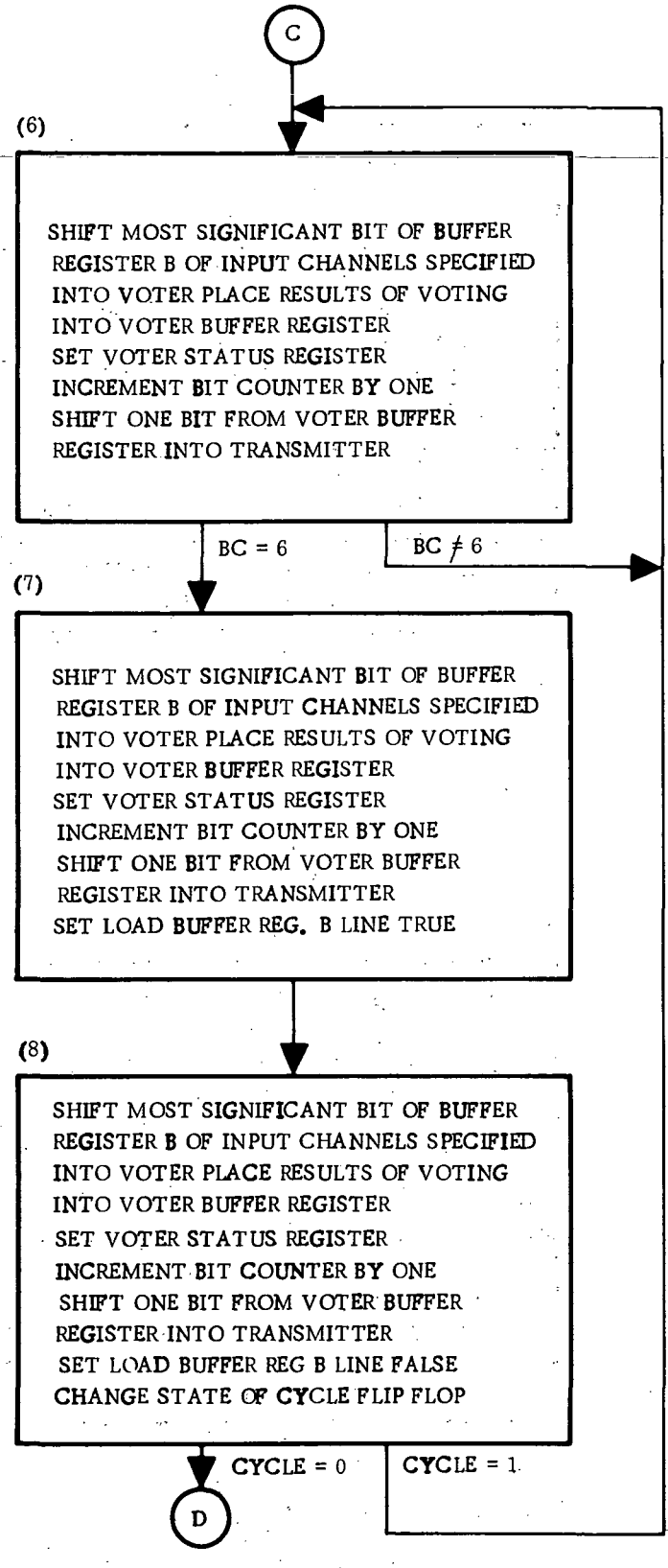
Input Channel Operation (Cont)



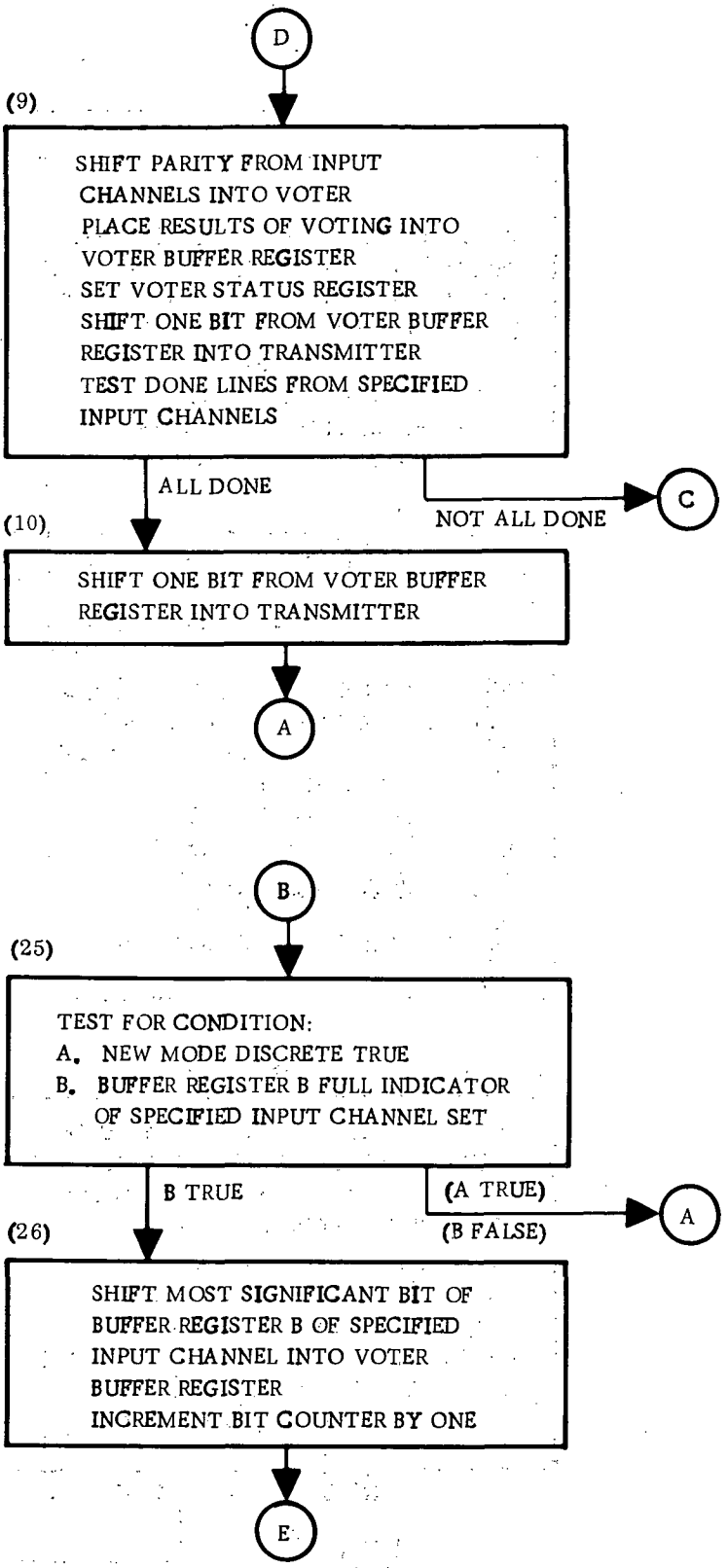
Input Channel Operation (Cont)



