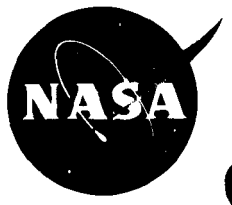
NASA SP-8070

NASA
SPACE VEHICLE
DESIGN CRITERIA
(GUIDANCE AND CONTROL)

SPACEBORNE DIGITAL COMPUTER SYSTEMS



MARCH 1971

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

GUIDE TO THE USE OF THIS MONOGRAPH

The purpose of this monograph is to organize and present, for effective use in spacecraft development, the significant experience and knowledge accumulated in development and operational programs to date. It reviews and assesses current design practices, and from them establishes firm guidance for achieving greater consistency in design, increased reliability in the end product, and greater efficiency in the design effort. The monograph is organized into three major sections that are preceded by a brief *Introduction* and complemented by a set of *References.*

The *State of the Art*, section 2, reviews and discusses the total design problem, and identifies *which* design elements are involved in successful designs. It describes succinctly the current technology pertaining to these elements. When detailed information is required, the best available references are cited. This section serves as a survey of the subject that provides background material and prepares a proper technological base for the *Design Criteria* and *Recommended Practices.*

The *Design Criteria*, shown in section 3, state clearly and briefly what rule, guide, limitation, or standard must be imposed on each essential design element to insure successful design. The *Design Criteria* can serve effectively as a checklist for the project manager to use in guiding a design or in assessing its adequacy.

The *Recommended Practices*, as shown in section 4, state how to satisfy each of the criteria. Whenever possible, the best procedure is described; when this cannot be done concisely, appropriate references are provided. The *Recommended Practices*, in conjunction with the *Design Criteria*, provide positive guidance to the practicing designer on how to achieve successful design.

Both sections have been organized into decimally numbered subsections so that the subjects within similarly numbered subsections correspond from section to section. The format for the Contents displays this continuity of subject in such a way that a particular aspect of design can be followed through both sections as a discrete subject.

The design criteria monograph is not intended to be a design handbook, a set of specifications, or a design manual. It is a summary and a systematic ordering of the large and loosely organized body of existing successful design techniques and practices. Its value and its merit should be judged on how effectively it makes that material available to and useful to the user.

For sale by the National Technical Information Service, Springfield, Virginia 22151 — Price $3.00

FOREWORD

NASA experience has indicated a need for uniform criteria for the design of space vehicles. Accordingly, criteria are being developed in the following areas of technology:

Environment
Structures
Guidance and Control
Chemical Propulsion

Individual components of this work will be issued as separate monographs as soon as they are completed. This document, *Spaceborne Digital Computer Systems,* is one such monograph.

A list of all previously issued monographs can be found at the back of this publication.

These monographs serve as guides to NASA design and mission planning. They are used to develop requirements for specific projects and also are cited as the applicable references in mission studies and in contracts for design and development of space vehicle systems.

This monograph was prepared for NASA under the cognizance of the Jet Propulsion Laboratory, California Institute of Technology. Principal contributors were Mr. William C. Hoffman of Aerospace Systems, Inc., Professor Albert L. Hopkins, Jr. of the Massachusetts Institute of Technology, and Mr. John P. Green, Jr. of Intermetrics, Inc. The program manager was Mr. John Zvara of Aerospace Systems, Inc.

The effort was guided by an advisory panel which was chaired by Professor Hopkins. The following individuals participated in the advisory panel and monograph review activities:

A. A. Avizienis	University of California, Los Angeles
D. O. Baechler	Bellcomm, Inc.
T. C. Bartee	Harvard University
J. M. Black	NASA Flight Research Center
J. V. Christensen	NASA Ames Research Center
B. M. Dobrotin	Jet Propulsion Laboratory
B. L. Dove	NASA Langley Research Center
B. J. Jansen	UNIVAC, Defense Systems Division
L. R. Manoni	United Aircraft, Hamilton Standard Systems Center
W. J. Patzer	IBM, Federal Systems Division
D. H. Schaeffer	NASA Goddard Space Flight Center
G. P. Talcott	Raytheon Co., Equipment Division
W. E. VanderVelde	Massachusetts Institute of Technology

Contributions in the form of design and development practices were also provided by many other engineers of NASA and the aerospace community.

Comments concerning the technical content of this monograph will be welcomed by the National Aeronautics and Space Administration, Office of Advanced Research and Technology (Code RE), Washington, D.C. 20546.

March 1971

CONTENTS

SPACEBORNE DIGITAL COMPUTER SYSTEMS

1. INTRODUCTION

As space vehicle missions have become more complex, the use of onboard digital computers has become more prevalent. The functions which these computers are assigned to perform are also expanding in number and magnitude. As a result, the problem of specifying and designing digital computers for space vehicles has increased in complexity.

Although most spaceborne digital computers are of the type often referred to as "general purpose," they have been in fact special-purpose machines in that a particular choice of design must reflect the requirements of the particular mission application. Thus, the program manager must be aware of the capabilities and limitations of spaceborne computer systems and the design tradeoffs which might affect his application.

The flight performance of spaceborne digital computer systems has generally been successful. However, a number of recurring problems have been experienced during the design, development, and testing of these machines. Previous systems have been very costly, have required major redesigns, and have caused significant schedule delays. Most difficulties have resulted from 1) lack of adequate capacity and flexibility to accommodate expanded requirements, 2) poorly defined subsystem and interface specifications, 3) the impact on software of changing mission requirements, and 4) reliability demands.

Important factors which influence the design and performance of spaceborne digital computer systems include:

- System architecture
- Computational capability (precision, speed, throughput, memory capacity, input/output capability, instruction repertoire, etc.)
- Adaptability (expandability, flexibility, compatibility, etc.)
- Provisions for interface with other components of the system
- Software (support and applications programs, ease of programming, etc.)
- Cost (money, weight, power, volume, time)
- Reliability related items (fault tolerance, failure rate, redundancy, ease of checkout, etc.)
- Environment (temperature, shock and vibration, electromagnetic and nuclear radiation, noise, power fluctuations, etc.)
- Packaging and cabling design

The preferred design should consider the expanding nature of the requirements, potential advances in the technological state of the art, and the entire spectrum of environmental requirements. It should strike a balance between hardware complexity and software simplicity, and facilitate simulation, testing, and checkout.

This monograph discusses considerations which form a basis for the specification, design and evaluation of digital computer systems for spaceborne applications. Detailed discussion of the following items are outside the scope of this monograph: software development, mechanical and electrical design, hardware technology, I/O equipment, displays, and test equipments or specifications.

Related documents are SP-8053, "Nuclear and Space Radiation Effects on Materials," June 1970 and SP-8054, "Space Radiation Protection," June 1970.

2. STATE OF THE ART

The state of the art of spaceborne digital computer systems has undergone a rapid development over the past decade and will continue to do so in the future. An onboard digital computer has now become essential for most new space vehicles. This section provides a brief description and appraisal of some of the design and flight experience of these systems, and the technology which presently exists.

To appraise the various design approaches to spaceborne computers, it is convenient to consider their common features. Table 1 summarizes the more important characteristics of selected digital computers which have been designed or are under development for space vehicle application. Although the data in table 1 was compiled from numerous sources, much of the information was drawn from references 1 and 2. Due primarily to the differences in requirements between launch vehicles and spacecraft, separate computer systems are usually designed for both the launch vehicle and spacecraft. The launch vehicle requires a large amount of complex high-speed computation, while the spacecraft has fewer computations over a long period of time.

2.1 Spaceborne Computer Functions

The functions or tasks of spaceborne computers have ranged from simple mission sequencing to complex multipurpose uses in manned spacecraft. With the expanding complexity of space missions, the number of tasks which are assigned to onboard digital computers is growing rapidly. In many cases, digital computers are taking over functions which were formerly performed by analog equipment. As time goes by, increasing experience is gained in sharing a computer or system of computers among numerous tasks. In spaceborne applications, most of the tasks involve real-time, sampled-data control problems, which place specific demands on, and yield certain advantages for, the computer system. Sections 2.1.1 through 2.1.6 summarize some of the most common onboard computer tasks.

TABLE 1.–Selected Spaceborne Computer Characteristics

Space Vehicle[a] (Computer Designation)	Physical Characteristics			
	Weight, kg (lb)	Size, m^3 (ft^3)	Power, W	Components[c]
ATLAS (ARMA MICRO)	9.1(20)	0.011(0.4)	50	Discrete
SATURN 1 (IBM ASC-15)	34.0(75)	0.06(2.12)	150	Discrete
TITAN 2/3-C (IBM ASC-15B)				Discrete
CENTAUR (LIBRASCOPE-3)	30.4(67)	0.052(1.83)	135	Discrete
GEMINI (IBM GDC)	26.7(59)	0.047(1.65)	85	Discrete
SABRE 2 (UNIVAC 1824-D)	38.5(85)	0.033(1.15)	260	DTL IC/ Discrete
SATURN 1B/5 (IBM LVDC)	36.2(80)	0.06(2.1)	138	DTL Hybrid
TITAN 3-C (UNIVAC 1824-MGC)	45.3(100)	0.057(2.0)	168	DTL IC/ Discrete
APOLLO CSM/LM (MIT/RAYTHEON AGC BLOCK I)	39.4(87)	0.028(1)	125	DCTL IC
MINUTEMAN 2 (AUTONETICS D-37C)				
X-15 (HONEYWELL HDC-801)	31.7(70)	0.04(1.4)	150	HLTTL IC
AGENA (HONEYWELL HDC-501)	11.3(25)	0.014(0.49)	92	DTL IC
APOLLO LM-AEA (TRW MARCO 4418)	14.7(32.5)	0.016(0.58)	91	DTL IC
APOLLO CSM/LM (MIT-RAYTHEON AGC BLOCK II)	26.3(58)	0.028(1)	100	DCTL IC
MOL (IBM SYSTEM/4Pi-EP)	75.6(167)	0.113(4.0)	763	TTL IC
GEMINI EXPERIMENT (CDC 449-1)	3.9(8.5)	0.002(0.08)	5	IC
APOLLO TELESCOPE MOUNT (IBM SYSTEM/4Pi-TC 1)	45.3(100)	0.07(2.5)	165	TTL IC
OAO-B (NASA/GSFC-WESTINGHOUSE OBP)	17.7(39)	0.015(0.52)	36	LPDTL IC
IMPROVED CENTAUR (TELEDYNE TDY-300)	26.8(59)	0.035(1.25)	185	Hybrid LSI
ADVANCED IMP (NASA/GSFC SDP-3)	1.8(4)	0.04(1.4)	2	LPDTL IC
MINUTEMAN 3 (HONEYWELL HDC-701)	22.6(50)	0.034(1.2)	270	IC
MARINER '71	10.2(22.5)	0.01(0.37)	22.5	IC/Discrete

TABLE 1.–(continued)

Space Vehicle[a] (Computer Designation)	Memory Features			
	Type[d]	Word Size, bits	Capacity, words	Access/Cycle Time, μsec
ATLAS (ARMA MICRO)	NDRO Core	22	2K-8K	/27
SATURN 1 (IBM ASC-15)	Drum	27	6886	
TITAN 2/3-C (IBM ASC-15B)	Drum	27	1160	
CENTAUR (LIBRASCOPE-3)	Drum	25	3K	
GEMINI (IBM GDC)	NDRO Core	39	4K	4/4
	Tape	13	85K	
SABRE 2 (UNIVAC 1824-D)	NDRO Film	48	4K	/4
	DRO Film	24	512	0.7/4
SATURN 1B/5 (IBM LVDC)	DRO Core	28	4K-32K	4/8
TITAN 3-C (UNIVAC 1824-MGC)	DRO Film	24	512	0.7/3.9
	NDRO Film	48	4K	0.7/3.9
APOLLO CSM/LM (MIT/RAYTHEON AGC BLOCK I)	DRO Core	16	1K	/11.7
	NDRO Core Rope	16	24K	/11.7
MINUTEMAN 2 (AUTONETICS D-37C)	Disc	27	6966	/78
X-15 (HONEYWELL HDC-801)	NDRO Core	24	4K-32K	1.0/2.0
	NDRO Plated Wire	24	4K-32K	1.0/2.0
AGENA (HONEYWELL HDC-501)	DRO Core	20	4K-8K	0.65/2
APOLLO LM-AEA (TRW MARCO 4418)	DRO Core	18	4K-8K	/5
APOLLO CSM/LM (MIT-RAYTHEON AGC BLOCK II)	DRO Core	16	2K	/11.7
	NDRO Core Rope	16	36K	/11.7
MOL (IBM SYSTEM/4Pi-EP)	DRO Core	32	16K-128K	0.9/2.5
GEMINI EXPERIMENT (CDC 449-1)	DRO Thin Film	24	256	4.0/
	NDRO Core	12	7680	4.0/
APOLLO TELESCOPE MOUNT (IBM SYSTEM/4Pi-TC 1)	DRO Core	8	8K-64K	0.9/2.5
OAO-B (NASA/GSFC-WESTINGHOUSE OBP)	DRO Core	18	4K-64K	0.85/2.5
IMPROVED CENTAUR (TELEDYNE TDY-300)	DRO Core	24	8K-16K	1.5/3.0
ADVANCED IMP (NASA/GSFC SDP-3)	DRO	16	4K-64K	5/10
MINUTEMAN 3 (HONEYWELL HDC-701)	NDRO Plated Wire	32	4K-16K	0.6/
MARINER '71	DRO Core	22	512	4/7

TABLE 1.–(continued)

Space Vehicle[a] (Computer Designation)	Arithmetic Features				
	Data Flow[e]	Data Type	Number of Instructions	Add/Multiply/ Divide Times, μsec	Index Registers
ATLAS (ARMA MICRO)	S	Fixed	19	27/135/324	
SATURN 1 (IBM ASC-15)	S	Fixed	39	156/1872/23K	0
TITAN 2/3-C (IBM ASC-15B)	S	Fixed	39	156/1872/23K	0
CENTAUR (LIBRASCOPE-3)	S	Fixed	12	625/4218/4218	0
GEMINI (IBM GDC)	S	Fixed	16	140/420/840	0
SABRE 2 (UNIVAC 1824-D)	P	Fixed	45	8/44-92/128	3
SATURN 1B/5 (IBM LVDC)	S	Fixed	18	82/328/656	0
TITAN 3-C (UNIVAC 1824-MGC)	P	Fixed	45	8/44-92/128	3
APOLLO CSM/LM (MIT/RAYTHEON AGC BLOCK I)	P	Fixed	11	23.4/117/210	0
MINUTEMAN 2 (AUTONETICS D-37C)	S	Fixed	57	~350/~1400/	0
X-15 (HONEYWELL HDC-801)	P	Fixed	89	4/12/30	6
AGENA (HONEYWELL HDC-501)	P	Fixed	59	4/24/24	4
APOLLO LM-AEA (TRW MARCO 4418)	P	Fixed	27	10/70/73	1
APOLLO CSM/LM (MIT-RAYTHEON AGC BLOCK II)	P	Fixed	34	23.4/46.8/81.9	0
MOL (IBM SYSTEM/4Pi-EP)	P	Fixed	72	5.8/9.5/18.3	16
GEMINI EXPERIMENT (CDC 449-1)	P	Fixed	38	28/604/	2
APOLLO TELESCOPE MOUNT (IBM SYSTEM/4Pi-TC 1)	P	Fixed	54	15/51/54	3
OAO-B (NASA/GSFC-WESTINGHOUSE OBP)	P	Fixed	50	6.25/42.5/90	1
IMPROVED CENTAUR (TELEDYNE TDY-300)	P	Fixed	33	6/22.5/40.5	3
ADVANCED IMP (NASA/GSFC SDP-3)	S	Fixed	54	78/ /	1
MINUTEMAN 3 (HONEYWELL HDC-701)	P	Fixed	56	2.4/10.8/21.4	3
MARINER '71	S	Fixed	16		1

TABLE 1.–(continued)

Space Vehicle[a] (Computer Designation)	Input/Output		Comments
	Channels	Inter-rupts	
ATLAS (ARMA MICRO)	2		Registers are delay lines. Two aperture core memory.
SATURN 1 (IBM ASC-15)	10		
TITAN 2/3-C (IBM ASC-15B)			25-bit data word; 9-bit instruction word.
CENTAUR (LIBRASCOPE-3)	32	0	Two-address instruction-operand & location of next instruction. Multiply & divide by series of additions.
GEMINI (IBM GDC)	14	0	13-bit instruction word; 26-bit data word. 13-bits of memory word are "read–only."
SABRE 2 (UNIVAC 1824-D)	8	2	24-bit data word; 16-bit instruction word.
SATURN 1B/5 (IBM LVDC)	25	7	26-bit data word; 13-bit instruction word. Memory expandable in 4K modules. Weight & size for 16K. TMR logic.
TITAN 3-C (UNIVAC 1824-MGC)	23	4	16-bit instruction word; 24-bit data word. Double precision add & subtract.
APOLLO CSM/LM (MIT/RAYTHEON AGC BLOCK I)	9	8	LM fixed memory capacity was 12K. Index instruction.
MINUTEMAN 2 (AUTONETICS D-37C)	14	1	3 parity bits per word; split data word available. Phase register may serve as limited index register. Tailored I/O.
X-15 (HONEYWELL HDC-801)	28	32	32K maximum total memory capacity. 4-character alpha-numeric data word.
AGENA (HONEYWELL HDC-501)	5	16	10- or 20-bit instruction word. Also designated ALERT.
APOLLO LM-AEA (TRW MARCO 4418)	1	1	Memory can be partially hardwired.

TABLE 1.–(continued)

Space Vehicle[a] (Computer Designation)	Input/Output		Comments
	Channels	Interrupts	
APOLLO CSM/LM (MIT-RAYTHEON AGC BLOCK II)	15	10	Index instruction. Double precision add.
MOL (IBM SYSTEM/4Pi-EP)	3	7	1-parity bit per 8-bit byte. Data word–2, 4, 8 bytes; instruction word–2, 4, 6 bytes. Floating point optional. Microprogram. Size, power, weight for 16K memory.
GEMINI EXPERIMENT (CDC 449-1)	1	1	Internal battery, keyboard, and dial readout. 12-bit parallel, 2-byte serial.
APOLLO TELESCOPE MOUNT (IBM SYSTEM/4Pi-TC 1)			2-byte serial; 8-bit parallel. Data word 2 bytes; instruction word 1, 2, 3 bytes. Size, weight & power for 16K memory.
OAO-B (NASA/GSFC-WESTINGHOUSE OBP)	9	16	Size, weight & power given for 4K memory. Scale register to locate binary point. Dual busses accommodate spare I/O, CPU, & memory units.
IMPROVED CENTAUR (TELEDYNE TDY-300)	36	5	Size, weight & power for 16K memory version and PCM telemetry master control capability.
ADVANCED IMP (NASA/GSFC SDP-3)	2	16	Size, weight & power estimates do not include memory. Double precision add & subtract.
MINUTEMAN 3 (HONEYWELL HDC-701)		8	16- or 32-bit instruction word. Microprogrammable.
MARINER '71	42	13	Parallel redundancy for maneuver. Weight contains sequencer. Interrupts include flight commands, telemetry system, and TV subsystems.

Notes: [a]Arranged in approximate chronological order by date of computer introduction.

[c]IC = Integrated circuit; DTL = diode–transistor logic; DCTL = direct–coupled transistor logic; TTI = transistor logic; HL = high level, as in HLTTL; LP = low power, as in LPDTL; LSI = large scale integration.

[d]DRO = destructive readout; NDRO = nondestructive readout.

[e]S = serial; P = parallel.

2.1.1 Guidance and Navigation

Digital computers were first placed on launch vehicles such as Atlas, Centaur, Minuteman and Titan to perform guidance and navigation (G&N) calculations. To accomplish this G&N task, the computer first processes data from various sources (e.g., inertial measurement units, star trackers, and horizon sensors) to estimate the present attitude, position, and velocity of the space vehicle. Using stored or computed trajectory data, it then determines any velocity corrections which are necessary and generates the appropriate steering commands, including thrust commencement and/or termination signals. Normally, the computation requirements for guidance and navigation do not by themselves dictate an excessively large or fast machine by present day standards. However, extreme accuracy and reliability demands can place stringent requirements on the spaceborne computer.

Guidance and navigation of a spacecraft in orbit involve calculations for such functions as 1) orbit parameters for use with experiment data, 2) steering commands for orbit maintenance, generation of ground tracks and display of latitude, longitude, and altitude, and 3) rendezvous guidance commands. Some missions require the capability of orbit changes in addition to orbit maintenance. In the Apollo guidance computer (AGC), the performance of all those G&N tasks requires some 24,000 words of memory and 26,000 operations per second (peak), out of the totals available for all calculations of 38,000 words and about 40,000 operations per second.

The Apollo spacecraft uses a stable platform for the primary inertial navigation system, whereas future vehicles may use strapdown systems which require additional calculations to update the inertial reference (e.g., see refs. 3 to 6). The memory requirement remains about the same as for a stable platform system, but the speed requirement depends heavily on the resolution desired and the rate at which the vehicle attitude is changing. For example, it is estimated that a system comprising three single-degree-of-freedom strapdown gyros and accelerometers in a vehicle rotating at 0.35 rad (20 deg)/sec would require approximately 120,000 operations per second to establish a resolution of 0.15 mrad ($\pm$30 sec of arc) (ref. 7).

2.1.2 Stabilization and Control

The purpose of a stabilization and control (S&C) system is to orient and maintain the space vehicle with a specified attitude and/or attitude rate. Conventionally, analog circuitry has been employed to provide the required control signals to the appropriate effectors (e.g., engine gimbal actuator, reaction jets, and control moment gyros). During the development of the Apollo spacecraft, however, it was decided to switch from an analog to a digital mechanization of the autopilot in the onboard computer (refs. 8 and 9). Digital S&C systems have since been designed and flown on a number of space vehicles, including both the Apollo command and lunar modules, Titan 3-C (ref. 10) and the improved Centaur (ref. 11).

Since the dominant natural frequencies of an S&C system are much higher than those of the G&N system, a digital S&C system has a significant effect on the speed requirements of the onboard computer. An example of the computer requirements for a reaction jet S&C system during

coasting flight is obtained from experience with the Apollo program; about 2,000 memory words and 15,000 operations per second are necessary to support the control system itself, and another 2,500 words provide associated functions such as star tracker calculations (ref. 7). The requirements for S&C during powered flight are even more demanding. For example, the Apollo lunar module digital autopilot uses about 3,500 memory words and on the order of 25,000 operations per second (ref. 12).

2.1.3 Man/Machine Interface

For manned space vehicles, online communication between the computer and the crew is required. Thus, the computer must possess the capability to request action or information, display real-time status and other data, and accept and execute manual commands. Because man is accustomed to alphabetic and decimal characters whereas computers operate with binary numbers, all information to be communicated must be converted to the proper form before it is intelligible. Several implications of the man/machine interface on the computer design are discussed in reference 13. The major effect on the computer design has been to increase both the memory and input/output requirements. Although the programs required to implement the man/machine interface are long, their duty cycle is low so they do not consume much of the computer's time (ref. 14).

