AD NUMBER

ADB003993

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to U.S. Gov't. agencies only; Administrative/Operational Use; JAN 1975. Other requests shall be referred to Federal Aviation Administration, Supersonic Transport Office 800 Ir. Idependence Avenue SW, Washington, DC 20590.

AUTHORITY

FAA ltr, 26 Apr 1977

Report No. FAA-SS-73-3

SST Technology Follow-On Program—Phase II

REDUNDANT FLIGHT-CRITICAL CONTROL SYSTEM EVALUATION

Task IV:

Analog and Digital Systems Failure Analyses and Preflight Test Designs

J. H. Husband, K. E. Andreasen, M. L. Beattie, D. W. Carr,
L. E. Grefsrud, K. A. Hill, J. E. Templeman, and L. R. Tomlinson

Boeing Commercial Airplane Company
P.O. Box 3707
Seattle, Washington 98124

D6-60287
January 1975

FINAL REPORT

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FAA-SS-73-3	Final rept. on	Phase 2, Task 4

4. Title and Subtitle
Redundant Flight-Critical Control System Evaluation: Analog and Digital Systems Failure Analyses and Preflight Test Designs

5. Report Date
Jan[illegible]75

6. Performing Organization Code

7. Author(s)
J.H. Husband, K.E. Andreasen, M.L. Beattie, D.W. Carr, L.E. Grefsrud, K.A. Hill, J.E. Templeman, B.R. Tomlinson.

8. Performing Organization Report No.
D6-60287

9. Performing Organization Name and Address
Boeing Commercial Airplane Company
P.O. Box 3707
Seattle, Washington 98124

10. Work Unit No.

11. Contract or Grant No.
DOT-FA 72 WA-2893

12. Sponsoring Agency Name and Address
Federal Aviation Administration
Supersonic Transport Office
800 Independence Avenue S.W.
Washington, D.C. 20590

13. Type of Report and Period Covered
Final Report–Task IV

14. Sponsoring Agency Code
296 Pp

15. Supplementary Notes
DOT/SST FCD task technical monitors: Messrs. Sid Platt and M.H. Lowe (ARD-500).

16. Abstract

The U.S. SST prototype commercial airliner under development from 1967 to 1971 employed redundant flight-critical control systems as an essential part of the airplane's airworthiness. The flight-control system electronics were analog for the flight-critical stability augmentation functions and digital for the automatic flight-control functions. The digital system, through an automated preflight test function, also served to establish the integrity of the flight-critical elements. The SST program was terminated before these systems became operational. This study deals with the mechanization of redundant electronic systems. Specifically, the study evaluates analog and digital electronic designs for implementing a triplex fail-operational flight-critical control system. The primary subjects studied were analog and digital systems, multiple failure fail-operational capabilities, and preflight integrity check requirements. This document deals with failure modes and effects analysis, and preflight test requirements of the systems studied.

17. Key Words	18. Distribution Statement
Redundant flight controls Digital control system Preflight test Failure modes and effect criticality analysis Automated system test	Approved for U.S. Government only. Transmittal of this document outside the U.S. Government must have prior approval of the Supersonic Transport office.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified			

Form DOT F 1700.7 (8-69)

DD 1473

DN 390145

PREFACE

This document is one of a series of documents reporting on work conducted under task IV of the SST Technology Follow-On Program, phase II. Task IV studies are an extension of the flight controls technology studies conducted during phase I of the SST Follow-On Program and were given the subject title: Flight Controls Development (FCD). The FCD study results are covered within the reports listed below:

- Report FAA-SS-73-1, *SST Longitudinal Control System Design and Design Processes (Hardened Stability Augmentation Design)*
- Report FAA-SS-73-2-1, *Redundant Flight-Critical Control System Evaluation (Analog and Digital Systems Descriptions)*
- Report FAA-SS-73-2-2, *Redundant Flight-Critical Control System Evaluation (Analog and Digital Systems Performance Comparisons)*
- Report FAA-SS-73-3, *Redundant Flight-Critical Control System Evaluation (Analog and Digital Systems Failure Analyses and Preflight Test Designs)*

The failure modes and preflight test investigations of the redundant analog and digital systems discussed in this document were not an attempt to determine which was the optimum system. Rather, the analysis and laboratory tests were to provide insights into the methods of performing the failure analysis, and the considerations involved for developing a preflight test based on the results of the analysis. Therefore, the reader must derive his own conclusions, along with those presented, in determining the system features that most benefit the application of interest.

CONTENTS

CONTENTS–Concluded

FIGURES

FIGURES–Continued

FIGURES–Concluded

TABLES

ABBREVIATIONS

A/D	analog to digital
AFCS	automat'c flight control system
BIT	built-in-test
CIU	computer interface unit
CMCU	computer monitor and control unit
CP	central processor
CU	computer unit
D/A	digital to analog
DCI	device controller interface
DFD	dynamic failure detector
DMA	direct memory access
DOT	department of transportation
ECS	electric command servo
ECSS	electric command and stability system
FAA	Federal Aviation Agency
FCD	Flight Controls Development
FMECA	failure modes and effect criticality analysis
HSAS	hardened stability augmentation system
ICP	incremental control processor
ICPS	incremental control processor (triplex) subsystem
I/O	input/output
LRU	line replaceable unit
LSB	least significant bit

MCP	modular control processor
MDR	memory data register
MLV	majority logic voter
MSB	most significant bit
MTA	minimum time available
MTBF	mean time between failure
na	nanosecond
OFD	oscillatory failure detector
PHT	peripheral hardware test
RAM	random access memory
ROM	read only memory
SA	sign adder
SFD	static failure detector
SLI	synchronous logic interface
SSCU	system status and control unit
SSFD	signal selection/failure detection
SST	supersonic transport (U.S.)
STRU	servotransmitter/receiver unit
WWCS	whole-word computer (triplex) subsystem (MCP-703)

1.0 INTRODUCTION

As aircraft designers are developing a new generation of airborne vehicles that employ redundant electronic subsystems as a basic part of the aircraft design, automated system test features must be added to the subsystem electronics to enable determining the systems operational status or to assist in the isolation of system failures to effect manageable system maintenance. The flight controls development study (task IV of the SST Technology Follow-On, phase II) deals with the mechanization of redundant electronic subsystems. Specifically, the study compares analog and digital electronic designs for implementing a triple-redundant flight-critical system. The study included assessing the failure modes and effects of the systems studied and investigated the preflight test requirements of those systems relative to a dispatch critical system operation. The application model is based on the stability augmentation systems under development for the U.S. supersonic transport at the time of the SST program cancellation in 1971.

The flight controls development (FCD) task statement of work was not directed toward developing a flight control system for a given vehicle, but was drafted to permit technology investigations into selected areas of triple-channel, fail-operational, analog and digital system designs. Interface requirements, operational performance, failure protection mechanizations, and preflight test requirements were fundamental areas to be investigated. Hardware/software relationships with respect to a digital systems failure analysis were a special item of interest. Laboratory testing of baseline analog and candidate digital systems was a part of the FCD statement of work. The hardened stability augmentation system (HSAS) of the prototype U.S. SST was selected as the baseline function (application model) for the FCD task studies.

The general background and scope of the FCD task are presented in FAA-SS-73-2-1 (ref. 1), along with detailed descriptions of the one analog and two digital systems studied. The two digital systems evaluated were a variable incremental control processor subsystem (ICPS) (developed during the early stages of the SST program) and a whole-word computer subsystem (WWCS), a general-purpose digital processor design similar to those currently under development by many flight-control system electronics suppliers. Specific areas identified for examination during the FCD task were:

- General control function operational performance
- Sensor signal selection and failure detection processing
- Redundancy management for fail-operational/fail-passive system response
- System preflight test requirements and methods, supported by failure modes and effects analysis

Analysis and laboratory test results on the first three areas are covered in FAA-SS-73-2-2 (ref. 2). This report presents the results of the failure modes and effects and preflight test studies.

Section 2.0 of this document highlights the background of the SST program, which led to the preflight test area of interest, with the scope and approach of the studies conducted presented in section 3.0. Section 4.0 summarizes the one analog and two digital subsystems failure modes and effect criticality analyses and preflight test requirements. Section 5.0 covers the application model, HSAS, preflight test mchanization. Sections 4.0 and 5.0 also provide an overview of the studies conducted, with the bulk of the detail analyses placed in the appendixes.

2.0 BACKGROUND

To achieve the most economically attractive commercial supersonic transport, the U.S. SST designers incorporated stability augmentation systems as part of the airplane safety-of-flight primary design. Such a design approach allowed performance parameters (i.e., weight, range, cruise efficiency, and takeoff/landing speeds) to be improved over that achievable with conventional designs not using full-time stability augmentation.

New approaches to safety assurance were necessary in the design of stability augmentation systems directly related to vehicle flight safety. The two most dominant configuration features that were to provide safety assurance were:

- System redundancy (multiple-failure fail-operational capability)
- Preflight system test (an integrity check of the flight-critical subsystems)

These were the primary study subjects of the FCD task.

Introductions to the Boeing SST flight-critical analog and digital subsystems are given below. This is followed by a section that briefly discusses system design/development areas, which potentially can be more effectively treated with a digital system design. A discussion of the SST system test design approach and the problems encountered with that approach follows.

2.1 SST FLIGHT-CRITICAL CONTROL SYSTEM

Three subsystems became involved in the development of a flight-critical control system for the Boeing SST: an electric command and stability system (ECSS); a hardened stability augmentation system (HSAS); and an automatic flight control system (AFCS). The ECSS was being designed to provide normal (desired) airplane handling qualities throughout the entire flight regime. The HSAS was to be an extremely reliable design providing minimum-safe airplane handling characteristics during subsonic flight, as a backup to the ECSS. The AFCS was to provide autopilot functions and, more significantly, to provide the supervisory control for conducting an automated preflight test of the HSAS and ECSS systems.

Both the HSAS and ECSS were four-channel, fail-operational-squared (operational after two failures) analog hardware designs. Four channels were required to achieve the functional reliability numbers specified for the SST's 2.7-hr nominal mission. Analog technology was selected for the HSAS and ECSS hardware primarily because sufficient insight into state-of-the-art digital hardware failure modes and effects, and sufficient piece-part reliability data on airborne digital computers did not exist. The design emphasis for the HSAS was placed on obtaining the greatest component reliability with regard to sensor selection, circuit design, mechanical layout, and manufacturing techniques. The reliability of the ECSS design, though stressed, would have been lower than that of the

HSAS, since the ECSS computational functions were more complex and had more interfaces with other airplane subsystems. Automated system test and development of simple maintenance procedures also received high priority in the HSAS and ECSS designs.

The AFCS was a triple-channel, fail-operational, whole-word digital design. The three-channel configuration was selected to meet automatic landing category III low-weather-minimums safety criteria. The digital design became necessary in order to effectively implement a preflight system test, determined essential to ensure mission reliability goals of the flight-critical control system.

The preflight test function became essential because in-flight monitoring could not detect all critical system failures while the airplane was on the ground, and because reliability requirements were based on beginning each flight with no critical failures (i.e., the integrity of the system functions, especially the in-flight monitors, had to be ensured prior to dispatch).

All three subsystems had essentially been designed and fabrication of the hardware was well under way at the time of the SST program cancellation. Boeing had initiated analytical and limited laboratory evaluations of these designs in preparation for system validation once all hardware was received from the electronics supplier. For a more detailed description of the SST flight-critical control system design, design requirements, and associated disciplines involved in developing such a system, refer to FAA-SS-73-1 (ref. 3).

2.2 DIGITAL SYSTEM APPLICATION

Experience during the SST program on the development phase of the HSAS, ECSS, and AFCS equipment brought out the design inflexibility of analog hardware once fabrication had begun and the attendant high cost for modifying high-quality (highly reliable, low-tolerance) analog circuits. This situation was magnified by channel redundancy requiring changes to each of four channels every time a change was implemented, and by system design sensitivity (changes required) for each iterative refinement of the airframe/subsystems definition as the final airplane configuration was being evolved.

Early configuration development of the stability augmentation systems for the SST included consideration of the use of digital computers. Trade studies were conducted on all-digital, all-analog, and hybrid systems. The final configuration (analog hardware for the augmentation functions and a digital subsystem for the AFCS, capable of administering preflight checkout of the analog equipment) was selected because of the very low confidence level (high risk) in the applicability of digital computers for flight-critical functions. The state of the art of digital equipment at that time (1968) raised unanswerable questions on cost, functional reliability, and development schedule risks. The selected SST flight-control electronics configuration did, however, initiate the development of a digital computer redundant system, AFCS, but in a role that could not directly jeopardize the development and flightworthiness testing of a prototype SST airplane.

In the past few years, digital control-system technology has advanced at an extremely high rate. Component procurement schedules and fabrication cost factors are more visible

and competitive with analog hardware. As more systems are involved with an airframe development program, the digital approach offers a potential cost saving because of the flexibility of its software downstream of the hardware design freeze point.

With the termination of the SST program AFCS development, meaningful application data on multiple-channel digital control systems to the particular problem of flight-critical controls are lacking. The intent of the FCD task for the SST Technology Follow-On Program was to obtain such data and determine the practicability of the use of digital control system electronics in a flight-critical control system application.

2.3 SYSTEM TEST DESIGN BACKGROUND

At the outset of the Boeing prototype SST development, stringent dispatch and block reliability requirements and flight safety goals were established for the flight control systems to ensure that the systems availability and safety capabilities would satisfy the inservice needs of a production SST. It was recognized early in the concept definition phase that the systems complexity and the time constraints imposed by commercial service operation would require that testing of the flight control system be automated to the greatest practical extent. Hundreds of individual tests are required to determine the operational health of a quadruplex system. These tests could not be performed manually in the time allotted for in-transit and turn-around unscheduled maintenance and preflight operational readiness testing. The digital automatic flight controls subsystem, therefore, was assigned the task of performing automated system test of all critical elements of the flight control system.

The term *system test* was used to collectively identify all operational tests conducted to determine that the flight control system was properly assembled and adjusted, was performing within specification limits, and had not suffered failures that would affect the functional capabilities of critical elements of the system. System test for the SST included a preflight system readiness test, an in-flight autoland test, a postflight maintenance test, and a periodic functional test. The latter test was to be used to perform required functional tests following initial system assembly and installation. These tests were also augmented by an automated acceptance test (bench test) conducted at the line replaceable unit (LRU) and flight-control electronics subsystem level.

The automated system test capability described above did not explicitly perform tests on essential support systems such as the hydraulic and electrical power generation and distribution systems or the environmental control systems. These support systems were to be self-sufficient with respect to system test capabilities. The automated flight control system would, however, determine the presence or absence of the outputs of these essential support systems.

2.3.1 Design Considerations for System Test

In order to meet overall mission reliabilities, the electronic design concept had a strict reliability budget both for performance and departure delay. Systemwise, the criteria was established as:

- A failure(s) with probability greater than 1 in 10^5 flight-hours shall not cause a flight deviation or discomfort to passengers or crew.

- A failure(s) with probability between 1 in 10^5 and 1 in 10^6 flight-hours may require flight deviation but no significant increase in passenger discomfort.

- A failure(s) with probability between 1 in 10^6 and 1 in 10^9 flight-hours may cause marginal passenger discomfort.

- A failure(s) that is catastrophic shall not occur with a probability greater than 1 in 10^9 flight-hours.

The reliability budget to meet these objectives was then partitioned into the electronic subelements of 600 hr mean time between failure (MTBF) for the automatic flight controls, 675 hr MTBF for the electric command and stability system, and 1100 hr MTBF for the hardened stability augmentation system. Dispatch reliability, which was defined in terms of a delay because of equipment failure, was likewise strict. Budget allocations for delays of 15 min or more were 8.5 per 10^6 departures for all of the dispatch required electronics. Added to these criteria was an additional one related to test circuitry. Test circuitry in itself was not to reduce the flight-control systems operational reliability by more than 2% or contribute to the propagation of failures between channels.

Given that departure delays because of an equipment failure or malfunction was not to exceed 15 min or more, and assuming that corrective maintenance would not begin until 5 min after arrival at the destination gate, the total times allocated for servicing and unscheduled maintenance were 40 min for transit service and 100 min for turnaround. This was based on a normal 30-min through stop and a normal 90-min turnaround stop. The assumed ratio of through stops to turnaround stops was 1:9. The maximum times required to perform the automated preflight readiness and maintenance tests were established as 5 min and 20 min, respectively. From these test time goals, it was evident that automation of the tests was mandatory.

It was also evident from the time available for unscheduled maintenance that a high probability of isolating the failure to the affected LRU was also required. This requirement established the criterion that failures should be isolated to the LRU with a 97% confidence level. It was also required that an in-flight failure diagnostic and annunciation capability be provided so that the flight crew could alert the maintenance crew in advance of arrival as to the LRU(s) requiring replacement during the unscheduled maintenance. Replacement LRU's could then be made available within the first 5 min of the service stop.

The system test (preflight test) provision was to have the capability of determining the operational status of all critical elements of the flight control system. To accomplish this with a 100% confidence level within the limited time specified above became an exceedingly difficult task, especially when many elements of the system were not amenable to fully-automated system test methods. Crew participation was required in some instances to perform required functions such as making manual control inputs, operating switches, observing displays, engine startup, and sequencing the reference power supply systems.

Restrictions on the operation of engines (for prevention of noise, personnel hazards, fire, excessive fuel consumption, etc.) further complicated the ability to perform system test under in-flight system interface conditions.

Automated and semiautomated tests were to be employed whenever possible with the AFCS administering the test sequence. Flight and maintenance crew participation in the test sequence, which was to be kept at a minimum, was to be indicated by an alphanumeric readout of the required test involvement. System failures and appropriate diagnostic information were to be supplied through an alphanumeric display and a subsystem failure recording numeric displays. Control of the test was to be provided on the same panel which was to annunciate the test in progress. Figure 2-1 illustrates the system test control and display panel under development at the time of the SST program termination.

The tests were divided relative to preflight and functional tests and as to the system elements being tested. Since the AFCS digital computers were to administrate the tests and judge test results, the computers were to receive a full-operational assurance test prior to any test on other elements within the flight control systems. The test programs were segregated from the flight control programs within the computer memory to minimize the possibility of programming errors being introduced into the flight programs when changes were made to a system test program.

Control of the test process by the AFCS was via a digital data link to digital interface electronics contained within each of the electronic LRU's being tested. The tests were organized to allow nearly simultaneous testing of all electronic LRU's. The tests were also selected so that mechanical and hydraulic system elements would be tested during the tests of related drive electronics. The system was divided into functional levels and subdivisions that were directed toward achieving the 97% LRU failure isolation criterion. A fixed test stimulus was to be applied to the circuit (function) being tested, and the response was to be measured against a desired response at an appropriate time interval(s). The design included circuitry to isolate elements of the system in order that given tests effectively check the integrity of specific functions. Because of this, end-to-end tests were to be performed following subdivision tests to ensure that the test segregating logic had restored the circuits to their intended configuration.

The preflight, postflight, and functional system test circuitry was to be isolated during flight to prevent the possibility of in-flight failures being introduced by the test electronics. The preferred method was to use two independent switches to remove electrical power from all test electronics during flight. The minimum acceptable approach was to provide one manual switch and one switch operated by the ground-to-air interlock switch (squat switch) system with provisions that system monitoring would verify that isolation had occurred.

The final evaluation of the system test capability was to have been accomplished by analysis and test using success path and fault tree analysis techniques. The purpose of these analyses was to analytically demonstrate that block reliability and system safety numerical goals could be satisfied by the specified preflight and periodic tests.

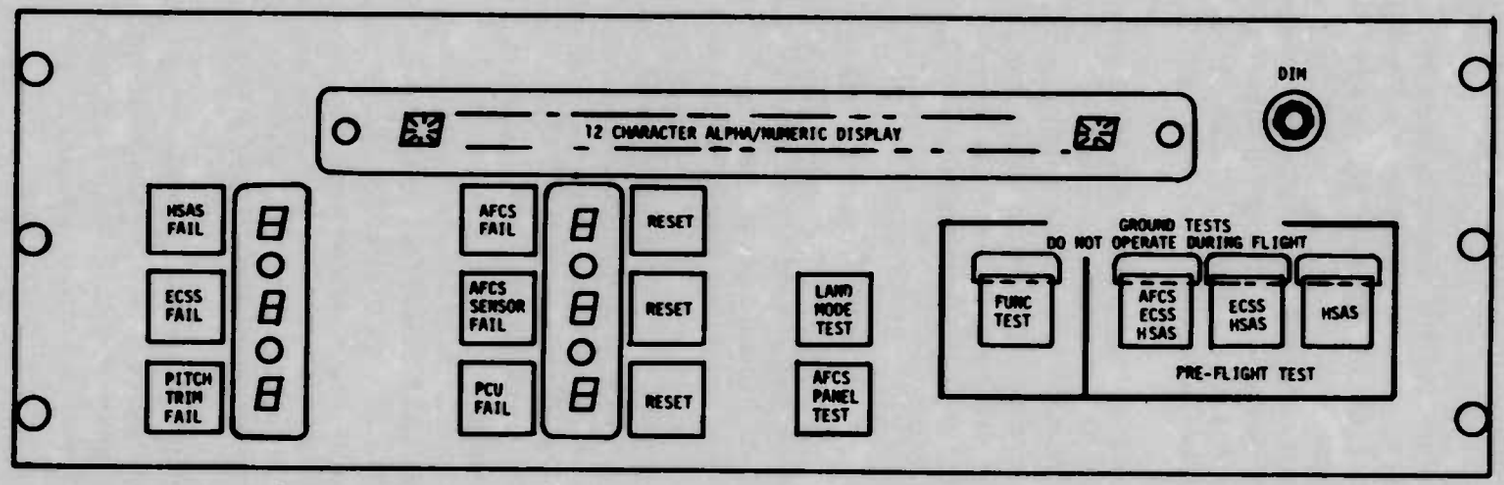


FIGURE 2-1 – SST SYSTEM TEST CONTROL AND DISPLAY PANEL

2.3.2 SST System Test Approach Assessment

Just prior to the SST program termination, the system test design/implementation definition work had progressed to a stage where several problems were becoming visible with the system test approach introduced above. Memory, interface wiring, and digital interface circuit requirements had all grown beyond desirable/acceptable boundaries with serious questions arising as to the completeness of the tests relative to ensuring system preflight integrity. It appeared that the approach taken was becoming a piece-part system check, which did not take advantage of the system redundancy monitoring that existed in the system. Each test had been defined to check specific elements such that failure of the test would immediately identify the faulty LRU.

The program terminated before a new approach (based on using system redundancy intelligence in conjunction with end-to-end testing) could be developed and before relative trades between analog and digital system mechanizations could again be reviewed. The following discussion relates specifics of the problems mentioned above.

Memory Requirements

The estimate of memory requirements escalated by a factor greater than three between initial hardware definition of 4,000 words of memory and program termination. The memory requirements were still only an estimate at termination. It was becoming apparent that the AFCS computer resident memory of 16K would have to be expanded. This could have been in the form of an additional memory LRU if all test programs were to be resident in core, or alternatively, a cassette tape where portions of the test program would be read in and executed in block form.

Interface Wiring

Because of the systems test signal interface requirements between the AFCS computer unit and HSAS/ECSS LRU's, a fourth computer interface unit had to be added to the system. But even with the additional LRU, all interface units required two rack and panel connectors having 336 pins each. Commercial experience to date indicate that field problems can be expected with units having such a high pin count.

Digital Interface Electronics

The quantity of interface electronics required within the ECSS, HSAS, and the pitch-trim control electronics to perform the system test had grown to the point that overall system reliability, LRU volume, and LRU weight were adversely affected. The systems test interface electronics had grown to approach 50% of the electronics associated with the system's operational design–a level that would significantly affect the HSAS and ECSS mean time between failure.

Completeness of Test

Even with the complexity of the subelement testing being implemented, many elements of the remaining system had test shortcomings that would have needed correction.

In some instances, particularly in the test of the in-flight monitors, the continuity of interface wiring between redundant channels was not adequately verified. The monitor circuitry in each channel should have been isolated and tested to ensure that continuity existed from the error signal source in one channel to the monitor circuit in another channel. Something that would require a systems approach for testing system integrity rather than a subelement/subcomponent verification test.

3.0 SYSTEM TEST STUDIES

The FCD task system test studies concentrated primarily on determining the preflight test requirements for the one analog and two digital systems evaluated. The study objectives were to develop an understanding of test concepts and test procedures, and to acquire data relative to subsystem tests, test time, and memory requirements (presuming a general purpose computer as the test administrator).

Preflight test is the portion of a system's automated system test function, which specifically deals with determining the flight-critical control-systems operational integrity prior to takeoff. The basic intent of a preflight test is to establish for the flight crew a go/no-go status of the flight control system.

3.1 SCOPE

The preflight test studies involved conducting failure modes and effect criticality analysis (FMECA) determining test steps that would check all system critical elements and developing test sequences that would minimize the overall test time and interface circuitry requirements. A system end-to-end test concept was adopted in that, where possible, tests would be defined utilizing system sensor outputs to drive the system with success determined by the servosystem response. Where this was not possible, test sequencing would be based on a building block approach where functions already tested would be used in determining the test success of functions yet to be tested.

Since the analog and incremental digital systems did not inherently have the capability to be programmed to administrate the system test functions, preflight test for these systems was to be defined but not operated as an integrated test in the laboratory. Laboratory testing of the whole-word system, however, was to include an operational preflight test function, coresident in memory with the flight control laws of the application model.

The SST HSAS function (ref. 1) was selected as the application model for the preflight test studies. This model included simulated sensors, flight-critical control-system computational electronics, and hydromechanical electric command servo-subsystem. Other ground rules adopted for the study were:

- Flight-control system hardware downstream of the electric command servoelements (power control units and interfacing linkage) would not be considered.
- The system was a triplex system satisfying fail-operational requirements.
- All elements of the triplex system were assumed essential for aircraft dispatch. Dual- or single-channel operational status would not be established.
- Preflight test would assure that the system is fail-operational and that all first and second failure detection/warning functions are operational.

- Crew participation in the preflight test administration should be held at a minimum.

3.2 APPROACH

The development of an adequate preflight test is intimately involved with the actual design of the subsystem on which the test is to be administered. Each of the functional blocks of figure 3-1 must be analyzed in depth to determine the tests required which establish that the function performs properly under normal and related failure state conditions. Test procedures/sequences are arrived at by developing a functional flow diagram of the system elements at the hardware mechanization level and evaluating the design through the FMECA. The FMECA is used to uncover possible failure modes that are not manifested in a timely manner by the system monitoring functions (latent failure states). If latent failure states are not discovered, a test is defined that provides an integrity check of the function and associated monitors. If latent failure states are found to exist, and if they compromise the predicted operation of the function following detectable failure conditions, two options are available. The system design can be changed to remove the possibility of incurring the latent failure, or a special test can be defined as part of the preflight test function to establish that the failure condition does not exist prior to dispatching the system. This latter approach must be supported with a failure probability analysis which forecast the probability of incurring the latent failure and subsequent hazardous failure conditions to be less than 1 in 10^9 flight-hours between the running of the preflight test and completion of the subsequent flight (see sec. 2.3.1).

This study treats the application model functions of figure 3-1, which are within the dashed enclosure, i.e., sensors, control function processing electronics, and electric command servo-subsystem. The control-function processing electronics are evaluated for analog, incremental digital, and whole-word (general purpose) digital designs.

Failure mode and effect analyses of the analog and digital systems present two very different analysis problems. In the analog mechanization, dedicated piece parts performed specific functions for all time. This permits the functional flow diagram to be a very strict representation of partitioned circuit groups. As a result, failure mode effect analysis and testing becomes fundamentally systematic; i.e., system operation for given functional failures can be determined by failing circuit components, through analysis and/or testing, and observing the system response. In addition, this circuit-related functional flow diagram allows tests to be defined that check the hardware integrity as well as the system function. However, if the functional operation is modified; i.e., hardware design changes made, this systematic failure analysis/test definition procedure must be repeated for the affected areas of the functional flow diagram.

Digital systems do not in general lend themselves to the same systematic failure analysis approach described above. First, most of the functional signal flow is facilitated through multiplexing the same component and constructing functions timewise by altering the interaction or sequential operation of the piece-part elements. Secondly, the timing and control logic, which paces the multiplexing process of the hardware, adds an additional dimension to the failure modes and effects analysis. Software, the computer resident

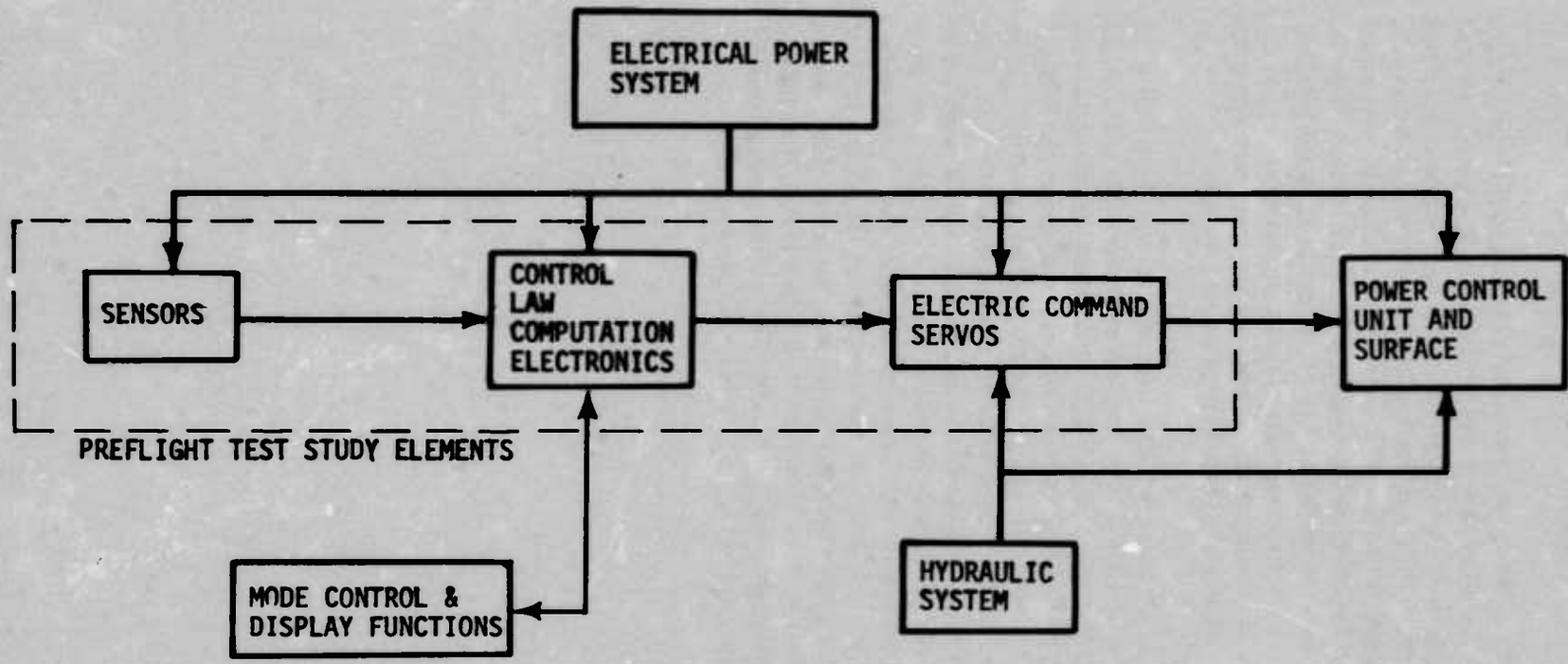


FIGURE 3-1 – FLIGHT CONTROL SYSTEM MAJOR FUNCTIONAL BLOCKS

program, provides the information from which the piece-part components interact for each discrete time pulse generated by the computer timing and control logic. A system functional flow diagram for a digital system, therefore does not represent the partitioning of dedicated circuit groups. It more closely represents the functions defined by software; i.e., defined by dedicated groups of computer program instructions. This means that checking a function with input/output tests like those used with an analog system will not necessarily verify that the function and associated hardware is operable for all states (e.g., amplitude/rate conditions) of the function. The piece parts are not viewed as continuous operating devices dedicated to the function, but a multitude of piece-parts which may vary for each computational pass in processing the input/output relationship of the function.

Because of the operational differences between the analog and digital systems presented above, it was judged that a functional flow diagram of the hardware alone could not be used as the primary reference in conducting a FMECA. The study approach taken in the FCD task was to use the overall system redundant configuration to direct the FMECA effort. Two levels of investigation become apparent relative to an overview of the system redundant architecture. The first level concerns itself with the channel cross-tie points. Each intertie between the channels (involving both hardware and software) must be analyzed to determine whether a failure mode in one channel or within the cross-tie interconnecting harness could encumber the operation of more than one channel. After the intertie points have been checked for design integrity and failure mode performance, the second level of investigation must be completed. This deals with determining whether latent failure points exist within single channel elements which, when combined with a detectable system failure, can encumber more than the operation of a single channel. The results of both levels of investigation are combined in developing appropriate preflight test requirements.

4.0 COMPUTATIONAL ELECTRONICS FMECA AND PREFLIGHT TEST REQUIREMENTS

The system configuration can be separated into major functional blocks as illustrated in figure 3-1. This section summarizes the computational electronics subsystems (analog, incremental digital, and whole-word digital) failure mode studies and preflight test requirements. The discussion on each subsystem presents a review of the system's redundancy operation, results of the failure modes analysis, and specific tests required for a preflight integrity check. The overall preflight test mechanization approach, which includes the subsystem integrated with the peripheral hardware of the application model, is presented in section 5.0.

A complete detailed failure mode analysis was not made on each subsystem since neither the application model nor the subsystems were part of a prototype/production development effort. The intent of the information presented is to provide an insight into the difficulty and complexity of such investigations relative to the potential of a flight-critical digital system.

Detailed descriptive information on the three subsystems hardware designs and comparative operational performance is presented in FAA-SS-73-2-1 (ref. 1) and FAA-SS-73-2-2 (ref. 2) FCD task reports.

4.1 ANALOG SUBSYSTEM

The analog subsystem is a *brickwall* triple-channel system design utilizing in-line monitors for failure detection. The redundant channels of the analog system only crosstie through a hydromechanically force summing point at the output of redundant electric command servos. Figure 4-1 illustrates the analog application model configuration. A failure mode analysis of the hydromechanical elements of the application model is not a part of this study, only the associated electronic elements that are involved with failure detection. Since the analog system channel monitoring is mechanized essentially at a single point and associated with the output of the electric command servo (ECS), preflight test requirements involved the consideration of the electric command servosystem normal/failure condition operation.

The application model analog control law processing electronics consisted primarily of a piece-part mechanization of the HSAS filter functions, built up on a breadboard level following design philosophy put forth for the SST (and most contemporary aircraft) analog circuit designs. The circuits were laid out to follow the functional flow diagram of the system (fig. 4-2). Such a circuit design approach permits laboratory verfication or investigation of the failure mode effects of opening or shorting specific components, where the components can be directly related to a parameter within the functional flow diagram.

4.1.1 Analog Subsystem Monitoring

The in-line failure detection monitors of the analog system are located with the servodrive electronics (fig. 4-3). A differential pressure sensing transducer across the

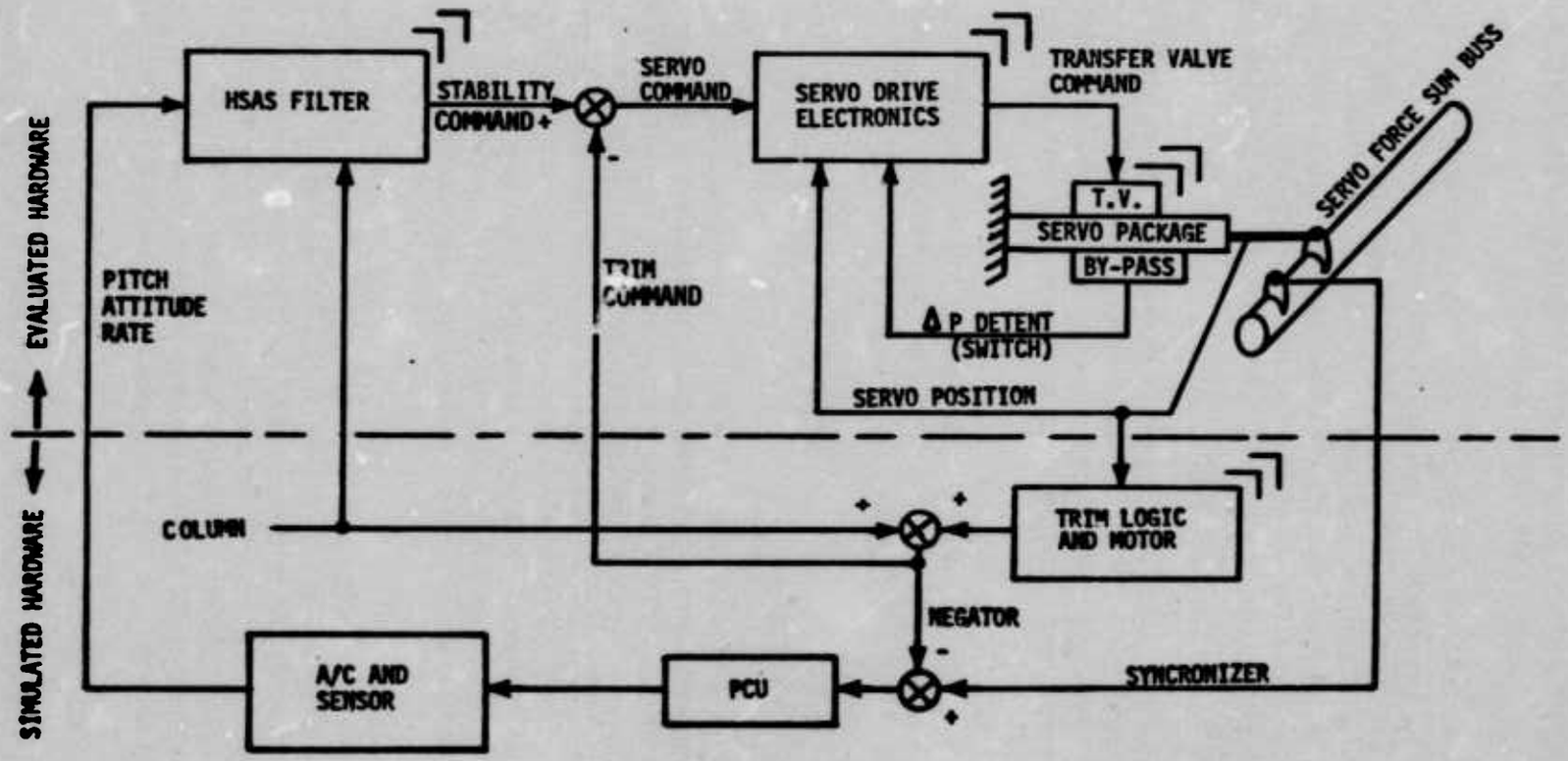
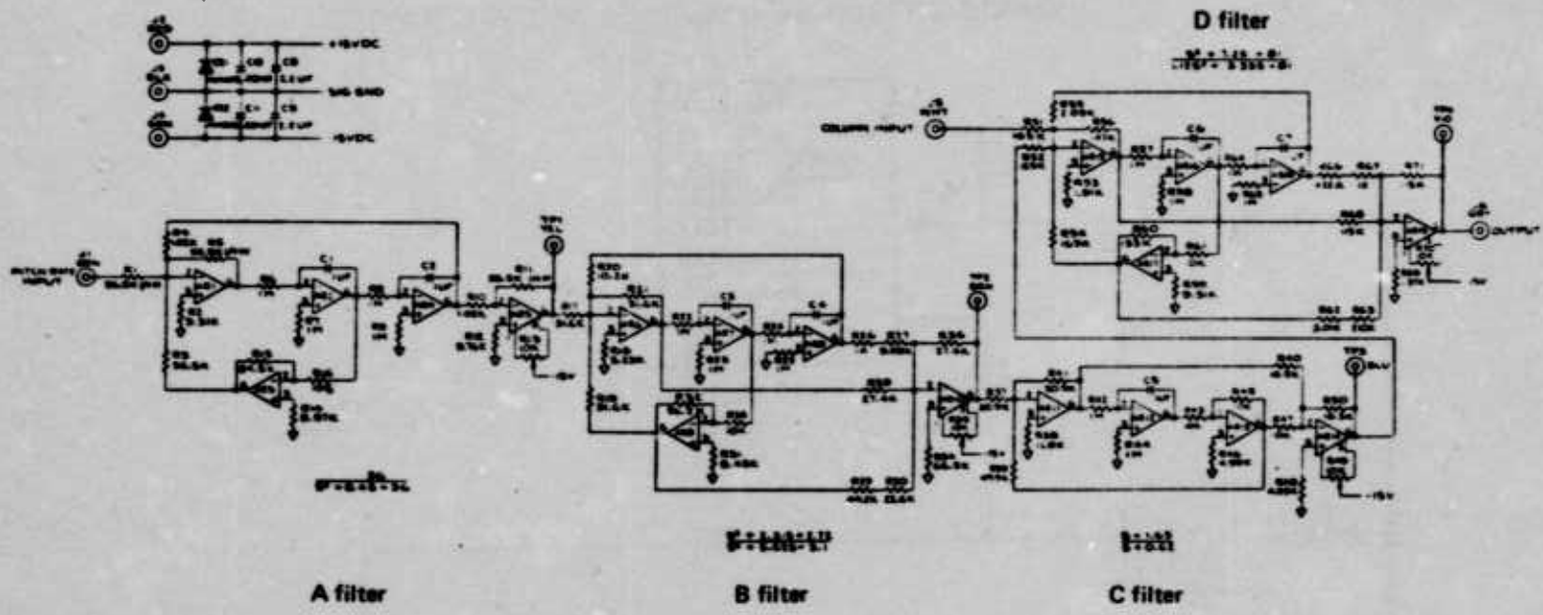


FIGURE 4-1–ANALOG APPLICATION MODEL CONFIGURATION



Note:
Except as noted, all resistors 1/8 W, 1% on all amplifiers.

FIGURE 4-2–HSAS FILTER BREADBOARD

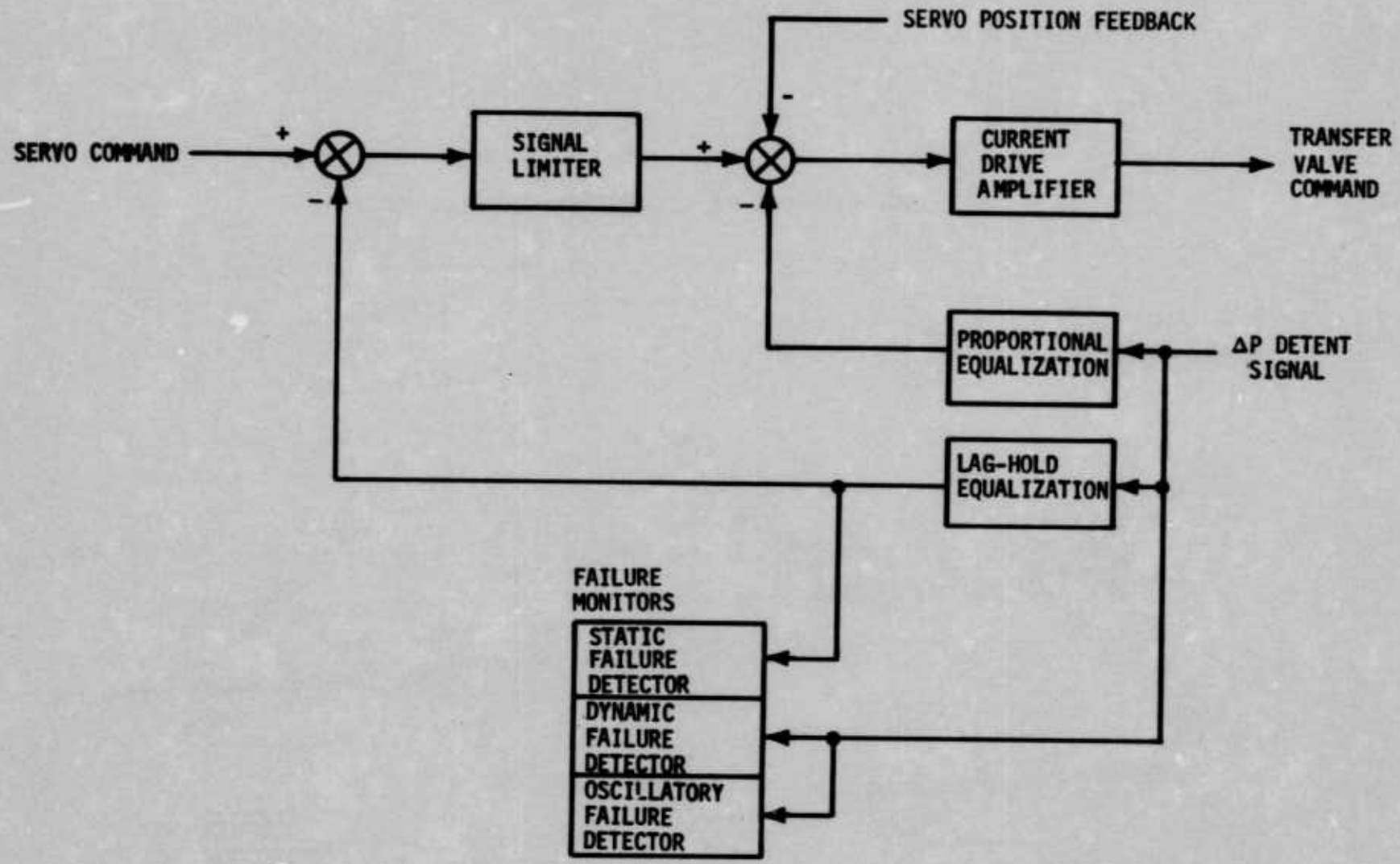


FIGURE 4-3–SERVO DRIVE ELECTRONICS

servoactuator provides the signal used for both failure monitoring and servoequalization. The sensor pickoff is associated with a differential pressure (ΔP), a pressure relief detent mechanism that opens and ports fluid across the servoactuator when the ΔP reaches a predetermined threshold. This hydromechanical detent action limits the actuator force level relative to the other redundant electric command servos. The ΔP electrical pickoff is used as a feedback signal to equalize the servocommands through both proportional and pseudointegration (lag-hold) signal paths. Equalization is required to balance normal dynamic and static system tolerances between the redundant channels.

Failure monitoring for the analog system has been subdivided into three failure-state detection areas; dynamic, static, and oscillatory. Dynamic and oscillatory failures are monitored directly from the ΔP detent signal, and static failures are monitored downstream of the lag-hold equalization function (fig. 4-3). Both the dynamic and static failure detectors utilize the same design (fig. 4-4) where the lag-hold function provides an integral accumulation of the static errors. These failure monitors register a failure when the input signal exceeds a specified threshold for one second. Should the input signal exceed the threshold for less than one second, the time delay is reset as the signal drops below the threshold. The oscillatory failure detector was designed to monitor failure states that resulted in a servodetent oscillation of greater than 0.2 cps. These monitors are the basis for detecting all failures within the analog system, sensors through servos.

4.1.2 Analog System FMECA

Failure modes and effects criticality analysis on the analog system followed the systematic approach of analyzing/testing piece-part hardware failures to determine their effect on the system performance. Breadboard circuits of the computational electronics were physically altered to represent typical failure conditions. Failure modes were inserted into the system while the system was operational in a closed loop laboratory set-up with a simulated airframe. As the failure modes were inserted, the failure mode response was observed from two viewpoints; failure detection (assuming the monitors had not failed) and airframe disturbance. Detail failure mode results are shown in tables 4-1 and 4-2.

Figure 4-2 is the schematic of the HSAS filter breadboard circuit (one channel). Each component on the breadboard was failed during laboratory testing. The capacitors were tested as shorts and as opens. The resistors were tested only as opens since this is their primary failure mode. Laboratory testing only considered total failure of the component; no testing was done based on component degradation (component value change with time). Failures of operational amplifiers were considered to be equivalent to passive component failures. An opening of a downstream resistor is like an operational amplifier failing to zero, and an opening of the plus (+) grid reference resistor is equivalent to an operational amplifier failure to saturation.

Test results, tabulated in tables 4-1 and 4-2, define hardover (HO) and slowover (SO) failure responses based on whether the dynamic or static failure detector (DFD or SFD) was the first to register the particular fault. Faults that are labeled no effect (NE) indicate that the failure was not detected by any monitor.

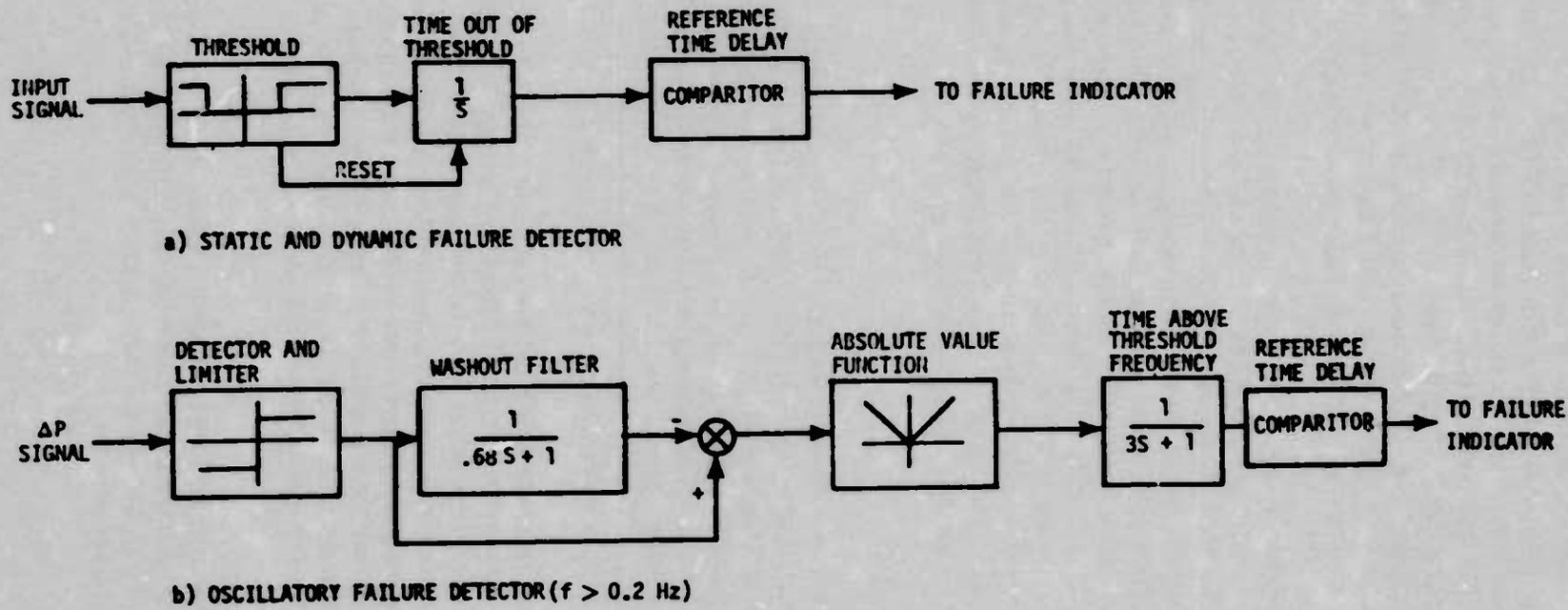


FIGURE 4-4—ANALOG FAILURE MONITORS

TABLE 4-1.–HSAS FILTER FMECA TEST

Component	Failure	Effect	Monitor Tripped				Comments
			OFO	SFO	OFO	NONE	
R1	O	Passive(P)				X	
R2		Hard-Over(HO)	X				
R3		Oscillatory(O)			X		
R4		HO	X				
R5		No Effect(NE)				X	
R6		HO	X				
R7		HO	X				
R8		HO	X				
R9		HO	X				
R10		P				X	
R11		HO	X				
R12		HO	X				
R13		RE				X	Op amp trim resistor
R14		HO	X				
R15		HO	X				
R16	O	O			X		
C1	O	O	X				OFO detected because of command offset. Oscillation beyond servo bandpass
	S	HO	X				
C2	O	O	X		X		Output is saturated squarewave
	S	HO	X				Output shift to 1/2 full scale
R17	O	P				X	
R18		HO	X				
R19		O			X		
R20		Slow-Over(SO)		X			Function of Op amp offset bias
R21		O	X		X		Output is odd shaped waveform
R22		HO	X				
R23		HO	X				
R24		HO	X				
R25		HO	X				
R26		NE				X	
R27		RE				X	
R28		RE				X	
R29		RE				X	
R30		NE				X	
R31		HO	X				
R32		O-SO		X			Oscillation above servo bandpass
R33		RE				X	
R34		HO	X				
R35		NE				X	Op amp trim resistor
R36	O	HO	X				
C3	O	O-(RE)	X				Detection depends on offset bias in Op amp
	S	SO		X			
C4	O	O-HO	X				Oscillation above servo bandpass
	S	RE				X	
R37	O	P				X	
R38		HO	X				
R39		SO		X			
R40		NE				X	
R41		O	X				Oscillation above sero bandpass
R42		SO		X			
R43		NE				X	
R44		HO	X				
R45		O	X				Oscillation above servo bandpass
R46		HO	X				
R47		NE				X	
R48		HO	X				
R49		RE				X	Op amp trim resistor
R50	O	HO	X				
C5	O	O				X	Oscillation above servo bandpass
	S	RE				X	
R51	O	P				X	
R52		P				X	
R53		HO	X				
R54		O			X		
R55		SO		X			
R56		O				X	Oscillation above servo bandpass
R57		HO	X				
R58		HO	X				
R59		O	X				Oscillation @ 3/4 full scale offset - oscillation above servo bandpass
R60		O-SO		X			Oscillation to 1/2 full scale offset - oscillation above servo bandpass
R61		O				X	Oscillation below OFO frequency threshold
R62		NE				X	
R63		NE				X	
R64		HO	X				
R65		HO	X				
R66		NE				X	
R67		NE				X	
R68		NE				X	
R69		HO	X				
R70		NE				X	Op amp trim resistor
R71		HO	X				
C6	O	O				X	Oscillation above servo bandpass
C7	O	HO	X				Oscillation full scale square wave above servo bandpass
	S	NE				X	

TABLE 4-2.–EC SERVO FMECA TEST

Component	Failure	Effect	Monitor tripped				Comments
			DFD	SFD	OFD	None	
Servo amp	Active	HO	X				
	Passive	NE				X	
Servo valve	Active	HO	X				
	Passive	NE				X	
EC piston	Jam	NE				X	
Position LVDT and demod	Active	HO	X				
	Passive	NE				X	
⚠1 P detent	Jam	NE				X	
Detent LVDT and demod	Active	HO	X				
	Passive	NE					
Equalization network	Active	HO/SO	X	X			Indication depending
	Passive	NE				X	

Laboratory FMECA results confirmed two points. First, no single failure created a safety-of-flight condition. Second, the failures that were not detected by the monitors produced little or no effect on the airplane dynamics. The analysis and test did, however, show that approximately 26% of the computational electronics failures (essentially filter functions) were not detected by the monitors, as were 50% of the failures (passive failures) associated with the servosystem. These undetected failures could accumulate in time and ultimately degrade the system's performance or be detrimental to the system's safety-of-flight failure response.

Failures not detected by the monitors under in-flight conditions are classified as latent failures. Such failures must be analyzed to determine their probability of occurrence relative to functional test verification check intervals, to satisfy the mission success criteria, or to be designed out of the system. Since other elements of the HSAS (sensors, pilot command paths, and monitors) possessed failure states that were latent, difficult to design out, and required a preflight functional verification check before each flight, no attempt was made to remove the computational electronics or servo latent failures through redesign.

4.1.3 Analog Preflight Test Requirements

The SST mission success criteria required that the full complement of redundant system elements be operational at dispatch. As a result of the analog system FMECA studies, specific tests were required and postulated that would clear the analog system monitors and computational electronics latent failure states. The tests were defined in detail, and their creadibility was substantiated in the laboratory. However, the tests were not implemented and integrated into a working preflight test operation since the analog system did not inherently have a means of administrating the test. A complete system preflight test organization and administration operation is covered in section 5.0.

The analog subsystem preflight test requirements could be satisfied by verifying that the monitors were operational and then utilizing the monitors to verify that the computational electronics were operational (no latent failures existed). Thus, two test series were developed.

4.1.3.1 Analog Subsystem Monitors Preflight Test

The task of checking the monitors required adding circuitry to control the state of the lag-hold equalization logic as a function of the ECS simulated engage/disengage state and to provide a ΔP detent signal negation stimuli upstream of the monitors pickoff points (fig. 4-5). A test was developed to be operated when the hydraulic system was depressurized since under this condition the ΔP signal was a constant 5 V. The ΔP signal could be systematically negated to control failure conditions to the in-line monitors, the monitors response checked for proper response, and the monitors response cross-checked channel-to-channel to further verify the test results.

Figure 4-5 shows the time dependent preflight test sequenced test stimuli, and table 4-3 indicates the function checked. In general, the test checks that the dynamic, static, and oscillatory failure detector (DFD, SFD, OFD) monitors will trip, that the DFD and SFD plus and minus thresholds are proper, and that the OFD is frequency dependent.

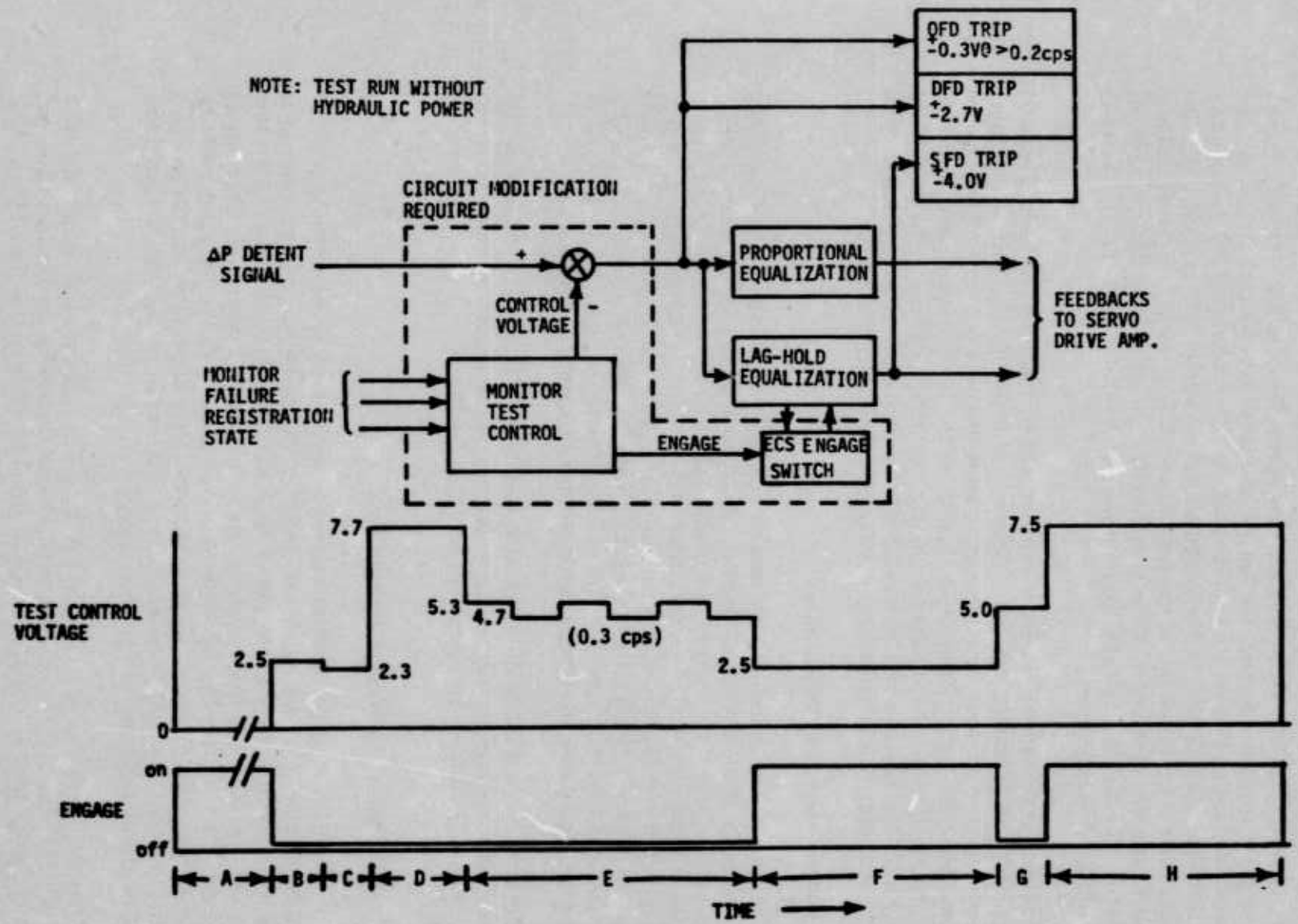


FIGURE 4-5—MONITOR PREFLIGHT TEST

TABLE 4-3.–MONITOR PREFLIGHT TEST

Time period	Monitor tripped				Comments
	DFD	SFD	OFD	None	
A	X	X			DFD and SFD
B				X	
C	X				DFD + threshold checked
D	X				DFD - threshold checked
E			X		OSD threshold and freq checked
F		X			SFD + equalization rate checked
G				X	Equalization reset
H		X			SFD - equalization rate checked

4.1.3.2 Analog Computational Electronics Preflight Test

Filter circuit normal operation and latent failure conditions could be checked by stimulating the rate gyro (input sensor) at specified fixed frequencies and observing that the monitors did not register a failure. By producing a 0.3- and 0.5-Hz square wave excitation into the analog subsystem electronics (fig. 4-6), all pertinent circuitry could be validated. The failure states that were labeled from the FMECA as being latent were retested under the system excitation conditions just mentioned. All were detectable by the OFD except for failures of the operational amplifiers trim resistors (table 4-4). Since the loss of the operational amplifiers trim resistors did not degrade the system's performance, they were not a necessary part of the design.

All but two latent failure modes could be detected with the 0.5-Hz signal. A 0.3-Hz oscillation was required to detect a shorted capacitor in the B filter (C4 of fig. 4-2). The other latent failure was the control column electrical input resistor (R51). This path required a control column input and could not be exercised by the rate gyro excitation.

4.2 DIGITAL SYSTEM FMECA APPROACH

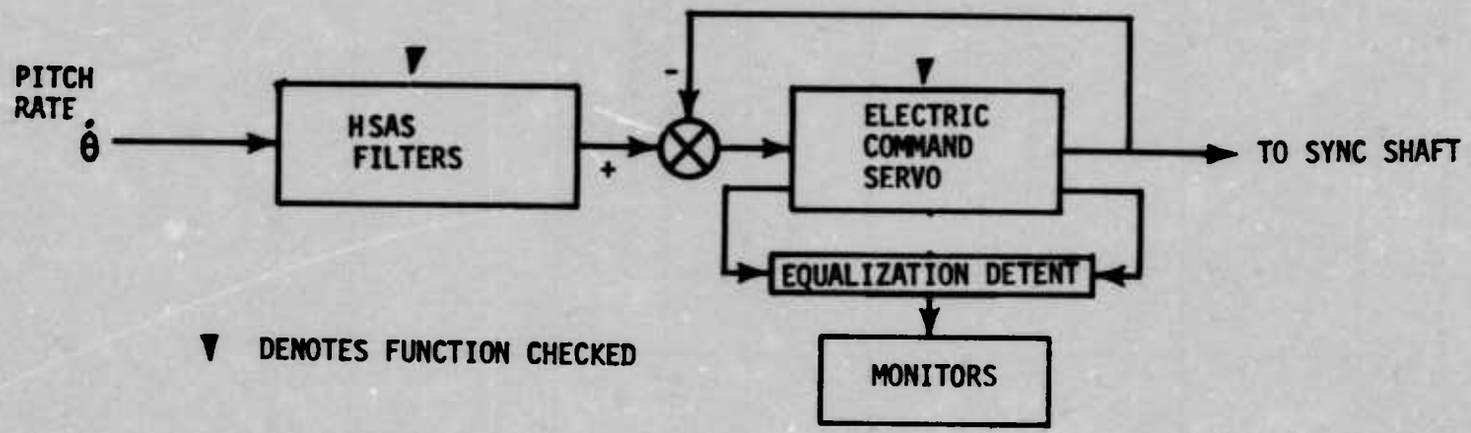
The analysis and prediction of failure effects for a digital system appears to be an order of magnitude more complex than that associated with an analog system. Whereas an analog mechanization dedicates hardware to a particular functional process, the digital system efficiency is achieved by multiplexing common hardware to systematically perform all functions. This time sharing of hardware presents a new dimension to the digital system failure analysis, potentially making the task easier or harder depending on the circuit utilization. Processing (multiplexing), timing, and control circuits can introduce failure modes that are both event and time dependent. The following discusses the various digital system circuit types and attempts to point out the nature of their failure modes.

If we exclude those circuits in a digital system whose signal flow is analog in nature (input signal conditioners, and output sample and hold devices), we can categorize the remaining digital circuits as:

- Combination logic
- Clocked sequential logic
- Asynchronous sequential logic

4.2.1 Combinational Logic

Combinational logic circuits are those circuits that contain no memory and have a truth table whose output is directly related to the inputs. These circuits can be either discrete components (resistors, diodes, transistors, etc.), or integrated circuits refered to as a a chip. If the logic is built on an integrated chip, the FMECA analyst may not be able to find out precisely how the chip is constructed. The same functional design by different manufacturers may have different failure modes depending upon the crystal line structure



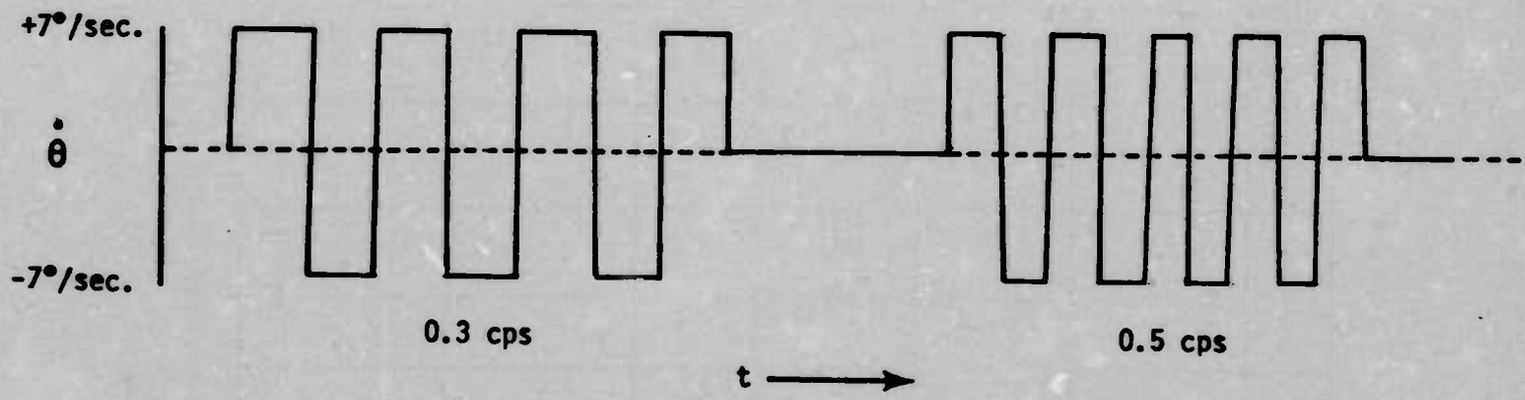


FIGURE 4-6—HSAS ELECTRONICS PREFLIGHT TEST

TABLE 4-4.–HSAS ELECTRONICS PREFLIGHT TEST

Component failed	Test frequency cps	Monitor tripped				Comments
		DFD	SFD	OFD	None	
R1	0.5			X		
R5				X		
R10				X		
R13					X	[1]
R17				X		
R26				X		
R27				X		
R28				X		
R29				X		
R30				X		
R33				X		
R35					X	[1]
C3 (open)				X		
C4 (short)	0.3			X		
R37	0.5			X		
R40				X		
R43				X		
R47				X		
R49					X	[1]
C5 (open)				X		
C5 (short)				X		
R51					X	To be checked during control column command path checks
R52				X		
R56				X		
R61				X		
R62				X		
R63				X		
R70					X	[1]
C6 (open)				X		
C7 (short)				X		

[1] No test stimuli could check an operational amplifier trim resistor. Since no degradation occurred with an open component, these items could be left out of the design.

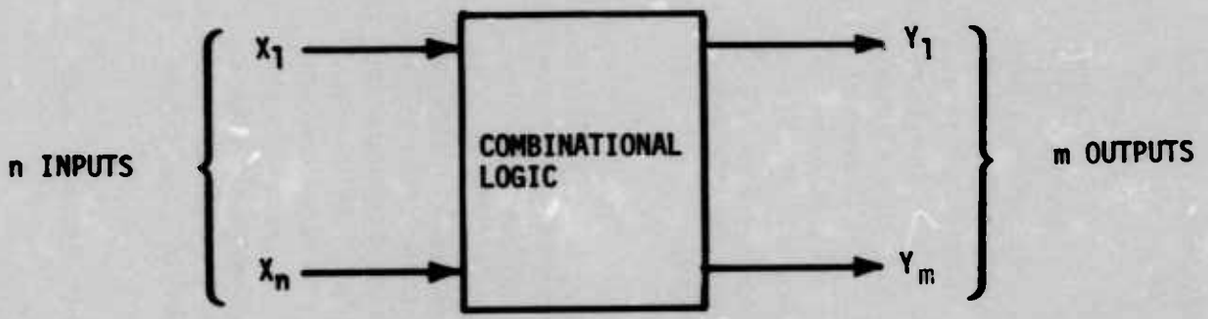
upon which the circuit has been developed. In many cases, these logic circuits will have internal (relative to the boundary about which the analysis is taking place) failure modes that will either fail the output(s) high or low, or change the truth table of the output relative to the input. Thus, the FMECA of combinationed logic circuits must be conducted by determining the system response relative to postulated truth table changes. In general, combinational logic circuits can be represented by a box with n inputs and m outputs as shown in figure 4-7. The truth table for this logic circuit will have 2^n rows and n + m columns. There are m x 2^n possible one bit variations to the truth table. If any or all of the m output bits are considered to change for any row, then there are 2^n x (2^m -1) possible changes that must be investigated for an exhaustive failure modes analysis by the truth table approach. For a small combinational logic circuit of three inputs and two outputs, there would be 2^3 x (2^2 -1) = 24 possible truth table changes that would have to be investigated for an exhaustive truth table failure modes analysis. Thus, exhaustive analysis can become very costly for large combinational logic circuits. Even so, circuits that are built upon chips where the analyst does not know how the circuit logic is built up, the truth table approach may be the only way of being certain that all possible failure modes have been considered. Special combinational logic circuits, which are constructed of Nand and Nor gates, however, can often be effectively analyzed by considering the inputs and outputs of each gate to have failed high or low. This approach can be simpler than the truth table analysis since it does not always require investigating all possible combinations that will fit into the truth table, but only requires investigating failures that can occur in the specified circuit.

4.2.2 Clocked Sequential Logic

Clocked sequential logic circuits contain combinational logic and memory circuits (usually flip-flops (F/F's)) and are wired together to process sequential information. The outputs of a sequential logic circuit not only depend on the present set of inputs, but also depend on the state of the circuit before it went through its last change. In general, a sequential circuit has as many binary state variables as it has memory devices; i.e., F/F's. If a circuit has n binary state variables, then it will have a maximum of 2^n states it can transition to. The functioning of a clocked sequential circuit can be described by a state-transition table or state-transition diagram, just as the functioning of a combinational logic circuit can be described by a truth table. Failures in the clocked sequential circuits can occur either in the combinational logic or in the memory devices. Failures in the clocked sequential circuits will cause changes in the state-transition diagram. The following is an example of how a state-transition diagram can be used as an aid for conducting the FMECA on a clocked sequential logic circuit.

Consider the shift register signal shaping circuit shown in figure 4-8. The circuit no-failure state-transition matrix can be expressed as:

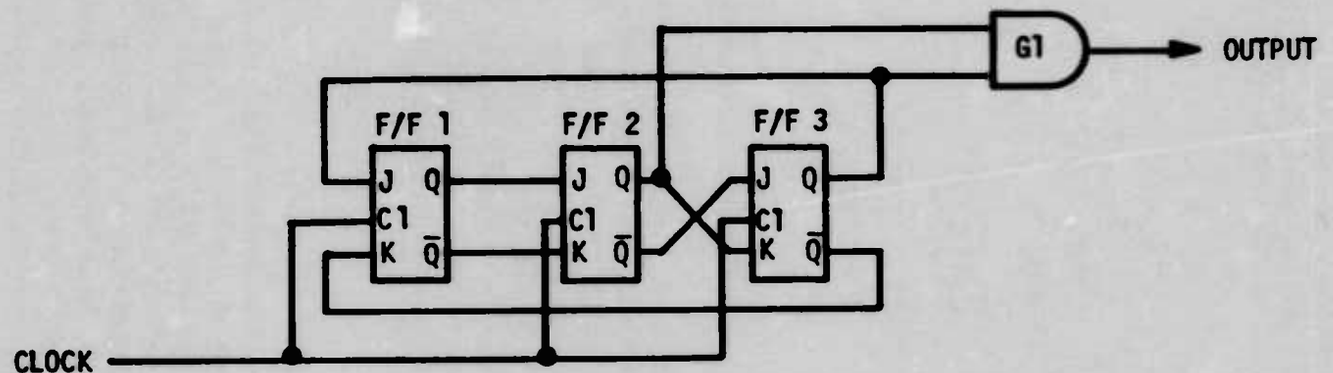
State	0	Transitions to State	4
State	1	Transitions to State	6
State	2	Transitions to State	0
State	3	Transitions to State	2
State	4	Transitions to State	5
State	5	Transitions to State	7
State	6	Transitions to State	1
State	7	Transitions to State	3



TRUTH TABLE

INPUTS						OUTPUTS			
X_1	X_2	•	•	•	X_{n-1}	X_n	Y_1	• • •	Y_m
0	0				0	0			
0	0				0	1			
0	0				1	0			
1	1				0	1			
1	1				1	0			
1	1				1	1			

FIGURE 4-7–GENERALIZED REPRESENTATION OF COMBINATIONAL LOGIC CIRCUIT



F/F TRUTH TABLE

AT t=n		AT t=n+1	
J	K	Q	$\overline{Q}$
0	0	Q_n	$\overline{Q}_n$
0	1	0	1
1	0	1	0
1	1	$\overline{Q}_n$	Q_n

GATE TRUTH TABLE

in		out
0	0	0
0	1	0
1	0	0
1	1	1

TRANSITION MATRIX

t_n					t_{n+1}				
STATE	F/F 1	F/F 2	F/F 3	G1	F/F 1	F/F 2	F/F 3	G1	STATE
0	0	0	0	0	0	0	1	0	4
1	1	0	0	0	0	1	1	1	6
2	0	1	0	0	0	0	1	0	0
3	1	1	0	0	0	1	0	0	2
4	0	0	1	0	1	0	1	0	5
5	1	0	1	0	1	1	0	1	7
6	0	1	1	1	1	0	0	0	1
7	1	1	1	1	1	0	0	0	3

OUTPUT WAVE FORM

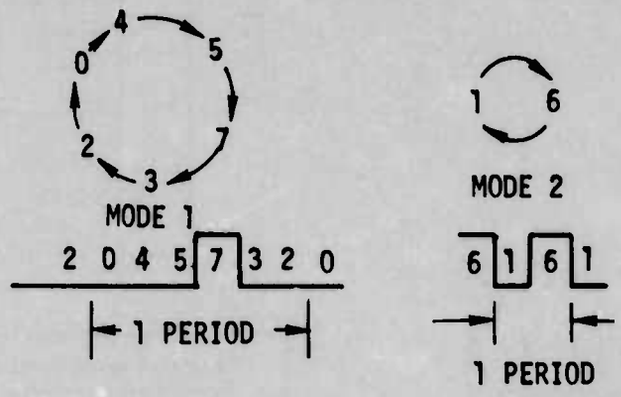
t_{n+1}	=	t_n
F/F 1		F/F 3
F/F 2		F/F 1
F/F 3		$\overline{\text{F/F 2}}$

NOTE: F/F = FLIP FLOP DEVICE

G1 = (F/F 2 • F/F 3)

FIGURE 4-8–EXAMPLE OF A CLOCKED SEQUENTIAL LOGIC CIRCUIT

Two possible stable modes of operation exist for this circuit. One mode has a period of six clock cycles, producing a one only once during the period. The second mode has a period of two clock cycles, producing an output of alternating ones and zeros. These modes, as developed from the above state-transition matrix, can be illustrated by the following diagrams.



In order to effect the mode desired, logic would have to be added to the circuit to initialize the flip-flops to a particular modes state as part of the circuits power-on cycle.

Figure 4-9 presents the failure mode state-transition matrices for failures associated with the third flip-flop (F/F 3) of the subject circuit. From looking at these failure analysis examples, two observations can be made. First, even though this circuit is rather simple, there are a large number of failure modes that may have to be investigated. Only 7 of 12 possible cases are illustrated for only one device in the circuit. Second failures can exist within a circuit analysis boundary such that the output is not necessarily stuck at zero or one, but may continue to produce periodic functions. Failure mode criticality of these failures can only be evaluated by analyzing the downstream circuits response to the failure states produced (a nontrivial task).

4.2.3 Asynchronous Sequential Logic

Asynchronous sequential logic circuits contain clocked sequential logic circuits. The latter are provided with clocking signals extraneous of the circuit boundary. The asynchronous circuits process through a state-transition on a conditional basis as a function of the input. These circuits generally appear as interfaces between nonsynchronized processes. One of the outputs from this type of circuit generally is an interrupt that is utilized to inform a processor that it has information (data) available for another processor. The ICPS-723 does not contain any of this type of interface relative to our laboratory study. The WWCS, discussed in section 4.4, does contain this type of interface, and it will be dealt with in that section of the document.

Fortunately, digital systems are usually put together with relative simple building blocks. The task in conducting the FMECA is to thoroughly understand each of these building blocks and how they fit together to build the system, and then describe the effects of the failures of each building block on the system operation.

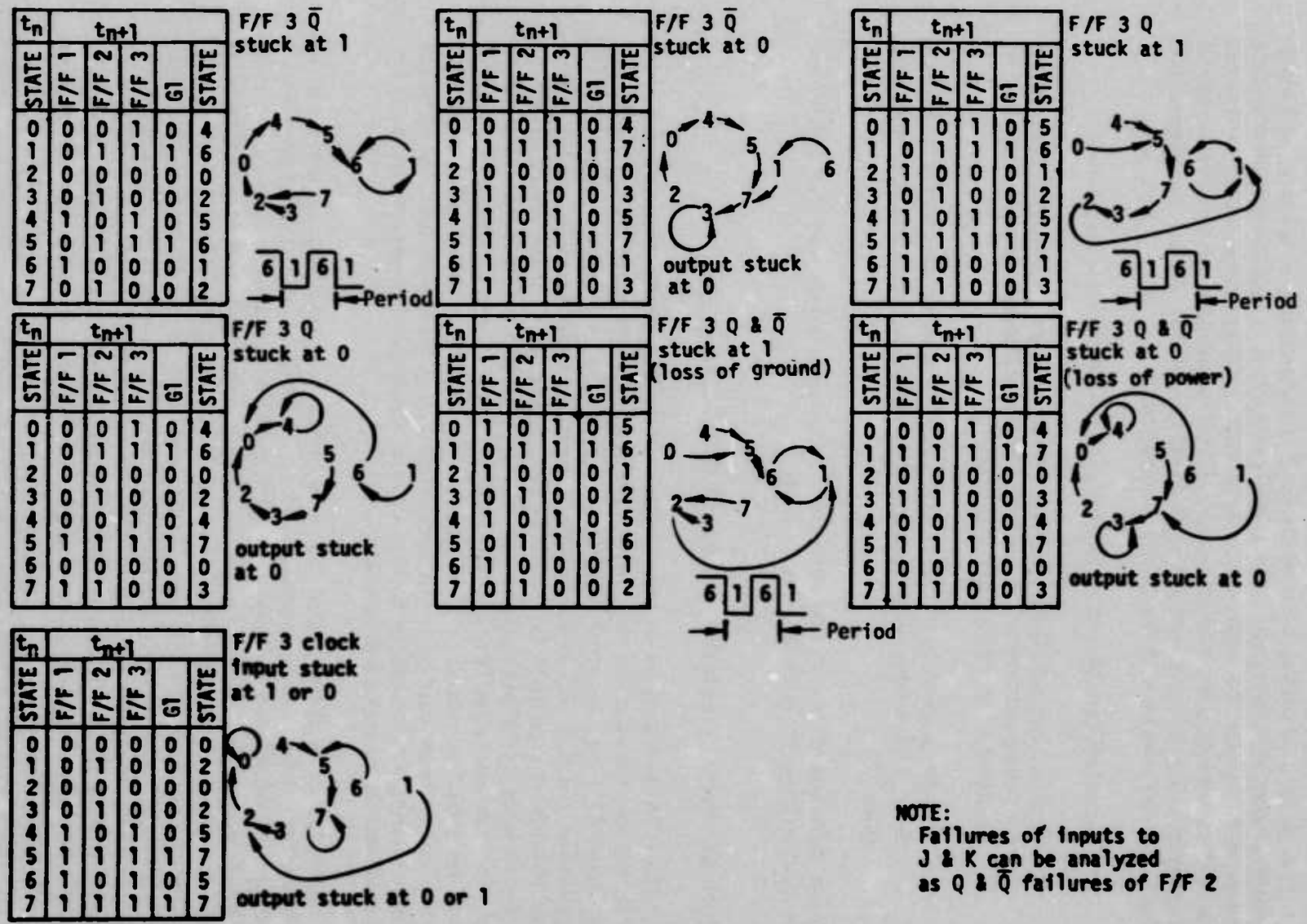


FIGURE 4-9.—FAILURE MODES OF F/F 3 (REF. FIG. 4-8 CIRCUIT)

The use of hand analysis tools, as discussed in the preceding paragraphs, can become unwieldy, laborious, error prone, and eventually impractical as logic circuits under analysis increase in complexity. To augment the hand analysis, failure studies were accomplished using a Boeing special computer simulation facility. This simulator was developed to handle the analysis of logical failure modes in large digital circuits. The simulator is a combination of hardware and software with the following capabilities:

- High-speed logic level simulation capability including propagation delays
- Register level simulation language, which provides a convenient medium for controlling and monitoring the logic level simulation and concurrent register level simulation of interfaces with other equipment.
- Modular description of simulation problems allowing the user to separately describe black boxes or subsystems, and constructing their interface at simulation time.

Further definition of the computer simulator is given in Boeing document D25-71060-1, An Introduction to the Computer Simulator (ref. 4).

4.3 INCREMENTAL CONTROL PROCESSOR SUBSYSTEM (ICPS)

The ICPS application model test configuration was functionally the same as the analog subsystem with the exception that the trim command logic was placed within the ICPS and thus removed from the simulation (fig. 4-10). Unlike the analog system brickwall organization, the ICPS utilizes cross-channel information exchanges as part of the systems basic redundancy operation. In this configuration, (fig. 4-11), input sensor signals are converted to an appropriate digital format and sent immediately to the other channels for signal selection processing. Two other cross-channel data exchanges took place following the input signal selection process before commands are passed on to the servo-subsystem. The electric command servosystem is identical to that used in the analog study, only here the servomonitoring is accomplished by cross-channel comparisons as opposed to the in-line approach used in the analog configuration.

Since the ICPS mechanization has cross-channel interconnections upstream of the hydromechanical force summing point, the first line of failure mode analysis must deal with evaluating these interfaces. Figure 4-12 illustrates the functions of the cross-channel information exchanges. One series of cross-channel processes relates only to control law data while another two are associated with the redundant system's timing and control synchronization. Each cross-channel exchange of information is accompanied by a signal voter (selector) and a monitor. The purpose of the voter is to prevent (block) upstream failures from propagating errors into downstream functions. Since the ICPS is time synchronized on a 1 μsec bit-time level, and the cross-channel interface is mechanized using a serial transmission buss, the voters are simple majority logic circuits providing a two out of three bit selection.

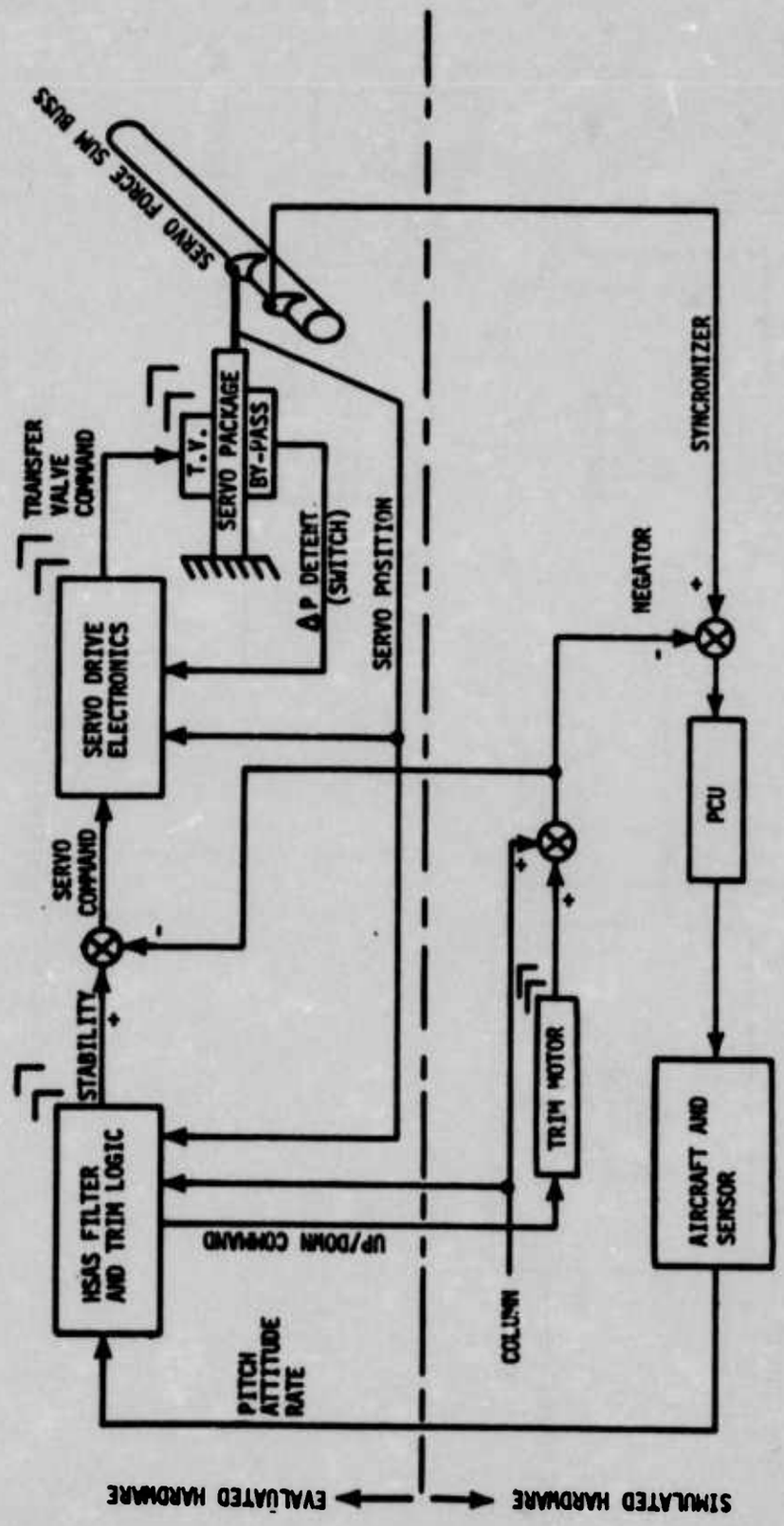


FIGURE 4-10—ICPS APPLICATION MODEL CONFIGURATION

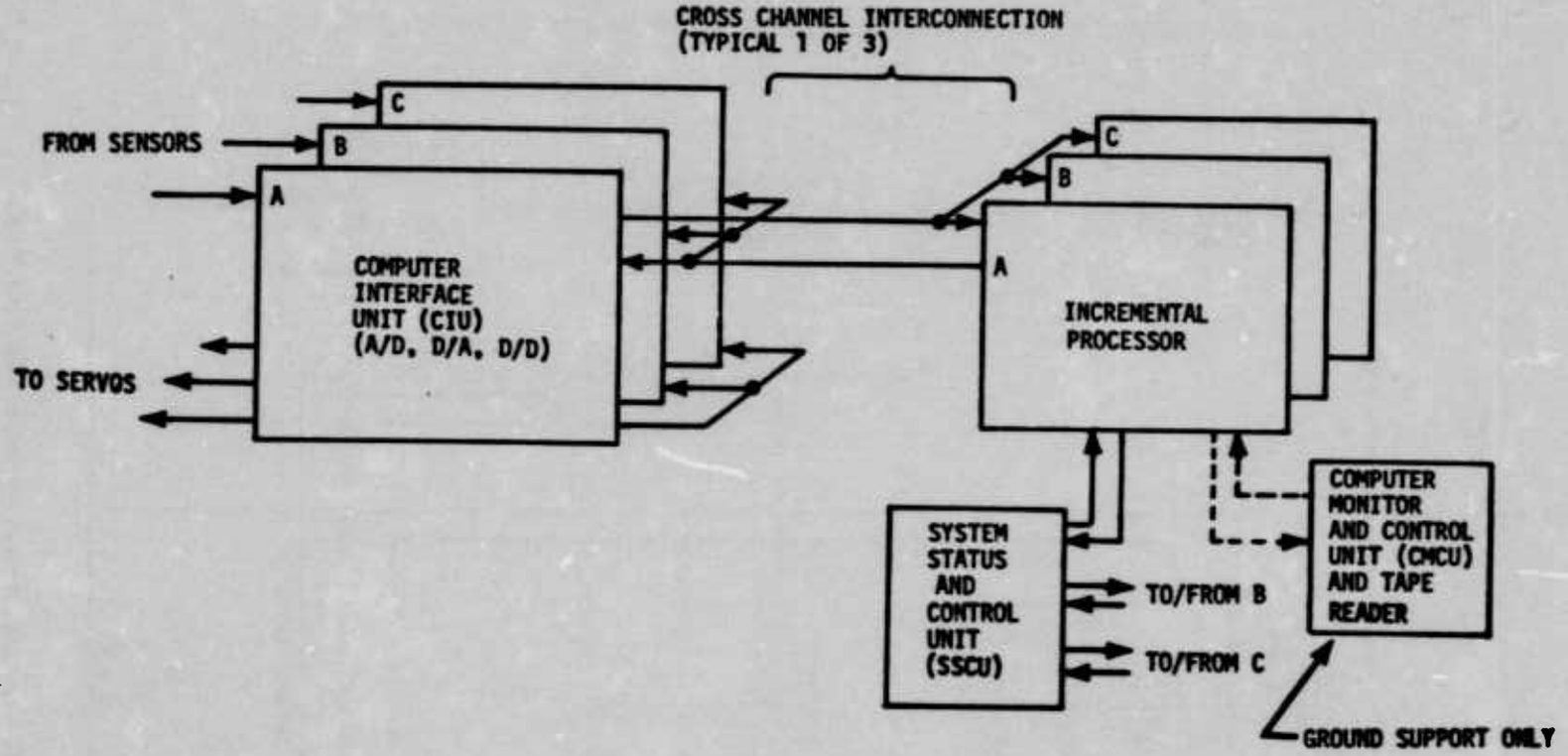


FIGURE 4-11—ICPS HARDWARE INTERCONNECTION

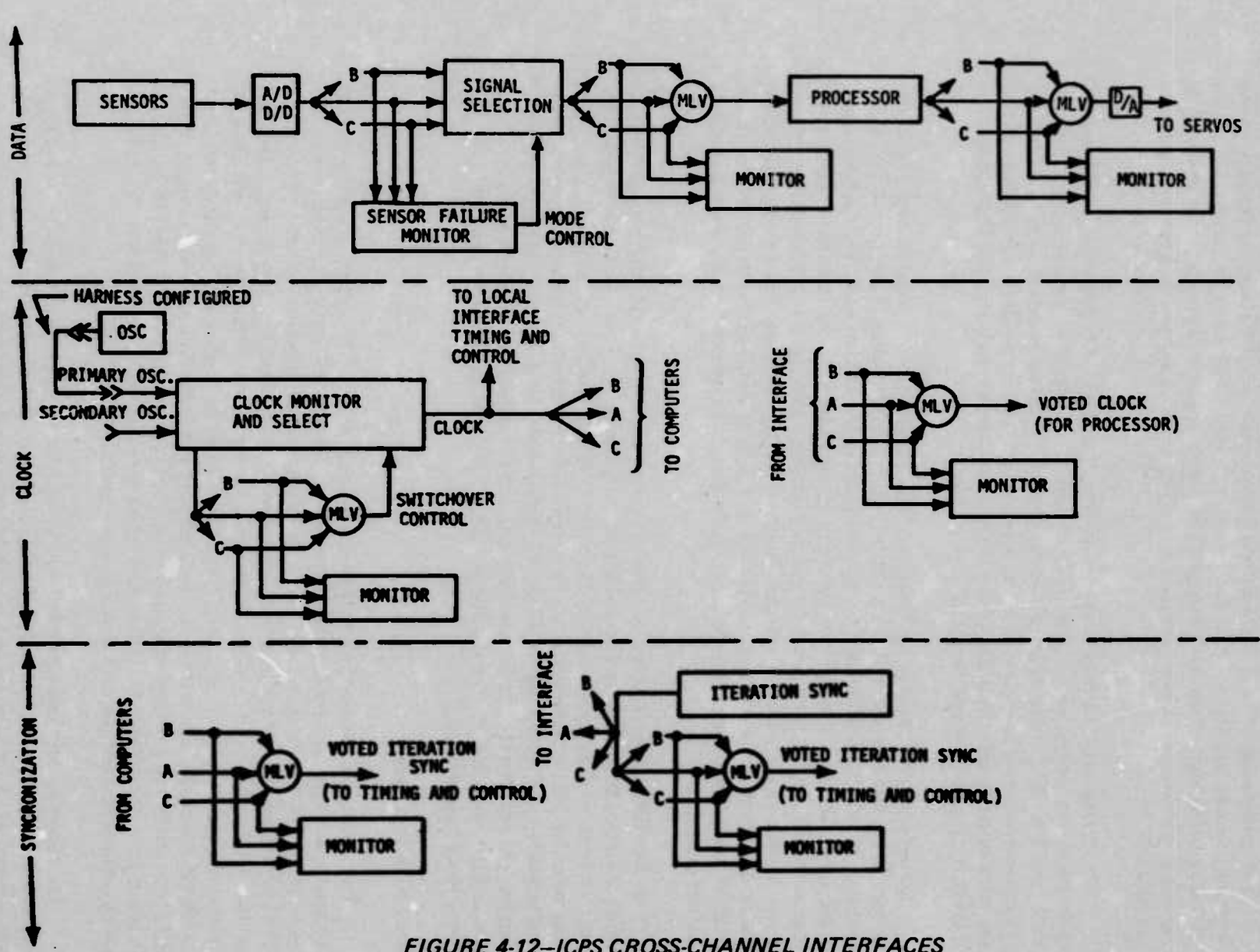


FIGURE 4-12--ICPS CROSS-CHANNEL INTERFACES

The positioning of the cross-channel ties within the ICPS was to separate major functional blocks (analog to digital A/D), signal selection, control law processing, and digital to analog (D/A) and to provide primary LRU failure isolation. The intent was to provide a maximum number of failure success paths without an inordinate amount of hardware complexity.