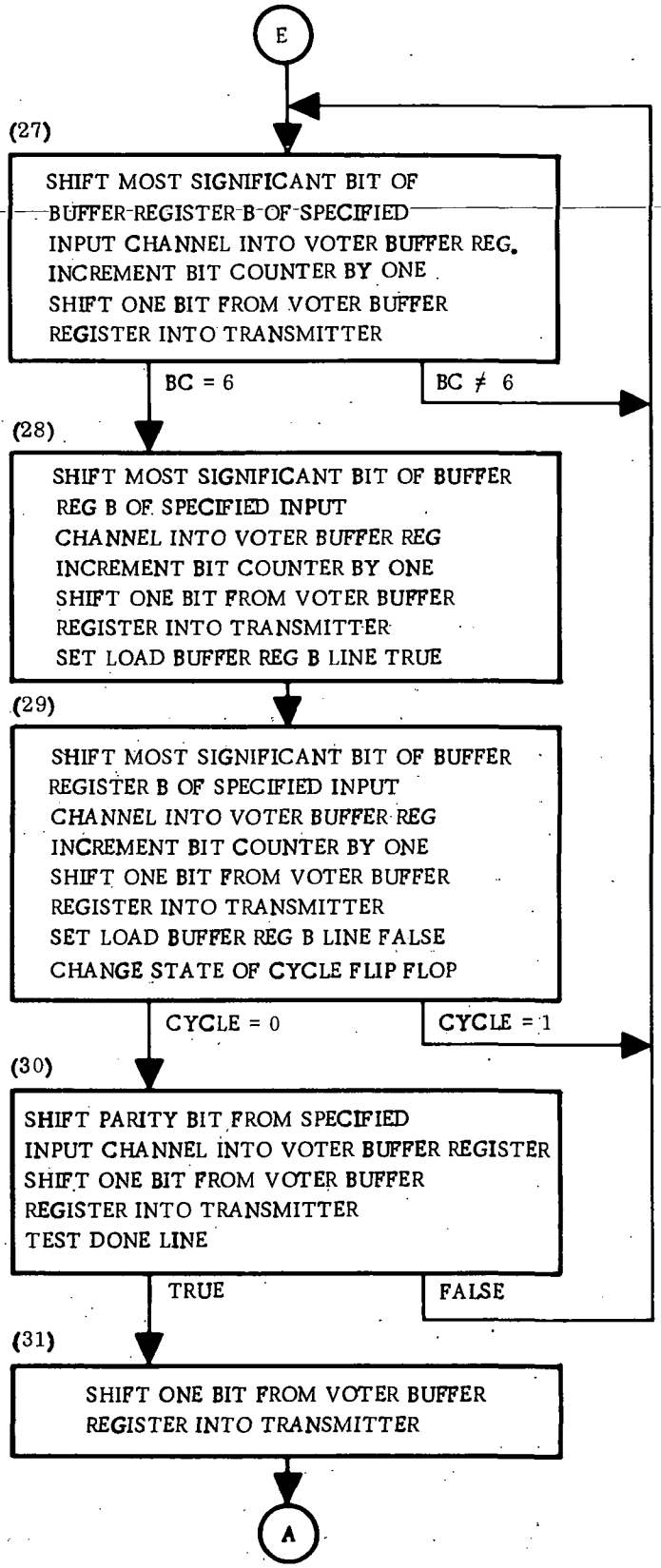
Voter Operation



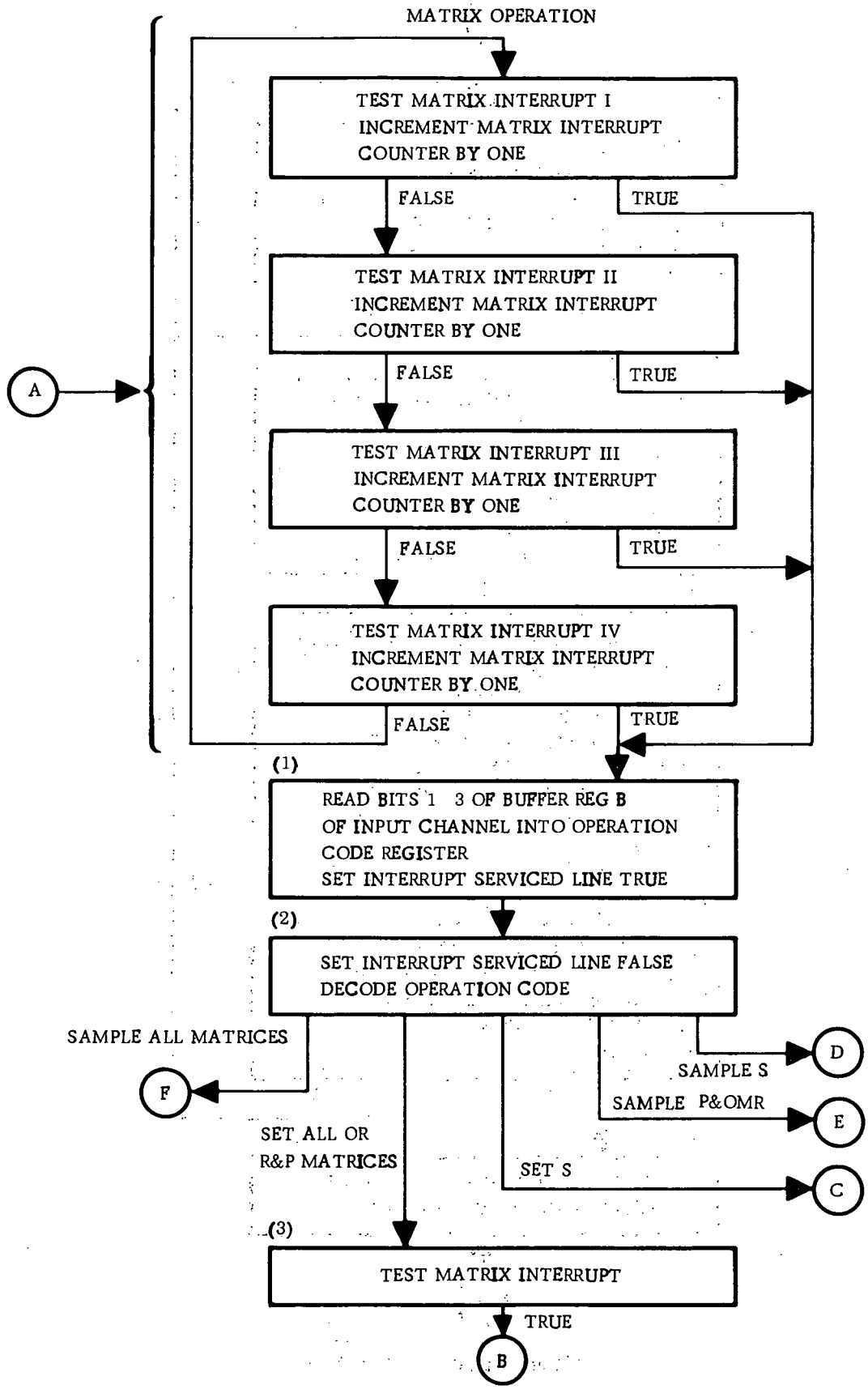
Voter Operation (Cont)