The Apollo guidance computer uses a peak rate of about 3,600 operations per second to interpret an input or handle an output in less than 0.1 sec, which appears as virtually an instantaneous response to the operator (ref. 7). The interface between the crew and the AGC consists of a display and keyboard assembly (DSKY) which has function keys for verb, noun, clear, standby, keyboard release, enter, and reset; numeric keys for zero through nine; and plus and minus signs. Three two-digit electroluminescent (EL) displays are used to indicate the verb, noun and program number that the computer is currently using, and three five-digit-plus-sign EL displays are used to show input and output data. Discrete outputs turn on status lamps to indicate a variety of events including uplink activity, gimbal lock, and operator error.

The data entry and display assembly used for communication with the abort electronics assembly (AEA) computer in the Apollo lunar module is similar to the DSKY but less elaborate. It also comprises a keyboard and EL address and data displays (ref. 15). A push-button keyboard was used for manual data insertion to the Gemini computer, but EL devices were not considered because the circuitry required was too complex, and the display intensity would have been insufficient in bright sunlight. Instead, electromechanical decimal wheel devices were used (ref. 16).

No alphabetic input or output has been used with the Gemini or Apollo computers, although the use of symbolic coding would have reduced training burdens as well as simplifying control and interpretation problems. The main drawback in the use of alphabetic or symbolic coding at the time of the Apollo design was the lack of reliable, solid-state display translators. However, the convenience of alphabetic information will undoubtedly lead to the use of symbolic codes in future spaceborne computers. The application of symbolic coding for spacecraft computer control and display has been investigated at the NASA Ames Research Center (ARC) in a simulation environment with considerable success (ref. 17).

2.1.4 Data Processing

A large amount of data can be generated in flight by scientific experiments as well as by other spacecraft subsystems, such as guidance and navigation instrumentation. Depending upon the type of data and the extent of processing desired, the onboard computer speed and size requirements may be significantly altered. The engineering and scientific data generally have a high bandwidth and are available in real time only, corrupted by noise and not in a convenient form. To reduce this data to a manageable and comprehensible form, without losing the information content, it may be processed through an onboard computer. Such processing can include filtering, smoothing, estimating, averaging, compression, and formating for display or transmission.

Even for small scientific spacecraft, an onboard computer can be used as the master control for collecting data and formating it for telemetry. Since the power consumption of the computer is generally much lower than that of the telemetry transmitter, it is easy to justify substantial computation to achieve a moderate amount of data compression. For example, the use of a small central computer in place of the individual processors on the three IMP-F satellite experiments on Explorer 34 would have reduced the average power consumption by about 70%, without increasing the size or weight requirements (ref. 18). A general-purpose stored-program computer, the SDP-3, has been designed at the NASA Goddard Space Flight Center (GSFC) for use as the core of the data system of small scientific spacecraft and will be flown as an

TABLE 2.–Summary of Computer Requirements for Typical Experiments (ref. 24)

Functions	Memory Requirements (words)			Speed Requirements (Operations/sec)
	Data	Program	Total	
Checkout	200	10	210	17
Experiment Control				
Sequencing	2	12	14	10
Mode control	2	12	14	10
Rock spectra analysis	17 000	3 000	20 000	200
Pointing control	1 000	2 000	3 000	12
Solar flare sensing	9 000	1 000	10 000	70 000
Data Compression				
Simple predictor		20	20	500
Sophisticated interpolator (ESSI system)	1 000	150	1 150	1 400
Data Reduction				
Land SSE analysis (continuous)	900	100	1 000	50 000
Land SSE analysis (intermittent)	1 100	100	1 200	25 000
Magnetic field autocorrelation experiment	100	50	150	340
Plasma experiment	25	115	140	140

engineering experiment on IMP-I (ref. 19). As an example of onboard data processing requirements, the reduction of 238 pieces of raw data from the IMP-F magnetic field autocorrelation experiment to one average value and nine other statistical values is estimated to require 150 memory words and 340 operations per second. Calculating a sum, a sum of squares, and their ratio for the plasma statistics experiment on that spacecraft (ref. 20) would require about 140 memory words and 140 operations per second (ref. 7).

The computer can also serve as the hub for data transfer through the spacecraft, such as the IBM primary processor and data storage system (ref. 21) on the Orbiting Astronomical Observatory (OAO). A more sophisticated, general-purpose central computer, the NASA/GSFC–Westinghouse onboard processor (OBP), was scheduled to be flown on the OAO-B (refs. 22 and 23). Had the launch been successful, the OBP would have taken a central role in the spacecraft's operation and control as well as data handling.

Reference 24 discusses some of the many ways onboard computers may be used to enhance experimentation in space and indicates the wide range of computer requirements which might be encountered. Estimates of the computer memory and speed requirements for several individual experiments are summarized in table 2 (from ref. 24).

2.1.5 Systems Checkout

Systems checkout includes the monitoring, testing and diagnosis of spacecraft systems to detect, and if possible to predict, failures or unsafe conditions. For most spacecraft systems, this consists of comparing current values of system parameters to their nominal ranges. When a value falls outside its acceptable range, a message or signal is sent to the crew or to the ground. However, for some systems, passive monitoring is inadequate and periodic active tests are required. An example of systems checkout is the orbital checkout mode of the Saturn 5 launch vehicle digital computer (LVDC). When the vehicle has achieved orbit, the computer checks out the propulsion system, the mid-course guidance and control system, and other related systems and sends the test results to the ground for analysis. If all tests are satisfactory, the next mission phase may be initiated from the ground (ref. 25).

The checkout of experiments is particularly important on long duration missions since some experiments may not be turned on until a substantial time has elapsed, and others may be operated throughout the mission (ref. 24). The computer requirements for checkout of experiments or other spacecraft systems depend on the number of test points, the type of testing being performed, and the frequency of the checkout. Typical requirements for a sophisticated experiment with fifty scientific data points are shown in table 2 (ref. 24). The memory requirements are dominated by the storage needed to indicate the acceptable range for each checkout point.

Checkout is not restricted to inflight needs but may be performed during certain phases prior to launch. For example, the AGC is used during prelaunch to check out the entire Apollo G&N system. Among other tasks, the AGC determines all drifts and biases in the inertial reference subsystem that can be measured in a 1-g field. In addition, the AGC is used in a simulated flight mode to check out various other vehicle and spacecraft systems. The Saturn 5 LVDC also operates in a prelaunch checkout mode to insure that all guidance system interfaces operate properly prior

to flight. The checkout program includes a computer self-test, complete mission simulation, and a system test, among others (ref. 25). With the development of more flexible and powerful spaceborne computers in future programs, the use of an onboard computer for prelaunch checkout of spacecraft systems is expected to grow, and thus reduce the dependence on extensive ground-based facilities. In the advanced Centaur computer, for instance, prelaunch checkout was a major design requirement (ref. 11).

2.1.6 Computer Self-Test

A special case of systems checkout is the checkout of the computer itself, which generally requires the running of a self-test program. When an error is detected, its cause may be diagnosed and the error corrected or the computer reconfigured to minimize its effects. This task generally has a minor impact on the computer speed requirements since the self-test program is typically executed every one to two seconds, and is assigned a relatively low priority. However, additional hardware requirements for diagnosis and reconfiguration may be substantial and memory must be provided to store the checking routines.

Typically, a self-test program for checkout or restarting is a boot-strapping procedure which begins with the verification of the most elementary set of instructions, i.e., those which rely on only a fraction of the computer hardware in order to operate. These instructions are then used to construct a decision-making subroutine which verifies some primitive condition on a YES-NO basis. Once verified, this subroutine (or several similarly constructed) is used to check all other instructions and variations in sequence, beginning with the next least complex instruction and working up to the most complex instruction. After all instructions are verified, input/output (I/O) and memory self-test programs check the remaining hardware.

Self-test routines are also important for detecting malfunctions during operation. In the Gemini project, for example, diagnostic subroutines were interleaved in the operational computer program. When they detected a fault, a discrete command was issued to light a malfunction indicator lamp on the control panel. The circuit had a manual reset capability to test whether it was set by a transient malfunction. Three self checks were performed during flight (ref. 15):

- A timing check, based on the noncoincidence of certain signals within the computer under proper timing conditions.
- A thorough diagnostic test which exercised all of the computer's arithmetic operations during each computer cycle in all modes.
- A looping-check, to verify that the computer was following a normal program loop. A counter in the output processor was designed to overflow every 2.75 sec. Each program was written to reset this counter every 2.7 sec; thus, any change in the program flow would cause an overflow and indicate a malfunction.

In addition, a prelaunch mode check was included to verify the contents of memory syllable 2 by summing them. Since this syllable was "read-only" (after loading the memory by ground equipment), the sums could be checked against their known values to insure correct data.

The Apollo guidance computer is equipped with a restart feature comprising alarms to detect malfunction and a standard initiation sequence which leads back into the programs in progress. The AGC has six malfunction detection devices that cause a restart (ref. 26), as follows:

- A parity test of each word read from memory. An odd-parity bit is added to each fixed-memory word at manufacture time and to each erasable word at write time.
- A looping check much like the one on Gemini. A specified register must be periodically tested by any correctly operating program. This register is "wired" and if it is not tested often enough will cause restart.
- A transfer control trap, which detects endless loops containing only control transfer instructions, such as a location L which contains the instruction "transfer control to location L."
- An oscillator fail check caused by stopping of the timing oscillator.
- Voltage fail circuits to monitor the 28-, 14-, and 4-V power levels which drive the computer.
- An interrupt check, which detects excessive time spent in the interrupt mode, or too much time spent between interrupts.

2.2 System Design

This section provides a short discussion of the system design of spaceborne digital computers. It includes such considerations as organization, storage capacity, logic design, input/output, and anticipated future trends.

2.2.1 System Organization

(1) Central vs Dedicated Systems

There are two extreme approaches to the design of spaceborne computer systems. One approach favors the use of highly reliable, large central data processors to control and process several subsystems on the entire spacecraft. The opposing approach favors the development of small yet flexible, low power, dedicated processors where the ratio of processors to computational functions in any given spacecraft is equal to or greater than one. Large-scale integration (LSI) and complementary metal oxide semiconductor (MOS) technology are making the development of such small programmable processors possible (ref. 27).

There are two major advantages in using one or more small processors per function. One advantage is the ability to integrate subsystems with their computers at an early test stage, and to avoid interferences with other subsystems as might happen in a shared computer. Having its own processor permits the experiment freedom to be flown on different missions without the difficulty

of interfacing with each unique spacecraft processor, since only one experiment to processor interface need be designed. Although interfacing with telemetry cannot be avoided, this is much simpler than interfacing with a central processor. Reliability is another advantage since, with a collection of small decentralized processors, a failure may not jeopardize the entire mission of the spacecraft. The major disadvantages of the decentralized approach are that problems of information exchange and coordination are left unsolved, and error control is difficult to achieve.

The alternate concept of a wholly centralized computer solves many of the information exchange and coordination problems, but creates another in the number and diversity of its interfaces with all the sensors and effectors; and it poses a large challenge to fault-tolerant computer and system design (ref. 28).

A comparison of the two approaches, which was made for an example of an orbital mission to Mars in the early 1970s, is discussed in reference 29. The conclusions indicated that the anticipated functional requirements only lightly load the capacity of a general-purpose computer, and that the effect of economies in power, weight, and size for a centralized approach does not fully offset the inherent reliability of the decentralized concept. However, another study (ref. 18) found that a scientific spacecraft telemetry system utilizing a centralized computer is desirable over the decentralized processing approach, even for small satellites with only a few telemetry experiments.

A compromise between the two extremes which seems to combine their advantages is a distributed system, where information processing is done at various levels (ref. 28). The level at which a particular process takes place is almost entirely a function of the sophistication, reaction time, and bandwidth requirements associated with the process. A central computer is best suited to processes whose data-rate requirements are not severe, but whose sophistication is high in terms of program complexity and/or multiplicity of information sources and destinations. A dedicated computer is correspondingly appropriate for high bandwidth processes with limited complexity.

(2) Computer Configuration

Although there are innumerable ways of configuring a computer system, all will contain one or more of each of the following units, plus the communication and data paths required to interconnect them (ref. 30):

- Memory
- Processor (arithmetic and control)
- Input/output

A variety of system configurations have been designed, developed, and flown. Figure 1 shows the conventional single processor or uniprocessor system; the processor (P) is connected to the memory (M) and to the I/O controller (I/O), which may also be connected directly to the memory. The connections of the I/O controller to I/O devices, and the I/O devices themselves, are not shown in the figure.

Nearly all spaceborne computers which have been flown to date have been uniprocessor systems, either with or without the direct access channel between I/O and memory. These include the

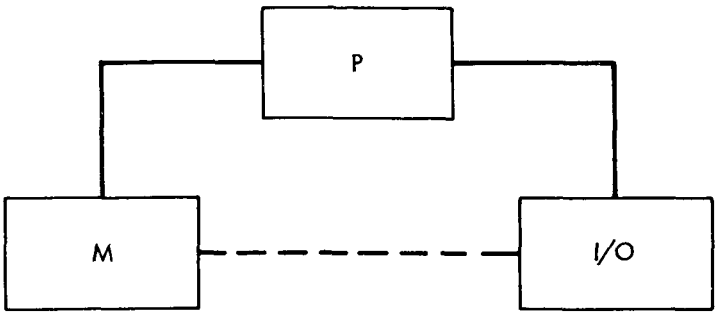


Figure 1.–Uniprocessor configuration.

computers on the Gemini spacecraft, the Apollo command and lunar modules, and the Atlas, Minuteman, Titan, Agena, Centaur, and Saturn launch vehicles.

A multicomputer system is shown in figure 2. The processors and memories are not necessarily identical since they are generally assigned different tasks to perform. This configuration is distinguished from a multiprocessor system in that the processors do not share memory. They may communicate directly or via a channel-to-channel adapter which makes each computer look like an I/O device to the other. The two systems may also share I/O devices, such as disk storage. The multicomputer configuration has been used in numerous ground based applications, such as the IBM direct-coupled 7094/7044, and in airborne systems, such as the F-111 Mark II Avionics System (ref. 31). During the latter phase of the X-15 program, a multicomputer system was flown on the No. 3 vehicle. A second computer was added to perform energy management calculations; it communicated with the Alert computer through a special I/O adapter. In the AGC design, provision was originally made for direct communication between the command module and lunar module guidance computers; however, this was never implemented. The Hamilton Standard Modular Flight Computer (ref. 32) is a breadboard multicomputer configuration which was originally designed for the Advanced Kickstage booster.

The general architecture for a symmetrical multiprocessor configuration is shown in figure 3. The distinguishing characteristic of the multiprocessor organization is the equal sharing of memory

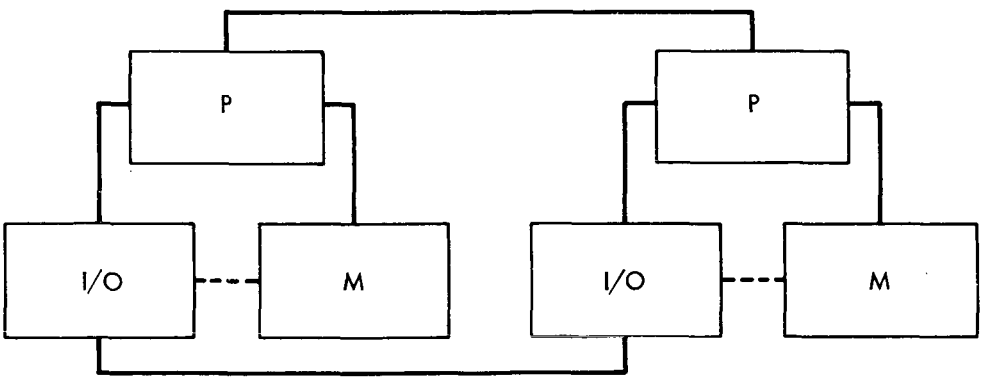


Figure 2.–Multicomputer configuration.

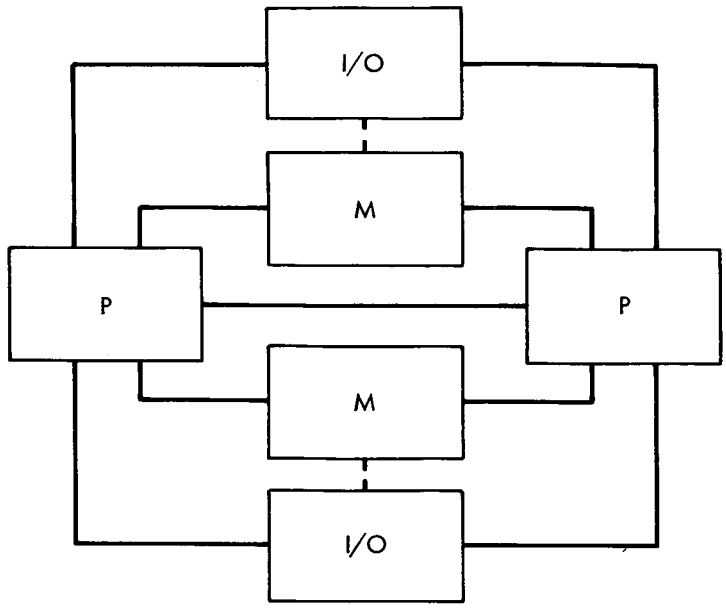


Figure 3.–Symmetrical processor configuration.

and I/O by each processor. Although the processors in this organization are often alike, they need not be. When they are, the operating system software usually treats them interchangeably, and tasks may be assigned to any available processor when they become ready for execution.

Multiprocessors have not as yet been flown in space vehicles although they have been used in ground-based and airborne installations (e.g., Control Data 6600 and 7600, UNIVAC 1108 and AN/UYK-7, IBM System 360/Model 65 Multiprocessor, and System/4 Pi VS/ANEW Multiprocessor). However, several multiprocessor configurations are in the design or development stages for spaceborne applications. These include the NASA Manned Spacecraft Center (MSC) Experimental Aerospace Multiprocessor—EXAM (refs. 33 to 35), the Massachusetts Institute of Technology (MIT) Instrumentation Laboratory Advanced Control, Guidance and Navigation Computer (refs. 28, 36 to 39), the NASA Marshall Space Flight Center (MSFC) Space Ultrareliable Modular Computer—SUMC (ref. 40), and the NASA/GSFC Parallel Ultra-Low-Power Processor—PULPP (ref. 27).

A survey of actual and proposed multiprocessor computer systems, multiprocessor theory, and problems related to the proposed space station and space base data management systems is contained in reference 30. Reference 41 describes an earlier study of the use of multiprocessing techniques for deep-space missions. Additional references to multiprocessors can be found in reference 42.

Figure 4 shows a closely related type of computer organization, the distributed processor, in which the logic elements are decentralized on an array basis. Each element has some memory associated with it, and its complexity can vary from the execution of a single instruction to a small computer. The execution of a program in the array can be centrally controlled. References 41, 43, and 44 discuss distributed processor designs which have been proposed for spaceborne applications.

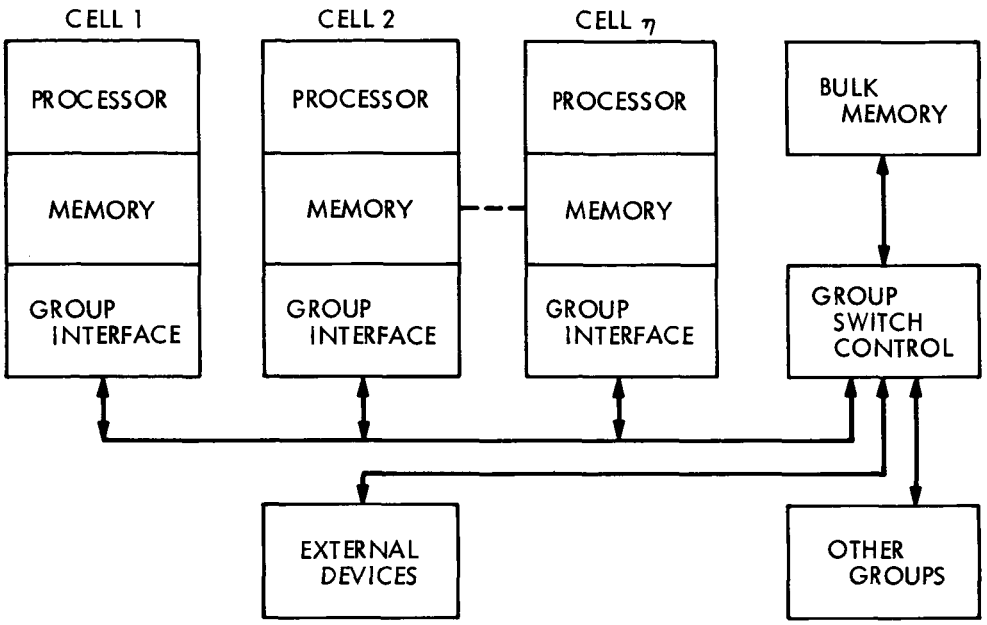


Figure 4.–Distributed processor configuration.

2.2.2 Memory

(1) Organization and Type

Spaceborne computer memories are functionally divided into program memory and variable memory sections. Sometimes the memories are physically separate and may even be constructed of different types of devices. In other cases, program and variable storage is incorporated in the same memory, as is generally the case with ground-based computers. The program memory is often either a fixed or a nondestructive read-out (NDRO) memory which contains the program instructions and constants. The variable memory is almost invariably a random-access read-write memory for temporary (erasable) storage of computational results. The size of the program memory is typically five to twenty times that of the variable memory, although the ratio will generally not exceed about ten.

Fixed or read-only memories usually have their contents manufactured into them; thus, changing the contents of a read-only memory requires physical modification of the device. The possible advantages of fixed memories lie in volume efficiency; retention of program following electrical malfunctions, power loss, or program error (sometimes shared by protected core memories); simplicity of construction and operation; and increased reliability (NDRO memories share some of these characteristics). Moreover, a fixed memory offers assurance that the computer program is identical through all phases of testing and in flight. On the other hand, a "read-only" memory requires that program and data be determined well in advance of use; consequently, it places a limitation on the computer's ability to accommodate changes in mission plan such as may be required periodically in most applications.

An example of a fixed memory is the core-rope program memory used in the Apollo guidance computer (ref. 14). This design achieved high density and reliability at a cost of having to procure new memory modules with about a four-week production cycle for program changes. Despite the fact that verification cycles were at least as long as production cycles, and that no launch postponements have yet been caused by this production cycle, the fixed memory has been a controversial issue owing to its potential for causing problems.

Read-write memories are classified as volatile or nonvolatile depending upon whether or not their contents remain intact when power is removed. Read-write memories may also be either destructive readout (DRO) or nondestructive readout. In general, NDRO memories enjoy two advantages over DRO memories. They are less vulnerable to power fluctuations or other interruptions of the read process, since at no time during the reading operation is the information cleared from the memory. Secondly, they can operate at higher speeds since it is not necessary to restore the contents following each read cycle.

As a compromise between reliability and flexibility, some read-write memories are designed so that their contents are electrically alterable only by special equipment; thus, in normal operation they behave as "read-only" devices. The Gemini guidance computer memory was an example. Each 39-bit memory word was divided into three 13-bit syllables; syllables 0 and 1 were multiaperture NDRO core read-write storage, while syllable 2 was "read-only" after being loaded by ground-support equipment.

