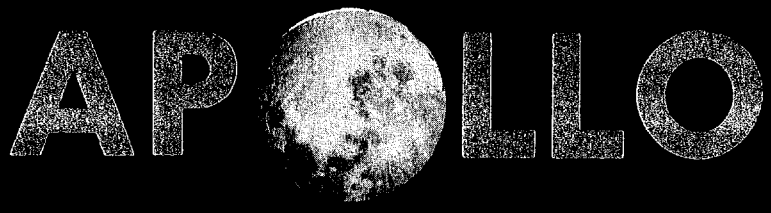

E-1758

ORGANIZATION OF COMPUTATION AND CONTROL IN THE APOLLO GUIDANCE COMPUTER

by

T. J. Lawton and C. A. Muntz

MIT

CAMBRIDGE 39, MASSACHUSETTS

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

APOLLO

GUIDANCE AND NAVIGATION

Approved: Milton B. Trageser Date: 4/7/65
MILTON B. TRAGESER, DIRECTOR
APOLLO GUIDANCE AND NAVIGATION PROGRAM

Approved: Roger B. Woodbury Date: 4/9/65
ROGER B. WOODBURY, DEPUTY DIRECTOR
INSTRUMENTATION LABORATORY

E-1758

ORGANIZATION OF COMPUTATION AND CONTROL IN THE APOLLO GUIDANCE COMPUTER

by

T. J. Lawton and C. A. Muntz

[redacted] INSTRUMENTATION LABORATORY

CAMBRIDGE *39*, MASSACHUSETTS

COPY #209

REPRINT #1
(BEGINS WITH COPY #201)

ACKNOWLEDGEMENT

This report was prepared under DSR Project 55-238, sponsored by the Manned Spacecraft Center of the National Aeronautics and Space Administration through Contract NAS 9-4065.

E-1758

ORGANIZATION OF COMPUTATION AND CONTROL IN THE APOLLO GUIDANCE COMPUTER

ABSTRACT

The digital guidance computer is the central control element in the Apollo Guidance and Navigation System. The difficulties in preparing computer programs for 'the complex Apollo missions are aggravated by the need to keep the computer hardware to a minimum. This paper demonstrates how an effective solution to these problems is achieved with a combination of appropriate programming techniques and computer design.

by T. J. Lawton and C.A. Muntz

April 1965

ORGANIZATION OF COMPUTATION AND CONTROL IN THE APOLLO GUIDANCE COMPUTER

by

T.J. Lawton and C.A. Muntz
Staff Members
Instrumentation Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

ABSTRACT

The digital guidance computer is the central control element in the Apollo Guidance and Navigation System. The difficulties in preparing computer programs for the complex Apollo missions are aggravated by the need to keep the computer hardware to a minimum. This paper demonstrates how an effective solution to these problems is achieved with a combination of appropriate programming techniques and computer design.

INTRODUCTION

The goal of the Apollo program is to place two men on the moon and return them safely to earth in this decade[1]. An important part of this program is the onboard guidance and navigation system[2]. It is the function of this system to maintain continual knowledge of position and velocity and to use this information to steer the vehicle during non free-fall portions of the mission. The guidance and navigation functions during a lunar landing mission are depicted in Fig. 1.

The major components of this system are:

1. The Inertial Measurement Unit
2. The Space Sextant and Telescope
3. The Display and Controls Unit
4. The Digital Guidance Computer

The guidance computer[3] plays the central role in receiving sensor inputs, processing them, and transmitting resulting commands to various control systems within and outside the guidance and navigation system. Figure 2 shows the devices which communicate with the computer.

Because the computer must be carried aboard the spacecraft, the primary design goals are low weight, volume, and power consumption and high reliability. In order to achieve these objectives it was necessary to limit the word length, operating speed, and instruction set (Fig. 3)[4]. To offset these apparent limitations, a unique high density memory[5] has been incorporated which provides a large program storage capacity (Figs. 4 and 5).

By appropriate programming techniques, together with an involuntary interrupt feature and unusual input/output techniques, it is possible to achieve the hardware design goals and still meet the system requirements.

EXAMPLE OF COMPUTER REQUIREMENTS

To illustrate the diversity of requirements imposed on the computer by the guidance and navigation functions, a specific phase of the mission will be examined - control of the vehicle during thrusting. This activity may be considered in three parts: computation of the required guidance and navigation information, communication with the guidance and spacecraft control system, and monitoring the performance of the computer in the overall control loop.

The information flow for this task is depicted in Fig. 6. Using present position and velocity, together with a desired terminal condition, the computer calculates an impulsive velocity-to-be-gained and sends corresponding steering signals to the spacecraft control system. This results in vehicle motion which is fed back into the guidance computer via the inertial sensors. The cycle is repeated until the magnitude of the velocity-to-be-gained vector is driven to zero[6].