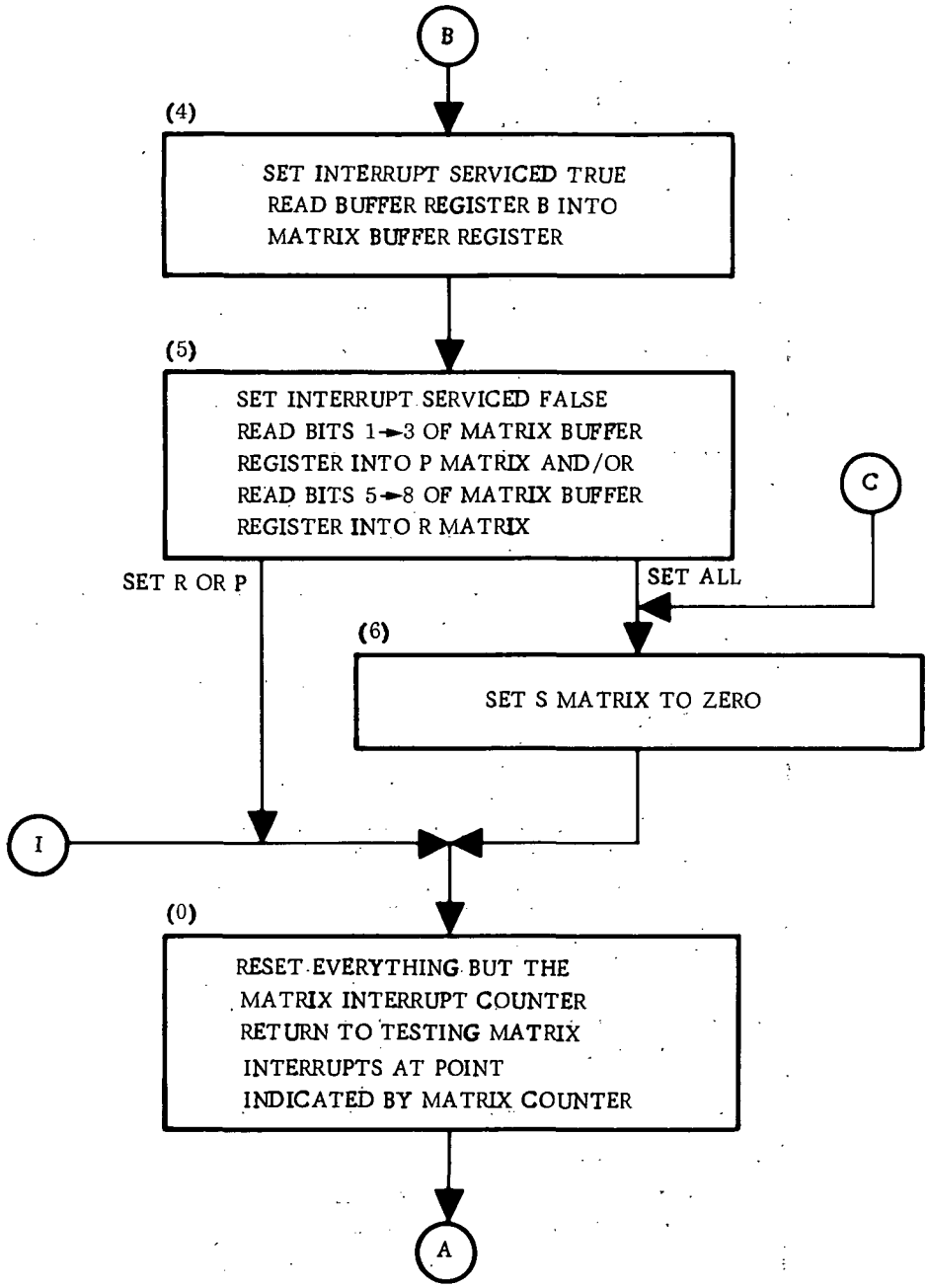
Voter Operation (Cont)



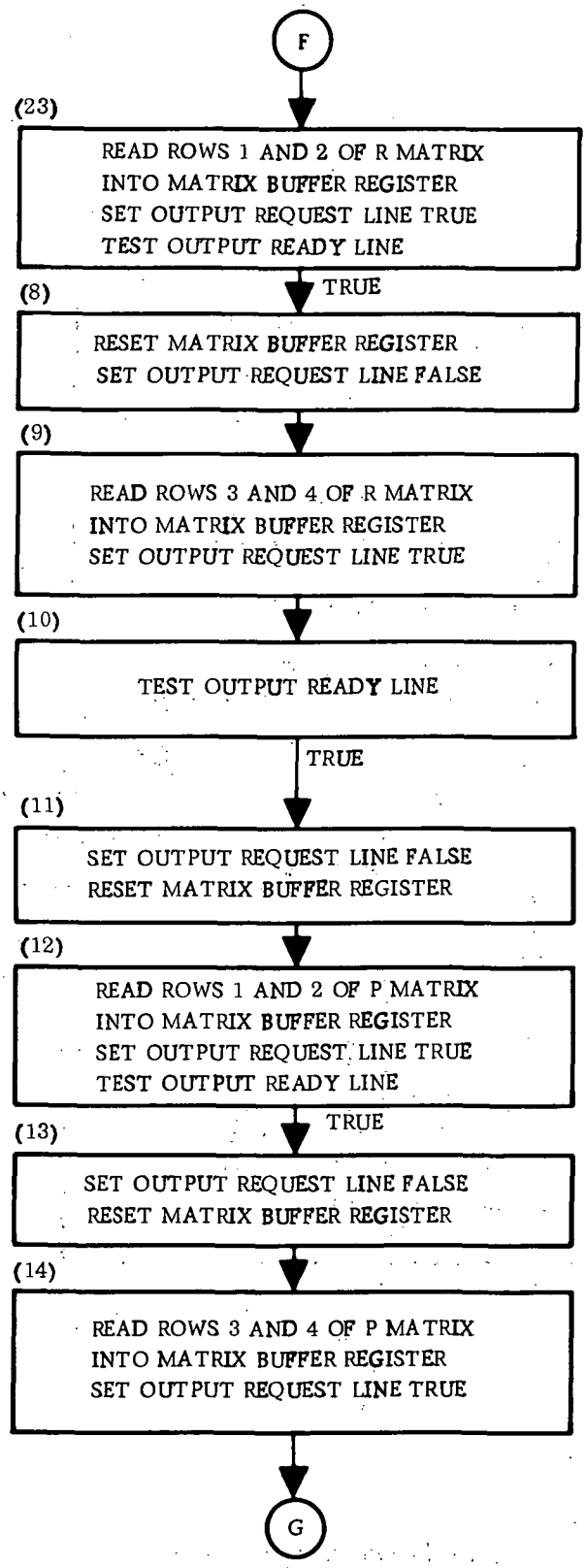
Voter Operations (Cont)