In addition to their main memories, many newer computers have "scratchpad" memories, which are small, high-speed memories associated with the calculating circuitry. These memories give additional operating speed to a computer and seem particularly desirable for spaceborne computers since only small segments of the variable memory are likely to be used during any one program subroutine. The size of these memories may be determined by the mission requirements for operating speed since there is some penalty in power, weight, and complexity for scratchpad memory words vs the regular, random-access memory of the computer. These memories are especially popular in larger systems because of their increased operating speed and reduced number of instructions (since two-address systems are generally used). However, the saving in program storage is partially offset since the two-address instructions require a greater word length.

(2) Capacity

One of the most difficult parameters to specify, and one which has been a constant problem in space vehicle applications, is the memory size or capacity. Basically, memory capacity problems have arisen because the number of tasks assigned to the computer increased substantially as the spacecraft program progressed. Also, the complexity of certain tasks has sometimes been underestimated.

One of the earliest computers to experience memory capacity problems was the original Centaur machine (ref. 45). As the major storage medium, this computer used a magnetic drum, which included a timing track, instruction register, accumulator, multiplier/quotient register, as well as a main memory. In the original design (1959/1960), the main memory consisted of 37 tracks of permanent storage and 3 tracks of temporary storage, at 64 words per track. However, this was found to be inadequate, and in the redesign in 1961/1962, seven permanent tracks and one

temporary track were added to the drum. With clever programming, this 3072-word capacity has been adequate to implement G&N tasks, and the Centaur has successfully launched several Surveyor spacecraft to the moon, two Mariner spacecraft to Mars, and an OAO into earth orbit. However, to expand the Centaur's capability for future missions, a new computer was required and is presently in development (ref. 11). The improved Centaur computer will provide not only guidance and navigation, but also flight attitude control, vehicle sequencing, propellant utilization, and telemetry format control.

The Gemini program experience is another example of the problems associated with growing computer memory requirements (ref. 15). These were increased significantly when the ascent guidance mode was added to the computer in May 1962 to provide backup guidance capability for the Titan 2 launch vehicle. Another early addition was the requirement for an orbit navigation mode. These additions, plus numerous small features and improvements to existing programs which required additional memory capacity, occupied all the memory space by January 1963. The programming of the second operational "math" flow in June 1963 revealed that the memory capacity was exceeded by over 700 words, even after significant savings had been achieved by reprogramming. Three solutions were considered for this problem:

- Reduce programming requirements at the expense of accuracy and/or versatility. This approach could have been accomplished by across-the-board reductions in all modes, or by dropping one of the computer modes (rendezvous and orbit navigation were not required for the early spacecraft).
- Provide additional usable memory locations by hard wiring some elementary functions presently programmed. Such an approach would have gained 825 instruction locations.
- Provide an auxiliary tape memory external to the computer under the control of the astronauts. This unit would contain sufficient storage to handle present requirements, as well as any contemplated future requirements.

A combination of the first and third alternatives was eventually adopted. On Spacecraft 2 through 7, the orbit navigation mode was not included. On Spacecraft 8 through 12, an auxiliary tape memory was added to the spacecraft.

The development of the tape memory, begun in mid-1964, was a costly and time-consuming process which could have been avoided if the original design had provided for easy memory expansion. However, when completed, it provided an additional capacity for storage of 85,000 13-bit words and permitted a high degree of computer flexibility, as evidenced by the inclusion of nine operational computer modes in the last four spacecraft. In addition to alleviating the memory capacity problem, the auxiliary tape memory provided a memory reload capability for use in the event of an inflight memory alteration, such as occurred during the flight of Spacecraft 4.

A similar experience was encountered during the design of the Apollo guidance computer. Table 3 presents a chronological summary of the AGC memory growth from 1961 to 1965. Throughout much of this history, at least prior to 1964, the memory growth was due more to demonstration of the feasibility of a larger memory than to well-defined system requirements.

TABLE 3.–History of Apollo Guidance Computer Memory Growth

Date	Variable	Program	Remarks
1961	512	4K	Paper design based on inertial navigation requirements. Only executive and similar tasks programmed.
1962	1K	12K	Original mechanical design.
1963	1K	24K	Block I design. Guidance system programming started.
1964	2K	36K	Block II design. Redefinition of functional requirements; change to digital autopilot, improved crew displays and controls, and others. Greater speed also required.
1965	2K	36K	Mission programming started.

The Block I Apollo G&N system was conceived about 1962 to furnish position, velocity, and attitude control of the command module using inertial and optical sensor inputs. Outputs were transmitted to an analog S&C system responsible for the direct excitation of control actuators. To meet these requirements, plus a variety of less challenging ones such as telemetry, the Block I AGC design provided a 1K coincident-current core variable memory, a 12K core-rope fixed program memory, eleven instructions, and a 12-μsec memory cycle.

The Block I computer was used in support of three suborbital unmanned missions and was programmed, but not used, for a manned orbital mission. As these programs were being prepared, it was concluded that more memory and speed resources would be required for the lunar missions.

In 1964, the Apollo G&N system was revised to conform with the lunar module and revised command module designs. Design decisions made at that time are of some interest because of the knowledge gained during the Block I design and checkout. The Block II computer requirements were larger than those of Block I for several reasons. One reason was the decision to incorporate the autopilot function, which was formerly in the analog stabilization and control system, into the AGC. Another was the evolution of the computer's display into a major mission sequencing device. In addition, several new interfaces were defined whereby the computer could deliver data to spacecraft displays and receive data from spacecraft manual controls.

A 50% increase in the fixed memory capacity and a doubling of the variable memory capacity were made without any major impact on the associated circuit techniques. At this stage it was felt that all functions could be performed within these new memory capacities. Not surprisingly, the increased memory resources were committed before the programming was complete, and a large effort was expended to maximize the utilization of the available memory.

A final example of memory capacity growth is the Abort Electronics Assembly computer which provides backup guidance for the Apollo lunar module (ref. 16). In the original design, based on request for proposal (RFP) requirements, 500 18-bit words of memory were determined to be adequate for the required computations. At that time, there were no requirements for variable mission changes since the Abort Guidance System was essentially an attitude reference system. However, due to major increases in system functional requirements, the memory grew to 2,000 words, including 500 words of erasable storage. Additional studies soon revealed that the original

guidance scheme was not adequate, and that explicit guidance was required to provide rendezvous via parking orbit or direct ascent. To implement the revised guidance system functions, it became necessary to provide an I/O unit for use by the crew in communicating with the computer, and to increase the size of the memory to a total of 4,096 words, with 2,048 words of hard-wired "read-only" memory and 2,048 words of read-write data memory. A significant by-product of these memory and I/O changes was the required redesign of the AEA power supply.

2.2.3 Logic Design

(1.) Word Length

Word lengths of aerospace computers have varied from 9 to 52 bits with most being around 24 bits. The major considerations have to do with precision of calculations (with due attention to roundoff and truncation errors), addressing formats, input/output precision, economy of size, and programming convenience. The size, speed, and power consumption of the memory all increase with longer memory words. Shorter word lengths impose limitations on the operand addressing and precision of data representation.

Although approximately 80 to 90% of spacecraft data-processing computations can be performed with word lengths no longer than 15 to 16 bits, critical navigation quantities (such as the earth's radius) generally require 24 to 32 bit precision (ref. 46). When a shorter word length is used, for example the 15-bit word in the Apollo guidance computer, greater precision is obtained when required by multiple precision operations. In many applications, sufficient accuracy can be achieved by computing various terms in single precision and accumulating them in double precision.

The precision of data from analog-to-digital converters (generally from 6 to 12 bits) is often important when the word length is being specified. The minimum word length is usually selected to permit handling of I/O data in single precision. However, this consideration is usually academic since the lower bound is effectively placed by the format needed for instruction addressing (ref. 14). If the instruction format is too short, operand-addressing limitations are imposed, and increased use of indexed addressing accompanied by a performance loss usually results. For example, in the redesign of the Block I AGC, the original 15-bit word was not lengthened. The expanded operation code set and memory capacity were accommodated by such devices as bank registers, an extension operation, and the shared use of some bits for either operations or addresses. The motivation was to achieve commonality of ground equipment for the Block I and Block II versions, but this advantage was probably outweighed by the added programming difficulties which would have been alleviated if the word length had been increased by three or four bits.

Another addressing feature which affects the choice of word length is the use of single or multiple address instructions. In typical problems, a double address format is likely to save up to 30% in program space and 10% in execution time over a comparable single address format (ref. 47). A double address machine, however, is more costly to implement in terms of hardware than a single address machine. Triple address and list processing structures provide only marginal improvement in program size and execution speed over double address at considerable hardware cost.

The ability to address shorter words or parts of words may be desirable, even though the memory uses longer words in its access or selection scheme. Thus, the memory may read out two or more data or instruction words during a cycle, after which a selection circuit chooses from these words. For example, the Gemini computer utilized a 39-bit memory word consisting of three 13-bit syllables, each of which could contain an instruction or data. This approach can both increase the program operating speed and simplify the memory construction.

Several spaceborne computers have used multiple word lengths for instruction and data. The memory devices of these machines must use a word size which is some multiple or fraction of the program and/or data words. For example, the UNIVAC 1824 guidance computers used for Sabre 2 and the Titan 3 launch vehicle have a 16-bit instruction word and a 24-bit data word. The main NDRO memory word is 48 bits in length. Hence a memory word may contain either three 16-bit instructions, two 24-bit data words, or one operand and one instruction. Each DRO memory word is 24 bits in length and may represent either one operand or one instruction (refs. 48 and 49).

The goals of minimizing power, weight, volume, and cost are leading manufacturers to shorter word length machines in order to reduce hardware. This may be accomplished successfully by increasing the speed of operation sufficiently to accommodate double precision arithmetic (where needed) and by developing a computer architecture which can be programmed efficiently with short instructions (e.g., a small address field). The word length trend seems to be either toward 16 bits, following the commercial minicomputers, or toward 32 bits with 1/2 and 1/4 word addressing.

(2) Instruction Repertoire

Early spaceborne computers had very limited instruction repertoires, with several computers having sixteen or fewer instructions. More recent computers have much larger instruction repertoires, ranging to over 100 instructions for some developmental machines, with most having around 60. Early computers all provided conventional arithmetic operations for binary fractional numbers. Addition, subtraction, and multiplication were offered on nearly all machines although division was sometimes programmed. The original Centaur computer was an exception since both multiplication and division were performed as series of additions or subtractions.

All machines to date have included basic instructions such as load, store, logic and arithmetic, I/O instructions, conditional and unconditional branch instructions, and shift instructions for scaling. Current spaceborne computers go well beyond this, generally to obtain more speed at a cost in hardware. In many machines, the instruction repertoire features certain instructions that are "tailored" to perform functions or sequences of operations that frequently occur in specific spaceborne applications. For example, a multiplication is often followed by the addition of the product to a prior accumulation. This sequence occurs in vector multiplication, matrix manipulations, and in numerous algebraic equations. A "multiply and accumulate" instruction to perform this sequence of operations will provide a saving in program space as well as execution time. Another example is to combine with instructions a commonly occurring function, such as decision making based on the result of the instruction. Since the tailored instructions invariably have control sequences that are almost identical to sequences that must be generated to achieve the standard instruction complement, they require only little additional hardware (ref. 47). For ex-

ample, the Apollo Block I computer repertoire consisted of only eleven regular machine instructions, which burdened both speed and storage. The Block II redesign had 34 instructions, which were implemented with only a modest hardware increase.

For extended precision data handling, certain instructions utilizing overflow bits are convenient. Because multiple words are used primarily to accumulate small increments or to sum products during polynomial evaluations, nearly all multiple precision computation can be performed directly by load, store, add, and subtract double-precision instructions. Long data-format shift instructions reduce the number of double-precision products that may be required (ref. 46).

Hardware floating point operations have not yet been utilized in spaceborne applications although programmed floating point routines have been made more convenient by inclusion of instructions such as "shift left to most significant bit." There is, however, a strong tendency to provide developmental computers with at least an option for floating point instructions to simplify flight software development.

The introduction of scratchpad memories in the newer computers leads to two-address instruction words where one address, or both, refer to the scratchpad memory. The use of small scratchpad memories generally results in a shorter instruction word and thus economy of storage along with a higher rate of instruction execution. Computers which make extensive use of scratchpad memories can effectively and efficiently use a combined half and full word instruction repertoire. If the repertoire is well chosen, such systems can reduce the amount of storage required for the instruction portion of programs to almost half (ref. 50). This savings is significant since memory is the most costly hardware item in the system, and the memory reduction can be an important weight saving consideration.

The need for separate index registers can be eliminated by the use of scratchpad memory registers as either index registers or base registers in instruction word formats. Addressing can use base registers with such options as direct indexed, indirect indexed, relative branches, and relative operands. The latter two types of instruction addressing are found in most smaller machines, while the first two are common only to the larger machines.

Although a short (e.g., 16-bit) word provides stringent constraints in the design of an instruction format, careful study reveals a number of attractive possibilities which lead to a powerful machine organization. Page addressing, which becomes necessary because of the small address field available, is much less burdensome if it is done in such a way that the page moves along with the program counter; thereby identifying memory locations relative to the program step being processed. The use of multiple accumulators which can serve as address pointers as well as data registers provides a freedom of memory accessing that is impossible with instruction word memory address fields alone. Such an addressing scheme, combined with appropriate instruction mechanization, can achieve pushdown stack operation with the pointer registers. The concept also frees bits in the instruction word, allowing multiple register addressing for interregister operation. Interregister instructions decrease memory size as well as access time.

A revived development in the spaceborne computer field is the use of microprogramming (ref. 51). Although microprogrammed computers have not been flown yet, the technique provides a flexibility not customarily available in conventional computers by allowing modification of such

processor parameters as the instruction set and operand length. Microprogramming provides a means of including macro instructions which are application-oriented operations of considerable complexity, such as root and trigonometric functions, vector/matrix operations, and coordinate conversions. Furthermore, microprogramming allows instruction-set modification much later in the design cycle than conventional machines, to permit optimization of the computer for an evolving set of requirements.

(3) Speed

The speed of a computer can be thought of as the time required to perform an assigned task (ref. 52). However, since the tasks vary widely, specific numbers such as instruction times and memory access or cycle times are normally used to indicate the speed of the computer. Speed is important in spaceborne computers, and logical complexities are employed in order to gain speed in virtually every design; but size and reliability restrictions are of sufficient importance to limit the number and extent of such complexities.

The speed of memory operation is generally specified in terms of the access time and/or the cycle time. Faster memories need additional power for selection circuitry and have lowered signal-to-noise ratios for read signals, while slower memories generally have simpler and less critical circuits, less power dissipation, and hence potentially more reliability. For DRO core or film memories, the cycle time is generally 2 to 3 times the access time, whereas for NDRO or fixed memories, the cycle time is only about 1-1/2 to 2 times the access time. As shown in figure 5, storage cycle times for aerospace computers have ranged from 0.6 to 27 μsec, with most times lying from 2 to 6 μsec (ref. 1). The advent of smaller cores and plated wire memories is yielding cycle times of a microsecond or less for more recent computers. In addition to plated wire, MOS and bipolar LSI show promise, particularly for small memories. Since the latter are NDRO memories, the write time, which can be from tens to hundreds of nanoseconds, is often quoted as well as the read or access time.

Early spaceborne computers (not including those with drum memories) performed such basic operations as addition and subtraction in tens of microseconds and multiplication and division in hundreds of microseconds. Present day computers have addition times on the order of 2 to 6 μsec and multiplication and division times in the tens of microseconds. Figure 5 also compares the add and multiply times of a number of aerospace computers introduced or under development through mid 1970 (ref. 1). Some computers of recent design have addition and subtraction times of 2 μsec or less, multiplication times of less than 10 μsec, and division times of less than 20 μsec. When comparing execution speeds for various computers, one must consider whether the arithmetic operation times given are for single-address or double-address instructions, whether or not memory banks are alternated, and whether the numbers given are average times as opposed to minimum times.

To assist in the comparison of speeds of various computers, several organizations (e.g., NASA and the Aerospace Corporation) have devised sample programs or tasks for which operation times may be calculated on either real or hypothetical machines. Generally, these sample programs consist of a given series of operations such as matrix inversion, or a typical guidance or navigation problem. The real or hypothetical computer is then programmed to perform this task, and the computer's speed is determined by 1) operation of the machine itself, 2) simulation on another

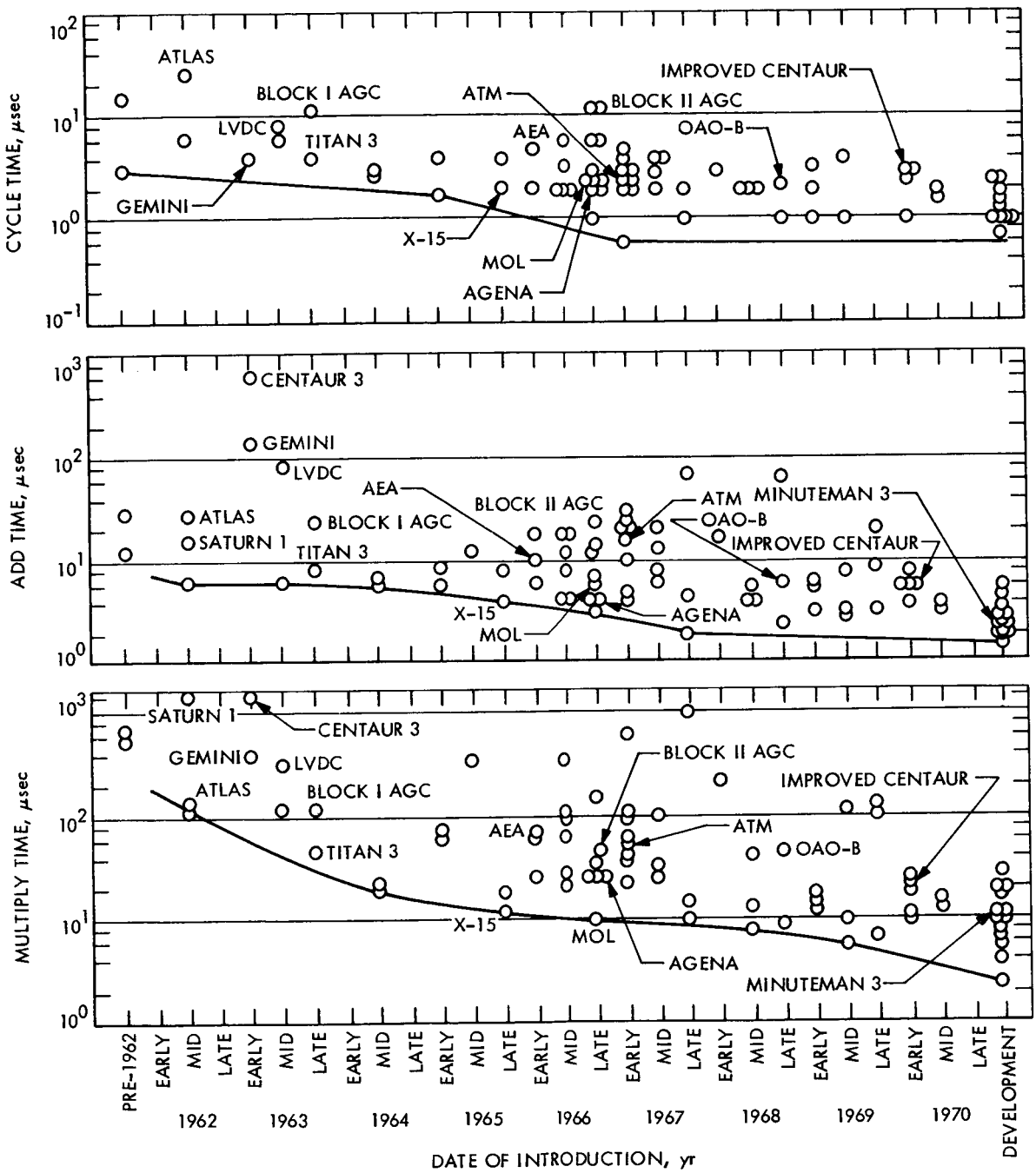


Figure 5.–Chronological comparison of aerospace computer memory cycle times, add times, and multiply times.

computer in which the times for each of the simulated instructions are accumulated as they are executed, or 3) manual calculation. Sample problems to date have been most pertinent and useful for specifying and evaluating computers when many arithmetic operations with numbers of widely varied precision are required. They have not been as appropriate when large amounts of data handling are required.

Speed problems with spaceborne computers have been minor and generally resulted from attempts to minimize ground-based precomputation. However, even minor speed problems are costly and can be expected as long as storage is critical. In the Apollo guidance computer, a speed increase was implemented by the addition of instructions, including a double-precision addition operation and a variety of single- and double-precision data-handling operations. However, the increase was limited by the memory addressing scheme devised to overcome word length restrictions.

The most dramatic example of speed problems in a spaceborne computer was the overload that nearly created an abort during the final moments of the Apollo 11 spacecraft's descent to lunar landing (ref. 53). At least four program alarms were triggered by a work overload, which was produced by the last-minute decision to operate the rendezvous radar in the auto-tracking mode during descent. Because of the necessity of responding to interrogations of the radar, the computer rejected some jobs of lower priority, and simultaneously flashed a warning light to the crew that its speed was being taxed beyond capacity.

(4) Data Flow

Whereas most earlier machines transmitted and processed data serially, practically all recent spaceborne computers have used parallel data flow to achieve higher speed. Two exceptions to this trend are the Saturn 5 LVDC and the OAO-B onboard processor, which illustrate the ability of serial machines to successfully handle complex tasks and, potentially, provide hardware savings and reduced complexity. A compromise serial-parallel configuration, which will be used on the Apollo telescope mount (ATM) computer, requires less hardware than is needed for parallel processing while it provides higher speed than a serial processor. In this approach, the data words are divided into serial "bytes," each of which is processed in parallel. Parallel processing probably will continue to dominate future designs although serial machines may still be used in special applications which do not demand high speeds but do require extreme simplicity for greater reliability (ref. 52).

(5) Number Representation

All spaceborne computers operate with the binary system; i.e., numbers are expressed to base two. However, for compactness and the convenience of programmers, the binary words are frequently represented in octal (base eight) or hexadecimal (base sixteen) by grouping the binary bits into bytes of three or four, respectively.

Fixed-point representation has been used almost exclusively to date. The number range provided has been adequate, with suitable scaling, for most aerospace applications, so that the additional hardware required for floating-point representation has not been considered justified. In those

cases where the range of certain variables has made it necessary to store numbers in floating-point form, subroutines have been used to perform arithmetic calculations.

The scaling of fixed-point variables imposes an extra burden on programmers, and also complicates program validation. However, the scaling process has been facilitated by the fact that the values through which variables may range are generally constrained by the details of the mission and can be predicted quite accurately. Scaling is also simplified by means of simulations on ground-based computers with floating-point representation; the simulation results are easily examined to determine the excursions of all variables under anticipated conditions. Contingency code is still required for critical programs to provide for inadequate scaling and the subsequent occurrence of an overflow. For high precision G&N calculations, which comprise about 30% of all AGC programs, the lack of floating-point capability requires the generation of approximately 35% more code, and at least a twofold increase in the programming and verification workload (ref. 12). The complex scaling used in the AGC is difficult to understand, requires excessive time and training to implement, and produces coding that is difficult to modify and/or verify. For example, the scaling strategy was responsible for a rendezvous convergence problem encountered during Apollo verification.