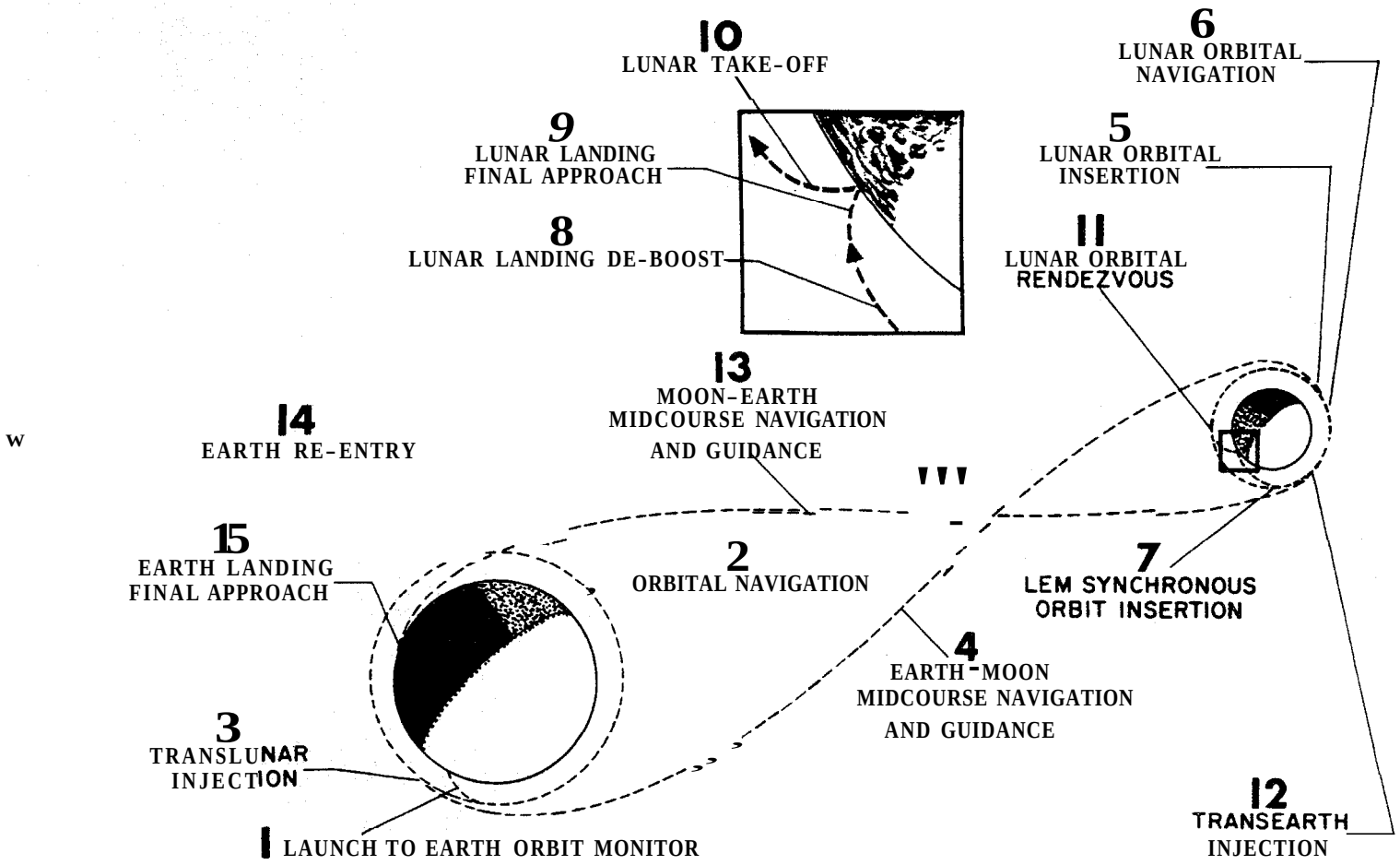

Fig. I Mission Phase Summary

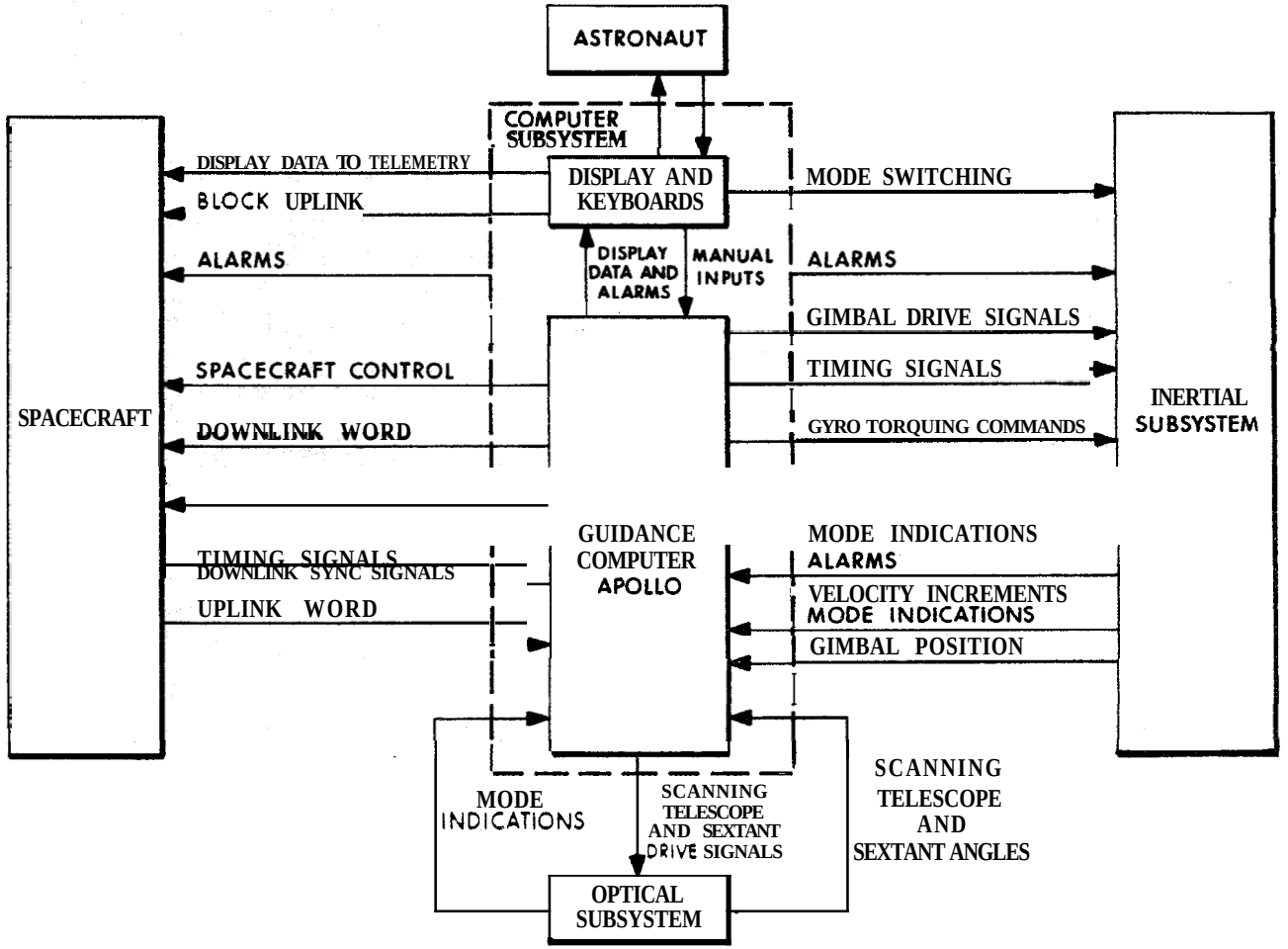

Fig. 2 Guidance Computer Interfaces

Arithmetic - Parallel, Binary, 1's Complement, Fixed Point.