The timing and control crossties provide the logic for power-on initial system synchronization and a distribution of the basic clock signal to permit LRU functional isolation. The ICPS utilizes an active-standby clock fail-operational scheme where the system operates with reference to a primary clock and switches over to a secondary clock if the primary clock fails. This design is a key area of concern relative to the FMECA because it involves a common thread function.

One other cross-channel interface, the failure status display logic transmitted to the system status and control unit, exists in the ICPS configuration. This interface is not an exchange of information but can be visualized as an extension of each channel housed in a single LRU. The only direct tie of this cross-channel link is at the functional failure lamp indicators that display the failure state at each voting node (fig. 4-13). Each channel's lamp driver is wire ORed to the functional lamps indicating function first and second failure status; i.e., clock, iteration reset, output voter, etc. This interface was lightly treated in the FMECA study since the appropriateness of the design (as opposed to a lamp per channel) was not evaluated.

In the FMECA evaluation of all of the cross-channel circuits, the boundary about which a functional analysis is constructed must be chosen with care. In fact, the boundary must be to some degree a variable in the study. A fixed boundary, improperly chosen, may hide potential failure states relative to a system single point failure. In conducting the clock and majority logic voter/monitor FMECA's, independent studies were made to evaluate the designs. The results were not identical as a result of the individuals selection of the hardware functional boundaries. A comparison of the efforts brought out the significance of the above procedural statement.

4.3.1 ICPS Monitoring

The primary monitoring principle utilized within the ICPS is cross-channel comparisons. There are two functional interface partitions that can be drawn relative to this monitoring. The first is that associated with the redundant elements external to the ICPS. This monitoring works in conjunction with the input signal selection process, and in the case of the application model, monitors the column input, pitch rate sensor, manual trim commands, and parameters of the ECS (i.e., ECS position, negator position, ΔP detent and equalization signals). The second is associated with the redundant elements internal to the ICPS, essentially the monitors related to the majority logic voters within the ICPS hardware. In addition to the cross-channel monitors, some in-line monitors exist that check memory parity, arithmetic overflow, and power supply operation.

The cross-channel failure monitor associated with the input signal selection process is illustrated in figure 4-14. For each set of input signals, the monitor function can establish which signal will be passed on to the control law computations (median, average of two, or

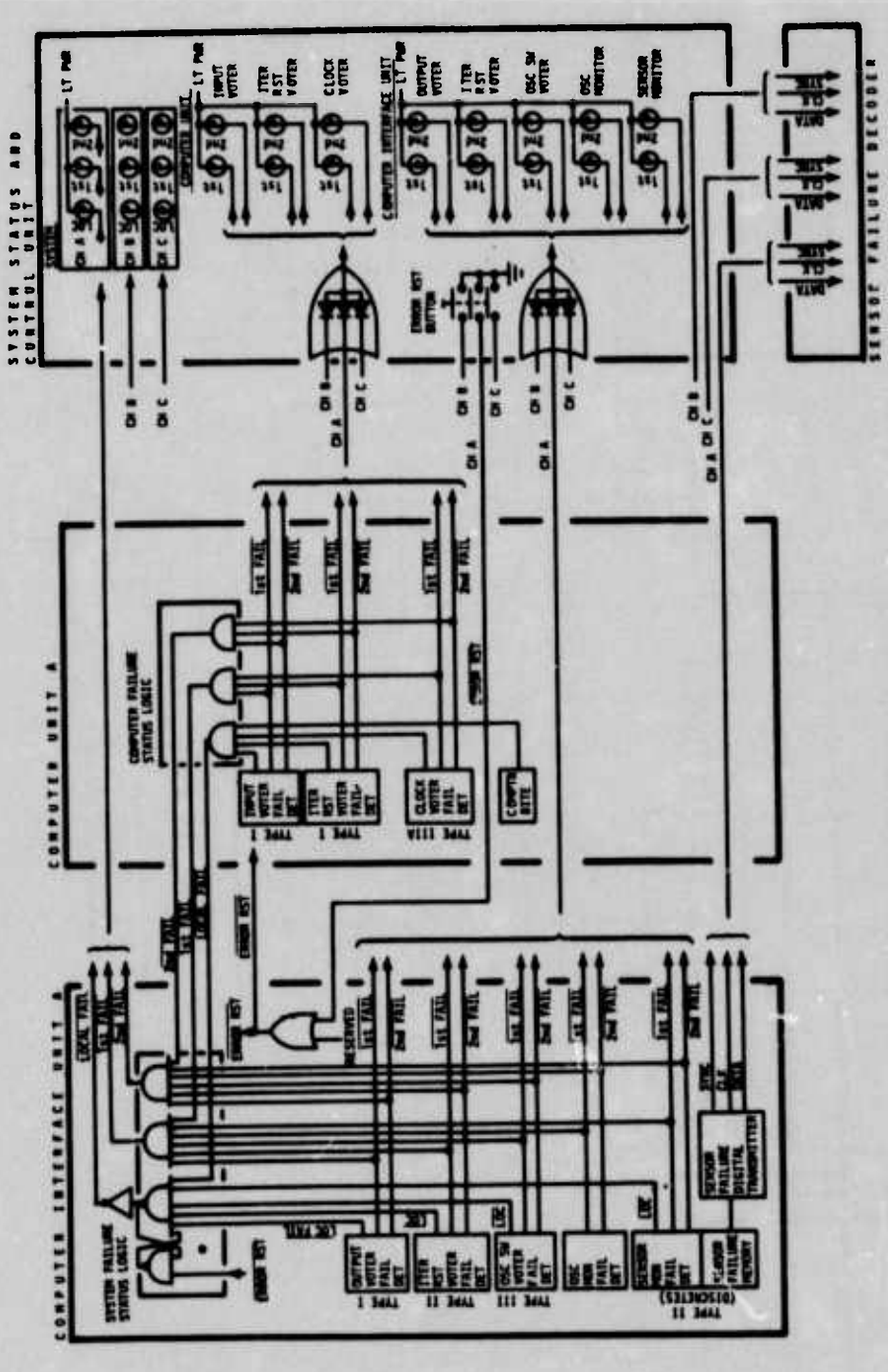


FIGURE 4-13.–ICPS FAILURE STATUS ORGANIZATION

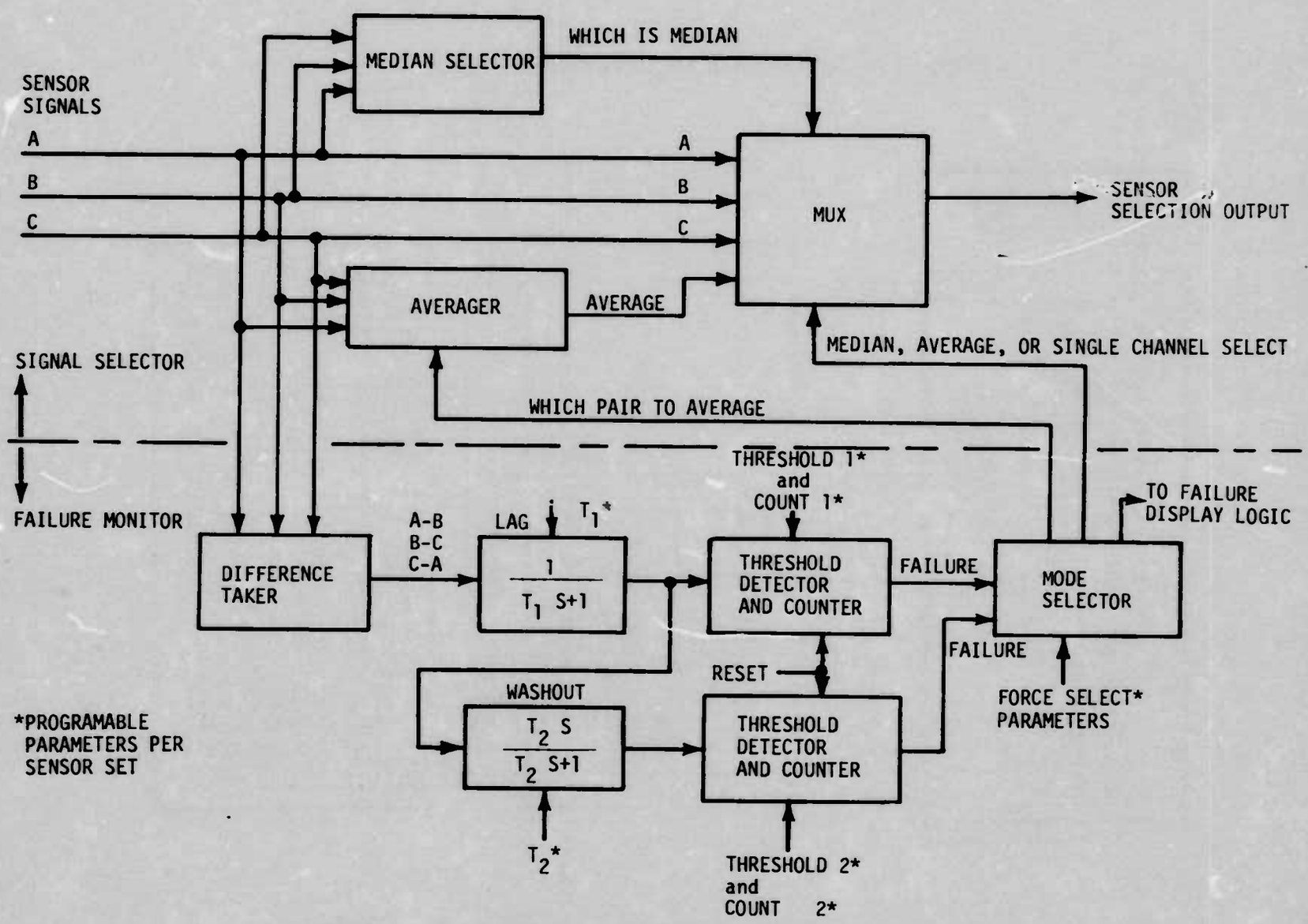


FIGURE 4-14.—SENSOR SELECTOR/FAILURE DETECTION CIRCUIT

single selected). Several parameters within the failure detection logic of the monitor are programmable for each input sensor set. These parameters include filter time constants (lag and washout), threshold level detection at the output of the filters, and allowable iteration counts above the threshold before a failure is registered. Under normal usage, the signal selection/failure detection function is used as a median selection switching to the average of two following first failure. Forced utilization of a specific signal input defeats the monitoring function.

The monitor associated with the ICPS majority logic voter points is shown in figure 4-15. This monitor will register a failure if any difference exists on the data input for more than 500 ns. The monitor provides local first and second failure status information. The registration of a failure can be removed without disturbing the voting function by activating a reset command. If the failure persists, the monitor will again register that fact.

4.3.2 ICPS FMECA

Since the ICP-723 triple-channel system was designed as a bit-for-bit synchronized subsystem, the timing and control structure in each channel anticipates data from the other channels at precise instances in time. Thus, the design of the cross-channel interfaces does not contain interrupt or transmitter/receiver control information. If a receiving channel does not get the anticipated data from one of the other channels, it simply flags the errant channel as failed.

Appendix A contains the FMECA data on the ICPS. Information presented in this appendix is placed in the order of its criticality with respect to assessing the ICPS fail-operational design integrity. The order was arrived at by separating the ICPS into subsystem functional blocks and making a judgement as to their FMECA level of interest (i.e., cross-channel paths or latent fault potentiality). The functional blocks are:

- Redundant clock
- Majority logic voter/monitor
- System failure status logic
- Sensor signal selection and failure detection
- Input/output circuitry and timing
- Computer unit processing electronics

The first four items are classified as level one failure mode functions because they involve cross-channel data paths. The last two items are second level failure mode functions because they represent circuitry between voting nodes and must be checked to determine their potential latent failure-mode states. In analyzing ICPS failure modes, an appropriately monitored downstream voter (hydromechanical subsystem) was assumed.

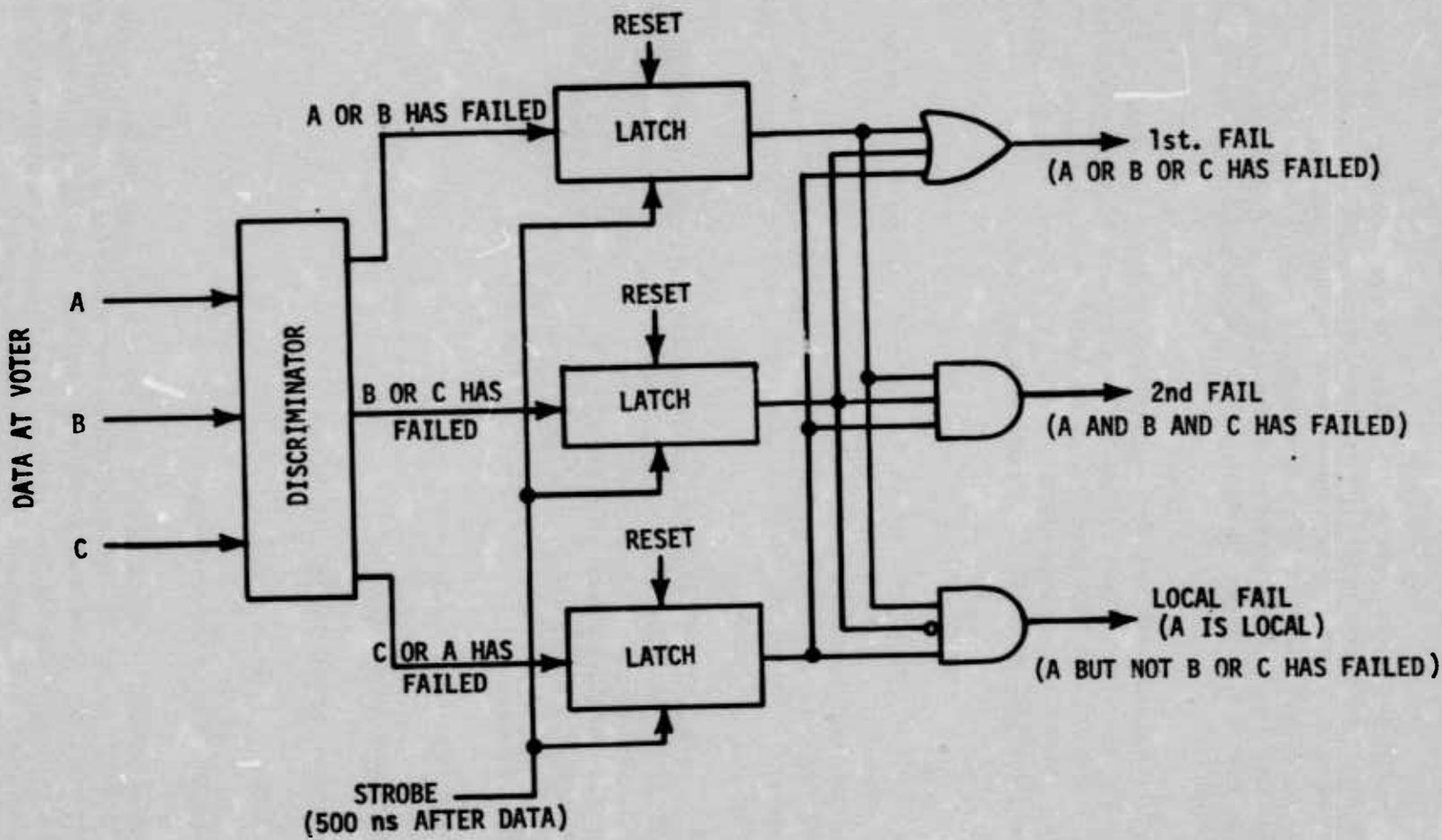


FIGURE 4-15.—MAJORITY LOGIC VOTER FAILURE MONITOR

4.3.2.1 Level One Functions

The circuit diagrams associated with each level one function were studied to gain the circuit/function understanding required to determine the function's significant hardware boundary. New drawings were then prepared that grouped the circuit details into the established functional boundary, and a systematic failure mode analysis began. Because the FCD task was not directed at developing a prototype or production system, the FMECA depth was carried to a level that, for some items studied, only provided an insight into the failure mode nature of the specific circuit/function. Several key areas, however, were covered in substantial detail (e.g., the clock and majority logic voter functions). These areas exemplify the technical differences between a digital and analog system.

The following is a summary of the level one functions FMECA results.

Redundant Clock

The ICPS fail-operational clock detail description and failure mode analysis is presented in section A.1. The clock circuits were considered a key area to be investigated because the design was an active standby operation, an operation (function) always suspect relative to having a single failure mode which may inhibit the switch over to the standby elements when the active elements fail.

The ICPS redundant clock scheme utilizes a primary/secondary oscillator configuration where the primary/secondary clock signals are routed to each of the ICPS interface units via external wire harness (ships wiring) paths. Both signals are independently monitored in each channel. If the primary clock signal fails, the secondary clock will be switched in. The design integrity of the circuit appeared to rest with the ability of the clock monitor to detect all failures. Rather than determining the clock signal generator circuit failures, the failure mode study began with investigating the monitor to determine if there were any postulated failure states that the monitor could not detect. If none could be found, then one does not have to investigate in detail the clock logic failure states.

The analysis of the primary clock signal monitor discovered that there was a failure situation whereby the monitor would not detect a particular clock waveshape change. This failure situation would permit the loss of the primary clock signal without detection; thus, the secondary clock signal would not be switched in, and the system would be without a clock signal (total system going dormant from a single failure).

As a result of this analysis, the clock waveshaping circuit had to be looked at to determine if a potential failure mode existed that would result in this waveshape change not detectable by the monitor. Such a failure state was discovered, providing a device failed during a specific time interval of the waveshaping circuit's periodic operation. Even though the specific device failure, coupled with the specific time period, could be shown to have a low probability of occurrence, the waveshaping circuit was reviewed to determine if a design change could obviate the failure mode.

A design change was developed that would remove the discovered failure state possibility. This change was incorporated into the ICPS hardware. Not only did the

modified circuit remove the potential failure mode, but with prudent selection of components, could reduce the waveshaping circuitry to one-third of the former design.

Majority Logic Voter/Monitor

The cross-channel data voter and its associated monitor were the next area of concern. Since this signal interface required electrical crossties between the redundant channels, the circuits become suspect for having failure states that could propagate from one channel to another in a daisy-chain fashion, or for having latent failure states that result in a simultaneous first/second failure situation. Results of the failure mode analysis showed that, of the possible failure states, 70% of the voter failures and 65% of the monitor failures go undetected during the processing of normal data (channel A = channel B = channel C). However, if the incoming data could be manipulated such that all incoming failure state situations were simulated, all voter failures and all but 10% of the monitor failures can be uncovered. Of the undetectable monitor failures, none represent a serious latent failure mode as they are associated with parallel redundant circuits which drive failure status displays (where one leg of the redundant path would most likely be removed in a production application).

By reviewing the voter/monitor detail description and FMECA in appendix A.2, it can be seen that for this relatively small circuit (14 elements), a complete analysis is no trivial task. The analysis did not uncover a design weakness that could be removed by a design change, but pointed out that the voter/monitor circuits must be tested to insure their integrity. The circuit test requirements brought out in the analysis become test specifications for the built-in-test/preflight-test function. No electrical failure mode could be postulated that would result in a daisy-chain failure mode propagation. This included considering the possibility of shorting a cross-channel data transmission buss to ground or to any power voltage level that would exist within the ICPS wiring interface. Such failure state possibilities were only treated from an analysis viewpoint since destructive tests were not a part of the FCD task laboratory work.

System Failure Status Logic

The system failure status logic functions were only lightly treated. The full implementation of these functions, as well as system control functions with respect to the man/machine interface, is very mission/vehicle dependent and goes beyond the bounds of the FCD task. However, the design utilized in this program does represent the typical nature of such logic and does provide some insight into associated circuit failure modes.

Appendix A.3 covers the system status logic description and associated FMECA. In general, the operational integrity of circuits designed to annunciate failures cannot be determined until the circuit is requested to respond to a detected system fault. Logic failures cannot undermine the fail-operational capability of the system but can miscue the flight crew as to the status of the system. Potential electrical failures (shorts to ground or shorts to power) on the three-channel system static logic lines which end up in close proximity to each other at the display panel, could effect system failure moding if the annunciating logic and system engage/disengage logic emanate from the same source. These circuits, in general, must be checked through built-in-test/preflight-test operations involving flightcrew participation.

Sensor Signal Selection/Failure Detection

All variable signals outside the ICPS computing subsystem pass through the signal selection/failure detection (SSFD) function. This function receives data from all three sensor sets (channels A, B, and C) and selects a single signal for each variable, which is then passed to the system computational sections. The incoming signals are cross-channel monitored which provide sensor failure identification and signal selection modification (median-select to average-of-two) when a fault is registered. A block diagram of the SSFD function is shown in figure 4-14. The SSFD selected signals are crossfed to the other channels of the system on a bit-for-bit synchronized basis, monitored and passed through a majority logic voter.

Two somewhat independent FMECA studies were made to investigate the SSFD circuits. Neither study resulted in an in-depth FMECA treatment. The first study began by developing a complete circuit level breakdown of the SSFD functions. This entailed constructing 12 detailed drawings, where each drawing covered the piece-part makeup of specific subfunctions within the SSFD circuitry. Specific piece-part failures were then postulated and the drawings were used to determine the failure propagation/manifestation. This approach soon became unmanageable because of the circuit complexity and associated time and manpower demands it placed on the FCD task effort. A second effort was therefore initiated to look at the SSFD function from a top-down functional point of view. In this approach, failures were postulated on a functional level and a further breakdown of the function only took place when a failure proved to compromise the SSFD operation. Details of this work are presented in appendix A.4.

From the investigations that were made, two generalizations became evident. First, since the SSFD circuits process data in a serial fashion, failures in the median selection logic, which is utilized to process normal data, would be detected at the downstream majority logic voter monitor. Second, since the proper operation of most of the SSFD monitor circuit only becomes evident when an upstream failure occurs, this circuit is prone to having latent failures.

It is easy to conclude that a functional test of all off-line functions must be conducted to verify the operational integrity of the SSFD monitor circuits. Off-line functions amount to 68% of the SSFD circuitry (5% associated with the signal selection mode switching and 63% associated with the monitor functions). Tests were derived that would result in checking the integrity of the off-line circuits on a preflight test basis. In fact, for the particular mechanization of the ICPS SSFD function, all off-line circuitry, except for the parity logic and failure latching logic, could be exercised (tested) during unused time slots within each 96 μsec SSFD processing window. Such a test could be implemented as an in-flight background process and would check the lag/washout computation and associated thresholds, failure count and associated thresholds, and SSFD signal output of both median and average-of-two for all combinational states of the three incoming signals. Running the complete test to check all time constant, threshold, and failure count values would take approximately 1 min. Test failures would be detected by the majority logic voting/monitor node downstream of the SSFD function.

4.3.2.2 Second Level Functions

Second level functions associated with failure modes and effect analysis are functions from a system configuration overview, whose failure cannot result in directly compromising the integrity of a system's fail-operational design. These functions are associated with the system's processes and circuitry that rest between signal selection or voting nodes and do not require the operation of adjacent channels to perform the function in a normal fashion. The FMECA conducted on these functions is concerned with identifying potentially latent failure states, which, when combined with other latent failure states, could result either in the system incorrectly responding to a detected failure, or in the system not detecting a performance degradation that could create a hazardous situation.

The following is a summary of the ICPS second level functions FMECA studies.

Input/Output Circuitry and Timing

Input/output circuits transform the signals between the outside world and the digital subsystem into compatible formats. The analysis of these circuits follow to a large extent the analysis methods used in conducting the FMECA for an analog system. Much of the circuit is dedicated to a specific function or signal path, and the multiplexing process does not alter the signal-path processes but stages (sequences) signals to and from the digital system through fixed time windows.

Failures in the input analog circuits (i.e., demodulators, prefilters, etc.) can result in a passive signal path (failed to zero), which may be difficult to detect depending on the failure detection threshold of the SSFD function. Such potential failures will require in-flight maneuvers at a sufficient level to uncover the dormant signal path, or an appropriate preflight test to verify the signal path integrity prior to system dispatch.

Failures of the A/D converters can cause all incoming signals to be plus or minus full scale or zero, or may cause specific signal level conversions to be wrong. Failing of the lower significant bits of an A/D process can result in the loss of signal resolution. Loss of resolution in two or more channels could allow undesirable airplane performance degradation if the signal path was involved with the stability loop of an unstable or lightly damped airplane dynamic characteristic. Detection of these lower-level bit failures may be difficult relative to the selected failure thresholds of the SSFD function. For example, if the SSFD monitor threshold was set at 5% of full scale, only failures in the four or five most significant bits of the A/D would be detected. Thus, 75% of the possible bit failures in a 12-bit A/D may fall below the monitor threshold.

Signals leaving the digital subsystem through a D/A converter can have failure induced degraded resolution like that produced by the A/D conversion. Here again failure detection is predicated on the downstream monitor threshold value.

Results of a cursory failure analysis of the timing logic (circuits) indicated that timing failures will generally produce massive miss-data management that would be detected by a number of system failure monitors. However, because of the multiple point usage of specific

timing logic, it is difficult to conduct a detailed hand analysis of the failure propagation effects. Therefore, it was recommended that failure response data relative to the major timing source logic be determined through laboratory tests. Since the timing functions are, in general, exercised iteratively as part of the ICPS hardwired executive, any failure in one channel should produce an out-of-sequence operation with the other two channels and thus be subject to detection by the majority logic voting node monitors. Details of the input/output and timing FMECA are presented in section A.5.

ICPS Computer

The ICPS computer performs the computational processes associated with mechanizing the control law equations. Within the computer subelements, data handling is a function of software controlled parameters (instructions). As a consequence, there may exist hardware elements (data paths) that may fail and go undetected because the software (resident program) does not utilize that function in the makeup of the control law equations. However, once the control law has been fixed, the computer, because of its multiplexed serial operation, is essentially monitored by the output majority logic voting/monitor node. The only areas that contain a potential for latent failures are the branching operation and initial condition operation. The latent failures associated with branching involves the setting of the most significant bit of the address register (the location where most branch loops would reside) and the integrity of the instruction bits residing in the upper reaches of the memory. The initialization latent failures are generally cleared (uncovered) during the computational reset operation at the start of the program execution. A detailed discussion on the ICPS computor failure analysis is presented in section A.6.

4.3.3 ICPS Preflight Test Requirements

As in the case with the analog subsystem, a number of hardware failure modes, which can go undetected during the processing of normal data, resides in the ICPS. These circuits are primarily related to error monitoring functions and signal paths that are only used after the detection of failures. Therefore, these circuit elements need to be subjected to a periodic test or preflight check to establish their functional integrity or be redesigned such that the failures become evident when they occur. It is impractical to redesign a system as complex as the ICPS with an intent that the system becomes 100% self-monitoring. Even if the ICPS elements could be monitored in such a way that internal latent failure states could not exist, such failures will exist within other elements of the flight-control system. Therefore, a preflight test (or system test of some nature, inflight, postflight, periodic, etc.) must be conducted to ensure a high level of confidence (high probability) that the system (flight-critical) is operationally sound. The following discusses the integrity check requirements envisioned for a preflight test on the ICPS.

The incremental computer cannot be programmed to function as an administrator of a preflight test operation. Therefore, the tests described, to a large part conceptual, assume appropriate test administration hardware. However, some laboratory tests were conducted using the ICPS hardware to verify specific preflight-test tests. These tests are described in appendix B. A block diagram of the test configuration is shown in figure 4-16.

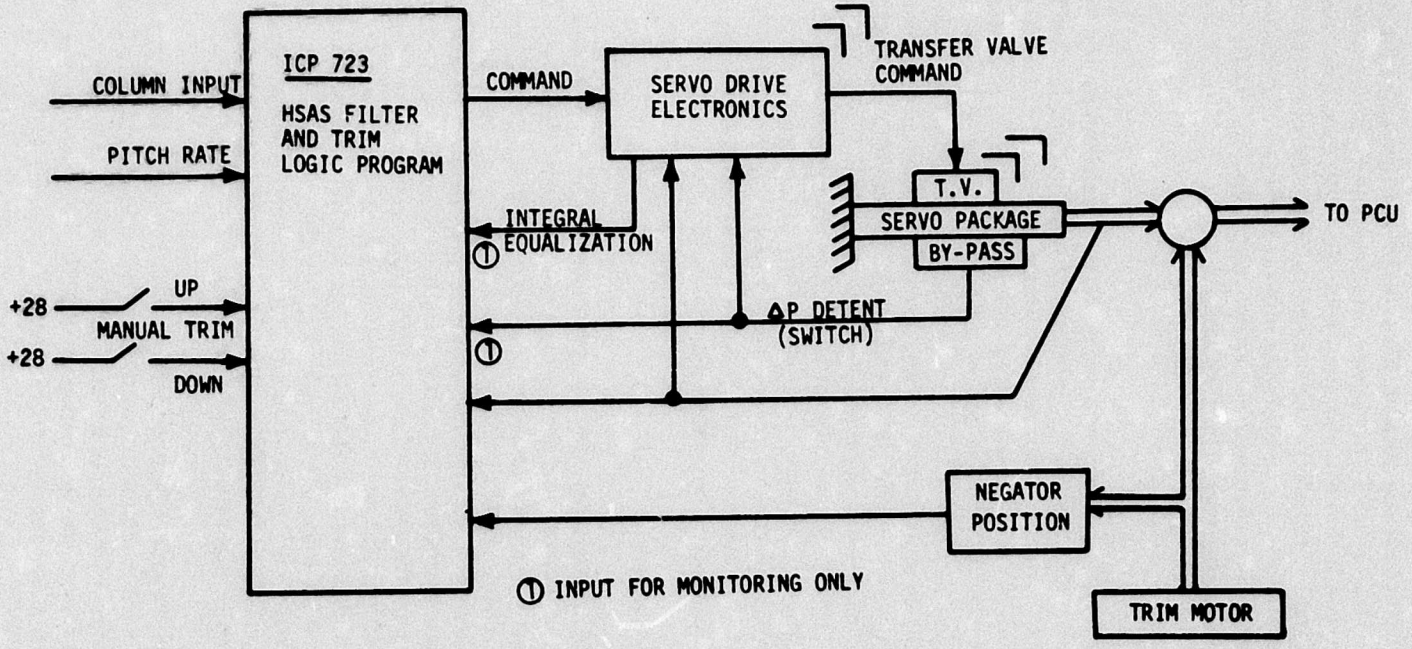


FIGURE 4-16.–PREFLIGHT TEST CONFIGURATION

Since the system monitoring is the key feature for ensuring system operational integrity, it is essential that the validity of the monitoring system be established in the initial phases of conducting the preflight test. The FMECA results pointed out that all possible data combinations were required on MLV's to clear the monitor of latent failures. In addition, the signal selection/failure detection circuitry operation must be completely verified. If built-in-test circuits were part of the subsystem design, a test program sequencer could be postulated, which would exercise all monitoring functions. This would include testing the parity, timing, power supply, and overflow monitors, as well as the system MLV and SSFD monitors. Tests of these items would fall into the category of *checking the checker.*

Thus with these considerations, the following sequence of testing events were postulated:

1) All electrical and hydraulic power on–system disengaged implies computational reset is active

2) Engage system–note that all test failure indicators are out

3) Activate ICPS preflight test–enables the test program sequencer and provides go/no-go testing of all monitors

4) Activate peripheral hardware preflight test–torque gyros, operate column (note elevator position plus or minus full throw), operate manual trim (note trim up or down operation)

To test whether the postulated preflight test would be sufficient to uncover system failures, several potential failures were tested in the laboratory. For each of the failures tested, an adaptation of the above procedures was utilized. Failures that were known to be active (i.e., sensor hardovers) were omitted from the testing exercise.

Categories of failures that were exercised included input/sensor failures, cable disconnects, and computer memory faults. In general, these types of failures were all detectable by the proposed test. Appendix B contains the tabulated results.

As expected from the FMECA analysis, the output MLV monitor caught pertinent program memory errors. Those memory faults not detected by the output monitor were ones that produced no change in the data. However, these faults were detected by the built-in read only memory (ROM) parity monitor.

4.4 WHOLE-WORD COMPUTER SUBSYSTEM (WWCS)

The WWCS application model laboratory configuration was essentially the same as that used for the ICPS. The signal selection and cross-channel data flow within the WWCS, however, are comparably different than that of the ICPS (fig. 4-17). Sensor signal-selection/failure-detection is processed in software, and the computer unit (CU) interface with the servo-subsystem is through a line replaceable unit, other than the

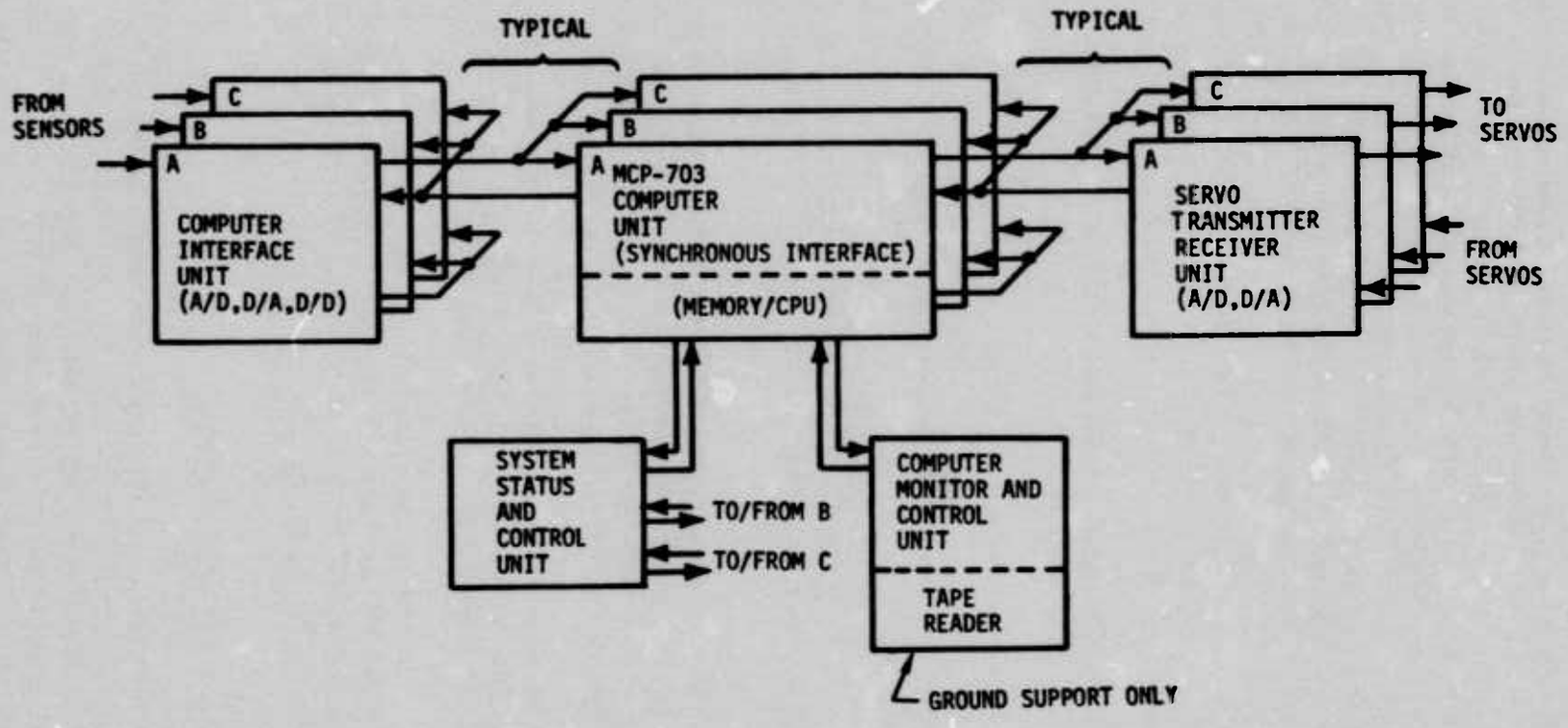


FIGURE 4-17.—WWCS HARDWARE INTERCONNECTION

computer interface unit (CIU), and is referred to as the servotransmitter receiver unit (STRU). The WWCS, however, does have bit-for-bit cross-channel data transfers identical to that found in the ICPS. In addition to this synchronous data transfer interface, the WWCS has the ability to transfer data between computer units under central processor control through a nonsynchronous serial digital interface.

Detail illustrations of the WWCS cross-channel signal interfaces are shown in figures 4-18 through 4-21. The first two, which show the CIU and STRU input/output signal interfaces, are identical except that the CIU contains the fail-operational clock logic where the STRU votes the clock information from all three CIU's. Figure 4-20 shows the computer unit's synchronous logic interface with the CIU and STRU. The significant difference between the computer unit to CIU interface and computer unit to STRU interface is in the number of computer unit output transmission paths. There exists only one output holding register/transmitter for transmitting data simultaneously to all three CIU's; three output holding registers/transmitters exist for transmitting data simultaneously to the three STRU channels. Thus, data that are being transmitted to the STRU can be individually tailored for each STRU channel.

Figure 4-21 shows the WWCS cross-channel interface, which is totally different from any within the ICPS. This interface permits a direct (under central processor control) computer-to-computer communication link. Its control (labelling), as well as data, is involved in the cross-channel transmission. Information transmitted via this route can gain the attention of the receiving central processor through the use of interrupt logic.

4.4.1 WWCS Monitoring

Like the ICPS, cross-channel signal comparisons are the basis for WWCS failure monitoring. Unlike the ICPS, however, the WWCS combines the failure state information for display in software not hardware (fig. 4-22). All the WWCS failure status information is placed in memory, appropriately combined under program control, and transmitted to the system status and control unit (SSCU) for failure state display. Failure status information from the CIU enters the computer unit through a device controller under central processor control. Failure status information from the STRU enters the computer unit via the synchronous logic interface and gets placed directly into the memory.

First, second, and local failure states are identified. The first and second states are generally system level failures and are primarily derived from the monitors associated with cross-channel interfaces. Local failure states are developed by combining, where possible, the logic of the first failure information and channel failure information (parity, power supplies, etc.) so that a channel can identify that the fault is within its elements.

The synchronous logic majority logic voter/monitors of the WWCS are identical to those used in the ICPS and are associated with intrasystem, WWCS, cross-channel signal paths. All redundant signals entering the WWCS are monitored using a software SSFD algorithm. Since software from the FMECA point of view cannot fail, in that it can only respond to hardware failures, the specific WWCS software SSFD used will not be described in detail here (ref. 1). However, the following FMECA results do point out the interrelationships between hardware and software relative to failure propagation and protection.

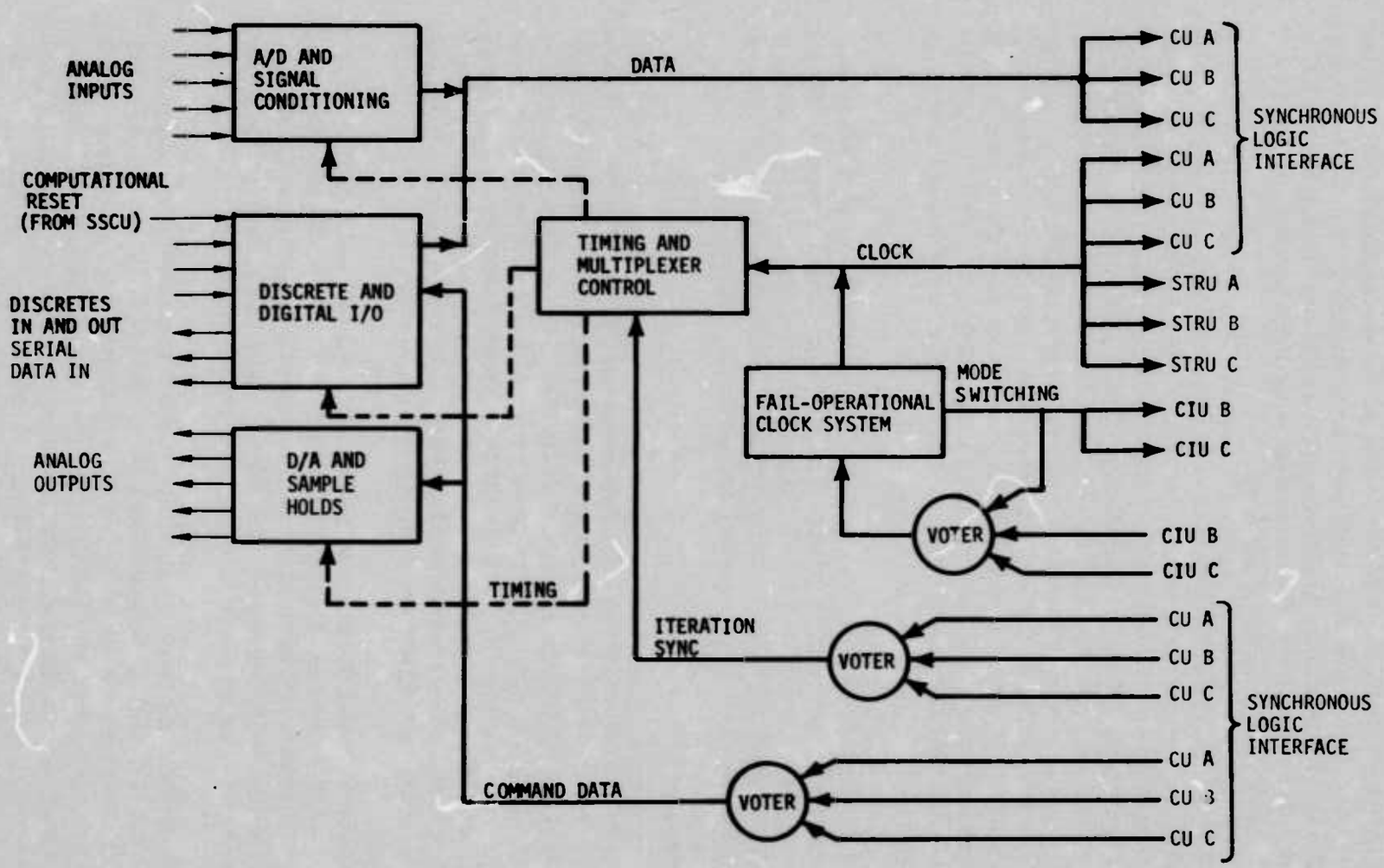


FIGURE 4-18.—CIU CROSS-CHANNEL INPUT/OUTPUT INTERFACE

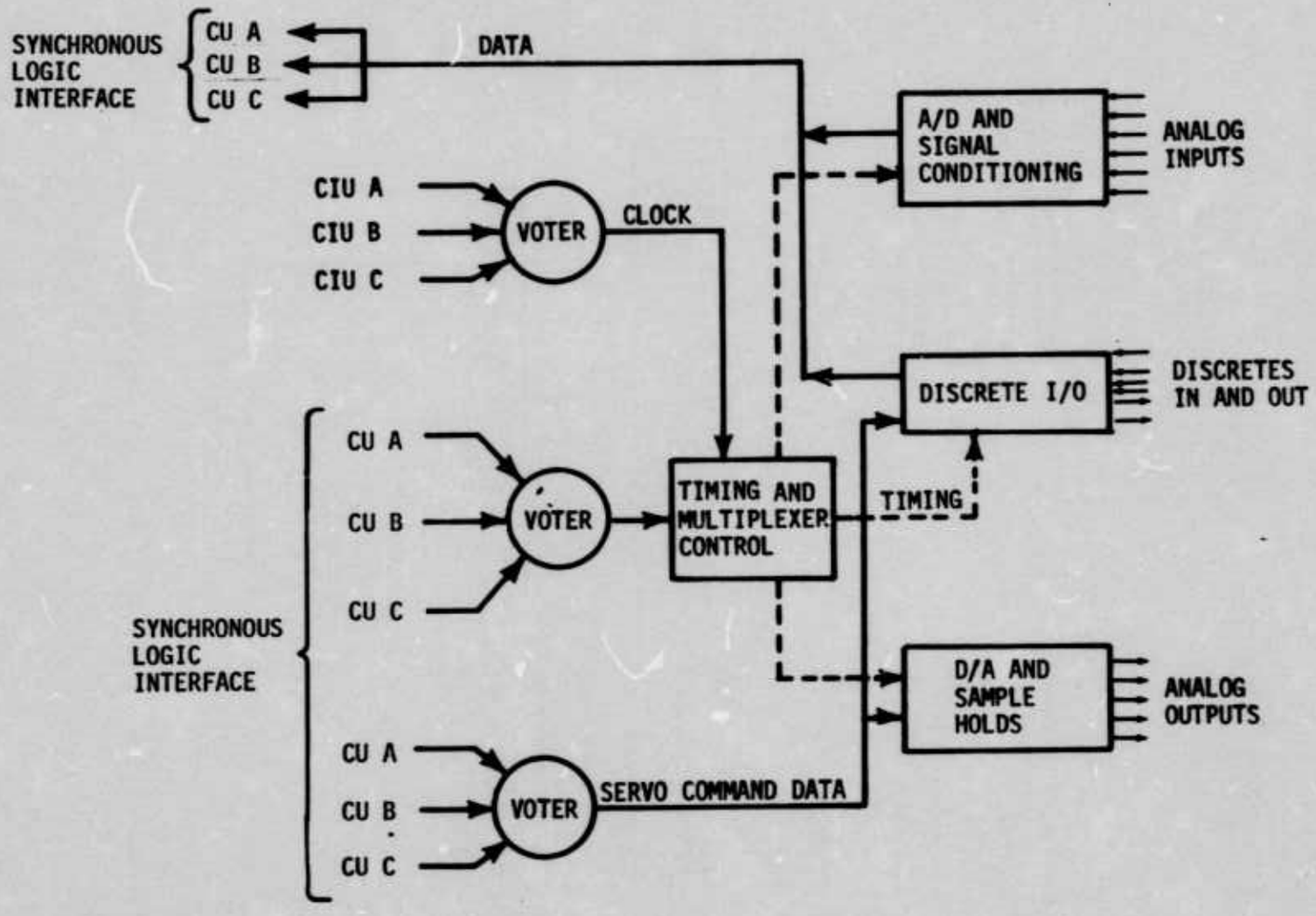


FIGURE 4-19.--STRU CROSS-CHANNEL INPUT/OUTPUT INTERFACE

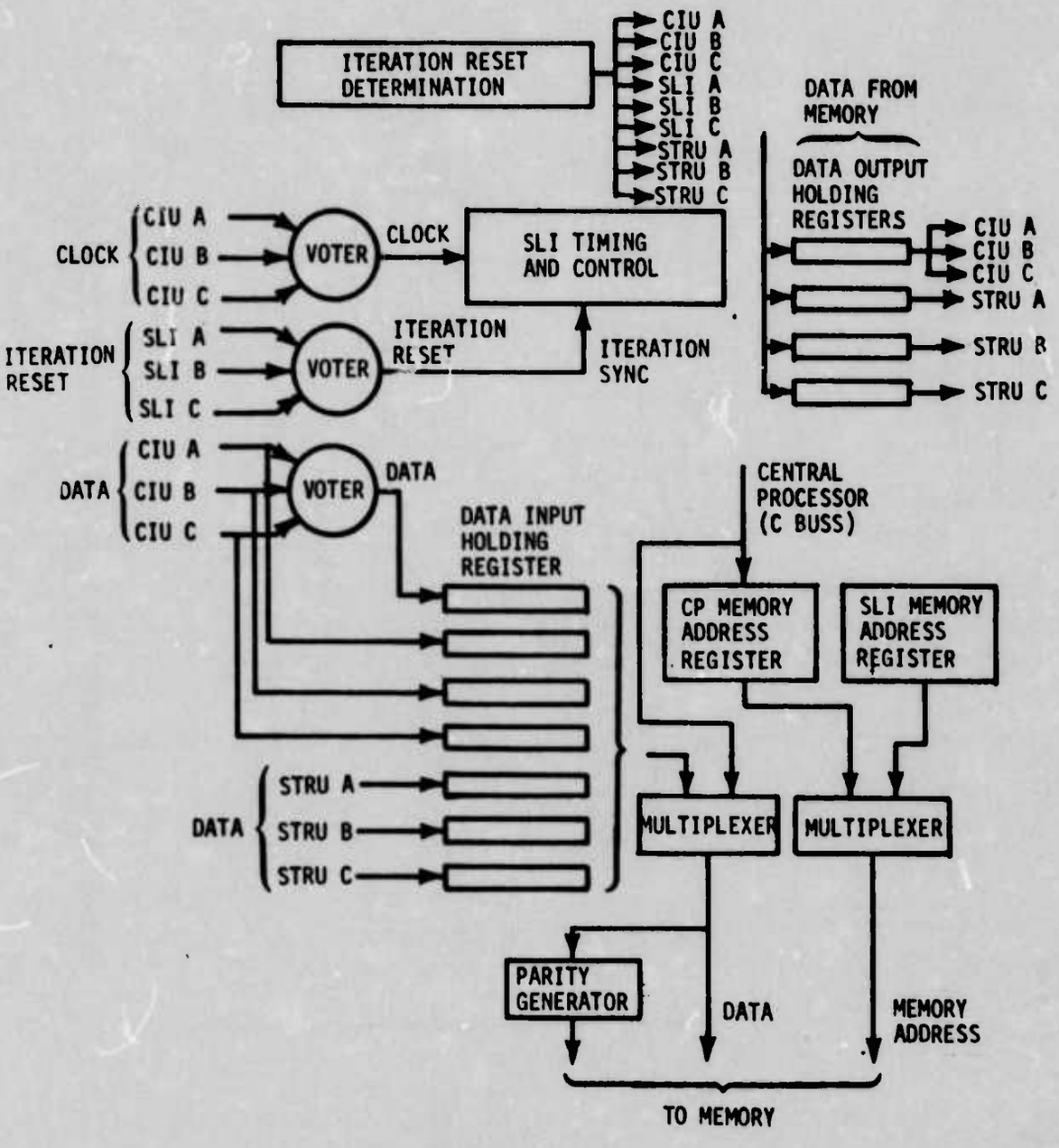


FIGURE 4-20.–COMPUTER UNIT SYNCHRONOUS LOGIC INTERFACE

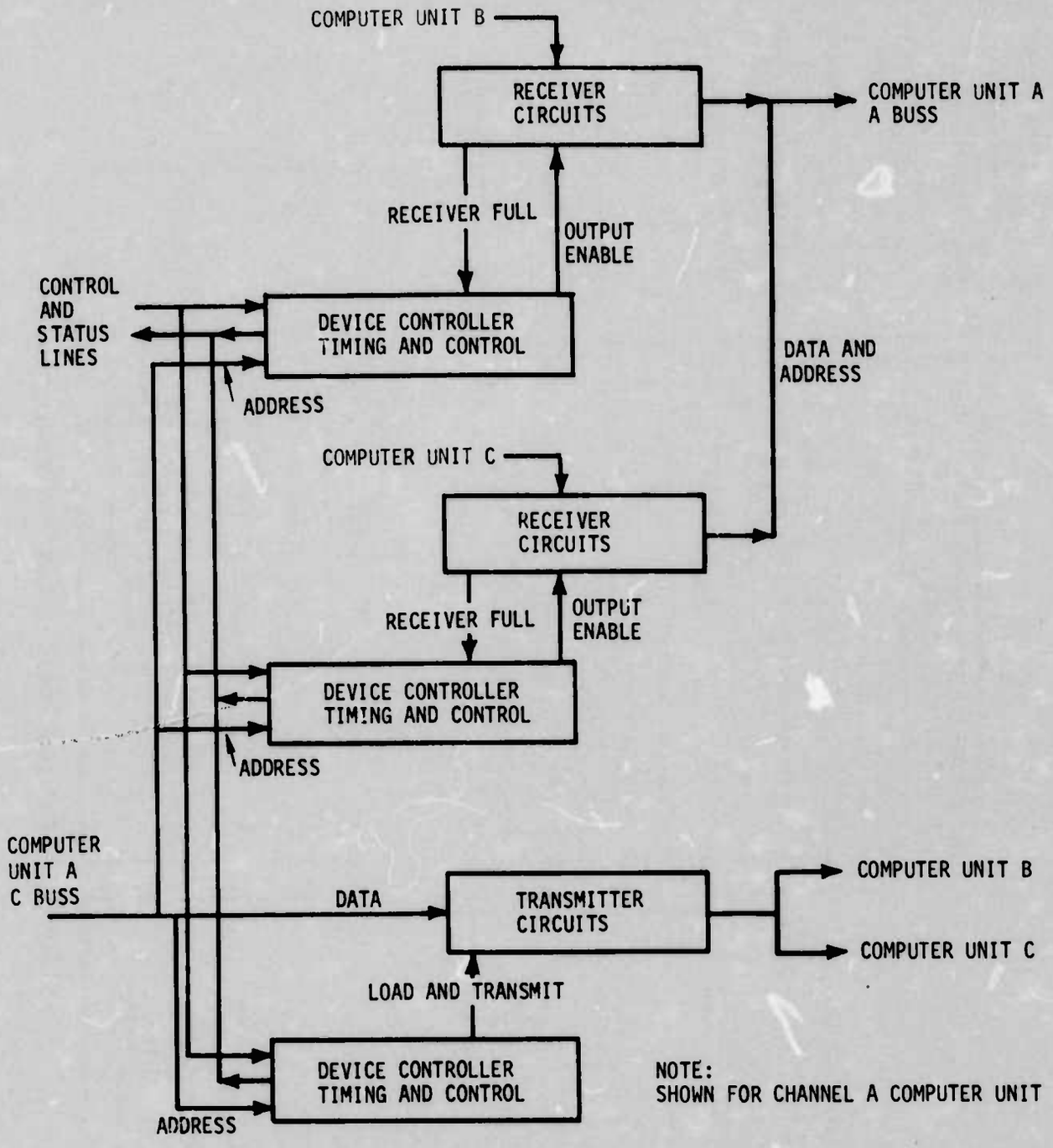


FIGURE 4-21.–CENTRAL PROCESSOR ASYNCHRONOUS CROSS-CHANNEL INTERFACE

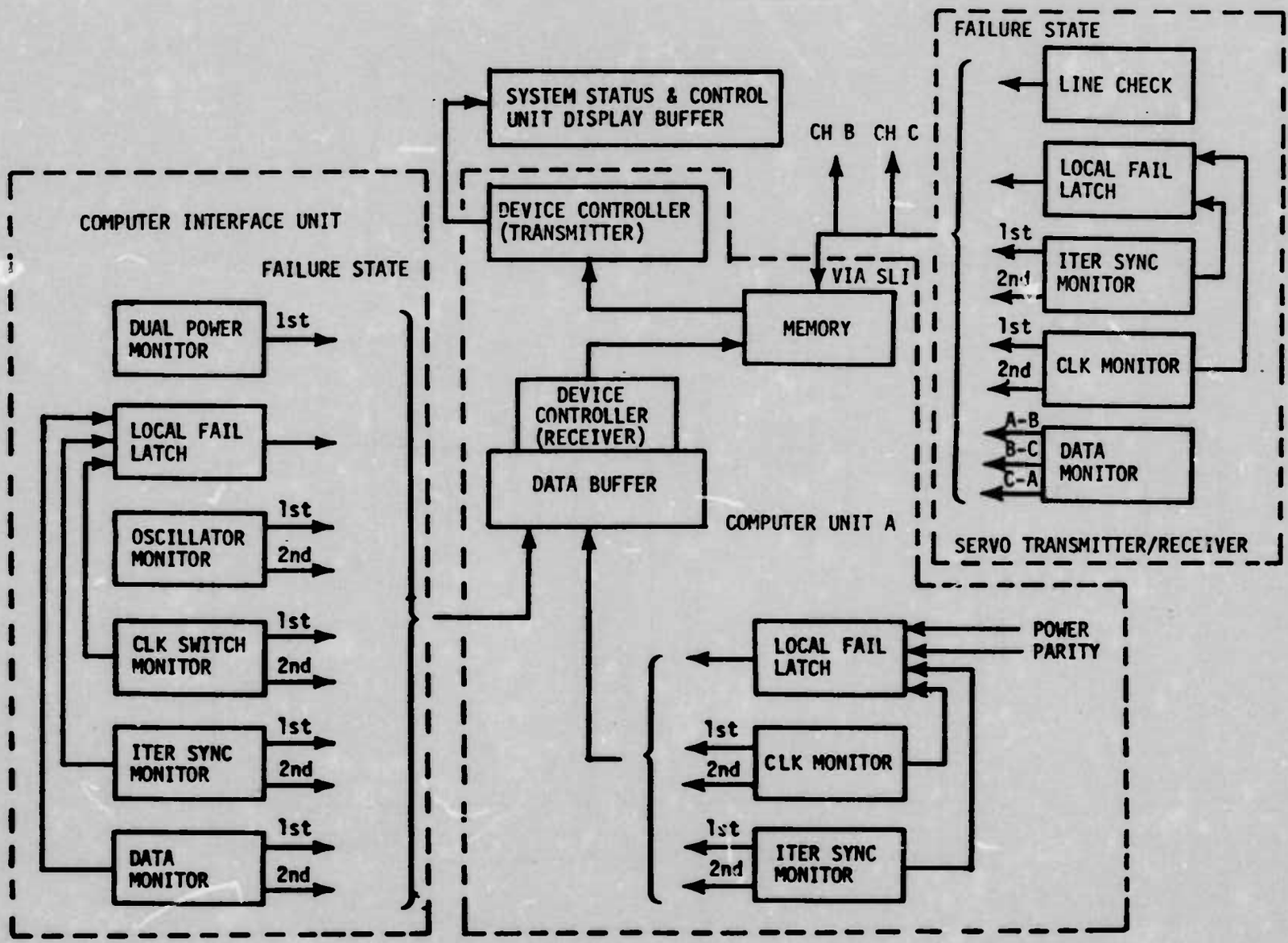


FIGURE 4-22.—FAILURE STATE INFORMATION PROCESSING

4.4.2 WWCS FMECA

Methods and techniques for analyzing the WWCS are essentially the same as those used on the ICPS and discussed in section 4.3.2. Therefore, failure mode study discussions on hardware functional subassemblies common to the ICPS, such as the redundant clock, majority logic voter/monitor, and input/output circuitry and timing, will not be repeated. A key area that is discussed, however, relates to how software can be used to alleviate potentially undesirable failure modes. The general purpose nature of the WWCS permits a great deal of flexibility relative to altering the hardware/software interface in light of failure mode situation awareness.

Appendix C contains the FMECA investigation data relative to the WWCS. Information presented in the appendix, like that for the ICPS, is placed in order of its assessed criticality with respect to understanding the WWCS fail-operational design integrity. Thus, the FMECA discussion is divided into a level one area of investigation covering the cross-channel interfaces, and a second level area of investigation covering the potentiality of latent failures in the redundant channel segments between cross-channel communication points.

4.4.2.1 Level One Functions

Level one functions for the WWCS, over those common to the ICPS, are the device controller cross-channel data link and the servotransmitter/receiver interface.

The device controller link constitutes a specific cross-channel (computer-to-computer) information communication path, where the other interface is associated with data handling circuitry between major line replaceable units within the WWCS. Therefore, the device controller interface FMECA provides an insight into a general class of asynchronous cross-channel data processing, where the other item relates specifically to the WWCS design.

The following is the level one functions FMECA results.

Device Controller (Cross-Channel Data Link and SSCU Interface)

The device controller cross-channel interface represents a major deviation to the bit-synchronous majority logic voter/monitor function in that the interface control, as well as the data, is transferred through this communication link. The central processor (CP) under program control loads and initiates execution of the cross-channel transmitter, where the signal train contains the control information to enable the receiving circuit to acquire the incoming data. The receiving CP under program control can fetch the incoming data at will or be forced to retrieve the data upon its arrival. It is this transmission/receiving process which must be scrutinized in order to determine if a failure in one channel can affect the operation of the other channels. Since servicing the receiver involves software, the servicing software program, as well as the interface hardware, must be clearly understood before the failure effects of the device controller cross-channel function can be established.

There are two modes in which the WWCS asynchronous cross-channel function can be utilized. One is through the use of an interrupt, where the device controller directly acquires the attention of the CP when the receiver has been loaded with incoming data. The interrupt

functionally works by forcing jump instruction into the instruction register following the completion of the instruction currently being executed. The address portion of the forced jump instruction, determined by which device controller interrupt has occurred, directs the CP to the starting location in memory for the appropriate program for servicing the associated receiver. In essence, the mainstream program is held up until the receiver is serviced. The CP, following the processing of the receiver routine, is returned to the mainstream task where corrections can be made on the mainstream data as the program picks up at the point from which the interrupt occurred. The second mode in which the cross-channel function can be utilized is upon CP (program) demand. In this mode, use of the device controller interface remains strictly under control of the mainstream program. The device controller interrupt is masked (the computer can be programmed to ignore an interrupt), and the receiver servicing takes place when the mainstream program desires the information that is held by the receiver.

In beginning the FMECA of the device controller cross-channel function, it becomes obvious that for the interrupt mode of operation a potential single point system failure could occur if a transmitter failure held the attention of the two receiving channels such that the interrupt routine was always on call. This could be overcome either by having a software or hardware escape capability, (i.e., setting a maximum time for the interrupt to be recognized), or by avoiding the interrupt mode altogehter. For a flight-critical system, the latter approach appears to be the more conservative route.

The control device receivers can be operated in the put-and-take (CP controlled) mode without the danger of a single point system failure hazard. For the case where other device controller interface functions require an interrupt mode, (e.g., to service ground support equipment), the hardware design would have to accommodate individual masking of the device controller interrupt function (the WWCS device controller interfaces utilize a single interrupt line).

For the FCD task, the WWCS cross-channel device controller interface was utilized in the noninterrupt mode. This cross-channel link was only used for the system redundant synchronization initialization, associated with the system computational reset function. Thus, the interface was only active during the execution of the mainstream code, which responded to the computational reset command discrete. Even though this WWCS interface was not used in the interrupt mode, there still exists failure states that could compromise the system operation, (e.g., latent failures in the interrupt masking logic). Therefore, future designs will have to be critically checked for their possible failure modes, which will have to be cleared using built-in or preflight-test tests. Details of the device controller cross-channel FMECA are presented in section C.1.

The WWCS computer unit to system status and control unit is another system interface where a device controller is used. This interface design does not include an interrupt mode and therefore can only be used on demand (PUT and TAKE) through the CP. Both the failure display transmitter and failure status receiver (failure status discretes) operate under mainstream code control. Any active failure in this interface is readily apparent (computer held in computational reset or lamp display locked on). The most prevalent potential latent is to have the error reset function fail to on. For this condition, all monitors within a channel will fail to register (display) a failure state. This condition can arise from hardware

failures or through software errors. Thus, the FMECA results support the requirement that a critical analysis must be conducted on both the hardware and software; and further, that great care must be taken in making software changes to avoid and to verify that software errors are not entered while implementing a software change. The SSCU FMECA details are covered in section C.2.

Servo Transmitter/Receiver Interface

The WWCS computer unit to STRU interface presents a potential hazard for level one system failures. The interface provides cross-channel data flow from each computer unit to all three channels of the STRU and from each channel in the STRU back to all three computer units; therefore, two situations exist where a single channel signal is feeding a redundant computational string. Going from the computer unit to STRU, a potential single point failure is avoided through the use of majority logic voter functions. However, in analyzing the STRU to computer unit design, there does exist a potential failure mode that could compromise the fail-operational design of the WWCS.

The STRU to computer unit interface is a digital serial data path that transports eight 16-bit data words of information. Seven words are associated with A/D converted variable signals, and one word is used for discrete and failure status information. The variable analog signals are converted to a 12-bit digital signal accuracy and packed into the lower portion of the 16-bit transmitted serial word. Each data word is simultaneously received by all three computer units and placed into memory via the synchronous logic interface direct memory access operation. With this design, it is possible to develop a failure in the data transmission circuitry that can transfer the 16-bit data word askew relative to the computer unit receiving timing window. Since the STRU interface is designed to receive servocommands tailored to each STRU channel, it is implied that each computing channel would utilize servo feedback data in its raw form; i.e., nonsignal selected in computing the STRU channel commands. If the above failure shifts the 12-bit data into the 16-bit memory location such that the signal significance has increased or the sign convention is altered, a potential exists to encounter a computational overflow in the subsequent processing of that data. Since all three computer units would be processing common data, there is a possibility that the single signal fault could force all three computers erroneously into an arithmetic overflow protection routine or other subroutine that may cause the system to lock up. For such a design, precautions must be taken in hardware or software to mask the upper bits of the 16-bit data word or to provide reasonableness tests (or cross-channel comparisons) on the data to ferret out failure states. Again, software may be involved in providing protection against undesirable failure modes. Detail STRU FMECA results are presented in section C.3.

4.4.2.2 Second Level Functions

Some second level FMECA functions of the WWCS are identical to those in the ICPS. These are primarily the A/D and D/A processes. However, failures within these areas were reviewed to determine their propagation effect into the downstream function. For instance, if an averaging signal selection algorithm (in software) was used, lower order bit failures in the A/D process would be passed to all computer units and would influence the selected signal value. The allowable error band would be established by the associated SSFD monitor thresholds.

The other WWCS second level functions investigated were the synchronous logic interface (SLI), memory, and central processor. As previously mentioned, an in-depth (exhaustive) FMECA study was not conducted; rather, each functional area was looked at to gain an insight into the function's latent failure potentialities and to acquire some insight into the required analysis approach. The following summarizes the FMECA studies conducted on the three WWCS functional areas just mentioned.

Synchronous Logic Interface

The SLI is a hardwired function that processes all bit-for-bit synchronous logic data into and out of the computer unit memory. Dedicated memory locations are allocated to accommodate the incoming and outgoing variable or discrete information. Since the SLI circuitry is active in nature and involves a multiplexing operation, failures within this circuitry tend to be discoverable just from the processing of normal data. In fact, no failure mode could be uncovered that would not result in the modification of incoming or outgoing signals. Therefore, faults in one-channel SLI would result in placing data into its computer unit memory different from that placed in the memories of the other two computer units or would result in different data being transmitted from the faulty computer unit's SLI. Errors on the input data, which only effect the lower order bits (signal resolution degradation), may not be detected by the software SSFD, but the errors would ultimately be picked up by the output majority logic voter/monitors located in the STRU and CIU, which will detect any bit-for-bit error on computer unit outgoing data. A more detailed SLI FMECA discussion is presented in section C.4.

Memory

The WWCS memory is builtup utilizing ferret cores. The memory is used to store the WWCS operational program (execution instructions for the central processor), input and output variable and discrete information, program constants, and intermediate data relative to calculations controlled by the operational program. The WWCS memory system is a destructive read/restore core memory with a built-in parity monitor.

The memory system has three potential latent failure areas. The first is whether the parity checker checks for parity. The second is associated with the loss of data in a core cell, either because the cell magnetic orientation has changed or the cell itself has cracked. The third involves the circuitry that must respond to the parity error detection. Core cell errors can only be detected when that cell, as part of a program word in memory, is exercised (read). Therefore, core cell failures can go undetected for extended periods of time if that portion of the program is not utilized during normal system operation; e.g., if the program function deals with the system operation following a system failure. Such failures can be detected by exercising a complete memory parity check as part of the preflight test or during a running background (self-test) program.

Parity error detection creates an interrupt to the central processor that forces the computer into a memory fault routine if a higher priority interrupt is not present. Parity error does not prevent the continued processing of instructions; i.e., recovery from a parity error is handled in software. Following the servicing of one parity error or other arithmetic fault interrupts, an instruction must be executed which clears the arithmetic fault logic. The

interrupt structure within the central process is mechanized such that only when there is a change in interrupt status between instructions will the program actually be interrupted. Therefore, if the *clear arithmetic fault* instruction is not processed after an interrupt has been serviced, successive parity failures would not be passed on to the central processor. Further details on the memory FMECA are presented in section C.5.

Central Processor

The CP is made up of circuits that control the interpretation and execution of the program instructions stored in memory. The instruction interpretation and processor register control is developed through read-only-memory microprogramming. As the details of the central processor circuitry were being laid out, it became obvious that a bottoms-up (piece part) analysis of this function could not be accomplished with the resources available. However, experience todate has shown that an effective FMECA can be pursued on a functional level. Therefore, an analysis of the WWCS CP was conducted based on a functional level study referred to as a middle-up approach. The basis for this approach was to look at the CP at the instruction level since a general purpose digital computer is nothing more than a device for sequentially processing given instructions. Each instruction can be described as a series of operational (functional) steps, where each step can be subject to a failure. Depending on the hardware design, the particular step failure may be related to a failure-in-one or one-of-many piece-part elements utilized in the mechanization of that functional step.

The failure mode analysis can be conducted in general relative to studying each instruction capable within the CP or can be conducted for only those instructions used in the operational program. Whichever is the case, the failure effect part of the analysis must include the resident software program in order to ascertain the propagation effect of the failure mode. A failure state for each instruction step would then be classified as being immediately detectable, latent, or trivial. Examples of the various failure classifications for given instruction steps, from the study conducted for the FCD task, follow.

One step of the MCP-701 central processor load upper register (LDU) instruction modifies the sign adder (SA) indicator bit of the status register. A failure in this step was classified as trivial if the significance of this bit was not utilized by any other instruction of the application (operational) program. Another step in the LDU instruction is to address a specified memory location whose content is to be placed in the upper register. If the desired memory location is not selected (step failure) and the data in the wrong memory location does not correspond identical to that in the right location, the step failure was classified as immediate since tracking through the application program showed that the failure would result in an output command different from the other two processors. This difference would be detected by the WWCS output MLV monitors.

An instruction step failure, which was classified as a possible latent failure, was one involved with the multiply (MPY) instruction. The step deals with multiplying the upper register by the contents of a specific memory location. Depending on the multiply instruction design and hardware implementation, a failure, which would only effect a unique solution, could occur; i.e., the solution of a specific number in the upper register multiplied

by a specific number in memory may be wrong. For a 16-bit multiplier, this may effect only one solution out of 2^{32} possible solutions. This type of latent failure case must be looked at in detail by analyzing the involved hardware or by defining a background self-test routine, which will periodically test all multiply states.

The FMECA exercise conducted on the WWCS central processor is presented in section C.6. Even though this exercise did not consider the entire application program used for the FCD task, it was concluded that the middle-up analysis approach had merit and should be applied in a critical system's FMECA as soon as the system's hardware/software definitions become visible.

4.4.3 WWCS Preflight Test

One of the primary advantages identified with a whole-word (general purpose) digital processor in a flight-critical control system application is its potential to manage an overall system test with a minimal impact on hardware additions. However, before a processor system can be relied upon to administrate such a system test, it must first be able to ensure its own integrity. For the WWCS development as part of the FCD task, a built-in-test (BIT) feature was specified that would functionally establish the operational integrity of the WWCS fail-operative design. This BIT capability was to check those elements within the WWCS that would have to be proven operational during a preflight test (dispatch test) of a flight-critical system.

The following sections provide a summary description of the WWCS BIT, where BIT refers to the set of hardware/software developed within the WWCS to, under program control, establish its own operational integrity. A basic requirement for the WWCS BIT feature was that the BIT function had to perform a self-test on the WWCS to a level which would assure that the subsystem could reliably administrate a preflight test on the entire flight-critical system. Thus, BIT was looked upon as the first series of test sequence that was to establish the flightworthiness of the flight-control system.

4.4.3.1 BIT Implementation Overview

The structure of the WWCS (MCP-703 system) and the nature of the flight-critical system application problem led to a fundamental decision to implement the BIT function wherein all computer units would play an equal role in the BIT processes. This decision was founded on the principle that all computer units had to be operational to achieve a dispatch state, and the idea that tests employing the system redundancy vs tests using singular reasonableness criteria provided the greatest level of test success confidence. Therefore, all computer units were to have an active role in the administration of BIT and were to interact (compare test success) at each test step. The BIT implementation philosophy and scope are summarized by the following BIT development guidelines:

- Testing was to be performed in all three channels simultaneously.
- All processors (computer units) were to play an equal role and interact at each test step.

- All WWCS voters and their associated monitors must be cleared of potential latent failures.
- A minimum of simplex testing should be employed.
- The design should entail a minimum of operator (flightcrew) intervention.
- Where possible, functions tested should be used in testing other functions.

Basic control and display functions to operate BIT and to ultimately operate the preflight test program were placed on the WWCS SSCU. These functions (fig. 4-23) were designed to be representative of those functions that may be required on the application vehicle's flightdeck. They consist of a rotary switch (fig. 4-23) labelled INTERLOCK, a guarded toggle switch labelled INITIATE, three channel NO-GO indicators, a test-in progress numeric display, and a push-to-continue button.

The Interlock switch controls the discrete data signal paths used to induce failures into the WWCS hardware/software. The function is representative of the interlocks that would be imposed in a production implementation to circumvent inadvertent initiation of BIT in flight. Without the Interlock closed, BIT/preflight test cannot be initiated. For a production system, the interlock function would be controlled by using landing gear ground sensing switches and/or by other aircraft configuration sensing fail-safe logic.

The Initiate switch activates a discrete through the interlock switch which creates a BIT initiate flag in the computer. In response to the BIT initiate flag, the executive program jumps to the BIT software routine and begins to automatically sequence through 147 specific hardware/software test steps. These tests induce failures, check for proper failure detection, remove failures and reset monitors, and retest for normal operation.

The three-digit numeric display annunciates the current test in progress. Should a test require operator action, the test number denoting that action will remain annunciated until the action is taken. Should a test fail, the appropriate channel no-go light will be illuminated, and the BIT will stop with the failed test number displayed.

The continue test button is used to acknowledge specific BIT tests calling for operator recognition and to continuing BIT following a failure indication once the failed test number has been recorded.

The WWCS functions checked by BIT are:

- SSCU light-emitting diode display
- Central processor arithmetic fault detection and interrupt
- Memory fault detection and interrupt
- Cross-channel data parity monitor

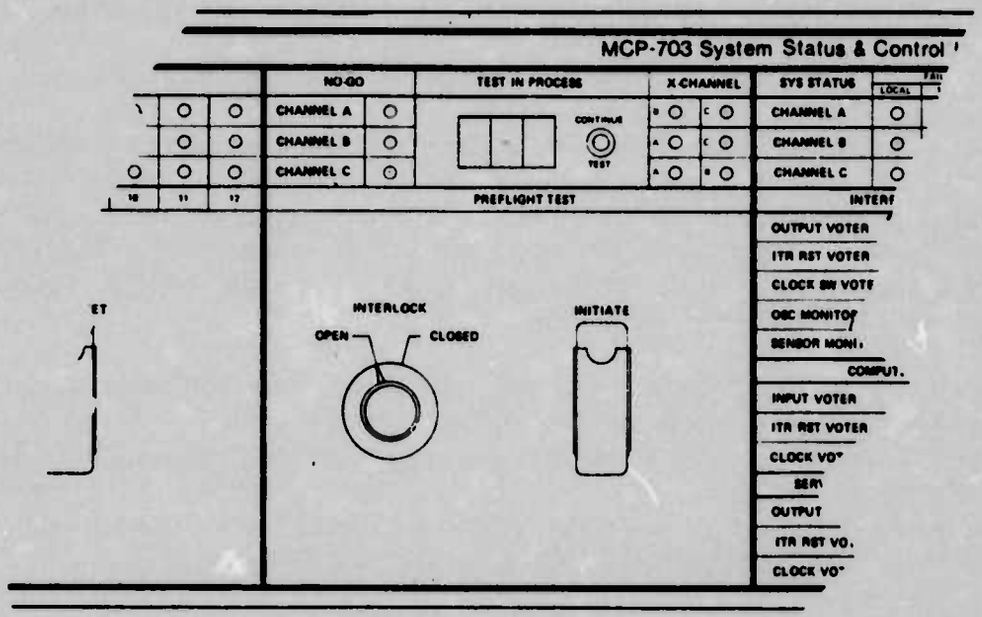


FIGURE 4-23.—BUILT-IN-TEST/PREFLIGHT TEST CONTROL PANEL

- Parity check for all of memory
- All majority logic voters and associated monitors
- Fail-operative clock switchover circuitry
- A/D and D/A circuits
- CP oscillator and input/output (I/O) oscillator frequencies

The computer's cross-channel data link provides the means to reliably compare the success of each BIT test step. Each computer initiates a given test stimulus, obtains test results, and trades the test success status with the other computers via the cross-channel data link. If a test proves unsuccessful, all three computers stop processing BIT, and identify and output the failed test number. The computer, which is able to locate the fault illuminates its no-go indicator.

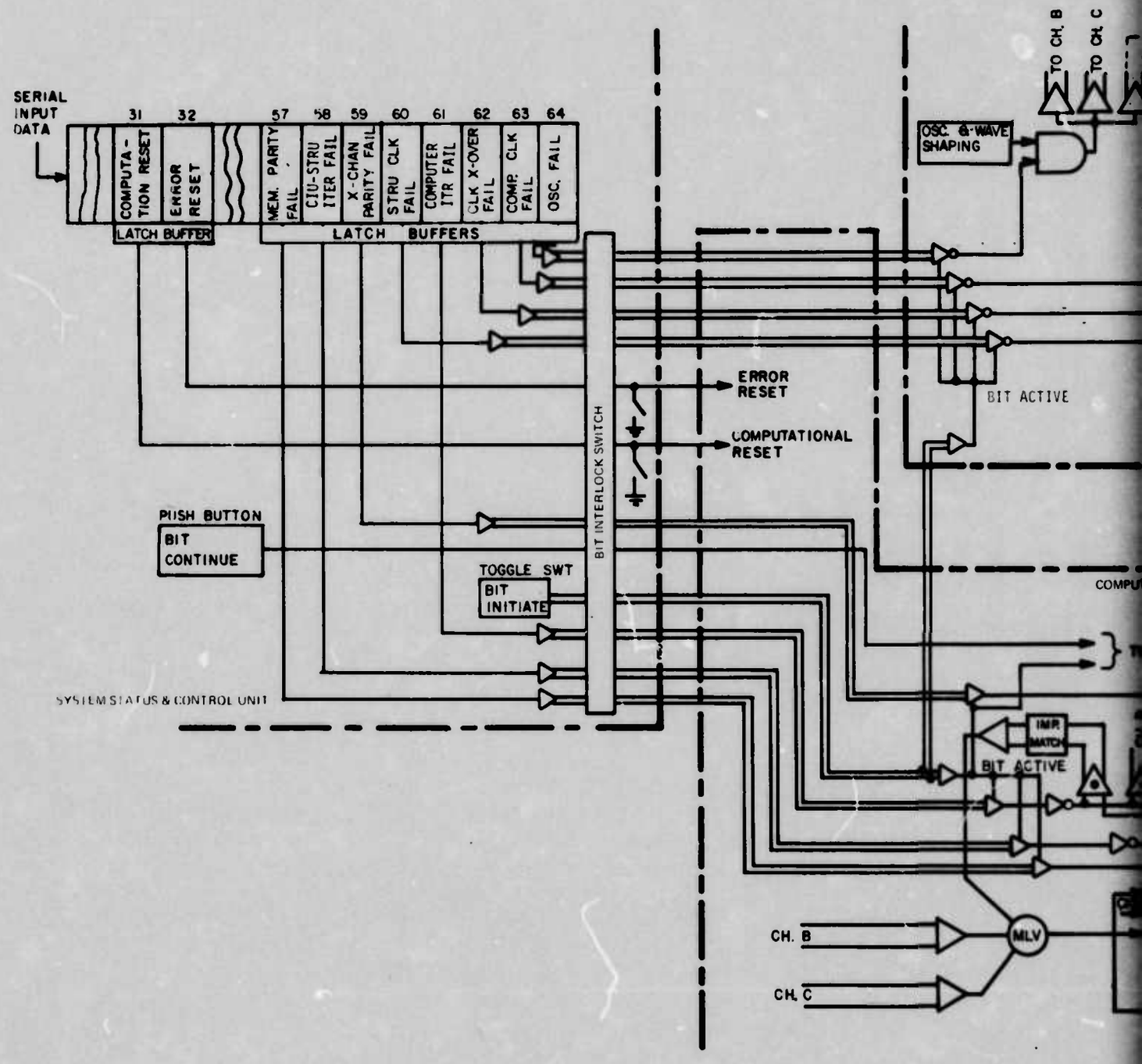
4.4.3.2 BIT Mechanization

In order to conduct the WWCS integrity checks required, as a result of the FMECA studies, additional hardware elements were added to the computer subsystem. These elements (fig. 4-24) allow the insertion of simulated failed signals so that potential latent failure states can be cleared. Failure inducing hardware was added to the following functional circuits:

- Primary/secondary oscillator
- Primary oscillator monitor (to test clock switchover voter/monitor)
- CIU clock output transmitters (to test clock MLV/monitors in CU's and STRU)
- Iteration reset drivers (to test iteration reset MLV/monitors in CIU's and CU's)
- Cross-channel data link parity generator
- Memory parity generator

In addition to these failure stimuli points, the system computational reset and error reset functions were modified to permit their activation by the software program. This substantially reduced the required BIT/operator interaction.

To prevent an inadvertent inducement of failure stimuli, BIT signals are controlled by a double interlock. First, all stimuli signal paths pass through contacts controlled by the interlock switch. The stimuli signals are all discretes originating in the BIT program, transmitted serially to the SSCU, and converted to discretes routed to the interlock switch. The second interlock is the BIT initiate switch. This switch must be activated to enable the BIT discretes and to initiate the processing of the BIT program. However, the initiate switch can only start the BIT program if the interlock switch is closed.


SERIAL INPUT DATA
31
32
57
58
59
60
61
62
63
64
COMPUTA-TION RESET
ERROR RESET
MEM. PARITY FAIL
CIU-STRU ITER FAIL
X-CHAN PARITY FAIL
STRU CLK FAIL
COMPUTER ITR FAIL
CLK X-OVER FAIL
COMP. CLK FAIL
OSC. FAIL
LATCH BUFFER
LATCH BUFFERS
BIT INTERLOCK SWITCH
ERROR RESET
COMPUTATIONAL RESET
PUSH BUTTON
BIT CONTINUE
TOGGLE SWT
BIT INITIATE
SYSTEM STATUS & CONTROL UNIT
OSC. & WAVE SHAPING
TO CH. B
TO CH. C
BIT ACTIVE
IMP. MATCH
BIT ACTIVE
CH. B
CH. C
MLV

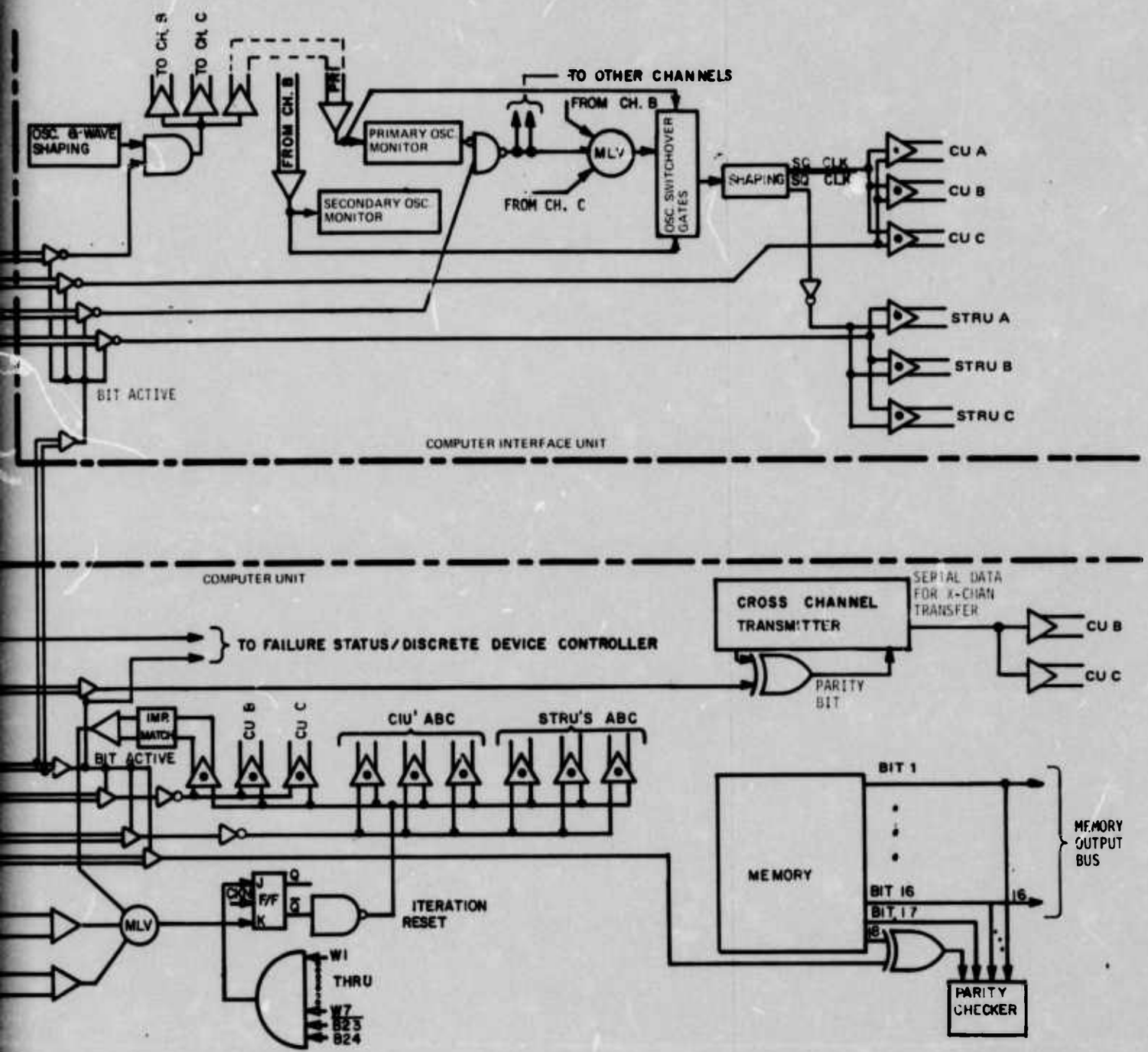


FIGURE 4-24.—WWCS BIT MECHANIZATION

The BIT control commands enter the computer unit through a discrete device controller interface. If the WWCS processors are executing a flight-control (real time) program, the BIT initiate flag is checked by the WWCS executive just after the I/O timer interrupt. Thus, when an operator initiates BIT, it will not commence until the end of the current I/O frame. The BIT initiate discrete continues to be interrogated after each interrupt, while BIT is being executed in order to permit exiting from BIT at the discretion of the operator. If a failure is detected, the processors jump to a wait loop, and the appropriate test number and no-go indicator are illuminated. The BIT program can be continued by activating the continue test button. This button is also used to continue BIT after a specified operator action has been satisfied.

4.4.3.3 BIT Sequence

The following is a summary of the WWCS BIT program execution sequence. A test-by-test detail description of BIT is presented in appendix D.

The program starts by testing the system status and control panel displays; i.e., sequencing the illumination of the display elements controlled by each channel (channel A, B, and C). The program waits as each channel drives the display until the operator acknowledges observing the display by pressing the continue test button. This is followed by internal computer unit test centered around the CP's arithmetic fault interrupts and memory fault (parity) interrupt. These tests check the central processor/memory monitors and then use the monitors to watch over the related hardware elements as they are exercised by the processor.

At this point, the triplex operational features of the WWCS begin to be checked. The computer's cross-channel data-link parity monitors are tested and checked. The synchronous logic interface clock voters and monitors are tested followed by the testing of the STRU clock voters and monitors. These circuits are systematically faulted to exercise the first, second, and local failure detection/annunciation capabilities. Further, the MLV's involved with these circuits are completely exercised for proper input and internal latent fault screening by systematically permuting the input first and second fault states for all input signal combinations.

The next series of tests deal with checking the integrity of the primary/secondary I/O oscillators and associated monitors and switchover logic. The secondary oscillator is faulted first, and a check is made to see that the failure is detected and that the I/O timer interrupt continues. Then, the fail-operative clock switchover logic is tested. The clock switchover voter is disabled, and the primary and secondary oscillators signals are systematically faulted. Proper response is determined by checking the timer interrupt signal. Then, all combinations of oscillator signal failures are exercised along with simulated failures of the switchover voter inputs to clear the switchover logic of latent failure potentialities.

The above test series is followed by a check of the iteration reset voter/monitors located in the computer units SLI. Again, proper response is determined by checking the timer interrupt iterative operation. (It should be noted that a proper monitor response requires that only specific failures be detected, wherein all monitors are checked during each significant test step.) The iteration reset voter/monitors in the computer interface units are

then checked. These checks are made by systematically controlling the SLI operation (disable/enable) one channel at a time in permuted combinations and observing the timer interrupt signal loss and resynchronization ability.

The final triplex tests are to verify the integrity of the computational data paths. These are achieved by loading different data into the computer units output buffers and checking for the proper response on the output majority logic voter/monitors. Again, all combinations are permuted.

The final test series are simplex in nature. An assessment is made of the analog interface A/D and D/A operation, and the synchronous clock and CP clocks are compared. The A/D and D/A checks are accomplished by closing a path between one input and one output of the analog I/O channels and by checking the relative signal conversion accuracies. The clock timing checks are made by counting the CP clock pulses between I/O timer interrupts.

The above WWCS tests were segmented into 146 specific test steps. The total test sequence processing time is approximately 18 sec, discounting the operator intervention time at the beginning during the display tests.

4.4.3.4 BIT Evaluation

A number of hardware elements within the computing subsystem are not checked, or at least not exhaustively checked by the BIT program; e.g., the power supply monitor, the entire instruction set, input signal conditioning circuits, and all sample and hold output circuits. Some of these were omitted because an appropriate FMECA had not been accomplished. The input/output functions were omitted because it is impractical to include such functions in a subsystem test; their testing was relegated to be a part of the overall flight-control system preflight test.

The current BIT program resides within two pages of the WWCS core memory (1024 words). With the visual inspection of the SSCU displays made and the appropriate operator action taken to continue the test (required 3 times), 30 sec represents a nominal execution time for the BIT program. The program, however, was not optimized and contains several delay processes, which were added to overcome hardware anomalies existing during the initial stages of the hardware/software integration checkout effort. The operating time could be reduced by approximately 6 sec if the program coding was recycled (refined). The initial hardware anomalies were corrected before the equipment was delivered.

