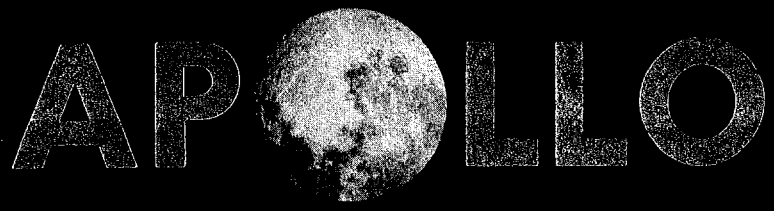


R-416

THE APOLLO GUIDANCE COMPUTER

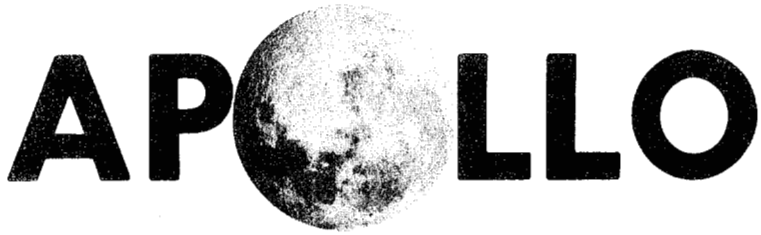
by

Ramon L. Alonso and Albert L. Hopkins

August 1963

MIT

CAMBRIDGE 39, MASSACHUSETTS



GUIDANCE AND NAVIGATION

Approved: Milton B. Trageser Date: 8/12/63

Milton B. Trageser, Director
Apollo Guidance and Navigation Program

Approved: Roger B. Woodbury Date: 8/15/63

Roger B. Woodbury, Associate Director
Instrumentation Laboratory

To be presented at the 1963 National Space Electronics Symposium, Miami Beach, Florida, October 1, 2, and 3, 1963.

R-416

THE APOLLO GUIDANCE COMPUTER

by

Ramon L. Alonso and Albert L. Hopkins

August 1963

MIT INSTRUMENTATION LABORATORY

CAMBRIDGE 39, MASSACHUSETTS

COPY # 213

ACKNOWLEDGMENT

This report was prepared under DSR Project 55-191, sponsored by the Manned Spacecraft Center of the National Aeronautics and Space Administration through contract NAS9-153.

The structure of the Apollo Guidance Computer and its relationship to the rest of the guidance system were strongly influenced by J,H. Laning, Jr. in the area of logical. organization, and by E. C, Hall in connection with circuitry and input and output principles.

R-416

THE APOLLO GUIDANCE COMPUTER

ABSTRACT

The general logical structure of the on-board Apollo Guidance Computer is presented, and the developments of fixed and erasable memory are described. Particular attention is given to the methods of input and output.

by Ramon L. Alonso
and
Albert L. Hopkins
August 1963

TABLE OF CONTENTS

R-416

THE APOLLO GUIDANCE COMPUTER

By Ramon L. Alonso and Albert L. Hopkins

INTRODUCTION

The Apollo Guidance Computer [1]*, [2], (AGC), is an airborne digital computer designed to control, test, and operate a guidance system and to determine and execute changes in spacecraft velocity for optimum performance in a lunar mission.

The computer is organized as a binary, fifteen digit (bit) "general purpose" computer employing parallel word transfer and single address instructions, Instructions and data are stored in a common body of memory composed of several thousand words of fixed, wired-in memory, and about one thousand words of erasable memory. The latter includes a small number of addressable central registers.

Two types of interrupt functions are used in the computer to resolve the fundamental conflict between efficient programming and real-time system requirements [3]. One type is a full program interrupt which stores certain central register contents, executes a short independent program, and then resumes the main program operation., The second type permits certain erasable memory registers to be altered upon external command. This is used principally to increment (by plus or minus one) memory registers used to accumulate positional

* Numbers in brackets refer to similarly numbered references in the Bibliography.

information from incremental shaft encoders, The increments are made in conjunction with extra memory cycles which are commanded by the interrupt circuits to occur between normal memory cycles.

The logic. of the computer is composed of three-input nor gates in microcircuit form, Erasable memory selection is done with core-transistor circuits, and current drivers for fixed and erasable memory are diode-transistor circuits..

ARITHMETIC

The AGC has an arithmetic unit composed of a parallel adder and a few auxiliary flip-flop registers. The four basic arithmetic operations (+, -, ×, ÷), plus complementation and logical disjunction are executed under control of the sequence generator, which furnishes gating pulses according to a prescribed micro-program for each instruction.

The simplified block diagram in Figure 1 shows the principal information and control paths used for computation, Instruction and data words each have fifteen bits, plus a parity bit. Certain central registers, including the accumulator, employ an extra bit for storage of overflows. Instruction words contain a three bit operation code and a twelve bit address, to cover a basic field of 4096 words. The basic field is expandable to several times its size by means of a Bank register.

The three bit operation code specifies one of eight instructions. Table 1 lists these instructions along with their major properties. Three so-called extracode instructions, described in Table 1b, are also available, but can be selected only by an index instruction, which adds a specified data word to the instruction word next in line to be executed. If this addition results in overflow, (exceeds the fifteen bit capacity), the interpretation of the operation code is not the same as if there

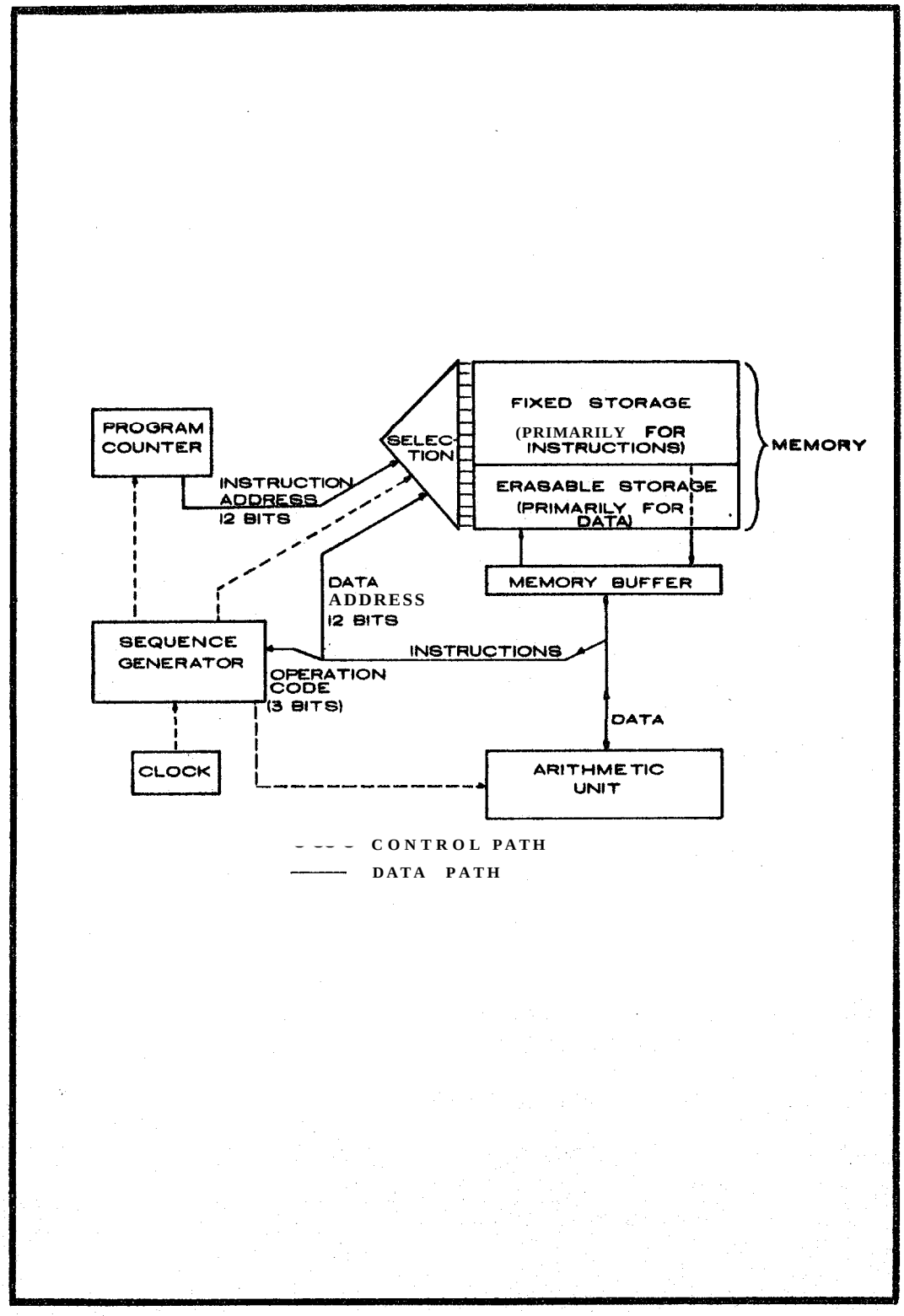


Fig. 1 Simplified block diagram of the AGC,

were no overflow. The extracode instructions require negative overflow for their selection.