Word Length - 16 bits - 15 bits and parity.

As Data = sign and 14 bits
As Instruction = 3 bit op. code and 12 bit address.

Memory Organization - High Speed Special Registers
Erasable Storage, including Counters-coincident current type.
Program Storage - read-only, core rope type.

Memory Quantity and Addressing - High Speed - order of 10^1, most directly addressable.
Erasable - order of 10^3, all directly addressable.
Program - order of 10^4, expansion capability practically limitless;
2000 directly addressable, remainder addressable
1000 words at a time in conjunction with Bank Register.

instruction - 11 Total
Arithmetic - Add, Subtract, Multiply, Divide
Information Transmission - Exchange, Transfer to Storage, Clear and Subtract.
Logical - Mask
Decision Making - Count, Compare, and Skip (I instruction)
Sequence Changing - Transfer Control
Indexing - Index Instruction

Fig. 3 Guidance Computer Characteristics

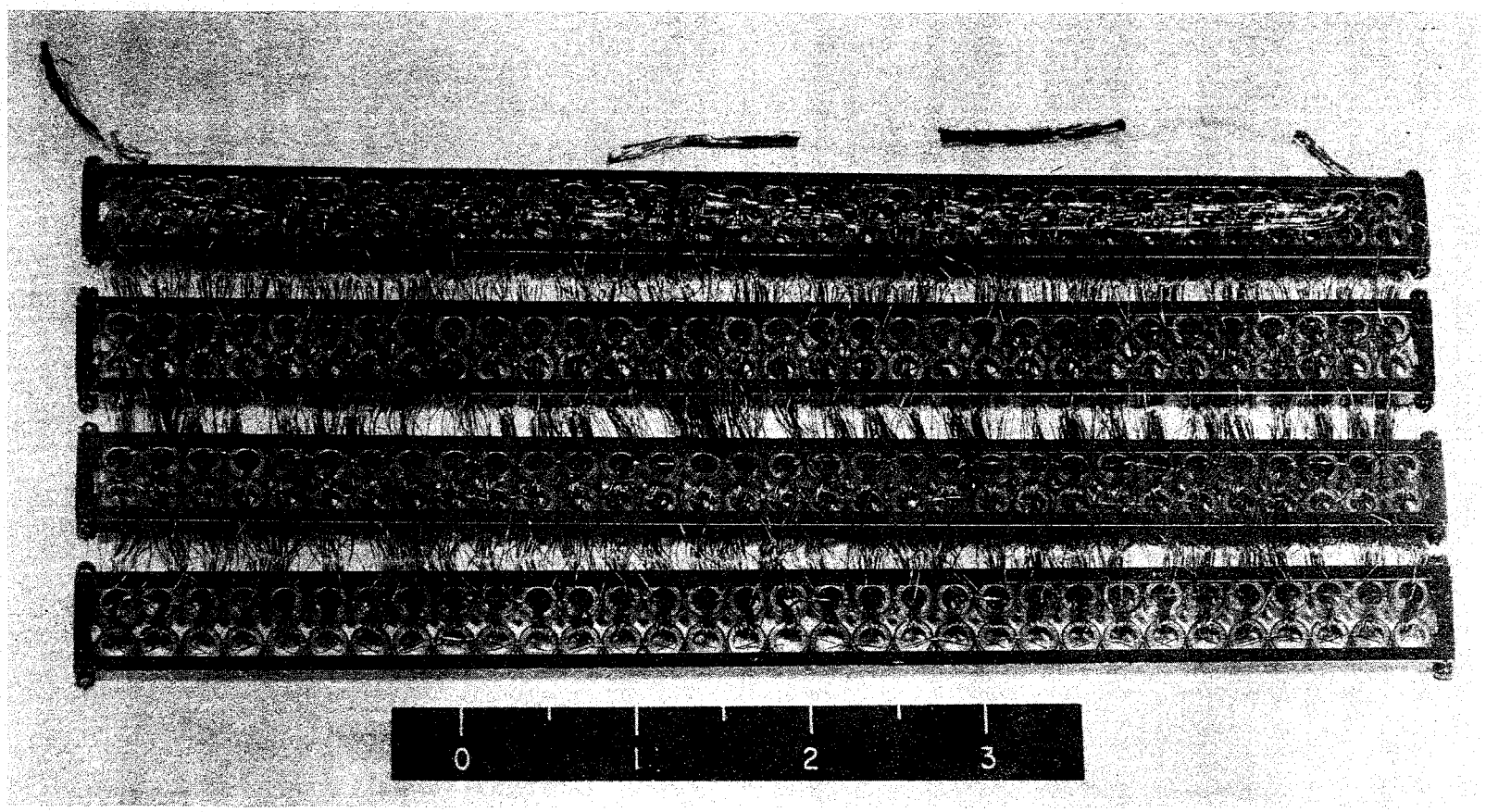

Fig. 4 Experimental 256 Core Rope (Made by Raytheon).