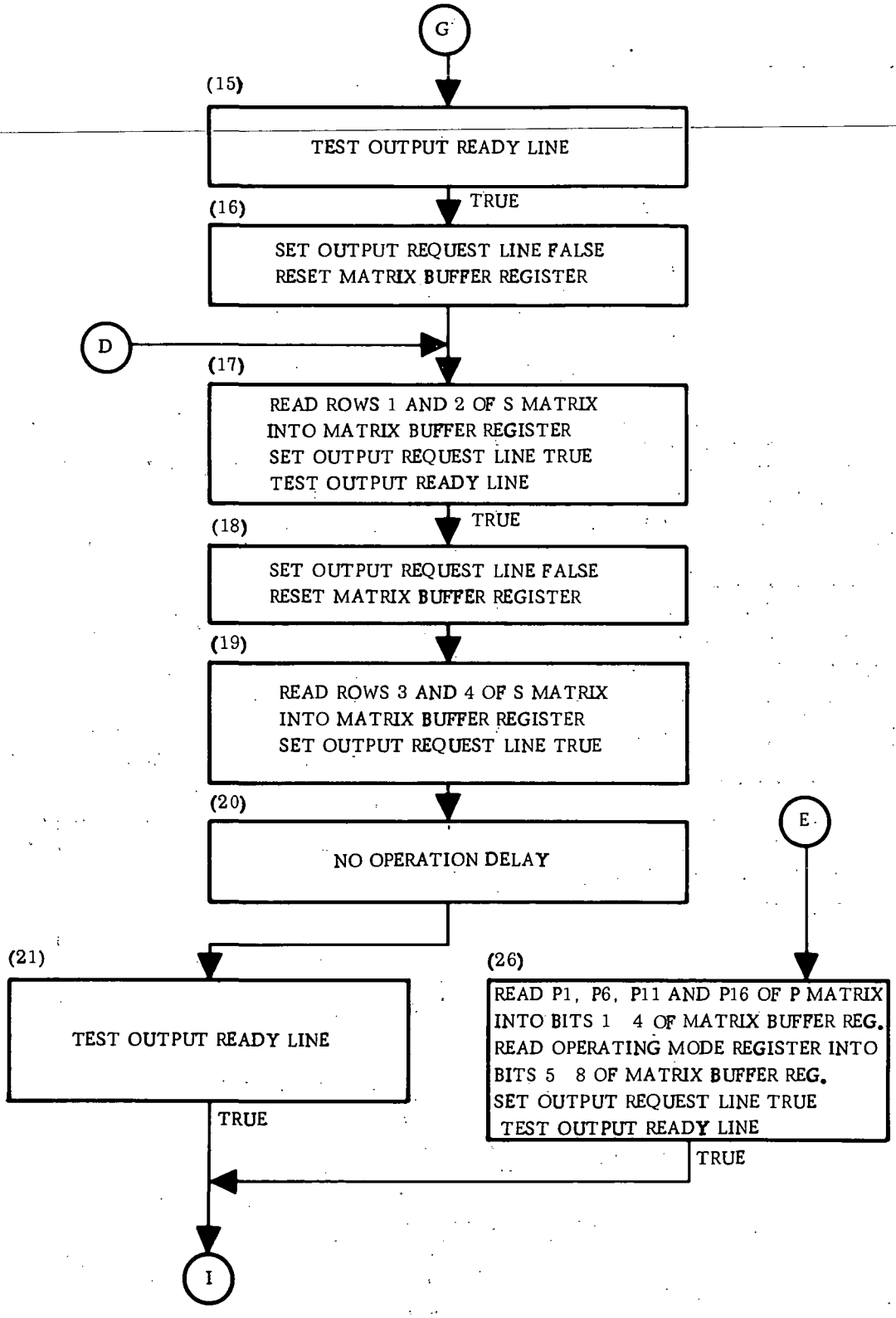
Matrix Operation



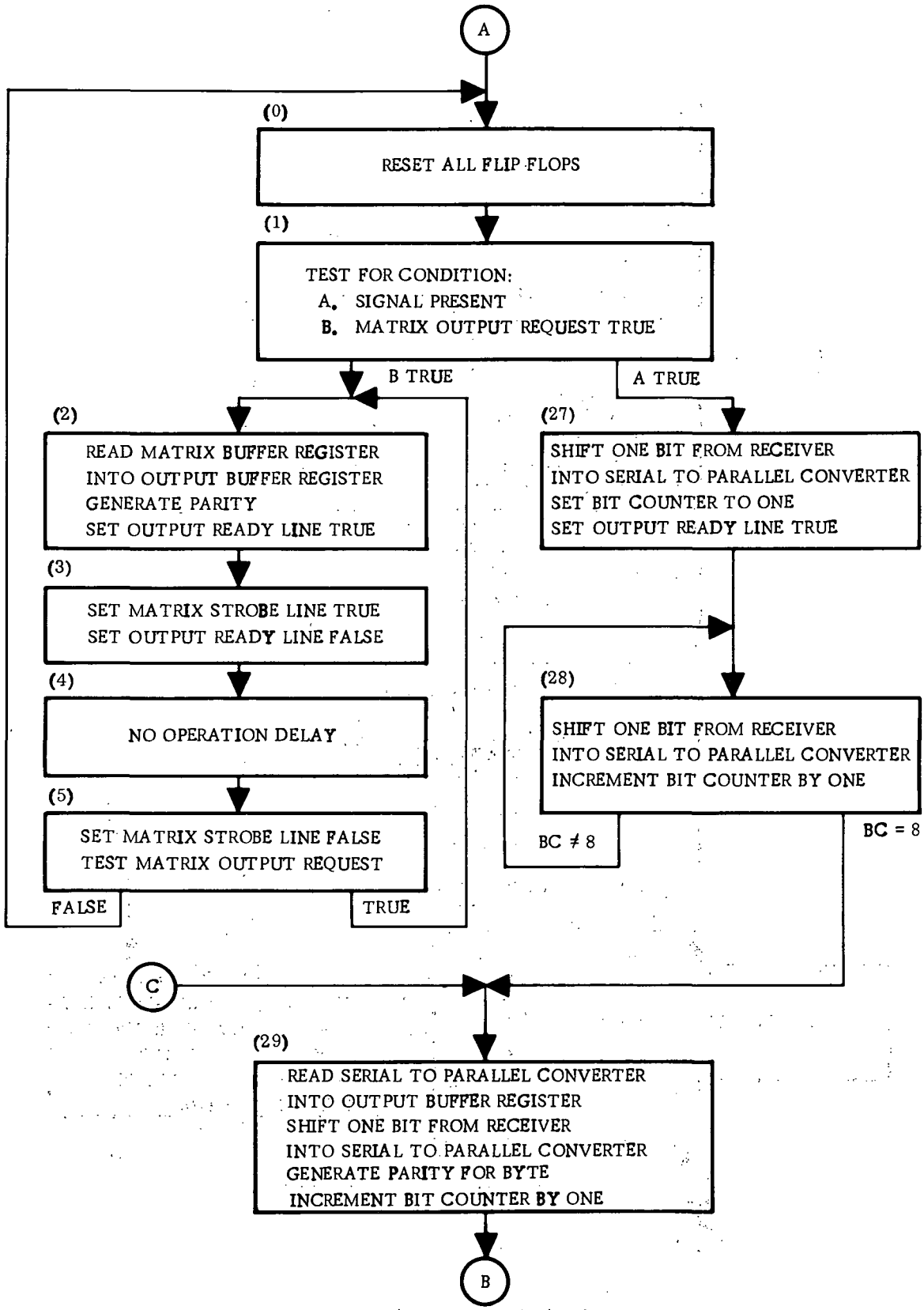
Matrix Operation (Cont)



