

2-12  
MIX

20 copies

NASA TECHNICAL  
MEMORANDUM

NASA TM X-58063  
Volume III

NASA TM X-58063  
Volume III

17

PROCEEDINGS OF THE SPACE SHUTTLE INTEGRATED  
ELECTRONICS CONFERENCE

NASA Manned Spacecraft Center  
Houston, Texas  
May 11-13, 1971

FACILITY FORM 602	N71-35052	N71-35068
	(ACCESSION NUMBER)	(THRU)
	383	G3
	(PAGES)	(CODE)
TMX 58063	31	
(NASA CR OR TMX OR AD NUMBER)	(CATEGORY)	

1 Report No TM X-58063 - Volume III	2 Government Accession No	3 Recipient's Catalog No		
4 Title and Subtitle  PROCEEDINGS OF THE SPACE SHUTTLE INTEGRATED ELECTRONICS CONFERENCE		5 Report Date	6 Performing Organization Code	
		8 Performing Organization Report No	10 Work Unit No	
7 Author(s)	9 Performing Organization Name and Address		11 Contract or Grant No	
12 Sponsoring Agency Name and Address  National Aeronautics and Space Administration Washington, D. C. 20546			13 Type of Report and Period Covered <b>Technical Memorandum</b>	
			14 Sponsoring Agency Code	
15 Supplementary Notes  Held at the NASA Manned Spacecraft Center, May 11-13, 1971				
16 Abstract  The symposium encompassed five specific categories within the general category of space shuttle integrated electronics. The five categories are published in three volumes as follows:  Volume I - Electronics Overview and Guidance, Navigation, and Control II - Instrumentation and Power Distribution and Communications III - Data Systems				
17 Key Words (Suggested by Author(s))		18 Distribution Statement  Unclassified - unlimited		
19 Security Classif (of this report)  Unclassified	20 Security Classif. (of this page)  Unclassified	21 No. of Pages  398	22 Price*	

\*For sale by the Clearinghouse for Federal Scientific and Technical Information  
Springfield, Virginia 22151

PRECEDING PAGE BLANK NOT FILMED

## FOREWORD

As a follow-up of the Space Transportation System Technology Symposium held at the NASA-Lewis Research Center, Cleveland, Ohio, July 15-17, 1970, a series of discipline-oriented conferences was planned, with the Office of Advanced Research and Technology/Office of Manned Space Flight (OART/OMSF) Space Shuttle Integrated Electronics Technology Conference being held at the NASA Manned Spacecraft Center, Houston, Texas, May 11-13, 1971. The Conference goal was to present a timely review of the status of Space Shuttle technology in the major areas of electronics and power systems for the benefit of the industry, Government, university, and foreign participants considered to be contributors to the program. In addition, the Conference offered an opportunity to identify the responsible individuals already engaged in the program. The Conference sessions were intended to confront each presenter with his technical peers as listeners, and this was substantially accomplished.

Because of the high interest in the material presented, it is being published essentially as it was presented, utilizing mainly the illustrations used by the presenters along with brief words of explanation. The document is unclassified, and each of the authors has determined that his paper can be published in this manner. This publication is aimed at revealing the substance and significance of the work in this manner now, rather than in a more refined form much later.

CONTENTS

	Page
<u>DATA SYSTEMS</u>	
MODULAR SPACEBORNE PROCESSORS	
W. R. Wadden and H. S. Zieper, RCA Advanced Technology Laboratories; J. M. Gould and R. Asquith, NASA-Marshall Space Flight Center . . . . .	3 ✓
FAULT-TOLERANT COMPUTER DESIGN FOR SPACE SHUTTLE	
Irving Goodman, General Electric . . . . .	27 ✓
CONFIGURATION ALTERNATIVES FOR DATA BUS SUBSYSTEM INTERFACE UNITS	
Edward Chevers and William Mallery, Manned Spacecraft Center . . . . .	47 ✓
DATA BUS CONCEPTS FOR THE SPACE SHUTTLE	
John R. Lane, General Electric Company . . . . .	71 ✓
DATA BUS TO SERVICED EQUIPMENT AND SENSOR INTERFACE	
Berry S. Yolken, TRW Systems . . . . .	87 ✓
SELF-SEQUENCING DATA BUS TECHNIQUES FOR SPACE SHUTTLE	
Robert H. Hardin, Martin Marietta Corporation . . . . .	111 ✓
SOFTWARE DEVELOPMENT AND VERIFICATION TECHNOLOGY	
H. Trauboth and B. Hodges, NASA-Marshall Space Flight Center . . . . .	141 ✓
SYNCHRONOUS EXECUTIVE AND BUS CONTROL SOFTWARE FOR REDUNDANT DATA MANAGEMENT SYSTEMS	
T. E. Daly, TRW Systems . . . . .	179 ✓
HAL, A COMPILER LANGUAGE FOR SHUTTLE	
Fred H. Martin, Intermetrics, Incorporated . . . . .	191 ✓
DESIGN OF CRT DISPLAYS FOR THE SPACE SHUTTLE	
G. F. Conron, Norden Division of United Aircraft Corporation . . . . .	221 ✓
ONBOARD DATA STORAGE FOR SPACECRAFT	
J. E. Holthaus and L. B. Gangawere, Westinghouse Defense and Space Center . . . . .	251 ✓
MULTI-PURPOSE DISPLAYS FOR SPACE SHUTTLE	
A. P. Yeager and Z. G. Tygielski, IBM/FSD Electronics Systems Center; and F. L. Holmes, IBM/FSD Electronics Systems Center . . . . .	289 ✓
SELF-TEST TECHNIQUES FOR REDUNDANT DATA MANAGEMENT SYSTEMS	
Myron Kayton and H. S. E. Tsou, TRW Systems Group . . . . .	309 ✓

	Page
UNIFIED TEST EQUIPMENT A CONCEPT FOR THE SHUTTLE GROUND TEST SYSTEM	
Edgar A. Dalke, Manned Spacecraft Center . . . . .	329 ✓
A STANDARD LANGUAGE FOR TEST AND GROUND OPERATIONS	
Henry C. Paul, NASA-John F. Kennedy Space Center . . . . .	355 ✓
SOLID STATE SEQUENCER SYSTEM	
David E. Leck, Martin Marietta Corporation . . . . .	359 ✓
<u>ATTENDEES OF THE SPACE SHUTTLE INTEGRATED ELECTRONICS</u> <u>TECHNOLOGY CONFERENCE</u> . . . . .	379

## DATA SYSTEMS

PRECEDING PAGE BLANK NOT FILMED

N71-35053

MODULAR SPACEBORNE PROCESSORS

W. R. Wadden, H. S. Zieper

RCA Advanced Technology Laboratories  
Camden, New Jersey

J. M. Gould, and R. Asquith

NASA-Marshall Space Flight Center  
Huntsville, Alabama

## PART I

The challenge which confronts computer system designers in developing the data processing configurations for support of the next generation of space missions is to design a mechanized capability which provides a spectrum of graded application throughput while delivering a significantly higher level of system availability than ever realized in the past. This capability must be available without serious penalty to the spacecraft environment (in terms of heat and power loads) and must not usurp undue volume and weight allowances. Such system realizations are now both feasible and economical with the use of the direct outgrowths of prior NASA-sponsored technological development in large-scale instrumentation (LSI) technology, supportive design automation aids, and system-level explorations. This paper presents a review of the LSI COS/MOS technology base upon which such a highly reliable baseline computer system is being developed. Extensions of the design technique are directly applicable to wide application areas where customized digital logic subsystems are required. Ongoing developments not only assure growth in the computer capability, but also will lead to the reduction to practice of the newer forms of MOS technology and packaging, thus assuring a state-of-the-art technology time line in the future.

### THE TECHNOLOGY BASE

4

The MOS technology offers a natural response to the needs of the digital spacecraft hardware designer. Its many forms offer:

1. High packing density
2. Low standby power
3. Ability to modulate active power according to application speed needs
4. Ready means of device modeling and characterization
5. Appropriate levels of circuit speed

A design automation system provides the mechanism by which the designer can readily translate system needs into economical hardware. Input to the system is by partitioned logic diagrams which the software automatically translates to a customized functional circuit layout; provides a series of user-oriented, confidence-building intermediate reports; accepts user-desired modifications; and produces the magnetic tape output required to drive an automatic precision artwork generator. After plotting, the artwork is ready for the mask fabrication,



wafer processing, part packaging, and testing cycle. The software system permits both P- and C-MOS LSI arrays to be implemented with totally arbitrary functional logic keyed to the specific user needs. The only limits are chip size and package pin count, both of which are constantly expanding with the rapid maturing of the technology. The design aid system places at the user's disposal the high efficiency of truly custom-made circuit layout coupled with the fast-computer-paced turn-around cycle for such costly and error-prone actions as placement, routing, and artwork command creation. It is thus practical to implement system improvements in logic and to quickly take advantage of new technology availability such as the SOS and silicon gate variants. A particular advantage is the ability to implement reliable, economical switching mechanisms that permit subsystem redundancy for reliability enhancement of system operation. Logic is presented to the design and system in a form consistent with the usual manner of logic detail, that is, partitioned to a chip level by user-convenient functions and sketched with conventional symbols.

The first step in the implementation cycle is to translate the user logic into a standard cell notation and prepare the input data decks (a matter of 1 or 2 hours of coordination effort followed by a short keypunch session). Translation is accomplished by a simple reference to a circuit catalogue. Each page contains the logic data and provides circuit configuration, timing, and the cell size which is used in estimating chip size. This catalogue contains a wide variety of standard logic circuits (cells) in both P- and C-MOS forms. As will be discussed shortly, the catalogue, which represents a technological library, is expandable by the user so that the new needs which will arise in the future can readily be included in the library set.

Once the logic has been prepared for input, the software generates an optimum placement of circuits for array fabrication; routes the interconnecting wires, using metal and P tunnels; and configures the emerging chip in an approximately square shape. A series of reports is generated which permits the user engineer to evaluate his degree of satisfaction with the software-generated chip configurations. For those cases where pads should be relocated or special critical leads must be run in particular ways, a manual intervention capability is available. In some cases, human intervention can effect significant chip size reduction by specific wiring relocation. Again, the manual intervention feature permits the user to exercise his options. This intervention procedure permits the ideal blend of large-scale automatic action while taking advantage of the unique pattern recognition capability of the human to meet those special cases where a computer algorithm is not maximally effective.

After acceptance by the user, the proposed chip configuration is translated to a tape containing the artwork commands, and the layout is plotted. Interface exists to both GERBER and D. W. MANN precision plotting systems. By means of simple control parameters, plots can be made at any convenient scale. Typically GERBER plots are made at 80x; D. W. MANN plots are at 10x. With the current state of the art, chips with the feature shown in the following table are practical.

<u>Parameter</u>	<u>C-MOS</u>
Number of pins	16, 24, 40
Active area	150 × 150 mils
Total chip area	170 × 170 mils
Number of devices	400 to 800
2-input NOR equivalent	80 to 200

A key notion referred to in the preceding text is that of the standard logic circuit (cell). This is a circuit designed such that its topology conforms to a set of predetermined rules that have been chosen to blend technology parameters and software needs together in an optimum fashion. Since all circuits are of a constant height, a simple computation of total cell length permits the user to estimate chip size. The variation is a function of the wiring complexity. It is within the cell area that the digital-to-analog system takes advantage of the area efficiency of custom layout. Each cell is designed to compress the specific function into the minimum area required to meet pad and fanout drive constraints.

A key point must be presented with respect to part qualification. The circuits are:

1. Designed once and stored on a tape library
2. Subjected to extensive characterization during design
3. Based on the basic process design geometry rules

Once the wafer process and packaging scheme is space qualified, any manner of logic array can be qualified since the circuit designs are fixed, tested, and replicated at each and every use with exact faithfulness. There is simply no variation of circuit geometry from spot A on a chip to spot B on a chip. Variations in circuit actions are based solely on process tolerance, not artwork.

Clearly, the design of LSI arrays requires the highest confidence in design philosophy and detail. Two key supports to the design automation system discussed previously are a circuit level and a gate (logic) level simulation. The circuit simulation, FETSIM, is the key to the cell design method and is used to develop the majority

of circuit behavior data supported by appropriate laboratory evaluation. Inputs to the FETSIM program are: process parameters, model characteristics, and exercising waveforms. The output data provide use, full times, et cetera.

Correlation between the prediction of simulation and laboratory evaluation was obtained by implementing a test vehicle. This laboratory in miniature not only confirmed the simulation predictions, but also served as a process evaluation and comparison vehicle. It is to be noted that the ability to model and simulate the MOS circuit accurately is the key factor that permits an unlimited variety of circuits to be included in the cell library - the limit being user need.

Because the cell library is really a carefully maintained file of geometric data, it is a direct matter for the user to add a new cell. Such an addition is made to implement specific functions that will provide a specific area saving, speed improvement, or simply a function replicated many times as local logic islands. Starting with a circuit sketch, a geometric layout is made on a grid medium. The coordinates are keypunched, checkplots made, the design reviewed, and a library addition made. FETSIM is liberally used to make optimum selection of device sizes needed to match system specifications.

Specific to the logic level of design, validation of function is accomplished by means of the LOGSIM simulator. This simulation permits the designer to create in software a laboratory version of the logic, to exercise the laboratory version in any form that is desirable, and to observe the action at any series of points of interest. The program analyzes any static or dynamic configuration, permits specific timing to be assigned to elements based on system use, and provides user-oriented reports with appropriate levels of data reduction. Input to the program is exhaustively checked for clerical accuracy in order to help build confidence in design validity.

Programs at the algorithm and subsystem levels provide the final layer of design support needed to implement digital systems.

## GENERAL CONSIDERATIONS

Spaceborne processors will be employed to implement a wide range of tasks such as:

1. Display support applications
2. On-board control of mission experiments
3. Monitoring of system condition
4. Local display interfacing
5. Navigation
6. Dedicated experiment control
7. Communication control
8. Data-link format conversion
9. Overall format conversion
10. Overall analysis and system control
11. Environment control

Such processors are expected to be components in the full spectrum of spacecraft from small, unmanned (e.g., TIROS) configurations to the large systems typified by an orbiting space station. Depending on the nature of the spacecraft, a single processor or a federated complex of such systems acting in consort with a large central multiprocessor may be required.

Technology, power budgets, and system organization interact to define the range of capability attainable for single-processor systems utilizing the current state of the COS/MOS technology. The processing speed lower limit of 10,000 operations per second indicates either that system clock speed has been reduced drastically or that serial machine organization is used. At a processing speed of 100,000 instructions per second, machine organization is parallel with clock rates in the 1- to 2-MHz region.

Analyses conducted to date show that for the throughput toward the low end of the spectrum, 500 to 1000 words of memory permit single dedicated tasks to be implemented. At the higher processing speeds, more tasks can be implemented, but additional storage is required. Current state-of-the-art technology will permit the designer to reach processing speeds of 200,000 operations per second if the power budget is appropriately modified and the machine organization suitably enhanced. For those applications where the small real-time system structure is still appropriate, a higher level of total system processing capability can be obtained by linking a number of microprocessors together in a network structure. In such a system network, the central control processor (which may indeed be a large-scale multicomputer) directs the total system and contains the large software base associated with large-scale operating systems oriented toward large file control. The multi-computer will contain massive operating systems, validation systems, and total resource allocation programs which implement the required fail-safe modes of operation such as subsystem task reassignment. Each of the local task processors then is assigned a small set of dedicated tasks to implement. The communication bus is used to pass status data and interim system reports upward in the chain of data management software. Such a system structure permits local task reassignment in the event of a single task subsystem failure. Because the federation of processors is in reality a "loosely coupled" structure, the net aggregate of capability of N task processors can be higher than that achieved in a multiprocessor connection of an equivalent processor capability. Such a federated network is not intended to eliminate the multiprocessor, rather it is intended to provide a complementary ability to do dedicated, relatively unsophisticated local tasks.

9 Another way to broaden the range of system capability that can be implemented in a single microprocessor is to extend the functional capability coded within a single instruction. A specific illustration of this technique is to implement a branch control feature within each instruction. Such capability, for example, permits an ADD instruction to be implemented with a built-in test for the nature of the result, and a branch made under specified conditions. This idea can be extended in sophistication to allow index control as well as to provide for combinations of such tests. The improvement in speed capability comes by virtue of the lack of need for staticizing time for the branch instructions, with the additional possibility of simultaneous branch algorithm execution, thus reducing the time for total function execution. System structures like this have been demonstrated to be effective in a family of real-time computers.

To meet the evident need for a family of computer systems tailored to specific levels of system performance a generalized system structure (the SUMC organization) has been evolved which:

1. Is modular at the sub-byte level
2. Facilitates realization of a variety of specific architectures
3. Permits availability of a wide gradation of functional capability
4. Takes advantage of the capabilities of MOS LSI technology

## COMPUTER PARTICULARS

The connections of versatility and reliability are modular organization and a firmware implementation technique for control structures. With these key concepts as a base, a variety of system architectures can be implemented by using a common base-level machine organization, and the spectrum of system reliability techniques can be overlaid to yield the total system availability factors needed for advanced space missions.

The heart of the machine organization is a modular, generalized data path configuration. Typical of the generalized features provided by this structure are:

1. Dual, cascaded adders for high-speed address calculation
2. Transfer paths designed to implement multiple-bit multiplication, division, and specialized arithmetic operations
3. Partitioning at the sub-byte level so that user-desired word lengths may be efficiently handled
4. Requirement of only three chip types for implementation
5. Logic structures that match advances in memory technology
6. Efficient means of implementing central processor unit (CPU) mechanized input-output with a variety of interface widths for those systems where separate input-output programs are not warranted

Supporting this data path are a scratch pad memory unit, a two-level microprogram control unit, and a large main memory subsystem. The interface to the user environment is a real-time interrupt system and a set of input-output capabilities featuring both channel and programmed type.

Central to the concept of a real-time system structure is the priority interrupt feature. The capability of the LSI processor provides for servicing of a multiplicity of programs on a priority basis. The primary purposes of the priority-interrupt system are to eliminate the need for program-controlled scheduling and status testing on input-output transfers, and to segregate programs on an execution-priority basis.

The complete set of indicator flags is examined, simultaneously with the execution of each instruction, to determine which program demanding attention had the highest priority. When the currently active program has the highest priority of those demanding attention, the program proceeds without interruption. Otherwise, the system automatically transfers, at the completion of the current instruction, to the program with the highest priority.

During the priority-interrupt cycles, the machine logic ensures that the address of the next instruction of the interrupt program is saved in its instruction-location counter. Because the register complement (including the status, program counter, and input-output channel registers) is located in a scratch pad memory and is replicated for each interrupt level, there are no programming constraints on the application software. When all the higher priority demands are satisfied, control returns to the interrupted program, and the latter proceeds oblivious of the interruption. In this way, the priority-interrupt feature enables the data processor to interleave a number of different programs.

Programmed input-output routines for the LSI processor are generally short and often consist of a single instruction. Once the data have been transferred to memory, suitable programs are called to process the new information. The various input-output and data-handling routines are assigned different priority levels; therefore, the programmer need not perform periodic tests for the arrival of data. The priority-interrupt system automatically activates the proper program to process the new data which have been transferred into the memory by other programs.

Because of these considerations, assignment of relative priority among the various input-output programs and devices is principally based on the allowable waiting time between data transfers. Computing programs are assigned lower priorities than those of input-output programs, and the priorities may be assigned arbitrarily, within this constraint, by the user. The lowest priority program, automatically executed when the flag register is cleared, is usually a self-check routine which automatically exercises and tests the computer during idle time.

This type of input-output operation provides a simple, low-throughput transfer mechanism particularly suited for background level application modules (in particular, communication between processors connected in a multicomputer network where the interrupt mechanism provides a natural asynchronous control technique). Provision for large-scale data transfers is made by means of a selector channel configuration.

The simultaneous input-output operations are performed with the CPU processing on a cycle-steal basis to the main memory. Channel command words are set up by the program prior to executing an I/O instruction. For each data transfer between the main memory and the peripheral device, the control words stored in scratch pad memory are accessed and updated. When the total amount of data requested by the program have been transferred, the peripheral device will generate an interrupt indicating that the desired transfer has been completed.

The complete input-output transfer is performed under read-only memory (ROM) control by using two channel control registers. Channel control register I (CCR I) contains the input-output command and the main memory address of the next data to be transferred. Channel control register II (CCR II) contains the physical address of the peripheral device (obtained from the I/O instruction) and the remaining count of data yet to be transferred.

Once the two control registers are set up in scratch pad memory, the input-output operation is initiated with a START DEVICE instruction. After initiation, the peripheral device will generate a service request when it is ready to transfer data. The CPU checks for the presence of a service request upon the completion of an instruction execution. If one is present, the microprogram control will jump to an input-output service routine prior to executing the next instruction. At any time, the input-output operation can be halted by the execution of a HALT DEVICE instruction.

As presented in the succeeding discussion on expandability, this basic channel structure can be replicated to provide massive data throughput capability. The basic architecture of the demonstration baseline is compatible with current third-generation structures. The baseline is a 16-bit system using a balanced subset of this family of architectures.

The demonstration vehicle is implemented to provide a variety of methods in which the capability of the CPU may be increased. Such expansion may be accomplished by several hardware additions, use of firmware, and the normal use of software.

The baseline capability may be directly enhanced by hardware additions which:

1. Increase the main memory to a maximum of 65,000 words possible ( $2^{28}$  words possible with a 32-bit interface)
2. Expand the data path width
3. Provide a faster ROM and less scratch pad cycle time
4. Increase the number and width of the scratch pad and ROM words
5. Increase the number of user interrupt levels
6. Expand the width of the input-output interface and add to the number of trunks
7. Supply floating-point and field instructions

For specialized architecture variants, the data path configuration can be expanded in increments, although the most direct change to maintain compatibility with third-generation architecture would be a doubling of a data path. Use of the baseline logic and system structure is unchanged; only the number of chips is increased. Current design of the timing structure of the baseline permits changes in scratch pad and ROM timing to be made, each of which will provide a system speedup. This independence permits advances in LSI memory technology to be taken advantage of in the most cost effective manner.



By increasing the size of the local scratch pad memory, the programming power of the system is increased, and the number of interrupt levels available to the user is increased. The combination of these provides real advantages in complex real-time programs. Expansion of the input-output interface, culminating in the inclusion of an independent input-output processor module, permits the DV vehicle to be expanded to meet the total needs of an advanced space station complex.

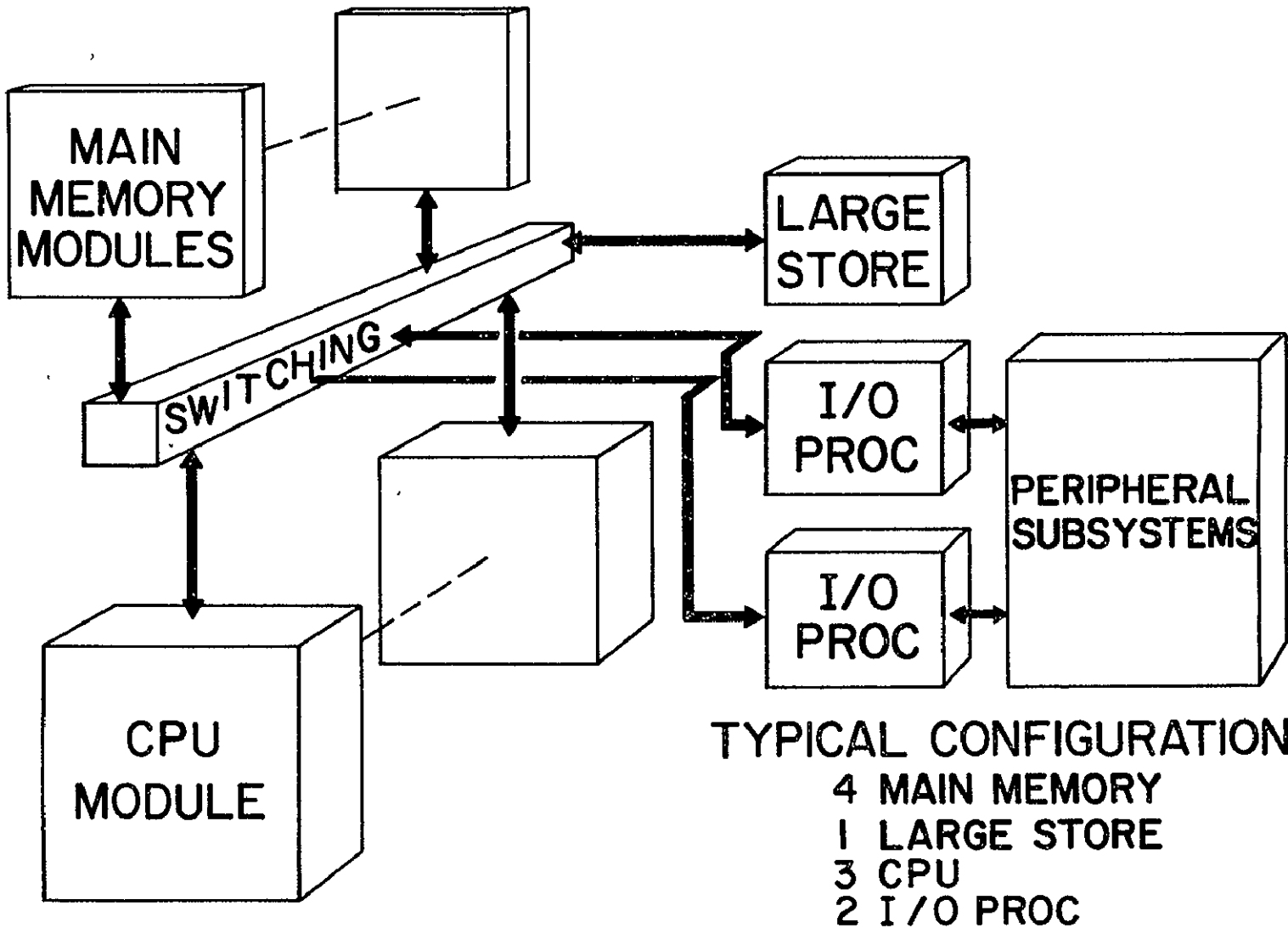
Several levels of firmware modification provide yet another dimension of baseline growth. These are:

1. Extended precision algorithms using the 16-bit data path
2. Extended repertoire
3. Specialized functions such as square root,  $e^x$ , sin, cos, et cetera
4. Diagnostic and recovery features

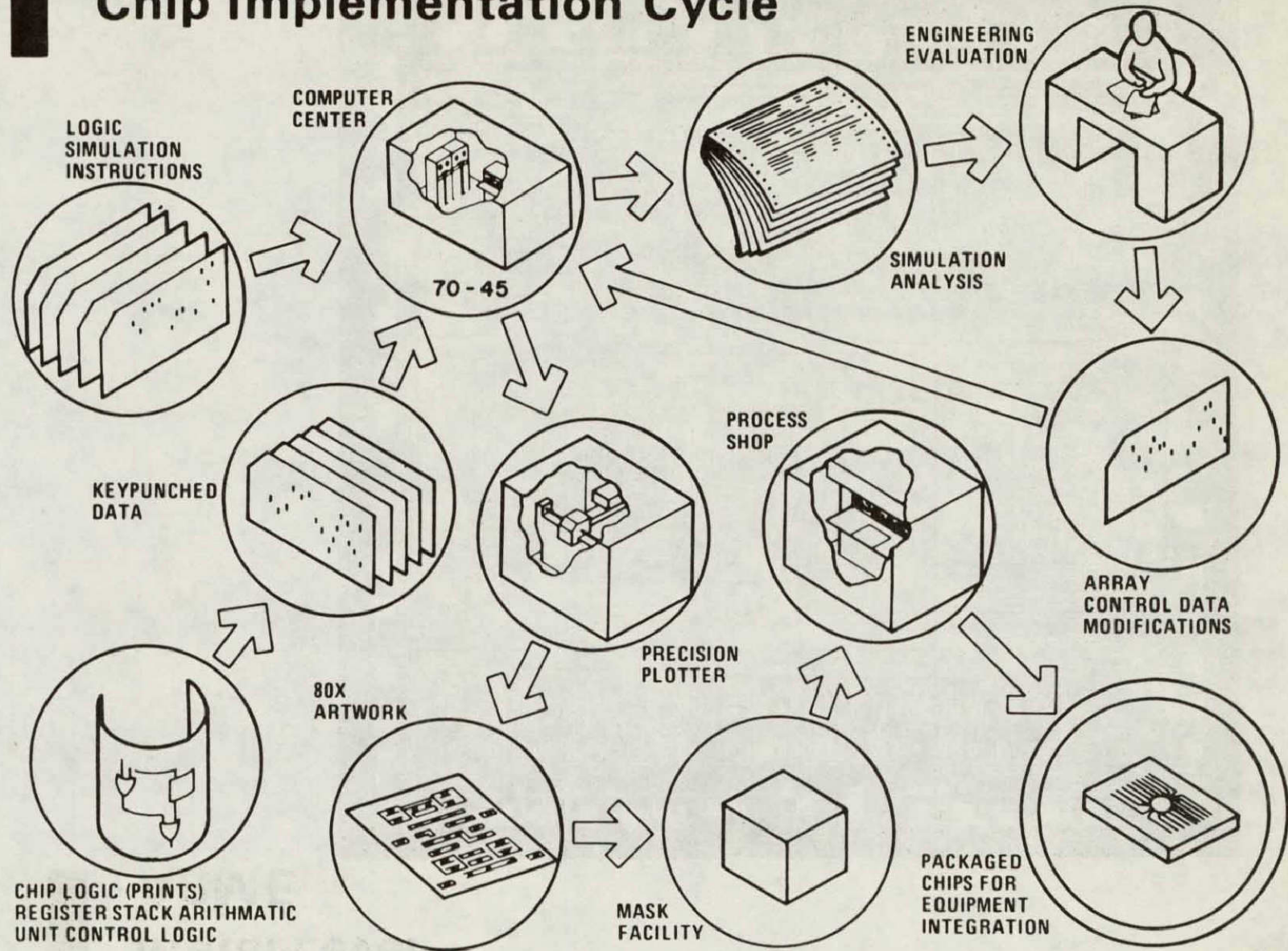
For any size data path, such additions are made easily for extended precision by adding ROM words. Inclusion of floating-point and field-organized instructions will require one or two new control chips in addition to more ROM.

Extensions of the system to include diagnostic and recovery features will require chip modification and new chips and will represent the more advanced capabilities which can be added.

# System Structure

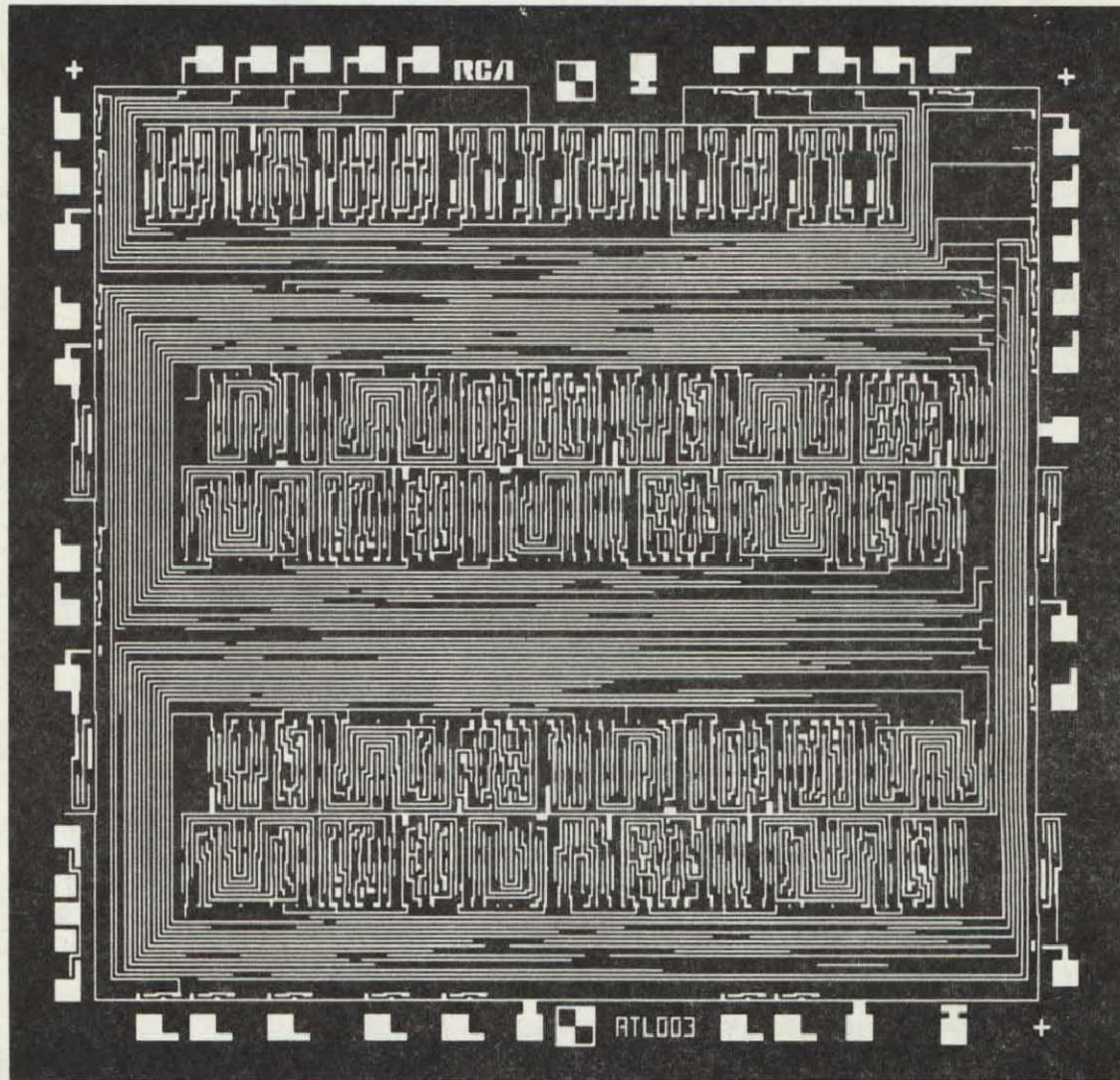


# Chip Implementation Cycle



**Metal Level**

**APME**



# SUMC

## Parameters

TYPE OF CPU	DIGITAL, STORED PROGRAM, SYNCHRONOUS BINARY, PARALLEL STRUCTURE	TYPICAL SPEEDS	RX ADD 3 $\mu$ SEC RX MPY 8 $\mu$ SEC FULLY INDEXED BC 3 $\mu$ SEC
INSTRUCTION FORMS	FULL SPECTRA 70, RCA SERIES 2-7 AND SYSTEM 360/370 NONPRIVILEGED COMPATIBLE	MPX CHANNEL RATE	2.6 x 10 <sup>6</sup> BYTES/SEC
DATA FORMATS	16/32/64/128 BIT ORGANIZATION	SELECTOR CHANNEL RATE	5 x 10 <sup>6</sup> BYTES/SEC
ARITHMETIC FORMS	FRACTIONAL BINARY, DECIMAL, BYTE (ASCII) AND FLOATING POINT	MAIN MEMORY ORG	4096 WORDS/MODULE (2 <sup>28</sup> WORDS MAX)
I/O TYPE	CHANNEL COMMAND ORIENTED	CONTROL TYPE	ROM-PLANNED FOR EMULATION
INTERRUPT TYPE	USER CONTROLLABLE, 16 INDEPENDENT LEVELS PER CPU	REGISTER STOCK	16 GENERAL REGISTERS/LEVEL
		SIZE	300 IN <sup>3</sup>
		WEIGHT	10 LB
		POWER	15 WATTS

} CPU

# Major System Features

- 16/32 bit word formats
- Spectra 70 non privileged opcodes
- 32 bit main memory
- ROM control scheme
- Data path modularity from 16 bits in 4 bit increments
- Add on floating point option
- Add on byte handling option
- Add on decimal option
- Provision to add specialty instructions ( $e^X$ , SINX, etc.)
- Multicomputer option
- Natural multiple precision modes
- Independent I/O processor capability
- Growth to total R215 software systems

## PART II

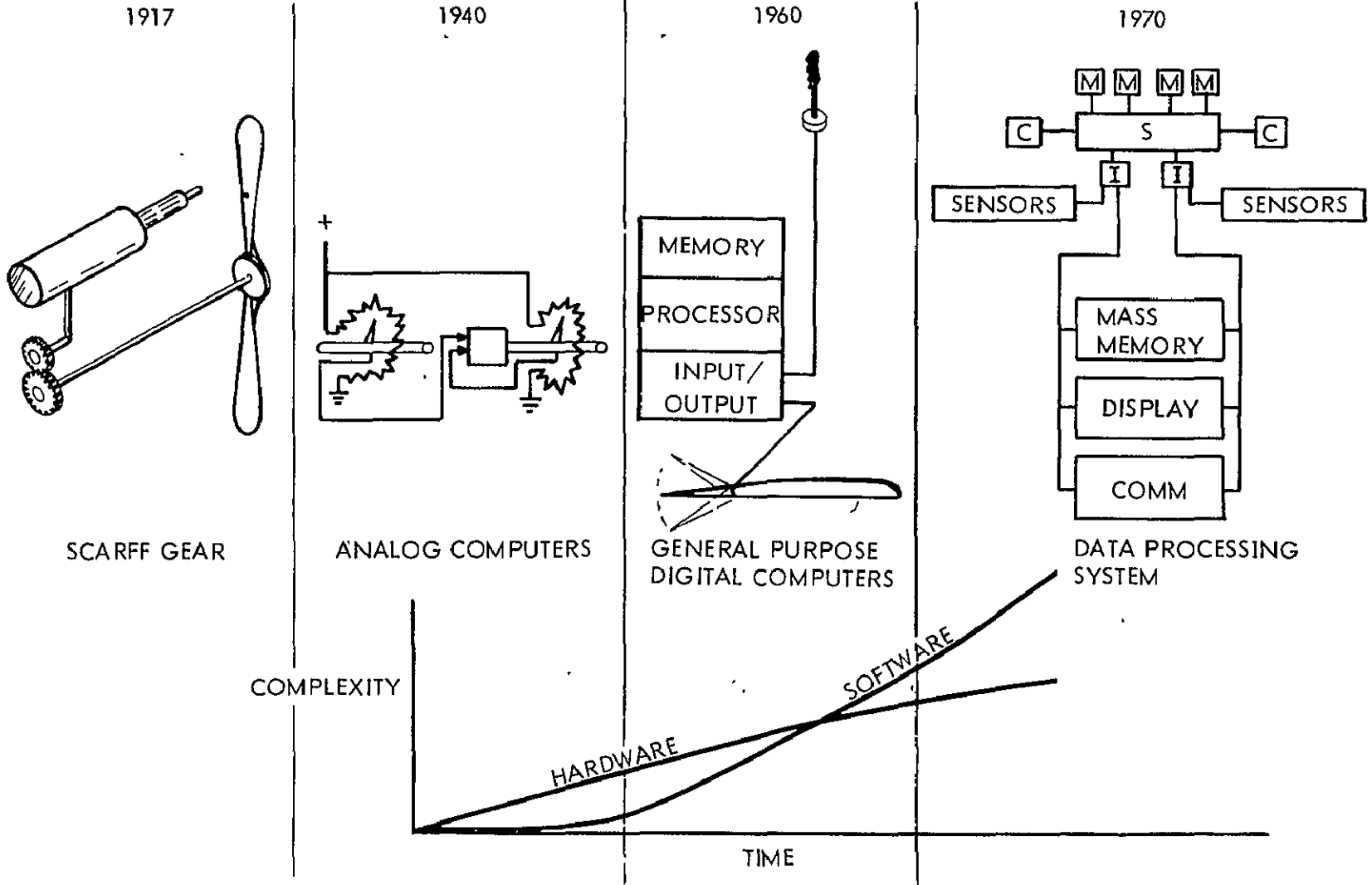
### AIRBORNE COMPUTER TECHNOLOGY

Advances in integrated circuit technology in the 60s has made possible the fabrication of general purpose, stored program, digital computers of small size and weight suitable for airborne applications. With these computers, it has been possible to solve problems for a variety of applications using standard hardware configurations with custom designed software. This departure from the previous practice of custom designing hardware for each new problem has shifted the emphasis in the complex and time consuming aspects of system design from the hardware design to the software design.

Further advances in component technology in the areas of medium and large scale arrays make possible far more sophisticated computer architectures for airborne and spaceborne programs in the 70s thereby permitting significantly more difficult problems to be addressed. The attendant increased complexity of software design and validation is cause for concern in project planning particularly in the case of the Space Shuttle where performance goals are in excess of anything previously attempted in airborne or space environments. Clearly the problem of software development must be addressed at the early stages of baseline specification.

# Airborne Computer Technology

20





## APPROACH TO SOFTWARE DEVELOPMENT

The requirement for autonomous operation of the Space Shuttle imposes on the computer software the management functions of automatic checkout, mission planning and mode control, error management and configuration control of redundant subsystems, and communications management in addition to guidance navigation and flight control. Clearly a large scale software effort is indicated.

To execute the software development in a timely manner at low cost and risk requires the application of software skills and disciplines that are state-of-the-art in industry. Also, previous experience on large software projects must be utilized to avoid repeating old mistakes. Specifically, the single factor that contributes most to software failures is attempting to do too much too soon with insufficient resources. Other factors that create problems are the tendency to underestimate the problem during the early stages and the failure to provide for enough expansion in memory and processing capability to accommodate the total problem as it becomes defined.

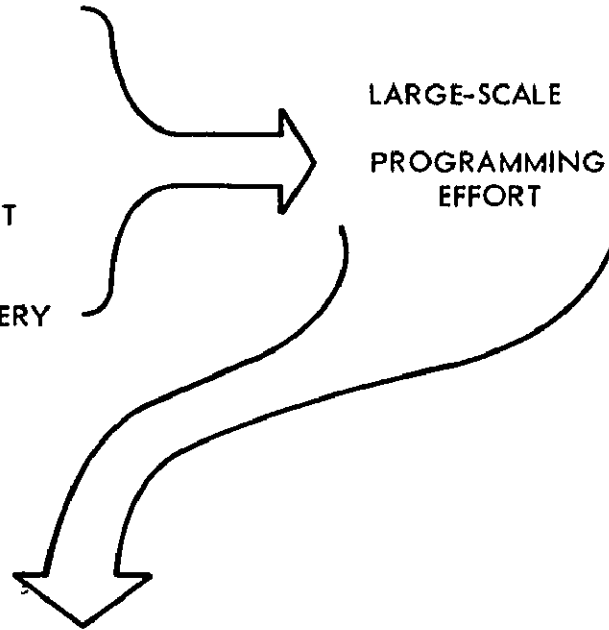
Flexibility, then, should be the single most important feature of the Space Shuttle computer/software baseline. Consistent with this approach is the selection of a general purpose operating system as the baseline software structure for the shuttle management functions. The operating system functions include the executive with multiprogramming capability, data management, display management, file/query management, error management, and system support modules. For the initial capability, the baseline has a separate minor executive for guidance, navigation, and control that operates in parallel with but under control of the prime executive. This separation of tasks by major categories of management and real time control functions simplifies the design of the system software and the design and verification of the application software.

# Approach to Software Development

## SCOPE OF TASKS

GUIDANCE AND NAVIGATION  
FLIGHT CONTROL  
DISPLAY CONTROL  
DATA PREPROCESSING  
CONFIGURATION MANAGEMENT  
EXPENDABLES MANAGEMENT  
MISSION PLANNING  
ERROR DETECTION AND RECOVERY

LARGE-SCALE  
PROGRAMMING  
EFFORT



## REQUIRES MODERN OPERATING SYSTEM

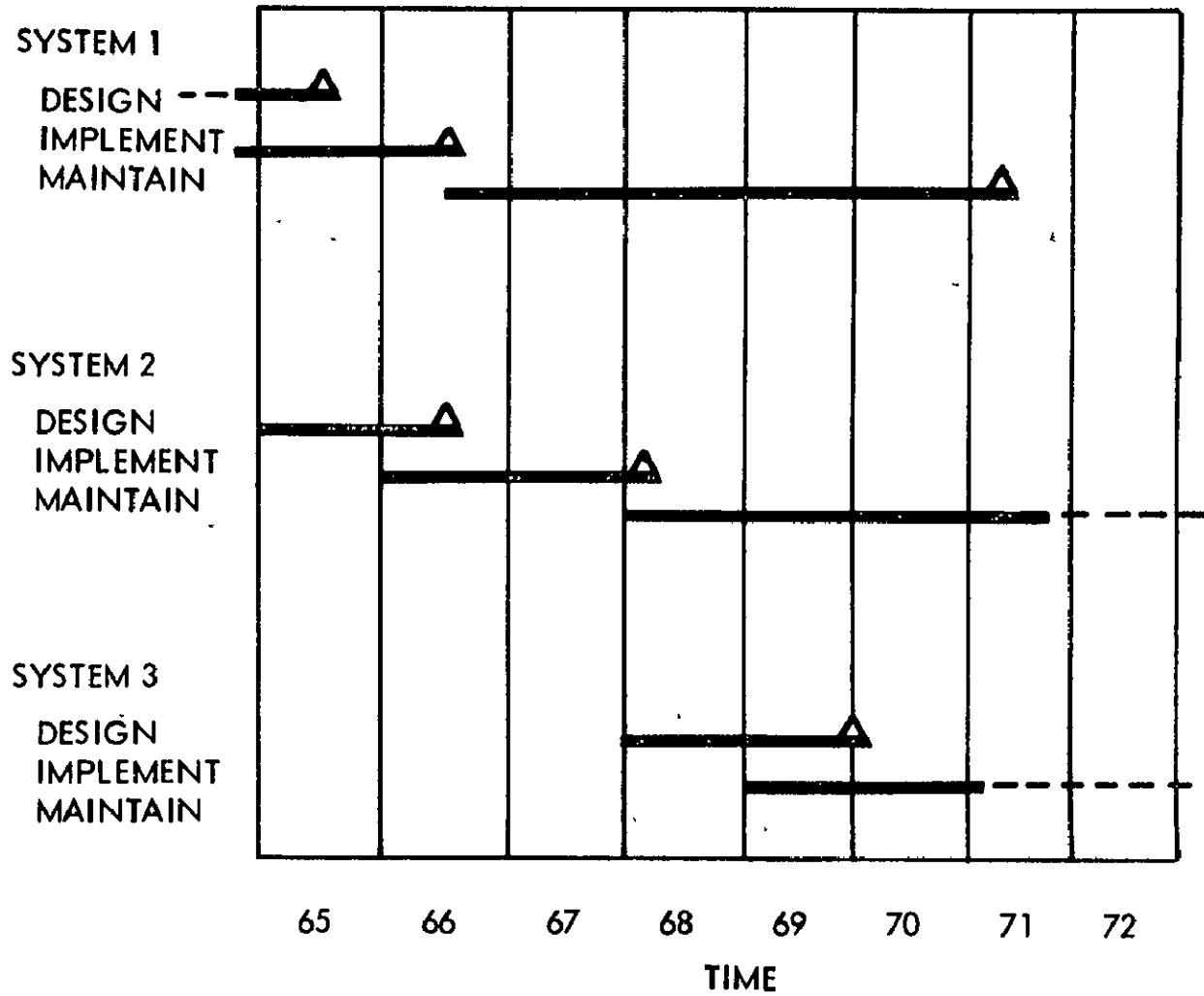
MULTIPLE USERS UNDER SINGLE EXECUTIVE  
EFFICIENT MANAGEMENT OF RESOURCES  
WELL DEFINED USER INTERFACES  
EXTENSIVE TEST AND SIMULATION

### OPERATING SYSTEM SCHEDULES

To illustrate the problems associated with the development of a complex operating system, the development schedules for three commercial operating systems developed for the same computer system are shown. Typically, a period of 1 to 2 years is spent in initial specification and design, an additional 1 to 2 years in implementation and test, and then the system is released. At regular intervals, the system is upgraded to incorporate corrections of program errors in the previous release and additions of new features or improvements. The complexity and performance of the operating system is increased in a significant way each time a new system is introduced but the total process is one of gradual evolution to increased capability.

These schedules point out the urgency in getting software development started early and suggest the value of gradual evolution to total capability with usable operating systems of increasing complexity available as required during development and operational phases of the Space Shuttle.

# Operating System Schedules



## DUAL JOB STREAM BASELINE

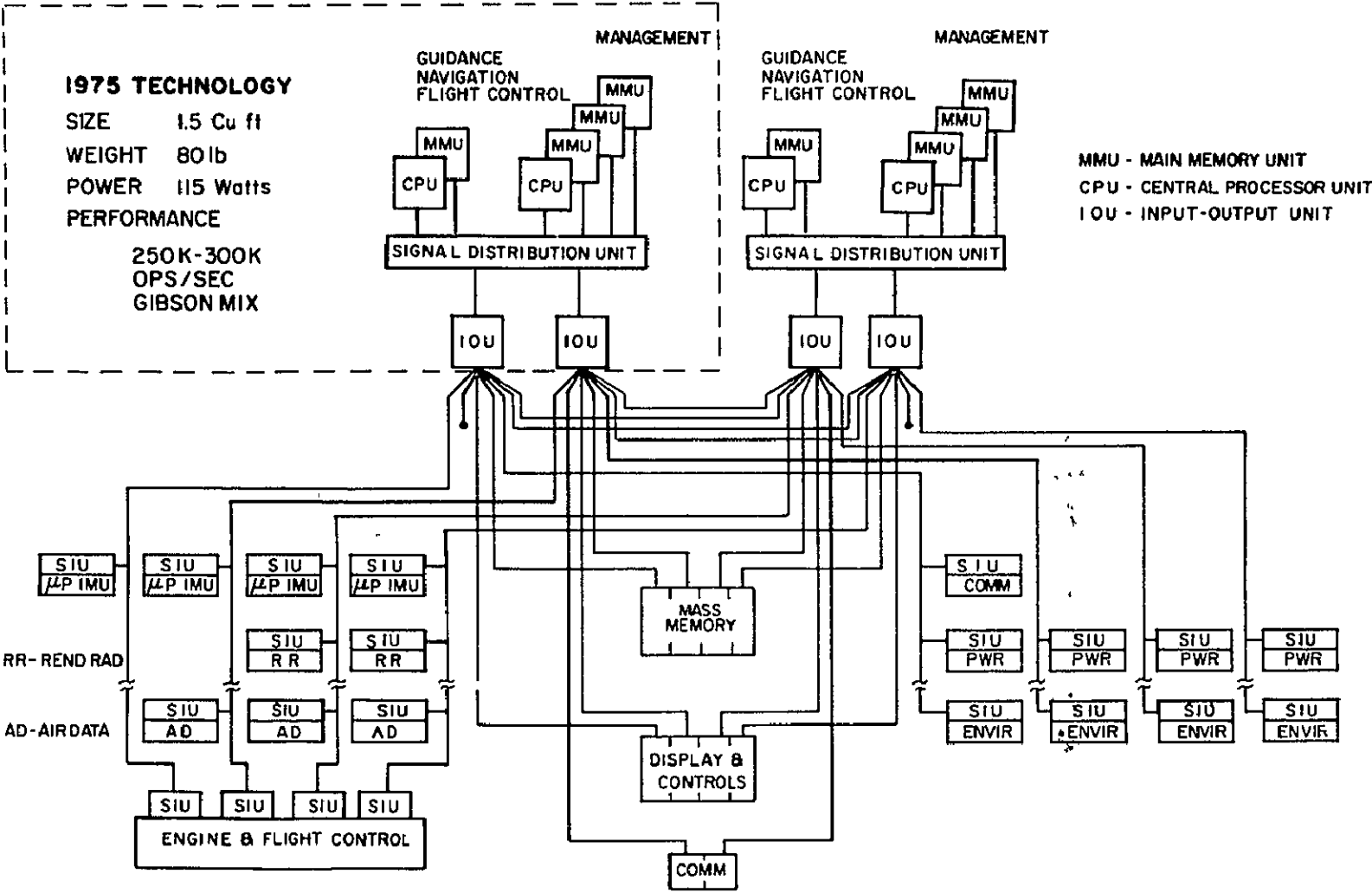
An approach to the implementation of the data management functions for the Space Shuttle is shown which features a centralized computer complex of modular design. The total processing load is pre-partitioned into dual job streams of nearly equal peak load with each job stream assigned a central processor unit. One processor is assigned guidance, navigation, and flight control functions. The other processor is assigned management functions such as mission planning, mode control, display and resource management, communications, data storage and retrieval, and checkout.

The centralized approach permits the computer system to make use of common spare units to satisfy the reliability requirement of fail operational, fail operational, fail safe with the attendant reduction in total number of units required to implement the system compared to a distributed computer approach.

The approach to redundancy is to provide redundancy at the unit level (CPU, MMU, IOU) in such a way as to permit the functional use of the redundant units, specified under program control, to be total duplication of the function of a principal unit or to share the total processing load with another identical unit and have the total system performance degrade as units fail, retaining a minimum essential level of capability up to a fixed number of failures.

The baseline computer depicted is the RCA 215 which is totally compatible with the RCA Spectra line of commercial computers, thereby capturing the total system and support software from the Spectra line. Another advantage of total compatibility with a major commercial computer system is that application software can be designed, tested, and validated using commercial computers and transferred directly to the space computer, thereby significantly reducing development time and costs.

# Dual Job Stream Baseline



N71-35054

FAULT-TOLERANT COMPUTER DESIGN  
FOR SPACE SHUTTLE

Irving Goodman

General Electric  
Utica, New York

## BASELINE AVIONICS SYSTEM

This chart is shown primarily to indicate the hardware scope of the Data Management Subsystem within the SSS Avionics System. Design of the DMS for fault tolerance must consider:

- . Solution rates, computation speeds, and memory capacities
- . Required speed of error detection/isolation and error recovery
- . Nature and criticality of computational and checkout functions for all mission phases
- . Geographical distribution/location of avionics and non-avionics subsystems serviced by the DMS
- . Man/machine interaction speed and capacity.
- . Flight safety and abort capability needs
- . Degree of autonomy

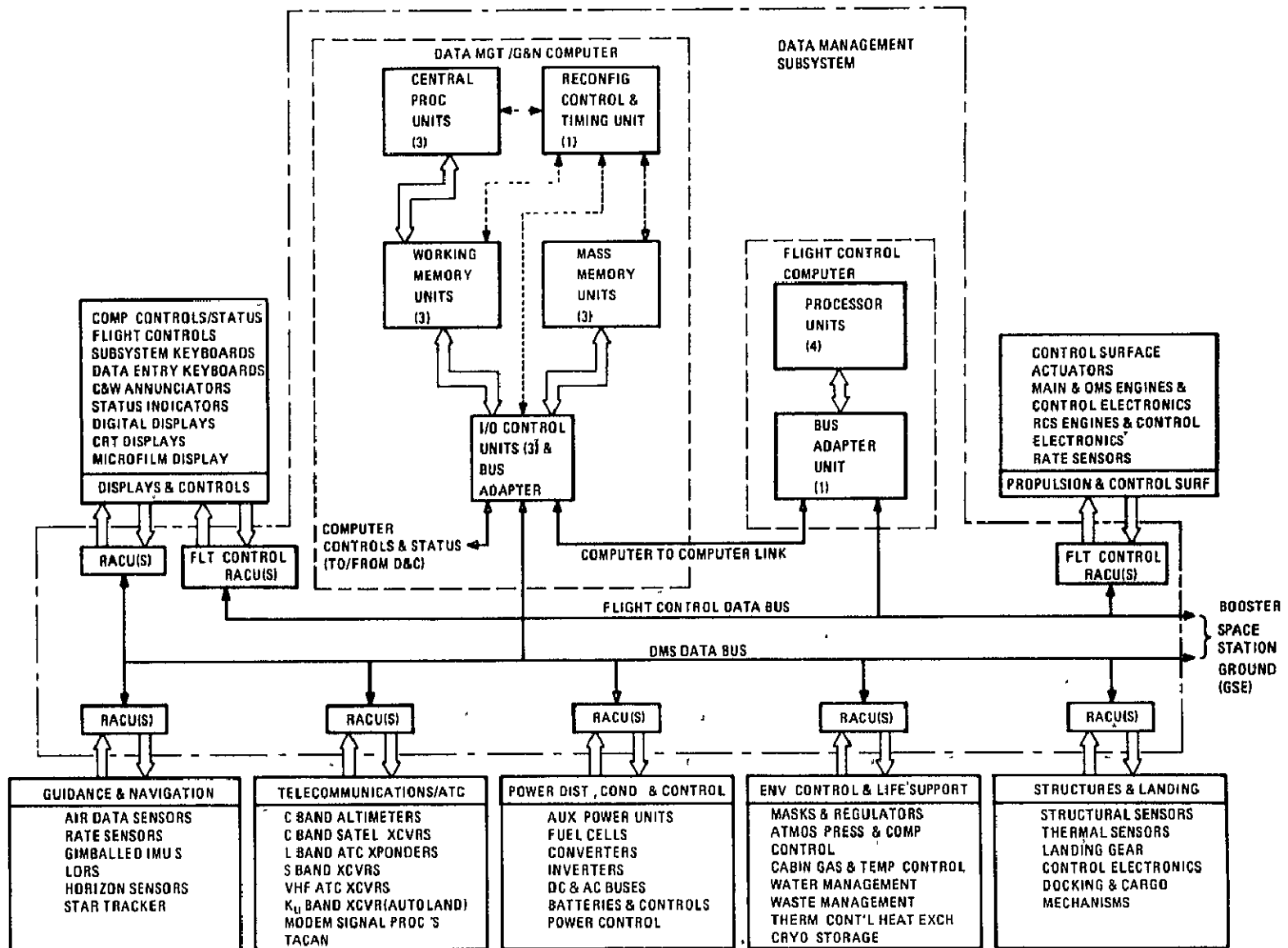
The tools used by the DMS designer to achieve fault tolerance with these considerations are:

- . Reliability and redundancy
- . Architecture for hardware/software modularity and interaction
- . Economical balance between hardware/software fault detection and error recovery configuration control

These considerations and tools will be reviewed in more detail in the following charts.



# BASELINE AVIONICS SYSTEM



## ARCHITECTURE SOLUTION STUDY FLOW

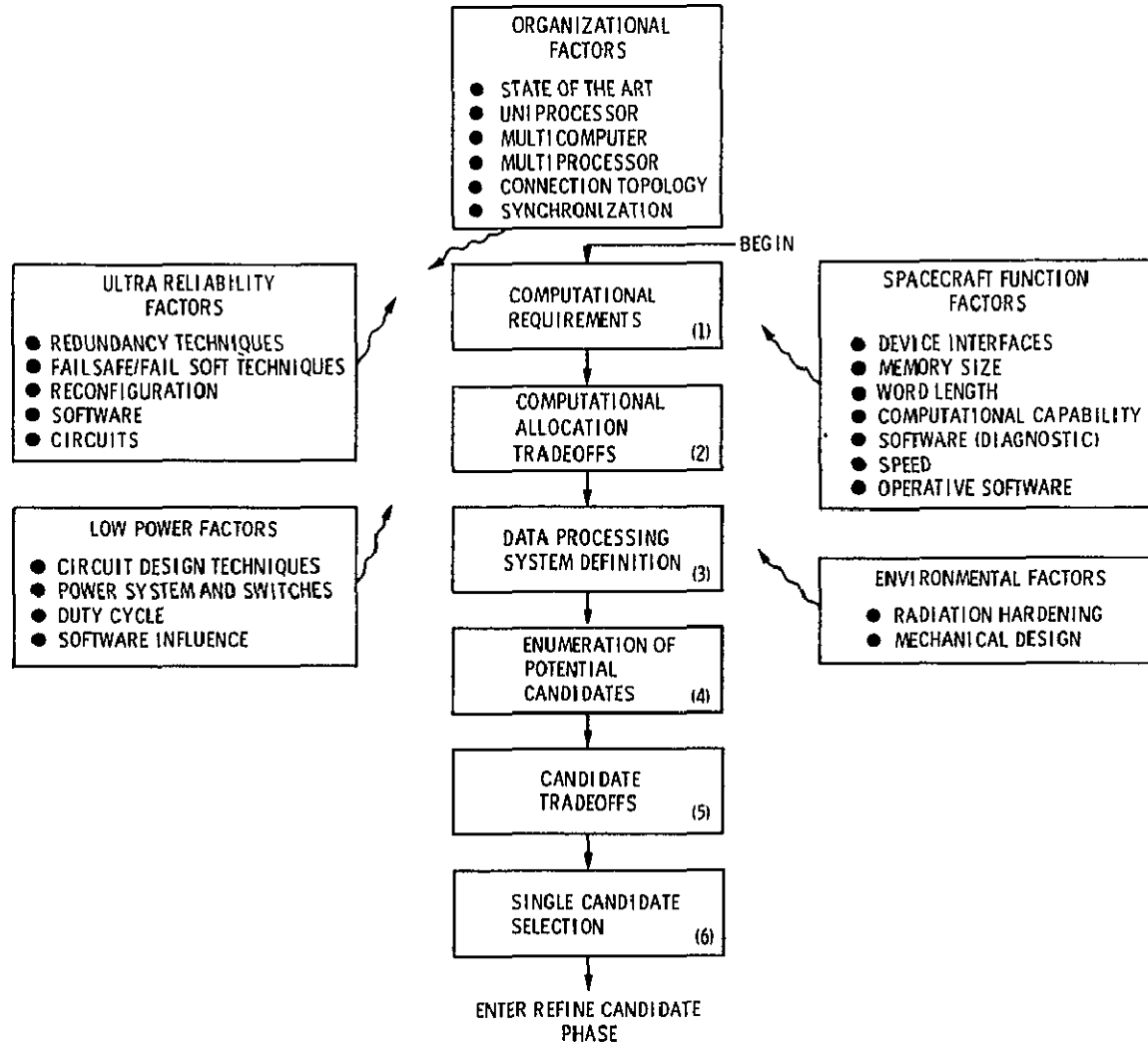
Selection of candidate computer organizations for Space Shuttle encompasses utilization of many disciplines and applying a satisfactory tradeoff methodology. The diagram highlights many of the factors which impact the candidate selection. Selection starts with assessing the state of the art and the basic standard configurations, such as the uniprocessor, multicomputer, multiprocessor and ascertaining their relevance in meeting computational requirements (Box 1). Standard interconnection topology networks are similarly assessed.

A computational allocation tradeoff (Box 2) is initiated. Its objective is to minimize hardware/software complexities by sharing the load through the use of centralized and decentralized regional computers.

The Data Processing System (Box 3) is finally arrived at by combining and weighting all the relevant factors, most of which are enumerated in the Study Flow Figure.

Several candidates are acceptable (Boxes 4, 5, 6), so a "fine tuning" process is needed. Establishing rejection thresholds such as utilizing a power-speed figure of merit helps to eliminate candidates. It is worth noting that most of the selection criteria enumerated will be used again during refining of the design of a selected candidate.

# ARCHITECTURE SELECTION STUDY FLOW



### COMPUTER ORGANIZATION SIMPLEX CANDIDATES

Shown here are some of the types of computer required/available performance parameters which must be considered in DMS design for fault tolerance.

# COMPUTER ORGANIZATION SIMPLEX CANDIDATES

- HIGH PERFORMANCE G & N/DMS CENTRAL COMPUTER  
(32 BIT MACHINE)
  - 30 - 150 LSI BIPOLAR CHIP CONSTRUCTION (4K MEMORY)
  - 15 - 20 LBS WEIGHT
  - 400 - 500 CU INCHES VOLUME
  - ADD TIME                      2 $\mu$ sec
  - MPY TIME                      5 $\mu$ sec
  - DVD TIME                      7 $\mu$ sec
  - MTBF  $\approx$                       20,000 HRS \*\*
  
- DEDICATED FLIGHT CONTROL COMPUTER                      (16 BIT MACHINE)
  - 50 - 75 LSI MOS CHIP CONSTRUCTION                      (2K MEMORY)
  - 0.5 LB WEIGHT
  - 8 CU INCHES VOLUME
  - 10 WATTS POWER
  - ADD TIME                      2 $\mu$ sec
  - MPY TIME                      11 $\mu$ sec
  - DVD TIME                      19 $\mu$ sec
  - MTBF  $\approx$                       35,000 HRS \*\*

\*\*MTBF NOT KNOWN - ESTIMATES ARE PROJECTIONS. FAILURE RATES GIVEN FOR LSI CHIPS - 30 PARTS/10<sup>9</sup> HRS. AND APPLYING A DERATING EXPERIENCE FACTOR

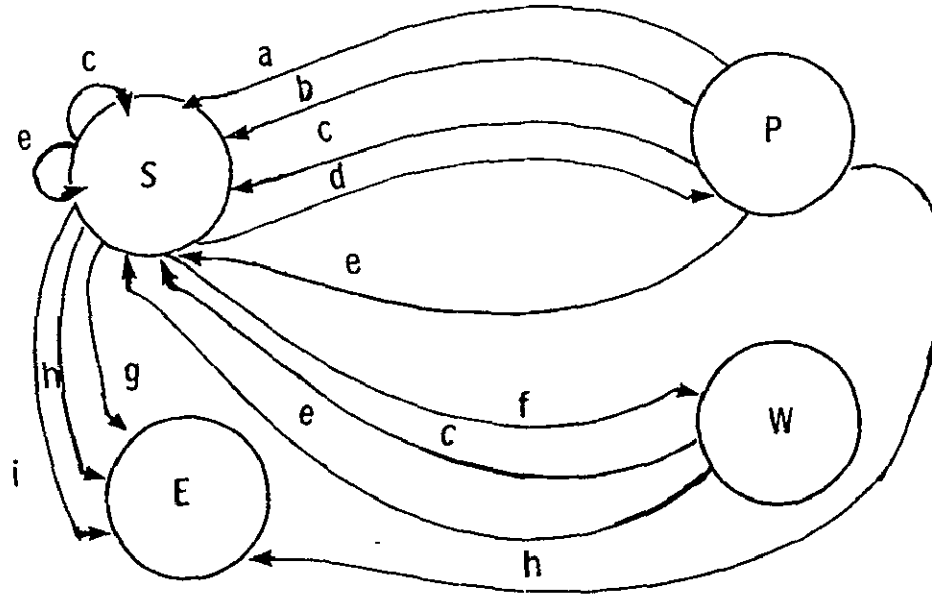
## STATE DIAGRAM INTERRUPT PROCESSING

The hardware/software error program must consider:

- Operating system programs (Supervisor State)
- Application-oriented programs (Problem State)

Error operations are basically interrupt driven. Interrupt processing is featured in most major computer hardware/software systems. This state diagram of interrupt processing of a typical operating system shows computational transitions. The emergency state interests us. It is in this state that reconfiguration of resources takes place.

## STATE DIAGRAM INTERRUPT PROCESSING



- a) PROGRAM INTERRUPTS (PROBLEM STATE)
- b) SVC INTERRUPTS (PROBLEM STATE)
- c) I/O INTERRUPTS
- d) INITIATING PROBLEM PROGRAM EXECUTION
- e) EXTERNAL INTERRUPTS
- f) IDLE
- g) PROGRAM INTERRUPTS (SUPERVISOR STATE)
- h) MACHINE CHECK INTERRUPTS
- i) SVC INTERRUPTS (SUPERVISOR STATE)

S = SUPERVISOR STATE  
P = PROBLEM STATE  
W = WAIT STATE  
E = EMERGENCY STATE

## GENERIC EQUIPMENT/SOFTWARE ERROR PROGRAM RELATIONSHIP

Reconfiguration decisions fall under operating system error operation programs. These programs are imbedded in WMU and RCTU storage resources. Normal operation features transitions between System Check and Monitor programs and application-oriented programs. Error operations programs are distinguished by fault monitoring, configuration sequencing restart, retry and initializing options. The accompanying diagram highlights these aspects by functional flow.



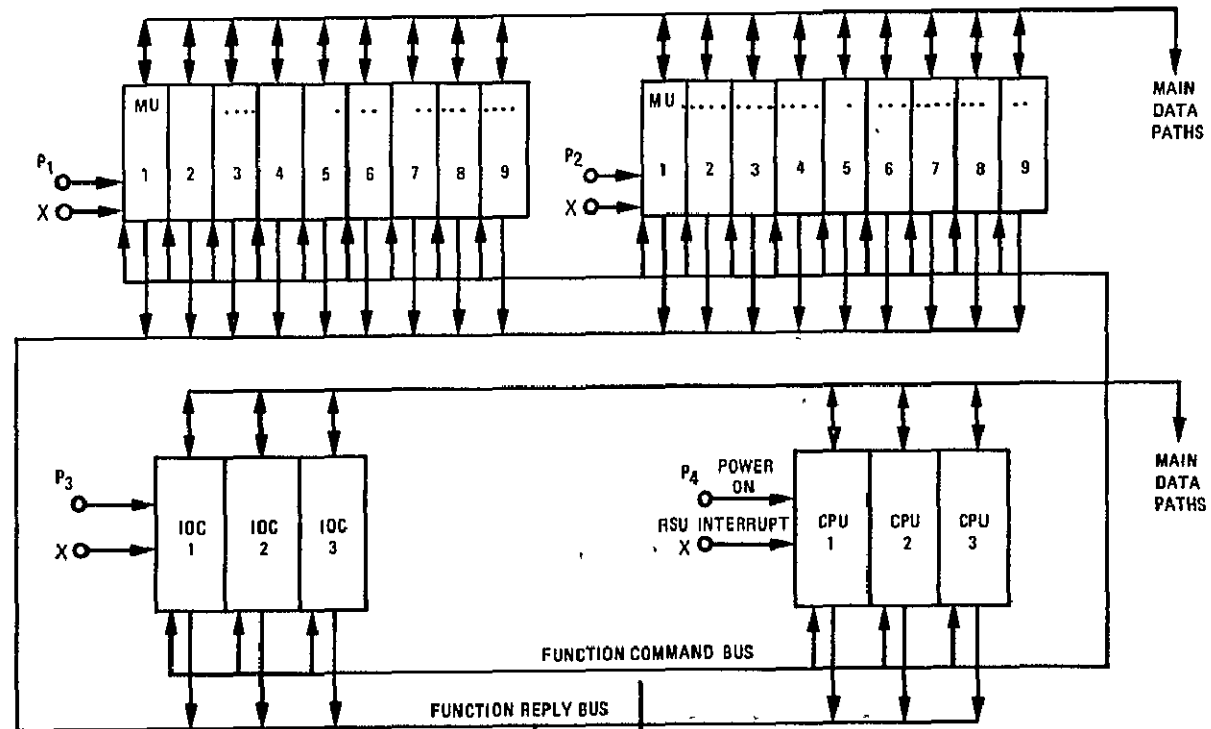


## HARDWARE RECONFIGURATION MODULE

Two hardware techniques are illustrated in this and the following slide. One is based on concurrent error detection delayed diagnosis and recovery - the Reconfiguration Switching Technique; the other represents concurrent error detection, immediate error diagnosis, and correction - the Error Masking and Voting technique.

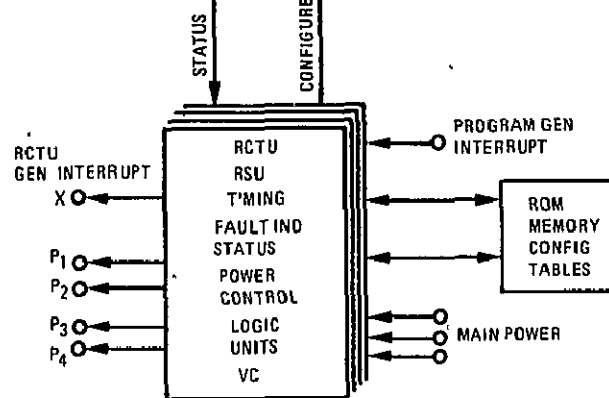
These techniques have been chosen to enhance the reliability of the proposed fault tolerant computer complexes. The fault masking technique is mandatory for the critical mission requirements of the Flight Control Computer complex. The G&N/DMS Computer complex has an alternative option, however, due to its ability to withstand equipment down times and delayed recoveries. Instead of utilizing a separate dedicated functional hardware reconfiguration module, some systems, particularly multiprocessors, have assigned this function to one of the operating processors. It is our contention that fault recovery is slower using this technique, and multiprocessors have been known to fail exercising this function.

# HARDWARE RECONFIGURATION MODULE



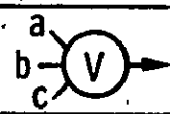
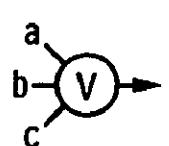
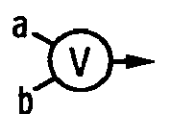
39

MU = MEMORY UNIT  
 CPU = PROCESSOR  
 IOC = INPUT OUTPUT  
 RSU = RECON UNIT  
 P<sub>1,4</sub> = POWER CONTROL  
 VC = VOLTAGE  
 CONVERTERS

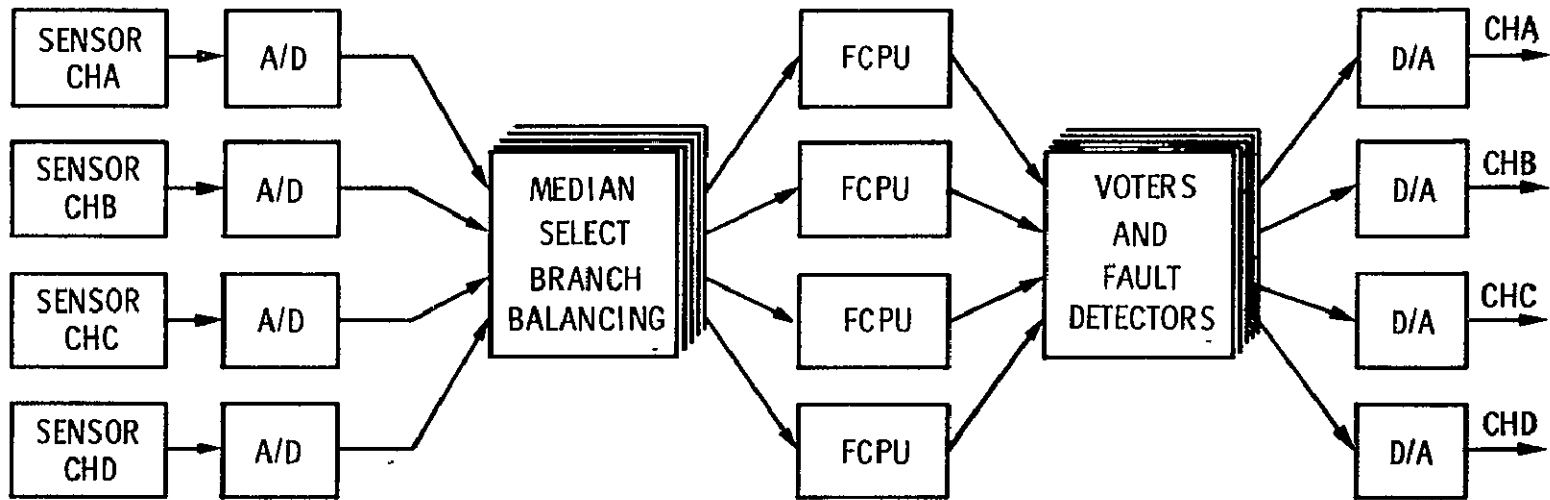


## QUADRUPLEX CONNECTIONS (FO/FO/FS) FOR F/C COMPUTERS

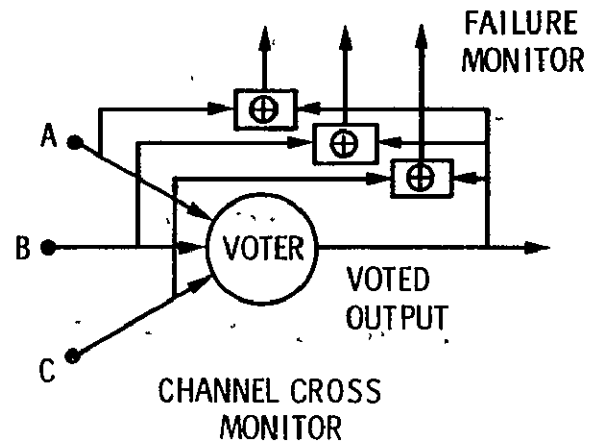
Quadruplex voting for flight control computer computations is indicated by this slide. Triplex voting is much simpler because of less sophisticated switching. The Flight Control Error Control is predicated on reaching a single nonredundant processing case after the third failure. Shown also is cross channel error monitoring. Balancing of data received from redundant sensors is also indicated. The zero to three error case logic equations follow. Isolating the third error requires self-test backup.