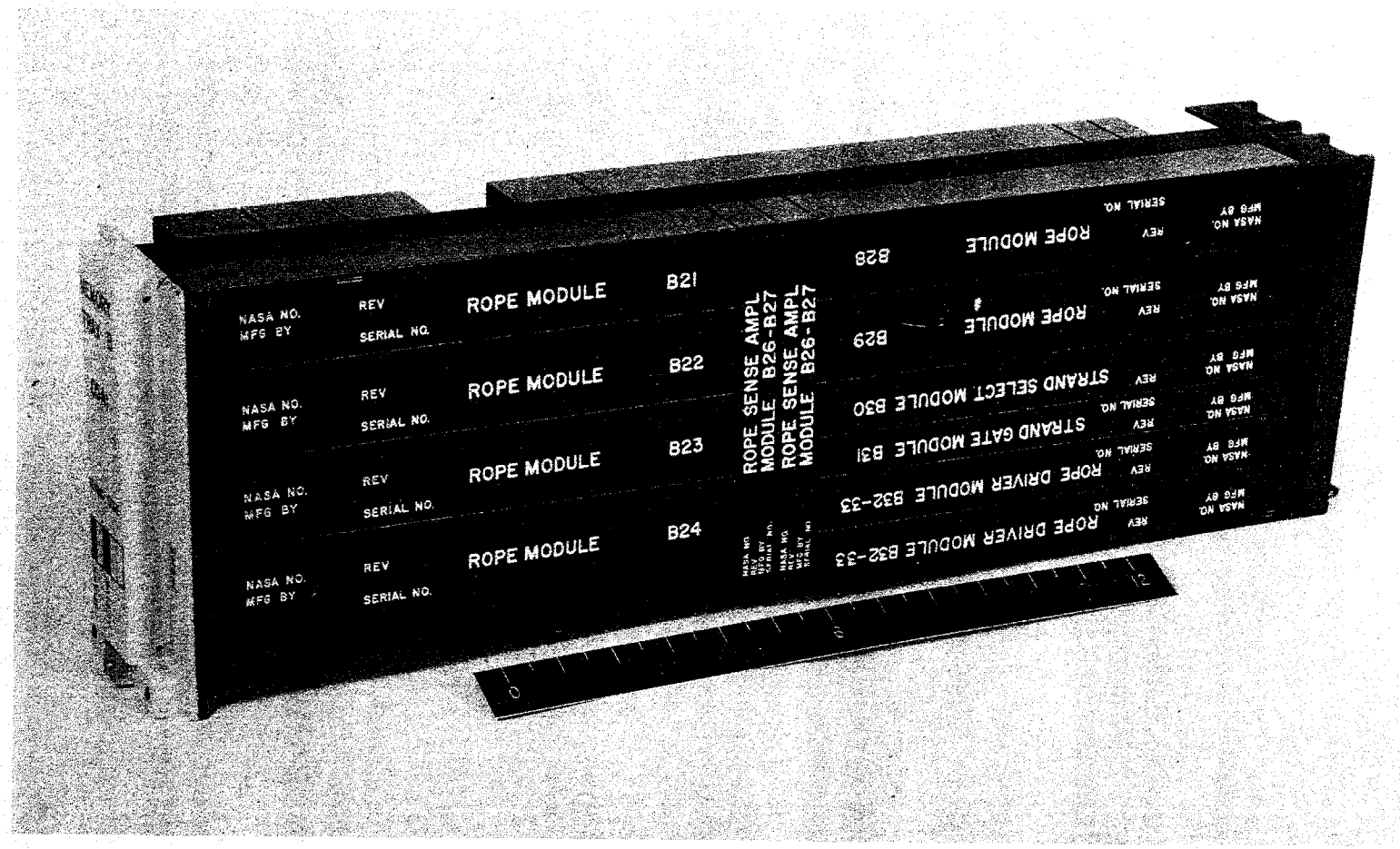

Fig. 5 Guidance Computer Memory Tray.

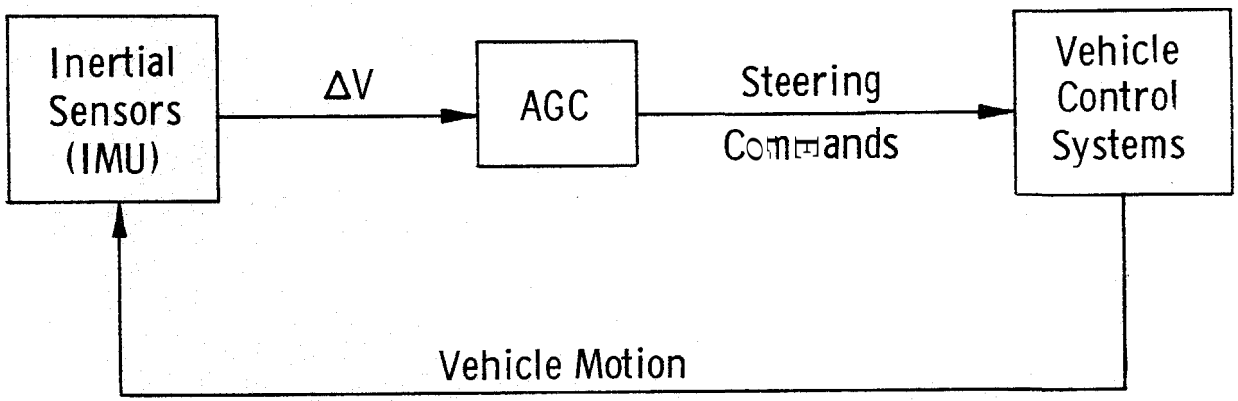

Fig. 6 Vehicle Control During Thrusting

The computation associated with each such iteration begins by updating the vehicle state vector at discrete points in time, using the velocity increments received from the inertial sensors. From the updated state the required impulsive vector velocity change is calculated. A thrust direction is then selected to align this vector and its derivative. The difference between the desired thrust direction and the actual orientation as measured by the inertial system is used to generate steering commands. Each of these computational phases has its own iteration rate: the state vector updating frequency depends on both the integration technique used and the characteristics of the accelerometers; the recalculation of the required velocity is time consuming and can not be performed too often; the rate at which steering commands are updated depends, to a great extent, on the dynamics of the overall guidance loop,

Throughout the powered maneuver, inputs must be received from and outputs delivered to equipment external to the computer. In particular, inputs may consist of:

1. velocity increments
2. gimbal angles
3. guidance and navigation status signals
4. astronaut keyboard commands
5. ground commands

These inputs are basically asynchronous with any internal computer activity and have widely disparate frequencies. Further, it should be noted that commands from ground based control centers are functionally indistinguishable from keyboard inputs initiated by the astronaut.