Along with the extracode instructions there are several kinds of "involuntary" instructions, i.e., microprogram sequences not under program control., These sequences are triggered by asynchronous external inputs or internal overflows, The specificat-ion of the operand address is part of the triggering logic,

TABLE I - AGC INSTRUCTIONS

a, Normal Instructions

Transfer Control	Break instruction sequence; replace number in Program Counter. Place previous program counter contents in Return Address Register,
Branch Skip and Count	If operand is > 0 no skip = +0 skip 1 instruction < 0 skip 2 instructions = -0 skip 3 instructions Replace accumulator contents by one less than its previous absolute value.
Index	Add operand to next instruction.
Exchange	Exchange operand with accumulator contents.
Clear and Subtract	Replace accumulator contents with negative of operand,
Transfer.to Storage	Replace operand with contents of accumulator, If accumulator stores overflow, skip one instruction and replace contents by a low order one with same sign as overflow.
Add	Add operand to accumulator contents and leave'sum in accumulator, If sum overflows, add a low order plus or minus one to the overflow counter register according to the sign of the overflow,

Mask		Form the bit by bit logical product of the accumulator contents and the operand, and leave the result in the accumulator.

b. Extracode Instructions

Multiply		Multiply operand by accumulator contents and place 29 bit product in the accumulator and the Low-Product register.
Divide		Divide contents of accumulator by operand, Place quotient in accumulator, and the negative absolute value of the remainder in the Return Address Register.
Subtract		Same as Add, but with negative of operand.

c. "Involuntary",Instructions

These instructions are executed not under program control, but as a program interrupting sequence, Addresses of operands are a wired-in part of the instruction.

Positive Increment	x_1	Increment the contents of register x_i by 1.
	x_n	
Negative Incrernent	x_1	Increment the contents of x_i by -1,
	. . x_n	
Shift and Increment	y	Shift the contents of y left one place, and insert a one in the vacant position.
Shift y		Shift c(y) left by one place, These two instructions are used for asynchronous serial inputs.

Interrupt z_1	Stop the present (noninterrupting) program. Save central register contents. Transfer control to Program starting at z_a.
z_k	
Resume	Resume interrupted program after restoring central registers.

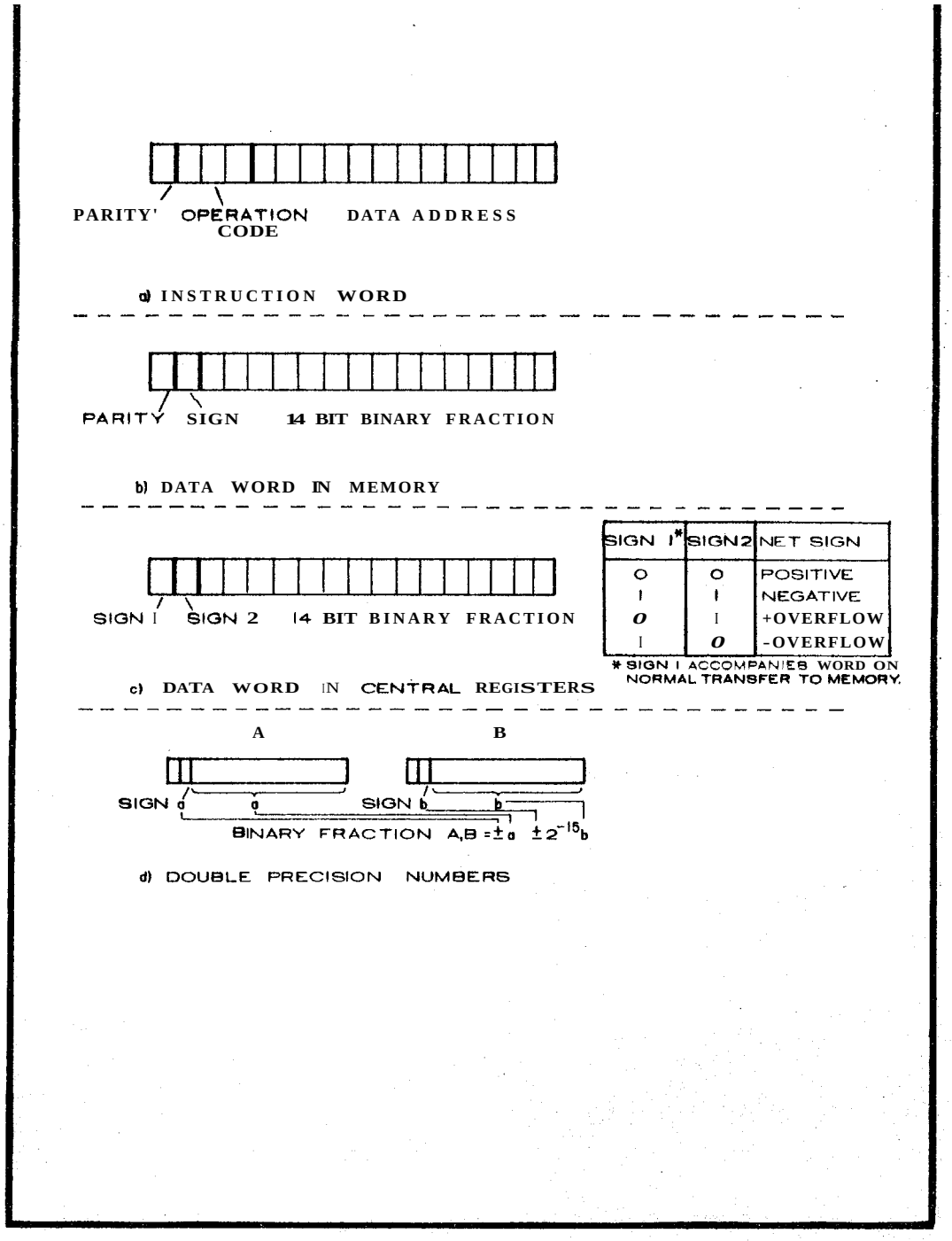
Data words employ a one's complement convention for representing negative numbers; that is, to change the sign of a number, every bit in the word is complemented. This requires that the adder bave "end around carry" from the high order digit position to the low order, a property easy to incorporate in the AGC and in most other machines with parallel word transfer,

The comparatively short word length in the AGC is advantageous from the point sf view of equipment minimization. Fifteen bit words are sufficient to deal with most of the control, telemetry, and display activities of the Apollo mission, Navigational calculations, however, require approximately twice the precision afforded by fifteen bits; and multiple precision numbers, composed of two or more computer words, are used to represent the variables used in such calculations, Arithmetic operations on.multiple precision numbers are carried out by subroutines which operate on. one component word at a time [4]. Figure 2 illustrates the various word formats'used in the AGC.

MEMORY

Three classes of memory are used in tine AGC: €lip-flops, a conventional coincident Current core memory, and a fixed core rope memory. They differ in cost and utility.

The use of flip-flops is limited to as few registers as possibie, owing to their high cost in terms of size, weight, and power, Their two advantages are their short cycle time and



Fig, 2 Word formats.

their steady state representation of variables, The short. cycle time is required in the arithmetic unit and other central registers which are directly gated by instruction and microprograms, The property of steady state representation is necessary in the output and memory bank registers; and is desirable in input registers, Only about a dozen registers of this sort are used,

The erasable memory is of the coincident current, ferrite core type,, operated with a very conservative cycle time [4]. Figure 3 shows the size of an early prototype ferrite stack,

Selection is accomplished by two banks of steering circuits. Twelve steering circuits operate on an eight by four coincidence principle to select one of 32 drive lines, as shown in Figure 4, The steering circuits are of some interest because the cores in them provide a memory for the address of the register after it is read out, The transistor driven by the switch core acts as a directional closure, and avoids problems of voltage references,

The "rope" memory is a form of wired-in storage that has proven to be compact and reliable, The principles of its operation are covered in References 3, 6, and 7, Briefly, a set of inhibit wires threads the cores in such a way as to saturate all but a selected one of the cores (Figure 5). A set line attempts to switch all cores, but only the selected core is switched, As it switches, the selected core transfers energy to whatever sense lines thread it, Such sense lines show a one (an output pulse), while those sense lines which do not thread the selected core show a zero (no output). Unlike the coincident current memory, in which each core corresponds to one bit, the rope cores each correspond to a multibit word, The size of the word in this context is the number of sense lines in the rope.