The introduction of medium and large scale integration techniques has made it possible at low cost for newer aerospace computers to offer floating-point number representation and floating-point operations wired in as instructions (e.g., IBM System/4 Pi-EP, UNIVAC 1832, and CDC ALPHA). The additional equipment and instructions necessary for floating-point representation can be justified on the grounds of 1) ease of programming and 2) faster operating speed when many operations are performed on numbers which range through wide intervals and when precision is required. This can only be determined by the details or requirements for a particular mission, but, in the future, floating-point representation will undoubtedly become more common as the hardware becomes smaller, lighter, cheaper, and more reliable.

Three possible ways for representing positive and negative fixed-point binary numbers are available to the logic designer: sign and magnitude, one's complement, and two's complement (ref. 14). Sign and magnitude representation, which is convenient when direct human interrogation of memory is desired, has not seen application in spaceborne computers. Two's-complement representation has been used in practically every spaceborne computer, with the notable exception of the Apollo guidance computer. This representation provides a unique representation for zero, does not require recomplementation, and simplifies multiple precision arithmetic.

For fixed-point arithmetic, the binary point can be positioned either between the sign and the high order magnitude bit (fractional representation) or to the right of the low order magnitude bit (whole number or integer representation). The former approach has been used in all spaceborne computers to date.

Occasionally, communications may require alphanumeric characters, and provision for 6- and 7-bit codes and their handling may be necessary. Binary-coded-decimal (BCD) number representation systems have not been popular for spaceborne applications due to the poor economy in number representation, and it is not foreseen that much use of the BCD technique will be made, except perhaps in conjunction with alphabetic or alphanumeric codes in future manned flights.

2.2.4 Input/Output Features

Input/output (I/O) features of spaceborne computers are extremely important since these machines act as real-time control elements. Due partly to the nondigital nature of such electromechanical machines as inertial measurement units and rocket steering servos, and partly to the strong desire to minimize interface circuits and cables, spaceborne computers spend a substantial amount of their time (or equipment) budgets on maintaining communication with these units. Furthermore, the central processor is usually time shared among several real-time jobs, and delays in responding to I/O requests often must be limited to fractions of a second.

(1) Channels and Interrupts

Several basic techniques exist for introducing input data into the computer and delivering output data from it. Program interrupts are useful for handling input data which might be lost if immediate servicing is not performed, or for requesting an immediate response from the computer. The interrupts are generally forcing, so that the program services the request upon demand. A priority schedule is required when simultaneous interrupts are possible on two or more channels. Interrupts from input data are most commonly used to interface asynchronous peripheral equipment and to notify the computer of the occurrence of external events such as operator console actions, ground-initiated communication signals, and certain sensor outputs, such as from alarm devices. Interrupts of various types share one important drawback; they cause discontinuities in instruction and data flow at times determined by external events. It, therefore, becomes necessary to make programs immune to such discontinuities and, equally important and perhaps more difficult, to verify this immunity for any possible event timing.

For most data, however, forced interrupts are undesirable although their use sometimes results in simplification of programming. Information which does not require immediate servicing, such as guidance and navigation data, is commonly accepted from and delivered to fixed buffer registers. An I/O buffer for storage of data which can be serviced during normal program operation is often very useful, particularly during mission phases wherein the computation load is high and servicing of the forced interrupts is of primary concern. The buffered nonobligatory interrupt feature sometimes increases efficiency in program operation since nonforcing data can be processed when convenient and at a rate suitable to the task being performed.

Another feature which can save program time consists of "cycle-stealing," wherein data is loaded directly into a specified part of the memory, without any program interruption. This characteristic facilitates program organization and efficiency and is of particular value in large systems.

Depending upon their urgency and the computer facility, discrete signals—including data from consoles, communication links, overload relays, and other on-off sensor devices—can be accommodated by forced interrupts, loaded into buffers or introduced directly into the memory by cycle stealing. In the past, discrete signals have generally been handled with program subroutines which store and read individual bits within words. However, bit manipulation instructions have been incorporated in some developmental computers to facilitate discrete signal handling.

(2) Interfaces

The interface provides the means for transfer of data between the computer and all peripheral devices, including whatever signal conversion is required. Typical functions are analog-to-digital (A/D) and digital-to-analog (D/A) conversion, I/O buffering, multiplexing, and signal level shifting. In some cases, input data are not in the conventional fractional binary system, and must be converted before they can be used. For example, incremental inputs must be scaled before operations can be performed. When additional conversion is required, it may be performed either by program subroutines or by external circuitry—the choice depending upon the frequency and complexity of the operation and the amount of hardware required.

The interface between a spaceborne computer and its associated I/O subsystems is a critical item which has a substantial effect on the overall design. Frequently, the interface conversion units, multiplexers, and data buffers occupy as much volume and consume as much power as the remainder of the digital computer system (ref. 54). To date, interfaces tend to be of a custom design and are generally quite complicated. The launch vehicle data adapter (LVDA), which serves as the interface for the Saturn 5-LVDC, transforms input and output signals, controls data flow, performs some simple computational and logical operations on data, and provides temporary storage of data (ref. 25). Packaged separately from the LVDC, the LVDA is almost twice as large, weighs nearly three times as much, and consumes more than twice as much power.

Because of the varied types of I/O devices which are used with spaceborne computers, interfaces have presented a recurring problem in design and systems integration. Experience in past designs has underlined the importance of (1) coordination between the manufacturers of sensor devices, communication links, consoles, A/D converters and the computer itself, and (2) early and clear specification of the interfaces between the computer and its auxiliary devices. An awareness of this problem of compatibility of the interfaces between subsystems, and concern for its resolution, may eventually lead to the establishment of standardized I/O requirements for spaceborne computers (ref. 53). Development of the common data bus concept is expected to hasten the introduction of a truly standardized interface.

2.3 Physical Characteristics

2.3.1 General

Spaceborne computers are usually designed to satisfy rather severe physical constraints. These are discussed below.

(1) Weight

Weight is related to size by the densities of materials used and the spatial efficiency of the structure. In the past, good packaging practice has led to densities in the neighborhood of the density of water (one gram per cubic centimeter). Early aerospace computers with 8K memories had masses on the order of 40 kg (88 lb), while equivalent present day machines have masses on the order of 20 kg (44 lb) or less.

(2) Size

Memory size, memory speed and processor operating speed are important factors in size since faster circuits, high-speed memories, and parallel operations all tend to lead to increased size and power dissipation. Low speed requirements may permit more efficient, serial or series-parallel processing and machine organization. Extremely small processors of the order of a few hundred cubic centimeters in volume, limited in throughput capability, can be made using large scale integrated-circuit packaging techniques.

Size reductions foreseen by the progress of the art of microminiaturization have been limited by problems in the areas of heat transfer, interconnections (including the availability of test points), ability to replace subsections, and power conversion. Consequently, the greatest overall economy is not necessarily achieved by designing to the minimum possible volume but rather requires tradeoffs in volume vs operating features.

(3) Power Consumption

Except for certain special low power logic systems, the advent of integrated circuitry, MSI and LSI, has not produced dramatic reductions in total power consumption. However, the power efficiency has been substantially improved since more computation can be performed for a given amount of power. Low power consumption may enhance reliability since, for instance, some semiconductor failure mechanisms are strongly temperature dependent. Since the memory often consumes the bulk of the power, it is a major factor in power requirements.

Another important consideration is speed since faster circuits nearly always require larger currents, and, in particular, faster core or film memories normally require larger drive currents. Therefore, a faster computer with otherwise the same operating characteristics generally requires more power. Occasionally, the power distribution to spaceborne computers is controlled during different phases of the flight. For example, the Apollo lunar module computers are powered down during most of the flight to the moon. For these applications, fixed memories or nonvolatile NDRO memories are usually desirable since they retain their contents when the power is turned off. The computer may not be switched off entirely in some cases but "idled" at low standby power; periodically, it will awaken itself to perform tasks such as midcourse-guidance calculations. A few special circuits have been developed which consume very little or no power in the standby condition. The newer circuit technologies such as low power bipolar logic and metal–oxide–semiconductor–field–effect–transistors (MOSFETs)—particularly complementary MOSFETs—consume less power than the bipolar circuitry which is most commonly used in spaceborne computers today.

In the interest of reliability, power supplies are often duplicated, as in the Saturn LVDC, and provision is made for switching in spare power supplies when those in use fail during a mission. Also, protective circuitry is often included to protect against power surges caused by certain types of power supply failures. Computers with a restart capability incorporate an alarm to warn of power supply failure and a temporary power source to allow retention of system status. However, the advantages of these protective features are gained at the expense of increased weight and volume as well as the added complexity.

2.3.2 Hardware

(1) Memory Devices

Most early spaceborne computers (e.g., Centaur, Titan 2 and 3, Minuteman 1 and 2, Saturn 1) used electromechanical drums or discs as memory devices. However, these are disappearing from space application for several reasons (ref. 14): a substantial increase in packaging density of core and film memories, the serial access nature of discs and drums, and the limited time that discs and drums can operate without maintenance. Consequently, in recent years, the ferrite coincident–current core memory has been the cornerstone of computer technology, providing fast random access at a few cents per bit in the megabit range. DRO cores used for program memory suffer the disadvantage of requiring a write cycle following each read to restore the information. More sophisticated geometries, such as the multiaperture devices (as used on the Gemini computer) or BIAX, can be used to obtain NDRO operation but at a cost in weight and volume (refs. 55 and 56). More often, NDRO memories are implemented with thin-film techniques, wired arrays, or semiconductors.

Thin-film memories have received a large research investment and significant advances have been made in their speed and bit density. However, they have not gained wide popularity, since their cost is relatively high and their capacity is more limited than core (ref. 14). The "Bicore" thin magnetic film element has been successfully used in the Titan 3-C computer and a similar device, "Quadralloy," is being used in the Phoenix missile guidance computer (ref. 55). These devices can be electrically altered, but the write time for NDRO operation is typically 1,000 times the read time. Therefore, NDRO information is usually electrically entered by external equipment and no provision is made for real-time write operations. However, DRO thin-film memories can be altered in the same time as read time. For example, the DRO-mated film memory used in the UNIVAC 1832 computer has a 750 nanosec read-write time (ref. 57).

If nonvolatility is a requirement, plated wire (refs. 55, 56, 58 to 62) is generally considered a good candidate for the next generation mainframe and mass memories. Plated wire memories can be constructed for either DRO or NDRO operation. Speed, low power, moderate density, and automated production give this device the potential advantage of use in all memory applications of future spaceborne computers. NDRO plated wire is being used for the memory of the Honeywell HDC-701 computer for Minuteman 3.

Two types of wired arrays have been used in which the information is physically contained in the memory. Missing core memories are constructed by removing or shorting specific cores. In core-rope memories, such as the Apollo guidance computer uses (ref. 14), magnetic cores are either threaded or bypassed by word lines in a manner that permits storing one or more entire words in one core. The resulting bit density is extremely high: approximately 100 bits per cubic centimeter (about 1500 bits per cubic inch) including all electronics, interconnections, and packaging hardware.

Recent developments in LSI have made the semiconductor memory extremely attractive and practical, particularly for scratchpad or high-speed control applications (refs. 55 and 56). These memories are generally constructed of MOS or bipolar devices and provide NDRO operation, but

they are volatile. However, several manufacturers are presently developing nonvolatile semiconductor memories. Two potential techniques are magnetic cylindrical domain "bubble" devices (ref. 63) and the somewhat controversial Ovonics devices based on amorphous material technology (ref. 64). There is currently a definite trend toward the use of semiconductor memories, and they may become the major memory technology by the next decade (refs. 52, 55, and 62).

Plated wire and semiconductor memories are expected to be the most suitable technologies for mass data storage within the next decade (ref. 62). Serial memories using ferroacoustic (ref. 65) and magnetic domain device (ref. 66) technologies may also provide nonmechanical alternatives to drums and tape by the mid 1970s.

(2) Logic Devices

Since about 1963, bipolar silicon integrated circuits, or microcircuits, have been adopted nearly universally by spaceborne computer designers for at least the logic portion of the machines (refs. 14 and 52). Prior to the advent of microcircuits, magnetic cores and all-transistor circuitry were both strong contenders as logic elements. Core circuits were no smaller, but they were capable of operating on substantially lower power. Although special applications may favor the magnetic core, the small size, high speed, and reliability of microcircuits make them preferable to cores in nearly all instances. Moreover, the gradual reductions in power consumption of new microcircuit logic devices has enabled them to consume less power at full speed than cores. However, cores still have the advantage of reduced power consumption at low speed operation (ref. 14), an advantage which complementary MOS circuits share.

Within the last few years, spaceborne computer designers have begun to utilize LSI techniques to reduce volume and weight (e.g., ref. 67). Moreover, MOS technology is now being used to design central processing units (e.g., the Autonetics D-200), and it is likely that power, volume, and weight will continue to decrease as MOS is used more widely (ref. 52). However, its susceptibility to radiation effects may limit MOS use in some critical and long-duration missions.

2.3.3 Packaging

The packaging design of a spaceborne computer is usually the result of a careful tradeoff among the requirements for cooling, reliability, volume, weight, mechanical integrity, cost, ease of maintenance, accessibility of test points, and interconnection schemes. The latter factor is extremely important since the number and type of interconnections have a strong effect on reliability and since the interconnection provisions take up a major part of the volume in the computer. For example, approximately 75% of the volume of the Apollo guidance computer is required for interconnection wiring (ref. 14).

Spaceborne computers have often been physically divided into modules corresponding to the organizational subsystems of the computer (i.e., logic, memory, power supplies, and input/output), for at least two reasons:

(1) Different packaging techniques can be used for each section.

(2) An individual subsystem (e.g., memory) can easily be replaced or revised without redesigning the whole package.

The number of packages for each subsystem is the result of a tradeoff since smaller modules reduce the cost of each throw-away unit and may allow for some module standardization but increase the interconnection problem.

In most early computers, the logic section was usually constructed of cordwood modules of discrete components mounted on single-layer, plug-in printed circuit boards. These techniques are now obsolete, having been replaced by the use of some form of integrated circuit. Typically, a number of these microcircuit flatpacks are mounted in a module, in one or more layers, and interconnected by a multilayer printed circuit board. These modules in turn plug into or are mounted on a header or main frame which is the basic structural element of the computer. The wiring in the main frame may be wire wrap, multilayer printed circuit, or wire and solder (ref. 68). In some small developmental computers using LSI circuitry, the entire logic subsystem may consist of only a few flatpacks, which are attached directly to the main header, thus omitting the intermediate module.

Few general statements can be made about packaging the memory subsection of the computer since there is such a variety of competing memory techniques—magnetic core, plated wire, semiconductors, drums, tapes, and many others in the development stage. Each of these techniques has its own special problems and characteristics which must be considered at the time of packaging and in the light of mission requirements.

Discrete components still require packaging in the power supply subsection primarily because of the high power levels and large physical sizes involved. Discrete component circuits are also used in the I/O section because some functions are not conveniently adapted to microcircuits. There is a tendency in some recent designs, however, to remove these special I/O functions from the computer in order to standardize the I/O interface and encourage off-the-shelf use. If this is done, the packaging for the part of the I/O section within the computer is governed by the same considerations as for the logic section.

The spaceborne computer is typically housed in a substantial case made of lightweight metal such as aluminum or magnesium, which is both mechanically and electromagnetically sealed. Many early designs and some recent computers that are intended for use on the smaller unmanned spacecraft have odd shapes to fit in the limited space available in the vehicle. Other computers, designed for larger vehicles, have generally rectangular shapes, which are usually more convenient to package and easier to seal properly.

2.4 Environmental Design Factors

A major factor in the design of a spaceborne computer is the precautions that must be taken to insure correct operation in the environment to which the computer will be subjected, both in space and during testing and checkout on the ground. This section discusses several aspects of this environment.

2.4.1 Thermal Aspects

Most spaceborne computers have been cooled by conduction either to a liquid-cooled cold plate (e.g., Gemini and Apollo) or to a heat sink which radiates directly to space. The requirement to interface with the cold plate or heat sink imposes some mechanical constraints on the design of the computer. A few designs have been cooled by liquid flowing through passages within the computer (e.g., Saturn LVDC), but this method is becoming less common due to the development of more efficient thermal interface materials (ref. 25). Only minor thermal problems have been experienced with spaceborne computers, and these have been solved by minor modifications. For example, the radiational cooling of the Centaur computer was improved by simply changing the external finish from gold plating to white polyurethane paint (ref. 45).

2.4.2 Electromagnetic Interference

Digital circuitry is particularly susceptible to electromagnetic interference (EMI), of which there are many sources in a typical space vehicle. Relays and solenoid-operated devices, such as electrically-controlled valves or electrically-fired explosive charges used for stage separation, are especially troublesome generators of impulse noise. These voltages can enter a computer either on the power lines or through the signal interfaces, and their magnitudes can be quite high. For example, noise spikes of 50 to 100 V were observed on the power distribution buses in the Apollo spacecraft, and impulses of up to approximately 150 V were produced by the turn-on and turn-off of the Gemini rate gyros (ref. 15). EMI problems have been experienced at one time or another, particularly during testing, in nearly every spaceborne computer system. References 15 and 69, for example, contain a fairly complete discussion of the EMI problems encountered with the Gemini guidance computer and the EMI susceptibility testing conducted on it. In general, most EMI problems have been either unique to ground checkout installations or solved by minor modifications such as shorter ground straps, decoupling capacitors on input lines, or precautionary flight procedures.

2.4.3 Power Bus Voltage Variations

The power supplies in a spacecraft are not ideal voltage sources, which leads to potential problems besides those associated with strictly EMI effects. The voltage output of a battery varies with the temperature and state of charge of the battery. Fuel cells, like those used on the Apollo spacecraft, are also temperature sensitive in addition to having transient voltage variations due to many factors such as the dynamics of the fuel supply system. Spacecraft power systems are also subject to occasional externally induced variations, such as low-voltage transients or temporary outages. To minimize the effect of these occurrences on the computations, many spaceborne computers have a circuit in the power input section which detects a low voltage condition before the voltage drops to a level where the computer will not operate, and which either shuts down the processing in a safe and orderly manner or initiates a switchover to a backup power source. For example, in

the Gemini spacecraft, an auxiliary computer power unit (ACPU) furnished backup power for the computer to operate during low voltage transients of up to 100 msec on the primary supply (ref. 15). If the power was interrupted for a longer period, the ACPU shut down the computer in a controlled manner to prevent loss of memory contents.

2.4.4 Other Environmental Factors

Most of the other environmental factors that affect the spaceborne computer have a similar effect on other types of medium to low power equipment in the spacecraft. Among those factors whose effects have been found to be important during the early stages of design are shock, vibration and acoustic noise, humidity, vacuum, radiation, magnetic fields, and sterilization. As an example, the Centaur computer was repackaged to seal and pressurize the I/O unit and general purpose section and thus eliminate a corrosion problem caused by operating in a humid environment (ref. 45). The acoustic vibration environment experienced by the Atlas computer produced intermittent diode failures on several flights (ref. 70). The recommended solution was to modify the computer isolation mounting to further attenuate circuit-board vibration, but this was not implemented due to impending deactivation of the Atlas fleet. The use of magnetic material may be restricted if a sensitive magnetometer experiment is included in the spacecraft payload.

2.5 Reliability and Fault Tolerance

The reliability of a system is defined as the probability of correct operation for a specified period of time under given conditions. Fault tolerance is a more recent and a more general concept of the system's ability to continue to operate without errors after a fault has occurred. Until recently, reliability has been used almost exclusively in describing the operational lifetime of computers, as well as other devices. Conventional techniques which have been used for achieving ultra-high reliability in spaceborne digital computer systems include:

- Use of conservative design practices (derating, simplicity, wide tolerances)
- Parts standardization
- One hundred per cent screening of parts and assemblies, including thorough burn-in
- Detailed laboratory analyses and corrective action for all failed parts
- Use of extreme care in manufacture of parts
- Thorough qualification of parts and manufacturing processes
- Thermal cycling and vibration testing of all completed assemblies
- Establishment of an efficient field service feedback system to report on equipment failures in the field
- Design of the equipment to minimize stress during assembly and to facilitate replacement of failed components

In the past, the measure of a computer's reliability has generally been expressed as a mean time between failures (MTBF), which is the reciprocal of the estimated failure rate of the computer. The reliability analysis typically begins with a study of the components and fabrication processes from which the computer is to be built, including tests to determine the failure rate of the individual components and examinations of failed and unfailed components to determine the causes of failures. From these data and estimates of the number of parts to be used in the design, the failure rate for the total assembly is based upon certain mathematical assumptions (often unstated) of independence of failures, etc. At the present state of the art in construction of equipment, representative MTBFs for a medium-sized digital computer constructed with various component reliabilities would be as tabulated below.

Reliability Level of Parts and Practices	Representative MTBF (hr)
Commercial	500
Military	2 000
High Reliability	10 000

The mean time to failure of the Apollo guidance computers in the field has been about 13,000 hr, and the failure rate of the integrated circuit gates has been less than 0.001% per 1,000 hr. This excellent performance is attributed to the fact that only one gate type was used, and that an unrelenting effort was made in screening, failure analysis, and the implementation of production, test, and flight-processing specifications.

2.5.1 Redundancy

Beyond these conventional reliability requirements, spaceborne computers have been designed to continue accurate operation, even after a transient or permanent failure, through the use of redundant elements. In this concept, two or more elements, each capable of performing the necessary function, are carried in the vehicle. In the event of a failure in one element, the others are able to carry on. The penalties in volume, weight, and power prevented extensive use of redundancy in early space projects, but the advent of integrated circuitry has made redundancy techniques practical. Two categories of redundancy have been utilized (ref. 71): 1) dynamic redundancy, in which the occurrence of a fault is detected, and the fault and/or its effect is subsequently corrected; and 2) static or masking redundancy, in which the effect of a faulty element (component, circuit or system) is immediately masked by permanently connected and continually operating replicas of that element.

The early studies of redundancy techniques concentrated on static redundancy at the component level. Thus, gates or flip-flops were triplicated and majority logic was used, diodes were quadded as on the OAO (ref. 21), etc. As the reliability of individual components improved, it was found that the probability of failure rates for voters vs gates or flip-flops was such that it would be preferable to introduce redundancy of large modules, such as whole memories or arithmetic

elements, thus reducing the complexity of the voting or checking circuitry relative to that which was being checked. At the present time, the reliability of equipment built with high reliability integrated circuit components is a function of the number of interconnections (welded joints, soldered joints, and especially connector pins), as well as the number of active and passive elements. Thus, there is a tendency to keep the number of interconnections to a minimum by applying redundancy at a very high level.

The simplest redundancy technique is to use two identical systems with a provision for switching from one to the other when a failure is discovered. This dynamic redundancy approach is used in the Apollo telescope mount (ATM) pointing computer now in development. In the ATM, two identical computers, an active and a spare, are connected to the workshop computer interface unit (WCIU), which is an executive device that monitors the active computer and switches operation to the backup in case of a malfunction (ref. 72). Alternately, two different systems can be used, with the backup system having less capability than the primary system but enough to allow the mission to be aborted safely. This is the case with the Apollo lunar module's abort guidance system in which the AEA computer provides a backup for the primary Apollo G&N system (ref. 16).