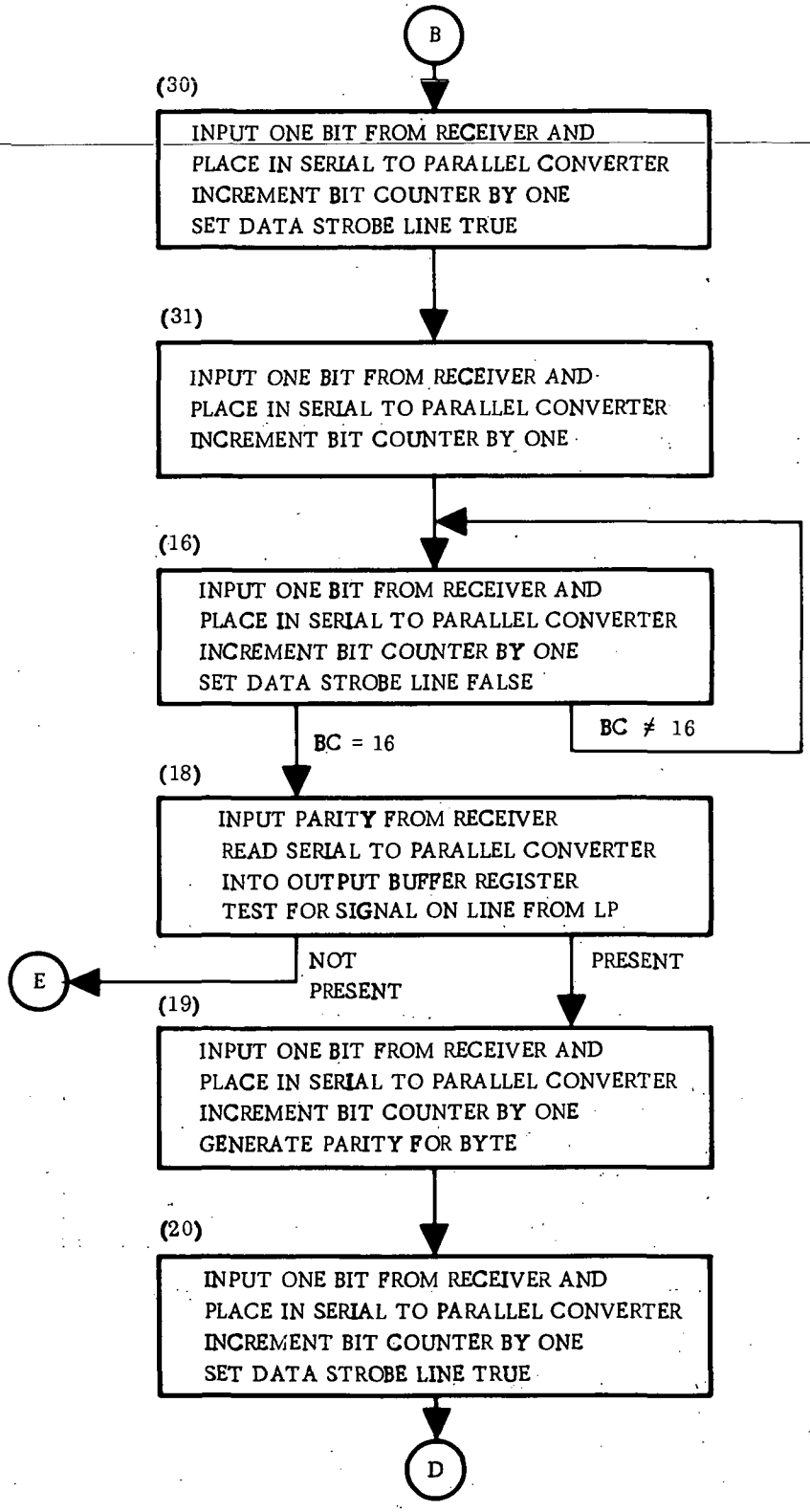
Matrix Operation (Cont)



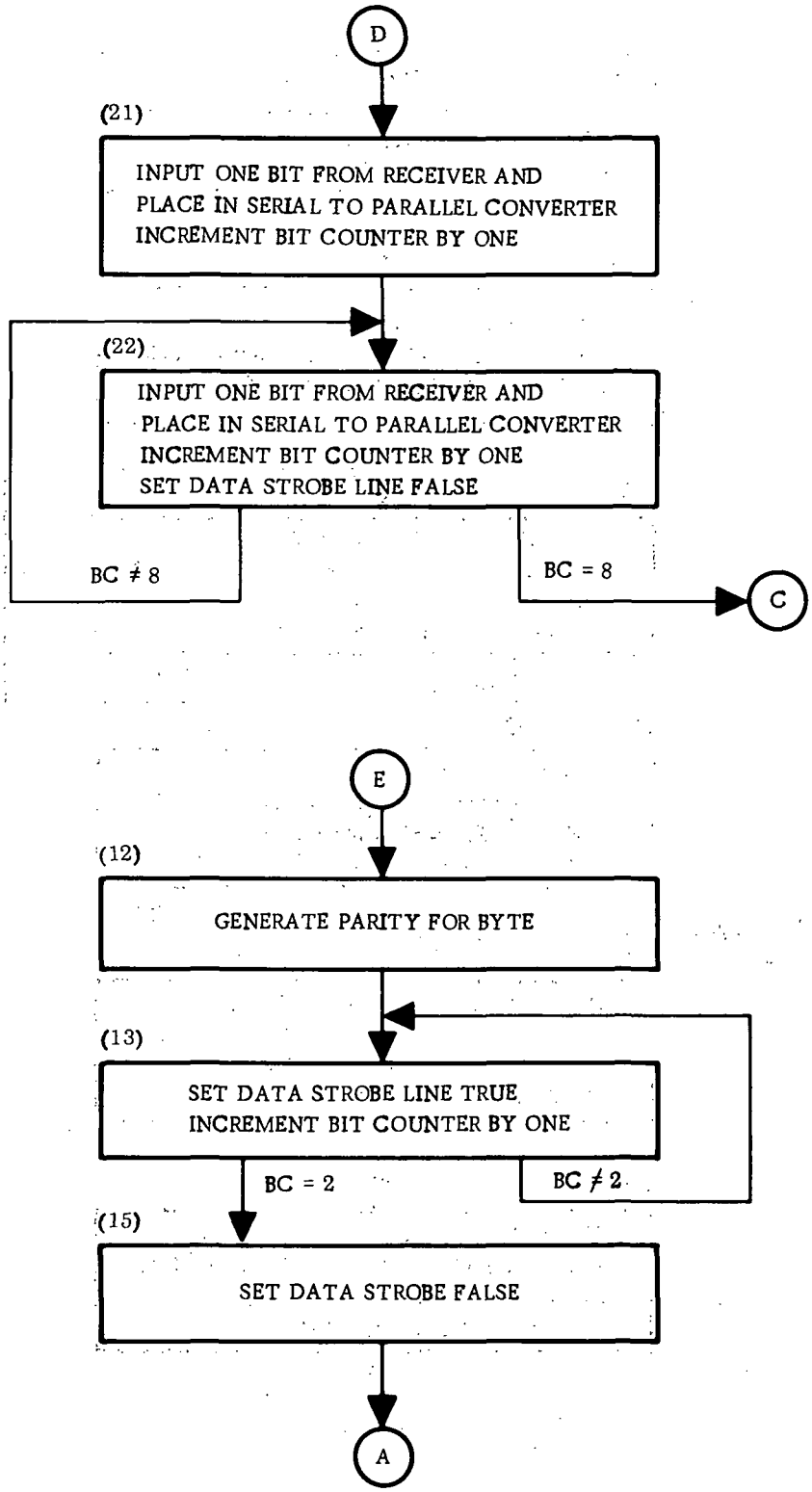
Matrix Operation (Cont)