The advantages of the ropes are numerous. The program, once wired in, cannot be electrically altered, a substantial asset

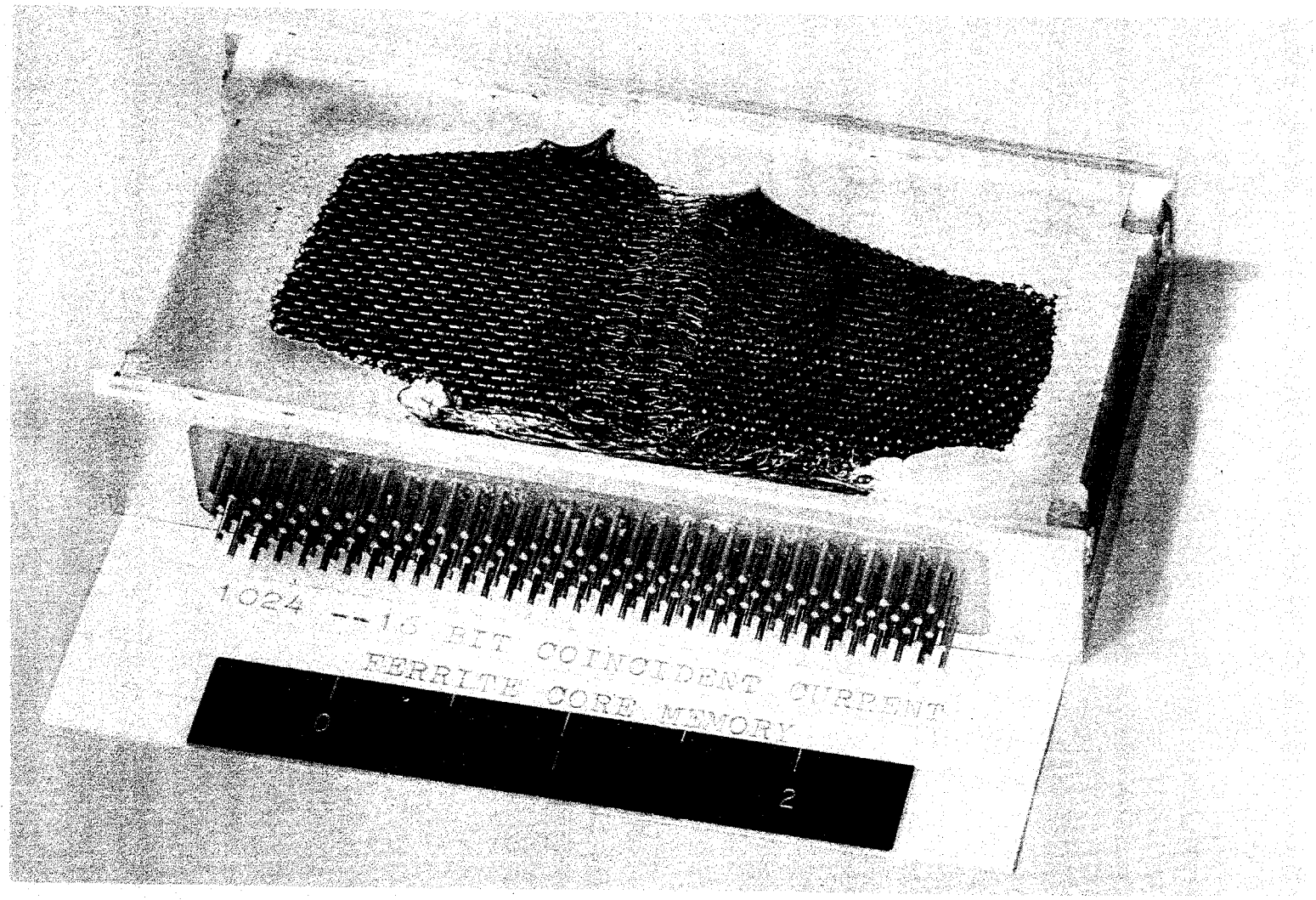


Fig. 3 Coincident ɔrrent memory.

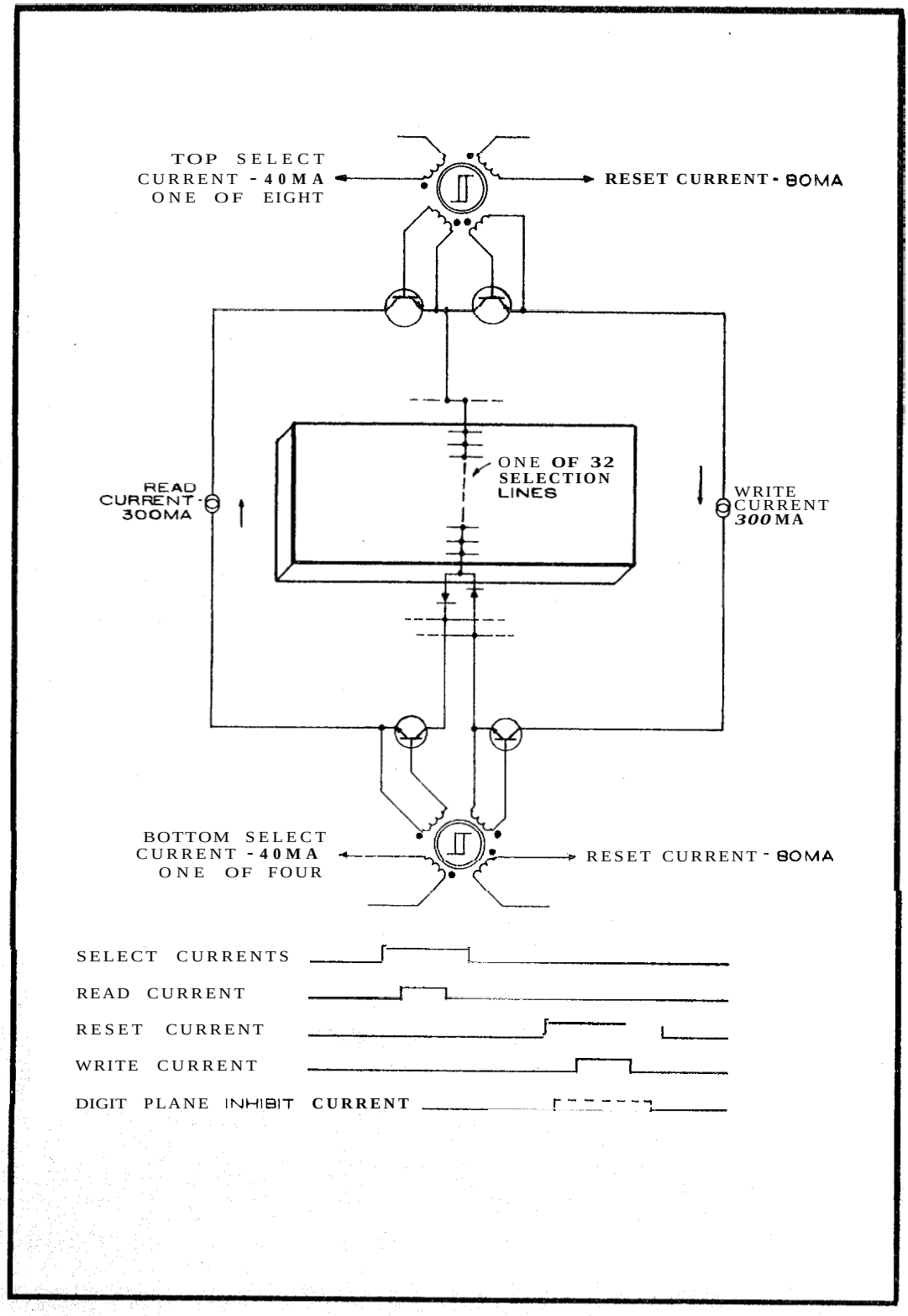


Fig. 4 Erasable memory current switching.