Dynamic redundancy and error-detection logic are also used to improve the reliability of the Saturn LVDC memory section (ref. 25). In this self-correcting duplex scheme, shown in figure 6, when both memories are operating properly, each memory is controlled by an independent buffer register and both are simultaneously read and updated. Initially, only one buffer register output is used, but both buffer register outputs are simultaneously parity-checked. When a parity error is detected in the memory being used, operation immediately transfers to the other memory. Both memories are then regenerated by the buffer register of the "good" memory, thus correcting transient errors. After the parity-checking and error-detection circuits have verified that the erroneous memory has been corrected, each memory is again controlled by its own buffer register. Operation is not transferred to the previously erroneous memory until the "good" memory develops its first error. Consequently, instantaneous switching from one memory output to another permits uninterrupted computer operation unless simultaneous failures at the same storage location in both memories cause complete system failure.

Some recently developed computers are designed with one or more spares of each subsystem—processors, memories, I/O units, etc. The spares are held in an unpowered state and are used to replace operating units when a permanent fault is discovered. Examples are the Jet Propulsion Laboratory's self-test-and-repair (STAR) breadboard computer (refs. 73 to 75) and the onboard processor for OAO-B (refs. 22 and 23).

A popular static redundancy scheme is triple modular redundancy (TMR) with majority voting. This approach is used to improve the reliability of the logic section in the Saturn 5 LVDC. Figure 7 shows part of the LVDC logic divided into subsections called modules (M). Three identical copies of each module are used, with each channel receiving the same inputs at the same time. The outputs of the three channels are transmitted to majority voting circuits (V), which check the inputs for agreement. If one input differs, it is disregarded; so a single component failure will not cause a system malfunction (refs. 25, 76, 77). The LVDC is divided into seven

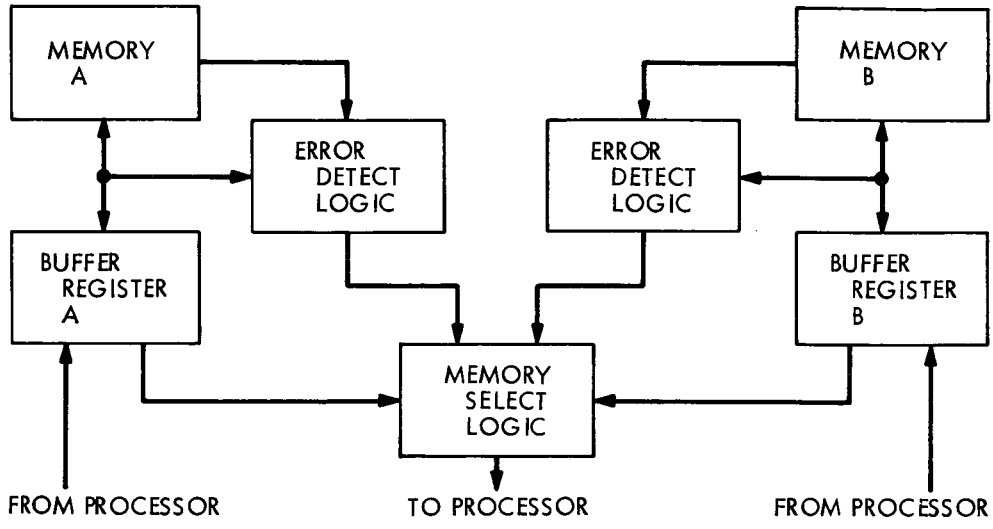


Figure 6.–Saturn 5–LVDC duplex memory diagram.

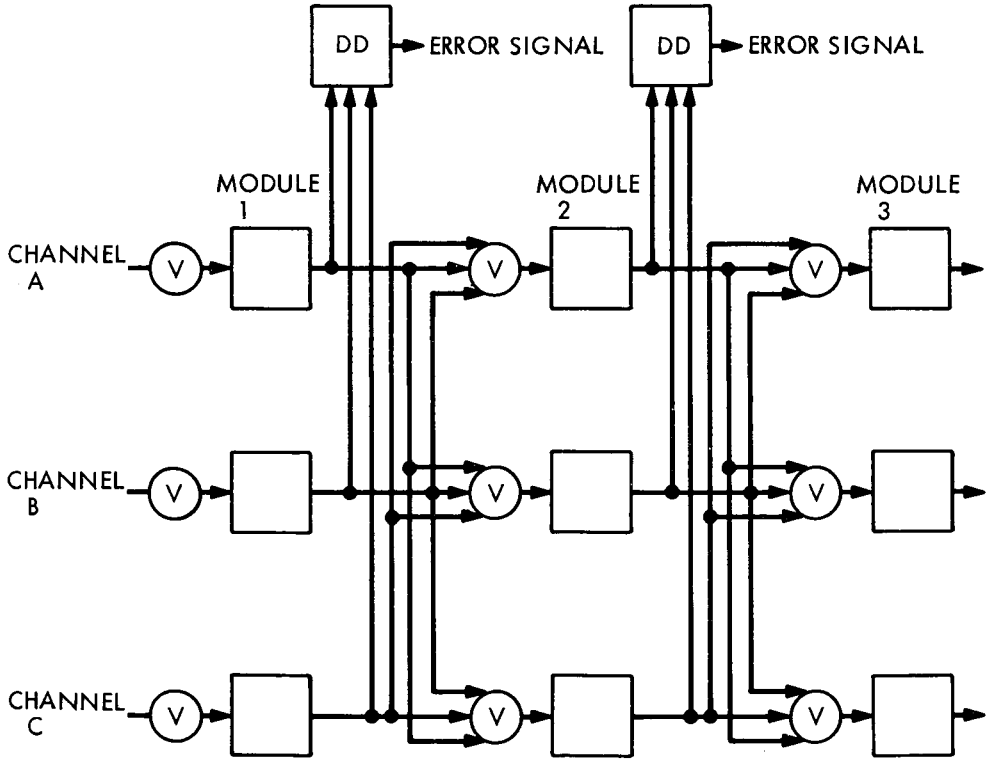


Figure 7.–Triple modular redundancy in Saturn 5–LVDC logic section.

modules with an average of ten voted outputs. Another circuit, called a disagreement detector (DD), monitors system performance by signaling the ground equipment whenever voter inputs are not identical. This is particularly important during prelaunch checkout to insure that a failure is not present and masked by the voters. If a masked failure did exist, the overall reliability of the system would be less than that of a nonredundant (simplex) configuration since a failure in either of the remaining good channels would cause the voter to give a faulty indication. The LVDC disagreement detector outputs are OR'd together so that malfunctions can be isolated to one, two, or three replaceable subassemblies.

TMR is also used to achieve the very high reliability required for the common section of the ATM WCIU, mentioned previously. However, instead of hardware disagreement detectors, the WCIU has a software capability for independently checking redundant channels. Over a short period, TMR can provide a substantial increase in reliability. For example, the TMR logic section in the LVDC is approximately twenty times more reliable for a 250-hr mission than an equivalent simplex system, with only about 3-1/2 times more components (ref. 25). However, for longer duration missions, TMR with majority voting is less reliable than simplex, as shown for a single module in figure 8 (ref. 78).

A substantial reliability increase over simplex operation can be achieved by reorganizing the TMR system to switch out one of the two remaining good modules, as well as the failed one, in the event of a malfunction (ref. 78). As a result, one or more modules may operate simplex while the remainder operate TMR (the "TMR/Simplex" curve in fig. 8). A somewhat greater improvement, shown by the "Switchable Spare" curve in figure 8, can be obtained by enabling the system to switch to the remaining good unit in the event of a second failure in a module. An extension of this concept, discussed in reference 79, consists of an N-tuply modular redundant (NMR) module with an associated bank of spare units. When one of the N active units fails, a spare unit replaces it and restores the NMR module to the all-perfect state. These approaches use a combination of static and dynamic redundancy to provide uninterrupted, long-duration reliability.

The ability to tolerate multiple failures may be implied by a long-duration high-reliability requirement, or directly stated by a fail-safe or fail-operational criterion. The latter case cannot strictly be called a reliability requirement, since no probabilities of failure are involved. Instead, this approach to fault tolerance requires that no single failure shall jeopardize the mission (fail-safe), or shall degrade system performance (fail-operational). These criteria are often stacked together, as in the fail-operational/fail-operational/fail-safe requirement for the proposed Space Station—Space Base computer system, which specifies normal operation after any two failures and safe operation after any three.

The use of redundancy is not always the answer to reliability improvement problems, however. Redundancy will improve the probability of success only if the simplex probability is high. The cost of a redundant system in terms of weight, volume, and power consumption can be quite severe, and a careful analysis might show that a more reliable system can be obtained by allocating the same resources to a simplex computer. Moreover, the checkout of a simplex system is often easier since there is no need to check each redundant element for failures that may be masked by some redundancy techniques.

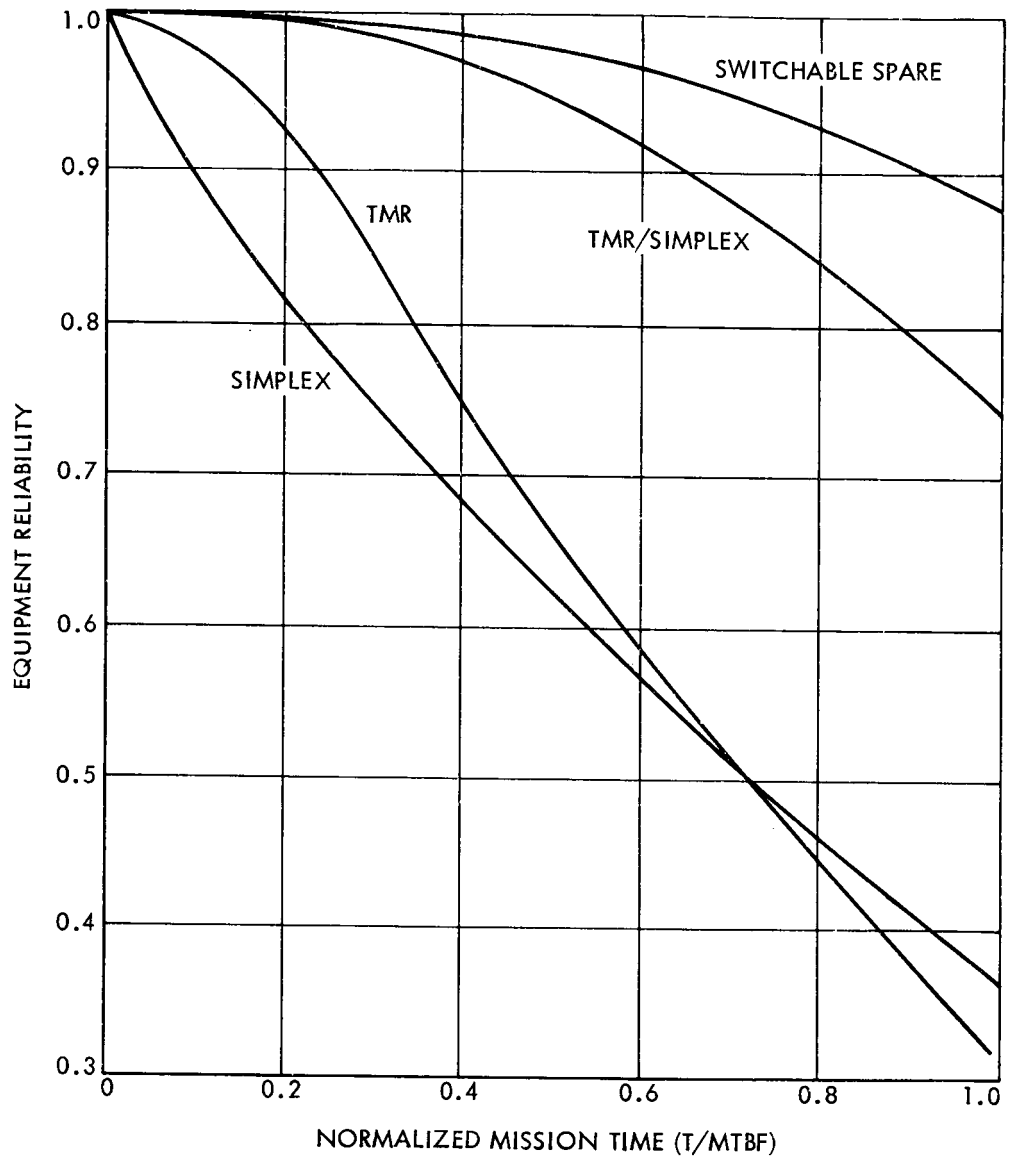


Figure 8.–Comparison of reliability techniques as functions of normalized mission time (ref. 78).

2.5.2 Restarting

The probability of successfully completing a mission which requires an onboard computer can sometimes be greatly increased by designing the computer in such a way that it is able to detect certain types of transient failures, errors, or power interruptions and to pick up the processing again after the transient has passed. This is called a restart capability. Sometimes, more than one level of restart capability is prescribed. The Apollo G&N computer, for instance, has two levels, restart and fresh start, which together are able to pick up and continue the program after certain hardware errors (parity error, oscillator fail), software errors (single instruction loop, excessive time in interrupt mode), or external occurrences (loss of power). The Gemini computer and the UNIVAC 1824 also had similar though less extensive capabilities.

Not all computers have this provision nor do they require it. The Saturn LVDC, for instance, is not designed with a restart capability because no practical restarting sequence could recover the mission in the event of a transient problem in the LVDC during the boost phase. Instead, the LVDC uses extensive redundancy, as described earlier, and is designed to be especially resistant to environmental effects such as EMI. To illustrate the success of this design, the LVDC was able to place the Apollo 12 mission into a nearly perfect parking orbit after the vehicle suffered several lightning strikes during the first several seconds after liftoff.

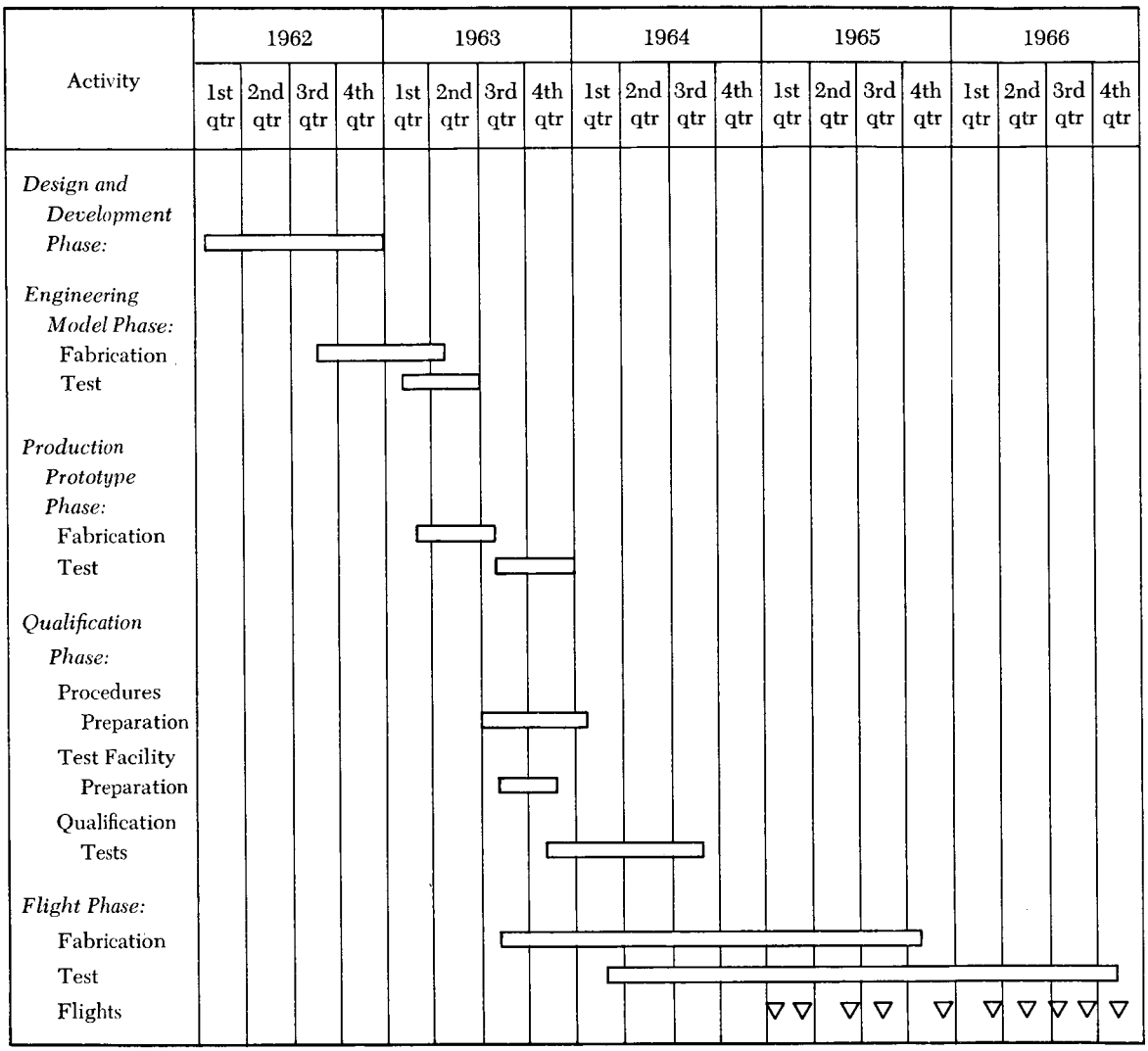
2.6 Testing and Checkout

During its development, construction and operation, a spaceborne computer is subjected to extensive testing to insure that it performs its assigned functions properly. Much of this testing is involved with generating, verifying and debugging the software, the details of which are outside the scope of this monograph. This section briefly describes the testing required for hardware development of a typical spaceborne computer. To illustrate the level of testing involved, table 4 (ref. 15) shows the history of the design, development, and flights of the Gemini guidance computer.

As soon as the basic logical configuration of the computer has been determined, a functional simulation of the operation of the proposed design is performed on a general purpose ground-based computer (e.g., ref. 80). A measure of the real time required to execute the program is maintained, and the static and dynamic characteristics of the proposed interfaces to the computer are simulated. The simulation is used to verify that the instruction set is adequate, that the speed and word length are satisfactory, that the I/O provisions will be sufficient, etc. (ref. 81). Since the mission software is generally developed in parallel with the hardware, it is sometimes possible to use preliminary versions of the operational programs in the simulation. Otherwise, programs which duplicate the anticipated mission as closely as possible are used.

Hardware development is started as soon as an initial configuration for the computer has been established. Each testable item of hardware fabricated in the prototype or engineering model state is tested to insure that it is properly designed, that the mechanization of the logic is correct, and that it will perform in the anticipated environment. Generally, several iterations are required at this stage. As soon as a complete computer is available, it is subjected to thorough design

TABLE 4.–History of Gemini Guidance Computer (ref. 15)



qualification tests, including verification of its ability to withstand the anticipated environment while operating, including:

- Realistic variations in the characteristics of the interfaces
- Electromagnetic interference
- Vibration, shock, acoustic noise
- Vacuum
- Variations in power input

- Temperature extremes
- Anticipated radiation levels
- Variations in clock frequency

Each production model of the computer is thoroughly tested to insure that no fabrication or design errors were committed. A typical series of tests includes:

- Testing of all computer capabilities with test programs designed to isolate errors
- High-low internal voltage tests to detect weak components
- Temperature cycling and vibration (operating if possible) to locate bad connections
- Burn-in to eliminate early life failures in components
- Special tests to check any marginal areas indicated by the design verification tests

Prelaunch checkout of a spaceborne computer typically consists of an extensive series of tests to verify that the computer and all of its interfaces operate properly. Special precautions are required in the checkout of redundant systems to ensure that the redundancy is not masking one or more failed components at the time of launch. In the Saturn LVDC discussed previously, the problem was solved by use of Disagreement Detectors (see fig. 7). A monitoring phase is generally engaged during which the computer runs test routines at regular intervals, interleaved into any operational programs that are required during the prelaunch phase. In some systems, the monitoring phase of checkout involves the loading of a separate program into the computer. In this event, a detailed verification that the program has been loaded properly is performed before the active phase of the testing is finished.

Inflight testing of the computer, one of the major operational functions, has been discussed previously in section 2.1.6.

3. DESIGN CRITERIA

The mission requirements, functions to be performed, and precision and reliability requirements must be defined early in the design. The spaceborne digital computer shall be capable of performing all the functions allocated it during the mission and during prelaunch testing and checkout. The design shall achieve the specified precision and reliability of performance within allotted space, weight, and power consumption constraints. The computer's speed of operation shall be adequate both for the critical parts of the mission and for handling restructuring or recovery when these facilities are provided.

3.1 Design Tradeoffs

The design shall operate satisfactorily in the mission environment and shall minimize susceptibility to factors inherent in the operational environment and the vehicle installation, including

various interfaces and design tradeoffs between the computer and the spacecraft, the spacecraft environment, and interference sources. Demonstration shall be made, by a suitable combination of analytical and experimental studies, that the design is entirely suitable for its intended mission. Recommended practices for the factors which can be used to evaluate the design tradeoffs (usually on a qualitative basis) are discussed in section 4.1.

3.2 System Design

3.2.1 Physical Features

Mechanical design of the computer shall take into account at least the following considerations: size, weight, acceleration and vibration, mechanical shock, humidity, ambient pressures and temperatures, corrosive atmospheres, and human factors. Consideration should also be given to adequate sealing, access to subunits for repair and access to connectors. Proven techniques of interconnection and packaging shall be utilized. The electrical properties of interconnections shall be controlled with respect to electromagnetic interference susceptibility, cross talk, and transmission properties. The power source and ground requirements for the system shall be compatible with spacecraft provisions. Standby or backup power is to be provided whenever needed.

3.2.2 Memory Features

The computer must have a memory subsystem capable of storing all programs including prelaunch testing programs. There shall be adequate random access, high-speed memory to hold all programs, and data which will be required simultaneously during peak load periods. The program memory shall exhibit adequate reliability with the capability of withstanding power supply transients and temporary power failures. If the memory is not fixed or "read-only," precautions shall be taken to preclude irrecoverable loss of the programs or critical data resulting from an electrical transient or a program flaw. The memory system shall be capable of being turned off and on without losing its contents. Provision for a backup or reload capability shall be considered for storage reliability.

Consideration shall be given to ways in which expanded memory requirements can be met, such as by the addition of memory modules or a bulk memory, or by reloading the memory from external sources.

3.2.3 Processor Features

The computer shall provide an instruction repertoire commensurate with the mission demands and operating speeds which are clearly adequate for peak loads. These shall include as a minimum the following basic instructions: load, store, logic, arithmetic, input/output, conditional and unconditional branches, and shifts. Special instructions to facilitate particular functions, such as data conversion for nonbinary inputs (e.g., man/machine interface), telemetry formating, and

driving displays, are to be provided if required. Floating-point or multiple precision instructions, or instructions to facilitate these operations, should be considered in the design if needed to satisfy precision requirements and to reduce software development costs. If a sample problem is used to demonstrate the efficiency and speed of the computer for a given mission, the sample problem's instruction mix must closely compare with that to be used during the actual mission.

3.2.4 Input/Output Features

The computer shall provide a means for timely subroutine entry when critical input signals are received and for adequate buffering of data which do not require immediate servicing. Priority sequencing of responses to critical inputs shall be provided if necessary.