ERROR CASE	LOGIC OPERATION
0	$ABC \vee ABD \vee ACD \vee BCD = 1$ 
1	$A = 0; BC \vee BD \vee CD = 1$ $B = 0; AC \vee AD \vee CD = 1$ $C = 0; AB \vee AD \vee BD = 1$ $D = 0; BC \vee BD \vee B \vee CD = 1$ 
2	$A = 0 \quad B = 0 \quad C = 0 \quad D = 0$ $BC = 1 \quad AB = 1 \quad AB = 1 \quad AB = 1$ $BD = 1 \quad AC = 1 \quad AD = 1 \quad AC = 1$ 
3	SELF-TEST DECISIONS

# QUADRUPLEX CONNECTIONS (FO/FO/FS) FOR F/C COMPUTERS



41



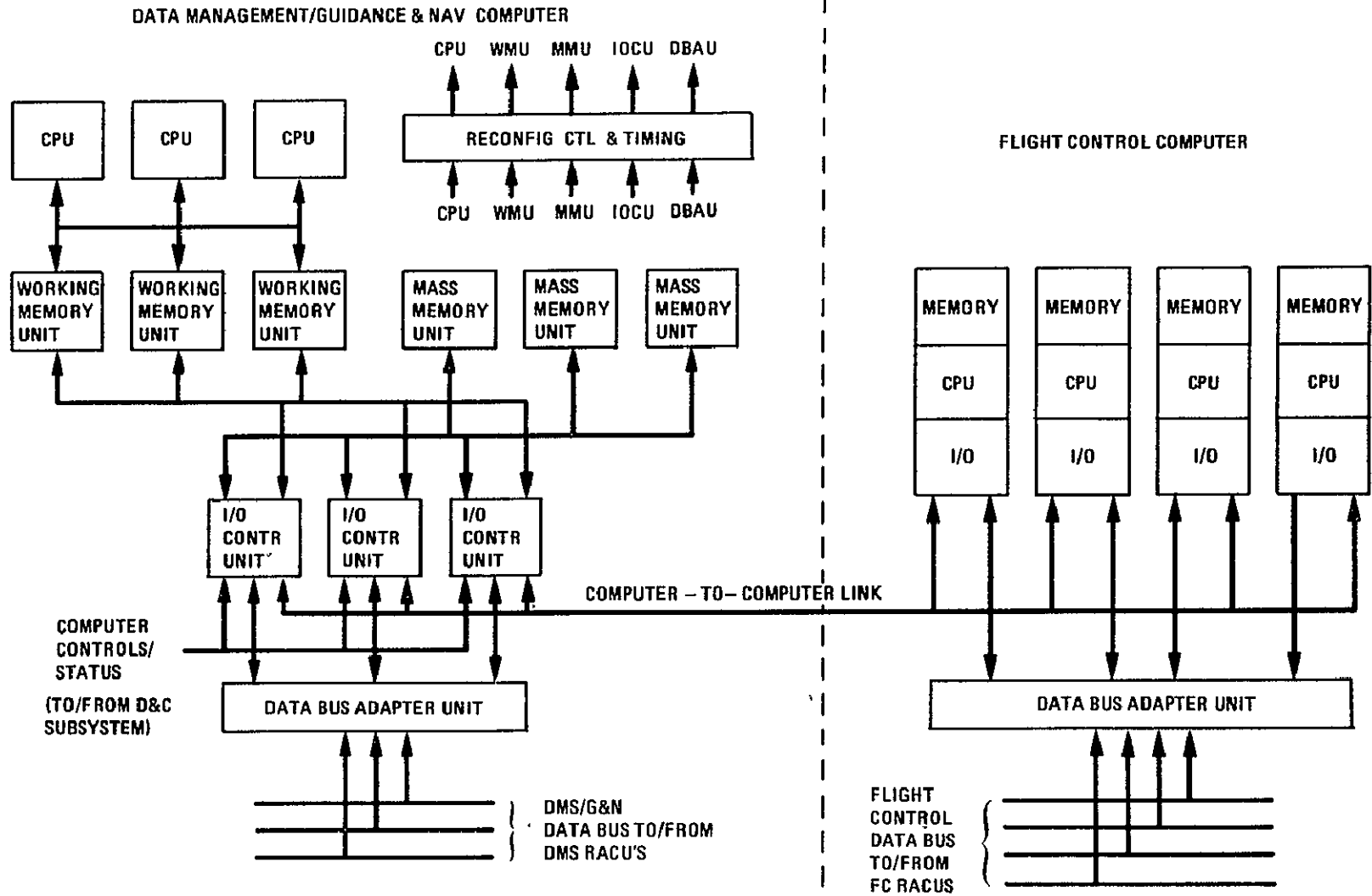
## FINAL BASELINE COMPUTER COMPLEX

Utilizing this fault-tolerant DMS design technique, the following baseline SSS DMS system was designed. It features:

- . A modular central computer with hardware/software error recovery reconfiguration within 1 second and a redundant uniprocessor, multicomputer, and multiprocessor hybrid computation capability
- . A quadruplex uniprocessor flight control dedicated computer with voter error masking recovery of 10 milliseconds
- . A dedicated quadruplex party line data bus for flight control
- . A triplex command/response line data bus for the central computer

Other redundancy features are as shown on this slide.

# FINAL BASELINE COMPUTER COMPLEX



## CONCLUSIONS

Redundancy techniques must be used in order to meet the mission reliability objectives. A candidate architecture which has been proposed has resulted in evaluation of various techniques to be used. Emphasis is placed on practicality of the technique.

Because of the reliability objectives for the Space Shuttle, space and aircraft redundancy techniques are applied. In one part of the computer system, reconfiguration and standby redundancy techniques are suggested. In the more critical flight areas, system transients are not allowed so masking techniques are emphasized. This is the basis for the architecture.

Meeting flight control objectives (aircraft redundancy criteria) reduces the burden on central computer operation. Flight critical areas have been defined as such.

- . Multiple redundancy system and module concepts are employed such that system transients caused by components or (modules) failures are minimized or eliminated
- . The computer will be defined such that the system will be operational after failure of the two most critical components and failsafe after the third failure

The table suggests applicable techniques to enhance computer system reliability. These techniques are suggested as a minimum for incorporation into the computer assemblies.



## **RELIABILITY ENHANCEMENT TECHNIQUES STUDIED AND APPLICABLE**

- REDUNDANCY AT MODULE AND COMPONENT LEVELS
- RECONFIGURATION SWITCHING AND STANDBY SPARING
- ERROR DETECTION CODES (RESIDUE, PARITY)
- SELF-CHECKING MICROPROGRAMS (RETRY, ROLLBACK, SELF-TEST, ETC.)
- TRIPLICATION VOTING , DUPLICATION COMPARISON CHECKS
- OUT OF TOLERANCE CHECKS - ANALOG TYPES, POWER CLOCK, SENSING
- SINGLE BIT CORRECTION TECHNIQUE(HAMMING & RESIDUE)

**N71-35055**

**PRECEDING PAGE BLANK NOT FILMED**

**CONFIGURATION ALTERNATIVES FOR  
DATA BUS SUBSYSTEM INTERFACE UNITS**

**EDWARD CHEVERS AND WILLIAM MALLARY**

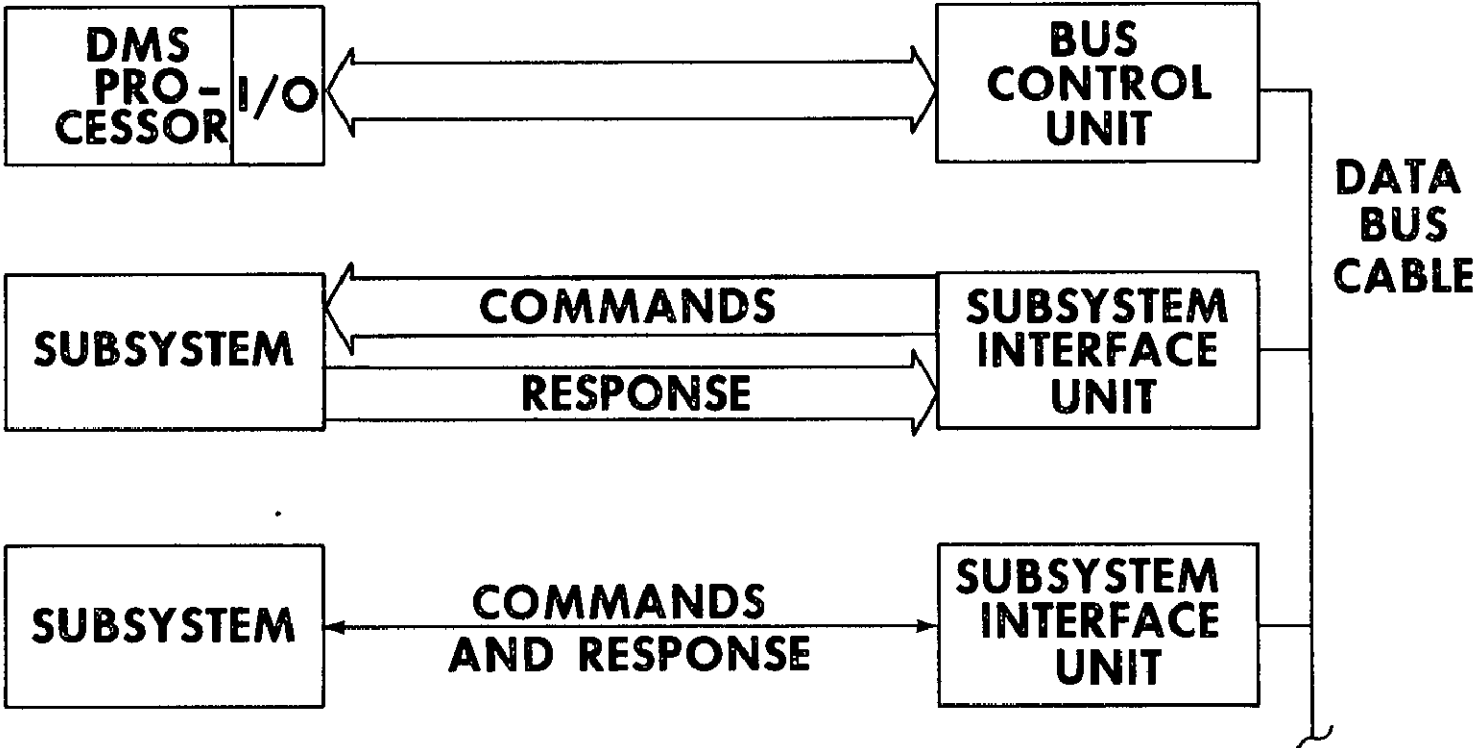
**MANNED SPACECRAFT CENTER  
HOUSTON, TEXAS**

The current space shuttle study and development efforts have evolved data management concepts which utilize a data bus system to acquire data from and to distribute data and commands to the vehicle subsystems. The data bus substantially reduces vehicle wiring by using time division multiplexing and pulse code modulation techniques to transfer data serially between the vehicle data management subsystem (DMS) central processors and the other vehicle subsystems. As illustrated on the following page, the data bus control unit (BCU) interfaces with the central processor and distributes data and commands over a twisted-shielded-pair cable to subsystem interface units (SIU's) which interface with the subsystems. In a similar manner, the SIU's, under BCU control, acquire data from the vehicle subsystems and transmit the data to the BCU where it is provided to the central processor.

This presentation will review three basic approaches to the solution of how to implement the functions of the SIU. Two configurations convert the digital data into analog voltages or discrete mode commands. The third configuration in effect utilizes a sub-bus to distribute commands and acquire data from the individual subsystem channels. For each configuration, the relative impact on data acquisition, command distribution, pertinent advantages and disadvantages of interfacing, reliability, and software implications will be noted.

No attempt is made to define an optimal solution. Our effort in this paper is directed toward presentation of the considerations necessary to effect a final design for a functional unit which all too often is looked upon as merely another black box in the data path. The Electronic Divisions at the Manned Spacecraft Center consider the SIU to be probably the single most important element in the integrated avionics functional path. This is the unit which bridges the gap between the nebulous world of computers and software and the very real world of hardware sensors, controllers, and actuators.

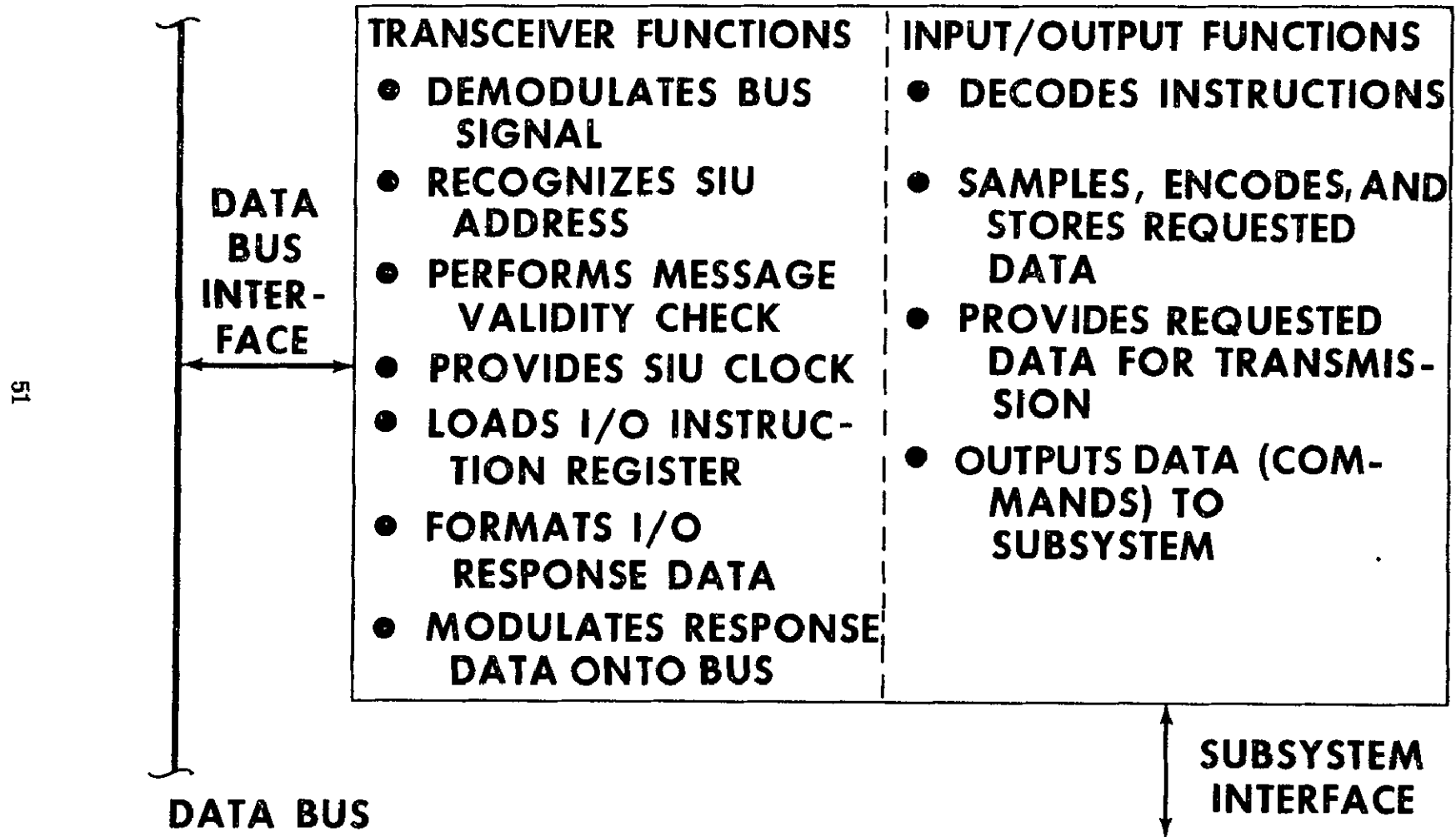
# DATA MANAGEMENT SUBSYSTEM SIGNAL FLOW



The SIU's, then, function in the shuttle to interface the DMS central processors with the vehicle subsystems via the data bus. To fulfill this requirement, the SIU's must accomplish the following tasks:

- a. Signal demodulation - The SIU must interface with the data bus cable and derive data from the bus signal.
- b. Address recognition - The SIU must decode the received data to determine if it is the intended destination of the bus message.
- c. Message validation - The SIU must determine that the received message is error-free.
- d. Data acquisition - The SIU must respond to data request instructions by sampling the requested data point, encoding or converting it to a digital form, and sending it via the data bus to the DMS data processor.
- e. Command/data distribution - The SIU must respond to command instructions by converting the command or data into a form usable by the vehicle subsystem and outputting it from the designated signal buffer. The SIU must also provide an indication to the BCU that the command was executed.
- f. Signal transmission - The SIU must convert the SIU response data into a form compatible with the data bus modulation scheme.
- g. Checkout - The SIU design and operation must provide the BCU a means to ascertain that the SIU is operating properly.

# SUBSYSTEM INTERFACE UNIT FUNCTIONS



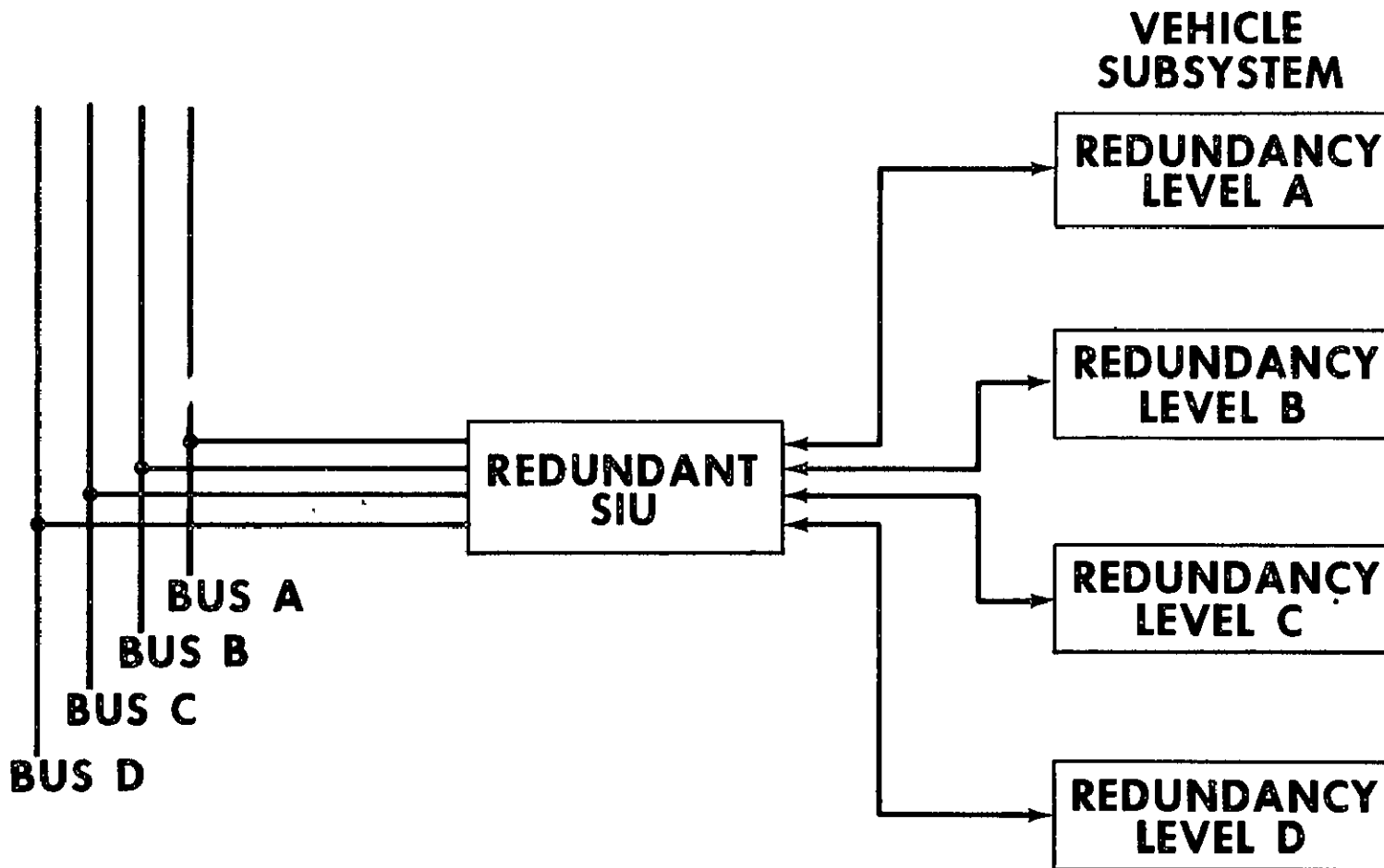
In addition to the previously discussed operating requirements, the SIU must satisfy shuttle reliability requirements which dictate that the DMS, which includes the SIU's, must permit the shuttle vehicle to perform its design mission after two failures and permit the shuttle to return to base after three failures. To satisfy this requirement, the current shuttle DMS design concepts use a data bus which incorporates four levels of redundancy. In general, four data bus cable sets interface with an SIU which contains four levels of redundancy. The SIU in turn interfaces with the subsystem which may also incorporate up to four levels of redundancy, depending on its mission criticality.

The SIU, then, is required to interface with four data bus cables and with up to four levels of subsystem redundancy in a manner which will permit the SIU to perform its functional requirements after as many as three failures.

There are three SIU design alternatives which are currently being considered by MSC. Each satisfies the basic requirements for redundant operation but imposes restrictions on DMS operation. The following text describes these alternatives in terms of the partitioning of SIU functions and then discusses the impact of each design on critical DMS and vehicle subsystem considerations.

To form a basis for describing each SIU design alternative, the SIU functions are categorized as either transceiver or input/output functions, and the SIU is partitioned into two functional elements. Each SIU design configures the SIU transceiver and input/output functions differently. For reference purposes, these alternatives will be designated as configurations 1, 2, and 3.

# REDUNDANT SIU INTERFACES





Configuration 1, which is illustrated on the following page, is the simplest of the three approaches to satisfying the shuttle redundancy requirements. A simplex (non-redundant) SIU interfaces with each data bus cable and each level of subsystem redundancy. No cross coupling is used at either the data bus/SIU interface or the SIU/subsystem interface. Data transfer between the DMS processor and a subsystem redundancy level is restricted to the single path provided by the associated SIU and data bus cable. Complete isolation is easily maintained between redundant levels of the data bus and subsystem.

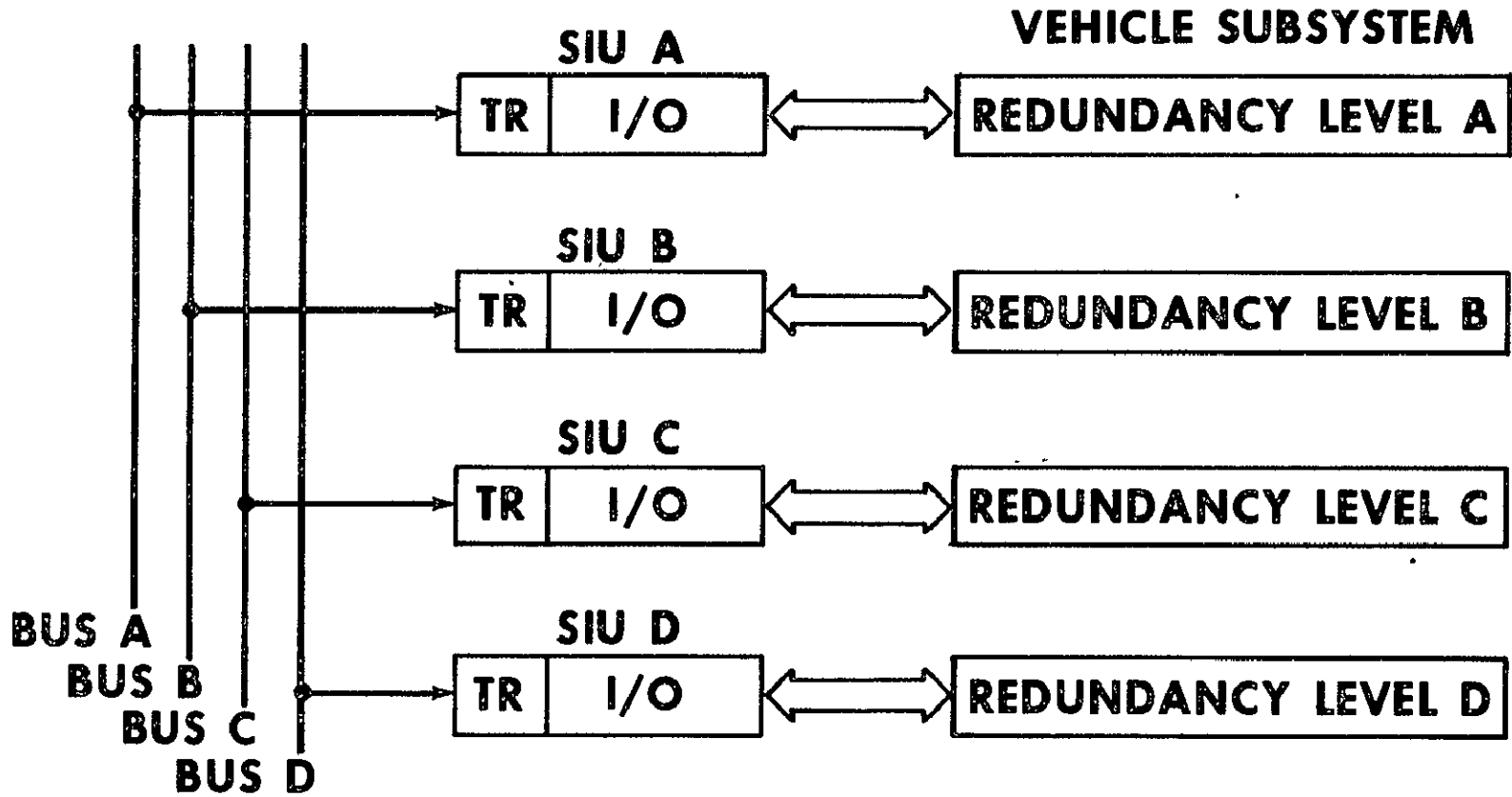
In this configuration, the DMS processor may either acquire data from and output commands to the redundant subsystem elements simultaneously, over all four bus cables and associated SIU's, or it may address each subsystem redundancy level individually. In the case of simultaneous addressing, both the DMS processor and the vehicle subsystem would use voting techniques to determine the validity of the received data and/or commands.

Simultaneous addressing also necessitates some form of data buffering at the BCU or each computer input/output so that each feedback signal from the SIU's can be kept isolated until ready for processing in the DMS processor.

## CONFIGURATION 1

- **SIMPLEX INTERFACE WITH EACH DATA BUS CABLE**
- **SIMPLEX INTERFACE WITH EACH SUBSYSTEM CHANNEL**
- **SUBSYSTEM CHANNELS MAY BE ADDRESSED INDIVIDUALLY OR SIMULTANEOUSLY**
- **COMPLETE ISOLATION BETWEEN BUSES AND SUBSYSTEM CHANNELS**

# SIU CONFIGURATION 1



Configuration 2 differs from configuration 1 in that each SIU interfaces with all four bus cables. The SIU's incorporate redundant transceiver sections to provide interbus isolation. Each input/output section, however, is still simplex and interfaces with only a single subsystem redundancy level.

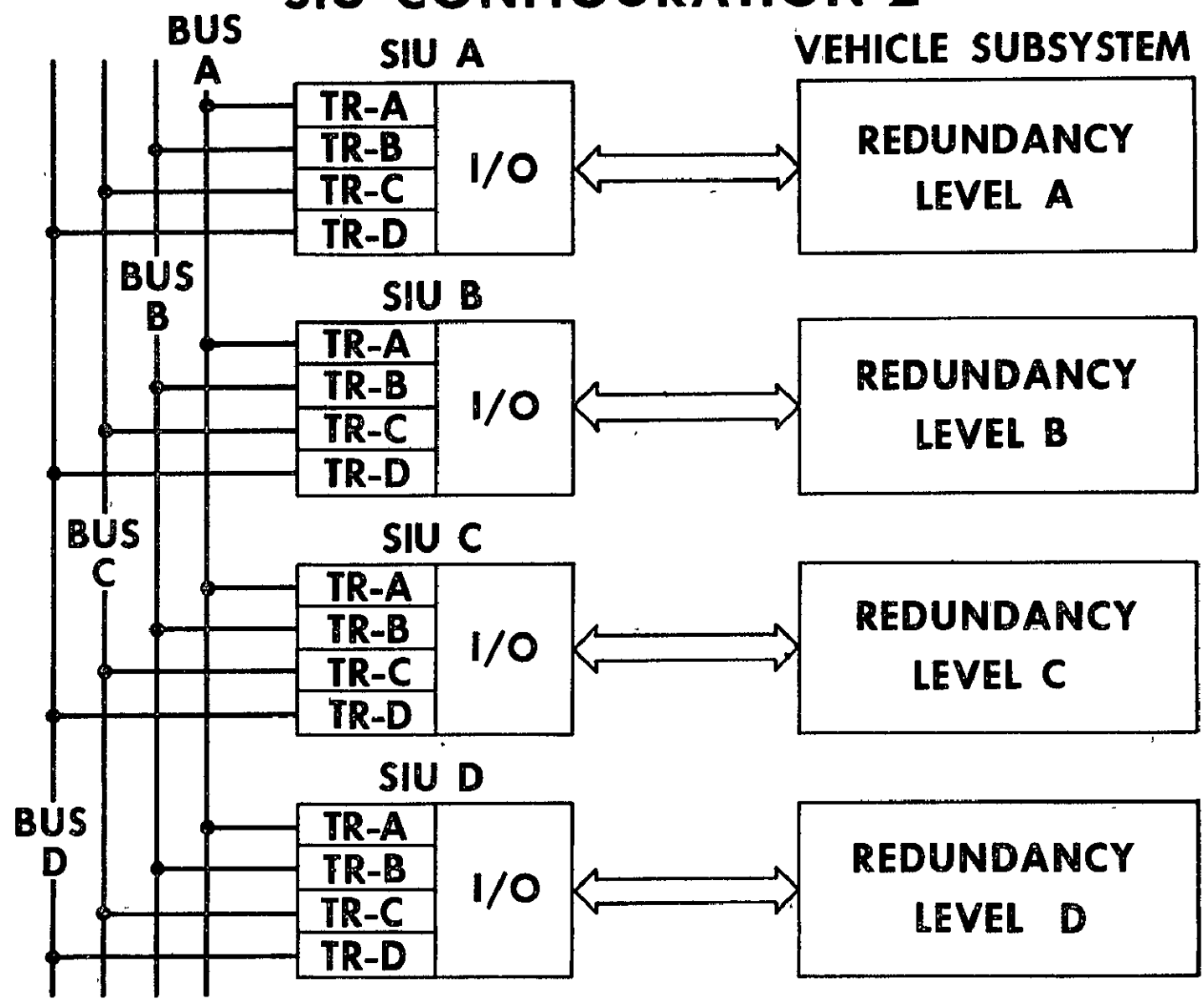
The DMS processor is now able to communicate with any subsystem redundancy level via any of the four data bus cables. Like configuration 1, the DMS processor may address the subsystem redundancy elements either individually or simultaneously. Each SIU must, however, have a unique address, and SIU complexity increases because of the additional transceiver circuits and the requirement for the SIU to know which data bus to transmit on. The primary advantage to configuration 2 is the protection against bus cable failures due to the multiple signal paths provided by cross coupling in the data bus/SIU interface.

## CONFIGURATION 2

- EACH SIU INTERFACES WITH ALL FOUR DATA BUS CABLES
- SIMPLEX INTERFACE WITH EACH SUBSYSTEM CHANNEL
- SUBSYSTEM CHANNELS MAY BE ADDRESSED INDIVIDUALLY OR SIMULTANEOUSLY
- EACH SIU REQUIRES A UNIQUE ADDRESS PLUS GROUP ADDRESS RECOGNITION CAPABILITY IF SIMULTANEOUS ADDRESSING USED

2  
⊕

# SIU CONFIGURATION 2



Configuration 3 actually involves two different techniques which are extensions of configurations 1 and 2. The diagram shows configuration 1 expanded to allow cross coupling of each SIU to each subsystem channel. This permits the DMS processor to address each subsystem channel through any one of the four data bus cables even after three successive SIU failures. Again, system flexibility is increased but at the expense of increased hardware and software complexity. The DMS processor now must address both an SIU station and a specific subsystem channel behind the SIU. The primary advantage of this configuration is the ease with which a transition may be made from four data buses to any level "n" of subsystem redundancy.

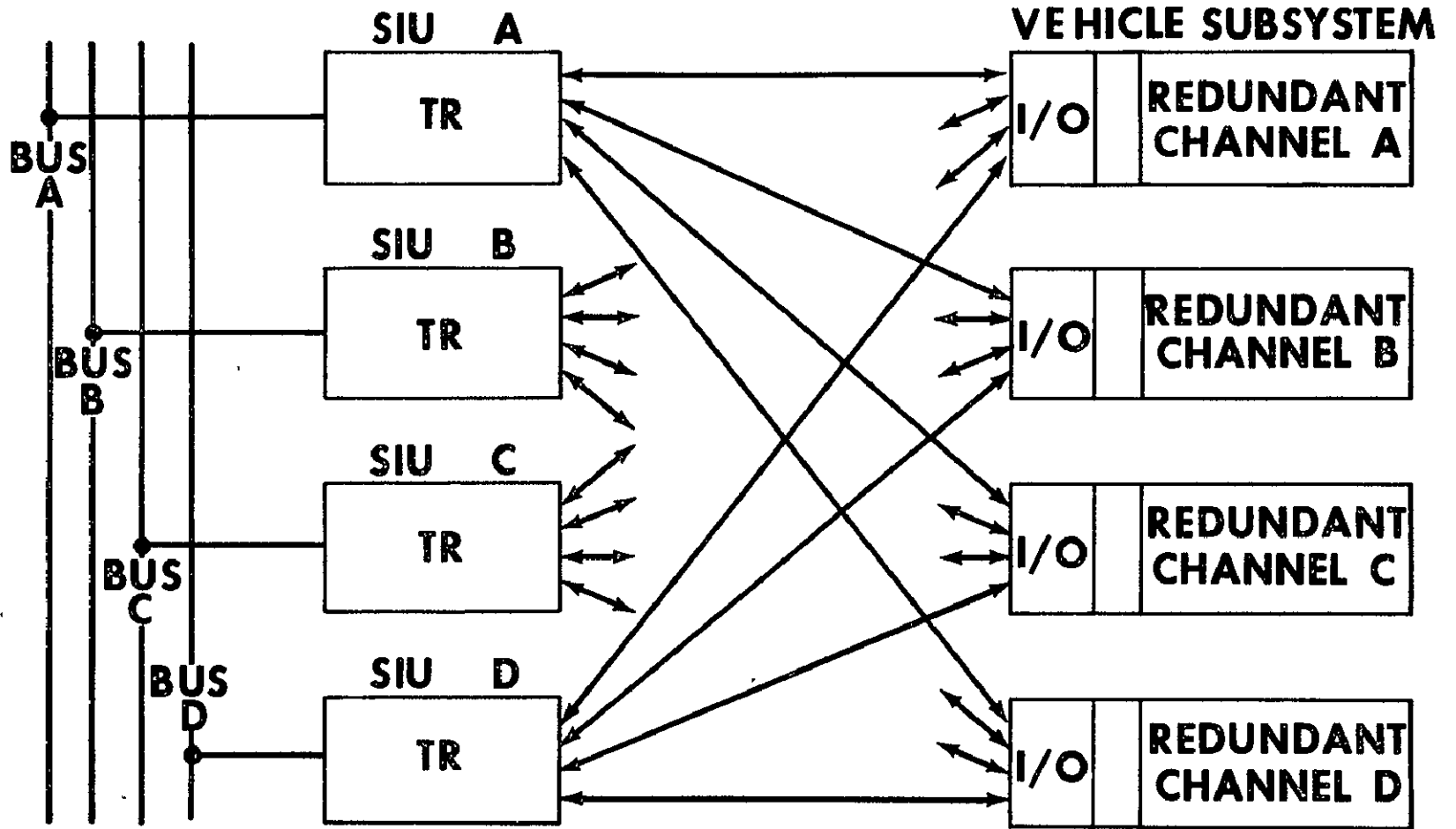
The increase in interconnection complexity is readily apparent in the diagram and is a significant disadvantage. However, this particular cross-coupling technique does appear to be the most practical approach for access to and usability of all the inherent reliability provided by a multiply redundant subsystem such as the triply redundant hydraulic actuators and quadruply redundant electromechanical actuators being considered for the shuttle.

The other variation of configuration 3 is an extension of the configuration 2 technique to allow cross coupling at both the data bus/SIU interface and the SIU/subsystem interface. This particular configuration is not considered a likely candidate for flight application since overall system flexibility is virtually unaffected when compared to the basic concept, and total system reliability may actually be decreased as a result of the additional hardware and interconnections. There are no firm reliability numbers to prove this argument; however, preliminary laboratory breadboard investigations tend to support the decreased reliability assumption.

### **CONFIGURATION 3**

- **SIMPLEX INTERFACE WITH EACH DATA BUS CABLE**
- **CROSS COUPLING OF EACH SIU TO EACH SUBSYSTEM CHANNEL**
- **SUBSYSTEM CHANNELS MAY BE ADDRESSED INDIVIDUALLY OR SIMULTANEOUSLY WITH GROUP ADDRESSING**
- **REQUIRES BOTH SIU ADDRESS AND SUBSYSTEM CHANNEL ADDRESS**
- **PERMITS EASY TRANSITION FROM FOUR DATA BUSES TO ANY NUMBER OF SUBSYSTEM CHANNELS**

# SIU CONFIGURATION 3



The software impact of configurations 1 and 2 is very nearly the same. The DMS processor will first select a subsystem channel to access for information or to issue commands. In configuration 1, the processor then selects one of the four buses to match the subsystem channel, and we are now assuming the processor has available sufficient information as to the operational health of each SIU/subsystem pair. When a command is issued, the SIU must have the capability to determine the validity of the command before releasing it to the subsystem for execution. This may be done through error detection codes and requests for retransmission, error correction codes, voting on multiple commands, verification by echo checking, plus numerous variations of one or more of these techniques. The error detection/correction codes and voting techniques all tend to increase SIU and subsystem hardware complexity, whereas the echo check decreases available data bus capacity. Studies are planned at the Manned Spacecraft Center to define the relative impact of each technique on both hardware and software for each of the mentioned data security methods, but initial investigations tend to indicate a slightly lower software penalty for techniques incorporating echo checks. This seems to be a result of continuously available information in the central processor in regard to the health status of each data bus cable/SIU combination.

69 Configuration 2 has essentially the same degree of software complexity as configuration 1. The only difference in this case is the available knowledge of SIU/subsystem interface status and reference to a data bus priority table in the computer memory. Interfacing each SIU with all four data buses does permit voting at the SIU of simultaneous messages, but again this tends to increase SIU hardware complexity. This would be especially true if all data bus traffic was not bit synchronized at the bus control unit. However, bit synchronization increases BCU and input/output controller complexity; therefore, changes in one area must be traded off against resulting impacts in other areas of the system.

As might be suspected, configuration 3 does impose the most severe software penalty. This is the price which must be paid to achieve the high level of flexibility available from this interconnect configuration. The DMS processor must maintain a status record of each SIU, each subsystem channel, plus the interconnecting cables between SIU's and subsystem channels. The added complexity of keeping a record of the cables is to prevent the needless discard of a good subsystem channel because a cable has failed, when there is an alternate path into the subsystem from each of the other SIU's.

From the descriptions of the three SIU configurations, it is evident that the number of selectable data paths between the DMS processor and subsystem redundancy elements is increased at the cost of both hardware complexity and bus operational efficiency. Also affected is the rate at which redundant data can be acquired from the vehicle subsystems. Configuration 1 permits the simultaneous operation of all four of its redundant data paths, and data can be acquired from all subsystem redundancy levels at the same time. Even when one or more signal paths fail, sufficient data are presented to the DMS processor to permit computation.

Configuration 2 can also simultaneously acquire data from all subsystem redundancy levels. The normal mode of operation, however, is the sequential acquisition of data from one subsystem redundancy level at a time, utilizing a single data bus cable. This mode is normally used because a substantial reduction in SIU transceiver electronics can be obtained when it is not necessary to decode bus signals from more than one data bus at a time. Bus efficiency does suffer, however, because the time required for the DMS processor to acquire data increases as a factor of the number of subsystem redundancy levels sampled.

Configuration 3, like configuration 2, acquires data from each subsystem redundancy level in a sequential manner. Effective data acquisition rates are further reduced in this configuration by the increase in bus message overhead caused by the requirement to uniquely address both the SIU transceiver and input/output sections.

The distribution of DMS processor-generated commands and data to the vehicle subsystems is also affected by the choice of SIU configurations. Configurations 1 and 3 are equally efficient in terms of command rates, as each can output simultaneous commands to each subsystem redundancy level within a single instruction time. Configuration 1 accomplishes this by transmitting simultaneous commands to each subsystem redundancy level over all four data buses. Configuration 3 does as well by transmitting a single command containing an input/output group address which causes all input/output sections to generate identical data or commands.

Configuration 2 outputs data to only one subsystem redundancy level at a time. A simultaneous command capability is possible by requiring that the SIU's respond to a group address code. This is not normally considered, however, because of hardware complexity considerations.

A comparison of redundancy levels between configurations 1, 2, and 3 is somewhat dependent on where in total system flow a redundancy boundary is drawn. By this we mean, at what point does the subsystem begin and the DMS and/or data bus end. For purposes of this presentation, the subsystem will be considered to include the SIU and its interface with the data bus. This approach is taken because of the various interface techniques being considered and the fact that data bus cable failures have significant impacts in both DMS software and the subsystem redundancy.

Configuration 1 is the least reliable when considered from a straight reliability string approach. This is a result of having the data bus, SIU transceiver, SIU input/output, and subsystem channel all in a single string. Therefore, a failure in any one of the four units renders the entire string useless. It should be noted that a failure of the data bus cable is even more significant since it not only eliminates the subsystem channel near the fault but also all other channels which are attached to that particular bus.



The one main advantage of configuration 1 is the relative simplicity of the hardware which does serve to offset some of the unreliability of a single string system. Power, packaging, and interconnections can all be simplified in this configuration, and subsystem reliability improved through good quality control procedures.

Configuration 2 tends to improve overall system reliability by eliminating the loss of one channel in all subsystems as a result of a single data bus cable failure. The penalty paid is increased hardware complexity at the SIU transceiver input. As mentioned earlier, care must be taken with this approach to insure against a failure at one input propagating across to all inputs such that a failure condition exists on all four data buses.

For each of these configurations, the most unreliable part of the SIU will normally be the transceiver section. This is especially true if linear amplifiers and drivers must be used for the transmitters and receivers. Therefore, configuration 2 should be more reliable than configuration 1 since the most unreliable portion has been made redundant. However, this configuration still permits the loss of a good subsystem channel in the event of a failure in the SIU input/output section. Also, the increase in reliability must be calculated carefully with due consideration for the added hardware complexity and circuit design incorporated with quadruply redundant transceiver sections. If high power components and extreme accuracy are necessary to satisfy design goals, it is possible to arrive at a quadruply redundant unit which is less reliable and an order of magnitude more expensive than the simplex SIU in configuration 1.

Configuration 3, like configuration 2, does provide the capability of being able to use each subsystem channel after failures in the data bus cable or the SIU transceiver section. The single transceiver section of configuration 3 tends to reduce the hardware complexity which provides some increase in system reliability when compared with configuration 2; however, this gain is offset to a certain extent by the interconnections required between the SIU transceivers and subsystem channel input/output sections. Therefore, the penalty paid for the high level of redundancy with configuration 3 is the unreliability of multiple connections, increased software complexity in checkout, and increased software complexity for inflight failure monitoring.

As can be seen from the diagram, there are 16 interconnecting cables between the four SIU's and a quadruply redundant subsystem. The increased software comes from the fact that failure isolation must be carried to the cable level for this configuration. If not, a good subsystem channel may be discarded because of a failed interconnect cable. From this it can be seen that failure detection and isolation logic increase at four times the rate of increase in subsystem redundancy levels. Subsystem checkout complexity is also increased in configuration 3 when compared with configurations 1 and 2.

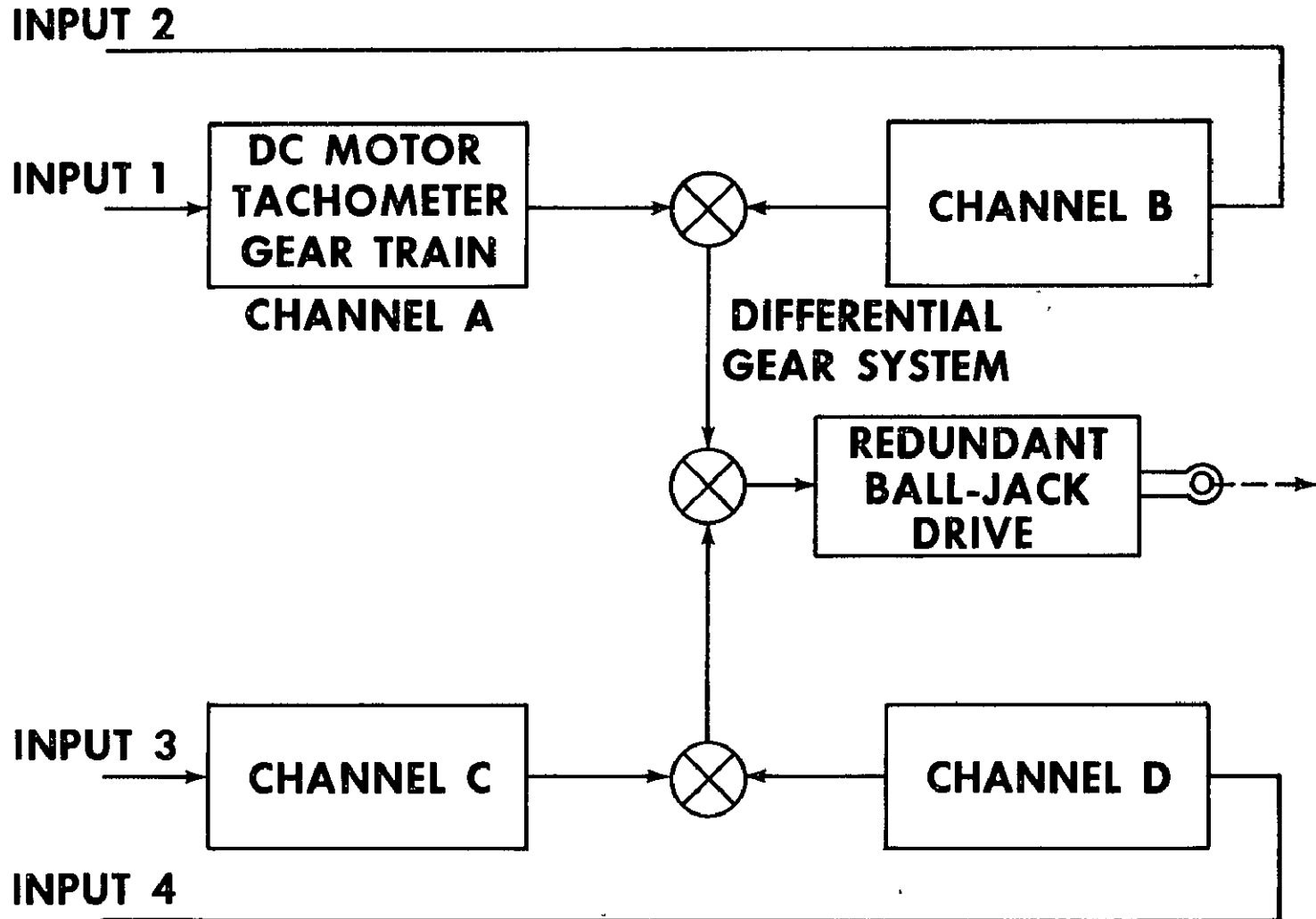
## SIU COMPARISON

	CONFIGURA- TION 1	CONFIGURA- TION 2	CONFIGURA- TION 3
<b>DATA BUS/SIU INTER- FACE COMPLEXITY</b>	<b>LOW</b>	<b>HIGH</b>	<b>LOW</b>
<b>SIU/SUBSYSTEM INTER- FACE COMPLEXITY</b>	<b>LOW</b>	<b>LOW</b>	<b>HIGH</b>
<b>SYSTEM FLEXIBILITY</b>	<b>LOW</b>	<b>MED/HIGH</b>	<b>HIGH</b>
<b>HARDWARE COMPLEXITY</b>	<b>LOW</b>	<b>MED</b>	<b>HIGH</b>
<b>SOFTWARE COMPLEXITY</b>	<b>LOW</b>	<b>LOW</b>	<b>HIGH</b>
<b>DATA ACQUISITION RATE</b>	<b>HIGH</b>	<b>LOW</b>	<b>LOW</b>
<b>COMMAND DISTRIBUTION RATE</b>	<b>HIGH</b>	<b>LOW</b>	<b>HIGH</b>
<b>OVERALL SYSTEM RELIABILITY</b>	<b>LOW/MED</b>	<b>MED/HIGH</b>	<b>HIGH</b>
<b>ABILITY TO INTERFACE WITH DIFFERENT LEVELS OF SUBSYSTEM REDUN- DANCY</b>	<b>LOW</b>	<b>MED</b>	<b>HIGH</b>

Now let us take a brief look at the subsystem interface problems associated with vehicle control. Control is segregated from other spacecraft functions because of the high risk to crew safety in the event of a subsystem failure. This is most easily seen by comparing an actuator subsystem with a sensor or status monitoring subsystem. Although both are vital to mission success, and in some instances sensors can warn the crew of impending or potential danger, a failure of a rocket steering actuator or an aerodynamic surface mover quite often will be catastrophic if it occurs during a time critical mission phase.

To overcome some of these problems, aircraft and aerospace manufacturing companies have been developing redundant actuators for several years. Two of the more advanced designs include a quadruply redundant electromechanical design and a triplex self-monitoring/self-correcting hydraulic unit. In the case of the quadruply redundant unit, the interface with configurations 1 and 2 is straightforward. Each SIU input/output is connected to an individual channel of the actuator. The one problem with this system is the lack of status information from a given channel if its associated SIU fails. This poses no problem if all four channels are commanded simultaneously, except now each SIU must be cross referenced by time or output command signal to insure against one actuator channel fighting another. The cost is increased complexity in the SIU plus increased complexity in the actuator to insure satisfactory operation after SIU failures.

# QUAD REDUNDANT ELECTROMECHANICAL ACTUATOR



To alleviate some of these problems, several redundant hydraulic actuators have been developed which incorporate internal self-monitoring and hydraulic voting techniques. One such arrangement is shown here. This particular unit has three hydraulic channels in an active-standby-standby arrangement. Six electrical signals are required for the paired active and monitor input commands. If either one of the actuator paired inputs fail, a differential signal forces channel 1 off and activates channel 2. Similarly, a failure in channel 2 activates channel 3, and channel 2 goes off. The system may be reset to channel 1 manually, after which it will proceed down through the sequence again if the failure still exists.

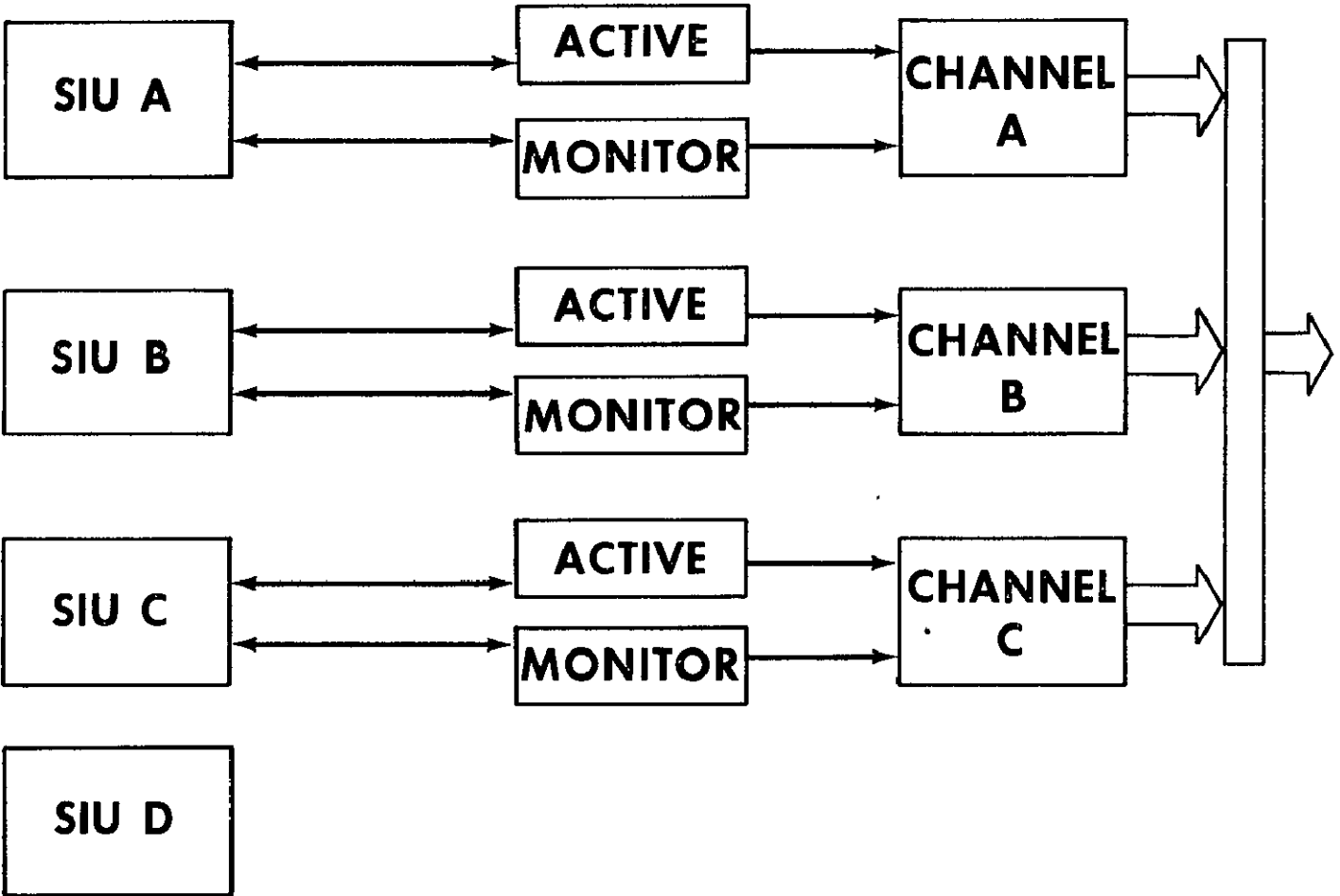
The two primary problems with this arrangement are the interfacing of three channels, with six command signals required, to four SIU's. Configuration 2 will work nicely if one SIU is eliminated. Notice, however, if SIU A fails such that identical signals go to both the active and the monitor inputs on channel 1, the actuator will not switch to channel 2, and the subsystem is lost. To prevent this happening, each SIU must be made internally redundant, which adds to its complexity. Other solutions include the use of six configuration 2 SIU's or the use of configuration 3. Configuration 3 solves the problem by interfacing input/output sections with each of the command channels.

66

These two examples were selected because they are presently undergoing extensive test and evaluation at MSC. The evaluation includes not only their capability to perform as designed in the flight control area, but also their impact on redundancy management and DMS software concepts. The electrical power distribution subsystem is also under study, and plans are underway to include other subsystems in the near future. These will probably be the non-avionic hydraulic, cryogenic, environmental control, or fuel cell subsystems.

# TRIPLY REDUNDANT HYDRAULIC ACTUATOR

## SELF-MONITORING SELF-CORRECTING HYDRAULIC ACTUATOR

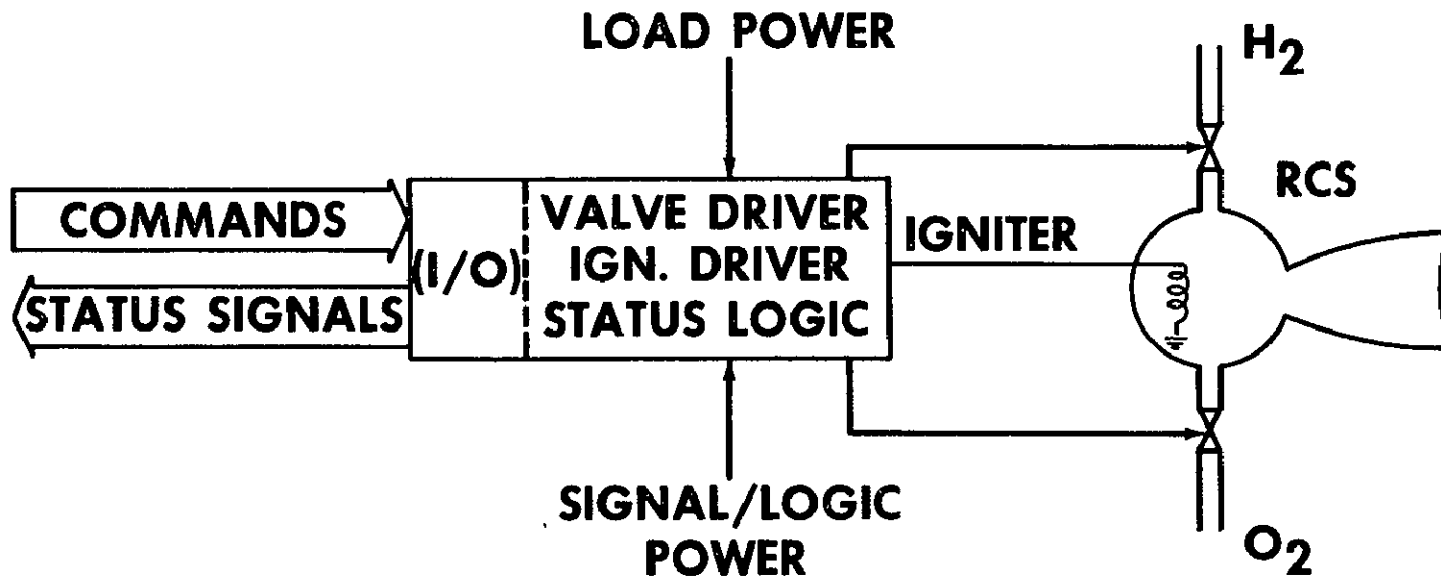


For completeness of the subsystem interface area, the next diagram depicts a typical single input subsystem. The RCS thrusters being considered for the shuttle do not use hypergolic fuels and must be started with some form of igniter. Although not shown here, one thruster configuration actually uses a small 25-pound-thrust engine inside the larger 2,000-pound-thrust engine as an igniter. Therefore, this subsystem while seemingly straightforward, based on past experience, will pose some intricate time sequencing problems, plus requiring interfaces with up to 10 different driver stages. The single input command channel is also awkward to interface with any one of the basic configurations while still maintaining a redundant path from the DMS processor down to the subsystem channel.

Another area which needs to be investigated in considerable detail is power distribution to the individual SIU's and subsystem channels of configurations 1, 2, and 3. This topic will not be taken up here since it deserves an entire presentation to itself, plus one of the conference sessions has been devoted to the subject of Instrumentation and Power Distribution. Let it suffice to say that power distribution and control techniques must be considered early in the design and development of an integrated avionics system. The approach taken and methodology employed can have serious repercussions in subsystem reliability and DMS software if each area is defined independently and then put together downstream in the design phase.

Here at the Manned Spacecraft Center, power distribution and power control have been taken into consideration in both breadboard systems designed by the Information Systems Division and the Guidance and Control Division. This area also represents one of the major design considerations in the integrated avionics breadboard which is being developed from the individual division efforts.

# SIMPLEX REACTION CONTROL THRUSTER INTERFACE





## SUMMARY

We have discussed the three most basic approaches to data bus subsystem interface units. Configuration 1 is the simplest and also least flexible. Configuration 3 is by far the most flexible but also the most complex in many ways. Configuration 2 generally falls in between and represents a satisfactory solution to some problems, and a not so satisfactory solution in other instances. We have tried to show some of the engineering decisions necessary to arrive at a final answer. These included data acquisition, message distribution, software implications, and hardware and reliability considerations. A very brief look was taken at redundancy interface problems in the flight control area. And finally, the importance of an integrated power distribution and power control subsystem was discussed.

In conclusion, let me say that it is realized that no definite results or conclusions have been presented by this paper. Our goal was to present a brief overview of an area which does not seem to have a clear-cut solution. The subsystem interface world is one of trade-offs, engineering judgment, reliability, cost, and hardware/software compromises. Yet, at the same time, it is probably the most vital link in the data bus approach to a true integrated avionics system. We can only urge all participants in the space shuttle program to keep in mind the need for cooperation and understanding between subsystem engineers and the need for a strong unbiased systems engineering organization.

## **SUMMARY**

- **THREE BASIC SIU CONFIGURATIONS REVIEWED**
- **NONE OF THE BASIC CONCEPTS SATISFACTORY FOR ALL REQUIREMENTS**
- **FINAL SELECTION MUST INCLUDE BOTH DMS AND SUBSYSTEM OPERATIONAL CRITERIA**
- **SIU MAY BE THE MOST VITAL LINK IN INTEGRATED AVIONICS FUNCTIONAL PATH**

DATA BUS CONCEPTS FOR THE SPACE SHUTTLE

John R. Lane

General Electric Company  
Daytona Beach, Florida

The work which is reported in this paper has been accomplished as part of a joint Grumman Aerospace Corporation/General Electric Company team effort on the Space Shuttle Program. The total scope of this effort was primarily centered about the Data Management Subsystem. While this subject deals with the Data Bus, it is important to underscore the fact that this is but a subset of a more encompassing investigation. Actually, the total study involved the identification of requirements, development of candidate configurations, conduct of trade-offs, and the definition of the baseline Data Management Subsystem.

## DATA BUS ADVANTAGES

Where interfaces are relatively complex and wiring runs are lengthy, use of a data bus can be advantageous. One early study estimated a weight saving of 900 pounds when a bus was employed rather than hardwire in a typical reusable orbiter configuration.

In a large, complex vehicle such as the Shuttle, a high degree of computation centralization for data management functions has been found to be desirable. For a central digital computer to service a diversity of widely dispersed equipments, data must be appropriately converted and transmitted in orderly fashion. The use of a time-shared digital data bus facilitates this data handling process; indeed, it is the bus that tends to make centralization practicable.

The use of a data bus provides flexibility to accommodate design changes and system growth. Extensive additional cabling is not needed when the number of subsystem interfaces is increased. The growth of bus loading is limited by the design data rate and the cable characteristics.

### DATA BUS ADVANTAGES

- LESS WIRING WEIGHT AND COMPLEXITY.
- FACILITATES ORGANIZATION OF COMPUTER AND SUBSYSTEM INTERFACES.
- HAS FLEXIBILITY TO ACCOMMODATE DESIGN CHANGES AND SYSTEM GROWTH.

## BASELINE DATA MANAGEMENT SUBSYSTEM

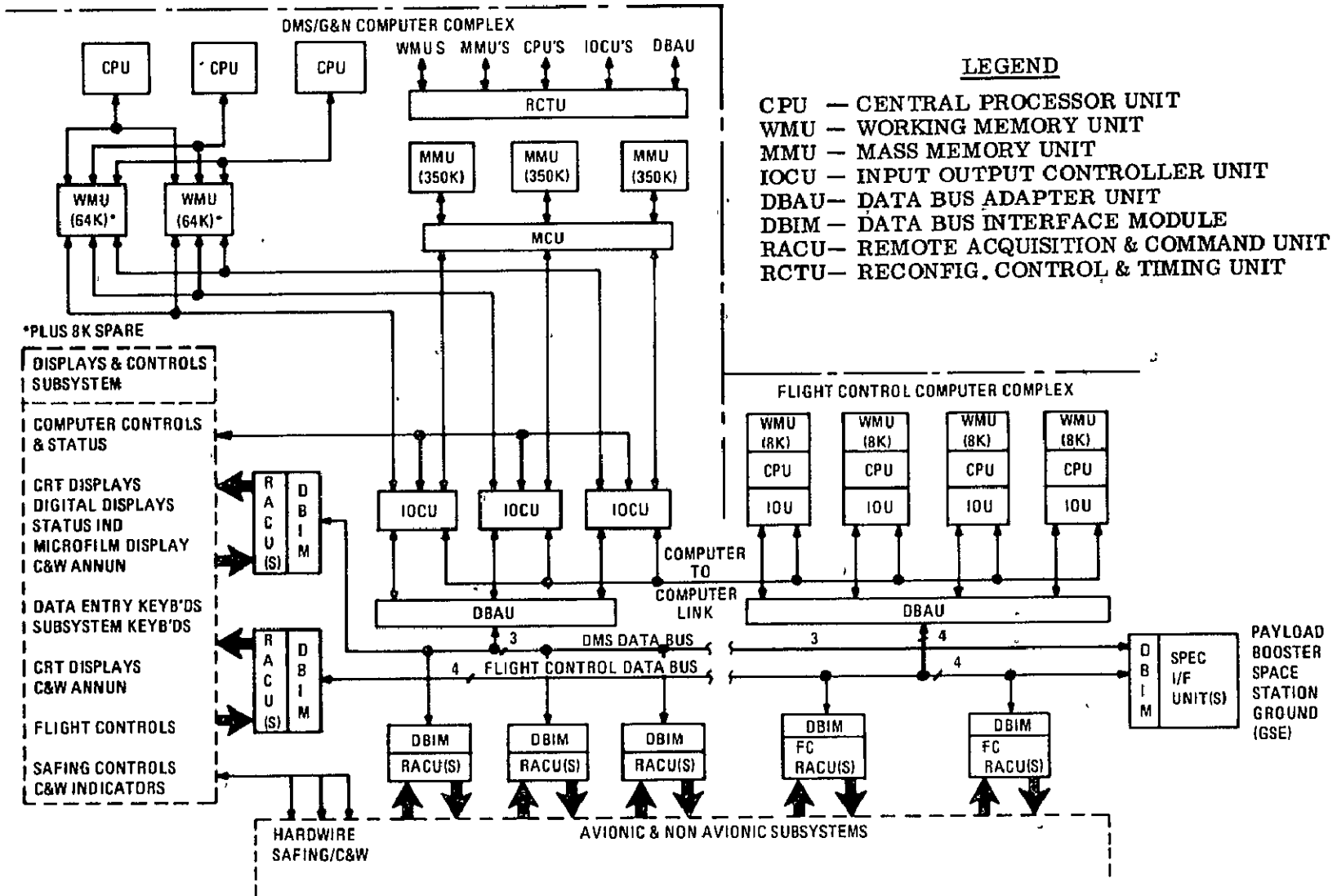
The Data Management Subsystem (DMS) configuration shown, which was discussed in terms of computer organization in the preceding paper, is now considered from a data bus viewpoint.

This data management concept utilizes four uniprocessor computers for Flight Control and three modularized (reconfigurable) computers for all other processing. Note that two separate bus arrangements, with redundancy to match that of the respective computer complex, complement the computer configuration.

Hardwire is employed for critical safing and for certain caution and warning functions - this is a "back-up" provision although these functions are adequately handled by the bus. A direct computer complex-to-computer complex link is shown for exchange of status and certain G&N parameters. In addition, control of DMS operation is maintained via a hardwire link to the DMS/G&N computer complex input/output equipment.

Vehicle subsystems are serviced by "regional" interface units (or RACU's). These units do not perform computation such as limit-checking. Among other functions, the bus interface portion (DBIM) votes on redundant words received via the bus, and outputs to the bus redundant responses acquired from subsystem sensors by the RACU's.

DATA MANAGEMENT SUBSYSTEM BASELINE



75

## DATA BUS REQUIREMENTS AND DESIGN CHARACTERISTICS

This chart summarizes the salient design characteristics for each bus, as well as rationale for each design decision. These choices were made during trade studies, the results of which are to follow.

Data rate estimates were based on analyses of vehicle subsystem requirements. For example, considering approximately 2000 measurements for a typical orbiter, an average DMS sampling rate of three-per-second, and 32-bits per word, a 250 kilobit-per-second rate results for the DMS bus. The selected one megabit design rate provides adequate margin for loading contingencies or growth. At this rate and considering distances of up to 500 feet, simplex operation was chosen to assure positive bus timing and/or operation. Due to the lower Flight Control bus data rate, half-duplex operation is satisfactory - resulting in a reduction of cable (and attendant connections) required.

Laboratory tests have shown that through use of bipolar split-phase code modulation, balanced low-capacitance transmission lines, and differential waveforms, a bit error rate of  $10^{-6}$  can be achieved. When required, however, this rate can be improved via voting or error coding techniques. Critical commands can be verified by echo-checking in the computer before an "execute" command is issued - or when multiple failures have negated the improvement in error rate made possible by word comparisons.

DATA BUS REQUIREMENTS AND DESIGN CHARACTERISTICS

<u>DESIGN CHARACTERISTICS</u>	<u>FLIGHT CONTROL</u>	<u>DMS/ G&amp;N</u>	<u>DRIVING REQUIREMENT</u>
• DATA RATE	150 KBPS	1MBPS	ESTIMATED 68 KBPS FOR FC; 250 KBPS FOR DMS/ G&N
• REDUNDANCY	4 BUSES	3 BUSES	FoFoFs FOR FC; FoFs FOR DMS/G&N
• TRANSMISSION MODE	HALF DUPLEX, COMMON COMMAND AND RESPONSE LINES	SIMPLEX, SEPARATE COMMAND AND RESPONSE LINES	LOWER FC DATA TRANSFER RQT. ALLOWS HALF DUPLEX OPERATION
• MODULATION CODING	BI-POLAR SPLIT PHASE	SAME	RELIABLE BIT SYNC.
• MAX. ONE WAY DISTANCE	500 FEET	SAME	VEHICLE CONFIGURATION
• OPERATIONAL TEMP. RANGE			VEHICLE CONFIGURATION
-CABLE	-175° C TO 290° C (CRYOFILL) (RE-ENTRY)	SAME	
-ELECTRONICS	-54° TO 71° C	SAME	
• BUS WORD LENGTH	~20 BITS	~32 BITS	FC COMPUTERS USE 16 BITS; DMS/ G&N COMP USE 32 BITS.
• MAX. BER W/ O CODING OR VOTING	<10 <sup>-6</sup>	<10 <sup>-6</sup>	RELIABLE DATA TRANSMIS- SION IN HIGH EMI/ NOISE EN- VIRONMENT WITH MINIMUM HARDWARE COMPLEXITY



## DATA BUS TRADE STUDY SUMMARY

This is a matrix of the trade studies performed with resulting selections indicated. The location of the check mark in Trade Study 4 implies a tie which must be resolved by further analysis.

In the Computer Subsystem Link Configuration Study, Alternative B was selected; hardware is necessary for initialization, safing, and back-up of certain critical functions.

The Bus Partitioning Study considered a single multipurpose data bus versus the use of a data bus for Flight Control and separate bus for DMS/G&N functions. The two-bus configuration was chosen over the single bus primarily to provide higher reliability for the critical functions served by the Flight Control Bus. A simpler bus design results from less extensive cabling, fewer terminals, and lower data rates. This choice also matches the chosen computer configuration and simplifies the computer interface with the bus.

The Bus Redundancy Study resulted in selection of Alternative B because of lower cost and development risk.

The Bus Control/Operation Study clearly indicated that C was undesirable due to DBIM complexity resulting in lower MTBF, higher cost, and higher development risk. However, a clear choice between A or B was not indicated. The final design will, in fact, probably be intermediate between A and B.

In the Bus Voting Study, A was selected since this technique offers a wider range of reconfiguration options. Configuration alternatives covered in this study will be described in the bus voting alternatives slide.

DATA BUS TRADE STUDY SUMMARY

<u>TRADE STUDY</u>	<u>ALTERNATIVES</u>
1. COMPUTER-SUBSYSTEM LINK CONFIGURATION	A. BUS ONLY. √B. BUS WITH HARDWARE FOR CRIT. SAFING AND C&W. C. HARDWARE ONLY.
2. BUS PARTITIONING	A. SINGLE BUS FOR FC AND DMS/G&N. √B. SEPARATE BUSES FOR FC AND DMS/G&N.
3. BUS REDUNDANCY	A. SINGLE FoFoFs BUS FOR FC AND DMS/G&N. √B. FoFoFs BUS FOR FC; FoFs FOR DMS/G&N. C. FoFs BUS FOR DMS/G&N; HARDWARE FOR FC.
4. BUS CONTROL/ OPERATION	√A. MAX. COMPUTER INTERACTION. √B. MAX. BUS CONTROLLER INTERACTION. C. MAX. SS INTERFACE INTERACTION.
5. BUS VOTING	√A. VOTE ON RESPONSES IN CPU AND VOTE AT SUBSYSTEM. B. VOTE ON RESPONSES IN CPU; NO VOTE AT SUBSYSTEM.

## BUS VOTING ALTERNATIVES

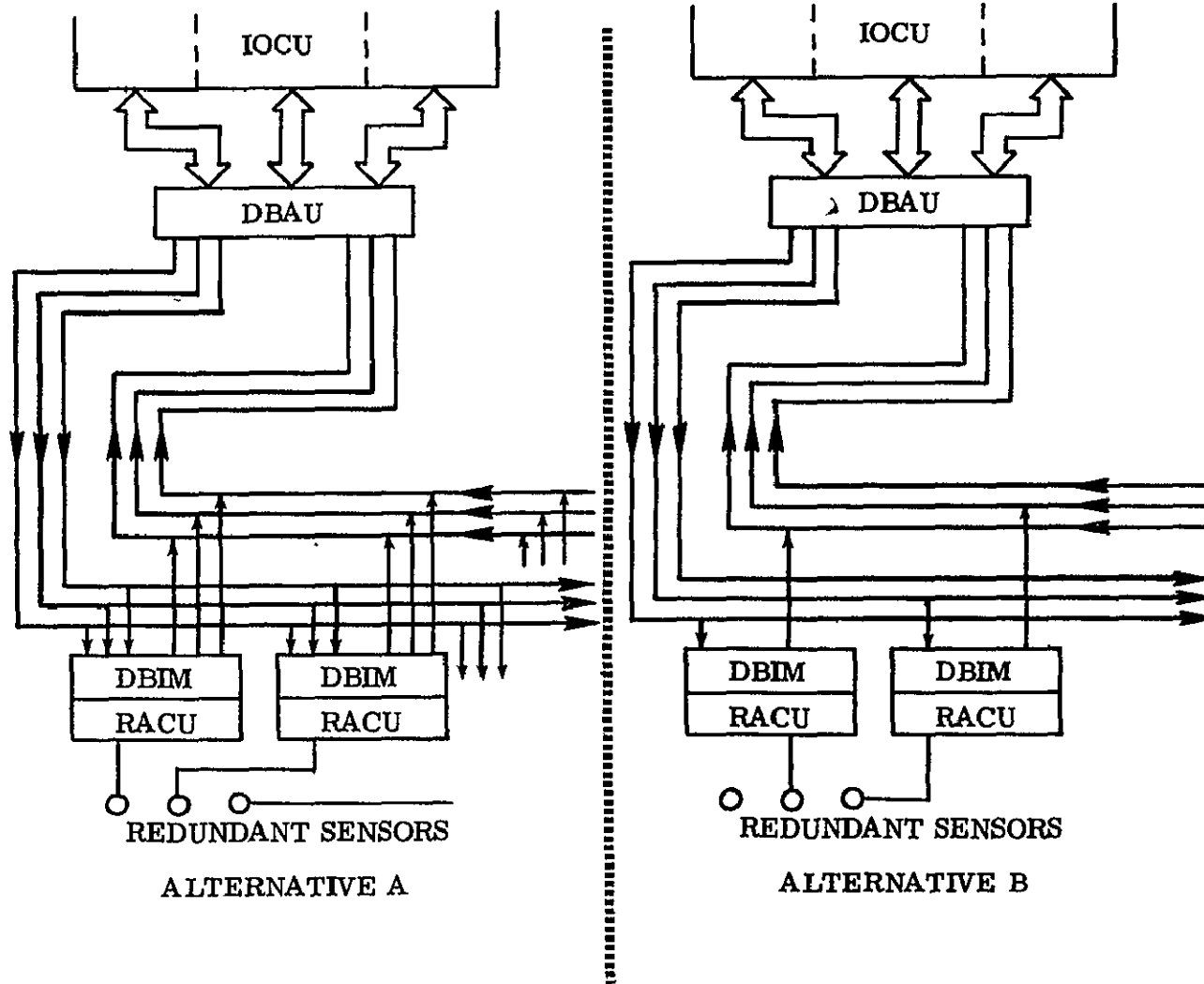
Two voting alternatives evaluated during the trade-off studies are depicted in this chart. Although only the DMS/G&N bus configuration is represented, with appropriate redundancy considerations the figures also apply to Flight Control.

In Alternative A each subsystem interface unit (DBIM/RACU combination) is connected to all three sets of buses, and voting (on commands/addresses) is accomplished by the DBIM. In Alternative B each unit is serviced by only one set of buses in a single-thread computer-to-bus-to-subsystem interface configuration, thus, no voting is performed at the DBIM level. In both A and B, however, responses from redundant sensors are compared in all three computers. Also, the DBAU votes on computer outputs such that only error-free words are issued to the buses at this interface.

The response from each redundant sensor is transferred as three identical concurrent words in Alternative A, thus, three sequential transmissions are required to read all three sensors. In B, fewer messages are transferred in acquiring a given subsystem parameter.

Fail-operational/fail-safe (Fo/Fs) operation is achieved with both alternatives. Although A appears potentially more reliable, the attendant voting circuit complexity may in fact degrade reliability; Fo/Fs must be met consistent with the MTBF required. In Alternative B the redundancy of the sensors and interface units must be carefully assigned according to criticality, since the loss of a bus results in loss of all attached interface units. On the basis of these considerations, Alternative A was selected as the baseline.

BUS VOTING ALTERNATIVES  
(DMS/G&N)

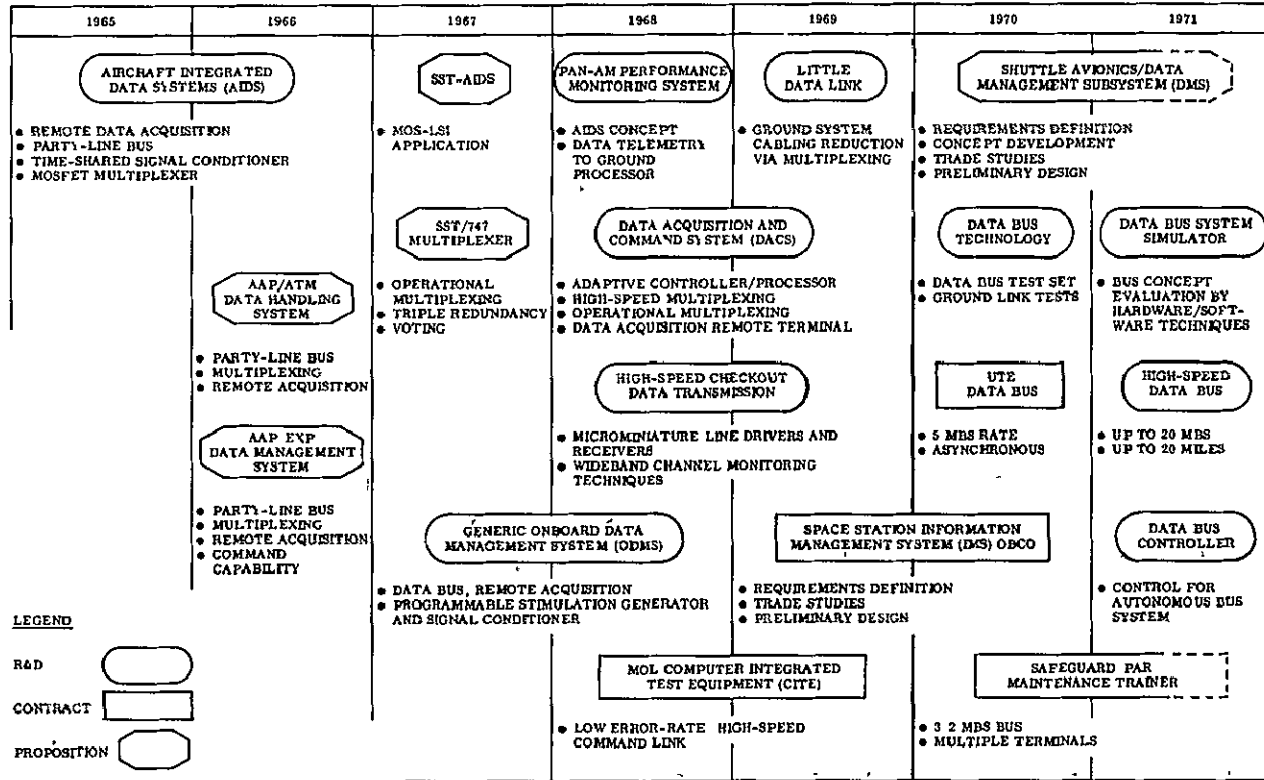


## DATA BUS TECHNOLOGY EVOLUTION AT GE-AGS

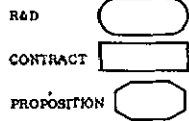
Apollo Program experience with the problem of transmitting vast quantities of data over considerable distances led to extensive IR&D, proposition, and contract work by GE-Apollo and Ground Systems directed toward advantageous utilization of multiplexing and data bus techniques. These techniques have been applied in aircraft integrated data systems, aircraft operational multiplexing systems, and spacecraft onboard checkout/data management systems, as well as ground checkout systems conceived, developed, and/or fabricated by GE-Apollo and Ground Systems since 1965.