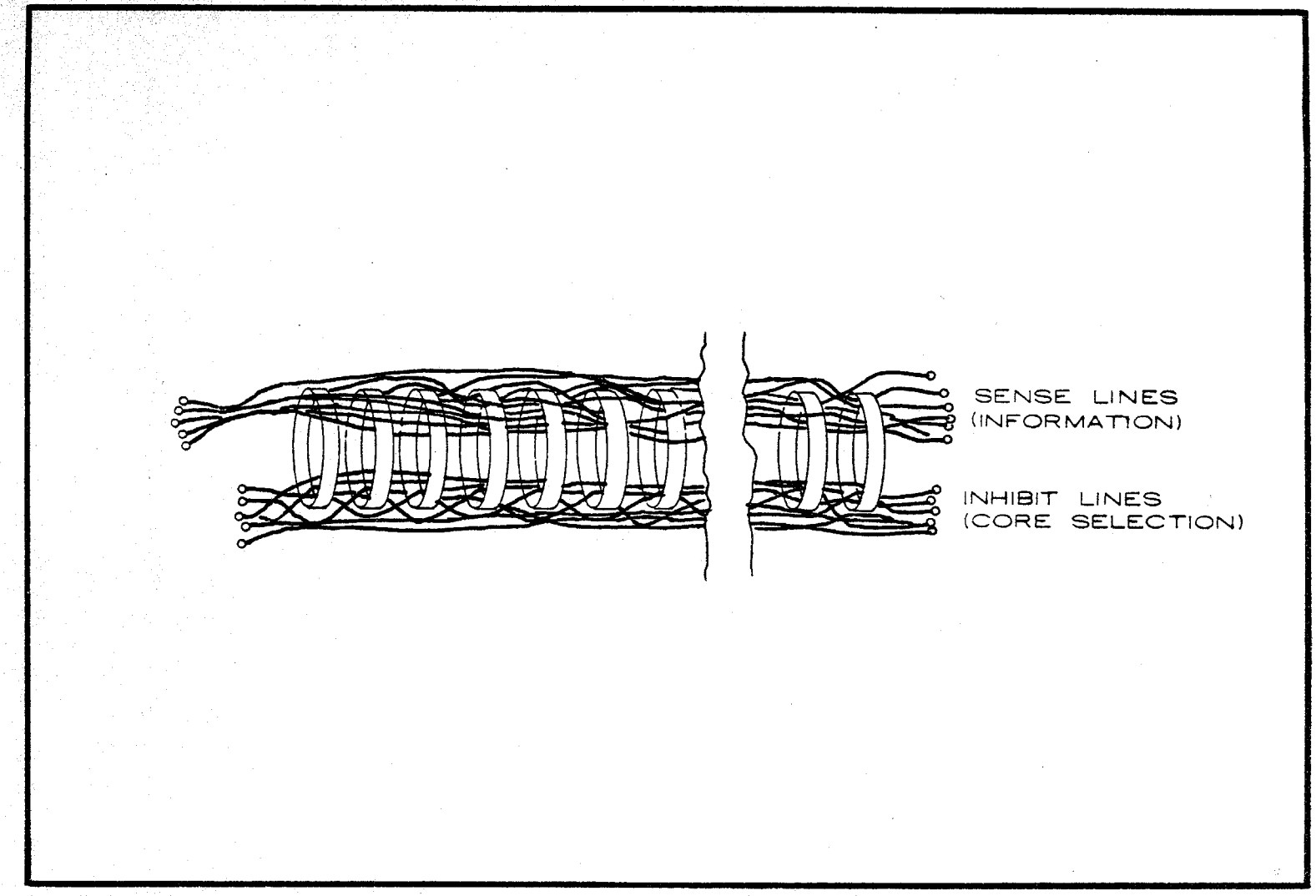


Fig. 5 Inhibit and sense lines through a rope.

for mission reliability. Since the programs cannot be affected as to content, prompt restarts are easily made,

The rope memory is of the random access type, of moderate speed, The rope cycle time was made identical to that of the erasable memory in order to simplify the system, The power consumption is proportioned to activity, independent of the number of ropes in the memory system, and about the same as that required for the erasable memory. The electrical circuitry €or operating ropes is simple and substantially smaller than that required to operate an equivalent coincident current memory.

Another advantage of the rope is its cost, A number of companies have made ropes for MIT (Figure 7), and many of these were made as packaging studies, Those made by Burroughs are electrically functional only, and cost between five and seven cents per bit, To our knowledge, other nondestructive random access memories cost about ten times as much, The above cost is mentioned because it does not represent an estimate, but an actual price paid,

The main disadvantage of the rope is that its contents may not be changed except by rewiring, Such change in general requires replacement o€ a module, which puts a greater than usual premium on correct programming, Modular construction and high speed production techniques are expected to render this characteristic inconsequential, especially in view of the time required for proper documentation, analysis, and simulation of changes in programs.

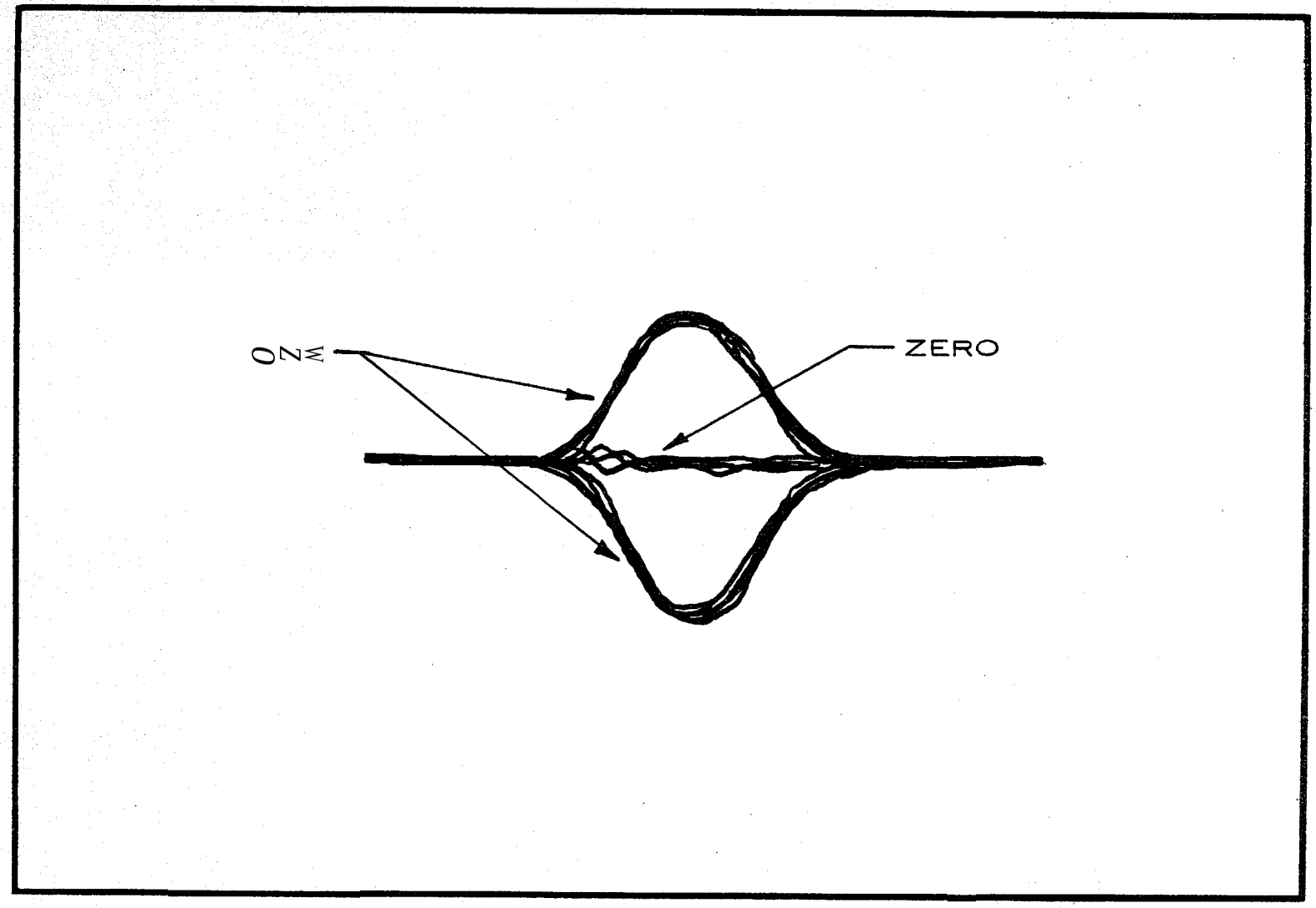


Fig. 6 Sequential output envelope of a 256 core rope memory.