The BIT program played a dual role during the development of the WWCS equipment. It provided an engineering exercise to gain experience in defining such a function, and it was used to systematically check the system operational state during the system hardware/software integration activity. The software development of BIT was completely independent of the actual hardware design; i.e., it was prepared from the system functional specifications and not drafted from a knowledge of the actual circuit designs. Therefore, it became instrumental in uncovering differences between the functional design intent and the actual circuit realization of the specified function. The differences were then resolved by reviewing the specifications or by making appropriate corrections to the hardware. What was

important about this process was that design inconsistencies were discovered and understood early in the total system integration period, much earlier than anticipated if BIT were not a subsystem requirement in the acceptance test of the hardware.

Another important aspect of BIT has been its ability to serve as a standard between equipment failures and application software programming errors. In the development of the operational software, BIT could be quickly used to determine if suspect hardware faults were really programmers software frustrations. Its use and value can be easily scoped when the alternative is to setup an oscilloscope or other pieces of laboratory equipment to check the hardware against gripes. The BIT program, as implemented, cannot be used as a good hardware failure diagnostic routine since it was designed as a systems go/no-go checker.

5.0 FLIGHT-CRITICAL SYSTEM PREFLIGHT TEST

The automated preflight test function within a flight-critical system is comprised of two parts. The first part is associated with the self-test, or BIT capability internal to the primary electronic subsystem. The second part is the system test operations that are required to conduct an integrity check of the peripheral hardware elements that interface with the central electronic subsystem. This latter part assesses the health of the flight-control hardware associated with sensors, servos, flightcrew controllers, etc. Since the previous sections discussed the central (flight-critical) electronic subsystem preflight test requirements, this section deals with the peripheral hardware test (PHT) requirements.

The PHT study included programming an actual test for the laboratory application flight-control system, HSAS. This program was implemented using the WWCS equipment and became an annex to the WWCS BIT program to form a HSAS preflight-test test. The following is a summary of the PHT implementation and evaluation.

5.1 PERIPHERAL HARDWARE TEST CONSIDERATIONS

The WWCS preflight test laboratory configuration is shown in figure 5-1. The primary elements included in the PHT tests were the WWCS hardware, simulated pitch-rate gyros, a breadboard mechanization of the pitch-control column electrical-mechanical interface, and the electric command servo-hydromechanical subsystem. The preflight test laboratory study objective was to develop a PHT program using these elements to better understand system test and test executive requirements.

In general, the PHT's were developed following the same guidelines used in the development of the WWCS BIT (sec. 4.4.3.1). The considerations listed below, however, were added to those guidelines.

- End-to-end testing (sensor signal to actuator command/response) should be used wherever possible. It was assumed BIT validates A/D and D/A conversion accuracies; therefore, the PHT would be concerned primarily with signal-path interface continuity checking.

- Preflight-test test time (BIT and PHT) should be minimized.

- Preflight test software will be loaded identical in all three computer units. It is assumed that error-free software (i.e., a fully-validated resident program) does not fail; therefore, software checks are not to be a part of the preflight-test tests. (However, tests for memory failures are conducted in BIT.)

5.2 PHT ORGANIZATION

Preliminary timing estimates of a PHT based on the application model extrapolated to a three axis flight-critical system soon indicated that a full-system preflight test would

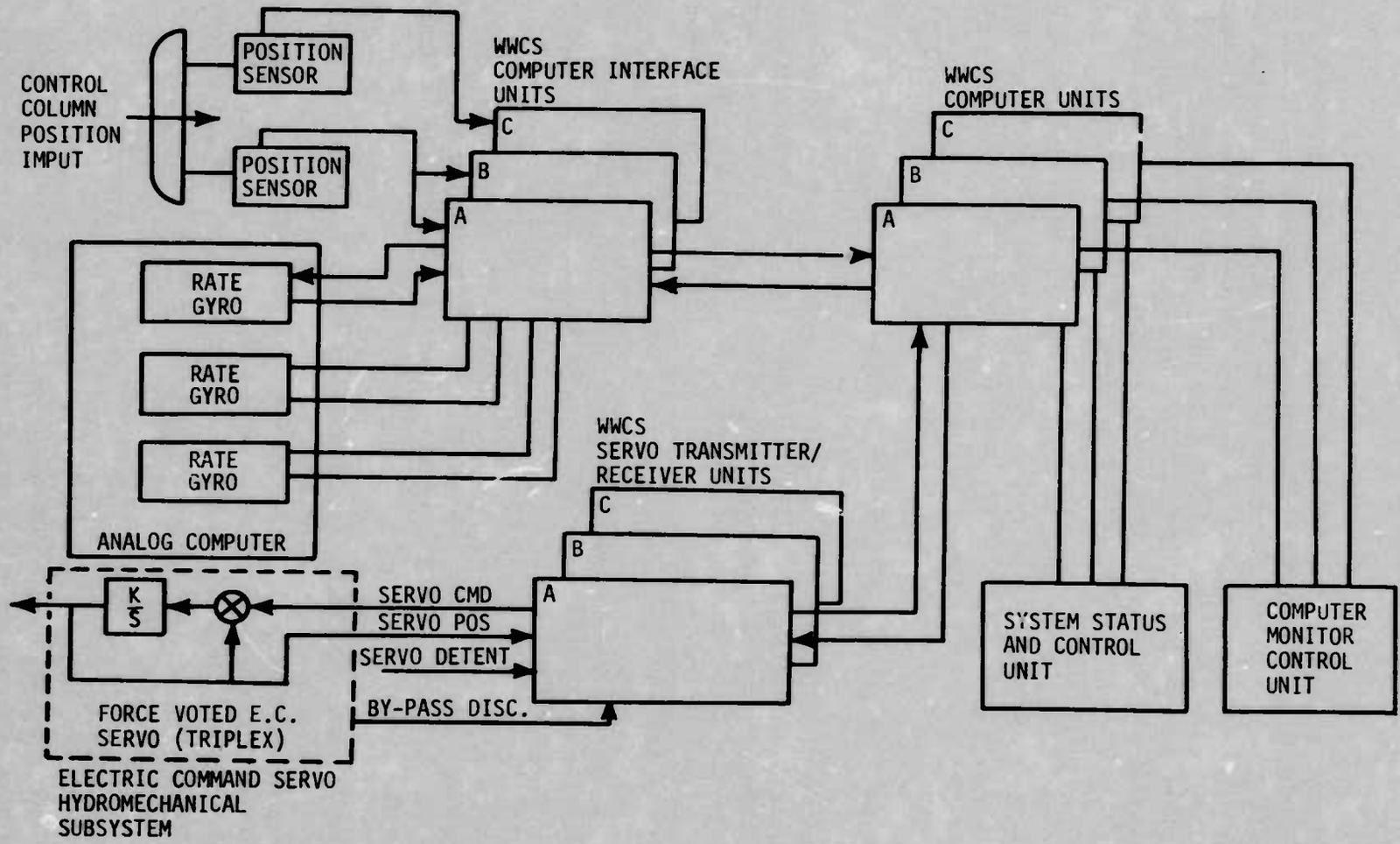


FIGURE 5-1.—PREFLIGHT TEST LABORATORY CONFIGURATION

become very time consuming if all tests are conducted in series. Therefore, it was concluded that an acceptable preflight test must involve executing a substantial number of tests in parallel. In order to better understand the series/parallel testing ramifications of an all-axis flight-control system, the system functions, shown in figure 5-2, were used as a reference in developing the organizational requirements of a PHT program. Both sequential (series) and parallel test executions are necessary. The rationale is as follows:

- Many functions (items of hardware) must be essentially simultaneously tested in order to minimize the test time.
- Some functions (items of hardware) must be tested in sequence because:
  - They must time-share flightcrew (operator) or equipment.
  - Their health must be validated before they are used to validate the integrity of other functions.
  - They must be sequenced to prevent the overloading of power supplies (e.g., hydraulic power, if all servos are simultaneously driven at their maximum rate).
- Some tests will have precondition requirements that must be satisfied before a test can begin. This is illustrated in figure 5-2 where manual (operator) action is specified; e.g., the servo testing must be completed before manual shutdown is exercised.
- Complete testing of some functions may require sequential execution of several different tests.

In order to organize a PHT program that would be adaptive to satisfy all of the above requirements and guidelines, the following definitions and program structuring was instituted.

Test Group

A test group is composed on one or more tests that must be executed sequentially. All groups could be executed simultaneously if the tests do not overload the computational capability of the CP. If a test within a group fails, it will not necessarily cause other tests within that group to fail.

Test

A test may be composed of one or more tasks that must be conducted sequentially. If any task within a test fails, the test is failed. All tests are independent from each other; i.e., no test failure will cause other tests to fail; however, the failure of a test may exclude the validity of other tests. If one test shows a display to be inoperative, all other tests requiring that display would be invalid.

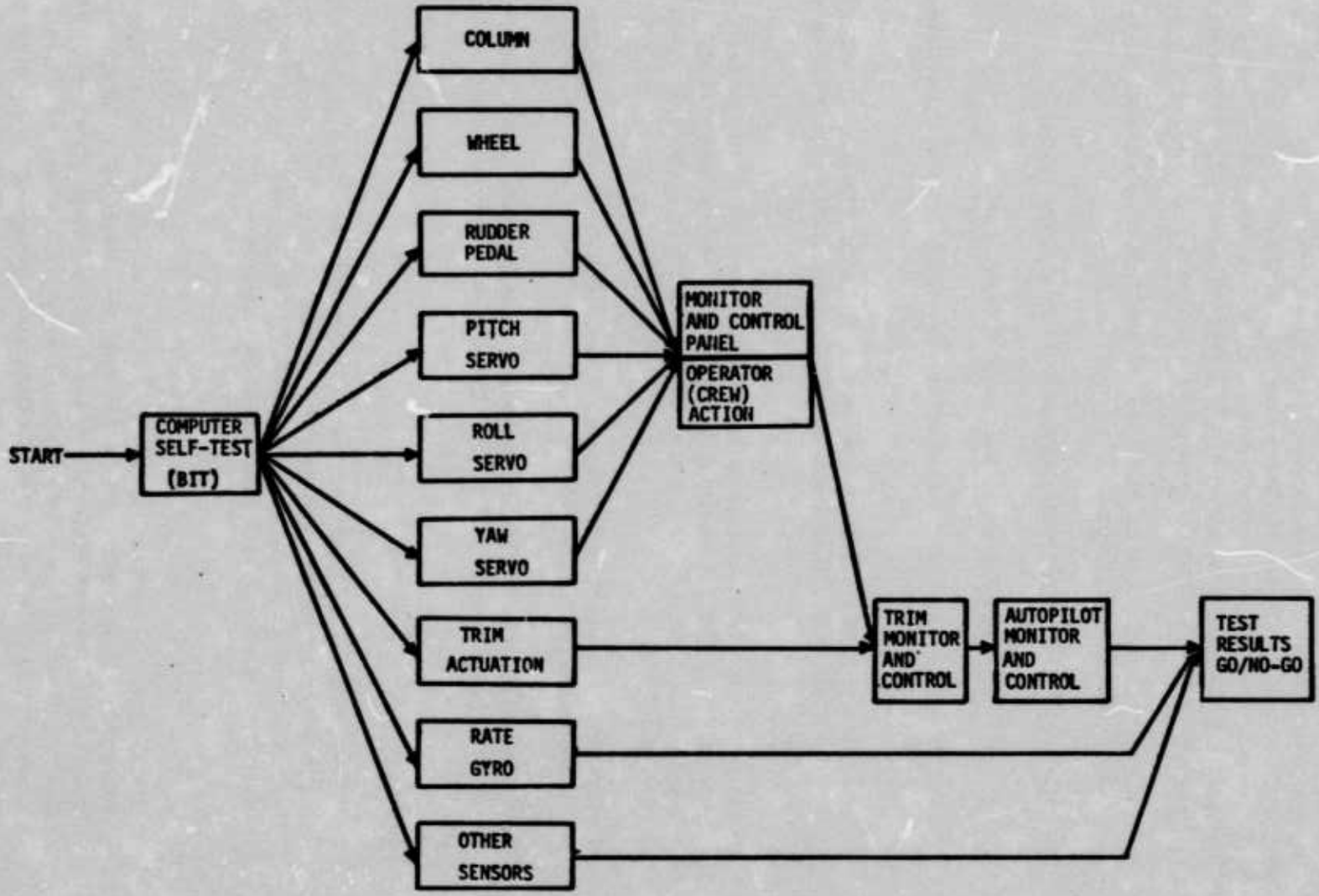


FIGURE 5-2—POSTULATED ALL AXIS FLIGHT-CRITICAL SYSTEM

Task

A task is the smallest item of work that can be programmed as a module within a test. All tasks, except for the initialization task, depend on the success of the preceding task before the task can itself be successful.

A PHT executive program was then developed to administrate the peripheral hardware test group, test, and task programs on a computational load priority basis.

5.3 PHT PROGRAM EXECUTION

Even though the preflight test program is not considered a real-time computation, reference to real time is required in order to validate the operation of specific system functions. This real-time measurement requirement was met by structuring the WWCS PHT executive to complete the processing of a test task for each test in work without permitting a task to be disrupted by the I/O timer interrupt signal. Since the I/O timer interrupt occurs at precise intervals (6.144 ms), it was used as the primary real-time reference within the PHT program.

The PHT executive was further structured to have a test-in-progress computational load management feature. This computational load management capability was developed in order to guarantee the consecutive execution of test tasks of tests within each test group, each I/O frame time. This feature was devised in order to permit a large number of test groups within the PHT without overloading the computational capability of the processor for any one I/O frame. The maximum task time for any test within a group was to be established and identified with that test; the PHT executive would control the number of tests executed in parallel, such that the sum of the maximum task times would not overrun the available I/O frame real time. Test task times were to be determined during the debugging of the individual tests. With this computational load management approach, the processor, in executing the PHT test groups, would be used to a reasonably efficient working capacity without requiring careful timing changes to be made every time a test group was added or deleted.

Preflight-test test failure for a flight dispatch-critical system could result in a simple no-go status annunciation. However, some degree of fault isolation is required before maintenance action can be taken. Therefore, the PHT program was designed to provide test failure identification along with the annunciation of a go/no-go status. Because different tests were to be processed in parallel, it was decided that a test failure should not stop the testing of functions unrelated to the failed test. Thus, it was concluded that the PHT executive would store test failure identification information for recall at the conclusion of all tests. Provisions were also included in the PHT executive to permit rerunning tests identified as failed without completely recycling through all other tests in the PHT program.

The PHT structure of the application model, HSAS, test problem involved working all of the requirements discussed above. Figure 5-3 shows a test sequence time diagram for the three test groups that made up the laboratory problem. The computer subsystem built-in-test, WWCS BIT, is executed first, followed by the processing of the PHT tests.

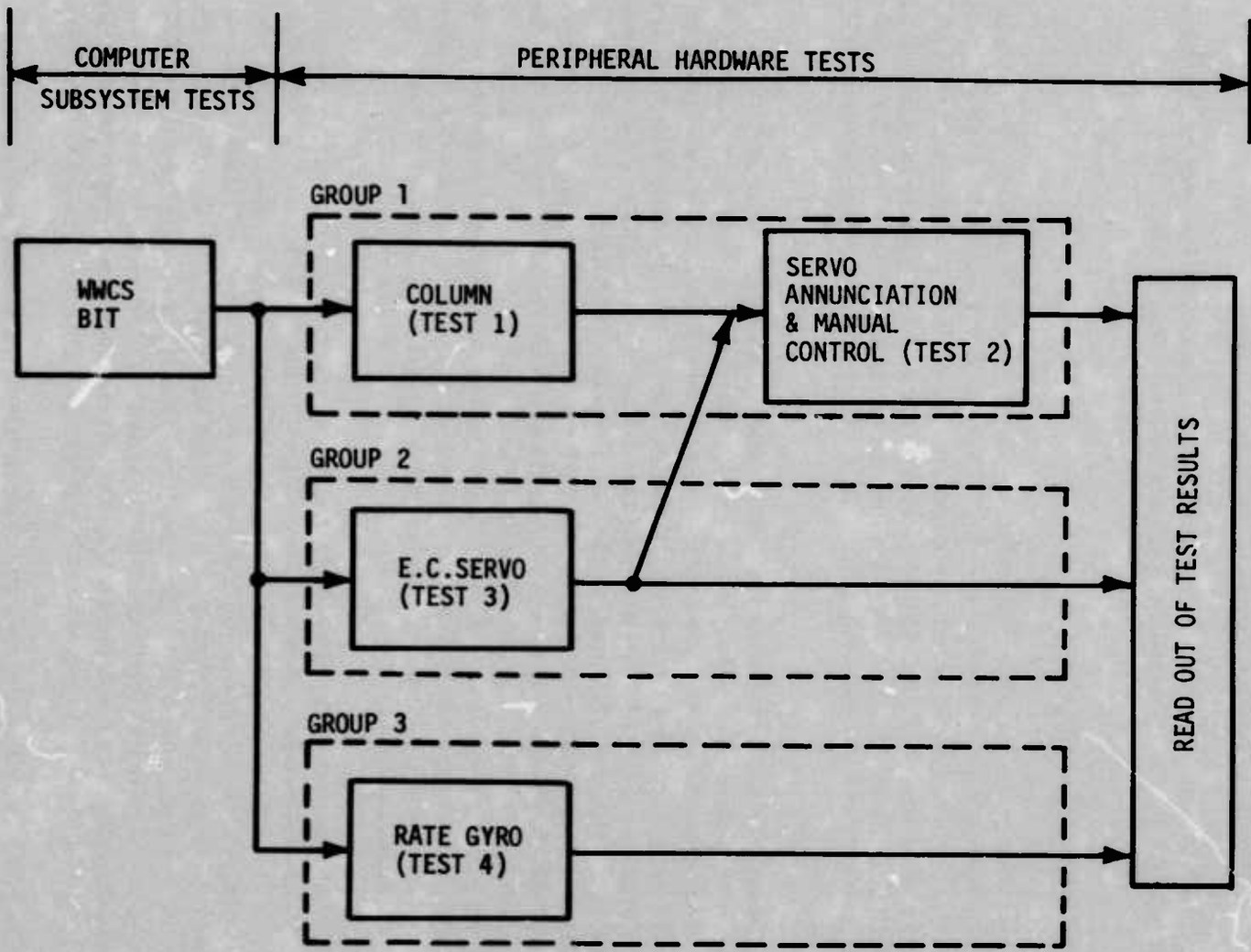


FIGURE 5-3.–PREFLIGHT TEST LABORATORY TEST SEQUENCE

Figure 5-4 is a block diagram that illustrates how the PHT executive, test groups, tests, and tasks are interrelated. The PHT executive handles initialization, timing, test group selection, computation load management, and test result readout. Each test group and each test have a small executive to direct the processing of their respective tests and test tasks. The PHT program processes through one task in each test-group-in-work during each I/O frame time. After the completion of the last test task, a display indicates the go/no-go status of the PHT tests. The operator then presses the continue test button, (fig. 4-22) and the program returns to the beginning of BIT (no failures) or the first failed test.

5.4 PHT LABORATORY EVALUATION

The peripheral hardware test covering the application laboratory HSAS model involved three test groups. These test groups, whose time relationships are depicted in figure 5-3, are:

Group 1—Pitch control column and servoannunciation/manual control tests

Group 2—Electric command servo-hydromechanical subsystem tests

Group 3—Pitch-rate gyro tests

The design procedure used in developing the PHT program for these test groups was as follows:

- Design the individual hardware functional tests
- Estimate the time required for each test and fit the tests into a sequence time diagram which complies with all precondition requirements
- Prepare software program code to implement the tests

The individual test group tests, time estimates, and program memory requirements are discussed in the following subsections.

5.4.1 Test Group Descriptions

5.4.1.1 Pitch Control Column Tests

The control-column interface test was designed to check the control-column redundant transducers travel limits, tracking accuracy, and dead zone (hysteresis) levels. The flightcrew is required to participate in this test group. At the start of the column test, the test program is initialized, and the crew is cued to exercise the control column. The column must be moved to the extremes of its travel and be given some short quick oscillatory motions about the neutral position. A time history of the column transducer test activity is shown in figure 5-5.

After the crew exercises the column, the continue test button is pressed. The computer checks that the column was adequately moved to its travel limits. If not, the test will be

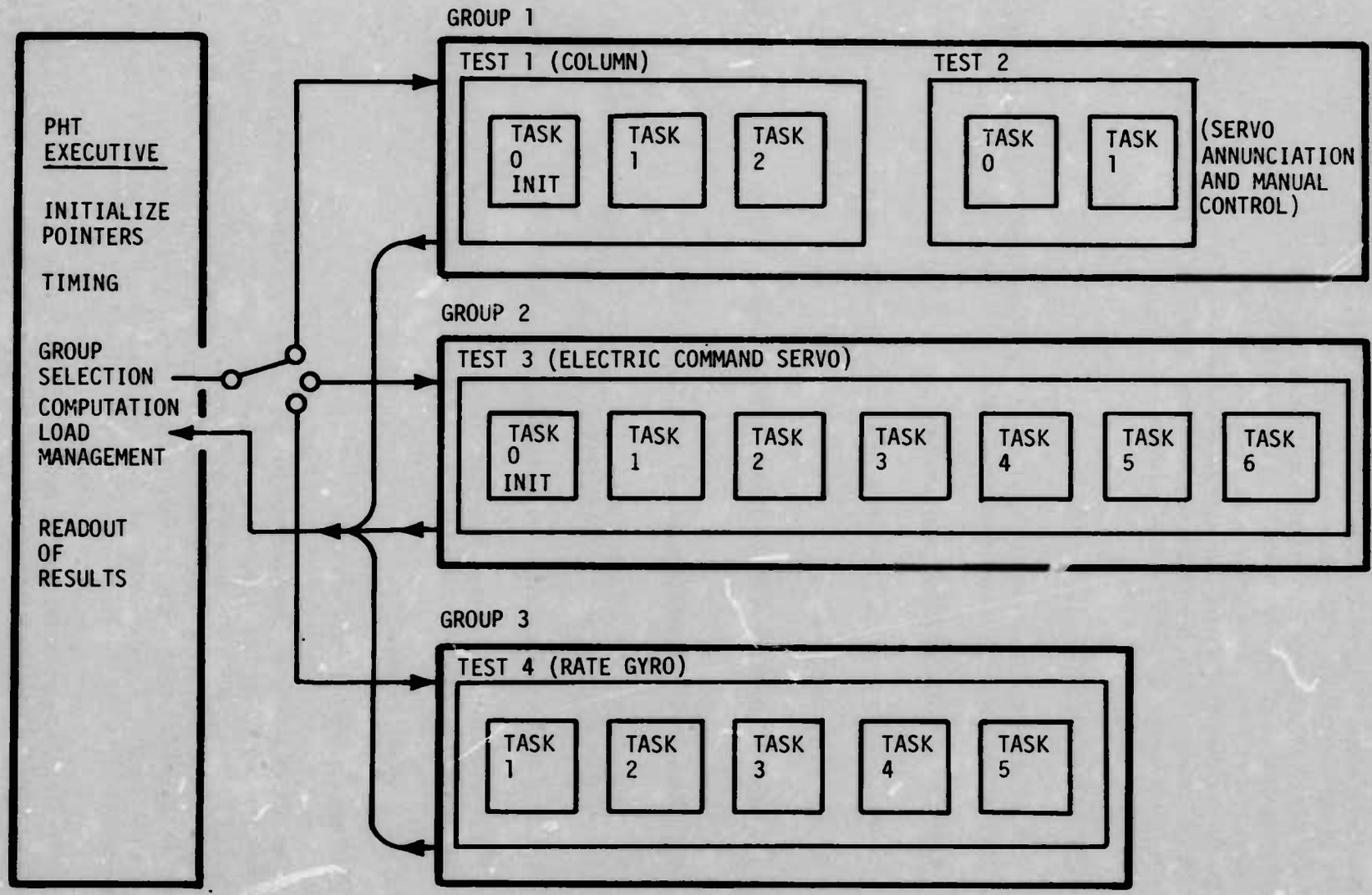


FIGURE 5-4.–PHT PROGRAM SEQUENCE

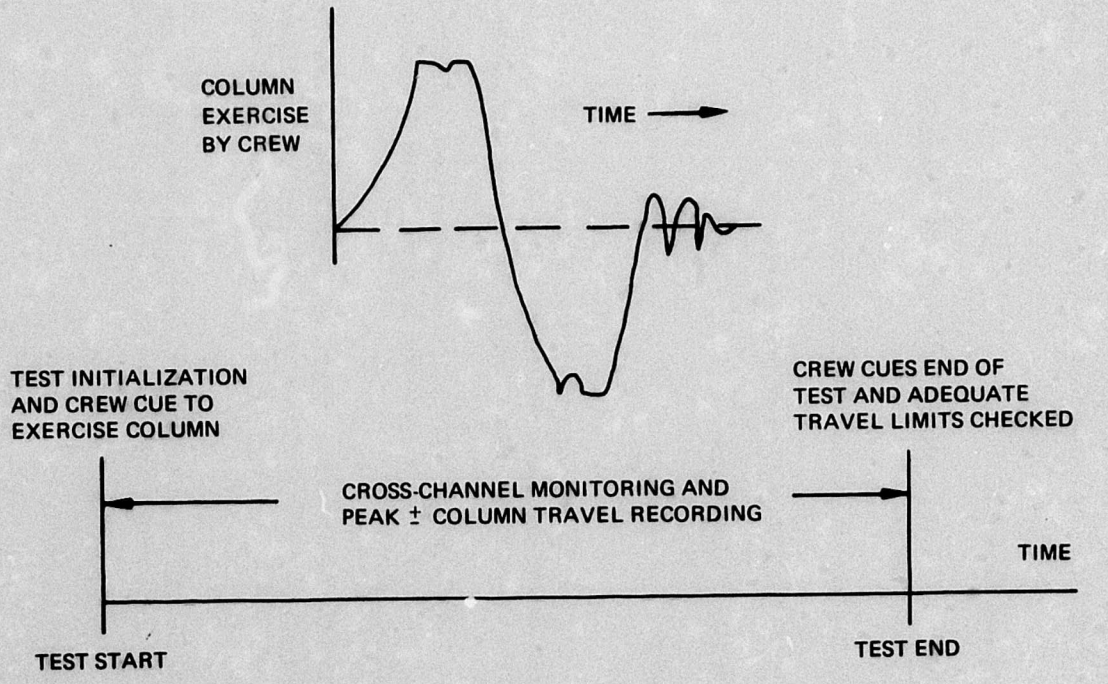


FIGURE 5-5.–PITCH CONTROL COLUMN TESTS

failed. Throughout the test, the computer program performs two types of cross-channel signal comparisons each I/O frame time. The first comparison checks that the transducer input to each channel, compared across-channels, remains within 8% of the transducers full-scale travel. The second comparison checks for signal nonlinearities by comparing the transducer signal difference, across-channels, and between successive I/O frames. If the difference exceeds 0.5% of full scale, the test would fail indicating an undesirable level of dead zone, a condition that could arise from a sloppy mechanical installation of the transducer.

5.4.1.2 Electric Command Servotests

Servo-subsystem tests were based on checking the following hydromechanical characteristics of the servopackage.

- Operation of the differential pressure (ΔP) detent mechanism
- Servocentering following channel depressurization
- Force authority of each channel

The time sequenced test tasks for the EC servotests, as illustrated in figure 5-6, are described below.

Tasks 0 and 1 initialize the servo-subsystem tests. Task 2 checks that the ΔP detent in channel 1 will bypass in the positive direction for a hardover command, and that the hardover does not disturb the redundant servooutput position beyond acceptable limits.

Task 3 checks that the servo-subsystem output tracks the two operative channels when one channel is hardover. Task 4 is the same test for one channel depressurized and the servocentering detent engaged.

Task 5 checks that the ΔP detent in channel 2 will bypass in the negative direction, and that this hardover does not drive the subsystem output. Channel 2 is then depressurized and task 6 checks that channel 3 cannot overcome the two centering detents and drive the subsystem output.

These test tasks are executed three times in succession, where the system redundant channels, A, B, and C, are permuted through the test series shown in figure 5-7 for channels 1, 2, and 3.

This series of tests verify that:

- Each ΔP detent will bypass in both the negative and positive direction
- Each combination of two servos has a load driving capability to drive the subsystem output with the other channel bypassed or depressurized

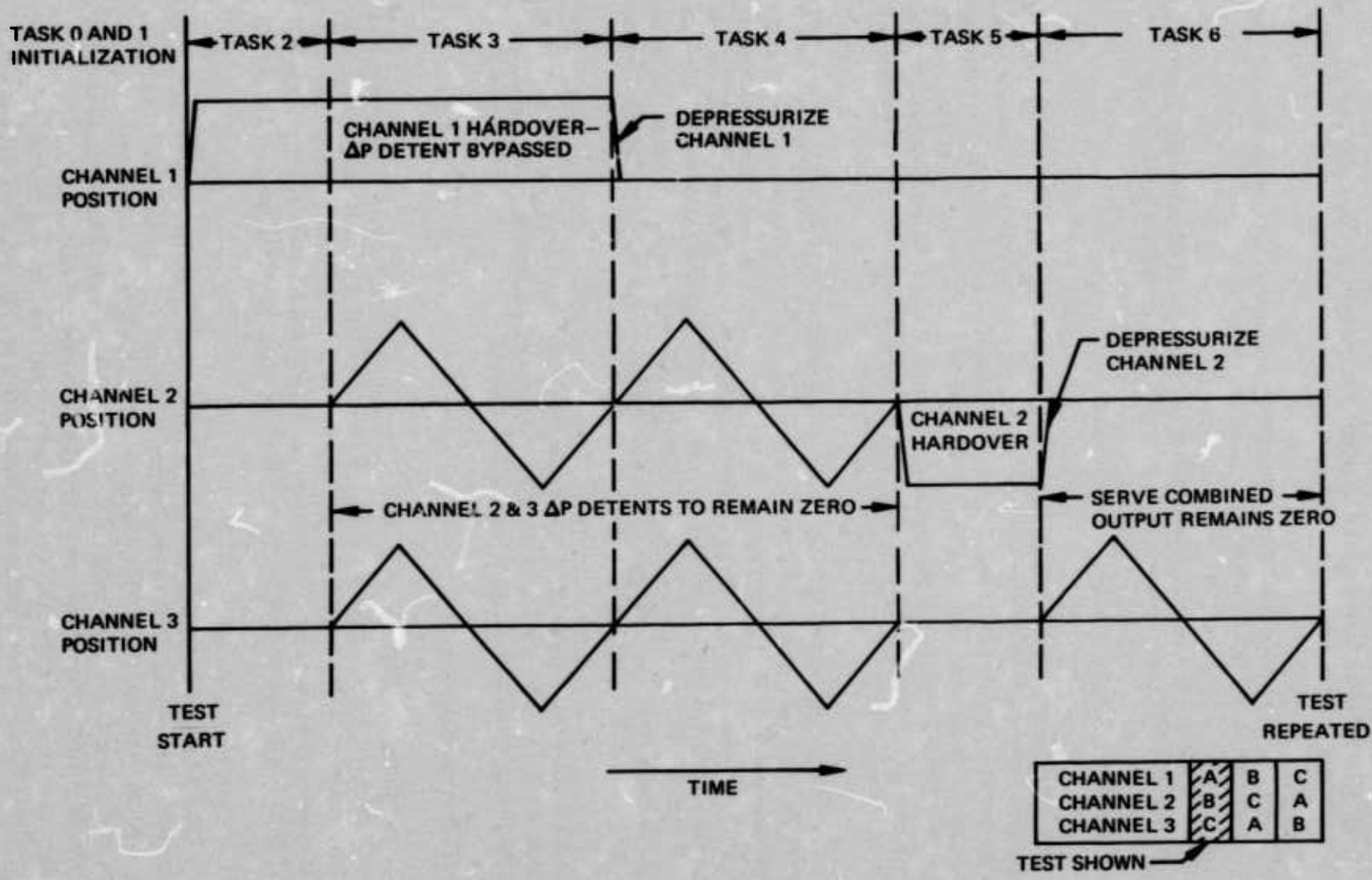


FIGURE 5-6.–ELECTRIC COMMAND SERVO TESTS

- Each centering detent has an adequate force capability to assure that a second hardover will not drive the subsystem output

- Each combination of two centering detents prevent the other channel from driving the subsystem output

It was assumed that servoload equalization (if required) and subsystem monitoring would be handled by software; therefore, these functions were not specifically tested.

5.4.1.3 Rate Gyro Tests

Rate gyro tests were developed based on the assumptions that the gyros could be automatically torqued, and that they would respond to a step disturbance following their specified dynamic response characteristics. The gyros were simulated on an analog computer as three second-order systems having a natural frequency of 18 Hz and a damping ratio of between 0.5 and 0.8. While such dynamic characteristics may not be truly representative of the response of a self-torquing gyro, the tests developed using the above assumptions permitted the evaluation of a preflight test program structure.

The gyro-test time history and associated test tasks are illustrated in figure 5-7. Task 0 initializes the gyro test. Task 1 activates the torquing coil to command a 1°/sec positive gyro output, and a cross-channel comparison is made to see that the three gyros remain within their specified step response boundaries over 16 I/O frame times. During the last frame time of task 1, the gyro output value is recorded and used as a steady-state reference value for task 2. This value must be within ±10% of the expected 1°/sec command value. Task 2 monitors that the steady-state output remains within ±2% of the steady-state reference for a period of 50 I/O frame times.

Tasks 3 and 4 repeat tasks 1 and 2 by driving the gyro output in the negative direction. Task 5 essentially repeats task 1 with the gyro torquing coil stimulus removed and the gyro output returning to null within acceptable null offset limits.

5.4.2 PHT Program Structure

The peripheral hardware test executive was set up to execute all PHT test groups and read out results. Coding was included to reexecute a failed test group without rerunning the entire PHT program. This procedure was selected so that improperly set preconditions, such as mode switch setting or improper flightcrew interaction, could be corrected, and the test reexecuted with a minimum loss of time. A flow chart of the PHT executive is shown in figure 5-8.

The PHT program design required that the program executive be executing when the I/O timer interrupt occurred. In the event the timer interrupt occurred during the execution of a test group program, the PHT program would stop and await crew action. Such a situation could arise if the PHT program tried to execute too many test group test tasks within one I/O frame, or excessive time was used in recording test failure states. Should this occur, the crew could depress the continue test button, and the interrupted test group would be rerun. If a timer interrupt does not occur before the last scheduled test group task

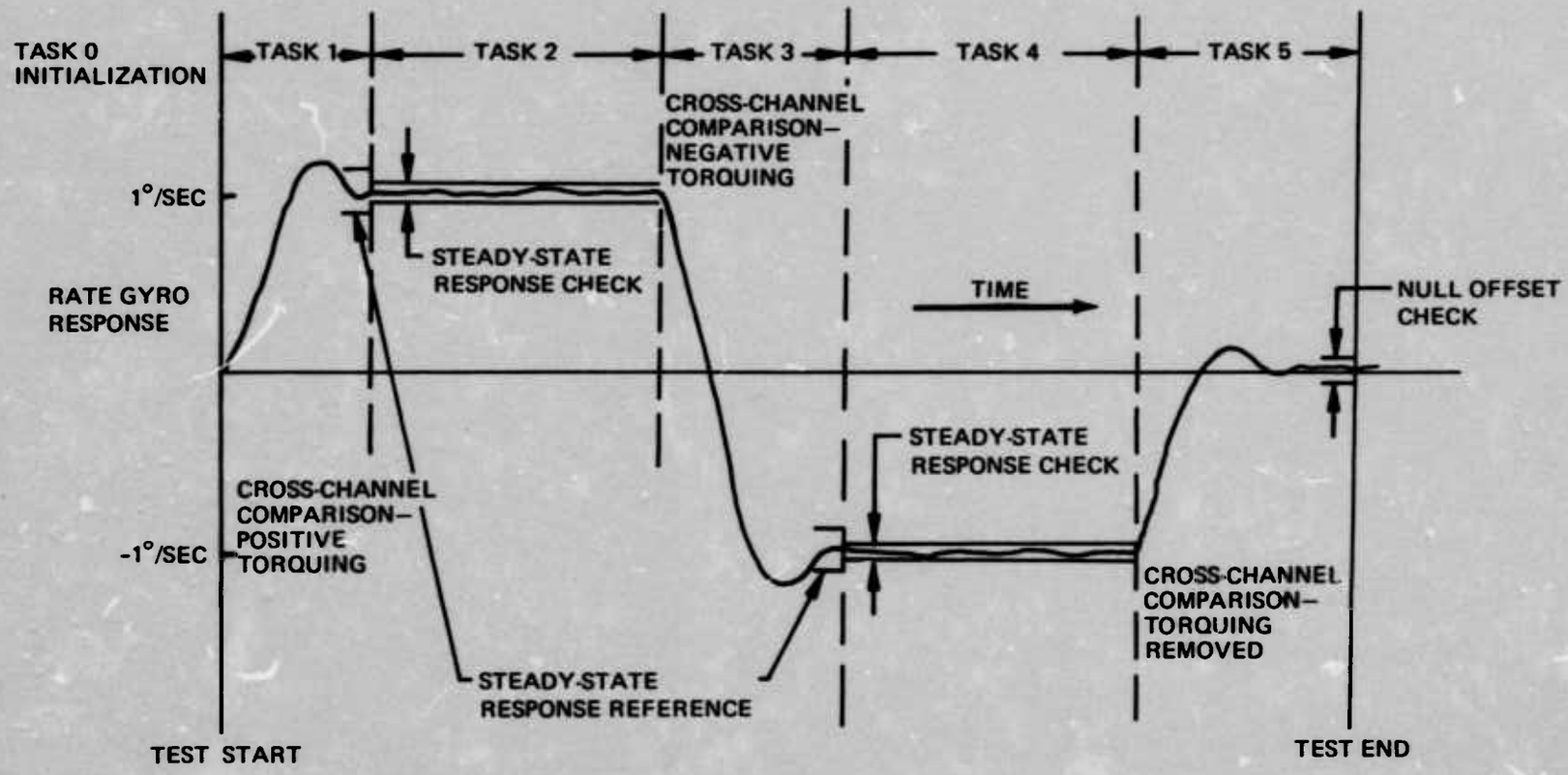
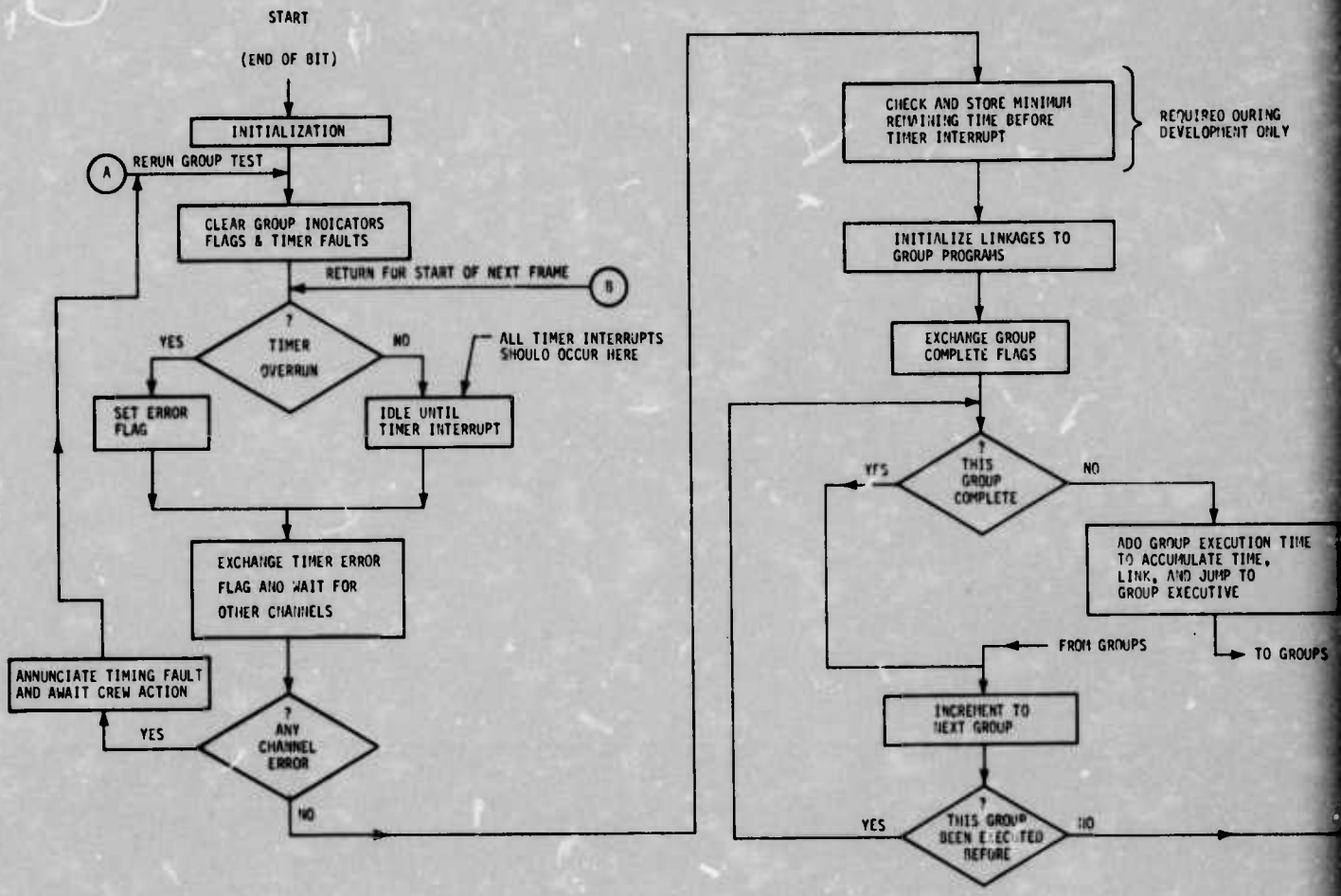


FIGURE 5-7.–RATE GYRO TESTS


START
(END OF BIT)
INITIALIZATION
A
RERUN GROUP TEST
CLEAR GROUP INOICATORS FLAGS & TIMER FAULTS
RETURN FOR START OF NEXT FRAME
B
? TIMER OVERRUN
YES
NO
ALL TIMER INTERRUPTS SHOULO OCCUR HERE
SET ERROR FLAG
IDLE UNTIL TIMER INTERRUPT
EXCHANGE TIMER ERROR FLAG AND WAIT FOR OTHER CHANNELS
ANNUNCIATE TIMING FAULT AND AWAIT CREW ACTION
? ANY CHANNEL ERROR
YES
NO
CHECK AND STORE MINIMUM REMAINING TIME BEFORE TIMER INTERRUPT
REQUIRED OURING DEVELOPMENT ONLY
INITIALIZE LINKAGES TO GROUP PROGRAMS
EXCHANGE GROUP COMPLETE FLAGS
? THIS GROUP COMPLETE
YES
NO
ADD GROUP EXECUTION TIME TO ACCUMULATE TIME, LINK, AND JUMP TO GROUP EXECUTIVE
FROM GROUPS
TO GROUPS
INCREMENT TO NEXT GROUP
? THIS GROUP BEEN EXECUTED BEFORE
YES
NO

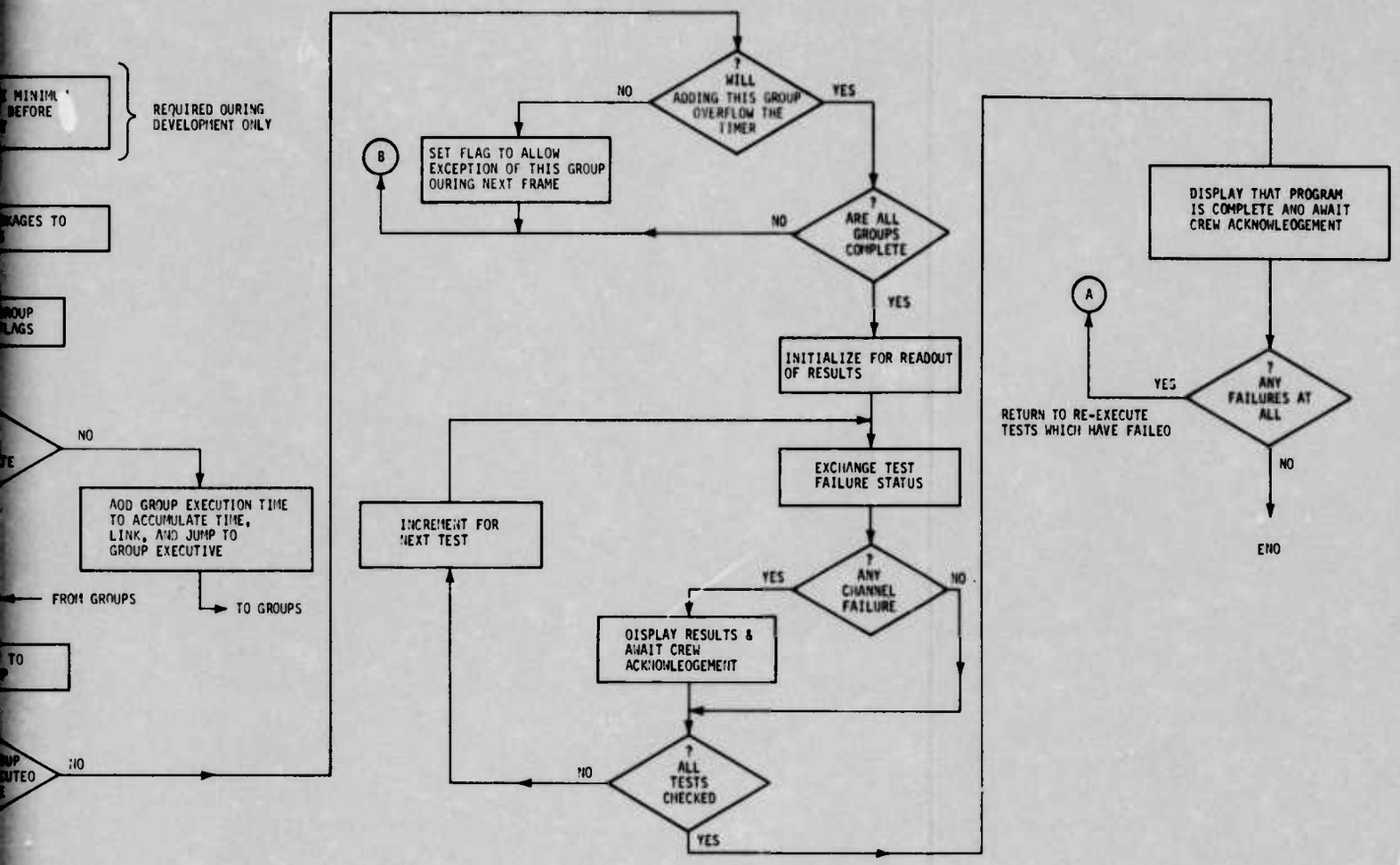


FIGURE 5-8.—PERIPHERAL HARDWARE TEST EXECUTIVE

is completed (normal case), the PHT executive idles in a counting routine until the interrupt does occur. This counting routing forms the base for establishing how much available time remains during an I/O frame times worth of PHT computations.

Following a timer interrupt, the PHT executive sequences back up to the test group subroutines. Since it was assumed that the first test task in any test group would initialize the test group tests, only one new test group was permitted to be added to the test group series during any one I/O frame. Before each new test group is added, its maximum use of the available test time is added to the available test time used by the test groups in process, and the test group would be skipped if its time with the others would overload the computational load during an I/O frame. When a sufficient time margin exists because other test groups have been completed, the new test group begins.

When all test groups have been completed, the test results are readout. If a test is identified as failed, the program stops and awaits crew action to acknowledge the failure information. After all failed tests have been recorded, the program will initiate, with an activation of the continue test button, a rerun of the failed tests.

The preflight test (BIT and PHT) can be terminated at any time by deactivating the BIT initiate switch on the system status and control panel.

The test group test and task program organization is shown in figure 5-9. The PHT executive passes program control to the test group program using a jump to subroutine operation. The test group program has a utility executive routine, which sequences the series of tests and test tasks within that test group. The general structure of a test series of tasks is as follows. A task is entered and a test is made to determine if the task execution will be the final pass. The task may be repeated if a preset number of passes is required, or if the program is awaiting a cue from the flightcrew. A task operation will usually be to control (initiate) test stimuli or to record a response and check it against expected response data. If the task detects a default, (a test has failed), the program jumps to a subroutine that will record the failure state information. The program will then return to the PHT executive and continue into the next test group test task. If no default is detected, the test is completed, and the program returns to the PHT executive to continue. The last test task includes the test-finish-work of resetting all task initial states.

5.4.3 Computational Load Estimation

Test time requirements are an important factor in planning a preflight test program for a peripheral hardware test having many test groups. The total test time depends on the computational load capability of the computer; i.e., the number of test groups the computer can handle during a given frame time. Part of the PHT development effort using the WWCS was to determine how the application model PHT burdened the WWCS I/O frame time. The following laboratory tests were conducted to determine WWCS PHT computational load.

The time available within the WWCS I/O frame for processing PHT test groups is that remaining following the execution of the primary executive that handles the I/O timer interrupt, and the execution of the PHT executive that initiates the first PHT test task and closes out the last test task before the next I/O interrupt occurs. This available test time was

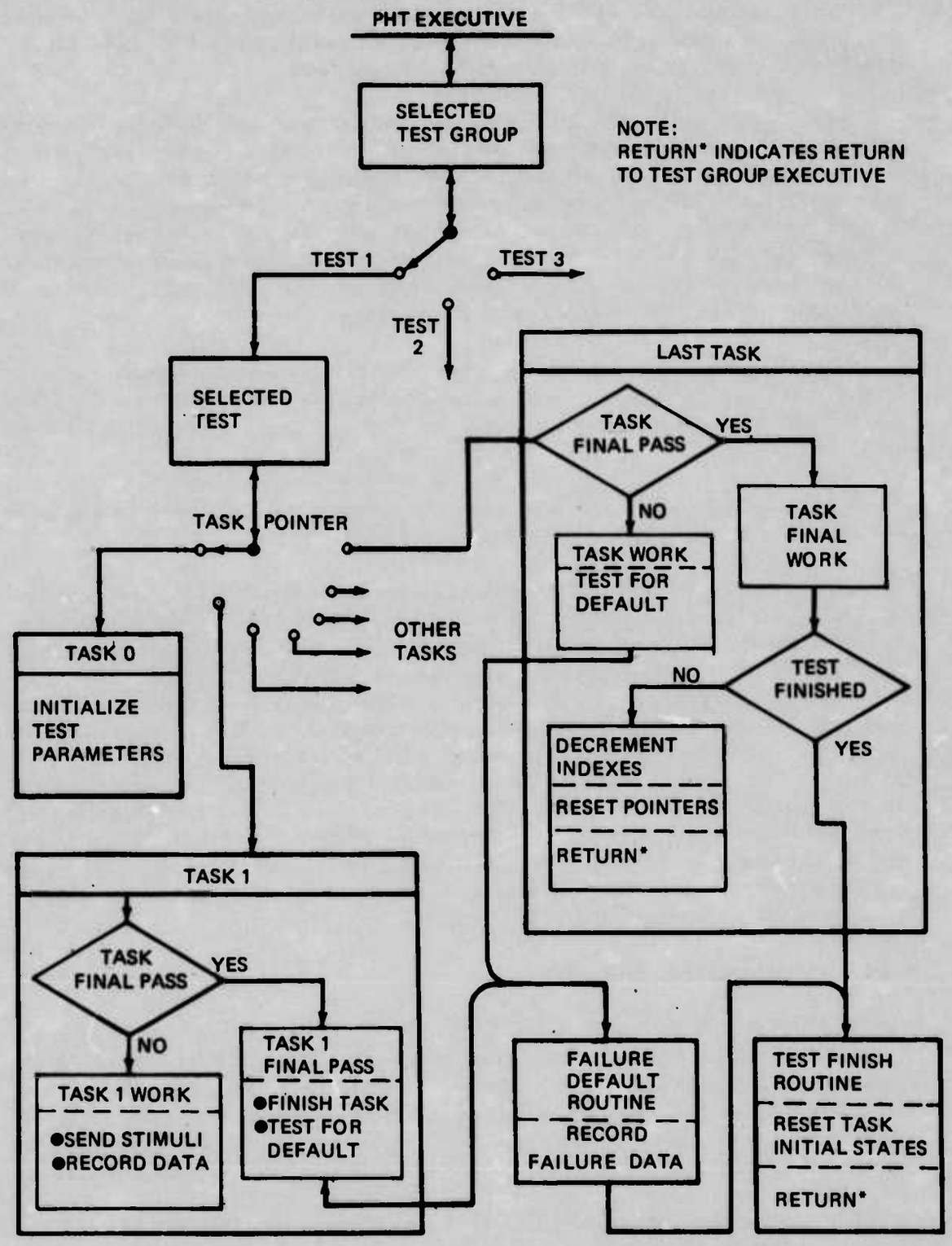


FIGURE 5-9.–PHT TEST GROUP TEST/TASK ORGANIZATION

determined in the laboratory by processing a dummy test group, a simple entry and exit, and recording a time remaining count between the dummy test group execution and the subsequent I/O timer interrupt. Each test group of the laboratory application model was then executed and the time remaining count recorded. This was followed by adding one test group at a time and by recording the time remaining count until all three groups were being executed simultaneously. The percent of available PHT test time used during the above tests are listed in table 5-1.

The data in table 5-1 show that 67% of the test time available (for the worst case I/O frame burden during the total test time) was used when all test groups were being executed together. A summation of the individual test group worst-case test times, however, adds up to 102% of the test time available. An explanation for this apparent anomaly is illustrated by figure 5-10, where the I/O frame worst-case computational load is shown to be a function of how the test group test tasks mesh together each I/O frame, over the total PHT test period.

In conducting the above tests, two test groups were not permitted to be initiated during the same I/O frame. This was the only special constraint imposed for combining the test groups, since some initialization tasks were prejudged to be time consuming operations.

From this laboratory study, it was concluded that if a PHT program was to be executed in the minimum real time, it would be necessary to fit the test group programs together by test task times rather than by total test group or test times. However, care would have to be taken to preserve test task continuity for real-time tests and to mesh test tasks such that the time available within one I/O frame time, WWCS, would not be overrun before the final task in that frame was completed. This conclusion diminishes the usefulness of the test-in-progress computational load management feature of the PHT executive once the laboratory development work reaches the final test sequencing time-optimization phase. The feature was, however, a useful tool during the initial laboratory work of combining the various test groups of the PHT program.

5.4.4 PHT Memory Utilization

A breakdown of the WWCS memory used for the PHT program is presented in table 5-2. The basic PHT executive required 353 words of code with an additional 78 words required to handle the test group test and task executive items. This gave an overall executive overhead of 431 words.

The memory required for each test is categorized in table 5-2 as to whether it is unique test code, subroutines, or data blocks. The right hand information column of table 5-2 contains an estimate of the additional words of memory required to add other PHT tests similar to those implemented in the laboratory. For instance, the column tests require 124 words of memory. To add an additional pilot controller transducer interface test, the column test code could be made into a subroutine for approximately 20 words of code; a new data block added 40 words. The new test, therefore, would only require about 60 words of code as compared to the initial test requiring 124 words. A similar approach can be taken for additional EC servotests resulting in approximately 82 words of added code for one additional EC servotest, and about 52 words of code per test for each EC servotest thereafter.

TABLE 5-1.–AVAILABLE PHT TEST TIME USED (I/O FRAME WORST CASE)

Test group	Test time used		
	Group executed alone (%)	Additional time as groups are added (%)	Test period duration (sec)
Control column	58	58	30 [1]
Electric command servo	20	2	30
Pitch rate gyro	24	7	1
	102	67	
	total	actual	

[1] Includes crew interaction time

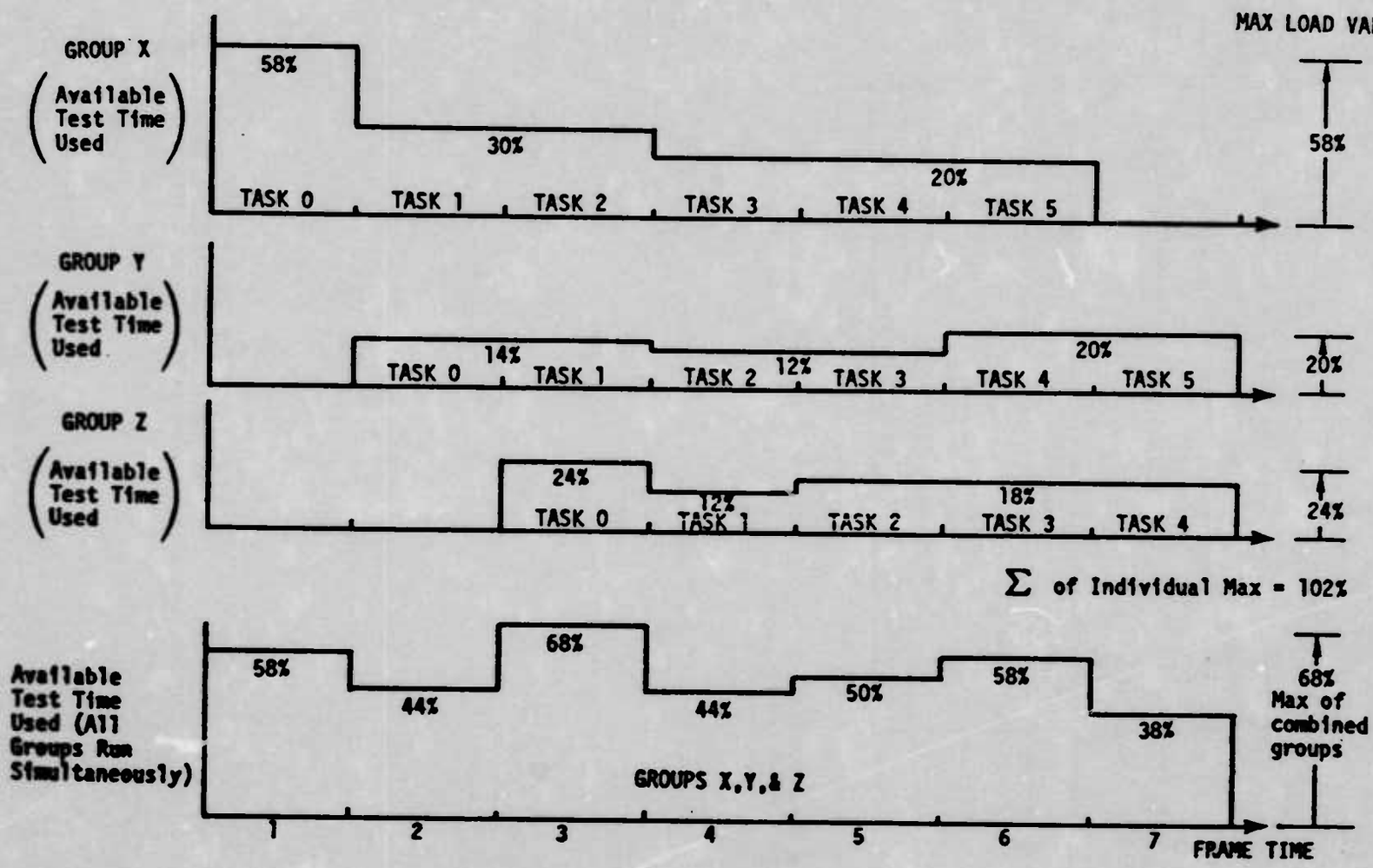


FIGURE 5-10.–PHT COMPUTATIONAL LOAD, MESHING ILLUSTRATION

TABLE 5-2.–PHT MEMORY REQUIREMENTS

Test	Words				
	Test code	Subroutines	Data blocks	Total	Additional like test
Control column	84	0	40	124	20 + 40
Servo annunciation/ manual control	29	0	0	29	——
Electric command servo	149	149	52	350	30 + 52
Rate gyro	90	99	26	215	30 ⚠
	Executive overhead				
	PHT executive			353	
	Test group executive (6/GRP)			18	
	Test task executive (15/test)			60	
	Summation of program indirects and constants			90	
	Total			1239	

⚠ Assumes identical gyros tested sequentially with shared data blocks

5.4.5 PHT Laboratory Test Conclusions

The laboratory evaluation of a preflight test (WWCS BIT and PHT tests) for the HSAS application model showed that the approach taken can result in a workable test sequence accomplished in a reasonable time period. Coding optimization for the BIT or PHT programs was not undertaken. The objective of the laboratory test was to investigate preflight test schemes and, since the approach taken proved satisfactory without cycling through code and timing optimization, no programming refinement was attempted. A strip chart recording of the preflight test PHT tests is shown in figure 5-11.

Per the SST preflight test design requirements, the application model preflight test resulted in a go/no-go indication. Maintenance diagnostic requirements were not investigated in the laboratory beyond those used in the PHT executive to support laboratory checkout of the PHT program. These fault identification/test-rerun features, however, could be easily adapted to maintenance testing to identify a faulty line replaceable unit.

The testing concept investigated was based upon an operational triplex system. Test requirements to determine the operational integrity of two out of three functional success paths of a system were not undertaken and are viewed as being an easy task for the PHT program and a difficult (involved) task for the WWCS BIT program. Future redundant system preflight test development efforts should include formulating a system test capable of determining the system's operational state even though a single failure exists.

The laboratory preflight test study demonstrated the versatility of a whole-word system's capability for conducting an automated system test. This capability, along with the recognition that the WWCS performance falls within that required to implement most (if not all) control-law/redundancy-management functions of a flight-critical system, points to the WWCS concept as being the most powerful subsystem electronics for mechanizing a redundant flight-control system. This position is based on the use of a processor having a processing speed comparable to the MCP-701 (80% or better).

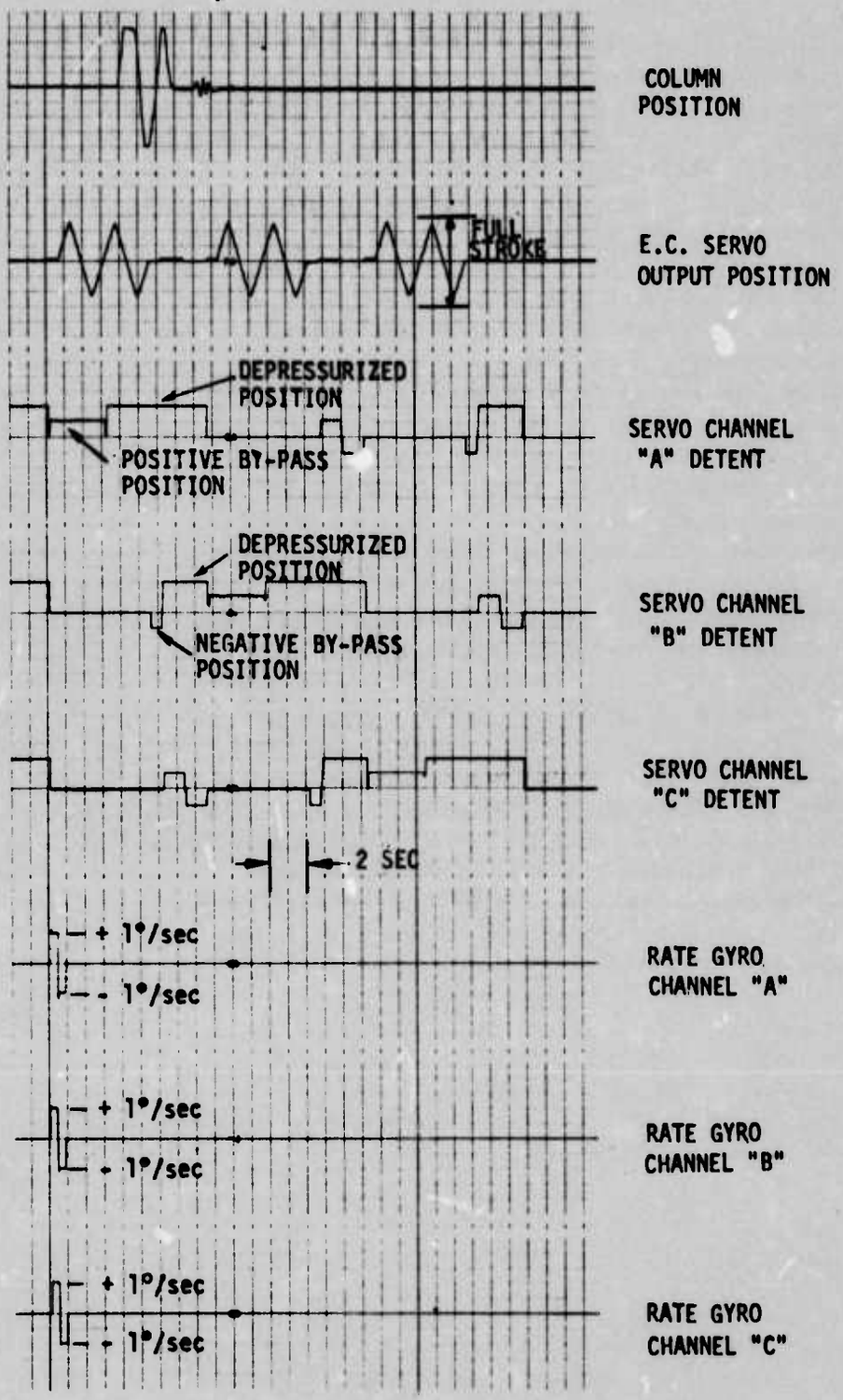


FIGURE 5-11.–PHT STRIP CHART RECORDING OF PERIPHERAL HARDWARE TESTS

APPENDIX A

ICPS FMECA

Contained within this appendix are the results of the various study areas associated with failure modes analysis for the ICPS. By partitioning the ICPS into major functional blocks as discussed in section 4.3.2, six study areas were identified. Four of these (redundant clock, majority logic voter, system status logic, and sensor selection and failure detection) fell within the first level of concern since they are involved in cross-channel tie points. The two remaining areas (input/output circuitry and timing, and the computer) are second level areas since they do not directly interface with other channels.

In presenting the failure modes data for each of these six sections, a brief description of that section's operation will be given prior to the failure modes study. These sections are discussed in a descending priority relative to the system's sensitivity to their failure modes and effects. In all cases, studies were only done as a paper analysis; i.e., no hardware was actually failed (destructive testing) to verify study results.

A.I REDUNDANT CLOCK

Throughout the mechanization of the ICPS, all functions and hardware are triplicated with the exception of the basic oscillator scheme. Even though the oscillator exists in each CIU (to permit interchannel exchangeability), the harness wiring does not utilize the channel C oscillator in developing the redundant clock scheme. The clock scheme is an active standby mechanization utilizing only two oscillators to develop the fail-operative clocking structure. This type of redundancy structure is always suspect relative to having a single failure mode, which may prevent the switchover to the standby elements when the active elements have failed. The viability of this design, as to its design integrity and latent free potential, lies strictly within the realization of the design intent. The active standby concept by itself does not imply that a single thread failure element exists. This can only be ascertained by careful study of the actual circuit design.

Figure A-I illustrates the operation of the clock system of the ICP-723. This circuit is repeated in each channel so that the computing subsystem is not dependent upon a single clock circuit. Within each channel, both the designated primary and secondary (as determined by harness wiring) are checked by an in-line monitor. The outputs from the monitors go to failure status logic for annunciation. In addition, the output of the primary oscillator monitor is used to initiate a command to switch to the secondary oscillator. To prevent single channel (common mode) failures in the monitor and switch-over circuits, the actual switching must be a majority of two out of three commands to switch. This is accomplished with the use of a majority logic voter (MLV) discussed in section A.2.

This switching operation, illustrated within figure A-I, is implemented through solid-state logic devices. In order to maintain functional independence for this clock system, the basic oscillator signals are utilized to operate the sequential logic elements within the command circuits. To ensure that sufficient signal is available to operate these circuits, the

primary oscillator is passed through a long delay function (equivalent to three clocking cycles). This allows the monitor time to achieve a switchover in a transient-free manner before the primary oscillator signal is lost to the switch-over logic.

The key to the design integrity centers about the ability of the monitor to detect failures. Thus, the failure modes study starts by determining if there are any postulated failure states that the monitor could not detect. If there are none, the clock logic circuits may not need to be analyzed.

A.1.1 Clock FMECA Approach

The design of the primary (secondary) clock monitors shown in figure A-1 verifies that the input waveform has a negative transition every 500 ns. This task is accomplished by toggling a flip-flop on each negative transition of the waveform, where the output of the flip-flop, ANDed with its negated value and delayed by 500 ns, represents a signal suitable to drive a failure latch. If transitions do not occur at the 500 ns interval (within a tolerance 50 ns), this ANDed signal makes a state change.

The FMECA study starts by first determining what waveforms can satisfy the operation of the monitor. By drawing the time history waveshapes through the monitor circuit, one can easily ascertain that there are several frequencies that meet the monitor design requirements. These frequencies are nominally 2MHz, 6MHz, 10MHz, 14MHz, 18MHz, or any other frequency higher than 20MHz. Therefore, any failure in the oscillator countdown and waveshaping circuit that would result in any of these signals will go undetected.

A second condition that needs to be checked is one in which the waveshaping circuits fail to a mode where it is still providing clocking transitions at the proper intervals but where the signal waveform will not be satisfactory for driving the square clock signal generator (fig. A-1). An examination of this circuit shows that if the signal into it fails to a mode where it is zero 250 ns before each negative (clocking) transition, it will produce a constant output (i.e., go dead). Furthermore, if such a signal fails to this mode without breaking the normal 500 ns sequence of clocking transitions, then the monitor will not detect the shift in waveshape. Thus, the monitor cannot request a switch to the secondary clock driver, and it will be a single point failure to all channels of the system. In summary, there are two failure modes that the waveshaping circuits must be checked for; transitions to other specified frequencies and a waveshape whose negative transition is always proceded 250 ns by a zero value.

A.1.2 Clock Countdown and Waveshaping FMECA

Figure A-2 illustrates the clock countdown and waveshaping circuitry. Since it contains memory elements (flip-flops), failure analyses using transition matrices were utilized. To keep the state transition diagram reasonably simple, it was necessary to narrow the circuit to be analyzed by breaking it at intermediate points where interfaces are simple and easy to understand. An example of such a process is given in the following paragraph.

By viewing figure A-2, it can be seen that flip-flop 1 (F/F 1) simply converts the 16-MHz oscillator square wave into an 8-MHz square wave. Therefore, it need not be included in the state transition analysis. (It is either working or it isn't working.) The remaining F/F's in the circuit are being clocked by the same 8-MHz square wave, except for the output stage, F/F 5. Since F/F's 2, 3, and 4 are being driven by a negated value of the clock, F/F 5 essentially is made to lag the clocking transitions of the other F/F's. (Clocking occurs on the negative transition of the clock signal.) Therefore, the output waveform at F/F 5 will be the waveform Z_0 (see fig. A-2) delayed by 62.5 ns (1/2 of an 8-MHz cycle). Thus, for a state transitional analysis, the problem is reduced to the boundary illustrated in figure A-2. The analysis conducted within this boundary will then show how Z_0 responds to the combination of signals from F/F 2, (X) and hardware failures of F/F 3, F/F 4, and gates G1 through G4. This boundary has reduced the circuit to one in which there are three memory elements constituting eight possible transition states.

As a reference base, table A-1 presents the operation of the circuit discussed above when there are no failed elements. From this table the waveform at Z_0 is derived which has a cycle of 1 μsec. During each cycle, there are two negative transitions 500 ns apart, which satisfies the monitor. Table A-2 presents operation as a function of 11 probable failure modes within this circuit. A review of this table shows that there are no failures that can produce any of the off-design nondetectable frequencies. However, there were a number of failures that produced a suitable signal for the monitor but not suitable for the square clock generator, the second area of concern. To better capture the significance of the data of table A-2, a summary of the table information is presented below.

Case number	Monitor tripped	Square clock signal generator
1	Yes	Stops
2	No (conditional)	Continues
3	Yes	Stops (conditional)
4	No (conditional)	Continues
5	No	Stops
6	No	Continues
7	No (conditional)	Continues
8	No	Stops (conditional)
9	Yes	Stops
10	No (conditional)	Stops (conditional)
11	No (conditional)	Stops

As can be noted, several of the failure modes are conditional. The conditional detection of these modes depends upon where the circuit is in the transition matrix at the time of the failure. Take failure case 4 for example; if the failure occurs while the circuit is in state 0, 3, 4, or 7, the monitor will not detect a change in the waveshape. This can be deduced by viewing the waveshape associated with failure case 4 and the nonfailed waveshape in table A-2. For instance, if the failure occurs while the circuit is in state 0, the monitor expects to see another negative transition in 375 ns (three states later). From failure case 4, a failure occurring during state 0 forces the circuit to state 6, then to state 3, then to state 7, at which point the negative transition occurs (375 ns later). On the other hand, if the failure had occurred while the circuit was in state 2, the next negative transition

would be anticipated one state later. However, this failure results in a negative transition three states later, thereby breaking the normal monitor cycle, and thus it is a detectable failure situation.

The conditional performance of the square clock signal generator depends upon the tolerance buildup in the delay line within that circuit; again see figure A-1. For example, during failure case 8, the output waveform at Z_0 (see table A-2, case 8) will be sufficient to produce clocking signals if the delay line has degraded to 245 ns. However, the clock will stop if the delay line has degraded to 255 ns. In any case, the most crucial of the failures is case 5 in which the failure is undetected by the monitor, and the final output clock signal stops.

As a consequence of the above analysis, the General Electric Company proposed an alternate countdown and waveshaping circuit. This circuit is illustrated in figure A-3. The intent of this new circuit is to provide the same normal clocking signal but without the undesirable failure mode. To check this, the analysis was again repeated.

Tables A-3 and A-4 contain the essential results of the repeated analysis. By utilizing essentially the same analysis boundary, as illustrated in figure A-3, no key failure could be discovered. The only failure mode discovered that went undetected was for case 1. However, for this failure mode, the clock signal is still produced. Probable detection would occur at the next power-up sequence. At that time, the outputs from the square clock signal generators in each of the three channels would not become synchronized and would be one-half cycle out of phase with the other two channels. (The designed specialized waveshape is required to force synchronization.)

In the preceding discussion, the study emphasis was based upon a single channel failure affecting multichannel operation. The remaining circuits in the oscillator monitor and switchover circuits (fig. A-1) are single channel oriented failures. (The exception being the majority logic voter, discussed in section A.2). A cursory investigation of these circuits found no failures that would encumber the other channels. The only failures found were either registered by downstream voter monitors or were latent. For the latter cases, preflight tests for cycling the majority logic voters and its associated monitors were developed (see app. D BIT).

A.2 MAJORITY LOGIC VOTER AND ASSOCIATED MONITOR

Within the ICPS, serial data, clock, and essential elements of timing and control are processed through a circuit, entitled an MLV, any time such information passes between system LRU. The system intent for this circuit is to prevent the propagation of failures from one LRU to another. At these MLV points, cross-channel monitors are utilized to register failures and to create system first and second failure status. Essential to the failure analysis is to ascertain failures that preclude timely failure annunciation so that simultaneous first/second failure situations will not arise because of unannunciated failures.

A.2.1 System Description

Figure A-4 illustrates the MLV circuit and its associated monitor. As can be seen, the section is simply implemented by the use of four NAND gates (G1, G2, G3, and data out [DO]). This implementation processes data so that the value out of gate DO is the same value as the majority value of the inputs. The monitor portion (the remaining 10 elements) interrogates the informational data string upstream of the voter (cross-channel signals).

Any time two channels do not contain the same logic levels to within the tolerance of the clocking signal, the monitor registers that fact into a latch. The latch output(s) is then logically combined to produce the first, second, and local failure status information. Once a latch has been set, it can only be cleared by manually actuating an error reset command.

A.2.2 FMECA Approach

Since the MLV and its associated monitor represented a relatively few number of circuit elements, it was used in a comparison study between hand analysis and the computerized aided design introduced in section 4.2.3. When the potential failure cases were postulated and tabulated, the number was so great that the hand analysis problem was scaled back to include only the voter circuits. Dealing with the entire circuit monitor and voter, were relegated to the simulator. Furthermore, for the hand analysis, it was assumed tnat the voter was processing signals that required the output to switch between the high and low state and that the monitor would detect any input (upstream) failures.

Results from the hand analysis (tabulations not included) were (see fig. A-4):

- Output failures of gate DO to either the high or low state will be caught by the next downstream monitor.
- Failures in the output of gates G1, G2 or G3 to the low state are also detected downstream since any low input to DO prohibits the circuit output from going low.
- The circuit can sustain a dual failure such that if the output from any two of the gates G1, G2, or G3 are high, it will still process normal, nonfailed, data.

If this latter condition is not tested, latent failures may buildup without being detected and cause a loss of fail-operational capability. For instance, if over a period of time the output of G1 fails to a high in both channels A and B, all channels will remain operative and there is no indication that a failure exists. If a short to ground on the cable containing channel C data were then to occur, total system operation would stop. This can be illustrated with the use of figure A-4. Since the output of G2 and G3 in all channels becomes high with a short on the channel C line, and since G1 is stuck high in channels A and B because of the previous latent failures, the output for DO becomes and remains zero. This zero will persist and continue to be passed on from A and B and, since it represents a majority of two out of three at the next MLV mode, the entire system becomes dormant.

A.2.3 Computer Aided Analysis

As the blocks of digital circuitry under analysis grow, the analysis becomes too complex to be treated by hand in a timely manner. To such ends, computer aided analysis will serve as a useful tool. To gain experience with computer aided analysis, the MLV and its monitor were evaluated using a logic simulator available at Boeing. The simulation (ref. 4) required logic equations of the circuit to be analyzed, a list of faults to be inserted, values of input signals (or sequence of), and a list of the outputs to be monitored.

The equations that represent the circuit in figure A-4 are listed in figure A-5. The equations are modified by a compiler for the simulator to convert them into a suitable format for insertion simulation. For example, in figure A-5 the NAND gate G1 was originally defined as

$$G1 = (A\ BUF \cdot B\ BUF)^*$$

From this, the compiler generates the fault model equation:

$$\begin{aligned} G1 = ZG000001 &\cdot ZI000001^* \\ ZG000001 &= ZG000002 + ZI000002 \\ ZG000002 &= (ZG000003 \cdot ZG000004)^* \\ ZG000003 &= A\ BUF + ZI000003 \\ ZG000004 &= B\ BUF + ZI000004 \end{aligned}$$

Note: The asterisk (*) is used to denote the logical complement.

By generating the fault model in this manner, see figure A-6, faults can be inserted into the simulation. With all ZI inputs to the gates zero, the fault model is functionally identical to the original NAND gate. But by applying a one to each of the ZI inputs, a failure mode is simulated. That is, ZI000003 = 1 functionally makes the A BUF input to the NAND gate G1 act as if it is stuck at one. Each of the input equations are modified in this manner. Figure A-7 illustrates how each device in the test cirvuit is made into a fault model. From this figure it can be seen that for a two-input gate, the failure modes to be simulated are:

1-S1	=	Input 1 stuck at one
2-S1	=	Input 2 stuck at one
X-S1	=	Output stuck at one
X-S2	=	Output stuck at zero

The next input to the simulator is a list of faults to be inserted. For this circuit there are 88 pertinent nonredundant failures. These are tabulated in table A-5. Since the input of a NAND gate stuck at zero is the same as the output stuck at one, both failure modes were not included. Submission of the input data sequence and outputs to be monitored completes the simulator data package. The simulator sequentially institutes each failure and applies the input signal sequence. The outputs are then recorded as a function of each test number. These outputs can then be compared to the results of the nonfailed case. Any difference in the outputs is then deemed sufficient as a detectable failure. Section A.2.4 contains the sequence and results of these simulation runs; a summary of this section follows.

Table A-6 contains the summary of detectable failures for the MLV circuits when processing normal data. As can be seen from this table, of the possible voter and monitor failures, 70% of those in the voter and 65% of those in the monitor go undetected during the processing of normal data (i.e., channel A = channel B = channel C). However, if it were possible to manipulate the incoming data (to the voter) such that a sequence of first and second failed information were present, the number of undetected failures becomes 0% for the voter and 10% for the monitor, see table A-7. In this table there are three failure modes that can be classified as *don't care.* These are K SAB X-S0, K SBC X-S0, and K SCA X-S0. These points in the circuit are normally grounded so that a failure to a zero is of no consequence. Three other nondetected failures (R SAB X-S1, R SBC X-S1 and R SCA X-S1) are detectable when interpreted properly. Each of these failure modes will prevent the clearing of a failure indication. However, this failure mode would be detected by the fact that the failure indicator lamp would not be extinguished when the reset button is depressed. The seven remaining failure modes are not possible to detect by any sequence of input conditions. This is because of the inherent redundancy in the functions performed by this part of the circuitry.

The redundancy involved can be illustrated by the following discussion, and the aid of figure A-4. Take the case for any failed input into NAND gate G7. The normal (no data failures) input state for this gate is all ones. This gate will then change output status (indicate a data failure) for any single input being zero. For instance, if channel A fails, both SAB* and SCA* become zeros. Therefore, if any one of these inputs were stuck at one, it would not affect the output status of gate G7. This similar redundancy design is what makes the first and second input into LFAILN, and SCA* and SAB* stuck at one latent failures. This is not deemed a deficiency since this error mode could be removed in a production system by removal of one of the redundant links.

A.2.4 Computer Fault Simulation Runs

This section contains the results of the fault simulation runs that were made for the MLV and its failure monitor. For each of the failure states of table A-7, the fault sequence of table A-8 was utilized. This table contains the test runs 1 through 14 with the simulation run procedures used as shown below.

Runs 1 and 2–Normal Data Inputs

The register level program was written to control the logic level simulation through the following sequence:

1. Insert first failure mode
2. Apply error reset pulse
3. Apply input data (channels A, B, and C)
4. Wait 500 ns, apply negative clock transition
5. Wait 500 ns, compare

6. Increment failure mode by one

7. Repeat items 2 through 6

This sequence was repeated for each of the possible failure modes, and the results are presented in table A-9.

Runs 3 through 14–Combinations of First and Second Fail Data Inputs

The register level program was written to control the logic level simulation through the following sequence:

1. Insert the first failure mode

2. Apply error reset pulse

3. Apply first fail input combination (channels A, B, and C)

4. Wait 500 ns, apply negative clock transition

5. Wait 500 ns, compare

6. Apply second fail input combination (channels A, B, and C)

7. Wait 500 ns, apply negative clock transition

8. Wait 500 ns, compare

9. Increment failure mode by one

10. Repeat items 2 through 9

This sequence was repeated for each of the possible failure modes and these results are presented in table A-10.

A.3 SYSTEM STATUS LOGIC

The full implementation of system status functions, as well as system control functions, is mission/vehicle dependent and goes beyond the bounds of the FCD task. However, the design utilized in this digital subsystem does represent the typical nature of such logic and does provide insight into associated circuit failure modes. Therefore, the section of the SSCU panel associated with first and second failure status was considered representative of the system problem.

Within section A.2, the results from the cross-channel MLV monitors are combined, see figure A-8. The latched status in each of these monitors is functionally ORed to produce an indication of failure status. The annunciation of first, second, or local failure is driven from

each channel within the system, even though only a single channel is illustrated in figure A-8. Local failure indication, which also includes in-line monitoring (power, overflow, parity, etc.) via computer BIT, are not truly considered part of the system failure status. In many respects they represent maintenance information, and the recombination of such failure states does not enter into the system functional operation.

The failure mode tabulation that details the effects associated with failures of the status circuitry is shown in table A-11. Only first-failure circuitry is included since the second-fail circuitry is of identical design. The health of the failure status logic is unknown until it is exercised to annunciate a failure. Thus, it becomes necessary to check this circuitry by tripping each of the failure monitors for both first- and second-failure cases to assure signal path continuity. This is extremely important if automatic system disengagement is going to be derived from the status logic.

A.4 SENSOR SIGNAL SELECTION/FAILURE DETECTION (SSFD)

All variable signals coming into the computing subsystem (ICPS) pass through the CIU SSFD circuit. Its function is to select or create a single signal from three similar, but not necessarily alike, digital signals. This selected signal is then passed on to the computer unit and enters through a majority logic voter. Cross-channel monitoring on the SSFD input signals provide failure state logic to mode the operation of the signal selection algorithm.

The signal selection algorithm, implemented in hardware, selects the median value from a set of three sensor signals. When one of the three sensor values exceeds a preset difference from the other two, the average value of the similar two signals is selected as the SSFD output. The determination of the out-of-tolerance signal is based upon a set of parameters applied to the differences between each pair of the signals of the sensor set. These parameters relate to:

- Lag filter time constant
- Allowable magnitude out of the lag filter
- Allowable time the lag filter output is above the allowable magnitude
- Washout filter time constant
- Allowable magnitude out of the washout filter
- Allowable time the washout filter output is above the allowable magnitude

This selector and its monitor circuits are multiplexed and utilized on all of the incoming sensor signals once each frame time. The timing, acquisition of data, retrieval of monitor parameters, and registering of failure codes are all accomplished by harness wiring. The parameters associated with the failure monitor, however, may be uniquely tailored to each set of triplex sensors through read-only-memory programming.

Two somewhat independent FMECA studies were made to investigate the SSFD circuits. Neither study resulted in an in-depth FMECA treatment. The first study began by developing a complete circuit level breakdown of the SSFD functions. This entailed constructing 12 detailed drawings where each drawing covered the piece-part makeup of specific subfunctions within the SSFD circuitry. Figure A-9 is the top drawing of this set of drawings, showing the breakdown of the SSFD subfunctions. Figures A-10 and A-11 are shown to illustrate the complexity of the circuits contained within each subfunction. The set of drawings was then used to analyze the failure propagation/manifestation for postulated piece-part failures. This approach soon became unmanageable because of the circuit complexity and associated time and manpower demands it placed on the FCD task effort.

The second effort utilized the same drawings but was an analysis of the SSFD functions from a top-down functional point of view. In this approach, failures were postulated on a functional level and a further breakdown of the function only took place when a failure proved to compromise the SSFD operation. The results of this cursory analysis are contained in table A-12. The results show that 68% of the total circuitry may contain latent failures. These latent failures primarily exist because the SSFD monitoring circuits are off-line functions under no failure situations (5% associated with the signal selection mode switching and 63% associated with the monitor functions).

From this study, two conclusions could be drawn. First, since the signal selection circuits of the SSFD process data in a serial fashion and all paths through this portion are utilized in the processing of normal data, failures within these circuits would be detected by the downstream MLV monitor (in the computer unit). Second, all the monitor related functions are off-line and can develop latent failures, which, if accumulated in two or more channels, could lead to hazardous situations with subsequent failures in one or more sensors. These latter circuits must be exercised periodically in order to establish their operational integrity.

A.5 ANALOG INPUT/OUTPUT CIRCUITRY AND TIMING

These circuits condition, level change, and transform the signals from outside the subsystem to a form suitable for manipulation within the computing subsystem. In a similar fashion, they transform signals within the computing subsystem to a form compatible to the external subelements. In general, the signal flow of any one particular input (or output) signal has its own dedicated path as long as it is in the analog signal form. Once converted to a digital form, the signals share the same data path by a multiplexing process that stages (sequences) signals to and from the digital processor through fixed time windows.

A substantial portion of these circuits lend themselves to analysis utilizing analog techniques. In developing the FMECA on these circuits, a top drawing was created as a guide, figure A-12. Hardware design drawings were then redrawn to functionally represent the subfunctions identified within the top drawing. Using these drawings, a failure modes table was developed (table A-13). The following paragraphs discuss the failure mode results presented in table A-13 relative to each subfunction illustrated in figure A-12.

A.5.1 Analog Input Circuitry

Both dc and ac sensor signals are processed within the circuit boundary shown in figure A-13. Each incoming signal is passed through a prefilter, which is utilized to condition the signal against possible line noise and/or aliasing errors created by digital sampling. In the case of the dc signal conditioner, a filter with a double order break near 110 rad is used. This double breakpoint was selected in order to alleviate aliasing of the signal for the 6.144 ms I/O frame rate. For the ac prefilter, the breakpoints were set to keep a 10 V rms signal at 400 Hz from having more than 2.5 mV ripple (1/2 LSB of the A/D converter). These signals then pass through an analog multiplexer to a single A/D converter. The A/D converter, upon command, converts the analog signal to its equivalent binary value represented by 12 bits including sign. At the end of the conversion, the digital value is loaded parallel into the lower 12 bits of the A/D register. (The upper 5 bits of the 16-bit word register are made identical.) From this register, the data is shifted LSB first to the signal selection circuitry via the serial gating circuits.

Failure modes of the analog processing circuits, demodulators and filters, are predominantly either high gain or passive. The ability to detect the passive failures will be dependent upon the sensor tolerances and downstream signal selection and failure detection levels. The sensitivity to these detection levels will be discussed later in this section. Failure effects of the A/D converter used in the ICPS could not be adequately defined because a schematic for this circuit could not be obtained. Therefore, in order to develop an appreciation of the types of failure modes possible, a converter design was postulated (fig. A-14). The analysis results presented in table A-13 are based upon this figure. Before the failure modes of this circuit are presented, a short explanation of this circuit is felt to be beneficial.

The input amplifier of this converter takes a ±10-V analog input signal and converts it to a zero to -10-V signal so that a 0-V input signal corresponds to a -5-V level. The biased analog signal is then summed with the current from a precision ladder network into a high gain comparator. These ladder network currents are controlled by F/F's such that if the F/F output is set high, the ladder input is connected to a precision voltage source; if the F/F output is low, the ladder input is grounded. The analog to digital converter has its own clock signal generator that is independent of the CIU clock signals, except that it must receive a convert command to start the conversion process.

Assuming a standing analog signal exists at the analog input, a convert command initializes the counter or shift register (see fig. A-14) and causes a clear signal to go from the timing logic through the OR gate to all F/F timing gates. All the F/F timing gates are opened at T_0 to clear all the F/F's to zero. The counter/shift register then starts counting or shifting, and the timing logic opens each F/F time gate (one at a time) in a sequence starting with the most significant bit (MSB) and progressing to the least significant big (LSB). During the open gate time interval for each F/F, a set pulse is utilized to trial set the F/F. If setting the F/F makes the ladder network output greater than the biased analog value, then the comparator causes the F/F to clear again before its time gate closes. Proceeding in this successive approximation mode, a 12-bit word is created to represent the input analog value. The resulting 12-bit digital output is biased to the positive half of the binary number system so the maximum positive output is all ones and the maximum negative output is all zeros.

The biased binary output is converted to the twos complement number system by negating the most significant bit shown as MSB in figure A-13. The 12-bit number is then parallel loaded into a 16-bit shift A/D register with the sign bit of the 12-bit word repeated in the five most significant bit positions of the 16-bit word.

Referring back to table A-13, one can see that the bulk of the failue modes is reflected by signal degradation. For instance, if the output of a F/F in the A/D converter fails high or low, it will cause the other F/F's to set so they are trying to correct the error introduced by the failed F/F. If the analog value is such that the failed F/F is in the correct state for the present conversion, then no error will be introduced. For example, if the MSB (bit 12) is failed into the high state, no error is introduced for any analog signal between 0 and +10 V, but all negative input signals would be set to zero. Thus, failing the MSB in a fixed state (zero or one) will make the A/D converter behave similar to a half-wave rectifier. This is illustrated in figure A-15 view (a). The effects of failing the output of either F/F 11 or 10 high are also shown in figure A-15. To understand the effects, consider that if bit 10 is failed high, it can cause bit 11 to set low when it is supposed to set high. If both bits 11 and 10 are supposed to be low but bit 10 is failed high, then all lower order bits will set low to try to correct the error. As a comparison, the effects of failing any of the lower three bits high are shown in figure A-16. From these drawings it can be concluded that single bit failures simply degrade the resolution capability of the A/D conversion, and if the bits fail high they prevent the digital output from having a zero or null value. This type of resolution loss would exist on all input signals of the failed channel. The capability to detect these failures with normal data depends on the bit significance being greater than the downstream monitor failure detection level and signal motion being great enough to cause the signal to jump in excess of the monitor threshold.

To better visualize the significance of each of the 12 bits of the A/D converter, in terms of the fractional part of a full-scale value, refer to figure A-17. If a signal has a monitor threshold setting of ±5% of full scale, only failures in the top four bits (i.e., bits 9 through 12) could be guaranteed as detectable. The significance of the first eight bits would be less than the failure detection level (10% of full scale). Thus 75% of the bit failures under these conditions would probably be latent passive. Such failures in two or more channels could cause control degradation unless downstream signal selection/failure detection guards against such degradation.

Failures associated with the internal timing signals might have the effect of mixing data from previous conversions with the present conversion. This results in an unpredictable output since the data significance of the previous word may have little relationship to the current word (i.e., the A/D is time shared between a maximum possible 32 different input signals) so one cannot make the assumption that there is a small change between the previous value and the present value of any conversion.