Data bus-related IR&D and proposition activities in 1970-71 have been chiefly directed toward evaluation of techniques in digital data transmission within the Space Station and Space Shuttle. In addition, consideration has been given to the ground interface with the onboard bus. An existing ground data link has also been evaluated experimentally for potential Shuttle usage.

DATA BUS TECHNOLOGY EVOLUTION AT GE-AGS



LEGEND



## CONCLUSIONS

The Shuttle data bus design can be mechanized using present-day technology. The data rates and distance requirements are not excessive; one-to-two megabit buses up to 500 feet in length operating at error rates of less than  $10^{-6}$  (without coding) can be implemented with standard baseband transmission techniques.

FoFoFs or FoFs operation is achievable, however, MTBF requirements must be met concurrently. Straightforward quadruple or triple redundancy with multiple cross-switching can result in excessive hardware - perhaps to the extent that the MTBF is degraded. Any proposed bus (or computer design) must be validated for both of these requirements. In the end, less complex designs may stand a better chance of meeting both the FoFoFs/FoFs and MTBF requirements; however, more detailed analyses of subsystems served are required.

### CONCLUSIONS

- THE DATA BUS CAN BE MECHANIZED WITH PRESENT-DAY TECHNOLOGY.
- FoFoFs/FoFs AND MTBF REQUIREMENTS MUST BE MET CONCURRENTLY.



# DATA BUS TO SERVICED EQUIPMENT AND SENSOR INTERFACE

Berry S. Yolken

TRW SYSTEMS

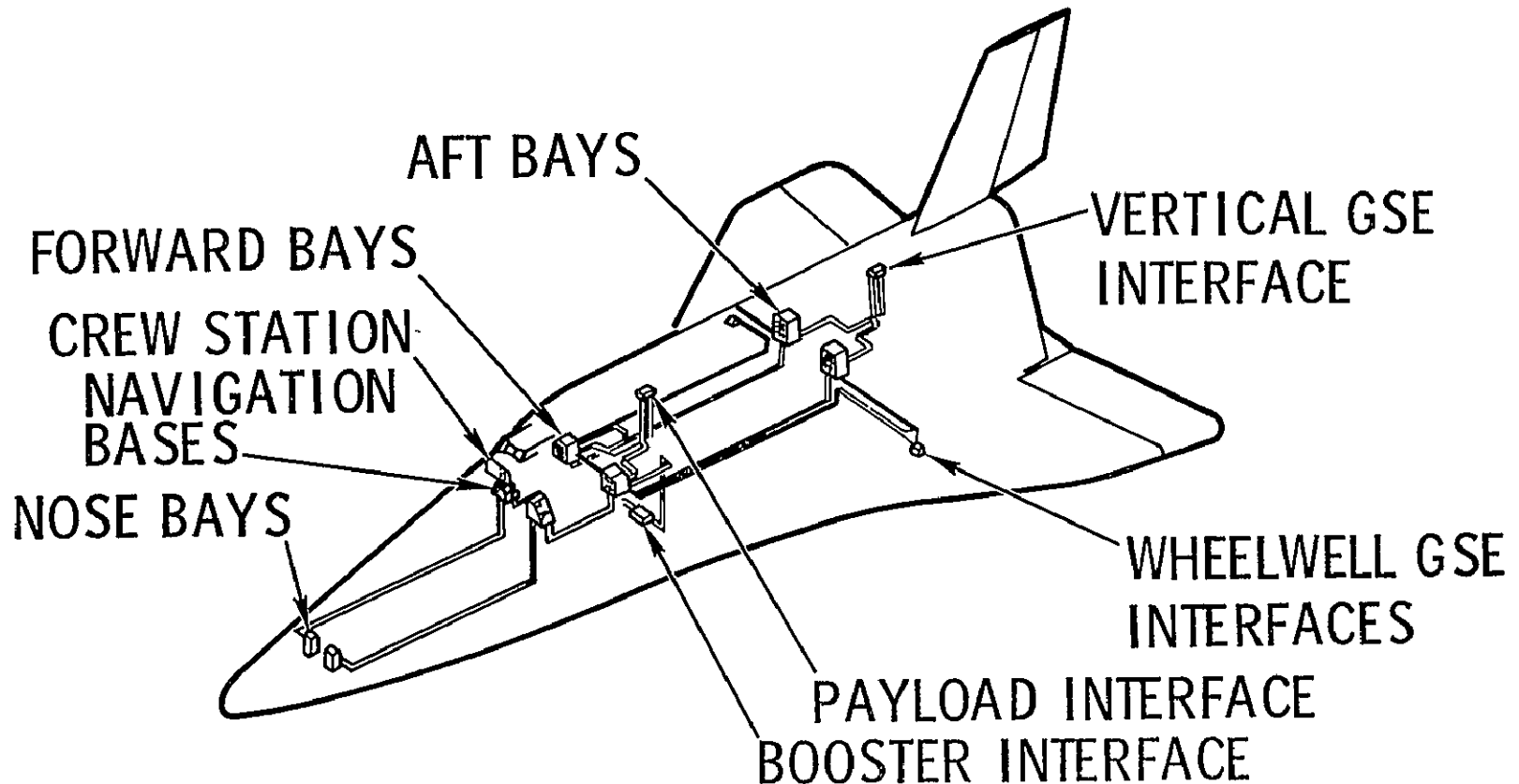
REDONDO BEACH, CALIFORNIA

N71-35052

PRECEDING PAGE BLANK NOT FILMED

THE DATA COLLECTION AND DISTRIBUTION SYSTEM FOR THE MC DONNELL DOUGLAS SHUTTLE ORBITER IS LOCATED WITHIN ENVIRONMENTALLY PROTECTED FUSELAGE EQUIPMENT BAYS TO AVOID THE SEVERE TEMPERATURES ENCOUNTERED DURING RE-ENTRY.

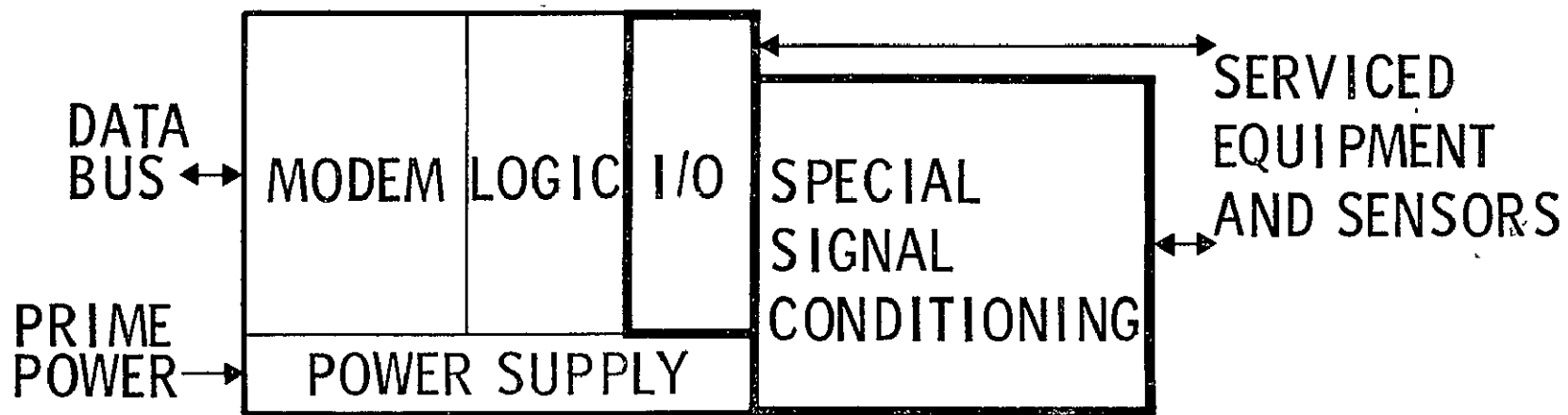
# DATA COLLECTION AND DISTRIBUTION AREAS - MC DONNELL DOUGLAS SHUTTLE ORBITER



A SIGNIFICANT PORTION OF THE TERMINAL DESIGN IS PREDICATED UPON THE EXTENT AND TYPE OF SIGNAL CONDITIONING INCLUDED AS PART OF THE REMOTE TERMINAL. IT APPEARS ADVANTAGEOUS TO INCLUDE SENSOR SIGNAL CONDITIONING WITHIN THE TERMINAL SINCE THE TERMINAL CONTAINS REGULATED POWER, HERMETIC CHASSIS, COLD PLATE CONNECTIONS, AND THE BASIC TIMING AND CONTROL.

THIS PRESENTATION WILL ONLY CONSIDER THE INTERFACE FROM THE TERMINAL TO THE SERVICED EQUIPMENT.

# DATA BUS TERMINAL



THE DATA BUS TECHNIQUE BY ITS VERY NATURE ENCOURAGES STANDARDIZATION OF BOTH SOFTWARE AND HARDWARE THROUGHOUT THE DATA MANAGEMENT SYSTEM. THERE IS, HOWEVER, ONE AREA WHERE STANDARDIZATION MAY BE DIFFICULT; THIS IS THE INTERFACE BETWEEN THE DATA BUS TERMINALS AND THE SERVICED EQUIPMENT. WE MUST CONTEND WITH A MULTITUDE OF NON-STANDARDIZED SIGNALS, GROUND DISCONTINUITIES, ASYNCHRONOUS DATA TRANSFER, AND METHODS OF PHYSICAL CONNECTION. THERE ARE TECHNIQUES WHICH HAVE BEEN UTILIZED IN THE TELEMETRY FIELD TO HANDLE MANY OF THESE PROBLEMS. IN ADDITION, THERE ARE COMPONENTS AND TECHNIQUES AVAILABLE TO THE DESIGNER TODAY.

# INTERFACE CONSIDERATIONS

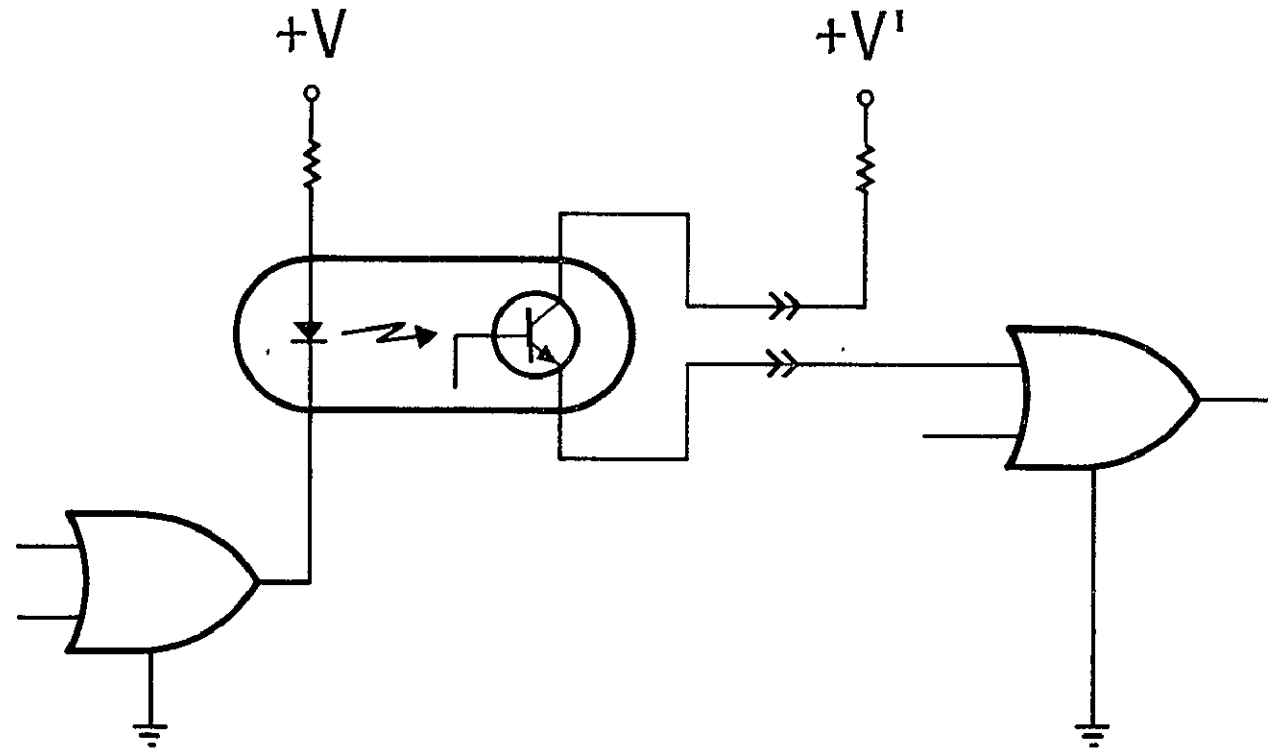
- GROUND DISCONTINUITIES
- ASYNCHRONOUS DATA TRANSFER
- ANALOG CONVERSION
- DATA MULTIPLEXING
- SENSOR SIGNAL CONDITIONING

GROUND LOOPS AND ATTENDANT NOISE CAN BE CIRCUMVENTED WITH PHOTO COUPLERS FOR DIGITAL DATA TRANSFER. THESE PHOTO COUPLERS PROVIDE ISOLATION GREATER THAN  $10^{12}$  OHMS. THE DEVICE CONSISTS OF A GALLIUM ARSENIDE LIGHT-EMITTING DIODE OPTICALLY COUPLED TO A PHOTSENSITIVE SEMICONDUCTOR (USUALLY SILICON). IN OPERATION, THE GaAs DIODE EMITS INFRARED RADIATION WHEN FORWARD BIASED. THE LIGHT GENERATES CURRENT FLOW WITHIN THE RECEIVING SEMICONDUCTOR JUNCTION. THE RECEIVER MAY BE EITHER A TRANSISTOR WITH OPEN BASE OR A DIODE.

THE CURRENT GAIN IS APPROXIMATELY ONE FOR THE PHOTOTRANSISTOR TYPES AND APPROXIMATELY 0.002 FOR THE PHOTO DIODE TYPE. THE HIGHER GAIN IN THE TRANSISTOR DEVICE IS ACHIEVED AT A SACRIFICE IN SWITCHING SPEED. EACH TYPE IS CAPABLE OF INTERFACING DIRECTLY WITH TTL.



# ISOLATION WITH PHOTO COUPLERS



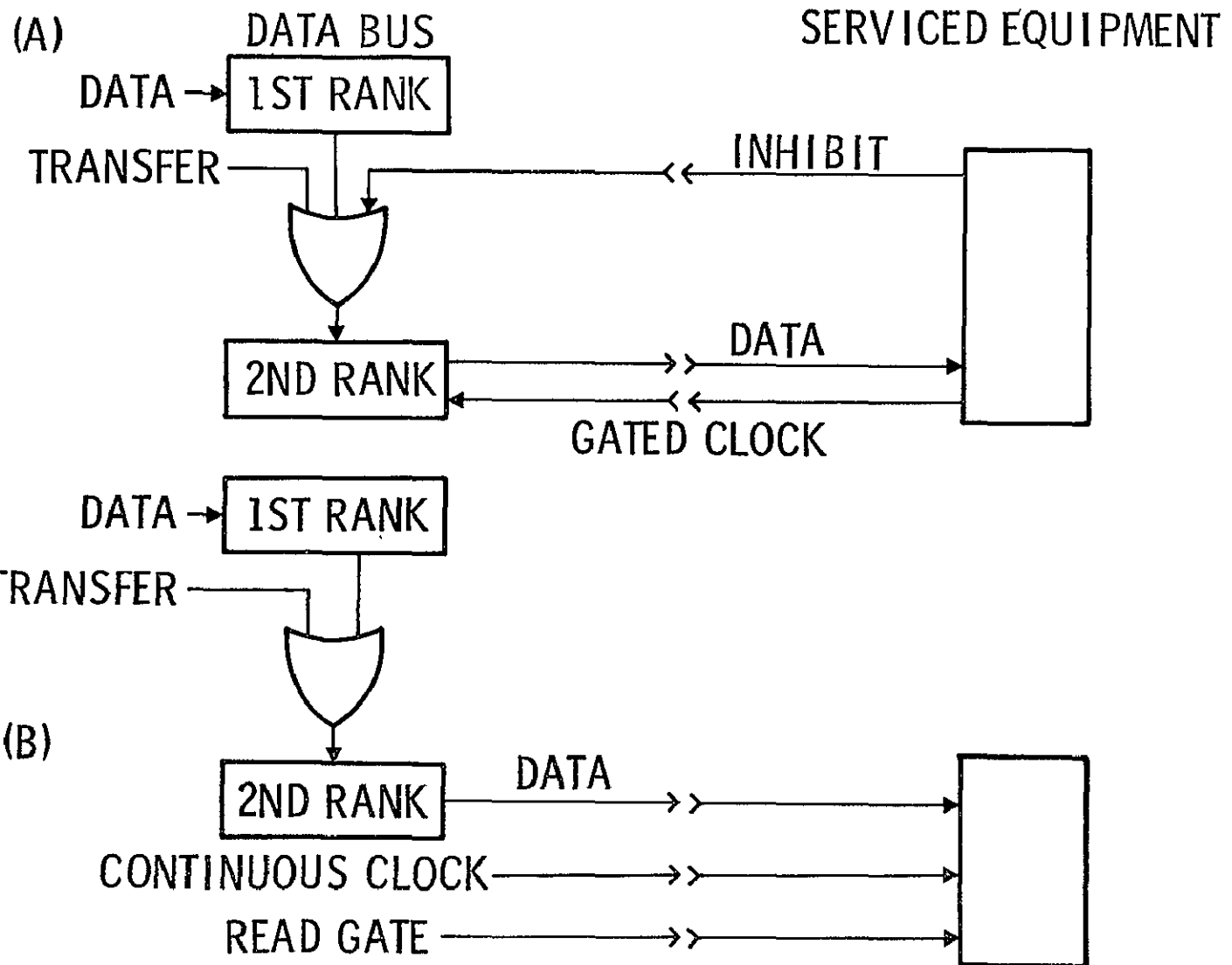
95

DATA BUS TERMINAL

SERVICED EQUIPMENT

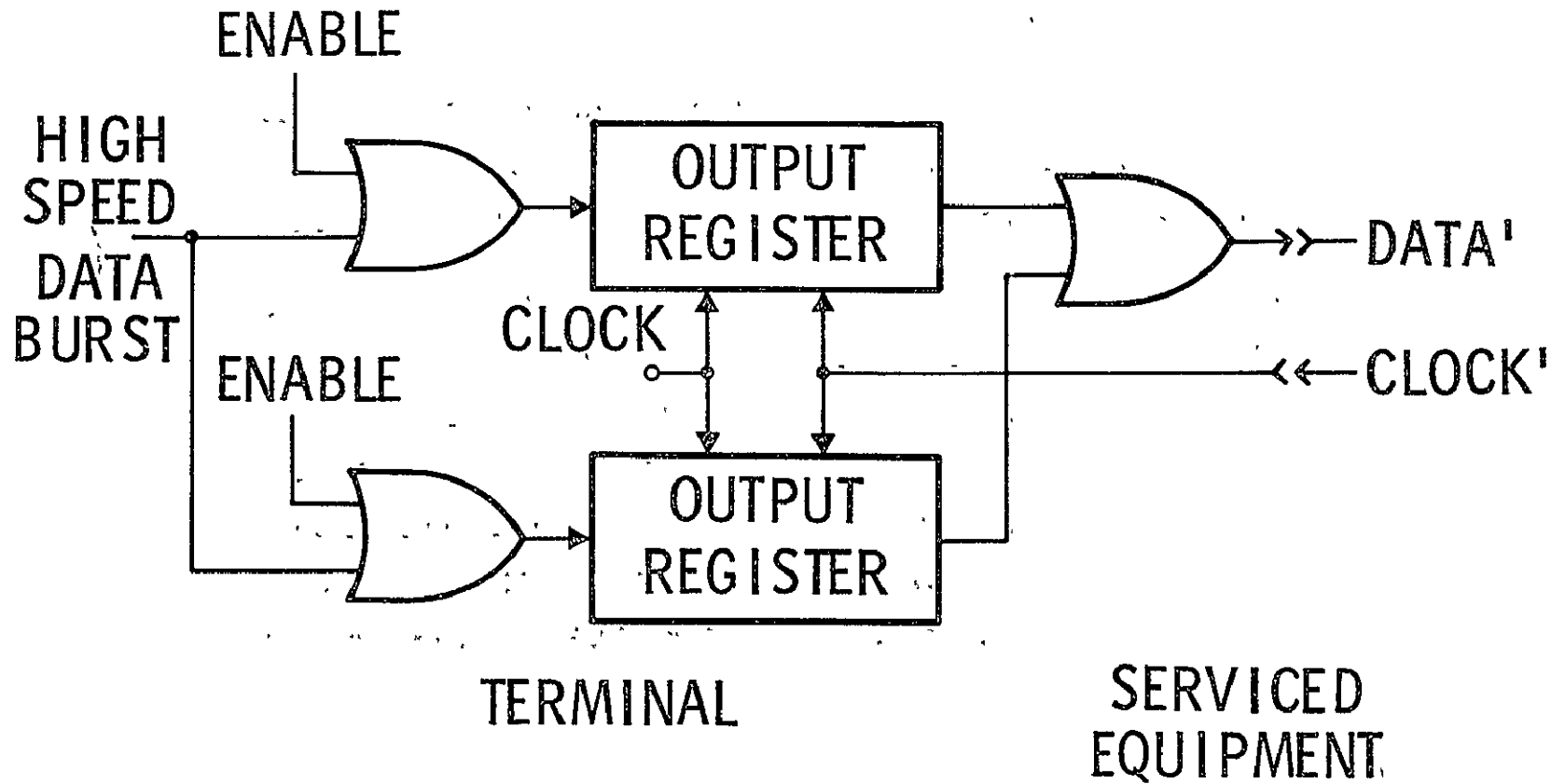
TWO METHODS FOR ASYNCHRONOUS DATA TRANSFER ARE SHOWN. THE FIRST IS WHERE THE SERVICED EQUIPMENT HAS CONTROL OF THE TERMINAL OUTPUT REGISTER BUT "HANDSHAKING" IS EMPLOYED, AND THE SECOND IS WHERE THE TERMINAL HAS CONTROL. A TYPICAL APPLICATION IS THE TRANSFER OF VECTOR INFORMATION BETWEEN AN INERTIAL PLATFORM AND THE TERMINAL.

# ASYNCHRONOUS DATA TRANSFER



THIS TECHNIQUE CAN BE USED FOR BUFFERING BURST HIGH RATE DATA INTO CONTINUOUS LOW RATE FORM. TYPICAL APPLICATION IS THE SELECTION OF SPECIFIC PARAMETERS FROM A 1 MBPS DATA BUS FOR SUBSEQUENT TRANSMISSION ON A SLOWER TELEMETRY LINK. REGISTERS IN EXCESS OF 1000 BITS ARE AVAILABLE WITH THE ADVENT OF LARGE SCALE INTEGRATION.

# HIGH SPEED TO LOW SPEED



THERE ARE MANY A/D TECHNIQUES USED FOR SPACECRAFT AND AIRBORNE APPLICATIONS INCLUDING RAMP TYPE AND SUCCESSIVE APPROXIMATION. SUCCESSIVE APPROXIMATION IS THE MOST POPULAR TECHNIQUE IN USE TODAY BECAUSE IT EMPLOYS DIGITAL TECHNIQUES WHICH ARE LESS SUBJECT TO DRIFT AND ARE MORE AMENABLE TO MICROMINIATURE TECHNIQUES. SIX TO 10 BIT CONVERTERS ARE AVAILABLE WHICH OPERATE AT 1 MBPS. THIS MAKES REAL TIME CONVERSION POSSIBLE.

EIGHT BIT D/A CONVERTERS ARE AVAILABLE ON ONE MONOLITHIC CHIP. THIS ALLOWS THE USE OF ONE D/A FOR EACH OUTPUT, THUS PRECLUDING THE NEED FOR DEMULTIPLEXERS AND SAMPLE AND HOLD CIRCUITS.

# ANALOG CONVERSION CONSIDERATIONS

## ANALOG TO DIGITAL

- ACCURACY AND RESOLUTION
- SAMPLING RATE
- REAL TIME VERSUS ENCODE AND STORE
- SAMPLE AND HOLD
- POWER GATING

## DIGITAL TO ANALOG

- ONE CONVERTER PER OUTPUT
- PRECLUDES NECESSITY FOR DEMULTIPLEXER AND SAMPLE AND HOLD

A MAJOR FUNCTION OF THE TERMINAL IS THE SAMPLING OF LARGE NUMBERS OF ANALOG AND DIGITAL DATA SOURCES IN SOME DESIRED SEQUENCE. TYPICALLY A MULTIPLEXER IS USED TO PERFORM THIS FUNCTION. IT CONSISTS OF A LARGE NUMBER OF SWITCHES ORGANIZED TO CONNECT MANY DATA SOURCES IN A DESIRED SEQUENCE TO A SINGLE OUTPUT.

DIAGRAM (A) ILLUSTRATES THE ANALOG MULTIPLEXER CONTAINING MOS FET OR J-FET TRANSISTOR SWITCHES. THIS TYPE MULTIPLEXER CAN ALSO ACCOMMODATE DIGITAL DATA.

DIAGRAM-(B) ILLUSTRATES A MULTIPLEXER TECHNIQUE USED FOR DIGITAL DATA ONLY.



# DATA MULTIPLEXER

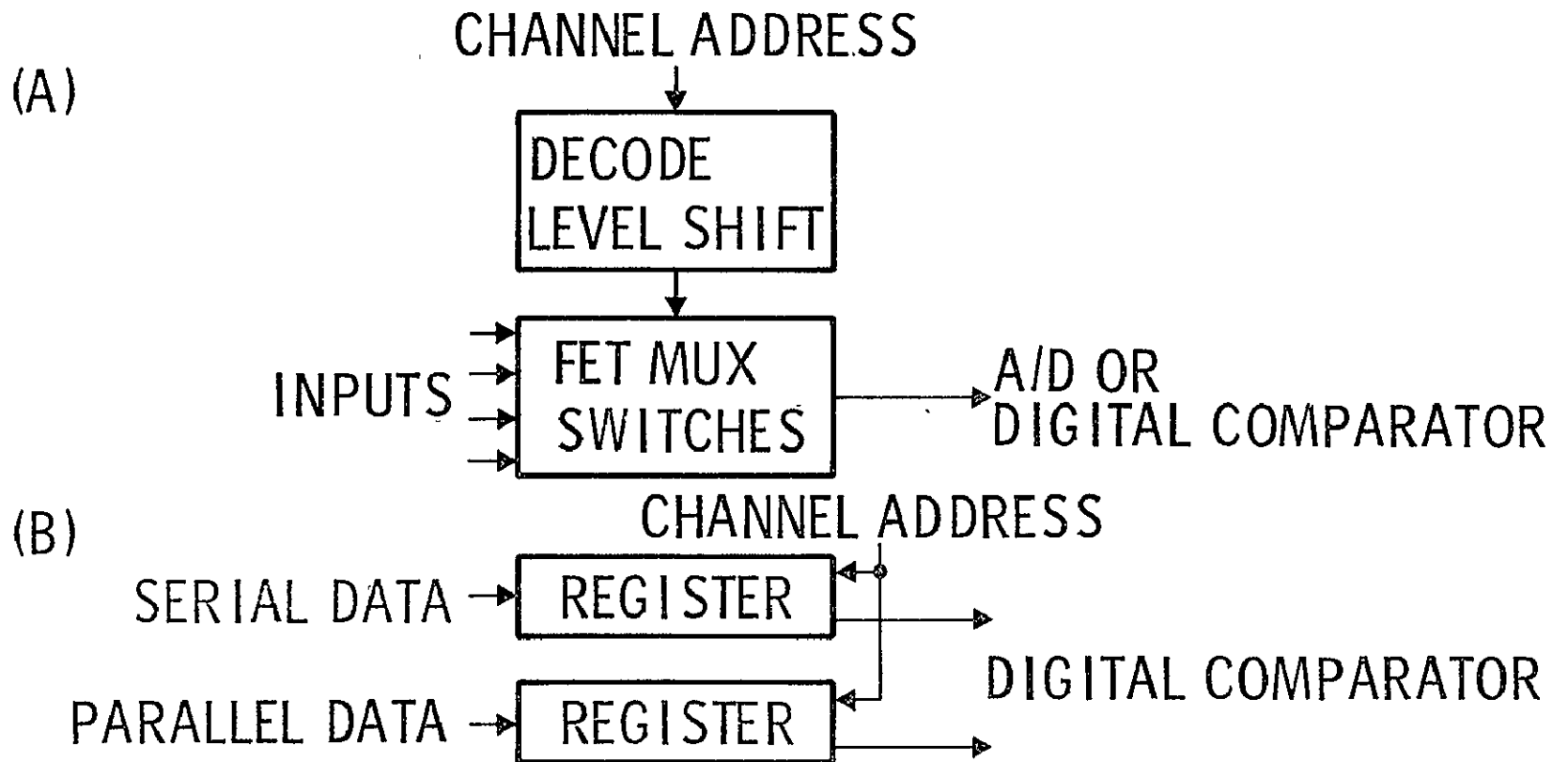


DIAGRAM (A) SHOWS A TECHNIQUE FOR CONDITIONING A SENSOR (TYPICALLY A PRESSURE OR LOW TEMPERATURE). THE SENSOR ELEMENT IS A COMPLETE BRIDGE AND IS ISOLATED FROM THE VEHICLE GROUND. THE DIFFERENTIAL AMPLIFIER PERFORMS TWO BASIC FUNCTIONS, AMPLIFICATION AND LOW PASS FILTERING.

DIAGRAM (B) SHOWS A TECHNIQUE FOR CONDITIONING A SENSOR SIGNAL IN ITS LOW LEVEL FORM. THE ONLY CONDITIONING ELEMENT PROVIDED BY THE SIGNAL CONDITIONER IS A LOW PASS FILTER. THE OUTPUT WILL BE A SIGNAL WHICH WILL VARY PLUS AND MINUS ABOUT THE GROUND AND REQUIRES A MORE COMPLEX MULTIPLEXER AND A/D CONVERTER.

# BRIDGE SENSOR SIGNAL CONDITIONING

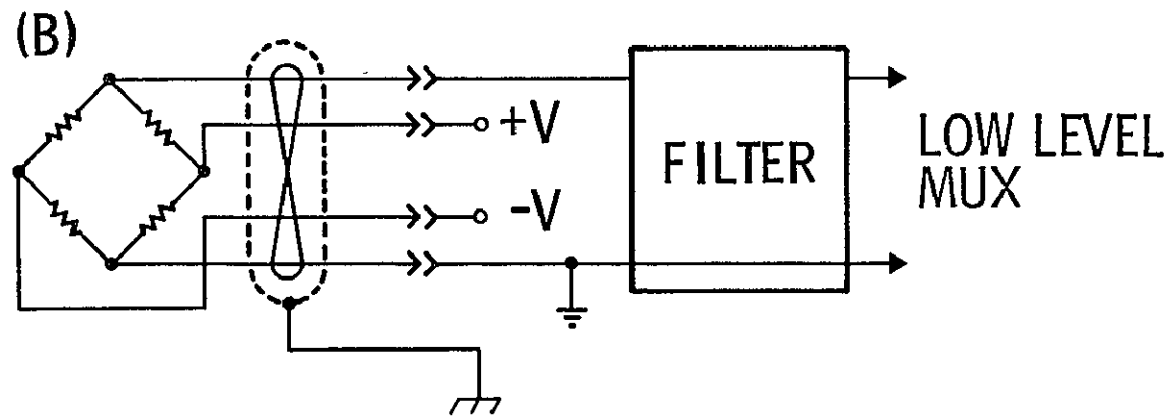
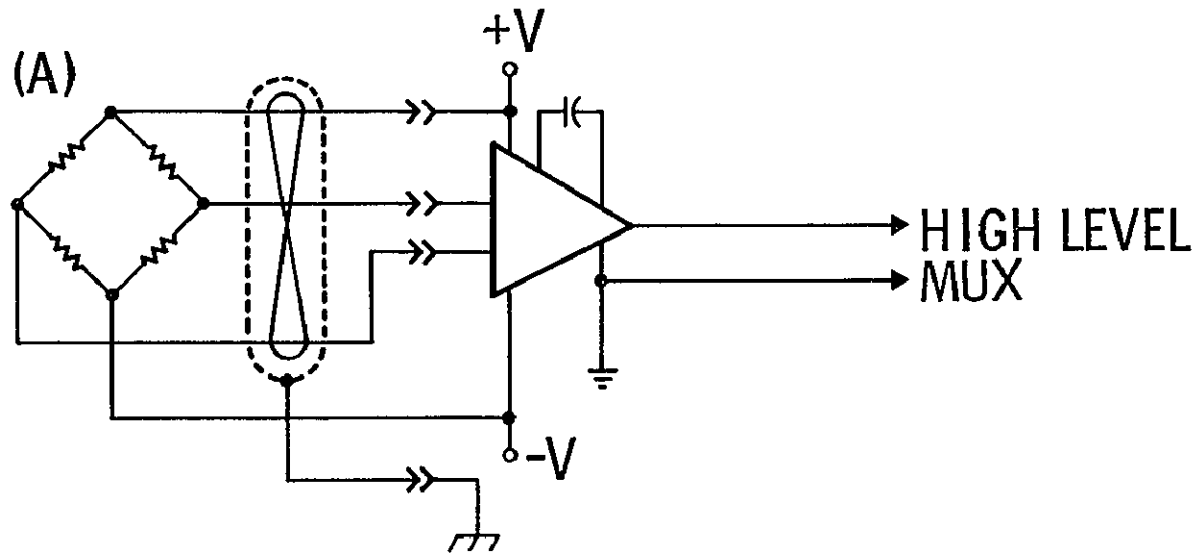


DIAGRAM (A) SHOWS A TECHNIQUE FOR CONDITIONING A SENSOR (TYPICALLY A PLATINUM ELEMENT THERMAL SENSOR OR A RESISTIVE ELEMENT STRAIN GAGE). THE SENSOR ELEMENT IS ISOLATED FROM THE VEHICLE GROUND. THE BRIDGE COMPLETION NETWORK IS LOCATED WITHIN THE SIGNAL CONDITIONER. THE DIFFERENTIAL AMPLIFIER PERFORMS TWO BASIC FUNCTIONS: AMPLIFICATION AND LOW PASS FILTERING.

DIAGRAM (B) SHOWS A TECHNIQUE FOR CONDITIONING A SENSOR SIGNAL IN ITS LOW LEVEL FORM. A LOW PASS FILTER IS PROVIDED TO REMOVE NOISE. THE SINGLE ENDED OUTPUT WILL VARY PLUS AND MINUS AROUND THE SIGNAL CONDITIONER GROUND AND WILL REQUIRE A MORE COMPLEX MULTIPLEXER AND A/D CONVERTER.

FOR BOTH METHODS, A THIRD WIRE IS PROVIDED TO CANCEL THE WIRING THERMAL DELTAS.

# SINGLE LEG SENSORS

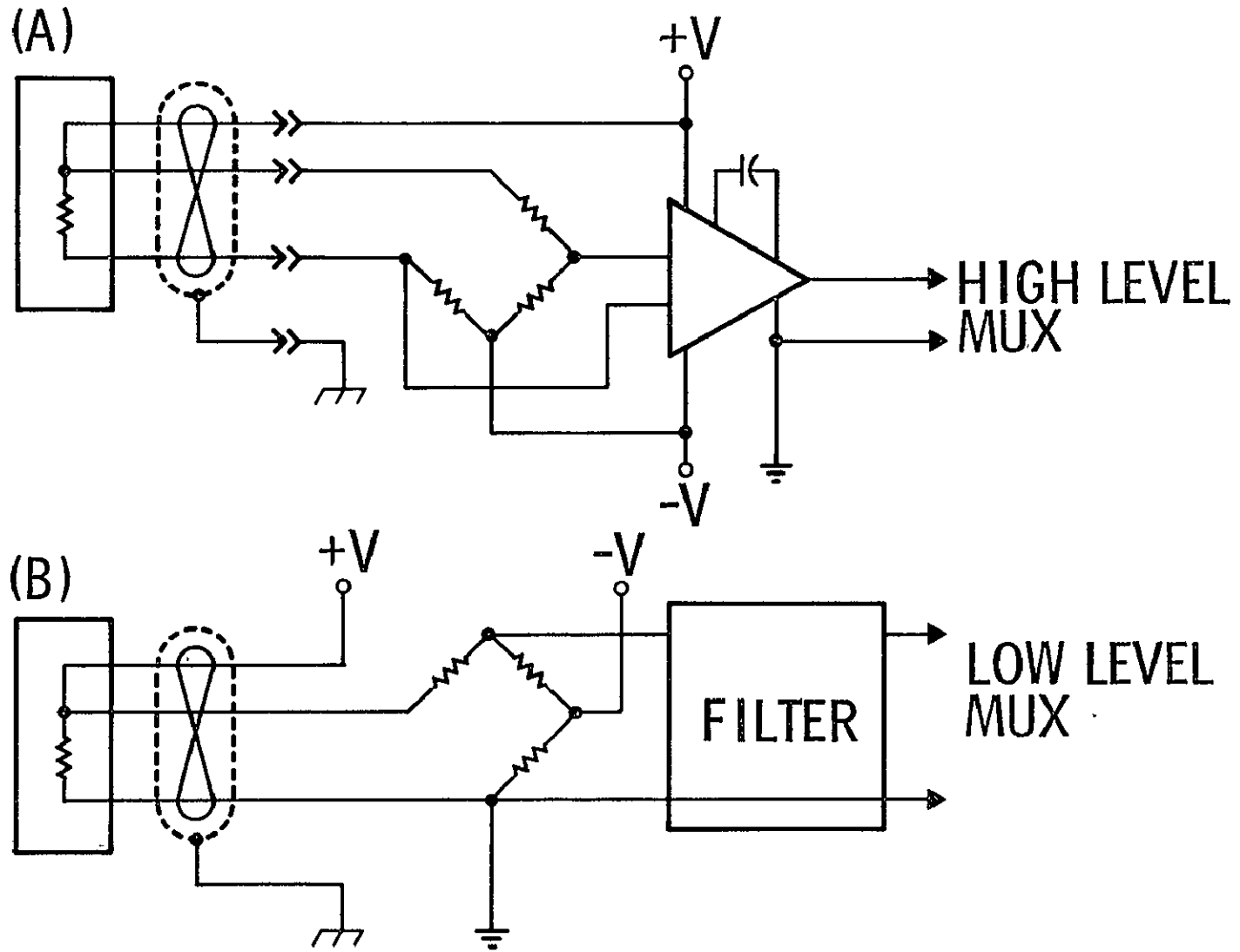
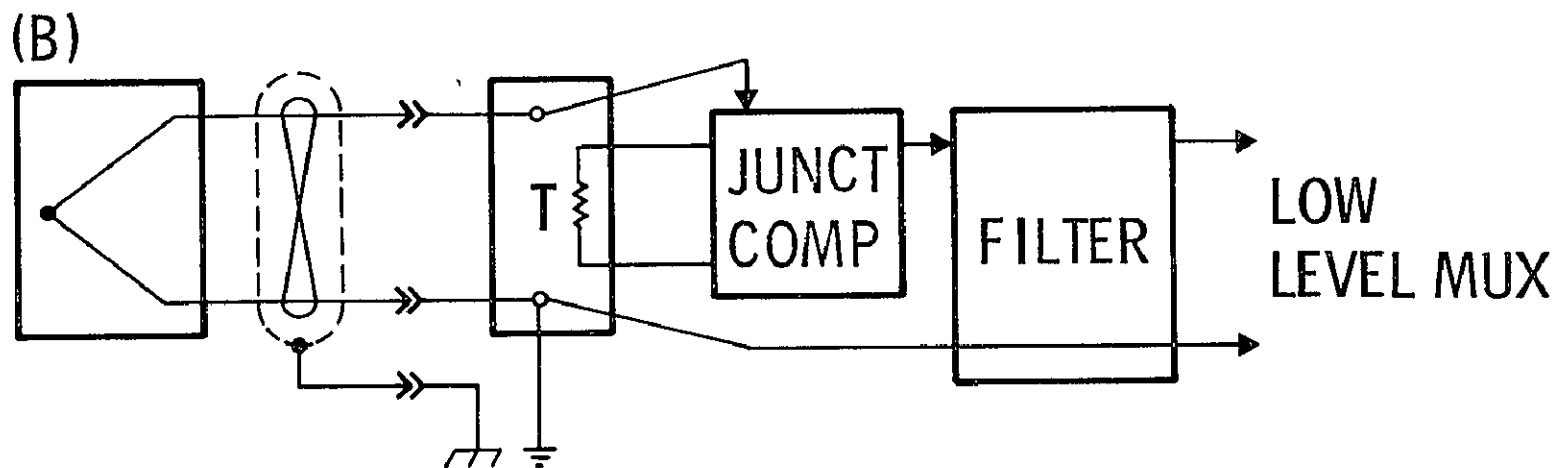
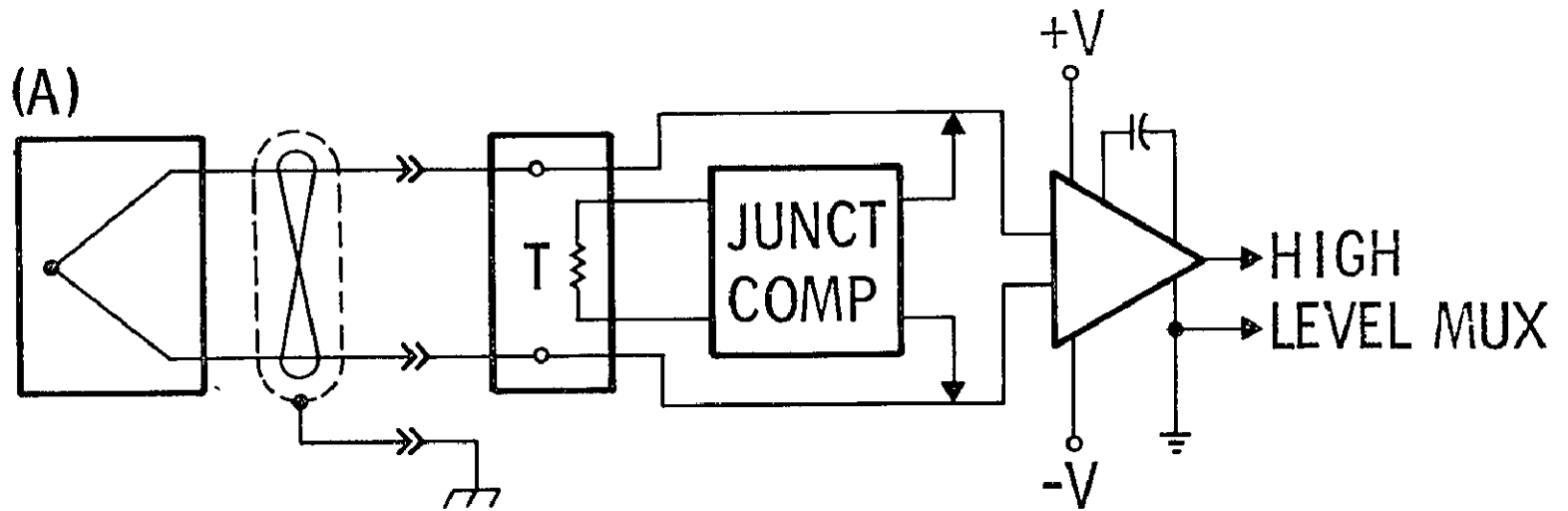


DIAGRAM (A) SHOWS A TECHNIQUE FOR CONDITIONING A THERMOCOUPLE OUTPUT SIGNAL DIFFERENTIALLY. THE THERMOCOUPLE IS ISOLATED FROM THE VEHICLE GROUND. THE SIGNAL CONDITIONER INPUT JUNCTION PROVIDES AMBIENT JUNCTION SENSING TO CORRECT THE OFFSET ERROR. AN AMPLIFIER IS EMPLOYED IN EACH CHANNEL TO PRECLUDE THE NECESSITY FOR A LOW LEVEL MULTIPLEXER.

DIAGRAM (B) SHOWS A TECHNIQUE FOR CONDITIONING A THERMOCOUPLE OUTPUT IN ITS LOW LEVEL FORM. A JUNCTION COMPENSATION NETWORK IS REQUIRED AT THE TERMINAL INPUT. THE THERMOCOUPLE OUTPUT IS THEN LOW PASS FILTERED TO REMOVE NOISE.

# THERMOCOUPLE SIGNAL CONDITIONING



## SELF-SEQUENCING DATA BUS TECHNIQUES FOR SPACE SHUTTLE

Robert H. Hardin

Martin Marietta Corporation  
Denver, Colorado

Data bus techniques which have grown out of remote multiplexed data acquisition and telemetry systems have several potential limitations in their application to the Space Shuttle. Because these systems are under the control of a central "programmer" or "controller", the whole system is vulnerable to failures in this central unit. Depending on design details, this central control approach may also require that the rate of data processing functions in the remote units, such as A/D conversion, be equivalent to the data transfer rate on the bus itself.

Data bus designs which distribute control among the remote units are called "self-sequencing" and can overcome these disadvantages. One self-sequencing technique will be examined in detail in this paper and will be compared to the centralized control approach. Reliability, redundancy management, and flexibility of operation will be the principal comparison criteria. Two other self-sequencing alternatives are discussed briefly.

The self-sequencing technique considered in detail is called Slot Assigned TDM, or SLAT. The only central control operation required by this technique is the emission of a data cycle synchronization word which is sent to all remote units. Each remote unit is assigned a time slot during which it will transfer data to the data bus. Using the sync word as a timing reference, each remote unit determines when to place data on the bus by using a simple internal clock and down-counter. Each remote unit then places data, and the corresponding destination address for each data word, on the data bus. Clocking for the data is always determined by and sent by the sending unit, so propagation delay problems are eliminated. Data acquisition, A/D conversion, and data processing within each remote unit can proceed continuously at a low rate; data can be buffer stored within each unit; and the transfer to the data bus can be done at a very high rate. The generation of the data cycle synchronization word can be a function of one of the remote units with back-up in other remote units in case of failure.



## I. Primary Requirements of A Data Bus System

The requirements for the data bus system for the shuttle have not yet been firmly determined by the Phase B studies and other program activities. To some extent, the determination of these requirements will be an iterative process, and changes will occur as other system features are determined. For the purposes of this paper, the primary requirements shown in Figure 1 are assumed. These requirements are independent of the data bus design features. Others are included (the secondary requirements) to force consideration of the salient features of self-sequencing data bus systems with decentralized control.

The fundamental requirement of any data bus system is that it should transfer a very large number of signals from signal sources to corresponding signal sinks on a very small number of cables, to eliminate what has recently been referred to as "wire pollution" (1). If this basic requirement is considered primarily as a replacement of point-to-point wiring to interconnect various elements of the total systems, another basic requirement is immediately evolved. This is the requirement for "device-to-device" data (or signal) transfer. Note that this basic requirement includes device-to-computer and computer-to-device data transfer, but the computer is given no special consideration in this basic requirement definition. The computer is treated as any other device here, and by viewing it this way, several system simplifications may be brought about.

Another basic requirement for multiplexed and sampled data transfer must be considered. Whenever a sampled data system is included in a real-time control loop, the required data samples must be transferred from source to sink on a continuing, periodic basis. Since guidance and flight control functions on the shuttle are included, the requirement for periodic data transfer is established. Many other data handling functions, such as display, status monitoring, and flight records stored for post-flight analysis, require non-periodic data transfer on an "as-required" basis. Thus, intermixed periodic and non-periodic data transfers from device-to-device are established as basic requirements.

Random access data acquisition is included to provide the in-flight flexibility to acquire data in accordance with real-time flight conditions which cannot be foreseen and pre-programmed. The capability to priority interrupt remote units by the central Data Management System (DMS) should also be included.

A quad-redundant bus system is assumed as a basic requirement as a minimum level of redundancy to meet the system requirement to fail safe after three failures.

(1) J. Ohlhaber, "A Solution To Wire Pollution", Signal, April, 1971

## I. PRIMARY REQUIREMENTS OF A DATA BUS SYSTEM

**MARTIN MARIETTA**

DENVER DIVISION

1. Transfer a very large number of Signals on a very small number of Wires
2. Device-to-Device Data Transfer (Includes Device-to-Computer and Computer-to-Device)
3. Intermixed Real-Time Periodic and Non-Periodic Data Transfer
4. Random Access Data Acquisition
5. Quad-Redundancy

## II Secondary Requirements

For space shuttle applications, data bus lengths of up to 300 feet must be considered, since most shuttle configurations now under consideration are approximately 250 feet in length, and cable runs may extend another 50 feet into wing and tail sections to control surfaces and related components. Special designs which break up the total bus system into parts which serve smaller regions of the shuttle may overcome some of the problems associated with these lengths, but only at the cost of more electronics complexity for bus-to-bus data transfer and similar problems.

Data bus rates of up to ten million bits per second (10 Mbps) were initially considered for shuttle applications. As the shuttle subsystem requirements for data transfer are being defined in more detail, 10 Mbps appears to be more than is required. Some estimates now are below one Mbps. The higher rates are considered here to emphasize the potential propagation delay problems, which have not received enough attention in system design studies. Serious bit- and word-skew problems can arise when the propagation delay along the data bus length becomes an appreciable part of a bit period. The data bus system design must be such that these problems are overcome. These considerations give rise to the last two requirements listed in Figure 2. The data bus system design should include decentralized control, and clock and sync should be contained in the data waveform. In this way, clock and sync will suffer the same delay as the data, and skew can be overcome if control of data transfer is at the same place as the data source.

## II. SECONDARY REQUIREMENTS

**MARTIN MARIETTA**  
DENVER DIVISION

1. Bus Length up to 300 Feet
2. Bus Data Rates up to 10 MBPS
3. Propagation Delay Problems Must Be Overcome
4. Decentralized Control
5. Clock and Sync Contained in Data Waveform

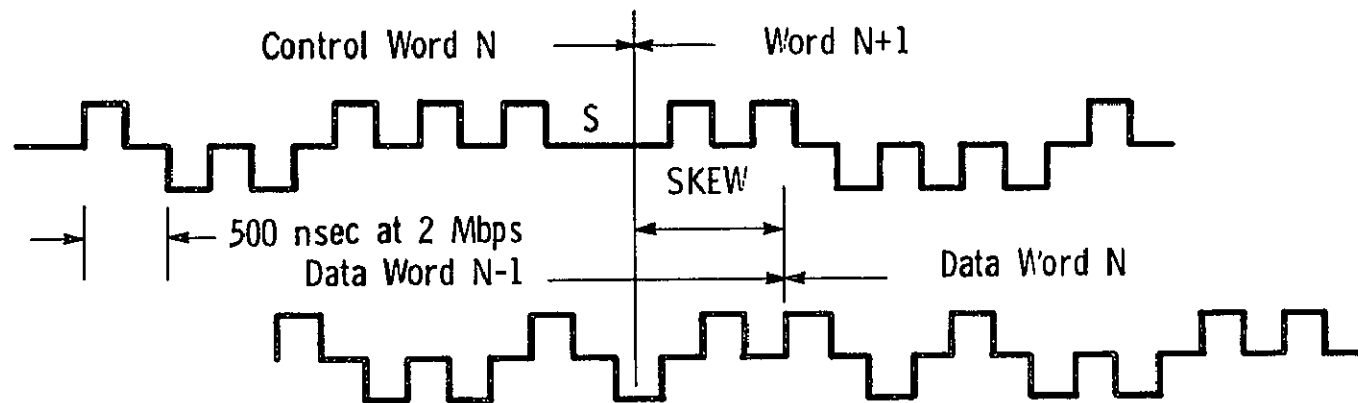
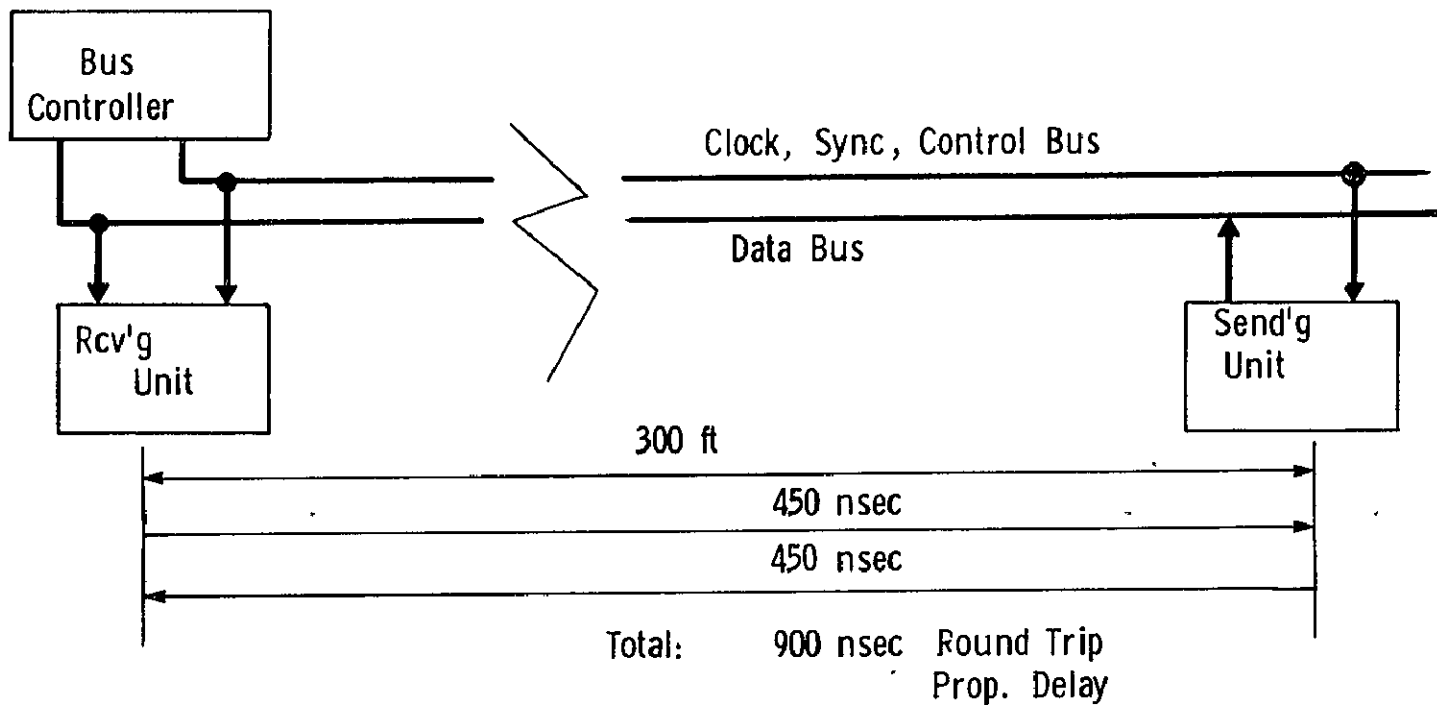
### III. Propagation Delay Problem

In Figure 3, the worst case propagation delay for a centrally controlled, synchronous data bus system is shown. The maximum length of 300 feet separating the remote sending and receiving units is shown as the worst case. If clocking and word synchronization from the control bus are used to control the transfer of data in the receiving units, the bit- and word-skew shown can result in data transfer errors. The worst case is when the receiving unit is located close to the bus controller and both are located near an end of the 300 feet of data bus. This is a probable configuration on shuttle, since the bus controller will probably be located in or near the cockpit, which will be near one end of the data bus.

The figure shows that clocking signals will arrive at the sending unit 450 nanoseconds after they arrive at the receiving unit due to the 300 feet of cable separating these two remote units. A propagation delay of 1.5 nanoseconds per foot is assumed, which is typical of shielded twisted pair and coaxial cable of the types likely to be used for the data bus system. Another 450 nanoseconds passes before the data reaches the receiving unit, making 900 nanoseconds the total time delay of the data at the receiving unit, relative to the clock and sync waveforms. There may be additional fixed delays due to various electronic operations such as A/D conversion and parallel-to-serial conversion of the data, but these delays will be fixed (to within a few nanoseconds and the variability in gate delays, etc.) and can be relatively easily provided for in the design. But the propagation delay is variable with the distance between receiving and sending units, and will vary between zero and 900 nanoseconds. This represents 9 bit periods at 10 Mbps; it represents nearly 2 bit periods at 2 Mbps. If not somehow accounted for, the most significant bits of one data word will be operated on in the receiving unit as the least significant bits of the adjacent word, or vice versa.

Waveforms on the data bus can be chosen such that timing signals used for data transfer in the receiving unit are derived from the data bus. However, this only partially overcomes the problems discussed above. If data acquisition is synchronously controlled from a centralized bus controller, data overlap can occur on the bus. This is possible because a sending unit located near the controller can be turned on while another unit located further away is still transferring data onto the data bus. The only way to overcome this problem at high data rates is to activate each sending unit only after it is verified that the data bus is inactive. Thus, the requirement for asynchronous operation is established.

### III. PROPAGATION DELAY PROBLEM



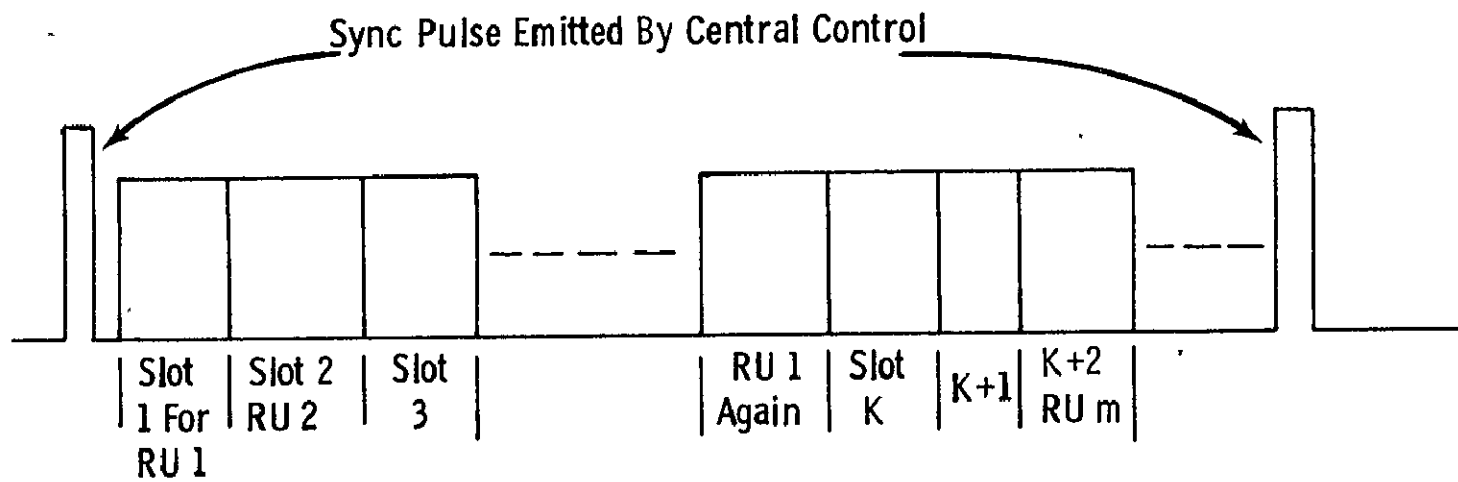
3  
⊕

#### IV. Timing Diagram For "SLAT"

These requirements, propagation delay problems, and other design considerations have led to the consideration of a data bus design technique called "SLAT" for Slot Assigned TDM. Each remote unit is pre-assigned a time slot in a complete data frame, as shown in Figure 4. Each unit determines its own time for transfer of data to the bus by measuring time relative to a frame sync pulse transmitted on the bus once each frame. This is accomplished by a simple clock contained in each unit and a down-counter which is always reset by the frame sync pulse. Any remote unit can transmit data several times each frame, or only once, as required. Each unit can also continuously acquire data at a slow rate, which can materially reduce power requirements, store the data in a small "scratch-pad" memory, and transfer the data to the bus in bursts at a very high rate. The sink address, or some means of identifying the data source and destination, is also stored in memory in the remote unit and transmitted on the bus with the data. To assure that no more than one unit transmits at the same time, two conditions must occur before any unit will transmit. One is that its own counter "times-out" to indicate its turn to transmit. The other is that an inactive bus detector indicates no other unit is transmitting data.

# IV. TIMING DIAGRAM FOR 'SLAT'

**MARTIN MARIETTA**  
DENVER DIVISION





## V. "SLAT" Advantages

The principal advantage of the SLAT technique is that it overcomes the propagation delay problem discussed earlier. This is achieved through asynchronous operation and by deriving all timing required in the receiving units directly from the waveform on the data bus. Several other important advantages also occur.

One of these is that only one bus is required. There is no need for a separate control bus on which clock, data addresses, word sync pulses, and other information is transmitted. The required timing data, and routing addresses are all carried on the same bus with the data. The total data rate on the single SLAT bus will be less than the sum of the rates on the two buses of centrally controlled synchronous systems, because much of the control and sequencing is accomplished within the remote units. The system simplification which results from only one bus will increase reliability because parts count is reduced. For example, only one set of bus receivers is required in each remote unit, and only one set of redundancy management electronics is required. The redundancy management circuits may be majority voters for triply redundant systems, or it may be more complex "Authentic Data Generators" for quad redundant systems, which compare the data from the four bus receivers, monitor the waveforms received, check parity, and perform other checking and switching functions to assure that correct data is used after the second (and for some cases, the third) failure. Since this equipment must be repeated for each set of bus receivers required, a reduction from two buses to one can result in a significant reliability improvement.

The SLAT design also permits continuous data acquisition and processing functions within each remote unit. This permits much lower rates of operation within the remote units, and consequently results in much lower power consumption. For example, the A/D conversion operation, if performed at an output rate of one to ten Mbps, requires considerably more power than if the rate is 10 to 100 kilobits per second. The power consumption of the required signal conditioning amplifiers and commutation switches is also similarly reduced at the lower operating rates.

Each remote unit in SLAT includes a small, simple data acquisition control system, or commutator. This can be a very simple device, since each remote unit will serve only a few channels, say 32 or 64. A counter and simple, easily programable, decoder is all that is required. The sum of all this circuitry in all remote units, compared to the rather complex centralized bus controller and all of the address decoding circuits in all of the remote units, results in a significant system simplification and reliability improvement.

## V. 'SLAT' ADVANTAGES

MARTIN MARIETTA

DENVER DIVISION

1. One Bus instead of Two - Higher Reliability

One Set of Bus Receivers and Redundancy Management Electronics

(Authentic Data Generators)

2. No Propagation Delay Problem

Achieved Through Asynchronous Operation

3. Decentralized, Simplified Control - Higher Reliability

4. Continuous Data Acquisition and Processing in Remote Units Results

in Lower Rate and Lower Power

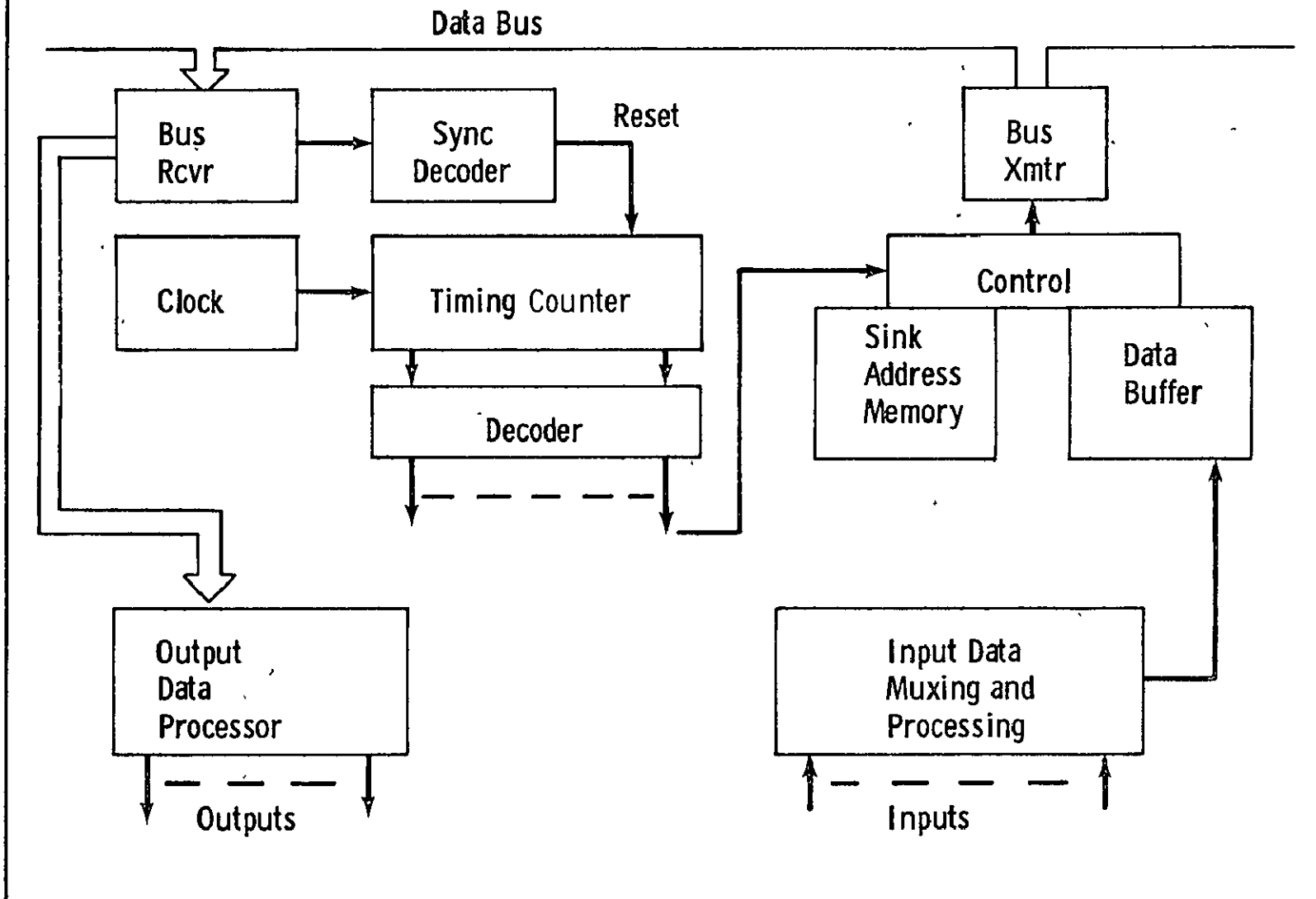
## VI. Remote Unit Block Diagram

A block diagram of a representative SLAT remote unit (RU) is shown in Figure 6. In addition to the usual input data multiplexing and processing circuits, and the output data demultiplexing and processing, the SLAT remote unit includes a frame sync decoder and a timing circuit. The timer consists of a clock, counter, and timing decoder. A timing decoder output is required for each time slot per frame assigned to the RU. An active decoder output enables the bus transmitter and initiates data transfer from the buffer memory to the data bus. A sink address memory is included to tag each data sample placed on the bus with a "sink" or destination address. Memory design will depend upon the implementation approach and the memory size required for data storage. However, it seems likely that a semiconductor read/write memory for data and a semiconductor read only or "read mostly" memory for sink addresses will meet most requirements.

General purpose data processors with built-in data monitoring, test routine sequences, and similar processing functions are entirely possible with this approach. The programming of input data multiplexing, A/D conversion, and buffer storage is carried out completely independently of bus activity, and at a much lower rate.

# VI. REMOTE UNIT FOR 'SLAT'

MARTIN MARIETTA  
DENVER DIVISION



## VII. BIPHASE L (Manchester) Waveform

Three basic requirements must be considered in selecting the data bus waveform used. The first is that a clock signal must be readily constructed from the waveform. The second requirement is that some form of word sync must be included in the waveform to identify the beginning and ending of data "words," or samples. The third requirement is that the waveform must include some unique characteristic which can be identified as frame synchronization to be used by all remote units as a timing reference.

Although frame synchronization could be accomplished by sending a special frame sync code word in the same waveform used to transmit all data, similar to techniques currently used for frame sync in telemetry formats, a unique waveform characteristic is recommended to assure no confusion with normal data, and to assure high reliability of the frame sync function. Long frame sync codes transmitted in the normal data waveform are subject to noise errors, require complex decoders, and may appear normally as data. The transmission of this code once each frame in a special waveform overcomes these potential problems.

Figure 7 shows one waveform candidate for the SLAT system. It is the "BIPHASE L" or "Manchester" code in which a logic ONE is one cycle of reference phase clock, and a logic ZERO is one cycle of inverted phase clock. Thus, a transition occurs every bit period, and clock can be easily derived from the waveform. Word sync is uniquely identified by increasing the period during which there is no transition. In the example shown, the no-transition period is increased to 1.5 bit periods; the total word sync duration is 3 bit periods.

Similarly, frame sync is uniquely identified in this waveform by increasing the no-transition period to 3 bit periods. The complete frame sync bit period is 6 data bit periods. Four or more frame sync bits should be transmitted as the complete frame sync code. The "BIPHASE L" waveform is particularly well suited to lossy line data bus systems in which attenuation is deliberately inserted along the data bus (for example, at every remote unit) so the bus can remain operative even after it suffers open circuits or short circuits along its length. This system design requires data bus receivers with an AGC dynamic range of about 40 db. Thus, no data or sync information can be carried in the waveform amplitude, which is true for the "BIPHASE L" waveform.



### VIII. BIPOLAR RZ Waveform

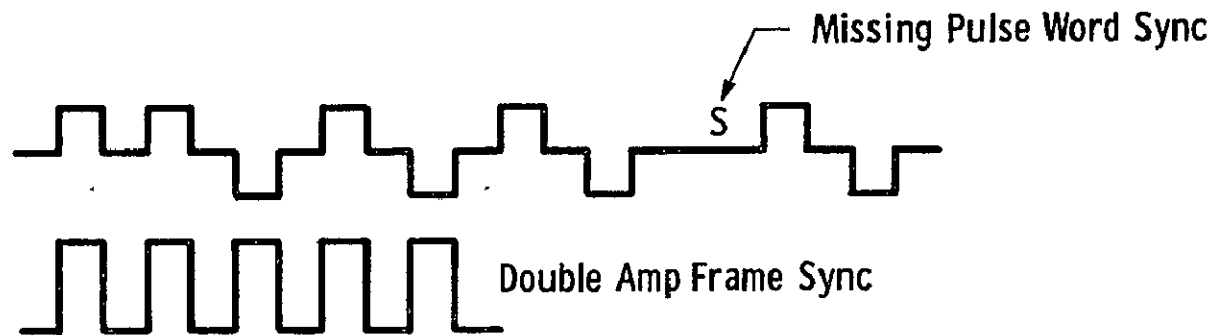
Another waveform that requires less complex data bus receivers and simpler circuits for clock reconstruction is "BIPOLAR RZ" shown in Figure 8. A logic ONE is represented by a positive clock pulse; a logic ZERO by a negative clock pulse. Word sync is represented by a missing pulse, and frame sync is represented by a number of positive double amplitude pulses. Clock pulses are obtained by ORing the outputs of the positive ONE detector and the negative ZERO detector. A "one-shot" multivibrator with its time constant selected at 1.5 bit periods serves as a word sync detector, and a properly biased detector can discriminate the double amplitude frame sync pulses.

One-shot multivibrators can be used as word sync and frame sync detectors with the "BIPHASE L" waveform using a time constant of 2.5 bit periods for word sync and 5 bit periods for frame sync. For both waveforms, a one-shot with a time-constant of 20 bit periods can serve as an inactive bus detector. That is, if no transitions occur on the bus for 20 bit periods, the "inactive bus detector" one-shot "times out" (changes state) and indicates an inactive bus.

With the "BIPOLAR RZ" waveform, the attenuation along the length of the data bus must be low enough that the double amplitude frame sync pulse is not confused with the normal amplitude data pulses. Both shielded twisted pair and coax cables are available which have attenuation factors below 1 db per 100 feet at 10 MHz. Thus, less than 3 db attenuation over the total data bus length of 300 feet can be expected. This is a voltage ratio of less than 1.4, which provides adequate detection margin and noise margin for the sync pulses.

## VIII. BIPOLAR RZ WAVEFORM

MARTIN MARIETTA  
DENVER DIVISION



Word Sync Detector: O. S. 1 -  $T_1 = 1.5T$

Frame Sync Detector: Double Amp (No One Shot Req'd)

Inactive Bus Detector: O. S. 3 -  $T_3 = 20T$

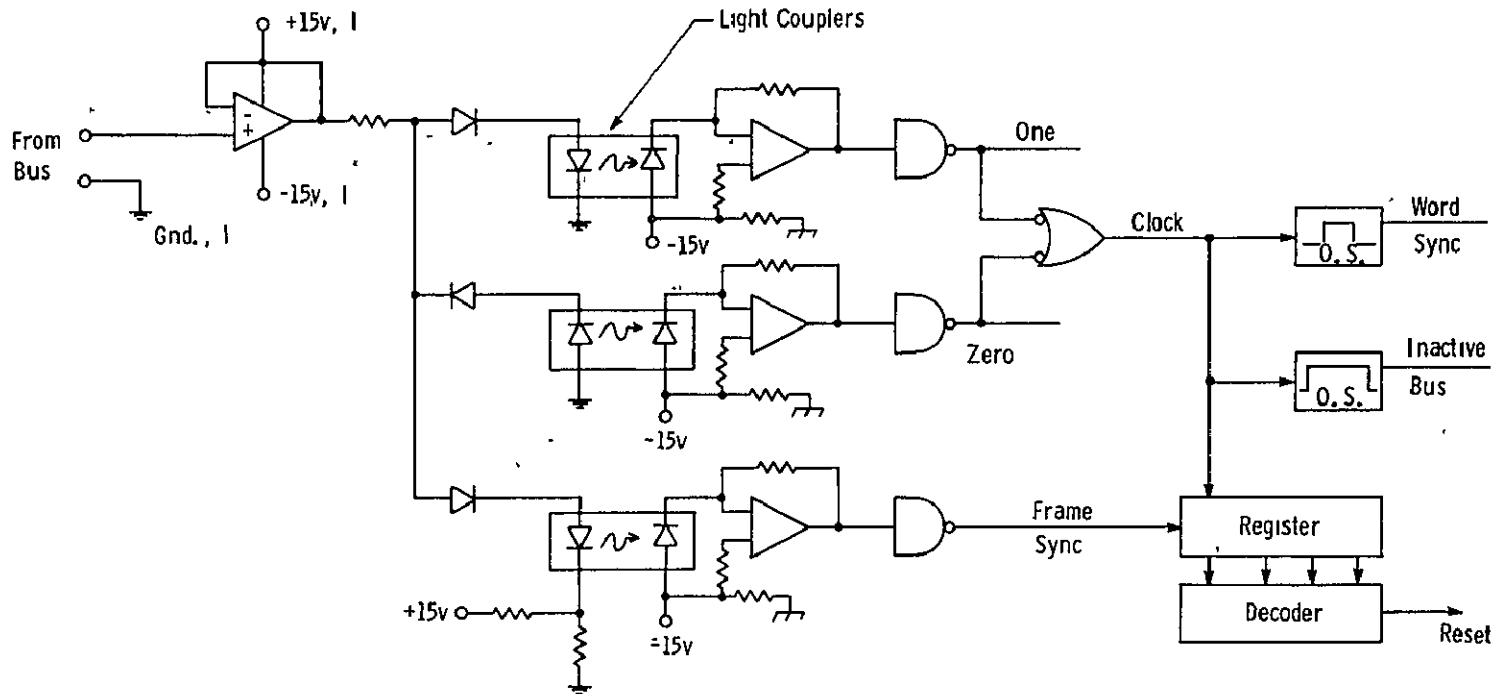


## IX. Bus Receiver for BIPOLAR RZ

A bus receiver for the BIPOLAR RZ waveform is shown in Figure 9. This design features light couplers instead of transformers as the isolating element between the data bus and the remote unit. These light couplers consist of a light emitting diode and a photo-diode or photo-transistor contained in the same package. Units are available which, with appropriate amplifiers, operate well at 10 Mbps. They provide better isolation than transformers, are smaller, and the total circuit requires less power. To achieve high isolation, the amplifier on the data bus side of the light couplers must be operated from a separate, isolated winding on the DC-DC converter power supply. Similar requirements hold for bus receivers using transformer signal isolation.

The polarity of the signals is used to separate logic ONES and ZEROS and to route them through separate light couplers. Then clock is obtained by ORing these together and sending the resulting signal to "word sync" and "inactive bus" one-shot detectors. The frame sync pulses are detected by sending them through an appropriately biased light coupler which will exclude ordinary data pulses. The frame sync pulses are sent to a special register and decoder, which will require that several (for example, four) frame sync pulses must be received consecutively to produce a frame timing reset pulse.

# IX. BUS RECEIVER FOR BIPOLAR RZ WAVEFORM



#### X. Bus Receiver for BIPHASE L

Figure 10 shows a representative bus receiver if the BIPHASE L waveform is used. Since only one amplitude, and one polarity, is used in the waveform, a single light coupler or transformer is adequate. The buffer amplifier shown does not include AGC as would be required if the lossy line approach is used.

The four "one-shots" labeled A, B, C, and D are used to reconstruct the clock. Together, one-shots A and B provide pulses for every transition, regardless of whether it is a rising or falling transition. One-shots C and D are timed and interconnected so that alternate pulses from A and B are removed if they occur one-half bit period apart. The result is one clock pulse in the center of each bit period, which is input to the data register, as shown.

Raw BIPHASE L data is also input to three other one-shots, with appropriate time constants to detect word sync, frame sync, and an inactive bus.

When Figures 9 and 10 are compared, it is seen that several more one-shots are required in the BIPHASE L receiver, but more light couplers and associated amplifiers are required in the BIPOLAR RZ receiver. If a one-shot is considered about equal in complexity to an operational amplifier, the bus receiver for the BIPOLAR RZ waveform is slightly less complex.