Output Channel Operation



Output Channel Operation (Cont)



Output Channel Operation (Cont)

APPENDIX C

VCS MODEL SPECIFICATION

APPENDIX C

VCS MODEL SPECIFICATION

1.0 SCOPE

This document establishes the requirements for the performance of a voter-comparator-switch (VCS) for application in fault tolerant data handling systems.

1.1 REQUIREMENTS

The VCS shall be capable of performing various types of voting operations upon digital data received from up to four data sources. The voting modes shall be commanded by the data sources and shall be modified as a function of the failure detected in the data sources.

2.0 ITEM DEFINITION

The VCS shall be made up of five basic functional areas:

1. Input channel
2. Matrix section
3. Voter section
4. Output channel
5. Clock generator

The normal configuration would be one of each section and four input channels. The number of input channels used can vary from one to four.

2.1.1 ITEM DIAGRAM

The relationship of the VCS to other equipment is shown in Figure C-1.

2.1.2 INTERFACE DEFINITION

2.1.2.1 INPUT POWER

The VCS shall be designed to operate from two regulated DC voltage levels:

$+5 \pm 0.5$ VDC
-20 ± 2 VDC

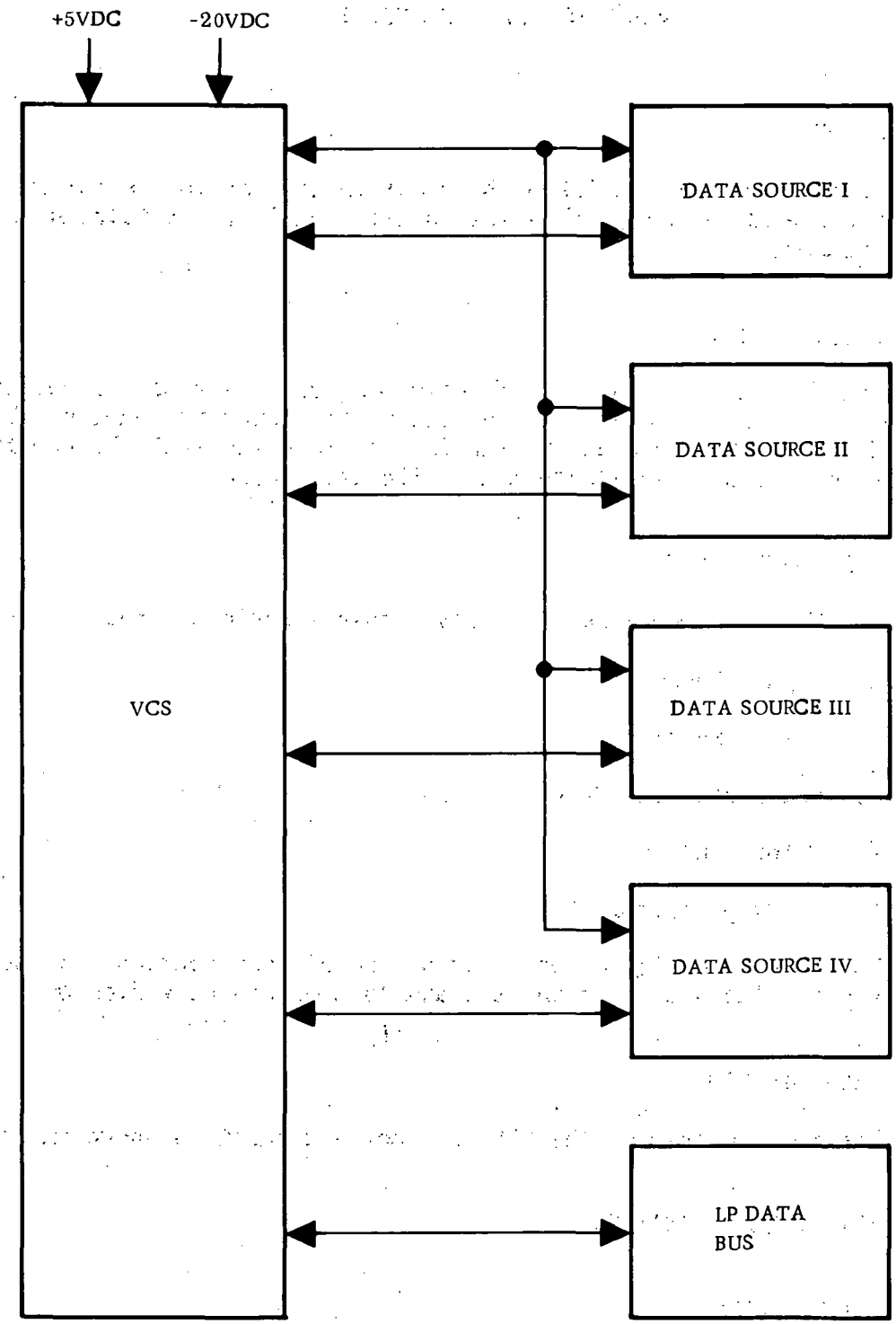


Figure C-1. VCS Interface

2.1.2.1.1 Power Interruption

The VCS shall not be damaged by the interruption of the input power for any time period. During application of power, the VCS shall be forced into an initialize mode with all outputs held in the false state.