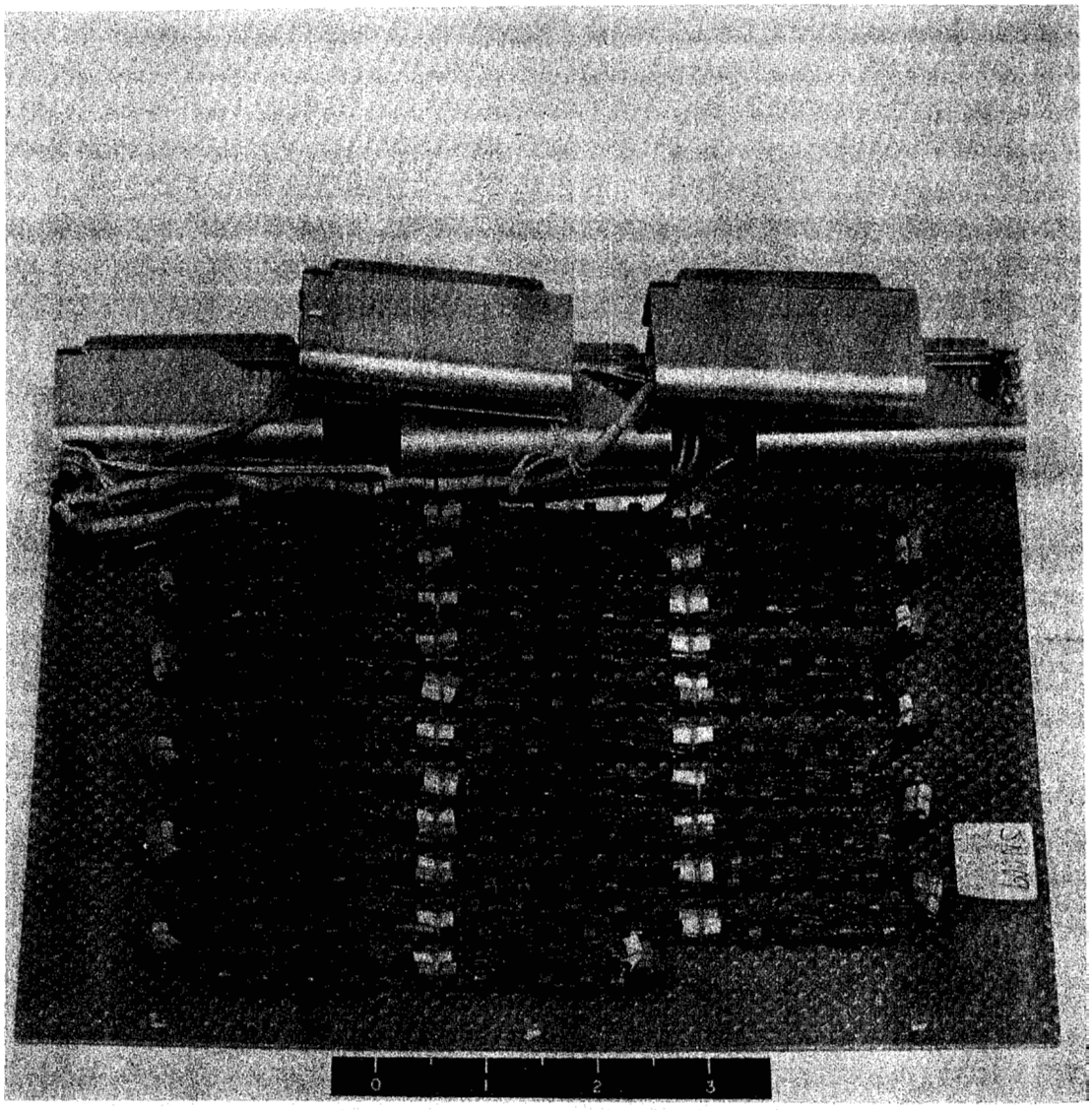


Fig. 7a Core rope made by Wheeler Corporation.

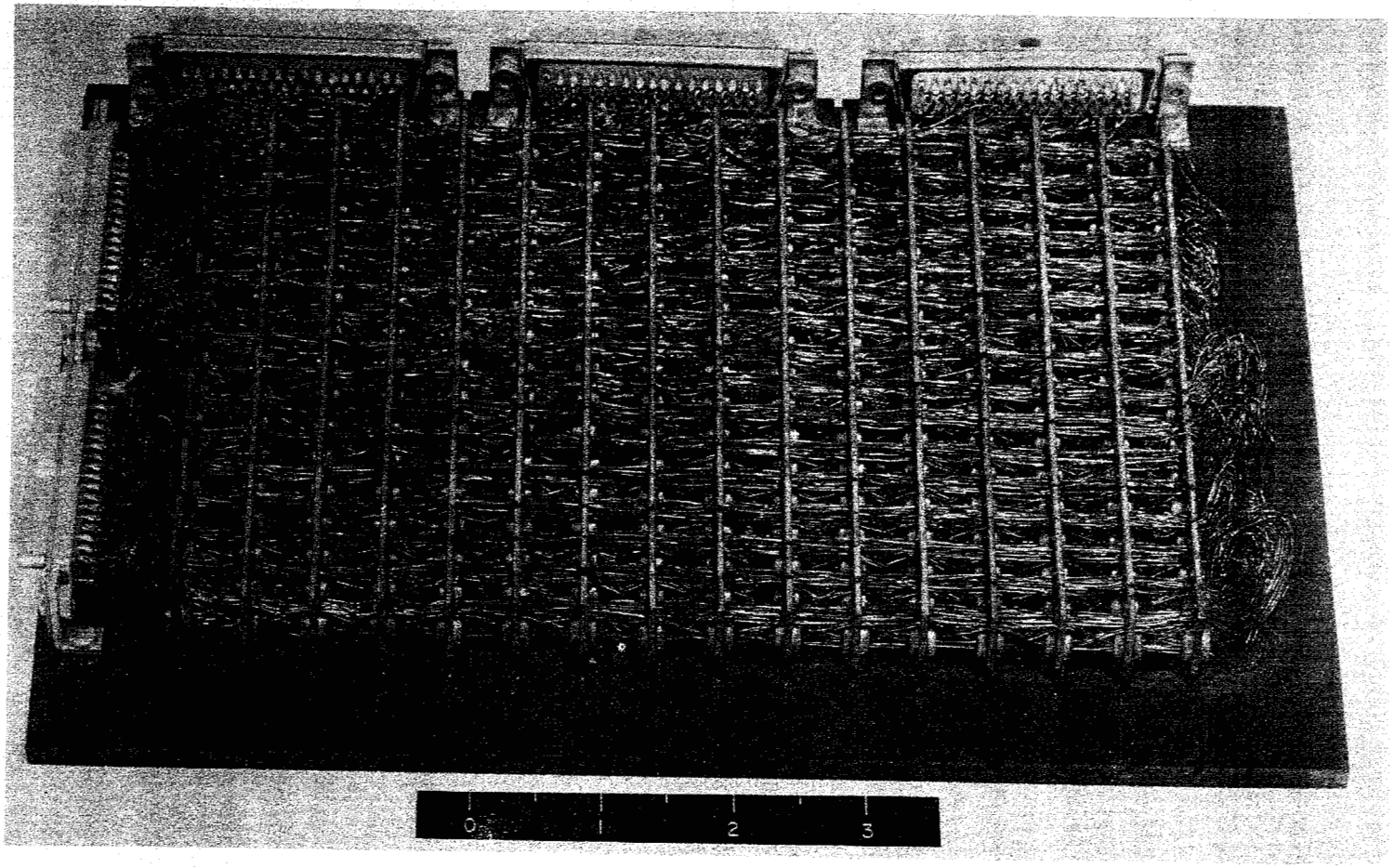


Fig. 7b Core Plane Head by Burroughs Corporation.