Failures of the A/D converter actually occurred during the laboratory tests conducted as part of the FCD studies. The most common failure was one manifested by a low frequency wandering offset. Its magnitude generally fell within the first five bits. Another failure mode was one in which the converter refused to respond to convert commands; a typically passive mode. However, since twos complement data were being extracted (MSB being complemented), this failure was manifested with full-scale negative output; a situation easily detected by the sensor monitoring circuits.

Failures in the A/D register (fig. A-13) are similar in nature to those described for the A/D converter. A failure to load bits into the register would normally be detected only if it occurred in the upper four or five most significant bits. However, failures that caused the bits to shift out as zeros or ones would have the effect of making the input vs output characteristics look like a sawtooth. Figure A-18 illustrates this example for the case of failures to zero. The peak of the sawtooth will depend upon the significance of the failed location. A failure in a lower order bit will make the peak of the sawtooth very low. A failure in bit position 12 will make the sawtooth peak at full amplitude (fig. A-18 view [b]). It should be noted, as illustrated in figure A-18, view (a), that a failure in a bit position above the normal 12-bit converter will produce an equivalent signal in excess of the nominal scaling range.

A.5.2 Discrete Input Circuitry

These circuits (fig. A-19) provide level changing of the incoming discretes (0 to 10 or 28 V dc) to 0 and +5-V logic levels. These signals then, like the analog signals discussed above, pass to a multiplexer. The multiplexer output is transmitted to an MLV signal selector. All failures of the input discrete circuits result in a fixed or wrong output value. Therefore, the downstream MLV and its associated monitor will prevent error propagation and provide failure detection.

A.5.3 Input Signal Serial Gating

This circuit (fig. A-20) multiplexes all incoming signals to be transmitted to circuitry. Manual initialization (system computational reset), as well as sensor data, is sequenced and placed on a serial data buss. In the case of the sensor data, a parity bit is added in the serial string in order to give further credibility to the transmission link. This parity check essentially clears all failures within this circuit with the exception of the gate through which the sensor data passes (see fig. A-20). Failures of this element force data to be either all zeros or ones. For twos complement notation, this is equivalent to a near zero value. Parity will not assist in this error detection because the calculation of the parity bit is downstream of the failure point. Detection for this error will require signals to be above their associated monitor threshold of the sensor selection circuitry.

A.5.4 Output Buffer and Discrete Circuitry

Data processed by the digital computer reenters the analog interface unit through the bit buffer, see figure A-12. This buffer serves to transform the serial data string into the time reference of the interface unit. From this buffer, the data is distributed to either the D/A interface (to be discussed in the next section) or to the discrete output circuits (fig. A-21). Discrete data, which arrives throughout the frame period, are shifted and stored into a receiving register one bit at a time. At the end of the frame, the contents of the entire register are simultaneously transferred into holding latches. From these latches, the discretes are further buffered and level converted to drive peripherals.

The majority logic voter upstream of the bit buffer prevents failure propagation of upstream failures. Voter output failures or the bit buffer failing high or low will cause all

serial data to output as ones or zeros. This will cause all discrete outputs to be zero. Detection of these failures will depend upon the system downstream monitoring.

The discrete shift registers could fail a bit location low or high so that all bits downstream of the failure location would shift in as zeros or ones. Since there are two shift registers feeding the quad latches, (fig. A-21), a single failure could cause a failure of one to eight of the possible 16 outputs. Loss of a clock signal to either shift register or to both shift registers could cause 8 or all 16 of the discrete outputs to be unchangeable in the failed channel. If the clock signal to the shift registers occur at the wrong time, then the registers will load in the wrong data. These failures can be tested by verifying that the eighth bit in each of the two eight-bit shift registers can be changed high or low. In general, the quad latch can fail and output high or low or all outputs or any group of four outputs could lose resetting capability. Detection of these failures will again depend on the downstream monitoring (external to the digital equipment).

A.5.5 Analog Output Circuitry

Figure A-22 illustrates the circuits that process analog output data. The first stage of signal processing through this circuit is to check for a potential overflow situation. This is required because only the lower 12-bits of the 16-bit word received from the computer is actually passed to the D/A converter. The potential overflow is checked by interrogating the five uppermost bits of a register, which is temporarily holding the computer output value. If they are not the same, an overflow is indicated (fig. A-22). The corrective action that is then instituted is to convert the data word to a full-scale value of a polarity indicated by the sign (16th) bit. This is accomplished by loading the maximum negative value into the temporary register for negative overflow, or by forcing all ones into the data string as it is passed to the D/A register.

Once the data has been passed to the D/A register, it is converted to its analog equivalent. The conversion process was accomplished by utilizing each bit position in the D/A register as voltage weighted information into an operational drive amplifier. The operational amplifier was then used in turn to drive sample/hold circuits. These circuits are sequenced one at a time so that they slew to the current value on the drive amplifier. After a sample/hold has acquired its output values, the input to it is disabled and the output made to retain its value. The D/A is then allowed to develop a different signal, which is then placed on the next sample/hold circuit.

Failure detection of these circuits (tabulated in item 2.4 of table A-13) depends upon monitoring in the next downstream subsystem. In summary, failures in the overflow detection circuits cause all analog outputs of the failed channel to ± hardover, zero, or the inability to detect overflows. Failures in the D/A register or the computer data buffer that prevent lower ordered bits from being shifted in produce a stairstep output (fig. A-23). The granularity of this failure, of course, depends upon the location of the bit failure. However, failures in the D/A register such that the register properly shifts in the data but one of the output bits to the converter is failed high (or low) produce an output waveshape illustrated in figure A-24. Such a waveshape would undoubtedly have tremendous impact upon any closed-loop control system if proper design at the servo-subsystem does not block such a

signal. Again, the effect of this error depends upon the location of the failed bit. For dramatization of the effect, figure A-24 illustrates failures in the upper bit positions. These failures would exist in all analog outputs of the failed channel since the failure is in the demultiplexing data string. The typical failures would be the inability to sample, or to hold.

The number of failures that fail all analog output paths in one channel could be reduced by deleting the sample/hold circuits and by using a separate D/A register, converter, and operational amplifier for each analog output.

A.5.6 Input/Output Timing

This is the final circuit function within figure A-12 to be discussed. It is within this subfunction (fig. A-25) that all the control of the hardware elements within the CIU are developed. The essential input to this circuit is the selected square clock signal from the redundant clock circuitry. (Iteration reset serves as an initialization signal to develop synchronization.) The square clock is waveshaped into three different 100 ns pulses as is illustrated in figure A-25. These pulses drive the clocked sequential circuits witnin the CIU. In addition, one of these clock signals (bit clock) is utilized to drive the bit time shift register. This shift register develops the basic 48 time partitions within a timing window identified as one algorithm time–one complete cycle of the register represents one algorithm time. Every cycle of the bit time shift register is utilized to increment the word counter (and advance counter). The significance of these counters identifies which algorithm time (0 to 127) is currently being processed. The advance counter and the word counter essentially perform the same function; the only difference being that the state of the advance counter is always two algorithms ahead. The purpose for this is to preinitiate the processing of incoming sensor data so that there is sufficient time to process data through the signal selection circuitry before it is required within the computers. The amalgamation of the clock signals, bit shift register, and word counters (through combinational logic) produces the actual device control lines. These control lines manage the various pieces of hardware within the CIU such as initiating A/D conversion, shift control to various registers, multiplexing signals on to data busses, etc. The sum of all these signals results in the fixed (hardwired) executive structure for the CIU.

Since the signals (control lines) from these circuits manipulate so many hardware elements, any failure in them that produces off design signals would have massive failure effects throughout the CIU. By reviewing item 3 in table A-13, it can be shown that failures in any of the clocking signals generally prohibit input and output processes because clocking of the signal transmission devices will stop. Thus, clock signals failing dead (high or low) are easily detected by normal monitoring. Further analysis was then conducted to discover potential failures that would change the clock signals but not destroy them. These failures center about the two-input NAND gates which are used to develop strobe and clock (CLK) signals. For instance, if the input into the strobe NAND gate from the delay line fails to a high, (refer to fig. A-25), then the strobe signal becomes the square clock signal. The leading edge of this failed strobe signal would occur at the same time as the leading edge of the normal strobe signal, but the trailing edge would occur 400 ns later than normal. This increased width of the strobe pulse brings its high time equal to the edge of the leading edge of the CLK pulse. Since strobe actuated devices are dependent on both the leading and

trailing edges of the strobe, it appears that this failed strobe signal would cause race problems in the signal processing. The net results cannot be easily predicted.

This long strobe pulse time opens up the possibility of the strobe devices having incorrect state changes caused by switching noise, synchronous noise, etc. Many of the possible consequences are ambiguous or unpredictable without laboratory test. In general, it was concluded that all failures would be detected. However, laboratory tests would have to be conducted to verify such a prediction for a production system.

Failures in the bit time shift register and word counters are detectable by normal monitoring. For instance, if a bit in the shift register fails to a high or to a zero in a manner that causes it to shift the failure into the downstream bit locations, the sequencing of the register output would soon end in a static pattern. After being reset to zero at iteration reset, the shift register would again reach a steady-state failure condition in NF-1 clock pulses where NF is the number of the failed bit location. Thus, since the register is not cycling, the word and advance counters could not count past the first algorithm time, and the input and output timing would stop in the failed CIU. Likewise, failures in the counter circuits either stop the count sequence, or they cause the wrong count sequence to occur. These failures in turn preclude the orderly calculation of the event time points, which are required to process data.

A.6 COMPUTER UNIT

All of the previous sections of this appendix have dealt essentially with hardwired data handlers and formatters. Within the computer unit, data handling is both hardware and software; i.e., data handling as a function of 2 programmed series of instructions that control the hardware operational states. As a consequence, hardware elements may exist that fail and go undetected until the software (instruction set) requires that they function. The study, therefore, was to identify possible latent failure states within the computer unit that would be associated with the processing of flight control functions.

A.6.1 Processor Description

In order to better comprehend the construction of the computer, it is beneficial to review the mathematical model of the incremental control processor (ICP) presented in reference 1. First, the processor executes mathematical operations utilizing incremental arithmetic. Second, the hardware to accomplish this task is mechanized to solve a general incremental equation, the rho equation, expressed below:

$$\rho_i = \rho_{i-1} (\pm) S_q \Delta Q (\pm) S_p \Delta P (\pm) V_i \Delta W (\pm) U_i \Delta T$$

where

ρ_i = value of ρ at the ith iteration

$U_i = U_{i-1} + \Delta U_i$

$V_i = V_{i-1} + \Delta V_i$

This equation is then further modified to create the output value, ΔZ, which is defined as:

$$\Delta Z = \rho_i / S_z$$

where

$$S_z = V_i \text{ or } S_q$$

and ΔZ is constrained to one of the values 0, ±1, ±2, ±4, ±8, ±16, ±32, or ±64.

The rho equation is executed 128 times per processor iteration. The output of one execution, an algorithm time, is ΔZ. This output can be used in succeeding algorithms as an input for the values of ΔQ, ΔP, ΔW, ΔT, ΔU, ΔV, or as a computer output. Computer programming manipulates the rho equation by specifying where in the data memory the values of the Δ's are to be found (either as variables or constants); the sign of summations; the values of constants (S_q, S_p, U_0, V_0, ρ_0); and the source of S_q/S_p as being either from the program memory or from the data input/output (X/Y) register. By selecting various combinations of these parameters, a highly sophisticated filter (second order over second order) can be implemented using two algorithms.

The limitation to the allowable values of ΔZ is a hardware consideration. Even though this limitation results in a degradation of equation resolution during any single equation execution, there are two advantages to its implementation. First, the values (0, ±1, ±2, ±4, ±8, ±16, ±32, ±64) can be decoded into a four-bit word by using 2's complement notation. This feature reduces memory volume requirements. Second, by having these input parameters to the rho equation expressed as a power-to-the-base-two, multiplication and division of parameters can be executed simply by shifting operations. This reduces computation time and hardware elements required for execution of the rho equation. Through programming, the resolution error difference between the selected ΔZ output and the ideal calculated value can be stored within a rho register to be used during succeeding computations of the same algorithm. This results in a servoing process of the output until there exists a zero difference between output value and the true calculated value after a number of iterations.

The last key feature of the incremental control processor is its hardwired executive. Since the execution of a control law function is developed by a series of algorithms, the ICP executive relieves the programmer of the task of linking these algorithms together by always executing a fixed number of algorithms per iteration (frame time). This fixed architecture allows the subsystem to be mechanized extensively with shift (ring) registers as a means for storing the variables (ρ, U, V, X, and Y). Since the executive always performs 128 algorithms, the data storage locations within the memory and the ring registers can be hardware assigned as a function of the time within the frame that the ΔZ_i is calculated.

A.6.1.1 Hardware Organization

There are essentially five major functional blocks that comprise the ICPS computer. These functions are illustrated in figure A-26. The functions enclosed within boxes are

discussed in the following paragraphs. The other primary elements are the serial MLV's, previously discussed in this appendix. The voters are utilized as functional barriers for isolating iteration, clock and data informational failures among LRU's.

As shown in figure A-26, sensor data enter from the interface units via a majority logic voter. (The interface units have already performed a signal selection process.) These data are then changed into increment form by the incremental I/O function illustrated in figure A-27. The increment value is created by subtracting a signal's past value from the current value. The difference is then passed through an increment selector whose output is stored into a random access memory (RAM) as a ΔX. During the next frame time, the ΔX increment is summed with previous ΔX's of the same signal. This sum is also stored in a long shift register (identified as the X/Y register in the figure) as X data. (Servocommand or output data are stored in the same shift register as Y data.) Discrete data are stored directly into RAM without the incremental selection process (the interface unit having already formatting data into a four-bit coded number).

Under selective programming, the current value of a sensor's input can be used in an algorithm's computation instead of ΔX's. A special 16-bit register (illustrated at the bottom of fig. A-27) can hold one full-sized variable until it can be utilized within the control law computation at some later algorithm. A new variable can be stored in the register after it has been utilized.

Command outputs are generated by summing ΔY's into the X/Y register. These outputs are then serially shifted to the interface at the appropriate timing window. It should be noted that X and Y values are interleaved within the same shift register. During the algorithms times when data are not sent from the interface unit to the computer unit, the X portion of the register may be treated like any other accumulator within the rho equation. Likewise for servocommand outputs, the Y register is also available during nonassigned (hardwired) algorithm times.

The outputs from the increment selector in the I/O section are stored in a random access memory within the increment storage section illustrated in figure A-28. The storage addresses for these data (both ΔX and ΔZ) are determined by the hardware (word time counter) as a function of time during each frame. These values are then retrieved (to be used during computation) as a function of software programming, illustrated in the bottom half of figure A-28.

The read only memories, illustrated in the top half of figure A-28, store the program instructions and constants. These memories contain the addresses for the RAM of the ΔZ values desired in the rho equation, as well as all the parameters for controlling the arithmetic section. These latter values are represented as initial conditions (U_0, V_0), gains (S_p, S_q and any Δ's that are to be constants), rho equation moding (signs of summations and selection of shift register outputs), and branching data. The addressing and moding data are utilized in a parallel fashion, while the initial condition values and S_p/S_q values are sent serially.

Addressing for the program memory is controlled in the timing section illustrated in figure A-29. This address is broken into two parts. The first address represents the major address for the ROM. This major address refers to the first location of a 12-word block of

memory in which all the parameters per algorithm are stored (192 bits). Each individual word within this block is retrieved by a minor address, an address formulated by a counter that updates once every 3 μsec. Under normal situations, addresses for instructions are sequential. These addresses are created by incrementing the major address as it shifts around a ring register (fig. A-29). During a branching instruction, a new address is inserted into this ring if the branch test conditions are met. When a branch has been executed, selective registers are automatically reset. Branching is facilitated by comparing the current next major address with the major address that was used in the previous frame during the same time period (a data value that had been stored in the rho register).

The remaining timing and control logic, illustrated in figure A-29 (bit shift register, word counter, and combinational logic), is similar in design to that used in the interface unit. Since timing and control logic is discussed in section A.5.6, it will not be repeated here.

The remaining functional element in figure A-26 is the arithmetic unit, illustrated in more detail in figure A-30. Within this section, the various variables are combined (products as well as sums) to make up the rho equation. The equation is so mechanized that as each input variable arrives, it is used in creating the partial sums. The mode control bits utilized within the expression are latched so that they persist throughout the computation. These bits are used to close switches, illustrated in figure A-30, and to complete the various data paths. (Note; control bits that can change the sign significance of incoming data were omitted from the figure for clarity purposes.)

A.6.1.2 Hardware Monitors

There are several monitors contained within the computer that check the health of all memory related devices. Checks are made on the program ROM (instructions), data RAM (results of computations), and on the major shift registers (U/V, X/Y, and rho registers). Figure A-31 illustrates the parity checker used for the ROM. This circuit is designed to check that the sum total of the 192 bits, which comprise an algorithm instruction, is odd. This is done by comparing the running total of the parallel ROM words to the serial ROM words once each algorithm. Thus, each bit of the 12-word instruction set is checked once each iteration. The only exceptions are the instructions that reside in branch loops, which are not normally executed. It should be noted that all of the serial data for algorithm (n) are not completely read out of ROM before some of the parallel data for algorithm (n+1) are being fetched. To account for this fact, the bit sum significance from the parallel word toggle is double buffered (not illustrated in fig. A-31) so that the proper sum is available to compare with the serial data.

The RAM is checked by the use of data comparison as illustrated in figure A-32. Any time data are stored into the RAM, the information remains on the write buss until the data can propagate out onto the read buss. If any of these bits disagree, an error is flagged. This ensures that proper data are stored within the RAM at the time it is loaded.

Data in the major shift registers are parity checked by the scheme illustrated in figure A-33. During each algorithm time, a running total on the number of bits going into the U/V,

X/Y, and rho registers is maintained. At the end of each algorithm time, a parity bit is generated such that the sum of all of these bits is odd. This is then stored into the rho register along with the address bits for the branch test. In a similar fashion, the data are brought from these registers and parity checked. This check assists in ensuring that data have not been improperly shifted through the registers during an iteration.

Arithmetic overflow is the remaining type of monitoring function that is illustrated in figure A-34. This monitor assists in detecting programming errors as well as registering some hardware failures. Not all of the summations within the rho equation are monitored, only those illustrated in figure A-34. The other summations (like $U\Delta T + S_p\Delta P$) do not require monitoring because the resulting word size is increased to accommodate any possible arithmetic overflow. In the same fashion, multiplication (integer in this machine) is not monitored because a larger word is generated.

The remaining monitoring functions (not illustrated) are for power supplies, clock signals (after waveshaping), and the cross channel data entering via the MLV's. The outputs from all of these monitors are combined to produce local and first failure status.

A.6.2 Processor FMECA

A failure analysis of the computer at the component level represents a highly difficult task because of the complexity of the circuits involved. A functional failure analysis was therefore conducted to determine if by having a program loaded (HSAS) would the computer be self-monitored? An affirmative answer to this question would depend on the monitoring system's ability to register failures. Thus, the analysis for latent failures was conducted by postulating failures in the computer hardware and determining whether the designed functions of the built-in-test equipment would register them through the processing of normal data and thereby validate the operational capability of the computer processor. For this analysis it was assumed that the monitoring system was operational. Further, the results of the analysis assume a downstream voter/monitor to prevent the propagation of failures from the computer and to register failures of the computer. Table A-14 contains the results of the analysis. Pertinent data from this table are presented below.

A.6.2.1 Incremental Input/Output

Within this section there are several latent failure possibilities associated with the whole-word input buffer. Because the input buffer was not used in programming the HSAS control laws, its associated failures were classified as *don't care* failures. If the register had been utilized, the latent failures would have essentially disappeared because any error to the input signal from the interface would create an output command different from the other two channels. Depending upon the nature of the failure, this would be detected either by the output monitor or by the overflow monitor.

The ability of the increment selector to select all significant values may be of some concern. However, any execution of more than 1.5% of the full scale of any sensor during any frame time will test this function. This error, if it was present, would then propagate through the computation until caught by the output monitor. Therefore, normal signal excitation should clear this potential failure mode.

The only other remaining potential latent failures within the incremental I/O are those associated with initial conditions. These failures can prevent the insertion of proper starting values during the actuation of computational reset. They become a latent failure condition when they occur while the computers are in the compute mode. However, when computational reset is again actuated, these failures will be manifested by the output monitor.

A.6.2.2 Program Memory

This area contains only one latent failure possibility that is associated with the integrity of the control bits of algorithms which are not in the mainstream of a computational cycle (instructions in a branch loop). The potential failure is whether the desired instruction bits are present in their assigned storage locations. Such failures can go undetected until that portion of the memory is exercised since some portion of each ROM chip are utilized during each iteration, their input/output buss structure, address decoding, and power status are checked by normal operation. All other element malfunctions within the memory circuit result in active failure indications becuase of the multiplexed serial architecture of the design.

The ROM parity checker is the key to memory fault detection. Each bit in a semiconductor memory occupies such a small area on the chip that faults due to heat, semiconductor imperfections, or thin oxide layers will generally encompass more than one bit. There are also circuit dependent failures, which can affect several bits. Often, several bits are electrically attached to the same metallization strip, so that a single short or open anywhere on this strip will cause all attached bits to fail. With this background, semiconductor memory designs and parity checking structures have been developed that will provide a very high probability of failure detection. The following discussion presents an overview of the associated failure detection success.

For the case of a one-bit failure in a memory word, a parity check will always detect the error. (It is assumed that if a zero or one bit fails to zero or one, there is no change to the data significance; therefore, this represents a don't care fault.) By looking at the case involving two failed bits in a single memory word (fig. A-35), it is possible to derive the generalized expression for the probability of detecting a failure as:

P (don't care failure/x bits fail) = $1/2^x$

P (detectable failure/x bits fail) = $1/2$

P (undetectable failure/x bits fail) = $1/2 - 1/2^x$

P (detecting the fault or don't care fault/x bits fail) = P (detectable failure) + P (don't care)
$1/2 + 1/2^x$

where x equals the number of failed bits/word.

Thus, a design using a single parity bit to check parity on a single word can offer little more than 50% error detectability on a single word. But, since several memory words reside within the same ROM device, the probability of detecting a ROM error within one frame time can be expressed as:

$$P(\text{detecting memory fault}) = 1 - [P(\text{undetectable failure})]^f$$

where f equals the number of words tested.

Now, if x equals 17, the probability of detecting the failure is:

$$P(\text{detecting the fault in any single word}) \geq 1/2$$

And, if f equals 128 (the times a ROM device is accessed/frame), the probability of detecting a memory fault becomes:

$$P(\text{detecting memory failure}) = 1 - 2.9 \times 10^{-39}$$

which is a very good probability.

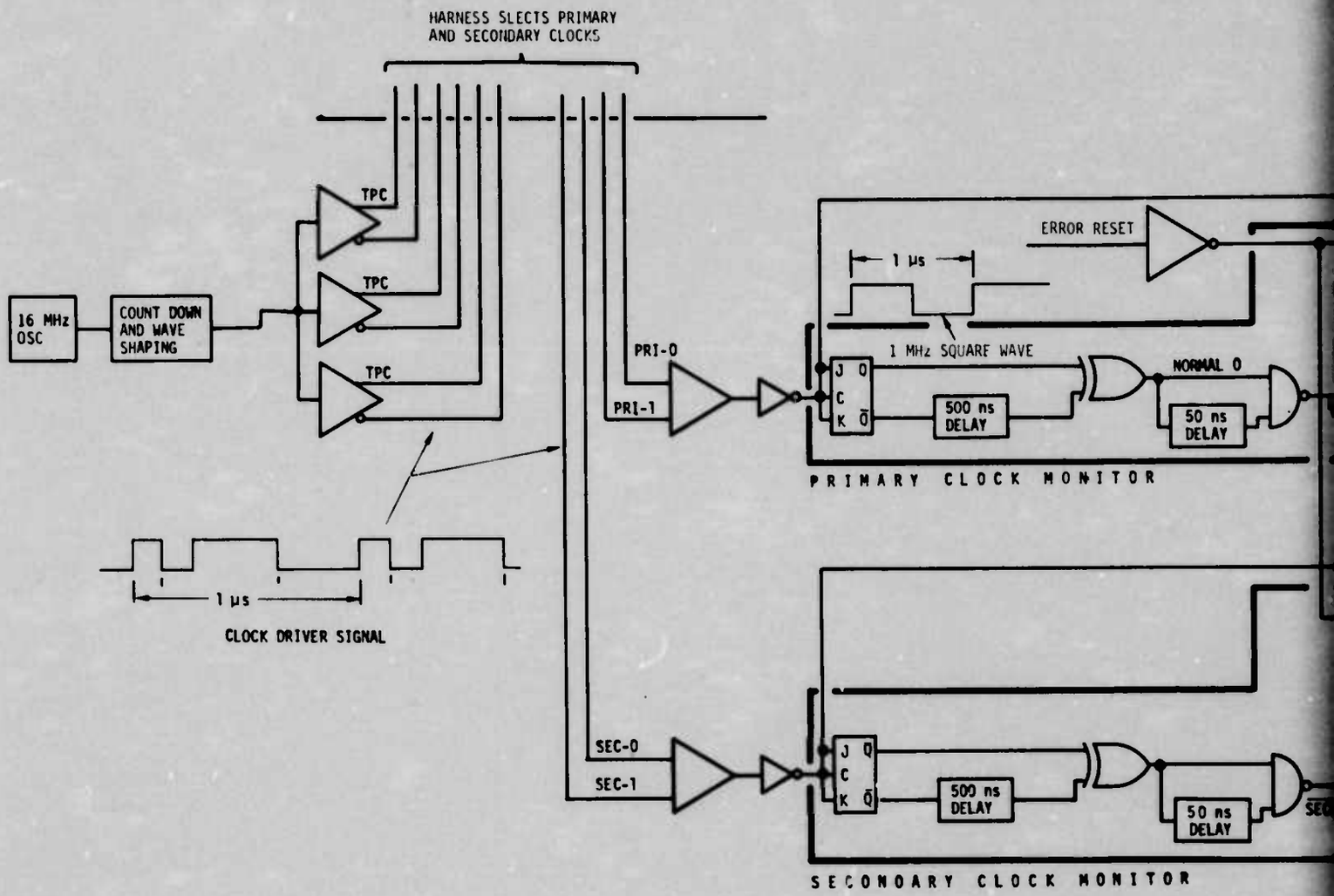
Therefore, from this representative analysis, the ROM parity checker is adequate in catching failures associated with semiconductor memories dropping or adding of bits.

A.6.2.3 Increment Storage

This section contains one potential failure mode that is somewhat self-correcting and therefore may go undetected. This is associated with the address decoding of memory cells. When data are stored in incorrect cells then retrieved from the same incorrect cells, the error becomes self-correcting. If the decoding error results in an overlay operation where a word is placed upon another, then the output monitor would be the main failure detector. However, overlays may go undetected if data placed in incorrect cells are retrieved before that cell is again used during the same iteration. Such undetected failures can be considered don't care failures.

A.6.2.4 Arithmetic Unit

The most prevalent latent failures within these circuits are associated with moding and functions unused during the FCD task (HSAS software program). However, it seems reasonable to extrapolate that these errors would be registered by the output monitor if the functions had been required in the program. One potentially significant latent failure is whether the increment selector can generate the MSB of the data. In order to establish this, an output value of at least 8, of a possible 64, would have to be exercised. All of the remaining latent failures relate to the insertion of initial conditions.


HARNESS SLECTS PRIMARY
AND SECONDARY CLOCKS
16 MHz
OSC
COUNT DOWN
AND WAVE
SHAPING
TPC
TPC
TPC
PRI-0
PRI-1
ERROR RESET
1 µs
I MHz SQUARE WAVE
J Q
C
K Q̄
500 ns
DELAY
NORMAL 0
50 ns
DELAY
PRIMARY CLOCK MONITOR
1 µs
CLOCK DRIVER SIGNAL
SEC-0
SEC-1
J Q
C
K Q̄
500 ns
DELAY
50 ns
DELAY
SECONDARY CLOCK MONITOR

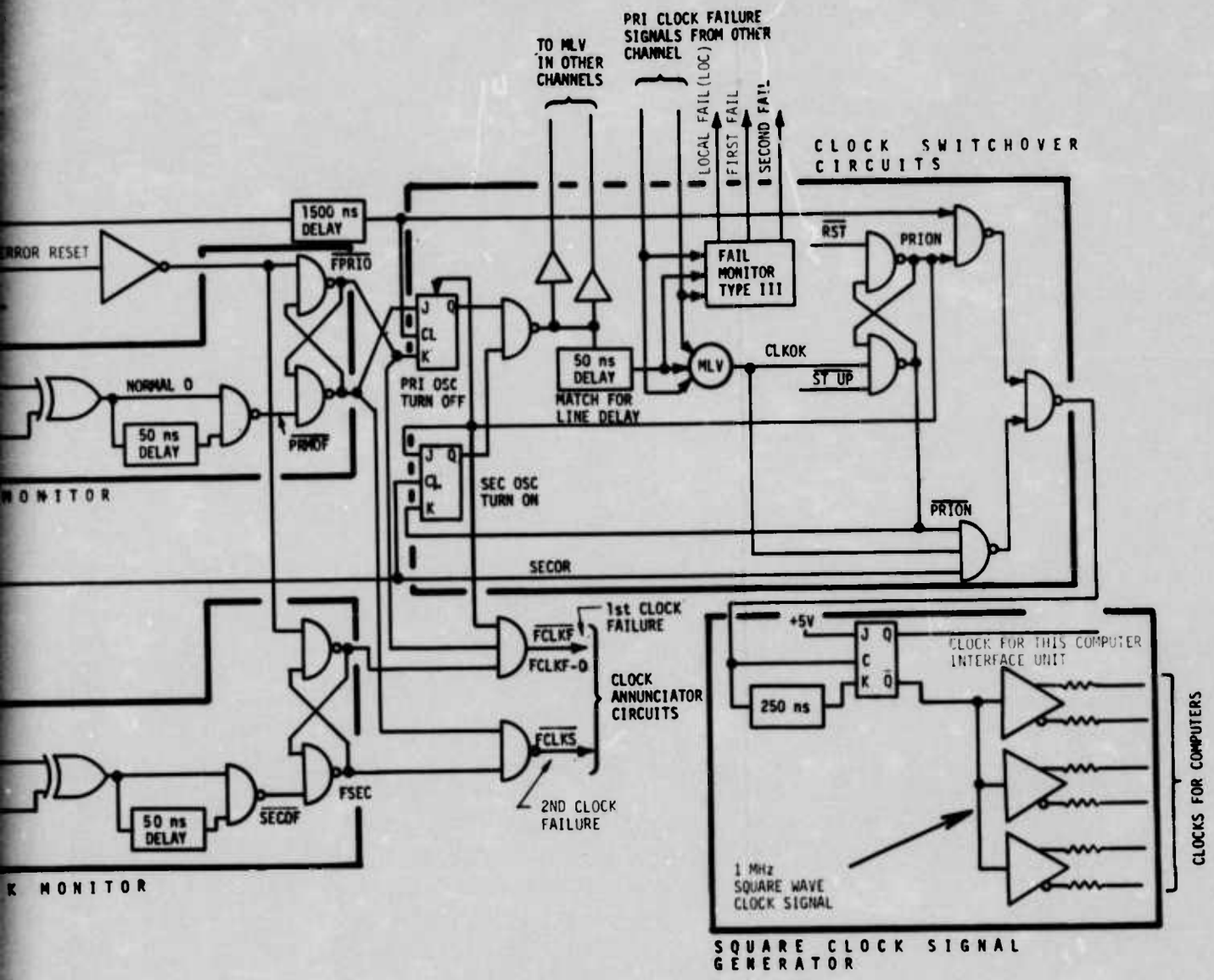


FIGURE A-1.–OSCILLATOR MONITOR AND SWITCHOVER CIRCUITS

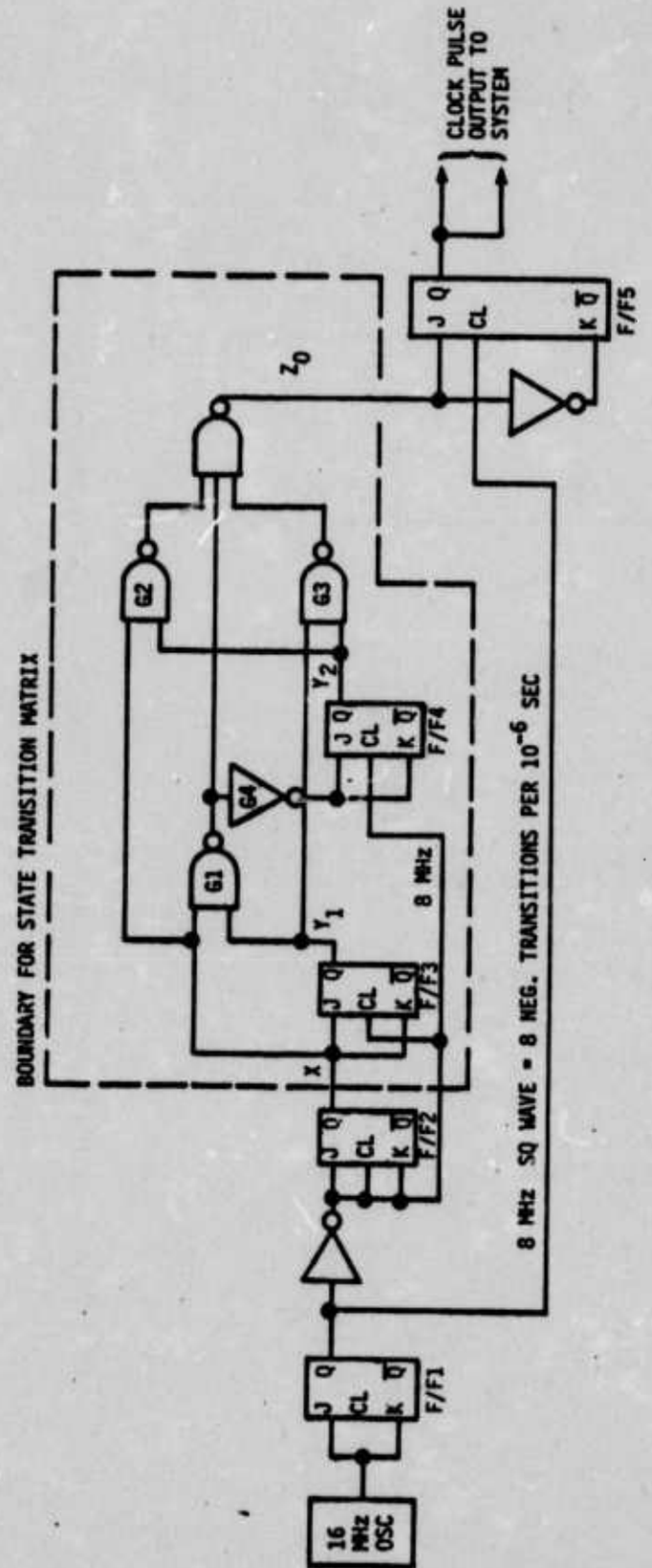


FIGURE A-2.–OSCILLATOR COUNTDOWN AND WAVE-SHAPING CIRCUIT

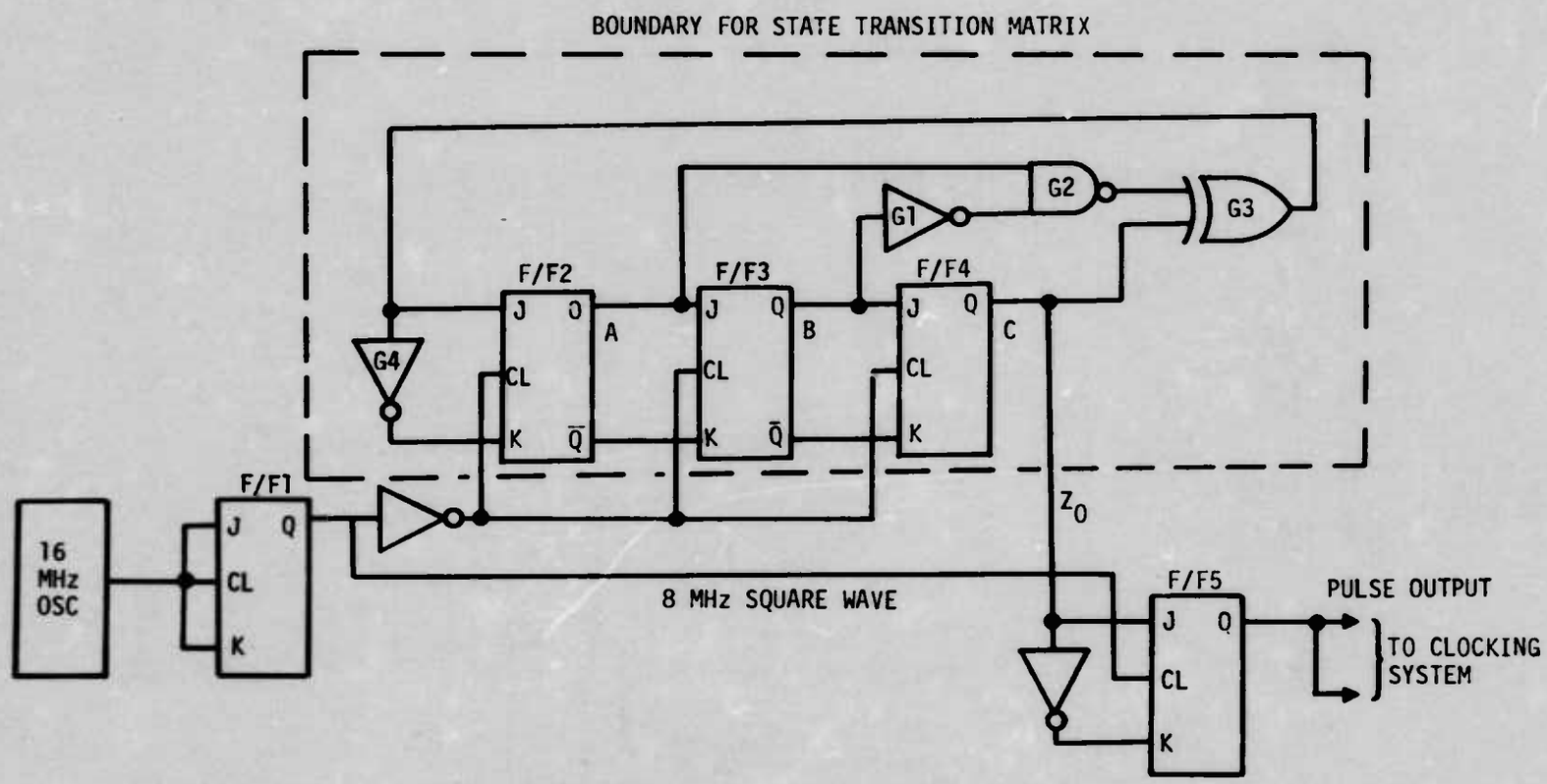


FIGURE A-3.–ALTERNATE OSCILLATOR COUNTDOWN AND WAVE-SHAPING CIRCUIT

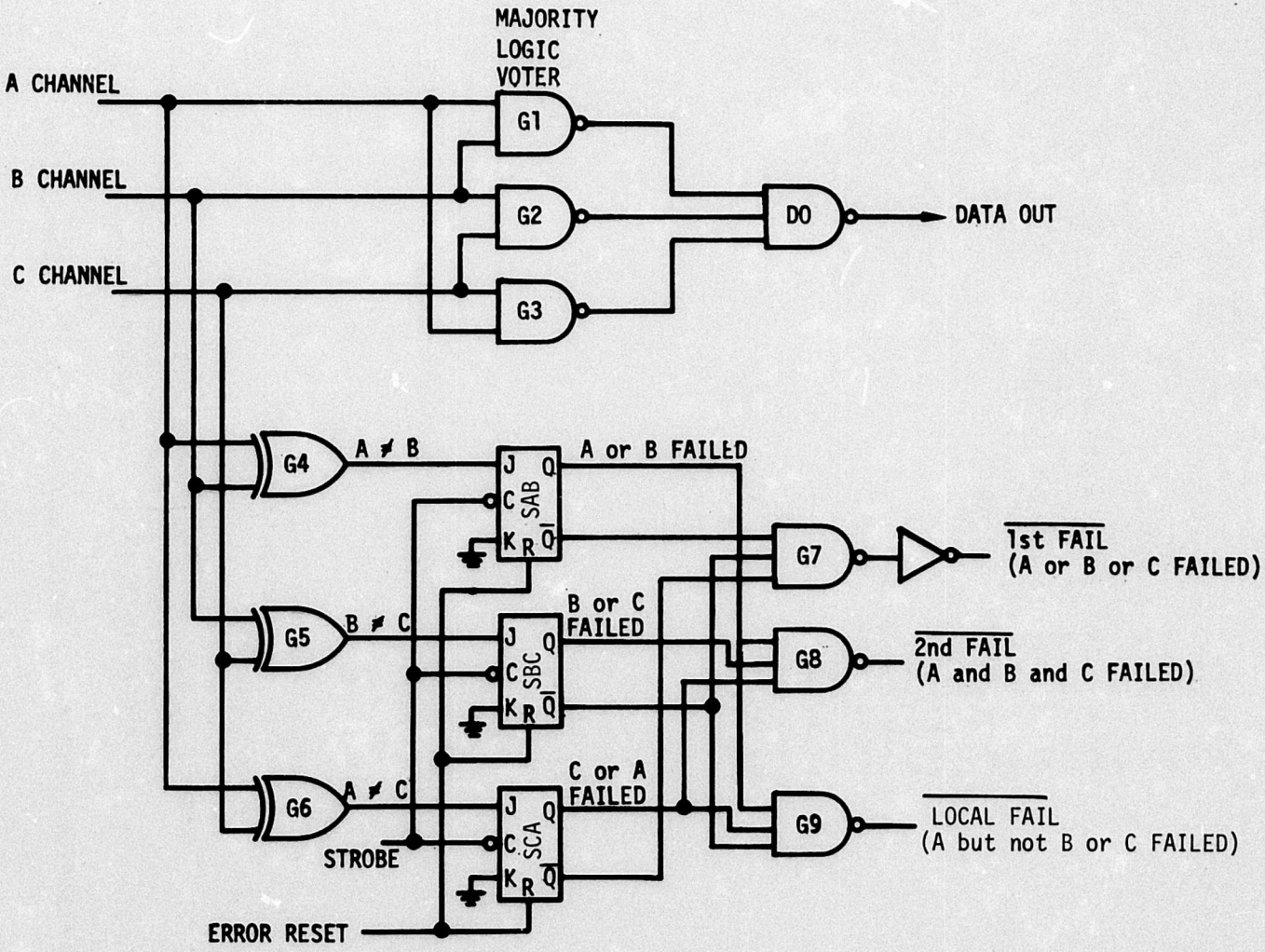


FIGURE A-4.–MLV AND FAILURE MONITOR

```
DELAY 10 N
G1   = (A BUF · B BUF)* Note 1
G2   = (B BUF · C BUF)*
G3   = (C BUF · A BUF)*
DO   = (G1 · G2 · G3)*
G4   = (A BUF · B BUF*) + (A BUF* · B BUF)
G5   = (B BUF · C BUF*) + (B BUF* · C BUF)
G6   = (C BUF · A BUF*) + (C BUF* · A BUF)
G7   = (SAB* · SBC* · SCA*) Note 2
G8   = (SAB · SBC · SBC · SCA)*
G9   = (SAB · SCA · SBC*)*
J SAB = G4
K SAB = GND
C SAB = CLCBUF
R SAB = RESETN
J SBC  = G5
K SAB = GND
C SBC = CLKBUF
R SAB = RESETN
J SBC  = G5
K SAB = GND
C SBC = CLKBUF
R SBC = RESETN
J SCA  = G6
K SCA = GND
C SCA = CLKBUF
R SCA = RESETN
FFAILN = G7*
SFAILN = G8*
LFAILN = G9  Note 3
```

The following equations were added for simulation purposes:

```
DELAY 1 N
A BUF  =  A
B BUF  =  B
C BUF  =  C
CLKBUF   =  STROBE
RESETN   =  ERROR RESET
GND = GND · GND*
```

NOTE: 1 The asterisk (*) is used to denote the logical complement.
2 SAB = A or B failed, SBC = B or C failed, SCA = C or A failed.
3 FFAILN = 1st fail, SFAILN = 2nd fail, LFAILN = local fail

FIGURE A-5.—MLV AND MONITOR EQUATIONS FOR LOGIC SIMULATION

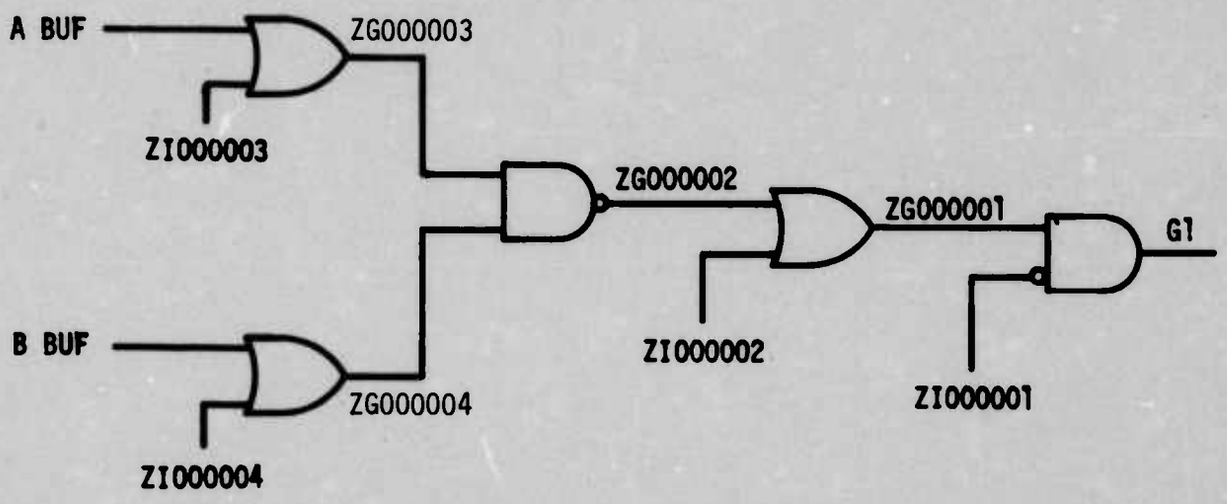
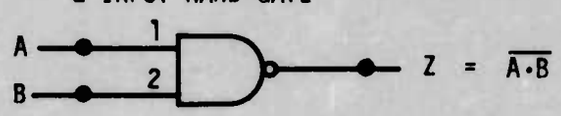
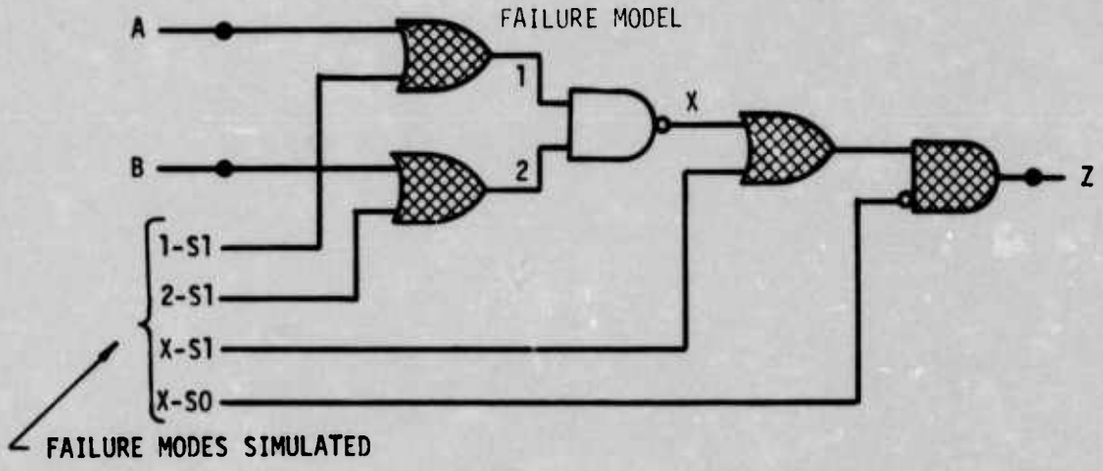


FIGURE A-6.—INPUT NAND-GATE FAULT MODEL

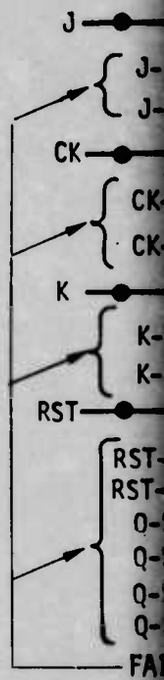
2 INPUT NAND GATE



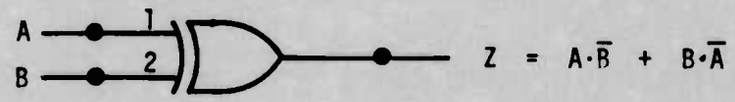
FAILURE MODEL



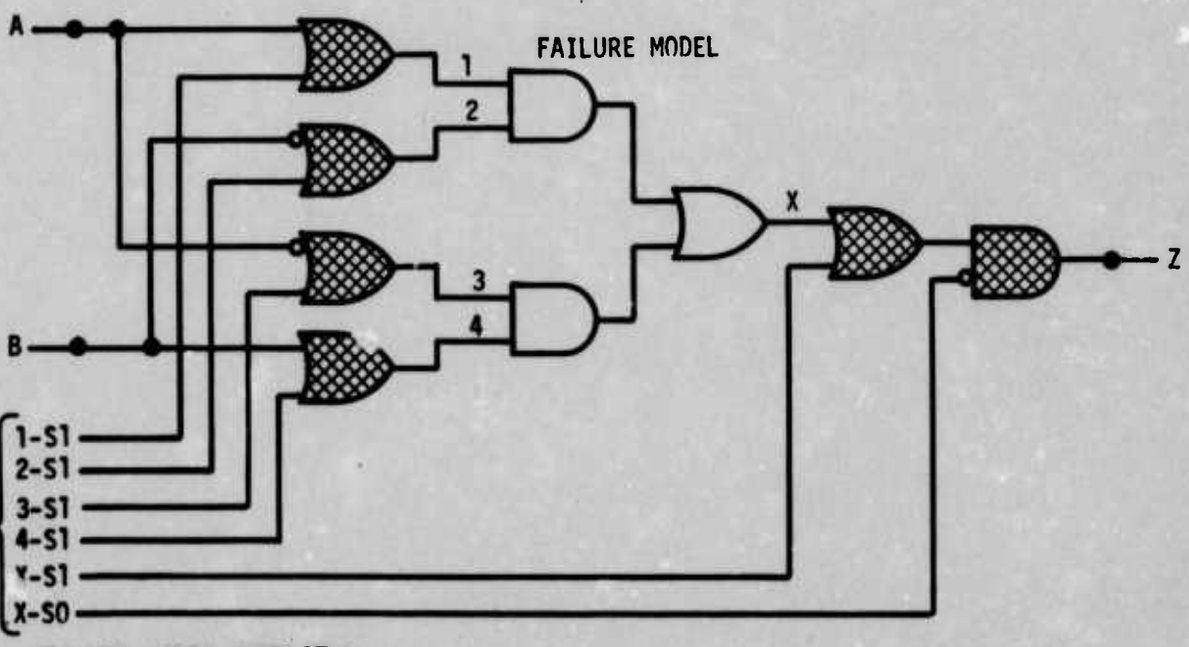
FAILURE MODES SIMULATED



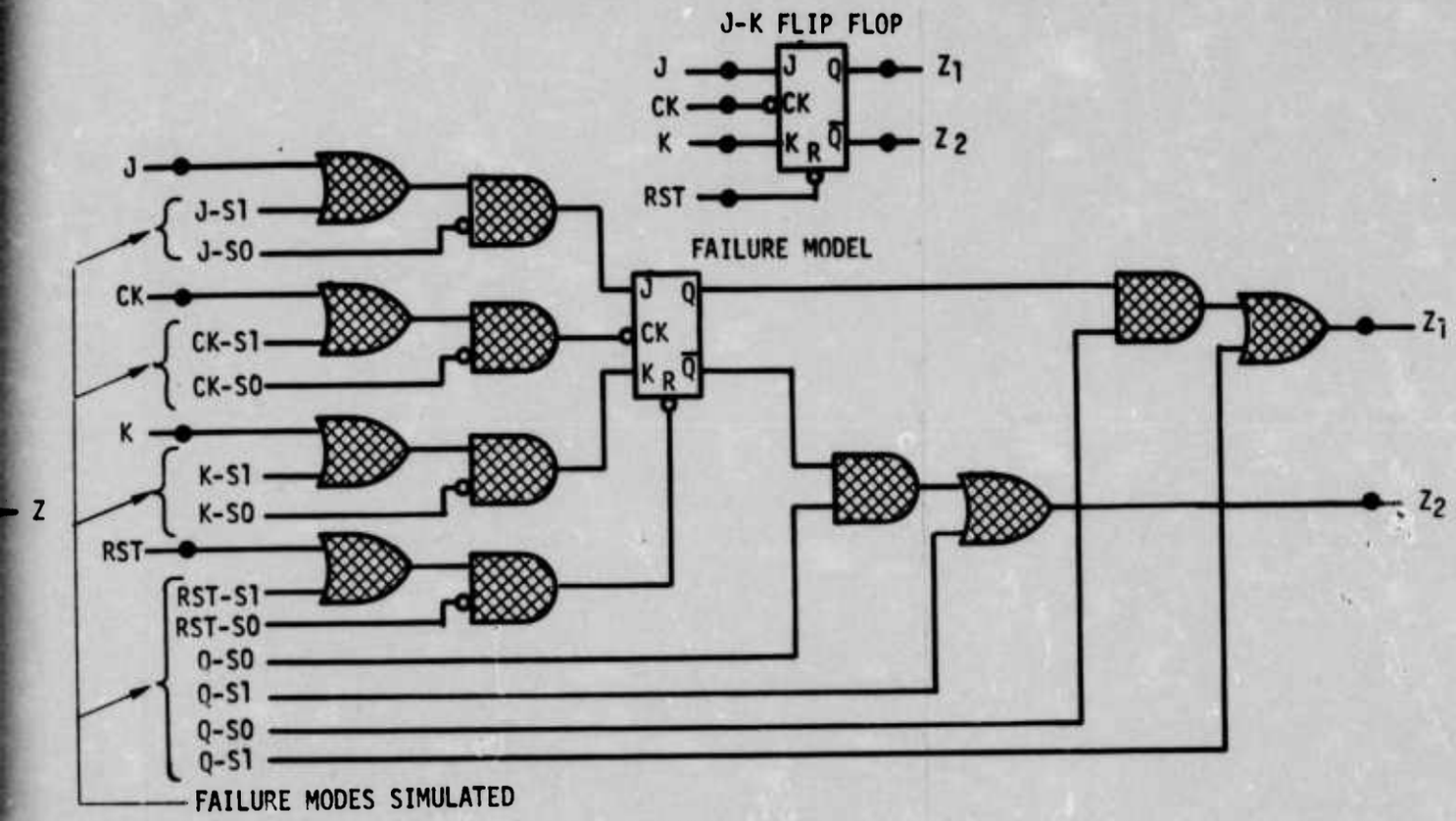
EXCLUSIVE OR GATE

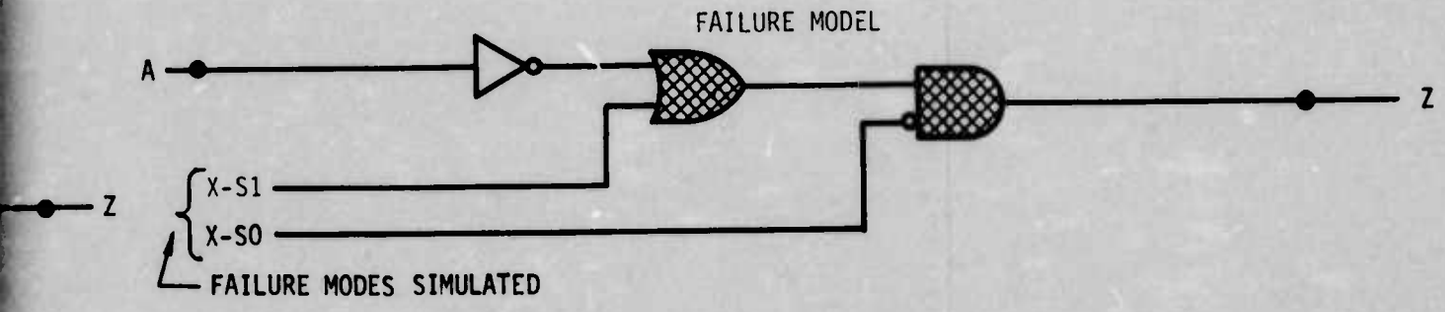


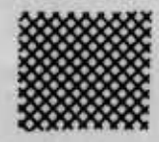
FAILURE MODEL



FAILURE MODES SIMULATED







SIMULATOR GENERATED ELEMENT

FIGURE A-7.—FAULT MODELS FOR DEVICES USED IN MLV AND MONITOR CIRCUIT

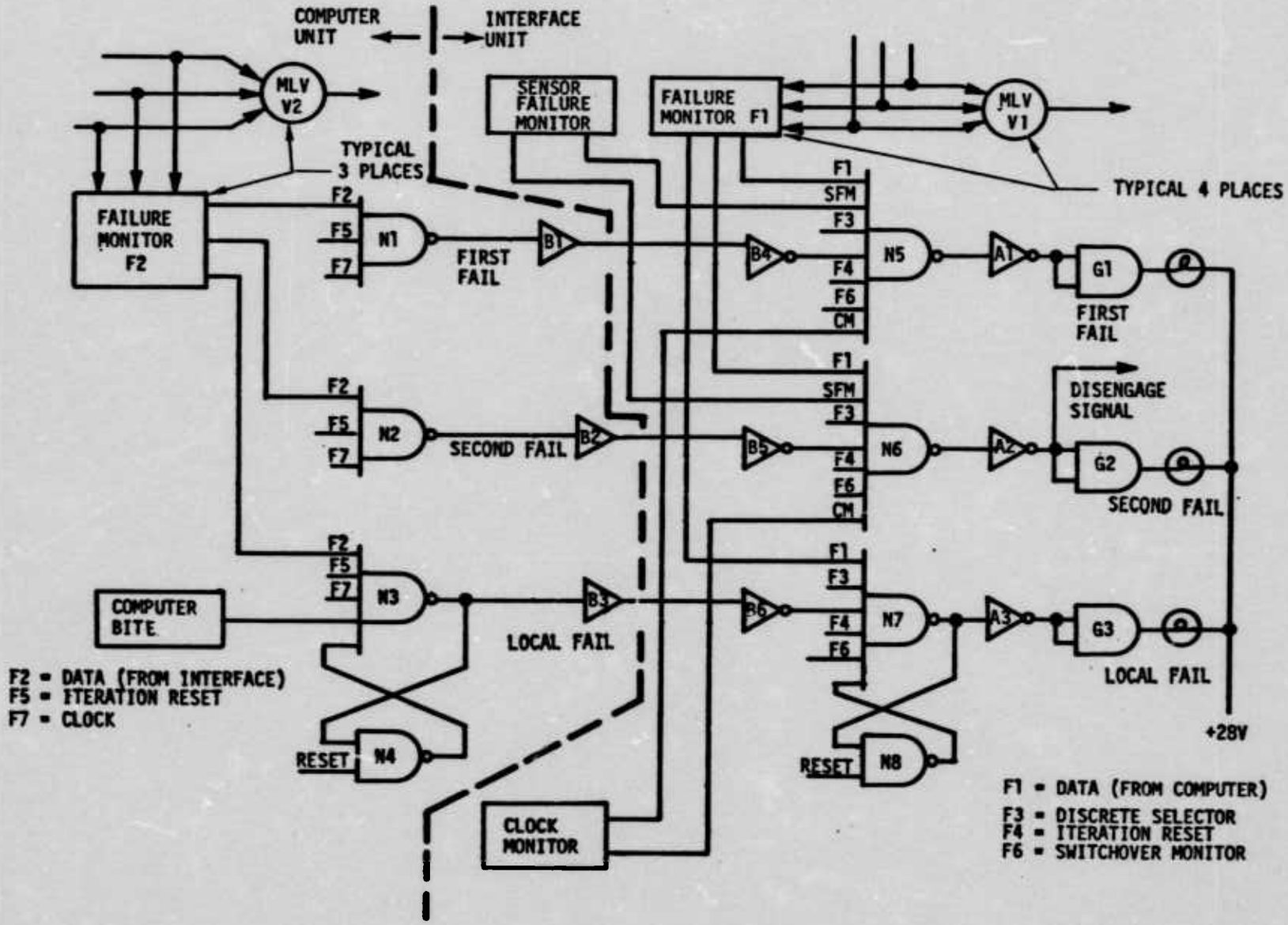
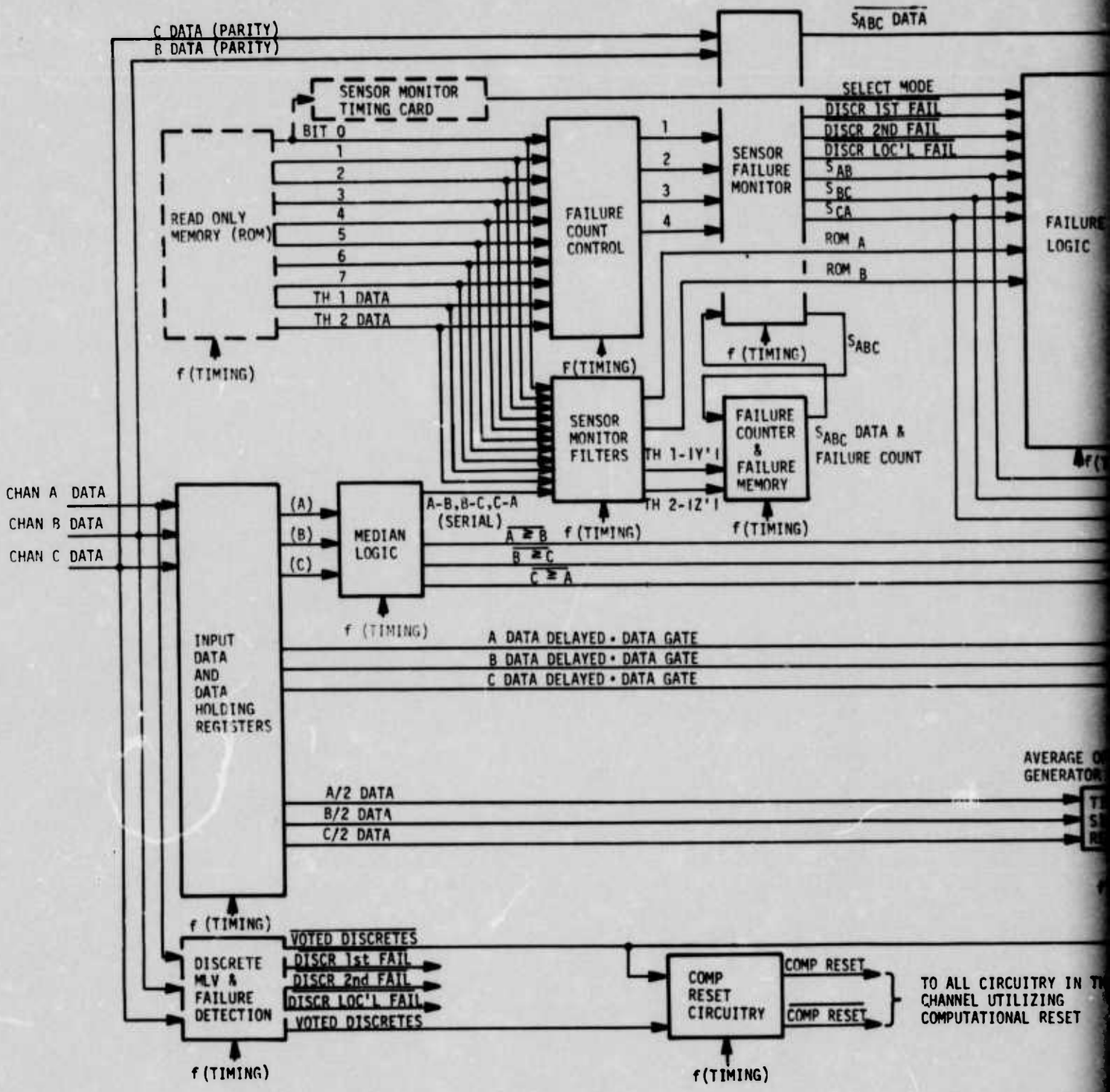


FIGURE A-8.—SYSTEM STATUS LOGIC



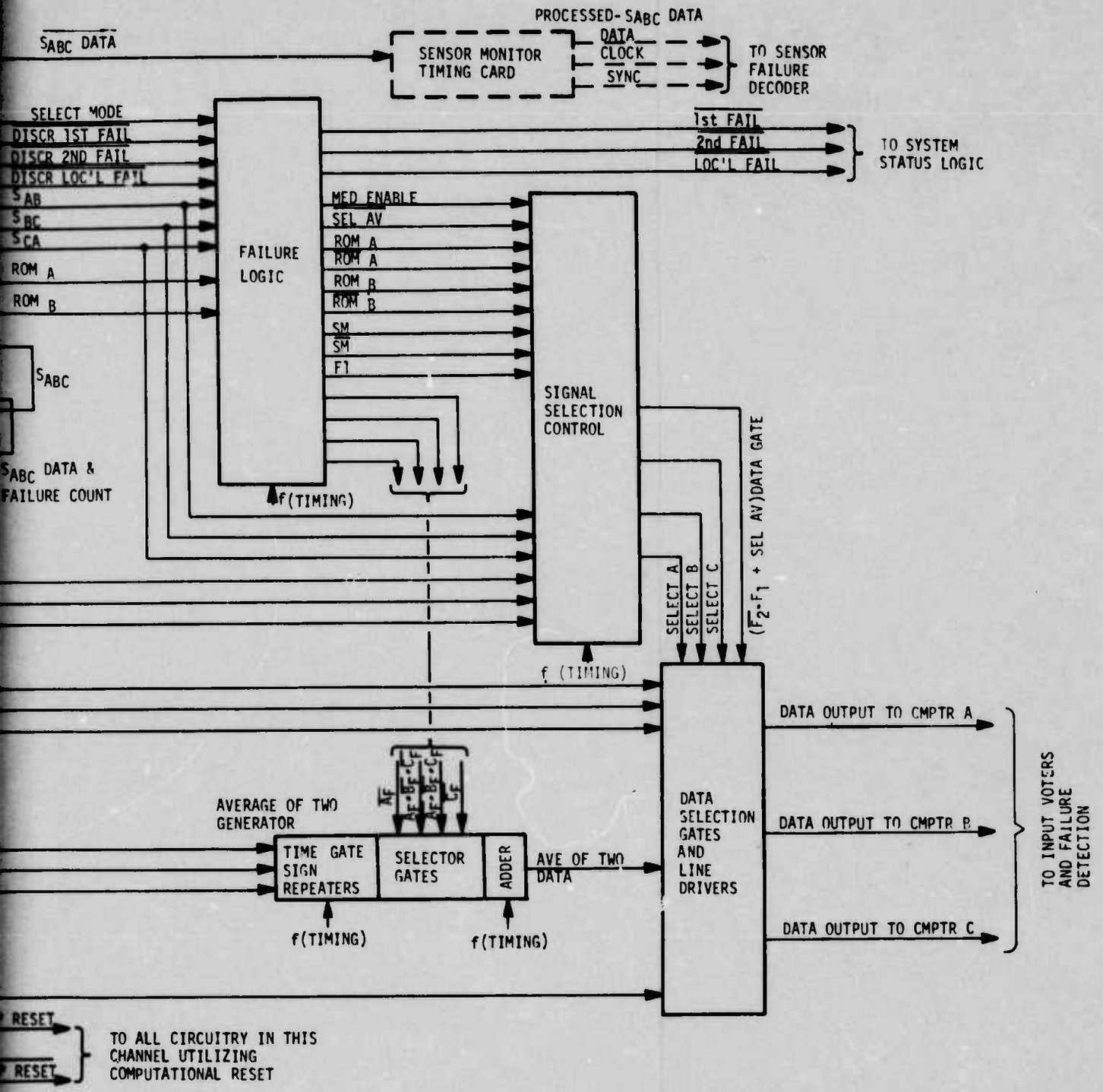
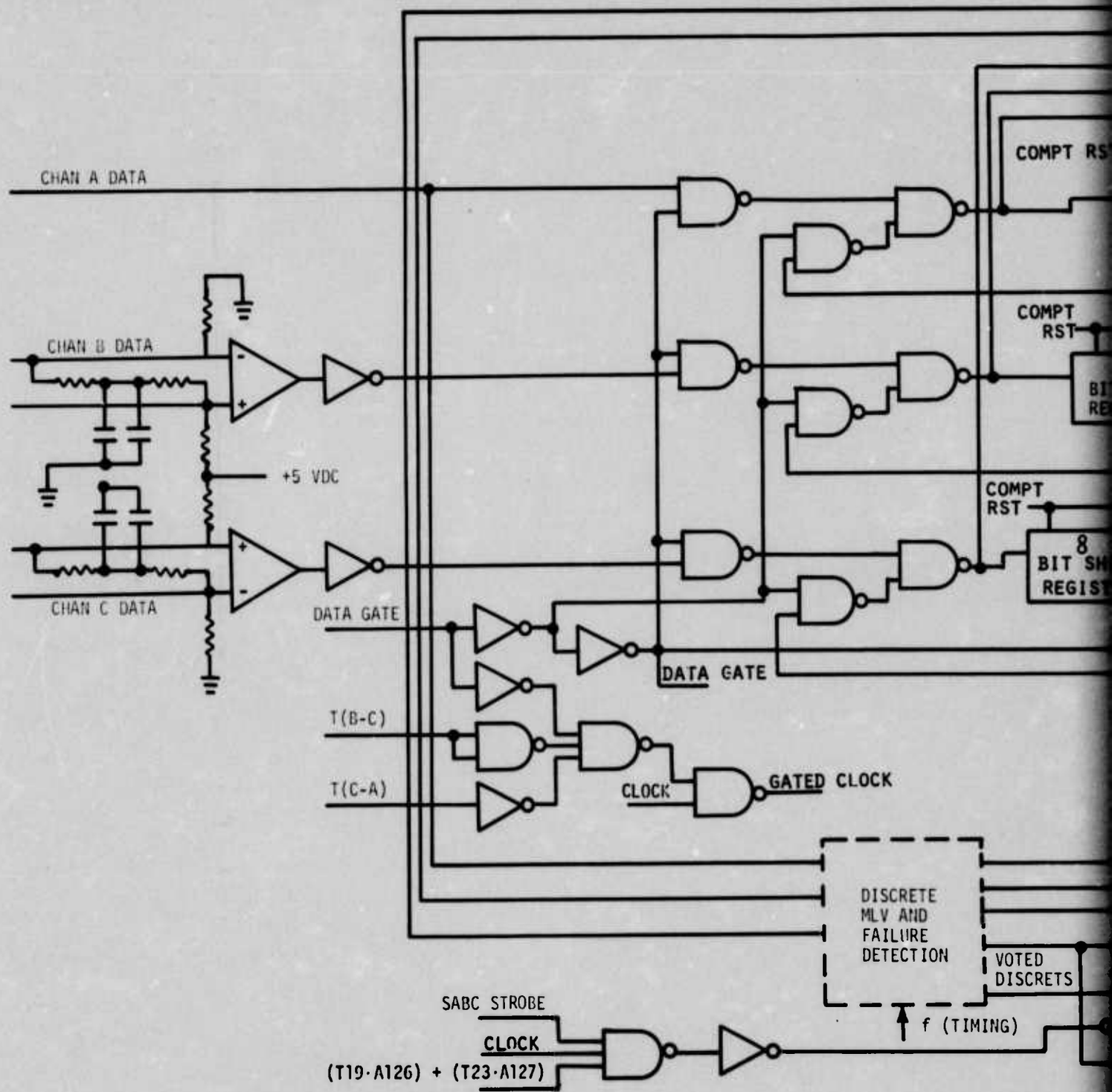


FIGURE A-9.—SENSOR SELECTION AND FAILURE MONITORING FUNCTIONAL BLOCK DIAGRAM



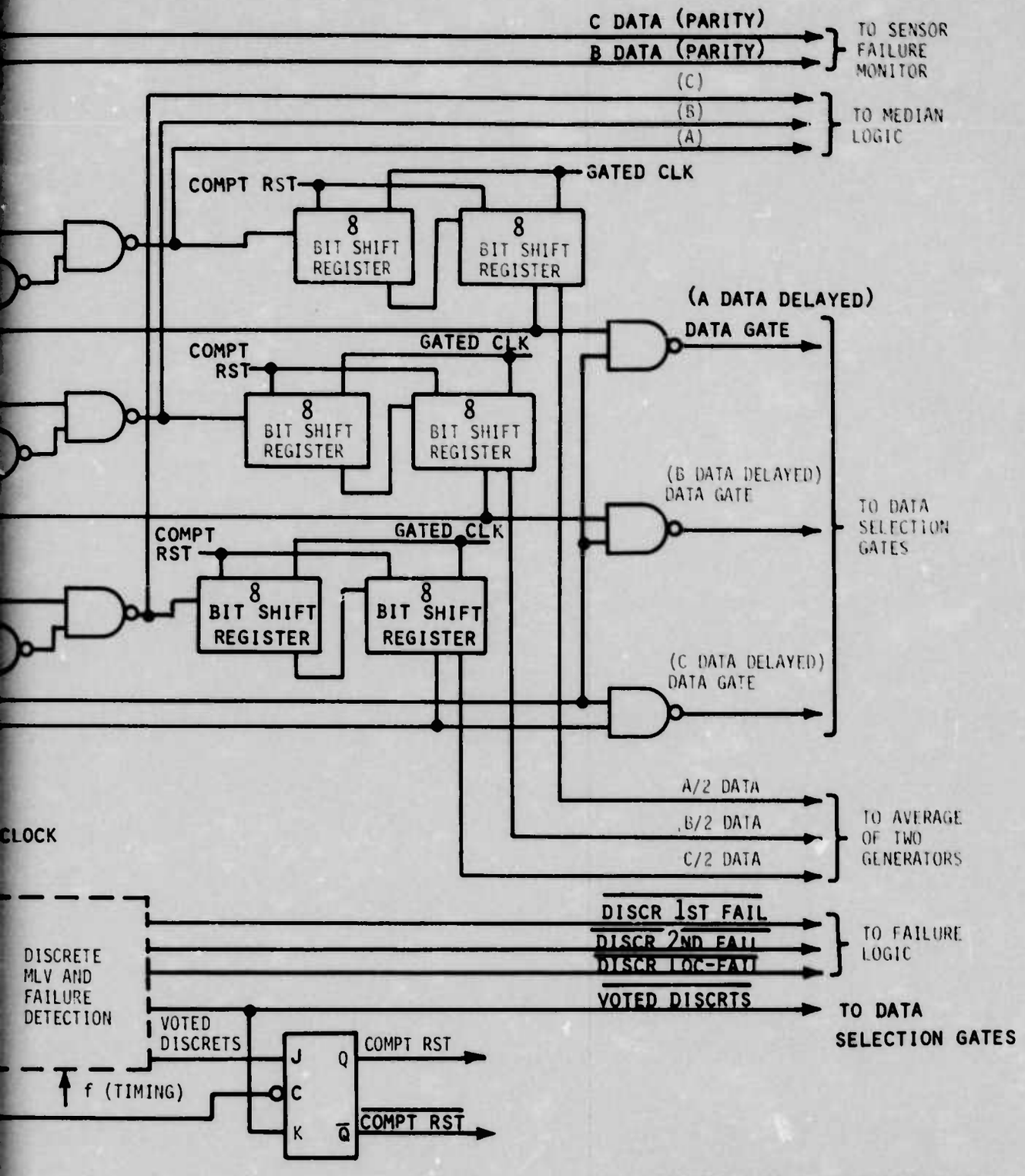
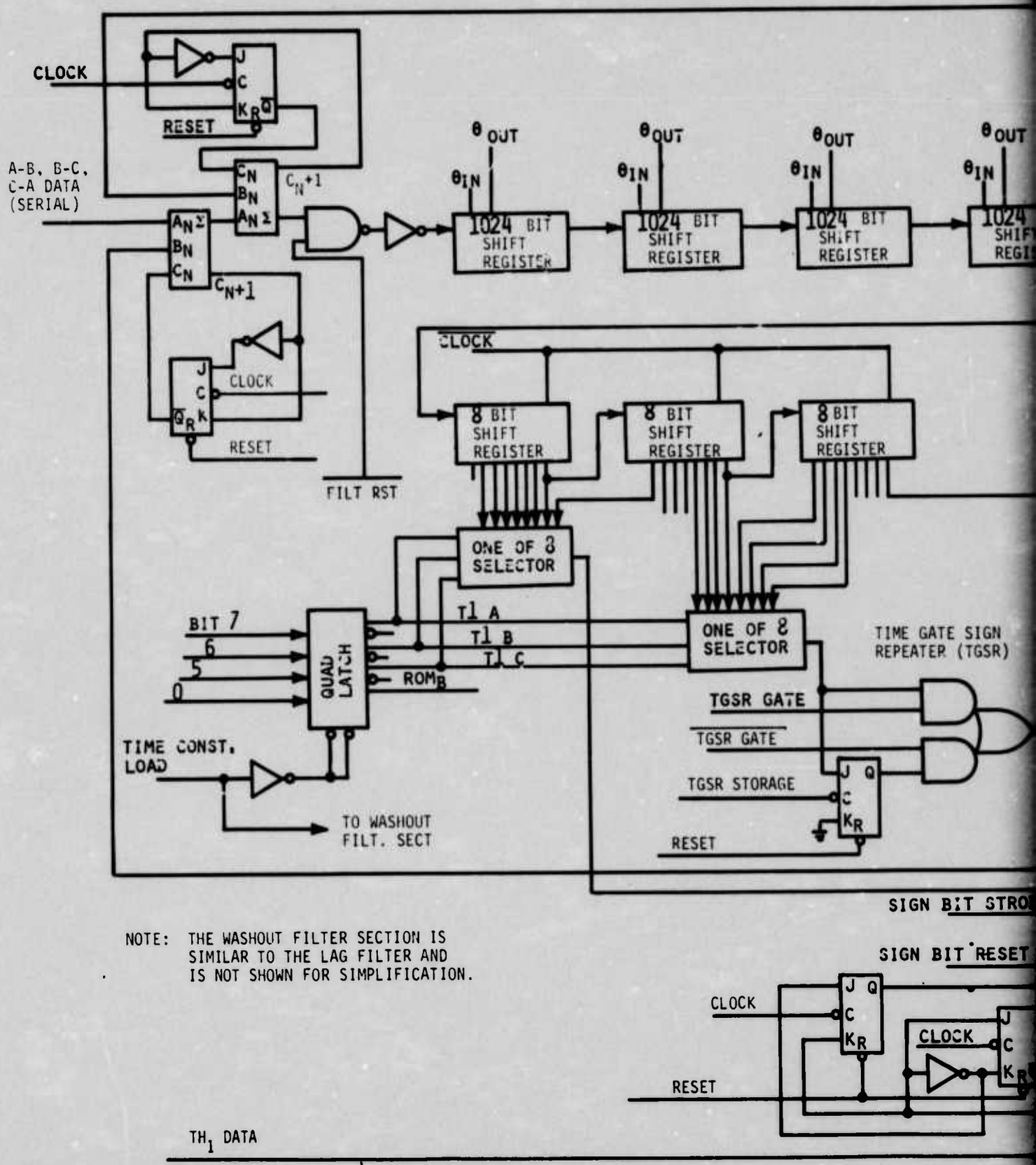


FIGURE A-10.—INPUT AND DATA HOLDING REGISTERS



NOTE: THE WASHOUT FILTER SECTION IS SIMILAR TO THE LAG FILTER AND IS NOT SHOWN FOR SIMPLIFICATION.