The outputs from the computer might be:

1. steering commands
2. maintenance of mode and caution lamps
3. digital display updating
4. digital downlink transmission

Performance verification is accomplished by periodically checking the operation of the logic and memory elements of the computer and by incorporating consistency checks of significant computed parameters. The mode and status of the guidance and

spacecraft systems must be continually examined for indication of failure while the overall steering loop must be checked for loss of control by examination of gimbal angle inputs. Any malfunction *so* detected must result in displayed alarms, engine shut-down, or other appropriate action.

ORGANIZATION OF COMPUTER AND PROGRAMS

A. Input/Output Considerations

The input signals described in the previous section may be divided into three classes:

1. The velocity increments and gimbal angles constitute information of wide band-width and are in the form of pulses representing increments of velocity and increments of angle. To accomodate these inputs, with minimum computer time and interface equipment, a digital differential analyzer type of counter has been incorporated[7]. They enter the computer as pulses which "steal" the next available computer memory cycle *so* that they may be summed in these counter registers. The only effect of the counter service cycle is to lengthen the program execution time by one memory cycle. The use of counter registers as input buffers permits economical serial interfaces with the inertial measuring devices. The arithmetic registers of the computer are time-shared for the maintenance of these counters, providing, thereby, a significant saving in hardware. Since computer time thus consumed is proportional to the rate of change of the input quantities, wide band-width signals are readily accomodated. Provision is made for several such counter inputs to be exploited for many other purposes, some of which are described below.

2. The system status signals are discrete inputs which may be serviced at a low frequency and are directly coupled with input registers, each corresponding to a particular bit in the register, No immediate response, in the sense of computer speed, is required to changes in state; these inputs need only be sampled periodically by the performance verification programs.

3. The keyboard inputs present a slightly different problem. As soon as one keyed character enters the associated input register of the computer, it must be stored away to prepare for the next. In this way the need for buffer storage is eliminated. To perform this function without frequent scanning of the relevant input register, the entry of a key character causes an involuntary interrupt of the program sequence in progress. The effect of the interrupt is to cause the next instruction to be selected from a standard location, after sufficient information has been saved to allow resumption of the interrupted program. The ensuing interrupt program may perform short-term processing and then resume the interrupted program via a special instruction. Several interrupt options are provided for similar applications. To minimize interrupt storage requirements and simplify associated programs, only one level of interrupt is allowed. If an interrupt occurs during the execution of another, the second interrupt is inhibited until the completion of the first.

Output commands may be divided into three categories similar to the input signals:

1. Steering signals are the output analog of the input gimbal angles. To utilize the same type of minimum, serial interface, the commands are sent in the form of pulse trains with each pulse representing an angular increment about one of the three spacecraft body axes. The counter feature, previously described for processing inertial sensor inputs, is used to meter out the pulses without requiring program supervision. To initiate such an output signal a program delivers a desired incremental body angle to a counter and then enables an output gate. Every n microseconds a counter service request is generated by the computer clock if such a gate is set. A pulse is sent to the spacecraft control system and the counter contents reduced by one in the following "stolen" memory cycle. This process is repeated until the required number of pulses has been delivered; i.e., during the counter increment cycle which reduces the counter to zero, the output gate is reset, removing future counter service requests. Note that after initiation of the pulse train, no further program supervision is required. In order that three-axis control may be accomplished with one counter, commands are transmitted serially at a high frequency.

2. Maintenance of mode and caution lamps is a low frequency operation. They are controlled by discretes coupled directly to bits in output registers and may be set as required by the appropriate programs. The process is the reverse analog of the method of handling system status inputs.

3. Transmission of information to the digital display, Fig. 7, takes place via a single output register. Essentially, the information in this output register specifies two digits on the display. Since the display lights are not switched by solid state devices, several milliseconds must transpire before a command word may be removed from the output register. To update the entire display, a computer program must place the display information in a buffer table in erasable memory. These characters will be sent out, two at a time, at a constant rate until no further changes in the display are required. This rate is sufficiently low to accomodate the switching time delay without compromising good visual presentation.

Down-telemetry operates on the same basic principle. A single output register is assigned to this task. During transmission, the contents of this register are shifted out serially to the transmitter (the serial to parallel conversion requires no program supervision). An end-of-transmission signal causes an interrupt and the associated program delivers the next word to the output register.

B. Computation Facilities

The wide range of variables, the high accuracy requirements and the general complexity of the Apollo mission dictate the need for a computer with substantial word length, large memory capacity and a flexible and versatile instruction set. Unfortunately, the short word length of the Apollo computer is not compatible with an easy solution to these problems.