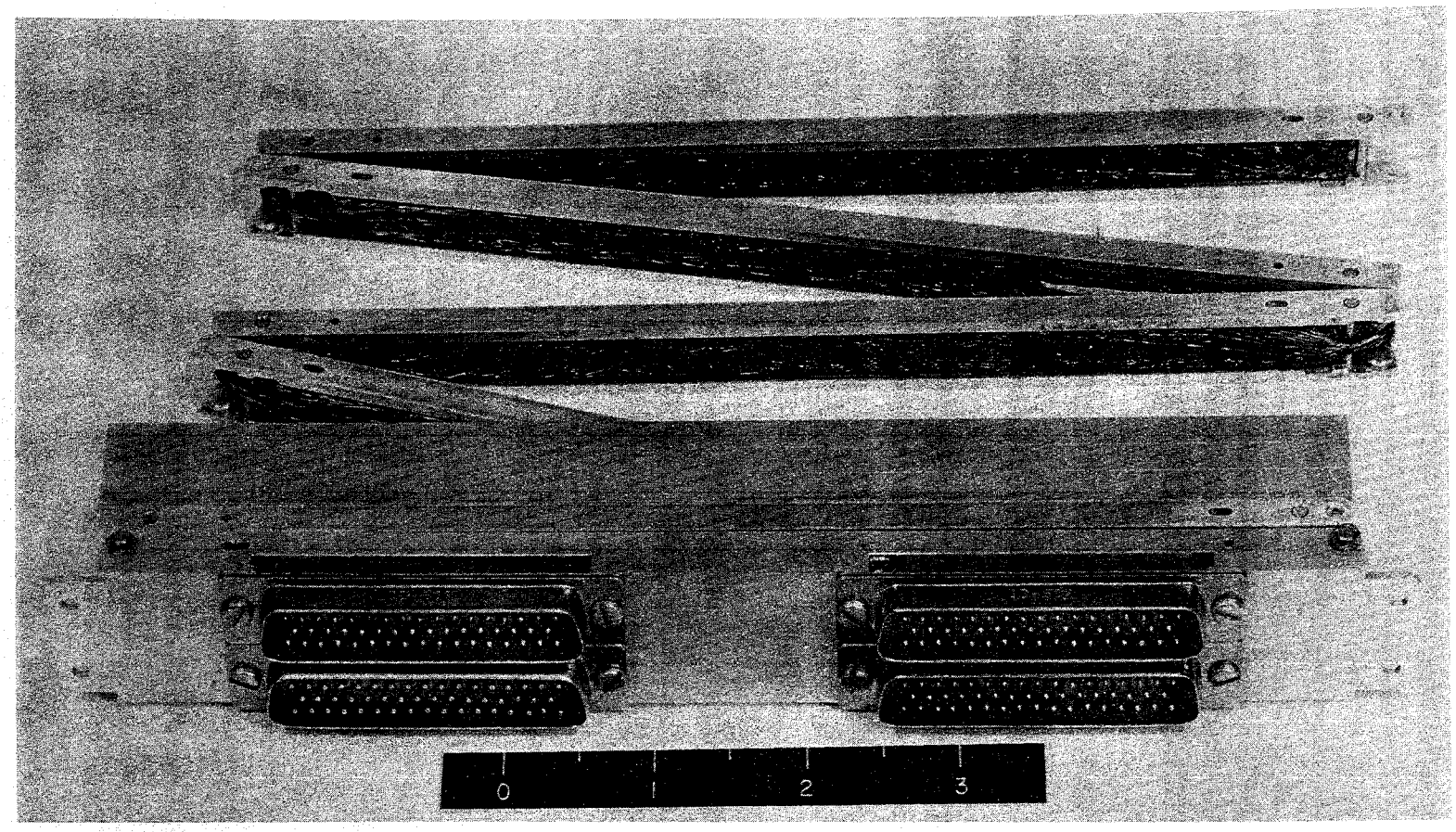


Fig. 7c Core rope made by Sippican Corporation.

Fig. 7c Core rope made by Sippican Corporation

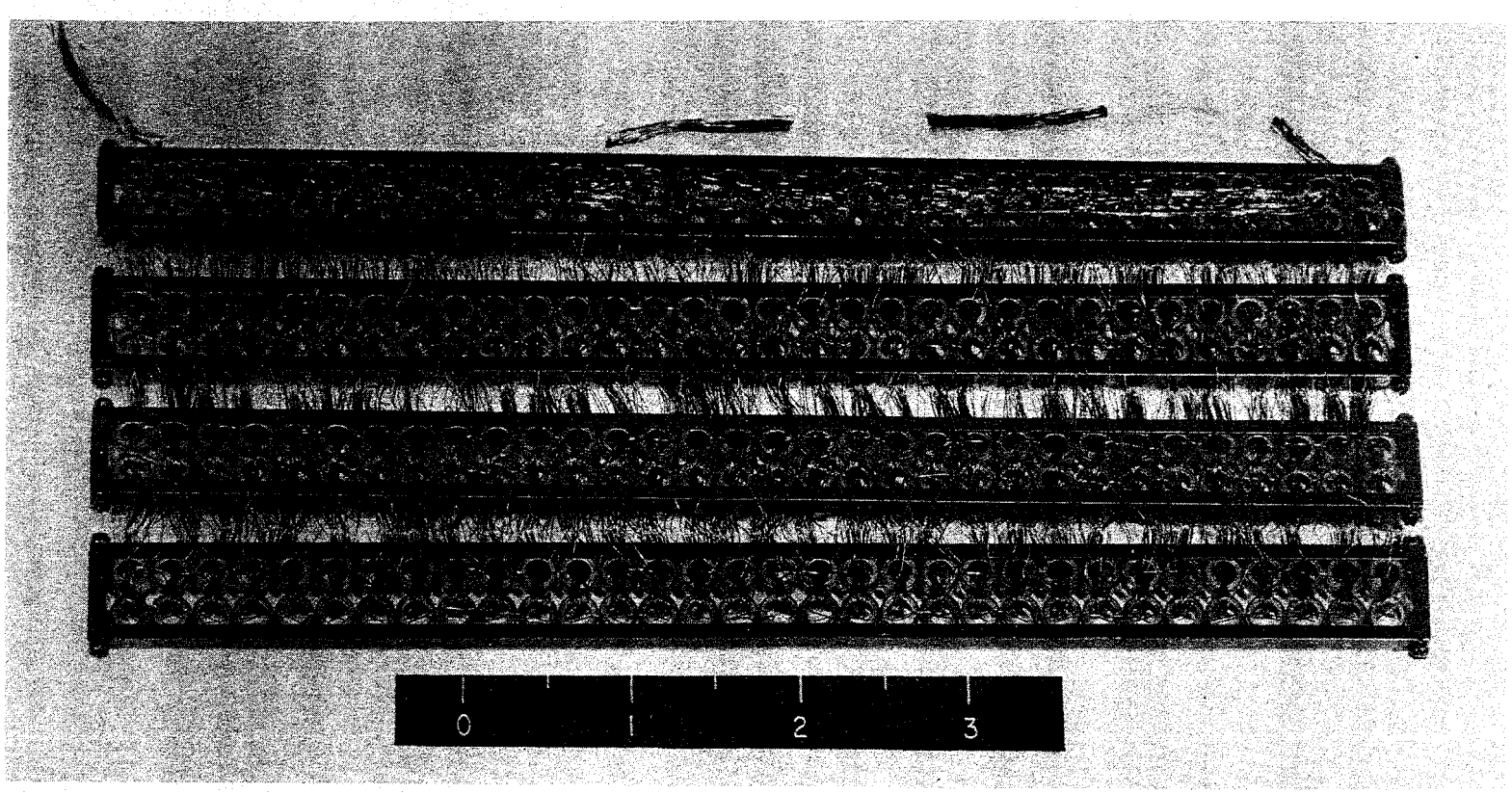


Fig. 7d Core rope made by Raytheon.

INPUT - OUTPUT

A. General Interface Problem

The practical matter of integrating a guidance computer into a system requires the definition and specification of as many as several hundred signals. The computer maker is usually faced with the task of coordinating circuits with several other subsystems, and it is of great importance that all of the input and output circuits and signals be as alike as possible. The development of the Apollo Guidance System has followed a number of principles which reflect expe'rience gained in previous missile-borne control systems.

1) Electrical isolation from other subsystems.

This is a very important point which has both electrical and logical implications. The computer is electrically tied to the power supply ground at a single place. and thus avoids ground loops. Isolation of interface signals is accomplished either by transformer coupling or by relays and switches.. When an input consists of a relay or switch closure, the contacts connected to the computer are connected nowhere.else.

The input and output circuits are designed *so* that no damage may be caused by improper connections at the interface; transformer coupling offers some'protection against wrong connections, and the circuits are further designed to withstand accidentally shorted secondaries.

2) Asynchronism.

The ability to accept inputs asynchronous to the computer timing signals is desirable because it affords a design with a minumum of reference to circuits outside the computer. Furthermore, it reduces the number of signals across the interface and simplifies their specification. Inputs whose functions are to interrupt normal program sequences are best designed for asynchronous operation.

3) Compatibility.

If the computer's input and output circuits are compatible, then the computer may be used to check its own interface by having the output circuits drive the various inputs. We first learned of this scheme with the VERDAN computer. and have come to believe it is a very powerful checking method, In the AGC all input and output signals (except power) are compatible and suitable for self checking,

4) No analog transmissions in magnitude form.

For the sake of accurate transmission and standardization of the many system interfaces, all analog signals are sent and received by the AGC in pulsed form. Analog inputs to the AGC are partially digitized and transmitted as increments to be counted, Analog outputs are in the form of durations, the conversion being performed by logic internal to the computer, The application of this principle affords a clear distinction between the computer and the subsystems it controls; a fact which works to the benefit of both.

5) Output feedback.

The ability to check on the results of output commands is clearly essential for fault detection and for self correcting operation.

B. Inputs

1) Input registers.

There are several input registers which may be interrogated under program control. The individual bits are flip-flops set either by external switch closures or by incoming pulses through transformers. The bits are reset only upon interrogating the register, This is an instance of the influence of electrical isolation upon the logic of the computer. In order to sample pulse trains and to treat the presence or absence of such trains as if they were switch closures, it is necessary

either to rectify and filter or to use the pulses to set flip-flops. Asynchronous sampling of the pulse train line may otherwise yield false results by sampling between pulses. Note, however, that having input bits reset only upon interrogation of the input register means that the information so obtained may be old and not reflect the present state of the input signals. The present state of the input lines is detected by interrogating twice in succession. The pulse frequency of the incoming train should be high enough to ensure at least one incoming pulse per instruction.; otherwise programs must provide suitable delays,

2) Interrupts.

Some incoming signals are used to trigger program interrupts. For example, the "or" of all the lines coming from the keyboard triggers one specific interrupting program, which then interrogates the particular input register involved. The keyboard character so obtained is stored elsewhere in erasable memory, and a request is entered into an executive, or dispatcher, program. The original interrupted program is then resumed at the point of interruption. Eventually, the program control is returned to the executive program which schedules further processing of the keyboard character. Interrupting programs are short and usually involve sampling of input bits and entering requests into the executive program.

3) Increments.