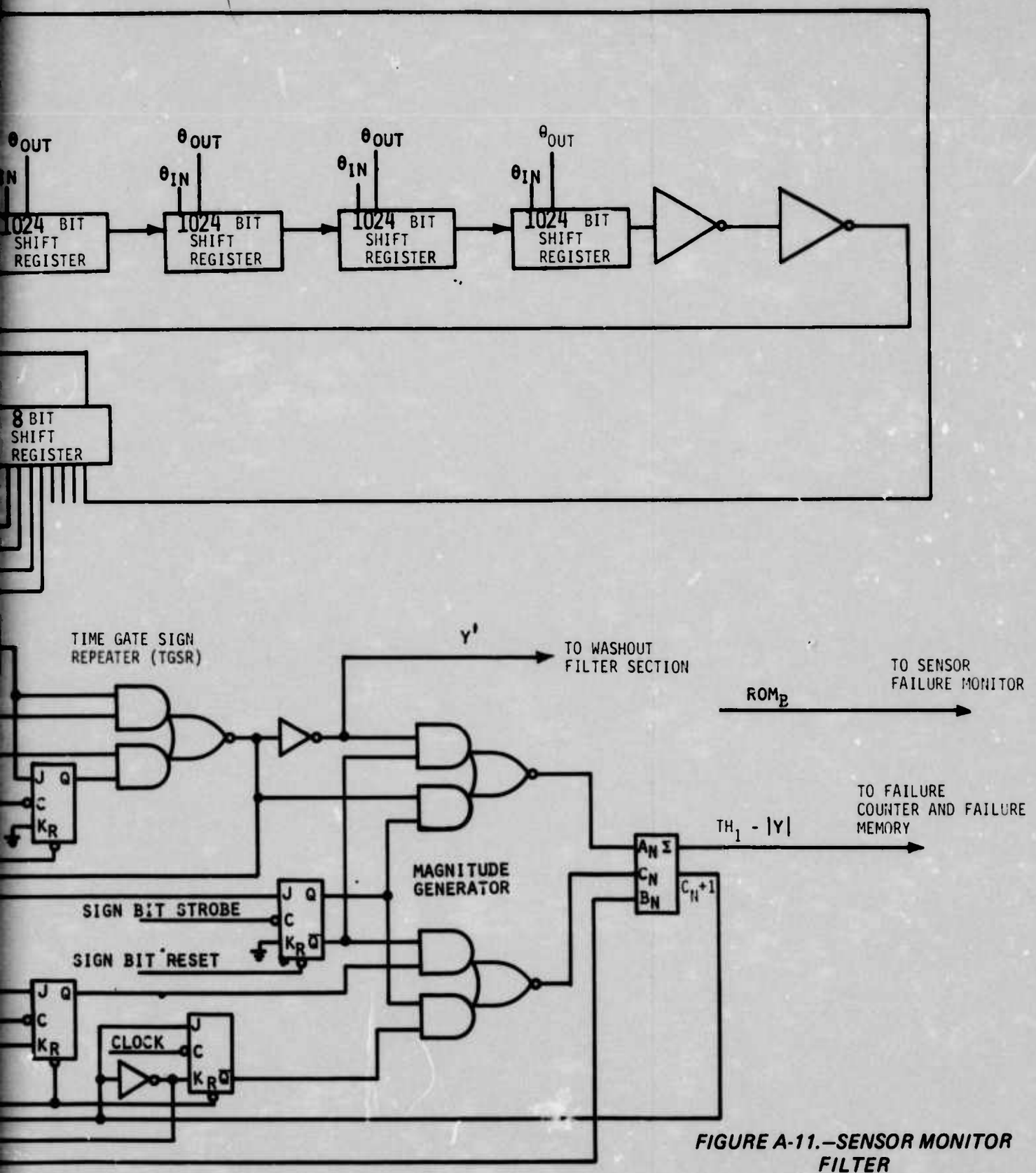


FIGURE A-11.—SENSOR MONITOR FILTER

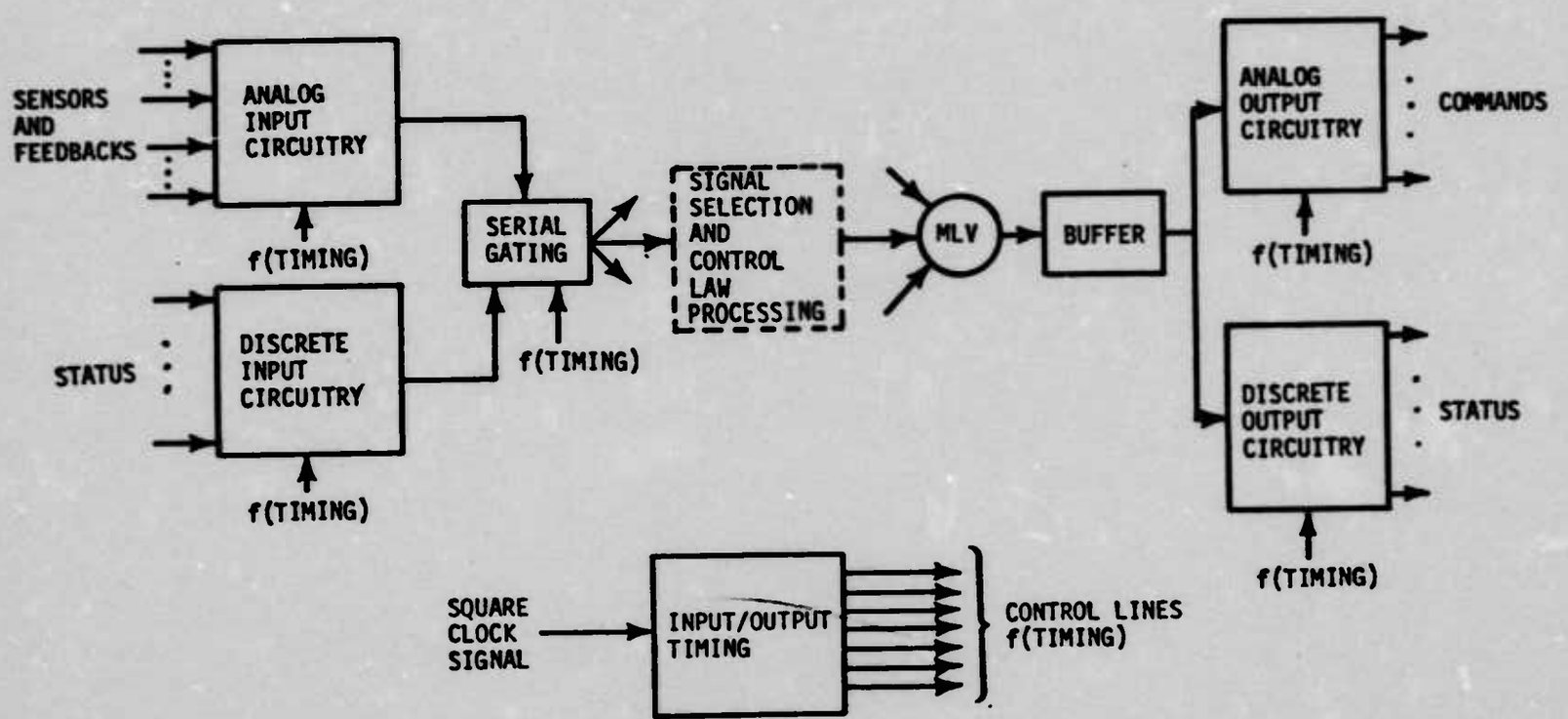
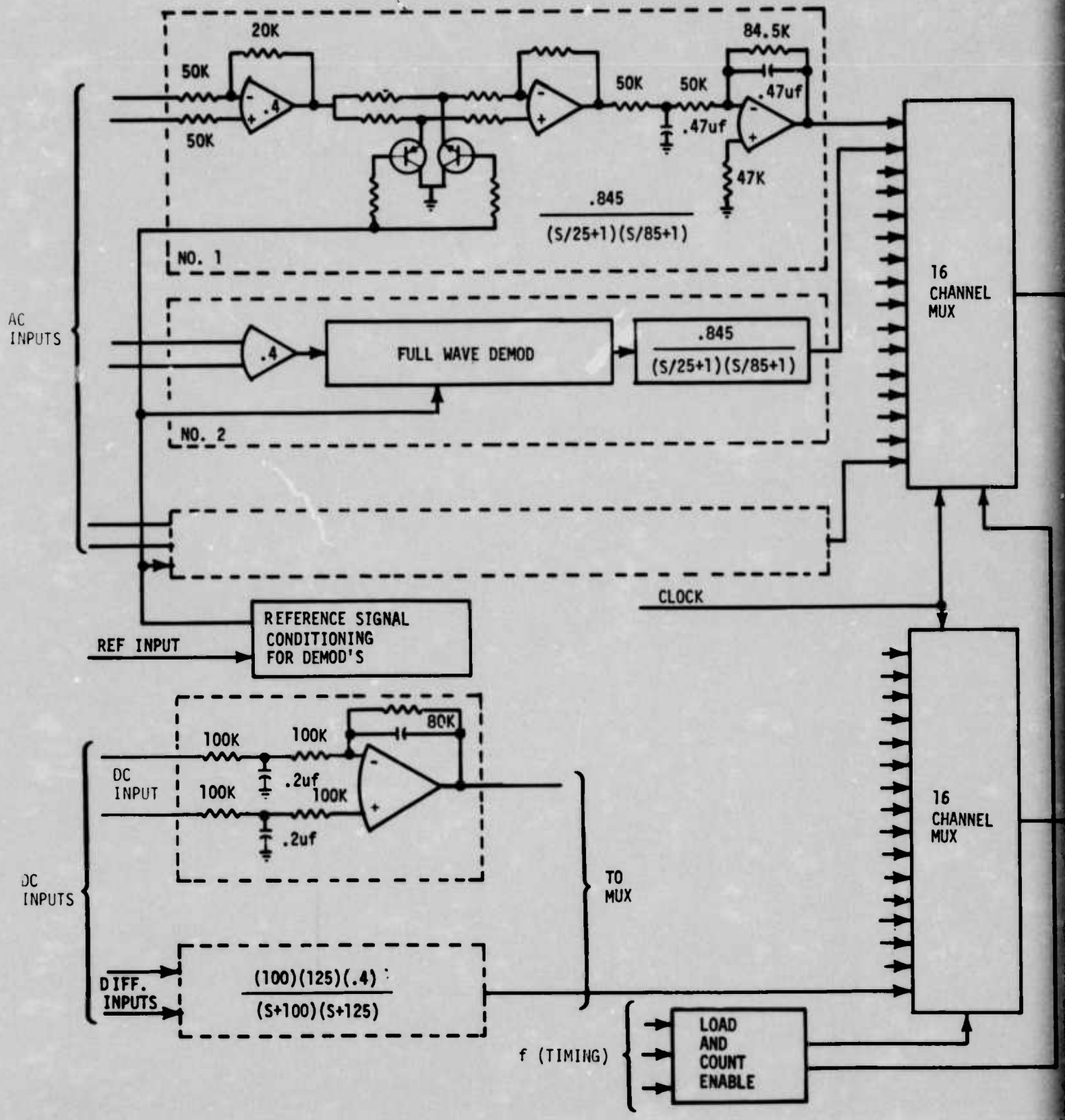


FIGURE A-12.—INPUT/OUTPUT CIRCUITRY AND TIMING


20K
50K
50K
.4
50K
50K
84.5K
.47uf
.47uf
47K
.845
(S/25+1)(S/85+1)
NO. 1
AC
INPUTS
.4
FULL WAVE DEMOD
.845
(S/25+1)(S/85+1)
NO. 2
16
CHANNEL
MUX
CLOCK
REF INPUT
REFERENCE SIGNAL
CONDITIONING
FOR DEMOD'S
80K
100K
100K
DC
INPUT
100K
.2uf
100K
.2uf
DC
INPUTS
TO
MUX
16
CHANNEL
MUX
DIFF.
INPUTS
(100)(125)(.4)
(S+100)(S+125)
f (TIMING)
LOAD
AND
COUNT
ENABLE

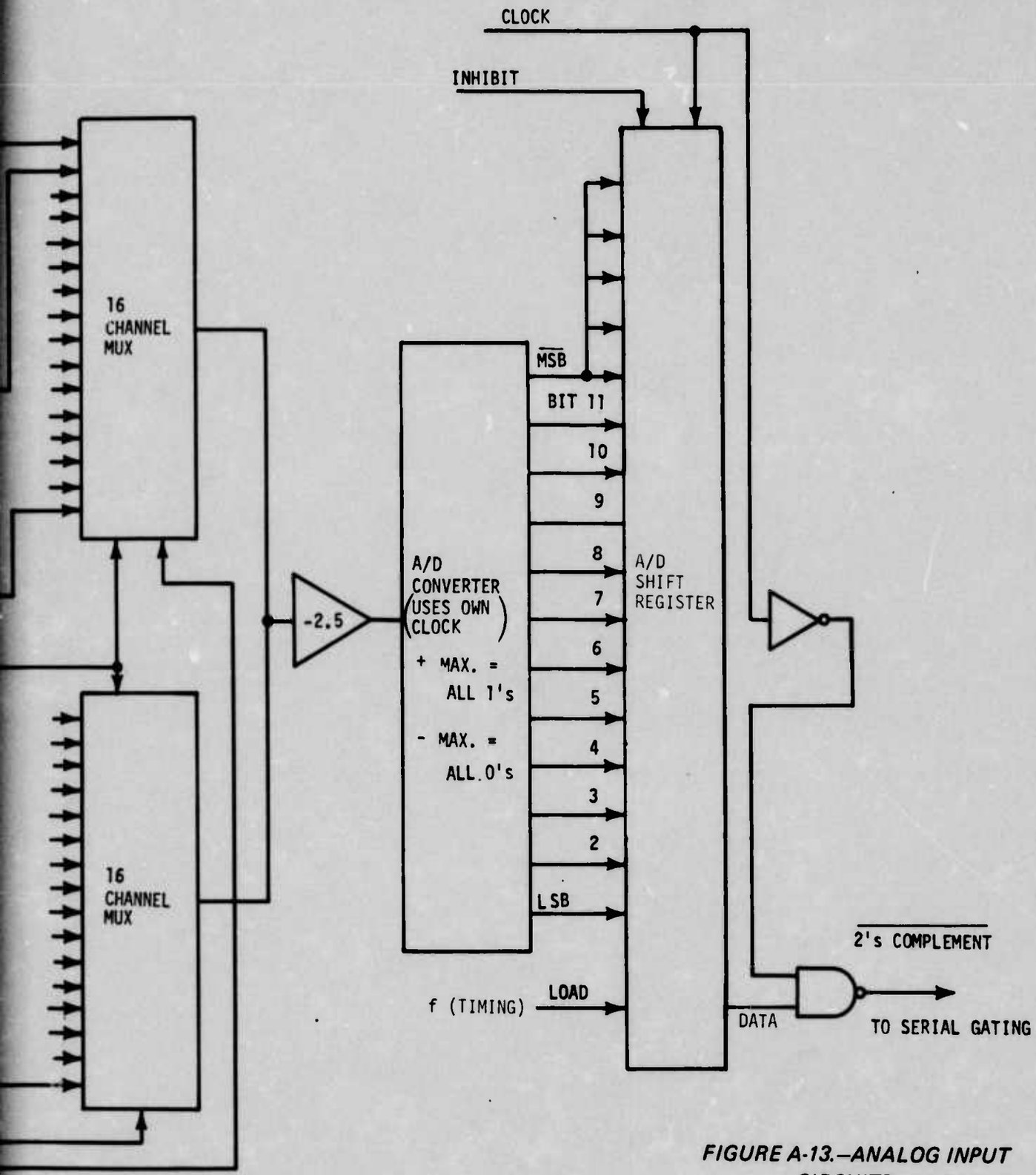


FIGURE A-13.–ANALOG INPUT CIRCUITS

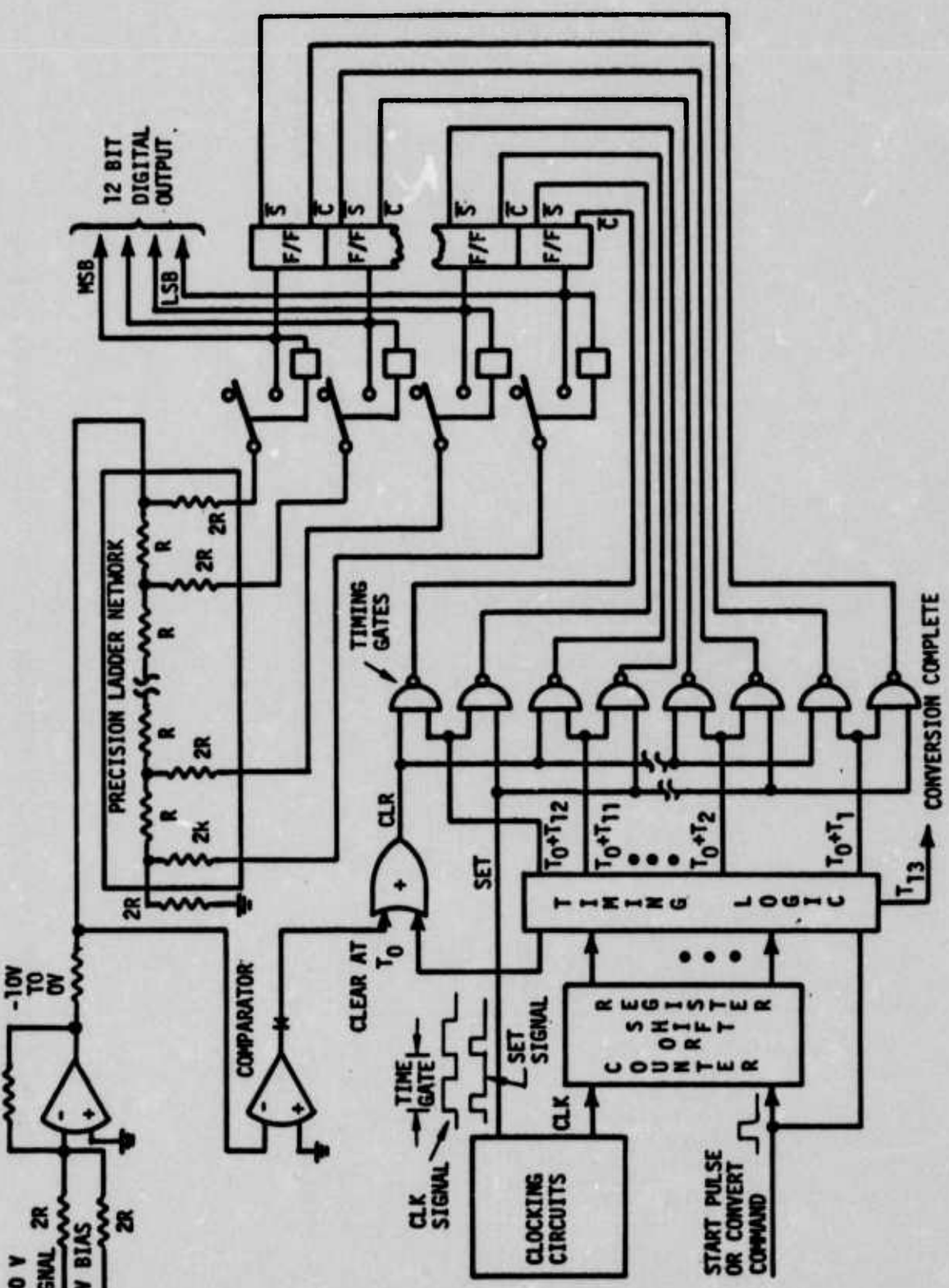
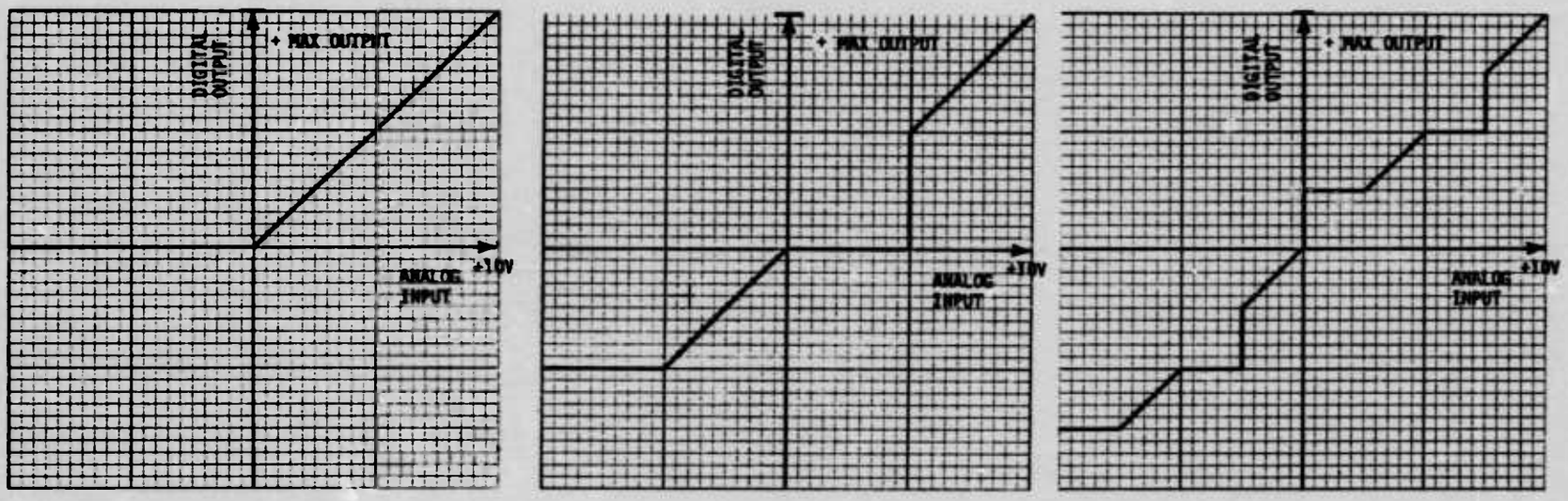


FIGURE A-14.–A/D CONVERTER

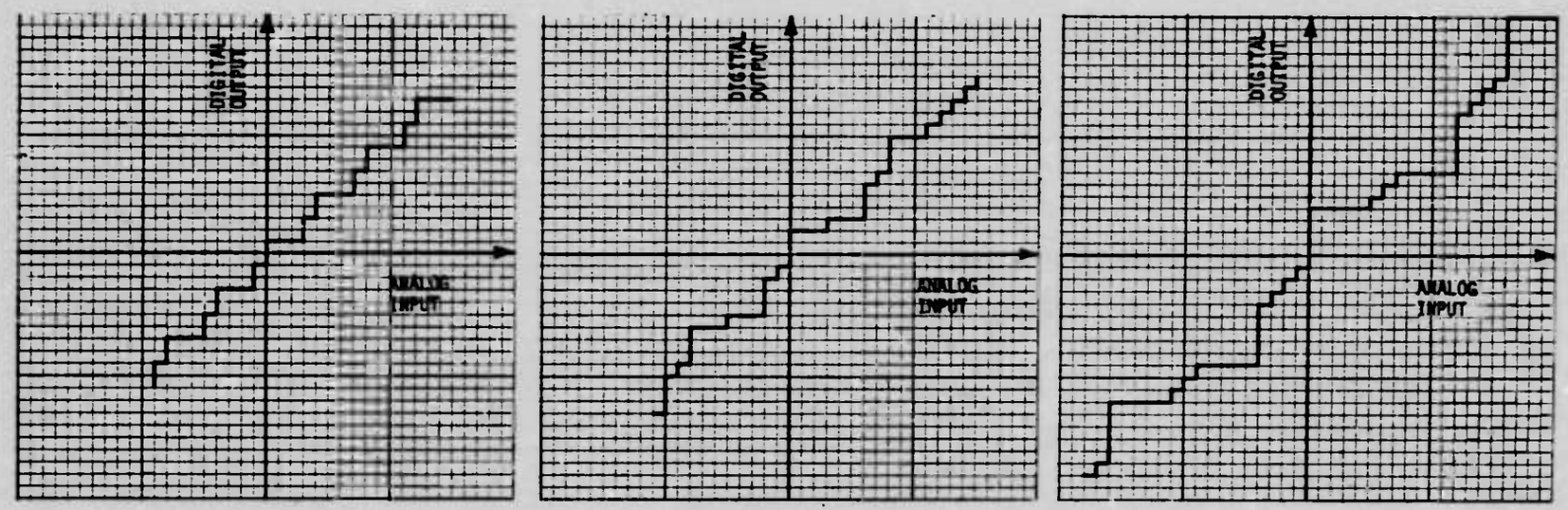


(a) EFFECTS OF FAILING BIT 12 HIGH

(b) EFFECTS OF FAILING BIT 11 HIGH

(c) EFFECTS OF FAILING BIT 10 HIGH

FIGURE A-15.—EFFECTS OF FAILED HIGH SIGNIFICANT BITS IN AN A/D CONVERTER



(a) EFFECTS OF FAILING BIT 1 (LSB) HIGH

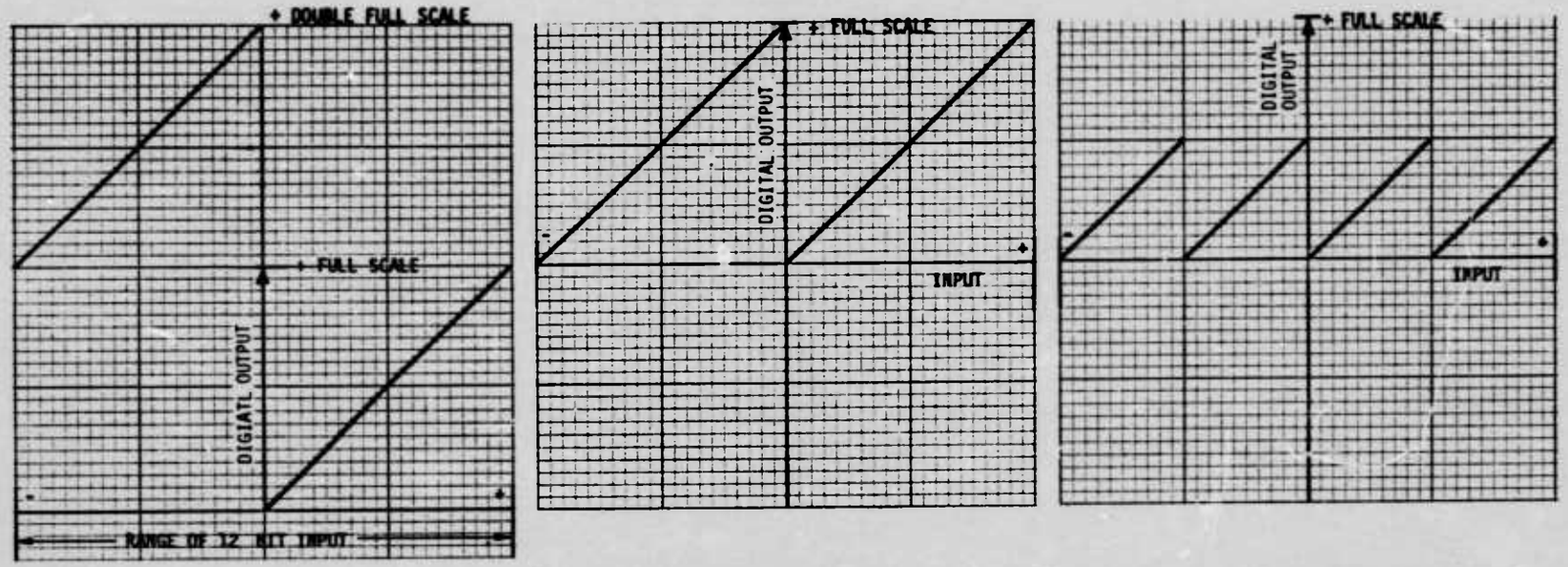
(b) EFFECTS OF FAILING BIT 2 HIGH

(c) EFFECTS OF FAILING BIT 3 HIGH

FIGURE A-16.—EFFECTS OF FAILING FIRST THREE BITS IN AN A/D CONVERTER

BIT	12	11	10	9	8	7	6	5	4	3	2	1
BIT SIGNIFICANCE	SIGN	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512	1/1024	1/2048

FIGURE A-17.—FRACTIONS OF FULL-SCALE OUTPUT IN 12-BIT A/D CONVERSION



(a) EFFECT OF BIT 13 FAILED TO ZERO

(b) EFFECT OF BIT 12 FAILED TO ZERO

(c) EFFECT OF BIT 11 FAILED TO ZERO

FIGURE A-18.—ANALOG TO DIGITAL REGISTER FAILURES THAT CAUSE ALL UPSTREAM BITS TO SHIFT OUT AS ZERO

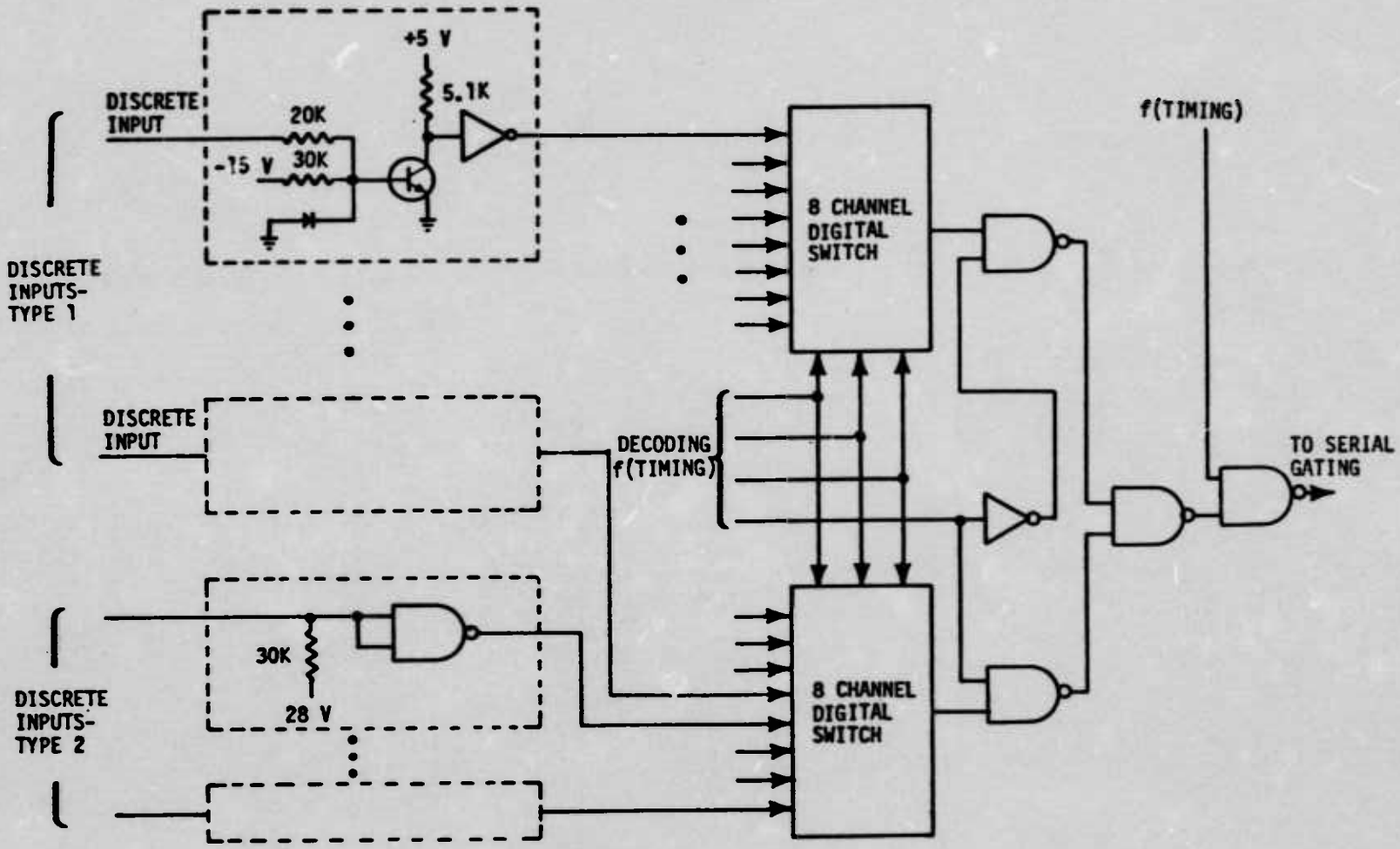


FIGURE A-19.–DISCRETE INPUT CIRCUITRY

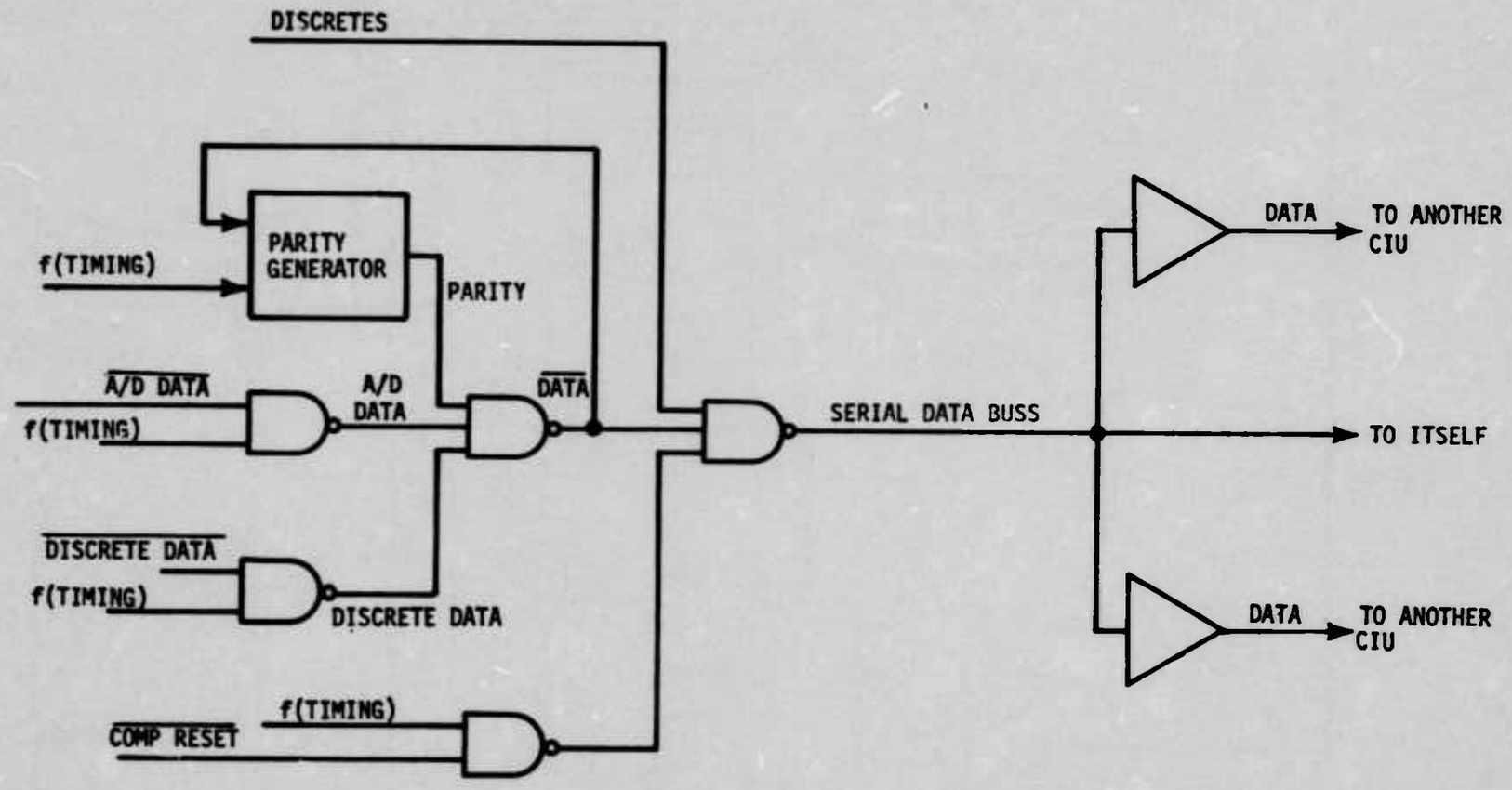


FIGURE A-20.–SERIAL GATING CIRCUITS

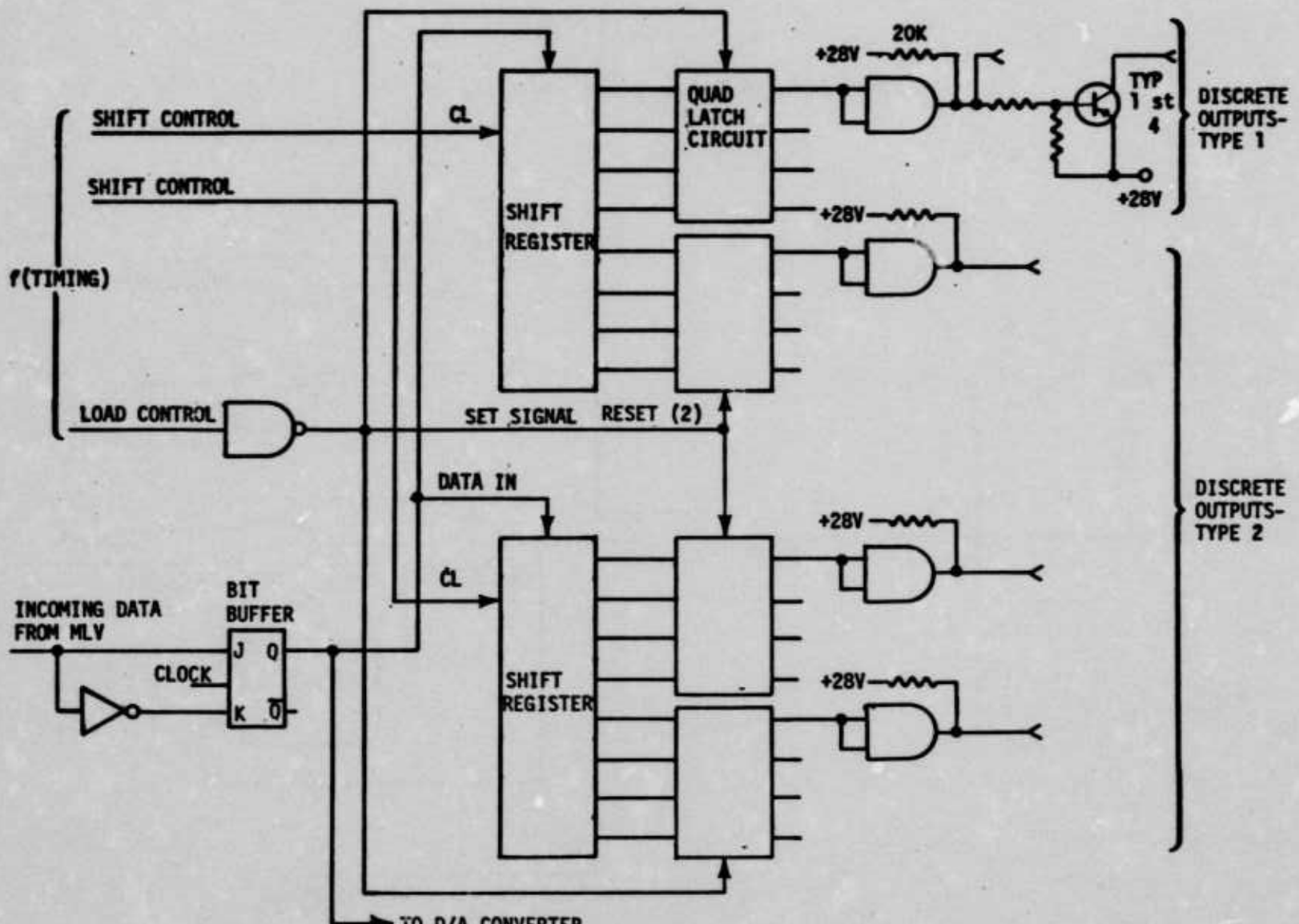
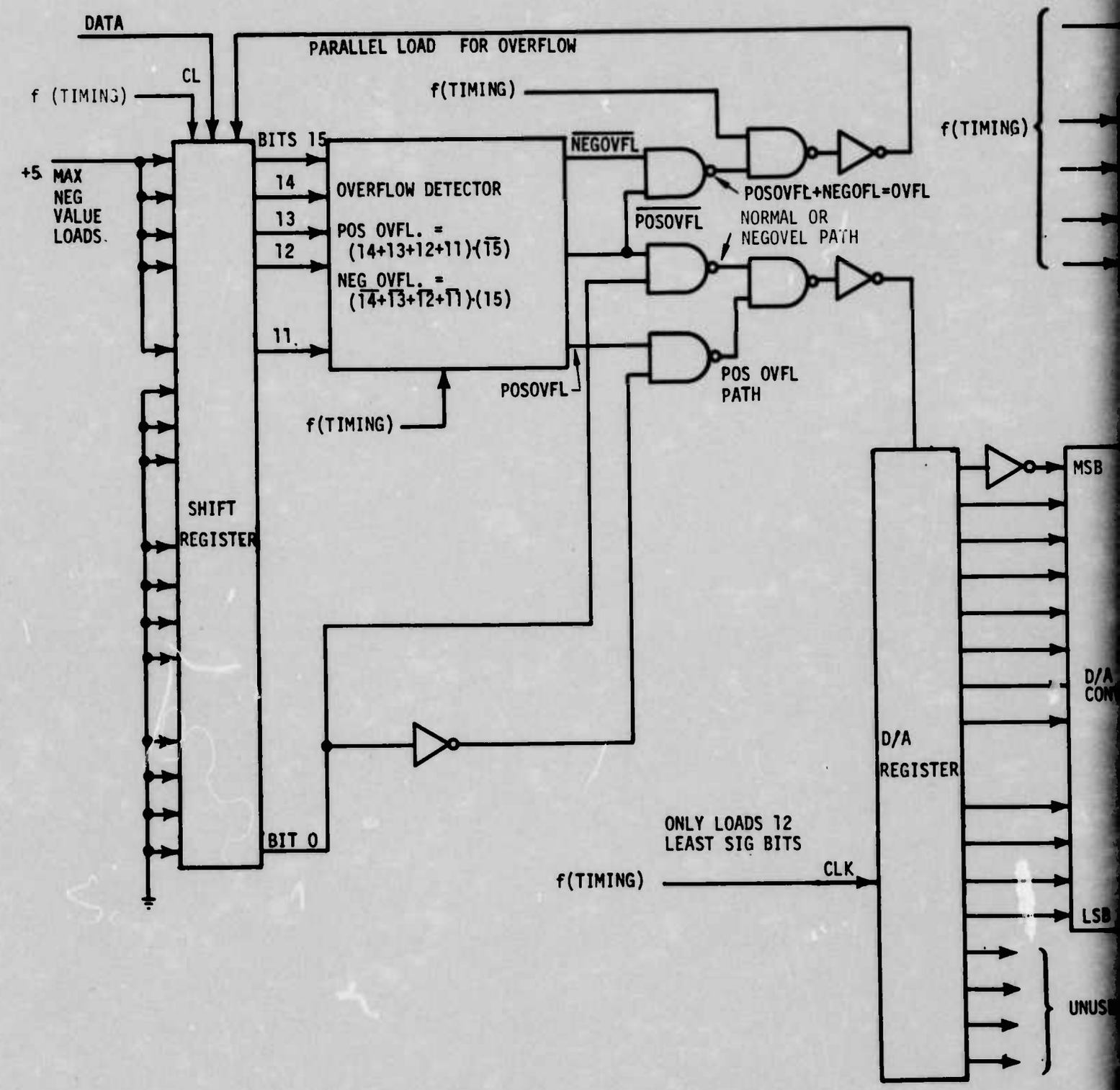


FIGURE A-21.–BUFFER AND DISCRETE OUTPUT CIRCUITS


DATA
PARALLEL LOAD FOR OVERFLOW
CL
f (TIMING)
f(TIMING)
f(TIMING)
BITS 15
14
13
12
11
+5
MAX
NEG
VALUE
LOADS
OVERFLOW DETECTOR
POS OVFL. =
(14+13+12+11)·(15)
NEG OVFL. =
(14+13+12+11)·(15)
NEGOVFL
POSOVFL+NEGOFL=OVFL
POSOVFL
NORMAL OR
NEGOVEL PATH
POS OVFL
PATH
POSOVFL
f(TIMING)
SHIFT
REGISTER
BIT 0
MSB
D/A
CON
D/A
REGISTER
ONLY LOADS 12
LEAST SIG BITS
f(TIMING)
CLK
LSB
UNUSE

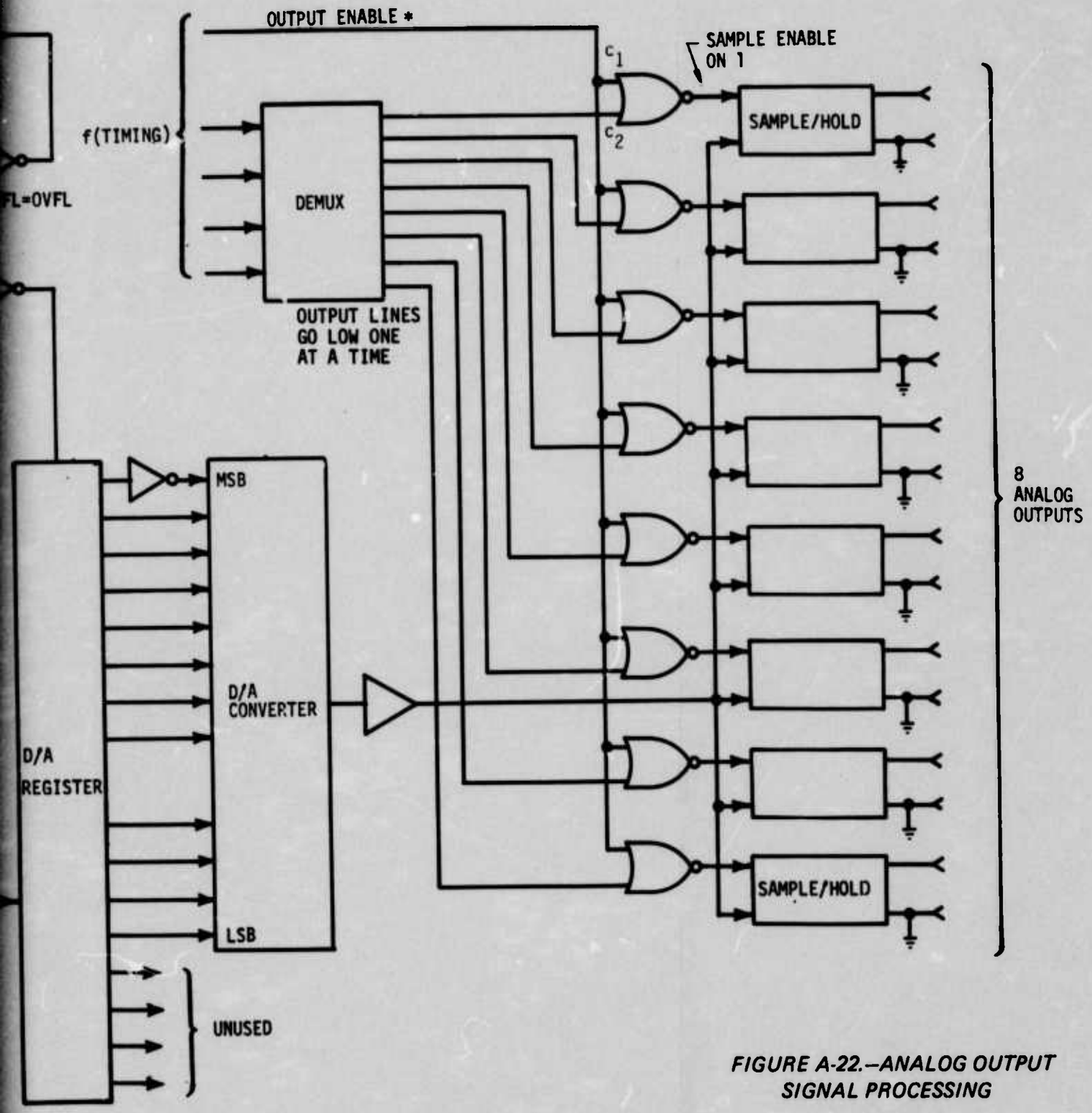


FIGURE A-22.—ANALOG OUTPUT SIGNAL PROCESSING

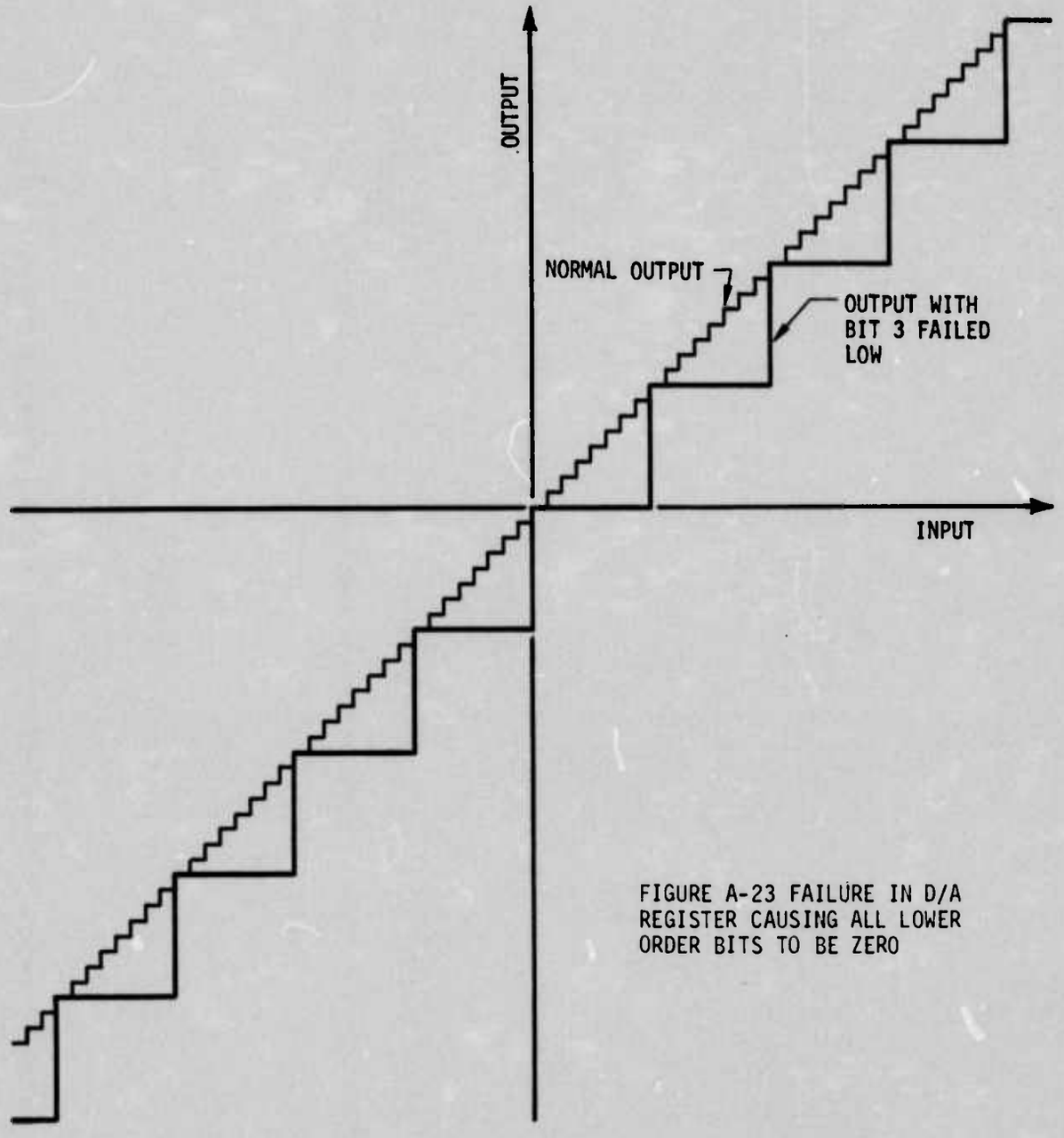
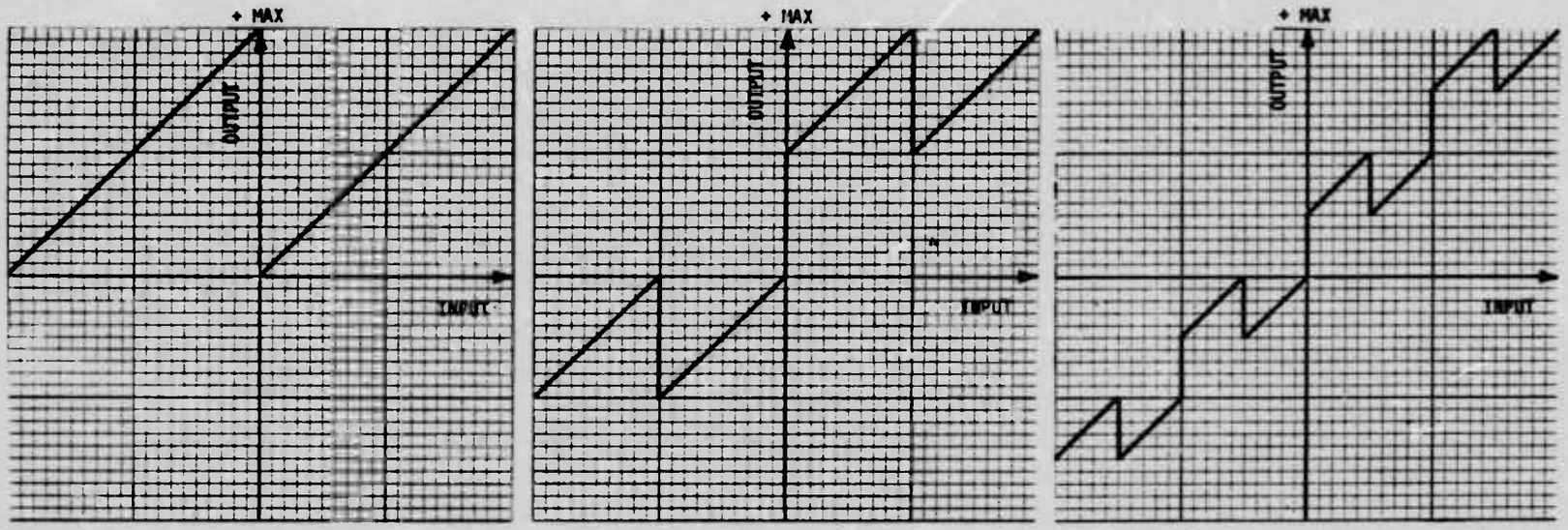


FIGURE A-23 FAILURE IN D/A
REGISTER CAUSING ALL LOWER
ORDER BITS TO BE ZERO

FIGURE A-23.–FAILURE IN D/A REGISTER CAUSING ALL LOWER ORDER BITS TO BE ZERO



(a) EFFECT OF D/A REGISTER BIT 12 (MSB) FAILED HIGH (b) EFFECT OF D/A REGISTER BIT 11 FAILED HIGH (c) EFFECT OF D/A REGISTER BIT 10 FAILED HIGH

FIGURE A-24.—EFFECTS OF FAILING INPUT BITS ON THE D/A CONVERTER

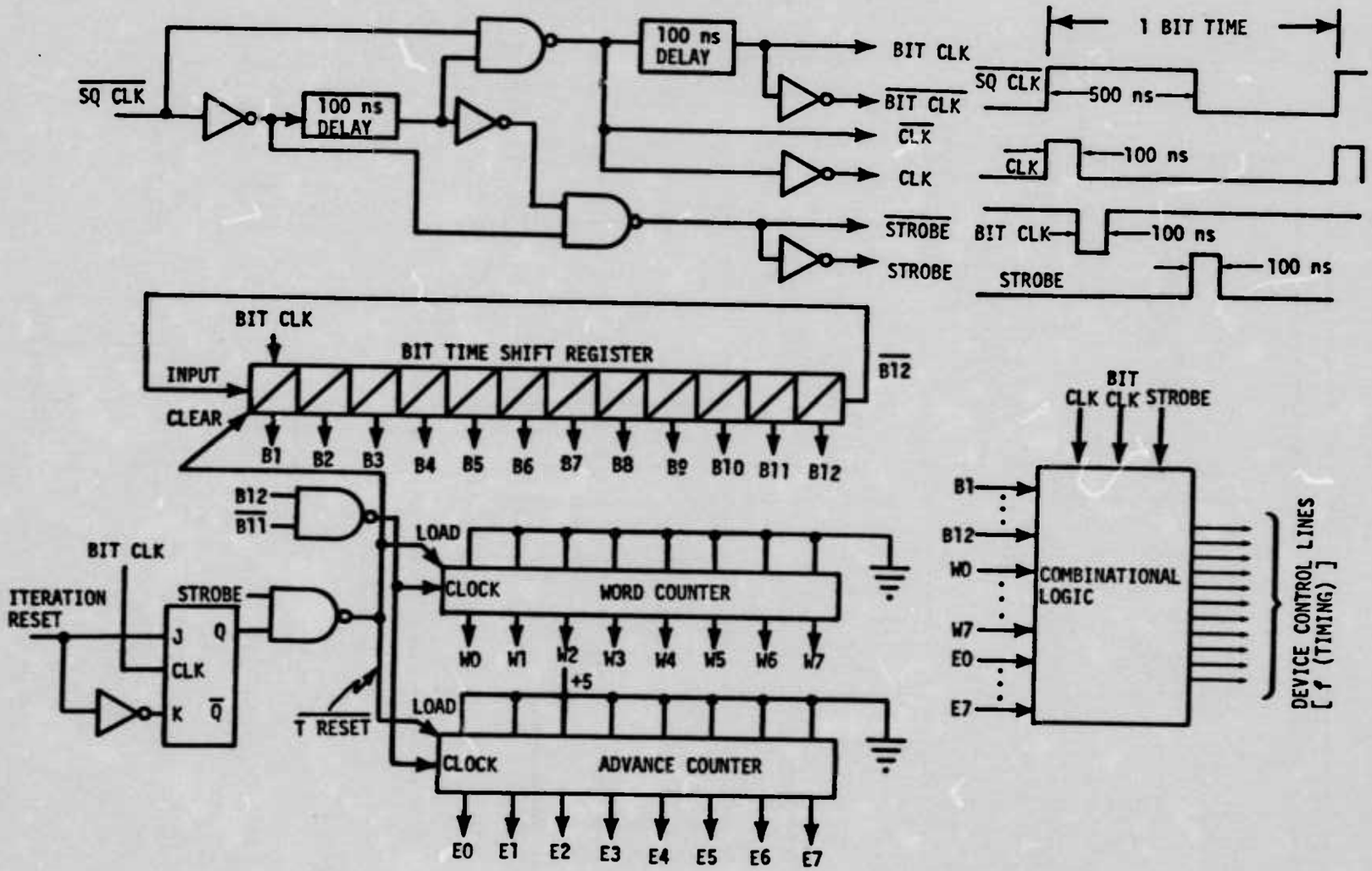


FIGURE A-25.—CIU TIMING

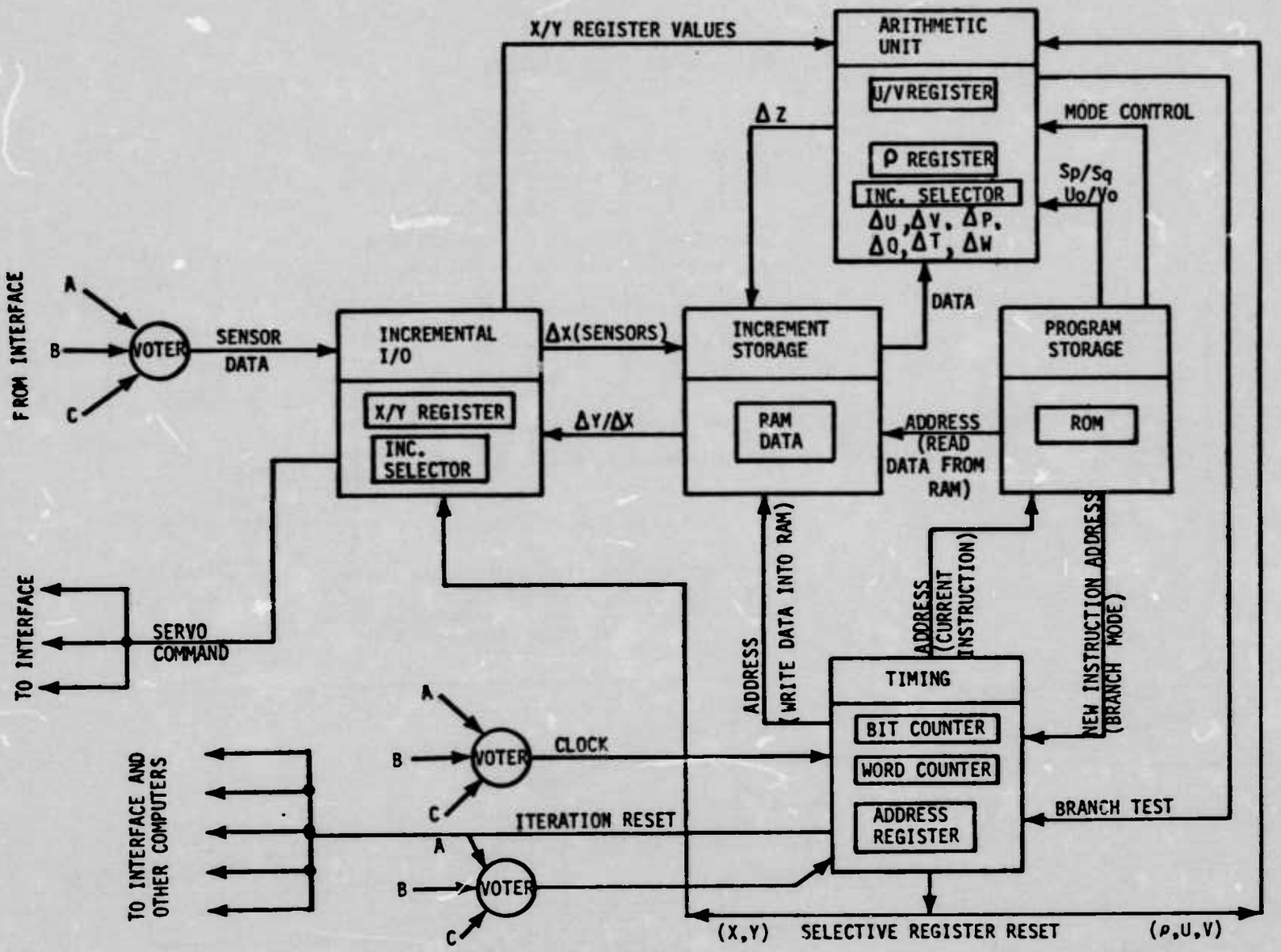


FIGURE A-26.—ICPS COMPUTER

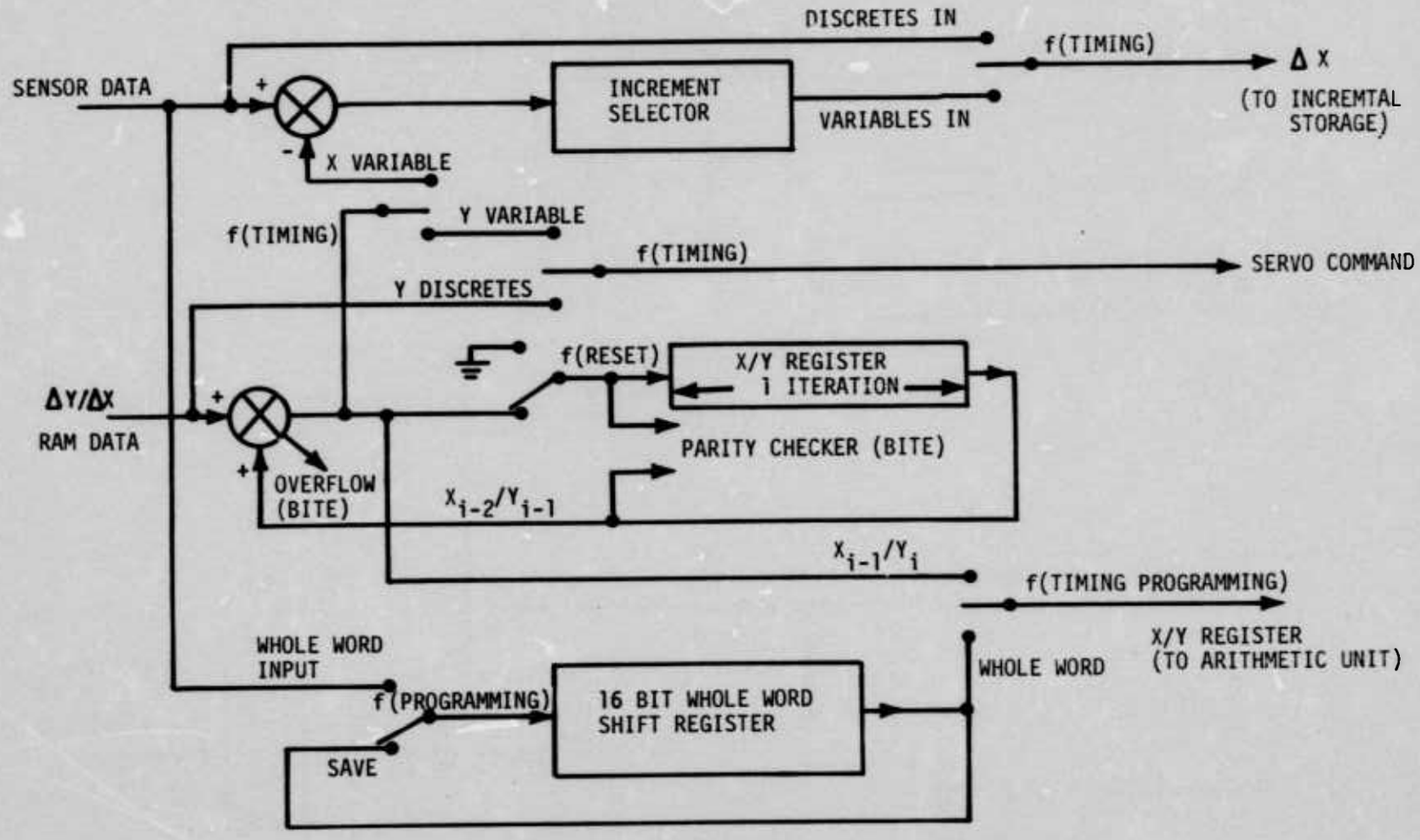


FIGURE A-27.–INCREMENTAL INPUT/OUTPUT

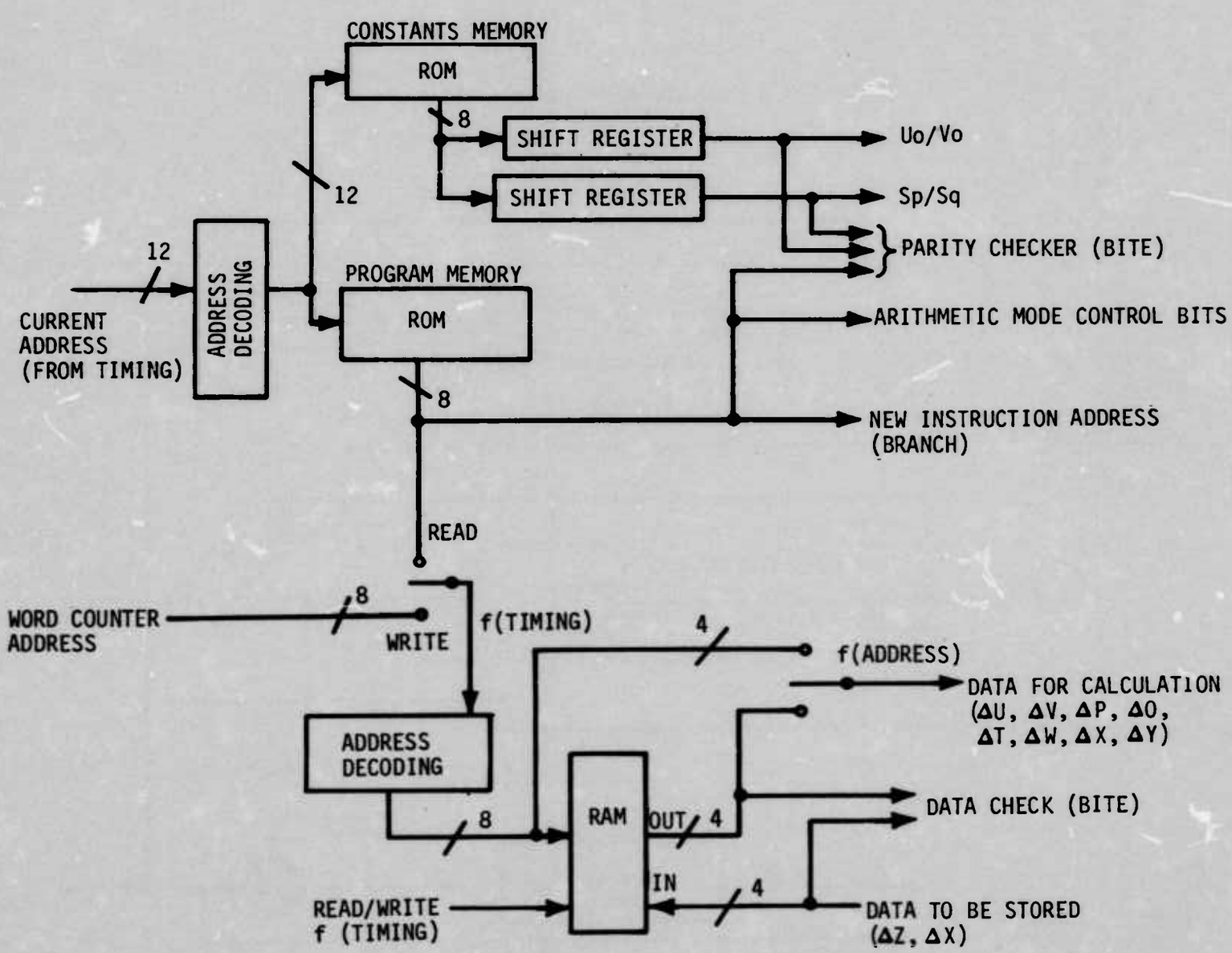


FIGURE A-28.–PROGRAM AND INCREMENT STORAGE

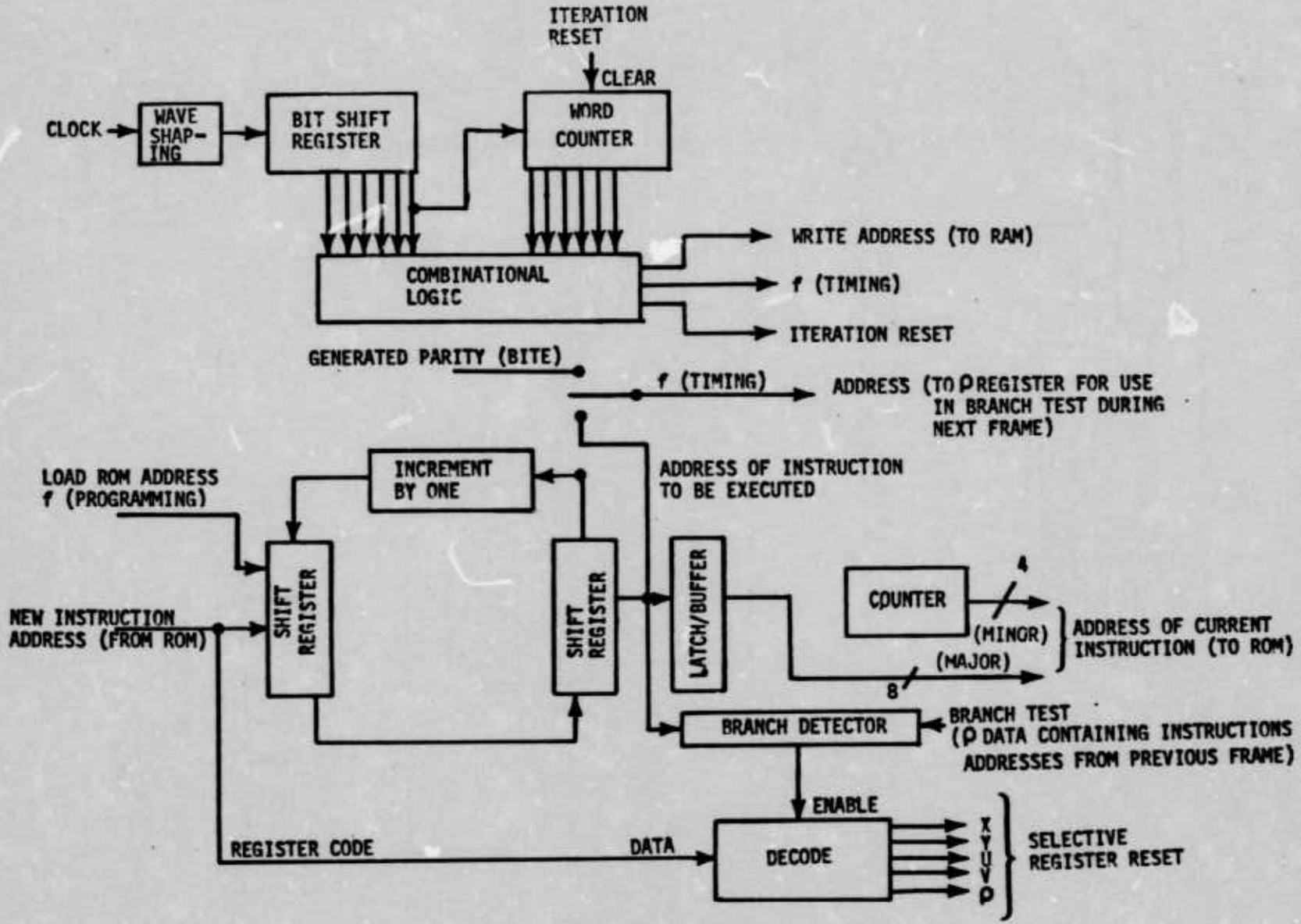


FIGURE A-29.—ICP COMPUTER TIMING

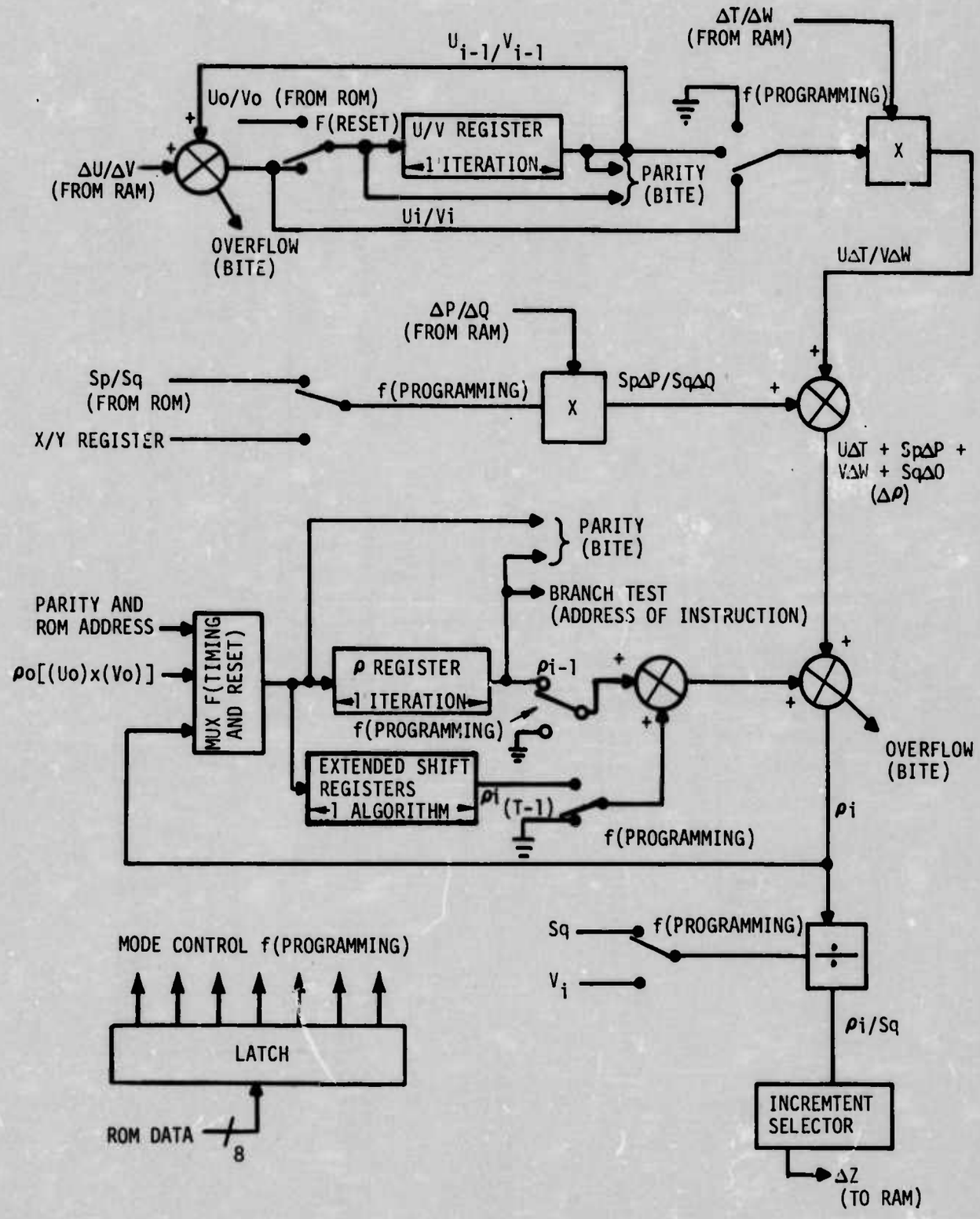


FIGURE A-30.–ARITHMETIC UNIT

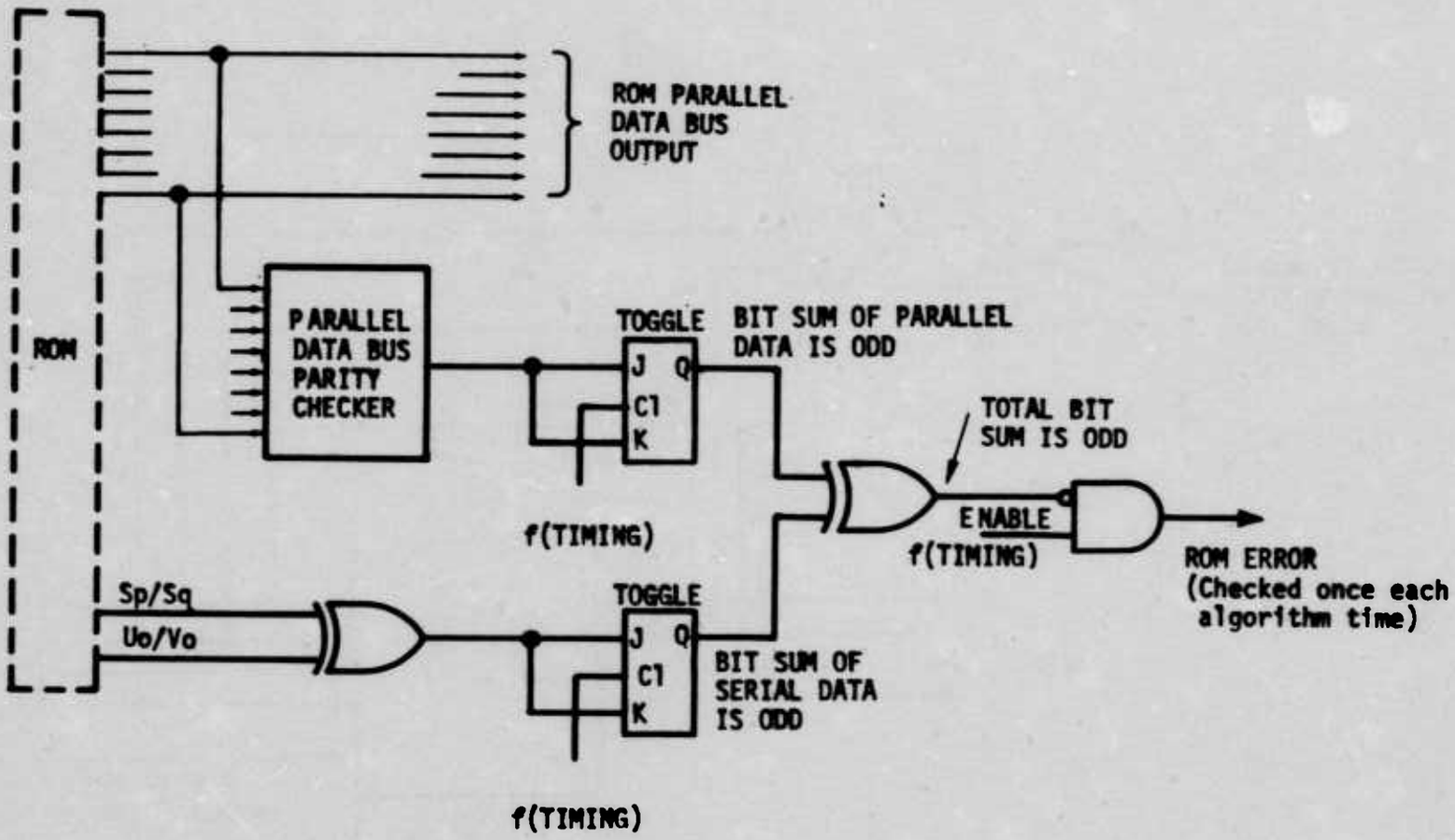


FIGURE A-31.—ROM PARITY CHECK

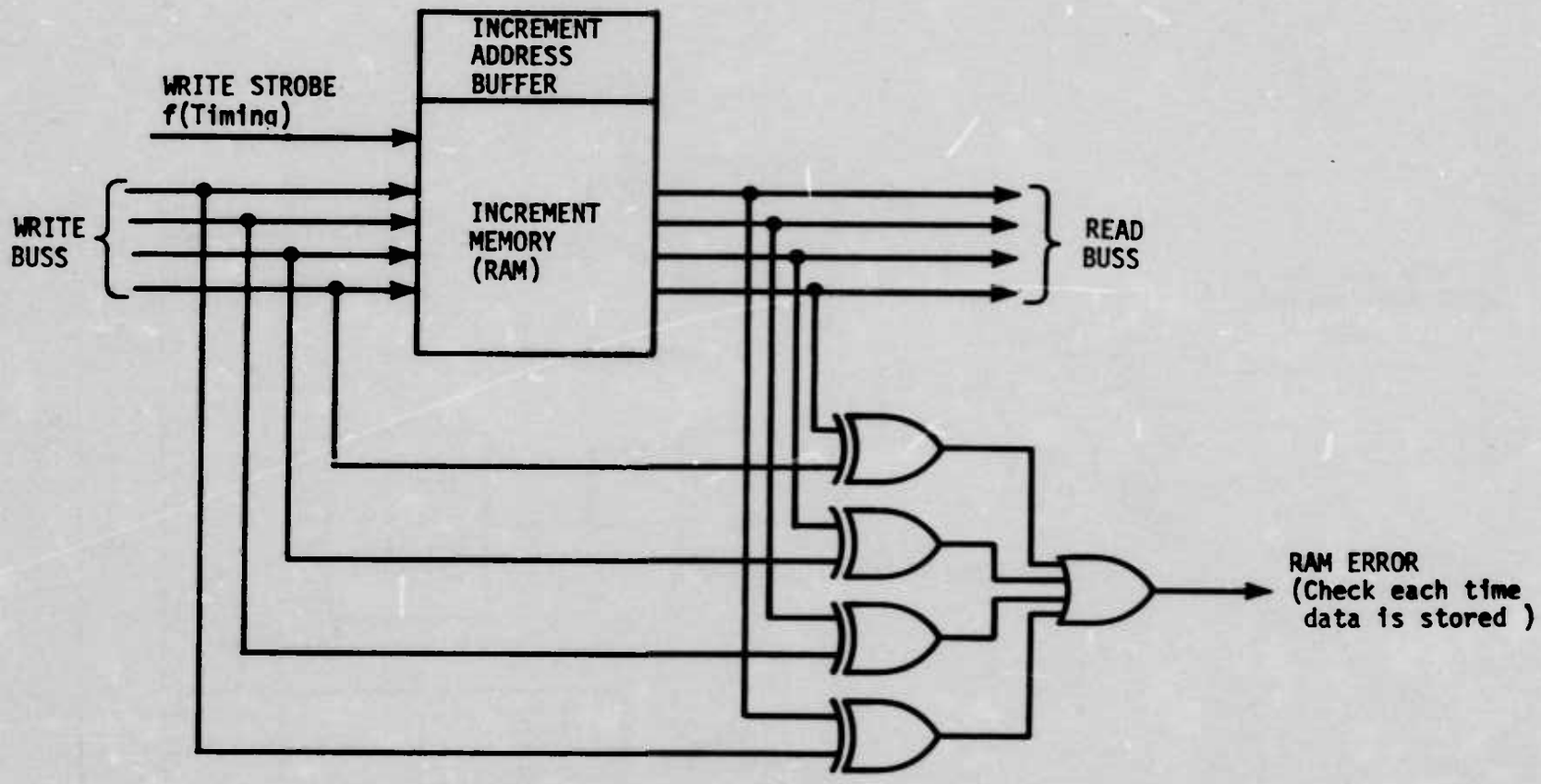


FIGURE A-32.—RAM DATA CHECK

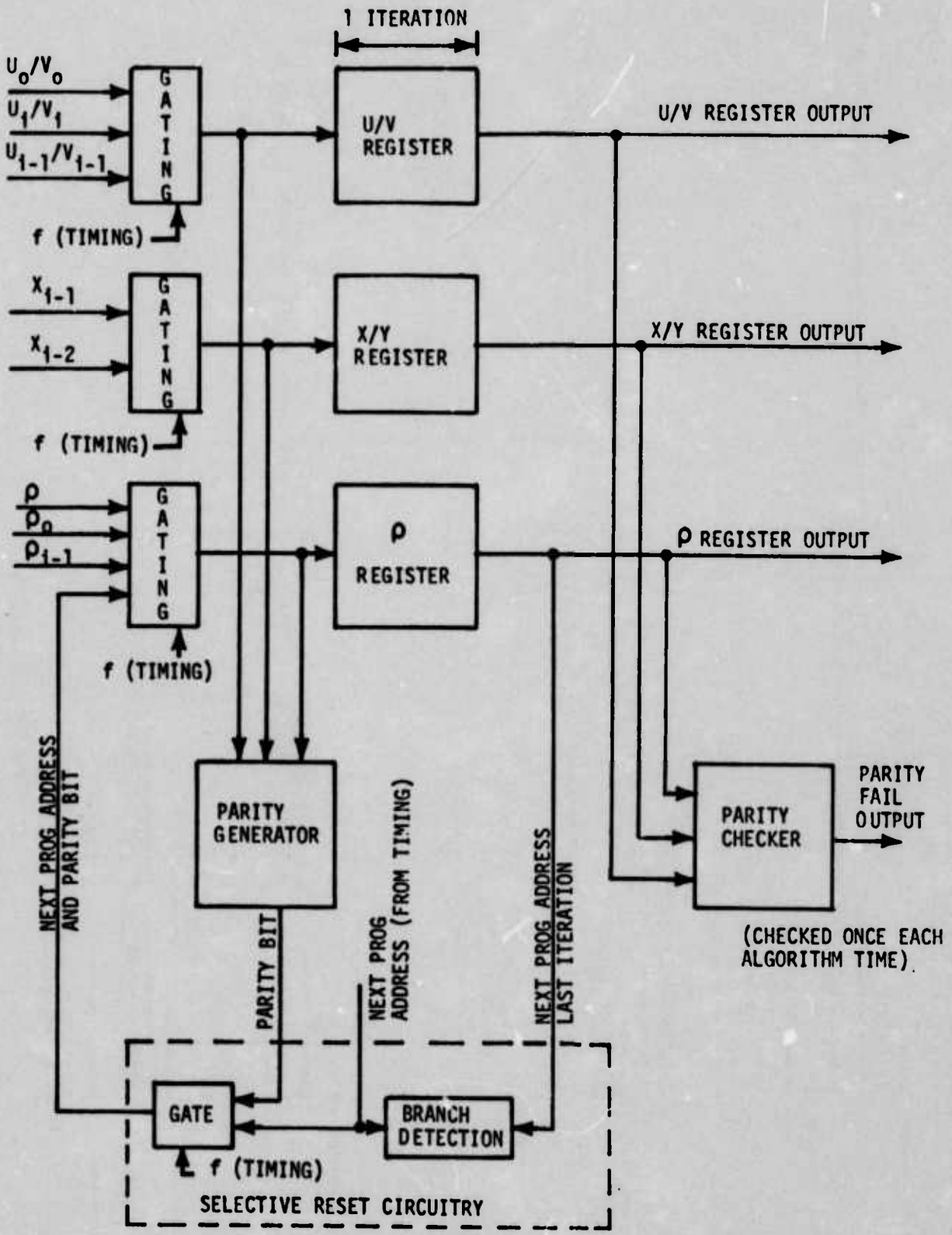


FIGURE A-33.–REGISTER PARITY CHECKING

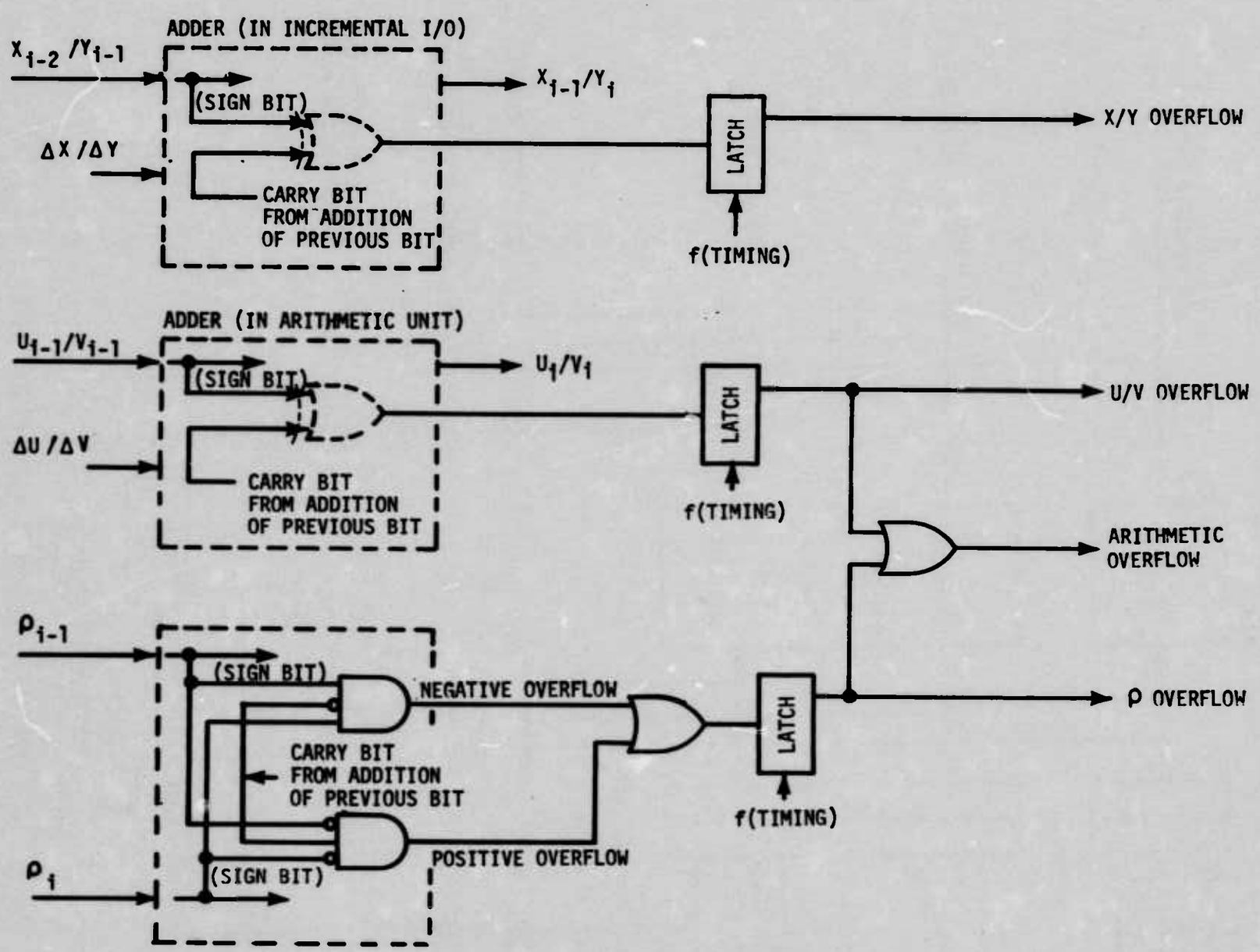


FIGURE A-34.—REGISTER OVERFLOW

CASE	DATA STORED	BITS LOCKED AT	FAILURE TYPE
1	00	00	dc
2	01	00	d
3	10	00	d
4	11	00	u
5	00	01	d
6	01	01	dc
7	10	01	u
8	11	01	d
9	00	10	d
10	01	10	u
11	10	10	dc
12	11	10	d
13	00	11	u
14	01	11	d
15	10	11	d
16	11	11	dc

dc = don't care
d = detectable failure
u = undetectable failure

FIGURE A-35.—POSSIBLE CASES FOR TWO FAILED BITS IN A MEMORY WORD

TABLE A–1.–COUNTDOWN AND WAVESHAPING TRANSITIONS–NO FAILURES

TRANSITION TABLE

t_n				t_{n+1}				
STATE	X	Y_1	Y_2	X	Y_1	Y_2	STATE	Z_0
0	0	0	0	1	0	0	4	0
1	0	0	1	1	0	1	5	1
2	0	1	0	1	1	0	6	1
3	0	1	1	1	1	1	7	1
4	1	0	0	0	1	0	2	0
5	1	0	1	0	1	1	3	1
6	1	1	0	0	0	1	1	0
7	1	1	1	0	0	0	0	0

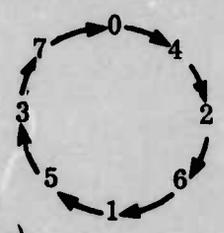
WHERE $X(t_{n+1}) = \overline{X}(t_n)$

$$Y_1(t_{n+1}) = \begin{cases} Y_1(t_n) \text{ if } X(t_n) = 0 \\ \overline{Y}_1(t_n) \text{ if } X(t_n) = 1 \end{cases}$$

$$Y_2(t_{n+1}) = \begin{cases} Y_2(t_n) \text{ if } X(t_n) \cdot Y_1(t_n) = 0 \\ \overline{Y}_2(t_n) \text{ if } X(t_n) \cdot Y_1(t_n) = 1 \end{cases}$$

$$Z_0 = X \cdot Y_1 + X \cdot Y_2 + Y_1 \cdot Y_2$$

TRANSITION MATRIX DIAGRAM



OUTPUT WAVEFORM (Z_0)

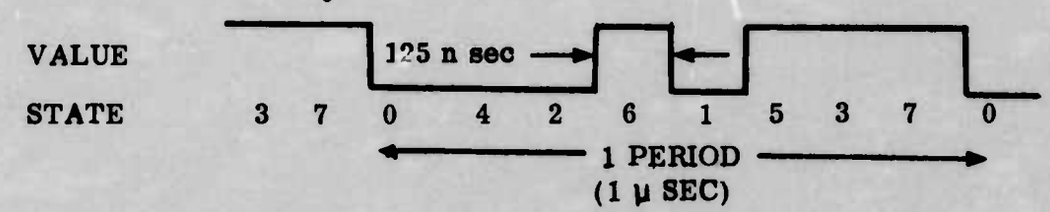
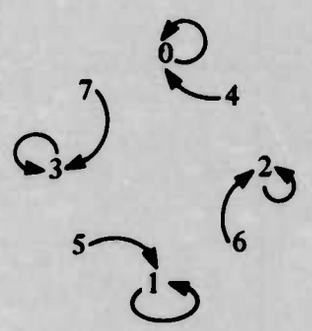


TABLE A-2.–FAILURE MODES OF FIGURE A-2

CASE #1 (X STUCK @ 0)

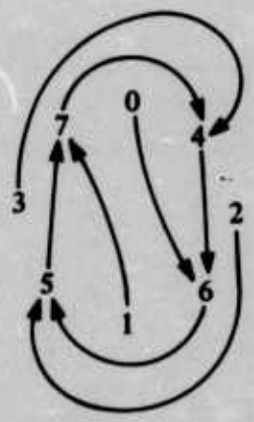
t_n	t_{n+1}				
	X	Y_1	Y_2	STATE	Z_0
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	2	0
3	0	1	1	3	1
4	0	0	0	0	0
5	0	0	1	1	0
6	0	1	0	2	0
7	0	1	1	3	1



OUTPUT: STOPS

CASE #2 (X STUCK @ 1)

t_n	t_{n+1}				
	X	Y_1	Y_2	STATE	Z_0
0	1	1	0	6	1
1	1	1	1	7	1
2	1	0	1	5	1
3	1	0	0	4	0
4	1	1	0	6	1
5	1	1	1	7	1
6	1	0	1	5	1
7	1	0	0	4	0



OUTPUT:

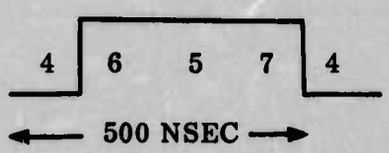
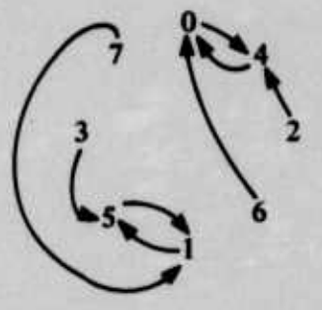


TABLE A-2.—Continued

CASE #3 (Y_1 STUCK @ 0)

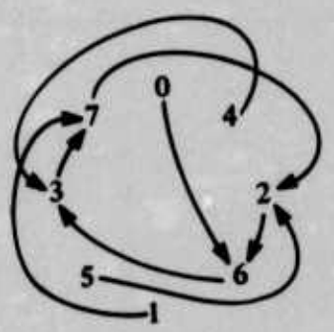
t_n	t_{n+1}				
	X	Y_1	Y_2	STATE	Z_0
0	1	0	0	4	0
1	1	0	1	5	1
2	1	0	0	4	0
3	1	0	1	5	1
4	0	0	0	0	0
5	0	0	1	1	0
6	0	0	0	0	0
7	0	0	1	1	0



OUTPUT:

CASE #4 (Y_1 STUCK @ 1)

t_n	t_{n+1}				
	X	Y_1	Y_2	STATE	Z_0
0	1	1	0	6	1
1	1	1	1	7	1
2	1	1	0	6	1
3	1	1	1	7	1
4	0	1	1	3	1
5	0	1	0	2	0
6	0	1	1	3	1
7	0	1	0	2	0



OUTPUT:

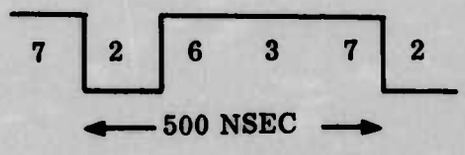
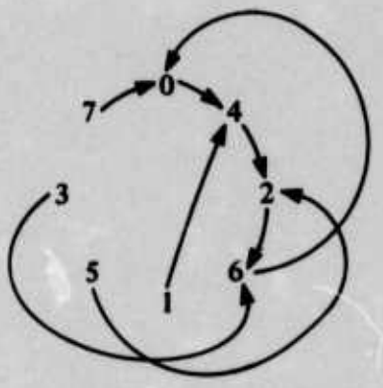


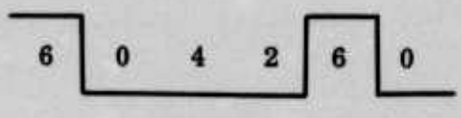
TABLE A-2.—Continued

CASE #5 (Y_2 STUCK @ 0)

t_n	t_{n+1}				
	X	Y_1	Y_2	STATE	Z_0
0	1	0	0	4	0
1	1	0	0	4	0
2	1	1	0	6	1
3	1	1	0	6	1
4	0	1	0	2	0
5	0	1	0	2	0
6	0	0	0	0	0
7	0	0	0	0	0



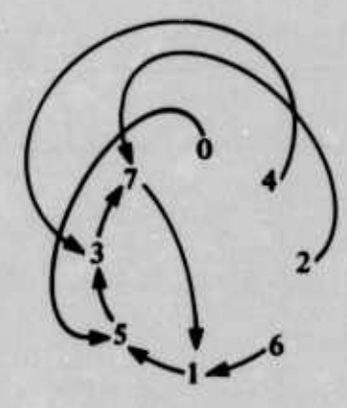
OUTPUT :



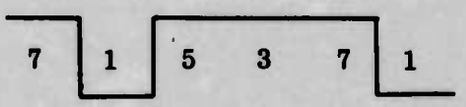
500 NSEC

CASE #6 (Y_2 STUCK @ 1)

t_n	t_{n+1}				
	X	Y_1	Y_2	STATE	Z_0
0	1	0	1	5	1
1	1	0	1	5	1
2	1	1	1	7	1
3	1	1	1	7	1
4	0	1	1	3	1
5	0	1	1	3	1
6	0	0	1	1	0
7	0	0	1	1	0



OUTPUT:

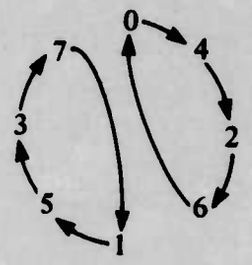


500 NSEC

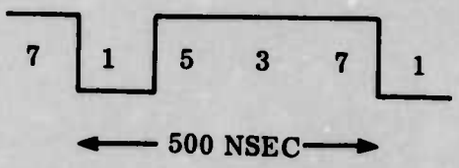
TABLE A-2.—Continued

CASE #7 (Gi STUCK @ 1)

t_n	t_{n+1}				
	X	Y_1	Z_2	STATE	Z_0
0	1	0	0	4	0
1	1	0	1	5	1
2	1	1	0	6	0
3	1	1	1	7	1
4	0	1	0	2	0
5	0	1	1	3	1
6	0	0	0	0	0
7	0	0	1	1	0

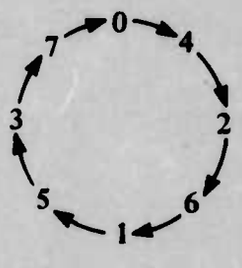


OUTPUT: ZERO OR



CASE #8 (G2 STUCK @ 1)

t_n	t_{n+1}				
	X	Y_1	Y_2	STATE	Z_0
0	SAME AS FOR NON FAILED CASE			4	0
1				5	0
2				6	1
3				7	1
4				2	0
5				3	1
6				1	0
7				0	0



OUTPUT:

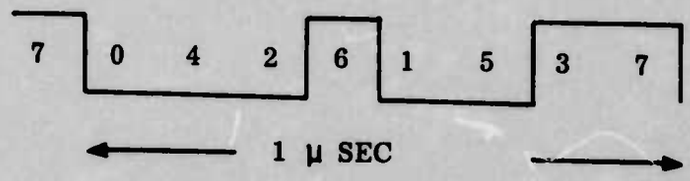
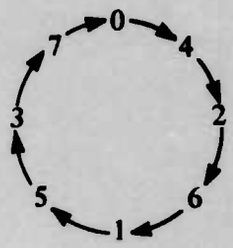


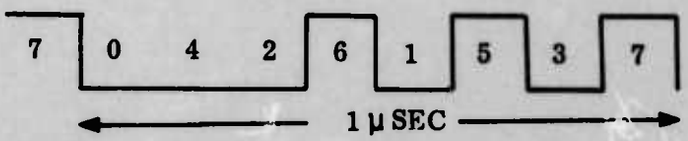
TABLE A-2.–Continued

CASE #9 (G3 STUCK @ 1)

t_n	t_{n+1}				
	X	Y_1	Y_2	STATE	Z_0
0				4	0
1				5	1
2				6	1
3	SAME AS FOR			7	1
4	NON FAILED CASE			2	0
5				3	0
6				1	0
7				0	0

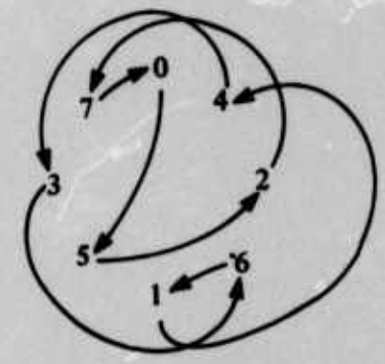


OUTPUT:



CASE #10 (G4 STUCK @ 1)

t_n	t_{n+1}				
	X	Y_1	Y_2	STATE	Z_0
0	1	0	1	5	1
1	1	0	0	4	0
2	1	1	1	7	1
3	1	1	0	6	1
4	0	1	1	3	1
5	0	1	0	2	0
6	0	0	1	1	0
7	0	0	0	0	0



OUTPUT:

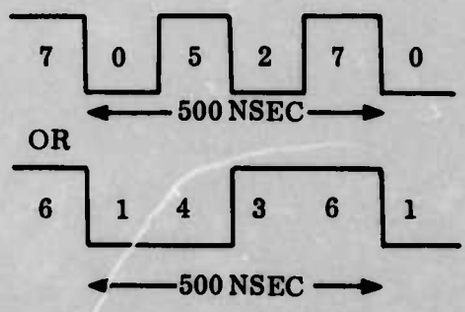
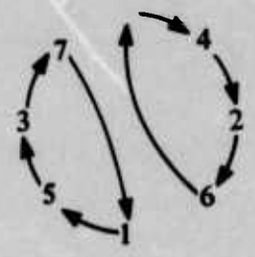


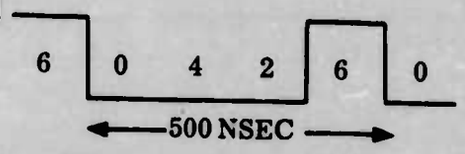
TABLE A-2.– Concluded

CASE #11 (G4 STUCK @ 0)

t_n	t_{n+1}				
	X	Y_1	Y_2	STATE	Z_0
0	1	0	0	4	0
1	1	0	1	5	1
2	1	1	0	6	1
3	1	1	1	7	1
4	0	1	0	2	0
5	0	1	1	3	1
6	0	0	0	0	0
7	0	0	1	1	0



OUTPUT:



OR

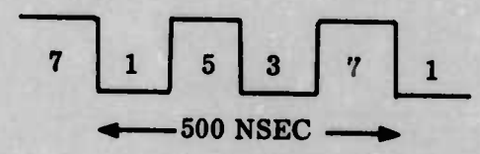


TABLE A-3.—COUNTDOWN AND WAVESHAPING TRANSITIONS (ALTERNATE CIRCUIT)—NO FAILURES

TRANSITION TABLE

t_n				t_{n+1}			
STATE	A	B	C	A	B	C	STATE
0	0	0	0	1	0	0	4
1	0	0	1	0	0	0	0
2	0	1	0	1	0	1	5
3	0	1	1	0	0	1	1
4	1	0	0	0	1	0	2
5	1	0	1	1	1	0	6
6	1	1	0	1	1	1	7
7	1	1	1	0	1	1	3

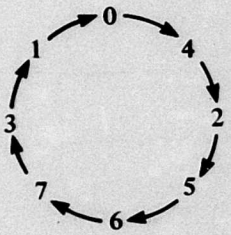
Where

$$B\ (t_{n+1}) = A\ (t_n)$$

$$C\ (t_{n+1} = B\ (t_n)$$

$$A\ (t_{n+1}) = \begin{cases} \overline{A\ (t_n)} + B\ (t_n) & \text{if } C\ (t_n) = 0 \\ A\ (t_n) \cdot \overline{B\ (t_n)} & \text{if } C\ (t_n) = 1 \end{cases}$$

$$Z_0 = C$$



TRANSITION MATRIX DIAGRAM

OUTPUT WAVEFORM (Z_0)

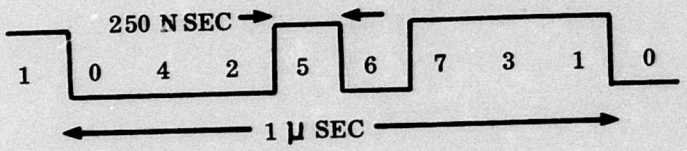
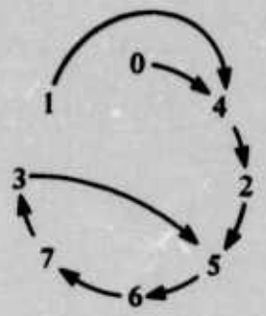


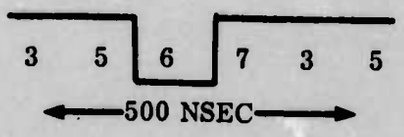
TABLE A-4.–FAILURE MODES OF FIGURE A-3

CASE 1 F/F 2 J STUCK @ 1

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	1	0	0	4
2	1	0	1	5
3	1	0	1	5
4	0	1	0	2
5	1	1	0	6
6	1	1	1	7
7	0	1	1	3

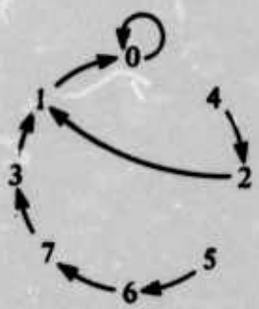


OUTPUT



CASE 2 F/F 2 J STUCK @ 0

t_n	t_{n+1}			
	A	B	C	STATE
0	0	0	0	0
1	0	0	0	0
2	0	0	1	1
3	0	0	1	1
4	0	1	0	2
5	1	1	0	6
6	1	1	1	7
7	0	1	1	3

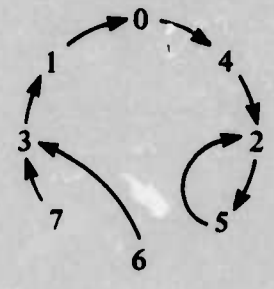


OUTPUT: STOPS

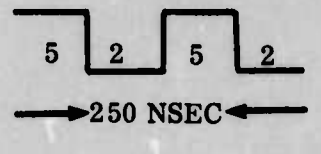
TABLE A-4.—Continued

CASE 3 F/F 2 K STUCK @ 1
(SAME AS G4 STUCK @ 1)

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	0	0	0	0
2	1	0	1	5
3	0	0	1	1
4	0	1	0	2
5	0	1	0	2
6	0	1	1	3
7	0	1	1	3

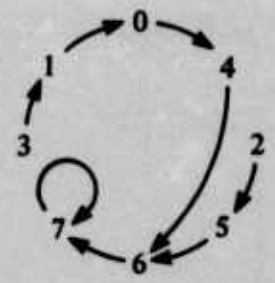


OUTPUT



CASE 4 F/F 1 K STUCK @ 0
(SAME AS G4 STUCK @ 0)

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	0	0	0	0
2	1	0	1	5
3	0	0	1	1
4	1	1	0	6
5	1	1	0	6
6	1	1	1	7
7	1	1	1	7

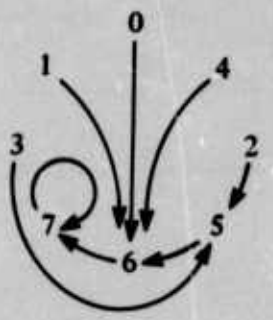


OUTPUT: STOPS

TABLE A-4.—Continued

CASE 5 F/F 1 Q STUCK @ 1

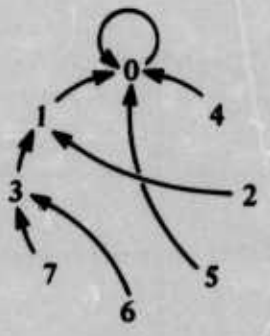
t_n	t_{n+1}			
	A	B	C	STATE
0	1	1	0	6
1	1	1	0	6
2	1	0	1	5
3	1	0	1	5
4	1	1	0	6
5	1	1	0	6
6	1	1	1	7
7	1	1	1	7



OUTPUT: STOPS

CASE 6 F/F 1 Q STUCK @ 0

t_n	t_{n+1}			
	A	B	C	STATE
0	0	0	0	0
1	0	0	0	0
2	0	0	1	1
3	0	0	1	1
4	0	0	0	0
5	0	0	0	0
6	0	1	1	3
7	0	1	1	3

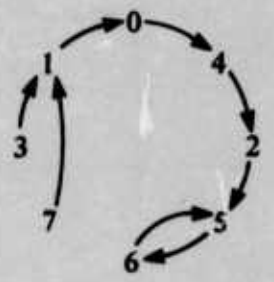


OUTPUT: STOPS

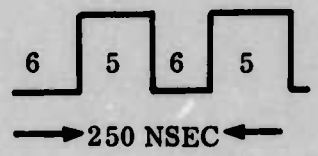
TABLE A-4.—Continued

CASE 7 F/F 2 $\bar{Q}$ STUCK @ 1

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	0	0	0	0
2	1	0	1	5
3	0	0	1	1
4	0	1	0	2
5	1	1	0	6
6	1	0	1	5
7	0	0	1	1

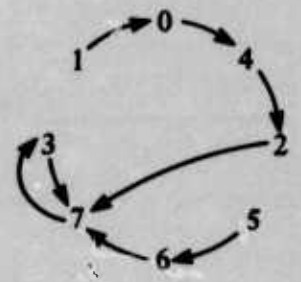


OUTPUT



CASE 8 F/F 2 $\bar{Q}$ STUCK @ 0

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	0	0	0	0
2	1	1	1	7
3	0	1	1	7
4	0	1	0	2
5	1	1	0	6
6	1	1	1	7
7	0	1	1	3

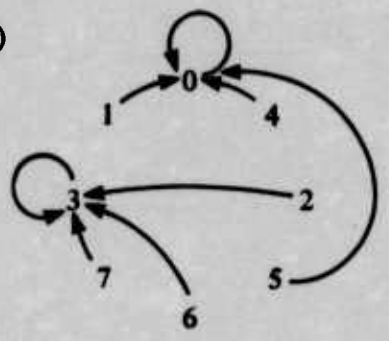


OUTPUT: STOPS

TABLE A-4.--Continued

CASE 9 F/F 2 LOSS OF POWER (Q & $\overline{Q}$ = 0)

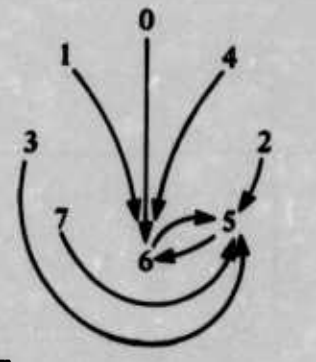
t_n	t_{n+1}			
	A	B	C	STATE
0	0	0	0	0
1	0	0	0	0
2	0	1	1	3
3	0	1	1	3
4	0	0	0	0
5	0	0	0	0
6	0	1	1	3
7	0	1	1	3



OUTPUT: STOPS

CASE 10 F/F 2 LOSS OF GROUND (Q & $\overline{Q}$ = 1)

t_n	t_{n+1}			
	A	B	C	STATE
0	1	1	0	6
1	1	1	0	6
2	1	0	1	5
3	1	0	1	5
4	1	1	0	6
5	1	1	0	6
6	1	0	1	5
7	1	0	1	5



OUTPUT

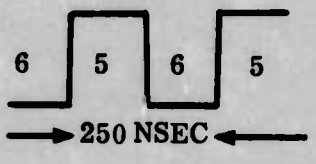
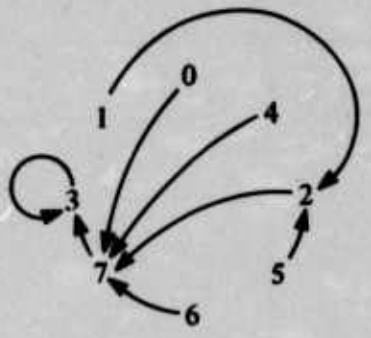


TABLE A-4.—Continued

CASE 11 F/F 3 Q STUCK @ 1

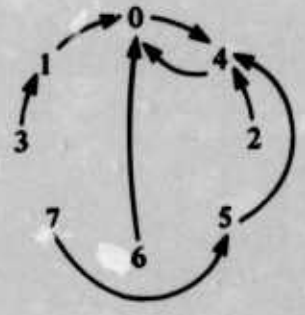
t_n	t_{n+1}			
	A	B	C	STATE
0	1	1	1	7
1	0	1	0	2
2	1	1	1	7
3	0	1	1	3
4	1	1	1	7
5	0	1	0	2
6	1	1	1	7
7	0	1	1	3



OUTPUT: STOPS

CASE 12 F/F 3 Q STUCK @ 0

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	0	0	0	0
2	1	0	0	4
3	0	0	1	1
4	0	0	0	0
5	1	0	0	4
6	0	0	0	0
7	1	0	1	5

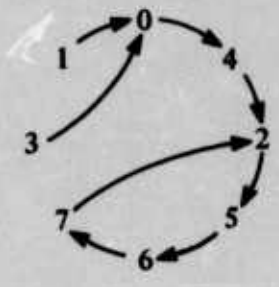


OUTPUT: STOPS

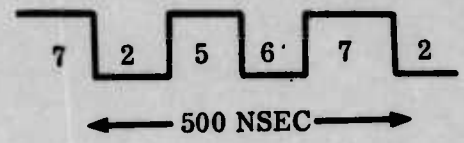
TABLE A-4.—Continued

CASE 13 F/F 3 $\bar{Q}$ STUCK @ 1

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	0	0	0	0
2	1	0	1	5
3	0	0	0	0
4	0	1	0	2
5	1	1	0	6
6	1	1	1	7
7	0	1	0	2



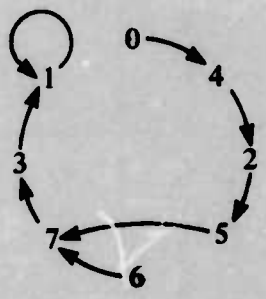
OUTPUT



500 NSEC

CASE 14 F/F 3 $\bar{Q}$ STUCK @ 0

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	0	0	1	1
2	1	0	1	5
3	0	0	1	1
4	0	1	0	2
5	1	1	1	7
6	1	1	1	7
7	0	1	1	3

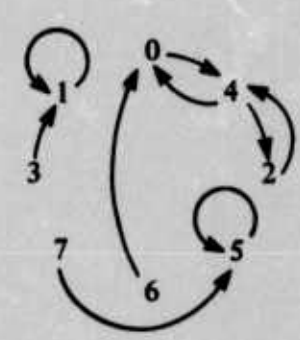


OUTPUT: STOPS

TABLE A-4.–Continued

CASE 15 F/F 3 LOSS OF POWER (Q & $\overline{Q}$ = 0)

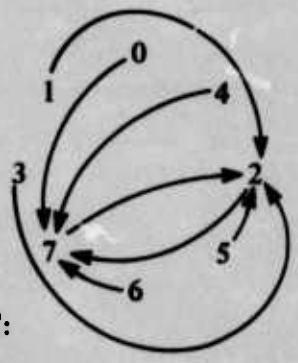
t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	0	0	1	1
2	1	0	0	4
3	0	0	1	1
4	0	0	0	0
5	1	0	1	5
6	0	0	0	0
7	1	0	1	5



OUTPUT: STOPS

CASE 16 F/F 3 LOSS OF GROUND (Q & $\overline{Q}$ = 1)

t_n	t_{n+1}			
	A	B	C	STATE
0	1	1	1	7
1	0	1	0	2
2	1	1	1	7
3	0	1	0	2
4	1	1	1	7
5	0	1	0	2
6	1	1	1	7
7	0	1	0	2



OUTPUT:

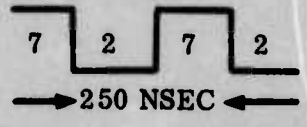
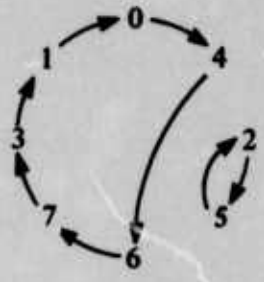


TABLE A-4.–Continued

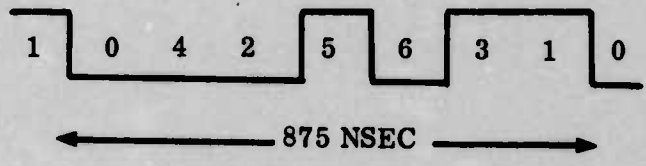
CASE 17 G1 STUCK @ 0

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	0	0	0	0
2	1	0	1	5
3	0	0	1	1
4	1	1	0	6
5	0	1	0	2
6	1	1	1	7
7	0	1	1	3



OUTPUT: STOPS

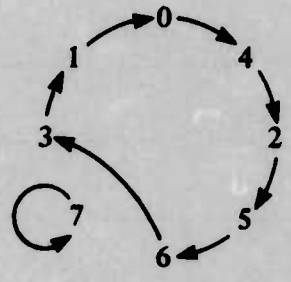
OR



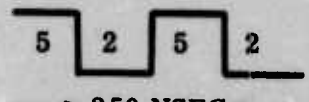
875 NSEC

CASE 18 G1 STUCK @ 1

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	0	0	0	0
2	1	0	1	5
3	0	0	1	1
4	0	1	0	2
5	1	1	0	6
6	0	1	1	3
7	1	1	1	7

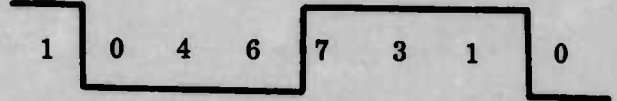


OUTPUT:



250 NSEC

OR

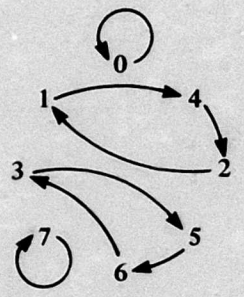


750 NSEC

TABLE A-4.—Continued

CASE 19 G2 STUCK @ 0

t_n	t_{n+1}			
	A	B	C	STATE
0	0	0	0	0
1	1	0	0	4
2	0	0	1	1
3	1	0	1	5
4	0	1	0	2
5	1	1	0	6
6	0	1	1	3
7	1	1	1	7

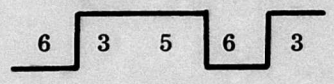


OUTPUT: STOPS

OR

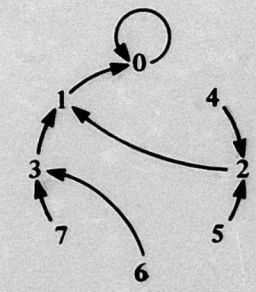
OR

375 NSEC



CASE 20 G3 STUCK @ 0

t_n	t_{n+1}			
	A	B	C	STATE
0	0	0	0	0
1	0	0	0	0
2	0	0	1	1
3	0	0	1	1
4	0	1	0	2
5	0	1	0	2
6	0	1	1	3
7	0	1	1	3

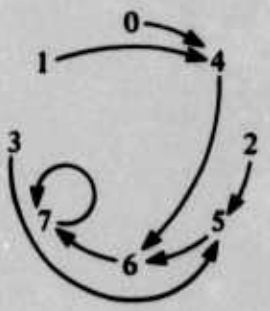


OUTPUT: STOPS

TABLE A-4.–Concluded

CASE 21 G3 STUCK @ 1

t_n	t_{n+1}			
	A	B	C	STATE
0	1	0	0	4
1	1	0	0	4
2	1	0	1	5
3	1	0	1	5
4	1	1	0	6
5	1	1	0	6
6	1	1	1	7
7	1	1	1	7



OUTPUT: STOPS

TABLE A-5.–MATRIX OF ALL POSSIBLE FAILURE MODES FOR THE CIRCUIT IN FIGURE A-4

S1 = STUCK - AT - ONE
Sø = STUCK - AT - ZERO

FAULT NO.												
ø	VOTER RELATED	G1	X-Sø	G1	X-S1	G1	1-S1	G1	2-S1	G2	X-Sø
5		G2	X-S1	G2	1-S1	G2	2-S1	G3	X-Sø	G3	X-S1
1ø		G3	1-S1	G3	2-S1	DO	X-Sø	DO	X-S1	DO	1-S1
15		DO	2-S1	DO	3-S1	G4	X-Sø	G4	X-S1	G4	1-S1
2ø	MONITOR RELATED	G4	2-S1	G4	3-S1	G4	4-S1	G5	X-Sø	G5	X-S1
25		G5	1-S1	G5	2-S1	G5	3-S1	G5	4-S1	G6	X-Sø
3ø		G6	X-S1	G6	1-S1	G6	2-S1	G6	3-S1	G6	4-S1
35		G7	X-Sø	G7	X-S1	G7	1-S1	G7	2-S1	G7	3-S1
4ø		FFAILN	X-Sø	FFAILN	X-S1	SFAILN	X-Sø	SFAILN	X-S1	SFAILN	1-S1
45		SFAILN	2-S1	SFAILN	3-S1	LFAILN	X-Sø	LFAILN	X-S1	LFAILN	1-S1
5ø		LFAILN	2-S1	LFAILN	3-S1	J SAB	X-Sø	J SAB	X-S1	K SAB	X-Sø
55		K SAB	X-S1	C SAB	X-Sø	C SAB	X-S1	R SAB	X-Sø	R SAB	X-S1
6ø		J SBC	X-Sø	J SBC	X-S1	K SBC	X-Sø	K SBC	X-S1	C SBC	X-Sø
65		C SBC	X-S1	R SBC	X-Sø	R SBC	X-S1	J SCA	X-Sø	J SCA	X-S1
7ø		K SCA	X-Sø	K SCA	X-S1	C SCA	X-Sø	C SCA	X-S1	R SCA	X-Sø
75		R SCA	X-S1	SCA	X-S1	SCA	X-Sø	SCA*	X-S1	SCA*	X-Sø
8ø		SBC	X-S1	SBC	X-Sø	SBC*	X-S1	SBC*	X-Sø	SAB	X-S1
85		SAB	X-Sø	SAB*	X-S1	SAB*	X-Sø				

TABLE A-6.—FAILURES DETECTED THROUGH THE NORMAL PROCESSING OF DATA (MLV)

* INDICATES LOGICAL COMPLEMENT

FAILURES DETECTED BY DATA BEING ALL "1's"

FAILURES DETECTED BY DATA BEING ALL "0's"

FAULT NO.										
		G1	X-S0	G1	X-S1	G1	1-S1	G1	2-S1	G2	X-S0
5	VOTER RELATED	G2	X-S1	G2	1-S1	G2	2-S1	G2	X-S0	G3	X-S1
10		G3	1-S1	G3	2-S1	DO	X-S0	DO	X-S1	DO	1-S1
15		DO	2-S1	DO	3-S1	G4	X-S0	G4	1-S1	G4	1-S1
20		G4	0-S1	G4	3-S1	G4	1-S1	G5	X-S0	G5	X-S0
25		G5	1-S1	G5	2-S1	G5	2-S1	G5	1-S1	G6	X-S0
30		G6	1-S1	G6	1-S1	G6	2-S1	G6	0-S1	G6	1-S1
35		G7	X-S0	G7	X-S1	G7	1-S1	G7	2-S1	G7	3-S1
40		FFAILN	1-S1	FFAILN	X-S1	FFAILN	X-S1	SFAILN	X-S1	SFAILN	1-S1
45	MONITOR RELATED	SFAILN	2-S1	SFAILN	3-S1	SFAILN	2-S1	LFAILN	X-S1	LFAILN	1-S1
50		LFAILN	2-S1	LFAILN	3-S1	J SAB	X-S0	J SAB	1-S1	K SAB	X-S0
55		K SAB	X-S1	C SAB	X-S0	C SAB	X-S1	R SAB	X-S0	R SAB	X-S1
60		J SBC	X-S0	J SBC	1-S1	K SBC	X-S0	K SBC	X-S1	C SBC	X-S0
65		C SBC	X-S1	R SBC	X-S0	R SBC	X-S1	J SCA	X-S0	J SCA	1-S1
70		K SCA	X-S0	K SCA	X-S1	C SCA	X-S0	C SCA	X-S1	R SCA	X-S0
75		R SCA	X-S1	SCA	X-S1	SCA	X-S0	SCA*	X-S1	J SCA	1-S1
80		SBC	X-S1	SBC	X-S0	SBC*	X-S1	J SCA	1-S1	SAB	X-S1
85		SAB	X-S0	SAB*	X-S1	J SCA	1-S1				

TABLE A-7.–UNDETECTED FAILURE MODES AFTER ALL POSSIBLE INPUT DATA SEQUENCES (MLV)

Undetectable failures

"Don't care"

FAULT NO.						
ø	VOTER RELATED	G1 X-Sø	G1 X-S1	G1 1-S1	G1 2-S1	G2 X-Sø
5		G2 X-S1	G2 1-S1	G2 2-S1	G2 X-S1	G2 X-S1
1ø		G3 1-S1	G3 2-S1	DO X-Sø	DO X-S1	DO 1-S1
15		DO 2-S1	DO 3-S1	G4 X-Sø	G4 X-S1	G4 1-S1
2ø	MONITOR RELATED	G4 2-S1	G4 3-S1	G4 4-S1	G5 X-Sø	G5 X-S1
25		G5 1-S1	G5 2-S1	G5 3-S1	G5 4-S1	G6 X-Sø
3ø		G6 X-S1	G6 1-S1	G6 2-S1	G6 3-S1	G6 4-S1
35		G7 X-Sø	G7 X-S1	G7 1-S1	G7 2-S1	G7 3-S1
4ø		FFAILN X-Sø	FFAILN X-S1	SFAILN X-Sø	SFAILN X-S1	SFAILN 1-S1
45		SFAILN 2-S1	SFAILN 3-S1	LFAILN X-Sø	LFAILN X-S1	LFAILN 1-S1
5ø		LFAILN 2-S1	LFAILN 3-S1	J SAB X-Sø	J SAB X-S1	K SAB X-Sø
55		K SAB X-S1	C SAB X-Sø	C SAB X-S1	R SAB X-Sø	R SAB X-S1
6ø		J SBC X-Sø	J SBC X-S1	K SBC X-Sø	K SBC X-S1	C SBC X-Sø
65		C SBC X-S1	R SBC X-Sø	R SBC X-S1	J SCA X-Sø	J SCA X-S1
7ø		K SCA X-Sø	K SCA X-S1	C SCA X-Sø	C SCA X-S1	R SCA X-Sø
75		R SCA X-S1	SCA X-S1	SCA X-Sø	SCA* X-S1	SCA* X-Sø
8ø		SBC X-S1	SBC X-Sø	SBC* X-S1	SBC* X-Sø	SAB X-S1
85		SAB X-Sø	SAB* X-S1	SAB* X-Sø		

TABLE A-8.–TABULATION OF FAULT SIMULATION RUNS MADE

	RUN NUMBER	CH A INPUT	CH B INPUT	CH C INPUT	FAILURE SEQUENCE	REMARKS
	1	1	1	1	——	NORMAL DATA, ALL "1's"
	2	0	0	0	——	NORMAL DATA, ALL "0's"
INPUT DATA FAILURES TO 0	3	0 0	1 0	1 1	FIRST FAIL SEC FAIL	A FAIL LO B LO, C HI
	4	0 0	1 1	1 0	FIRST FAIL SEC FAIL	A FAIL LO B HI, C LO
	5	1 0	0 0	1 1	FIRST FAIL SEC FAIL	B FAIL LO A LO, C HI
	6	1 1	0 0	1 0	FIRST FAIL SEC FAIL	B FAIL LO A HI, C LO
	7	1 0	1 1	0 0	FIRST FAIL SEC FAIL	C FAIL LO A LO, B HI
	8	1 1	1 0	0 0	FIRST FAIL SEC FAIL	C FAIL LO A HI, B LO
INPUT DATA FAILURES TO 1	9	1 1	0 0	0 1	FIRST FAIL SEC FAIL	A FAIL HI B LO, C HI
	10	1 1	0 1	0 0	FIRST FAIL SEC FAIL	A FAIL HI B HI, C LO
	11	0 0	1 1	0 1	FIRST FAIL SEC FAIL	B FAIL HI A LO, C HI
	12	0 1	1 1	0 0	FIRST FAIL SEC FAIL	B FAIL HI A HI, C LO
	13	0 0	0 1	1 1	FIRST FAIL SEC FAIL	C FAIL HI A LO, B HI
	14	0 1	0 0	1 1	FIRST FAIL SEC FAIL	C FAIL HI A HI, B LO

TABLE A-9.–FAILURES DETECTED BY PROCESSING NORMAL DATA

RUN 1 & 2

	RUN 1	RUN 2
CH A	1	0
CH B	1	0
CH C	1	0

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.		RUN 1 CH A		RUN 1 CH B		RUN 1 CH C		RUN 2 CH A		RUN 2 CH B	
ø	VOTER RELATED	~~G1~~	~~X-Sø~~	G1	X-S1	G1	1-S1	G1	2-S1	~~G1~~	~~X-Sø~~
5		G2	X-S1	G2	1-S1	G2	2-S1	~~G2~~	~~X-Sø~~	G3	X-S1
1ø		G3	1-S1	G3	2-S1	~~DO~~	~~X-Sø~~	~~DO~~	~~X-S1~~	DO	1-S1
15		DO	2-S1	DO	3-S1	G4	X-Sø	~~G4~~	~~X-Sø~~	~~G4~~	~~1-S1~~
2ø	MONITOR RELATED	~~G4~~	~~2-S1~~	~~G4~~	~~3-S1~~	~~G4~~	~~4-S1~~	G5	X-Sø	~~G5~~	~~X-Sø~~
25		~~G5~~	~~1-S1~~	~~G5~~	~~2-S1~~	~~G5~~	~~3-S1~~	~~G5~~	~~1-S1~~	G6	X-Sø
3ø		~~G6~~	~~2-S1~~	~~G6~~	~~1-S1~~	~~G6~~	~~2-S1~~	~~G6~~	~~2-S1~~	~~G6~~	~~4-S1~~
35		G7	X-Sø	~~G7~~	~~X-S1~~	G7	1-S1	G7	2-S1	G7	3-S1
4ø		~~SFAILN~~	~~X-Sø~~	FFAILN	X-S1	~~SFAILN~~	~~X-Sø~~	SFAILN	X-S1	SFAILN	1-S1
45		SFAILN	2-S1	SFAILN	3-S1	~~LFAILN~~	~~X-Sø~~	LFAILN	X-S1	LFAILN	1-S1
5ø		LFAILN	2-S1	LFAILN	3-S1	J SAB	X-Sø	~~C SAB~~	~~X-Sø~~	K SAB	X-Sø
55		K SAB	X-S1	C SAB	X-Sø	C SAB	X-S1	R SAB	X-Sø	R SAB	X-S1
6ø		J SBC	X-Sø	~~J SBC~~	~~X-S1~~	K SBC	X-Sø	K SBC	X-S1	C SBC	X-Sø
65		C SBC	X-S1	R SBC	X-Sø	R SBC	X-S1	J SCA	X-Sø	~~J SCA~~	~~X-S1~~
7ø		K SCA	X-Sø	K SCA	X-S1	C SCA	X-Sø	C SCA	X-S1	R SCA	X-Sø
75		R SCA	X-S1	SCA	X-S1	SCA	X-Sø	SCA*	X-S1	~~SCA~~	~~X-Sø~~
8ø		SBC	X-S1	SBC	X-Sø	SBC*	X-S1	~~SBC~~	~~X-Sø~~	SAB	X-S1
85		SAB	X-Sø	SAB*	X-S1	~~SAB~~	~~X-Sø~~				

* Indicates logical complement.

Failures detected by data being all "1's"

Failures detected by data being all "0's"

TABLE A-10.–FAILURE DETECTED BY VARIOUS COMBINATIONS OF FIRST AND SECOND FAILURE SEQUENCE

RUN 3

	First Fail Sequence	Second Fail Sequence
CH A	0	0
CH B	1	0
CH C	1	1

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.										
0	VOTER RELATED	~~G1 X-S0~~		G1 X-S1		G1 1-S1		G1 2-S1		~~G2 X-S0~~
5		~~G2 X-S1~~		~~G2 1-S1~~		G2 2-S1		~~G2 X-S0~~		G3 X-S1
10		G3 1-S1		~~G3 2-S1~~		~~DO X-S0~~		~~DO X-S1~~		DO 1-S1
15		~~DO 2-S1~~		DO 3-S1		~~G4 X-S0~~		G4 X-S1		G4 1-S1
20		G4 2-S1		G4 3-S1		G4 4-S1		~~G5 X-S0~~		~~G5 X-S1~~
25		G5 1-S1		~~G5 2-S1~~		~~G5 3-S1~~		G5 4-S1		~~G6 X-S0~~
30		G6 X-S1		G6 1-S1		G6 2-S1		G6 3-S1		G6 4-S1
35		~~G7 X-S0~~		G7 X-S1		G7 1-S1		G7 2-S1		G7 3-S1
40		FFAILN X-S0		~~FFAILN X-S1~~		~~SFAILN X-S0~~		~~SFAILN X-S1~~		SFAILN 1-S1
45		~~SFAILN 0-S1~~		SFAILN 3-S1		~~LFAILN X-S0~~		~~LFAILN X-S1~~		LFAILN 1-S1
50		LFAILN 2-S1		~~LFAILN 2-S1~~		~~J SAB X-S0~~		J SAB X-S1		K SAB X-S0
55		~~K SAB X-S1~~		~~C SAB X-S0~~		~~C SAB X-S1~~		~~K SAB X-S0~~		R SAB X-S1
60		~~J SBC X-S0~~		~~J SBC X-S1~~		K SBC X-S0		K SBC X-S1		~~C SBC X-S0~~
65		~~C SBC X-S1~~		~~R SBC X-S0~~		R SBC X-S1		~~K SCA X-S0~~		J SCA X-S1
70		K SCA X-S0		~~K SCA X-S1~~		~~C SCA X-S0~~		~~C SCA X-S1~~		~~R SCA X-S0~~
75		R SCA X-S1		SCA X-S1		~~SCA X-S0~~		SCA* X-S1		SCA* X-S0
80		~~SBC X-S1~~		~~SBC X-S0~~		~~SBC* X-S1~~		~~SBC* X-S0~~		SAB X-S1
85		~~SAB X-S0~~		SAB* X-S1		SAB* X-S0				

* Indicates logical complement

Failures detected by first fail sequence

Additional failures detected by second fail sequence

TABLE A-10.—Continued

RUN 4

	First Fail Sequence	Second Fail Sequence
CH A	0	0
CH B	1	1
CH C	1	0

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.								
~~ø~~	VOTER RELATED	~~G1 X-Sø~~	G1 X-S1	~~G1 1-S1~~	G1 2-S1	~~G2 X-Sø~~		
5		~~G2 X-S1~~	G2 1-S1	~~G2 2-S1~~	~~G2 X-Sø~~	G3 X-S1		
1ø		G3 1-S1	G3 2-S1	~~DO X-Sø~~	~~DO X-S1~~	DO 1-S1		
15		~~DO 2-S1~~	DO 3-S1	~~G3 3-Sø~~	G4 X-S1	G4 1-S1		
2ø	MONITOR RELATED	G4 2-S1	G4 3-S1	G4 4-S1	~~G5 X-Sø~~	~~G5 X-S1~~		
25		G5 1-S1	~~G5 3-S1~~	~~G5 3-S1~~	G5 4-S1	~~G6 4-Sø~~		
3ø		G6 X-S1	G6 1-S1	G6 2-S1	G6 3-S1	G6 4-S1		
35		~~G7 X-Sø~~	G7 X-S1	G7 1-S1	G7 2-S1	G7 3-S1		
4ø		FFAILN X-Sø	~~FFAILN X-S1~~	~~SFAILN X-Sø~~	~~SFAILN X-S1~~	SFAILN 1-S1		
45		~~SFAILN 2-S1~~	SFAILN 3-S1	~~LFAILN X-Sø~~	~~LFAILN X-S1~~	LFAILN 1-S1		
5ø		LFAILN 2-S1	~~LFAILN 3-S1~~	~~J SAB X-Sø~~	J SAB X-S1	K SAB X-Sø		
55		~~K SAB X-S1~~	~~R SAB X-Sø~~	~~R SAB X-S1~~	~~R SAB X-Sø~~	R SAB X-S1		
6ø		~~J SBC X-Sø~~	~~J SBC X-S1~~	K SBC X-Sø	K SBC X-S1	~~C SBC X-Sø~~		
65		~~C SBC X-S1~~	~~R SBC X-Sø~~	R SBC X-S1	~~R SBC 2-Sø~~	J SCA X-S1		
7ø		K SCA X-Sø	~~K SCA X-S1~~	~~R SCA 2-Sø~~	~~R SCA 2-S1~~	~~K SCA X-Sø~~		
75		R SCA X-S1	SCA X-S1	~~SCA 3-Sø~~	SCA* X-S1	SCA* X-Sø		
8ø		~~SBC X-S1~~	~~SBC X-Sø~~	~~SBC X-S1~~	~~SBC* X-Sø~~	SAB X-S1		
85		~~SAB 2-Sø~~	SAB* X-S1	SAB* X-Sø				

* Indicates logical complement

Failures detected by fail sequence

Additional failures detected by second fail sequence

TABLE A-10.—Continued

RUN 5

	First Fail Sequence	Second Fail Sequence
CH A	1	0
CH B	0	0
CH C	1	1

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.										
0	VOTER RELATED	~~G1~~	~~X-S0~~	G1	X-S1	G1	1-S1	G1	2-S1	~~G0~~	~~X-S0~~
5		G2	X-S1	~~G0~~	~~1-S1~~	G2	2-S1	~~G2~~	~~X-S0~~	~~G2~~	~~X-S1~~
10		G3	1-S1	~~G0~~	~~2-S1~~	~~DO~~	~~X-S0~~	~~DO~~	~~X-S1~~	DO	1-S1
15		DO	2-S1	~~DO~~	~~3-S1~~	~~G4~~	~~X-S0~~	G4	X-S1	G4	1-S1
20	MONITOR RELATED	G4	2-S1	G4	3-S1	G4	4-S1	~~G5~~	~~X-S0~~	G5	X-S1
25		G5	1-S1	G5	2-S1	G5	3-S1	G5	4-S1	~~G6~~	~~X-S0~~
30		~~G6~~	~~X-S1~~	G6	1-S1	~~G6~~	~~2-S1~~	~~G6~~	~~3-S1~~	G6	4-S1
35		~~G7~~	~~X-S0~~	G7	X-S1	G7	1-S1	G7	2-S1	G7	3-S1
40		FFAILN	X-S0	~~FFAILN~~	~~X-S1~~	~~SFAILN~~	~~X-S0~~	~~SFAILN~~	~~X-S1~~	SFAILN	1-S1
45		SFAILN	2-S1	~~SFAILN~~	~~0-S1~~	~~LFAILN~~	~~X-S0~~	LFAILN	X-S1	LFAILN	1-S1
50		LFAILN	2-S1	~~LFAILN~~	~~2-S1~~	~~J SAB~~	~~X-S0~~	J SAB	X-S1	K SAB	X-S0
55		~~K SAB~~	~~X-S1~~	~~Q SAB~~	~~X-S0~~	~~Q SAB~~	~~X-S1~~	~~R SAB~~	~~X-S0~~	R SAB	X-S1
60		~~J SBC~~	~~X-S0~~	J SBC	X-S1	K SBC	X-S0	~~K SBC~~	~~X-S1~~	~~Q SBC~~	~~X-S0~~
65		~~Q SBC~~	~~X-S1~~	~~R SBC~~	~~X-S0~~	R SBC	X-S1	~~J SCA~~	~~X-S0~~	~~J SCA~~	~~X-S1~~
70		K SCA	X-S0	K SCA	X-S1	~~Q SCA~~	~~X-S0~~	~~Q SCA~~	~~X-S1~~	~~R SCA~~	~~X-S0~~
75		R SCA	X-S1	~~SCA~~	~~X-S1~~	~~SCA~~	~~X-S0~~	SCA*	X-S1	SCA*	X-S0
80		SBC	X-S1	~~SBC~~	~~X-S0~~	~~SBC*~~	~~X-S1~~	SBC*	X-S0	SAB	X-S1
85		~~SAB~~	~~X-S0~~	SAB*	X-S1	SAB*	X-S0				

* Indicates logical complement

Failures detected by first fail sequence

Additional failures detected by second fail sequence

TABLE A-10.—Continued

RUN 6

	First Fail Sequence	Second Fail Sequence
CH A	1	1
CH B	0	0
CH C	1	0

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.										
0	VOTER RELATED	~~G1 X-S0~~		G1 X-S1		G1 1-S1		~~G1 2-S1~~		~~G2 X-S0~~
5		G2 X-S1		G2 1-S1		G2 2-S1		~~G2 X-S0~~		~~G2 X-S1~~
10		~~G3 1-S1~~		G3 2-S1		~~DO X-S0~~		~~DO X-S1~~		DO 1-S1
15		DO 2-S1		~~DO 3-S1~~		~~G1 X-S0~~		G4 X-S1		G4 1-S1
20	MONITOR RELATED	G4 2-S1		G4 3-S1		G4 4-S1		~~G5 X-S0~~		G5 X-S1
25		G5 1-S1		G5 2-S1		G5 3-S1		G5 4-S1		~~G6 X-S0~~
30		~~G6 X-S1~~		G6 1-S1		~~G6 2-S1~~		~~G6 3-S1~~		G6 4-S1
35		~~G7 X-S0~~		G7 X-S1		G7 1-S1		G7 2-S1		G7 3-S1
40		FFAILN X-S0		~~FFAILN X-S1~~		~~SFAILN X-S0~~		~~SFAILN X-S1~~		SFAILN 1-S1
45		SFAILN 2-S1		~~SFAILN 3-S1~~		~~LFAILN X-S0~~		LFAILN X-S1		LFAILN 1-S1
50		LFAILN 2-S1		~~LFAILN 3-S1~~		~~J SAB X-S0~~		J SAB X-S1		K SAB X-S0
55		~~K SAB X-S1~~		~~G SAB X-S0~~		~~G SAB X-S1~~		~~R SAB X-S0~~		R SAB X-S1
60		~~J SBC X-S0~~		J SBC X-S1		K SBC X-S0		~~K SBC X-S1~~		~~G SBC X-S0~~
65		~~G SBC X-S1~~		~~R SBC X-S0~~		R SBC X-S1		~~J SCA X-S0~~		~~J SCA X-S1~~
70		K SCA X-S0		K SCA X-S1		~~G SCA X-S0~~		~~G SCA X-S1~~		~~R SCA X-S0~~
75		R SCA X-S1		~~SCA X-S1~~		~~SCA X-S0~~		SCA* X-S1		SCA* X-S0
80		SBC X-S1		~~SBC X-S0~~		~~SBC* X-S1~~		SBC* X-S0		SAB X-S1
85		~~SAB X-S0~~		SAB* X-S1		SAB* X-S0				

* Indicates logical complement

Failures detected by first fail sequence

Additional failures detected by second fail sequence

TABLE A-10.—Continued

RUN 7

	First Fail Sequence	Second Fail Sequence
CH A	1	0
CH B	1	1
CH C	0	0

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.										
0	VOTER RELATED	~~G1 X-S0~~		~~G1 X-S1~~		~~G1 1-S0~~		G1 2-S1		~~G2 X-S0~~
5		G2	X-S1	G2	1-S1	~~G2 0-S1~~		~~G2 X-S0~~		G3 X-S1
10		G3	1-S1	G3	2-S1	~~DO X-S0~~		~~DO X-S1~~		~~DO 1-S1~~
15		DO	2-S1	DO	3-S1	~~G4 X-S0~~		~~G4 X-S1~~		G4 1-S1
20		~~G4 0-S1~~		~~G4 0-S1~~		G4	4-S1	~~G5 X-S0~~		G5 X-S1
25	MONITOR RELATED	G5	1-S1	G5	2-S1	G5	3-S1	G5	4-S1	~~G6 X-S0~~
30		G6	X-S1	G6	1-S1	G6	2-S1	G6	3-S1	G6 4-S1
35		~~G7 X-S0~~		G7	X-S1	G7	1-S1	G7	2-S1	G7 3-S1
40		FFAILN	X-S0	~~FFAILN X-S1~~		~~SFAILN X-S0~~		~~SFAILN X-S1~~		~~SFAILN 1-S1~~
45		SFAILN	2-S1	SFAILN	3-S1	~~LFAILN X-S0~~		LFAILN	X-S1	LFAILN 1-S1
50		LFAILN	2-S1	~~LFAILN 0-S1~~		~~J SAB X-S0~~		~~J SAB X-S1~~		K SAB X-S0
55		K SAB	X-S1	~~G SAB X-S0~~		~~G SAB X-S1~~		~~R SAB X-S0~~		R SAB X-S1
60		~~J SBC X-S0~~		J SBC	X-S1	K SBC	X-S0	~~K SBC X-S1~~		~~G SBC X-S0~~
65		~~G SBC X-S1~~		~~R SBC X-S0~~		R SBC	X-S1	~~J SCA X-S0~~		~~J SCA X-S1~~
70		K SCA	X-S0	~~K SCA X-S1~~		~~G SCA X-S0~~		~~G SCA X-S1~~		~~R SCA X-S0~~
75		R SCA	X-S1	SCA	X-S1	~~SCA X-S0~~		SCA*	X-S1	SCA* X-S0
80		SBC	X-S1	~~SBC X-S0~~		~~SBC X-S1~~		SBC*	X-S0	~~SAB X-S1~~
85		~~SAB X-S0~~		SAB*	X-S1	SAB*	X-S0			

* Indicates logical complement

Failures detected by first fail sequence

Additional failures detected by second fail sequence

TABLE A-10.—Continued

RUN 8

	First Fail Sequence	Second Fail Sequence
CH A	1	1
CH B	1	0
CH C	0	0

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.												
0	VOTER RELATED	~~G1 X-S0~~		~~G1 X-S1~~		G1	1-S1	~~G1 2-S1~~		~~G0 X-S0~~	
5		G2	X-S1	G2	1-S1	G2	2-S1	~~G2 X-S0~~		G3	X-S1
10		~~G0 1-S1~~		G3	2-S1	~~DO X-S0~~		~~DO X-S1~~		~~DO 1-S1~~	
15		DO	2-S1	DO	3-S1	~~G4 X-S0~~		~~G4 X-S1~~		G4	1-S1
20		~~G4 2-S1~~		~~G4 2-S1~~		G4	4-S1	~~G5 X-S0~~		G5	X-S1
25	MONITOR RELATED	G5	1-S1	G5	2-S1	G5	3-S1	G5	4-S1	~~G6 X-S0~~	
30		G6	X-S1	G6	1-S1	G6	2-S1	G6	3-S1	G6	4-S1
35		~~G7 X-S0~~		G7	X-S1	G7	1-S1	G7	2-S1	G7	3-S1
40		FFAILN	X-S0	~~FFAILN X-S1~~		~~SFAILN X-S0~~		~~SFAILN X-S1~~		~~SFAILN 1-S1~~	
45		SFAILN	2-S1	SFAILN	3-S1	~~LFAILN X-S0~~		LFAILN	X-S1	LFAILN	1-S1
50		LFAILN	2-S1	~~LFAILN 3-S1~~		~~J SAB X-S0~~		~~J SAB X-S1~~		K SAB	X-S0
55		K SAB	X-S1	~~C SAB X-S0~~		~~C SAB X-S1~~		~~R SAB X-S0~~		R SAB	X-S1
60		~~J SBC X-S0~~		J SBC	X-S1	K SBC	X-S0	~~K SBC X-S1~~		~~C SBC X-S0~~	
65		~~C SBC X-S1~~		~~R SBC X-S0~~		R SBC	X-S1	~~J SCA X-S0~~		J SCA	X-S1
70		K SCA	X-S0	~~K SCA X-S1~~		~~C SCA X-S0~~		~~C SCA X-S1~~		~~R SCA X-S0~~	
75		R SCA	X-S1	SCA	X-S1	~~SCA X-S0~~		SCA*	X-S1	SCA*	X-S0
80		SBC	X-S1	~~SBC X-S0~~		~~SBC* X-S1~~		SBC*	X-S0	~~SAB X-S1~~	
85		~~SAB X-S0~~		SAB*	X-S1	SAB*	X-S0				

* Indicates logical complement

Failures detected by first fail sequence

Additional failures detected by second fail sequence

TABLE A-10.—Continued

RUN 9

	First Fail Sequence	Second Fail Sequence
CH A	1	1
CH B	0	0
CH C	0	1

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.										
0	VOTER RELATED	~~G1 X-S0~~		G1 X-S1		G1 1-S1		~~G1 2-S1~~		~~G2 X-S0~~
5		G2 X-S1		G2 1-S1		G2 2-S1		~~G2 X-S0~~		~~G2 X-S1~~
10		~~G3 1-S1~~		G3 2-S1		~~DO X-S0~~		~~DO X-S1~~		DO 1-S1
15		DO 2-S1		~~DO 3-S1~~		~~G4 X-S0~~		G4 X-S1		G4 1-S1
20	MONITOR RELATED	G4 2-S1		G4 3-S1		~~G4 4-S1~~		~~G5 X-S0~~		~~G5 X-S1~~
25		~~G5 1-S1~~		G5 2-S1		G5 3-S1		~~G5 4-S1~~		~~G5 5-S0~~
30		G6 X-S1		G6 1-S1		G6 2-S1		G6 3-S1		G6 4-S1
35		~~G7 X-S0~~		G7 X-S1		G7 1-S1		G7 2-S1		G7 3-S1
40		FFAILN X-S0		~~FFAILN X-S1~~		~~SFAILN X-S0~~		~~SFAILN X-S1~~		SFAILN 1-S1
45		~~SFAILN 0-S1~~		SFAILN 3-S1		~~LFAILN X-S0~~		~~LFAILN X-S1~~		LFAILN 1-S1
50		LFAILN 2-S1		~~LFAILN 0-S1~~		~~J SAB X-S0~~		J SAB X-S1		K SAB X-S0
55		~~K SAB X-S1~~		~~K SAB 0-S0~~		~~K SAB 0-S0~~		~~K SAB 0-S0~~		R SAB X-S1
60		~~J SBC X-S0~~		J SBC X-S1		K SBC X-S0		K SBC X-S1		~~K SBC X-S0~~
65		~~K SBC X-S1~~		~~R SBC X-S0~~		R SBC X-S1		~~K SCA X-S0~~		J SCA X-S1
70		K SCA X-S0		~~K SCA X-S1~~		~~K SCA X-S0~~		~~K SCA X-S1~~		~~K SCA 0-S0~~
75		R SCA X-S1		SCA X-S1		~~SCA X-S0~~		SCA* X-S1		SCA* X-S0
80		~~SBC X-S1~~		~~SBC X-S0~~		~~SBC* X-S1~~		~~SBC* X-S0~~		SAB X-S1
85		~~SAB 0-S0~~		SAB* X-S1		SAB* X-S0				

* Indicates logical complement

~~Failures detected by first fail sequence~~

Additional failures detected by second fail sequence

TABLE A-10.—Continued

RUN 10

	First Fail Sequence	Second Fail Sequence
CH A	1	1
CH B	0	1
CH C	0	0

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.										
0	VOTER RELATED	~~G1~~	~~X-S0~~	~~G1~~	~~X-S1~~	G1	1-S1	~~G1~~	~~2-S1~~	~~G2~~	~~X-S0~~
5		G2	X-S1	G2	1-S1	G2	2-S1	~~G2~~	~~X-S0~~	G3	X-S1
10		~~G3~~	~~1-S1~~	G3	2-S1	~~DO~~	~~X-S0~~	~~DO~~	~~X-S1~~	~~DO~~	~~1-S1~~
15		DO	2-S1	DO	3-S1	~~G4~~	~~X-S0~~	G4	X-S1	G4	1-S1
20	MONITOR RELATED	G4	2-S1	G4	3-S1	G4	4-S1	~~G5~~	~~X-S0~~	~~G5~~	~~X-S1~~
25		~~G5~~	~~1-S1~~	G5	2-S1	G5	3-S1	~~G5~~	~~4-S1~~	~~G6~~	~~X-S0~~
30		G6	X-S1	G6	1-S1	G6	2-S1	G6	3-S1	G6	4-S1
35		~~G7~~	~~X-S0~~	G7	X-S1	G7	1-S1	G7	2-S1	G7	3-S1
40		FFAILN	X-S0	~~FFAILN~~	~~X-S1~~	~~SFAILN~~	~~X-S0~~	~~SFAILN~~	~~X-S1~~	SFAILN	1-S1
45		~~SFAILN~~	~~0-S1~~	SFAILN	3-S1	~~LFAILN~~	~~X-S0~~	~~LFAILN~~	~~X-S1~~	LFA.LN	1-S1
50		LFAILN	2-S1	~~LFAILN~~	~~0-S1~~	~~J SAB~~	~~X-S0~~	J SAB	X-S1	K SAB	X-S0
55		~~K SAB~~	~~X-S1~~	~~C SAB~~	~~X-S0~~	~~C SAB~~	~~X-S1~~	~~R SAB~~	~~X-S0~~	R SAB	X-S1
60		~~J SBC~~	~~X-S0~~	~~J SBC~~	~~X-S1~~	K SBC	X-S0	K SBC	X-S1	~~C SBC~~	~~X-S0~~
65		~~C SBC~~	~~X-S1~~	~~R SBC~~	~~X-S0~~	R SBC	X-S1	~~R SBC~~	~~X-S0~~	J SCA	X-S1
70		K SCA	X-S0	~~K SCA~~	~~X-S1~~	~~C SCA~~	~~X-S0~~	~~C SCA~~	~~X-S1~~	~~R SCA~~	~~X-S0~~
75		R SCA	X-S1	SCA	X-S1	~~SCA~~	~~X-S0~~	SCA*	X-S1	SCA*	X-S0
80		~~SBC~~	~~X-S1~~	~~SBC~~	~~X-S0~~	~~SBC*~~	~~X-S1~~	~~SBC*~~	~~X-S0~~	SAB	X-S1
85		~~SAB~~	~~X-S0~~	SAB*	X-S1	SAB*	X-S0				

* Indicates logical complement

Failures detected by first fail sequence

Additional failures detected by second fail sequence

TABLE A-10.—Continued

RUN 11

	First Fail Sequence	Second Fail Sequence
CH A	0	0
CH B	1	1
CH C	0	1

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.										
0	VOTER RELATED	~~G1 X-S0~~		G1 X-S1		~~G1 1-S1~~		G1 2-S1		~~G2 X-S0~~
5		~~G2 X-S1~~		G2 1-S1		~~G2 2-S1~~		~~G3 X-S0~~		G3 X-S1
10		G3 1-S1		G3 2-S1		~~DO X-S0~~		~~DO X-S1~~		DO 1-S1
15		~~DO 0-S1~~		DO 3-S1		~~G4 X-S0~~		G4 X-S1		G4 1-S1
20		G4 2-S1		G4 3-S1		G4 4-S1		~~G5 X-S0~~		G5 X-S1
25	MONITOR RELATED	G5 1-S1		G5 2-S1		G5 3-S1		G5 4-S1		~~G6 X-S0~~
30		~~G6 X-S1~~		~~G6 1-S1~~		G6 2-S1		G6 3-S1		~~G6 4-S1~~
35		~~G7 X-S0~~		G7 X-S1		G7 1-S1		G7 2-S1		G7 3-S1
40		FFAILN X-S0		~~FFAILN X-S1~~		~~SFAILN X-S0~~		~~SFAILN X-S1~~		SFAILN 1-S1
45		SFAILN 2-S1		~~SFAILN 0-S1~~		~~LFAILN X-S0~~		LFAILN X-S1		LFAILN 1-S1
50		LFAILN 2-S1		~~LFAILN 0-S1~~		~~J SAB X-S0~~		J SAB X-S1		K SAB X-S0
55		~~K SAB X-S1~~		~~G SAB X-S0~~		~~G SAB X-S1~~		~~R SAB X-S0~~		R SAB X-S1
60		~~J SBC X-S0~~		J SBC X-S1		K SBC X-S0		~~K SBC X-S1~~		~~G SBC X-S0~~
65		~~G SBC X-S1~~		~~R SBC X-S0~~		R SBC X-S1		~~J SCA X-S0~~		~~J SCA X-S1~~
70		K SCA X-S0		K SCA X-S1		~~G SCA X-S0~~		~~G SCA X-S1~~		~~R SCA X-S0~~
75		R SCA X-S1		~~SCA X-S1~~		~~SCA X-S0~~		SCA* X-S1		SCA* X-S0
80		SBC X-S1		~~SBC X-S0~~		~~SBC* X-S1~~		SBC* X-S0		SAB X-S1
85		~~SAB X-S0~~		SAB* X-S1		SAB* X-S0				

* Indicates logical complement

Failures detected by first fail sequence

Additional failures detected by second fail sequence

TABLE A-10.—Continued

RUN 12 — First Fail Sequence

	First Fail Sequence	Second Fail Sequence
CH A	0	1
CH B	1	1
CH C	0	0

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.								
0	VOTER RELATED	~~G1 X-S0~~	~~G1 X-S1~~	~~G1 1-S1~~	G1 2-S1	~~G2 X-S0~~
5		G2 X-S1	G2 1-S1	~~G2 2-S1~~	~~G3 X-S0~~	G3 X-S1
10		G3 1-S1	G3 2-S1	~~DO X-S0~~	~~DO X-S1~~	~~DO 1-S1~~
15		DO 2-S1	DO 3-S1	~~G4 X-S0~~	G4 X-S1	G4 1-S1
20		G4 2-S1	G4 3-S1	G4 4-S1	~~G5 X-S0~~	G5 X-S1
25		G5 1-S1	G5 2-S1	G5 3-S1	G5 4-S1	~~G6 X-S0~~
30		~~G6 X-S1~~	~~G6 1-S1~~	G6 2-S1	G6 3-S1	~~G6 4-S1~~
35		~~G7 X-S0~~	G7 X-S1	G7 1-S1	G7 2-S1	G7 3-S1
40	MONITOR RELATED	FFAILN X-S0	~~FFAILN X-S1~~	~~SFAILN X-S0~~	~~SFAILN X-S1~~	SFAILN 1-S1
45		SFAILN 2-S1	~~SFAILN 3-S1~~	~~LFAILN X-S0~~	LFAILN X-S1	LFAILN 1-S1
50		LFAILN 2-S1	~~LFAILN 3-S1~~	~~J SAB X-S0~~	J SAB X-S1	K SAB X-S0
55		~~K SAB X-S1~~	~~C SAB X-S0~~	~~C SAB X-S1~~	~~R SAB X-S0~~	R SAB X-S1
60		~~J SBC X-S0~~	J SBC X-S1	K SBC X-S0	~~K SBC X-S1~~	~~C SBC X-S0~~
65		~~C SBC X-S1~~	~~R SBC X-S0~~	R SBC X-S1	~~J SCA X-S0~~	~~J SCA X-S1~~
70		K SCA X-S0	K SCA X-S1	~~C SCA X-S0~~	~~C SCA X-S1~~	~~R SCA X-S0~~
75		R SCA X-S1	~~SCA X-S1~~	~~SCA X-S0~~	SCA* X-S1	SCA* X-S0
80		SBC X-S1	~~SBC X-S0~~	~~SBC* X-S1~~	SBC* X-S0	SAB X-S1
85		~~SAB X-S0~~	SAB* X-S1	SAB* X-S0		

* Indicates logical complement

Failures detected by first fail sequence

Additional failures detected by second fail sequence

TABLE A-10.—Continued

RUN 13

	First Fail Sequence	Second Fail Sequence
CH A	0	0
CH B	0	1
CH C	1	1

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.												
0	VOTER RELATED	~~G1 X-S0~~		G1 X-S1		G1 1-S1		G1 2-S1		~~G2 X-S0~~	
5		~~G2 X-S1~~		~~G2 1-S1~~		G2 2-S1		~~G2 X-S0~~		G3 X-S1	
10		G3 1-S1		~~G3 2-S1~~		~~DO X-S0~~		~~DO X-S1~~		DO 1-S1	
15		~~DO 3-S1~~		DO 3-S1		~~G4 X-S0~~		~~G4 X-S1~~		~~G4 1-S1~~	
20		G4 2-S1		G4 3-S1		~~G4 4-S1~~		~~G5 X-S0~~		G5 X-S1	
25	MONITOR RELATED	G5 1-S1		G5 2-S1		G5 3-S1		G5 4-S1		~~G6 X-S0~~	
30		G6 X-S1		G6 1-S1		G6 2-S1		G6 3-S1		G6 4-S1	
35		~~G7 X-S0~~		G7 X-S1		G7 1-S1		G7 2-S1		G7 3-S1	
40		FFAILN X-S0		~~FFAILN X-S1~~		~~SFAILN X-S0~~		~~SFAILN X-S1~~		~~SFAILN 1-S1~~	
45		SFAILN 2-S1		SFAILN 3-S1		~~LFAILN X-S0~~		LFAILN X-S1		LFAILN 1-S1	
50		LFAILN 2-S1		~~LFAILN 3-S1~~		~~J SAB X-S0~~		~~J SAB X-S1~~		K SAB X-S0	
55		K SAB X-S1		~~K SAB X-S0~~		~~K SAB X-S1~~		~~R SAB X-S0~~		R SAB X-S1	
60		~~J SBC X-S0~~		J SBC X-S1		K SBC X-S0		~~K SBC X-S1~~		~~K SBC X-S0~~	
65		~~K SBC X-S1~~		~~R SBC X-S0~~		R SBC X-S1		~~J SCA X-S0~~		J SCA X-S1	
70		K SCA X-S0		~~K SCA X-S1~~		~~K SCA X-S0~~		~~K SCA X-S1~~		~~R SCA X-S0~~	
75		R SCA X-S1		SCA X-S1		~~SCA X-S0~~		SCA* X-S1		SCA* X-S0	
80		SBC X-S1		~~SBC X-S0~~		~~SBC* X-S1~~		SBC* X-S0		~~SAB X-S1~~	
85		~~SAB X-S0~~		SAB* X-S1		SAB* X-S0					

* Indicates logical complement

Failures detected by first fail sequence

Additional failures detected by second fail sequence

TABLE A-10.—Concluded

RUN 14

	First Fail Sequence	Second Fail Sequence
CH A	0	1
CH B	0	0
CH C	1	1

Monitoring: DO, FFAILN, SFAILN, LFAILN

FAULT NO.		First Fail Sequence				Second Fail Sequence			
ø	VOTER RELATED	~~G1~~	~~X-Sø~~	G1	X-S1	G1	1-S1		
5		G2	X-S1	~~G2~~	~~1-S1~~	G2	2-S1	~~G2~~	~~X-Sø~~
1ø		G3	1-S1	~~G3~~	~~2-S1~~	~~DO~~	~~X-Sø~~	~~DO~~	~~X-S1~~
15		DO	2-S1	~~DO~~	~~3-S1~~	~~G4~~	~~X-Sø~~	~~G4~~	~~X-S1~~
2ø		G4	2-S1	G4	3-S1	~~G4~~	~~4-S1~~	~~G5~~	~~X-Sø~~
25	MONITOR RELATED	G5	1-S1	G5	2-S1	G5	3-S1	~~G5~~	~~1-S1~~
3ø		G6	X-S1	G6	1-S1	G6	2-S1	G6	3-S1
35		~~G7~~	~~X-Sø~~	G7	X-S1	G7	1-S1	G7	2-S1
4ø		FFAILN	X-Sø	~~FFAILN~~	~~X-S1~~	~~SFAILN~~	~~X-Sø~~	~~SFAILN~~	~~X-S1~~
45		SFAILN	2-S1	SFAILN	3-S1	~~LFAILN~~	~~X-Sø~~	LFAILN	X-S1
5ø		LFAILN	2-S1	~~LFAILN~~	~~3-S1~~	~~J SAB~~	~~X-Sø~~	~~J SAB~~	~~X-S1~~
55		K SAB	X-S1	~~G SAB~~	~~X-Sø~~	~~G SAB~~	~~X-S1~~	~~R SAB~~	~~X-Sø~~
6ø		~~J SBC~~	~~X-Sø~~	J SBC	X-S1	K SBC	X-Sø	~~K SBC~~	~~X-S1~~
65		~~G SBC~~	~~X-S1~~	~~R SBC~~	~~X-Sø~~	R SBC	X-S1	~~J SCA~~	~~X-Sø~~
7ø		K SCA	X-Sø	~~K SCA~~	~~X-S1~~	~~G SCA~~	~~X-Sø~~	~~G SCA~~	~~X-S1~~
75		R SCA	X-S1	SCA	X-S1	~~SCA~~	~~X-Sø~~	SCA*	X-S1
8ø		SBC	X-S1	~~SBC~~	~~X-Sø~~	~~SBC*~~	~~X-S1~~	SBC*	X-Sø
85		~~SAB~~	~~X-Sø~~	SAB*	X-S1	SAB*	X-Sø		

* Indicates logical complement

Failures detected by first fail sequence

Additional failures detected by second fail sequence

TABLE A-11.–FAILURE MODES – SYSTEM STATUS LOGIC

Item description	Failure type	Detection method (no stimuli)	Test required	Remarks
N1	High	Causes first failure indication	None	Should be failed serially to show each can light L1
	Low	None	First failure on F2 or F5 or F7	
B1	High	Causes first failure indication	None	
	Low	None	First failure on F2 or F5 or F7	
B4	High	None	First failure on F2 or F5 or F7	
	Low	Causes first failure indication	None	
N5	High	Causes first failure indication	None	
	Low	None	First failure on all inputs to N5	Should be failed serially to show each can light L1
A1	High	None	Any first failure	
	Low	Causes first failure indication	None	
G1	High	None	Any first failure	
	Low	Causes first failure indication	None	
Lamp	Open	None	Any first failure will test light	
+28 V	Open	None	Any first failure will test 28 V	ICP 723 not triply redundant in this area
Logic string		Second failure		Same as 1st fail results
Logic		Local failure		Not analyzed–maintenance only

TABLE A-12.–SSFD FMECA RESULTS

Section	Total elements in circuit	Latent failure possibility	Comments
Input data and data holding registers	32	0	
Median logic	28	11	Latent failures are in circuits associated with providing difference signals for failure monitor
Signal selection control	39	5	Latent failures are associated with circuits which become active after first failure
Average of two generators	13	13	Latent failure possibilities could be reduced to two if any force average were commanded
Data selection and line drivers	5	0	
Sensor monitor filters	66	66	
Failure counter and failure memory	16	66	
Failure count control	8	8	
Sensor failure monitor	20	20	
Failure logic	45	45	Latent failures could be reduced to 25 if any force average were commanded

TABLE A-13.—INPUT/OUTPUT CIRCUITRY AND TIMING FAILURE MODES

Channel failure item	Failure mode	Failure effect and classification	Means of detection or preflight test
1.0 CIU inputs			
1.1 AC inputs ● Input ampl	Open feedback	± maximum output for ± small inputs. Will cause aircraft limit cycle if signal selection is not hard vote and signal operates about zero. (active, conditionally detectable)	Normal failure monitoring and signal selection. Oscilatory failure detector may be beneficial for inflight monitoring.
	Open input	Signal stays at zero. Failure will be detected by normal monitoring only if good signals differ from zero by more than the detection level for longer than failure latching delay time. (passive, conditionally detectable)	Test by exercising each input above detection level.
● Demod	Half wave rectification	Demod gain goes to about half normal value. Detection by sensor signal comparison requires signal excursion greater than twice detection level. Also, increased carrier frequency noise on demod output may cause beat frequency on sampled output. (undetermined, conditionally detectable)	Test by exercising signals to ± full excursion.
	Loss of reference	Loss of sensor signal will be detected by cross-channel comparison only if signal motion is greater than detection level. (passive)	Same as above
● Filter	Open capacitor	Loss of filter roll off, i.e., less filter, increased noise to A/D. Increased allasing affect may cause signal oscillation. (indecisive, probably passive)	Signal comparison monitoring may not detect failure unless oscillatory failure monitor is included. Should be investigated in laboratory. Test may require high frequency sensor excitation if no OFD is provided.
1.2 DC inputs		Typically same as ac inputs if upstream failures are considered, i.e., dc inputs may have demods upstream.	Same as items under 1.1
1.3 32-channel MUX	Fails to switch channel	Output = 0 (passive)	Verify correct transmission from sensor to memory or sensor monitor.
	Switches to wrong channel	Wrong data (indeterminant)	Same as above
	Fails to isolate a channel	Will cause erroneous information on several channels (indeterminant)	Same as above

TABLE A-13.—Continued

Channel failure item	Failure mode	Failure effect and classification	Means of detection or preflight test
1.4 A/D converter	An output bit (from F/F) fails high or low	MSB will make the A/D conversion a half-wave rectifier for all analog inputs. Any other bit will make answer wrong when bit is wrong. Error will be related to significance of bit and will cause other bits (lower and 1 higher) to be wrong in conversion process. (may be active or passive depending on bit location)	Verify that digital output word can be set to values of 0 and - 1.
	Time gate signal to F/F fails high	Output bit continuously sets and clears during convert period which causes lower order bits to falsely clear if failed bit is supposed to be clear. If failed bit is supposed to be set, output may be correct if LSB equals one. (indeterminant)	Signal monitoring will detect if failed bit has sufficient significance to trip monitor, and if LSB equals zero. For preflight test, verify monitoring will not indicate failure with test design offset tolerance and LSB equals zero.
	Time gate signal to F/F fails low for T_i, but not at T_o	Bit with failed gate and many lower bits will be wrong if bit was supposed to be high. If bit was supposed to be low, then correct answer will result (indeterminant)	Test that all bits will set high and low.
	Time gate signal to F/F fails low for T_i and T_o	F/F won't change states	Test that all bits will set high and low.
	Time gate signal fails low at T_o, but not at T_i	F/F won't clear at T_o. Next more significant bit and several less significant bits may be set wrong if failed bit is supposed to be zero but was previously set to one. Could cause apparent intermittent failure indications, or could be masked by time delay on monitoring; 25% of conversions will be wrong. (indeterminant)	Test 4 to 6 preselected conversion values to check out most of A/D. Values will be selected to check as many of the conversion bits as possible.
	Open line to resistor network but not open to digital output	Failed bit always sets high; lower order bits will be set high if failed bit is supposed to be set high. (indeterminant)	Normal failure monitoring if bit has sufficient significance to cause detection.

TABLE A-13.—Continued

Channel failure item	Failure mode	Failure effect and classification	Means of detection or preflight test
1.5 A/D register	Open input line	Bit loads as zero; error will be proportional to bit significance. (indeterminant)	Normal monitoring will detect if bit has sufficient significance. Preflight test should verify that monitoring does not trip with design tolerances.*
	Zero or one bit failure in serial shift register	All bits upstream of failure would shift out as zero or one. This would cause the serial output to have a sawtooth relationship to the parallel input if the failure occurs in bit 12 or less. Failures above bit 12 can cause the output to exceed scale values. (indeterminant)	Failure will be actively detected by normal signal monitoring if significance of failed bit exceeds monitoring detection level. If bit significance is small, then the signal excursion will have to exceed the detection level minus the bit significance to assure detection. System test by exercising signal beyond detection level.
	Bit always loads high	Error will be proportional to bit significance if bit was supposed to be low. (indeterminant)	Same as above
1.6 Discrete inputs	Any	Discrete won't switch properly. (detectable)	Failure voted out and detected by input voters if signal is to switch in operation. Test by putting all discretes low or high, and verifying each will switch high or low.
1.7 Serial gating			
● Timing gate upstream of parity generator or shift CLK	Fails high or low	Data will pass as all zeros i.e., zero or all ones minus (LSB). If the signal is nominally zero, the failure will not be detected unless the good signals have an excursion greater than the detection level. (passive)	Test by exercising any channel greater than failure detection level. Verify no failure indications. Must first check out failure monitoring.
● Timing gate or line driver downstream of parity generator	Fails high or low	All bits pass as high or low including parity bits. (detectable)	Parity check receiver will detect failure
● Parity check generator	F/F won't toggle	Incorrect parity on interchannel data transfers. (detectable)	Normal failure monitoring.

*Design tolerance is buffer tolerance between what normally exists for sensors and what is considered safe for flight.

TABLE A-13.—Continued

Channel failure item	Failure mode	Failure effect and classification	Means of detection or preflight test
2.0 CIU outputs 2.1 Data line from computer	Open, high, or low	None except channel failure; MLV votes out failure effects. (excluded and annunciated)	Normal failure monitoring—test by verifying signal transmits from computer to output with no failure indications.
2.2 Bit buffer	Fails to toggle	1) Analog output registers would load as all zeros or all ones corresponding to zero or minus LSB will probably be passive failure if nominal signal is zero and normal excursions are small. (probably passive) 2) All discrete outputs would either be in high or low state.	1) command at least one analog output in all three channels to change more than a down stream failure detection level and verify that change occurs by end around check loop, or do item 2 below. 2) Depends on how discretes are used. Can verify that at least one discrete can be set high or low.
2.3 Discrete outputs ● Data line	Open or high	Loads all discretes high or low.	Verify last discrete in upper and lower register can be switched.
● Output discrete register	Zero or one bit fails in serial shift process	All downstream bits shift in as zero or one.	Test by verifying that the last bit in each of the registers can be switched.
	Dead clock signal	Discretes won't change, i.e., shift regist. won't shift.	Test by verifying that the last bit in each of the registers can be switched.
	Clocks at wrong time	Loads wrong data into discrete register.	Verify that all discretes can be switched high or low.
	Zero or one bit fails in serial shift process	All downstream bits shift in as zero or one.	Same as for data line.
● Quad latch	High or low	Discretes won't change.	Verify that all discretes can be switched high or low.
2.4 Analog outputs ● Overflow detector	Won't detect overflow	No overflow detection. If scaling allows overflow beyond 12 bits, then output will be erratic relative to desired value.	Detect by downstream voters (e.g., ECS) test by forcing ± overflow in all three channels, one bit at a time for bits 11, or 12, 13, and 14 while verifying that maximum output occurs with no down stream failure indications.

TABLE A-13.—Continued

Channel failure item	Failure mode	Failure effect and classification	Means of detection or preflight test
	Logic failure that indicates erroneous overflow	Would cause a ± maximum output on all analog signals in failed channel.	Detected by external downstream failure monitoring.
	Logic failure that outputs zero or one	Causes zero or minus LSB output on all analog outputs.	Same as 2.2. Propose having at least one analog output that operates more than detection. level away from zero.
	Shift regular bit fail high or low	If failure is in bit 10 or lower then output would be zero or minus LSB unless input data overflow occurs. If failure is in bits 11 through 15 then output may be Imaximum value. Failure effects all outputs of failed path.	Detection by external downstream monitoring depends on normal signal excursions exceeding detection levels when output fails to zero. Test by verifying no failure indications for large signal excursions.
● D/A register	Won't shift or CLK line failed	All analog outputs in failed channel put out same constant value (can be any value).	Downstream failure monitoring. Test by exercising analog outputs beyond failure detection levels.
	Regular bit fails fixed high or low	Failed bit and all less significant bits would output output fixed values of one or zero. Would cause stair-step output. (indeterminant)	Downstream monitoring or end around check on single output. Preflight test not required if continuous end around test provided, otherwise see above.
● DEMUX	Time signals to or time gate failures within DEMUX	One or more sample holds load at two or more time intervals; thus loading wrong data and holding wrong data for most of output time.	Can only detect with downstream monitoring, or use end around check on every output line in system test.
● D/A	MSB input failed low or high	D/A output forced to remain in negative or positive output region. When signal tries to cross zero going positive or negative the output jumps from zero to hardover negative or positive output. Failure is simultaneously common to all analog outputs. For nominal zero signal, the output tends to jump back and forth between zero and negative or positive hardover. Could easily cause limit cycle oscillation if downstream servos are not equalized with hard vote.	Downstream monitoring should easily detect hardover jumps if limit cycle oscillation is not above frequency for detection time delay. Oscillatory Failure Detector (OFD) may be needed to detect failure. System test by verifying peak-to-peak output control. Do with end around test.

TABLE A-13 (Concluded)

Channel failure item	Failure mode	Failure effect and classification	Means of detection or preflight test
	Any D/A input bit except MSB failed low	Output glitches occur which are proportional to significance of failed bit. Glitches can be as high as 50% of 0 to full-scale value. Failure is simultaneously common to all analog outputs. Failure may cause limit cycle problems by causing jumps near zero.	If bit significance exceeds downstream monitoring level, the failure should be easily detectable. If bit significance does not exceed detection level, then failure may be difficult to detect inflight.
● Sample hold	Won't sample	A single output slowly decays to zero.(probably passive)	Test by exercising all outputs beyond downstream detection levels.
	Won't hold	Output rapidly decays toward zero after each sample. Will cause oscillatory effect each time output transitions away from zero. (probably passive)	Downstream monitoring. OFD in downstream monitoring would enhance detection.
3.0 Input/output timing			
3.1 Waveshaping circuit			
● Square clock	Fails high or low	All functions stop. (detectable)	Normal failure monitoring.
● CLK	Fails high or low	Bit CLK and zero value outputs from sensors. (detectable)	Same as above
● Bit CLK	Fails high or low	All function stop. (detectable)	Same as above
● Strobe	Fails high or low	All peripheral output functions stop: D/A overflow detection; D/A register; discrete output register; T-reset; cross-channel parity. (detectable)	Same as above
● Strobe NAND gate	Any input high	Will advance leading edge 400 ns or delay trailing edge 400 ns of Strobe. Race problem will exist. (probably detectable)	Any cross-channel data voter.
● CLK NAND gate	Any input high	Will advance leading edge 400 ns or delay trailing edge 400 ns of CLK and Bit Clock. Time race problems with basic timing of LRU. (probable detectable)	Same as above
3.2 Bit time shift register	Any bit failure	All functions stop. (detectable)	Normal failure monitoring.
3.3 Word counters	Count stop or count	Miscoordinate processing. (detectable)	Same as above

TABLE A-14.—COMPUTER FMECA

Failure item	Failure mode	Failure effect	Detection						Remarks
			RAM	ROM	Parity	Overflow	Output monitor (downstream)	None	
1.0 Incremental input/output									
1.1 Increment selector	Selects improper value.	Data passing through computation will not track other channels.					X		May require several iterations before effects of input data error are detected.
1.2 X/Y register	Drops or adds a bit.	Improper data out.			X		X		Output monitor will catch error only if error occurs in a "Y" time slot dedicated to servo outputs.
	Fails to shift.	No data out.			X		X		
1.3 Whole word register	Any.							X	Register not used.
1.4 Mode switching f(timing)									
• Btw Y variable and Y discrete.	Stuck any position.	Improper data sent to CIU.					X		
• Btw X variable and Y variable.	Stuck at X variable.	No output.					X		
	Stuck at Y variable.	Open feedback loop on X register.				X	X		
• Btw reset and $\Delta Y/\Delta X$	Stuck at zero.	No effective output and no nulling of sensor data for use in computation.				X	X		
	Stuck at $\Delta Y/\Delta X$.	No initializing of register.					X	X	This error will go undetected if it occurs after the last successful reset. During the next reset (after failure) residual values in the register will trip the output monitor in the interface unit.
• Btw whole word input and save	Stuck any position.							X	Register not used.
• Btw X_{i-1}/Y_i and whole word	Stuck at X_{i-1}/Y_i	Normal mode of operation.						X	Don't care type of failure.
	Stuck at whole word.	All X register values will be wrong.				X	X		All X-register data will be the last data pattern entered, such as random values at power turn-on.
1.5 $\Delta Y/\Delta X$, X_{i-2}/Y_{i-1} adder	Adds incorrectly.	X/Y register contains erroneous data.				X	X		May require several iterations for overflow monitor to trip.
2.0 Program memory									
2.1 Address decoding	Improper address.	Erroneous instruction bits (control and/or variables) utilized in ρ equation.		X		X	X		Most prevalent detector would be the parity check on all bits from ROM to be used during an algorithm. The other monitors may trip depending upon the nature of the reaction of the ρ equation to the erroneous bits.
2.2 ROM	Drop or adding bits.	Same as above		X		X	X	X	Same as above. In the case of a bit failure in an instruction which resides in branch loop not normally executed, a latent possibility exists. As soon as the loop is executed, the ROM parity detector should flag the failure.
2.3 Parallel data buss	Any bit position stuck to a one or a zero.	Improper variable data and control mode bits used by ρ equation.		X		X	X		Most prevalent detector would be ROM parity check.

TABLE A-14.—Continued

Failure item	Failure mode	Failure effect	Detection						Remarks
			RAM	ROM	Parity	Overflow	Output monitor (downstream)	None	
2.4 Shift registers U_o/V_o register	Fails to shift or shifts improper data.	Proper initial conditions would not be set.		X					This failure would have no computational effect if it occurs after a successful reset.
• S_p/S_q register	Same as above	Scaling factors for ΔZ incorrect.		X		X	X		
3.0 Increment storage									
3.1 Address decoding	Improper address.	Data stored/read from location other than anticipated.			X	X	X		Errors may be self-correcting. The same decoding error would exhibit itself during both read and write operations. Thus, data stored in the incorrect location will be retrieved from the same incorrect location. However, if the decoding error is one of an address bit sticking at a one or zero, multiple use of the same storage cell will result. This error will manifest itself by an overflow or output error only if the first set of data is not read out before an additional set is read in during the same frame time.
3.2 RAM	Data value stored incorrectly.	Data on input and output buss will not match.	X			X	X		Nearly all of the internally circuitry is tested under normal use (input buss, output drivers). Any input sensor changing $>$ ±1.5% of its full range value per iteration will place bits in each position of the 4-bit data buss. The only possible latent would be the unique bit storage location for a particular word whose bit significance is not cycled in a normal course of handling data.
3.3 Mode switching f(timing)									
• Btw read and write	Stuck at write.	Data utilized in ρ equation not coming from outputs as specified by programming but only as a function of word counter.				X	X		
	Stuck at read.	Data stored at locations specified during Y and X branch address.	X				X		Address of RAM would change between data write command and data check. Data on output buss would be different than on input buss.
• Btw output data and address buss.	Stuck at output	Variable data substituted for constants anticipated by programming.				X	X		May require several iterations to manifest error.
	Stuck at address	Data set to ρ equation are static.				X	X		Same as above.

TABLE A-14.—Continued

Failure item	Failure mode	Failure effect	Detection						Remarks
			RAM	ROM	Parity	None	Output monitor (downstream)		
4.0 Timing									
4.1 Basic timing • Wave shaping • Bit shift register • Word counter	See section 3.0 in table C-9								
• Combinational logic	Any failure.	Develops improper f(timing) causing out-of-sequence processing.					X		Internal failure monitoring may be inoperative since the f(timing) in those circuits may be bad.
4.2 Address generators • Shift register	Drops or adds a bit.	Improper data out.					X		Branch detector will trigger reset of registers.
	Fails to shift.	Processing stops; same algorithm is repeated over and over.					X		
	Loads branch addresses continuously.	Erratic processing				X	X		Branch detector will trigger reset of registers.
• Incrementor	Adds incorrectly.	Same as above				X	X		Same as above
• Latch/buffer	Any bit failure.	Same as above				X	X	X	Latent possibility only in higher order bit if branching is not normally encountered.
• Counter	Any bit failure.	Same as above		X		X	X		
4.3 Branching circuits • Detector	Output fails to off.	No initial conditions upon branching.				X	X	X	Potential latent if branching is not normally exercised. After branching, the ring registers would not initialize.
	Output fails to on.	Registers reset during each algorithm.					X		Outputs would be fixed.
• Decoder	Improper output.	Register(s) reset during each algorithm.					X		
4.4 Mode switching—f(timing) • Btw register parity and instruction address	Stuck to parity. Stuck to address.	Same as above No register parity bit inserted into ρ register.			X		X		Output would be fixed.

Table A-14—Continued

Failure Item	Failure mode	Failure effect	Detection						Remarks
			RAM	ROM	Parity	Overflow	Output monitor (downstream)	None	
5.0 Arithmetic unit									
5.1 Mode control latch	Output failures.	Improper moding in ρ equation.				X	X	X	There are a number (8) of these settings that were unused during FCD task programs. Failures to these positions represent "don't care" conditions. Any other miss-moding should eventually show up at the output monitor.
5.2 Shift registers									
• U/V or ρ	Drops or adds a bit.	Improper data out.			X	X	X		
	Fails to shift.	ΔZ outputs in error.				X	X		May require several iterations before failure flagged.
o Extend	Any.	None.						X	Register not used.
5.3 Adders									
• $\Delta U/\Delta V$, U_i/V_i	Adds incorrectly.	U_i/V_i register receives incorrect data.				X	X		
• ρ_{i-1}, $\rho_{i_{(t-1)}}$	Same as above.	ρ_i incorrect.				X	X		Overflow would most likely occur in another register during a different algorithm time.
• ρ_{i-1}, $\Delta\rho_i$	Same as above.	Same as above.				X	X		
• $S_p\Delta P/S_q\Delta Q$ $U\Delta T/V\Delta W$	Same as above.	Same as above				X	X		
5.4 Multipliers									
• $\Delta T/\Delta W$, U_i/V_i	Multiplies incorrectly.	Same as above.				X	X		
• $\Delta P/\Delta Q$, S_p/S_q	Same as above.	Same as above.				X	X		
5.5 Divider	Divides incorrectly.	Wrong value to increment selector.				X	X		May require several iterations for error to be flagged.
5.6 Increment selector	Any bit stuck high.	Passes incorrect value into RAM.				X	X		
	Any bit stuck low.	Same as above.					X	X	Potentially latent in that the MSB of the coded data word may not be required during normal control law signal perturbation ; at least a value of eight is required.
5.7 Mode switching f(programming)									
• Btw U_i/V_i, U_{i-1}/V_{i-1}, and zero.	Stuck in any position.	Incorrect data into ρ equation.					X		Doubtful that any overflow would be generated. May require several iterations to propagate error to output.
• Btw S_p/S_q and X/Y	Same as above.	Same as above.				X	X		Overflow detection may depend on magnitude of the constants S_p and S_q.
• Btw ρ_{i-1} and zero	Same as above.	Same as above.				X	X		Overflow will flag stuck to ρ_{i-1}.

Table A-14—Concluded

Failure item	Failure mode	Failure effect	Detection						Remarks
			RAM	ROM	Parity	Overflow	Output monitor (downstream)	None	
• Btw $\rho_{i_{(T-1)}}$ and zero	Stuck to zero.	None.						X	Not used for FCD Task. If this register had been used, it would be flagged by output monitor.
	Stuck to $\rho_{i_{(T-1)}}$	Incorrect data into ρ equation.				X	X		
• Btw S_q and V_i	Stuck in any position.	Incorrect data to increment selector.				X	X		
• Any switches that negate signal prior to multiplication	Stuck in any position.	Incorrect data into ρ equation.				X	X		
5.8 Mode switching f(timing, reset)									
• Btw U_o/V_o and U_i/V_i	Stuck at U_o/V_o.	U/V register does not update.					X		
	Stuck to U_i/V_i.	U/V register won't initialize.				X	X	X	Potentially latent until reset initiated.
• ρRegister MUX	Stuck any position.	ρ register does not obtain its full complement of data.			X		X		

APPENDIX B

ICPS PREFLIGHT TEST

The FMECA study results presented in appendix A show several design areas of the ICPS that contain undesirable failure mode effects. Notable among these was the fail-operative clock circuit. In this case, the circuit was redesigned, reevaluated, and a change implemented to remove the damaging failure mode. All other undesirable failure modes are latent in nature, where the associated circuits cannot be easily changed to circumvent the problem. These areas of the system must be tested to ensure operational integrity. This appendix presents the results of the limited preflight test laboratory investigations conducted with the ICPS, preceded by a discussion of the required tests on the MLV and its monitor, to establish that those circuits do not contain latent failures.

B.1 MAJORITY LOGIC VOTER MONITOR

In appendix A, it was pointed out that the processing of normal data (channel A = channel B = channel C) would not uncover certain failures within the MLV and its monitor. In order to clear such faults, all combinations of incoming data states would be required. A method to check the MLV, short of manipulating all input failure states, was proposed with the addition of the circuit elements illustrated in figure B-1. These elements allow the insertion of a fail-to-high condition on the channel A and B paths. The proposed built-in test electronics were evaluated to determine what test series would be required to detect previously undetected failure modes.

Applying the same failure mode analysis methods used in section A.2, the effectiveness of using the BIT elements to uncover the possible failure modes is tabulated in table B-1. (Nomenclature for this table is the same as used in tables A-9 and A-10.) This table was compiled by using simulation runs 1, 2, 10, and 12 from table A-10 (assuming no faults in the BIT circuits). These runs represent the MLV circuit operation with normal (nonfailed) data inputs and the first and second failure sequences where channels A and B fail high through the use of the BIT circuitry. Each of the potential failure modes that are detected are annotated in table B-1 by identifying the various conditions under which the failures were detected. The failure modes that can be detected through the processing of normal data (channel A = channel B = channel C) are crossed out with a thin line while those detected by the proposed BIT circuitry are crossed out with a heavy line. The failure modes in table B-1 that are shown closed in a rectangle are the never detected failure modes discussed in section A.2.3. The remaining undetected failure modes of concern have been circled by a squiggly line. As can be noted from the table, many of the undetected potential failures are still within the voter section of the design.

By reviewing the testing sequence made possible with the BIT circuitry, one can ascertain that there are four possible ways to force a first and second failure sequence by causing input failures to only two of the three channels. The proposed BIT circuitry, discussed above, failed channel A and B high. The three additional sequences are:

- Failed channel A high; failed channel B low (condition II)
- Failed channel A low; failed channel B high (condition III)
- Failed channel A low; failed channel B low (condition IV)

These three additional combinations were explored to see if they resulted in a more optimum BIT sequence. The results are condensed in tables B-2 through B-4. By viewing these results, it is obvious that none of the alternate BIT test conditions, uniquely or collectively, present a complete test. Therefore, it appears that the earlier analysis in appendix A requiring failure states to be placed on all data paths must be adhered to.

However, not all input data combinations need to be tested. If the extra redundancy links (those elements inside the rectangles in table B-1) are deleted either by removing one of the first two inputs to gate G-7 (see fig. A-4) or by removing one of the first two inputs into gate G-9, the minimum test sequence presented below will suffice.

Test 1	Failed channel A high; then second failed channel B high
	Failed channel B high; then second failed channel C high
	Failed channel C high; then second failed channel A high
or	
Test 2	Failed channel A low; then second failed channel B low
	Failed channel B low; then second failed channel C low
	Failed channel C low; then second failed channel A low

(Test 1 and 2 are complements of each other.) Thus, only three failure sequences of the possible six are required to totally check the MLV and its monitor.

B.2 LABORATORY ICPS PREFLIGHT TEST

An abbreviated preflight test of the ICPS was evaluated in the laboratory. The test could not be automated because two essential elements were not availabe. The first was the lack of an automatic test administrator. It is anticipated that such a device would supply the needed stimuli and process the resultant responses to ascertain the go/no-go status of the subsystem. To account for this shortcoming, the stimuli had to be manually induced. The second element missing was the BIT test circuits (like those mentioned above) to control and check the monitoring functions within the ICPS. Therefore, the laboratory test had to assume that the monitors were working even though a proper preflight test would include checking the monitors as a first step. This is required since the monitors are used to detect any improper response resulting from preflight test stimuli.

The test procedure developed assumed on-the-ground conditions and that all hydraulic and electrical power were on. System stimuli for the test sequence involved exercising the column and pitch-rate gyro signals to generate a triangular waveform input to the ICPS of ±4V at 0.2 Hz. The settings of the hardware SSFD parameters (illustrated in fig. 4-11) for the signals monitored are tabulated in table B-5. For each of the postulated failures in table B-6, the preflight test was cycled to check if the failure was detected by any ICPS monitor.

The results of the ICPS laboratory preflight-test tests are presented in table B-6. The following paragraphs summarize conclusions drawn from the test results, by failure category, as referenced to the test numbers of table B-6.

Sensor Input Failures (t sts 1 through 7)

Single-sensor passive failures are easily detected, and the appropriate faulty channel are identified. Two like passive-sensor failures will be detected, but the good channel will be identified as the local failure. All other input sensor failures were detected and identified.

Inter- and Intra-Cable Disconnects (tests 8 through 12)

Disconnected cables resulted in failure indications for all cases. As anticipated, system input stimuli were required to uncover data path failures to interfacing subsystems, where data path separations within the subsystem were immediately detected. Some cable separations produced a second failure indication. The system remained operational, but the failure annunciation panel indicated a second failure from one channel. This points up the need to vote second-failure failure status logic if it is going to be used to automatically shut down a channel or system.

Power Failures (tests 13 and 14)

Total power failure in any ICPS LRU is immediately detected. Since input power is dual, loss of a single power source did not result in either a system failure or a first fail light. The only indication of this condition was displayed by the computer mechanical BIT indicator on the LRU and the channel local failure annunciator.

Computer Memory Failures (tests 15 through 24)

Changes in program memory were hand inserted into various memory locations to simulate failures. These tests were included in order to obtain a sensitivity of the system monitoring to memory failures. In all cases, except one, the bit-by-bit output voter monitor detected the changes, and a first failure was annunciated. This was true even for test 15 where the large S_q term was changed from 4737 to 4738. Gyro torquing was necessary, however, to produce the bit difference required for the monitor to trip. The case that did not produce a first failure indication (test 16) related to a change in an algorithm not used in the operational program. This, however, produced no system failure indications and is considered a don't care failure. In all cases, the computer BIT circuitry detected a parity error and annunciated the ROM and local failure lights.