The computer instruction word contains 16 bits: a parity bit, three operation code bits, and 12 address bits, Thus, with a direct addressing capability of 4096 words, the computer, nevertheless, has almost an order of magnitude more memory locations. The problem of generating addresses for all storage locations is successfully handled by means of a special bank register. The first 3072 computer memory locations carry unique addresses, while the rest

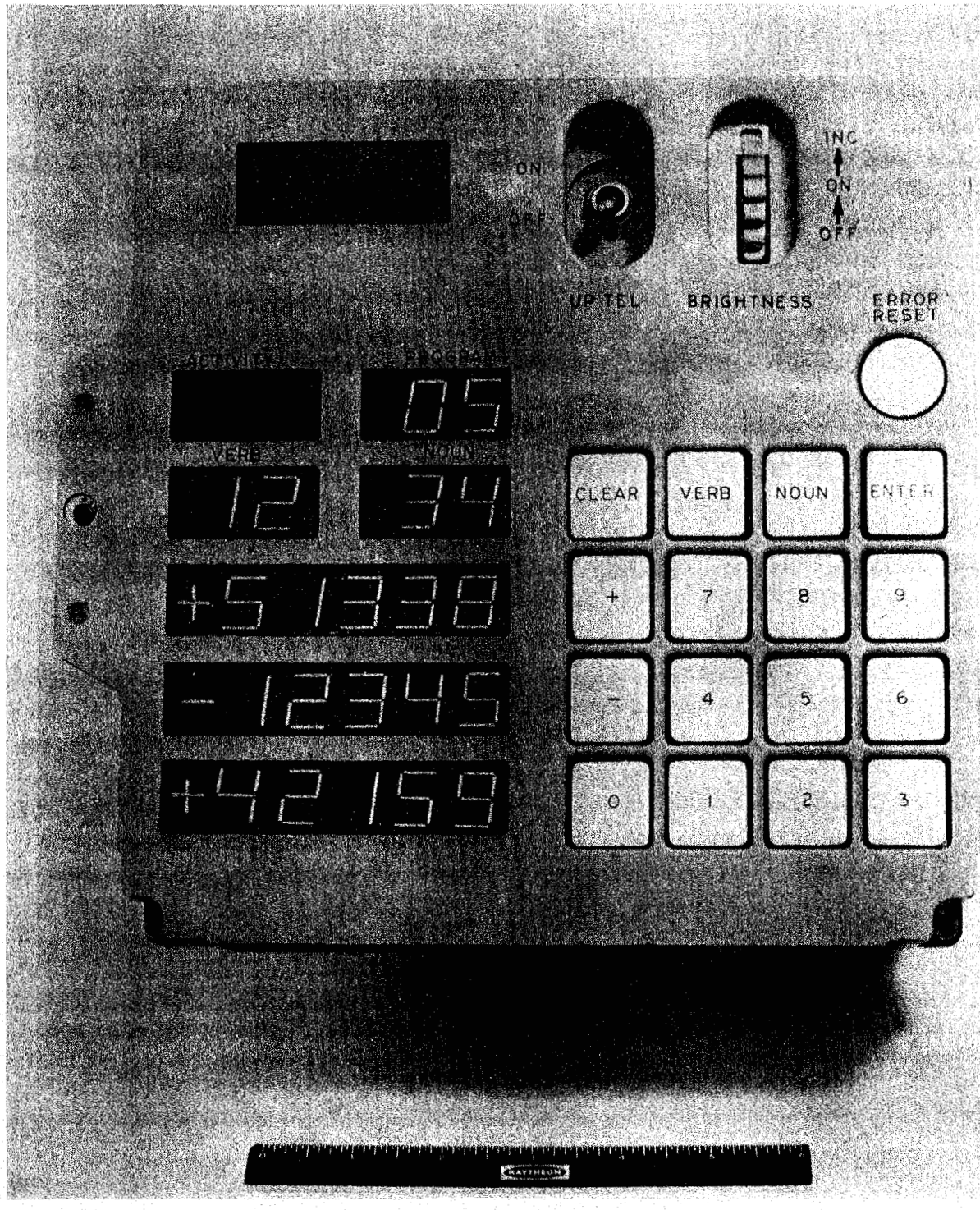

Fig. 7 Display and Keyboard.

of the memory is divided into banks of 1024 words, each with an individual address determined by a combination of the address bits in an instruction word and the bank register setting.

The operation code set cannot be extended in the same manner as the addressing capability. If only basic language coding is used, the programmer must perform the many tasks required using only eleven simple, single precision instructions. The mathematical and logical complexity of the problem dictates a need for a large and powerful operation code set.

A pseudo-language and a set of routines to accomplish these various ends have been developed. The advantages of a sophisticated algebraic compiler, which would translate equations written in a convenient problem-oriented language into basic machine language, are well known. However, the basic program resulting from such a translation is generally wasteful of storage. As an alternative, an interpretive system was chosen as a suitable compromise between the luxury of the compiler and the tedium of basic language programming.

The interpreter resembles a compiler in that programs are written in a problem-oriented language, but an encoded form of this language is stored directly in the computer memory. The final translation of such a program is done by a group of basic language programs at execution time. This solution requires far less storage than the compiler-generated program but at the expense of increased execution time. The time increase is largely brought about by the need for the translation process. Although the translating program must be carried in memory, it requires only a few hundred words and is certainly a small price to pay for the convenience afforded by the interpreter.

Typically, interpretive systems are based on one pseudo-operation code and one or more addresses placed in the same word, Since an interpreter must have many operation codes to be effective, the short word length of the guidance computer precludes this organization. To provide the desired full-word address and still allow a diversity of operation codes, a form of polish (parenthesis free) notation has been adopted[a]. In this language an equation is

written as a number of consecutive operations followed by the required operand addresses. One bit is reserved in each word of the source language to distinquish operation codes from addresses *so* that list-processing techniques may be employed, thereby providing additional savings in storage. Two operation codes are packed in each word, yielding 128 possible pseudo-instructions. A full word is available for operand address storage, *so* that 16,384 registers may be directly addressed.

The bulk of the interpretive instructions is listed in Fig. 8 together with a sample interpretive program in Fig. 9. An examination of these figures clearly shows the many advantages that have been obtained: double and triple precision arithmetic, explicit vector and matrix operations, and index register addressing.

C. Execution Control

Efficient control by the computer of several different functions occuring at approximately the same time is an inherent requirement of the guidance system. To provide this control, the necessity for some effective technique for initiating these different functions is apparent. Furthermore, with a limited amount of erasable storage available, it is desirable that this storage be time-shared for the various functions.