The interface circuitry and devices shall provide both for adequate reception and transformation to internal form of all signals from sensors, external communications from the ground or crew, and other system inputs, and for output signals of proper electrical characteristics for all actuators, telemetry, displays, and other output devices. When analog-to-digital and/or digital-to-analog conversion is necessary, the speed, precision, and accuracy of conversion shall be within necessary tolerances. The input/output electrical characteristics shall provide for adequate matching of converters to the input or output electrical interface circuitry.

The computer and the peripheral units shall be designed in such a way that the time required to respond to an input request or to generate an output signal shall be within the physical requirements of the vehicle system. Likewise, the interface bandwidth shall be adequate to support the required data volume, including recovery modes, without unnecessarily burdening the computer.

3.3 Simulation

Means shall be provided to facilitate the verification of flight hardware and software by appropriate real-time simulation facilities, support equipment, and cooperative test features in the computer, such as clock controls and memory dumping sequences. All programs shall be thoroughly specified and verified at all levels from subroutines to total program assemblies. Means shall be provided to guarantee that the program which has been verified is identical to the program loaded into the flight computer and that the flight computer has been correctly simulated during verification.

3.4 Testing and Checkout

Tests shall be conducted which sufficiently establish that the computer performs as required under all expected operating conditions. Functional and diagnostic testing of the hardware and software shall be performed. Checkout programs shall be utilized to exercise the computer capabilities, to simulate malfunctions, to restructure the system in case of malfunctions, and to restart the program in case of temporary failure of the power supply or loss of sensor inputs. Self-test software shall be provided for computers which cannot be adequately monitored by passive means. Means

shall be provided to verify the correct operation of all features of the computer, including all redundancy and repair mechanisms, after its installation in the flight vehicle and with other systems operating.

The computer shall be designed for ease of checkout and for accessibility. Utilization of the computer for the testing and checkout of other vehicle subsystems, both prior to and after launch, shall be considered. Compatibility of the computer with other subsystems shall be verified by testing.

3.5 Reliability and Fault Tolerance

The available lifetime of the computer shall be adequate for the mission requirements and shall be demonstrated by approved analyses and tests. If required by the tasks to be performed, the computer shall have the capability to detect certain transient internal or external failures (including loss of power) and to take appropriate action to minimize the effects of such occurrences. Malfunction detection equipment or software shall be provided to enable the crew or ground control to recognize the need for repair or change in operational procedure. If necessary, the computer should have the capability to restart or recover after such a failure with a minimum specified effect on the tasks being performed. The computer operation shall be as insensitive as possible to malfunctions and transients in other subsystems.

Extensive qualification testing of components, circuits, assembled computer hardware, and software, use of redundancy, and provision of self-test and malfunction detection programs are to be considered as primary methods of providing reliability.

4. RECOMMENDED PRACTICES

The design of a spaceborne digital computer entails a series of decisions involving interacting disciplines including stabilization and control, guidance and navigation, power, data processing, display, telemetry, and testing. A satisfactory configuration can best be achieved if consideration is given to all interacting aspects of design throughout the design process, so that the merits of various options may be assessed and their full impact on all systems identified. Therefore, it is recommended that personnel from the computer design group and all concerned groups work closely together throughout development and participate in all tradeoff decisions.

The experience, inventiveness, and general ability of a designer can greatly influence the number of failure-prone components, close tolerance items, interconnections, power requirements, and general simplification for a given system. However, to specify design methodology presupposes *a priori* knowledge and may result in unduly restricting a highly qualified designer. The difficulties in specifying how a system is to be designed, or who is to accomplish the effort, have led to a general practice of accepting the resultant design and restricting reliability discussion to the areas of component testing, environmental system testing, and mean time to failure statements. Notable exceptions have been made by the NASA Flight Research Center (FRC) and the NASA Lewis Research Center (LeRC) during recent procurements wherein, noting that a better grade of

designers were on proposal staffs, they made it a contract requirement that these same engineers participate in the actual design. Such a practice is recommended in future procurements.

The mission requirements should be established as soon as possible in the program so that computer functions, operations, and architecture can be identified. The consideration of the computer as a part of the overall system is the first step in formulating the computer requirements and is the primary factor in most design decisions. Care should be taken to insure that the computer system requirements are as liberal as possible since specifying tight requirements or choosing the lowest priced computer that barely meets the requirements usually results in a more costly system over the range of the program (ref. 82). Design freedom should be encouraged by not overspecifying computer requirements, particularly in the areas of technology or architectural design. Standardization should be sought wherever possible—especially on interfaces, languages, and sample problems—in order to facilitate checkout, repair, and evaluation. To insure a reasonable performance match, it is important to specify the software operating system in some detail before "freezing" the hardware design.

If mission requirements are not firmly established, as is typical of long-range programs, the computer design should be approached with flexibility and foresight. The designer should anticipate the potential growth in use of the computer by carefully reviewing previous programs, particularly those which are similar to the present program (e.g., table 1 in section 2). The use of reference 83 is encouraged where more specific guidelines are unavailable.

The designer should consider the use of advanced technology to meet performance objectives or to reduce cost, especially if the requirements are difficult to fulfill within state-of-the-art technology and where the development program is long. The selection of advanced technology should be based on as much developmental testing as is currently available. If possible, its use should be restricted to minimize potential harm in the event of failure. Where a high design risk is involved, it is recommended that a competitive development be initiated, utilizing more proven technologies but converging on one design as soon as possible.

4.1 Design Tradeoffs

Conceptual design involves a series of tradeoff decisions among significant parameters—such as operating speeds, memory size, power, and I/O bandwidth—to obtain a compromise design which best meets the performance requirements. Both the uncertainty in these requirements and the important tradeoff factors should be ascertained. Those factors which can be used to evaluate the design tradeoffs (usually on a qualitative basis) include:

- Reliability
- Expandability
- Programmability
- Maintainability
- Compatibility

- Adaptability
- Availability
- Development Status and Cost

Recommended practices for achieving *reliability* are given in section 4.5. The remainder of these features are discussed below.

Expandability measures the computer system's ability to conveniently accommodate increased requirements by higher speed or by physical expansion, without the cost of a major redesign (ref. 84). The original design of the computer should provide for this type of growth, especially with regard to the memory and I/O sections. The general procedure is to determine all the functions that foreseeably could be demanded of the computer system, such as by reviewing growth problems of past programs, and to establish a range of possible requirements for each of the functions, which may double the present requirement. If possible, the likelihood of these expanded requirements should also be estimated. Modularity is a desirable method for providing expandability and should be incorporated whenever feasible.

Programmability, or the ease of programming the computer, should be considered early in the design. Past experience has shown that a balance between programming simplicity and hardware complexity is essential to prevent the costs of programming from becoming overwhelming. For example, sufficient memory capacity should be provided to accommodate program changes necessitated by increased performance or mission requirements; memory architecture should be designed to facilitate programming; and hard-wired memories should be avoided if many program changes are anticipated, because of the time and cost involved in implementing the changes. Adequate addressing facilities without artificial boundaries and a simple subroutine linkage are recommended. Considerations of programmability should include the efficiency of the source language, of the object code, and of converting from the source language to the object code, and the ease of using the source language and obtaining a completely coded computer program. If the computer is to be programmed in flight, the use of a compiler should be considered. A standard programming language, such as JOVIAL, SPL, or CLASP, is desirable and should be utilized for future applications if available. The degree of software sophistication and the availability of support software should be considered during the design.

Maintainability should not be neglected when designing the computer. Repair should be readily accomplished during ground operation, and if inflight maintenance is desired, this should be specified as a design requirement. Inflight repair or reconfiguration is closely associated with reliability and, as such, the extent of reconfiguration made possible will be dependent upon the reliability required. Malfunctions can often be detected by self-check programs; inflight repair can be effected by automatic switching, or by manual operation on manned missions. The tradeoff should consider the use of a degraded mode of operation. Generally, the recommended prelaunch maintenance procedure is to remove bad components or subsystems from the system and replace them with backup equipment. To facilitate manual maintenance, subassemblies should be pluggable, require a minimum of disassembly for access, and be replaceable without adjustment. The design should provide ease of accessibility and should minimize the possibility of damage to other parts during maintenance. If replaced assemblies are to be discarded rather than repaired, maximum cost goals for a replaceable module should be established.

Compatibility should be developed between the computer and its interfaces, software, power levels, and, where necessary, ground computers. Standard interfaces and power levels should be implemented. Interface compatibility reduces the need for data conversion with peripheral equipment and is highly recommended. Data compatibility between models of a computer family should be provided to simplify the design of peripheral equipment. This consideration is particularly important when computers of different performance and architecture are interconnected. Source and object code compatibility between the spaceborne and ground-based computers is advantageous to facilitate programming.

Adaptability is defined as the ability of the system to meet a wide range of functional requirements without requiring physical modification. Adaptability is needed when requirements are not well defined or if it is anticipated that the computer will be applied to a variety of missions and/or a number of space vehicles. Although this is similar to the need for growth discussed under "expandability," in this case, potential requirements should be anticipated by providing reserves in memory capacity, computational speed, word length, and I/O capability. Moreover, the design should consider specific features which allow tailoring a basic machine to different situations, such as an adjustable word length through byte organized operations, alterable or unused operation codes, reserved fields within formats, and adjustable speed. Caution must be taken that the increase in computer capabilities is in accordance with other development considerations.

Availability is the probability that the computer is operating satisfactorily at a given time. It is closely related to reliability and repair time and should be considered in establishing reliability requirements. Since it takes into account the time required for malfunction detection and reconfiguration or repair, availability is particularly important during time-critical mission phases.

Development Status and Cost are complex management-related factors which can have significant effects on the design. They require the estimation of a number of items such as the extent of off-the-shelf hardware use, design risks in developing new equipment using advanced technologies, potential progress in the state of the art during the design and development of the computer, etc. In estimating cost, the manager should consider the total long-range expenditures as well as initial outlays, and also the cost of potential delays in developing advanced techniques, etc.

In addition to the above qualitative factors, tradeoffs should be determined on the basis of specified quantitative factors, such as precision, speed, capacity, weight, volume, and power.

4.2 System Design

The selection of the computer organization and configuration depends upon the mission requirements; the choice should be most carefully and thoroughly planned in light of available technology. The designer should consider the available diagnostic capability before partitioning the computer in terms of physical modules since the diagnostic program is sensitive to module size and functional consistency. Multiple processors should be considered since they may be capable of continuous performance despite failures in single units and have additional potential in areas such as onboard autonomy, checkout, and flexibility. If a group of computers is to be used, it is not possible to recommend any one organization since paths and functions are not uniquely

related. However, the following organizations should be considered: federated, dedicated, multicomputer, multiprocessor, distributed processor, and combinations of these. If dedicated computers are used, backup compatibility should be provided for critical functions. Similarly, if degraded modes are permitted in any organization, the design should provide sufficient reliability to insure that critical functions will not be affected. If a multiprocessor is being considered, development of the executive software required for the assignment of tasks must not be neglected. The use of special purpose data processors controlled by a central computer should be considered for routine and repetitive tasks because of 1) bandwidth of circuits, 2) need for computer availability in case of anomaly, and 3) fewer interface circuits and simplified subsystem testing.

Working with system designers and a knowledge of the mission requirements, a list of functions which the computer must perform should be prepared. Typical computational tasks which might be mechanized on a digital computer system for a long-range manned mission are shown in table 5 (ref. 85). A list of this type can be used as a basis for the block diagrams of the system programs as well as for preliminary estimates of the speed and memory requirements of the computer.

TABLE 5.–Typical Computational and Data Processing Functions of Spaceborne Computers (ref. 85)

Command and Control	Vehicle Guidance and Control	Guidance and Navigation	Navigation
			Targeting
			Required Velocity
			Velocity To Be Gained
			Flight Sequencing
			Steering
			G&N Controls and Displays
		Vehicle Attitude Control	Optical Sensors Orientation
			Angular Rate Stabilization and Control
			Translation Control
			Thrust Vector Control
			Attitude Controls and Displays
	Telecommunications	Telecommunications	Antenna Orientation
			Data Processing
			Communications Controls and Displays
Mission Data Processing	Scientific-Sensors Data Processing	Image-Sensor Data Processing	Image Sensors Orientation and Sequencing
			Image Sensors Data Correlation/Analysis
			Image Sensors Data Compression
			Image Sensors Controls and Displays
		Scientific Sensor Data Processing	Scientific Sensors Orientation, etc.
			Scientific Sensors Data Correlation
			Scientific Sensors Data Compression
			Scientific Sensors Controls and Displays
	Status Monitor	System Self-Test Operations	Automatic Self-Test Operations
			Self-Test Controls and Displays
		System Performance Monitor	Performance Data Compression
			Monitor Controls and Displays

The next step to be taken in the design process is the preparation of a block diagram showing the relation of the digital computer to the complete system. All I/O paths between the computer and the other major subsystems should be shown in this diagram in order that the overall functioning of the computer is apparent. An I/O system concept should be developed to satisfy the functional requirements. A detailed functional block diagram should then be drawn in which the individual I/O signals are shown and the overall blocks of the prior diagram are broken into specific components. When the functional block diagram has been prepared, the types of signals which each of the input devices will present to the computer and which the computer must deliver to other parts of the system must be specified. These specifications should include 1) the levels of the analog signals and their expected ranges, 2) the desired terminating impedances, 3) the required precision of the signals, 4) the frequencies at which the signals must be sampled, and 5) the grounding, shielding, and cabling requirements.

Restart capability in case of operational difficulties should be considered for all applications. Such provision should be made as early in the development program as possible, and the effects on the computer memory should be evaluated. At the same time, the effects of power supply transients should be estimated and provisions made to provide the computer with a recovery mode in case of temporary power failures. The power supply should be protected from injurious stress being imposed by an overload. Electronic limiting is recommended over fusing because it leaves the system operational if the overload is removed. Early consideration should be given to the number and size of voltage levels required. Primary and secondary power supplies should also be designated.

When the functions the computer must perform and the details of the I/O data have been specified, a block diagram of the software for the system should be drawn to estimate the characteristics of the programs required. A preliminary estimate of the program memory size should be made by roughing out the steps in the programs. The number of operands required by the subprograms should be summed to obtain an initial estimate of the variable memory requirements. If a scratchpad or a small high-speed working memory is to be used, the optimal size should be approximated. Initial estimates for the speed of the computer should be obtained from this block diagram. However, such estimates should be verified by simulation as early in the program as possible.

4.2.1 Physical Features

The size, weight, and power limitations of the computer should be specified to the designer, and, if possible, the relative priorities of these should also be provided to guide the designers in tradeoff studies. Design risks should not be taken in order to save small amounts of weight. The environment in which the computer is expected to operate, and to which it will be subjected in acceptance tests, should be specified including acceleration and vibration, mechanical shock, humidity, ambient pressures and temperatures, corrosive atmospheres, electromagnetic interference, radiation, and human factors. A means for heat dissipation from the computer is essential. Cooling methods which should be considered include conduction to a cold plate, convection (if an atmosphere is available), direct radiation, and combinations of these. Other systems, such as a wet system using ethylene glycol, should be considered.

Transients produced by relays and other electrically actuated devices should be defined and tested. EMI problems can usually be reduced by very careful design, especially in the areas of grounding, shielding, filtering, and impedance levels. Impulse noise on power lines, ground lines, and signal lines should be considered, as should the potential deleterious effect of long cables on both pulse amplitude and waveform of high frequency systems. EMI design and test requirements are difficult to specify; the use of MIL Standards 461 to 463 is suggested if no other specifications are available.

Components which usually pass initial qualification testing but deteriorate with use should be avoided. Examples of suggested alternatives are 1) the use of solid state devices to avoid mechanical contacts in relays and choppers, circuit simplification, and layout to minimize the number of connectors and conductor terminations, and 2) the use of circuitry that does not require fine wire transformers. Connectors should be keyed to prevent incorrect interconnection of modules.

4.2.2 Memory Features

To protect the system from additional demands on the memory, either the memory must be initially specified with an ample margin, or some provision for expanding its capacity must be made. The latter option is generally to be preferred, and most designs make provision for expansion at the onset. The importance of allowing ample margin for growth in the computer's memory capacity cannot be overemphasized. This margin should be a function of how well the computer requirements have been defined and is particularly important in developmental programs, such as Gemini and Apollo, in which expanded capabilities are added as the requirements evolve. To aid in evaluating the memory capacity, the expected variance in mission and function requirements should be determined. General purpose design is enhanced by electrically alterable program storage; therefore, mechanically altered "read-only" techniques, such as core-rope or missing core memory systems, should be selected with caution. Consideration should also be given to providing onboard bulk storage devices. Tape storage should be used for non time-critical items only.

Since the memory often consumes the bulk of the system power, techniques for reducing memory power requirements should be considered, such as the use of smaller cores, or partial switching of the memory elements. To reduce power consumption on long-duration missions, the computer should be capable of powering down at least part of its operation during inactive periods.

4.2.3 Processor Features

Instructions should be provided for the following basic operations: load, store, add, subtract, multiply, I/O, branch conditional, branch unconditional, and shift. As the use of additional hardware becomes feasible, serious consideration should be given to instructions for loop control, indexing, masking, and additional branching. Addition of these functions usually results in decreased program size and increased performance as well as improved programmability. For guidance and navigation applications, divide, subroutine linkage, and extended precision arithmetic should be considered as well (ref. 46).

Two's complement is recommended for data representation since it provides a unique representation for zero, does not require recomplementation, and simplifies multiple precision arithmetic. Fixed-point arithmetic should be used whenever hardware is at a premium and mission requirements are relatively stable. Floating-point arithmetic, using either hardware or software implementation, should be considered to ease the programmer's burden and to simplify software verification.

The expected rate at which inputs will arrive should be defined, and the types of signals to be provided to external devices should be stipulated. If the system must provide communication between the computer and the crew, the devices for manual data insertion and for display should be specified in detail. Data conversion techniques for manual displays should be specified; they may be handled by hardware or software.

Increased speed usually results in increased power consumption; therefore, sophisticated arithmetic algorithms should be investigated to reduce the requirement for a very high clock rate. The following techniques are recommended for enhancing the speed of the computer: 1) separate adders in a parallel machine for indexing and arithmetic, 2) parallel circuits to speed up multiplication by processing several multiplier bits at a time, and 3) single tailored instructions which perform complex jobs.

The precision requirements should be stipulated early in the design so that the optimum tradeoff between word length and double precision can be resolved. Moreover, the necessary precision of all I/O data should be established early for the same purpose.

4.2.4 Input/Output Features

System interrupts should be accommodated either by frequently executed test instructions to detect service requests from peripheral devices, or by an interrupt system which forces a branch from the main program to a servicing routine upon demand. When multiple channels may attempt to interrupt simultaneously, an interrupt priority system should be established. A tradeoff is required between the use of additional hardware to decrease instruction execution time and thus permit programmed polling of requests, or to implement the interrupt system without altering instruction execution time. To the extent possible, the location of all possible interrupts during a program should be determined and their effect on the program operation examined, including interrupts within interrupts. Protection against interrupts should be provided at points in programs where their occurrence would be harmful. Direct control of I/O data should be considered for computations in which the data output is small compared to the amount of calculation, since I/O-to-memory channels are cumbersome to start and stop. An adequate number of external test points should be provided, even at the expense of a small weight penalty.

The interface between the computer and all subsystems with which the computer must communicate, including test equipment, should be specified functionally, mechanically, and electrically. Standardization of interfaces is emphatically recommended. Moreover, the interface should be asynchronous or self clocking to minimize hardware changes due to system modification. Whenever possible, the interface design should be independent of cable length or logic delays internal to the peripheral equipment. If the computer is to be self checking or tested

manually, the I/O techniques which are to be used should be specified. The tradeoff between performing routine I/O format calculations at the peripheral device or within the central processor should be investigated. If A/D devices external to the computer are provided, both the output levels of these devices and the technique for interfacing them with the computer should be specified. The sampling rate should be designated as well as the ability of the A/D converters to buffer information during intervals in which the computer cannot respond immediately.

4.3 Simulation

Verification of the spaceborne computer by simulation on a general-purpose ground-based host computer is highly recommended. Simulation techniques should be considered before the computer characteristics are frozen. Since simulation is the only way to evaluate the computer before hardware has been fabricated, it is recommended that the effects of any proposed modification or expansion of the computer be examined by the simulation before a final decision is made. Simulation is most expeditious if the spaceborne computer is program-compatible with the host computer.

The host computer should be programmed to duplicate the results of each executed instruction of the spaceborne computer. It should maintain a measure of the real time that would be required to execute the program undergoing checkout on the spaceborne computer. The simulation should also include either static or dynamic external inputs, such as from gyroscopes and accelerometers, and manual data insertion and display devices. Since the I/O equipment on the host computer is usually not the same as on the spaceborne computer, the simulation should at least duplicate the data transfer portion of the I/O operation and allow interruption of the simulated machine. The simulation should allow detailed snapshots of the memory, core dumps, and branch traces.

Sample problems containing typical calculations for the expected mission are recommended for use in the simulation to compare organization, memory size, word length, functions, etc., of competing designs. The sample problems should be as representative as possible of the functions to be performed on the class of missions under consideration, such as boost guidance and vehicle control. These problems should consist of frequently used subroutines for each function. Executive control functions should also be included among the sample problems as well as problems requiring significant I/O operations. Estimates for computational speed, memory capacity, throughput, etc. should be made from the same sample problem to achieve a fair overall evaluation. In defining the sample problems the following information should be provided: 1) equations, flow, and decision logic for the problem, 2) I/O parameters along with the required precision, 3) iteration rate, and 4) percentage of total function represented by sample problem.

As a first step in developing the programs for the spaceborne computer, an assembler and/or compiler for the system should be prepared for the ground-based host computer. The compiler should execute on the ground-based computer to generate the object codes for the spaceborne computer and for checkout with the simulator. For cases where direct execution on a ground-based computer is desired, FORTRAN or PL/1 is usually sufficient to check out mathematical procedures, etc. Floating-point arithmetic on the ground computer is suggested to assist in developing a scaling strategy by indicating the expected ranges of the variables. The effort required to analyze and program mathematical procedures will be minimized if the same compiler-

generated code used in developing the procedures on the ground-based computer is executed by the spaceborne computer. The simulator should be used to debug and test programs since the host computer can be provided with extensive tracing routines and diagnostics which are not practical for the smaller aerospace computer.

As the development of the space vehicle computer progresses, the simulation should include an accurate representation of the vehicle dynamics, sensors, and environment for the entire mission using the actual flight program (ref. 81). Simulation at this level provides a detailed picture of the dynamic interaction and the response of the programmed equations as they have been implemented on the spaceborne computer. Long-term effects of error propagation should be evaluated and subtle error sensitivities may often be uncovered.

Finally, a real-time simulation, using the computer itself and as much of the actual flight hardware as is feasible, should be conducted in order to validate the all-digital simulation. This may involve a hybrid analog-digital (A/D) computing system to represent the remainder of the vehicle and its environment.

4.4 Testing and Checkout

Preparation of test procedures and construction of test facilities should begin early in the program. The test specifications should clearly distinguish between the tests required at various stages of design, qualification, production, installation, and flight. Feasibility tests of components and circuits may be conducted prior to completion of a breadboard computer. Engineering prototype computers may be used for hardware logical and electrical verification and for software verification in hybrid simulations. Full engineering qualification testing should be conducted using an early production model, and every production model should be subjected to an acceptance test which is as thorough as possible without overstressing the computers. Finally, the computer system and its interfaces should be tested in the vehicle to give sufficient confidence that no significant degradation has occurred since acceptance. It may be advantageous at any stage of testing to test the computer with another flight computer or a ground-based computer.