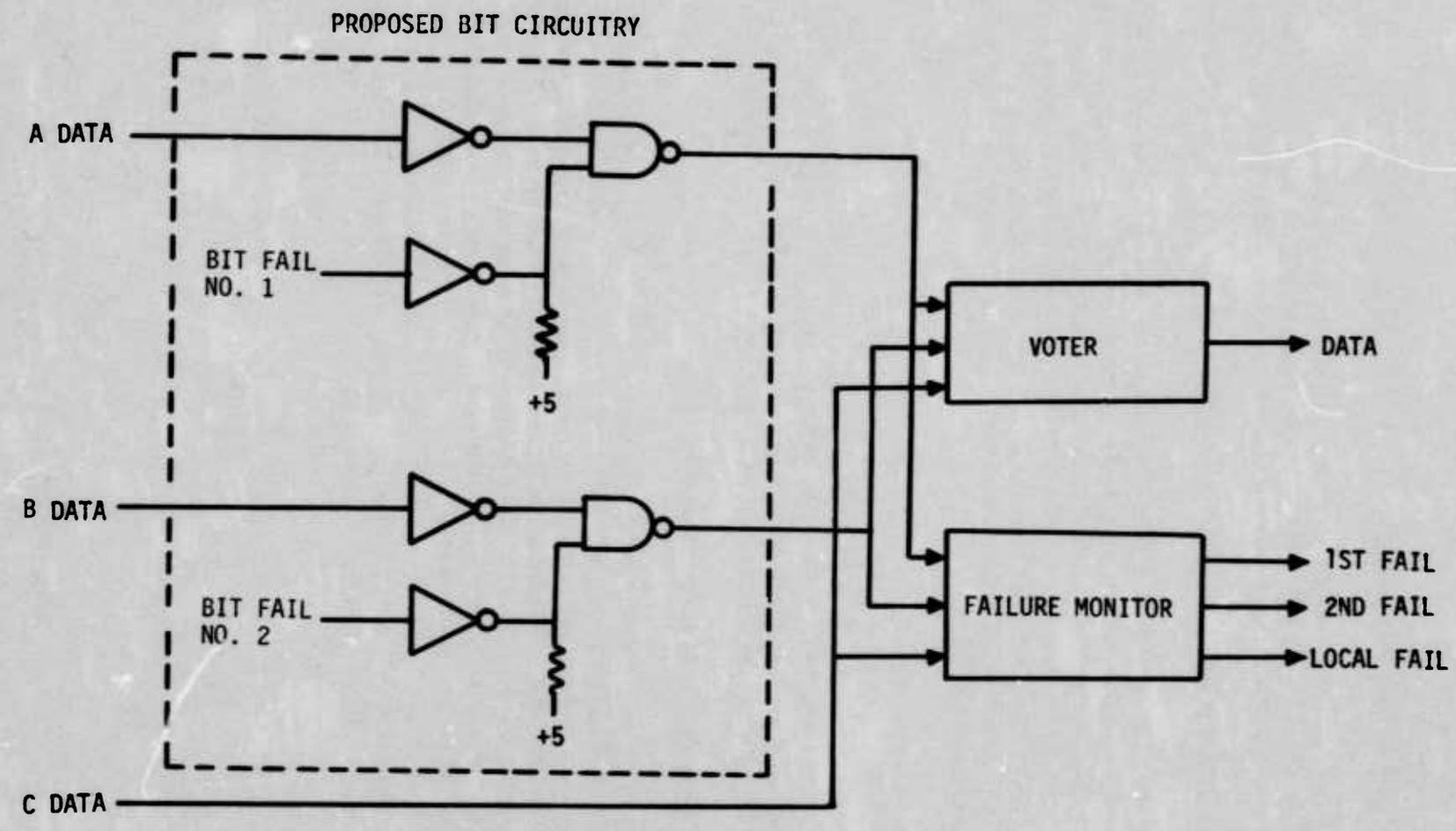


FIGURE B-1.–PROPOSED ICPS BUILT-IN-TEST ELECTRONICS FOR MAJORITY LOGIC VOTERS

TABLE B-1.—UNDETECTED FAILURE MODES WITH USE OF BIT CIRCUITRY

——— FAILURE MODES DETECTED WITH NORMAL DATA - Runs 1, 2

——— ADDITIONAL FAILURE MODES DETECTED WITH BIT CIRCUITRY (A FAIL HI, B FAIL HI) - Runs 10, 12

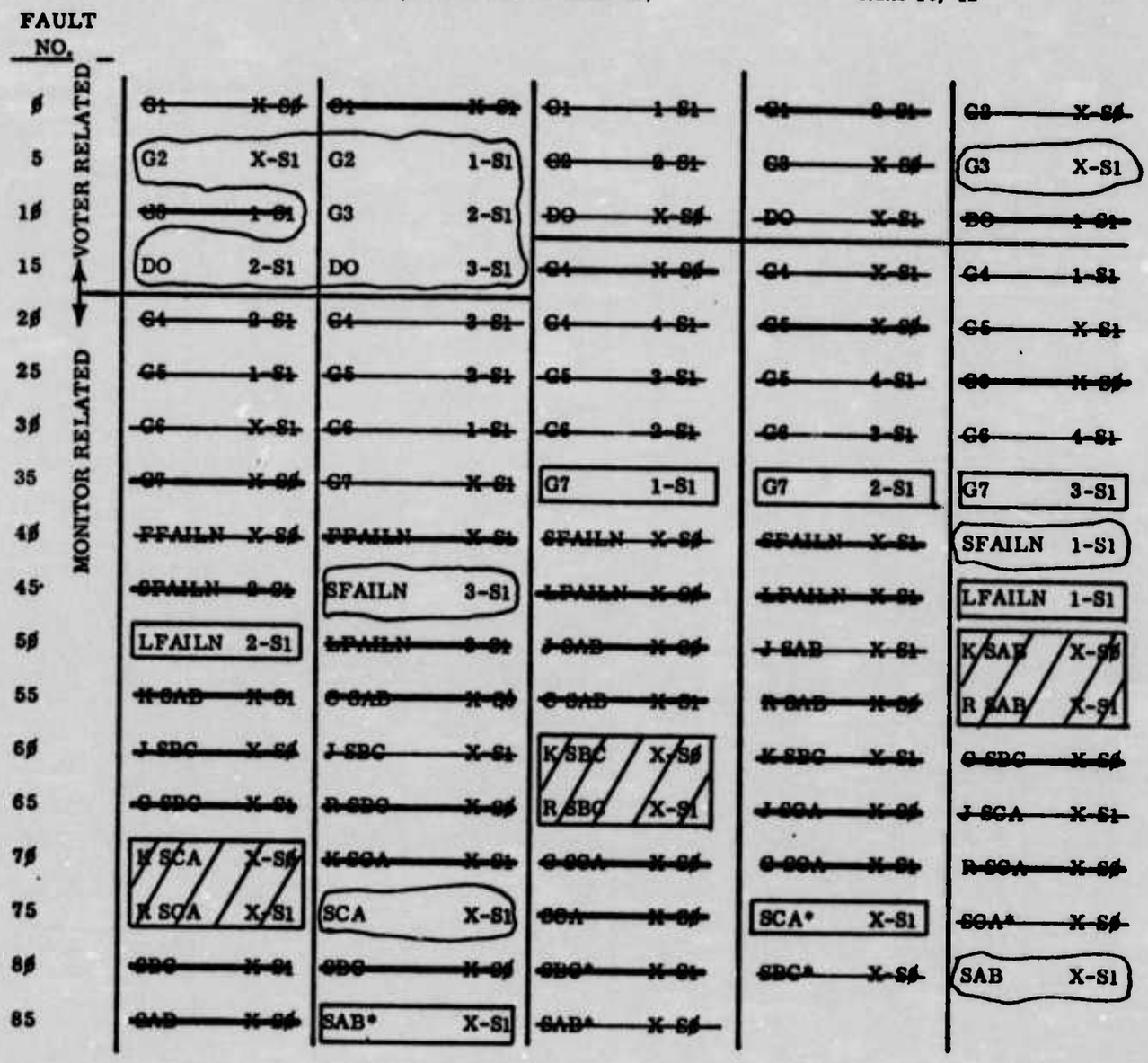


TABLE B-2.–PATTERN OF UNDETECTED FAILURE MODES OF CONCERN AFTER BIT CONDITION II

Compiled from Runs 1, 2, 6, and 9.

FAULT NO.												
ø	VOTER RELATED	G1	X-Sø	G1	X-S1	G1	1-S1	G1	2-S1	G2	X-Sø
5		G2	X-S1	G2	1-S1	G2	2-S1	G3	X-Sø	G3	X-S1
1ø		G3	1-S1	G3	2-S1	DO	X-Sø	DO	X-S1	DO	1-S1
15		DO	2-S1	DO	3-S1	G4	X-Sø	G4	X-S1	G4	1-S1
2ø	MONITOR RELATED	G4	2-S1	G4	3-S1	G4	4-S1	G5	X-Sø	G5	X-S1
25		G5	1-S1	G5	2-S1	G5	3-S1	G5	4-S1	G6	X-Sø
3ø		G6	X-S1	G6	1-S1	G6	2-S1	G6	3-S1	G6	4-S1
35		G7	X-Sø	G7	X-S1	G7	1-S1	G7	2-S1	G7	3-S1
4ø		FFAILN	X-Sø	FFAILN	X-S1	SFAILN	X-Sø	SFAILN	X-S1	SFAILN	1-S1
45		SFAILN	2-S1	SFAILN	3-S1	LFAILN	X-Sø	LFAILN	X-S1	LFAILN	1-S1
5ø		LFAILN	2-S1	LFAILN	3-S1	J SAB	X-Sø	J SAB	X-S1	K SAB	X-Sø
55		K SAB	X-S1	C SAB	X-Sø	C SAB	X-S1	R SAB	X-Sø	R SAB	X-S1
6ø		J SBC	X-Sø	J SBC	X-S1	K SBC	X-Sø	K SBC	X-S1	C SBC	X-Sø
65		C SBC	X-S1	R SBC	X-Sø	R SBC	X-S1	J SCA	X-Sø	J SCA	X-S1
7ø		K SCA	X-Sø	K SCA	X-S1	C SCA	X-Sø	C SCA	X-S1	R SCA	X-Sø
75		R SCA	X-S1	SCA	X-S1	SCA	X-Sø	SCA*	X-S1	SCA*	X-Sø
8ø		SBC	X-S1	SBC	X-Sø	SBC*	X-S1	SBC*	X-Sø	SAB	X-S1
85		SAB	X-Sø	SAB*	X-S1	SAB*	X-Sø				

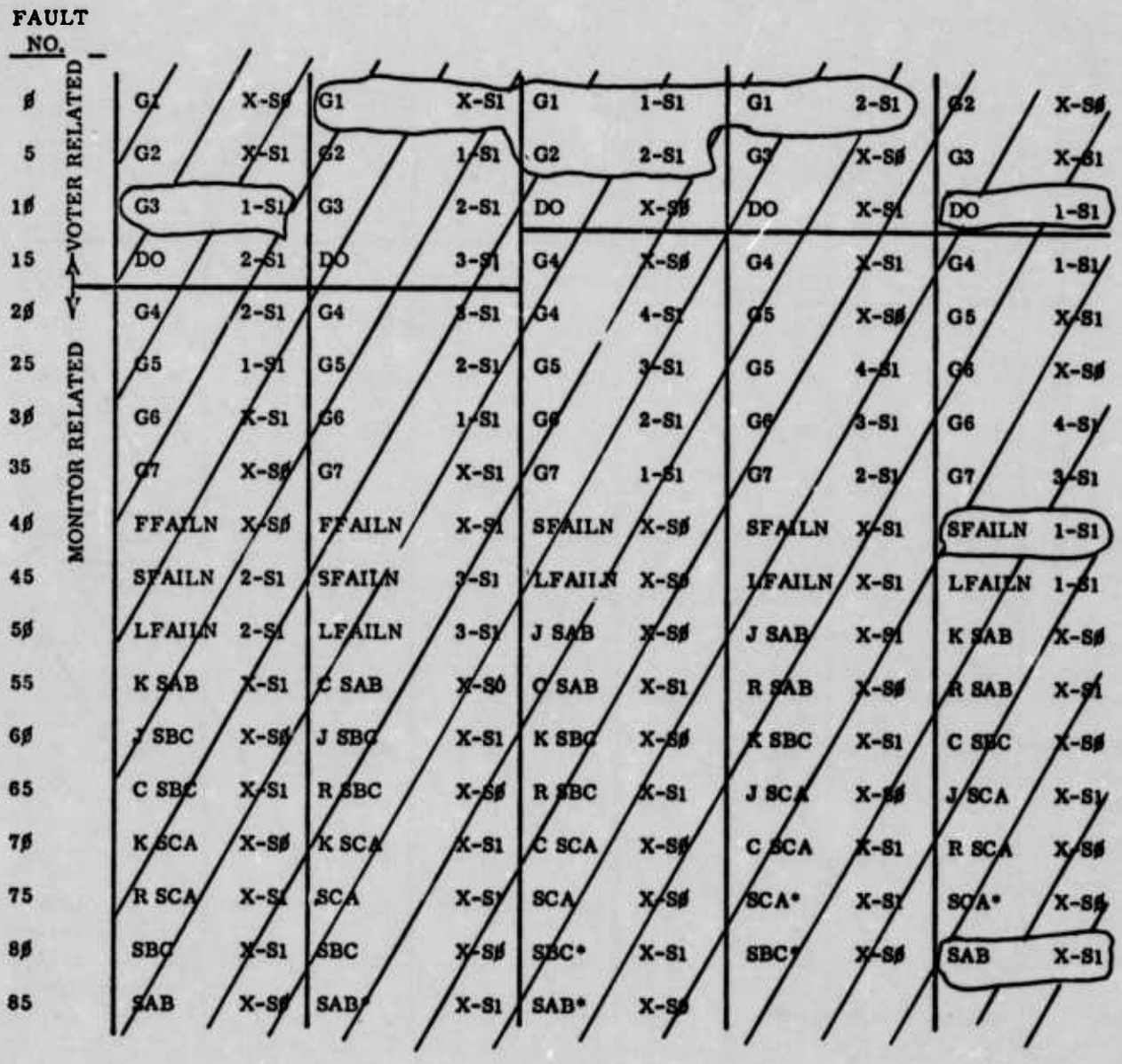
TABLE B-3.–PATTERN OF UNDETECTED FAILURE MODES OF CONCERN AFTER BIT CONDITION III

Compiled from Runs 1, 2, 4 and 11.

FAULT NO.												
ø	VOTER RELATED	G1	X-Sø	G1	X-S1	G1	1-S1	G1	2-S1	G2	X-Sø
5		G2	X-S1	G2	1-S1	G2	2-S1	G3	X-Sø	G3	X-S1
1ø		G3	1-S1	G3	2-S1	DO	X-Sø	DO	X-S1	DO	1-S1
15		DO	2-S1	DO	3-S1	G4	X-Sø	G4	X-S1	G4	1-S1
2ø	MONITOR RELATED	G4	2-S1	G4	3-S1	G4	4-S1	G5	X-Sø	G5	X-S1
25		G5	1-S1	G5	2-S1	G5	3-S1	G5	4-S1	G6	X-Sø
3ø		G6	X-S1	G6	1-S1	G6	2-S1	G6	3-S1	G6	4-S1
35		G7	X-Sø	G7	X-S1	G7	1-S1	G7	2-S1	G7	3-S1
4ø		FFAILN	X-Sø	FFAILN	X-S1	SFAILN	X-Sø	SFAILN	X-S1	SFAILN	1-S1
45		SFAILN	2-S1	SFAILN	3-S1	LFAILN	X-Sø	LFAILN	X-S1	LFAILN	1-S1
5ø		LFAILN	2-S1	LFAILN	3-S1	J SAB	X-Sø	J SAB	X-S1	K SAB	X-Sø
55		K SAB	X-S1	C SAB	X-S0	C SAB	X-S1	R SAB	X-Sø	R SAB	X-S1
6ø		J SBC	X-Sø	J SBC	X-S1	K SBC	X-Sø	K SBC	X-S1	C SBC	X-Sø
65		C SBC	X-S1	R SBC	X-Sø	R SBC	X-S1	J SCA	X-Sø	J SCA	X-S1
7ø		K SCA	X-Sø	K SCA	X-S1	C SCA	X-Sø	C SCA	X-S1	R SCA	X-Sø
75		R SCA	X-S1	SCA	X-S1	SCA	X-Sø	SCA*	X-S1	SCA*	X-Sø
8ø		SBC	X-S1	SBC	X-Sø	SBC*	X-S1	SBC*	X-Sø	SAB	X-S1
85		SAB	X-Sø	SAB*	X-S1	SAB*	X-Sø				

TABLE B-4.—PATTERN OF UNDETECTED FAILURE MODES OF CONCERN AFTER BIT CONDITION IV

Compiled from Runs 1, 2, 3, and 5.



FAULT NO.												
ø	VOTER RELATED	G1	X-Sø	G1	X-S1	G1	1-S1	G1	2-S1	G2	X-Sø	
5		G2	X-S1	G2	1-S1	G2	2-S1	G3	X-Sø	G3	X-S1	
1ø		G3	1-S1	G3	2-S1	DO	X-Sø	DO	X-S1	DO	1-S1	
15		DO	2-S1	DO	3-S1	G4	X-Sø	G4	X-S1	G4	1-S1	
2ø	MONITOR RELATED	G4	2-S1	G4	3-S1	G4	4-S1	G5	X-Sø	G5	X-S1	
25		G5	1-S1	G5	2-S1	G5	3-S1	G5	4-S1	G6	X-Sø	
3ø		G6	X-S1	G6	1-S1	G6	2-S1	G6	3-S1	G6	4-S1	
35		G7	X-Sø	G7	X-S1	G7	1-S1	G7	2-S1	G7	3-S1	
4ø		FFAILN	X-Sø	FFAILN	X-S1	SFAILN	X-Sø	SFAILN	X-S1	SFAILN	1-S1	
45		SFAILN	2-S1	SFAILN	3-S1	LFAILN	X-Sø	LFAILN	X-S1	LFAILN	1-S1	
5ø		LFAILN	2-S1	LFAILN	3-S1	J SAB	X-Sø	J SAB	X-S1	K SAB	X-Sø	
55		K SAB	X-S1	C SAB	X-Sø	C SAB	X-S1	R SAB	X-Sø	R SAB	X-S1	
6ø		J SBC	X-Sø	J SBC	X-S1	K SBC	X-Sø	K SBC	X-S1	C SBC	X-Sø	
65		C SBC	X-S1	R SBC	X-Sø	R SBC	X-S1	J SCA	X-Sø	J SCA	X-S1	
7ø		K SCA	X-Sø	K SCA	X-S1	C SCA	X-Sø	C SCA	X-S1	R SCA	X-Sø	
75		R SCA	X-S1	SCA	X-S1	SCA	X-Sø	SCA*	X-S1	SCA*	X-Sø	
8ø		SBC	X-S1	SBC	X-Sø	SBC*	X-S1	SBC*	X-Sø	SAB	X-S1	
85		SAB	X-Sø	SAB*	X-S1	SAB*	X-Sø					

TABLE B-5.–FAILURE MONITOR PARAMETER VALUES

Input	T1 sec	T2 sec	TH 1 (% full-scale)	TH 2 (% full-scale)	TD 1 counts	TD 1 counts
Pitch rate	0.098	0.8	2.5	1.95	1	1
Column input	0.098	–	5.0	–	1	Inhibit
Negator	0.098	–	5.0	–	1	Inhibit
ECS position	0.098	–	5.0	–	1	Inhibit
Detent (OFD)	0.098	0.8	–	3.0	Inhibit	64
Detent (DFD)	0.098	–	28.0	–	128	Inhibit
Detent (SFD)	0.098	–	28.0	–	128	Inhibit

Note: full scale = 10 V

TABLE B-6.—ICPS LABORATORY PREFLIGHT TEST RESULTS (HSAS)

Test number	Failure description	Test stimuli	Failure display																												Remarks
			System monitoring									Maintenance monitoring																			
												Interface unit										Computer unit									
			Ch A			Ch B			Ch C			Output voter		ITR RST voter (10)		Clock SW voter		OSC monitor		Sensor monitor		Input voter		ITR RST voter		Clock voter		Sensor	Bite	None	
			L	1st	2nd	L	1st	2nd	L	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd				
1	θ passive; Ch B	Gyro torquing (±4V, 0.2 Hz, triangular wave)		X		X	X			X											X							B2			
2	θ passive; Ch A and B	↓		X			X		X	X											X							C2			Falsely indicates channel C
3	δ_{col} passive; Ch A	Column input (Same as gyro torquing)	X	X			X			X											X							A3			
4	Negator passive; Ch A	↓	X	X			X			X											X							A4			
5	ECS position passive; Ch A		X	X			X			X											X							A6			
6	Manual trim up or down low; Ch A	Manual trim command	X	X			X			X											X										Does not indicate which discrete input.
7	Manual trim up low; Ch. A—trim down low; Ch B	↓		X	X	X	X	X		X	X										X	X									
8	Cut cable; btw Ch A and SSCU	Column input (Same as gyro torquing)					X			X											X										
9	Cut cable; btw Ch A and analog inputs	↓	X	X			X			X											X							A3			No indicators without column stimulus
10	Cut cable; btw Ch A and analog	↓	X	X			X			X											X							A7,8			Same as above.
11	Cut cable; btw Ch A and discrete interface	Manual trim command	X	X			X			X											X										No indicators without trim stimulus.
12	Cut cable; Ch A CIU to Ch B CIU	None		X			X		X	X	X	X	X	X		X		X		X		X	X	X		X	X	C, all			Loss of one cable gives second failure.
13	All power off; Ch A computer	None	X	X			X			X		X												X				PS	PS		

TABLE B-6.—Concluded

Test number	Failure description	Test stimuli	Failure display																													Remarks
			System monitoring									Maintenance monitoring																				
												Interface unit										Computer unit										
			Ch A			Ch B			Ch C			Output voter		ITR RST voter (10)		Clock SW voter		OSC monitor		Sensor monitor		Input voter		ITR RST voter		Clock voter		Sensor	Bite	None	
			L	1st	2nd	L	1st	2nd	L	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd				
14	One power source off; Ch A	None	X																									PS		No first failure.
15	Alg 5; S_q = 4738	Gyro torquing	X	X			X			X		X																ROM		All failures below are for Ch A; memory faults.
16	Alg 3; negate ΔP	None	X																									ROM		No light except BITE. P is not used in program, thus no failure.
17	Alg 3; negate ΔP & ΔT	None	X	X			X			X		X																ROM OVFL		
18	Alg 3; ΔW not negative	None	X	X			X			X		X																ROM OVFL		
19	Alg 3; ΔO not negative	None	X	X			X			X		X																ROM OVFL		
20	Alg 3; S_q = 637	Gyro torquing	X	X			X			X		X																ROM		
21	Alg 3; V not no. 6	Gyro torquing	X	X			X			X		X																ROM		
22	Alg 3; U not no. 4	Gyro torquing	X	X			X			X		X																ROM		
23	Alg 3; U_{i-1} = 0	None	X	X			X			X		X																ROM		
24	Alg 3; V_{i-1} = 0	None	X	X			X			X		X																ROM		
25	Alg 3; ΔP = 001	None	X	X			X			X		X																ROM		

Note: Refer to HSAS Algorithm map, figure 3.7 in FAA-SS-73-2-2, Performance Document for Identification of Points in Tests 15 through 25.

APPENDIX C

WWCS FMECA

This appendix contains the various FMECA studies conducted on the WWCS. The redundancy structure was partitioned into the first- and second-level FMECA areas of concern as discussed in section 3.2. Elements of the WWCS similar or identical in design to elements of the ICPS were not reevaluated. These elements included the redundant clock, majority logic voter/monitor and analog input/output circuitry, and associated timing. Level one WWCS elements analyzed and presented are the device controller interface, SSCU interface, and servotransmitter/receiver interface. Second level functions covered included the WWCS synchronous logic interface, memory, and central processor.

C.1 DEVICE CONTROLLER

The device controller interface is used as a communication path to and from the WWCS central processor (CP) and other subsystems (or system elements) over which the CP has little or no timing control. While the device controller interface to the CP is in general a common design, the transmitter/receiver circuits, which interface with each external subsystem, are specifically designed to satisfy the unique characteristics of each subsystem. Further, the failure effect of the circuits within the device controller will depend in large upon the software design associated with the particular interface. A block diagram showing the general design of a device controller is shown in figure C-1.

In general, the device controller timing and control section obtains the attention of the CP through the generation of an interrupt signal. The processor will respond to the interrupt by interrogating the requesting device controller's address and by jumping to the appropriate software subroutine which handles that I/O interface. The interrupt feature can be interlocked to prevent an outside subsystem from interrupting the CP, whereby the device controller is serviced strictly under CP program control.

To resolve the problem of having a number of device controller simultaneously requesting CP service, a priority system is used. The priority is established through computer unit harness wiring. The ordering of these priorities in the WWCS are:

First and second:	Cross channel receivers (from the other two channel computers)
Third:	Computer monitor and control unit (CMCU)
Fourth:	Tape reader
Fifth:	Paper punch

The priority system is mechanized so that a higher order device disables any lower ordered device from requesting service. The remaining device controller interfaces (SSCU, X-channel transmitter, and failure status and BIT discretes) do not have priorities because their design is such that they do not generate an interrupt. These interfaces are serviced exclusively by program sequence in the WWCS executive routine.

Once the processor responds to an interrupt, it locks out the priority structure so that the device being interrogated is not disabled by a higher priority device. Once the CP has finished with a device, any devices still requesting service, under established priorities, can send an interrupt to the processor. The processor also has the ability to either selectively mask a device interface or to mask them all. This capability allows any single device or the entire processor-controlled interface section to be ignored or masked during critical program calculations.

Interaction of the hardware/software utilization of a device controller interface can be typified by an evaluation of the WWCS computers cross-channel circuits. The following, therefore, presents a description of the WWCS cross-channel device controllers, an FMECA for their utilization within the WWCS laboratory studies, and the ramifications of the cross-channel hardware/software if the interface were used within a flight-critical program.

C.1.1 Cross-Channel Device Controller Transmitter/Receiver Description

There are two cross-channel receivers in each MCP-703 computer unit for receiving data transfers from the other two computer units. The receiver circuits are activated by the leading one in an 18-bit data transmission train. This marker bit is latched in order to provide an enabling signal to allow the shift register to acquire the data and to allow a coordinating counter to run. Upon reaching the count of 16, the enable is cleared and a desire-for-service request (SET) is sent to the device controller timing circuits to establish an interrupt. The last bit of the 18-bit transmitted word is a parity bit. Figure C-2 shows the logic of the cross-channel interface receiver design.

Upon CP recognition of the interrupt, the first word taken *(take)* will contain the valid (parity) designator. The first *take* will also toggle the multiplex control that clears the parity checker and enables the data to the buss buffer. The second *take* can then acquire the data.

The receiver device controller timing and control circuits are shown in figure C-3. These circuits are relatively complex because the receiver device controller interface operates under interrupt control or under computer service control, with the CP also able to selectively prevent interrupts. Operation of this section centers about the three events (states) of *ready*, *select*, and *mask*. The receiver sets the *ready* state when it desires service by the computer. The *select* state is then set by the *ready* state if no other device is already being serviced (I/O *lock*) and the device is not masked to prevent interrupts. The combination of *ready* and *select* is used to drive the interrupt line to the computer *(request)*. Once the computer has responded to this interrupt, the higher priority devices are prevented from disturbing the system by having them disabled (I/O *lock*). The I/O *lock* prevents the *ready* signal from passing onto the *select* logic.

The priority system is established about the *select* state. It is so constructed that any *select* will clear any other *selects* that are lower in priority. Thus, only one *select* can be set at any one instant. Thereby the *select* state indicates which device interface is going to be serviced by the computer.

The CP can also set the *select* with the *sel* instruction. The address portion of this instruction directs which interface is to be set. Also, the execution of this *sel* instruction unselects any other device that might be set.

The third state, *mask*, performs a similar function as the I/O *lock*, except that this state is set by the computer on an individual device interface basis.

The *ready* state is cleared either by CP servicing or by a *reset* instruction. The *select* state is cleared by the reset instruction or by the selection of another device. *Mask* is cleared by the *sel* instruction of its own device or globally (all mask's) with a device enable instruction.

The transmitter side of the cross-channel interface (fig. C-4) is completely controlled by the CP through the computer program. The transmitter device controller must be addressed using the *select* instruction, which enables the transmitter circuitry, and must be activated through the use of a *put* instruction. Upon the execution of the *put* instruction, data in the MCP-701 upper register are transferred to the transmitter shift register, and transmission is initiated.

The 16-bit data word is transferred to the cross-channel receivers as an 18-bit word. The first bit is a marker bit ("1") that is required to turn on the receiver. This is followed by the data (LSB first) then the parity bit. This data string is sent at a 2.5 MHz clock rate derived from the CP clock.

Provision is made to prevent the CP from initiating a second transmission before the first is complete. This is accomplished through the use of ANDing the *select* state and the fact that the transmitter is idle. This ANDed signal is then placed onto the interrupt buss for device controllers. Signals on this buss set the device ready bit of the computer (CP) status register. This bit can be interrogated through software to defer a second transmission (put instruction) until the device is ready.

The transmitter timing sequence is essentially a counter and a couple of flip-flop devices mechanized as a shift register. Receipt of the initiate signal inserts a control bit into the shift register that sends out the marker bit. It then shifts to enable the counter and transmitter for 16 counts. Following this, the control bit is shifted to enable the parity bit onto the transmission line. Finally, the control bit clears the counter and the parity checker.

The integrity check on the transmitter is by a parity check at the receiver. As a means to provide a check on this monitor, the ability to complement the parity bit is provided. This function is exercised by the computer self-test, BIT program.

C.1.2 Cross-Channel FMECA

Chief concern of the cross-channel interface is whether there is a failure possibility in a single transmitter that could cause the receivers in the other two channels to encumber their respective CP's. Such an event would present a single point failure that could result in a total system malfunction. A failure effect analysis of the transmitter/receiver failure modes, delineated in table C-1, established that there were several failures that could cause a continual request for servicing of the attached receivers.

To assess the impact on the system, the resident software was checked to see how the cross-channel interface was utilized. If the interface is operated strictly on an interrupt basis (i.e., service when requested), system operation can be severly impacted by the time required to service the interface. Even if the validity test indicates a failure, time is still required to link up the interface to determine that fact. If the interface is used totally under CP control, no hardware failures appear to affect the system operation.

A review of the WWCS in-flight application resident program showed that cross-channel exchanges are only used by the WWCS executive (level I) program for initialization after power turn-on and for frame synchronization of the three computers following a computational reset command. Both of these cross-channel exchanges utilize CP control of the interface and mask the interrupt operation.

Device controller interrupts were not allowed during the processing of the flight-control law (level II) program to satisfy strict redundant-operation timing requirements. Device controller interrupts were, however, required in the background (level III) program to service the computer monitor and control unit when this unit is coupled to the WWCS equipment. The CMCU is a laboratory support unit and would not be a part of the system in actual flight operations. Further, the CMCU program does not utilize the cross-channel interface.

The only utilization of the cross-channel interface, outside the executive (level I) program use mentioned above, was within the preflight test (level IV) program. This program is made up of the WWCS BIT and the application model PHT. These tests are conducted as a ground/laboratory operation only and are double interlocked to prevent their operation in-flight.

It was determined from this analysis of the WWCS mechanization of the application (HSAS) flight-control system that no critical failure modes existed within the cross-channel interface. This, however, reflects a specific WWCS configuration use of the cross-channel interface. In order to provide a better understanding of the software/hardware interaction for such an interface from the FMECA point of view, the following discussion is presented on the use of the cross-channel interface with the level II (control law) resident program.

C.1.3 Flight-Critical System Use of Cross-Channel Interface

If the flight resident software were to be restructured so that cross-channel transfers were to be made between computer units to improve redundancy management during level II program execution, several ramifications would have to be considered. In the process of

executing a cross-channel transfer within the redundant system, two receptions of that transfer must be anticipated by the receiving computers software. Since only one I/O device can be serviced by a computer at one time, it would be necessary for each channel (computer) to sequentially service its two receivers. Thus, the first receiver must be selected, data retrieved, and reset with this process repeated for the second receiver.

The simplest possible cross-channel interchange of data could be executed by each computer initiating their transmitter, followed by each computer processing the data out of both of their receivers. In order for this interchange to be successful, the computation time between processing the transmitter routine and processing the first receiver routine must account for the actual transmission delay plus the worst-case timing skew between channels (asynchronous operation within the synchronized frame time). To understand this, consider what would happen if the fastest computer executes the receiver routine before the slowest computer has completed the transmitter routine. There would be no data or erroneous data in the fastest computer's receiver.

The above transmitter/receiver routines assume CP control of the cross-channel process, where time delays or unrelated instructions have to be set up to provide the required transmit-receive separation. It is, however, difficult to know worst-case timing skews, and providing wide timing margins could result in a compromise of the total program execution time, especially if the cross-channel information is keyed to continuing the main-line program. Thus, a more elaborate cross-channel software structure could be developed to synchronize the fastest computer with the slowest computer during cross-channel exchanges. The following requirements for such a process were therefore postulated:

1) Cross-channel exchange synchronization must occur by transmitting and then receiving (both receivers), and by checking the reception to determine its validity.

2) If a reception is not valid, the receiver routine(s) would again be repeatedly executed until a good reception occurs.

3) A loop execution count for item 2 above must be made with provisions to exit the loop after a specific number of iterations.

4) A signal selection or reasonableness criteria routine would have to be used in item 1 above to establish the reception validity.

This software approach potentially minimizes the cross-channel information transmission/reception time and protects the system from a single point failure for any transmitter failure. Operating the cross-channel interface utilizing the I/O interrupt would result in saving processor time, but the single-point failure hazard makes this approach impractical.

C.2 SSCU INTERFACE DESCRIPTION/FMECA

The WWCS failure status and error reset/computational reset control functions are provided by the system status and control unit (SSCU) shown in figure C-5. The SSCU also has the control/display functions associated with preflight test. The following discussion covers only the former functions and pertains to the failure indicators (LOCAL, 1ST, and 2ND) in the upper right corner of the SSCU panel (fig. C-5) and to the control functions found in the lower left of that panel.

The error reset and computational reset controls are spring return, single action, triple contact switches (fig. C-6). Error reset attempts to reset the failure detection/indication on those monitored functions that are considered off-line and do not affect the system configuration. Computational reset, on the other hand, initializes the basic state of the WWCS configuration by removing all software time history and by activating the reset logic for all monitored functions that, upon failure detection, modify the WWCS redundancy structure. The computational reset action is visualized as a ground operation only and would be interlocked to prevent its activation in flight.

Failure status information on the CIU, STRU, and CU is all combined in software to formulate a system/channel failure state. The resultant system failure state, along with functional level failure information, is then transmitted to the SSCU via a device controller interface that is under CP control (fig. C-7).

Any active failures within the computational reset control logic or failure state information paths will be readily apparent. The computer system or one channel will be held in computational reset (system main-line software will not execute), or a portion or all of the display lamps will be illuminated. Potential latent failures exist in both the error reset logic and the failure display information paths. Error reset failures can activate the reset lines to the various monitor functions whereby an actual failure would not be registered or displayed. Such latent failures must be cleared through a preflight check process.

Control and display failure modes generally do not represent a hazard within a redundant system as long as basic isolation (redundant signal independence) ground rules are followed. However, preflight checks of these functions ultimately require flight crew interaction.

C.3 SERVOTRANSMITTER/RECEIVER UNIT

The STRU is an input/output interface between the WWCS and the flight-control system hydromechanical servosystem. Its design utilizes the same A/D and D/A conversion electronics discussed for the ICPS computer interface unit in section A.5. The following STRU description and FMECA cover only the differences between the STRU and the ICPS CIU.

C.3.1 STRU Description

The STRU performs the same I/O operations as found in the CIU. However, the STRU interface to the WWCS computer units is such that each channel of the STRU can receive different signal (command path) information during the same time interval. Thus, each channel of the STRU can have a unique voted input at any given instant in time; whereas each CIU channel presents the same (identical) output for any instant in time. This difference is illustrated in figure C-8. Essentially, the difference amounts to three transmitter registers in the computer unit dedicated to the STRU and only one transmitter register dedicated to the CIU.

Failure status information of the STRU is also treated different than that of the CIU. The failure information is transferred to the computer unit via a packed 16-bit data word rather than as independent discrete signals. The 16-bit word also contains the STRU to computer unit discrete signal transfers. Figure C-9 presents the bit assignment for this failure-state/discrete information data word. Figure C-10 illustrates the CIU and STRU discrete input interfaces to the computer unit.

The clock signal generation circuits within the STRU differ slightly to that of the CIU in that only one clock pulse is used in the STRU compared to a clock pulse and strobe pulse in the CIU. The algorithm time counter is also different. The STRU counter counts through eight time windows with 16 algorithm times in each window, where the CIU counts through a cycle of 128 algorithm times. However, the effects of failure modes in these generation circuits will be similar to the failure modes discussed for the CIU in appendix A.

The final difference between the STRU and CIU is that the STRU does not contain a 12-bit overflow detector and limiter for the D/A operation. The difference is covered in the illustration presented in figure C-8. Not having the overflow protection places the burden of limiting the output digital command on the programmer.

C.3.2 STRU/CIU FMECA Comparison

The STRU has all the A/D and analog input shift register and timing gate failures that were discussed for the CIU in section A. The analog output failures for the voter, D/A register, and D/A will also be the same as discussed in appendix A.

A failure in the computer unit's CIU transmitting register causes a first failure detection/indication in all three CIU's, where a failure in one of the STRU transmitting registers will only cause a first failure in one channel of the STRU.

A failure in STRU to computer unit transmitter, which causes all zeros to be transmitted, is detected by the loss of the line check bit provided in the discrete data word (bit 15, fig. C-9, always a one). For the alternate case where the transmitter transmits all ones, LOCAL, 1ST, and 2ND failures will be indicated; thus, the failures will be detected. Latent failures can occur in the failure status lines to the transmitter, where a failure will not be reflected by a zero-to-one state change of the failure registering logic.

As with the CIU, any failure of the STRU timing structure should be detected by monitors in the computer unit. However, these type of failures in one channel of the STRU

must be properly considered in defining the signal selection/failure detection functions to be performed in software in order to prevent failure cascading possibilites. For instance, it is possible for the STRU A/D converted data to be transmitted to reflect a value greater than plus or minus the analog full-scale value of 10 V. Such an event, without software reasonableness tests on the data, could cause a CP arithmetic overflow when combined with like data in a signal selection process. Such an unplanned failure effect could result in a common mode single point system failure. The failure possibility comes from the fact that the A/D signals of the STRU are placed in the lower 12-bits of the 16-bit transmitted word, where the sign bit is the 12th bit and the remaining higher order bits repeat the sign bit. This potentially hazardous situation can be easily overcome with either software or hardware protection.

The significant aspect of such failure effect possibilities is to remember that both hardware and software have to be defined when conducting the FMECA study. Further, once the FMECA has been conducted and the system design validated, software changes cannot be made without a thorough review of the failure effect ramifications of the change.

C.4 SYNCHRONOUS LOGIC INTERFACE

The SLI circuits are contained in the WWCS computer unit. They process the bit-for-bit synchronous input/output data between the computer unit memory and the CIU and STRU. This memory I/O operation is a hardwired program that utilizes a direct memory access link to the memory. The SLI circuit timing is developed from the clock and timing logic in the CIU.

C.4 1 SLI Functional Description

The SLI executes a fixed iterative program that stores/retrieves data into and out of specific core cells of the memory. A top view function flow diagram of the SLI is presented in figure C-11. Clock signals from the redundant CIU's and iteration resets from the three SLI's are voted and combined within each SLI to produce the timing and control signals used throughout the SLI. Dedicated holding registers are used to temporarily store the serial data received from the CIU's and STRU. The data are then transferred in parallel to the computer unit memory through a multiplexer operation. In a similar fashion, data from the memory are parallel loaded into output holding registers and then serially transmitted to the CIU's and STRU.

Three majority logic voters and associated monitors are used in the SLI receiver circuits (fig. C-12). The voter/monitors are the same or very similar to those discussed in section A.2. The monitor for the clock signal has been modified, however, to provide channels A, B, or C failure identification in order to disable the failed-clock signal going to the voter. The voter on the CIU data is only of value if a hardware signal selection algorithm is placed in the CIU. (Provisions for such a hardware signal selection algorithm does exist in the WWCS CIU but was not exercised during the FCD studies.) However, dedicated memory cells were established for both the voted CIU data and the individual CIU data.

The SIL timing and control operation (fig. C-13) essentially performs two functions. The first is to generate SLI clocking signals. The clocking signals are used to drive a bit-time shift register and a word time counter. The outputs of the shift register, word counter, and clock signals are then used to perform the second function, to control the data manipulation within the SLI electronics.

A key SLI control signal is *compare*. During each algorithm time (where there are 128 I/O algorithm times every 6.144 μsec, as set forth by the CIU design), a counter is run that registers the number of SLI put and take accesses to the memory. The SLI has primary control of the memory during this series of put and/or take operations, where each operation consumes one memory cycle (1 μsec). When the counter has incremented to a preset number of SLi operations for that algorithm time, the *compare* signal is generated. This signal is used to release memory accesses to be made by the SLI each algorithm time vary from one to eight and are established through the programming of read-only-memory devices.

Control of the memory by the SLI is scheduled through the memory switchover logic shown in figure C-14. The memory address register is multiplexed to pass SLI generated addresses or CP generated addresses depending upon which function is using the memory. The effective SLI address is controlled by a counter that is incremented for each SLI memory access, reset, and repeated every I/O frame time iteration. The SLI addresses decoded in a ROM to identify which I/O unit is being processed (CIU A, CIU B, etc.) and to set whether the data are to be written into or fetched from memory.

Write protection logic for the lower 32 words of memory exists to permit address overlaying of the core memory and a semiconductor read only memory. This logic controls which memory will be accessed. If data are to be processed (read/written), the core memory will be accessed. If an instruction is to be fetched, the ROM memory will be accessed. Control for this logic is contained in the microprogramming circuits of the CP.

C.4.2 SLI FMECA

Table C-2 summarizes the SLI failure modes and their effect on the system. Nearly all failures are detectable by the output data monitors in the STRU and CIU's. In fact, failures upstream of any of the output data paths will be detected whether or not they are being used in the application configuration.

No latent failures could be found in the SLI circuits, outside the majority logic voter/monitors used in the receiver circuits. The active nature of the multiplexed data processing, which generally extends from the incoming signals to one or more of the output commands, forces any failure (even on a bit level) to be propagated to the output voter/monitor where it will be detected. The integrity of the MLV functions in the SIL would have to be established through a preflight test.

C.5 WWCS MEMORY SYSTEM

The memory used in the MCP-703 computer unit is fabricated from ferrite cores. This memory is used to store program instructions (execution instructions for the CP), data for intermediate calculations from the CP, constants, and data pertaining to the peripheral interface input and output operations. The memory has a capacity for approximately 8,000 words (8K), with each word 18-bits long. The MCP-703 CP utilizes 16-bits of the word; one bit is used for parity checks; the remaining bit is unused.

C.5.1 Memory Description

The memory system is essentially comprised of the four functional sections shown in figure C-15. The timing and control section operates the core stack independent of the CP and administrates the memory/data available logic going to the computer system. The core stack decoding logic selects the particular core cell location(s) as a function of the memory address register. The core stack holds binary data (information) as a function of the magnetic flux orientation of each core cell. The memory data register retrieves or modifies data within the core as a function of the desired operation of the associated interface.

The memory timing is mechanized using a delay line constructed from a distributed LC network. At the arrival of an initiate signal, the memory timing chain commences (fig. C-16). A JK flip-flop latches the arrival of this signal, and 100 ns later a time tap from the delay line clears the flip-flop. This puts a uniform 100-ns pulse through the delay line. The pulse is then sent to a series of latches whose functions are to develop the memory control signals. When the chain completes its cycle, it will not restart until the initiate signal is recycled.

When the power to the memory is good (as determined from a power monitor), a power status signal enables the initiate signal to enter the timing chain. However, this signal is implemented so that when the power goes down, the memory will stay active for at least 50 μsec longer. When power first comes up, a power on reset signal is used to clear the timing control latches so that incompatible control modes will not exist.

The memory address register (fig. C-16) passes the selected memory word to a decoding module, which translates the selected word identification into specific core stack cell locations. The decoded signals pass through driver/decoder circuits that control the power transistors that drive the memory control lines. The current generator for the line drivers is simply power supply voltage with a series resistor to limit the current to the appropriate level. This resistance is trimmed as a function of stack temperature to allow for changes in the inductive coupling (geometry situation) caused by heat.

The memory data register (MDR) can be loaded from memory, from the SLI logic, or from the CP. When data are loaded into the MDR from memory, (memory read cycle) the contents of that memory location are destroyed (characteristic operation of a core memory). The data, however, are restored back into the core (memory restore cycle) at the same time it is distributed to the using interface. Data loaded into the MDR by either the SLI or CP will be placed into the core during the restore cycle of the MDR/memory operation. In this case, the read cycle can be reviewed as a clearing operation. The MDR

therefore distributes data to four locations: back to the memory core; to the SLI for transmission to the interface units; to the central processor B buss for CP use; and to a parity checker to establish that the core data holds to an odd parity. When a parity error has been detected, an interrupt is sent to the CP that forces the computer into a memory fault routine if a high priority interrupt is not present. The parity error does not prevent further processing of instructions, as recovery from this error is handled in software. The interrupt is cleared by execution of the clear arithmetic fault instruction.

C.5.2 Memory FMECA

The memory system failure modes and effects are summarized in table C-3. Three potential latent failure areas were identified. The first is whether the parity error checker is operational; the second is whether data have been lost in a core cell or group of core cells because of a loss of magnetic orientation of physical damage to the cell(s) such as a crack; and the third relates to the circuits (or software) that must respond to a parity error detection.

The loss of core information with parity error circuits operational can only become a hazard if it occurs in a section of code that will not be processed under normal circumstances over an extended period of time (e.g., code which is to respond to a failure detection). The hazard arises if the core loss, over time, occurs in two computers and then a failure causes the faulty area to be processed. A parity error would most likely be detected, and two computers responding to a parity error would be considered a single system failure.

Once a parity error has occurred and the CP responds by activating the appropriate interrupt software routine, the parity error detection has to be reset using a clear arithmetic fault instruction. If this logic fails, such that the clearing action did not take place, future parity error detections will not initiate an interrupt signal. This type of latent failure situation not only blocks the parity error detection but all lower priority arithmetic fault interrupts.

C.6 MCP-703 (WWCS) CENTRAL PROCESSOR

The CP of the MCP-703 consists of circuits that control the interpretation and execution of the instructions and data, which are stored in the memory. These circuits are referred to as the MCP-701 central processor. The basic architecture of the CP is a three-buss system as shown in figure C-17. This structure cycles all data processing through the arithmetic section.

The instruction repertoire and register control is developed through microprogramming. The operation code of an instruction is used as an address for a ROM. The output of this ROM forms a starting address for an additional ROM. The output of this second ROM develops the actual control logic which manipulate the electronics comprising the CP. With the use of this microprogramming design, modifications to instruction execution can be made by substitution of the ROM element without actually changing the wiring. Thus, there exists some potential for modifying the CP through changes of the software, which define the ROM program.

Failure modes analysis on previous functional elements have been conducted primarily on a bottoms-up approach. With such a method, failures are postulated at the component level, and the failure response propagation analyzed until its effect on the system operation could be determined. Essentially, all possible input combinations must be tried for all possible failures. For complicated circuits, such an exhaustive analysis would take a very long time. For instance, a 16-bit hardware multiplier can have 2^{32} possible output states. If we were to check this function at an analysis rate of 10^5 cases/sec, it would take about 1190 hr to complete the task.

For the MCP-701 central processor shown in detail in figure C-18, a middle-up analysis methodology was postulated for investigation of possible latent failures. This approach was undertaken due to the enormous manpower and time that would be involved to complete a bottoms-up analysis easily understood after observing the complexity of figure C-18. The results from either method (i.e., an identification of the latent failure prone areas in hardware) should be the same.

C.6.1 MCP-701 FMECA Approach

The basis of a middle-up analysis is to look at the digital computer at the instruction level, since a general purpose digital computer is nothing more than a device for sequentially processing instructions. Each instruction can be defined as a series of functional steps where it can be assumed that any of the functional steps can fail. Depending on the specific hardware design, the definition of a single functional failure may relate to one or many actual hardware failures--normally many. The instruction functional failues must then be evaluated with the computer system-flight resident software to determine whether the instruction failure mode would be detectable. In doing this, failure states can be classified as being immediately detectable, latent, or trivial.

To relate these failure states to the MCP-701 CP, the following examples are given. An immediately detectable failure would be a failure of the add-to-upper register (ADU) instruction, where the sum of two numbers were in error in one computer; three computers were assumed to be operational with an output bit-for-bit monitor such as that used in the WWCS.

A trivial failure would be a failure of the ADU instruction to set the sign adder (SA) bit of the CP status register, where no other instruction of the application program utilized (or tested) the state of the SA bit. A latent failure would be where the setting of the SA bit had failed, and where the SA state would be utilized in a portion of the program that would be active only after a detectable failure.

In this case, the SA bit setting latent failure could occur in time, in two computers, and result in the wrong system response to the detected failure.

Following the above failure mode analysis and failure state classification identification, latent failure possibilities must be looked at in great detail to:

- Establish that the failure mode is physically impossible and can be rejected.

- Identify specific piece-part or software elements that cause such a failure and redesign the processor to eliminate the failure mode.

- Develop a self-test hardware/software program to specifically exercise the function, as required, to preserve system functional reliability goals.

C.6.2 MCP-701 FMECA Example

A simple example was selected as an exercise of the FMECA approach discussed above. The flight-control system function, shown in figure C-19, was implemented using the instruction repertoire of the MCP-701. The specific code necessary to represent the function is shown in figure C-20. For this exercise, the executive and I/O code has been ignored.

Some further assumptions are made. It is assumed that the input signals, INPUT1 and INPUT2, are moving about zero (i.e., are not biassed positively or negatively). This implies that all bits in the word will be active all the time. A further assumption is that a system bit-for-bit output voter monitor is being used in a triplex system so that any error in any calculation will be detected at the output of the triple-channel computations.

The following presents the details of the analysis of the first instruction, LDU, of the sample application program. The instruction execution is based on the hardware architecture of the MCP-701.

Instruction	Operation	Explanation
LDU INPUT1	(INPUT1) → UR ZA, SA	Bring the contents of location INPUT1 to the UR. Set flags ZA, SA as appropriate.

Failure Modes	Explanation/Analysis	Failure Detection
(INPUT1)? → UR	Fail to select correct address – INPUT1	Immediate
(INPUT1) → ?UR	Data transfer failure (see assumptions)	Immediate
(INPUT1) → UR?	Failure to select UR for data	Immediate
SA?	SA set incorrectly (SA is not tested after LDU)	Trivial
ZA?	ZA set incorrectly (ZA is not tested after LDU)	Trivial

A summary of the analysis of the remaining sample problem instructions is presented in table C-4.

From this example, it was concluded that the middle-up approach had merit. Even though the analysis was not extended over the FCD task application model, an extrapolation of the results indicated that considerable manpower and time savings could be realized over an analysis, which would be based totally on the detailed hardware design. Further, this approach can be initiated as soon as the system hardware/software becomes visible (i.e., complete hardware and software details are not needed before the study can begin). It was determined in the example problem, that the potential latent failure in the multiply instruction could be uncovered using a random number multiply check routine as a self-test operation during a software background mode.

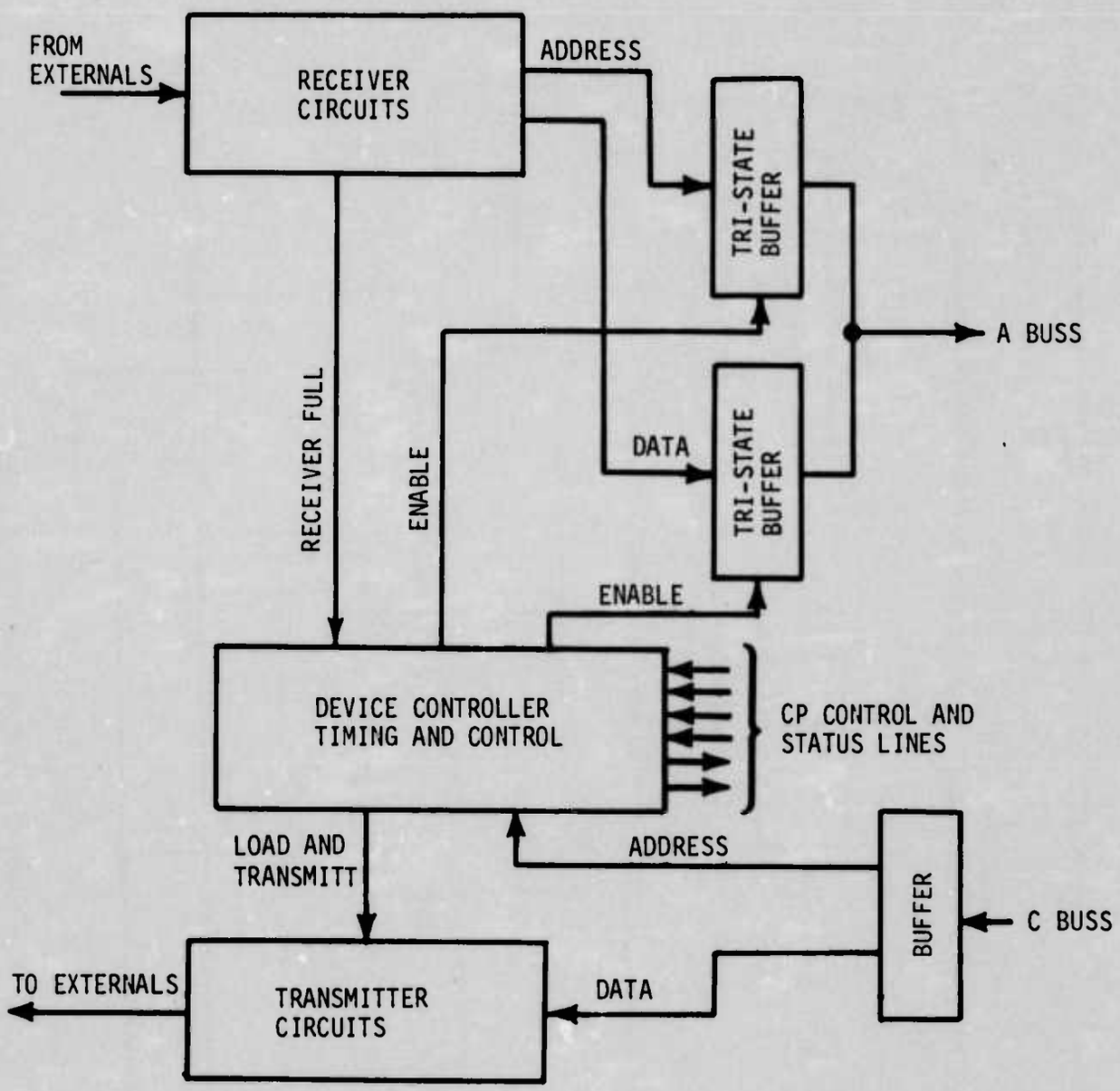


FIGURE C-1.—DEVICE CONTROLLER INTERFACE

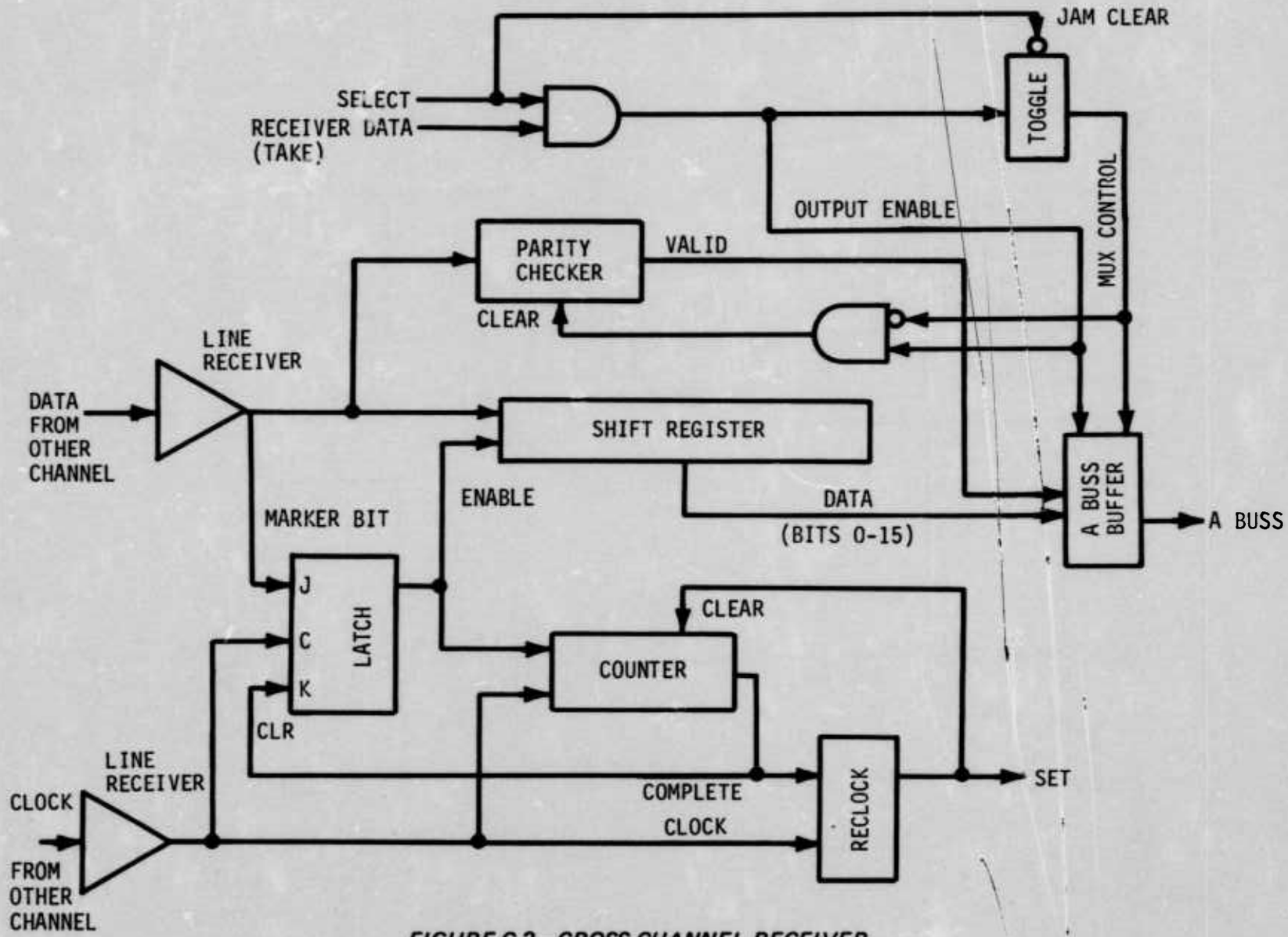


FIGURE C-2.–CROSS-CHANNEL RECEIVER

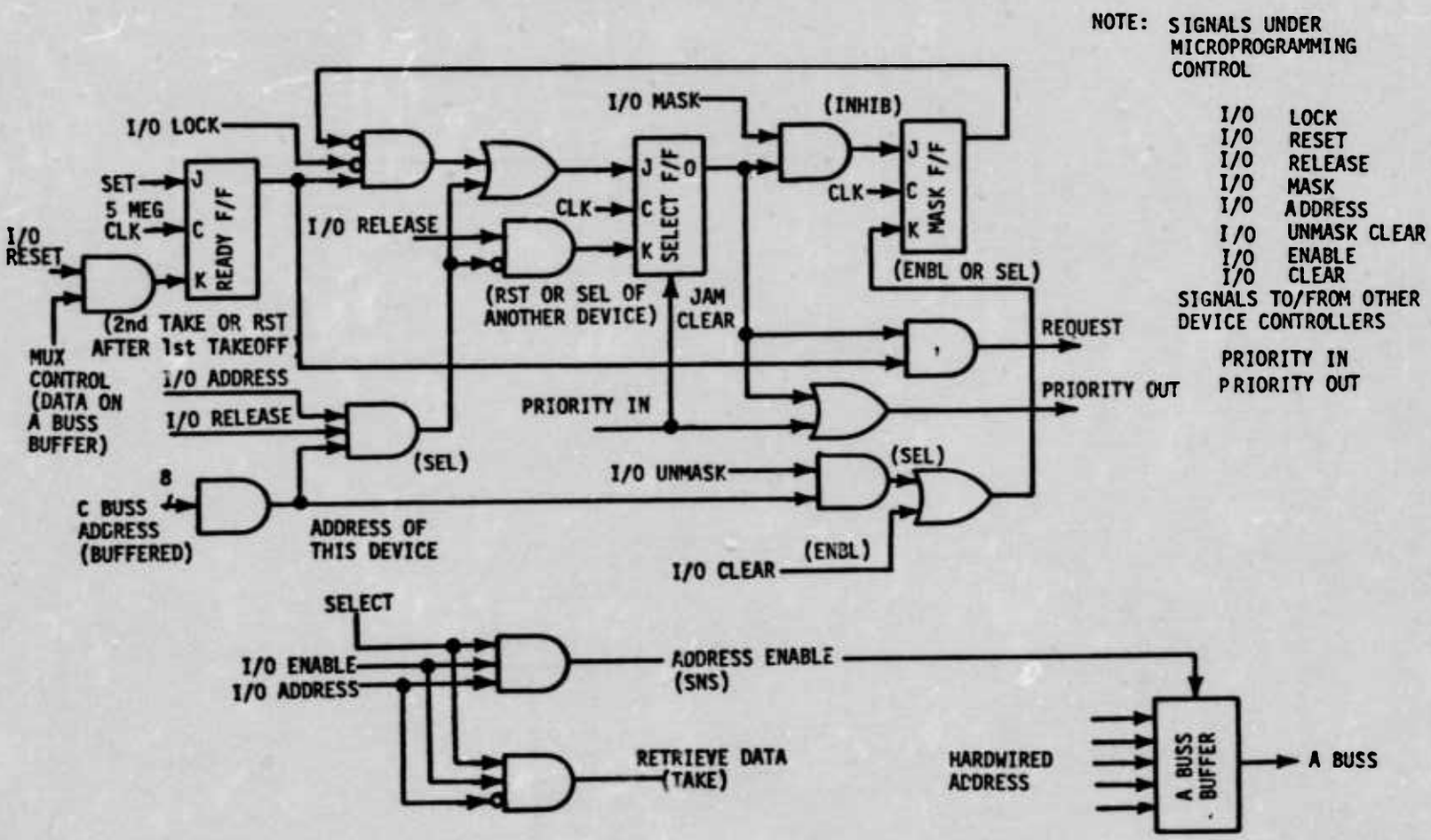


FIGURE C-3.–CROSS-CHANNEL RECEIVER DEVICE CONTROL TIMING AND CONTROL

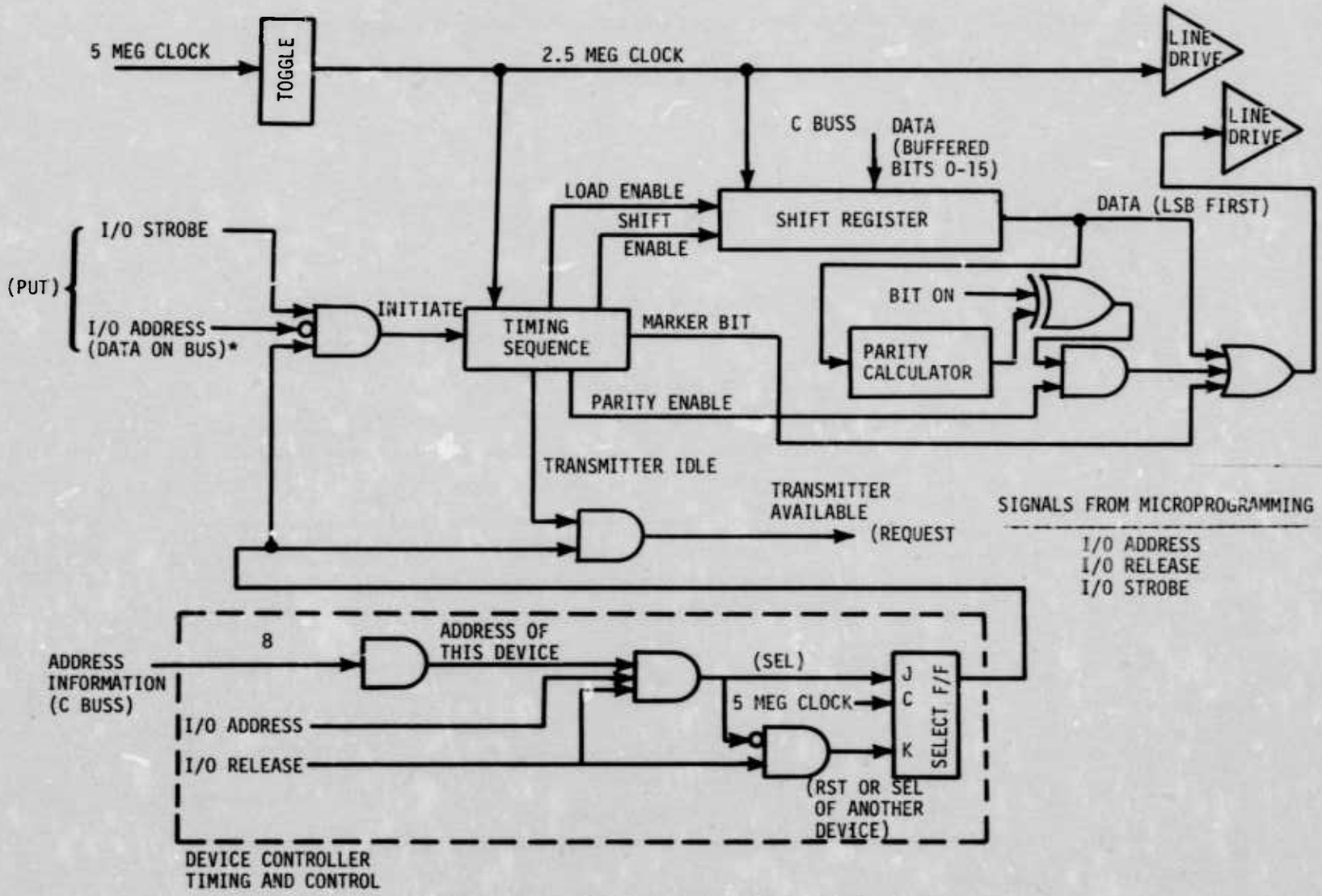


FIGURE C-4.–CROSS-CHANNEL TRANSMITTER

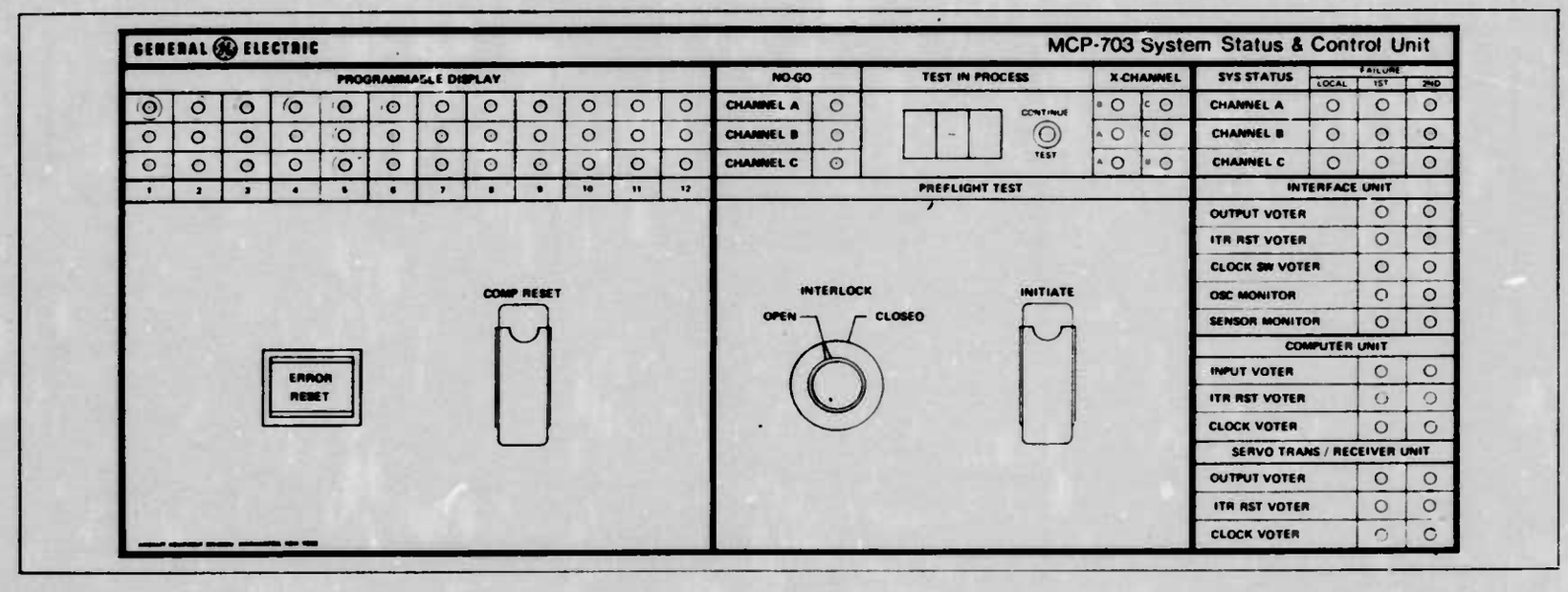


FIGURE C-5.–SYSTEM STATUS AND CONTROL UNIT FRONT PANEL

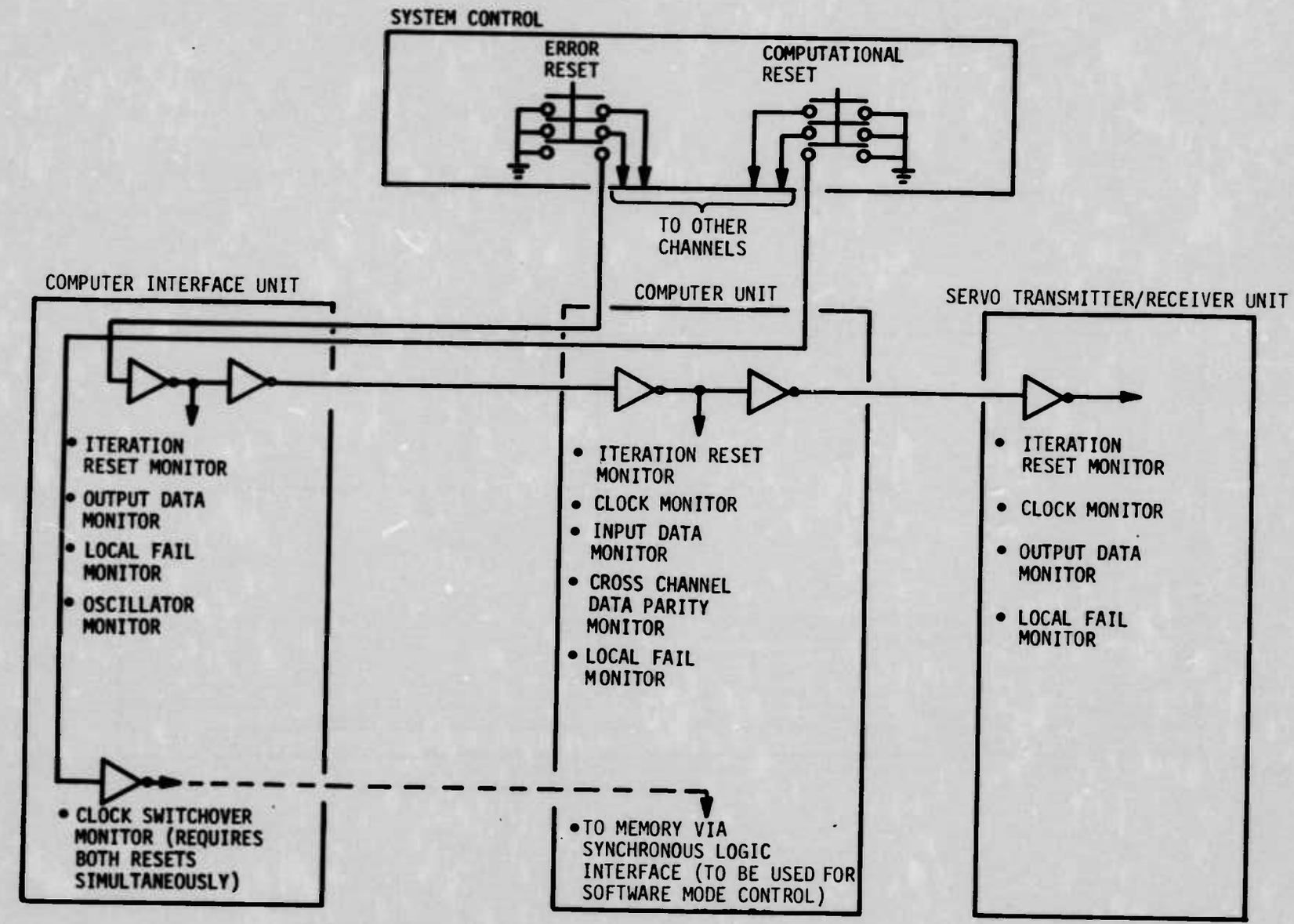


FIGURE C-6.—WWCS ERROR/COMPUTATIONAL RESET CONTROLS

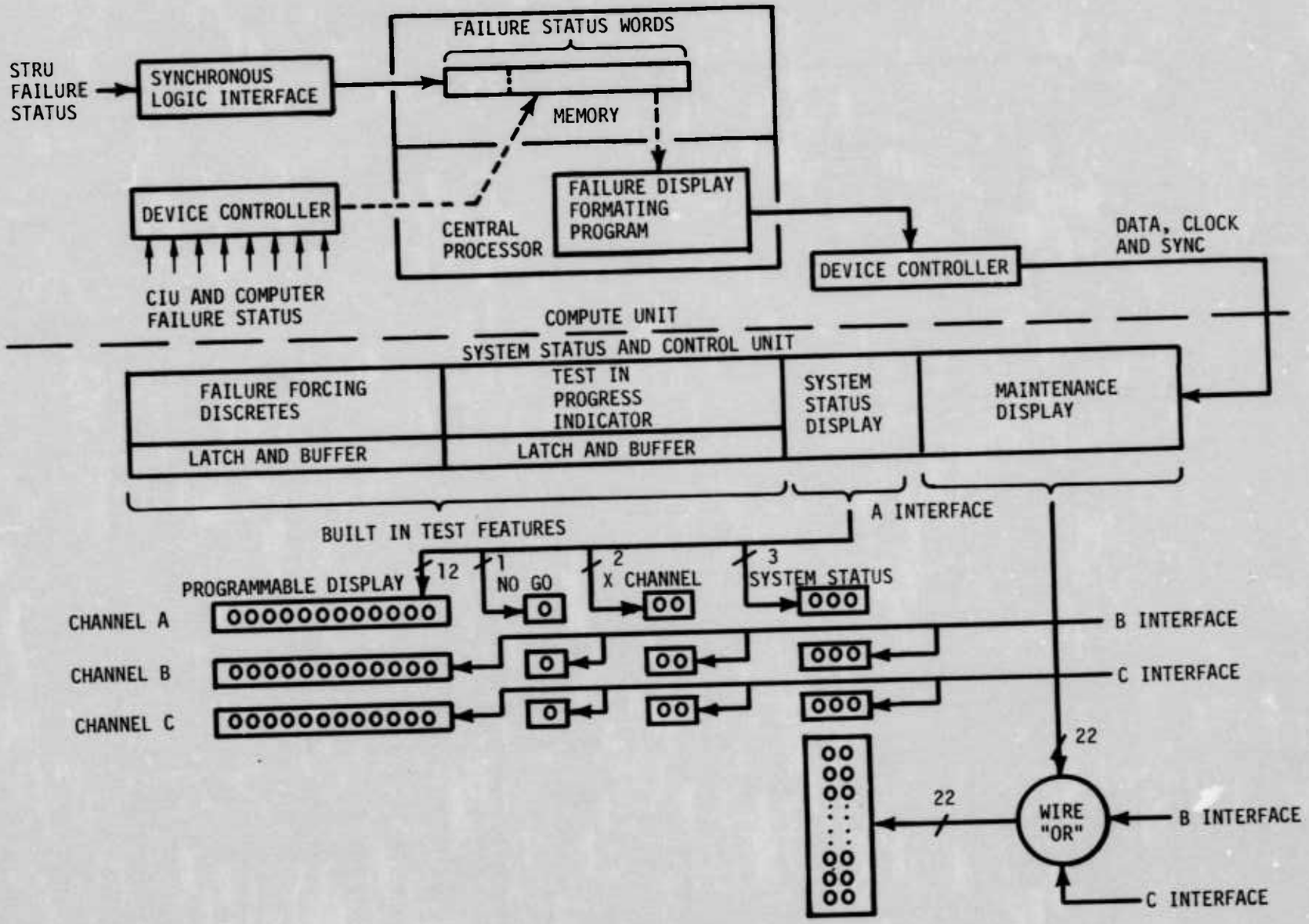


FIGURE C-7.–SYSTEM STATUS INFORMATION FUNCTIONAL FLOW

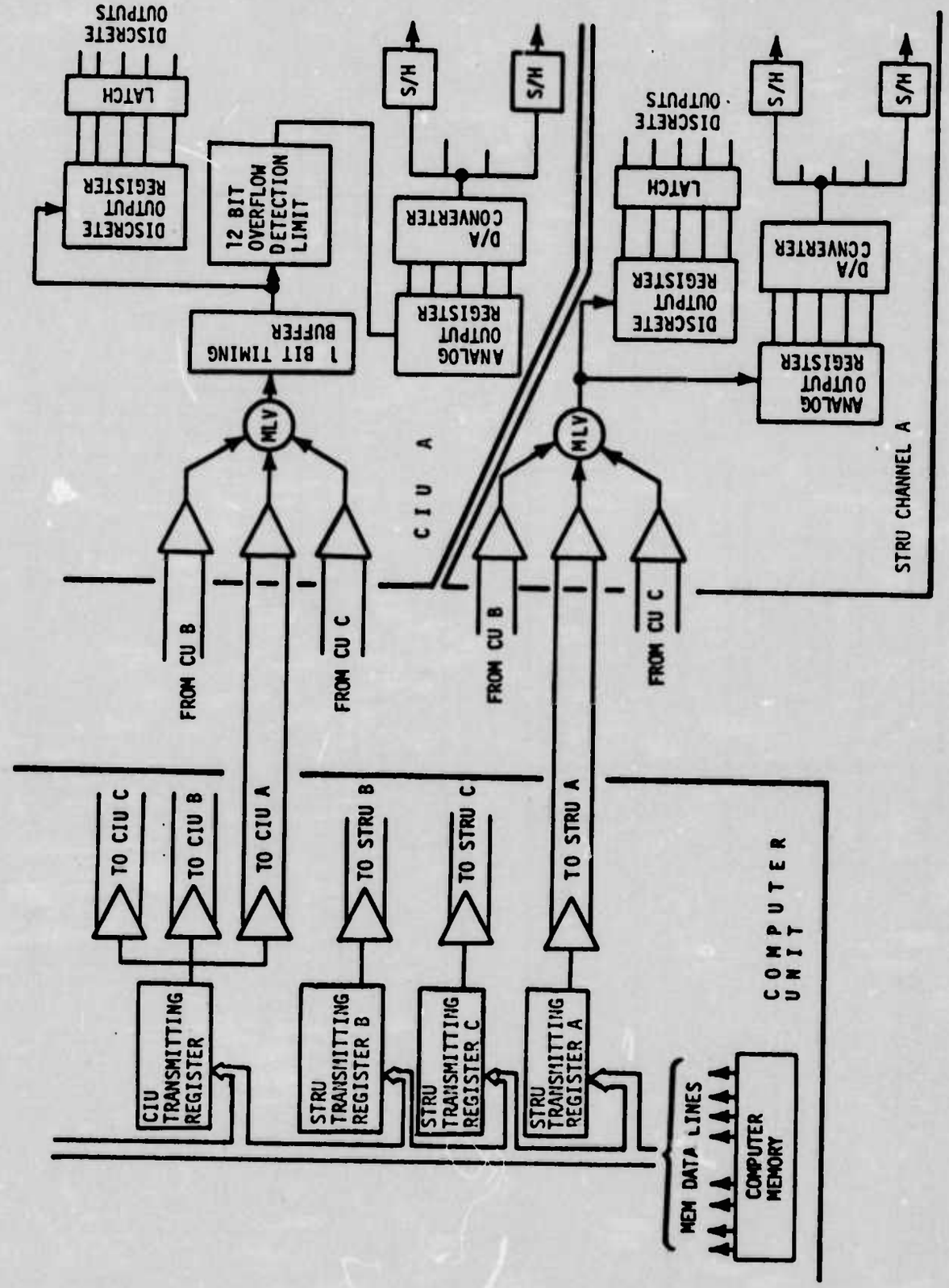


FIGURE C-8.–DIFFERENCE IN COMPUTER UNIT OUTPUT INTERFACE WITH CIU AND STRU

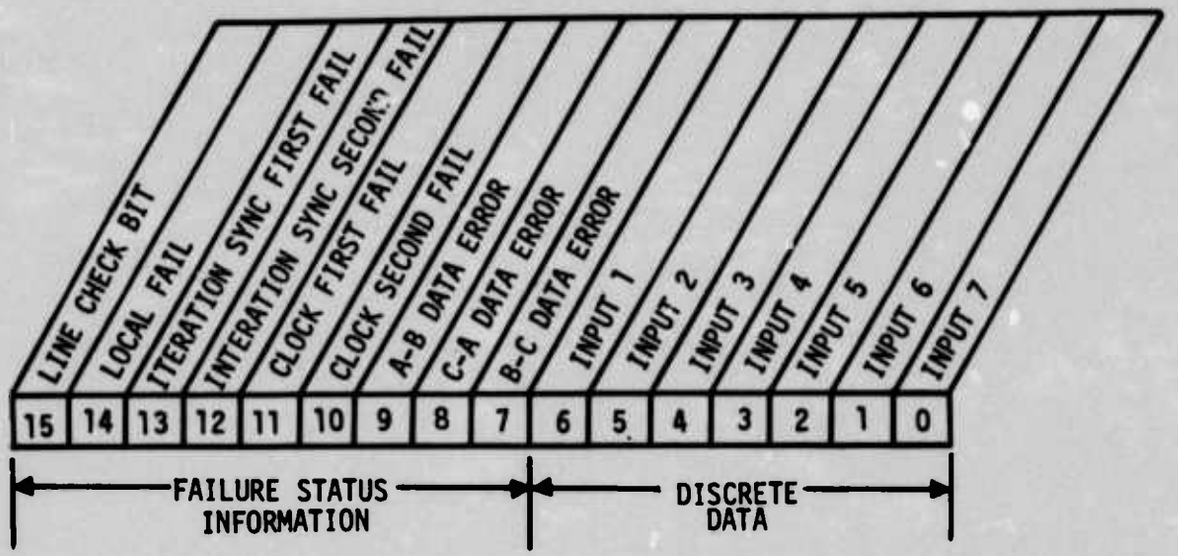
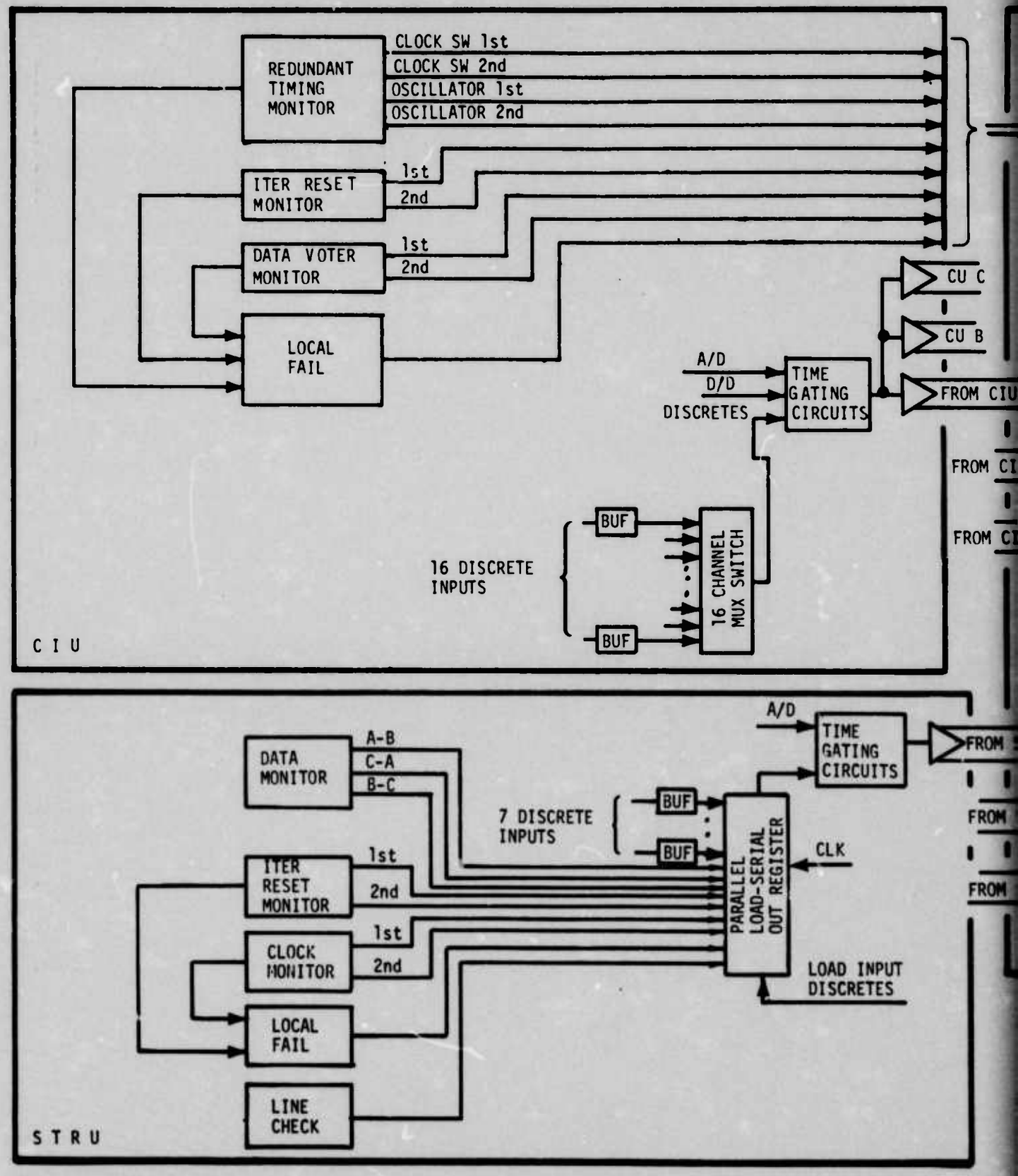