## XI. Data Acquisition And Storage

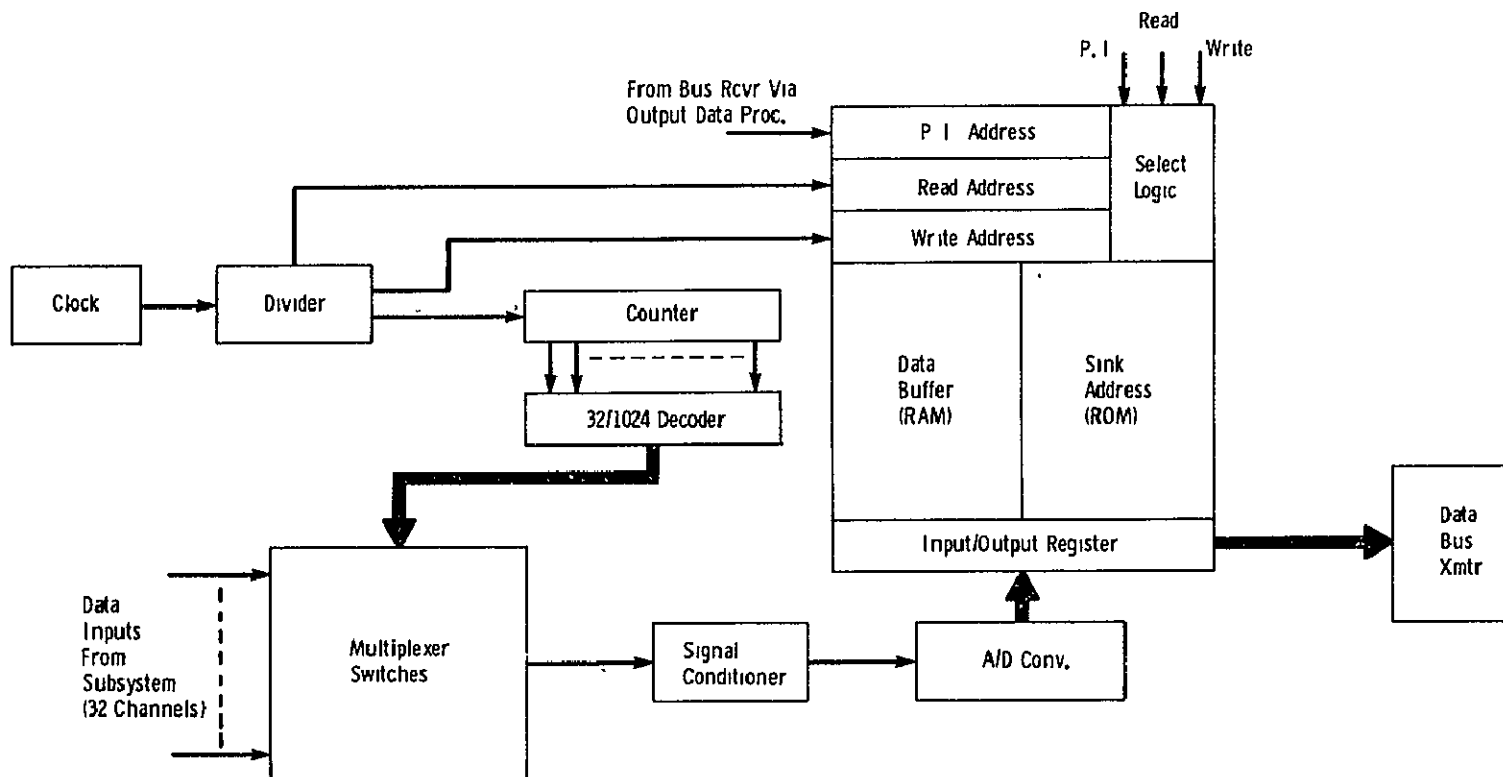
Those portions of the SIAT remote unit concerning data acquisition and buffer storage follow conventional designs as shown in Figure 11. A clock with a divider chain with multiple outputs can provide the very fast clock required for memory read and data transfer to the bus, as well as the much slower clock required for local data acquisition and processing. For controlling the sampling (commutation) of input data, a simple counter and a programmable "any 32 out of 1024" type of decoder is shown. This type of programmer is very simple and yet can accommodate complexities arising from very fast sampling rates intermixed with very low sampling rates. For shuttle applications, some channels may be sampled at 500 samples per second while others may be as low as 1 sample per second. Further simplifications in the programmer are possible if all 32 channels are sampled at the same rate, but in the interest of making all remote units identical, the type shown is recommended.

The write address register is incremented in sequence with the channel commutation. Details of the data storage pattern will depend on the relative sampling rates and whether high rate data must be transferred to the bus in real time, as for control, or whether all channels are scanned and stored, and then transferred to the bus. The number of words of random access memory (RAM, or "scratch pad") required will also depend on these details, but in all cases it appears that a small semiconductor memory will be well suited to this application. A word of sink address, shown here stored in a read only memory (ROM), is used for each data buffer word. This address is used to control the routing of data on the bus. In systems that permit only device-to-computer and then computer-to-device data transfer, the sink address memory would not be required in the remote units. However, the equivalent of the sum of all the ROMs in all the remote units will be required in the computer. While a ROM is shown, this sink address memory could also be a "read mostly memory" or a magnetic memory. These devices would permit easy reprogramming of the sink addresses

The priority interrupt (P.I.) address register permits unscheduled acquisition of any channel of data from the buffer. The P.I. address is obtained via the output data processor in the same remote unit.

Although not indicated in the figure, nothing in the design precludes a general purpose data processor with some arithmetic, test sequencing, and other functions operating in concert with the data acquisition and storage functions.

# XI. DATA ACQUISITION AND STORAGE

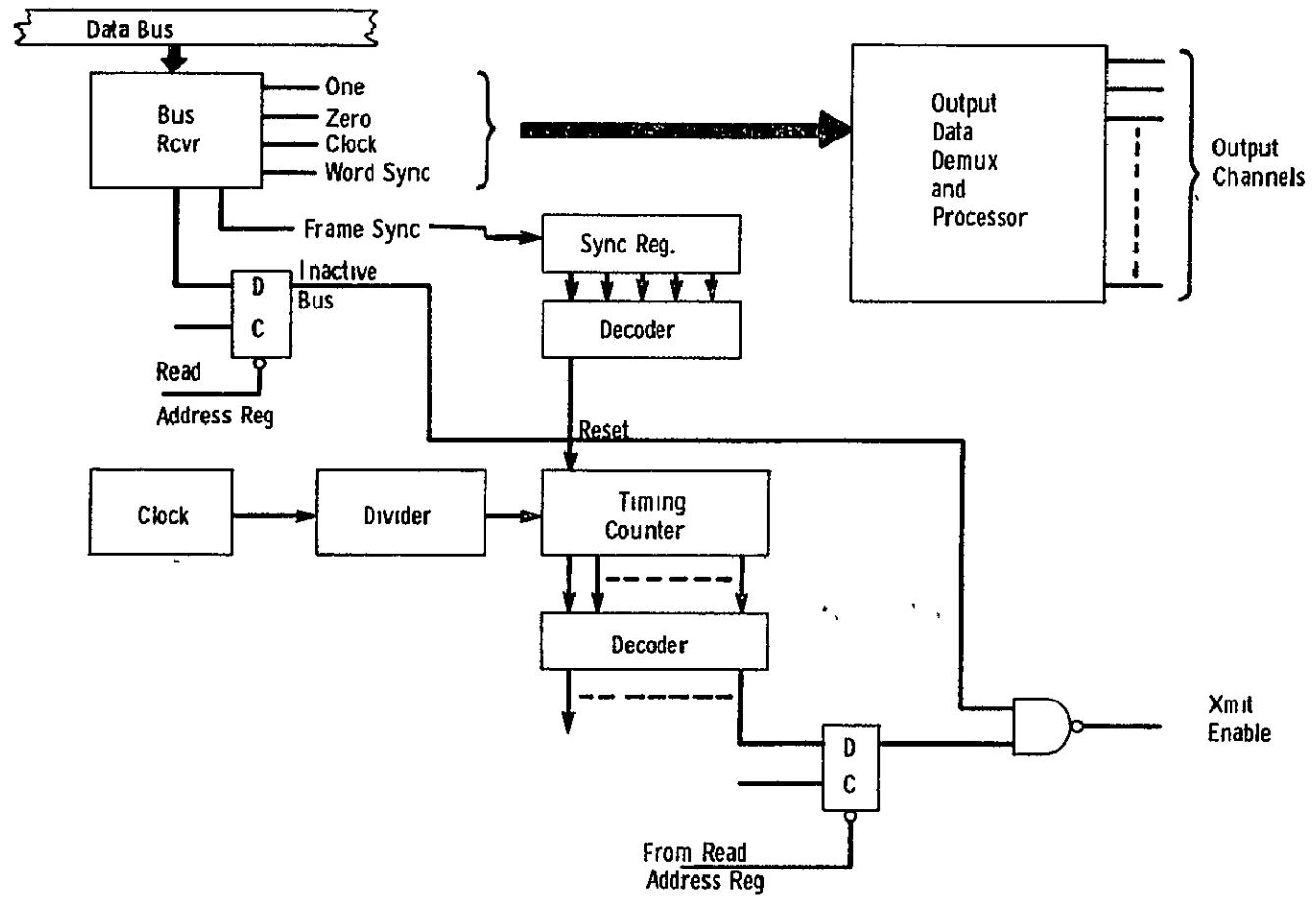


## XII. Data Transmission Timing

A clock and timing counter which is reset by the frame sync pulse are the principal elements required for each remote unit to determine when it is its turn to transmit data onto the bus. Both the inactive bus detector and the timing decoder must be in the appropriate state to enable transmission. These flip-flops are cleared when a prescribed address is reached in the buffer memory read cycle, returning the remote unit to its non-transmitting state. Note that several timing decoder outputs are possible, permitting each remote unit to transfer data to the bus several times each data frame, if required.

The timing accuracy required of the clock can be determined from the permissible "dead bus periods" between each unit's transmission. The total data frame will probably be somewhere in the range from 0.1 second to 1.0 second, and there may be about 100 remote units. As few as 10 rather large regional data bus interface units have been suggested. In any case, the average duration of a data burst onto the bus may be about one millisecond. If the "dead bus periods" are to be kept to 1%, the timing accuracy must be 1% of 1 millisecond, or 10 microseconds. Thus, we require 10 microseconds accuracy over a period of 0.1 to 1.0 second, or one part in  $10^{-4}$  to  $10^{-5}$ . An oscillator with a thermal coefficient of one part per million per degree centigrade then provides the minimum required accuracy over 100 degrees centigrade. Simple, low power oscillators can provide this accuracy.

## XII. DATA TRANSMISSION TIMING

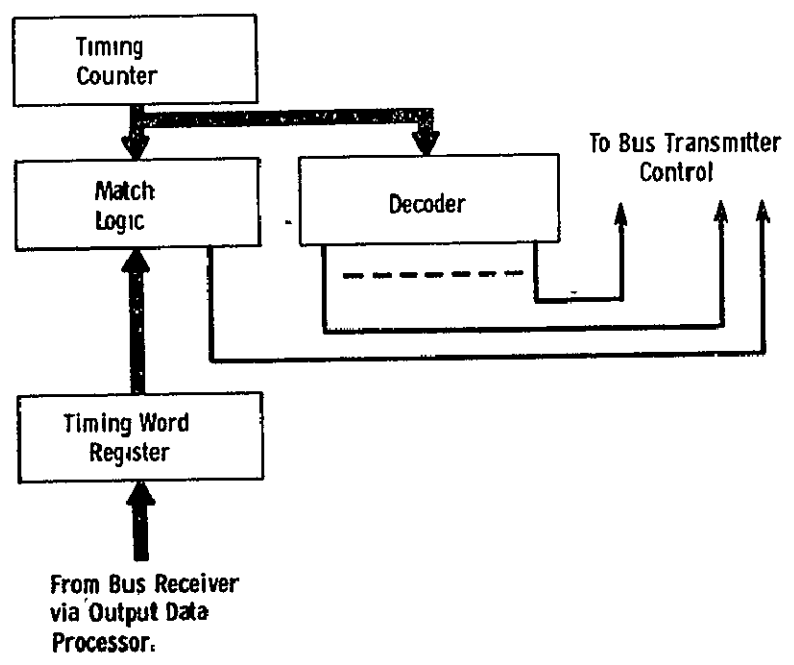




### XIII. Random Access With "SIAT"

Figure 13 shows how any remote unit can be randomly accessed by a centralized data management system (DMS). To accomplish this, a portion of the available time slots per frame would be unassigned. These unassigned slots could then be randomly assigned at any time to any remote unit as shown in the figure. The DMS unit sends a timing word corresponding to one of the available random access slots to the desired remote unit. This word is stored in the remote unit, and match logic continuously compares the timing counter contents with the random access timing word. When the match occurs, the output signal is combined with the signal from the inactive bus detector to enable data transfer to the bus.

### XIII. RANDOM ACCESS WITH "SLAT"



#### XIV. Design Alternatives

Two alternatives to the SLAT design, which retain most of the autonomous features of the SLAT RUs, have been considered. The first employs a central unit which addresses each remote unit (RU) and causes the RU to dump data onto the bus. Only the RU is addressed, not individual data channels served by the RU. The local data acquisition, programming, and processing would be carried on in the RU just as in the SLAT design. However, the RU timing counter and decoder would be replaced by an RU decoder. The central addressing unit would wait for an inactive bus following each RU transmission to assure no data overlap on the bus. The waveforms recommended for SLAT should be used to include timing with data transmission to eliminate propagation delay problems. This design alternative results in a simpler RU, because the RU decoder is simpler than the timing device. However, it adds a central RU address generator, so the total system complexity is about the same.

The other alternative is sequential addressing of the RUs, by the RUs themselves. That is, each RU would address the next RU in turn. The RU design would be essentially the same as for central addressing. But the concept suffers from a serious failure mode. The failure of any RU to address the next RU in sequence could result in the whole system stopping. Even though the redundancy planned for shuttle makes any system failure extremely unlikely, the effect of the failure is catastrophic. Therefore, this technique is rated far below the other alternatives. Some techniques for overcoming this failure mode based on dead bus durations may be worked out, but even with this, the technique is considered a poor reliability risk.

## **XIV. DESIGN ALTERNATIVES**

**MARTIN MARIETTA**  
DENVER DIVISION

### **1. REMOTE UNIT ADDRESSING**

Eliminate Timing Circuits in RUs

Requires RU Decoders

Central Addressing Unit would wait for Inactive Bus

### **2. SEQUENTIAL ADDRESSING**

Each RU Addresses the Next RU

Failure can Break the Chain and Stop All Units

PRECEDING PAGE BLANK NOT FILMED

SOFTWARE DEVELOPMENT AND VERIFICATION TECHNOLOGY

H. Trauboth and B. Hodges

NASA-Marshall Space Flight Center  
Huntsville, Alabama

N71-35059

# **CONTENTS**

- 1. INTRODUCTION**
- 2. TYPES OF SOFTWARE**
- 3. DESIGN CONSIDERATIONS**
  - MODULARITY**
  - RELIABILITY**
  - LANGUAGE**
- 4. SOFTWARE DEVELOPMENT TOOLS**
  - LANGUAGE PROCESSORS**
  - DESIGN ANALYSIS AND VERIFICATION TOOLS**
  - SOFTWARE CONFIGURATION CONTROL TOOLS**
- 5. DEVELOPMENT AND VERIFICATION PROCESS**
- 6. CONCLUSIONS**

## INTRODUCTION

### Purpose of Presentation

In the development and operation of the Space Shuttle, computer software will play a major role. For example, the guidance and navigation system will be more flexible for different missions. The propulsion system will have more and throttleable engines for soft landing, and the checkout system will go on-board. These are a few major areas where a powerful flight computer system performs considerably more functions in number and complexity than the computer of the SATURN V vehicle. It is estimated that close to \$1 Billion; i. e., about 10% of the total Shuttle development costs, will go into computer software and about half of these costs into checkout and verification of software. This shows that software has grown from mere programming to a new technology.

The purpose of this paper is to point to some critical areas in software development which should receive more attention and be funded for further research. We will present here only conceptual solutions to the problems we are working on because of limited time available. We wish to emphasize that proper tools for the development and verification of complex software is needed and that it is now the time to prepare and perfect these tools.

NASA has to make sure that the software development by the various contractors is well coordinated and integrated, and that the software products delivered are meeting the specifications. For this reason, NASA has to evaluate flight and ground software extensively at all development phases. It should, therefore, actively engage in developing and perfecting the tools for this evaluation and coordination such as higher-level languages, compilers, executive systems, and verification and simulation systems.

### Problems in Software Technology

The problems we will list here are well-known to most systems software people and have been presented elsewhere such as in the article by Barry Boehm of RAND Corporation on "Some Information Processing Implications of Air Force Space Missions in the 1970's" published in *Astronautics & Aeronautics*, Jan. 1971. We will repeat them briefly because hardware people should be alerted to them.

The costs of the software development, verification, and maintenance are enormous, close to \$1 Billion. At that price, the confidence in software reliability; i. e., its logical integrity, is still relatively low. The software size, complexity, and development efforts are most often underestimated which leads to unforeseen schedule slippages and hardware expansions. Though the documentation of software is usually massive, gaining insight and visibility into its functions is very difficult - especially for the manager.

The software development process usually does not start together with and is not synchronized with the hardware development process so that the software is seldom ready in time.



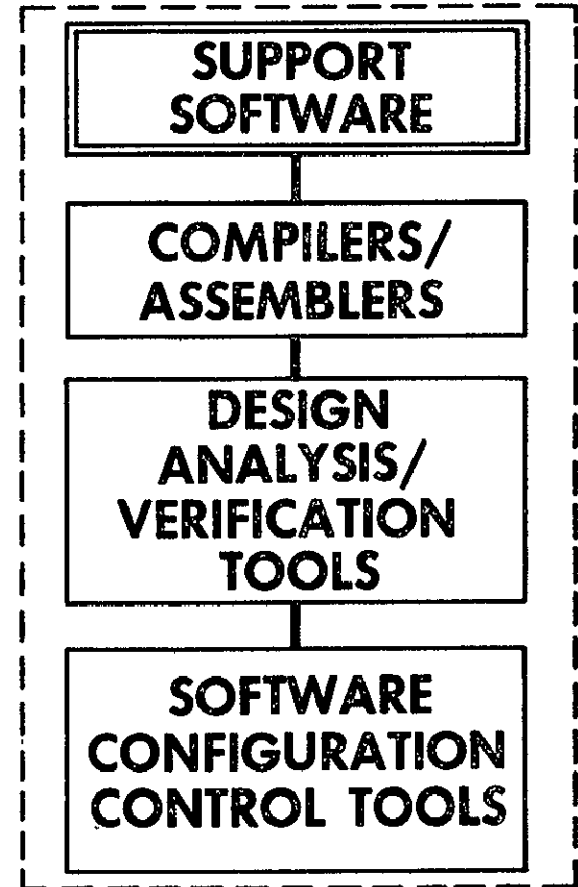
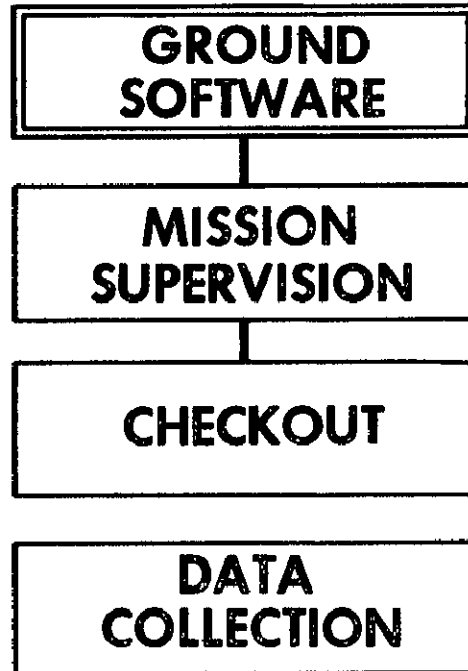
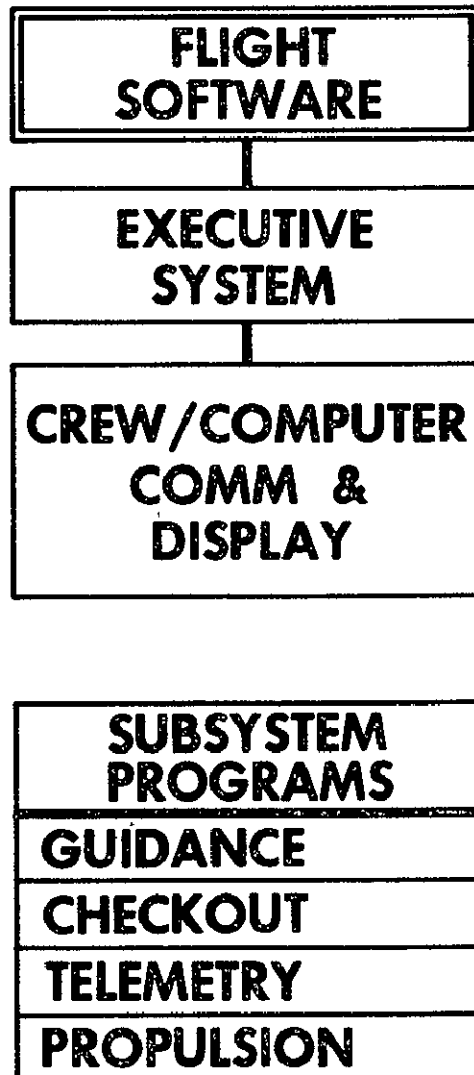
# **PROBLEMS IN SOFTWARE TECHNOLOGY**

- **COSTS OF SOFTWARE DEVELOPMENT, VERIFICATION, AND MAINTENANCE ARE ENORMOUS**
- **CONFIDENCE IN RELIABILITY IS RELATIVELY LOW**
- **SOFTWARE SIZE AND EFFORT ARE UNDERESTIMATED, TIMELINESS OF DELIVERY IS LACKING**
- **VISIBILITY INTO COMPLEX SOFTWARE IS POOR**
- **SOFTWARE DEVELOPMENT PROCESS NOT SYNCHRONIZED WITH HARDWARE DEVELOPMENT PROCESS.**

### TYPES OF SOFTWARE

When we talk about software we often think of different types of software. Here, we distinguish between three major categories: The Flight Software, the Ground Software, and what we call the Development Support Software. Each of these categories can then be broken further into sub-groups. In this paper, we want to concentrate on the last group, Development Support Software, since we feel its importance should be put into proper perspective.

# TYPES OF SOFTWARE



## DESIGN CONSIDERATIONS

### Modularity of Design

Objectives.- The software design should be modular for various reasons, even at the expense of some slight inefficiency in the final computer programs. A proper modularity concept of the software allows a more systematic and less expensive verification process of the software, since only those modules will have to be verified which are affected by changes. For mission changes, the software does not have to be redesigned. Complex software structures will become better visible and can be better documented (by less paper work). Since the large amount of software work has to be distributed and be performed by several contractors, proper modular design and interface definitions are essential for better management and configuration control of software development. In order to achieve this modularity, standard rules obeyed by everybody participating in the total software development have to be developed. All these factors will then improve the reliability of the software.

# **MODULARITY OF DESIGN**

## **OBJECTIVE**

- **REDUCTION OF VERIFICATION COSTS**
- **REDUCED REDESIGN FOR MISSION CHANGES**
- **BETTER VISIBILITY AND DOCUMENTATION OF COMPLEX SOFTWARE**
- **BETTER MANAGEMENT AND CONFIGURATION CONTROL OF SOFTWARE DEVELOPMENT**
- **STANDARD RULES FOR SOFTWARE DESIGN**
- **BETTER RELIABILITY OF SOFTWARE**

Approach.- The modularity concept has to follow certain ground rules to achieve the objectives mentioned previously. The programs containing instructions should be clearly separated from the files containing data and parameters. One might distinguish further between those data that control the processing flow (such as switches and indicators) and those data that are being processed for final output. The structure of both, the programs as well as the files, has to be hierarchical. The structures of the interfaces between programs and subprograms and between programs and files have to follow certain standard rules. The modularity boundaries should be function-oriented as well as mission-oriented, which should allow an easy exchange of mission-dependent parameters and program modules. To facilitate verification, certain check variables (like test points in hardware) have to be generated by the programmer and/or the compiler. In order to determine the coupling and interconnections of the various parts of the software, the functions and connections of the software should be described by simple diagrams and tables as several hierarchical levels. This will make the effect of changes more quickly visible. The next three slides should just portray the basic idea of this approach. The modularity rules should be defined in such a way that the compiler can check a program for the fulfillment of these rules. The compiler may also perform certain bookkeeping functions which are necessary to aid the verification process such as generating the diagrams and tables that follow.

# **APPROACH TO SOFTWARE MODULARITY**

- **CLEAR SEPARATION OF PROGRAMS AND DATA FILES**
- **HIERARCHICAL STRUCTURE OF PROGRAMS AND DATA FILES**
- **STANDARD INTERFACES BETWEEN PROGRAMS AND FILES**
- **MODULES WHICH ARE FUNCTION-ORIENTED AND/OR MISSION-ORIENTED**
- **EXCHANGE OF MISSION-DEPENDENT PARAMETERS AND PROGRAM MODULES**
- **GENERATION OF CHECK VARIABLES FOR VERIFICATION PROCESS**
- **FUNCTIONAL DESCRIPTION OF SOFTWARE BY SIMPLE DIAGRAMS AND TABLES AT SEVERAL HIERARCHICAL LEVELS**
- **CHECK OF MODULARITY RULES AND VERIFICATION BOOKKEEPING FUNCTIONS BY COMPILER**

Modular software description.- Here we want to sketch a concept of software description at a higher functional level in a different and more engineer-like way, which is useful for managers who need not understand the design at the detail flow diagram level but rather at the "sub-routine" level. It should also serve the purpose to assess quickly the effect of a change in a subprogram or file on other subprograms and files.



# **MODULAR SOFTWARE DESCRIPTION**

- 1. SUBPROGRAM CONNECTION DIAGRAM**
- 2. SUBPROGRAM 1 TABLE AFFECT DIAGRAM**
- 3. SUBPROGRAM EXIT CONDITION LIST**
- 4. SUBPROGRAM SEQUENCE DIAGRAM**
- 5. CHANGE EFFECT DIAGRAM**

---

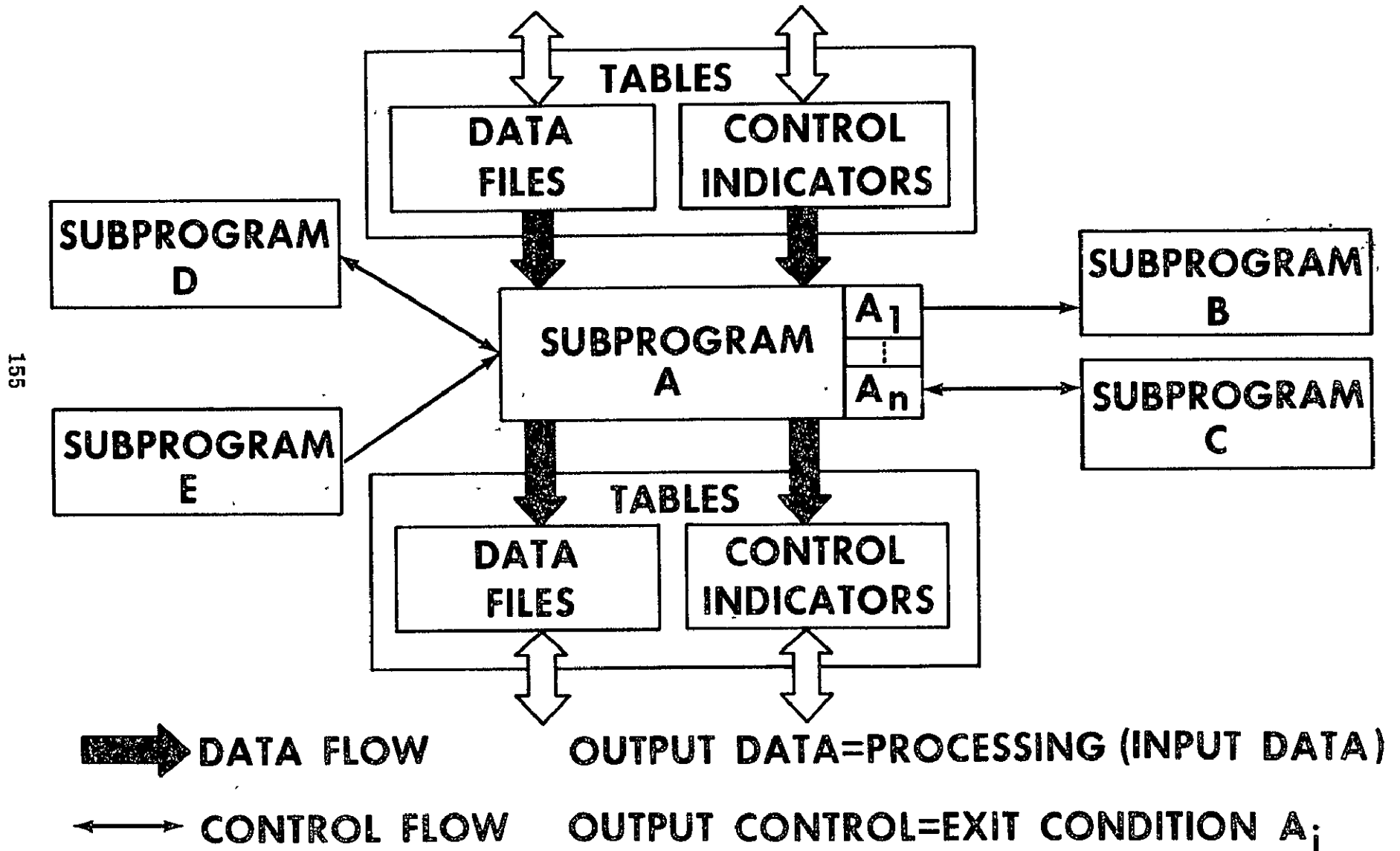
**6. SUBPROGRAM FUNCTIONAL DESCRIPTION  
(TEXT & FLOW DIAGRAMS)**

**7. TABLE FUNCTIONAL DESCRIPTION  
(TEXT & FORMATS)**

The software is viewed in a simplified way as a collection of subprograms and tables. Tables include files and indicators as shown in the following slide. Horizontally, we have depicted the control flow and vertically, the data flow. Control is transferred from one subprogram to another by exit condition  $A_1$ .

A subprogram takes information from data files as an input, processes this data depending on the settings of certain control indicators and outputs the processed information into the same or other files. During the execution, a certain exit condition is reached which transfers control to another subprogram of the next lower level and returns to the same (assuming here a main driver program and subroutine calls only as means of program control transfer).

# CONTROL AND DATA CONNECTIONS OF A SUBPROGRAM

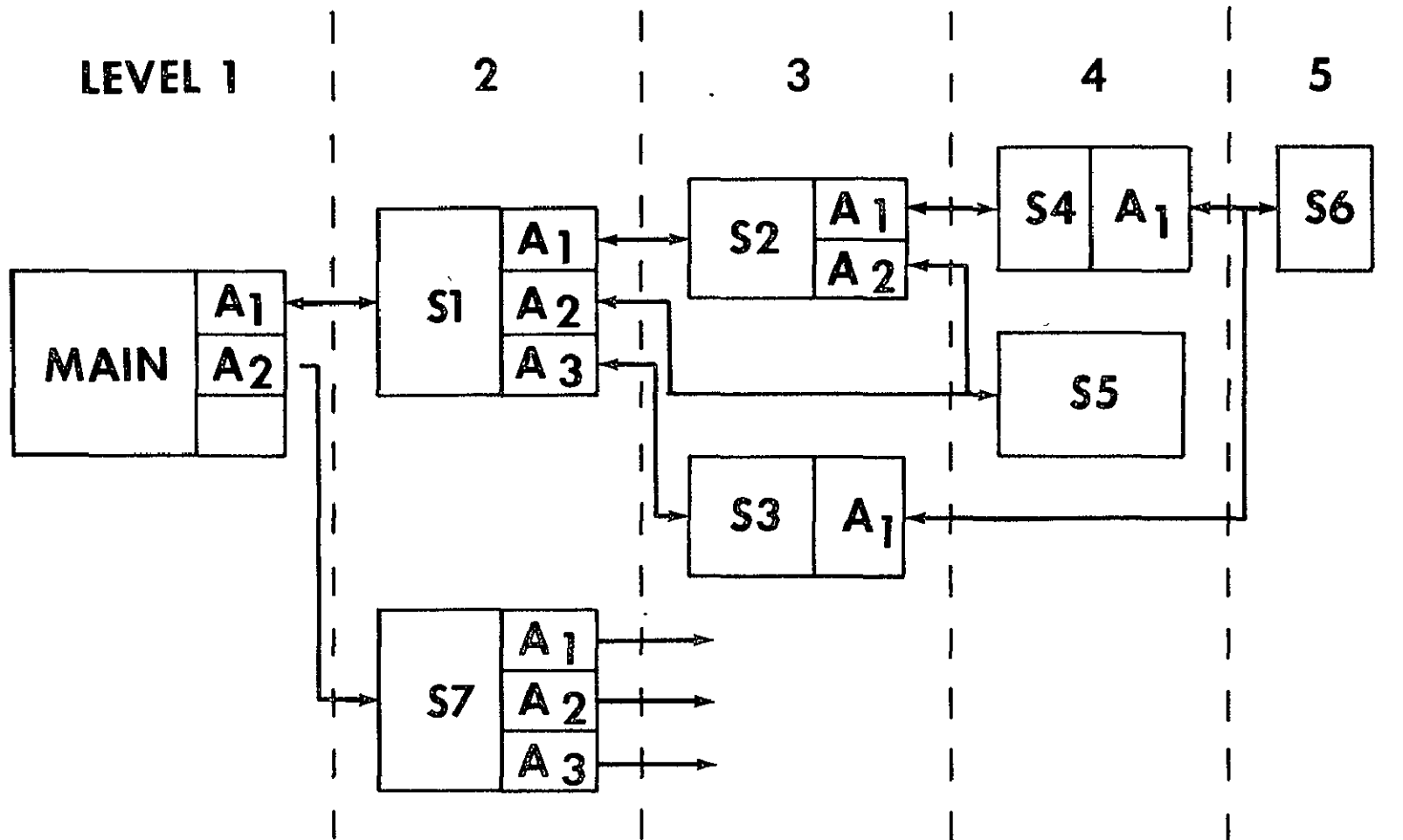


Several diagrams and tables should now describe the software system.

The Subprogram Connection Diagram describes the static interconnections of all subprograms.

It is like a road map which tells which connections are possible.

# EXAMPLE OF SUBROUTINE CONNECTION DIAGRAM



The Subprogram/Table Affect Diagram shows which tables are being used as a data input or data output by which subprograms. A "reading" from a file does not change or affect the file, whereas a "writing" does. For instance, a change in subprogram S2 might affect the two tables T4 and T5 which again affect those subprograms which read from these files; i. e., subprograms S3, S5, and S7. The Subprogram Exit Condition List describes verbally for each subprogram the conditions for exiting the subprogram at different points. The Subprogram Sequence Diagram lists the major control flows through the network of subprograms based on the combinations of exit conditions and the Subprogram Connection Diagram. It is like following a path through the road map for different destinies. The Change Effect Diagram can be built directly from the Subprogram/Table Affect Diagram. It tells more explicitly which subprograms and files affect each other if a change in a subprogram or file is made. The Subprogram Functional Description then describes in the usual way by text and flow diagrams the processing functions of each subprogram as the functional relationship between the input data and the output data. The Table Functional Description describes by text and diagrams the contents and formats of the data in the various files and indicators.

# EXAMPLE OF SUBPROGRAM/TABLE AFFECT DIAGRAM

		TABLES					
		T1	T2	T3	T4	T5	
<b>SUBPROGRAMS</b>	<b>S1</b>	0	1	1			
	<b>S2</b>		0		1	∅	<b>CHANGE IN S2</b>
	<b>S3</b>			∅	1	∅	<b>AFFECTS S3</b>
	<b>S4</b>	1		0			
	<b>S5</b>		0	1	0	1	<b>AFFECTS S5</b>
	<b>S6</b>	0	1		1	1	
	<b>S7</b>	1	0			∅	<b>AFFECTS S7</b>

**0 = READ FROM TABLE**

**1 = WRITE INTO TABLE**

**∅ = READ AND WRITE**

### Software Execution Reliability by Executive System

Since the regulation of process flow is the dominant purpose of an Executive, it is important that this area of software technology be given considerable treatment. Implementation of reliability techniques within the Executive requires that software development begin early in the design cycle with analysts/programmers working with engineers in the development of the Avionics system. Areas where reliability could be designed into the Executive functions include inter-system information checks, process controls, failure detections and responses, resource allocation controls, and configuration controls.

Many of these techniques are closely aligned with the hardware design, such as inter-subsystem information checks. Although the processes required for inter-subsystem communication are somewhat hardware dependent, each message sent and received will have hardware and software checks. These checks include status, longitudinal record check and message number check. Other methods of failure detection in software execution can also be included in the Executive. Executive call sequences can be checked for validity to detect both hardware and software induced errors. Similarly, other active error detection procedures such as limit-checking critical variables, main storage and central processing unit control, and peripheral reconfiguration can all be an integral part of the Executive functions.



# RELIABILITY OF SOFTWARE EXECUTION BY EXECUTIVE SYSTEM

- INTER-SUBSYSTEM INFORMATION CHECKS  
MESSAGE NUMBER  
LONGITUDINAL RECORD CHECK
- PHASE-DEPENDENT PROCESSES SEGREGATED FROM  
PHASE INDEPENDENT PROCESSES
- FAILURE DETECTION/RESPONSE  
EXECUTIVE CALL SEQUENCES CHECKED FOR VALIDITY  
LIMIT CHECKS ON CRITICAL VARIABLES
- RESOURCE ALLOCATION CONTROL  
MAIN STORAGE  
CENTRAL PROCESSING UNIT

## • CONFIGURATION CONTROL

### RELIABILITY CONSIDERATIONS

- REDUCTION OF ERROR GENERATION BY USE OF  
HIGHER-LEVEL LANGUAGE AND MODULAR DESIGN
- BETTER VERIFICATION BY MODULAR DESIGN AND BY  
BETTER VERIFICATION TOOLS
- ENHANCEMENT OF HARDWARE AND SOFTWARE  
RELIABILITY BY CONSISTENCY CHECKING OF  
EXECUTIVE SYSTEM DURING EXECUTION

### Use of Higher Level Language

While higher level languages such as FORTRAN, ALGOL, and COBOL are nearly exclusively being used in ground computation facilities, they have not been applied to flight computers because flight computers had very critical memory and real-time limitations. The compilers did not generate efficient enough code to cope with these limitations. However, during the last year's computer hardware technology and compiler technology, have made great progress to diminish these limitations. Born out of the experience in software development for large scale projects such as APOLLO, it has also been realized now that other aspects besides code efficiency are important where higher level language programming has advantages over assembly language programming. The amount of labor and time can be reduced effectively. The probability of errors is reduced and the error diagnostics can catch errors at a higher logical level which aids verification. The modularity of the design can be better accomplished which results in easier program modification. The programs themselves can be better read and documented by the programmer and his supervisor.

However, there are some problems which have to be considered.

In certain program areas, the efficiency of the object code might be lower than if it had been written by a skilled assembly language programmer. The compiler has to be error free so that it does not affect the reliability of the software. The compiler itself is an additional investment which has to be modified whenever the language or the target computer changes.

But new compiler generator systems promise to overcome these disadvantages to a large degree. These new compiler techniques should be thoroughly evaluated by extensive testing on real flight and ground software utilizing a Shuttle Avionics system breadboard before final flight software is written and committed.

# USE OF HIGHER LEVEL LANGUAGE

## ADVANTAGES

- EFFORT OF CODING IS REDUCED CONSIDERABLY
- PROBABILITY OF GENERATING ERRORS IS REDUCED
- ERROR DIAGNOSTICS AT HIGHER LOGICAL LEVEL TO AID VERIFICATION
- ASSISTS IN MODULAR SOFTWARE DESIGN
- EASIER PROGRAM MODIFICATION
- BETTER READABILITY AND DOCUMENTATION OF PROGRAMS

## DISADVANTAGES

- EFFICIENCY OF OBJECT CODE MAY BE LOWER
- RELIABILITY OF COMPILER INFLUENCES SEVERELY RELIABILITY OF SOFTWARE
- COMPILER IS ADDITIONAL SOFTWARE INVESTMENT

## SOFTWARE DEVELOPMENT TOOLS

As in hardware development and production, manual routine work has been substituted by machine tools; so can handcrafted routine software development and production be aided by computer program tools. We distinguish here between three major classes of tools:

- a. The language processors which consist of compilers, interpreters, and macro-assemblers and compiler generators (or translator systems or compiler-compilers),
- b. The design analysis and verification tools which consist of interpretive computer simulators (software packages), subsystems simulators (software packages), and a hardware Avionics systems breadboard,
- c. The software configuration control tools which comprise an interactive information management system for quick-look analysis of the impact of software design changes and an information management system for detail software design changes.

Since the design of the Shuttle is still in a fluid state, it is now the time to develop and perfect the tools to make them flexible enough so that they can be adapted quickly when the Shuttle design is finalized. Since software development usually requires much lead time, it is very important that powerful software tools are available as soon as possible.

# **SOFTWARE DEVELOPMENT TOOLS**

- **LANGUAGE PROCESSORS**
  - **COMPILERS, MACRO-ASSEMBLERS, INTERPRETERS**
  - **COMPILER GENERATORS**
- **DESIGN ANALYSIS AND VERIFICATION TOOLS**
  - **INTERPRETIVE COMPUTER SIMULATOR**
  - **COMPUTER ENVIRONMENT SIMULATOR**
- **SOFTWARE CONFIGURATION CONTROL TOOLS**
  - **QUICK-LOOK CHANGE IMPACT ANALYSIS**
  - **DETAIL CHANGE CONTROL**

### Language Processors

The flight and ground software has to be developed for different subsystems with substantial different characteristics by programmers or engineers of different disciplines. For instance, the navigation, guidance, and control subsystems require extensive array arithmetic in form of matrix calculations, coordinate transformations, and vector rotations with a relatively few real-time commands, while automatic checkout covers all subsystems and requires real-time monitoring of many signals, many stimuli commands, and output data formatting. The telemetry subsystem requires another different type of data handling, such as transmission data multiplexing, formatting (compressing), and channel control. The propulsion subsystem has other language requirements, and so on.

One could now try to find one language which at an average would meet all these different requirements to a limited degree. This language would then be implemented by a single compiler, macro-assembler, or interpreter system. This approach has the advantages of reducing the impact of transferability from one host computer to another, combining the language commonalities (if they exist), and reducing the programmer training. Certain disadvantages include the complexity of structure and size of the translator, lack of language characteristics common among the various disciplines, and the translator diversity which must be present to accommodate program development by various contractors and NASA agencies.

# LANGUAGE PROCESSORS

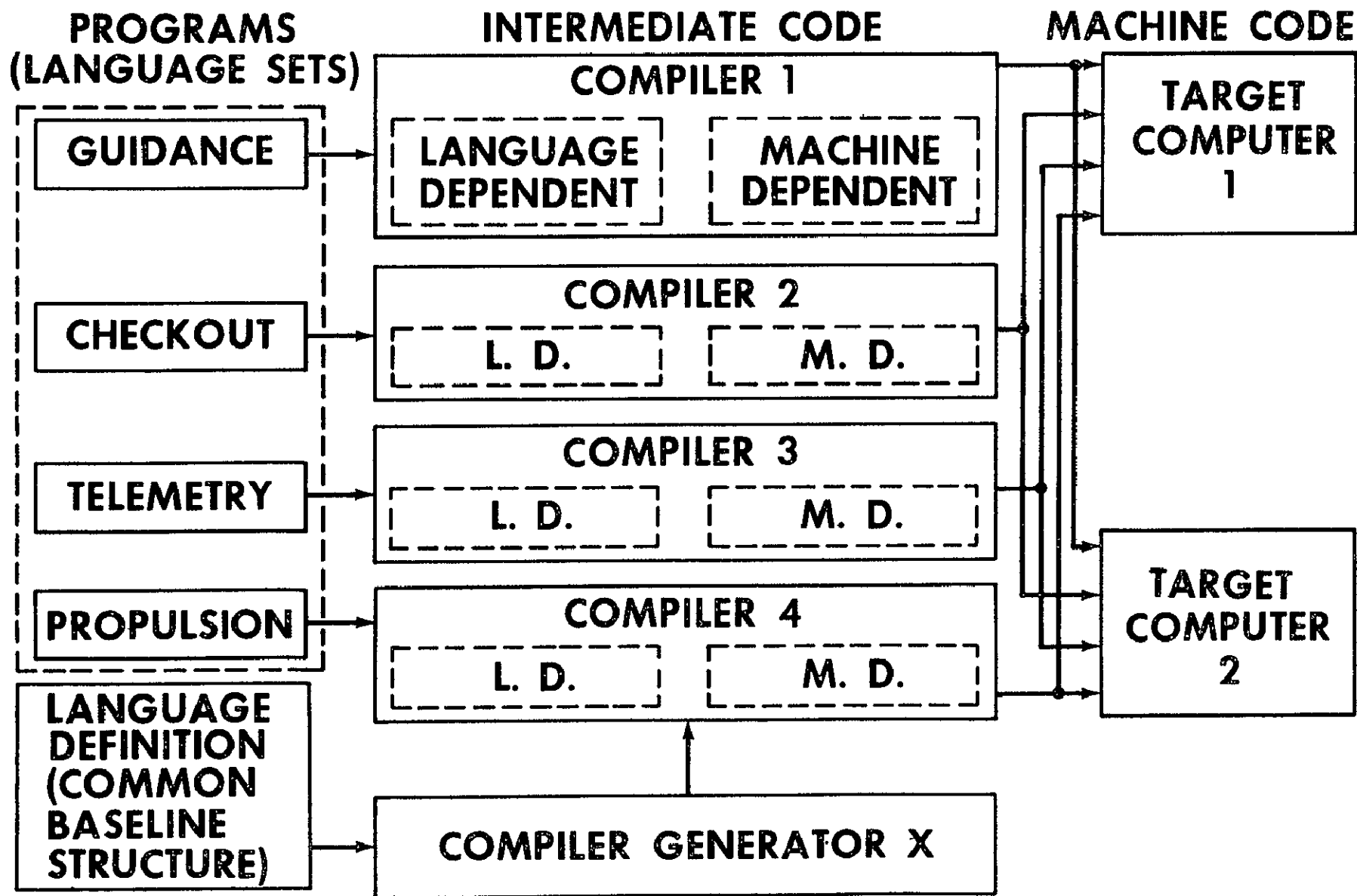
- DIFFERENT USER-ORIENTED LANGUAGE SETS
  - GUIDANCE — MATRIX CALCULATIONS, COORDINATE TRANSFORMATIONS
  - CHECKOUT — REAL-TIME MONITORING, MANY I/O COMMANDS
  - TELEMETRY — TRANSMISSION DATA FORMATTING AND CONTROL
- IMPLEMENTATION BY DIFFERENT CONCEPTS
  - SINGLE COMPILER, MACRO-ASSEMBLER, OR INTERPRETER SYSTEM
  - MULTI-COMPILER SYSTEM
- DESIGN CONSTRAINTS
  - TIME AND MEMORY EFFICIENCY OF OBJECT CODE
  - LANGUAGE EXPANSION CAPABILITY
  - REAL-TIME CONTROL CAPABILITY
  - EXTENSIVE DIAGNOSTICS AND VERIFICATION AID CAPABILITY
  - TRANSFERABILITY TO OTHER TARGET COMPUTERS

Or, one can take another approach, which is illustrated in the following slide. Here we assume we have for each discipline such as guidance, checkout, telemetry, propulsion, etc. a different language set for the different types of users. The word language "set" should express that the languages have a common baseline structure and obey certain rules which make it possible to use one compiler generator or translator system. This compiler generator then enables us to generate relatively fast a compiler for each language set. The compiler itself consists of two major parts, the language-dependent part and the machine (computer hardware)-dependent part, so that for different target computers only the machine-dependent part has to be changed. The language-dependent part of the compiler consists basically of the lexical and syntax analyzers, the intermediate code generator, and code optimizer; the machine-dependent part comprises the object code generator.

The compiler generator is a software tool for the development of modular compilers, which provides building blocks for automating those functions which are common to all compilers. We are presently evaluating various existing compiler generator systems such as AED ("Automated Engineering Design" by MIT, now Softech, Inc.), XCOMP (by Stanford University), GULP (by Martin-Marietta, Denver), and Syntax Directed Compiler (by McDonnell-Douglas). We think that such an existing system could serve as a basis for a Shuttle language translator system. Since the various language sets are still in the specification phase (CLASP, SPL, and HAL are considered for guidance, control, and command; and TOTAL and ALOFT for automatic checkout) and the flight and ground computers are not yet defined, we need a flexible and efficient compiler generator system. It is now the time to develop and perfect such a system.



# MULTI-COMPILER SYSTEM



## Design Analysis and Verification Tools

General.- The design analysis and verification tools allow the execution of flight programs on a large host computer such as the IBM 360/75 or UNIVAC 1108 even before the flight computer is manufactured. Such tools can, therefore, be used for checkout and verification of flight software. They are rather flexible in their design insofar as changes in the computer instruction set and in the subsystem design can be relatively easily adapted. They can also serve in the evaluation of the logical integrity and the efficiency of the compiler. At a relatively early phase of software and system design they can be used for systems design analysis.

The Interpretive Computer Simulator, which is a software system, executes flight computer instructions on a large host computer in simulated time; one could also say that the computer functions are being simulated to the "register level." It traces the data and program control flow and prints out register and memory cell contents under user control. It sends out and receives signals to and from its environment. Usually this simulator can be used for different flight computers of the same class (single processor computers only presently).

Though these simulators have been used extensively, also for the APOLLO program, they have certain deficiencies. Their running time is long, and multiprocessing and time-sharing are very difficult to implement efficiently.

The Subsystems Simulator is also a software system which runs on a large host computer either together with the Interpretive Computer Simulator or with an actual flight computer. It simulates in simulated time the continuous and discrete dynamics of the hardware subsystem surrounding the flight computer. In the APOLLO program, the subsystems dynamics were written in FORTRAN or in assembly language. For the Shuttle, we propose to use a higher-level simulation language such as MARSYAS which we developed. This simulation system accepts model descriptions in the engineer-oriented language MARSYAS and performs error diagnostics on the model description and simulation program prior to execution on a higher level and, therefore, more extensively than FORTRAN.

There are still some general problems. For large systems with detail models, the running time will be of long duration. For mixed continuous/sampled-data systems, it might be difficult to obtain efficiently accurate numerical solutions.

# **DESIGN ANALYSIS AND VERIFICATION TOOLS**

## **PURPOSE**

- CHECKOUT AND VERIFICATION OF FLIGHT SOFTWARE ON LARGE HOST COMPUTER
- EVALUATION OF COMPILER INTEGRITY AND EFFICIENCY
- SYSTEMS DESIGN ANALYSIS

## **INTERPRETIVE COMPUTER SIMULATOR (SOFTWARE)**

**FUNCTIONS:** EXECUTES AND TRACES INTERPRETIVELY FLIGHT PROGRAM ON LARGE HOST COMPUTER

**PROBLEMS:**

- RUNNING TIME IS LONG
- MULTI-PROCESSING IS VERY DIFFICULT TO IMPLEMENT EFFICIENTLY

## **SUBSYSTEMS SIMULATOR (SOFTWARE)**

**FUNCTIONS:** SIMULATES DYNAMICS OF SUBSYSTEMS SURROUNDING THE FLIGHT COMPUTER

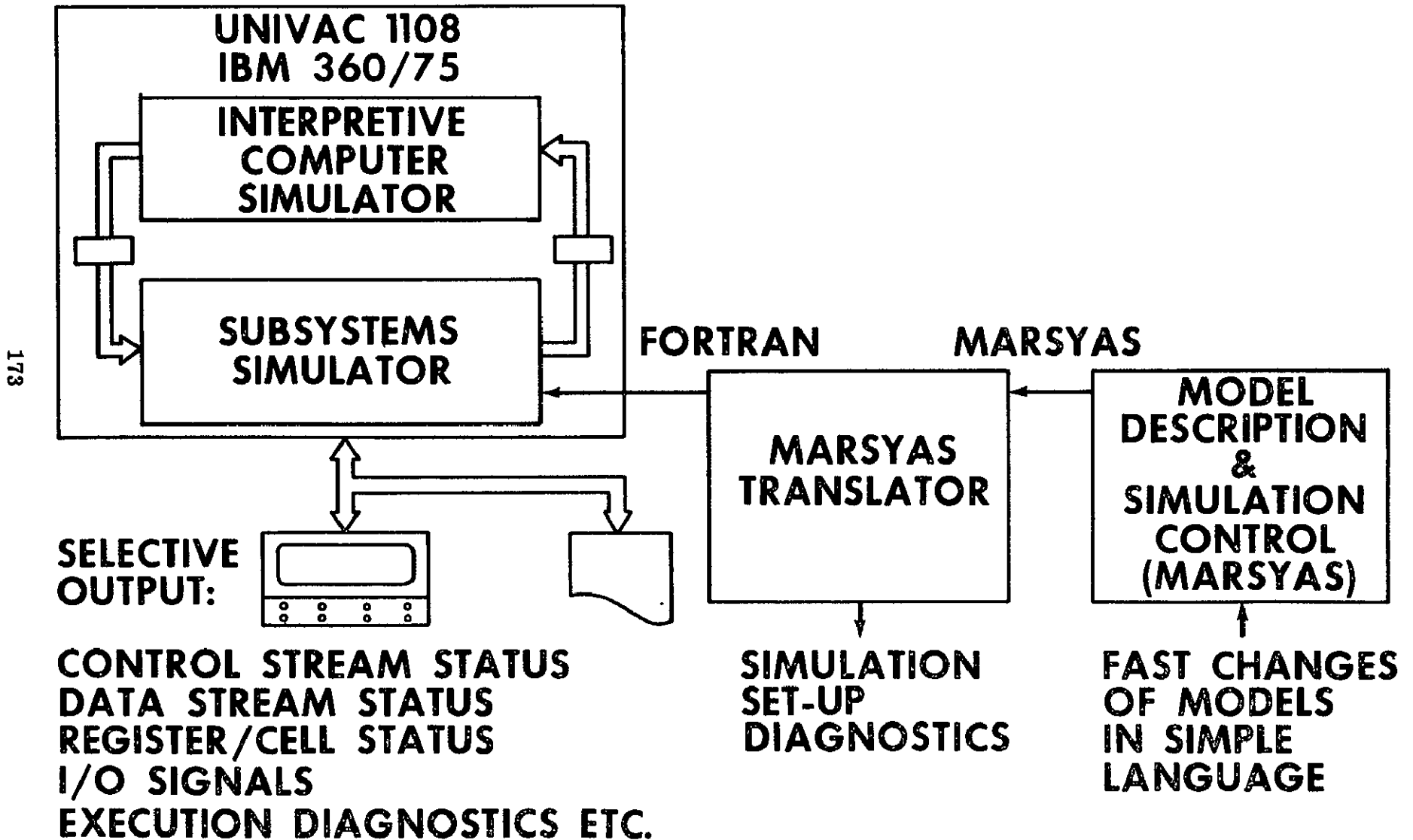
**PROBLEMS:**

- RUNNING TIME IS LONG FOR LARGE AND DETAIL MODELS
- ACCURATE NUMERICAL SOLUTION FOR MIXED CONTINUOUS/SAMPLED-DATA SYSTEMS

Integrated computer and subsystem simulation system.- In the integrated computer and subsystem simulation system, the interpretive computer simulator and the subsystem simulator run on a large computer like the UNIVAC 1108 or IBM 360/75 together under interactive control by the "software test engineer." Through a graphic display and keyboard he can control the simulation and call up register and memory cells to determine from their contents at particular times the control stream and data stream status of the flight programs. At the same time, he can observe the values of certain dynamic subsystem signals. The results can be either displayed on the CRT or printed.

The subsystem simulator uses a large number of models being described by differential and algebraic equations, transfer functions, and logical functions. These models could be described by a higher-level language like MARSYAS, which allows fast changes of models when the subsystem design changes and gives more extensive error diagnostics for the simulation setup. The MARSYAS- translator is written in FORTRAN and generates FORTRAN object code; therefore, it can be adapted quickly to any large digital computer with at least 32K words of core memory.

# INTEGRATED COMPUTER AND ENVIRONMENT SIMULATION SYSTEM

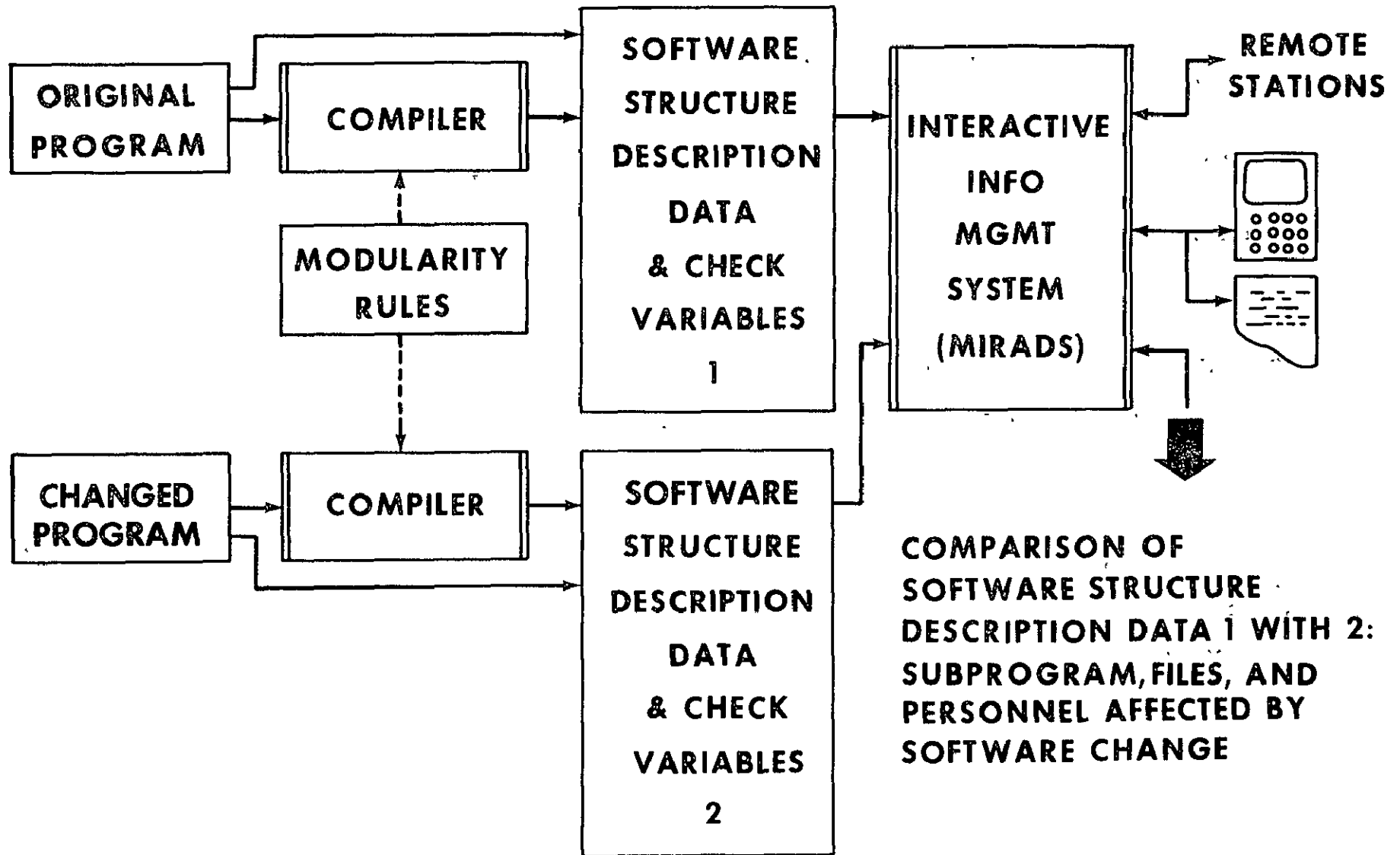


### Software Configuration Control Tools

Quick-look software change impact analysis.- It is assumed here that the software is described on a functional subprogram level by diagrams and tables as outlined in "Modular/Software Description" previously. These diagrams and tables, partly generated by the compiler, are stored as "software structure description data" into an interactive information management system such as MIRADS, which runs on a time-shared computer and can be accessed by programmers and managers via remote stations throughout the country. This would allow a rapid assessment of the effect of software changes and also would notify those programmers who work on programs being affected by a software change.

Detail software change control.- The quick-look analysis system would not replace the software change control systems that were used already in the APOLLO program but would augment their effectiveness.

# QUICK-LOOK SOFTWARE CHANGE IMPACT ANALYSIS



## DEVELOPMENT AND VERIFICATION PROCESS

### Software Development Schedule

The flight and ground software should go through the same phases in their development as the hardware does, as shown in the following slide. At the early design phase, trade-offs can be made between hardware and software. In order to insure software operational together with the hardware, an early design of the software is mandatory. It should be stressed here that the software development and verification tools have to be reliably operational much earlier; i.e., the compilers must be thoroughly tested and ready before programming starts, and the verification systems must be operational when debugging of the programs starts. This shows that the work on compiler generators, compilers, verification and simulation systems, and their extensive evaluation is needed immediately.

### Stages in Verification of Software

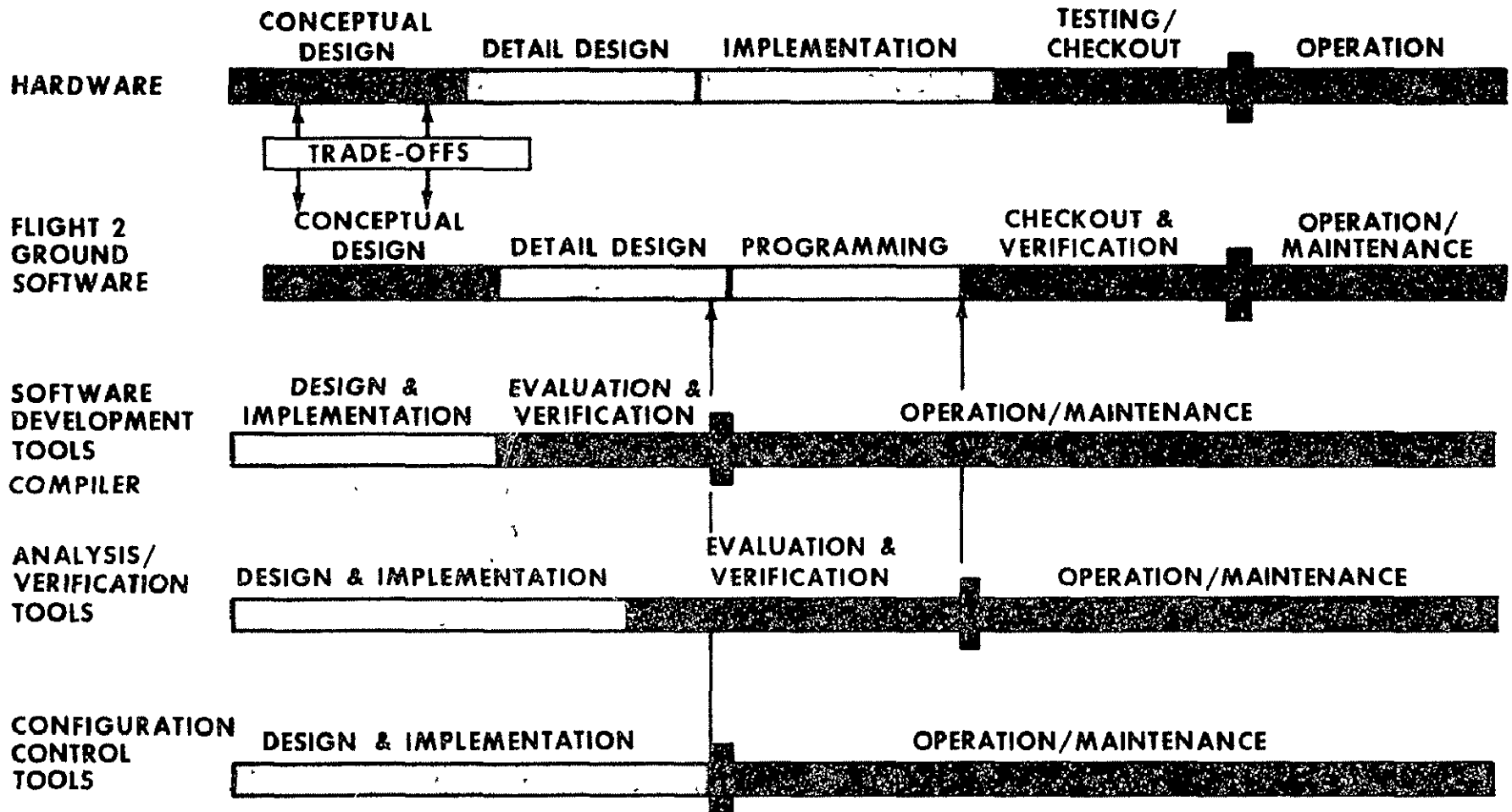
The verification of the software is accomplished in several stages according to a test plan:

1. The functions at all levels of the hierarchical design and an integrated system are checked.
2. The higher level compiler then diagnoses for errors which violate the semantics, syntax, and modularity structure.
3. The execution of the individual programs and their integration is simulated and traced on the Integrated Computer and Subsystems Simulation System.
4. The programs are then executed on an actual flight computer with the subsystem simulation on a separate large computer.
5. The actual programs are then executed and traced in real-time on the actual flight computer with hardware simulation of the computer environment (breadboard).



# SOFTWARE DEVELOPMENT SCHEDULE

177



4  
②

## CONCLUSIONS

The software development must go through the same phases as the hardware development and must go hand-in-hand with the hardware development in order to have the software ready in time and with less costs than for APOLLO. For the software development, several tools in the form of compilers, compiler generators, verification, and simulation systems are essential. These tools have to be made ready and perfected now since they have to be available once the Shuttle design is firmed up and programming for flight and ground computers starts. These tools still have certain deficiencies which have to be overcome by more vigorous research and development effort.

SYNCHRONOUS EXECUTIVE AND BUS CONTROL SOFTWARE FOR REDUNDANT DATA MANAGEMENT SYSTEMS

T. E. Daly

TRW Systems  
Redondo Beach, California

INITIALS

## 1.0 Introduction

The definition of executive and bus control software for the shuttle application must facilitate the generation of the software in a cost effective manner while at the same time satisfying the technical considerations imposed by the concept of the redundant data management system. Software reliability and ease of verification must be designed into the software. To achieve these objectives, it is necessary that the executive software have a simple organization and well defined behavior.

The utilization of a simple executive structure with no external interrupts and with predictable behavior reduces verification costs and improves software reliability. The executive approach advocated in this paper is referred to as a "Synchronous Executive Routine". The synchronous executive divides time into repetitive cycles. In order to facilitate management of the software, software modules are permanently assigned to various cycles. Bus transmission intervals are assigned to corresponding time intervals.

## SOFTWARE APPROACH

### OBJECTIVES

- ENHANCE SOFTWARE RELIABILITY
- REDUCE VERIFICATION & MODIFICATION COSTS

### CHARACTERISTICS OF APPROACH

- SIMPLIFY SOFTWARE ORGANIZATION
- CREATE WELL DEFINED BEHAVIOR
- ENCOURAGE MODULARITY & SOFTWARE SEPARATION BY MISSION PHASE

### METHOD OF ACHIEVING OBJECTIVES

- "SYNCHRONOUS EXECUTIVE ROUTINE"
- "SYNCHRONOUS BUS CONTROL ROUTINE"

## 2.0 Executive Requirements

The requirements that the executive must satisfy are shown in the figure. They include the conventional executive tasks of scheduling, providing program linkages, initiating I/O operations, and conditioning processing with directives received through the man/machine interface.

In addition, certain requirements are generated by the redundant nature of the data management system. These include the ability to initialize redundant machines through inter-computer communication and the ability to participate with a higher level controlling device to evaluate redundant machine performance. In order for redundant data management systems to redesignate a controlling computer in a transientless manner, it is sufficient that all redundant computers are synchronous to the minor cycle. The nominal portions of the data processing occur synchronously. However, non-nominal actions can occur that require additional data processing, such as failure diagnosis or shut-down after a power transient. Usage of a synchronous executive dictates an additional requirement to process unpredictable tasks within the synchronous structure.

# EXECUTIVE REQUIREMENTS

- SIMPLE DESIGN WITH WELL DEFINED BEHAVIOR
- SYNCHRONOUS DESIGN WITH ASYNCHRONOUS CAPABILITY ON DEMAND BASIS
- SCHEDULE SOFTWARE TASKS
  - PROVIDE LINKAGE BETWEEN PROGRAM ELEMENTS
  - SATISFY SUBSYSTEM DATA EXECUTION RATE REQUIREMENTS
  - CONTROL I/O OPERATIONS
  - PROVIDE RESCHEDULING CAPABILITY
- RESPOND TO MANUAL AND AUTOMATIC SELECTION OF ALTERNATE MODES OF SUBSYSTEM OPERATION
- PERFORM COMPUTER INITIALIZATION
- TRANSFER INITIALIZATION INFORMATION TO REDUNDANT COMPUTERS
- SUPPLY INFORMATION TO A SYSTEM CONTROL UNIT
- ASSURE THAT REDUNDANT COMPUTERS OPERATE IDENTICAL PROGRAMS SIMULTANEOUSLY

### 3.0 Executive Organization

The synchronous executive consists of a minor cycle and a major cycle which are constantly repeated with a fixed period. Program iteration rates are established by their position within these loops. The minor cycle represents the 50 times/second loop while the major cycle is slower with a 1 sec period. These fixed computational cycles eliminate the need for external interrupts. All subsystems are serviced in a sequential manner at rates established by the executive and bus control software. This manner of operation simplifies the verification process by eliminating test cases for interrupts occurring at various times in the cycle.

The basic executive cycle for the Space Shuttle is 20 msec. This repetitive time frame contains the minor cycle computations together with a single segment of the major cycle computations. It is the responsibility of the executive routine to schedule routines within these cycles so that necessary iteration rates are achieved and that the necessary interrelationships between modules are maintained.

The executive software is implemented with three components: a master controller, phase executives, and task lists. The program is divided according to mission phases with a separate phase executive and task list for each phase. The functions of these routines are described in the corresponding chart. The master controller is responsible for handling transitions between mission phases and for dealing with non-periodic or unpredictable events. This capability is superimposed on the synchronous structure in a non-interfering manner to provide a fast flexible real time response to non-periodic events.



# EXECUTIVE ORGANIZATION

## EXECUTIVE CHARACTERISTICS

PERIODIC STRUCTURE WITH MINOR LOOP ENTRY BY HARDWARE ASSIST  
NO EXTERNAL INTERRUPTS. SUBSYSTEM SAMPLING UNDER SOFTWARE CONTROL  
MINOR LOOP COMPUTATIONS OCCUR ON A REGULAR BASIS  
MAJOR LOOP SEGMENTS PERFORMED BETWEEN MINOR LOOP ENTRIES  
SCHEDULING OF MAJOR LOOP SEGMENTS IS EXECUTIVE FUNCTION (TABLE)

## EXECUTIVE COMPONENTS

MASTER CONTROLLER - HIGHEST LEVEL OF AUTHORITY, RESPONSIBLE FOR  
DEMAND PROCESSING AND MISSION PHASE TRANSITIONS  
TASK TABLES - TABLES, ORGANIZED BY MISSION PHASE, WHICH DEFINE THE  
CONTENTS OF THE MAJOR CYCLE SEGMENTS  
PHASE EXECUTIVES - LOCAL EXECUTIVES RESPONSIBLE FOR DECISIONS RELEVANT  
TO PARTICULAR MISSION PHASES

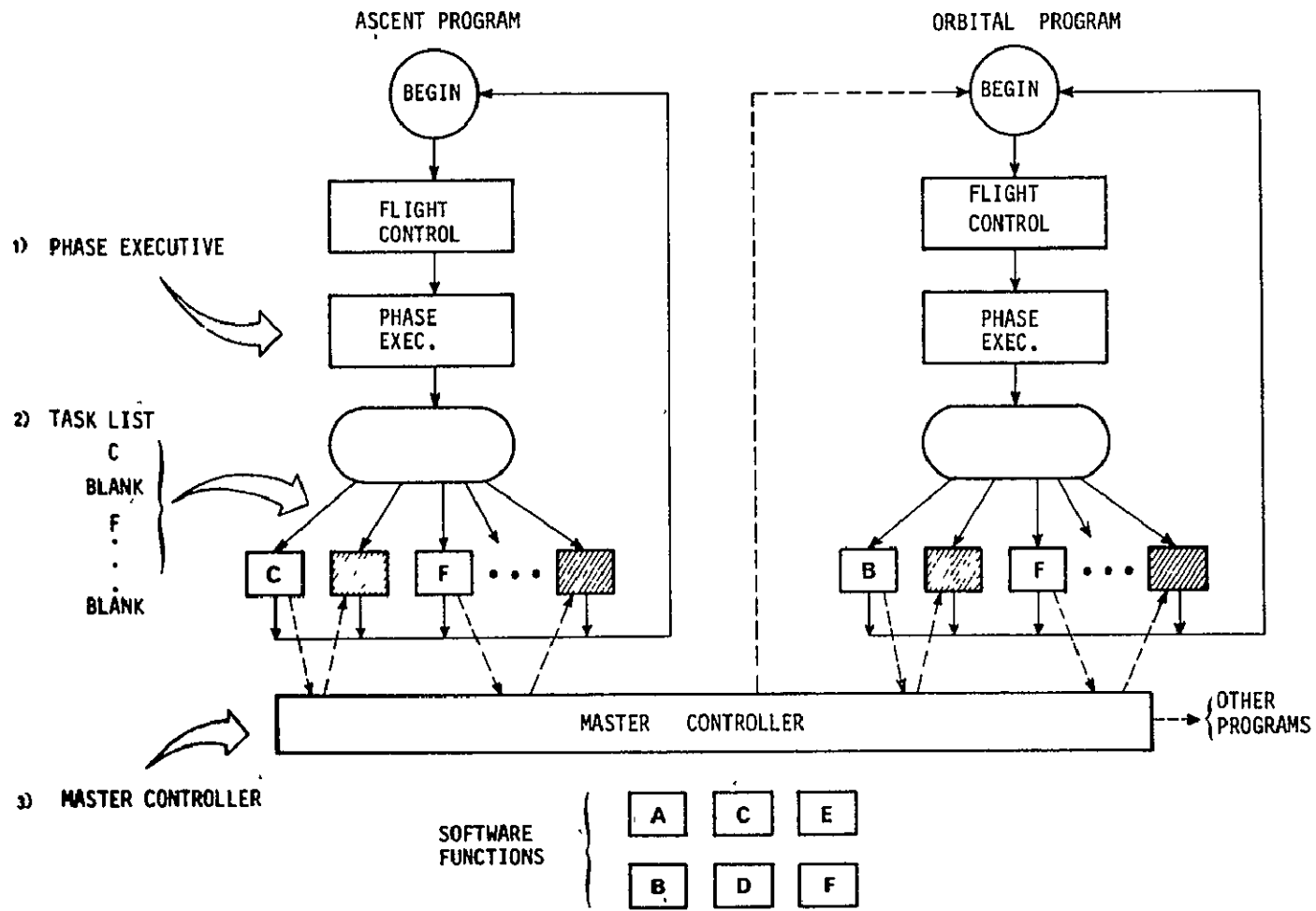
#### 4.0 Executive Structure

The executive schedules high speed computations (example: flight control) in the minor cycle and slower computations into the segments of the major cycle (shown in the diagram as modules A, B....F). The time constraint is that the minor cycle plus a major cycle segment must be completed within 20 msec. Major cycle modules are linked together by the task list for the appropriate phase.

Alternate major cycle segments are reserved for processing unpredictable events such as failure diagnosis, mass memory utilization, and mission planning activities. If, in the course of the regularly scheduled computations, it is determined that a need exists to schedule an unpredictable event, a flag is set. During the next reserved major cycle segment, the master controller detects this situation and schedules the appropriate software. This processing is accomplished without interfering with the periodic functions scheduled within the synchronous framework. Thus, tasks such as reconfiguration and mission planning can be performed without altering the real time execution of the basic program which contributes to maintaining synchronization between redundant computers.

The time reserved for these non-periodic functions is not wasted by the structure since any organization will have to provide processing time for such tasks. In the flight program this time will be released for background tasks such as continuous self test when not needed. This background task will be superseded as required for demand events. The capability of providing demand processing in a non-interfering manner can be taken advantage of during software checkout or hardware/software integration by building monitoring routines into these slots.