Generally, worst-case conditions should be used for testing. For qualification, the components should be tested to failure to realistically establish their actual limits. Where test specifications are not given, MIL Standard 883 contains suggested test methods and procedures for microelectronics. The computer should be tested while operating under environmental conditions including temperature cycling, vibration, vacuum, and radiation. If a failure occurs during a test, a logic proof should verify the failure point, repair should be effected, and elimination of the failure should be proven by demonstration. The test should be recycled to the previous checkpoint unless a major repair was required, in which case, the entire test cycle should be repeated. Special tests should be devised to verify each of the computer's functions, such as guidance, control, etc. Interfaces should be represented as realistically as possible, including waveforms, impedances, noise, voltage variations, etc.

A functional test program should be initiated early in the development phase. The availability of such a program should be reflected in the design to maximize ease of checkout. For acceptance testing, this program should exercise all the I/O units and the entire memory, while operating

under environmental conditions. If possible, the test conditions to be withstood should be specified, rather than arbitrary. Special purpose hardware subsystems should be considered for monitoring and diagnosing the rest of the computer, so that acceptance testing can be largely carried out by the special purpose part. If this procedure is adopted, only the special purpose equipment need be validated.

The ease of checkout during acceptance and qualification tests and, particularly, during prelaunch activities should be considered during the design of the computer. Prelaunch checkout should be as complete a checkout of the computer and its interfaces as can be conducted without removing the computer from the vehicle or disconnecting interfaces for any operation. The external equipment required and the ability of the computer to aid the checkout procedure should be considered. It is necessary to specify the electrical capabilities of the signals and circuits in the computer when operating with ground-support equipment cables and external loads, and, whenever possible, to specify the operating routines for checking various parts of the computer system using ground-based computers prior to launch. Ground-based checkout systems should have such specific features as: checks for proper operation of arithmetic, control, memory, and I/O sections; verification of any index or extension registers and of testing and jump instructions; testing of multiplexers and A/D converters; and testing of the system outputs including any D/A output devices and discretes delivered to telemetry or control devices. If a redundant system is used, each channel should be verified prior to launch. If necessary, external indicators should be used to verify the proper operation of clocks and power supplies in the system through the external harness. Provision should be made for varying all primary and secondary voltages for onboard margin testing. Self-test programs for onboard testing should include parity checks and internal coding checks on transmission. Use of an integrated testing concept in which certain prelaunch and inflight tests are carried out in a common manner by essentially the same automatic equipment is desirable, both for economic reasons and to maintain continuity of testing.

4.5 Reliability and Fault Tolerance

A suggested technique for defining reliability requirements is to provide a list of functions to be performed by the computer during each mission phase and a corresponding listing of the probability of successful performance required. The reliability specifications should be reasonable and appropriate for the mission. For example, the hardware and software should be highly reliable for guidance and navigation, whereas lower reliability specifications may be acceptable for experiments. If possible, the extent of coverage or fault tolerance should be specified. Caution should be used in comparing the reliability of different systems unless their reliabilities have been predicted for the same assumed conditions and those conditions are carefully described. An unqualified specification of MTBF should be considered to be inadequate.

To establish a meaningful measure of reliability, the following items should be considered: 1) the required operational time before failure; 2) the repair time, if repairs are possible; 3) the mission phases; 4) the probability of operating during a critical time period; and 5) environmental factors. For multiple channel organizations, it should be noted that reliability is generally based on independent failure assumptions and lack of interference between channels. For multiphase missions, a state-phase calculation may be used to determine how many computers or how many redundant modules should be operable at each critical phase. A thorough reliability analysis of

all anticipated failure modes should be conducted. During initial testing and checkout, all failures should be attributed to these specific modes, or additional modes should be added to provide a complete understanding of all experienced failures. The failure effects and modes analysis should take into account the topology of the devices and similar hardware-related mechanisms. The reliability of the design is finally verified by diagnostic and functional tests of the system under expected environmental conditions.

To achieve reliability in production of the computer, it is recommended that closely controlled manufacturing and assembly processes, together with quality workmanship and practices, should be required in all facets of the hardware development. Components should be selected for inherent reliability; they should use proven methods for their interconnection and packaging; and they should be subjected to rigid quality inspection and electrical checks. Highly recommended practices are: parts standardization, component derating with respect to both voltage levels and speed, and 100% screening of all parts and assemblies. Circuits should exhibit tolerance to transients. Adequate margins should be placed on the environmental and electrical characteristics, so that chances of failure at later stages of assembly are noticeably reduced. Extensive research is recommended to establish: the types of semiconductors to be used; the techniques for preparing and processing wafers; mask characteristics; failure probabilities for specific IC, MSI or LSI technologies; and the reliabilities of these devices in the space environment. Where more current data are unavailable, the use of MIL Handbook 217A or RADC Handbook, vol. 2, is recommended for failure-rate data.

The use of redundancy to enhance reliability should be considered, if necessary, to meet the mission specifications. The particular form of redundancy to be used will be a function of the time allowed for repair and may depend on the hardware technology with which the computer is to be mechanized. Both the method of redundancy and its level of application should be chosen carefully. The increased power supply and cooling requirements, and the effects of additional complexity on system checkout and personnel training, etc., should also be evaluated when various schemes are considered. Power switching should be considered for removing failed units from the system and for incorporating spares into it. The software should provide for recovery from specified malfunctions and for subsequent restart. If neither degraded performance nor computer failure can be tolerated, a means of repair should be provided. Inflight maintenance, i.e., manual replacement of failed modules, should be considered for future manned missions.

Self-check programs, error detecting codes, duplication and comparison, and/or voting among multiple units should be used in redundant systems to detect errors and/or malfunctions. It may be necessary to specify the possible configurations which the computer may assume in case of inflight failure of external or peripheral equipment, as well as internal subsystems. The additional reliability gained by such restructuring of the system should be studied both analytically and by means of simulation, and the software requirements for providing these alternatives should be considered early in the design. The response time for reconfiguring and restarting should be evaluated since even a temporary outage of the computer could be unacceptable during critical mission phases. Complicated redundant systems should be extensively tested to verify that the reconfiguration or fault-masking technique operates correctly in all cases and does not introduce errors. It is a good practice to use techniques whereby faults can be introduced artificially to insure that the system masks or reconfigures correctly. Combinations of these artificially introduced faults should be tried.

REFERENCES

1. Baechler, D. O.: Aerospace Computer Characteristics and Design Trends—Case 105-8. Memorandum B70 11043, Bellcomm, Inc., Washington, D. C., November 18, 1970 (published in January/February 1971 issue of Computer).
2. Frost, C. R.: Military CPUs. Datamation, vol. 16, no. 7, July 15, 1970, pp. 87-103.
3. Crisp, R.; Gilmore, J. P.; and Hopkins, A. L., Jr.: SIRU—A New Inertial System Concept for In-Flight Reliability and Maintainability. E-2407, MIT Instrumentation Laboratory, May 1969.
4. Gilmore, J. P.; and McKern, R. A.: A Redundant Strapdown Inertial System Mechanization—SIRU. AIAA Paper No. 70–1027, Presented at AIAA Guidance, Control, and Flight Mechanics Conference, Santa Barbara, Calif., August 17–19, 1970.
5. Dawson, W. F.; and Parke, N. G.: Development Tools for a Strapdown Guidance System. AIAA Paper No. 68-826, Presented at AIAA Guidance, Control, and Flight Dynamics Conference, Pasadena, Calif., August 12-14, 1968.
6. Matthews, J. B.; and Taylor, G. R.: Development Program for a General Purpose Computer and Strapdown Inertial Measurement Unit. AIAA Paper No. 69-850, Presented at AIAA Guidance, Control, and Flight Mechanics Conference, Princeton, N. J., August 18-20, 1969.
7. Baechler, D. O.; and Schaenman, P. S.: Functional Requirements for the Spaceborne Computer System of a Mid-70s Space Station. AAS Paper No. 69-581, Presented at AAS National Meeting, Las Cruces, N.M., October 23-25, 1969.
8. Martin, F. H.; and Battin, R. H.: Computer–Controlled Steering of the Apollo Spacecraft. J. Spacecraft and Rockets, vol. 5, no. 4, April 1968, pp. 400-407.
9. Vander Velde, W. E.: Space Vehicle Flight Control. Part 7 of Space Navigation, Guidance, and Control, J. E. Miller, ed., AGARDograph 105. Technivision Ltd., Maidenhead, England, 1966.
10. Freeman, N. G.; and Teets, P. B.: Titan 3 C Digital Flight Controls. AIAA Paper No. 69-878, Presented at AIAA Guidance, Control, and Flight Mechanics Conference, Princeton, N.J., August 18-20, 1969.
11. Matthews, S. W.: Improved Centaur Computer Operating Systems. AIAA Paper No. 69-943, Presented at AIAA Aerospace Computer Systems Conference, Los Angeles, Calif., September 8-10, 1969.
12. Anon.: STS Software Development (Study Task 5). E-2519, MIT Draper Laboratory, July 1970.
13. Gaddes, W. M.: Techniques for Developing Optimum Man/Computer Relationships in Aerospace Avionics Systems. In Air and Spaceborne Computers, E. Keonjian, ed., AGARDograph 127, Technivision Services, Slough, England, April 1970, pp. 57-72.
14. Hopkins, A. L., Jr.: Guidance Computer Design. Part 6 of Space Navigation, Guidance, and Control, J. E. Miller, ed., AGARDograph 105, Technivision Ltd., Maidenhead, England, 1966.
15. Anon.: Mercury/Gemini Program Design Survey. NASA CR-86041, McDonnell Douglas Astronautics Co., January 1968.
16. Anon.: Lunar Module/Abort Guidance System (LM/AGS) Design Survey. Report 06414-6008-T000, TRW Systems Group, September 1968.
17. Christensen, J. V.; and Kipping, E. D.: Mid-Course Navigation Using Statistical Filter Theory, A Manual. Theodolite and Symbolic Computer Control. NASA TN-D-3875, Ames Research Center, February 1967.

18. Stabler, E. P.; and Creveling, C. J.: Spacecraft Computers for Scientific Information Systems. Proceedings of the IEEE, vol. 54, no. 12, December 1966, pp. 1734-1742.

19. Cliff, R. A.: The SDP-3—A Computer Designed for Data Systems of Small Scientific Spacecraft. Presented at CNES Calculateurs Embarques Sur Fusees et Satellites Colloque International, Paris, France, December 3-6, 1968.

20. Schaefer, D. H.; and Snively, J. W., Jr.: On-Board Plasma Data Processor. The Review of Scientific Instruments, vol. 39, no. 4, April 1968, pp. 452-458.

21. Kuehn, R. E.: Computer Redundancy. Design, Performance, and Future. IEEE Transactions on Reliability, vol. R-18, no. 1, February 1969, pp. 3-11.

22. Styles, F.; Tharpe, M.; Taylor, T.; and Trevathan, C.: A General Purpose On-Board Processor for Scientific Spacecraft. NASA TM X-55843, Goddard Space Flight Center, July 1967.

23. Muller, R. M.: On-Board Processor 1—Philosophy, Design, and Operation. Presented at CNES Calculateurs Embarques Sur Fusees et Satellites Colloque International, Paris, France, December 3-6, 1968.

24. Baechler, D. O.: The Potential for Computer Use in Spaceborne Experiments, Case 730. TM-69-1031-1, Bellcomm, Inc., March 11, 1969.

25. Dickinson, M. M.; Jackson, J. B.; and Randa, G. C.: Saturn 5 Launch Vehicle Digital Computer and Data Adapter. Proceedings of the Fall Joint Computer Conference, 1964, pp. 501-516.

26. Copps, E. M., Jr.: Recovery From Transient Failures of the Apollo Guidance Computer. MIT E-2037, Presented at AIAA Guidance, Control, and Flight Dynamics Conference, Pasadena, Calif., August 12-14, 1968.

27. Lesniewski, R. J.: A Flexible LSI Spaceborne Data Processing System Concept. NASA TM X-61192, Goddard Space Flight Center, 1968.

28. Hopkins, A. L., Jr.: A New Standard for Information Processing Systems for Manned Space Flight. R-646, MIT Draper Laboratory, September 1969.

29. Hurst, S. R.; Shuck, R. J.; and Tanguy, J. S.: An Interplanetary Spacecraft Central Computer—When? AIAA Paper No. 68-840, Presented at AIAA Guidance, Control, and Flight Dynamics Conference, Pasadena, Calif., August 12-14, 1968.

30. Miller, J. S.; et al.: Final Report, Multiprocessor Computer System Study. Intermetrics, Inc., March 1970.

31. Daggett, D. H.; and Lee, R. Q.: F-111D Computer Complex. J. Aircraft, vol. 6, no. 5, September-October 1969, pp. 432-438.

32. Bersoff, E. H.; Hope, M. E.; and Tung, F. C.: Modular Computer Research. IEEE Transactions on Aerospace and Electronic Systems, vol. AES-6, no. 1, January 1970, pp. 29-36.

33. Wang, G. Y.: An In-House Experimental Air Space Multiprocessor—EXAM. ERC Memo KC-T-031, NASA Electronics Research Center, September 20, 1967.

34. Wood, P. E., Jr.: Interconnection of Processors and Memory in the Multiprocessor System. ERC Memo KC-T-041, NASA Electronics Research Center, February 5, 1968.

35. Wood, P. E., Jr.: Input/Output System for An Aerospace Multiprocessor. ERC Memo RC-T-062, NASA Electronics Research Center, May 19, 1969.

36. Anon.: Control, Guidance, and Navigation for Advanced Manned Missions. R-600, vol. 2, MIT Instrumentation Laboratory, January 1968.

37. Mallach, E.: Analysis of a Multiprocessor Guidance Computer. T-515, MIT Instrumentation Laboratory, June 1969.

38. Anon.: A Fault-Tolerant Information Processing System for Advanced Control, Guidance, and Navigation. R-659, MIT Draper Laboratory, May 1970.

39. Anon.: STS Data Management System Design (Task 2). E-2529, MIT Draper Laboratory, June 1970 (Preliminary Report).

40. Eastin, E. I.; et al.: MSFC Advanced Aerospace Computer. Report No. SP-232-0384, Sperry Rand Space Support Division, July 6, 1970.

41. Koczela, L. J.: Study of Spaceborne Multiprocessing—Phase II, vol. II—Technical Description. NASA CR-1159, North American Rockwell Autonetics Division, September 1968.

42. Anon.: Multiprocessing Bibliography. TR No. AU-T-13, Auburn University Digital Systems Laboratory, July 1970.

43. Williman, A.; Koczela, L.; and Burnett, G.: Multiprocessor Organization for Manned Mars Mission. Presented at Spaceborne Multiprocessing Seminar, Museum of Science, Boston, Mass., October 31, 1966.

44. Butsch, L. M.; et al.: A Distributed Aerospace Computer System. Presented at AICA-IFIP Conference on Hybrid Computation, Munich, Germany, September 2, 1970.

45. Anon.: Centaur in Retrospect. NASA CR-86040, Honeywell Aerospace Division, January 1968.

46. Patzer, W. J.: Structure of an Aerospace Computer Family. In Air and Spaceborne Computers, E. Keonjian, ed., AGARDograph 127, Technivision Services, Slough, England, April 1970, pp. 39-56.

47. Charney, H. R.; Lambert, D. W.; and Stanten, S. F.: Systems Design of a General Purpose Aerospace Computer. U67-4337, Raytheon Company, June 23, 1967.

48. Anon.: UNIVAC 1824 Computer Technical Description. PX 4620, UNIVAC Federal Systems Division, June 1969.

49. Anon.: Technical Manual for Titan 3 Missile Guidance Computer and Ground Support Equipment. PX 5671-1-2, UNIVAC Federal Systems Division, June 1970.

50. Joseph, E. C.: Evolving Digital Computer System Architectures. IEEE Computer Group News, vol. 2, no. 8, March 1969, pp. 2-8.

51. Patzer, W. J.; and Vandling, G. C.: Aerospace Systems Implications of Microprogramming. In Air and Spaceborne Computers, E. Keonjian, ed., AGARDograph 127, Technivision Services, Slough, England, April 1970, pp. 87-97.

52. Baechler, D. O.: Trends in Aerospace Digital Computer Design. IEEE Computer Group News, January 1969.

53. Anon.: Computer Overload Laid to Radar Mode. Aviation Week and Space Technology, August 4, 1969, pp. 87-88.

54. Yarosh, N. P.; Gregory, S. E.; Casey, E. J.; et al.: Interface Optimization Investigation. AFAL-TR-67-152, UNIVAC Federal Systems Division, February 1968.

55. Ricci, R. C.: Present and Future State of the Art in Guidance Computer Memories. NASA TN D-4224, Electronics Research Center, November 1967.

56. Kim, B. W.: Future Spaceborne Memories with 10^3–10^7 Bit Capacities. TM-68-1031-4, Bellcomm, Inc., July 26, 1968.

57. Anon.: UNIVAC 1832 Avionics Computer (Single and Multiprocessors). PX 5627A, UNIVAC Defense Systems Division, February 1968.

58. Kim, B. W.: The Plated Wire Memory and Its Potential for Aerospace Application—Case 105-8. B70 05046, Bellcomm, Inc., May 20, 1970.

59. Lamson, R.: A Plated Wire Memory System. R-551, MIT Instrumentation Laboratory, June 1966.

60. Smetzer, J.: Plated Wire Memory and Its Development at Aero-Florida. Honeywell Aerospace Division, 1970.

61. England, W. A.: Applications of Plated Wire to the Military and Space Environments. Presented at IEEE International Magnetics Conference, Washington, D. C., April 21-24, 1970.

62. Kosmala, A. L.; Green, J. P.; and Martin, F. H.: Final Report. Engineering Study for a Mass Memory System for Advanced Spacecrafts. Intermetrics, Inc., August 1970.

63. Blecher, F. H.: Magnetic Cylindrical Domain "Bubble" Devices. The Reflector (Published by Boston Section of the IEEE), vol. XIX, no. 1, September 1, 1970, pp. 10ff.

64. Anon.: Session 5—Bubbles, Bangles, and Ovonics (A Report on the Computer Elements Technical Committee Workshop, Litchfield Park, Arizona, January 1970). IEEE Computer Group News, vol. 3, no. 2, March/April 1970, p. 27.

65. Gratian, J. W.; and Freytag, R. W.: Ferroacoustic Memory. Presented at 1968 Government Microcircuit Applications Conference, Washington, D.C., October 1-3, 1968.

66. Bobeck, A. H.; et al.: Application of Orthoferrites to Domain Wall Devices. IEEE Transactions on Magnetics, vol. MAG-5, no. 3, September 1970, pp. 544-554.

67. Spignese, E. E.; Dawson, W. F.; and Edry, R. J.: Preview of a 1970 LSI Digital Computer. Presented at Northeast Electronics Research and Engineering Meeting—NEREM, November 5-7, 1969.

68. Collum, C. E.; and Kraus, A. D.: Packaging and Physical Design of Digital Electronics. A Space Odyssey for 1969. Honeywell Aerospace Division, 1969.

69. Anon.: Gemini Computer Large Amplitude Transient Susceptibility Test Report. IBM No. 65–565-213, IBM Federal Systems Division, Owego, N.Y., Sept. 1964.

70. Anon.: Atlas Series E/F Computer Liftoff and Staging Problems (CLASP) Final Summary Report. Division Report No. GDA-APZ64-070, General Dynamics/Astronautics, December 1964.

71. Avizienis, A.: Design Methods for Fault-Tolerant Navigation Computers. TR 32-1409, Jet Propulsion Laboratory, October 15, 1969.

72. Anon.: Skylab A ATM Digital Computer Program Requirements Document (PRD). S&E-ASTR-SG 50M-37941, NASA Marshall Space Flight Center, July 1, 1970.

73. Avizienis, A.: An Experimental Self-Repairing Computer. TR 32-1356, Jet Propulsion Laboratory, August 1968 (Reprinted from Preprint Booklet E (Hardware 2) of the 1968 Congress of the International Federation for Information Processing, Edinburgh, Scotland, August 1968).

74. Avizienis, A.; et al.: Automatic Maintenance of Aerospace Computers and Spacecraft Information and Control Systems. AIAA Paper No. 69-966, Presented at AIAA Aerospace Computer Systems Conference, Los Angeles, Calif., September 8-10, 1969.

75. Avizienis, A.: The STAR Computer: A Self-Testing-and-Repairing Computer for Spacecraft Guidance, Control, and Automatic Maintenance. Presented at 10th Meeting of AGARD Guidance and Control Panel, London, England, June 2-5, 1970.

76. Ball, M.; and Hardie, F.: IBM Proposes Triple-Redundancy Aerospace Computer. Computer Design, November 1967, pp. 34-36.

77. Ball M.; and Hardie, F.: Redundancy for Better Maintenance of Computer Systems. Computer Design, January 1969, pp. 50-52.

78. Ball, M.; and Hardie, F. H.: Self-Repair in a TMR Computer. Computer Design, February 1969, pp. 54-57.

79. Mathur, F. P.; and Avizienis, A.: Reliability Analysis and Architecture of a Hybrid-Redundant Digital System: Generalized Triple Modular Redundancy with Self-Repair. AFIPS—Conference Proceedings, volume 36, AFIPS Press, Montvale, N. J., 1970, pp. 375-383.

80. Widnall, W. S.; Glick, F. K.; Henize, J., and Drane, L. W.: The Digital Simulation for the Verification of Program SUNBURST (Unmanned LM, AS-206). E-2146, MIT Instrumentation Laboratory, July 1967.

81. Ferrier, B. E., Jr.: Operational Flight Program Validation Plan for Missiles. J. Spacecraft and Rockets, vol. 6, no. 5, May 1969, pp. 628 - 630.

82. Keonjian, E., ed.: Air and Spaceborne Computers. AGARDograph 127, Technivision Services, Slough, England, April 1970, pp. 173-175.

83. Anon.: Apollo Configuration Management Manual. NHB 8040.2 (Formerly NPC 500-1), NASA Office of Manned Space Flight, January 1970.

84. Anderson, M. D.; and Marek, V. J.: Evaluation of Aerospace Computer Architecture. AIAA Paper No. 68-836, Presented at AIAA Guidance, Control and Flight Dynamics Conference, Pasadena, Calif., August 12-14, 1968.

85. Williman, A.; Koczela, L.; and Burnett, G.: Multiprocessor Organization for Manned Mars Mission. Presented at Spaceborne Multiprocessing Seminar, Museum of Science, Boston, Mass., October 31, 1966.

GLOSSARY

–A–

Access Time. Generally, the time interval between a request for the content of a location in a memory device and the delivery of this information (read operation); also time between command to store data in a memory location and the completion of the storage (write operation). Access time is thus the sum of the waiting time and the transfer time. For disc and drum storage, the access time depends upon the location specified; for random access storage devices, the access time is essentially constant.

Accumulator. A register (or registers) and associated equipment in the arithmetic unit of a computer in which are formed the results of various arithmetic and logical operations, such as addition, subtraction (complementing), and shifting (multiplication).

Accuracy. The degree of freedom from error of a quantity, as distinguished from precision.