Much of the incoming information consists of trains of pulses which must be counted, and the running total must be available to the program at all times. There are several such inputs plus certain internally generated signals which are counted in order to keep time. The requirement of asynchronism is met by a circuit known as the Counter Priority Circuit, which stores incoming pulses until they can be processed. Processing consists of interrupting the normal sequence of instructions at a time when both the adder and the address register are free

and executing a special incrementing sequence. The sequence consists of reading out of a specified register in the erasable memory.(the "counter"), adding (or subtracting) one to (from) it, and writing the new word back into memory. The normal instruction sequence is then resumed. Each increment requires a memory cycle (~ 12 μ sec). The time spent incrementing counters is proportional to the number of pulses counted, each of which requires a twelve microsecond memory cycle. Naturally, this time is not available for normal program executions, but this system is efficient for average aggregate pulse rates below 5 Kpp. It is economical of equipment, and it can withstand peak rates an order of magnitude larger than the average rate.

Since each running total is kept in a specific location in erasable memory, all such totals are available to programs. Interference between a possible instruction fetch and an interrupting increment does not occur because incrementing can only take place following a complete memory cycle.

Counter incrementing is a specialized form of interrupt: such interrupting programs may last up to several hundred memory cycle times, as opposed to one memory cycle time per increment.

4) Serial to parallel conversion.

In addition to making analog to digital conversions, the counter incrementing system is also useful for processing asynchronous serial information. The incrementing sequence is modified to shift the word coming from the "counter," adding the incoming information (a zero or one) to the vacated position, and restoring the word back to memory. Serial information must be on two separate lines, one for zeros and one for ones. A leading one may be used to alert the computer to the fact that a whole word has been entered, by causing an overflow when it

is shifted past the highest bit position. The overflow causes an interrupt which removes the completed word from the "counter," leaving it empty and ready to accept more information.

C. Outputs

Outputs from the AGC may be divided into two major logical categories: those which are under program control, and those which are not. The latter, which form a substantial portion of all outputs, include a number of different fixed frequencies, reset pulses, and some alarm indications. The kinds of outputs under program control are pulse bursts and relay contact closures.

Program control of outputs is exercised by means of output registers. These registers are addressable, and their contents may be sampled without affecting them. The individual bits (output bits) serve as the inputs to some further logical networks which culminate in an output circuit consisting of either a relay driver or a pulse transformer.

The main problem in relating the AGC to various peripheral subsystems is that of converting a result which is an n-bit word into suitable serial commands. For purposes of this discussion it will be assumed that the peripheral subsystems consist of several imperfect stepping motors. These motors have encoders on their shafts which feed the shaft motions back to the computer in pulse form to be counted as described in the section on increments. The motors are imperfect in that a pulse sent to them may not always cause a corresponding shaft motion, and in that the pulse rate may have an effect on the motor response.

Typical commands to be sent to the motors range from tens of pulses to a few thousand pulses, at frequencies which may reach several kilocycles. The time necessary for sending the pulses is large enough to require that the AGC be able to

send the series of pulses at the same time that it proceeds with further computations. Simultaneous control of several motors aggravates the problem.

One solution is to provide a rate multiplier [8], [3], in which the output bits control the selective addition of fundamental frequencies into a single line. By setting output bits, the outgoing pulse rate may be made to have any desired average frequency from the lowest fundamental to the sum of all fundamentals. In this scheme the number of necessary output bits to control a motor is determined by the desired dynamic range of the output.

A scheme which uses fewer output bits is that of sending pulses at some high fixed rate and varying the number of pulses sent.* This scheme is illustrated in Figure 8; it consists of counting the outgoing pulses in a counter preset by the program, When'the counter overflows, it sets all the output bits which control the active motor to zero, thus suppressing further pulses.

The pulse burst method is a form of binary to pulse-duration conversion, provided that each pulse sent out lasts the whole of the interval between pulses. It is electrically inconvenient to transmit long ($\approx$0.5 msec) pulses through transformers because of the excessive amount of iron required. To achieve the effect of pulse-duration modulation the system is arranged as in Figure 9. The various motors, as well as other subsystems, have receiving flip-flops which are set by the pulses of the burst and reset by a common reset pulse source. Because both set and reset pulses are derived from a single oscillator,

*This "Pulse Burst" system is more closely related to position control than to rate control, so for direct comparison between it and the rate multiplier method, one should take into account the overall control problem and motor dynamics'.

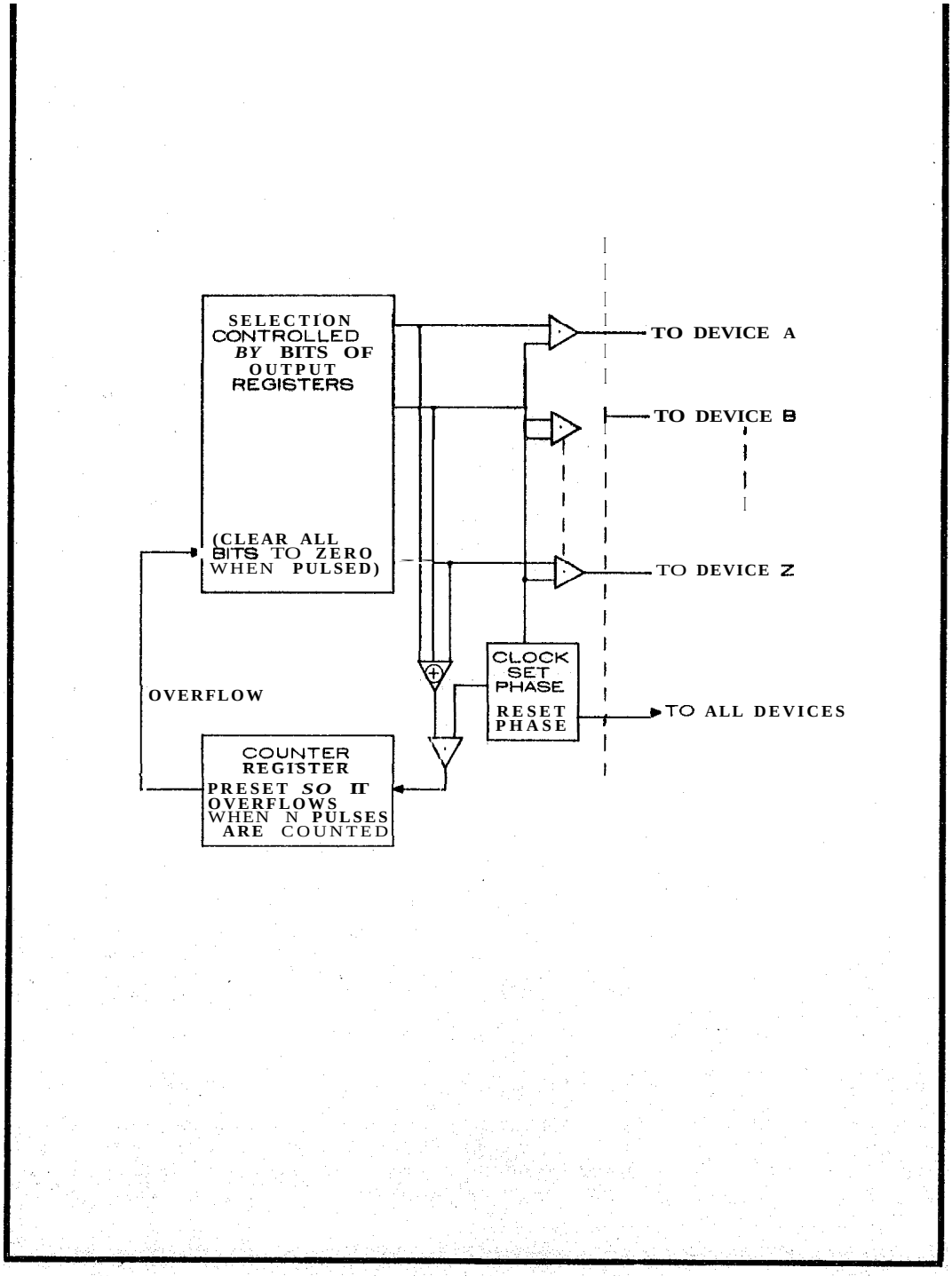


Fig. 8 Method for sending bursts of N pulses to a selected device.