# SOFTWARE STRUCTURE



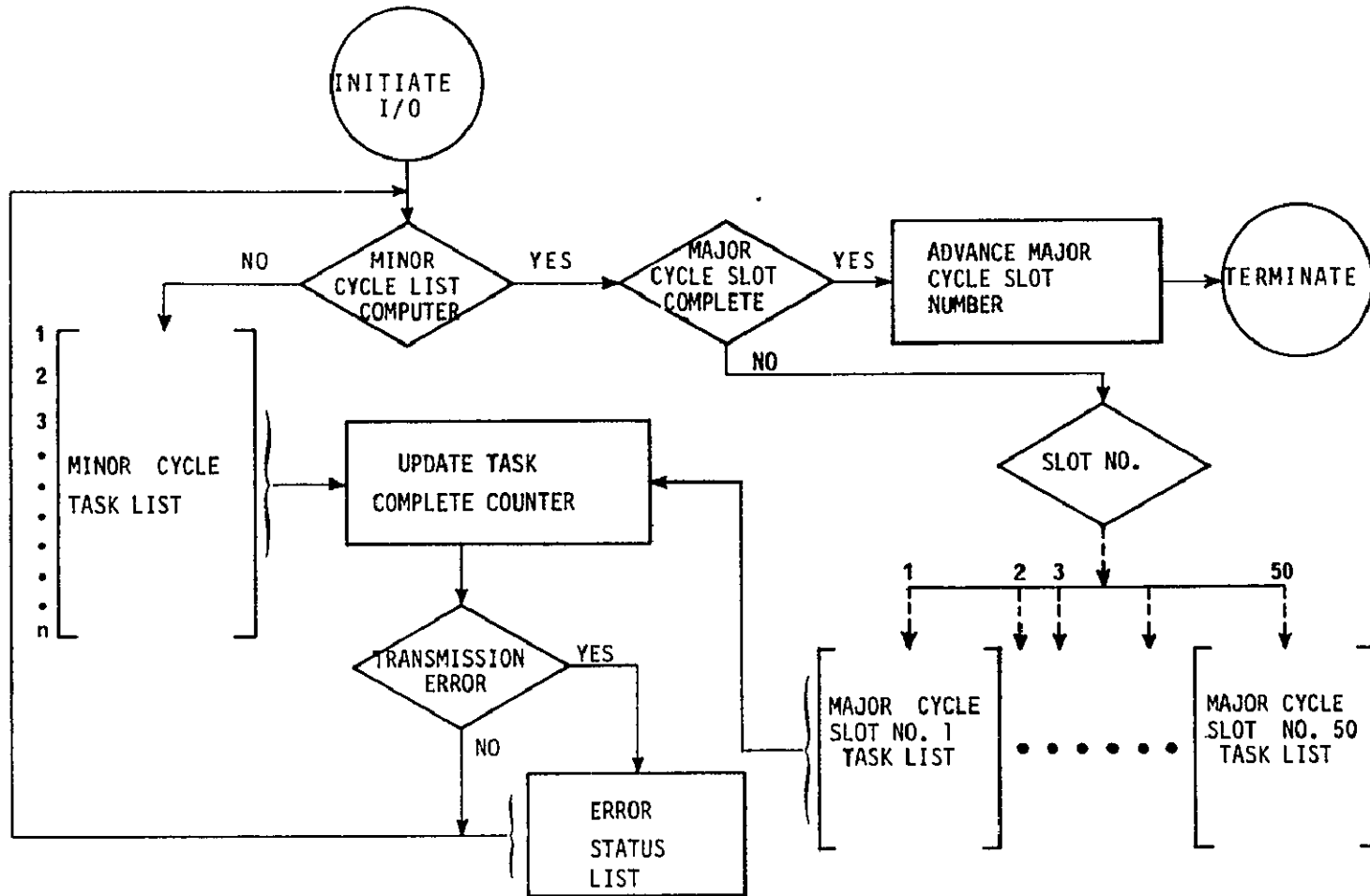
## 5.0 Bus Control Software

The bus control software consists of a list of instructions for the I/O channel to execute. Each entry in this list defines a transmission between the computer and a subsystem. Each item in the list contains the address of the subsystem, the function to be performed, the direction of transmission, the number of buses involved, the address of the data, and the number of words involved. Exceptions to this format are special commands to the channel which tell it to terminate or jump to a different list.

In order for I/O processes to coincide with the needs of the internal computations, it is necessary that the bus control list have a structure similar to the synchronous executive organization. As shown in the flowchart, a bus control list is established for the minor cycle, and separate lists are formed for each segment of the major cycle. The transmissions in these lists correspond to the organization of the computations within the executive structure. Upon initiation of the I/O channel, the minor cycle list is executed in addition to 1 major cycle segment list. At the next initiation (1 minor cycle later) the minor list is repeated and the next major cycle segment list is executed.

As shown in the flowchart, transmission errors associated with any tasks are recorded and an error message is placed in the computer memory. Since the I/O channel runs independently of the computer, it is necessary to assure that I/O transactions are completed prior to using data. When the I/O channel completes a list, a flag is set. In addition, a counter which designates which transaction is in process is available to the program. These aids, coupled with initiating channel activity at the appropriate times, facilitate the necessary coordination between internal computations and I/O transactions.

# BUS CONTROL SOFTWARE ORGANIZATION



HAL, A COMPILER LANGUAGE FOR SHUTTLE

Fred H. Martin

Intermetrics, Incorporated  
Cambridge, Massachusetts

191

1 N71 - 35061

PRECEDING PAGE BLANK NOT FILMED

## PREFACE

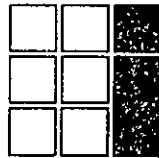
The HAL Programming Language has been developed by the staff of Intermetrics, Inc.

HAL accomplishes three significant objectives: (1) increased readability, through the use of a natural two-dimensional mathematical format; (2) increased reliability, by providing for selective recognition of common data and subroutines, and by incorporating specific data-protect features; (3) real-time control facility, by including a comprehensive set of real-time control commands and signal conditions. Although HAL is designed primarily for programming on-board computers, it is general enough to meet nearly all the needs in the production, verification and support of aerospace, and other real-time applications.

The design of HAL exhibits a number of influences, the greatest being the syntax of PL/I and ALGOL, and the two-dimensional format of MAC/360, a language developed at M.I.T. With respect to the latter, Intermetrics wishes to acknowledge the fundamental contribution, to the concept and implementation of MAC, made by Dr. J. Halcombe Laning.

## PREFACE

- \* HAL developed by Intermetrics, Inc.
  - \* Language design
  - \* Compiler design and implementation
  
- \* Significant Objectives
  - \* Increased readability
  - \* Increased reliability
  - \* Real-time control
  
- \* Capabilities
  - \* Primarily designed for on-board computer
  - \* General enough for:
    - ground support and verification
    - other real-time applications



SLIDE 1

**INTERMETRICS**



## SHUTTLE LANGUAGE REQUIREMENTS

As a result of an extensive language requirements analysis, Intermetrics designed HAL in order to satisfy the following requirements:

1. The principal application of HAL is for the development of manned spaceflight computer software for the 1972-1980 period and this includes Shuttle and Space Station applications. (Initial orientation will be toward the Shuttle system.)
2. Software applications should include: (a) navigation, guidance, targeting and general mission programming; (b) vehicle control and stabilization; (c) operating systems; (d) on-board checkout and system monitoring; (e) data management; (f) communications and displays; (g) compiler and support software.
3. The language and compiler should be designed for a wide range of flight computer systems and should be capable of supporting simplex configurations as well as advanced multi-computer and multi-processor computer systems.
4. The language should be machine-independent with a minimum of exceptions.
5. The language and compiler must contain specific features to aid in achieving high software reliability. The design shall:
  - a) strive toward clarity and readability in the language;
  - b) enforce programming standards and conventions;
  - c) perform extensive automatic checking;
6. The output format of the language should strive toward presenting data types, attributes and operations in an unambiguous way. An equation should look like an equation. A character string (or text) should be easily differentiated from a vector, or array. The compiler will annotate the output listing accordingly.
7. The language should be oriented toward a general class of technical personnel involved in manned spaceflight projects, not solely highly trained programmers.

## SHUTTLE LANGUAGE REQUIREMENTS

### \* Software Applications

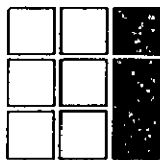
- \* Navigation, guidance, targeting
- \* Vehicle control
- \* Operating systems
- \* Data management
- \* Communications and displays
- \* Support software
- \* On-board checkout and monitor

### \* Computer Environment

- \* Wide range of computers (Flight and Ground)
- \* Fixed- and floating-point
- \* Simplex, multi-computer, multi-processor

### \* Language Characteristics

- \* Clarity and readability
- \* Enforcement of standards and conventions
- \* Extensive automatic checking (compile- and run-time)
- \* Facilitate software management
- \* Promote modularization



**INTERMETRICS**

SLIDE 2

CHRONOLOGY OF SOFTWARE DEVELOPMENT  
(PROGRAM DOCUMENTATION)

The format of a space programming language; i.e., its source input and printed appearance, should be designed to achieve maximum readability, ease in transfer of knowledge and understanding, and it should provide a basis for program documentation. Most existing higher order programming languages strive towards these goals as secondary objectives, with program composition as the first priority. Certainly any new programming language must be easy to use but composition is only the "front-end" of a long process to develop reliable space software. In perspective, stronger emphasis must be placed on software control techniques and accompanying documentation.

With reference to Slide 3, the important point is that the anticipated operational life-span of software will far exceed in time and effort that taken to generate the programs. Consequently, the language should emphasize readability and clarity for maintaining the software rather than emphasizing ease in program preparation.

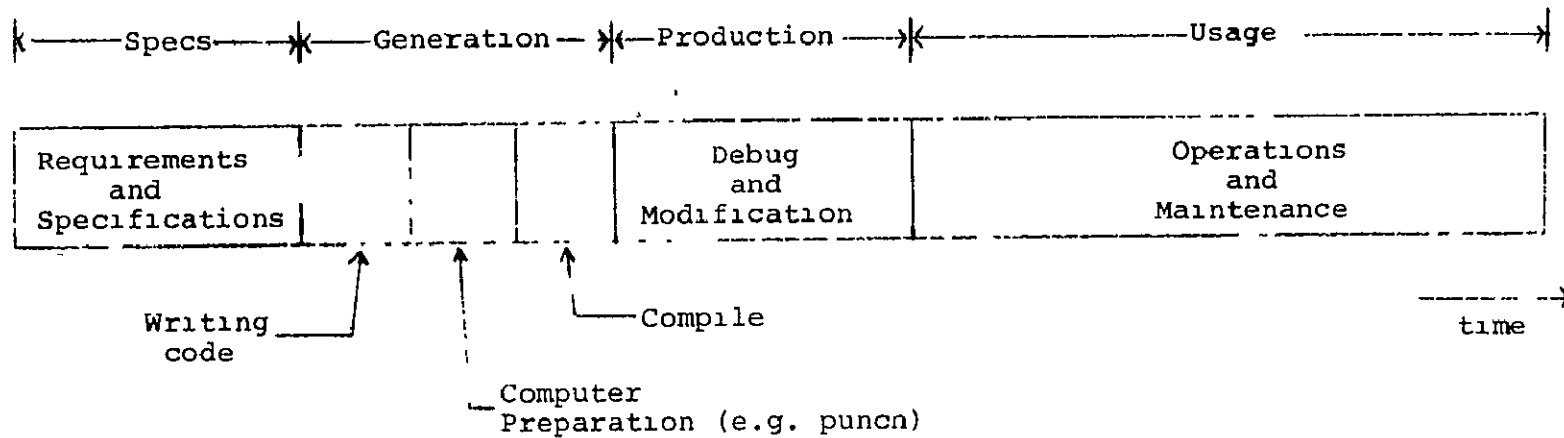
The printed appearance of a language greatly influences its usefulness as a communications medium. Most of today's higher order languages, being fundamentally compatible with card punches or data terminals, are written in single line format. A multi-line, or two dimensional, format can bring a programming language closer to being natural in expression and mathematical form. Equations, whether scalar or vector-matrix, look like equations and promote understanding among programmers and technical managers. Subscript and superscript lines, in which exponents and subscripts appear in standard mathematical notation, provide an opportunity to make data forms self-evident. For example:

$$\bar{V}DOT = -\bar{G} + K_3^2 \bar{M} \bar{C}$$

is obviously a differential equation involving the vector data types  $\bar{V}DOT$ ,  $\bar{G}$ ,  $\bar{C}$ ; the matrix data type  $\bar{M}$ ; and the square of the subscripted scalar  $K_3$ .

The appeal of the two-dimensional format is not simply esthetic; the conventional mathematical notation for input and output will be decisive factors in promoting software reliability. Coded operations will become visible to managers and supervisors who have ultimate project responsibility. The language can provide the basis of communication for a broad spectrum of engineers, scientists and technicians contributing to the shuttle program.

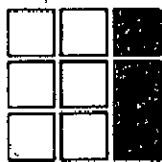
## CHRONOLOGY OF SOFTWARE DEVELOPMENT



197

### Some Observations

1. The writing of code is closely tied to the specifications.
2. The time required for computer preparation is small compared to the program life.
3. A lengthy period of debug and modification must be provided.
4. Period of program usage extends many times that of program generation.
5. Many more people will use a program than generated it.



### Conclusion

The computer language should promote understanding of the software. The listing should tend toward self-documentation.

## **INTERMETRICS**

## SALIENT FEATURES OF HAL

In order to meet the stated language requirements HAL includes the following features:

### 1. Two-dimensional input-output and annotation of variables

An equation which involves exponents and subscripts may be written, for example, as

$$C_I = (X A_J^2 + Y B_K^2)^{3/2}$$

(HAL also provides for an optional one-line format.)

In addition, in an effort to increase program reliability and promote HAL as a more direct communications medium between specifications and code, the HAL program listing is annotated with special marks. Vectors, matrices and arrays of data are instantly recognized by bars, stars and brackets. Thus, a vector becomes  $\bar{V}$ , a matrix  $\bar{M}$ , and an array [A]. Further, bit strings appear with a dot, i.e.,  $\dot{B}$  and character strings with a comma,  $\dot{C}$ . With these special marks as aids, the source listing is more easily understood and serves as an important step toward self-documentation.

### 2. Complete vector-matrix arithmetic

HAL can be used directly as a "vector-matrix" language in implementing large portions of both on-board and support software. For example, a simplified equation of motion might appear<sup>†</sup> as:

$$\bar{A} = \dot{B} \bar{A} \dot{C} \dot{C}; \quad \bar{G} = -MU \text{ UNIT}(\bar{R})/\bar{R} \cdot \bar{R}; \quad \bar{V} \text{DOT} = \bar{A} + \bar{G}; \quad \bar{R} \text{DOT} = \bar{V};$$

### 3. Bit and character manipulations

For handling I/O devices, communications (up/down telemetry and text messages), and support programs (executive, compiler, etc.) HAL provides for the necessary "bit-pushing" and character manipulations. Individual bits may be treated as Boolean quantities or grouped together in strings. Strings of bits and/or characters can be concatenated, deconcatenated, and converted to other data types.

### 4. Data arrays and structures

In anticipation of the need to process large volumes of measurement and experimental data (on advanced Shuttle or Space Station missions) and to facilitate general file handling or parallel computations, HAL provides for organizations of data. Multi-dimensional arrays of any single type can be formulated, partitioned, and used in expressions. A hierarchical organization called a structure (similar to the title, chapter, section, paragraph organization of a book) can be declared, in which related data of different types may be stored and retrieved as a unit or by individual reference.

---

<sup>†</sup> Where B transforms acceleration from spacecraft to reference coordinates.

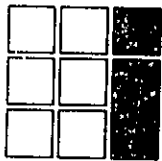
## SALIENT FEATURES OF HAL

### Capability

1. Two-dimensional Input-Output  
Annotation of variables
2. Complete vector-matrix arithmetic
3. Data array and structure handling
4. Bit and character manipulations
5. Real-time control statements
6. Data-Pool (COMPOOL), controlled  
sharing and name scope

### Requirement

- \* Increased readability
- : Targeting, guidance and control
- \* Data management
- \* Systems, communications and I/O
- \* Command and control
- \* Increased reliability



SLIDL 4

## **INTERMETRICS**

#### 5. Real-time control

HAL is a real-time control language; that is, certain defined blocks of code called programs and tasks can be scheduled based on time or the occurrence of anticipated events. These events may include external interrupts, specific data conditions, and programmer-defined software signals. Undesirable or unexpected events, such as abnormal conditions, may be handled by instructions which enable the programmer to specify appropriate action.

HAL's real-time control features permit the initiation and scheduling of a number of active tasks. This will be a necessity in order to cope with the computational tasks required of the Shuttle's integrated avionics system.

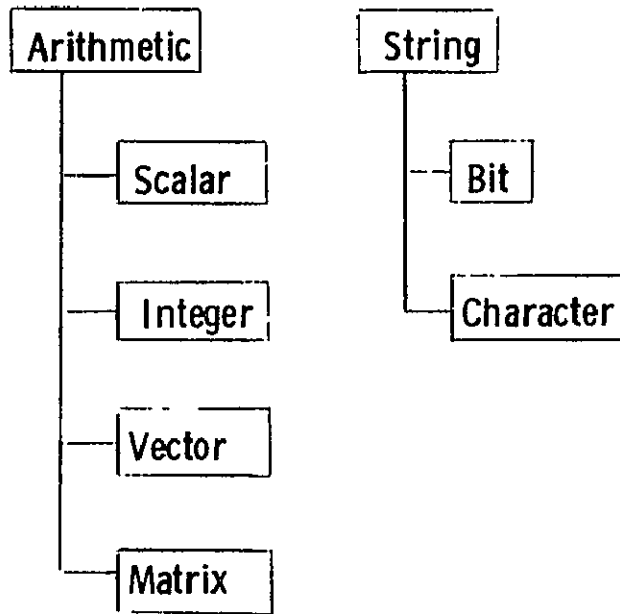
#### 6. Controlled data sharing

Program reliability is enhanced when a software system can create effective isolation for various subsections of code as well as maintain and control commonly used data. HAL is a block-oriented language in that a block of code can be established with locally defined variables that cannot be altered by sections of program located outside the block. Independent programs can be compiled and run together with communication among the programs permitted through a centrally managed and highly visible data pool. For a real-time environment, HAL couples these precautions with a locking mechanism which can protect, by programmer directive, a block from being entered, a task from being initiated, and even an individual variable from being written into, until the lock is removed.

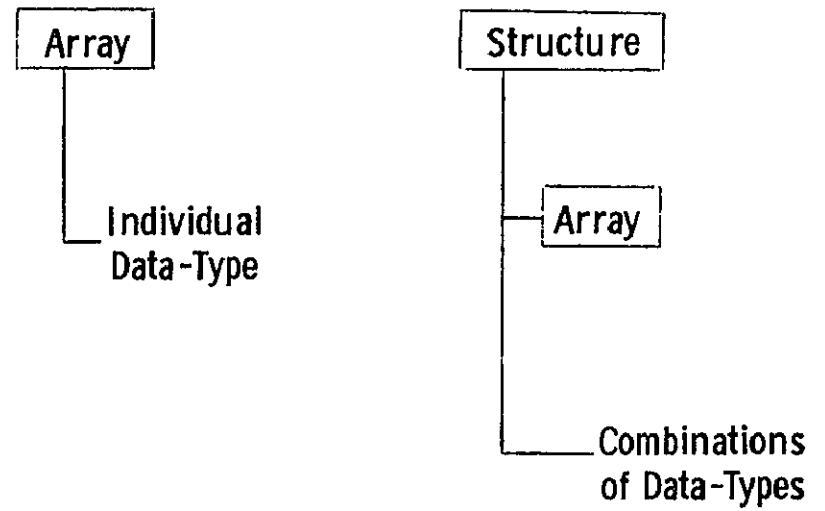
These measures cannot in themselves ensure total software reliability but HAL does offer the tools by which many anticipated problems, especially those prevalent in real-time control, can be isolated and solved.

# HAL Data Types and Organizations

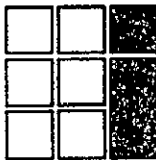
## Types



## Organizations



201



**INTERMETRICS**

SLIDE 5



## HAL PROGRAM ORGANIZATION

In order to accommodate all Shuttle programs in a single computer, or substantial portions in distributed computers, it is imperative that programs be isolated from one another except at controlled and visible interfaces. This isolation should prevent the unrestricted access of common data and the arbitrary transfer of control to any location in the instruction logic.

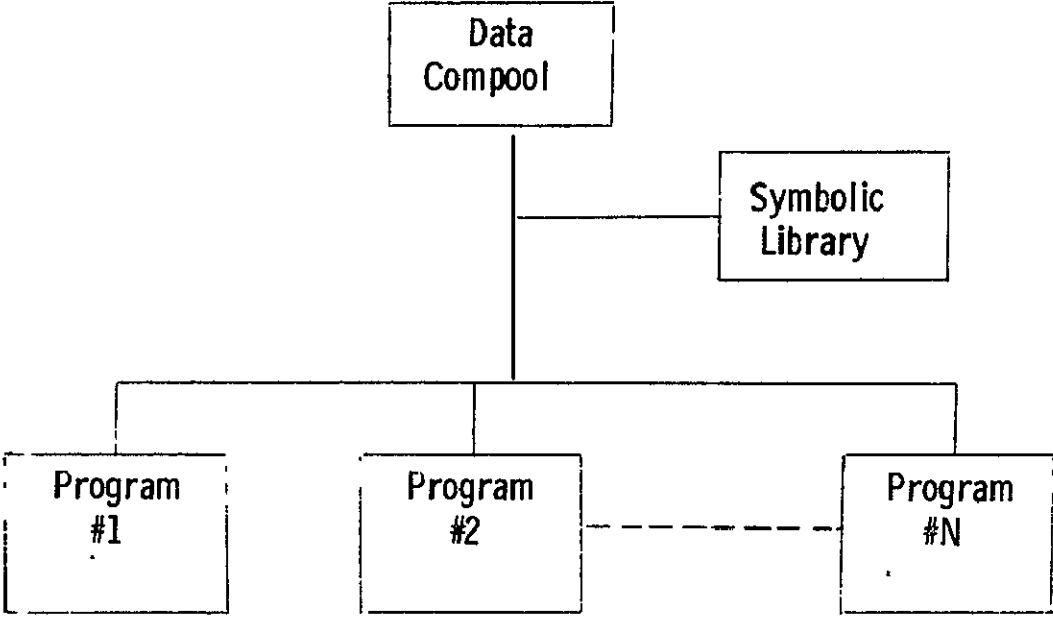
Software techniques now exist which allow many programs, designed to do various related and unrelated functions, to be written and incorporated in a single computer without conflict. The apprehension that the Shuttle DMS might be a bigger and more complicated Apollo-type effort with even more erasable conflicts and control interferences is blunted by the introduction of effective software modularity through language and compiler.

Slide 6 illustrates the HAL program organization. The individual numbered programs represent independently compilable units. Thus, for example, Program #1 might be rendezvous navigation, Program #2 - autopilots, Program #3 - environmental system monitoring. Independent compilation permits divergent groups of people to contribute to the whole and yet progress at varied paces with measures of local management control.

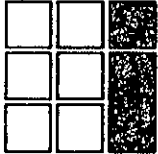
The communication between programs is provided through a common data pool (COMPOOL). The COMPOOL is a centrally defined and centrally maintained group of definitions. Variable names and location labels in the COMPOOL are potentially known to all programs and, in fact, provide the only means of communication between programs.

Thus, for the Shuttle, the many tasks can be apportioned into programs which are managerially or functionally convenient. Information interfaces among programs then become visible at the COMPOOL level and can be monitored with respect to definition and usage by a central authority.

HAL PROGRAM ORGANIZATION



203



SLIDE 6

**INTERMETRICS**

### BLOCKS OF CODE (NAME SCOPE)

For the purposes here, tasks, procedures and functions may be considered as subroutines (or blocks). As stated, names defined in the COMPOOL are potentially known in every program. Names defined at the program-level are potentially known within all included (or nested) subroutines, and so on. The region in which a name is known is referred to as its scope. Names are only potentially known because any particular name can be declared again in an inner block and then its scope would become all the nested blocks within this block. The example on Slide 7 may help to illustrate these principles:

Two desirable effects of the scope rules are:

1. Common data must be declared at the highest level and only once. This contributes to more direct management control and better visibility.
2. Local variables may be defined within inner blocks and remain unaffected by outside definitions. For example, a programmer declaring X in procedure CHARLIE (Slide 7) need not fear that any other program will overwrite his quantity. That is, this particular X is not addressable from outside this block. In fact, the X in GRAB must refer to different memory cells.

For the Shuttle application, a name scope, or block-oriented, language means that many programs and subsections of programs (i.e., subroutines) can "live" in the same computer, isolated, and unaware of each other, incapable of writing-over or otherwise interfering with variables or locations that are not mutually defined.

BLOCKS OF CODE (NAME SCOPE)

ABLE:

```
PROGRAM;  
  DECLARE VECTOR (5) A,B,C;  
   $\bar{A} = \bar{B} + \bar{C};$   
  :  
  :  
  BAKER:  
    TASK;  
    DECLARE A INTEGER;  
    CHARLIE:  
      PROCEDURE;  
      DECLARE X;  
      DECLARE A BIT (10);  
      :  
      :  
      END CHARLIE;  
    END BAKER;  
  GRAB:  
    PROCEDURE;  
    DECLARE X VECTOR (4);  
    :  
    :  
    END GRAB;  
END ABLE;
```

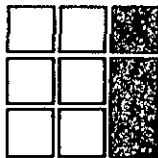
→ A,B,C are vectors (5)

→ B,C are vectors (5)  
A is now an integer

→ B,C are vectors (5)  
A is now a bit string  
X is a scalar

→ A,B,C are vectors (5)  
X is a vector (4)

205



SLIDE 7

**INTERMETRICS**

## CONTROL OF SHARED DATA

In order to illustrate the problems, in a general way, that can arise in sharing data, consider the examples shown in Slide 8.

In both examples TASK B interrupts TASK A during the execution of a statement. The interruption may be caused by a hardware or software interrupt or by a "job,swap" based on priority. In Example 1, presume that the interruption occurred while the matrix  $N$  was being read. When TASK A resumes, the computation of  $M$  will continue using some "old"  $N$  data and the "new"  $N$  data assigned in TASK B. In order to prevent this conflict, initiation of TASK B would have to be stalled until the reading of  $N$  in TASK A is completed.

In Example 2, presume that the interruption occurs just after the current value of  $Y$  is loaded into the accumulator. When TASK A resumes, the "old" value of  $Y$  (i.e., not reflecting the update of  $Y$  in TASK B) is restored into the accumulator,  $X$  is subtracted and the result assigned to  $Y$ . In order to prevent this conflict, the initiation of TASK B would have to be stalled until the value of  $Y$  is updated in TASK A.

The approach taken, in HAL, towards solving the problems represented above, is to confine the read and write accesses of shared variables to identified UPDATE blocks. Consider Example 1 in Slide 8 and suppose that the statements in question (in TASKS A and B) were enclosed within UPDATE blocks.

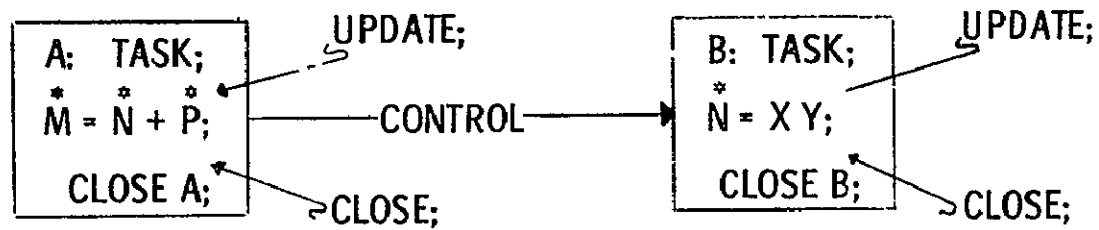
In TASK A a read-lock is established for  $N$ , because it will be read only. After the interruption, a write-lock is established for  $N$  and TASK B proceeds toward completion using copy-data for  $N$  rather than active data. At the end of the update-block in TASK B, the process stalls because of the read-lock imposed in TASK A. As a result, TASK A is allowed to continue with consistent "old"  $N$  data. After completion of TASK A, a copy-cycle is effected in TASK B and  $N$  is updated. All conflicts are eliminated.

The use of an UPDATE block is not a simple solution to the data sharing problem and presumes a sophisticated compiler; and yet the goal is worth the effort.

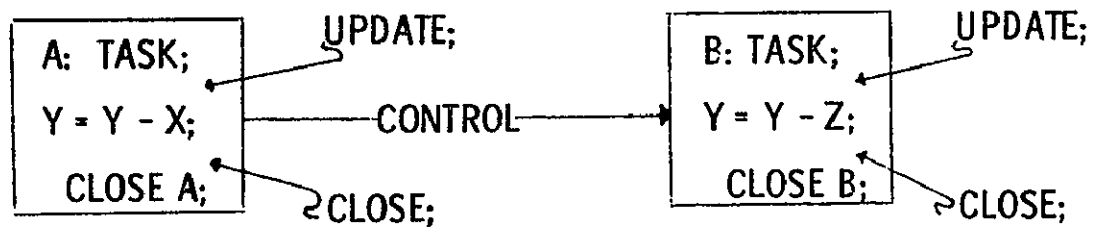
For the Shuttle, data sharing will be a necessity. A unified approach, through a compiler, as outlined above, will permit safe operation in multi-program and even multi-processor environments.

## CONTROL OF SHARED DATA

### EXAMPLE 1: READ AND WRITE CONFLICTS

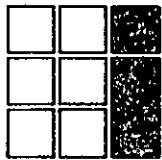


### EXAMPLE 2: UPDATE CONFLICTS



### NOTES:

1. B "INTERRUPTS" A IN BOTH CASES
2. #1 TASK A RESUMES USING OLD AND NEW VALUES FOR N\*
3. #2 TASK A RESUMES "CLOBBERING" THE VALUE FOR Y SET BY TASK B



**INTERMETRICS**

SLIDE 8

HAL CODE CAN LOOK LIKE THE SPECIFICATION

HAL exhibits full scalar-vector-matrix capabilities; vectors and matrices of arbitrary size may be declared. As part of the HAL syntax, vector-matrix operations include: inner and outer products, cross products, transpose and inverse, matrix-vector products, etc. By raising exponents onto an exponent line and not requiring a special symbol for multiplication (e.g., the FORTRAN \*); HAL arithmetic code can look remarkably like a written specification. Consider Slide 9; a set of rendezvous navigation equations has been reproduced from the Apollo GSOP Specification document (MIT). This set incorporates the measurement data,  $\delta Q$ , and updates state and error covariance information. HAL code parallels the specification almost line-for-line with few formalisms. Note that the vectors, matrices and scalars are apparent and their marks aid in understanding the programmer's intent. The adjacent vectors  $\underline{\Omega}$  and  $\bar{Z}$ , in the last line, imply an outer product.

Some further examples of arithmetic expressions are:

	<u>MATHEMATICAL NOTATION</u>	<u>HAL EXPRESSION</u>
1.	$ab$	$A B$
2.	$a(-b)$	$A(-B)$ or $-A B$
3.	$-(a + b)$	$-(A + B)$
4.	$a^{x+2}$	$A^{X+2}$
5.	$a^{x+2} c$	$A^{X+2} C$
6.	$ab/cd$	$A B/C D$
7.	$(\frac{a+b}{c})^{2.5}$	$((A+B)/C)^{2.5}$
8.	$1 + \frac{a}{1 + \frac{b}{2.7 + c}}$	$A / (1 + B / (2.7 + C))$
9.	$(\underline{v}^T \underline{y}) M^{-1} (\underline{v} + \underline{y})$	$(\bar{V} \cdot \bar{Y}) M^{*-1} (\bar{V} + \bar{Y})$
10.	$a (\underline{y} \underline{v}^T)^T (\underline{v} \times \underline{w})$	$A (\bar{Y} \bar{V})^T (\bar{V} * \bar{W})$

EXAMPLES OF ARITHMETIC OPERATIONS

(From Apollo Navigation Equations)

HAL

$$\bar{Z} = \bar{W}^{*T} \bar{B};$$

$$OMEGA = \bar{Z} \bar{W}^{*T} / (ZMAG^2 + ALPHA^2);$$

$$DELX = OMEGA DELQ;$$

$$\bar{X} = \bar{X}' + DELX;$$

$$F = 1 + (ALPHA^2 / (ZMAG^2 + ALPHA^2))^{1/2};$$

$$\bar{W}^* = \bar{W}' - OMEGA \bar{Z} / F;$$

GSOP Specification

$$\underline{z} = W'^T \underline{b}$$

$$\underline{\omega}^T = \frac{1}{z^2 + \alpha^2} \underline{z}^T W'^T$$

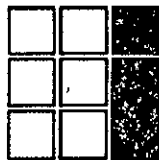
$$\delta \underline{X} = \underline{\omega} \delta Q$$

$$\underline{X} = \underline{X}' + \delta \underline{X}$$

$$W = W' - \frac{\underline{\omega} \underline{z}^T}{1 + \sqrt{\frac{\alpha^2}{z^2 + \alpha^2}}}$$

where  $\underline{b} \equiv$  geometry vector  
 $W \equiv$  square root of covariance  
 $\alpha^2 \equiv$  measurement variance  
 $X \equiv$  state vector

209



**INTERMETRICS**

SLIDE 9



## CONTROL, LOGIC AND COMPUTATION

Slide 10 illustrates the applicability of HAL in implementing a time critical routine. The example selected is that of cross product steering of the Apollo Command and Service Module. When XSTEER is entered TGO, the time-to-go to engine cut-off, is compared with 4 seconds. If TGO satisfies the inequality then all the statements between DO and END are executed. That is, the vehicle command rate in navigation-base coordinates is zeroed, and an indicator switch SW is turned off. (Note that the "over-dot" indicates a bit string; in this case, a Boolean, or flag.) A routine ENGINE\_OFF is then scheduled (via a HAL real-time control statement) to be executed at a proper time hence, with priority 20. E OFF\_ID identifies this particular scheduled job, uniquely, so that it might be referenced at a future time, perhaps to terminate it. If  $TGO > 4$  seconds then none of the above instructions would have been executed. Instead the steering command rate  $\bar{\Omega}_C$  would be computed based on the cross-product of the "velocity-to-go" ( $\bar{V}_G$ ) and the acceleration related term ( $\bar{D}_{ELM}$ ). Finally this rate is transformed from reference to stable member coordinates (REFSMMAT) and then from stable member to navigation base coordinates ( $\bar{S}_{MNB}$ ) for application to the autopilot within another routine.

Although the mnemonics may be unfamiliar to the reader, it is easy to see that HAL's expressiveness makes an important contribution toward self-documentation.

CONTROL, LOGIC AND COMPUTATION  
(Cross product steering of Apollo vehicle)

Involves scalars, 3-d vectors, 3x3 matrices, "Booleans"

```
XSTEER:  IF TGO < 4 THEN DO;
           $\bar{\Omega}_{CNB} = 0;$ 
           $\dot{S}W = OFF;$ 
          SCHEDULE ENGINE_OFF AT (TIME+TGO),
          PRIORITY (20), E_OFF_ID;
          GO TO START;
          END;
```

$\bar{D}ELM = C \bar{B} DELT - \bar{D}ELV;$

$\bar{\Omega}_C = K(\bar{V}G * \bar{D}ELM) / (ABVAL(\bar{V}G) ABVAL(\bar{D}ELM));$

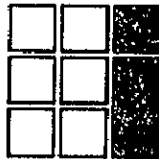
$\bar{\Omega}_{CNB} = \dot{S}MNB \dot{R}EFSMMAT \bar{\Omega}_C;$

GO TO START;

where TGO  $\equiv$  "time-to-go"

$\bar{V}G \equiv$  "velocity-to-be-gained"

$\bar{\Omega}_C \equiv$  rate command



SLIDE 10

**INTERMETRICS**

## SUBSCRIPTS AND PARTITIONS

The elements of vectors, matrices, bit and character strings, arrays and structures may be referenced by appropriate subscripting.

The first component of a vector or a one-dimensional array, is given the subscript 1, the second 2, etc. up to the total number of elements. Thus for a 9 element vector, i.e.,

```
DECLARE V VECTOR(9);
```

the components may be written as,

$$V_1 \ V_2 \ V_3 \ \dots \ V_9.$$

A matrix or two-dimensional array may be thought of as being composed of horizontal rows and vertical columns. The first of the two subscripts refers to the row number, the second to the column number. For instance, a matrix of two rows and three columns would require the declaration

```
DECLARE B MATRIX(2,3);
```

and the elements could be referred to by writing:

$$B_{1,1} \ B_{1,2} \ B_{1,3} \ B_{2,1} \ B_{2,2} \ B_{2,3}$$

A range of subscripts may also be selected and used for partitioning; e.g.,  $\bar{V}_1$  TO 4 partitions a larger vector  $\bar{V}$  and selects the first four components to form a vector. The same basic approach is used to subscript bit and character strings; i.e.,  $B_{16}$  selects the 16<sup>th</sup> bit of a string, and  $\bar{C}_5$  TO 10 selects characters 5,6,7,8,9,10 from the original character string.

Slide 11 illustrates the partitioning of a covariance matrix in order to compute rms errors after a landmark measurement and initialization prior to the next measurement.

EXAMPLES OF MATRIX PARTITIONING

Given: 9x9 covariance matrix E of errors in position, velocity and landmark location. That is,

$${}^*E = \begin{bmatrix} {}^*E_{p-p} & {}^*E_{p-v} & {}^*E_{p-l} \\ {}^*E_{v-p} & {}^*E_{v-v} & {}^*E_{v-l} \\ {}^*E_{l-p} & {}^*E_{l-v} & {}^*E_{l-l} \end{bmatrix}$$

1. RMS Errors

$$\text{RMS\_POS} = \text{SQRT}(\text{TRACE}({}^*E_{1 \text{ TO } 3, 1 \text{ TO } 3}));$$

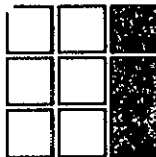
$$\text{RMS\_VEL} = \text{SQRT}(\text{TRACE}({}^*E_{4 \text{ TO } 6, 4 \text{ TO } 6}));$$

2. Initialize  ${}^*E$  for new landmark

$${}^*E_{1 \text{ TO } 6, 7 \text{ TO } 9} = 0;$$

$${}^*E_{7 \text{ TO } 9, 1 \text{ TO } 6} = 0;$$

$${}^*E_{7 \text{ TO } 9, 7 \text{ TO } 9} = \text{MATRIX}_{3,3} (A^2, 0, 0 \\ 0, B^2, 0 \\ 0, 0, C^2);$$



**INTERMETRICS**

SLIDE 11

## BIT AND CHARACTER MANIPULATIONS

It is anticipated that for the Shuttle, bit and character strings will be required for: logical decision sets, interfaces with hardware, up and down telemetry formulation and decoding, processing of text as status information for ground and on-board display (or recording), complex logical decisions for checkout, monitoring and equipment reconfiguration, interface with machine code (if necessary), and the writing of special support software like an executive, assembler, compiler, simulator, etc.

The manipulation of bit strings, in HAL, is accomplished using the following four operators:

<u>Operator</u>	<u>Definition</u>
NOT ( $\neg$ , ^)	complement
CAT (  )	concatenation
AND (&)	logical AND
OR (  or  )	logical OR

and certain built-in functions and conversions to other data types.

NOT complements each bit in the string; CAT joins two strings; AND and OR perform bit-by-bit logical operations on the corresponding bits of two bit operands. For example,

- 1) NOT B  
Each bit in the string is complemented
- 2) A = B<sub>4 TO 8</sub> AND BIN'10110';  
A logical AND is performed on a bit-by-bit basis.

The manipulation of character strings is accomplished using the concatenation operator, CAT or (||). For example:

- 1) T = C<sub>4 TO 9</sub> || A<sub>1 TO 5</sub>;  
The joining together of two character "sub"-strings.
- 2) WRITE(DISPLAY)'BATTERY VOLTAGE EQUALS' || VOLTS;  
The value of VOLTS is joined to the standard message.

Slide 12 illustrates the use of bit and character strings in decoding a systems status variable and displaying an appropriate status message. On entering DECODE the first three bits of SYSTEM\_STATUS are examined and the proper CASE (i.e., system) selected depending on the integer value 1,2,3,4, etc. A partial message is created. The last three bits are then examined to determine status. Thus the bit string 011001 would cause the message

IMU SYSTEM STATUS: O.K.

to be displayed.



## REAL TIME CONTROL AND ERROR RECOVERY

The real-time control of HAL programs consists of the interrelated scheduling of PROGRAMS and TASK blocks, the reliable sharing of common data, and the recovery from abnormal error conditions.

The concepts and language features have been designed for general applicability to real-time control programming. It is recognized that depending upon specific hardware environments and operating system designs, certain features may not find utility.

HAL real-time control commands permit the scheduling of independent or dependent tasks (i.e., dependent on the existence of another task), based on time and events, establishing absolute and relative priorities, and assigning unique I.D. words for future reference. Before or after the initiation of a task, its priority may be changed or the task may be terminated using the appropriate I.D. word. Once operating, a task may be stalled and/or reactivated based on time or events. Events may be defined to be interrupts or programmer-initiated occurrences. Tasks may then be scheduled or stalled based on single events or combinations of events. The HAL real-time commands include:

SCHEDULE	unconditionally	WAIT FOR...	an event
SCHEDULE...ON	event (interrupt)	SIGNAL	enables occurrence of
SCHEDULE...AT	time		programmer-defined event
SCHEDULE...IN	time increment	PRIO_CHANGE	changes priority of
WAIT...	time increment		task
WAIT UNTIL...	a time	TERMINATE	terminates task

Slide 13 illustrates an example of HAL real-time control.

During execution of HAL programs an error condition may be detected by the system. Examples of errors might be:

overflow/underflow  
divide by zero, or subscript out of range

Depending upon implementation such errors may be hardware or software detected. In any case, execution cannot continue and the system must offer generally applicable alternatives (e.g., aborting the current task, etc.). In order to provide the programmer with some control after the occurrence of an error, perhaps to reset flags or previously initiated I/O commands (e.g., engine jets), HAL permits programmer-defined error conditions and alternatives. An example might be

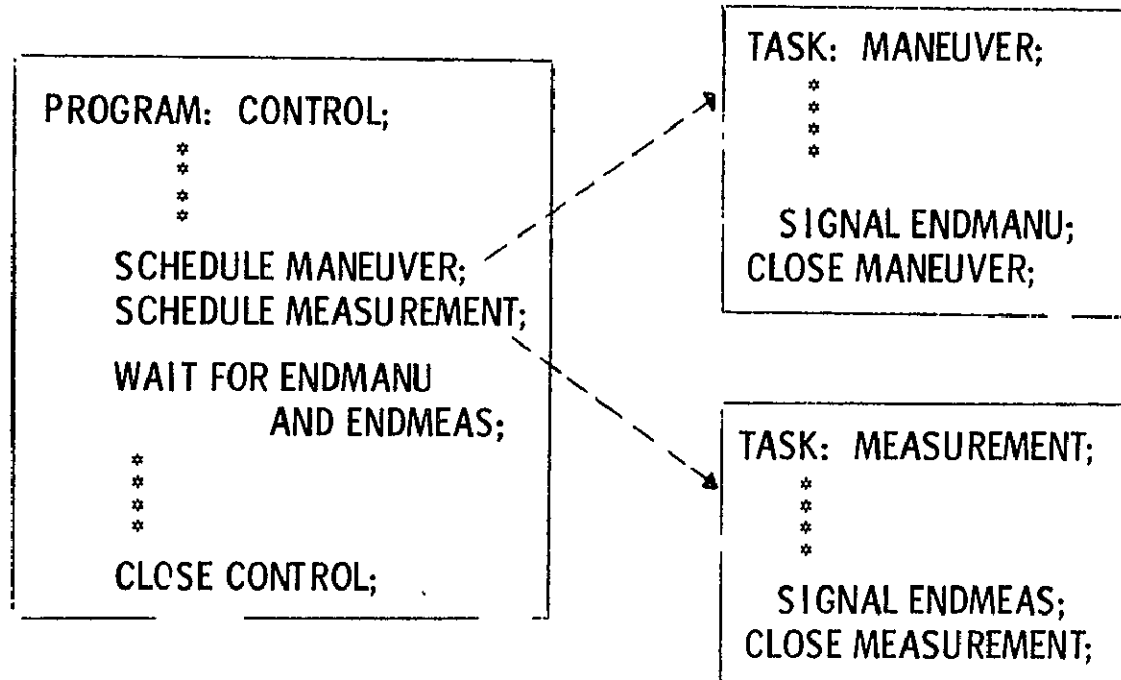
ON ERROR<sub>1</sub> TO 5 GO TO RECOVERY;

which sets up a remedial action on the occurrence of 5 errors, and

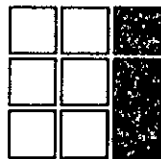
ALARM ERROR<sub>5</sub>;

which signals the actual occurrence of the particular ERROR<sub>5</sub>.

EXAMPLE OF HAL REAL-TIME CONTROL



217



ENDMANU and ENDMEAS are programmer-defined events

**INTERMETRICS**

SLIDE 13



## SUMMARY

HAL has been designed for applicability to advanced manned space missions. As such, reliability in terms of readability and data protection has been emphasized. By incorporating a full spectrum of data types, including bits and characters, HAL finds utility for on-board software, in fixed- or floating-point machines as well as for ground support, simulation, analysis, test and checkout, compilers and assemblers. HAL provides an excellent programming language with which to build special purpose 'user languages' for display, checkout, etc.

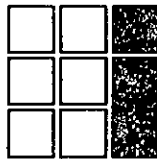
Currently, the first phase of HAL development is being completed. A working compiler will be available at MSC by June, 1971. In this version, code is being generated for the IBM 360/75. The compiler has been designed taking advantage of automatic compiler generating techniques. As such, the total grammar is expressed in a meta-language and changes are easily incorporated and automatically checked for consistency. The HAL compiler generates a universal intermediate code which drives a distinct object code generation module. Transfer to other machines is straightforward by substitution of other object-code-modules. The initial implementation of HAL reflects inherent FORTRAN compatibility and provides linkage to already developed FORTRAN libraries, at minimum cost.

It is anticipated that development of HAL will continue on a schedule necessary to support Shuttle software design and implementation; those specific features, for example real-time control, not included in the first version will be added. Following the selection of an on-board computer, HAL will require only a specific object-code-module.

Intermetrics recommends HAL as the Shuttle higher order language and is confident it can meet Shuttle program requirements.

## SUMMARY

- \* HAL emphasizes reliability
  - \* Readability
  - \* Data protection
  
- \* HAL is a full-capability language
  - \* Includes all data types
  - \* Real-time control statements
  - \* Supports on-board computer software
  - \* Floating- or fixed-point syntax
  - \* Supports ground, checkout, simulation software
  
- \* Schedule of Events
  - \* First version delivery to MSC in June, 1971
  - \* Development to continue compatible with Shuttle schedule
  - \* Object-code-module required for selected on-board computer



**INTERMETRICS**

SLIDE 14

DESIGN OF CRT DISPLAYS FOR THE SPACE SHUTTLE

G. F. Conron

Norden  
Division of United Aircraft Corporation  
Norwalk, Connecticut

PRECEDING PAGE BLANK NOT FILMED

N71-35062

## OUTLINE

This paper discusses the development of a display system for the Space Shuttle Vehicle. The design has been based on the systematic evaluation of display system technologies, adapted, as required, to the Space Shuttle display requirements. Several configurations have been compared on the basis of cost effectiveness. The use of the system for crew/computer communication is discussed.

# **DESIGN OF CRT DISPLAYS FOR SPACE SHUTTLE OUTLINE**

- **DISPLAY SYSTEM ELEMENTS**
- **SHUTTLE DATA DISPLAY REQUIREMENTS**
- **SELECTION OF SYSTEM CONFIGURATION**
  - SYSTEM TRADE-OFFS**
  - COST EFFECTIVENESS STUDIES**
- **CREW/COMPUTER COMMUNICATION**
- **SUMMARY**

## TYPICAL DISPLAY ELEMENTS

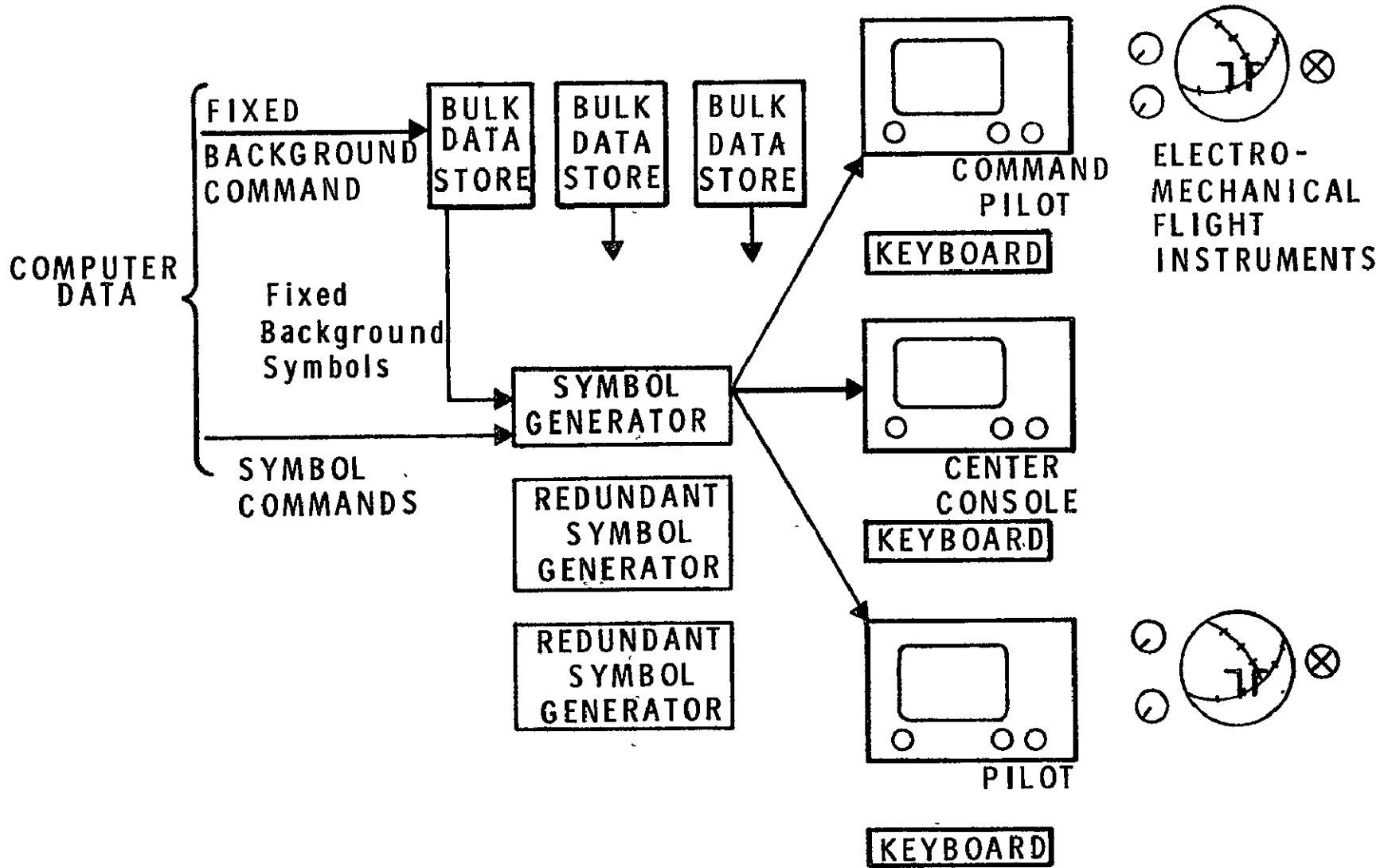
The system shown contains a representative collection of the display elements that have been considered.

Each crew member is provided with a CRT indicator, plus electro-mechanical instruments for flight control. Each CRT indicator has its own keyboard for display mode selection, and for crew communication with the computer.

Three identical, redundant symbol generators are provided, each capable of supplying different symbolic displays to the three indicators simultaneously. Two generators are operated in a standby mode for back-up.

The symbol generators supply variable symbols to the indicators, as commanded by computer data. Each of the three identical bulk data storage units stores data describing 500 frames of fixed background displays. These backgrounds may be superimposed on the symbols provided by the symbol generator.

# TYPICAL DISPLAY ELEMENTS



## DATA DISPLAY REQUIREMENTS

Three primary display modes are required: a) a flight control display, which may be an electro-mechanical attitude-director indicator (ADI) and associated instruments, or an integrated electronic attitude-director indicator (EADI); b) a central data display (CDD) which functions in the manner of a teletype repeater; and c) a systems monitor display (SMD) which provides handbook data, alphanumeric and graphical status data, and displays for computer communication.

There are two redundancy constraints on the design. There must be three independent paths that provide the modes required for mission success. There must be four independent paths that provide the modes required for mission safety.

Systems have been considered that use both electro-mechanical instruments (ADIs) as the primary flight control instruments, and EADIs as back-up.



# DATA DISPLAY REQUIREMENTS

- **3 PRIMARY DISPLAY MODES**
  - ADI OR EADI (ATTITUDE DIRECTOR INDICATOR)
  - CDD (CENTRAL DATA DISPLAY)
  - SMD (SYSTEMS MONITOR DISPLAY)
- **KEEP MISSION SUCCESS ITEMS AFTER TWO FAILURES**
- **KEEP MISSION SAFETY ITEMS AFTER THREE FAILURES**

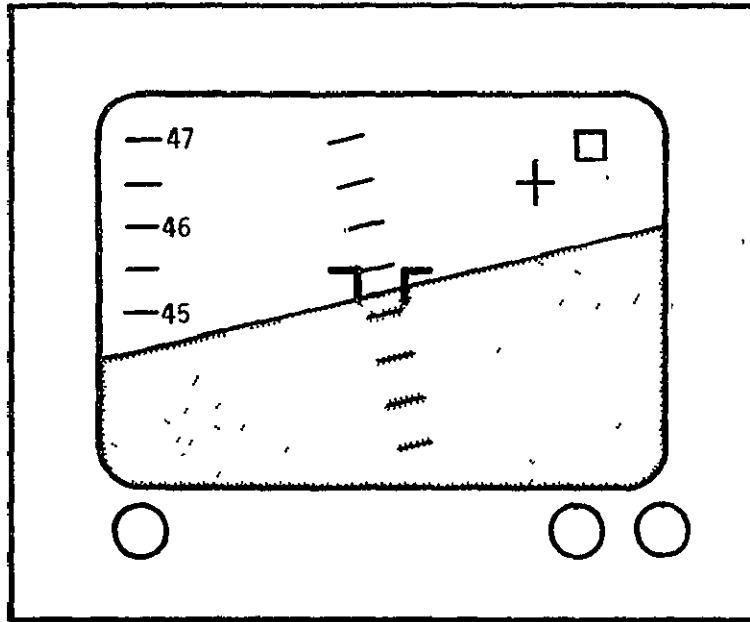
### EADI and CDD MODES

The EADI is an integrated CRT display of roll, pitch, and yaw, plus director commands and additional flight parameters such as attitude, range, angle of attack, etc.

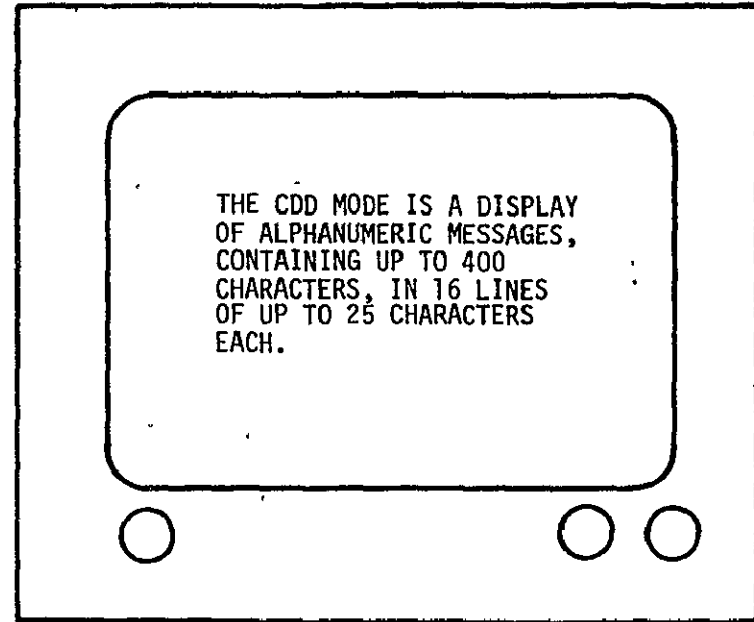
The CDD is a CRT display of computer composed alphanumeric messages, positioned in teletype format, and originating, for example, via data link.

# EADI AND CDD MODES

229



**EADI MODE**



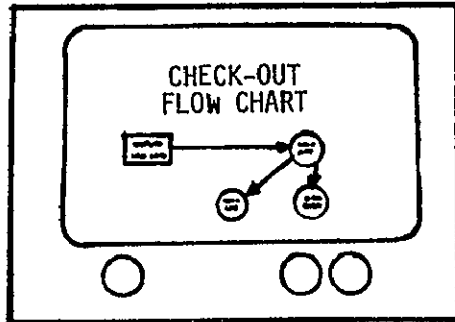
**CDD MODE**

## SMD MODE

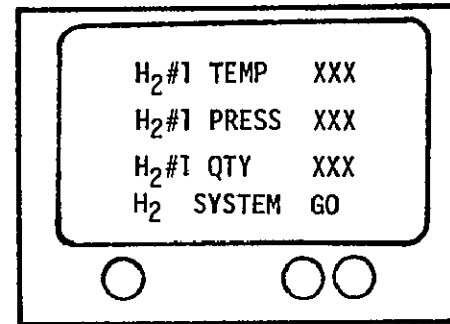
The SMD is a CRT display, provided by the bulk data storage unit, with superimposed symbols from the symbol generator. The SMDs may be separated into four categories:

- HANDBOOK - Handbook pages are provided by the bulk storage units. These include text procedures, flow charts, schematics, etc.
- STATUS (1) - Parametric data on subsystem status is provided by the symbol generator, including digital readouts and alphabetic discrettes such as "ON", "OPEN", "CLOSED", etc. Background legends are provided by the bulk data storage units.
- STATUS (2) - Parametric data is presented, as in STATUS (1), plus graphical data, consisting of points plotted by the symbol generator and background curves provided by the bulk data storage units.
- CREW/COMPUTER-COMMUNICATION - Interactive alphanumeric displays are provided. Background questionnaires are provided by the bulk data storage units. Keyboard inputs are displayed via the symbol generator.

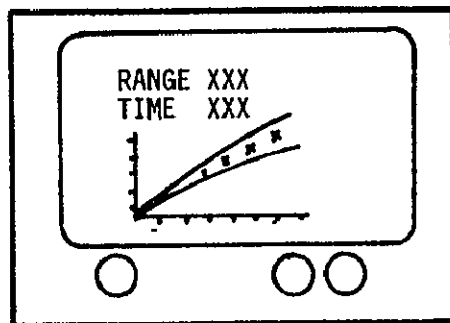
# SMD MODE



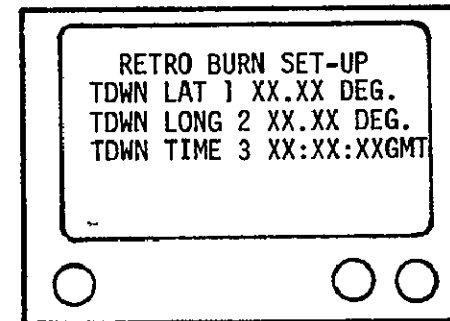
**HANDBOOK**



**STATUS**



**STATUS (WITH GRAPHS)**



**CREW/COMPUTER  
COMMUNICATION**

## SYSTEM TRADE-OFFS

Performance and cost data were established for all candidate display system elements. Cost effective elements were combined in a variety of configurations that represented the system alternatives. The major system alternatives were:

- . All electronic displays, versus a combination of electronic and electro-mechanical displays.
- . Separate symbol generators for each mode, versus a single multi-mode generator.
- . Allocation of display functions to display system or computer — primarily the allocation of the SMD formatting function. In this case, uniquely, the alternatives were not clear cut, and a new formatting technique was developed, as shown in subsequent illustrations.

## **SYSTEM TRADE-OFFS**

- **COCKPIT ARRANGEMENT-ALL ELECTRONIC,  
OR ELECTRONIC PLUS  
ELECTRO-MECHANICAL INSTRUMENTS**
- **SEPARATE EADI,SMD,CDD GENERATORS,OR  
SINGLE,MULTIMODE GENERATOR**
- **ALLOCATION OF FORMATTING FUNCTIONS  
TO DISPLAY SYSTEM AND COMPUTER**

## PRIMARY COCKPIT ALTERNATIVES

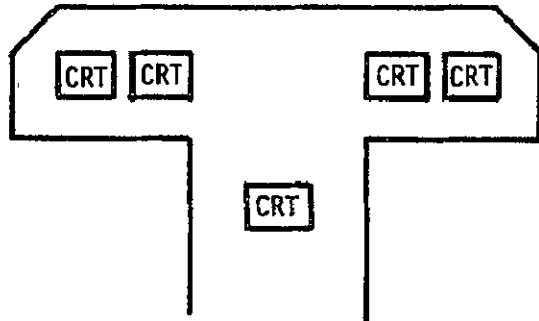
The cockpit arrangements that were considered fall into two categories:

- . An all-electronic system, in which each crew member has two CRT displays at his station, plus a center console display for back-up. One CRT at each crew station is used as an EADI, and the second as an SMD/CDD.
- . A hybrid system in which each crew member has a set of electro-mechanical instruments for primary flight control, plus a CRT for SMD/CDD, and the center console for back-up. The CRT at each crew station can also provide EADI to back up the electro-mechanical instruments.

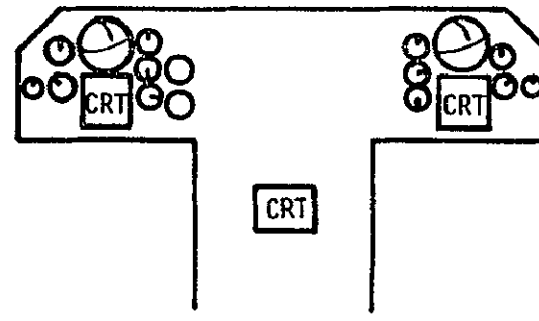


# PRIMARY COCKPIT ALTERNATIVES

285



**ALL ELECTRONIC**



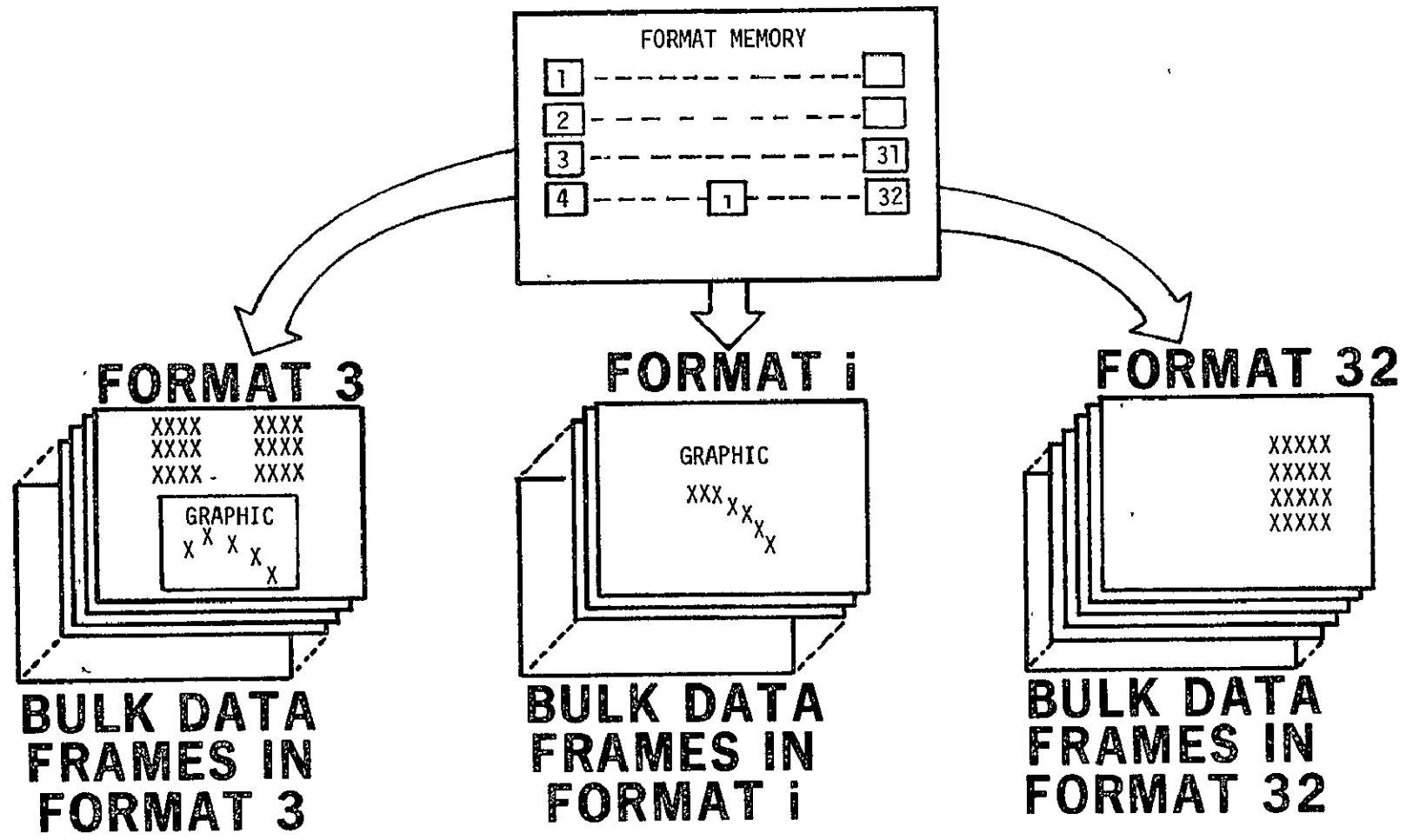
**ELECTRONIC PLUS  
ELECTRO-MECHANICAL  
(EADI BACKS UP ADI)**

## SMD FORMATTING

The EADI and CDD modes have well established formats that are most economically programmed by hard-wiring in the display system. Since the SMD mode provides all of the mission-variable display, it should not be frozen in hard-wired programs. The bulk data storage units provide easily reprogrammed SMD backgrounds. The superposition of symbol generator characters on these backgrounds must be accomplished with a minimum of inflexible hard-wired programming, and a minimum of additional computer software and data bus positioning commands. This is accomplished with the hard-wired programming and computer message structure discussed below.

Hard-wired SMD programming within the display system describes the structure of 32 formats. Each bulk data storage frame is designed to be congruent with one of these 32 formats.

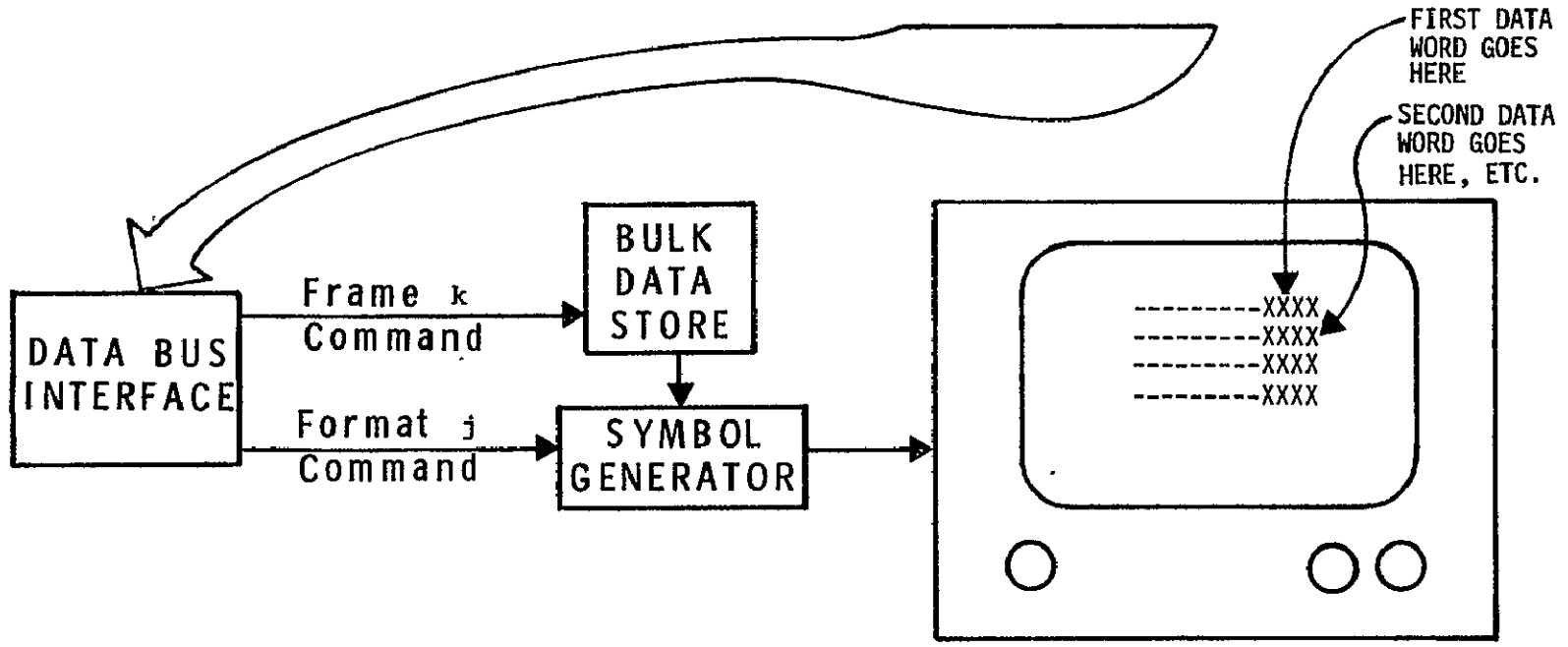
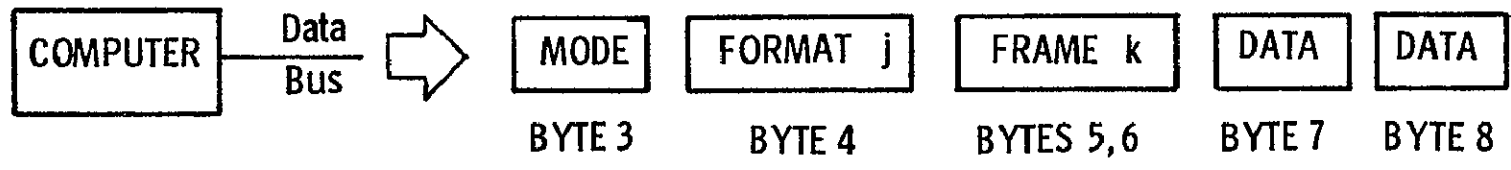
# SMD FORMATTING FUNCTIONS. EXTENT OF GENERATOR HARD-WIRED PROGRAMMING



In the SMD mode, the structured computer message signals that the SMD mode is to be generated (byte 3), identifies which of the 32 hard-wired formats is to be used by the symbol generator (byte 4), and commands the background frame that is to be superimposed by bulk data storage unit (byte 5 and 6). The succeeding bytes contain data for symbols, to be positioned by the symbol generator in a prearranged sequence within the commanded format.

# SMD FORMATTING (cont'd)

## STRUCTURED COMPUTER MESSAGE



### COST EFFECTIVENESS FACTORS

The alternate system configurations were compared on the basis of their total program cost. Factors were included for weight, power, reliability, and procurement costs. Energy consumption and mission probabilities were calculated for a seven day mission, with three different levels of crew work load per day.

# **COST EFFECTIVENESS FACTORS**

- **WEIGHT (DOLLARS PER POUND)**
- **ENERGY ( DOLLARS PER KILOWATT HOUR)**
- **PEAK POWER (DOLLARS PER PEAK KILOWATT)**
- **COST OF DDT&E**
- **COST OF PRODUCTION HARDWARE**
- **PROBABILITY OF MISSION SUCCESS (DOLLARS,  
SCALED BY IMPACT ON UNITY PROBABILITY)**
- **PROBABILITY OF MISSION SAFETY (DOLLARS,  
SCALED BY IMPACT ON UNITY PROBABILITY)**

## SUMMARY OF COST EFFECTIVENESS STUDIES

Over twenty system configurations were evaluated. The configurations were in two categories, all-electronic systems, and electronic displays plus electro-mechanical ADIs. The original baseline configurations (A) used available techniques. A new multi-mode generator was designed (B), and this brought an improvement in cost effectiveness. Finally, the new SMD formatting technique was introduced and the electronic elements were redesigned for minimum power consumption (C). This last step gave a significant improvement in cost effectiveness.

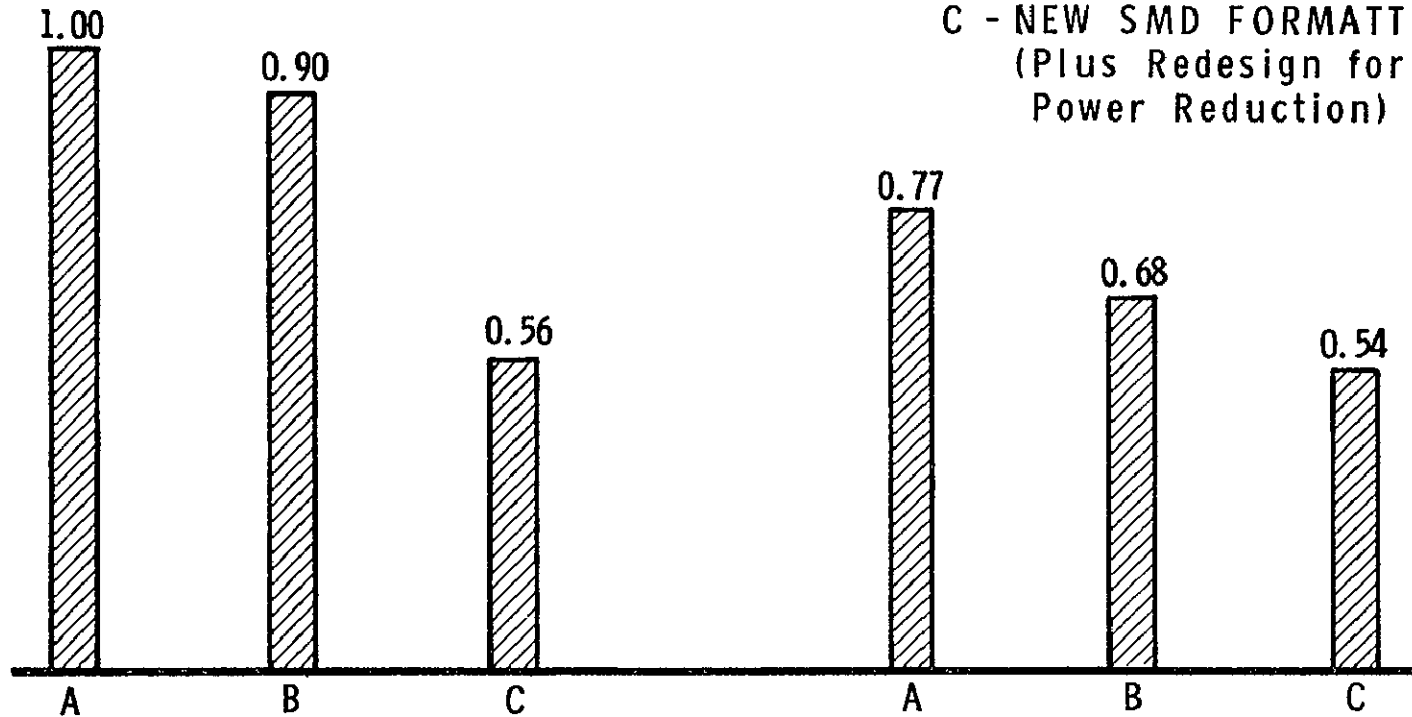
The final comparison showed little difference between the best all-electronic configuration and the best configuration using ADIs. As a result, the selection will be made on the basis of uncosted factors such as training costs, flexibility, crew preference, and simulator experience.



# SUMMARY OF COST EFFECTIVENESS STUDIES

- A - ORIGINAL BASELINE
- B - MULTI-MODE GENERATOR
- C - NEW SMD FORMATTING  
(Plus Redesign for Power Reduction)

RELATIVE PROGRAM COST



**ALL  
ELECTRONIC**

**WITH ADI**

## CREW COMPUTER COMMUNICATION

When a crew member presses the phase/function button and selects the SMD mode, he receives a master list of mission phases, with instructions for selecting each phase via the keyboard. When he selects a mission phase, he receives a display options index containing instructions for selecting SMD displays available for that mission phase. Displays available via the display options index may include sub-indexes, as required.

# CREW/COMPUTER COMMUNICATION

## STEP 1

### MISSION PHASE

1. CHECK-OUT
2. LAUNCH
3. ORBIT MANEUVER
4. FLIGHT PLAN
5. NAVIGATION
6. RE-ENTRY
7. LAND

## STEP 2

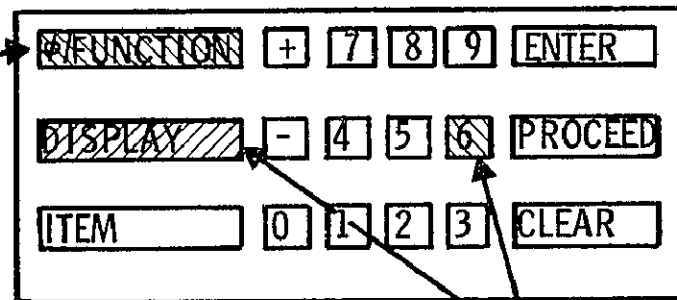
### RE-ENTRY DISPLAY OPTIONS

1. -----
2. -----
3. -----
4. RETRO BURN SET-UP
5. -----
6. -----
7. -----

245

## KEYBOARD

STEP 1  
SET SMD MODE  
PRESS "φ/FUNCTION"  
SELECTS MISSION  
PHASE DISPLAY



STEP 2 PRESS "6" FOLLOWED BY  
"DISPLAY" - SELECTS RE-ENTRY  
DISPLAY OPTION LIST

Specific displays for a mission phase are selected by following the instructions on the display options index.

Entry of computer data is achieved by means of SMD displays that are in questionnaire format. The questionnaires include the names of the parameters, instructions for entering each parameter, and spaces for displaying the values received by the computer. When the crew member, by means of the display, has verified computer receipt of the correct parameter value, he presses "ENTER" and proceeds to the next item.

# CREW/COMPUTER COMMUNICATION

## STEP 3

RETRO BURN SET-UP  
TDWN LAT 1           XX.XX DEG  
TDWN LONG 2         XX.XX DEG  
TDWN TIME 3         XX.XX DEG

(cont'd)

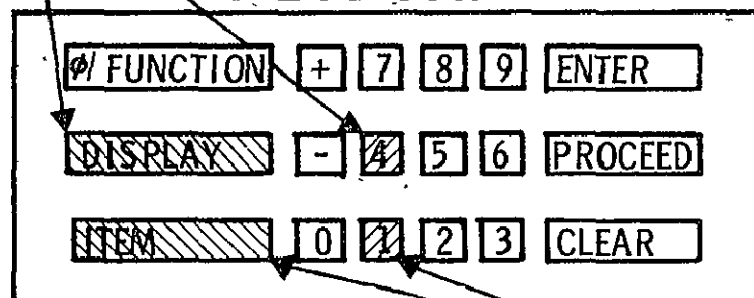
## STEP 4

RETRO BURN SET-UP  
TDWN LAT 1    34.28 DEG  
TDWN LONG 2   XX.XX DEG  
TDWN TIME 3   XX:XX:XX GMT

247

STEP 3 PRESS "4" FOLLOWED  
BY "DISPLAY"- SELECTS  
RETRO BURN SET-UP DISPLAY

### KEYBOARD



STEP 4 TO ENTER TDWN LAT,  
PRESS "1" FOLLOWED BY  
"ITEM", FOLLOWED BY "3",  
"4", "2", "8"

### SUMMARY

This paper has discussed the systematic development of a display system for the Space Shuttle Vehicle. A cost effective configuration has been developed that is specifically adapted to the Space Shuttle requirements. In particular, the system meets Space Shuttle redundancy requirements, provides systems monitoring displays that are easily pre-programmed between missions, and provides an integrated, easily assimilated method of crew/computer communications.