FIGURE C-9.–STRU TRANSMITTED DISCRETE WORD



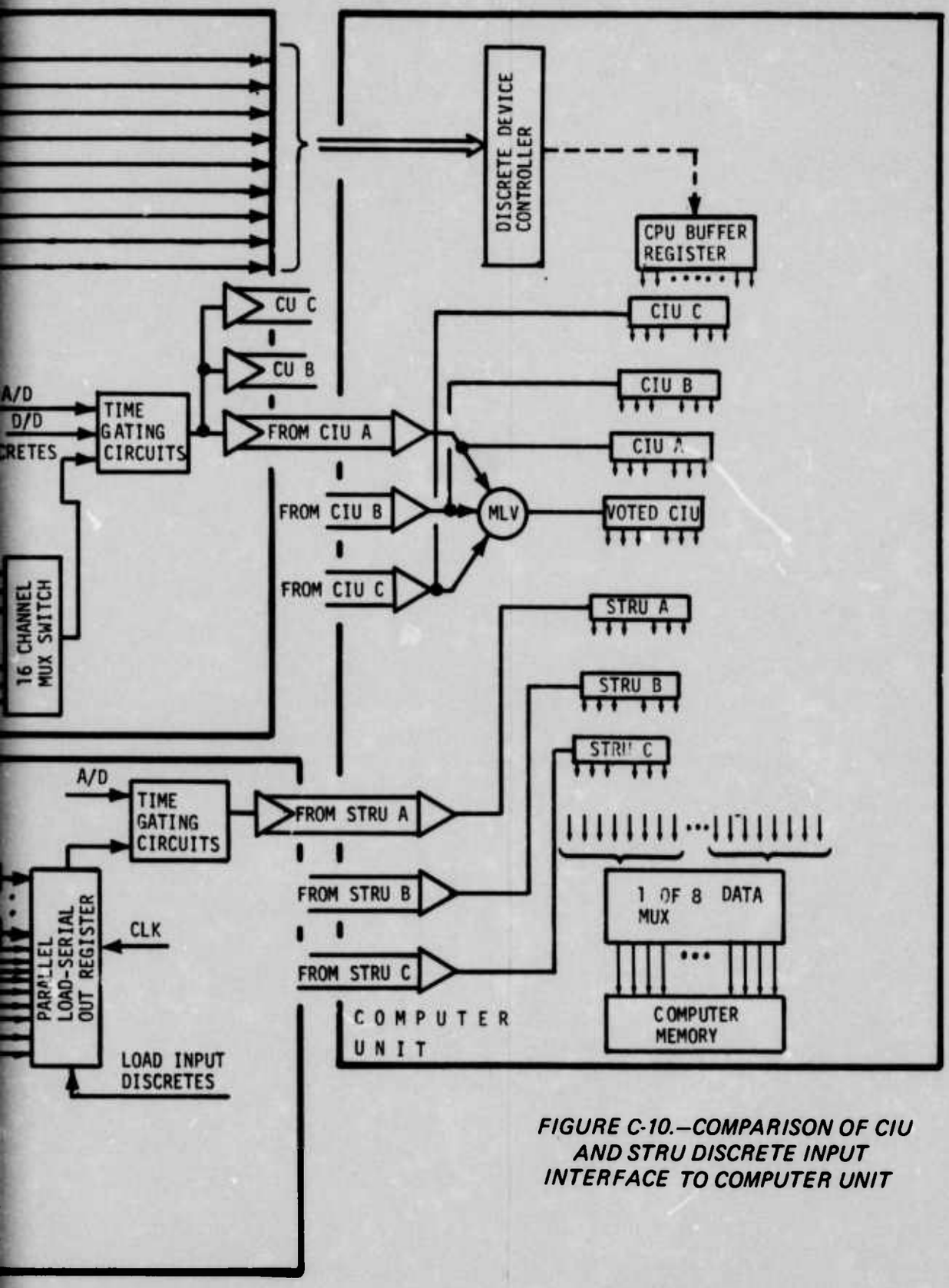


FIGURE C-10.—COMPARISON OF CIU AND STRU DISCRETE INPUT INTERFACE TO COMPUTER UNIT

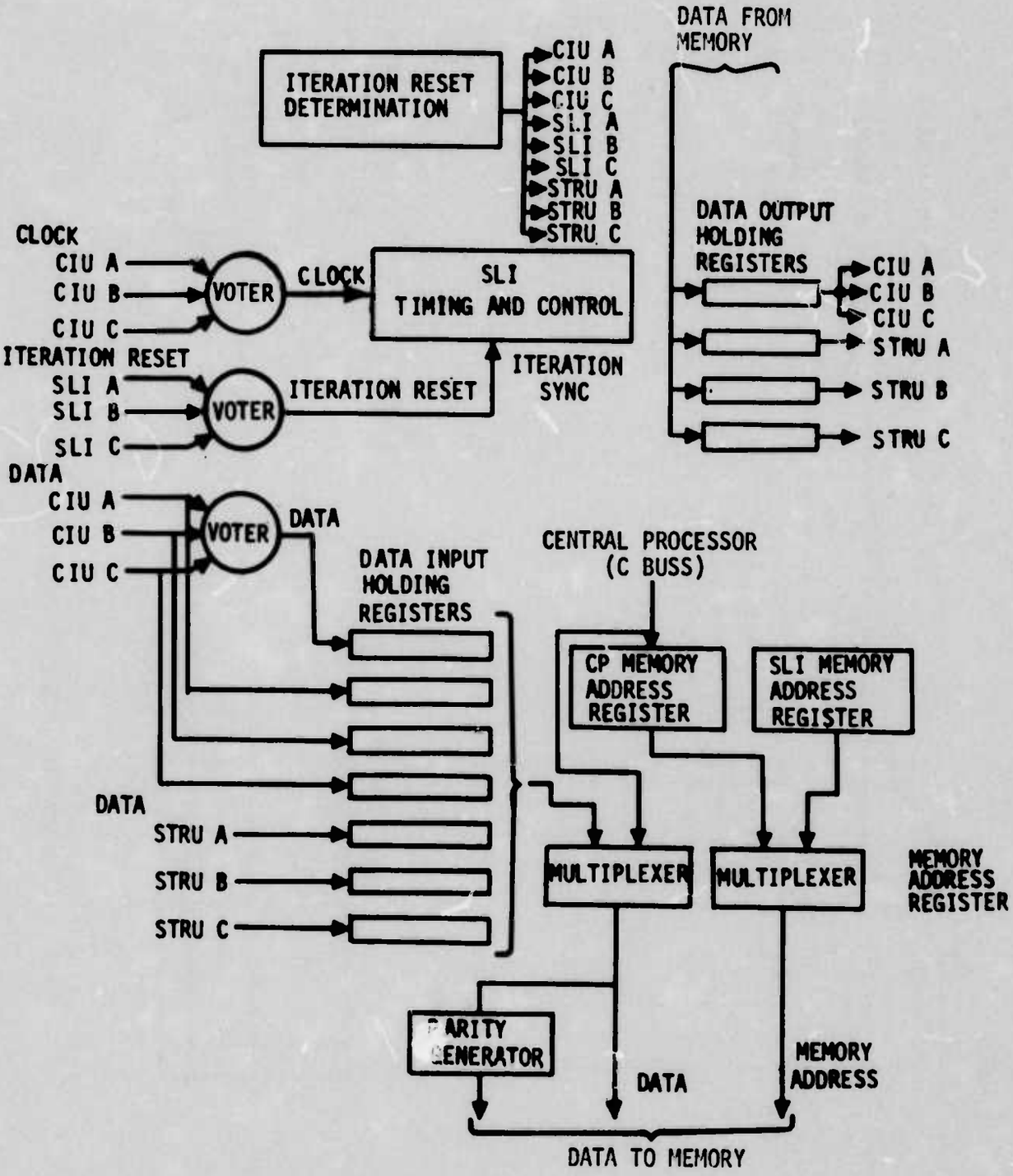


FIGURE C-11. SYNCHRONOUS LOGIC INTERFACE FUNCTIONAL FLOW DIAGRAM

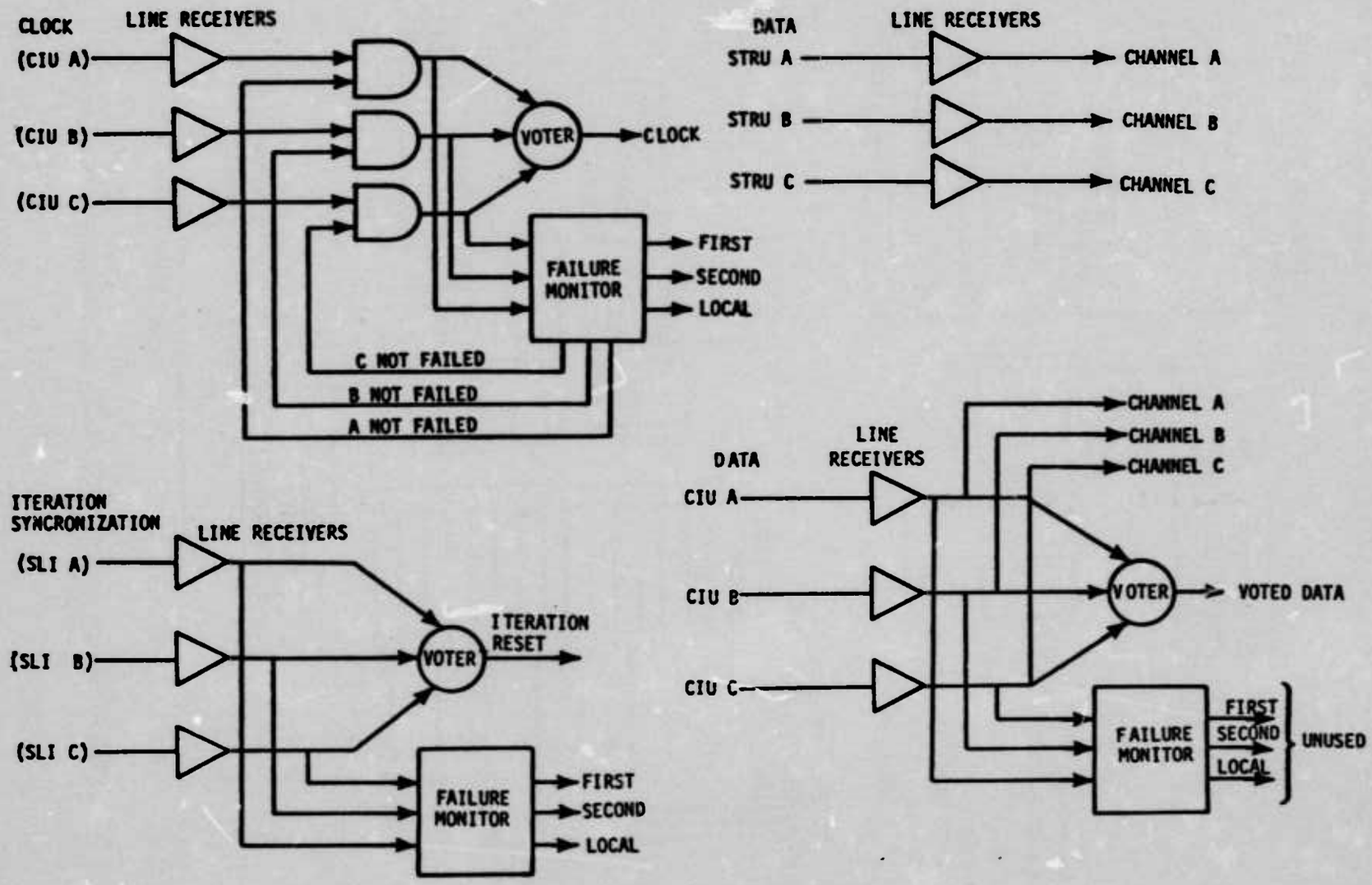


FIGURE C-12.–SLI RECEIVER CIRCUITS

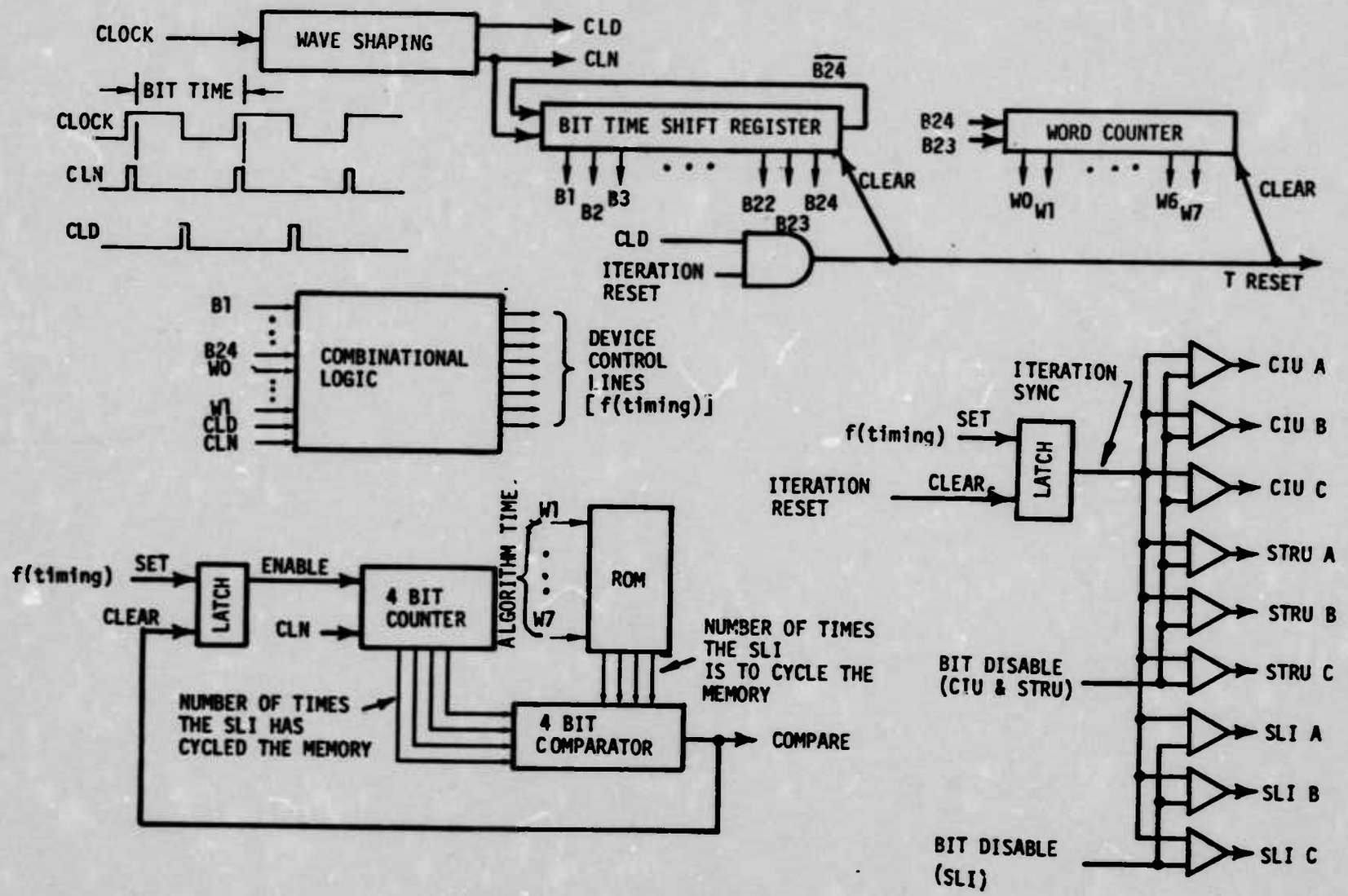
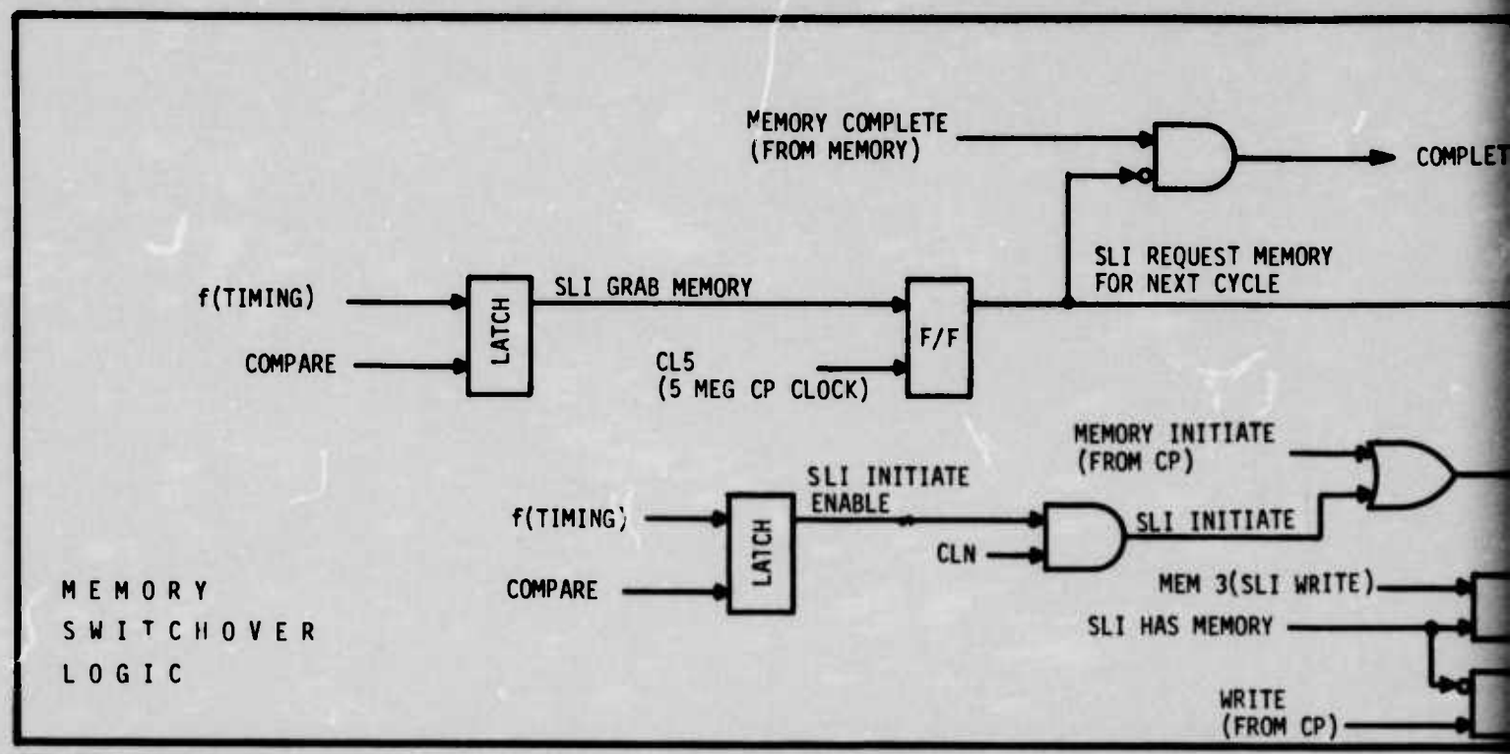
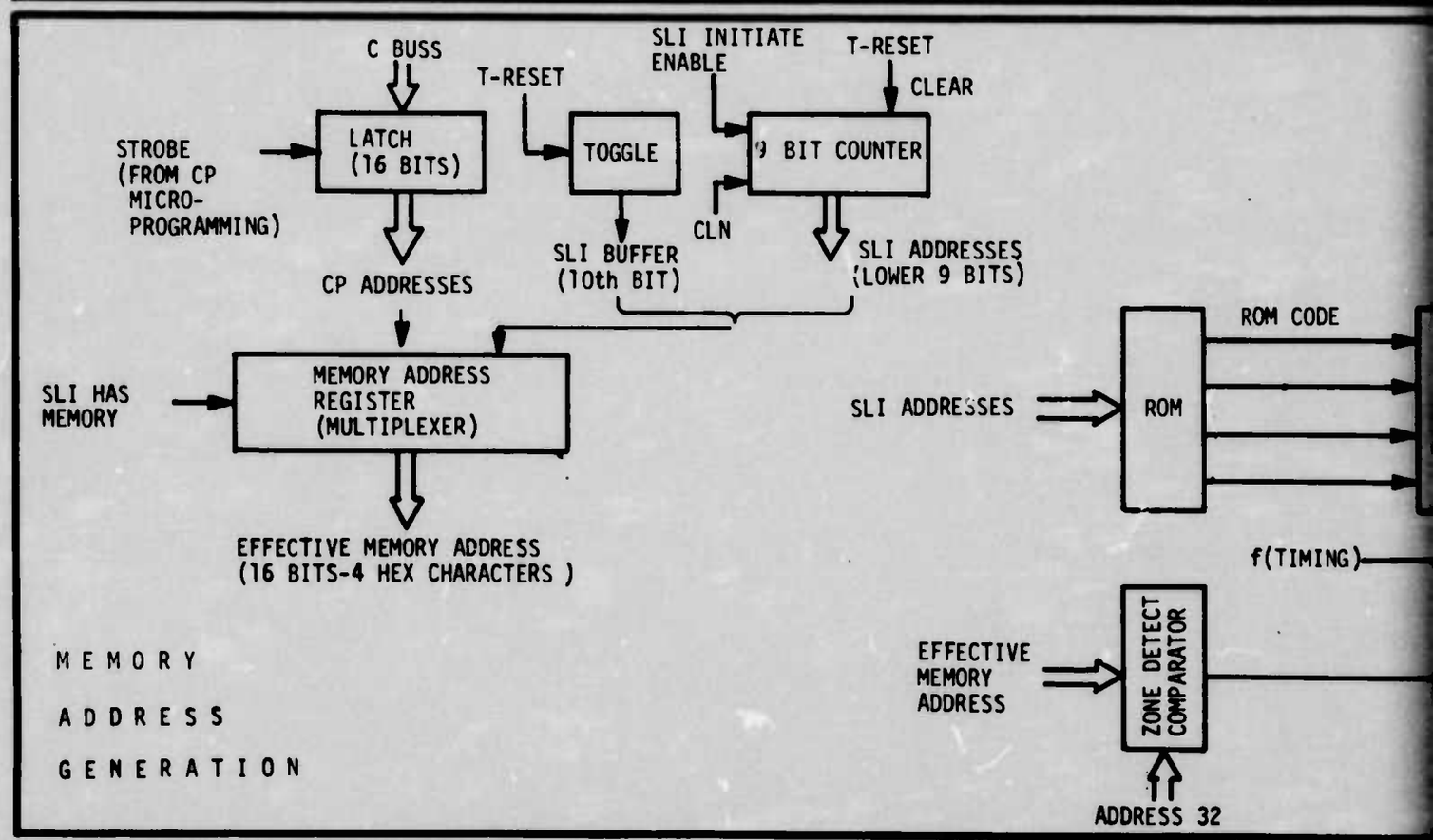


FIGURE C-13.–SLI TIMING AND CONTROL





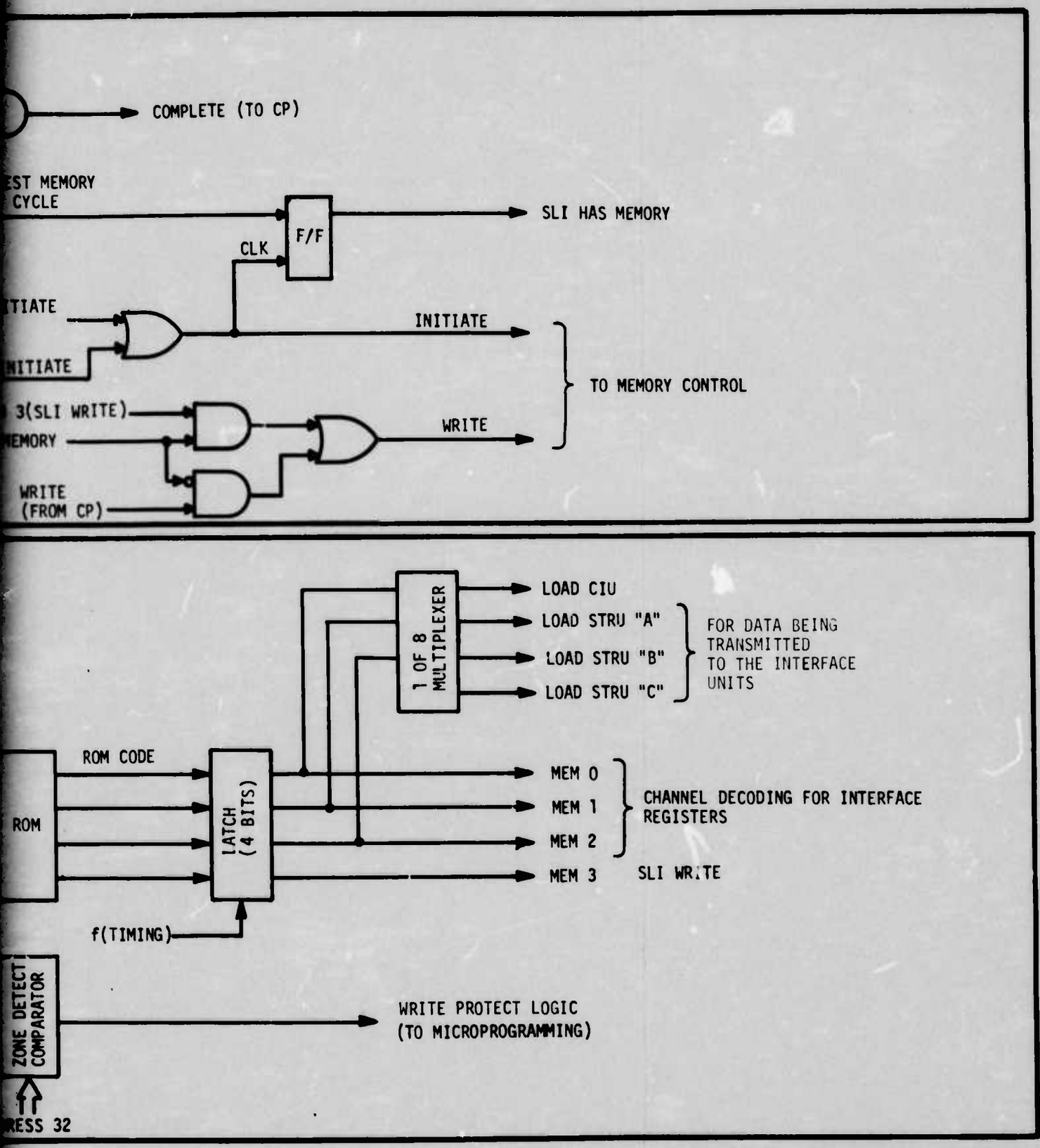


FIGURE C-14.–SLI MEMORY CONTROL

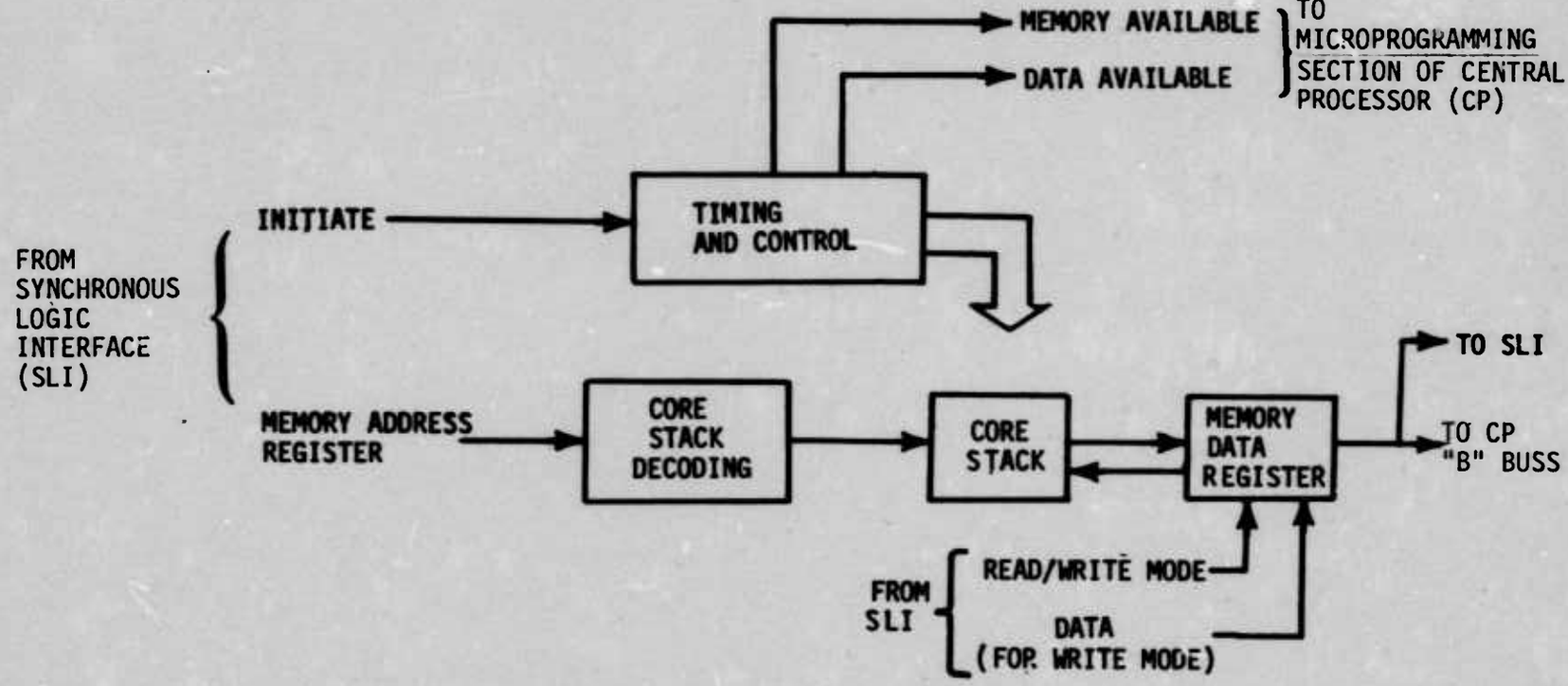
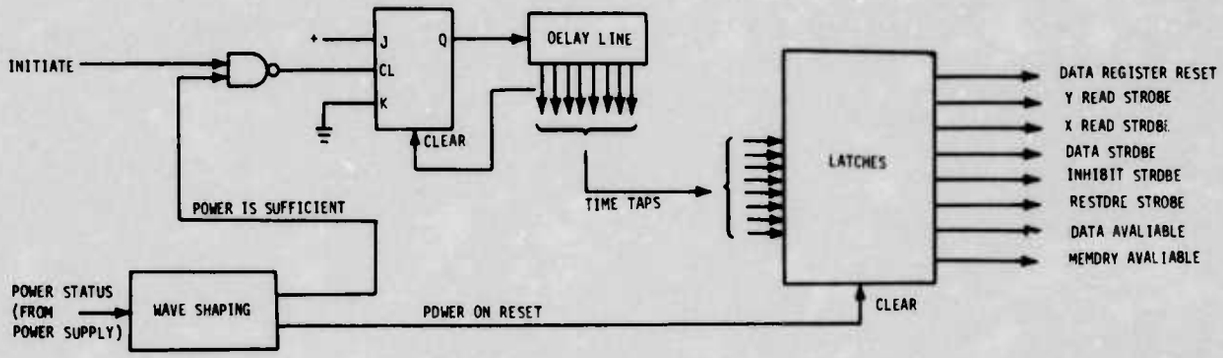
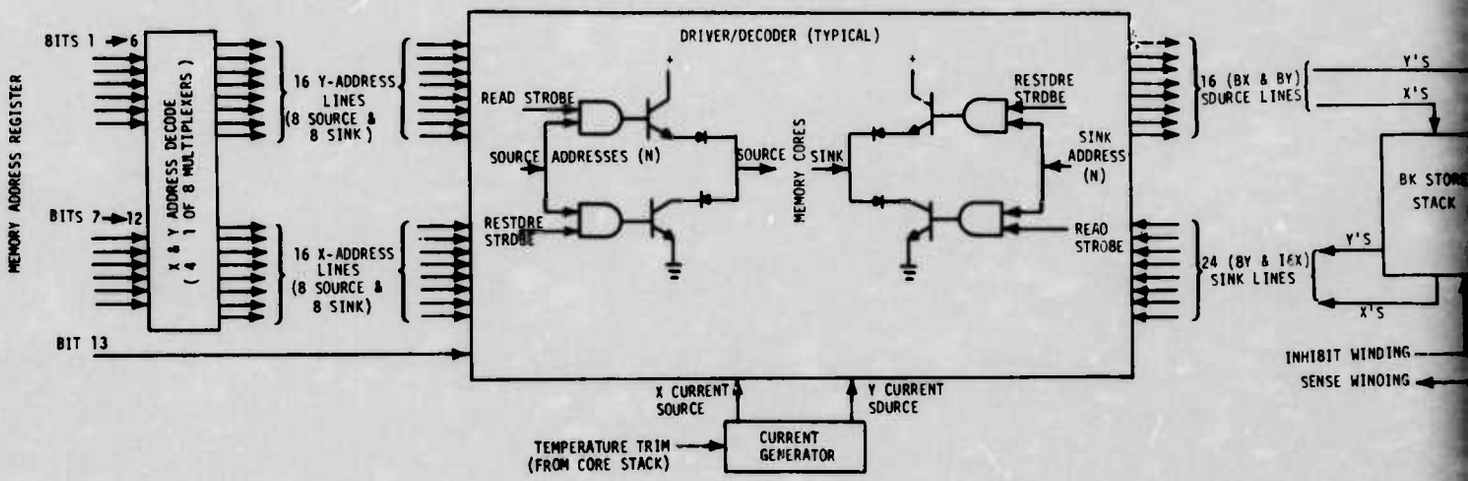


FIGURE C-15.—WWCS MEMORY SYSTEM





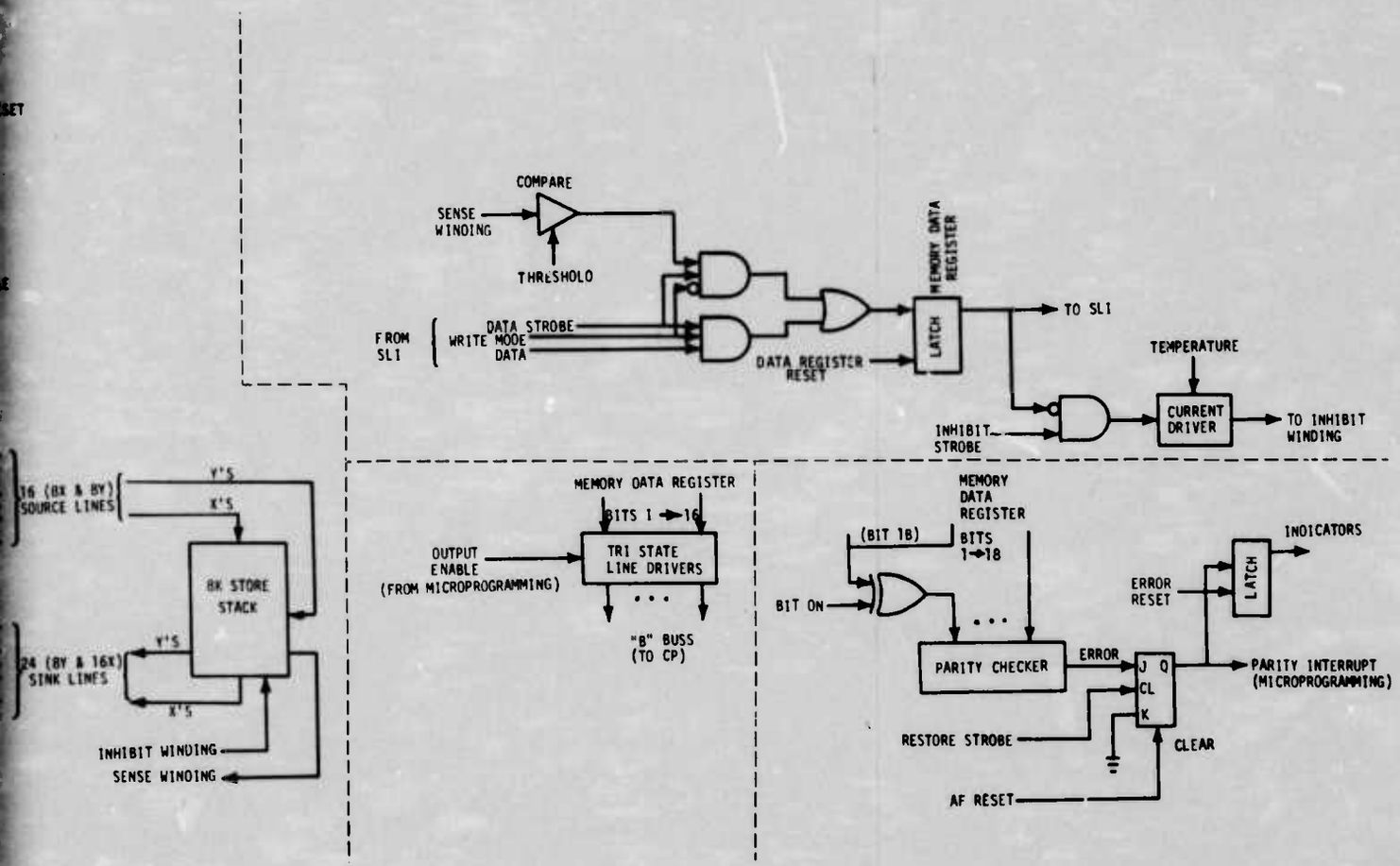


FIGURE C-16.—MCP-703 MEMORY OPERATION

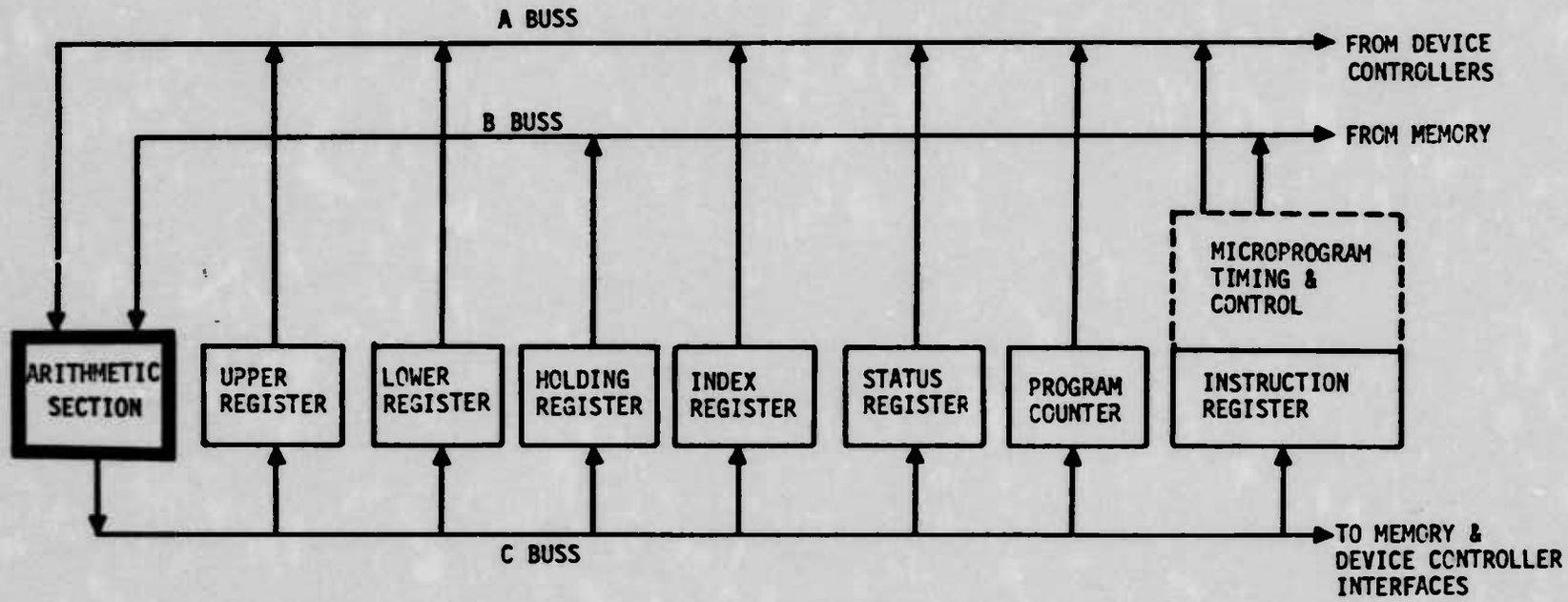
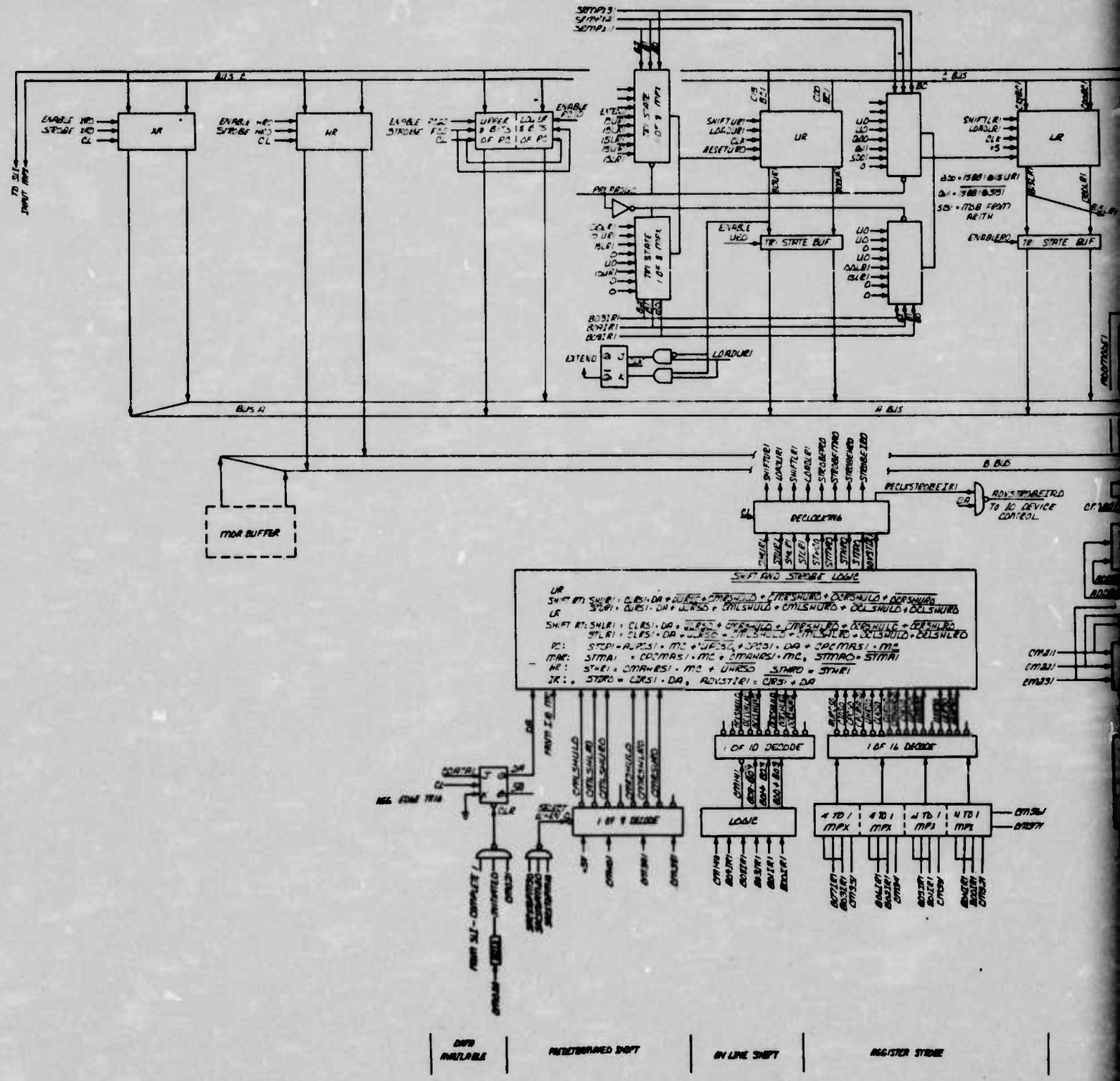


FIGURE C-17.—MCP-701 CONTROL PROCESSOR BUSS STRUCTURE



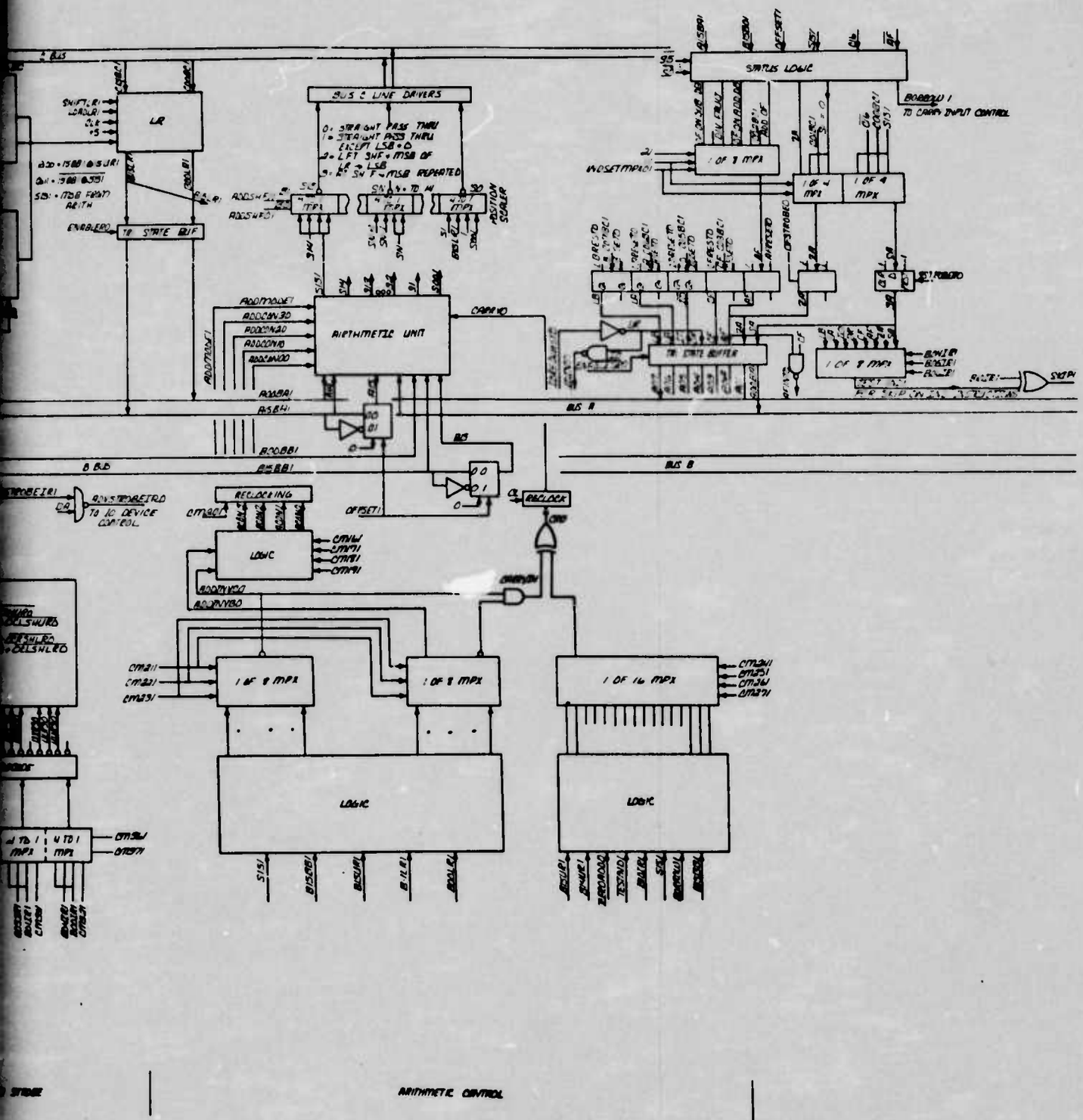
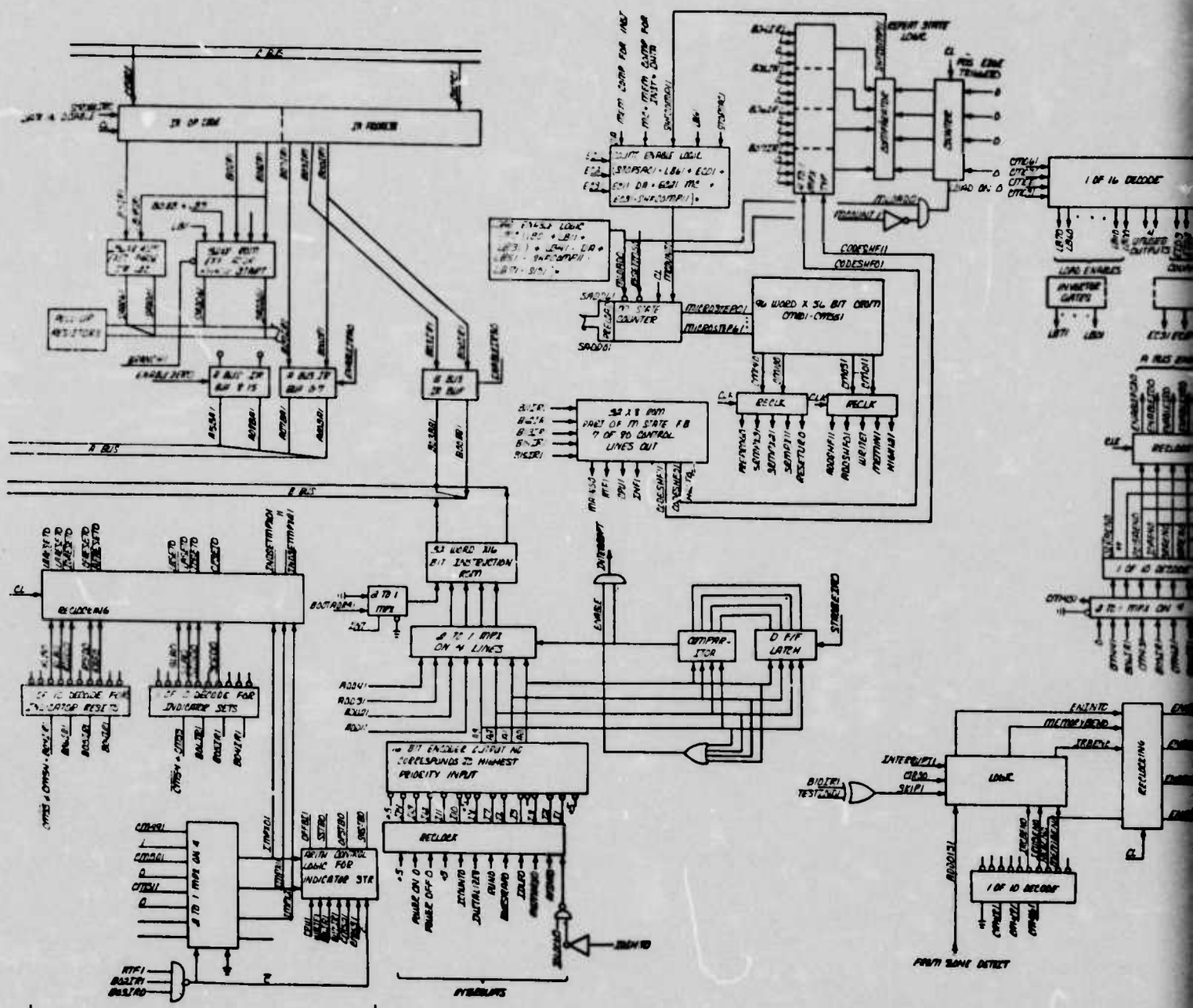
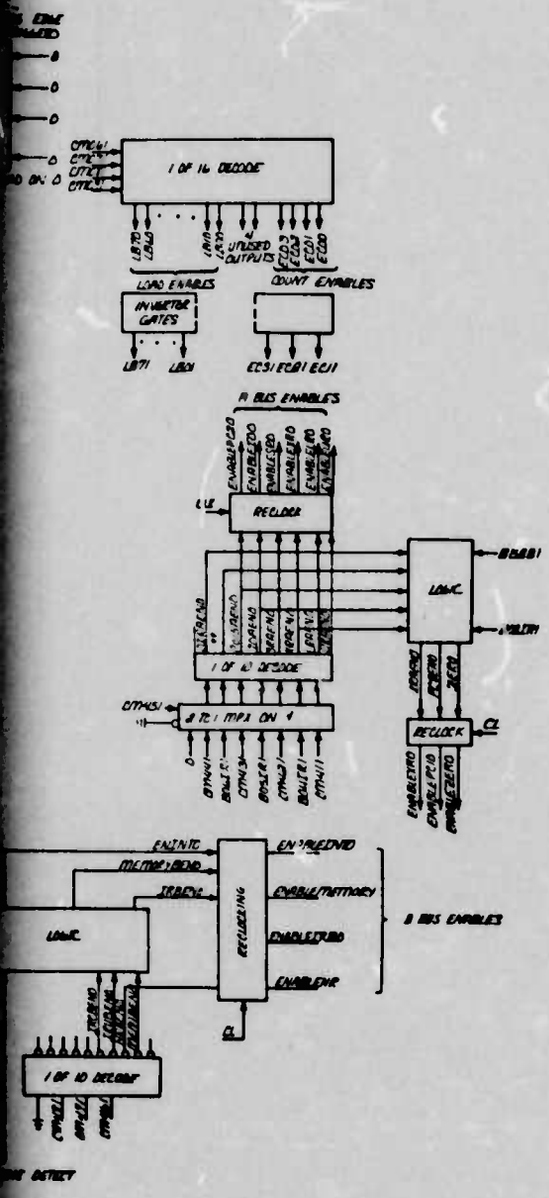
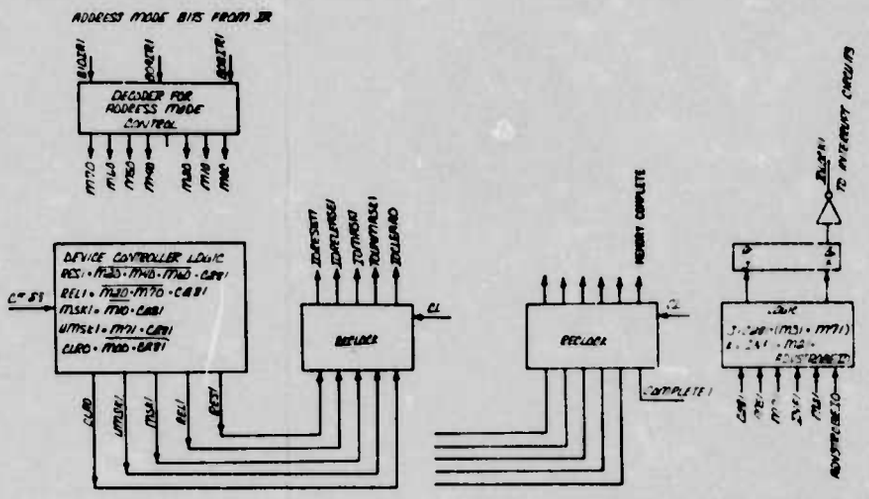


FIGURE C-18.—MCP-701 DESIGN STRUCTURE





A AND B BUS ENABLE



I/O DEVICE CONTROL

FIGURE C-18.–CONCLUDED

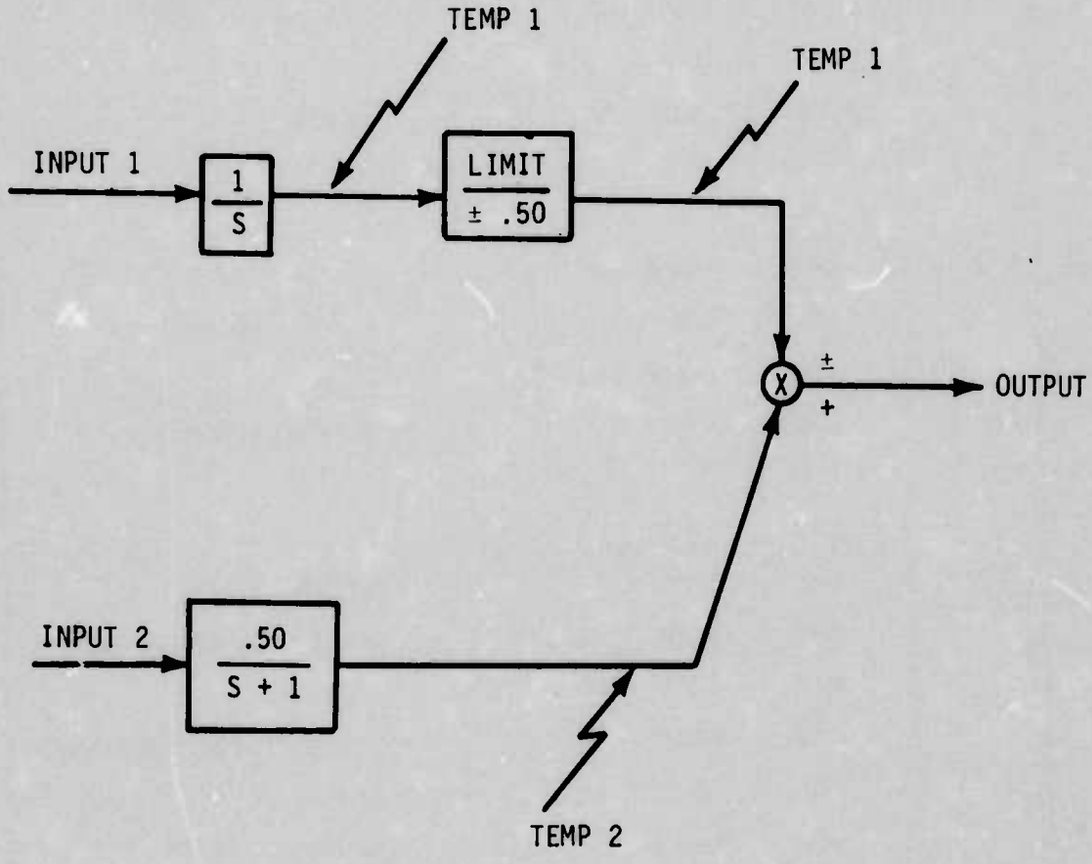


FIGURE C-19.–SIMPLE FLIGHT CONTROL SYSTEM

CODE			COMMENTS
	LDU	INPUT1	
	MPY	=DS15' .006144'	(6.144 m/s FRAME TIME)
	ADU	TEMP1	
	STU	TEMP1	INTEGRATOR
*			
	CPU	=DS15' .50'	LIMITER
	SFMI	GT1	
	LDU	=DS15' .50'	+.5 LIMIT
	JMP	GT2	
*			
GT1	CPU	=DS15' -.50'	
	SFNM	GT2	SKIP FOR NO LIMIT
	LDU	=DS15' -.50'	-.5 LIMIT
GT2	STU	TEMP1	PREVENT SATURATION
*			
	LDU	INPUT2	
	SRS	1, UR	MULTIPLY BY 0.5
	SBU	TEMP2	START FIRST ORDER LAG
	MPY	-DS15' .006144'	
	ADU	TEMP2	INTEGRATE FOR FIRST ORDER
	STU	TEMP2	LAG
	ADU	TEMP1	SUMMING JUNCTION
	STU	OUTPUT	
*			
INPUT1	DS	1	
INPUT2	DS	1	
TEMP1	DS	1	
TEMP2	DS	1	
OUTPUT	DS	1	

FIGURE C-20.—MCP-701 CODE REQUIRED FOR FIGURE C-19

TABLE C-1.—CROSS-CHANNEL DATA LINK FMECA

Failure item	Failure mode	Failure effect	Remarks
1.0 Transmitter			
1.1 Clock toggle	Any	No clock signals.	No operation of transmitter; receiver never contains any data.
1.2 Timing sequencer			
• Initiate signal	Stuck low	Timing sequence never starts.	Same as above.
	Stuck high	Continually initiates sequence.	Marker bit is ORed in every other bit time during the transmission processes. Transmission will be continuous with the validity test most likely failing in the receives.
• Transmitter idle	Any	Proper status relative to transmitter availability not available.	Of no consequence. The time required to make successive transfers in software is much longer than the time to clear the transmitter of previously loaded data.
• Marker bit	Stuck high	Data out; always a one.	Receiver continuously receives data. Validity test fails.
	Stuck low	No turn-on control for receiver.	Receiver left shifts data until first bit appears. Validity test fails.
• Shift enable	Stuck high	Shift register continually sends data.	When loaded, data will begin shifting one bit time early (on top of the Marker bit). Thereafter, only zeros will be sent. Validity may or may not pass depending on the particular element failure in the enable control line (also part of the parity enable chain).
• Load enable	Stuck low	No data shifted.	Only the marker bit sent.
	Stuck high	Shift register acquires all data on C buss.	Indeterminent data pattern transmitted.
• Parity enable	Stuck low	No data loaded.	Only the marker bit sent.
	Stuck high	Parity continually on line.	Parity significance ORed on to transmitted data one bit time delayed.
	Stuck low	No parity transmitted.	Validity test fails.
1.3 Shift register	Fails to shift or fails to load	No data out.	Data significance is zero.
	Output stuck low	No meaningful data out.	Data significance is zero.
	Output stuck high	No meaningful data out.	Forces continual servicing by receiver.
1.4 Parity calculator	Fail to low	Improper parity out.	Validity check fails.
	Fail to high	Improper parity out.	Forces continual servicing by receiver.
1.5 Bit	Stuck high	Complements parity.	Forces continual servicing by receiver.
	Stuck low	Parity can not be complemented.	Validity checker won't pass during computer self-test.
1.6 Line drivers			
• Clock	Any	No clock.	No data transmission.
• Data	Stuck low	No data.	No data transmission.
	Stuck high	No data.	Same as above; except the receiver continues to be turned on.

TABLE C-1.—Continued

Failure item	Failure mode	Failure effect	Remarks
1.7 Device controller timing and control			
• Select F/F	Stuck high	Initiates transfer on any PUT instruction.	Extraneous transmissions to receivers.
	Stuck low	No transfers.	
• Microprogramming control lines	Stuck any position	Improper operation of entire device.	Most likely to be global chaos throughout machine, i.e., winking display of the SSCU panel.
2.0 Receiver			
2.1 Line receivers			
• Clock	Any	No operation.	Any information from channel to which it's attached will not be received.
• Data	Stuck low	No operation	
	Stuck high	Receiver circuits always active.	Requests service every 7.2 μsec.
2.2 Enab'. latch	Stuck high	Receiver circuits always active.	Same except validity check will fail.
	Stuck low	No operation.	
2.3 Shift register	Fails to shift	No data available.	
	Any output stuck	Incorrect data available.	Effect on data significance will depend on location of bit that is failed and the significance of the data transferred.
2.4 Counter	Counts improperly	Data in wrong location in shift register.	Validity check will pass if the count is too long, but will eventually be wrong on short counts.
	Doesn't reset	None	Counter mechanized to put out COMPLETE signal as it overflows. Failure to respond to the clear signal has only an effect at power turn on where the counter may come up with a nonzero value. This means that the first reception will be invalid, like the counter counting short.
	Output stuck high	Continual request for service.	
	Output stuck low	No request for service.	
2.5 Parity checker	Output stuck high	Valid indication always good.	Won't pass a computer self-test.
	Output stuck low	Valid indication always bad.	
	Doesn't respond to clear signal	Every other validity check will indicate incorrectly.	
2.6 Buss buffer			
• Any output bit	Stuck	Improper data to GP.	Same as 2.3.
• Output enable	Stuck high	Information continually on A buss	There are a total of three output enables becuase the buffer is constructed from several devices. One enable is strictly for the validity bit. The other two are associated when address or data are to be on the buss. When data is on, all of the registers are enabled. Address information only enables the lower eight bits. Any of these buss enables stuck on will make total loss of control of the CP because the A buss is required for processing of instructions.
	Stuck low	No information on buss.	

TABLE C-1.—Concluded

Failure item	Failure mode	Failure effect	Remarks
• MUX control	Stuck low	No data available.	The first TAKE will acquire the validity character and the second TAKE will acquire all zeros.
	Stuck high	No validity character.	
2.7 Device controller timing and control			
• Ready F/F	Won't set	No request for service.	
	Won't clear	Continual request for service.	
• Select F/F			
– Output	Won't set	No request for service.	
	Won't clear	Priority out always set.	Prevents any lower priority device from ever obtaining service in either interrupt or by CP demand.
– Priority in	Stuck high	Can never be serviced.	Same as above.
	Stuck low	Priority chain broken.	Eventually results in multiple device interfaces being on the A buss simultaneously (more than one SELECT state).
• Mask F/F	Output stuck high	No interrupt capability.	The receiver can only be serviced by demand from the CP.
	Output stuck low	Can always interrupt.	Processor cannot prevent interrupts from this interface except by the global mask on all device controlled interfaces.
• Priority out	Stuck high	Lower ordered devices not serviceable.	
	Stuck low	Priority chain broken.	Eventually result in multiple device interfaces being on the A buss simultaneously (more than one SELECT state).
• Request	Stuck high	Few interrupts.	If the REQUEST line always remains high, a continual I/O interrupt is *not* generated. Interrupts created for the CP last for only one instruction. If the same interrupt persists, i.e., I/O interrupt, it is ignored on successive instruction fetch cycles. If there is a change in interrupts, such as a memory or arithmetic fault, an I/O interrupt will be generated after the other interrupt is cleared.
• Microprogramming control lines	Stuck any position	Improper operation of entire device controller interfaces.	

TABLE C-2.–SLI FAILURE MODE EFFECTS ANALYSIS

Failure item	Failure mode	Failure effect	Detection					Remarks
			Output data monitor	Sensor failure monitor	Iteration reset	Clock	None	
1.0 Receiver circuits								
1.1 Line receivers								
• Clock	Any	Improper wave shape to voter.				X		
• Iteration reset	Any	Improper wave shape to voter.			X			
• CIU and STRU	Any	Improper data pattern.		X				Since all bits of a single word pass serially through a single device, the data significance for a failure should be ascertainable by the software monitoring program. Hard failures to zero or one output potentially represent the harder value to detect since the significance of the data is nearly zero (2's complement). Any normal perturbation should unmask this case, however.
1.2 Voters/failure monitors	Discussed in appendix A							
2.0 Timing and control								
2.1 Wave shaping	Discussed in appendix A		X	X				Most errors produce no clock signal, which, of course, would force the interface to nonfunction. Off design clock signals would end up producing erroneous data from all channels with the resulting outputs to the interface units flagging the fault.
2.2 Bit time and word counter	Discussed in appendix A							Same as above.
2.3 Generation of COMPARE signal								
• ROM	Any incorrect value	Excludes or extraneous I/O placed into memory. SLI Address counter not synchronized for remainder of frame.	X	X				Input data windows from interface units would not contain the proper data.
• Counter	Doesn't count	COMPARE never obtained.	X					Depending upon when the counter stops, either the SLI or the CP will have the memory all the time.
• Comparator	Stuck low	COMPARE never obtained.	X					Same as above.
	Stuck high	No data stored or retrieved.	X					
2.4 Combinational logic	Any control line stuck high or low	Some SLI data handling function will cease.	X	X	X			Depending upon the functions not working, the appropriate monitor will flag the error.
2.5 Iteration reset								
• Transmitters	Stuck high or low	Iteration sync not sent to some LRU's.			X			Iteration reset in some LRU, if not in this one.
• Latch	Stuck low	No iteration sync.			X			
	Stuck high	Iteration sync sent continuously.			X			

TABLE C-2.—Continued

Failure item	Failure mode	Failure effect	Detection					Remarks
			Output data monitor	Sensor failure monitor	Iteration reset	Clock	None	
3.0 Memory control								
3.1 Memory switchover logic								
• SLI grab memory	High or low	No CP funcitons or no SLI functions.	X					
• SLI request memory	High or low	No CP functions or no SLI functions.	X					
• SLI has memory	High or low	Memory data register locked to either SLI or CP.	X					
• SLI initiate enable	Low	No SLI memory functions requested.	X					
	High	Overlapping memory requests.	X					Will generate excessive parity errors and may result in total loss of program.
• Initiate	High or low	No memory functions (machine stops).	X					Memory control requires the initiate signal (to go both high and low).
• Complete	Low	No CP operations (machine stops).	X					Will generate excessive parity errors and may result in total loss of program.
	High	Overlapping memory initiates.	X					
3.2 Address generation								
• CP address latch	Any bit stuck	Improper data (or instruction) fetched.	X					Entire program may run with control.
• T-reset toggle	Stuck to any value	SLI would have use of only one buffer.	X					Software, as mechanized during program, would use the buffer that is receiving no updates. If the software was using the same buffer as the SLI, there would probably be no degradation of performance, but the data monitor would trip on the LSB's, which would be different from the other channels.
• SLI address counter	Miscounts	Excludes or extraneous I/O placed into memory. SLI Address counter not synchronized for remainder of frame.	X					
	Any bit stuck high or low	Data stored (retrieved) from wrong memory location.	X	X				If the failure is in the upper six bits, which are fixed, the entire data buffer is moved to either a nonexistent location in memory or it will overlay program instructions. If the failure is in the lower nine bits, half of the buffer will be unused and the other half overlayed twice each frame.
• Memory address	Multiplexer stuck	Either SLI or CP utilizing improper memory region.	X	X				May result in total loss of program.
	Any output stuck	Improper data (or instruction) fetched.	X					
• Channel decoding circuitry (ROM or latch)	Improper value	Wrong register retrieved/stored into a particular cell.	X	X			X	Monitor tripped depends on location failed whether the WRITE control bit is involved. No failure indication for any unused input; however, an unused output would most likely be flagged by the data monitor.
	Any output stuck	Loss of data from half of the interfaces, or the inability to read or write from/to memory.	X					

TABLE C-2.—Concluded

Failure item	Failure mode	Failure effect	Detection					Remarks
			Output data monitor	Sensor failure monitor	Iteration reset	Clock	None	
• Register load control multiplexer	Any output stuck	Improper data sent to interface units.	X					
• Zone detect comparator	Output stuck high	Instructions fetched only from ROM.	X					Machine stops functioning.
	Output stuck low	ROM never utilized.	X					Machine would not execute any interrupt routines.
	Compares on incorrect value	Improper entry into ROM.	X					Most likely to affect the zero page indirects, thereby causing the program to lose control.
4.0 SLI data registers								
4.1 Input shift registers	Fails to shift data in	No values from that interface.		X				
	Any output bit stuck	Improper values from that interface stored into memory.	X	X				If the bit failure is in the lower order positions such that signal selection does not catch the failure, then the output monitor would detect because the CP would process and put out erroneous data in one channel
4.2 Oata multiplexer	Any	Improper values from that interface stored into memory.	X	X				
4.3 Parity generator	Any	Improper parity stored into memory.						Detection by parity detection circuitry in memory interface.
4.4 Output registers	Any	Improper data sent to interface units.	X					
4.5 Line drives	Any	Improper data sent to interface units.	X					

TABLE C-3.—WWCS MEMORY FAILURE MODES ANALYSIS

Failure item	Failure mode	Failure effect	Detection			Remarks
			Output data monitor	Memory parity	None	
1.0 Timing						
1.1 Initial latch	Fails to set; clear	Memory will process data.	X			
	Switches to frequently	Multiple timing pulses.		X		Result of error in microprogramming such that the memory complete signal is ignored, and the memory is requested to perform more frequently than once every 950 ns.
1.2 Power status wave shaping	Outputs fail low	Memory stops.	X			
	Outputs fail high	None.			X	Only indication will be a parity error on the first execution of memory. Control latches are in an improper state. After the first memory cycle, the timing chain will have self-corrected.
1.3 Delay line	Any tap open	Some control states not set; cleared.	X	X		Machine stops.
	Delay time between taps shortened	Control strobes shortens.		X	X	Some amount of tolerance would go unnoticed until the strobes become so short that the magnetic operation of the core stacks fall outside the strobe windows.
	Delay time between taps lengthened	Memory cycle time increases.	X	X		SLI will use the machine at 1 μsec rate regardless. If timing sequence is not complete, improper control states will develop.
1.4 Latches	Any	Sames as 1.2.				
2.0 X-Y address decode	Improper multiplexer decoding	Different cells utilized.	X		X	Depends upon time of failure. If failure occurs after the program is loaded, program is loaded, program execution control will be lost. If failure is before loading, the program will just be distributed in a different fashion throughout core. This should have no effect on processing as long as information is not overlayed.
	Any output stuck high	Two simultaneous words retrieved from memory.		X		The contents of the two words will probably not be ORed into the memory data register, for the current source won't have enough drive capability for two simultaneous words. Thus, the MDR will end up cleared.
	Any output stuck low	Inability to access 1/16th of the core distributed throughout memory.	X		X	Would go undetected only if a very, very small program was loaded so that the unaccessible core was not required. Then, this represents a "don't care" failure.
3.0 Driver/decoder	Any output open	Loss of access to 1/16th or 1/24th of memory, depending if one is on source or sink line.		X		MDR would receive only zeros.
	Any output shorted	Same as above.		X		Same as above.
4.0 Current generator	Output current too high	Excessive core stack noise.		X		The effect will OR in addition bits into the MDR from neighboring cores.
	Output current too	Not enough energy to switch cores.		X		Some bits not read or restored to or from memory.

TABLE C-3.—Concluded

Failure item	Failure mode	Failure effect	Detection			Remarks
			Output data monitor	Memory parity	None	
5.0 Core stuck	Break in X or Y line	Same as 3.0.		X		
	Broken core cell	Loss of bit significance.		X	X	Latent to "don't care" failure potential. If the cell significance is lost, and if it is in the mainstream of processing, parity will detect failure. If the failure is to a bit significance identical to the extended use, it is a "don't care" failure. If the significance is lost, and if it is not in the mainstream, the failure is latent.
	Broken sense/inhibit winding	Loss of bit significance in all words.		X		
6.0 Memory data register						
6.1 Sense amplifier	Any	Erroneous data in the same bit position for all words.		X		
6.2 Control lines						
• Data strobe	Stuck low	No data into MDR.		X		
	Stuck high	Extraneous data into MDR.		X		
• Write mode	Stuck low	Loss of data to be stored.		X		
	Stuck high	No information read from memory.	X	X		Parity will come up if this control line is lost at only the individual bit locations in the MDR. If the failure mode is global, parity will be satisfied, but the program execution will be without control.
• Data register reset	Stuck low	Same as above.		X		
	Stuck high	Register never clears.		X		
6.3 Latch	Output stuck any position	Improper data out.		X		
6.4 Inhibit driver						
• Output drive	High or low	Improper data stored into memory.		X	X	The loss of the data will not be detected until it is requested at a later time. This is not considered latent, because it would be lost in the same bit position for all words. The time between restoring and then requesting any of the previously requested words will be short.
• Control inputs	High or low	Same as above.		X		
7.0 Output buss drives	Any output stuck	Improper data to CP.	X			Loss of execution control.
8.0 Monitor						
8.1 Parity checker	Output fails low	No monitoring			X	Latent.
	Output fails high	Continual interrupt.		X		
8.2 Interrupt latch	Output high or low	Same as 8.1.		X	X	
8.3 Indicator latch	Output fails high or low	Improper operation of indicator.			X	Machine operation unencombered.
8.4 BIT circuitory						
• Output	Stuck high or low			X		
• BIT on	Fails high	Continual interrupt.		X		
	Fails low	None.			X	Inability to generate error during test of memory parity.

TABLE C-4.–FUNCTIONAL FAILURE MODES ANALYSIS OF MCP-701 INSTRUCTIONS

Instruction	Operation	Failure	Failure mode/comment
ADU X (add to upper register)	(X) + UR → UR SA, ZA, AF	(X) ? + UR → UR (X) + ? UR → UR (X) + UR ? → UR (X) + UR → ? UR (X) + UR → UR ? SA ? ZA ? AF ?	Immediate Immediate–but could be latent if the signals were not active Immediate Immediate Immediate Trivial (not used in program) Trivial (not used in program) Immediate–depends on AF software
MPY X (multiply)	(X) + UR→UR SA, ZA, AF	(X) x UR → UL (X) x ? UR → UL (X) x UR ? → UL (X) x UR → ? UL (X) x UR → UL ? SA ? ZA ? AF ?	Immediate Latent possibility–because of high number of states Immediate Immediate Immediate Trivial Trivial Immediate
STU X (Store upper register)	(UR) → X SA, ZA	(UR)?→X (UR)→? X (UR)→X? SA ? ZA ?	Immediate Immediate Immediate Trivial Trivial
CPU X (Compare to upper register)	UR - (X) SA ZA	UR ? - (X) UR - ? (X) UR - (X) ? SA ? ZA ?	Immediate Latent possibility/trivial Immediate Immediate Trivial (not used in this program)
JMP X (Jump unconditionally)	X → PC	X ? → PC X → ? PC X → PC?	Immediate–program loses control Immediate–program loses control Immediate

SFMI N
(Skip forward on minus)
	IF SA = 1
	
	PC + N → PC
	ELSE
	PC + 1 → PC
		IF(SA=0)PC+N → PC
		IF(SA=1)PC?+N→PC
		IF(SA=1)PC+?N→PC
		IF(SA=1)PC+N?→PC
		IF(SA=1) PC+N→?PC
		IF(SA=1)PC+N→ PC?
		SA?
			immediate
			Immediate
			Immediate
			Immediate
			Immediate
			Immediate
			Trivial–not retested after SFMI until set again

SFNM N
(Skip forward on not minus)
	IF(SA=0)
	PC+N → PC
	ELSE
	PC+1 → PC
		IF(SA=1)PC+N→PC
		IF(SA=0)PC?+N→PC
		IF(SA=0)PC+?N→PC
		IF(SA=0)PC+N?→PC
		IF(SA=0)PC+N→ ?PC
		IF(SA=0)PC+N→ PC?
		SA ?
			Immediate
			Immediate
			Immediate
			Immediate
			Immediate
			Immediate
			Trivial–not retested after SFNM

TABLE C-4.—Concluded

Instruction	Operation	Failure	Failure mode/comment
SRS 1, UR (Shift right, repeat sign)	UR shifted 1 → UR	UR?shifted 1 → UR UR shifted 1? → UR UR shifted 1 → UR? UR shifted 1 → UR	Immediate Immediate Immediate Immediate
SBU X (Subtract from upper register)	UR - (X) → UR SA, ZA, AF	UR ? - (X) → UR UR -?(X) → UR UR - (X) ? → UR UR - (X) → ?UR UR - (X) → UR ? SA ? ZA ? AF ?	Immediate Immediate—but could be latent if signals were inactive Immediate Immediate Immediate Trivial Trivial Immediate

APPENDIX D

WWCS BUILT-IN-TEST

Built-in-test (BIT) refers to those elements of hardware and software within the MCP-703 system, WWCS, that directly contribute to the subsystem self-administered operational integrity check. The primary responsibility for the design/development of the BIT was given to the General Electric Company as part of their role in the development of the WWCS. The design of BIT was not the direct result of a detailed FMECA but evolved parallel to the hardware design and various failure mode studies in progress as the hardware was being constructed. The basic structure of BIT followed the guidelines presented in section 4.3.3 of this document.

Table D-1 contains a detailed tabular sequence of the tests within the BIT software program. Tests that are looped through three times in order to test the permuted combinations of channel failures are considered a test group. There are six such groups; they deal with testing the MLV and their associated monitors. These MLV test groups are, in general, conducted as follows:

The normal MLV logic is defined as $V = A \cdot B + B \cdot C + C \cdot A$, where,

V = voted output

A, B, and C = inputs from channels A, B, and C, respectively

$+$ = logical OR operation

$\cdot$ = logical AND operation

The MLV test steps are:

Pass through group	Step number	Fail	Operations verified
Pass 1	1	A	$V = B \cdot C$ and channel A will indicate first failure
	2	A B	$V \neq C$ and second failure will indicate
Pass 2	3	B	$V = C \cdot A$ and channel B will indicate first failure
	4	B C	$V \neq A$ and second failure will indicate
Pass 3	5	C	$V = A \cdot B$ and channel C will indicate first failure
	6	C A	$V \neq B$ and second failure will indicate

Test steps 2, 4 and 6 verify that no single channel can drive the output.

TABLE D-1a.—MCP-703 BUILT-IN-TEST TEST SEQUENCE

TESTING OF SSCU DISPLAY CHECKOUT (TEST 1)			
Test number	Test step description	Comments	Expected indicator lights
001 step A	The SSCU display is cleared and program waits for operator verification	The SSCU display should be clear, except the executive may have reloaded the run or timer lamps. The operator should verify that the display is clear and push BIT continue.	TIP* = 001
001 step B	Indicator lamps are driven from channel A	The TIP* display is set to 888 to verify that all segments of each numeral will light. The top row of lights and the two columns of 11-failure indicators will be lit from channel A. The operator should verify the display and push BIT continue.	TIP = 888 All lights driven by channel A
001 step C	Indicator lamps are driven from channel B	The TIP display will remain 888. The second row of lights and the two columns of 11-failure indicators will be lit from channel B. Operator should verify the display and push BIT continue.	TIP = 888 All lights driven by channel B
001 step D	Indicator lamps are driven from channel C	The TIP display will remain 888. The third row of lights and the two columns of 11-failure indicators will be lit from channel C. Operator should verify the display and push BIT continue.	TIP = 888 All lights driven by channel C

*TIP—Test in progress display

TABLE D-1a.—Continued

TESTING OF PRIORITY INTERRUPTS (TESTS 002 THRUGH 007)			
Test number	Test step description	Comments	Expected failure indicators
002	Arithmetic fault by ADU overlfow	A large positive number is incremented to overflow in the upper register using the ADU instruction. A priority interrupt will occur and be verified.	None
003	Arithmetic fault by ADU sum = X'8000'	A negative number is decremented using the ADU instruction until the sum = X'8000'. A priority interrupt will occur and be verified.	None
004	Arithmetic fault add overflow	A double length positive number is incremented to overflow in the upper and lower registers using the add instruction. A priority interrupt will occur and be verified.	None
005	Arithmetic fault logic indicators LA and LB check	The program sets and resets logic indicators (LA and LB) and checks that the indications respond accordingly.	None
006	Arithmetic fault SBU underflow	A negative number is decremented in the upper register using the SBU instruction until underflow occurs. A priority interrup will occur and be verified.	None
007	Arithmetic fault ABS overflow	The absolute value of X'8000' is taken. A priority interrupt will be generated and verified.	None

TABLE D-1a.—Continued

TESTING OF PRIORITY INTERRUPTS (TESTS 008 THROUGH 013)			
Test number	Test step description	Comments	Expected failure indicators
008	Arithmetic fault SBD underflow	A double length negative number is decremented in the upper and lower registers using the SBD instruction until underflow occurs. A priority interrupt will occur and be verified.	None
009	Arithmetic fault CPL overflow	The complement of X'8000' is executed. A priority interrupt will be generated and verified.	None
010	Arithmetic fault RTF to SR	The status register will be modified using the RTF instruction in such a way that the AF indicator is set. A priority interrupt will be generated and verified.	None
011	Arithmetic fault multiply with 1 argument = X'8000'	One is multiplied by X'8000'. A priority interrupt will be generated and verified.	None
012	Arithmetic fault divide with result G.T.1	Four is divided by one. A priority interrupt will be generated and verified.	None
013	Memory fault	A memory parity fault indication is induced through the SSCU. A memory fault priority interrupt will occur and be verified. Note: No changes occur in the actual memory; therefore, the fault is easily cleared.	None

TABLE D-1a.—Continued

TESTING OF CROSS-CHANNEL DATA LINKS (TEST 014 THROUGH 016)			
Test number	Test step description	Comments	Expected failure indicators
014	Cross-channel data parity	The parity generators for the cross-channel data links are failed. Bad parity indications on cross-channel transmissions from each computer unit will occur and be verified.	X channel B C A C A B Sys. 1st (3) Sys. 2nd (3)
015	Clear induced failures and error reset	Failure annunciation will be cleared.	None
016	Cross-channel data transmission	Each processor transmits a word to the other two processors. The received words are compared to the transmitted words. Exact agreement is expected.	None

TABLE D-1a.—Concluded

TESTING OF MEMORY AND ITERATION RESET (TESTS 017 THROUGH 019)			
Test number	Test step description	Comments	Expected failure indicators
017	None	This test was deleted.	None
018	Parity check for all of memory	Every location in memory is read. After reading each word, the memory is checked to ensure that no memory fault has occured.	None
019	Computer iteration reset check Normal operation	A flag will be cleared which will be reset as part of the response to an iteration reset interrupt. After clearing the flag, the processor will execute a time delay loop and then check the flag to see if it has been reset. If it has not been reset, the test default flag is loaded. This test is in preparation for MLV group 1 testing.	None

TABLE D-1b.—MCP-703 BUILT-IN-TEST TEST SEQUENCE

CLOCK MLV.S AND MONITOR'S IN COMPUTER UNITS (GROUP 1: TESTS 020 THROUGH 034)				
Chan. role	Test numbers and indexing	Test step description	Comments	Expected failure indicators
Fail Test	020 025 030 A* B* C* M* M M	Fail first computer selected clock	Verifies that monitor will properly detect computer unit clock voter (CUCV) first fail and LOCAL (LCL) fail for failed channel	CUCV 1st LCL sys 1st (3)
Test	021 026 031 B C A C A B	Computer iteration reset check	Verifies that MLV has voted out first failure and clock signal is provided by remaining two channels.	No change
Fail Test	022 027 032 A B C B C A M M M	Failure second computer selected clock	The SLI memory grab must be inhibited before second failure to assure that the CP will not be locked out of memory. Verify second failure indication. Loss of SLI clock in the computer will cause loss of iteration reset in all units, and SLI information will cease to be updated.	CUCV 1st CUCV 2nd LCL (3) Sys 1st (3) Sys 2nd (3)
Test	023 028 033 C A B	Verify iteration reset will not occur	Verifies that single channel cannot cause iteration reset, thus checking that inputs to MLV are not failed high.	No change
	024 029 034	Clear failures and error reset	All induced failures will be cleared. Error reset will be set to a one then to a zero. Failure annunciation is checked to assure that a proper reset has occurred. The SLI memory grab should be enabled after finishing the group 1 tests.	None

*M = Monitors; A, B, and C = channels.

TABLE D-1c.—MCP-703 BUILT-IN-TEST TEST SEQUENCE

PREPARATION FOR GROUP 2 TESTS (TEST 035)			
Test number	Test step description	Comments	Expected failure indicators
035	CIU and STRU loop check	A digital number is transmitted to the CIU's and STRU where it is converted to an analog voltage and routed back to the computer unit through the respective A/D converters. The returned word is compared to the transmitted word within a specified tolerance. This preliminary loop check is in preparation for group 2 tests which will also use a loop check. The tolerance is set loose to allow a rapid check through the input filters to the A/D converters.	None

TABLE D-1d MCP-701 BUILT-IN-TEST TEST SEQUENCE

CLOCK MLV'S AND MONITORS IN THE STRU'S (GROUP 2: TESTS 036 THROUGH 050)				
Chan. role	Test numbers	Test step description	Comments	Expected failure indicators
Fail	036 041 046 A* B* C* M* M M	Fail first STRU selected clock	The clock signal from each CIU channel to the STRU is systematically failed to zero. Proper STRU clock voter (STRUCV) failure detection is monitored.	STRUCV 1st LCL Sys. 1st (3)
Test	037 042 047 B C A C A B	Data transmission test with first clock failure	A loop check (same as test 035) is conducted. If the loop check is not satisfied for any channel, then the associated clock voter has a zero failure.	No change
Fail Test	038 043 048 A B C B C A M M M	Fail second STRU selected clock	The clock signal from a second CIU channel to the STRU is failed to zero. This should stop the STRU clock signal and leave the STRU serial data lines to the SLI either high or low, thus, STRU response is unpredictable and can be all zeros or all ones. Experience is ones.	Sys. 1st (3) Sys. 2nd (3) LCL (3) STRUCV 1st STRUCV 2nd STRUIV 1st STRUIV 2nd STRUOV 1st STRUOV 2nd
Test	039 044 049 C A B	Verify data will not transmit with two clock failures	A loop check (same as test 035) is conducted. If the loop check is satisfied for any STRU channel, the associated clock MLV has an input failed to one.	No change
	040 045 050	Clear failures and error reset	This procedure is the same as test 024.	None

*M = Monitors; A, B, and C = channels.

TABLE D-1e.—MCP-701 BUILT-IN-TEST TEST SEQUENCE

VERIFICATION THAT SYSTEM IS ON PRIMARY OSC (TESTS 051 THROUGH 053)			
Test number	Test step description	Comments	Expected failure indicators
051	Fail secondary oscillator	The secondary oscillator signal will be failed to zero at the line drivers, which drive from CIU channel B. Proper failure monitoring will be checked.	OSC 1st Sys. 1st
052	Computer iteration reset check	This procedure is the same as test 019. If no iteration reset is detected, the system was run on the secondary oscillator.	No change
053	Clear induced failure and error reset	This procedure is the same as test 024.	None

TABLE D-1f.—MCP-701 BUILT-IN-TEST TEST SEQUENCE

TESTING OF REDUNDANT CLOCK SYSTEM (GROUP 3: TESTS 054 THROUGH 101)				
Chan. role	Test numbers and indexing	Test step description	Comment	Expected failure indicators
Fail for setup only	054 070 086 A* B* C*	Fail first clock switchover	The clock switchover signal is failed to a *do not want to switch* state. There should be no failure monitors tripped.	None
Fail test setup only	055 071 087 A B C	Fail primary oscillator	The system should switch to the secondary oscillator. Indications from CXVR1 are unpredictable because the switchover pulse width is marginal relative to the failure detection time.	OSC 1st Sys. 1st
Test	056 072 088 B C A C A B	Computer iteration reset check	This procedure is the same as test 019. If no iteration reset is detected, either a latent failure is in the clock switchover MLV, or the secondary oscillator is failed.	No change
	057 073 089	Clear failures and error reset	This procedure is the same as test 024.	
	058 074 090	Actuates computational reset	The computational reset routine will cause an error reset, which will force the system to switch back to the primary oscillator.	None

*A, B, and C channels.

TABLE D-1f.—Continued

Chan. role	Test numbers and indexing	Test step description	Comments	Expected failure indicators
Fail	059 075 091 A* B* C*	Fail first clock switchover	This procedure is the same as in test 054. Fails to a *do not want to switch* state.	None
Fail	060 076 092 A B C	Fail second clock switchover	This procedure is the same as test 054. Fails to a *do not want to switch* state.	None
Fail	061 077 093 M* M M	Fail primary oscillator	The SLI memory grab must be inhibited before failing the primary oscillator. Two lines are failed to a *do not want to switch* state so no switchover will occur. The channel without a clock switchover failure will indicate local failure. The clock switchover voter (CSWV) will indicate a failure.	OSC 1st LCL CSWV 1st
	062 078 094	Computer iteration reset check	This procedure is the same as test 019. If an iteration reset is detected, either the system was not operating on the primary oscillator or there is a latent failure in more than one clock switchover MLV.	No change
	063 079 095	Clear failure and error reset	This procedure is the same as test 024.	OSC 1st Sys. 1st
	064 080 096	Computational reset	This procedure is the same as test 058. Must enable memory grab and SLI before computational reset can be accomplished.	None

*M = Monitors A, B, and C = channels.

TABLE D-1f.—Concluded

Chan. role	Test numbers and indexing	Test step description	Comments	Expected failure indicators
Fail	065 081 097 A* B* C*	Fail first clock switchover	Procedure wiht same as test 054. Fails to a *do not want to switch* state.	None
Fail	066 082 098 A B C B C A	Fail second clock switchover	This procedure with same as test 054. Fails to a *do not want to switch* state.	None
Test	067 083 099 C A B M* M M	Fail secondary and primary oscillator, monitor for first clock switchover failure	Inhibit SLI memory grab. Fail secondary oscillator and then fail primary oscillator.Clock switchover MLV inputs will be 1,1,0.	CSWV 1st OSC 1st OSC 2nd LCL Sys.1st Sys.2nd
Fail Test	068 084 100 A B C C A B B C A	Force second clock switchover; monitor indication by resetting second switchover failure	Resetting second clock switchover failure will cause MLV inputs to be 1, 0, 0.	CSWV 1st CSWV 2nd OSC 1st OSC 2nd LCL Sys. 1st Sys. 2nd
	069 085 101	Clear induced failures and reset indicators and computational reset	Memory grab is enabled, and computational reset is activated.	None

*M = Monitors; A, B, and C = channels.

TABLE D-1g.—MCP-701 BUILT-IN-TEST TEST SEQUENCE

ITERATION RESET MLV'S AND MONITORS IN THE CU'S (GROUP 4: TESTS 102 THROUGH 116)				
Chan. role	Test numbers and indexing	Test step description	Comments	Expected failure indicators
Fail Test	102 107 112 A* B* C* M* M M	Fail first computer iteration reset	The iteration reset signal, which is generated in one computer interface unit and transmitted to all three CU's, is failed to zero at the line transmitters. Proper failure annunciation is monitored.	CUIV 1st LCL Sys. 1st
Test	103 108 113 B C A C A B	Verify iteration reset still occurs with first failure	This procedure is the same as test 019. If iteration reset is not detected in one of the computers, either the iteration reset MLV in that channel has a zero latent failure, or one iteration reset signal is not present.	No change
Fail Test	104 109 114 A B C B C A M M M	Fail second computer iteration reset	This fails the iteration reset to the CU's and also fails iteration reset to the CIU's and STRU.	CUIV 1st CUIV 2nd LCL (3) Sys. 1st (3) Sys. 2nd (3)
Test	105 110 115 C A B	Verify single channel cannot cause iteration reset	This procedure is the same as test 019. If an iteration reset is detected in one computer, the iteration reset MLV in that computer has a one latent input failure.	No change
	106 111 116	Clear failures and error reset	This procedure is the same as test 024.	None

*M = Monitors: A, B, and C = channels.

TABLE D-1h.—MCP-701 BUILT-IN-TEST TEST SEQUENCE

ITERATION RESET MLV'S AND MONITORS IN THE CUI'S AND STRU (GROUP 5: TESTS 117 THROUGH 131)				
Chan. role	Test numbers and indexing	Test step description	Comments	Expected failure indicators
Fail Test	117 122 127 A* B* C* M* M M	Fail first CIU and STRU iteration reset	The iteration reset signal, which is generated in a one computer unit, is failed to zero at the line transmitters to the CIU's and STRU. Proper failure annunciation will be monitored.	STRU IV 1st CIUIV 1st LCL SYS. 1st (3)
Test	118 123 128 B C A C A B	Verify iteration reset still occurs with first failure	The iteration reset serves only an initial synchronization function. This test unsynchronizes the CIU's and STRU from the CU's. It then checks to see that iteration reset MLV's will perform the synchronization with the above failure.	No change
Fail Test	119 124 129 A B C B C A M M M	Fail second CIU and STRU iteration reset	The iteration reset signal which is generated in a second-computer unit, is failed to zero at the line transmitters to the CIU's and STRU. Some failure annunciation will be monitored. Response from the STRU is not predictable.	STRUIV 1st STRUIV 2nd CIUIV 1st and 2nd LCL (3) Sys. 1st (3) Sys. 2nd (3)
Test	120 125 130 C A B	Verify single channel cannot cause iteration reset	This procedure is similar to test 118. If synchronization is achieved in any unit, there is a one failure in that iteration reset MLV.	Same as above STRUOV 1st STRUOV 2nd STRUCV 1st STRUCV 2nd
	121 126 131	Clear failures and error reset	This procedure is the same as test 024.	None

*M = Monitors; A, B, and C = channels.

TABLE D-1i.—MCP-701 BUILT-IN-TEST TEST SEQUENCE

OUTPUT DATA MLV'S AND MONITORS IN THE CIU'S AND STRU (GROUP 6: TESTS 132 THROUGH 140)						
Chan. role	Test numbers and indexing			Test step description	Comments	Expected failure indicators
Fail Test	132 A* B C	135 B* C A	138 C* A B	CIU and STRU loop check with first channel output = 0	The outputs from the one computer to all CIU's and STRU will be zero, and the outputs from the remaining two computers will be equal to and different from zero. Proper failure annunciation and proper return signal will be monitored for all units. The MLV's should vote out the failure and return the same data in all three channels.	STRUOV 1st CIUOV 1st Sys. 1st (3)
Fail Test	133 A B C	136 B C A	139 C A B	CIU and STRU Loop check with first and second channel outputs = 0	The outputs from two computers to the CIU's and STRU will be zero, and the output from the third computer will be different from zero. Proper failure annunciation and proper return signal (i.e., zero) will be monitored for all units.	STRUOV 1st STRUOV 2nd CIUOV 1st CIUOV 2nd LCL (3) Sys. 1st (3) Sys. 2nd (3)
	134	137	140	Restore all outputs to test value and error reset	Failures are cleared by restoring all outputs to the nominal test value.	None

*A, B, and C = channels.

TABLE D-1j.—MCP-701 BUILT-IN-TEST TEST SEQUENCE

A/D AND D/A CONVERSION TESTS (TESTS 141 THROUGH 145)			
Test number	Test step description	Comments	Expected failure indicators
141	CIU and STRU loop check A = B = C = X_1 X_1 = X'0700'	The outputs of computers A, B, and C to all CIU's and STRU will be equal to a known quantity X_1, which will be routed to the D/A converters and back through the A/D converters to each computer. The returned number will be compared to the transmitted number to a close tolerance.	None
142	CIU and STRU loop check A = B = C = X_2 X_2 = X'00FF'	Same as for test 141.	None
143	CIU and STRU loop check A = B = C = X_3 X_3 = X'FF00'	Same as for test 141.	None
144	CIU and STRU loop check A = B = C = X_4 X_4 = X'F8FF'	Same as for test 141.	None
145	Error reset	There should be no failures or indications to clear; the error reset routine will update the test number.	None

TABLE D-1k.—MCP-701 BUILT-IN-TEST TEST SEQUENCE

OSCILLATOR FREQUENCY TESTS (TEST 146)			
Test number	Test step description	Comments	Expected failure indicators
146	Oscillator frequency checks	The frequencies of the CP and SLI oscillators are monitored by checking the number of counts the CP can make between timer interrupts from the SLI. The comparison is repeated six times to enhance the check. The SLI memory grab must be inhibited so the CP will not be blocked from counting while the SLI is using the memory. SLI should be enabled at end of test.	None

REFERENCES

1. *Redundant Flight-Critical Control System Evaluation,* "Analog and Digital Systems Descriptions," FAA-SS-73-2-1, June 1974.

2. *Redundant Flight-Critical Control System Evaluation,* "Analog and Digital Performance Comparisons," FAA-SS-73-2-2, October 1974.

3. *SST Longitudinal Control System Design and Design Processes,* "Hardened Stability Augmentation Design," FAA-SS-73-1, June 1973.

4. J.G. McKinney, *An Introduction to the Computer Simulator,* D25-71060-1, Boeing Commercial Airplane Company, January 1970.