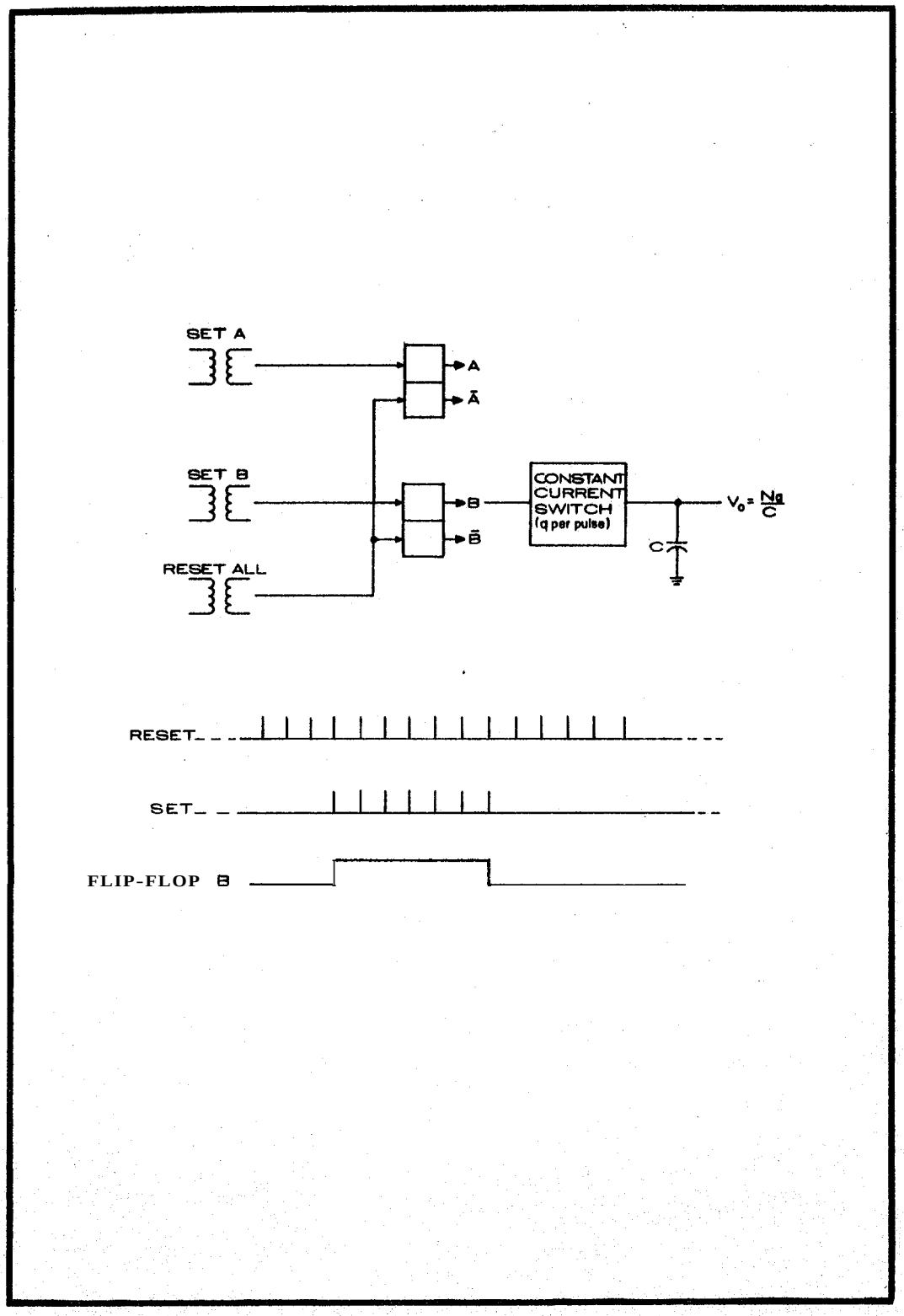


Fig. 9 Example of pulsetrain to du.ration to voltage conversion.

the interval between their leading edges may be well controlled, which in turn permits a variety of relatively simple digital to analog conversion techniques, For example, conversion from pulse duration to a d-c voltage requires but a single accurate constant current switch. The dynamic range of such a system is limited by the stability of the single current source, the stability of the set-reset timing characteristics, and the discharge characteristics of the load supplied by the integrating capacitor (Figure 9). Ey comparison, an N-resistor ladder converter depends on the relationship between the N resistors, as well as on the quality of the voltage reference source,

BIBLIOGRAPHY

[1] A. L. Hopkins, R. L. Alonso, H. Blair-Smith, Logical Description of the Apollo Guidance Computer (AGC 4), MIT Instrumentation Laboratory Report R-393, March, 1963.

[2] A. L. Hopkins, Design Concersts of the Apollo Guidance Computer, MIT Instrumentation Laboratory Report R-408, July, 1963.

[3] R. L. Alonso and J. H. Laning, Jr., Design Principles for a General Control Computer, MIT Instrumentation Laboratory Report R-276, April, 1960.

[4] R. L. Alonso, H. Blair-Smith, A. L. Hopkins, Some Aspects of the Logical Design of a Control Computer; A Case Study, to be published.

[5] D. Shansky, Erasable Store Mod 3C, MIT Instrumentation Laboratory Report E-1158, July, 1962.

[6] E. T. Walendziewicz,, "The D210 Magnetic Computer," Proceedings of the Spaceborne Computer Conference, IRE, October, 1962.

[7] "Core Rope Memory," Computer Design, June, 1963.

[8] B. M. Gordon, "Adapting Digital Techniques for Automatic Controls," I and II, Electrical Manufacturing, November and December, 1954.

R-416

DISTRIBUTION LIST

Internal

R, Alonso
J. Arnow (Lincoln)
R. Battin
W. Bean
E. Berk
P. Bowditch
A. Boyce
R. Boyd
P. Bryant
R. Byers
G. Cherry
E. Copps
W. Crocker
G. Cushman
J. Dahlen
M. Drougas
E. Duggan
J. Dunbar
K. Dunipace (MIT/AMR)
R, Euvrard
P. Felleman
S. Felix (MIT/S&ID)
J, Flanders
J. Fleming
L, Gediman
F. Grant
Eldon Hall
I. Halzel
D. Hanley
W. Heintz
E. Hickey
D. Hoag
A. Hopkins
F, Houston
M. Johnston
B. Katz
A. Koso
M. Kramer
W. Kupfer
A. Laats
D. Ladd
T. Lawton
D. Lickly
R. Magee
G. Mayo
J. McNeil
R. McKern
James Miller
John Miller
J. Nevins
G. Nielson
J. Nugent
E. Olsson
C. Parker
J. Potter
K. Samuelian
P. Sarmanian
R. Scholten
J. Sciegienny
N. Sears
D. Shansky
T, Shuck
W. Stameris
E. Stirling'
R, Therrien
W. Toth
M. Trageser
R. Weatherbee
L. Wilk
R. Woodbury
W. Wrigley
Apollo Library (2)
MIT/IL Library (6)

External

(ref. APCAN; 2 July 1963)

P. Ebersole (NASA/MSC)		(2)
W. Rhine (NASA/RASPO)		(1)
S. Gregorek (NAA S&ID/MIT)		(1)
AC Spark Plug		(10)
Kollsman		(10)
Raytheon		(10)
WESCO		(2)
Capt. W. Delaney (AFSC/MIT)		(1)
NAA RASPO:	National Aeronautics and Space Administration Resident Apollo Spacecraft Project Officer North American, Inc, Space and Information Systems Division 12214 Lakewood Boulevard Downey, California	(1)
CAPE:	National Aeronautics and Space Administration Atlantic Missile Range Operations Port Canaveral, Florida Attn: Mr. B. P. Brown	(3)
HDQ:	NASA Headquarters 1520 H Street Washington, D. C. Attn: Mr. G. M. Low, MD(P)	(6)
AMES:	National Aeronautics and Space Administration Ames Research Center Moffett Field, California Attn: Mr. Matthews	(2)
LEWIS:	National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio	(2)
FRC:	National Aeronautics and Space Administration Flight Research Center Edwards AFB, California	(2)
JPL:	National Aeronautics and Space Administration Jet Propulsion Laboratory Pasadena California Attn: Mr. H. R. Lawrence	(2)
LRC:	National Aeronautics and Space Administration Langley Research Center Langley AFB, Virginia Attn: Mr. A. T. Mattson	(2)

GSFC:	National Aeronautics and Space Administration Goddard Space Flight Center Greenbelt, Maryland	(2)
MSFC:	National Aeronautics and Space Administration George C. Marshall Space Flight Center Huntsville, Alabama Attn: Dr. Kuettner	(2)
GAEC:	Grumman Aircraft Engineering Corporation Bethpage, Long Island New York Attn: Mr. A, Whitaker	(1)
NAA:	North American Aviation, Inc. Space and Information Systems Division **12214** Lakewood Boulevard Downey, California Attn: Mr. R, Berry	(1)
GAEC RASPO:	National Aeronautics and Space Administration Resident Apollo Spacecraft Project Officer Grumman Aircraft Engineering Corporation Bethpage, Long Island New York Attn; Mr. Jack Small	(1)
WSMR	National Aeronautics and Space Administration Post Office Drawer D White Sands Missile Range White Sands, New Mexico	(2)
MSC:	National Aeronautics and Space Administration Manned Spacecraft Center Apollo Document Control Group (SDG) Houston 1, Texas	(45)