# **SUMMARY**

- **DISPLAY SYSTEM ELEMENTS**
- **SHUTTLE DATA DISPLAY REQUIREMENTS**
- **SELECTION OF SYSTEM CONFIGURATION**
  - SYSTEM TRADE-OFFS**
  - COST EFFECTIVENESS STUDIES**
- **CREW/COMPUTER COMMUNICATION**

## ONBOARD DATA STORAGE FOR SPACECRAFT

J. E. Holthaus and L. B. Gangawere

Westinghouse Defense and Space Center  
Baltimore, Maryland

This paper presents the summary of a design study and laboratory program to evaluate and select a technique for presenting static manual/handbook data in the crew station of the Space Shuttle vehicle. The information was compiled in conjunction with MDAC, and a model of the selected approach is under construction for demonstration in early June.

1 N71-35063



## BULK DATA STORAGE REQUIREMENTS

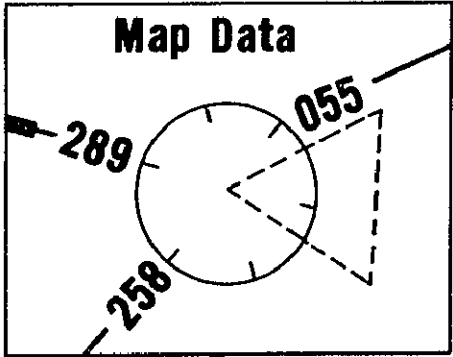
Onboard storage of several thousand pages of technical and instructional data can be required for extended spacecraft flight operations. The recall and presentation of this data must be compatible with the most reliable control and display techniques currently developed. Obviously it is not practical to carry volumes of paper instruction books aboard a spacecraft nor is it cost effective to have ground control personnel relay the information through a communications link. Therefore, the data must be reduced, stored and carried aboard in such a way that it is readily available for presentation when required.

The data may take many forms and be required during all portions of the Shuttle flight. The information, when graphical, must be registered very precisely with overlay information from the central computer or, as the case may be, another onboard sensor.

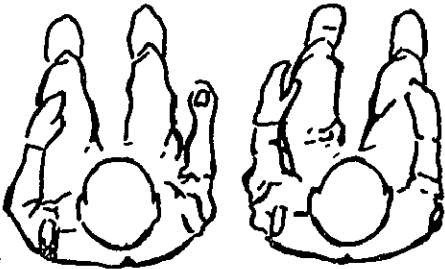
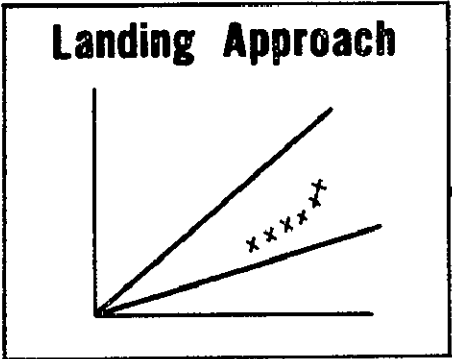
Particular emphasis therefore must be placed on overall cockpit integration and matching the input/output characteristics to the Shuttle's avionics system. Since several displays will be utilized by the crew the bulk storage data will be required to be displayed on one or all three of the displays depending upon the flight situation. Size, weight, power and flexibility will naturally determine the type of storage unit and video display system implemented.

**Entry Data**  
**Req'd DV**  
  
**Residual DV**

**Abort Data**  
  
**Dump**  
**Ditch**  
**Orbit**  
**Stage**



253



**Flight Plan**  
**Alternate**  
  
**UHF**  
**VHF**  
**Comm**

# **BULK DATA STORAGE UNIT (BDSU) INFORMATION**

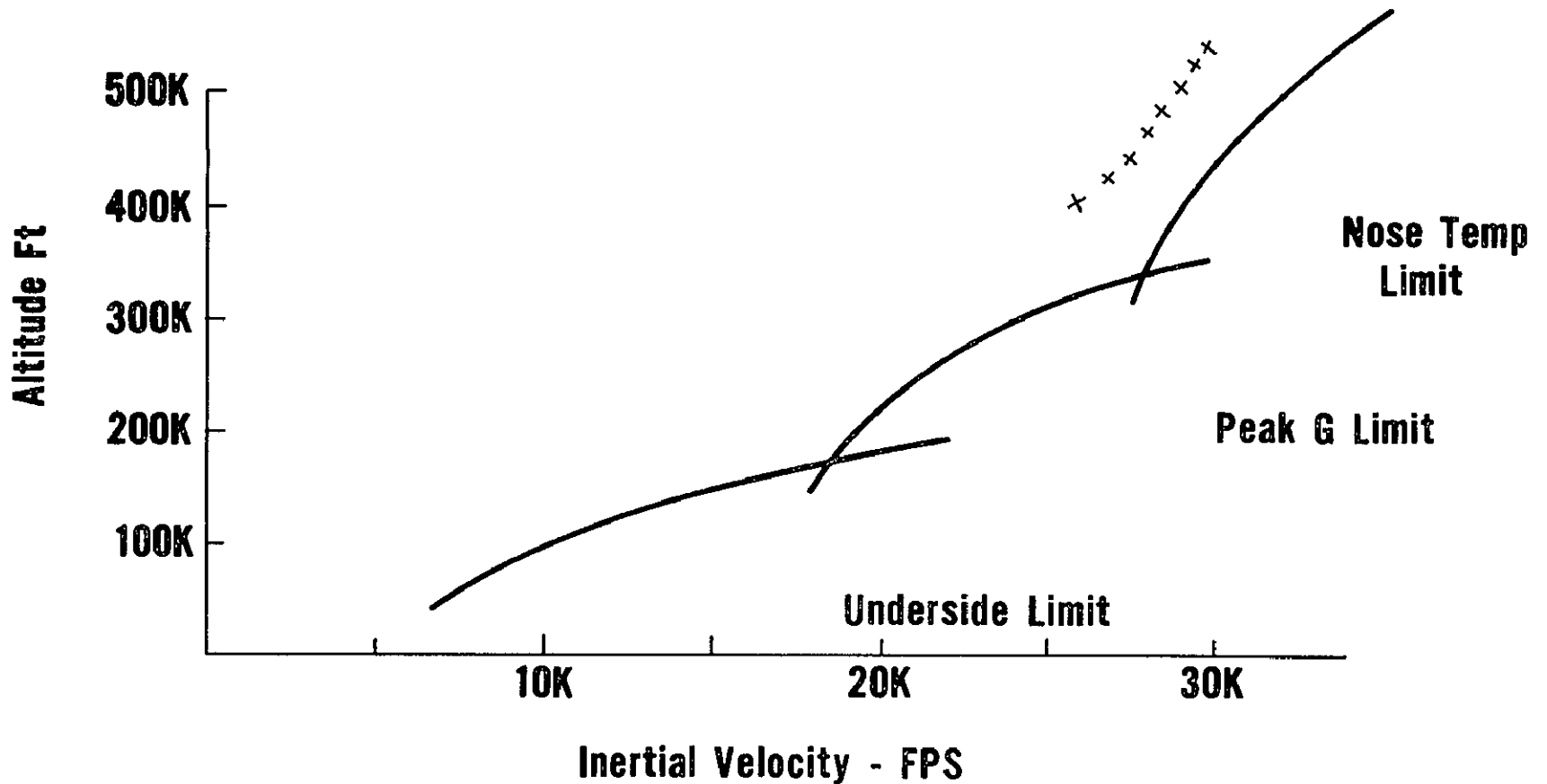
## TYPICAL FLIGHT DATA

Typical data that could be stored is illustrated. The information required for storage for the RE-ENTRY curve shown is the graph of altitude vs velocity in addition to the callout for numerical entries such as RANGE, HDOT, ETA and CRNG. This data will be displayed to the shuttle crew. The computer will determine a position track and overlay the actual flight path and the corresponding numerical readouts. The registration of the computerized data and the graphical data must therefore be precise. Initial studies have indicated that the registration should be within 1% of picture dimensions.

# ENTRY FLIGHT DYNAMICS

Impact Lat	x x x	Deg	Cross Range Error	x x x	NM
Impact Long	x x x	Deg	Down Range Error	x x x	NM
HDOT	x x x	FPS	Range To Go	x x x	NM

255



## BULK DATA STORAGE WITH COMPUTER READOUT OVERLAY

For crew/computer communications the bulk data storage unit provides the data as shown. The computer is addressed to provide the numerical values required.

Using a display size of 5 by 7 in. and a viewing distance of 18 in., it has been shown that alpha numeric characters which subtend an angle of 12 to 15 minutes of arc can be easily read. This yields a character height of approximately 0.08 in. If half of this value were allowed for separation between lines then a total of 0.12 in. would be allotted for each line or 40 lines on a 5 in. vertical height. Assuming a 5:7 character aspect ratio and minimum separation between characters in a line, approximately 0.08 in. should be allowed for each character along a line. This allows approximately 85 characters to be written along a line.

Using 40 lines of 85 characters each then 3400 characters can be resolved. This was proven during laboratory display experiments; however, the display was extremely busy and difficult to read rapidly in text form. After a series of operator visual experiments both from a timed recognition and fatigue standpoint, the number of lines and characters was selected as approximately 50% of the maximum. The text readout was therefore limited to 20 lines and 40 characters per line for a 5 by 7 in. display.

# TYPICAL DATA

## Landing Target Data

Latitude	yyy.yy	Deg
Longitude	yyy.yy	Deg

## Attitude and Orbital Conditions at Burn

Pitch Attitude	xxx.xx	Deg
Roll Attitude	xxx.xx	Deg
Yaw Attitude	xxx.xx	Deg
Altitude	xxxx.xx	N M

### TYPICAL CONVENTIONAL LANDING APPROACH DATA

Other projected data can take the form of maps, instruction book data, emergency procedures, staging information, ascent curves, landing aids and landing site descriptions.

Photographic and pictorial aids can also be stored for quick reference. For mapping and navigation purposes chart data can be prepared and projected and the computer can indicate the shuttle's present position with respect to required heading either for orbital or suborbital flight plans.





## STORAGE SYSTEMS CONSIDERED

Two practical methods of bulk information storage were considered for use within the next 5 years. These were (1) microfilm and (2) magnetic tape. Other currently practical methods of mass storage utilizing rotating discs or drums were not considered because of their volume, weight, gyroscopic effects, and lack of mechanical airworthiness.

### Magnetic Tape

Large amounts of information may be stored efficiently and economically on magnetic tape. If the tape is protected from extraneous magnetic fields the information stored will remain indefinitely without deterioration. Long term storage is obtained during readout with a high signal to noise ratio (>40 db) when operating in a continuous feed mode. A typical polyester plastic tape has a temperature coefficient of  $12.44 \times 10^{-6}$  expansive/ $^{\circ}$ F. This should not result in any problems with sync.

Several problem areas exist however, the major problem being the result of stop action display. In this mode, a single field is repeatedly scanned and a complete, interlaced frame is not available for viewing. In a 729 line system, approximately 670 lines are active. Since only 1 field is read out, there are effectively only 335 active scanning lines which would clearly be insufficient for high readability.

Another problem resulting from stop action display is clogging of the heads. In normal, continuous feed tape usage, the heads are self-cleaning. However, in stop action, the head accumulates material from the tape and clogs up. Operation of up to 15 minutes is feasible without manual head cleaning. If the heads are not cleaned, destruction of the tape may result. If the tape must operate in humid environment with elevated temperatures, tape coating is loosened and the head clogs up much faster. In a dry atmosphere, the tape will operate at temperatures to 150 $^{\circ}$ F before damage results.

### Microfilm

Perhaps the most efficient storage medium used today is microfilm. The stored information does not deteriorate with use except as the physical properties of the film deteriorates. Retrieval of the information is accomplished by some form of projection. Any projection scheme must consist of a film transport unit, a light projector and a suitable means for selecting a particular frame. The methods vary in the way information is viewed. For instance, the information may be projected on a wall, special screen, the face of a CRT or the target of a TV camera. Therefore, it is seen that any projection system may be considered as consisting of 3 main groups.

1. Frame selection electronics.
2. Film transport and light projection.
3. Screen or TV camera.

# BULK DATA STORAGE SYSTEMS

## Microfilm

- Conventional Projection
- Rear Ported Cathode Ray Tube Projection
- Camera/Reader

## Magnetic Tape

- Analog
- Digital

## BULK DATA STORAGE COMPARISON CHART

A conventional projection device can be mechanized utilizing microfilm, a mirror and a viewing screen. Static data is displayed through digital logic selection or computerized commands. The major problems associated with this approach are (1) that it is a single display unit and cannot be remotely projected on existing spacecraft monitors and (2) that additional digital input data cannot be registered with the static data as required.

The Rear Port Cathode Ray Tube may be used as a video display and at the same time have recorded data projected through a clear port in the rear of the tube. The need for rear ported tubes was established when cockpit film recordings of video data presented on the phosphor were required without putting a camera over the face of the tube. A major problem associated with this approach is the distortion caused by either the projection being off center or the electron gun being off center. Also the mechanical assembly of a projector on the rear of the tube has vibration characteristics that are not indicative of high reliability and registration.

The Magnetic Tape approach provides for a flexible unit, yet it is extremely difficult to mechanically phase lock a high frequency signal for adequate registration. High power is required in the servo drive for high speed head or tape operation. An investigation into the digital tape approach also indicated the requirement for high speed heads, the associated clogging problem, and the additional power required for an X-Y deflection type monitor.

The camera reader provides advantages in most of the categories investigated. It is easy to register video data with computer generated data since it depends solely on the linearity of the camera sweeps.

The unit operates at 20 watts normally and 40 watts when slewing. Data can be easily edited and prepared. It is ideal for use with a digital interface such as computerized data and overlay data. From these advantages the approach was selected as the most feasible and effective.

# BDSU COMPARISON CHART

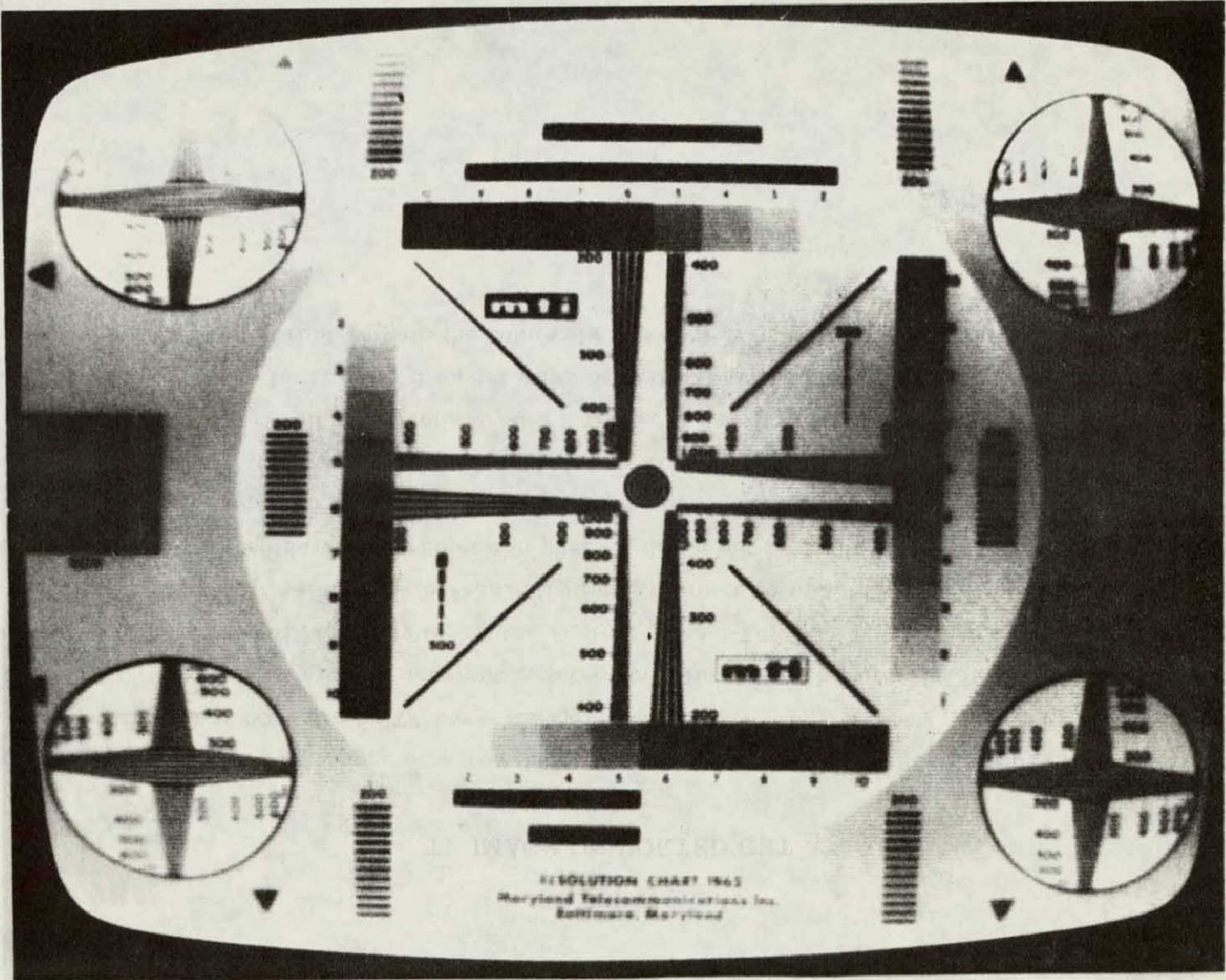
	<b>Registration</b>	<b>Power</b>	<b>Remarks</b>
<b>Conventional Projection</b>	<b>Difficult -Analog</b>	<b>Lamp 25-100W</b>	<b>-Single Display -Non Complex</b>
<b>Rear Port CRT</b>	<b>Distortion Problem</b>	<b>Lamp + CRT</b>	<b>-Multi Display -Mech. Problem -Single Projector</b>
<b>Magnetic Tape</b>	<b>Phase Lock Problem</b>	<b>Power For Speed</b>	<b>-Multi Display -Current Tech. -Easy To Update</b>
<b>Camera Recorder</b>	<b>Easy Camera Linearity</b>	<b>20-30W</b>	<b>-Multi Display -Easy To Edit -Compact</b>

## PROJECTED RESOLUTION CHART

The next three slides illustrate some of the data recorded in preliminary tests made last year and are included in this report because they illustrate the type of video and symbology.

The following resolution chart presentation was made on a prototype unit. The photograph when adequately reproduced shows 500-600 TV lines limiting resolution and 7 shades of gray. Dynamic focus was not used in the monitor and consequently the edge resolution is poorer than should be expected.

The resolution is characteristic of a Camera/Reader unit available at this time. The resolution and gray shade limitations are a function of the camera only, since the film development can be controlled and the corresponding low resolution losses are not realized in the projection optics.



NOT REPRODUCIBLE

**RESOLUTION CHART BDSU DISPLAY**

### TV IMAGE ON PORTED CRT

This slide illustrates a TV image on a rear port cathode ray tube. For the intent of the test, dynamic focus was not included; however the TV image was quite adequate.

If the information to be projected does not require coordination with information presented by the electron beam, the port will normally be off the center of the faceplate. Distortions due to off axis projection are not serious if data is only to be read. If linearity must be maintained, for superposition of video information over projected data, the projection port would be put in the center of the tube and the electron gun off axis. This is required since the electron optics necessary for the electrical correction may be accomplished with more accuracy than the expensive optics and critical alignment needed to correct the projected picture. As a result of various vendor surveys it was determined that the tube could be made within specification. The location of the port was not critical although the bulk data storage projector presented an interference problem with the tube yoke assembly.

NOT REPRODUCIBLE



267

**TV DISPLAY ON PORTED CRT**

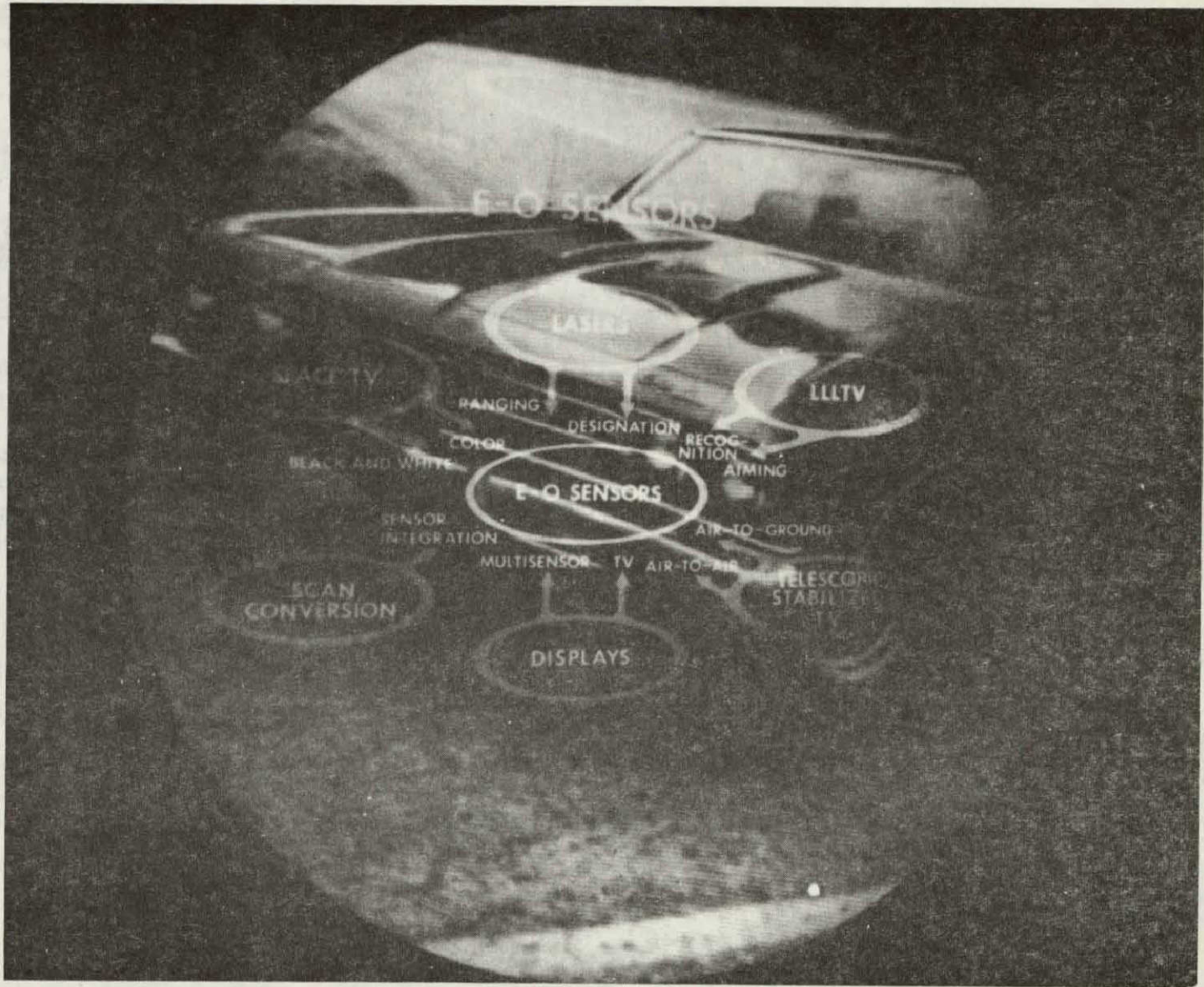


## PROJECTED TV IMAGE WITH SYMBOLOGY ON REAR PORT CRT

From this illustration it can be seen that much keystoneing was prevalent when projected data was overlaid with live video. This however would be much less if corrective optics were used. For the purpose of this test the projection was made to evaluate feasibility. The ported CRT faceplate was flat and this caused a further lack of concentric images.

From the results of the tests, the following was determined:

1. Projection of microfilm or slides on TV video is feasible and easily accomplished.
2. The readability of slides with or without the TV format is very good.
3. Color slides project excellent images. The selection of the tube phosphor can effect this according to the user's choice.
4. Resolution of TV video was approximately 500-600 TV lines limiting, and brightness was over 40 ft-lamberts. The dynamic range was eight shades of grey.
5. Keystoneing will have to be corrected and appears to be a significant problem.
6. The projector used was 100 watts. This was very satisfactory for the tests and it would be feasible to lower this to 25 watts.
7. The microfilm projector to be mounted to the rear of the tube would be approximately the same complexity as that used in the conventional or vidicon camera projection. Interference with the yoke was expected to be another major problem.
8. Environmental protection from vibration and shock in a finished assembly will require a major mechanical design effort.



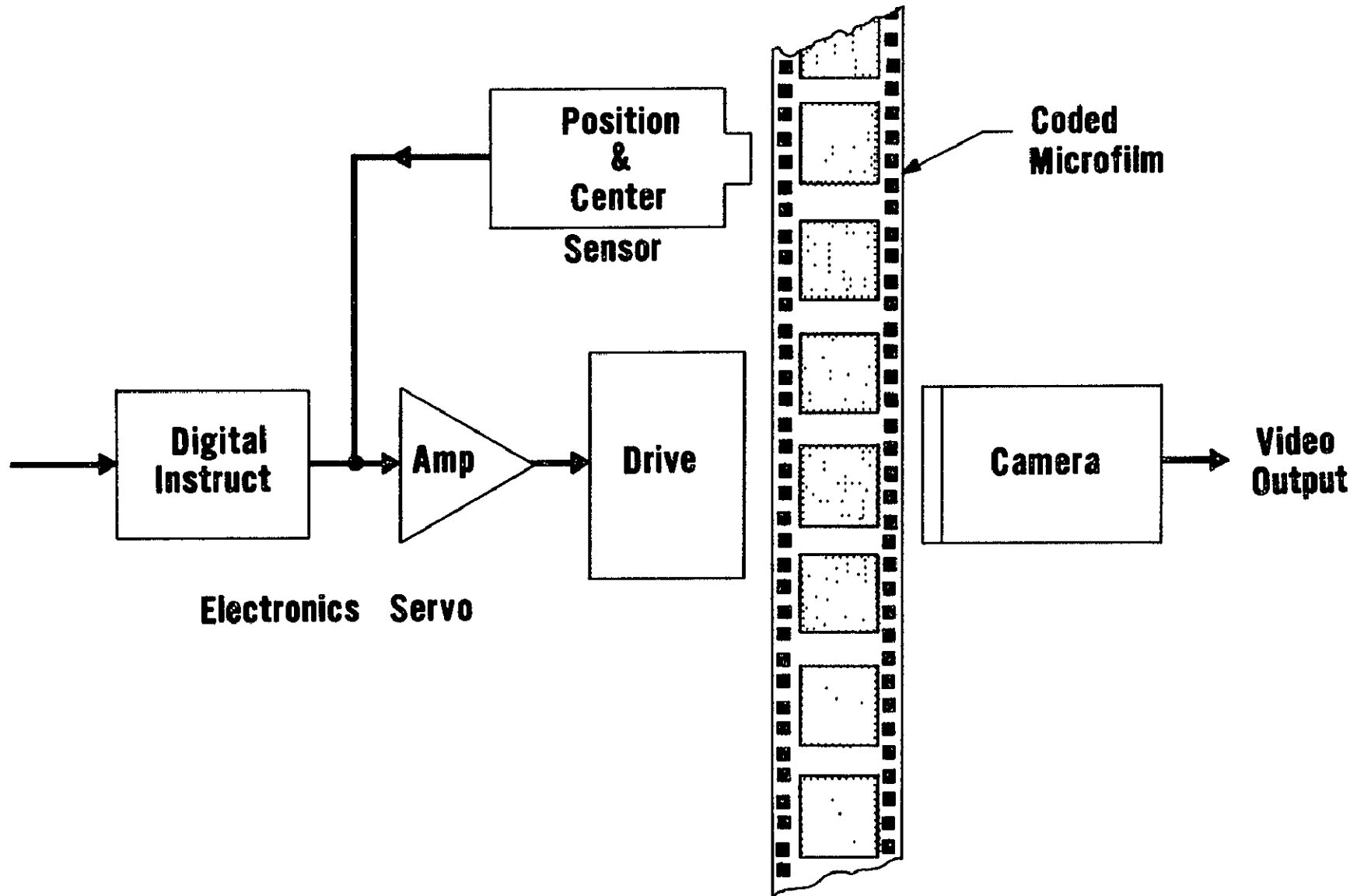
**TV & INSTRUCTIONS ON PORTED CRT**

## SELECTED BULK DATA STORAGE UNIT (BDSU) FUNCTIONAL DIAGRAM

At the conclusion of the tests it was determined that a camera reader, later named a Bulk Data Storage Unit (BDSU) should be designed, fabricated and used in a Space Shuttle Simulator for demonstration. The following is a description of the unit.

From the functional block diagram of the BDSU, it can be seen that the device is composed of three major subunits; the servo-driven film transport, the automatic frame selection electronics, and the film reader (a subminiature TV camera). Microfilm data of written text, tables of references, headings for dynamic data and graphs with limit values is imaged on the tube by a coded film cassette and then displayed in TV format on a CRT. The unit provides up to 500 registered frames of information individually selected either manually or by the computer. The frames of data can be addressed from 0 to 500 in less than 5 seconds with precise accuracy.

The bulk data storage unit provides up to 500 frames of information which may be selected for display either manually or automatically. In the manual mode, a 9-bit digital word is set into the local frame selector. When the unit is activated the film will be slewed to a computed location, framed, and then verified for correct address. If the address is the selected one, the video from the frame is available for display on the selected monitors. If the frame address is different than the selected address (assume a condition where the tape may have been edited), the unit will compute the proper location and repeat the cycle. If the selected frame cannot be found in four attempts, it will be assumed that it does not exist (may have been edited out) and file searching will cease. A visual and/or audible indication of the situation will then occur.



**BDSU FUNCTIONAL DIAGRAM**

## BDSU DESCRIPTION

The selected frame is read by a miniature TV camera. A low-power collimated light source illuminates the frame so that the vidicon camera will receive about 4 fc (peak white) for 40 db signal-to-noise ratio. The output of the camera is a standard 729 line, 30 frame, 2:1 interlace composite video signal.

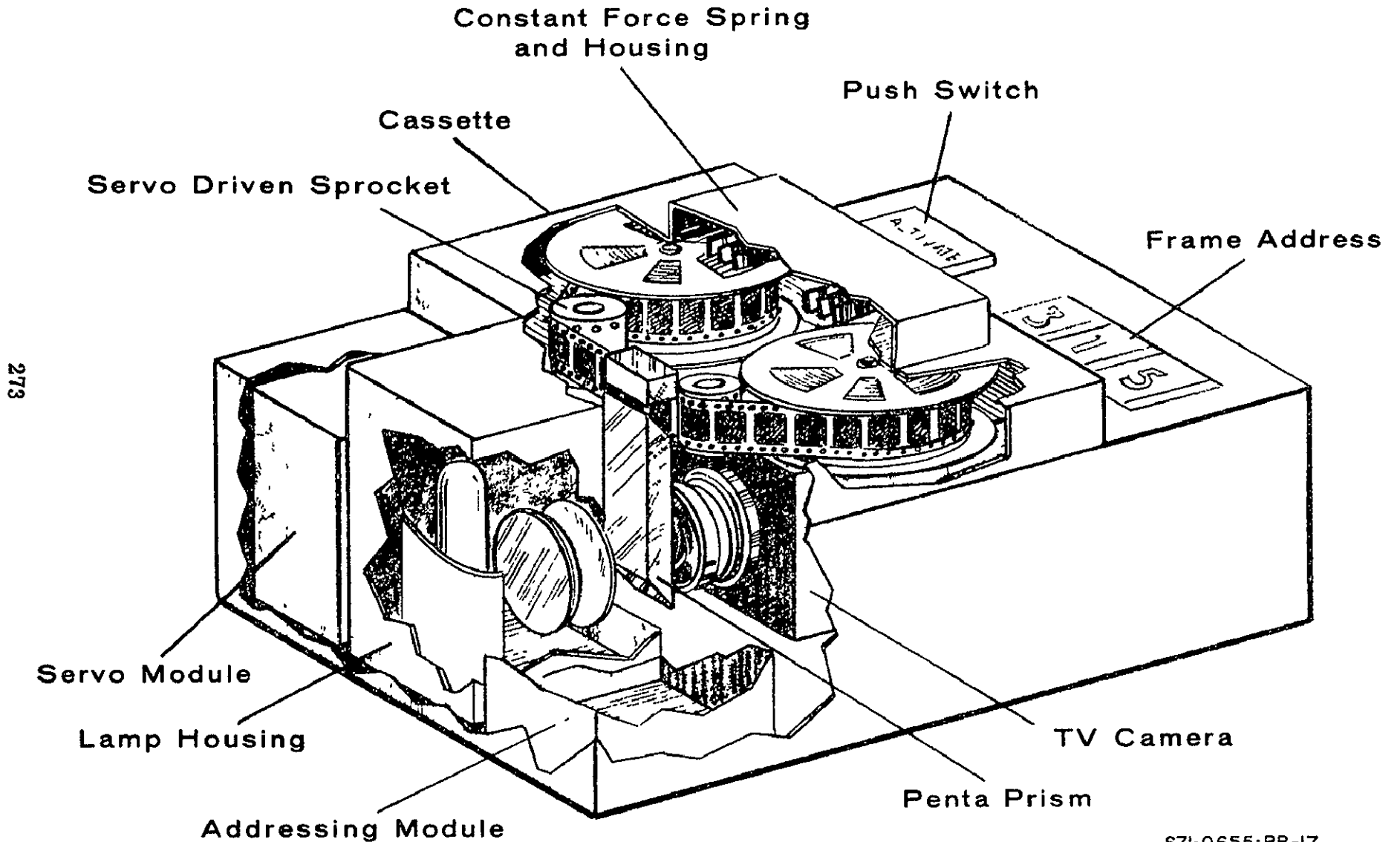
The film transport is a sprocket-type, gearless, servo-driven mechanism. Constant force springs are used for film control during high-speed slew. The sprocket is used for positive film control and, in conjunction with other film guides, provides the transverse film framing. Lateral framing is accomplished electronically.

Initial alignment of the system is done mechanically. After that, alignment of all film frames is assured through the use of differential phototransistor sensors, which sense the location of a pair of illuminated slots in the film. These slots are located precisely with respect to a frame edge during exposure of the filmed data. A relatively simple fixture used during the filming will provide this precise location along with a digitally coded light transparent digital word representing the address of the frame. This address is also read by photo-transistor sensors to enter into the electronics the present address of the reader.

Once the frame is aligned, the video output is transmitted to a symbol generator. This unit provides the symbol generation for the various modes and the synchronizing pulses for the system. The basic clock frequency for the symbology will run typically at about 17 MHz. This represents 640 complete cycles per line on a 729 line system so that data stored can be positioned very accurately with respect to the horizontal sync pulse for each television line by a digital counter. (To less than 1/2 resolution element.) All that is necessary then to obtain accurate registration is that the amplitude and linearity of the camera sweep be closely controlled. Linearity and amplitude of the sweeps are controlled by current feedback.

The electronics of the BDSU includes a digital velocity feedback so that small changes (such as one or two frames) receive maximum output from the servo until a pulse fed back to the servo system indicates the film is moving. Then, slew rates normal for the frame error are introduced to the system.

# BULK DATA STORAGE UNIT (BDSU)



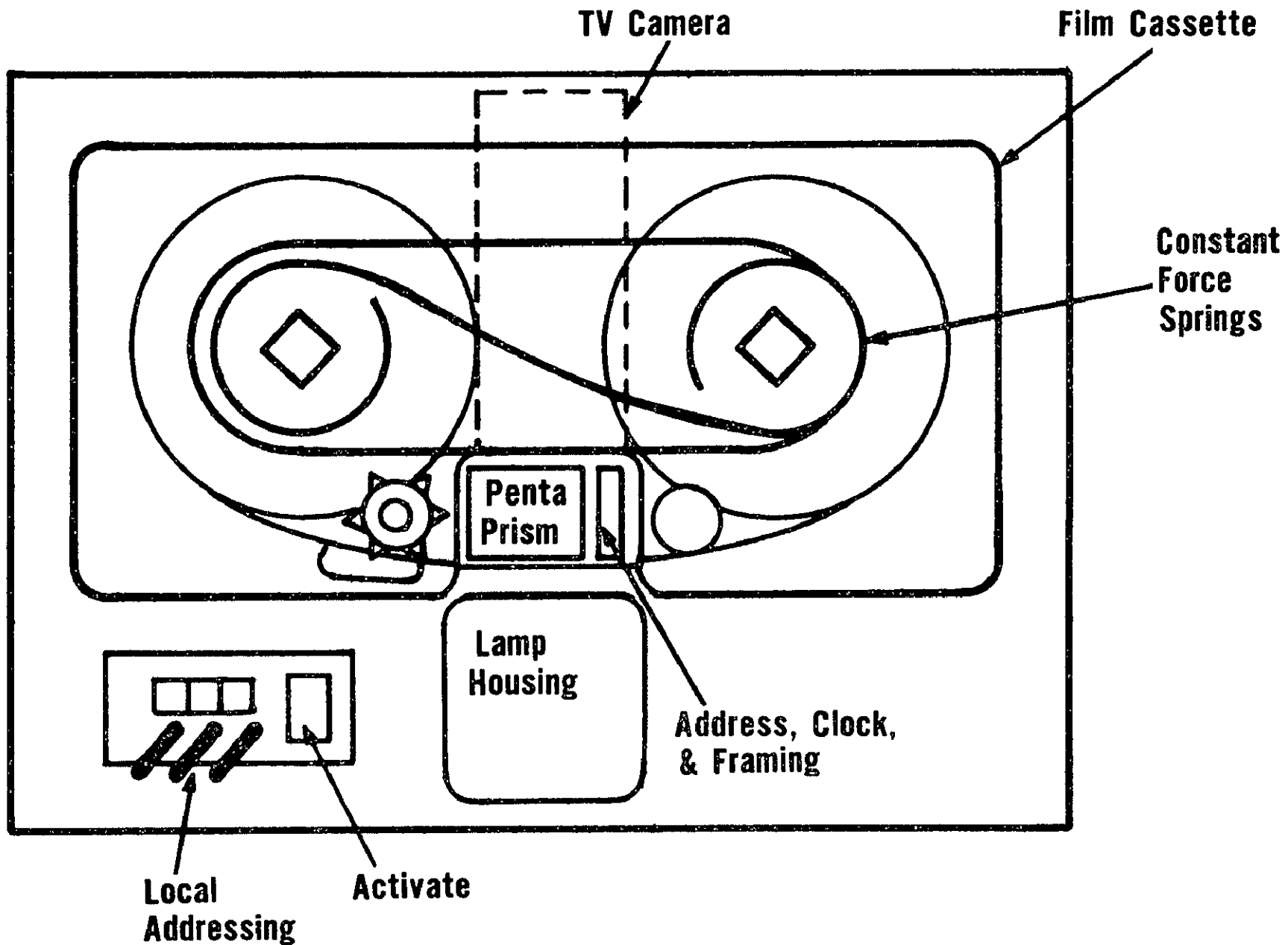
## BDSU SERVO DRIVEN FILM TRANSPORT

Schematically, the transport is as shown.

The tape reels are contained within a cassette-style holder complete with film guides, constant force springs, sprocket drive, and mechanical brake mechanism. The constant force springs are preloaded prior to film assembly into the cassette. The brake prevents film travel when the cassette is removed from the transport or after the framing has been completed.

Either reel acts as a takeup or a supply reel even though neither reel is directly driven. The film is driven through the sprocket directly coupled to the servo motor. The constant force springs are used to control the film during high speed slew and negate the requirement for mechanical clutches and belt coupling of the reels. Film tension will be relatively constant throughout the travel of the film. This mechanization requires no gears and permits easy change of bulk data storage. The removal of four guide screws, which serve to clamp the cassette to the deck, also assures vertical frame alignment with the TV camera and address sensor.

The address, clock, and framing sensor is a light pipe assembly located in proximity to the film plane to form a compact readout of the present film address, to count the frames while slewing, and to perform the transverse alignment of the frame after reading the designated frame address. Light pipes are used to minimize the actual area of the sensors in the film gate and to reduce the possibility of crosstalk in the addressing characters. The output of the light pipes is coupled directly to the phototransistors in the film transport electronics package.



# MICROFILM DATA TRANSPORT



## FILM CONSIDERATIONS AND CONSTRAINTS

Several potential problems were investigated with the types of film and the tape transport mechanisms which are used. One problem concerns registration of the film in the mechanism. Because of stretch and shrinkage caused by several factors, registration of the film is made more difficult. Another problem has been stretching of the film due to load at the sprocket holes. In order to design the drive mechanism properly, certain mechanical properties of the film must be known.

The results of these problems were investigated for possible misregistration and loss of accuracy. Although results are not finalized it is the opinion of the film suppliers that existing film can now meet the Space Shuttle cockpit environment.

# FILM CONSIDERATIONS

## Physical Properties

**Modulus of Elasticity**

**Temperature Coefficient of Expansion**

**Humidity Coefficient of Expansion**

**Creep**

## Factors

**Temperature (-65 to 210°F)**

**Humidity (very lo to very hi)**

**Load (5 to 50 lbs)**

**Time (Seconds to Hours)**

**Film Specimen History**

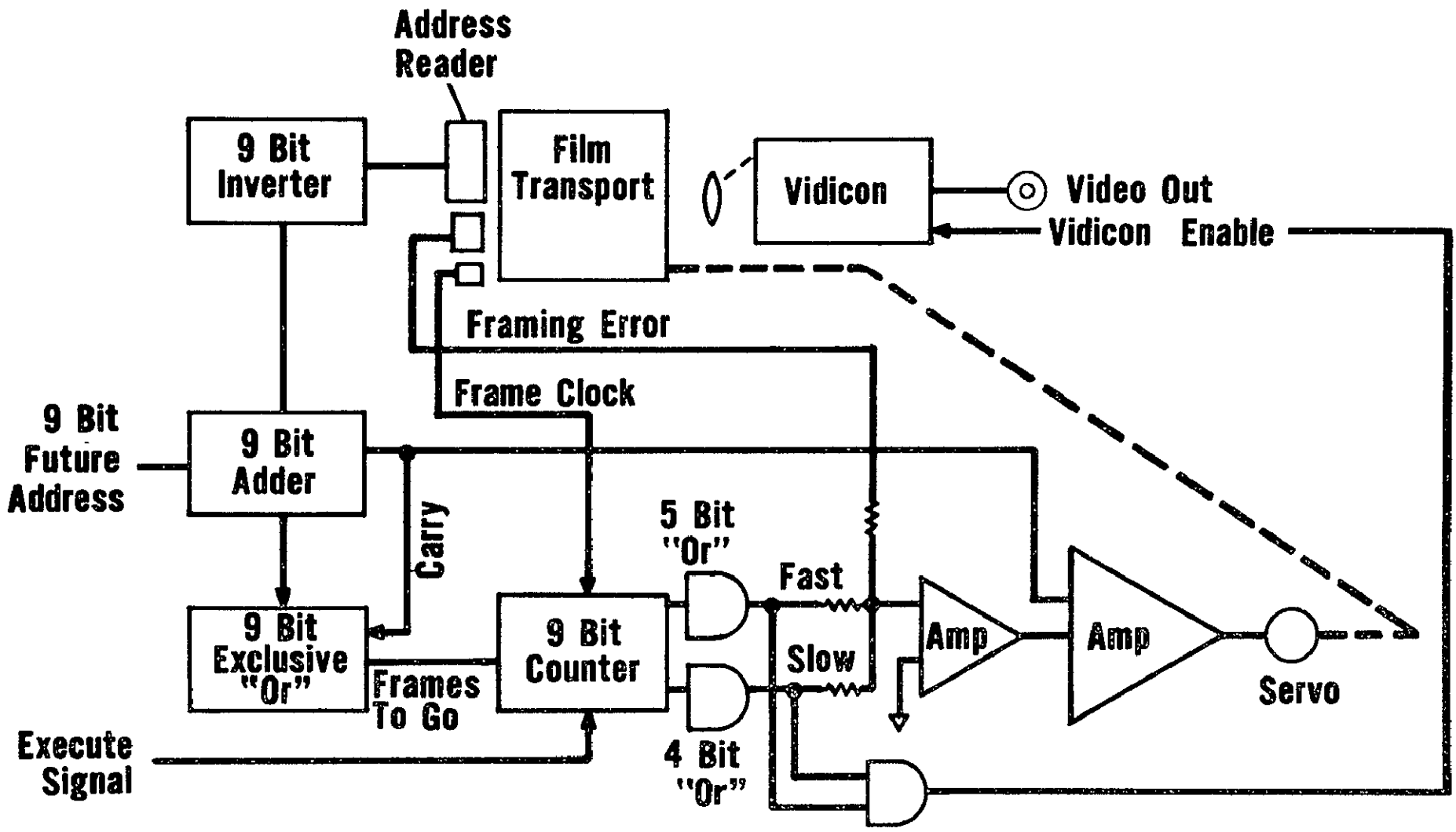
## BDSU AUTOMATIC FRAME SELECTION

The frame selection mechanism may be either locally or computer activated. For either mode, a frame number will be entered into the unit which will search the film file, locate, frame, and display the addressed frame. A block diagram of the frame selector is shown.

When the system is at rest, the address sensor is reading the present address of the film data. This data is complimented and presented as one input to a 9-bit adder. The second input to the adder is the desired address, either from the computer or the local control panel.

The output of the adder is the sum of the complement of the present address and the desired address and is the input to an exclusive "or" gate. The exclusive "or" will either pass or complement the input depending on whether the "carry" from the adder is a one or a zero. The output of the exclusive "or" then presets a counter to the number of frames to go from the present address to the desired or future address.

The counter output, however, remains zero until the execute signal is activated. At that time, the count in the counter is presented to the 5-bit and the 4-bit "or" gates. Until the first frame clock pulse is received by the counter, the output of the 5-bit "or" gate will always be a 'one' on receipt of the execute signal. This places the system in high slew for maximum accelerations from a zero frame rate. After the first count, the output of the counter and therefore the slew rate is determined by the total error between present and future addresses. If the error is greater than 4 bits (15 frames), the system remains in fast slew until the counter has been counted down to within 15 frames of the desired address. At that time, the output of the 5-bit "or" becomes zero and the system is in slow slew. When the count reaches zero, the error is entirely from the framing error sensor. In addition, the output of the vidicon enable gate is activated allowing the frame to be viewed.



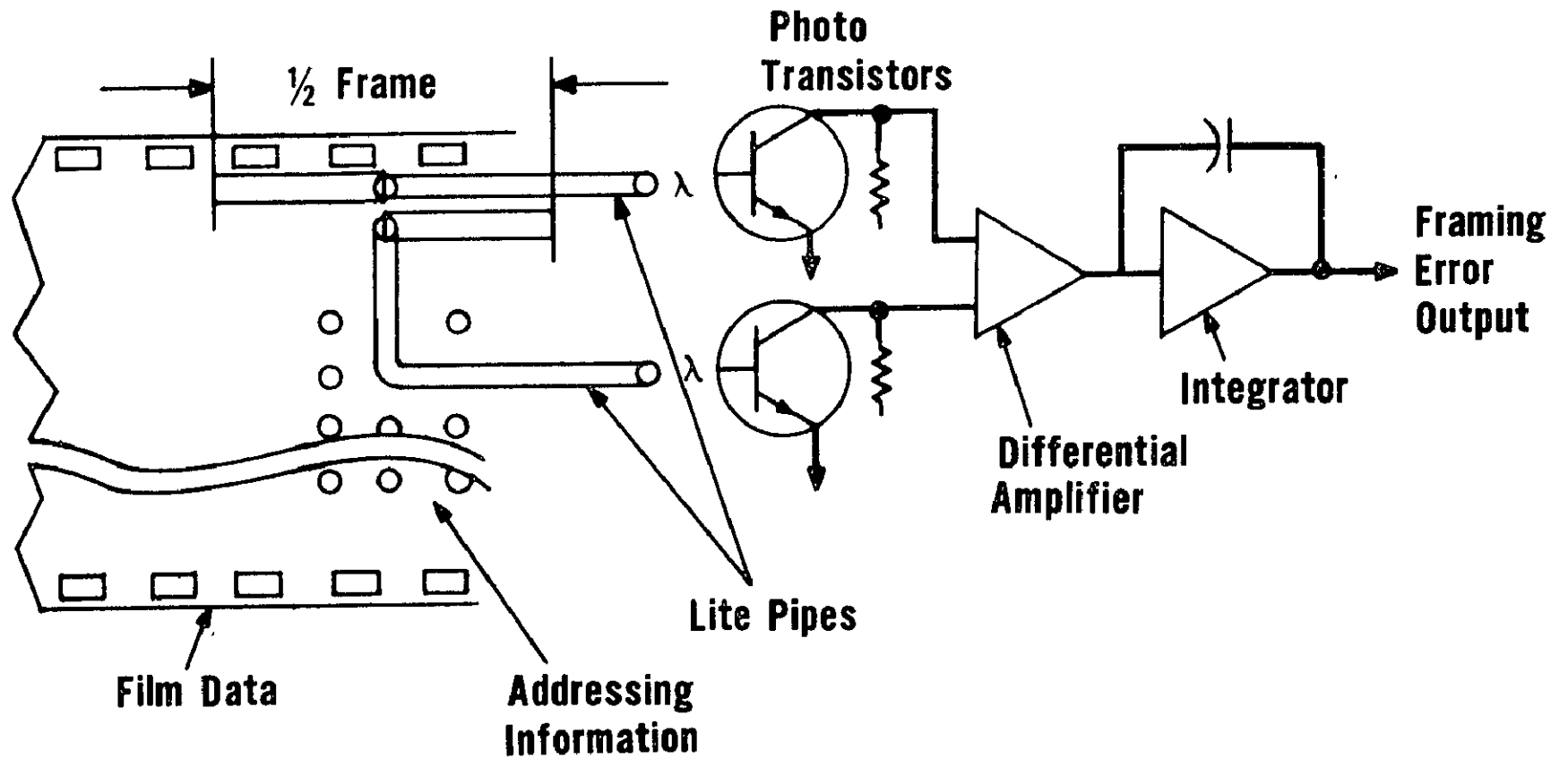
# BDSU AUTOMATIC FRAME SELECT

## FRAME SELECT ERROR SENSOR

The framing error sensor is a dual phototransistor that views a pair of framing slots in the film and corrects the transverse position of the film to null out the output frame sensor. The diagram describes the system schematically.

The integrator is used to minimize the static error in the system, and when the system is slewing, the framing error is approximately zero, since equal pulses to each input to the difference amplifier will average to zero on the integrator.

Transverse frame alignment is accomplished mechanically through film guides integrally a part of the film cassette. This assures that each frame of information is projected on the target of the vidicon in exactly the same position as all other frames, provided that the framing error slots are precisely located with reference to the film data. If the phase locking of the camera and monitors is not left to human judgement, centering of the frame on the monitor will not be a problem. Precise control of camera sweep linearity and amplitude assures registration of the computed data and the film data.



# FRAME ERROR SENSOR

## BDSU SUBMINIATURE VIDICON CAMERA

A subminiature television camera has been developed by Westinghouse. This design is the cornerstone of the microfilm/vidicon camera reader.

In effecting a significant size and weight improvement in camera electronics, heavy emphasis was placed on the rapidly emerging hybrid integrated circuit fabrication techniques. The EIA sync generator was the first production-oriented hybrid unit designed specifically for use in television systems. Housed in a 1- by 1-in. ceramic flatpack, it has been utilized in a series of military and space cameras, including the Apollo series. Following the sync generator, an all-digital motor logic and phase signal hybrid package was designed and used in later Apollo color cameras. In addition, for BIT, an all-digital gray scale generator hybrid package has been designed and implemented. A list of the camera's characteristics is as follows:

a. Sensor

1/2-in. diameter  
S-18 photoconductor  
0.32-in. scan diagonal  
Electrostatic focus  
Magnetic deflection

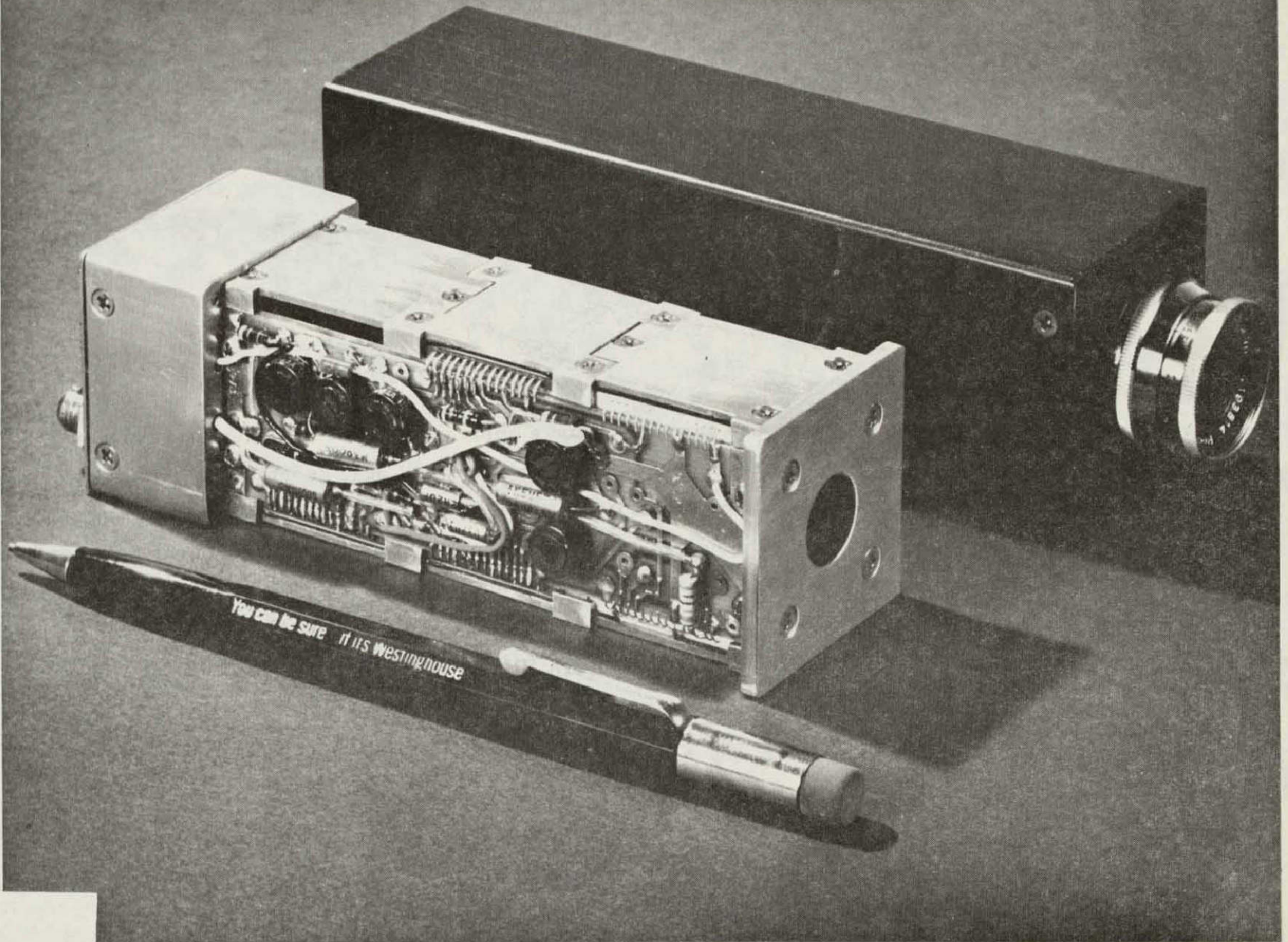
c. System - 450 TV lines/raster height resolution

40 dB signal to noise at 4 fc  
0.6 lb weight  
1.5 by 1.5 by 4.5 in. total size  
6-watt power input

b. Electronics

4.3 aspect ratio  
729 line, 30 frame/second scan format  
2:1 interlace ratio  
21,870 Hz horizontal frequency  
1% linearity  
6 MHz video bandwidth  
IV video into 75-ohm EIA std composite format  
< 6 dB change over 300 l light range

# SUBMINATURE CAMERA

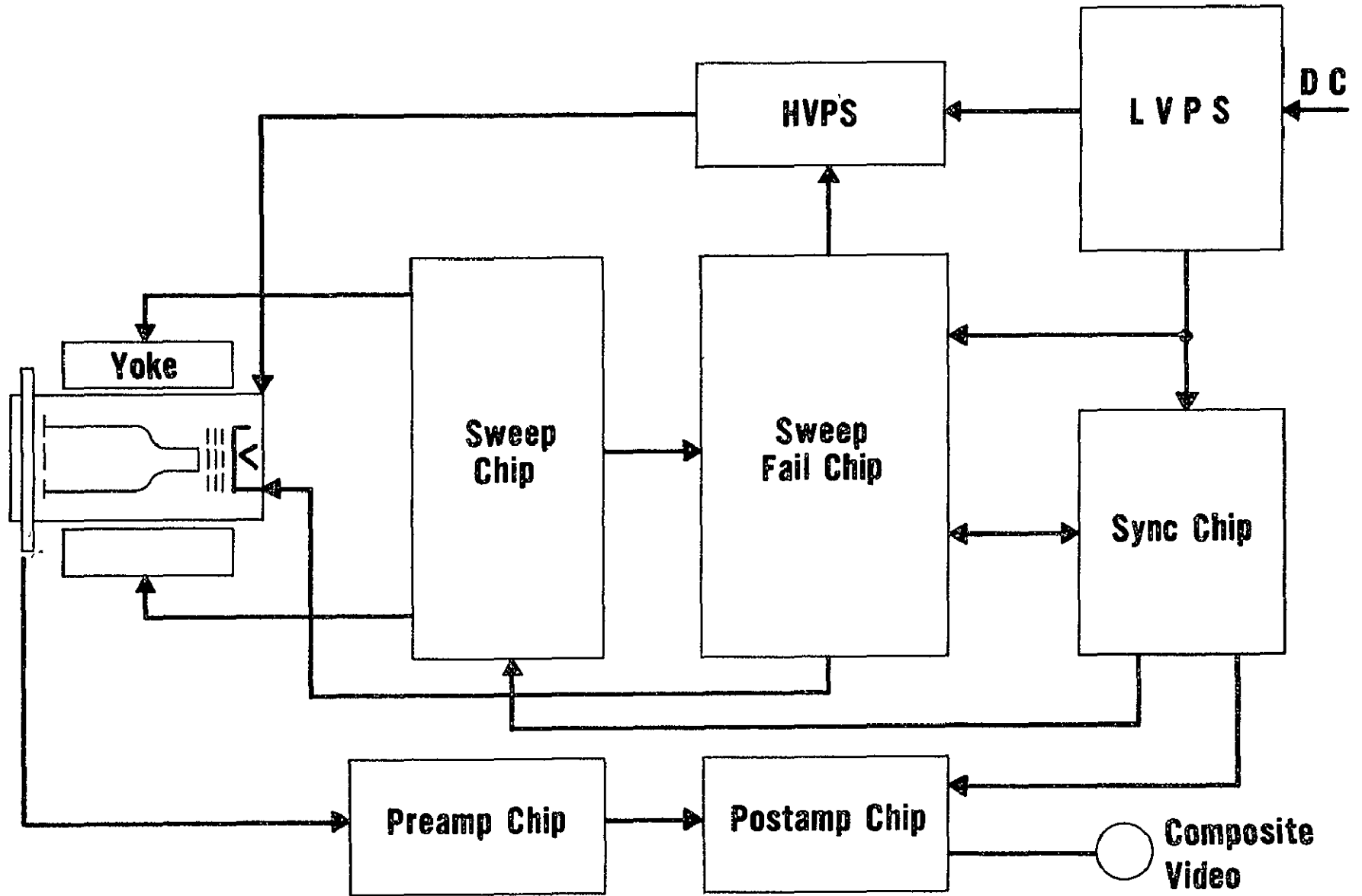




### SUBMINIATURE VIDICON CAMERA BLOCK DIAGRAM

The successful implementation of the EIA sync chip, logic, phase and BIT chips provided the impetus for the development of a subminiature system which will be used in this bulk data storage unit.

The block diagram illustrates the sync generator chip, the sweep fail protection chip and the sweep chip driving the vidicon. The preamplifier package and postamplifier chip provide the output chain of the diagram. The low voltage and high voltage power supplies are hybrid packages and are fed by +28 VDC.



**CAMERA BLOCK DIAGRAM**

## BDSU SYSTEM SUMMARY

In summary the BDSU is a unique device for presenting the required cockpit data in a television format. For the Space Shuttle application this unit can be addressed with digital words and then the respective video can be viewed by the crew. A preliminary prediction of the reliability is 480 failures per million hours or an MTBF of 2080 hours. The absence of gears in the film transport, the accuracy of the framing, the digital I/O, and the utility of the subminiature camera suggest that the unit will have additional applications in future commercial and military vehicles.

# BDSU SYSTEM SPECIFICATION

<b>Slew Rate</b>	<b>100 Frames/Sec</b>
<b>Framing Accuracy</b>	<b>H 1% of Width V 1% of Height</b>
<b>Film Size</b>	<b>16MM Cassette</b>
<b>Address</b>	<b>Digital Code</b>
<b>Drive</b>	<b>Direct Servo</b>
<b>Power</b>	<b>20W→40W</b>
<b>Size</b>	<b>8.8" x 6.5" x 4.5"</b>
<b>Packaging</b>	<b>MSI</b>
<b>Capacity</b>	<b>500→1000 Frames</b>

PRECEDING PAGE BLANK NOT FILMED

MULTI-PURPOSE DISPLAYS FOR SPACE SHUTTLE

A. P. Yeager and Z. G. Tygielski  
IBM/FSD Electronics Systems Center  
Owego, New York

and

F. L. Holmes  
IBM/FSD Electronics Systems Center  
Huntsville, Alabama

1 N71-35064

## ABSTRACT

A concept is described utilizing state-of-the-art technologies to form an integrated display system suitable for computer management and optimized for pilot acceptability. A capability is provided to display on command/demand any one of several flight displays, e.g., Aero Mode Flight Attitude, Space Flight Attitude, and Horizontal Situation type displays. For flexibility and growth, a general graphic alphanumeric capability is provided to format such displays as Engine Parameters, Subsystem Summaries and detailed Subsystem Status displays.

## INTRODUCTION

The Programmable Integrated Display System Design presented in this paper, provides the two man Space Shuttle crews with information necessary for effective vehicle control and subsystem management. The design has been adapted for computer interface and maximum on-board mission management. The Data Management Computer (DMC) complex would process related display data and form composite displays for effective crew comprehension and response. The main body of this paper describes a Multi-purpose Display System design approach/concept which will satisfy Space Shuttle requirements. (Figure 1 lists, in order, this paper's contents).

Figure 1. INTRODUCTION

- Space Shuttle Display System Requirements
- Space Shuttle Display System Design Approach
- Integrated Display Format Requirements
- Multi-purpose Display System Functional Block Diagram
- Display Electronics Functional Schematic

## SPACE SHUTTLE DISPLAY SYSTEM REQUIREMENTS

A two man flight crew in the Orbiter as well as a two man flight crew in the Booster must be capable of controlling their respective vehicles in Aero Flight Mode as well as Space Flight Mode with current avionic systems modified to accommodate the Space Shuttle requirements. To increase the probability of mission success, a Fail Operational/Failsafe fault tolerant requirement is placed on the Display and Control System. Computer managed integrated displays are required to reduce the cockpit real estate required for effective aerospace flight and sub-system management. Related data would be computer processed and organized before presentation to avoid overloading the crew with unnecessary data. These integrated mission related data sets would be presented on a single display surface to ease interpretation. For critical safety-of-flight functions, backup displays which can not be disabled by malfunctions; by damage to automatic, computation/control equipment; or by damage to normal power sources are required. Automatic Checkout and Fault Isolation (COFI) is required to simplify the crew's task of system reconfiguration.

It is purposely pointed out here that it is neither necessary nor desirable to integrate all displays. The final integrated displays would be determined from Mission Time Line Analysis, Pilot Work Load Analysis, and simulation studies.



Figure 2. SPACE SHUTTLE DISPLAY SYSTEM REQUIREMENTS

- Two Man Flight Crew
- Redundant Fault Tolerant System (FO/FS)
- Use Available Avionic Equipment/Technology Modified for SS to Reduce Risk & DDT&E Costs
- Computer Managed Integrated Information Presentation
- Automatic Checkout & Fault Isolation
- Data Management Computer Interface via Std. Data Bus

## SPACE SHUTTLE DISPLAY SYSTEM DESIGN APPROACH

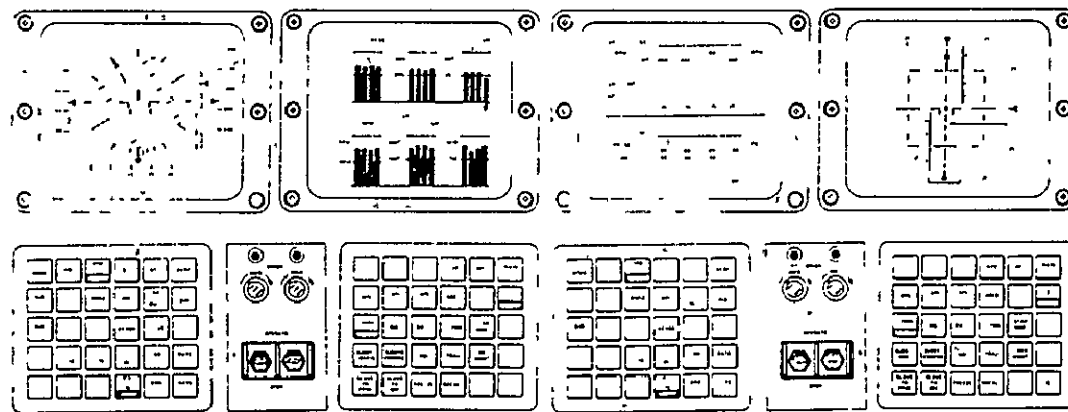
A multi-purpose display system utilizing four CRT's, two per crew member, is the design approach taken in this paper. Primary flight and subsystem data displays would be integrated for presentation to the crew. Each CRT would be capable of displaying either flight or subsystem displays, i.e., formats are interchangeable from one CRT to the other. Such a configuration exploits the inherent flexibility of modern CRT's and the capability of digital computers to manage such a display system.

The display equipment/technology presented in this paper is drawn from the F-14 Multi-purpose Display Indicator Group (MDIG) currently in production at IBM, Owego, New York. The F-14 Display Electronics would be made programmable and modified to be compatible with the Space Shuttle Computer complex. This approach provides a maximum of flexibility for future growth, e.g., new/modified formats would be added by modifying DMC display software.

The design approach is basically the same for the Booster and Orbiter. The four CRT displays, illustrated in Figure 3, are located on the main forward panel. Representative Display Select Keyboards, located below each CRT, are also illustrated in Figure 3 for completeness.

Advantages of computer managed integrated displays are flexibility, improved systems management, and the capability to process/format large amounts of data prior to presentation to the crew members.

Figure 3. SPACE SHUTTLE DISPLAY SYSTEM DESIGN APPROACH



- Primary Flight/Subsystem Data Displays Integrated
- Four Multi-purpose CRT Displays (Two Pilot, Two Commander)
- Use Modified Current A/C Equipment/Technology (e.g., F-14 MDIG)
- Subsystem Data and Flight Displays Interchangeable
- Growth for Display Format Changes thru Software

## INTEGRATED DISPLAY FORMAT REQUIREMENTS

Although all of the precise formats presented on the crew's CRT's are not known at this time . . . as was mentioned previously, they would be determined through mission, time-line, and pilot workload analyses and simulation . . . some general comments may be made and some representative formats illustrated.

As a baseline concept, information requiring proportional judgment tasks should be presented in analog form. Other information displays such as check lists, would be in alphanumeric form. Where possible, display elements should resemble if not duplicate comparable analog displays. This would insure pilot's acceptance through familiarity. For example, an Electronic Attitude Display should be comparable with a conventional mechanical ADI. Each format should then be optimized to reduce clutter, to reduce power required in the electronics and to remove unnecessary data which may cause pilot errors. The particular format presented on each CRT would normally be automatically selected as a function of mission phase with a manual override capability.

The exact number of unique integrated display formats required for the Space Shuttle mission would be dependent upon the final system design. Fifteen to 20 of these might be guidance and flight control displays, of varying complexities, which would be dependent upon mission phase. The remaining display formats would be Mission Situations and Potentials, Critical Systems Configuration and Condition, Consumable Status, Subsystem Summary Status, Subsystem Detail Status, etc. Formats would be constructed using vectors, alphanumerics, special symbols, bar graphs, and conics.

Representative format types and their construction are listed in Figure 4. Figures 5 through 8 illustrate particular displays which might be presented during various phases. It should be emphasized that these formats are preliminary and are a result of Space Shuttle Phase B Display studies conducted by North American Rockwell and IBM. They do, however, reflect the versatility of an integrated display approach.

Figure 4. INTEGRATED DISPLAY FORMAT REQUIREMENTS

- Number of Unique Formats Required are a Function of Final System Configuration
- Format Construction (Example)
  - Vector/Graphic + Alphanumeric (Flt. Attitude)
  - Alphanumeric Tabular (Subsy. Status Summary)
  - Multiple Bar Graph +  $\alpha/N$  (Subsy. Status Summary)
  - Graphic Conics +  $\alpha/N$  (Subsy. Configuration)
  - Special Symbols +  $\alpha/N$  (Star Chart)
- Format Selection Normally Automatic, as a Function of Mission Phase/Flt Mode, with Manual Override/Select

Figure 5. AERO MODE ATTITUDE

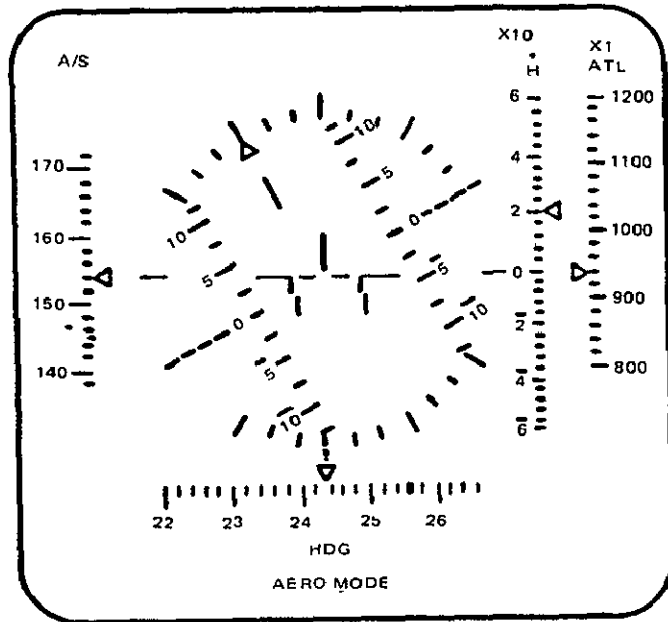


Figure 6. SPACE MODE ATTITUDE

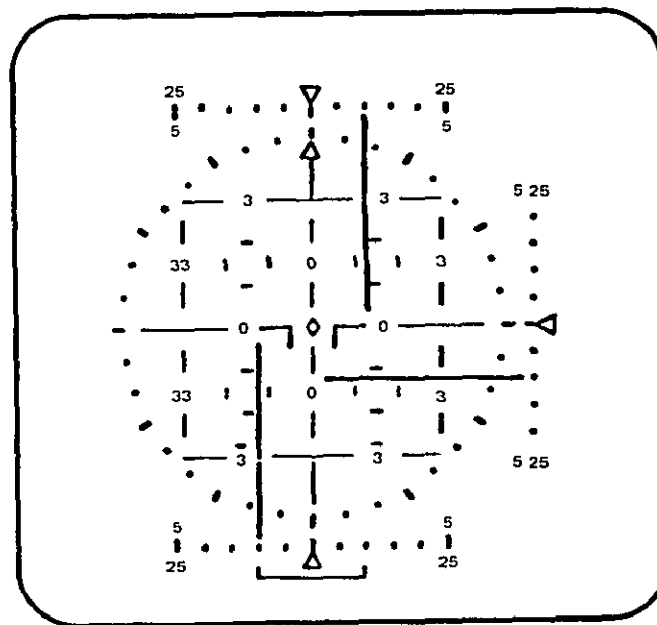


Figure 7. HYD/APU SUMMARY

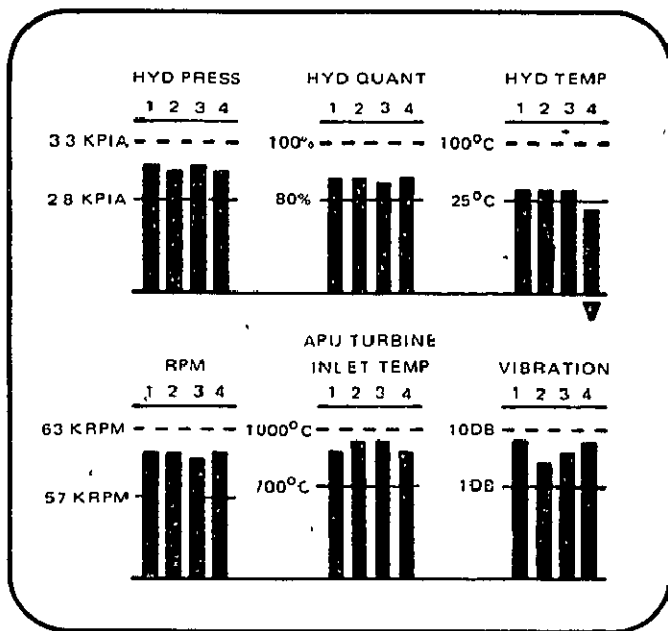
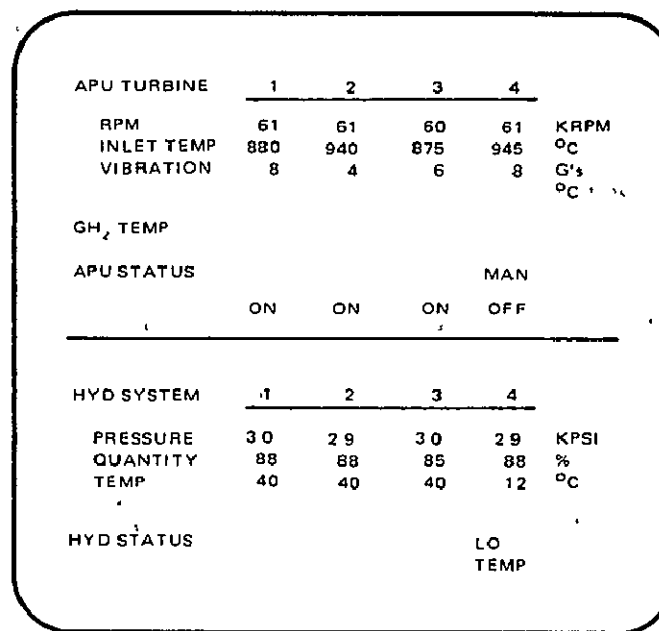


Figure 8. HYD/APU STATUS 2



## MULTIPURPOSE DISPLAY FUNCTIONAL BLOCK DIAGRAM

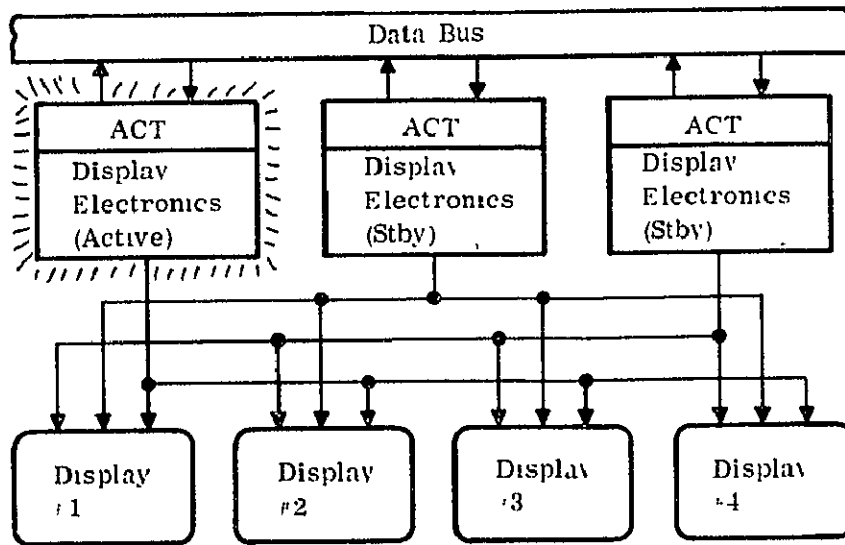
The Multipurpose Display System, functionally illustrated in Figure 9, is configured to meet the Space Shuttle FO/FS requirements. Each Display Electronics Unit (DEU) is capable of refreshing the four display units with independent display formats. Should the active DEU fail, a standby DEU would resume the task of display refresh.

Functionally, the DEU will: (1) interface with the DMC via the data bus; (2) store display formats; (3) provide display refresh; (4) perform the operations necessary to format position, rotate and translate characters and symbols; and (5) provide the appropriate deflection and blanking signals to the display indicator. Selected formats will be permanently stored in the DEU to minimize DMC interface. The remaining formats will be stored externally in a mass memory and selectively loaded into the DEU via the data bus under DCM control. The Indicator Units contain the necessary deflection amplifiers, high voltage, and video circuitry required in a standard CRT display.

To meet the requirements of Space Shuttle, a random positive stroke writing technique has been selected for character generation. This technique is very versatile since, in effect, an infinite character/symbol repertoire is available for format construction. Once the system character/symbol repertoire has been determined, the appropriate stroke data is stored in the Display Electronics memory. In addition, the circuitry designed for rotation and translation is independent of character changes when a stroke technique is implemented. An advantage of this technique is that the display dynamics are provided with less hardware and software than, for example, a digital TV approach.