A/D. Abbreviation for analog/digital.

Address. A specific location in the computer (usually a memory location) where data or instructions are stored.

Address Modification. The changing of the address portion of a computer word before the instruction is executed by the use of index registers, indirect addressing, or other techniques.

AEA. Abbreviation for abort electronics assembly, the digital computer in the backup guidance system for the Apollo lunar module.

AGC. Abbreviation for Apollo guidance computer.

Alphanumeric. A symbolic code that contains both alphabetic characters (letters) and numeric characters (digits). Alphanumeric codes generally include additional special characters such as commas, periods, ampersand, mathematical characters, etc.

Analog/Digital (A/D) Converter. A device for converting an electrical analog input signal to a corresponding digital data word.

Architecture. The conceptual and functional structure of the computer system, excluding the equipment internal organization and detailed implementation. Multiple implementations are possible for most architectural specifications.

Arithmetic Unit. The portion of the computer that performs the arithmetic and logical operations.

Assembly Language. A computer language which permits the writing of symbolic addresses (such as X or A1) for absolute binary addresses (such as 01100 or 11010) and also the writing of symbolic operation codes (such as ADD or SUB) instead of binary machine operation codes (such as 111 or 011). One assembly language statement normally translates into one machine instruction.

Assembly Program. A computer program which translates a program written in an assembly language into a machine language program.

ATM. Abbreviation for Apollo telescope mount, also referred to as Skylab.

Availability. Probability that the computer is functioning properly during specific time periods. Availability depends on the time to detect faults and repair or reconfigure as well as MTBF.

–B–

Base. See Radix.

Base Register. A register containing an address which is modified by the contents of an address field in the instruction to determine the effective address. It is also used to retain linkage addresses for subroutine entry and return. A base register generally contains a complete address that is modified by a displacement location in the instruction (cf., Index Register).

BCD. Abbreviation for binary coded decimal.

Binary. Pertains to a number system based on the radix 2. Two symbols are used, usually 1 and 0.

Binary Coded Decimal (BCD). Pertains to an encoding technique whereby each of the decimal digits 0 through 9 is represented by a unique group of binary digits.

Bit. Abbreviation for binary digit. A bit is a single character in a binary number and has the value 0 or 1.

Buffer. A temporary storage device used to make possible transfer between two devices whose input and output speeds are not matched.

Bus. A common path for transfer of information between several sources and several destinations.

Byte. A group of binary digits which are handled as a unit. Generally, a word is composed of an integral number of bytes.

–C–

Centralized. Refers to a computer system organization in which all computational tasks are performed by a general-purpose computer. (cf., Dedicated).

Channel. Usually a transfer path for specific I/O data. Also, applies to tape recording of information on a single track.

Chip. A single monolithic semiconductor element. A chip may be in the form of an IC, MSI or LSI device or a transistor.

Code. An arrangement of basic symbols to convey a system of notation for example, to use binary 1 and 0 symbols for generating computer codes such as BCD, octal, etc.

Compiler. A computer program which translates a compiler language program into a machine language program.

Compiler Language. A procedure-oriented computer programming language such as FORTRAN, JOVIAL, PL/1 or SPL. A single compiler language statement is generally translated into several machine instructions.

Complement. A number or quantity that is derived from another number or quantity by subtraction in accordance with special rules. Complements are used in computers to represent negative numbers and to perform subtraction by addition.

Computer. A machine which is able to perform sequences of arithmetic and logical operations upon information. A digital computer uses integers to express all variables and quantities of a problem, whereas an analog computer calculates by using physical analogs of the variables. In the latter, a one-to-one correspondence exists between each numerical quantity occurring in the problem and a varying physical measurement (e.g., voltage level).

Control Unit. A major functional unit of a computer that is the traffic controller of the system. It produces timing, control, and command signals for execution of the computer program. It causes all elements to function together as an integrated system.

Core Memory. A memory device consisting of an array of ferromagnetic cores. Generally, each core stores a single binary digit; the direction of magnetic polarization determines whether the bit is a 0 or 1.

Coverage. The probability that a failure will be detected and circumvented. Coverage is related to but distinguished from availability, which is often calculated on the basis of the number of spares available. Spares are useless unless 1) improper operation of the system can be detected, 2) the failed unit can be switched out to effect the repair, and 3) the system can be restarted to continue the mission satisfactorily.

Cycle Stealing. A technique for transferring data between an I/O device and a read-write memory without interrupting the program in control. The data are transferred to or from the memory during a cycle taken (stolen) from the normal programmed sequence. The program continues from where it was before the cycle was stolen.

Cycle Time. The time interval required to perform a complete "read" cycle for a memory, or the minimum time interval between the starts of successive accesses to a storage location.

–D–

D/A. Abbreviation for digital/analog.

Data Memory. See Variable Memory.

Data Word. A computer word containing an ordered set of bits used to represent a data quantity.

Dedicated. Refers to a computer system organization consisting of multiple computers, each of which is permanently assigned to a single computational function. The individual computers are usually small, special-purpose devices. (cf., Centralized)

Destructive Read Out (DRO). Refers to memories in which the read process destroys the information in the storage medium. If the information is to be retained, it must be temporarily stored in an external register (the memory buffer register) and then rewritten into the memory.

Diagnostic Routine. A routine or special program designed to check out computer operations. These programs usually isolate and indicate malfunctioning areas of a computer, and designate the specific faults. Machine diagnostic programs check the computer itself, and program diagnostics verify the software.

Digital. Representation of a quantity using digits or discrete steps.

Digital/Analog (D/A) Converter. A device for converting a digital value to a corresponding electrical analog output signal.

Discretes. Output control levels generated by the computer or input control levels interpreted by the computer. Discretes are constrained to binary values (i.e., 0 or 1), and generally indicate the occurrence of a specific event in the system or the state of some particular device or switch.

Disc Storage. A form of magnetic storage in which the information is stored in concentric circular tracks located on the surface of a rotating disc.

Direct Memory Access. Refers to a type of I/O channel which permits data transfer directly between memory and external devices under external device control.

Double Precision. Refers to the use of two data words to represent a single number, thereby gaining increased precision.

Drum Storage. A form of magnetic storage in which the information is stored in adjacent circular tracks located on the circumference of a rotating cylindrical drum.

Dynamic Redundancy. A hardware redundancy technique requiring two consecutive actions: 1) the presence of a fault is detected; 2) a recovery action either eliminates the fault or corrects the error which was caused. The redundancy is usually introduced in a selective, rather than massive, fashion.

–E–

EMI. Abbreviation for electromagnetic interference.

Error. A miscalculation in the program being executed by the computer; either an instruction is not executed correctly or an incorrect result is computed. Both types of errors may be caused at once by some faults.

Error Detecting Codes. Codes wherein data words contain additional checking bits to allow detection of errors that occur in data handling, and often to determine which bit is in error. Many coding techniques for adding redundancy digits are in use and differ in their ability to detect and/or correct multiple errors.

Executive. A supervisory program which allocates the processor resources between programs and controls the peripherals to be employed for a specific program.

–F–

Failure. A computer malfunction caused by component failure or degradation. Failures are considered "hard" if the malfunction is continuous or "intermittent" if it only occurs occasionally. An intermittent failure is typified by a soldered joint which opens momentarily under vibration and temperature stress.

Fault. The deviation of a logic variable from its prescribed value. Most faults will cause an error. A transient fault is a temporary logic deviation caused by an intermittent component failure or by external interference (e.g., power supply irregularities or EMI). A permanent fault is the result of a hard component failure.

Fault Tolerance. Ability of a computer to execute error-free programs in the presence of a fault. Fault tolerance in digital computers is achieved by means of protective redundancy, and must be qualified by the classes of faults that are tolerated and the parts of the computer in which they may occur.

Film Memories. Memories that store information by segmented polarization of a thin film of ferromagnetic material, which may be either flat or tubular in shape. Film memories allow an improvement in speed over core memories which are limited by the geometry of the core during flux reversals and the size required to thread wires through the holes in the core.

Fixed Memory. See "Read-Only" Memory.

Fixed-Point. Referring to the representation of a number by a single set of digits with a constant implied location of the radix point.

Floating-Point. Referring to the representation of a number by two sets of numbers, one containing the mantissa and the other the location of the radix point.

Functional Test. A test designed to check out the operation of the computer hardware.

–G–

Gate. A circuit which has the ability to produce an output that is dependent upon a logical function of the inputs; e.g., an AND gate has an output when all inputs assume a logical ONE or TRUE state.

G & N. Abbreviation for guidance and navigation.

–H–

Hexadecimal. Pertains to a number system with a radix of 16. The hexadecimal system is convenient for compactly representing a binary number by dividing it into 4-bit bytes.

–I–

IC. Abbreviation for integrated circuit.

Index Register. A separate register whose contents is used to modify an explicitly specified address without changing the program in memory. It generally contains a count which is added to the address in the instruction itself or in the instruction plus an extension register, as distinguished from a base register which generally contains a complete address. Index registers are often used to provide loop control in iterative programs and to designate return addresses at the conclusion of subroutines.

Indirect Addressing. Designating an address that contains the location of the desired operand.

Input/Output (I/O). All information transmitted between the computer and its interfacing systems.

Instruction. A set of characters in a computer that specifies an operation to be performed by the computer and usually the location or value of some of the operands and/or results.

Instruction Repertoire. The set of instructions which can be performed by a particular computer.

Instruction Word. A computer word containing an instruction.

Integrated Circuit (IC). An electronic circuit which is fabricated in an integrated process and which is capable of performing the functions of a conventional circuit composed of discrete components such as transistors, resistors, diodes, etc.

Interface. The matching circuitry required to allow the transmission of data between two devices.

Interrupt. An externally or internally generated signal that interrupts the current sequence of the program being performed and causes a new sequence to be performed.

I/O. Abbreviation for input/output.

–L–

Large Scale Integration (LSI). The fabrication of more than 100 integrated logic gates together in one assembly.

Logic Levels. The nominal voltage levels which are used to represent binary 0 or 1 in logic circuits. For instance, in commercial resistor-transistor-logic (RTL) circuits, the levels are generally 0 and +2.5 V; in diode-transistor-logic (DTL) circuits, they are generally 0 and +3.5 V.

LSI. Abbreviation for large scale integration.

LVDC. Abbreviation for launch vehicle digital computer on the Saturn 1B and 5 boosters.

–M–

Machine Language. Coded instructions in binary digit form for use in the computer.

Magnetic Core. A ferromagnetic ring or core used to store a bit of data.

Masking. Use of static redundancy to cover up an error. Also denotes the selection of particular bits of a computer word.

Mean Time Between Failures (MTBF). Reciprocal of the average rate of failure of a piece of equipment. MTBF is a frequently used measure of a computer's reliability, but to be meaningful a quoted MTBF must be accompanied by a full description of the assumptions and conditions used in the calculation.

Medium Scale Integration (MSI). The fabrication of 25 to 100 integrated logic gates together in one assembly.

Memory Protect. The technique for sensing potential power failures and preventing the loss of data in the memory and in critical registers.

Memory Word. An ordered set of bits in the computer's primary storage device. It can contain either data words or instruction words.

MOS. Abbreviation for metal-oxide-semiconductor.

MOSFET. Abbreviation for metal-oxide-semiconductor-field-effect-transistor.

MSI. Abbreviation for medium scale integration.

MTBF. Abbreviation for mean time between failures.

Multicomputer. A computer configuration having two or more sets of memories and processors, which are generally assigned different tasks. A multicomputer is distinguished from a multiprocessor in that the processors do not share memories.

Multilayer. Printed circuit board construction technique whereby the wiring is in the form of etched lines and where many layers make up the complete board unit. Interconnection between layers is performed normally by plated-through holes.

Multiplexed Data. Data from several devices which have been combined to be transmitted through a single channel, either by interleaving them or by sampling them in sequential order.

Multiprocessor. A computer configuration having two or more processors which share memory or memories and I/O.

–N–

NDRO. Abbreviation for nondestructive readout.

Nondestructive readout (NDRO). Refers to memories from which information can be read without destruction of any word in the storage device.

Nonvolatile. Refers to memories which do not lose their information contents when power is removed.

–O–

OAO. Abbreviation for an orbiting astronomical observatory.

OBP. Abbreviation for onboard processor, the digital computer scheduled to be flown on the OAO-B.

Octal. Pertains to a number system with a radix of 8. The octal system is convenient for compactly representing a binary number by dividing it into 3-bit bytes.

One's Complement. The complement of a binary number obtained by changing each 1 to 0 and each 0 to 1. An alternate method is to subtract the number from all ones.

Operand. Data used in an operation.

Operation. The arithmetic, logic, or transfer action that the computer performs as a result of interpreting a single instruction.

Overflow. A condition that occurs when a computation produces an answer that exceeds the storage capacity of a register.

–P–

Parallel. Refers to the simultaneous transmission and/or processing of all bits of a word via a separate line or channel for each bit.

Parity Check. A method for checking the validity of a binary word or byte, usually by summing the ONE bits. The check is usually made by use of a parity bit suffixed to the original word, which indicates whether the sum is odd or even. This technique can be used to generate more complete checks.

Peripheral. Refers to equipment external to the computer but directly associated with the I/O section, such as magnetic tape transports or paper tape punches and readers.

Plated-Wire. A type of film memory where information is stored in a thin magnetic film deposited over the bit wire.

Precision. The degree of discrimination with which a data quantity is represented. For instance, a two-digit decimal number discriminates among 100 possible values. Precision should be distinguished from accuracy.

Processor. That portion of a computer that consists of control and arithmetic units. The basic I/O interface is often included.

Program. A sequence of instructions and necessary numerical constants that will cause the computer to operate on a given problem.

Program Memory. That portion of the computer memory which is used for storage of program instructions and constants. It may be either "read-only" or read-write, as distinguished from the variable memory which is always read-write.

–R–

Radix. The base number of a number system. Example: 2 in binary, 10 in decimal, 8 in octal, 16 in hexadecimal, etc.

Random Access. Refers to a storage device in which the time necessary to "access" any memory location is constant and independent of the relative locations of the last addressed location and the next location to be addressed. Magnetic core memories have this characteristic while drum memories do not. In the latter, physical location influences the amount of delay of access.

Read. To sense and transfer information contained in memory to another storage or operating register of the computer.

Read-Only. Refers to a memory device that outputs a selectable word, but this word cannot be altered by the processor during its operation.

Read-Write. Refers to a memory device which can be both read from and written into during normal operation.

Reconfiguration. Reorganization of the computer into a new system without the failed part. Sometimes the computing capacity of the system is reduced by reconfiguration, and the fault is thus only partially tolerated. Partial fault tolerance is sometimes referred to as graceful degradation.

Recovery. The actions necessary to maintain information continuity in a computer system following a transient error or reconfiguration.

Redundancy. The use of additional circuits and/or components, that would not be needed in a "perfect" computer, but which serve to provide fault tolerance in a real computer. The two distinct approaches are static redundancy and dynamic redundancy.

Register. A device for temporarily storing a single word in preparation for operating on it. It may store data, instructions, memory addresses, or any other ordered set of bits. Usually it can be loaded or emptied very quickly.

Relative Address. An address indicated by an incremental change from the last address rather than an absolute location.

Reliability. The probability that a piece of equipment or system will perform as specified for a given period of time when used in the specified manner.

Repertoire. See Instruction Repertoire.

—S—

Scaling. Multiplying variables by an appropriate constant (the scale factor) to allow their representation in a given fixed-point numerical system.

Scratchpad Memory. A high-speed (generally small) memory which can be directly addressed by the processing circuitry.

Self Test. A test exercised by the computer itself, designed to check its logical operation.

Serial. Refers to the sequential transmission and/or processing of the bits of a word through a single line or channel.

Serial-Parallel. Refers to the simultaneous transmission and/or processing of bits in a byte through parallel lines or channels. The transmission or processing of a complete word requires several sequential bytes. This type of data flow is a combination of serial and parallel operations.

Shift. Moving the characters of a unit of information column-wise right or left from one storage cell to another, usually in a register. For a number, this is equivalent to multiplying or dividing by a power of the base of notation.

Single Address. Signifies that only one address is contained in each instruction.

Simulation. The process of modeling or logically duplicating a system (including another computer) by programming its features on a general-purpose computer.

Software. All of the computer programs written for use in a computer system. *Support* software includes assembly programs, compilers, utility routines, etc. *Operational* software refers to flight programs, test programs, etc.

S & C. Abbreviation for stabilization and control.

Static Redundancy. The use of massive replication of each component or circuit to two or more copies, which are permanently connected and powered. A component failure or logic fault is instantaneously and automatically masked by the presence of the redundant copies of the same item.

Subroutine. A subprogram that can be part of another routine. Subroutines can be closed, which means they are stored in one place and accessed by other programs when needed, or open which means they are inserted each time they are used.

–T–

Temporary Memory. The part of memory which contains data to be processed or computational results. Sometimes called the data memory, it provides read-write storage.

Throughput. The total flow of useful information through a computer during some given period of time.

TMR. Abbreviation for triplicated modular redundancy, a type of static redundancy.

Translate. To convert one type of language (special codes, other machine languages, etc.) to another language suitable for operations within the computer.

TTL. Abbreviation for transistor-transistor logic.

Two Address. Signifies that two addresses are contained in each instruction. For example, the address of one operand and the address of the next instruction.

Two's Complement. The complement of a binary number found by changing each 1 to 0 and each 0 to 1 and adding 1 to the number.

–V–

Variable Memory. That portion of the computer memory which is used for storage of temporary data or computational results. It is always a read-write memory, as distinguished from the Program Memory which is often "read-only."

Volatile. Refers to a memory which loses its information contents when power is removed.

–W–

WCIU. Abbreviation for workshop computer interface unit, which connects the two ATM workshop computers.

Word. A series of bits of prescribed length which is treated by the computer circuits as a unit. The word is the basic format in which the computer transmits information. Ordinarily a word is treated by the control unit as an instruction and by the arithmetic unit as a data quantity.

Word Length. The number of bits in a word. Word lengths may be fixed or variable depending on the particular computer.

Write. To store information into the system memory from the input data or from an operating register of the system.

NASA SPACE VEHICLE DESIGN CRITERIA MONOGRAPHS ISSUED TO DATE

SP-8001 (Structures)	Buffeting During Atmospheric Ascent, revised November 1970
SP-8002 (Structures)	Flight-Loads Measurements During Launch and Exit, December 1964
SP-8003 (Structures)	Flutter, Buzz, and Divergence, July 1964
SP-8004 (Structures)	Panel Flutter, July 1965
SP-8005 (Environment)	Solar Electromagnetic Radiation, revised May 1971
SP-8006 (Structures)	Local Steady Aerodynamic Loads During Launch and Exit, May 1965
SP-8007 (Structures)	Buckling of Thin-Walled Circular Cylinders, revised August 1968
SP-8008 (Structures)	Prelaunch Ground Wind Loads, November 1965
SP-8009 (Structures)	Propellant Slosh Loads, August 1968
SP-8010 (Environment)	Models of Mars Atmosphere (1967), May 1968
SP-8011 (Environment)	Models of Venus Atmosphere (1968), December 1968
SP-8012 (Structures)	Natural Vibration Modal Analysis, September 1968
SP-8013 (Environment)	Meteoroid Environment Model–1969 (Near Earth to Lunar Surface), March 1969
SP-8014 (Structures)	Entry Thermal Protection, August 1968
SP-8015 (Guidance and Control)	Guidance and Navigation for Entry Vehicles, November 1968
SP-8016 (Guidance and Control)	Effects of Structural Flexibility on Spacecraft Control Systems, April 1969
SP-8017 (Environment)	Magnetic Fields–Earth and Extraterrestrial, March 1969
SP-8018 (Guidance and Control)	Spacecraft Magnetic Torques, March 1969
SP-8019 (Structures)	Buckling of Thin-Walled Truncated Cones, September 1968
SP-8020 (Environment)	Mars Surface Models (1968), May 1969
SP-8021 (Environment)	Models of Earth's Atmosphere (120 to 1000 km), May 1969
SP-8022 (Structures)	Staging Loads, February 1969
SP-8023 (Environment)	Lunar Surface Models, May 1969
SP-8024 (Guidance and Control)	Spacecraft Gravitational Torques, May 1969
SP-8025 (Chemical Propulsion)	Solid Rocket Motor Metal Cases, April 1970

SP-8026	(Guidance and Control)	Spacecraft Star Trackers, July 1970
SP-8027	(Guidance and Control)	Spacecraft Radiation Torques, October 1969
SP-8028	(Guidance and Control)	Entry Vehicle Control, November 1969
SP-8029	(Structures)	Aerodynamic and Rocket-Exhaust Heating During Launch and Ascent, May 1969
SP-8030	(Structures)	Transient Loads From Thrust Excitation, February 1969
SP-8031	(Structures)	Slosh Suppression, May 1969
SP-8032	(Structures)	Buckling of Thin-Walled Doubly Curved Shells, August 1969
SP-8033	(Guidance and Control)	Spacecraft Earth Horizon Sensors, December 1969
SP-8034	(Guidance and Control)	Spacecraft Mass Expulsion Torques, December 1969
SP-8035	(Structures)	Wind Loads During Ascent, June 1970
SP-8036	(Guidance and Control)	Effects of Structural Flexibility on Launch Vehicle Control Systems, February 1970
SP-8037	(Environment)	Assessment and Control of Spacecraft Magnetic Fields, September 1970
SP-8038	(Environment)	Meteoroid Environment Model–1970 (Interplanetary and Planetary), October 1970
SP-8040	(Structures)	Fracture Control of Metallic Pressure Vessels, May 1970
SP-8041		Captive-Fired Testing of Solid Rocket Motors, March 1971
SP-8042	(Structures)	Meteoroid Damage Assessment, May 1970
SP-8043	(Structures)	Design Development Testing, May 1970
SP-8044	(Structures)	Qualification Testing, May 1970
SP-8046	(Structures)	Landing Impact Attenuation for Non-Surface-Planing Landers, April 1970
SP-8047	(Guidance and Control)	Spacecraft Sun Sensors, June 1970
SP-8048		Liquid Rocket Engine Turbopump Bearings, March 1971
SP-8049	(Environment)	The Earth's Ionosphere, March 1971
SP-8050	(Structures)	Structural Vibration Prediction, June 1970
SP-8051		Solid Rocket Motor Igniters, March 1971
SP-8053	(Structures)	Nuclear and Space Radiation Effects on Materials, June 1970
SP-8054	(Structures)	Space Radiation Protection, June 1970
SP-8055	(Structures)	Prevention of Coupled Structure-Propulsion Instability (Pogo), October 1970
SP-8056	(Structures)	Flight Separation Mechanisms, October 1970

SP-8057 (Structures)	Structural Design Criteria Applicable to a Space Shuttle, November 1970
SP-8058 (Guidance and Control)	Spacecraft Aerodynamic Torques, January 1971
SP-8059 (Guidance and Control)	Spacecraft Attitude Control During Thrusting Maneuvers, February 1971
SP-8060 (Structures)	Compartment Venting, November 1970
SP-8061 (Structures)	Interaction With Umbilicals and Launch Stand, August 1970
SP-8065 (Guidance and Control)	Tubular Spacecraft Booms (Extendible, Reel Stored), February 1971
SP-8071 (Guidance and Control)	Passive Gravity-Gradient Libration Dampers, February 1971