2.1.2.2 Signal Interface

The signal interface of the VCS with other equipment shall be TTL compatible. VCS outputs shall be capable of driving up to five standard TTL loads. VCS inputs shall present one standard TTL load.

2.2 CHARACTERISTICS

2.2.1 BLOCK DIAGRAM

A block diagram of the VCS is shown in Figure C-2.

2.2.2 INPUT CHANNEL

The input channel shall provide the interface between the data source and the VCS for data transfers to VCS.

2.2.2.1 Operation

The input channel shall operate under the control of the data source. The data source will command starting and stopping of operations. Data are received by the input channel as nine bit bytes in bit parallel and byte serial fashion. The first byte transferred during an operation shall be a control byte. Each byte will contain odd parity indication. The input channel shall perform all operations according to the logical equations in Paragraph 3.1.

2.2.2.2 DATA FORMAT

The input channel shall be capable of accepting and handling control and data bytes. The control byte is comprised of the following:

Bit Position in Byte	Field	Definition
1, 2, 3	Operation	These three bits specify the matrix operation to be performed
4	Type	This bit specifies a voting operation (binary zero) or a matrix operation (binary one).
5, 6, 7, 8	VCS Address	These four bits specify the VCS being accessed
9	Parity	This bit forms odd parity for the byte

The data byte consists of eight bits of data and one bit for odd parity.

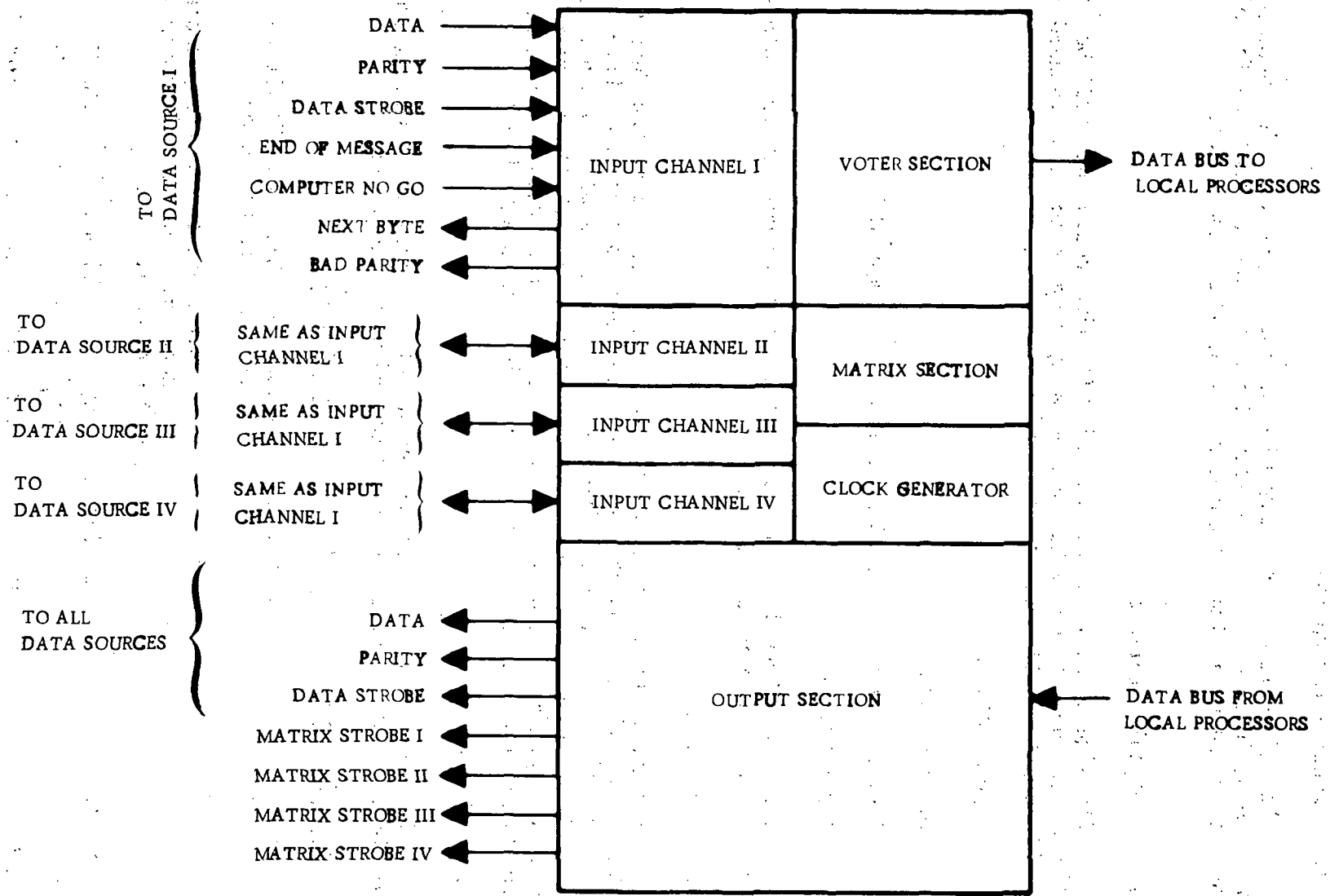


Figure C-2. VCS Functional Diagram

2.2.2.2.1 Operation Field Codes

The operation field of the control byte shall have the following definition:

Bit	Operation to be Performed
3 2 1	
0 0 0	Set R and P matrices to value; set S matrix to zero
0 0 1	Set R and P matrices to value
0 1 1	Set S matrix to zero
1 0 0	Sample all matrices
1 1 0	Sample P matrix diagonal and operating mode register
1 1 1	Sample S matrix

2.2.2.3 DATA SOURCE INTERFACE

The interface between the input channel and the data source shall consist of the following signals:

1.	Data	8
2.	Parity	1
3.	Data Strobe	1
4.	End of Message	1
5.	Computer No-Go	1
6.	Next Byte	1
7.	Bad Parity	1

2.2.3 MATRIX SECTION

2.2.3.1 FUNCTION

The matrix section shall define the voting mode of the voter section according to the contents of the P and R matrices. The S matrix shall be set according to the status signals from the voter section. All matrices shall be capable of being set or sampled by the data sources in a random fashion. The matrix section shall perform all operations according to the logical equations of Paragraph 3.2.

2.2.3.2 ORGANIZATION

The matrix section shall consist of the voting mode (R) matrix, the data source status (P) matrix, the voting status (S) matrix, a buffer register, and a mode control.