Figure 9. MULTIPURPOSE DISPLAY FUNCTIONAL BLOCK DIAGRAM



- Selected Formats Stored in Display Electronics Memory Remaining Formats Stored in Mass Memory
- Formats Loaded into Display Electronics Storage via Data Bus under DCM Computer Control
- CRT Displays Constructed and Refreshed by DEU
- Display Electronics Constructs Blanking & Deflection Signals to Position, Translate and or Rotate Symbols from Format Control Words
- Data Update Required for Dynamic Displays Supplied by DCM Computer via Data Bus

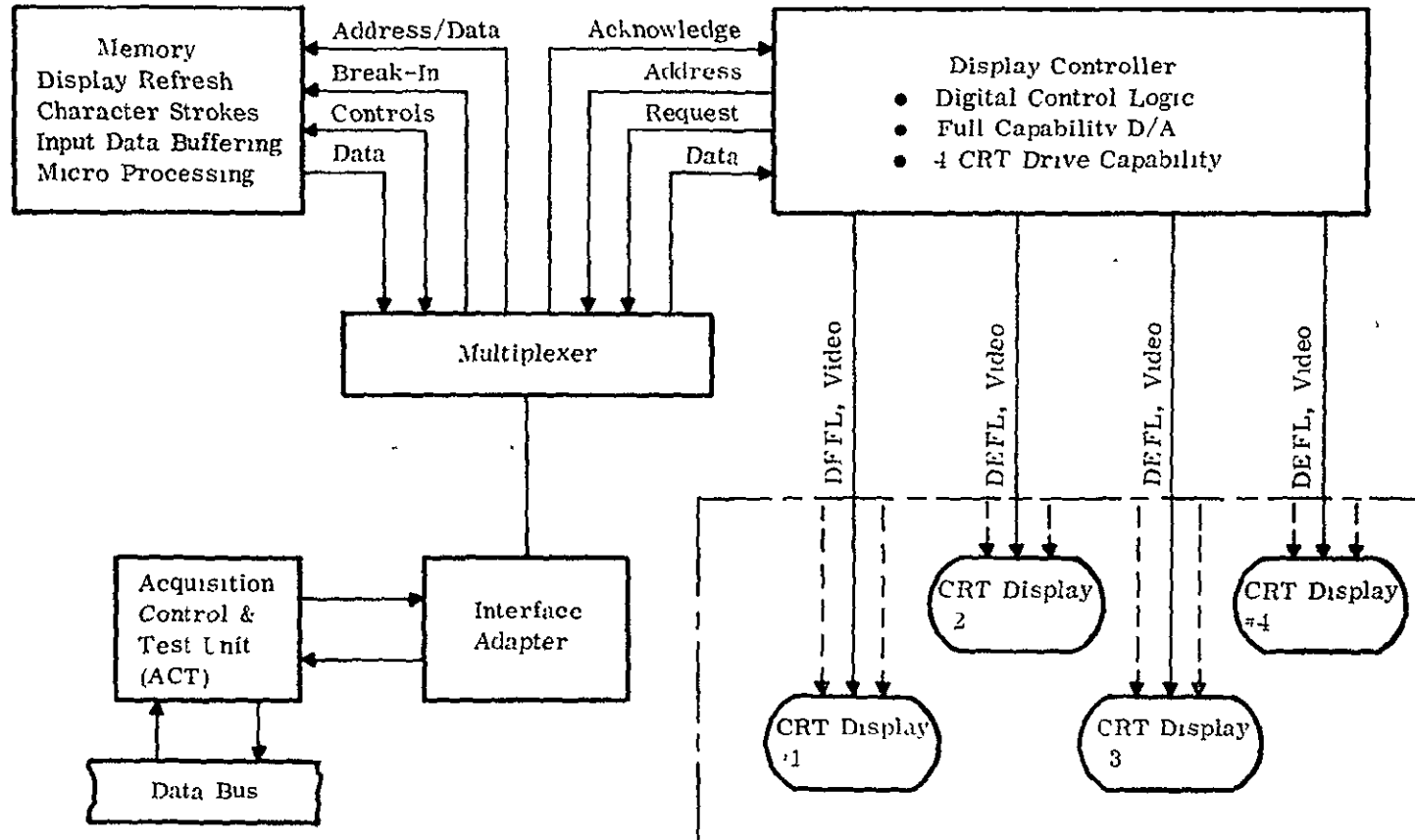
## MULTIPURPOSE DISPLAY ELECTRONICS FUNCTIONAL SCHEMATIC

A functional schematic of the Display Electronics is illustrated in Figure 10. It includes six functional units as follows: Interface Adapter Unit, Display Storage Unit, Data Multiplexing Channel, Format and Symbol Controller and Display Controller.

The Interface Adapter Unit is the element which receives and transmits signals between the MPDS and the DCM computer. All I/O operations are routed to the Data Multiplexing Channel via a request-acknowledge interlocking type interface. The computer inputs are the primary sources for display control information. Updates for all variable display parameters and control discrettes are communicated to the MPDS via the computer interface. In addition, the computer is capable of loading, verifying or changing the MPDS memory.

The Display Storage Unit is composed of a mixture of read/write and read-only storage. This mixture provides a storage unit which is flexible in the various display formats it can construct and the diverse symbols it can draw because of its read/write memory. Yet, it also provides non-volatile and economical read-only storage (ROS) for display symbols and formats which are not subject to change. The memory mixture shares common control hardware, address registers, and data registers.

Figure 10. MULTIPURPOSE DISPLAY ELECTRONICS FUNCTIONAL SCHEMATIC



The Read/Write section of the DSI is easily loaded or changed in two ways - either via the computer interface of using a special memory loader/verifier unit. This feature enables display formats and symbols to be changed by merely altering the memory contents. The memory is functionally partitioned into six areas - Input Data Buffer, Display Refresh Buffer, Symbol Stroke Information, Standard Format Controls, Processing Programs and Read/Write Memory Pool.

The Input Data Buffer is a memory block used to store variable input data as it is received from the computer. The Format and Symbol Controller continually interrogate this buffer for new input information.

The Display Refresh Buffer contains a list of Format Control Words which are decoded every refresh cycle and used to specify display formats. Format control words define position, angle of rotation, and character form for each display symbol.

The Symbol and Stroke Control portion of memory contains Stroke Control Words (SCW) which contain the detailed stroke information required to generate each symbol in the display system's character repertoire.

The primary function of the Format and Symbol Controller is to interrogate the input data buffer for newly transmitted data, to process the data and to make the required changes, additions, or deletions to the refresh buffer list.

The Display Controller sequentially reads the Format Program list and one-by-one executes each instruction until the desired display format is constructed. It contains both digital logic and analog circuitry for converting digital input words -- both format and stroke controls -- to analog deflection signals.

Information required to refresh each display indicator is stored as a sequential program list in memory. The program repetition rate, i.e., display refresh rate, is controlled by timing circuitry internal to the Display Controller. Fifty-five times per second (once every 18 milliseconds), the Display Controller interrogates the first sequential format control word stored in the format program. It then continues down the program list executing the operation defined by each word until one of the format control words signifies the end of the list. The Controller then becomes idle and awaits the beginning of the next refresh cycle.

Another portion of memory area stores standard formats and portions of formats which are not subject to change. It is a non-changeable supplement of the refresh buffer and contains format control words. It includes detail information for constructing such standard display as the compass and scale markings.

The Processing of variable input data words is controlled by a set of programs, stored in their own unique memory area, which include instruction words and control for operations such as: calculation of sine, and cosine, binary-to-binary coded decimal conversions, vector calculations, intensity control, mode control, and symbol motion control.

A special read/write memory pool is provided to add flexibility to the system. Any of the six above mentioned areas can use any portion of it as a supplement to their existing memory allocations. This provides the following possibilities: increased input data or refresh buffer size, changeable symbols or formats under program control, and/or changeable programming capabilities.

The Data Multiplexing Channel controls the data flow to and from memory. It receives requests to perform memory access operations from the Interface Adapter Unit and the Display Controller. At any given time, the data channel gives priority to the Display Controller.

The analog portion of the Display Controller performs digital-to-analog conversion as well as analog-to-analog conversion and is the signal interface to the display heads. Functions performed in the analog circuitry are as follows:

- Beam Position
- Vector, Conic and Character Generation
- Symbology Rotation
- Video Generation and Scaling
- Deflection and Video Multiplexing

## DESCRIPTION OF DATA MANAGEMENT SYSTEM

The function of the data management system for the Phase B Space Shuttle is to provide a data transmission and data processing service for the avionics equipment and for other vehicle subsystems. It performs all the on-board data processing except that assigned to special processors for engine control, and for analog stability augmentation.

The system consists of quadruply-redundant data buses and computers, and associated equipment. An Input-Output Control Unit (IOCU) is dedicated to each computer. It can operate or monitor all four data buses under software control. It has direct access to a portion of the computer's memory. The System Control Unit (SCU) designates the computer-IOCU that is in control of the data bus and compares status information from redundant computers.

Digital interface units (DIU) in each bay connect the line replaceable units (LRU) to the data bus. The DIU receives and transmits bus messages, multiplex and demultiplex data, performs the analog-digital conversion, applies excitation and stimuli to some of the sensors, and collects status information.

The Bulk Memory stores a copy of the flight program for use if a reload is necessary. It also stores alternate landing site constants and provides a means for loading maintenance tapes in the hangar for subsystem checkout.

The data management system monitors and displays checkout parameters of the LRUs, performs signal switching in case of failure, and performs power switching.



PRECEDING PAGE BLANK NOT FILMED

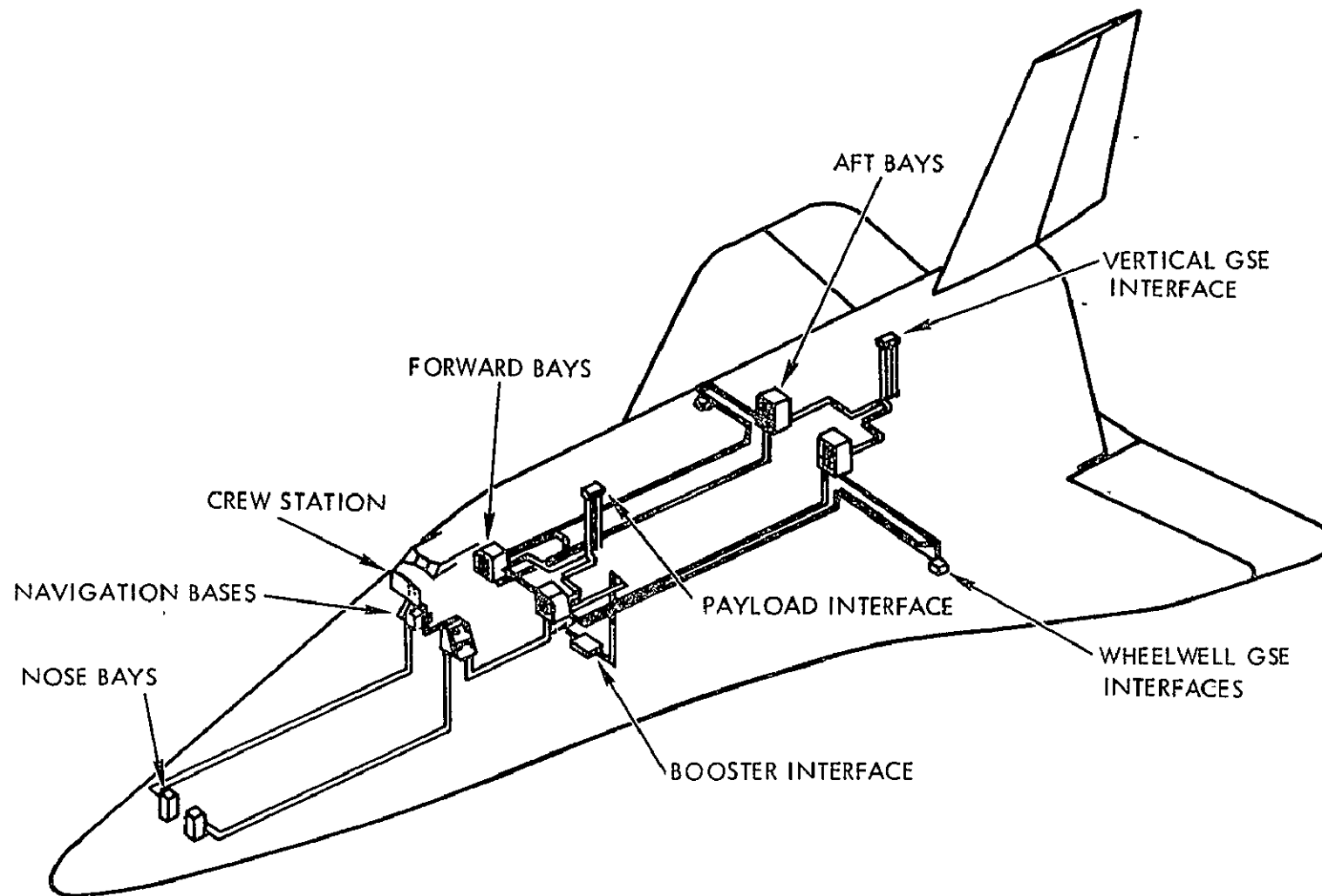
SELF-TEST TECHNIQUES FOR REDUNDANT DATA MANAGEMENT SYSTEMS

Myron Kayton and H. S. E. Tsou

TRW Systems Group  
Redondo Beach, California

N71-35065

# DATA MANAGEMENT SYSTEM



## DATA MANAGEMENT DESIGN AND SELF-TEST

The data management system is required to process engine control, autopilot, navigation, and subsystem monitoring signals, of which most are essential for crew safety. As a result, the principal consideration in the system design has been to provide means of detecting failures, and of switching the elements of the data management system. The design utilizes a hierarchy of control:

Crew controls computers via SCU panel, keyboard, and cathode ray tubes.

Crew may select the manual or automatic mode of reconfiguration after a failure.

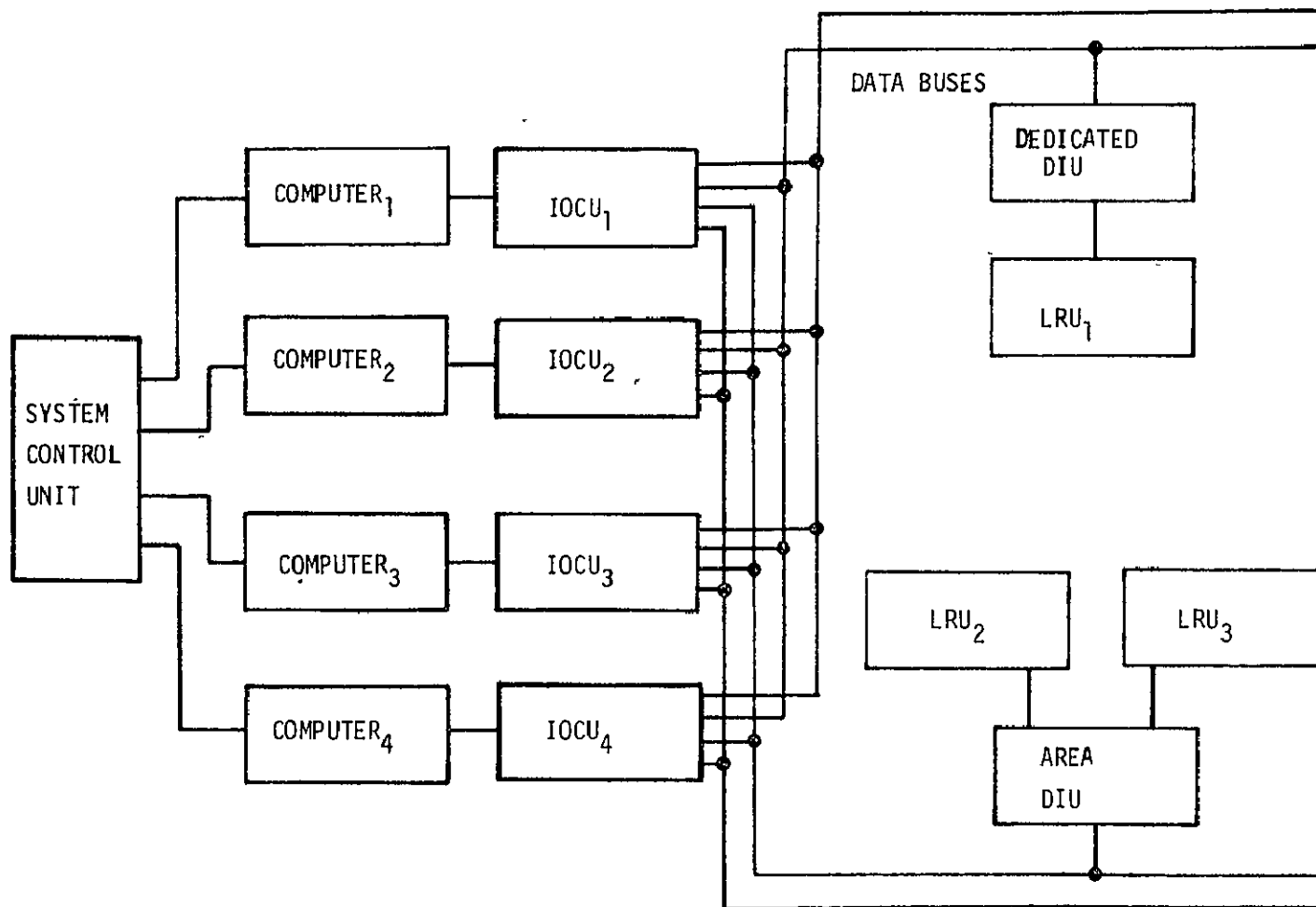
Computer controls testing of IOCU and data buses.

DIU controls its own self-test and also generates stimuli for testing LRUs.

LRUs may test themselves and may vote multiple bus inputs.

The avionic system is designed in two physically separate halves, such that mechanical damage to one half will not affect the other. Replicative redundancy is used in order to meet the "fail-operational, fail-operational, fail-safe" requirement within the two-bay constraint, without excessive equipment. No cross-strapping among the four redundant sheets is used except at the computer complex and at those LRUs that require voters.

DATA MANAGEMENT DESIGN AND SELF-TEST



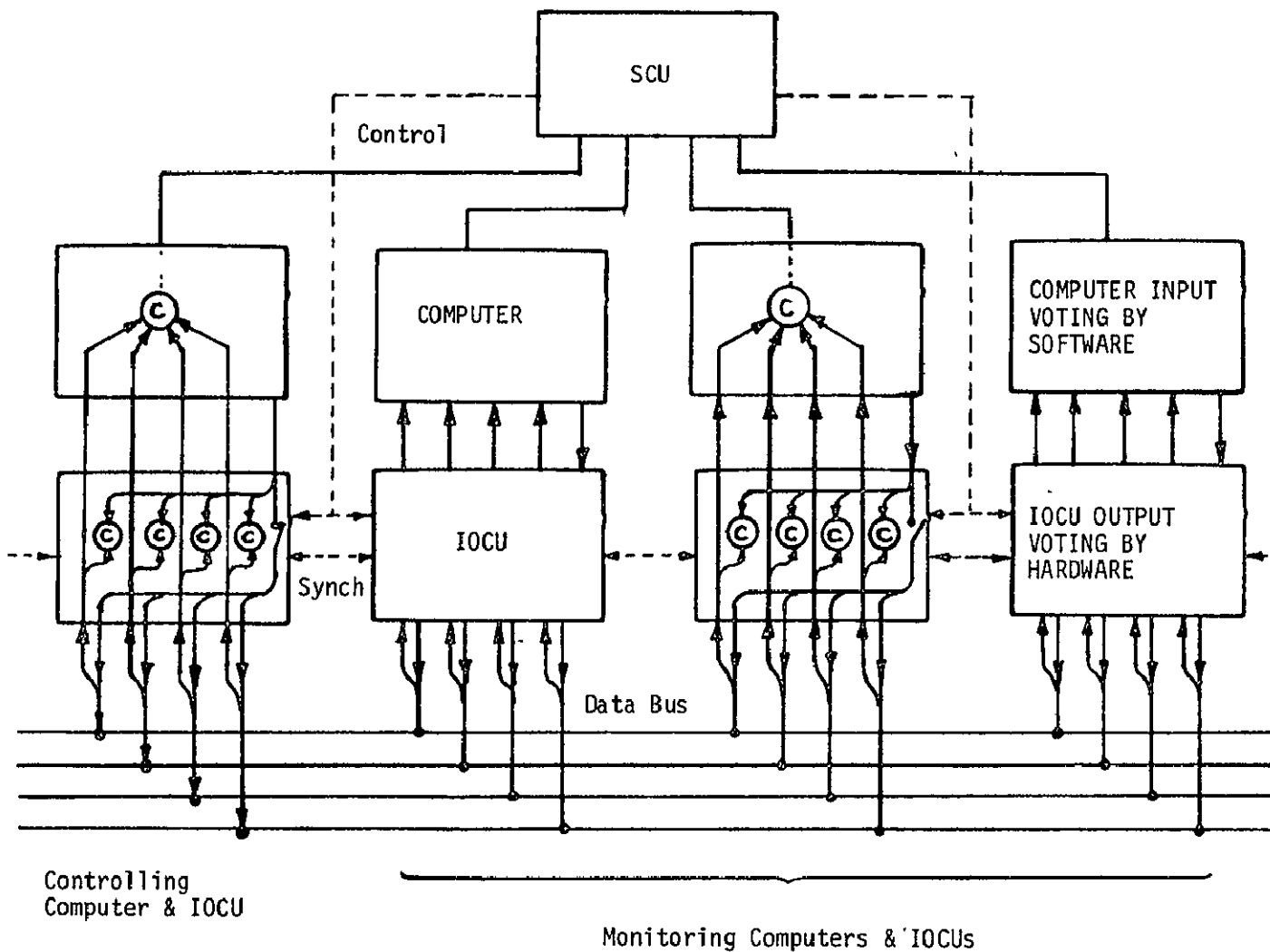
## TEST OF COMPUTER AND IOCU

The controlling computer and the monitoring computers perform the same computations and receive the same inputs. However, only the controlling computer-IOCU issues commands and data to the subsystems. The monitoring IOCU receives the commands and data generated by its own computer in addition to receiving the information on the buses. A discompare will result in switching of the controlling computer or elimination of a monitoring computer-IOCU.

All input data from four (or fewer) data buses are received by all operating computers. These redundant data are compared by software, using the same algorithm in each computer. The end results of the computations are transferred to the IOCU's for transmission on the buses.

Thus, errors in the IOCU and computer are detected by discompares in the IOCU that are sent to the SCU for logical evaluation. A software self-test program in the computers, and self-test circuitry in the computers and IOCU's assist in evaluating the nature of the failure.

TEST OF COMPUTER AND IOCU



315

## FAILURE DETECTION BY THE SYSTEM CONTROL UNIT

The function of the System Control Unit (SCU) and Panel is to apply power to the computers, to display status of the data management system, and to designate the controlling computer. The SCU monitors the following computer parameters at all times:

Built-in test discretes from computer and IOCU

power

clocks

IOCU status words

Software self-test

20 millisecond time-ticks

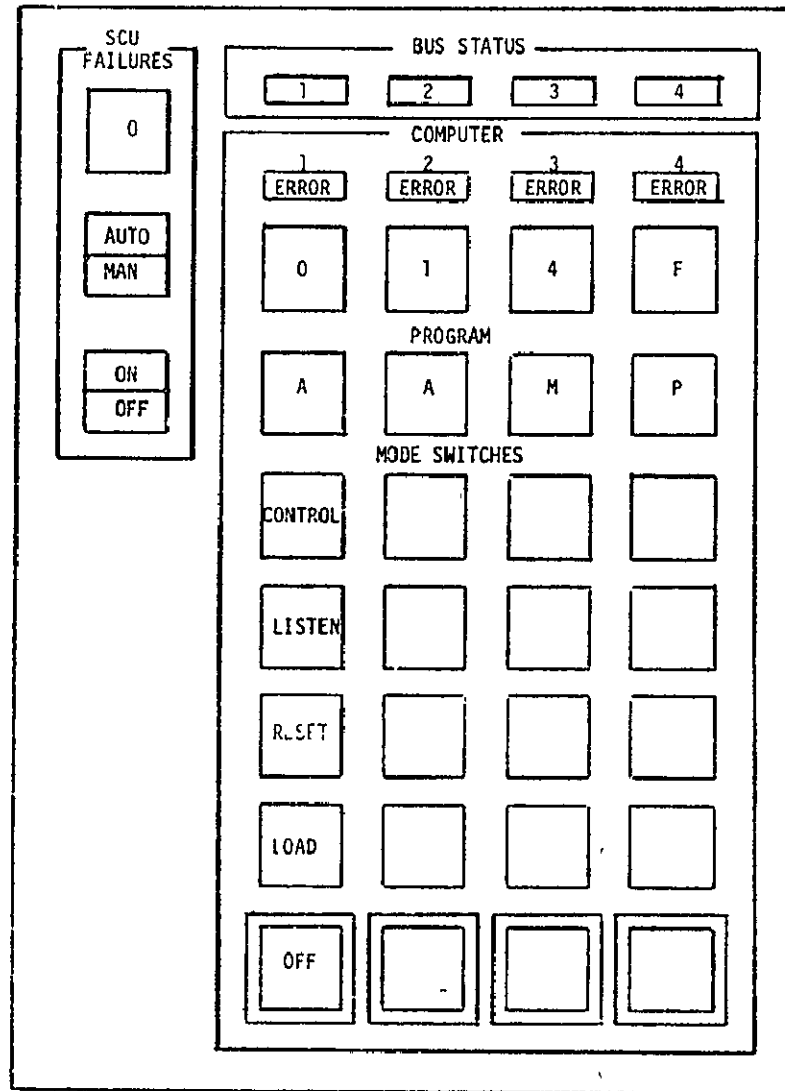
Row and column checks of the reconfiguration tables

The panel allows the computers to be reset after a transient error, and controls the loading of programs into the core memories.

In the manual mode, the crew selects the computer-in-control using the interlocked command switches. In the automatic mode, the SCU chooses the computer-in-control based on the monitored data.

The SCU also displays the bus and DIU status. The crew can monitor failures on the caution-and-warning panel and can perform detailed diagnosis using formats displayed on the cathode ray tubes.

FAILURE DETECTION BY THE SYSTEM CONTROL UNIT





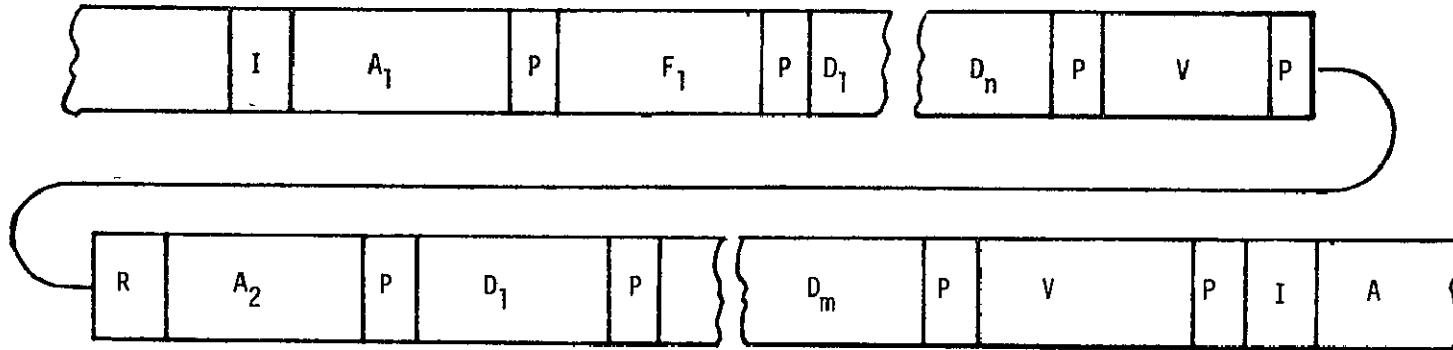
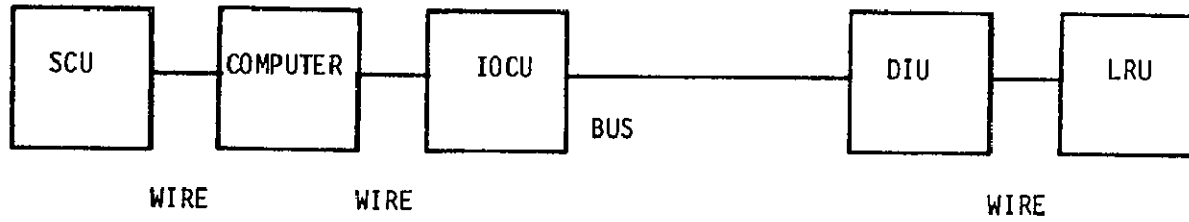
## SELF-TEST OF IOCU AND DATA BUS

The input/output control unit incorporates a number of tests to help detect failures in the transmission system. It detects parity errors resulting from accessing the computer's memory. It uses parity checks, an echo check, and a time-out check for data bus operation.

A typical subsystem message format is shown in the figure. It contains a DIU address, a Function Code, a variable number of data bytes, and a message parity byte. The first field contains the address of the DIU being interrogated. The IOCU stores this address and compares it with the responding DIU address. The IOCU monitors the input/output functions and the word count of the data to be transferred. Each 8-bit data byte in the message is checked by use of a horizontal parity bit, and each message is checked by a vertical parity byte. The IOCU also performs a time-out check of the data bus operation to ensure that a response is received within a preset time. This check prevents a non-responding DIU from halting data bus traffic. The modem in the IOCU receives the data which is being transmitted and performs bit-by-bit comparison checks on some of the data. Commands that constitute potential flight hazards are transmitted several times and compared by the DIU before execution to guard against burst errors of long duration.

The probability of an undetected error is 1 in  $10^{15}$  bits for a single-bit error rate of  $10^{-5}$ .

SELF-TEST OF IOCU AND DATA BUS



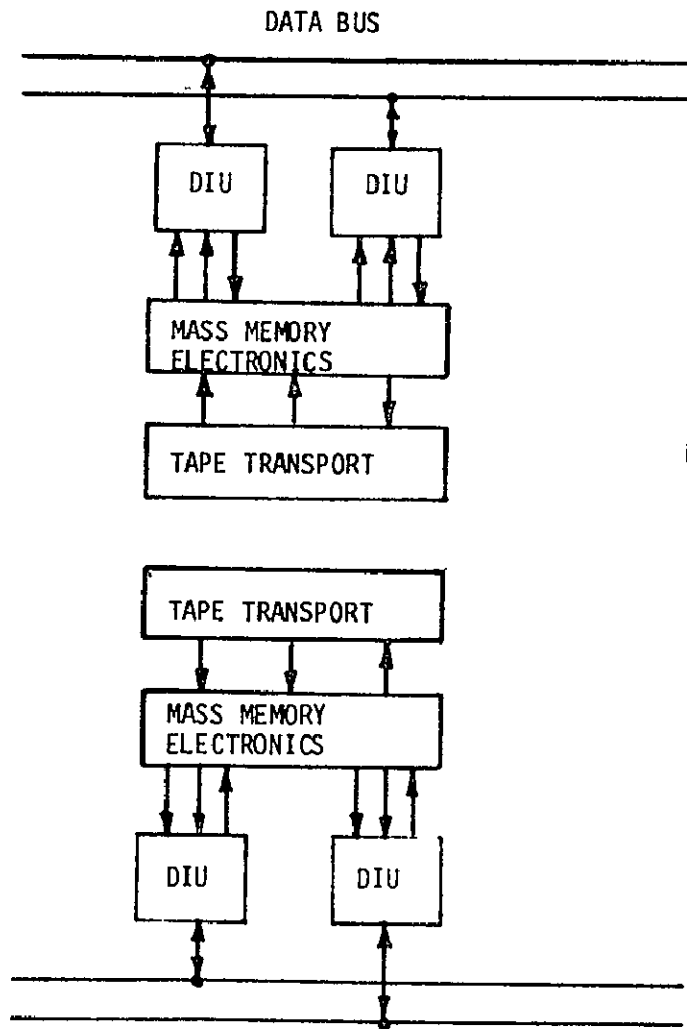
- I : Inter-message gap: minimum 2-bit time
- A<sub>1</sub> : Address of the polled DIU
- F<sub>1</sub> : Function Code
- P : Horizontal (or byte) parity bit
- D : Data bytes
- V : Vertical (or message) parity byte
- R : DIU response gap: 2-bit time

## SELF-TEST OF MASS MEMORY UNIT

The mass memory is not required for mission success or for crew safety. Therefore, the degree of in-flight self-test is not as encompassing as for other data management equipment. Because the two mass memories operate in the read-only mode, the self-test is concerned only with reading from the tapes. The information on the tape is stored in block form - each block containing 128, 32-bit words and a sum check. In order to ensure the accuracy of information read from the tape, the data read into core are compared to the data read from a separate track of the same tape unit via a separate DIU and data bus.

The mass memory units have built-in test monitors to assist self-test. These include motor rotation, temperature, pressure, tape movement, operational mode, and power status.

SELF-TEST OF MASS MEMORY UNIT



DIU issues commands and checks BIT.

Memory Electronics controls tape and generates BIT.

## DIU-LRU FAILURE DETECTION

Redundant sensors are connected as in Figure (a). The sensor outputs are compared in the central computers. This method of connection permits each LRU to be placed into an independent state. Common actuators such as landing gear, drag brake, chute, engines, control surfaces, and attitude control jets are actuated as in Figure (b). Commands are delivered to these devices by one to four paths. The actuators have the option of comparing the redundant signals in order to protect against faults in the DIU and in the wiring harness from DIU to LRU.

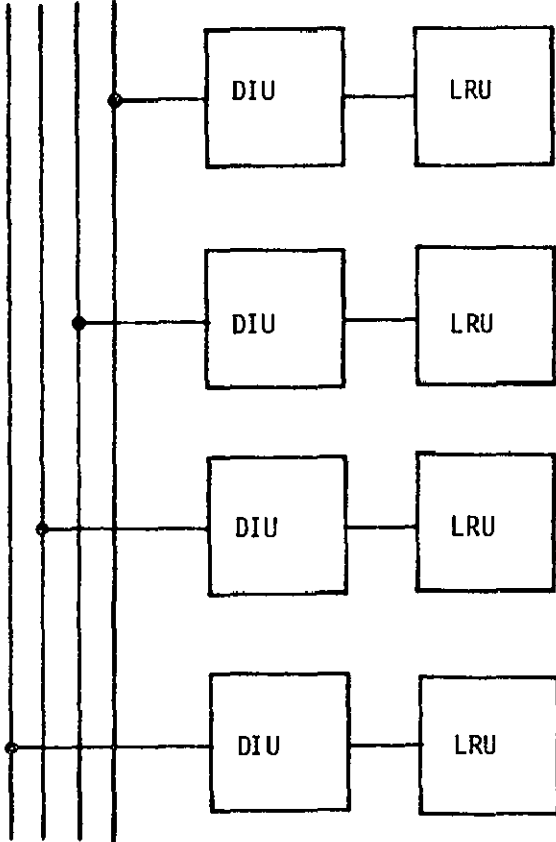
Actuators that incorporate voters must be designed to vote one to four signals depending on the power-on and failure status of each bus. "Data-good" discrettes go from each DIU to the voter, and voter status is sent to the DIU for re-transmission to the control computers.

A further test of the DIU itself is provided by built-in test circuits that monitor the power supply and clock, and check the DIU memory. Analog-digital encoders and digital-analog encoders are checked by sending a test signal down the bus once per second to be decoded, recoded, and sent back to the computer. The test also verifies a portion of the multiplexer in the DIU.

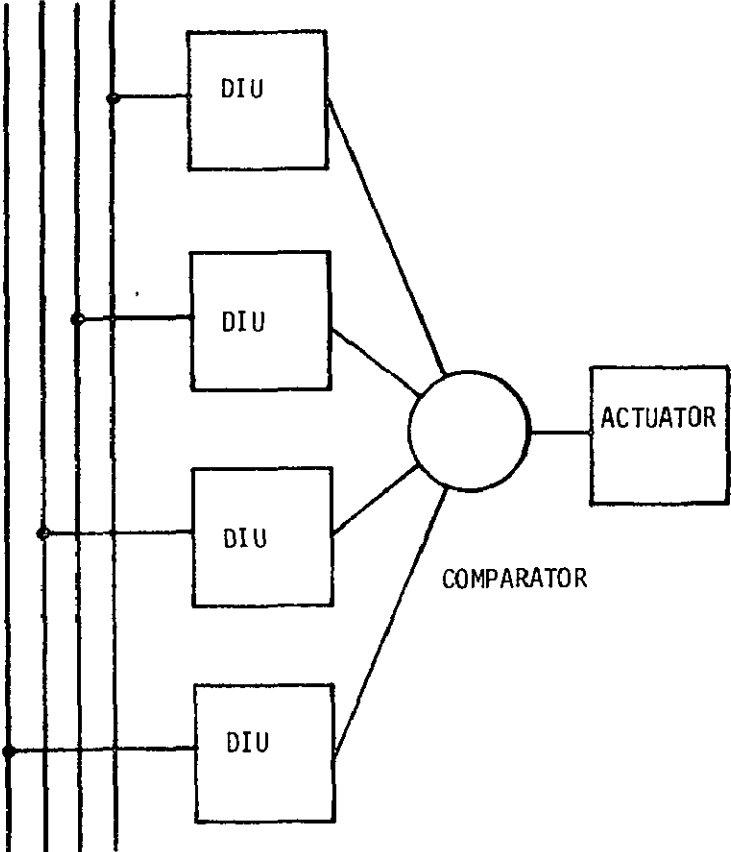
Design of the DIU circuits that apply stimuli to the LRU and measure LRU response is subsystem-peculiar.

DIU-LRU FAILURE DETECTION

a. SENSORS AND INDEPENDENT ACTUATORS



b. COMMON ACTUATORS



## SUMMARY OF DATA MANAGEMENT SELF-TEST

### Computer Receives Data (Figure a)

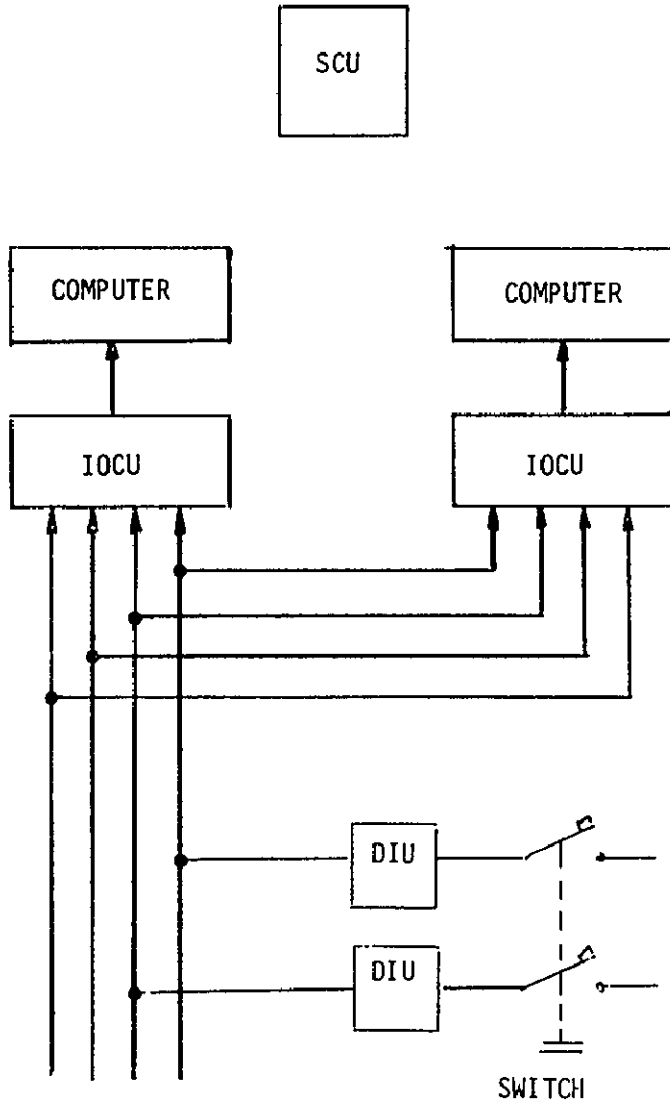
1. The controlling IOCU interrogates DIUs.
2. Each IOCU receives all data from LRUs.
3. Each IOCU transfers data to its computer.
4. Each computer compares redundant data.
  - a. Reasonability test on each signal
    - variable limit test
    - change from last measurement
  - b. Comparisons among redundant information
    - averaging
    - deviation from mean
  - c. Algorithms are subsystem-dependent

### Computer Sends Commands (Figure b)

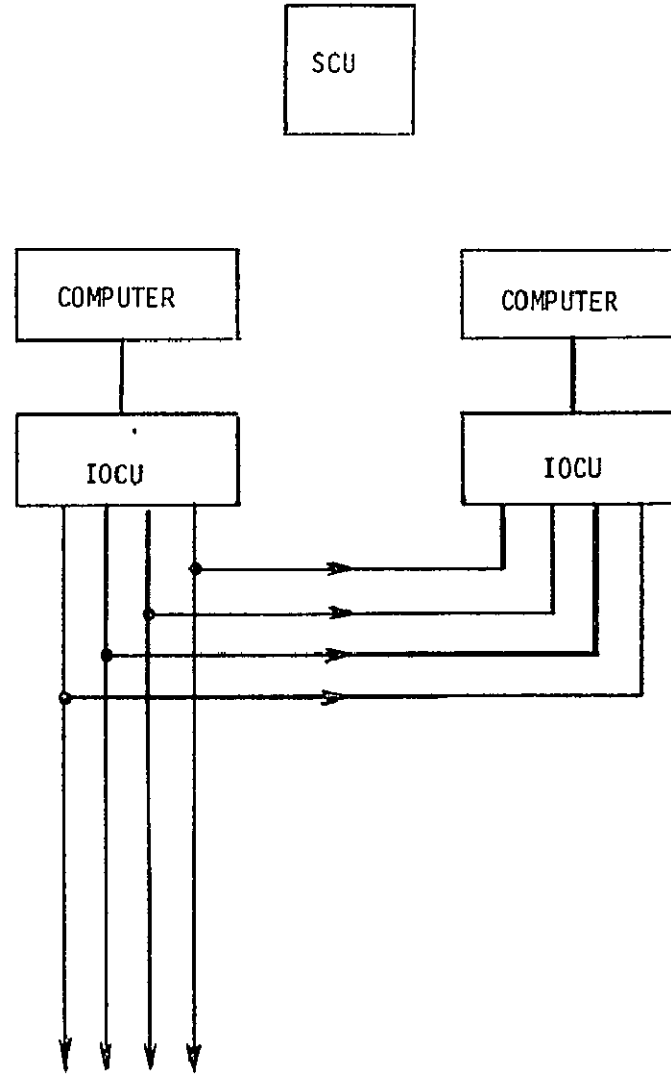
1. Each computer performs identical calculations.
2. Controlling IOCU interrogates panel switches and transmits commands on the bus.
3. Each computer sends commands to its IOCU.
4. Controlling IOCU transmits on four buses (or fewer if some are powered-down).
5. Monitoring IOCU's compare output commands.

SUMMARY OF DATA MANAGEMENT SELF-TEST

(a)



(b)





SUMMARY OF DATA MANAGEMENT SELF-TEST (Continued)

DIU Receives Commands (Figure c)

1. Parity check on each bus.
2. Address echo check on each bus.
3. IOCU repeats critical commands; DIU stores and compares repeated commands.
4. DIU performs closed-loop test and built-in tests.

LRU Receives Commands (Figure d)

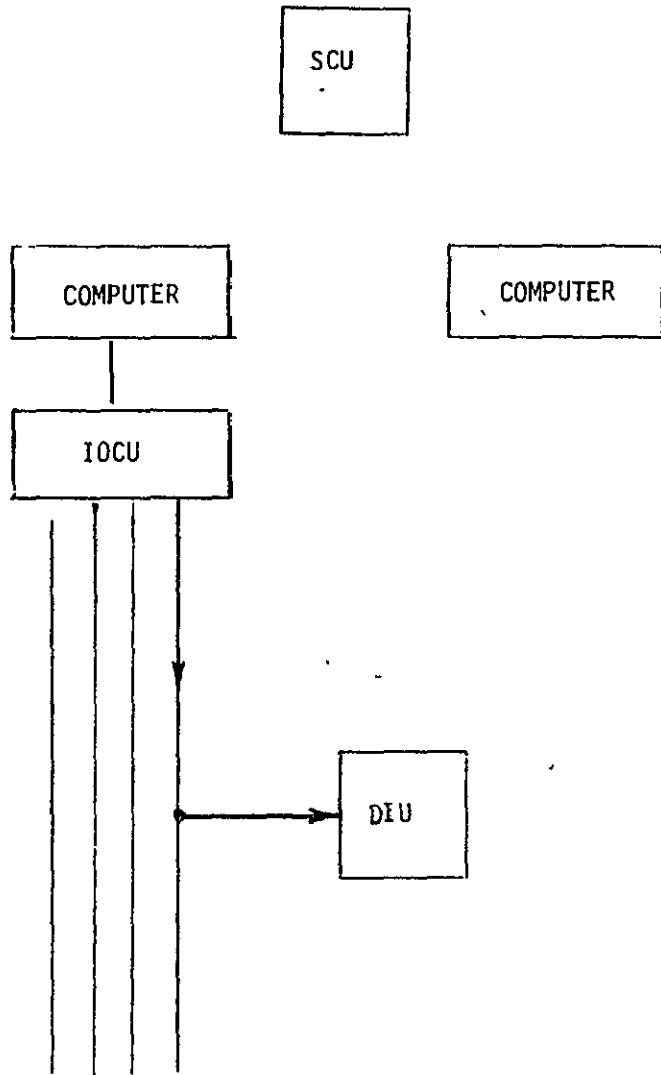
1. LRU votes multiple transmissions.
2. LRU responds to DIU stimuli.

LRU Replies (Figure d)

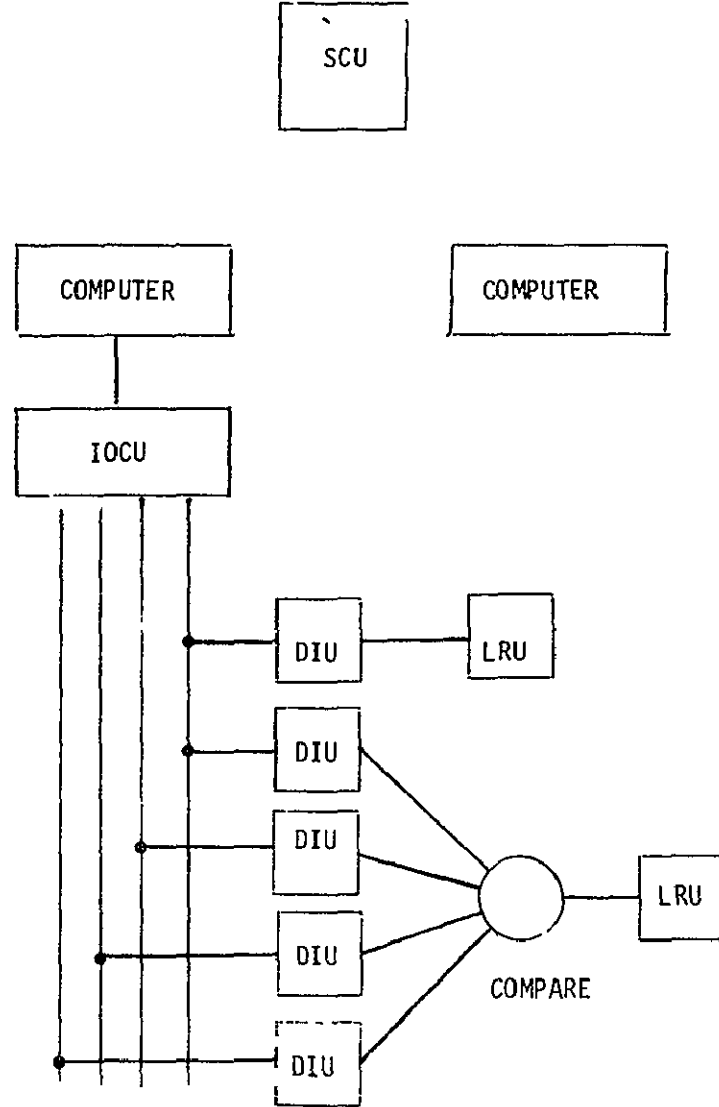
1. DIU samples LRU data.
2. LRU responds to DIU test signal.
3. Bus protection for LRU reply.  
    parity  
    data word count
4. Return to Figure (a) and repeat.

SUMMARY OF DATA MANAGEMENT SELF-TEST (Continued)

(c)



(d)



## CONCLUSIONS

### Proposed Method of Self-Test

1. Simplest solution to Phase B requirements.
2. Protects all data paths.
3. Adapts to crew participation in testing and to crew switching of computers and buses.
4. Well-defined hierarchy of control for operation and for self-test.
5. Does not change if subsystems are modified or added.

### Problem Areas

1. Voter is potential single-point failure.
2. Internally redundant design of SCU.
3. Form of power transients and dropout, and of noise bursts is unknown. Affects data bus.
4. Redundancy implementation and interlocking of panel switches.

### Recommendations

1. Early breadboard to identify hardware and software problems.
2. Early flight test to gain experience with real equipment.

**UNIFIED TEST EQUIPMENT  
A CONCEPT FOR  
THE SHUTTLE GROUND TEST SYSTEM**

**EDGAR A. DALKE**

**MANNED SPACECRAFT CENTER  
HOUSTON, TEXAS**

**N71-35066**

## SUMMARY

A unified test equipment (UTE) concept is described that has the characteristics required for a cost-effective ground support system through the reduction of hardware inventories and operational cost, and planned ground/onboard compatibility. Baseline test and support requirements and hardware flows have been developed that illustrate the feasibility of the UTE to meet overall Shuttle Program objectives. The design approach emphasizes modular building blocks for maximizing commonality to allow flexibility in efficient configuring of the hardware to satisfy early vendor/prime-contractor test requirements, as well as prelaunch and refurbishment support. The concept provides inherent compatibility to permit subsystem emulation as required to enhance the buildup and integration of vehicle subsystems, particularly during the R&D phases. Hardware elements that are described include the universal test console, the automatic RF test set, digital interface units for remote control and monitoring of GSE, and data bus interfaces, which are currently being integrated and evaluated in the Checkout Systems Development Laboratory.

## INTRODUCTION

330

The emphasis on low-cost ground operations for the Shuttle Program dictates an intensive review of ground support equipment (GSE) requirements and ground test and checkout operations. Low-cost techniques must be developed as well as new test and operational philosophies which are compatible with autonomous vehicle and a 2-week turnaround for support activities. A unified test equipment (UTE) concept is presented which is being developed at MSC in an attempt to satisfy the criteria described previously. In presenting this concept, the paper will (1) address some of the underlying cost issues of a program gathered from previous experience, (2) provide a definition of the concept being developed in terms of its general philosophy, design criteria and requirements, and applications, and (3) provide a review of in-house laboratory development activities supporting this concept.

In order to develop a cost-effective concept, a total review of program requirements to maintain required control to assure efficient expenditures is necessary. It should be realized that the start of a ground support system (GSS) commences with the first black box fabrication, is developed through the support of subsystem buildup and vehicle manufacturing, and is finalized in the operational launch support and recycle/refurbishment activities. It is interesting to note that almost without exception GSE and checkout/test are the first things needed for program support, yet receive minimal consideration until a subsystem or vehicle is through design and in the fabrication process. Potential schedule impacts therefore dictate fast design and work-around which account for one of the factors causing unnecessary expenditures. A concentrated effort is being made on this program to identify during Phase B the GSE requirements and test philosophies and to contribute to the overall vehicle/ground test system design.

## GROUND TEST EVALUATION - UTE GOALS

Within three generations of spacecraft, there has evolved a comprehensive ground test system encompassing test philosophy, state-of-the-art technology in test equipment, and a detailed familiarity of test operations from factory through launch. The success achieved by these ground systems to date is self-evident.

A brief review of some of the more salient features of previous ground test systems is in order to better understand the impact of program requirements on the design of ground equipment. The Mercury Program system is, of course, a baseline. The inexperience and uncertainties associated with manned spacecraft systems, coupled with the necessary emphasis on safety, prompted a test philosophy which involved systems performance verification and monitoring up to the time of launch. Intensive end-to-end subsystem tests and mission operation simulations provided the necessary confidence prior to launch. This approach to testing is in practice today.

The Gemini Program had considerable emphasis on reduction of launch pad testing which assisted in the successful Gemini 6/7 mission. Reduced pad time can be partially credited to the resolution of a recognized Mercury constraint involving system access for fault isolation. The Gemini subsystems design provided ground test access points which permitted troubleshooting with minimal invalidation of operational system interconnects. The use of factory-type test equipment at the launch pad was instituted to permit continuity of testing.

The Apollo spacecraft presented several additional challenges in the form of increased hardware complexities and the emphasis on system readiness for a relatively restricted launch window for lunar landing. Digital computers and flexible display devices enabled test personnel to have necessary insight into volumes of systems data in a meaningful manner, allowing continuous real-time systems assessment and decisions encompassing the total CSM/LM vehicle status.

Basic program goals of the Shuttle Program dictate that we again assess our previous experience to identify any new requirements for the ground test system. Several key differences are:

1. Spacecraft Usage - One time for the Apollo Program versus multiple missions for the Shuttle Program.
2. The Apollo Program is primarily R&D while the Shuttle Program is operationally oriented with minimal flight test.
3. The Apollo was single mission oriented of specified duration, while the Shuttle is multi-mission and of effectively open-ended duration.

Many similarities between the Shuttle Program and commercial airline practices have been noted. The ground test and operations philosophy is no exception. Unified test equipment is a concept which has this airline-type practice as its primary goal. More specifically, the concept goals are:

1. To minimize on-line ground support costs (equipment and people) through onboard checkout realization
2. To reduce off-line costs by minimizing equipment inventory for all program phases
3. To increase the efficiency of ground support operations to permit realization of a 2-week turnaround

# **A SHUTTLE GROUND TEST SYSTEM CONCEPT**

- CHECKOUT EXPERIENCE
- ADVANCED CHECKOUT CONCEPT
- CHECKOUT SYSTEMS LABORATORY DEVELOPMENTS
- CONCLUDING REMARKS

## **GROUND TEST EVOLUTION**

- MERCURY
- GEMINI
- APOLLO

## **UNIFIED TEST EQUIPMENT GOALS**

- ONBOARD CHECKOUT REALIZATION
- REDUCE COST
- TWO WEEK TURNAROUND



## SUPPORT EQUIPMENT - SCOPE

In approaching the development of the UTE concept, our first effort was to identify the ground test operation and its inter-relationships. Using our Apollo experience and available data, a total tabulation of identifiable CSM/LM GSE along with its usage and purpose was accomplished. Several interesting points were noted: (1) A significant amount of GSE is required for behind the scene operation such as vendor end item acceptance, verification before vehicle installation, and the act of vehicle manufacturing itself. (2) Although the functions were basically the same, the equipment was generally defined to support a particular need in the equipment flow, resulting in unique design and accompanying R&D costs. (3) A rather sizable portion of a program cost is GSE related. This information leads us to believe there is need to relegate the role of defining program requirements for GSE to a higher level than the design engineer.

The viewgraph points out the expanded scope envisioned for the support equipment role. In addition to the factory/launch activities, the need to support long-term operations involving diverse payloads as well as the vehicle precludes the use of traditional GSE approaches. This expanded scope implies the requirement for support equipment which assures continuous availability of program resources to maintain the flow of support services. This, together with traditional GSE concepts, has the potential of expanding support equipment inventories. Approaches and techniques must be developed to meet the expanded scope with reduced inventories.

# **SUPPORT EQUIPMENT - SCOPE**

## **● PREVIOUS PROGRAM**

- FACTORY/LAUNCH**
- BASICALLY R&D**
- LARGE INVENTORIES**

## **● SHUTTLE**

- SUPPLIER/FACTORY/  
LAUNCH/REFURB**
- R&D - EXTENDED  
OPERATION**
- ACHIEVE MIN INVENTORY**

## VEHICLE TEST FLOW/PREPARATION

Recognizing that time on the ground reflects money, plus the need for a ground support system design which supports a program objective of fast turnaround, a review of Apollo spacecraft experience was made to determine any impact on test hardware design. Of particular concern was the relative time required for the various activities as shown by the graph. Test setup and preparation are obvious candidates to consider for reducing ground involvement.

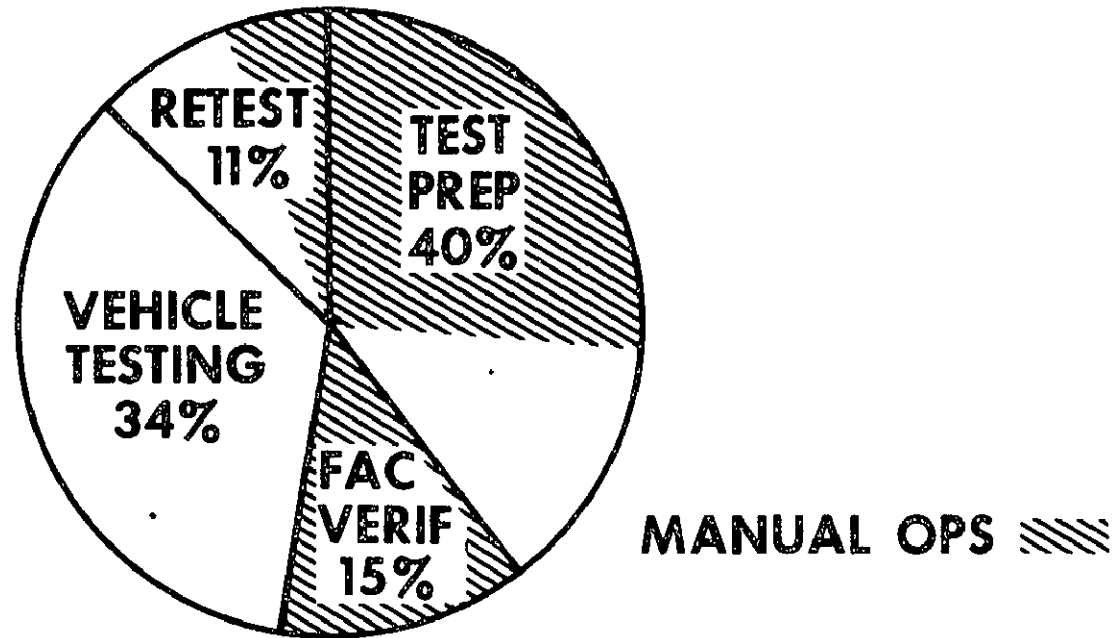
Test setup and preparation involved revalidation of all sites, associated Ground Support Equipment, and software programs that will be utilized to check out and launch the spacecraft. Revalidation consists of verifying by closed-loop testing and open-loop testing that (1) all ground equipment is in the prescribed configuration, (2) all electrical/mechanical connections are satisfactory, (3) software is compatible with spacecraft testing requirements, and (4) the total ground system is within specifications. In addition to utilizing the on-line ground test equipment, the following equipment is required to perform the validation testing: breakout boxes, special test equipment (STE), and bench maintenance equipment (BME).

The complex ground equipment interfaces encompass GSE to vehicle, GSE to facility, and GSE to GSE. The unique requirements of each test location have dictated the requirement for unique configurations of GSE, both hardware and software. More specifically, the time involved can be correlated with the number of interfaces and the method of verification. Multi-conductor cables carrying analog data accomplish most of the interfacing, while manual operation consisting of switch setting, meter readings, and pin-to-pin continuity testing accounts for operator involvement.

Considerable amounts of time are involved in test preparation and validation. Apollo experience has revealed that manually operated sets, such as RF GSE, account heavily in test preparation. Characteristics of these operations stimuli include signal (RF) generation and calibration, verification of RF cable patches, and a general assessment of measurement and signal source capabilities requiring lab standard calibration.

One of the factors which contributes to the retest time is the result of manually induced human errors. These can generally be categorized by manual operations consisting of (1) data logging and status monitoring, (2) verbal communication between test operator and test conductor, and (3) maintenance of proper order of test sequences.

# VEHICLE TEST FLOW/PREPARATION



**SITE TIME UTILIZATION**

## UTE APPLICATIONS

The UTE concept provides for the availability and application of versatile ground support hardware dedicated to achieving maximum onboard checkout and operation while providing mandatory ground support with minimum hardware inventory. Major functional applications include detailed test and checkout at the vendor/factory level, development testing, and vehicle operations support.

### Vendor/Factory Tests

During the vendor/factory test cycle where major portions of the avionics systems are not available, a test support capability is required to provide intersystem interface simulation. The UTE modular design feature provides the capability for GSE architecture configured to vendor and factory level testing for hardware validation and acceptance testing, as well as initial operating system software development. The modules utilized are a part of the common-use GSE inventory intended to be moved and integrated into GSE support required for integrated testing.


### Development Testing

During the development test cycle including ground and flight vehicle operations, the UTE is utilized for vehicle system functional/electrical interface validation test and evaluations. The experience gained at the vendor/factory level enhances utilization for bench test support at test locations. The interface compatibility of the UTE to vehicle data bus operations and the display and control console capabilities are particularly applicable to the development and validation of onboard checkout techniques through computer-to-computer communications, and onboard checkout simulation and/or emulation for ground/onboard functional allocation trade-offs as well as comparison of results.

### Operations

The role of UTE will be primarily that of refurbishment to recertify vehicle hardware elements and for payload integration. Since the goal of maximum vehicle autonomy is assumed, only those ground pre-launch support functions that are mandatory, cost effective, and/or critical to the 2-week' turn-around and launch window operations will be planned.

# UNIFIED TEST EQUIPMENT APPLICATION

	VENDOR/FACTORY	DEVELOPMENT TEST	OPERATIONS
TEST ARTICLE	SUBSYSTEM BUILDUP PIT TESTS	HORIZONTAL FLIGHTS VERTICAL FLIGHTS	PAYLOAD MISSIONS ONBOARD AUTONOMY
GSE	HARDWARE VALIDATION  ACCEPTANCE TESTING	FUNCTIONAL/ ELECTRICAL I/F VERIFICATION  ONBOARD CHECKOUT VALIDATION	MINIMAL LAUNCH SUPPORT  PAYLOAD INTEGRATION  REFURBISH- MENT
<div style="border: 1px solid black; padding: 5px; display: inline-block; margin-top: 10px;"> <b>REUSE OF COMMON GSE (UTE)</b> </div> 			

## UTE COMMON TEST EQUIPMENT

A major cost savings factor is commonality of test equipment for different levels of test and at different test sites. Maximum use of identically designed test equipment reduces the total inventory and prevents redundant design efforts. In past programs, GSE design was the responsibility of the subsystem engineer, resulting in the development of special test equipment, designed for specific functions and applications. Lacking complete program requirements visibility, several redundant designs, each carrying a high engineering overhead for engineering and development, were developed.

The accompanying figures show the standard or common elements of test equipment utilized in different operations activities. These standard items are typical of those required at all test sites from vendor through launch. They consist of:

Test Console(s) - control and monitor of test article and test equipment

RF Test Set - generates RF signals, evaluates RF responses

Test Article I/F - interconnect test article and test equipment

Standard Support Equipment - AC and DC power, electronic equipment cooling, etc.

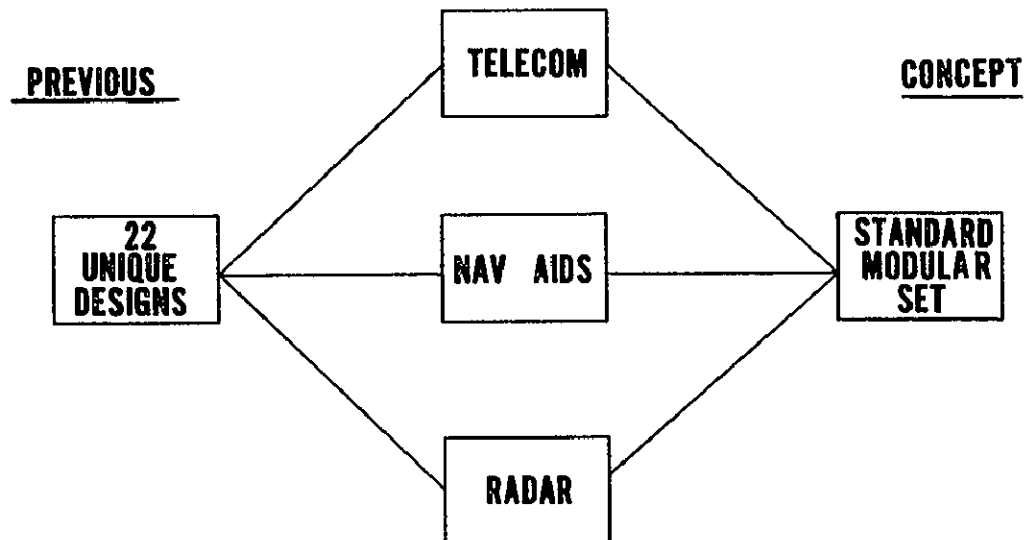
The special test equipment modules consisting of fixtures, simulators, and standards are unique to the test articles assigned to a particular shop. However, these modules are potential candidates for common use in the component, assembly, subsystem, and vehicle level testing.

The acceptance checkout equipment (ACE) used for Apollo spacecraft checkout is an example of past approaches to GSE commonality. These checkout systems were utilized for vehicle level testing at the prime contractor's facilities, thermal vacuum testing at MSC, and launch site operations at KSC.

## EQUIPMENT COMMONALITY

	VENDOR/ FACTORY	FLIGHT TEST		OPERATIONAL		REFURB— ISHMENT
		VEHICLE	SUPPORT	VEHICLE	SUPPORT	
TEST CONSOLE	X	X	X		X	X
TEST ARTICLE I/F	X	X	X		X	X
SUPPORT EQUIP.	X	X	X	X	X	X
TEST FIXTURE	X					X
TEST SIMULATOR	X	X				X
STANDARDS	X	X				X

## STANDARD DESIGN





## UTE - REDUCE SUPPORT PERSONNEL

Another program element which heavily affects ground operation costs involves the number of personnel required to perform on-line testing. Techniques must be developed that provide tools and visibility for fewer people to do the required test, checkout, and operation support functions. Prime areas for consideration in the ground environment include automation and simplification of GSE interfaces.

### Automation

The utilization of increased automation must be planned at the startup of the program so that the detailed procedures can be developed and verified early in the R&D phases. A key to the feasibility of implementation is the capture of the test requirements at the process specification stage, committing the requirements to computer media, and producing test procedures and related supporting documentation automatically. This early capture of requirements provides the means to translate procedures into formats compatible for computer-controlled callup and display techniques using mass memory, microfilm, or other state-of-the-art methods. On-line review and modifications can be achieved, with a major gain being the automated data logging of test results relieving the test operators of the vast majority of the mountain of paper currently needed to run a test. Additional gains are achieved by the fact that the personnel skills can be better utilized in that the routine and repetitive work is done with machines, thus giving people time to concentrate on those activities requiring decisions and judgment.

The techniques can be extended to the overall GSE problems by providing automated control and monitoring of those functions, as an integral part of the test procedures. Minimizing the manual GSE functions through the use of standard interface units also reduces the number of operational personnel required.

### Interface Requirements

Prime factors of the UTE concept are those of modularity and commonality. By employing the concept as far down in the test flow as possible, the need for specialized GSE becomes minimized. The less specialized GSE required, the fewer specialized personnel are required.

In extrapolating out into the operational phase of the program, the goal of vehicle autonomy is of prime importance. The degree of success in simplifying and reducing the number of the GSE interfaces will directly influence the degree to which the onboard data management system can be utilized to achieve maximum vehicle autonomy during ground operations.

# REDUCE SUPPORT PERSONNEL

<u>FUNCTION</u>	<u>PRESENT</u>	<u>CONCEPT</u>
<b>OPERATIONS</b>		
<b>COMMAND GENERATION</b>	<b>MANUAL INITIATION</b>	<b>PROGRAM CONTROL</b>
<b>RESPONSE EVALUATION</b>	<b>VISUAL</b>	<b>PROGRAMMED DECISION</b>
<b>GSE MONITOR AND CONTROL</b>	<b>MANUAL</b>	<b>AUTOMATED</b>
<b>DOCUMENTATION</b>		
<b>DATA LOGGING</b>	<b>MANUAL</b>	<b>MACHINE</b>
<b>TEST PROCEDURES</b>	<b>COOK BOOK VERBAL</b>	<b>CRT DISPLAYS PROGRAMMED OPTIONS</b>
<b>DISPLAYS</b>		
<b>TEST INFORMATION</b>	<b>ENG'G UNITS RAW DATA</b>	<b>RESULT SUMMARIES PROCEDURE MILE- STONES</b>

UTE - SIMPLIFIED INTERFACES - REDUCED VERIFICATION

At the launch site, in a fully operational phase, the UTE concept presents a much reduced time and operational costs expenditure to verify facilities/GSE as being ready to accept a next vehicle and the conductance of tests (includes servicing) against the vehicle once in that facility.

In the fully operational mode, with a 75-launch-per-year schedule, our projected baseline analysis reveals the following synthesized time line allocations.

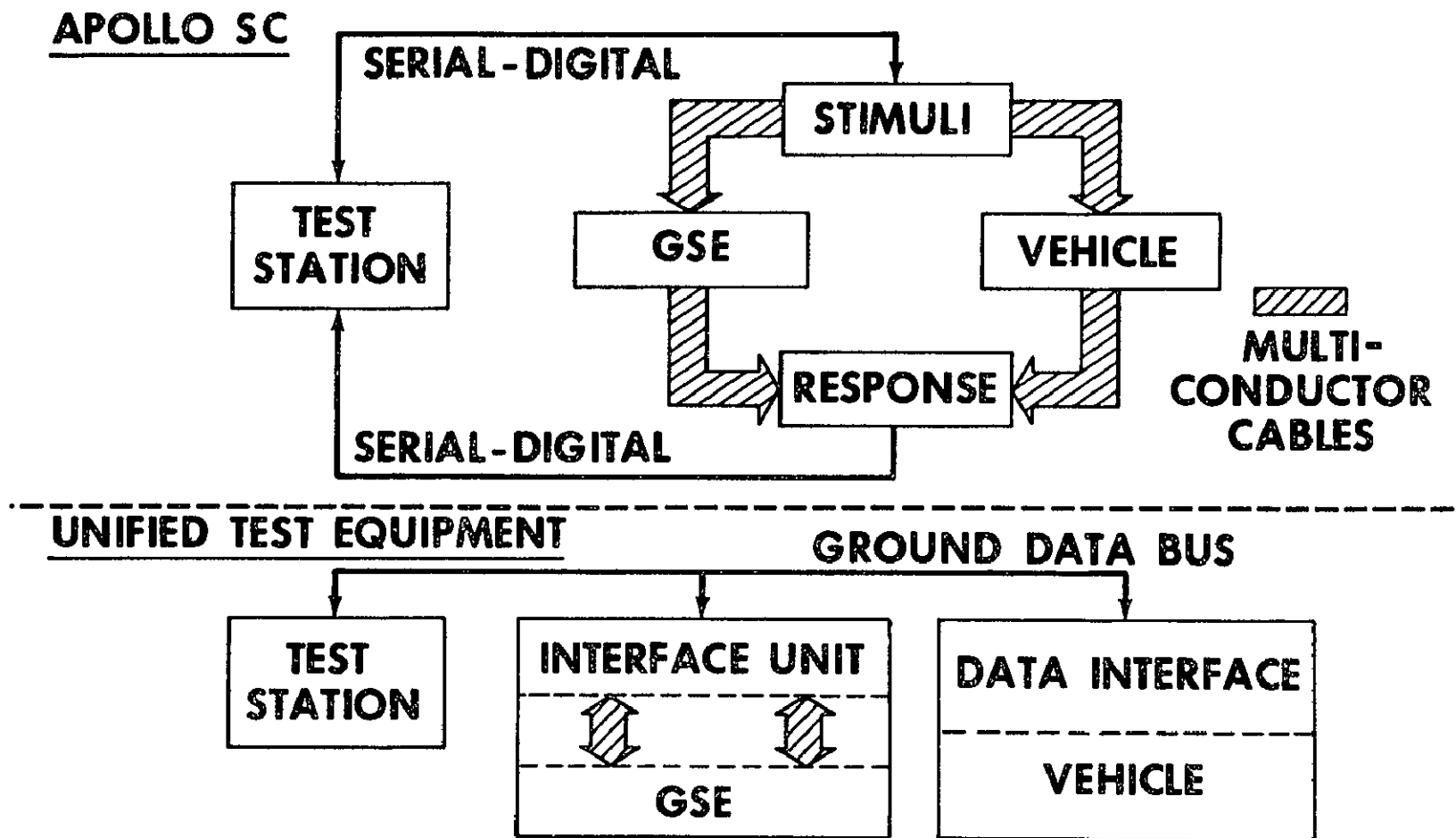
Area	Time Between Vehicles	Time to Prep & Verify	Available Free Time
Safing Area	Max = 108 Hr	10-1/2 Hr	97-1/2 Hr
	Min = 6 Hr	10-1/2 Hr	- 4-1/2 Hr
M&R Line	Max = 170 Hr	31 Hr	139 Hr
	Min = 108 Hr	31 Hr	77 Hr
Launch Pad	Max = 192 Hr	44 Hr	148 Hr
	Min = 120 Hr	44 Hr	76 Hr

The area preps and test setups include refurbishment, inspections, scheduled maintenance and servicing, physical interface checks (such as templates or umbilical end "go, no-go" tool mates), closed-loop connections for GSE self-tests, and certain external leak checks followed by restoration to vehicle acceptance configuration. The area GSE verifications are closed-loop full-range functional tests conducted in the automated mode (except certain leak checks). The vehicle placement and test/servicing encompasses the actual functions performed on the vehicles. This includes vehicle erections, cargo placement, vehicle mating, servicing, and test/checkout as demanded for flight.

Techniques which will allow a reduction in the test preparation and verification include:

- Integration of GSE and the data acquisition/control element (standard interface unit (SIU)). This reduces interfaces and cables which are widely recognized as problem generators.
- Dedicated usage of GSE. This accomplishes a reduction of reactivation and allows the level of validation to be one of automatic closed-loop self-test.
- Data bus interface to vehicle. Maximum use of the onboard data management and control system interfaces should eliminate the need for carryon cables to interface control/monitor signals to the test console.

# CONCEPT-SIMPLIFIED INTERFACE



## SPACE VEHICLE CHECKOUT

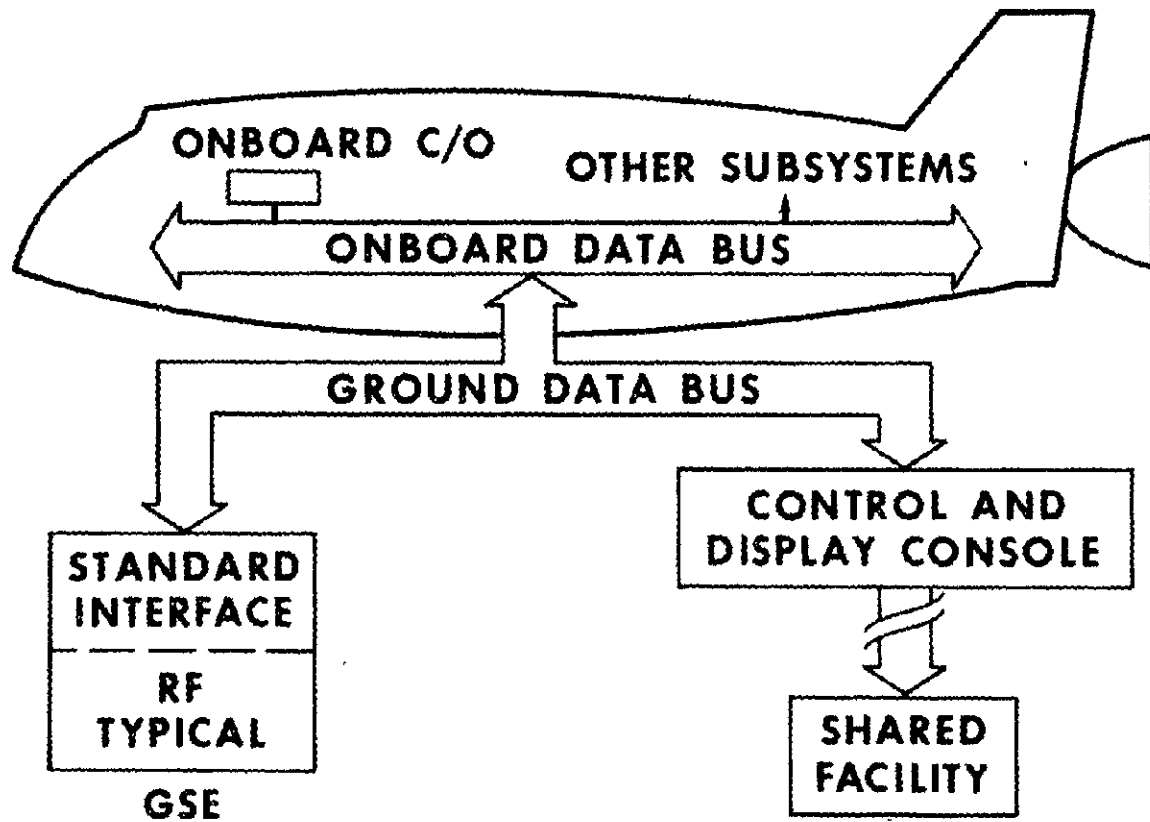
The UTE is a system concept which defines a total GSE overview encompassing hardware design, program planning, and test philosophy. The design requirements identified have dictated several basic technology developments in order for UTE to be realized. Developments which provide a baseline for test and evaluation of the concept and are presently in progress at MSC include:

- Test operator and control console
- Standard interface units (SIU's) and data bus
- An adaptive automatic RF test set

Autonomous vehicle operations (and its associated reduction in GSE) and rapid turnaround are the basic forcing functions behind the implementation of this concept. The inherent GSE design approach is one of providing GSE which is (1) readily configurable with minimum hardware to satisfy vehicle support requirements during all phases of test activity, (2) automated and integrated for repetitive functional activities based on cost effectiveness, and (3) interfaced for functional capability compatible with onboard system concepts.

An integrated approach to vehicle testing recognizes various levels and purposes for testing, all of which require some form of GSE. Testing encompasses system performance assurance during manufacture of the systems, interface verification during vehicle integration, and servicing support during launch operations. A key to the achievement of simplified test operations and test hardware configurations will be standard levels of interface for the control/monitor and transmission of vehicle systems and GSE data. This will be accomplished through the use of SIU's capable of performing control and monitor functions. The integration and control of ground test and checkout operations, including GSE control, are prime functions of the control and display console.

# SPACE VEHICLE CHECKOUT CONCEPT



347

### Control and Display Console

The control and display console provides for the remote display of real-time and support test information utilizing color, graphics, status, and alphanumeric combinations that allow operator visibility consistent with the test activity. Major design features include modular packaging for flexibility in sizing equipment to meet test needs, simplified man/machine interface considerations to minimize operator actions while maximizing information transfer, and added computational and storage capability via modem to large central computer facilities for common data base development and operations.