1. Scheduling

There are, in general, two ways to initiate a program - through an externally generated command or signal, or, internally, on the basis of the elapse of a specific interval of time. The mechanism for the first type of program initiation has already beendescribed in the discussion on keyboard characters. The measurement of elapsed time within the computer is handled by the counter structure. One such counter is fed by a stage of the computer clock divider chain. On receipt of a pulse from the selected stage, the counter is incremented by one during the next memory cycle. When the counter overflows, it causes an interrupt. The program activated by this interrupt then takes whatever action is appropriate and resets the timing counter for the next event in a queue, which can accomodate several such events.

Scalar Arithmetic*

Add
Subtract
Subtract From
Multiply
Multiply and Round
Divide By
Divide Into
Shifting
Normalization

Vector Arithmetic*

Add
Subtract
Subtract From
Vector Times Scalar
Dot Product
Cross Product
Matrix Pre-multiply
Matrix Post-multiply
Unit Operation
Vector Magnitude
Vector Complement
Shifting

Scalar Functions*

Square Root
Sine
Cosine
$Sine^{-1}$
$Cosine^{-1}$
Absolute Value
Complement

Decision Operations*

Branch Positive
Branch Negative
Branch Zero
Branch if Overflow

Indexing Operations

Load index directly
Load index indirectly
Store index
Exchange index
Add to index
Subtract from index
Count on index

*Double Precision Throughout

Fig. 8 Typical Interpretive Pseudo-Operations

Problem: Compute $\underline{z} = aM(\underline{x} + \underline{y})$
where a is a scalar and M a 3 x 3 matrix

Program (requires 7 words of storage)

```
        MXV   }
VXSC          }   Operation Codes

        X     }
        Y     }   Operand
        M     }   Adresses
        A     }

STORE   Z     }   Left-over address used to store result
```

Explanation

1) The first address of an equation is used to load an accumulator; VAD requests a vector load.

2) Each op code results in a subroutine call with the corresponding address left in a standard location.

3) After all op codes have been "executed," the remaining address is used to store the result. Since the result of the last operation is a vector, a vector will be stored in Z.

Fig. 9 Sample Interpretive Program

Programs are always initiated when the computer is operating in the interrupted mode. If the program to be initiated is of very short duration (a few milliseconds), its function will be completed in this mode. If, as is normally the case, a longer period of time is required, a request is made of the priority control program to resolve any scheduling conflict and, eventually, allow the program to operate. Clearly, the computer cannot be allowed to be in the interrupt mode for longer than a few milliseconds since, during this time, all other interrupts are inhibited.

2. Dynamic Storage Allocation

When a request is made to the priority control program, one of the first tasks is to perform the erasable memory time-sharing functions. A moderate number of registers form a pool which is time-shared by programs requesting execution via priority control. Requesting programs are assigned a number of registers which are available to the job throughout its execution period but are relinquished when the program terminates., Almost all guidance computer programs operate in this fashion.

3. Priority Control

A program scheduled for execution through priority control may not commence until the program currently in progress terminates or allows itself to be suspended. To resolve scheduling conflicts, each program is assigned a priority number which accompanies the scheduling request. If the newly scheduled program is of higher priority, a flag is set to indicate to the current program that it must temporarily suspend its activity, To assure speedy access to the computer, each program running in this fashion must test the indicator every few tens of milliseconds. Program suspension is, functionally, a programmed interrupt with the required information retained for resumption of the program when it is again of highest priority. If, on the other hand, the new program is of lower priority, it will not gain access to the computer until at least the current program is completed. Approximately ten such programs may be scheduled for execution or in partial stages of completion.

A program may optionally suspend itself with its dynamically allocated storage protected, even if it has highest priority. This may be desirable while awaiting the completion of an input/output event, for example. At the occurrence of the desired event, the program is rescheduled to resume at its original priority.

At the completion of the program the time-shared storage is made available to other programs and the completed program removed from the scheduling list.

APPLICATION

The programs required to implement the guidance steering mission phase discussed previously can now be readily organized within the framework that has beenconstructed, Computations may be coded in a convenient and readable language with adequate precision. The computational flow is accomplished using the scheduling and prioritycontrol programs. In the remainder of this section, the example problem will be discussed in some detail.

A. Computational Flow

During each navigation cycle a timed interrupt occurs resulting in a request to priority control forintegration of the state vector. In addition, steps are taken to ensure that a similar operation will take place at the next cycle time. The calculation then proceeds through to completion, unaffected by higher priority tasks such as a special display request, The guidance law calculation is treated similarly but its priorityis higher to prevent changesin the state vector from occurring during the computation cycle. The steering commands are left in buffer registers for servicing by the output control program,

B. Input/Output Control

The output control program has a fast repetition rate but each cycle is completed in the interrupted mode and is, therefore, independent of the priority control scheme. Among the several tasks, performed at integral multiples of its basic cycle time, are:

1. Status monitoring of all subsystems, comparing the desired and actual states. Any disagreement results in a suitable alarm and other appropriate actions as required.

2. Transmission of the steering commands to the spacecraft control system via the single output channel available. This program acts as a traffic controller determining which spacecraft axis is to be serviced each time.

3. Updatingof the digital displays, transmitting two characters at a time from the erasable memory buffer.

As described before, the down telemetry functions independently of these other tasks, being a self contained program.

C. Performance Verification

The requirement that some program must always be in an operational status is a special feature of priority control and is used to'advantage. An extensive self test routine is entered whenever no other non-interrupt program is active. Thus, a significant share of the monitoring job is performed during time in which the computer would otherwise be idle. Hence, all machine time is used effectively in one fashion or another to accomplish the overall goal.

CONCLUSION

The keystone of a space guidance computer is a large memory store for programs. Given this single advantage and the involuntary interrupt, the restrictions of a small operation code, limited input/output facilities, short word length, and addressing difficulties can be cast aside, Of course, a price is paid in operation time and the basic computer design must provide enough speed to counter this problem. Nevertheless, adequate memory and sufficient speed allows incorporation by programming of all those features of a digital computer which are typically accomplished within the circuitry at a substantial price in weight, volume and reliability.