2.2.3.2.1 R Matrix

The R matrix shall consist of 16 data bits, four bits set by each computer in the complex. These 16 bits shall determine the operating mode of the VCS and which computers are involved. The mode determining logic shall be modified by the P matrix. A mode shall be entered if a majority of the non-failed computers input identical four bit codes to the R matrix. The modes are:

a. Four way voting - data from all four computers must be compared in the voter to determine a majority.

b. Three way voting - data from three computers shall be compared in the voter. The computers involved shall be indicated by the presence of ones in the matrix.

c. Two way voting - data from two computers shall be compared in the voter. The computers involved shall be indicated by the presence of ones in the matrix.

d. No voting - one input channel is connected to the LP data bus.

2.2.3.2.2. P Matrix

The P matrix shall consist of 16 bits, four set by each computer. These bits shall represent the operating conditions of all computers as determined by each computer. A binary zero in any position of the matrix shall indicate a failed computer as determined by another computer. The matrix takes the form as follows:

Computer Doing the Fault Determination

		I	II	III	IV
Computers Being Studied	I	*			
	II		*		
	III			*	
	IV				*

*Indicate the diagonal positions

A binary zero in a diagonal position as shown shall indicate that that computer is failed and its outputs shall be disregarded. The failed computer's entry in the R matrix shall be ignored in the mode determination. The diagonal terms shall be set to zero as follows:

a. The computer no go discrete is true or

b. A majority of the non-failed computers have indicated the computer is bad by the entries in the P matrix

2.2.3.2.3 S Matrix

The S matrix shall consist of 16 bits, four for each computer. This matrix shall indicate the status of the data received by the VCS according to the voting process. Binary ones shall be placed in the matrix to indicate that a computer's data did not agree with the majority data from the voter.

2.2.4 Voter Section

The voter section shall compare the data from the input channels according to the P and R matrices. Data meeting the requirements of the voting mode are placed on the LP data bus. The voting modes are those described in paragraph 2.2.3.2.1. The voter section shall perform according to the logical equations of paragraph 3.3.

2.2.5 Output Channel

2.2.5.1 Function

The output channel shall be the formatting and interface capability for data sent to the data sources from the VCS and for data sent to the VCS over the LP data bus. The output channel shall be capable of transferring data from the matrix section to the data source. The output channel shall operate according to the logical equations of paragraph 3.4.

2.2.5.2 Data Format

The output channel shall receive serial data consisting of seventeen bits per word in multiple word messages over the LP data bus. The seventeen bits include sixteen data bits and one parity bit for odd parity. The output channel shall send data to the data sources as nine bit bytes in bit parallel and byte serial fashion. Each byte shall have eight bits of data and one parity for odd parity.

2.2.5.3 Data Source Interface

The interface between the data source and the output channel shall consist of the following signals:

1.	Data	8
2.	Parity	1
3.	Data Strobe	1
4.	Matrix Strobe I	1
5.	Matrix Strobe II	1
6.	Matrix Strobe III	1
7.	Matrix Strobe IV	1

2.2.6 Clock Generator

The clock generator shall be capable of accepting a square wave signal having a frequency of nine megahertz and generating a four phase clock as shown in Figure C-3.

3.0 Logical Equations

3.1 Input Channel

These equations are part of Appendix A and will not be repeated here.

3.2 Matrix Section

See Appendix A

3.3 Voter Section

See Appendix A

3.4 Output Channel

See Appendix A

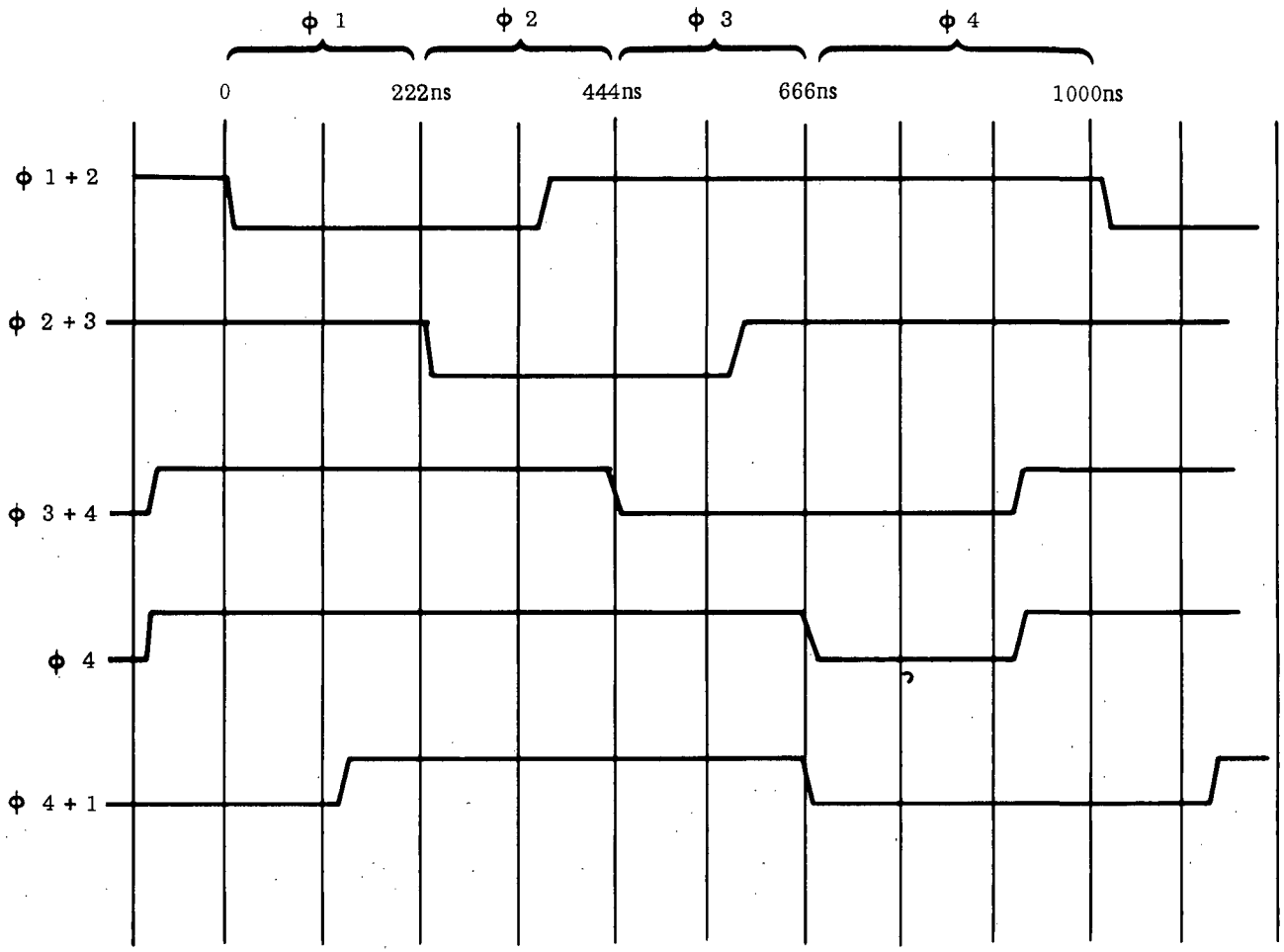


Figure C-3. Four Phase Clock