### Modular Packaging

The packaging concept provides for common-use stand-alone modules that can be configured for minimum test loads and expanded by add-on progressively to meet maximum test loads. Add-on modules include the CRT display, additional computational capability, additional memory, acquisition models, storage and recording, and selected peripherals. The processed display information can be remoted to various users for presentation on standard 525-line TV receivers as a result of the video data handling techniques utilized. Complete test history files for test support as well as documentation operation will be achieved via the storage and recording modules.

The acquisition module will be designed modularly to provide flexibility in the data input and command output format conversion and handling. The design approach includes provision for multiple data bus communication capability and for separate PCM and command transmission features for possible R&D Shuttle phases.

### Man/Machine Interface

The man/machine interface is accomplished via the keyboard and special function keys, fixed and variable. The basic keyboard provides the standard alphanumeric, numeric, and special graphics. The fixed function keys provide for repetitive action calls. The variable display function keys are under software control and are configured based on selected activity. Versatility is achieved through selection by the test operator and/or on-line composition of information-oriented displays.

### Central Computer Facility Interface

The central computer facility interface is via standard modems, with one modem as an integral part of the control and display console. In addition to common data base operations, the central computer will provide computational and procedural support as needed.





## Standard Interface Units (SIU's)

The standard interface units provide for a manageable interface point for the automated control and monitoring of GSE elements. Through the use of standard digital interfaces, certain inherent program problems are reduced. A common working interface is established when the unit is involved at different locations and multiple users are supported. Design requirements, responsibilities, and subsystem data interchange identification becomes more manageable. Major design features of the SIU include (1) modular packaging for sizing based on functional needs; (2) programmed capability for controlled decentralized operations including closed-loop control, data compression, self test, and critical safing sequences; and (3) data bus interface functions.

Modular Packaging - The design approach requires modularity for basic hardware elements including memory expandable to 64K, and input/output capability to achieve complete flexibility in sizing.

Programmable - The SIU has a basic repertoire of 32 instructions. Since the instructions are implemented by using plug-in ROM techniques, the instruction set can be changed to meet special applications. However, the design permits manual override and local display capability if desired.

Data Bus Interface - The data bus interface function includes command decoding and encoding, error checks, and serial-to-parallel and parallel-to-serial conversion.

## Data Bus

The data bus concept provides a versatile communications technique compatible with the shuttle onboard communications scheme. The use of this interface allows a logical vehicle integration and development. Through this unique communications interface, a standard ground facility can emulate/simulate the onboard system functions (including data management and control) during subsystem buildup and vehicle manufacture, and permits a progressive development whereby the onboard system assumes more responsibility to accomplish vehicle autonomy.

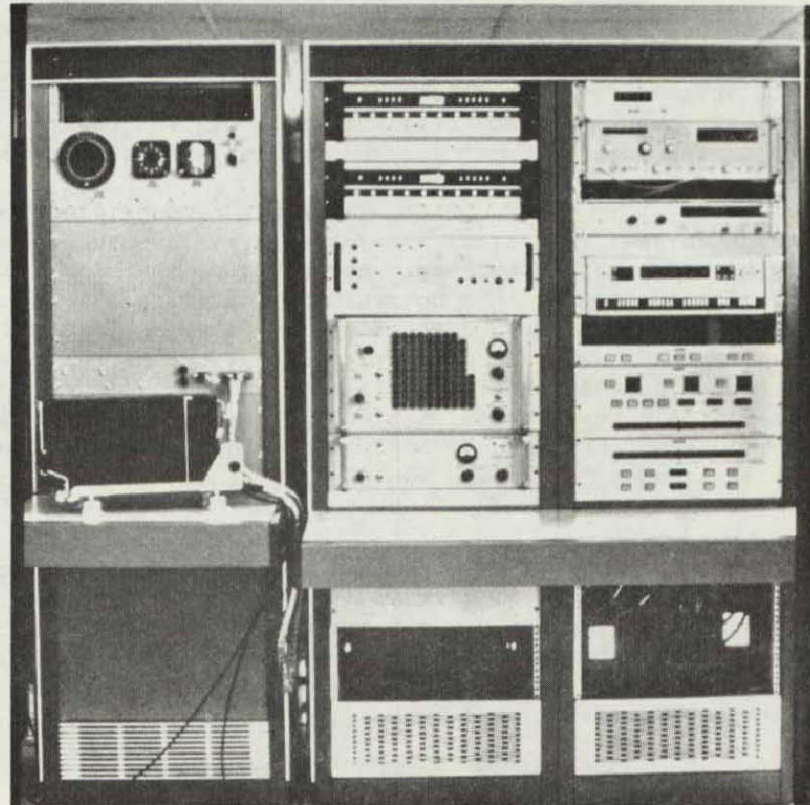
## AUTOMATED RF TEST SET

The RF test set development for UTE implements the concept of a common, modular design capable of multiple system support. The test set organization and design is oriented around common functional categories of RF system support requirements. This is identified by (1) frequency generation sources, (2) a measurement group, (3) a modulation group, and (4) an attenuation and switching group. Maximum utilization of available commercial test sets which functionally satisfy the previously discussed categories has been realized. Conventional RF test operations are performed, e.g., power and sensitivity, VSWR, bandwidth, and signal-to-noise-ratio tests.

Automation of the test set is accomplished through the use of an in-house designed and fabricated standard interface unit (SIU) capable of programable closed-loop control and monitor functions. Initialization and calibration is an adaptive process using iterative routines which select a requested function or parameter, measure the result, and provide any required delta to cause a comparison between request and result. Periodic status checks allow continuous calibration updates. System-under-test response parameters are measured by the test set and evaluated by the SIU for correctness of data. The data evaluation is processed by the SIU to reflect test status information (GO, NO GO, FAILED, etc.) and made available to the test console for appropriate test coordination. Because of its decentralized functional capabilities, the RF test set is capable of automatically performing individual testing independent of the test console if system organization and operations so dictate.

The initial development encompasses VOR/ILS and VHF/UHF communication systems checkout. Test system accuracies are equal to or exceed the FAA specifications for equivalent test gear. Based upon this baseline development, a modular expansion is in process, which will provide pulse modulation and radar frequency spectrum capabilities with little increase in system size. With this addition, it is expected that the total spectrum of RF systems test and checkout can be accomplished through the use of one basic configurable test set.

AUTOMATED RF TEST SET



# CONCEPT DEVELOPMENT AND EVALUATION

## LABORATORY TOOLS

STAND ALONE CONSOLE

INFORMATION DISPLAYS

RF TEST SET

SOFTWARE SYSTEM

## EVALUATION

ARCHITECTURE-MODULAR  
DESIGN VENDOR  
THRU REFURB

RAPID COMPREHENSION  
AUTOMATED TEST  
PROCEDURES

AUTOMATED GSE DATA BUS  
INTERFACE-REMOTE  
CONTROL

CONVERSATIONAL/TUTORIAL  
MAN/MACHINE INTERFACE

## CONCLUDING REMARKS

- SHUTTLE EMPHASIS ON COMMERCIAL AIRLINE TYPE OPERATIONS DICTATES THAT THE GROUND TEST/SUPPORT SYSTEMS PROVIDE FOR
  - EXPEDIENT RE-CYCLE OPERATIONS
  - EFFICIENT MANPOWER UTILIZATION
  - MINIMAL GROUND EQUIPMENT INVENTORY
  - RE-USE CAPABILITY FROM R&D THROUGH OPERATIONS
  - MAXIMUM UTILIZATION OF ONBOARD SYSTEMS
  
- KEY TECHNOLOGY FUNCTIONS IDENTIFIED WITHIN THE UTE APPROACH ARE BEING DEVELOPED FOR EVALUATION AND FEASIBILITY ASSESSMENT

## A STANDARD LANGUAGE FOR TEST AND GROUND OPERATIONS

Henry C. Paul

NASA-John F. Kennedy Space Center  
Cape Kennedy, Florida

The Space Shuttle characteristics of maximum onboard autonomy, two weeks turnaround capability, and a computer controlled data management system demand effective computer hardware and software for onboard operations as well as for launch checkout and preparations.

It is proposed that a standard language for test and ground operations be specified by KSC and that this language be used for test procedure definition for all levels of space shuttle flight and ground support hardware from line replaceable units (LRU's) up. This language should be defined and documented adequately to allow its use at the outset of the Phase C/D Space Shuttle contract.

The language specification shall contain syntax definitions with structure to facilitate computer implementation of all valid language statements. The choice of language primitive and sentence order shall be test engineer oriented, unambiguous, and as simple as possible.

The initiative to produce a standard test language for the Space Shuttle emerged from experience gained in test automation during the Saturn/Apollo program. A test language (Acceptance Test or Launch Language - ATOLL) was introduced to the launch site by the Marshall Space Flight Center (MSFC) during the development phase of the ground support system for the Uprated Saturn I program. Automation, using this language, has grown to a place of prominence in the testing and launching of Apollo space vehicles even though the language is not English-like in structure and requires some special training.

During early Saturn automatic checkout operations the typical user-programmer communication gap was experienced firsthand as test personnel attempted to comprehend the programmer's interpretation of their test requirements. Generally, changes were difficult to implement and to understand. Full control over the test came only after much actual experience. Even then the details of some operations were obscurely embedded in the test programs so that test engineers had difficulties in comprehending the subtleties of the programmer's logic. At the time when automatic checkout could have eased the mounting strain associated with the pressing schedule, the lack of an adequate common language to communicate requirements and to describe the computer programs further burdened the launch team. The high order language, ATOLL, provided the communications necessary to improve this situation significantly; however, ATOLL lacks some needed capabilities and is not readily readable by the engineer untrained in the language.

For Saturn/Apollo we must place ourselves in the same category for language development as most of the other groups who develop and use procedure oriented languages within industry or Government. That is, the language was developed only after the equipment and applications were firmly established.

We are now presented with the rare opportunity of standardizing the basic communications among all facets of ground and in-flight testing at a point in the system acquisition cycle where it will become a natural part of the program. This will avoid costly retrofits and difficult "unlearning" exercises at a later time. The language will serve as the basic tool in insuring commonality for Orbiter/Booster/GSE Test and Ground Operations Procedures while inherently providing the capability to: (1) efficiently automate manual procedures, (2) readily adapt design procedures for operational use, (3) reduce supporting documentation, (4) efficiently cross-train test personnel, (5) minimize impacts from changes, and (6) in general, will be a prime contributor in support of the rapid turnaround requirements.

Cost effectiveness considerations which justify a standard test language include the following:

- a. Economy realized in programming manhours for the high order language over the machine symbolic language. The ratio in favor of the high order language is approximately ten to one for an equivalent job.
- b. Program turnaround time significantly reduced for programs written in the language.
- c. Documentation costs significantly reduced because the language provides its own documentation.
- d. The language allows programs to be written and maintained by the responsible system engineers, thereby reducing the need for a programming group dedicated for this purpose.
- e. Computer programs more readily and efficiently controlled and maintained at the high order language level.

The primary language objectives which are general statements of philosophy to be used in implementing the language are tabulated below.

- a. The language requirements and specifications must be consistent with and support the design concepts and requirements of the Space Shuttle.
- b. The language will be independent of test equipment.
- c. The language will allow the same procedure to be used for both manual and automatic testing.
- d. The language shall provide for a flexible monitoring capability.

- e. The language will provide the capability for test personnel to communicate with mission software.
- f. The language will be easy to use by test-oriented personnel not necessarily skilled in programming techniques.
- g. The language must be compatible with the philosophy of performing concurrent testing.

The complete language requirements are too voluminous for inclusion here; however, the major breakdown of subject matter is as tabulated below. The complete language requirements are contained in a Kennedy Space Center Technical Report, KSC-TR-1111.

- a. English-like words, structure, and punctuation.
- b. Comments.
- c. Control of the system under test.
- d. Data sampling from the system under test.
- e. Data comparison.
- f. Time controlled events.
- g. Performance monitoring.
- h. Information presentation and recording.
- i. Console interaction.
- j. Data manipulation.
- k. Computer-to-computer communications.
- l. Interface with other languages.
- m. Test sequence designation.
- n. Language redundancy.



- o. Identification of language packages and components.
- p. Data dictionary.
- q. Table definition.
- r. Writer aids.
- s. Reaction to system changes.
- t. Language character set.

For the Space Shuttle Program, we must make maximum use of the lessons learned through the years of design and launch experience. The very nature of the Space Shuttle design and the essence of the operational concept dictate that more be accomplished in a shorter period by fewer people than ever before. Automation, then, becomes a requirement for operations, not an elective. To effectively apply extensive automation, a test language has no suitable alternate.

SOLID STATE SEQUENCER SYSTEM

David E. Leck

Martin Marietta Corporation  
Denver, Colorado

The Solid State Sequencer System (S4) was designed under Contract NAS9-9334 to perform the functions of the present Apollo sequencers which are relay implemented. The S4 controls the EDS ABORT, SP3 ABORT, CM-SM SEP, LM-SLA SEP and ELS sequences. However, because it is a stored program type system, it can be easily adapted to other applications requiring the same high reliability such as sequencing crew safety functions aboard the Space Shuttle.

359

N71-35068

7  
⊕

800-82008

### I. System Description

The S4 consists of two identical channels which operate in parallel for reliability purposes. Most of the following description applies to an individual channel.

The S4 is controlled via input events which are used to initiate sequences or to control branching conditions within a sequence. A crystal controlled clock is used for timing. The program is written in a sequential manner with the time intervals stored being defined as the time between an event and an output or as the time between two outputs. Ground isolation of outputs permits power busses to be separated as required for a particular application. Sequence position is maintained despite power dropouts of any duration. The status of the outputs is stored and at the end of a power dropout each output is automatically restored to its condition prior to the dropout. Time counts in progress are recycled for power dropouts of 0.5 sec or longer. This function may be disabled so that time counts are continued from the point of interruption if desired for a particular application.

Types of output switches used in the S4 include pyro firing circuits, motor switch drivers, solenoid drivers, 0.5 amp switches, and general purpose 100 milliamp switches. The system can control up to 64 outputs. The types of output switches may be varied without affecting the rest of the system.

360

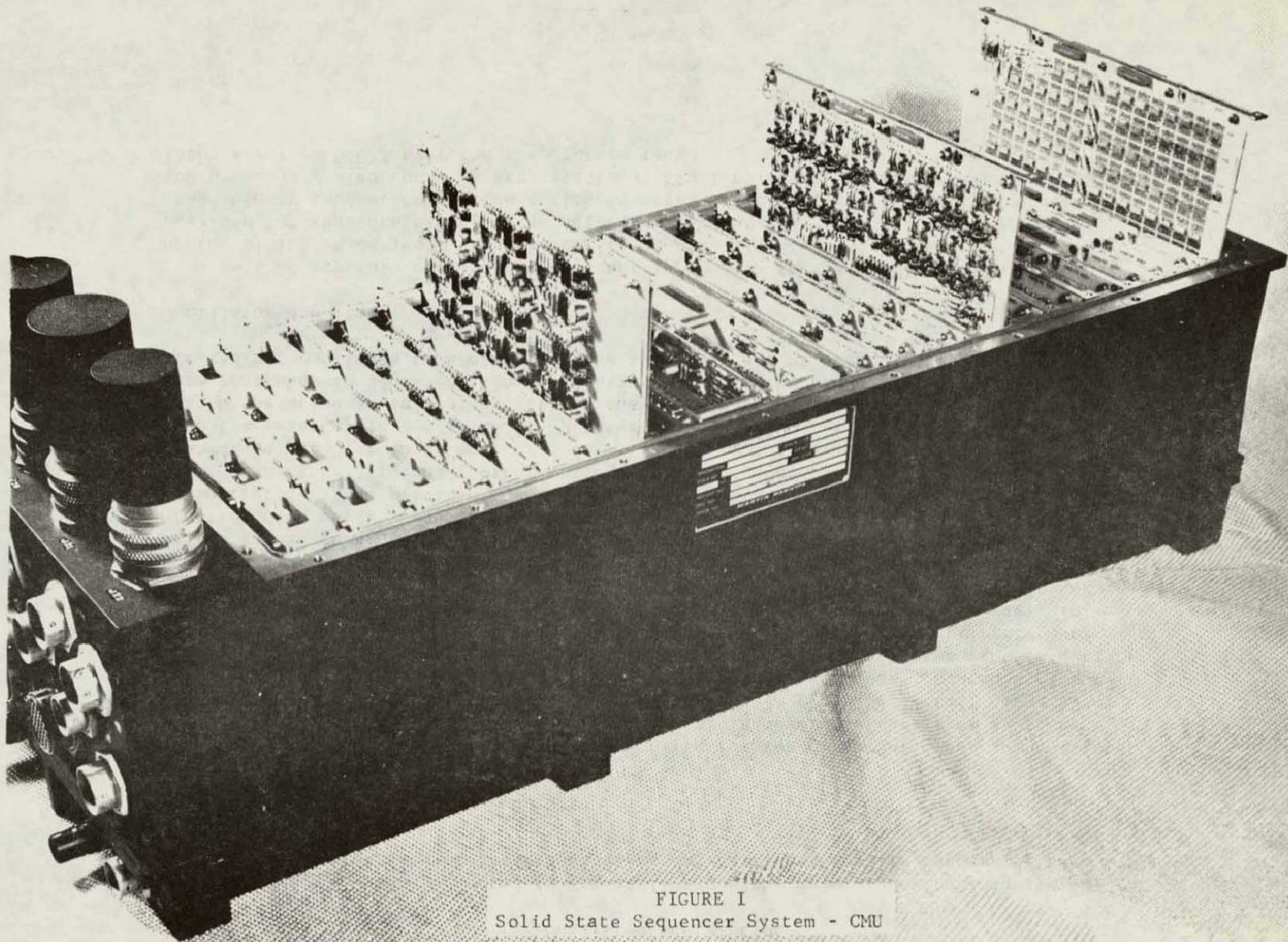


FIGURE I  
Solid State Sequencer System - CMU

## II. Subsystem Description

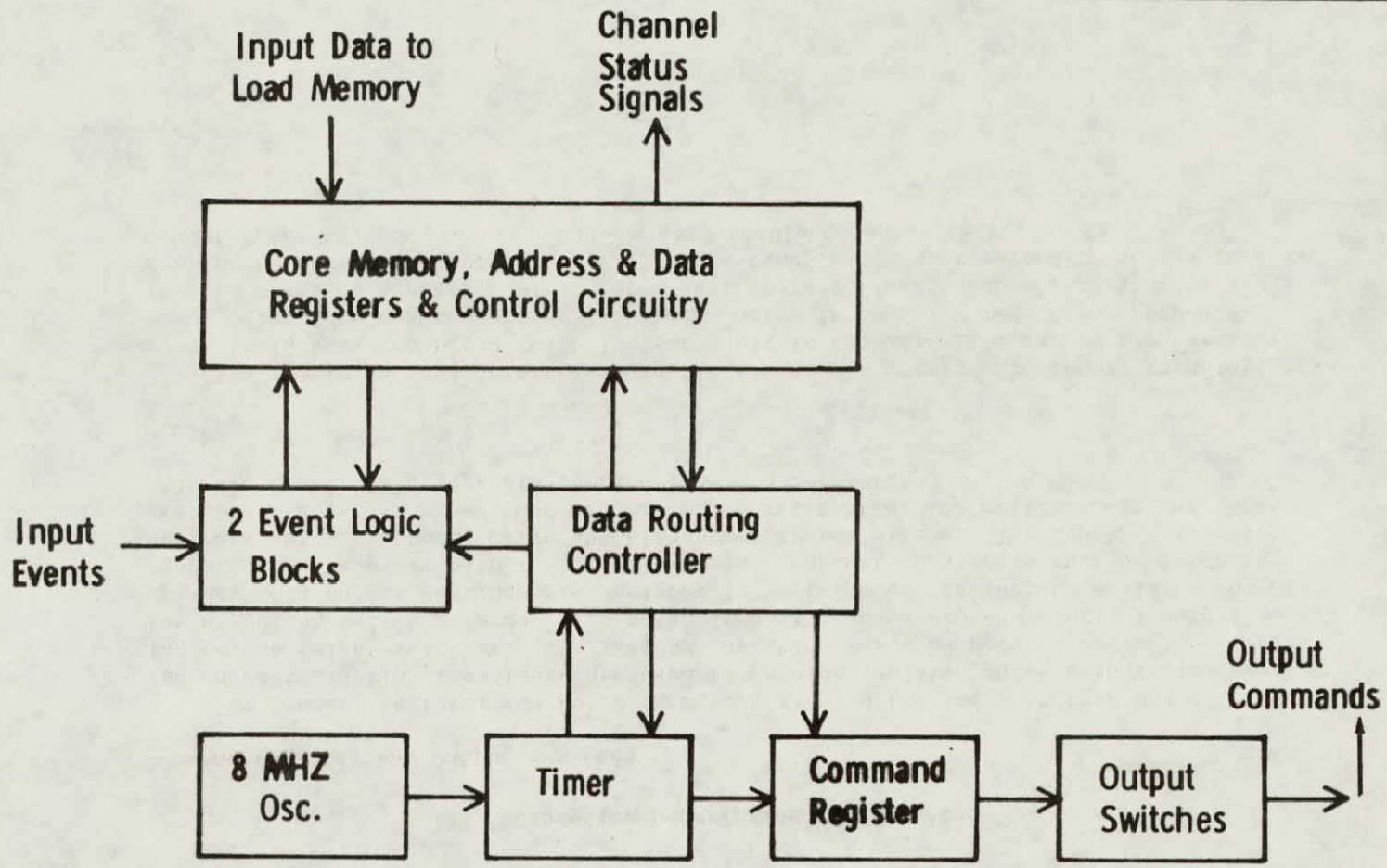
The block diagram presented in Figure II shows the major functional subsystems in the S4. The function of these subsystems and pertinent details concerning them are described below.

### Event Logic Blocks (ELB's)

Two ELB's are provided to permit monitoring for two different event combinations at one time. The inputs to these event blocks are generated either from switches controlled by the astronaut, or from external events (e.g. - the Liftoff signal), or from sequencer controlled outputs. Each input event signal passes through an input filter which provides a nominal one millisecond delay to prevent noise from triggering the system. The two ELB's operate independently with one ELB having priority over the other. Program construction allows for 17 event inputs per ELB. However, in the actual system only 13 and 8 inputs have been implemented for the two ELB's. Each ELB can be programmed to look for an OR combination of any number of its input events, or for an AND combination of two or three events, or for an N-1 or N-2 out of N combination of events. When an event equation has been satisfied, a branching operation is initiated. This provides the capability of branching into a particular sequence or of conditional branching within a sequence. The event logic may be used in an "immediate mode" or in a "monitor mode". In the immediate mode, an equation is loaded into the event logic and a YES or NO answer is obtained. If the answer is YES, a branch operation is initiated. If the answer is NO, the ELB is cleared and the program continues in sequence. In the monitor mode, if the answer is NO, the ELB is not cleared but continues to monitor for the specified set of event conditions.

### Oscillator and Timer

An 8MHZ crystal controlled oscillator is used as the basic clock from which all system timing is derived. Four clock frequencies (10KHZ, 1KHZ, 100HZ, and 10HZ) are provided for sequence timing. The selected frequency is counted down in a twelve stage binary counter to provide a time interval selection from 100 microseconds to 409.6 seconds. The count is accurate to within 1 pulse of the 10KHZ clock  $\pm$  an oscillator error of 0.01% over the temperature range.



CHANNEL BLOCK DIAGRAM

FIGURE II

363

### III. Subsystem Description Continuation

#### Command Register and Output Switches

The Command Register and Decoding Matrix are implemented with tape wound cores, and current steering techniques are used to provide isolated pulse outputs for turning on the output switches. The reasons for this are: low power consumption; isolation of output switch grounds; high noise immunity; and a stored record of output switch status. All output switches are designed for zero power dissipation while in the OFF condition. The pulse output from the magnetic decoder is used to turn on latching switches which, in turn, drive the high power output stages. Two types of latching circuits are used: those that automatically reset after 150 milliseconds and those that are reset via a separate command from the decoder.

#### Memory

The memory is used to store up to 512 eleven bit words. The memory capacity, however, is 1024 twelve bit words. The extra bit in the word is a parity bit used for error detection, and the extra 512 word capacity is used for redundancy purposes. This is a random access, 4 wire, coincident current, ferrite core memory using read/restore and clear/write cycles. Full cycle time is 1.8 microseconds. Access time is 0.4 microsecond. A picture of the memory module is shown in Figure III.

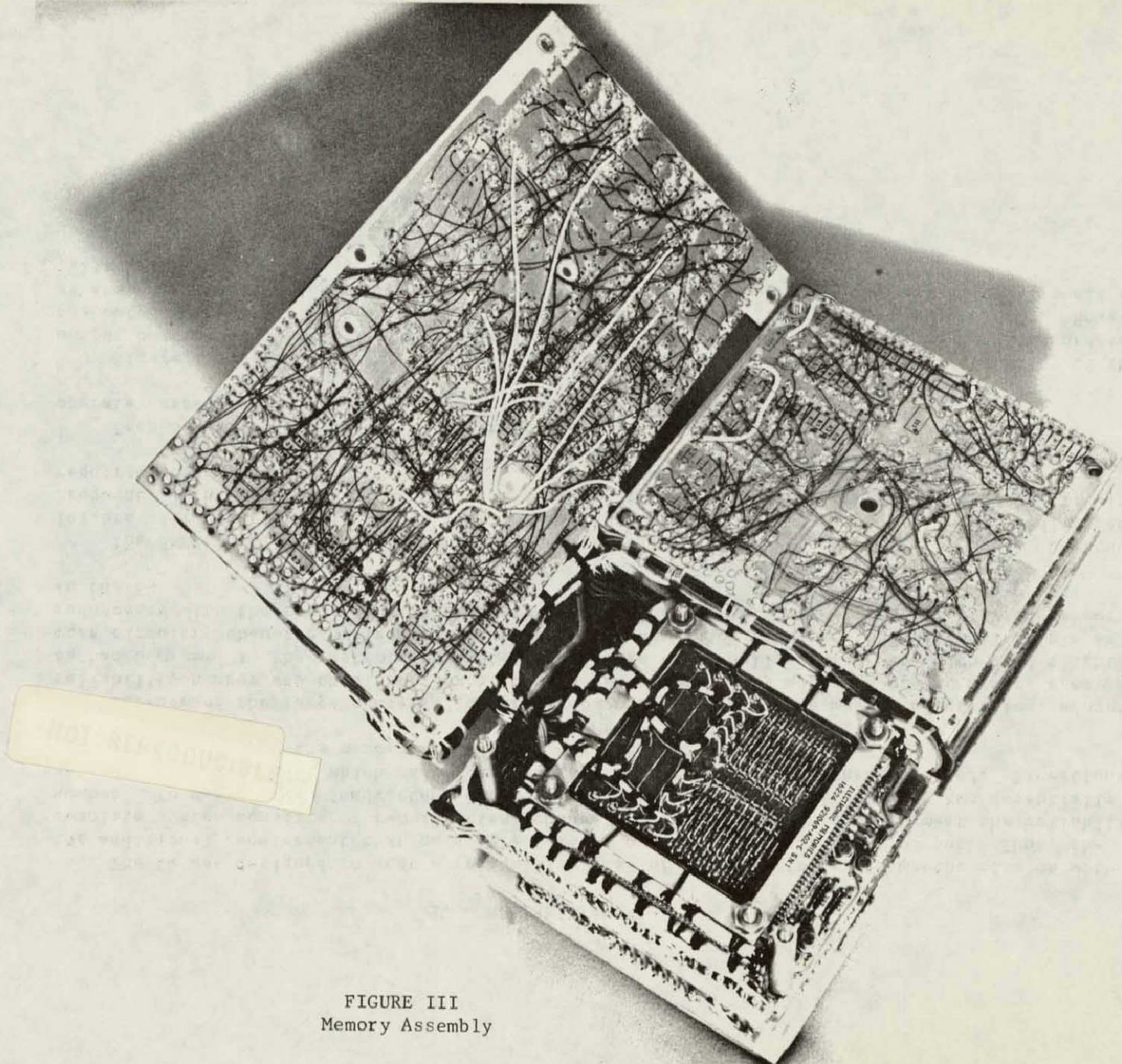


FIGURE III  
Memory Assembly



#### IV. Reliability

The S4 was designed to meet a reliability goal of 0.999999 for a six months mission with the additional requirement that no single failure produce an inadvertent output. Thus, the complete system consists of two identical channels operating in parallel to meet the reliability number. To meet the no inadvertent output requirement, each channel contains two essentially separate paths, both of which must operate properly for an output to be generated. Exceptions occur in two areas: the memory electronics and the 8MHZ oscillator.

Because of the large number of discrete components used in the memory electronics, a poor reliability number was obtained when using the same reliability approach to the memory area as had been used in the rest of the system. A design was finally worked out which used slightly more circuitry than the series redundant approach, but which resulted in functional triple redundancy with the result that the memory electronics is one of the more reliable subsystems in the S4.

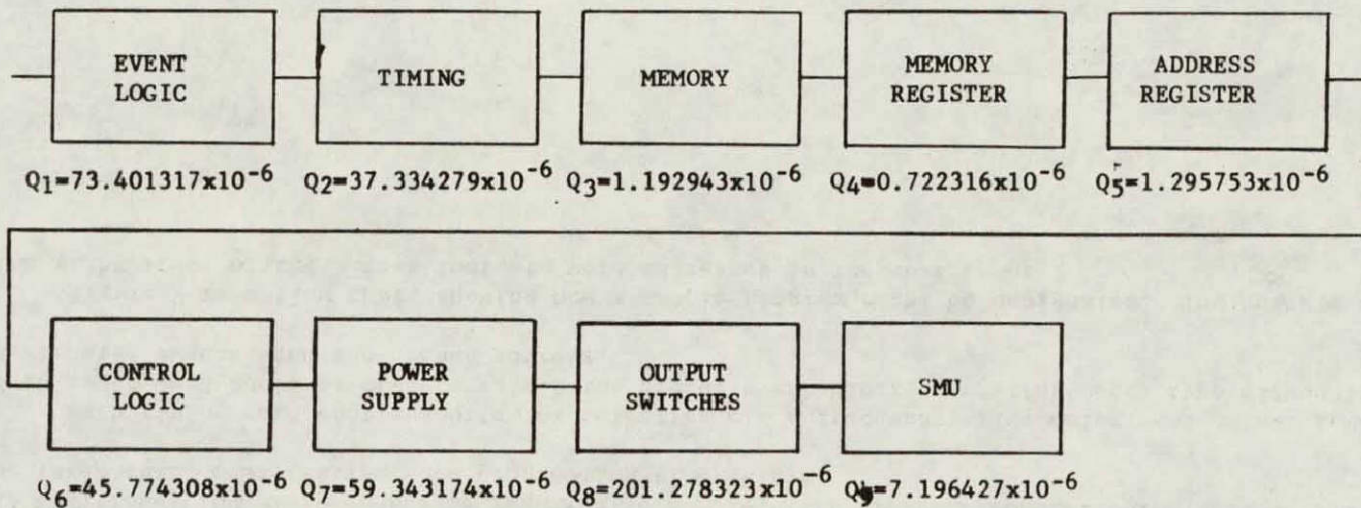
The 8MHZ oscillator was designed so that the circuit would operate only at 8MHZ. Component failure will cause the circuit to stop oscillating but cannot cause oscillations at a different frequency. This approach eliminates the need to provide a separate oscillator and the circuitry required to compare them.

Each output switch is series redundant, and both paths of logic within a channel must operate correctly for an output switch to be energized.

Figure IV presents a breakdown of system reliability by subsystem. The reliability of the output switches should be factored into the reliability equation separately since a failure in one output switch will not prevent all other output switches from functioning properly. However, as shown in the calculation, even if all outputs are lumped together, the system still meets the reliability goal.

MARTIN MARIETTA

DENVER DIVISION



$$\sum_{i=1}^9 Q_i = 427.538840 \times 10^{-6}$$

$$R_c = 1 - Q_c = 0.999,572,461; \quad Q_s = Q_c^2 = 0.182,789,460 \times 10^{-6}; \quad R_s = 1 - Q_s = 0.999,999,817$$

FIGURE IV  
Reliability Block Diagram of a Sequencer Channel

## V. Programming

Two types of system words are used; timing words and event words. Each timing word uses two memory words while each event word uses three memory words. System operation is such that 30 microseconds is required to read and process a timing word and 70 microseconds is required to read an event word and obtain an answer from the event logic. If the answer from the event logic is YES, an additional 100 microseconds is required to complete the branching operation. These delays must be taken into consideration when programming time delays.

Each timing word contains bits for selecting clock frequency, time delay, and output command. Each event word contains bits to select the proper event block, operating mode, type of equation, designated events, and the branch address.

Figure V is a flow chart showing how a sample program might be designated. The numbers on the right side of the blocks indicate word addresses in the core memory.

SAMPLE PROGRAM

MARTIN MARIETTA  
DENVER DIVISION

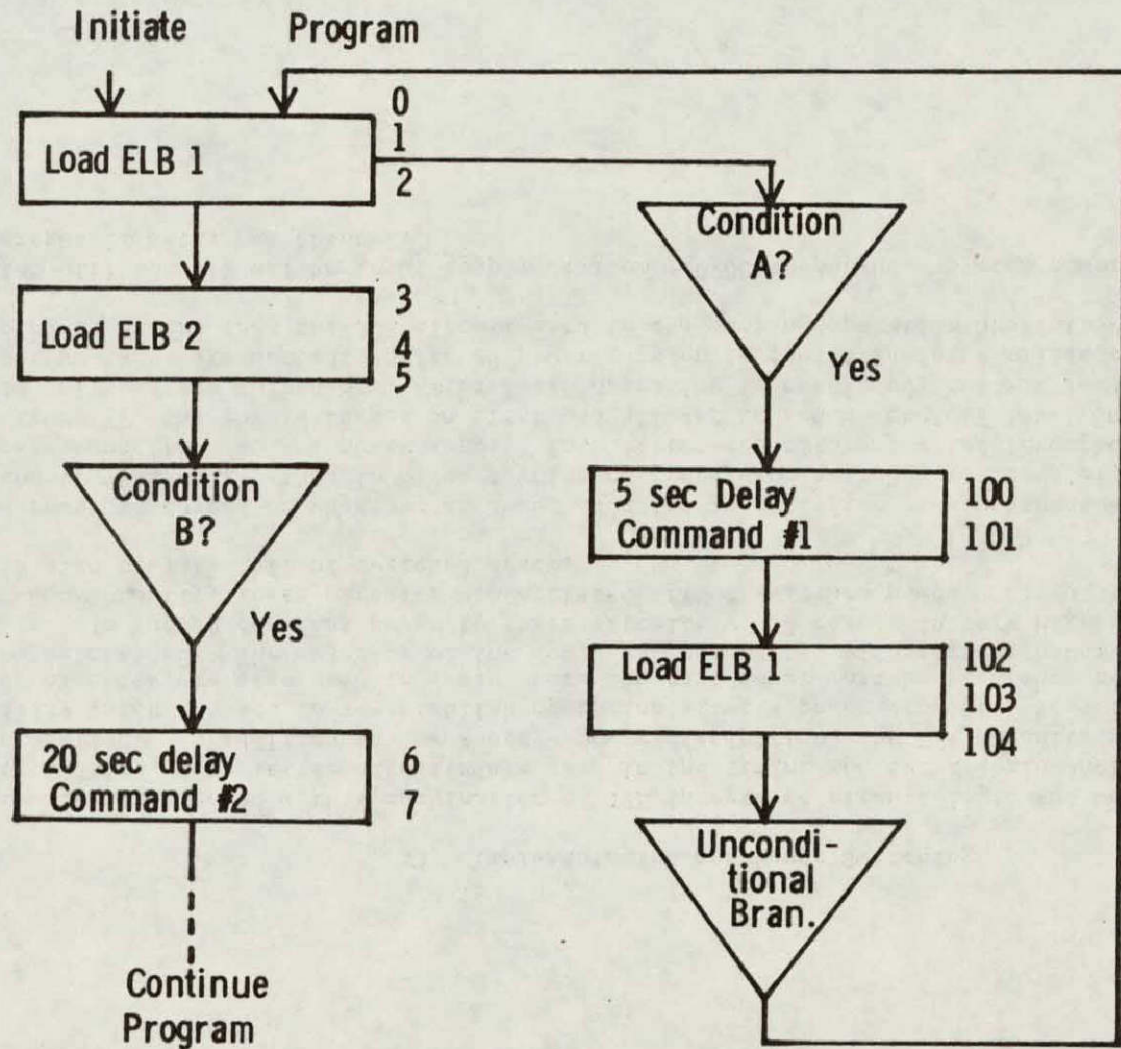


FIGURE V

369

## VI. Implementation and Power Switching

The system is implemented with a combination of TTL integrated circuit logic and core transistor logic circuits (CTL's). The latter circuits are used in the timing and event logic where constant power must be applied. In addition to low average power dissipation, the CTL circuits provide a memory capability which is used to re-establish operation after a power dropout. Some low power TTL integrated circuits are also used in areas where constant power must be used and where signal levels must be maintained. The majority of the logic is implemented with regular power TTL integrated circuits. To supply constant power to these circuits would result in very high system power consumption. Consequently, these circuits are operated off of switched power. All of the memory electronics is also operated off of switched power.

Switched power is turned on whenever an event equation is satisfied or upon the completion of a timing count. Basically, this involves turning on transistor switches to transfer energy from the storage capacitors in the power supply into filter capacitors located throughout the system. As shown in Figure VI, the +5V is turned on first and allowed to reach its full level before the other switched voltages are turned on. After logic operation is concluded, the +5V is maintained while the other voltages are dumped. Switched power turn on is inhibited for 6 milliseconds from the start of dumping to ensure that the SCR circuit used in the dumping operation has fully turned off.

The system will operate off an input supply of from +20VDC to +40VDC. System power consumption at +28VDC averages 11 watts per channel.

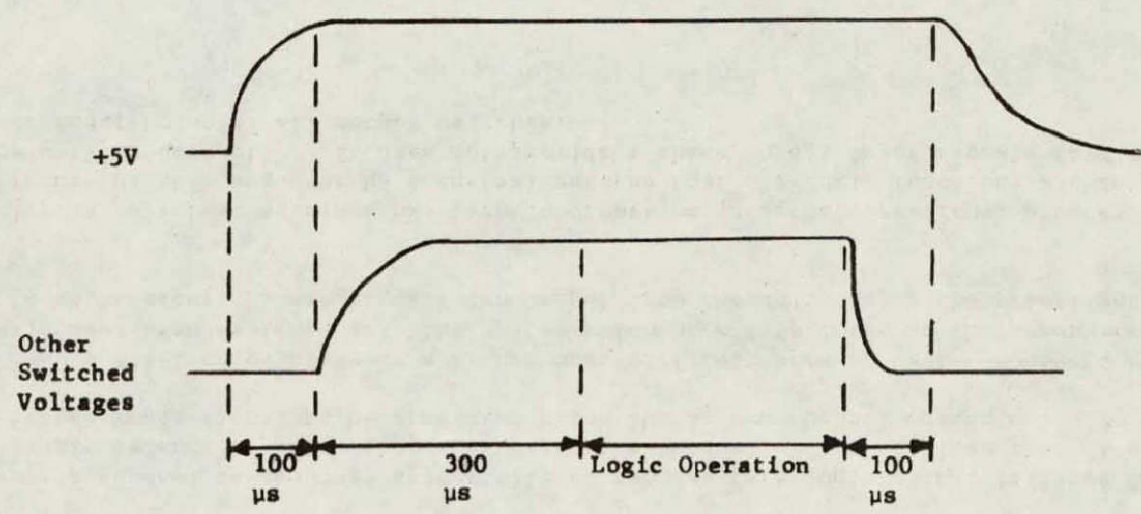


FIGURE VI  
Switched Voltage Operation

371

## VII. Testing and Fault Detection

The S4 has been designed with three levels of testing in mind. These are bench testing and maintenance, pre-flight checkout, and in-flight self test. Maintainability has been specified and designed to be at the module level.

Figure VII is a picture of a typical board containing integrated circuits. The two connectors at the top of the board are test point connectors which permit access to critical signals while the board is plugged into the housing. These test points permit failures to be isolated to the PC board level.

Because a channel is composed essentially of two separate logic paths, failures in a logic path are easily detected. This is done throughout a channel and any failure that is detected causes a System Fault signal to be generated which shuts down channel operation.

The memory electronics presents a rather more difficult area to test. However, special test signals have been designed into the system which permit failures in the memory redundant circuits to be detected. These signals can be put into the unit during the pre-flight test operation.

Finally, a self-test sequence has been incorporated into the operational program. Successful performance of this sequence by a channel ensures that all logic functions within the sequence are working correctly. It does not provide a check on all input signals (events) or on the correct functioning of all output switches.

NOT REPRODUCIBLE

373

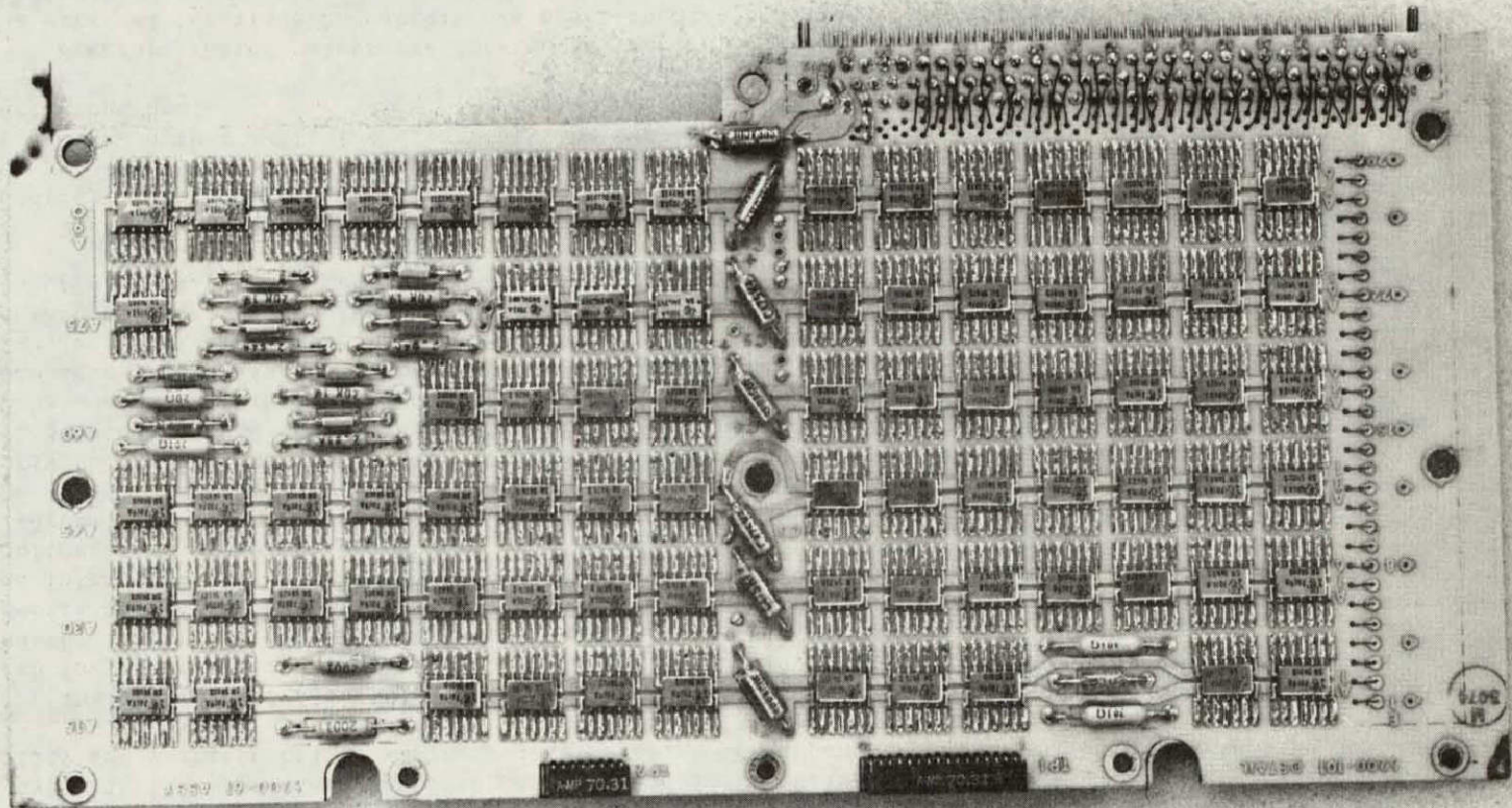


FIGURE VII  
Integrated Circuit Board



### VIII. Command Module Unit Packaging

The Command Module Unit (CMU) of the S4 (one channel) weighs 46.5 lbs and has a volume of 1330 in.<sup>3</sup> The CMU is packaged in a machined aluminum housing consisting of an upper housing with a cover and a base housing with a cover-shield. In Figure VIII, the base housing is in the foreground with the upper housing in the middle and the base cover-shield in the background. To ensure adequate checkout capability and maintainability at the module level, most components are mounted on plug-in type modules.

The upper housing contains the pyro firing circuit modules, the output switching circuit modules with low profile components, the memory assembly module, the decoding logic module, the CTL modules, and the integrated circuit modules. The modules consist of two printed circuit boards mounted back-to-back on a frame across the top and spacers in between. The connectors are offset on each board and interleaved to conserve headroom. The only exception to this is the pyro firing circuit module configuration where the components on each board face one another and are interleaved to conserve headroom. These modules plug into a mother board which provides the power and ground interconnection; the signals are hard wired. The memory assembly module is not a plug-in module but is mounted permanently to the housing structure. This module is located behind the name plate in the upper housing. All printed circuit board plug-in modules employ the use of a heat strippable, point-to-point wiring technique which permits packaging densities otherwise obtainable only with multilayer boards. This technique is a reliable way of making interconnections, reduces cost of boards, reduces procurement time, and facilitates making changes during build and checkout of a prototype system. The modules containing the fuse and fuse-resistors used in conjunction with the pyro firing circuits plug into the upper housing, at the top-front of the unit.

The base housing contains the power supply and the balance of the output switching circuits. The majority of the output switching components were large and required heat sinking. The base provided the proper environment for these. The power supply is completely shielded from the rest of the unit as are the output switching circuits. The base plugs into the upper housing to interconnect the two parts together.

Pressure relief valves are located at the rear of the case to provide for pressure differential. The external interface connectors are positioned on the front face of the unit. The modules containing the fuses and fuse-resistors that are used in conjunction with the pyro firing circuits plug externally into the upper housing at the top-front of the unit.

375

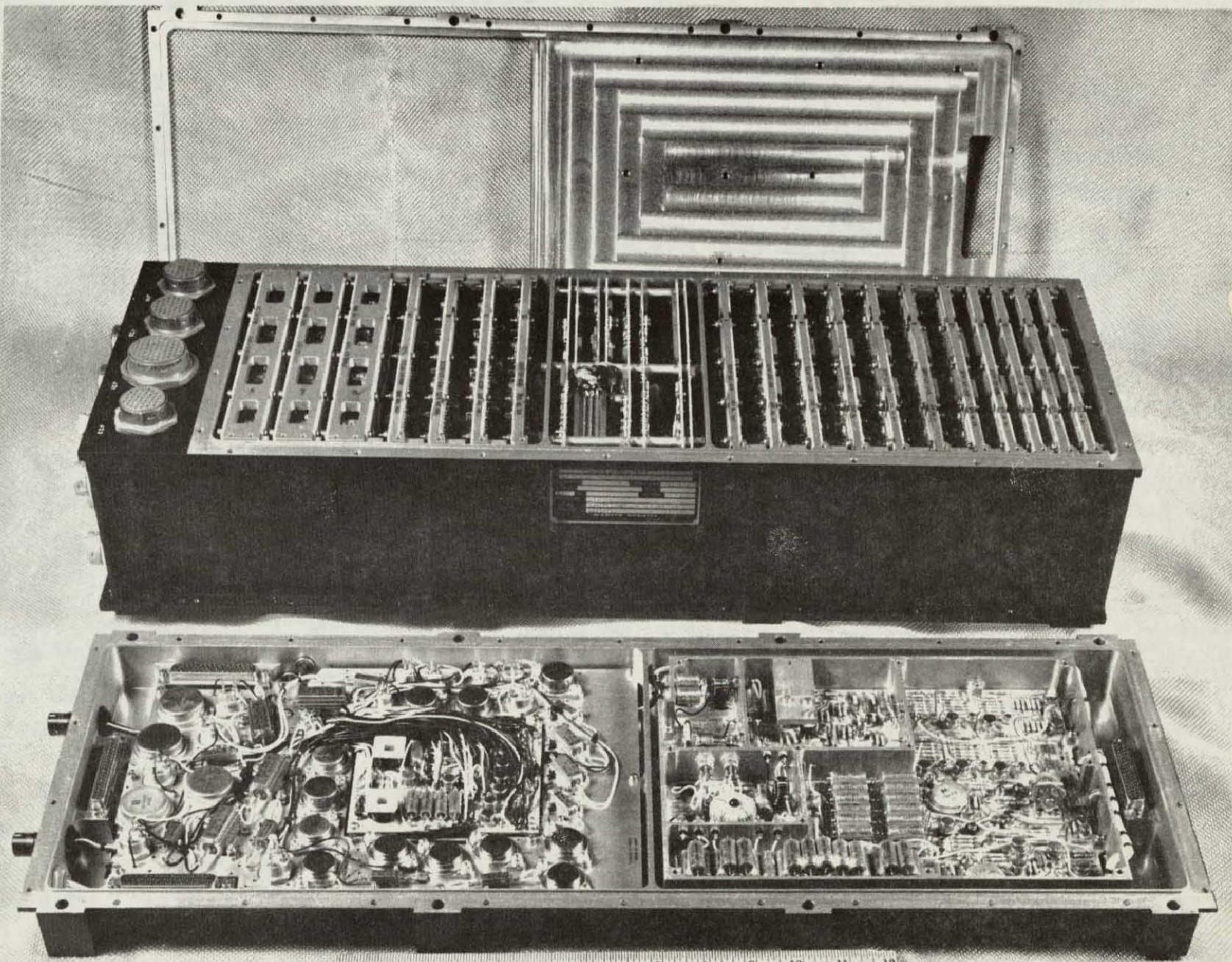


FIGURE VIII  
Disassembled CMU

### IX. Service Module Unit

In addition to the CMU, the S4 contains a separate small unit to perform the Service Module Jettison Controller functions. This unit called the SMU is shown in Figure IX. The SMU contains a unijunction relaxation oscillator, a CTL counter, power conditioning circuitry, and four solenoid drivers. All circuitry is serially redundant to prevent inadvertent outputs due to a single component failure. The SMU (one channel) weighs 3.5 lbs. and has a volume of 93 in<sup>3</sup>.

The SMU is packaged in a low profile machined aluminum case with a top cover. The low power dissipation components are mounted on two printed circuit boards in a stack configuration in the center of the case. The external connectors are at one end and the TO-3 devices are mounted on the other end on two plates in such a manner as to provide a heat path to the base mounting. Other components which require heat sinking are mounted directly to the case below the printed circuit boards. Interconnections are accomplished with a wire harness. The case is designed to provide for the pressure differential.

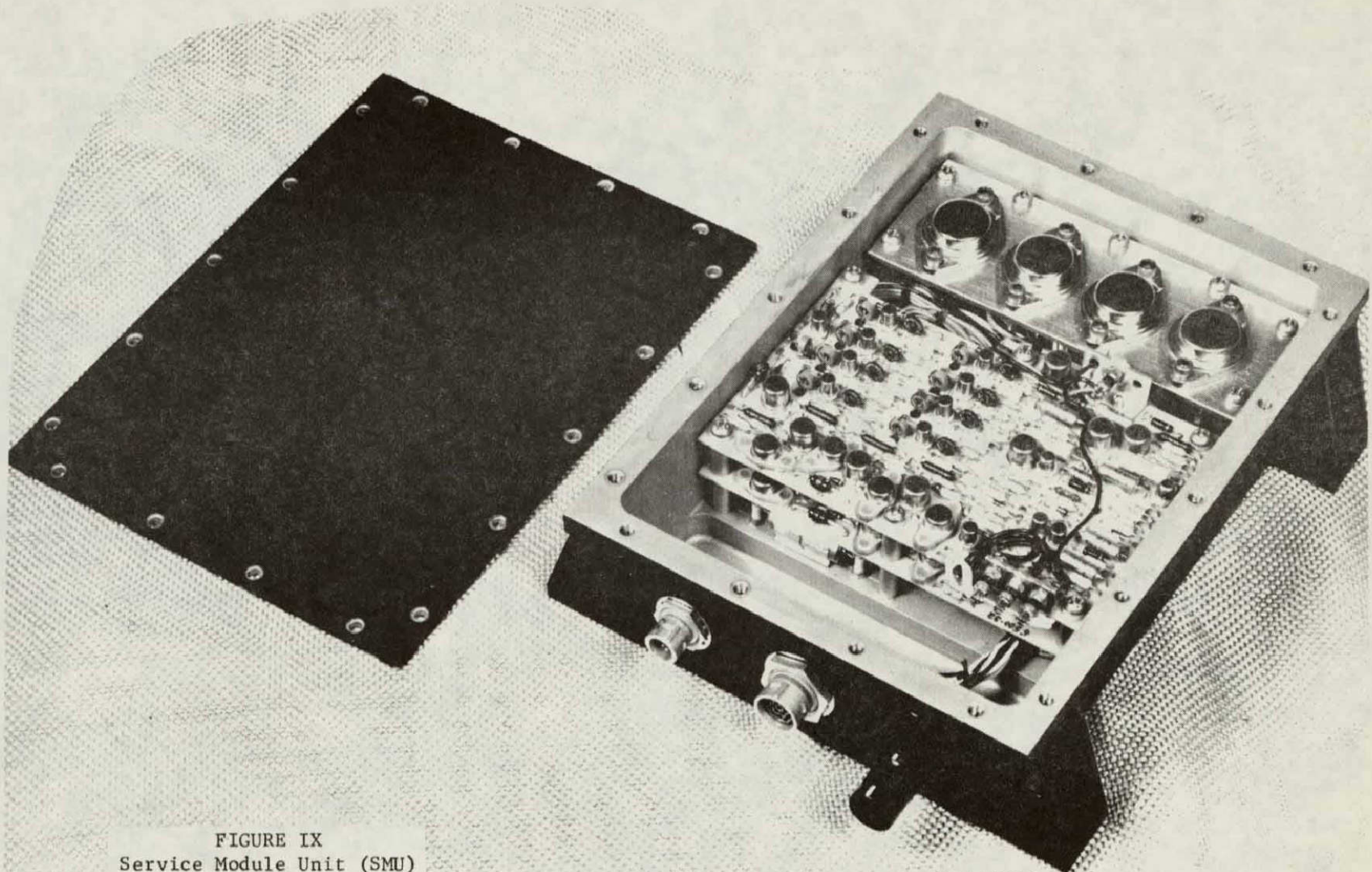


FIGURE IX  
Service Module Unit (SMU)



DOI SCIENCE/STALE

ATTENDEES OF THE SPACE SHUTTLE  
INTEGRATED ELECTRONICS TECHNOLOGY CONFERENCE

Government Attendees

NASA Headquarters

Hensley, R.  
Livingston, R. C.  
Michaels, T. S.  
Miller, W. E.  
Sullivan, F.  
Tischler, A. O.

Showman, R.  
Smith, D.  
Stevens, V. I.

Flight Research Center

Gee, S. W.

Wright-Patterson Air Force Base

Biernacki, J.  
Blatt, P. E.  
Doran, R. J.  
Gangl, E. C.

Goddard Space Flight Center

Chi, A. R.  
Dalle Mura, P. H.  
Major, F. G., Dr.

Edwards Air Force Base

Dyes, O.

John F. Kennedy Space Center

Bruns, R. H., Dr.  
Durrett, W. R.  
Edwards, M.  
Greenfield, T. D.  
Harrington, R.  
Hershey, T.  
Keenan, T.  
LaPorte, C.  
Lealman, R.  
Messing, W.  
Michalek, T.  
Nicholas, L.  
Paul, H.  
Penovich, F.  
Roseland, N.  
Smith, R. D.

Space and Missiles Systems Organization

Collins, D. R.  
Kuehner, W. E., Lt.

Jet Propulsion Laboratory

Carl, C.

Office of Naval Research

Scoggins, L.

Langley Research Center

Croswell, W. F.  
Gilreath, M. C.  
Moore, W. M.

Ames Research Center

Dodge, J. E.  
McGee, L.

Lewis Research Center

Stover, J. B.

George C. Marshall Space Flight Center

Adair, B. M.  
Aden, R. M.  
Baggs, E. T.  
Baker, D.  
Beasley, B.  
Beltran, A.  
Blackstone, J. H.  
Blair, J. C.  
Brown, H. E.  
Carlile, C. D.  
Davis, P. R.  
Doane, G. B., III  
Doran, B. J.  
Elms, C.  
Emens, F. H.  
Felch, J. L.  
Gallaher, B. E.  
Gause, R. L.  
Gilino, N. R.  
Hafner, A.  
Hall, G. E.  
Hamiter, L. C.  
Hight, H.  
Huie, H. A.  
Kaufman, J. J.  
Kerner, H.  
Knadler, J. R.  
Lovingood, J. A.  
Lowery, D. O.  
Lowery, H. R.  
McPeak, B. J.  
Mitchell, R. E.  
Moore, F. B.  
Nichols, D.  
Porterfield, H. D.  
Powell, J. T.  
Reed, B.  
Shockley, W. J.  
Stulting, J. B.  
Thompson, H. E.  
Trauboth, H. H.  
Turner, G. L.  
Van Orden, E. R.  
Vinz, F. L.

Williams, D. E.  
Willis, A. E.  
Wolf, R. K.  
Woodruff, L. D.

Manned Spacecraft Center

Aldrin, E. E.  
Algranti, J. S.  
Allday, C.  
Arndt, G. D.  
Bachman, S.  
Baiamonte, F. L.  
Barron, D.  
Barry, T.  
Batson, B. H.  
Becker, A. R.  
Becker, B.  
Beckman, D. A.  
Beers, C. A.  
Bell, J.  
Bertrand, O. J.  
Bigham, J. P.  
Boyd, D. L.  
Boykin, J. C.  
Bradford, W. C.  
Brandenburg, J. R.  
Burghduff, R. D.  
Burtzlaff, I. J.  
Cabra, O., Jr.  
Campos, A. B.  
Carl, R.  
Cassetti, M.  
Chambers, T. V.  
Chapman, J. T.  
Cheatham, D. C.  
Chevers, E. S.  
Chilton, R. G.  
Cook, D. R.  
Cox, K.  
Cree, D.  
Cubley, D.  
Dalke, E. A.  
Davila, H. J.  
Dawson, C.  
DeMoss, J. F.  
Dietz, R. H.  
Duncan, S. K.  
Duval, R. W.  
Dvorkin, D.

Dyer, R. C.  
 Eastman, F. E.  
 Edmiston, C.  
 Eickmeier, A. B.  
 Erickson, J. E.  
 Erwin, H.  
 Essmier, C., Maj.  
 Evans, J.  
 Evans, W. B.  
 Farley, A. R.  
 Farmer, N.  
 Fenner, W.  
 Ferguson, J. E.  
 Fitzgerald, P.  
 Fowler, J.  
 Fox, J. B.  
 Fullerton, C. G.  
 Gammon, E. P.  
 Gardiner, R. A.  
 Garland, J.  
 Gaudiano, S.  
 Gilbert, D.  
 Goad, J. W.  
 Graves, C.  
 Grimmin, J.  
 Hackler, C.  
 Hambleton, A.  
 Hamilton, L.  
 Hamilton, M.  
 Hanaway, J. F.  
 Hannigan, J.  
 Harlan, C.  
 Harpold, J.  
 Harrin, E. N.  
 Harris, C. D.  
 Hartsfield, H. W.  
 Hector, G. D.  
 Henderson, M.  
 Henderson, S.  
 Hendrix, M. K.  
 Heselmeyer, R.  
 Hill, W. A.  
 Hobokan, A.  
 Hohmann, R.  
 Holley, M.  
 Holloway, G.  
 Holmes, Q. A.  
 Hood, B.  
 House, S. G.  
 Howard, H.  
 Hughes, J.  
 Humphries, T.  
 Huss, C.  
 Hyle, T.  
 Irvin, R.  
 Jenkins, M.  
 Johnson, B. E.  
 Johnson, D. A.  
 Johnson, G. W.  
 Johnson, L.  
 Jones, M.  
 Jordon, K.  
 Kahanek, J.  
 Kaiser, T. H.  
 Keathley, M.  
 Keetan, T.  
 Kennedy, R.  
 Klinar, J.  
 Koontz, C.  
 Kosinski, R.  
 Kramer, P. C.  
 Kyle, H. C.  
 LaComb, D. R.  
 Lamereux, J.  
 Land, C. K.  
 Lang, D. D.  
 Langdoc, W. A.  
 Lattier, E.  
 Laubach, C.  
 Lawrence, J.  
 Leverich, W. B.  
 Levy, C. D.  
 Lewis, J. L.  
 Lewis, R.  
 Lewis, T.  
 Lively, C.  
 Loden, H.  
 Loftus, J.  
 Long, D.  
 Long, W.  
 Mallery, W. E.  
 Marcantel, B.  
 Marlowe, G. D.  
 McConnell, W. R.  
 McCreary, B.  
 McCullough, C. E.  
 McElwee, E. M.  
 McLain, D.  
 Mechelay, J.  
 Melliff, V. C.  
 Melugin, J. F.  
 Middleton, W. A.

Milhoan, J.  
Miller, W.  
Minar, L., Maj.  
Mollberg, B. H.  
Moore, D.  
Moorehead, R. W.  
Munford, R. E.  
Myers, B.  
Nance, S. C.  
Nassiff, J.  
Neily, C. M.  
Nelson, D.  
Nicolson, D.  
Nolting, A. G.  
North, W. J.  
O'Brien, D. E.  
Oldham, W. B., Jr.  
Olsen, A. B.  
Olslewski, O.  
Osgood, D.  
Otten, H.  
Packham, L. E.  
Parker, C. B.  
Patterson, H.  
Patterson, O.  
Peck, J. C.  
Peters, B.  
Petterson, D. H.  
Pitts, D. L.  
Pixley, P.  
Poulos, P.  
Price, C.  
Price, T.  
Pringle, L. M.  
Riegert, D.  
Ritterhouse, C.  
Ritz, W.  
Robinson, W. G.  
Rosenberg, H. R.  
Ruetz, L.  
Sabionski, G.  
Savely, R.  
Sawyer, R. S.  
Shelton, D.  
Sheppard, J.  
Shone, J. L.  
Simpson, R.  
Smith, D. R.  
Smith, H.  
Smith, J. R.  
Smith, R. J.

Stagg, W. C.  
Stokes, R. E.  
Strickland, J.  
Swigert, J. L.  
Swint, R.  
Takao, K.  
Talbert, R.  
Teasdale, W.  
Thomas, C. C.  
Tiedt, E. W.  
Trahan, W. N.  
Uzzel, B.  
Vavra, P. H.  
Vernon, J.  
Vincze, J.  
Wadle, R.  
Waite, J.  
Walsh, C.  
Walter, R. T.  
Warnick, C. D.  
Warren, D. K.  
Wasson, C.  
Watkins, C. W. L.  
Weldon, J. W., Jr.  
West, J.  
Wetterstroem, A.  
Woodfill, J. R.  
Xenakis, G.  
Zrubek, W.



## Foreign Attendees

Mr. H. Do  
Centre Spatial de Bretigny  
91 Bretigny S Orge, France

Mr. P. G. Bish  
British Embassy  
Washington, D. C.

Mr. Y. Ferraton  
Thomson-CSF  
Malakoff, France

Prof. Falleni  
Motdel  
Milan, Italy

Mr. Hans E. W. Hoffmann  
ELDO  
Neully, France

Mr. G. Pellegrineschi  
ELDO  
Neully, France

Mr. S. Hieber  
ELDO  
Neully, France

Dr. C. Reinhold  
ESC - Liaison Office  
Washington, D. C.

Chikao Ohara  
Electronics Equipment Lab.  
Toshiba Research and Development Center  
Kawasaki 210, Japan

Dr. Rupf  
Bundesministerium fur Bildung und Wissenschaft  
Heussalle 2 - 10, Germany

Mr. B. G. Cameron  
Electronics Division  
Electrical and Electronics Branch  
Department of Industry, Trade and Commerce  
Ottawa, Ontario, Canada

Mr. John H. Crysdale  
Office of Science and Technology  
Department of Industry, Trade and  
Commerce  
Government of Canada  
Ottawa, Ontario, Canada

Mr. Takaski Ishikawa  
Embassy of Japan  
Washington, D. C.

I. A. Marri  
Selenia S. P. A. via Tiburtina  
Rome, Italy

Industry Attendees

AEROJET GENERAL CORPORATION  
Houston, Texas

Boyce, C.  
Stephenson, G.

AEROJET NUCLEAR SYSTEMS COMPANY  
Sacramento, California

Porter, L. K.

THE AEROSPACE CORPORATION  
Los Angeles, California

Farrar, R. J.  
Foy, R. H.  
Rose, E. H.

Burlington, Massachusetts

Wadden, W. R.  
Zvara, J.

AIL (CUTTLE HAMMER)  
Farmingdale, L. I., New York

Barlam, T.

AMBAC INDUSTRIES  
Garden City, New York

Berke, L. C.  
Litman, B.

AMPEX CORPORATION

Quillen, E.

ATLANTIC RESEARCH (SUSQUEHANNA  
CORPORATION)  
Alexandria, Virginia

Crenca, J. J.

AUTONETICS (NORTH AMERICAN  
ROCKWELL)  
Anaheim, California

Jurison, J.  
Streeter, J. R.

AUTONETICS (NORTH AMERICAN  
ROCKWELL)  
Dallas, Texas

Massey, H. R.

AVCO CORPORATION  
Cincinnati, Ohio

Bopp, C. C.  
Mullaney, J. W.

Houston, Texas

Lipovsky, V.

BELL AEROSPACE COMPANY  
Buffalo, New York

Engen, D. T.  
McKee, D.  
Rusnak, W. J.

BENDIX CORP.  
Denver, Colorado

Ellman, A.

Ann Arbor, Michigan

Grant, R. A.  
Linder, R.  
Loughry, B.

Teterboro, New Jersey

Epstein, M.  
Fithian, D. H.  
Haefeli, J.  
Kendall, K.  
Lamberski, S.  
Shay, J.

Davenport, Iowa

Schebler, B. J.  
Stanley, N. E.

BOEING COMPANY  
Houston, Texas

Anderson, R. H.  
Baird, B.  
Elliott, E.  
Golub, R.  
Gunderson, J. M.  
Klopfenstein, H. W.  
Krier, C.  
Linton, A. T.  
Neale, W.  
Prongay, D. M.  
Snyder, J.  
Toliver, C. L.

Seattle, Washington

Brane, F. H.  
McMahon, R. M.  
Petrie, D. M.

New Orleans, Louisiana

Pastorick, W. J., Jr.  
Taliancich, P.

BRITISH AIRCRAFT CORPORATION

Young, G. M.

BROWN AND ROOT - NORTHROP  
Houston, Texas

Lamprose, S. L.

BUNKER RAMO CORPORATION  
Chatsworth, California

Baur, R.  
Bower, R.  
Greenblum, C.  
Silbert, W. C.

Oak Brook, Illinois

Casillas, F. C.

CHRYSLER SPACE DIVISION  
New Orleans, Louisiana

Juenglinz, W.

COLLINS RADIO COMPANY  
Cedar Rapids, Iowa

Albinger, R. A.

COMPUTER SCIENCES CORP.  
Washington, D. C.

Baumeister, J. K.

CONTROL DATA CORPORATION  
Houston, Texas

Cassola, R. L.  
Mendoza, A., Jr.

CORNELL AERONAUTICAL LAB., INC  
Buffalo, New York

Deazley, W. R.

CUBIC CORPORATION  
San Diego, California

Grady, T.  
Krenz, D.  
Troester, R. H.

DELCO ELECTRONICS DIVISION (GMC)  
Houston, Texas

Denissen, J. W.  
Korth, K. G.

DELCO ELECTRONICS DIVISION (GMC)  
Milwaukee, Wisconsin

Cavalier, M.  
Genna, J. F.  
Hanley, T.  
Larson, P.  
Lynn, M.  
Smith, L.  
Stridde, J.  
Swackhamer, P. K.  
Wachholz, J.  
Ziemer, D.

ELECTRONIC SYSTEMS INTEGRATION  
Burbank, California

McLeod, R. H.  
Stewart, R. A.  
Storker, S. A.

FAIRCHILD HILLER CORPORATION  
Germantown, Maryland

Metzger, W.

FAIRCHILD INDUSTRIES  
Houston, Texas

Colovin, J.

GENERAL DYNAMICS  
Orlando, Florida

Sherlin, R. E.

San Diego, California

Cantrell, W.  
Grunsky, C.  
Herz, E.  
Huggin, F. E.  
Jones, D. J.  
Klawa, R. F.  
Mawson, T. J.  
Munday, T. W.  
Mraz, J. Z.  
Newman, H.  
Peterson, R. I.  
Walter, R.

GENERAL ELECTRIC  
Binghamton, New York

Ragland, H. F.

Houston, Texas

Ettus, M.  
Feldman, S.  
Johnson, A. W.  
Joslyn, J.  
Kelly, C. R.  
King, J.  
Krueger, E.  
Salzar, D.  
Sawberger, F. H.  
Scherrer, O. D.  
Schramm, G.  
Warzecha, L. W.

Utica, New York

Frey, C.  
Goodman, I.  
Heurung, W.  
Kroeger, R.  
Shaw, A. L.  
Wimmer, W.

Daytona Beach, Florida

Lane, J. R.  
Towles, R.

Philadelphia, Pennsylvania

Macklis, S.  
Slanker, E.

Huntsville, Alabama

Rosing, S.  
Scott, J.

GRUMMAN AEROSPACE CORPORATION  
Bethpage, New York

Abrahams, S.  
Alexandrovich, A.  
Anderson, D.  
Bernstein, J.  
Caraceni, J.  
Cotter, G.  
Friedenreich, G.  
Gran, R.  
Kaline, J.  
Mooney, C.  
Prasinos, T.  
Reilly, G.  
Rossoff, A.  
Stein, D.  
Tellalian, R.  
Vitale, P.  
Wright, H. T.

HAZELTINE CORPORATION  
Little Neck, New York

Sherman, H. P.

HONEYWELL, INCORPORATED

Edwards, P.  
Mellen, D. L.  
Phelps, R. K.  
Smith, P. R.

St. Paul, Minnesota

Moynihan, F. A.  
Rupert, J. R.  
Sowada, D.  
Stone, C. R.

HUGHES AIRCRAFT COMPANY  
Culver City, California

Anderson, L.  
Harney, E. D.  
Julian, R. L.  
McShane, P.

INTERMETRICS  
Cambridge, Massachusetts

Martin, F. H., Dr.  
Miller, J., Dr.  
Miller, John  
Widnall, W., Dr.

IBM CORPORATION  
Huntsville, Alabama

Hundley, J. C.  
Jacowitz, L. A., Dr.  
Pachuta, J. R.  
Stuver, P. C.

Owego, New York

Gallagher, J. P.  
Patzner, W. J.  
Yeager, A. P.

Washington, D. C.

Grimesey, R. P.

Houston, Texas

Hayes, M.  
Kagg, R. C.  
Leedlam, L. D.  
Muths, G. A.  
Young, W. A.

ITT

Presto, A. F.

INSTRUMENTS SYSTEMS CORPORATION  
Huntington, New York

Margeson, J. H.

KAISER AEROSPACE  
Palo Alto, California

McAfee, C.  
Sable, J.

KOLLSMAN INSTRUMENT CORPORATION  
Jackson Heights, New York

Englert, L. J.

LEACH CORPORATION  
Dallas, Texas

Caton, R.

LITTON SYSTEMS, INCORPORATED  
Woodland Hills, California

Wheeler, D.

LOCKHEED ELECTRONICS CO., INC.  
Houston, Texas

Baumel, L.  
Bennett, R. W.  
Bonin, L. J.

Bolton, G. R.  
Bowes, H. N.  
Brown, J.  
Chang, J. C.  
DeRousse, R. H.  
Dills, R. S.  
Doland, G. D.  
Errington, A.  
Franklin, P.  
Haddican, P. J.  
Hansen, F.  
Harton, P. L.  
Hopkins, P. M.  
Lyden, J. A.  
Lynch, J.  
Martin, A.  
Naumann, A.  
Prock, G.  
Rao, V. R.  
Rathbone, W. M.  
Shellhase, M. W.  
Simpson, R.  
Steadman, B. E.  
Taqui, S.  
Theobald, R. T.  
Thompson, R. D.  
Varsos, S. G.  
Vincent, J.  
Wallingford, W. M.  
Witty, R. D.  
Wood, M. E.

LOCKHEED MISSILES AND SPACE CO.  
Sunnyvale, California

Brandon, J. W.  
Courtney, J. D.  
Guill, J. H., Dr.  
Hill, L. J.  
Trapp, A.  
Wilson, D.

LOCKHEED AIRCRAFT CORP.  
Burbank, California

Childs, H. A.

LOCKHEED GEORGIA CO.  
Marietta, Georgia

Speer, T. K.

LOGICON, INCORPORATED  
San Pedro, California

Stuart, W. H.

LORAL ELECTRONIC SYSTEMS  
Bronx, New York, and Dallas, Texas

Stewart, R.  
Storper, S. A.

M AND S COMPUTING, INCORPORATED  
Huntsville, Alabama

Thurber, R. E.

MIT/C. S. DRAPER LABORATORY  
Cambridge, Massachusetts

Kriegsman, B.  
Ragan, R. R., Deputy Director.  
Weinstein, W.

Houston, Texas

Lawton, T.

MARTIN MARIETTA  
Denver, Colorado

Hallmark, B.  
Hardin, R.  
Hart, W.  
Korgel, C. C.  
Murphy, D.

McDONNELL DOUGLAS ASTRONAU-  
TICS CO.  
Houston, Texas

Flowler, M.  
Glowczwski, R.  
Hayes, W. E.  
Klint, E. O.  
Puschinsky, R.  
Schaeffer, C.  
Soule, P.

McDONNELL DOUGLAS ASTRONAU-  
TICS CO.  
Huntington Beach, California

Hartman, R. M.  
Todd, H. J.  
Jerome, R. W.

St. Louis, Missouri

Doumerc, A. R.  
Freeman, D. J.  
Kuehl, J.  
Kuhlman, E. A.  
McKee, G. G.  
Reed, P. J.  
Rosamond, D.  
Wamser, R. C.

MOTOROLA, INC.  
Scottsdale, Arizona

Gallup, B.  
Toberman, A.

NORTH AMERICAN ROCKWELL  
CORPORATION  
Downey, California

Anderson, G. C.  
Binns, D.  
Engels, B. A.  
Florey, J.  
Githens, S. E.  
Knowlen, T. H.  
Levine, B.  
Livingston, J.

NORTH AMERICAN ROCKWELL  
CORPORATION  
Downey, California

Minor, R. J.  
Mehrbach, E.  
O'Hern, E.

Houston, Texas

Hazel, F. J.

Huntsville, Alabama

Myers, J. F.

NORTHROP CORPORATION  
Norwood, Massachusetts

Herweg, J. B.

RADIATION, INC.  
Washington, D. C.

Barnes, B. P.  
Stumpe, J.

NORTHROP CORPORATE LABORATORIES  
Hawthorne, California

Malm, R., Dr.

RAYTHEON COMPANY  
Sudbury, Massachusetts

Maynard, O. E.

PHILCO/FORD

Denson, W. J.  
Poirier, R. F.

RESEARCH TRIANGLE INSTITUTE  
Research Triangle Park, North Carolina

Smith, J. R.

PRD ELECTRONICS (HARRIS INTERTYPE)  
Westbury, New York

Lowitt, P.

SCI ELECTRONICS, INC.  
Houston, Texas

Hickman, D.

RCA  
Burlington, Massachusetts

Aronson, J. D.  
Heavy, J. J.  
Wadden, W. R.  
Zvara, J.

SCI SYSTEMS, INC.  
Huntsville, Alabama

Gage, R.  
Olson, R.

Camden, New Jersey

Hartshorne, F.  
Holt, S. B.  
Nossen, E. J.  
Saultz, J. E.  
Sullivan, W. E.  
Zieper, H. S.

S.I.P., INC.  
Houston, Texas

Fourrier, F. L.  
Lederer, A., III  
Walker, V.



SINGER-GENERAL PRECISION, INC.  
Link Division  
Houston, Texas

Dailey, W.  
Morris, R.  
Ragsdale, A.

SINGER-GENERAL PRECISION, INC.  
Kearfott Division

Knapp, R.

Dallas, Texas

Mahoney, T. R.

Wayne, New Jersey

Guberman, M.  
Zitovsky, S.

SPERRY RAND (UNIVAC)

Evans, J. D.  
Perriss, K.

St. Paul, Minnesota

Wolfe, T. O.

Huntsville, Alabama

Armstrong, T. R.

Phoenix, Arizona

Boudreaux, H.  
Boyle, P.  
Reynolds, C.  
Shuck, L.

Dallas, Texas

Courtney, K. C.  
Suggs, K.

SYSTEMS DEVELOPMENT CORPORATION  
Houston, Texas

Haake, J.  
Harman, H. A.

TRW SYSTEMS GROUP  
Houston, Texas

Arbuthnot, G. L.  
Baker, K.  
Boudreau, R. J.  
Clark, C. W.  
Coffman, J. L.  
DeVillier, J.  
Girod, W.  
Harwood, W. F.  
Herwig, H.  
Kastelein, J.  
Lee, R.  
Leisenring, J.  
McClure, D.  
Mellon, E. A.  
Phellips, D.  
Scheppau, J. E.  
Schook, G. R.  
Smith, D.  
Snygg, J.  
Widdifield, J.

Redondo Beach, California

Bettwy, T. S.  
Brady, H.  
Daly, T. E.  
Ergin, E.  
Greenbaum, H.  
Kayton, M., Dr.  
Morrison, R.  
Yolken, B. S.

TELEDYNE SYSTEMS CO.

Palmer, W.

UNITED AIRCRAFT (NORDEN)  
Norwalk, Connecticut

Conron, G. F.  
Kingston, W. W.  
Richardson, C. J.

UNITED AIRCRAFT (HAMILTON-STANDARD)  
Houston, Texas

Carrol, R.  
Dougan, D. P.  
Holsing, J.

UNITED AIRCRAFT (PRATT AND WHITNEY)  
Houston, Texas

Kernodle, B.  
Lassen, M. M.

WESTINGHOUSE ELECTRIC CORPORATION  
Baltimore, Maryland

Hoffman, C. P.  
Holthaus, J. E.  
Rassa, R. C.  
Shapiro, G.

Lima, Ohio

Geyer, M. A., Dr.

Houston, Texas

McDuffie, G. L.

ALABAMA A&M UNIVERSITY

Lawsine, L.