REFERENCES

1. Sullivan, W., America's Race for the Moon, Random House, New York, 1962.
2. Sears, N. E., "Technical Development Status of Apollo Guidance and Navigation", MIT Instrumentation Laboratory Report R-445, April 1964.
3. Alonso, R. L., and Hopkins, A.L., "The Apollo Guidance Computer", MIT Instrumentation Laboratory Report R-416, August 1963.
4. Alonso, R.L., Blair-Smith, H., and Hopkins, A. L., "Some Aspectsof the Logical Design of a Control Computer: A Case Study," IEEE Transactions on Electronic Computers, Vol. EC-12, December 1963.
5. "Core Rope Memory", Computer Design, June 1963.
6. Battin, R.H., "Explicit and Unified Methods of Spacecraft Guidance Applied to a Lunar Mission", Proc. 15th International Astronautical Congress, Warsaw, 1964.
7. Alonso, R.L. and Laning, J.H., Jr., "Design Principles for a General Control Computer", Institute of Aeronautical Sciences, New York, New York.
8. Muntz, C.A., "A List Processing Interpreter for AGC4", MIT Instrumentation Laboratory Report AGC Memo 2, January 1963.

E-1758

DISTRIBUTION LIST

Internal

M. Adams
R. Alonso
J. Arnow (Lincoln)
R. Battin
P. Bowditch
R. Boyd
G. Cherry
E. Copps
R. Crisp
W. Crocker
G. Cushman
J. Dahlen
E. Duggan
J. Dunbar
K. Dunipace (MIT/AMR)
R. Euvrard
J. B. Feldman
P. Felleman
S. Felix (MIT/S&ID)
J. Flanders
J. Fleming
Eldon Hall
W. Heintz
D. Hoag
A. Hopkins
F. Houston
L. B. Johnson
M. Johnston
A. Koso
M. Kramer
A. Laats
A. La Pointe
J. Larsen
L. Larson
J. Lawrence (MIT/GAEC)
T. J. Lawton (5)
T. M. Lawton (MIT/MSC)
D. Lickly
R. Magee
G. Mayo
James Miller
John Miller
J. Nevins
J. Nugent
E. Olsson
C. Parker
W. Patterson
J. Rhode
R. Scholten
E. Schwarm
N. Sears
J. Shillingford
W. Shotwell (MIT/ACSP)
W. Tanner
N. Thacher
R. Therrien
W. Toth
M. Trageser
R. Weatherbee
R. White
L. Wilk
R. Woodbury
W. Wrigley
Apollo Library (2)
MIT/IL Library (6)

External

(ref. PP1-64; April 8, 1964)

P. Ebersole (NASA/MSC) (2)

W. Rhine (NASA/RASPO) (1)

L. Holdridge (NAA S&ID/MIT) (1)

T. Heuermann (GAEC/MIT) (1)

AC Spark Plug (10)

Kollsman (10)

Raytheon (10)

Major W. Delaney (AFSC/MIT) (1)

NAA RASPO: (1)

National Aeronautics and Space Administration
Resident Apollo Spacecraft Program Office
North American Aviation, Inc.
Space and Information Systems Division
12214 Lakewood Boulevard
Downey, California

FO: (3)

National Aeronautics and Saace Administration. MSC
Florida Operations, Box MS
Cocoa Beach, Florida 32931
Attn: HB 23/ Technical Document Control Office

HDQ: (6)

NASA Headquarters
600 Independence Ave., SW
Washington 25, D.C. 20546
Attn: MAP-2

AMES: (2)

National Aeronautics and Space Administration
Ames Research Center
Moffet Field, California
Attn: Library

LEWIS: (2)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio
Attn: Library

FRC: (1)

National Aeronautics and Space Administration
Flight Research Center
Edwards AFB, California
Attn: Research Library

LRC: (2)

National Aeronautics and Space Administration
Langley Research Center
Langley AFB, Virginia
Attn: Mr. A. T. Mattson

GSFC: (2)

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland
Attn: Manned Flight Support Office Code 512

MSFC: (2)

National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Huntsville, Alabama
Attn: R-SA

ERC: (1)

National Aeronautics and Space Administration
Electronics Research Center
575 Technology Square
Cambridge, Massachusetts
Attn: R. Hayes/A. Colella

GAEC: (1)

Grumman Aircraft Engineering Corporation
Bethpage, Long Island, New York
Attn: Mr. A. Whitaker

NAA: (15)

North American Aviation, Inc.
Space and Information Systems Division
12214 Lakewood Boulevard
Downey, California
Attn: Mr. R. Berry

GAEC RASPO (1)

National Aeronautics and Space Administration
Resident Apollo Spacecraft Program Officer
Grumman Aircraft Engineering Corporation
Bethpage, Long Island, New York

ACSP RASPO: (1)

National Aeronautics and Space Administration
Resident Apollo Spacecraft Program Officer
Dept. 32-31
AC Spark Plug Division of General Motors
Milwaukee 1, .Wisconsin
Attn: Mr. W. Swingle

WSMR: (2)

National Aeronautics and Space Administration
Post Office Drawer MM
Las Cruces, New Mexico
Attn: BW 44

MSC: (45)

National Aeronautics and Space Administration
Manned Spacecraft Center
Apollo Document Control Group
Houston 1, Texas 77058

Mr. H. Peterson (1)
Bureau of Naval Weapons
c/o Raytheon Company
Foundry Avenue
Waltham, Massachusetts

Queens Material Quality Section (1)
c/o Kollsman Instrument Corporation
Building A 80-08 45th Avenue
Elmhurst, New York 11373
Attn: Mr. S. Schwartz

Mr. H. Anschuetz (1)
USAF Contract Management District
AC Spark Plug Division of General Motors
Milwaukee 1, Wisconsin 53201