UNCLASSIFIED

[redacted]

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

APOLLO

GUIDANCE AND NAVIGATION

Approved Milton B. Trageser Date 3/5/63
MILTON B. TRAGESER, DIRECTOR
APOLLO NAVIGATION AND GUIDANCE PROGRAM

Approved Roger B. Woodbury Date 3/5/63
ROGER B. WOODBURY, ASSOCIATE DIRECTOR
INSTRUMENTATION LABORATORY

Declassified
~~DOWNGRADED~~ PER AUTHORIZATION OF
NASA | MSC LETTER
DATED July 8, 1966

(Unclassified Title)

R-393

UNCLASSIFIED

LOGICAL DESCRIPTION FOR THE APOLLO GUIDANCE COMPUTER (AGC 4)

by
Albert Hopkins
Ramon Alonso
Hugh Blair-Smith

[redacted]

INSTRUMENTATION LABORATORY

CAMBRIDGE 39, MASSACHUSETTS

COPY # 291 OF 423 COPIES
THIS DOCUMENT CONTAINS 116 PAGES

UNCLASSIFIED

[redacted]

ACKNOWLEDGMENT

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

CONTENTS

Chapter Page

ILLUSTRATIONS

TABLES

PREFACE

This report discribes the logical structure of the APOLLO guidance and navigation computer. A previous computer, AGC 3, designed for the APOLLO mission, was predominately composed of core-transistor logic. The computer design described here employs miniature integrated NOR logic, whose use will result in the next APOLLO computer (AGC 4) being just over half the size of AGC 3.

The decision to change over to integrated circuitry was made in October, 1962. About a year ago, it was deemed inadvisable to commit the APOLLO Guidance Computer (AGC) to integrated circuitry. Its desirable attributes of small size, high speed, and universality were then offset by its high cost, the difficulty in regulating power consumption as a function of speed of computation, and the absence of operational experience in large scale systems. Because of its potential, however, a computer-design investigation was conducted with integrated circuits at the Instrumentation Laboratory during the development of AGC 3.

Now, a year later, the price of integrated circuit elements has changed from high to moderate; and enough experience has been gained in their use, by MIT and by others, to permit extrapolation of their reliability data with substantial confidence. The adoption of the new technology, with the consequent redesign of the computer, is being undertaken at a time when it is felt that it can still be effected without causing undue delays in the program.

Since the first design of AGC 3 of about a year ago, much has been learned about the capabilities demanded of the APOLLO computer; enough programming experience has been gained to warrant the inclusion of programming features not present in AGC 3 and the exclusion of others that were. Consequently, AGC 4 is sufficiently different from AGC 3 to make existing AGC documents inadequate for use in further developing the guidance system and its production and support facilities. The prime purpose of this report is to furnish necessary information to members of the Laboratory and its contract and industrial support associates. Fine detail and internal consistency have been underemphasized for the sake of promptness so that this report could be written within a few weeks of the inception of the design.

GENERAL REFERENCES

MIT/IL Report E-1077	Preliminary Mod 3C Programmers Manual by R. Alonso, J. H. Laning, Jr., and Hugh Blair-Smith
MIT/IL Report E-1126	AGC Mod 3C Computer Circuits - General by A. Hopkins

ACKNOWLEDGMENTS

The design described here came about through the efforts of many people. The work was done under the general direction of Eldon C. Hall, who is in charge of the Digital Development group.

The electrical design of AGC 4 has been greatly aided by David Shansky, who has produced several circuits, including the Erasable Memory, and who has engaged in extensive negotiations with component manufacturers.

L. David Hanley carried out the unified evaluation of integrated components by redesigning sections of AGC 3 with them. Many of the present AGC 4 circuits are based on his earlier designs; the decision to change to integrated circuits was influenced by his work in a major way.

Robert Oleksiak, Robert Scott, Samuel Scott, and Edwin Smally have helped in developing memory circuits and test equipment and in formulating the AGC 4 memory cycle. Alan Green, Charles Muntz, Joseph Rocchio, and Richard Warren programmed simulations and generated display and test routines for AGC 3, which have influenced the design of AGC 4. John Deyst and Thomas Lawton first proposed the present output pulse rate generation method. Dr. J. H. Laning, Jr., who was a partner in earlier computer design efforts, devised the original key WAITLIST and EXECUTIVE routines for AGC 3.

Finally, the authors want to acknowledge the substantial contribution made to this work by Herbert Thaler of the Raytheon Missiles and Space Division. His collaboration in all of the areas of this report has made its rapid preparation possible and has stimulated the progress of the entire computer design. William Lund, also of Raytheon, has aided in later phases of the design.

Chapter 1

INTRODUCTION TO AGC 4

Because of the many similarities between AGC 3 and AGC 4, and because most of the people for whom this report is primarily intended are familiar with the characteristics of AGC 3, the essential features of AGC 4 can be discussed by comparing them with those of AGC 3.

The principal difference between the two computers is in the difference between core-transistor logic and NOR logic. In the former, a binary ONE is represented by a current path established by a transistor switch in a conductive state. Complemented variables are not generally available, and variables stored in cores are available one time only, at the time the cores are reset. In NOR logic, a binary ONE has a voltage representation. Each stage performs inversion so that complements are readily available; variables are available on a DC basis. Central registers consist of flip-flops instead of cores, and are more expensive in terms of cost, size, and power. They are faster, though, and will be operated at a 1-μsec word rate as compared to the 5-μsec rate of the core registers. Moreover, they may be read nondestructively and cleared without putting their contents onto the Read Bus.

The Fixed and Erasable Memories of AGC 3 are used in AGC 4. To take advantage of the faster word-transfer rate in central registers, the cycle times are reduced from an average of 19.5 μsec to a fixed 11.7 μsec. By fixing the memory cycle, the timing of the various control functions may be set for maximum reliability and speed.

The memory cycle is broken down into twelve steps, or time pulses, each approximately one microsecond long. Within each step, a number may be transferred from one flip-flop register to another. AGC 4 instructions consist of an integral number of memory cycle times (generally two) instead of an integral multiple of eight 5-μsec pulse times (generally one) as in AGC 3. Because of the high cost of flip-flop registers, the number of central registers in AGC 4 is less than in AGC 3; and all of the editing registers which are not central are relocated into the Erasable Memory.

The timing of the memory cycle does not permit reading directly from memory to the Write Buses without reserving the buses for several time pulses which could otherwise be used for data transfer. For this reason, a memory-buffer register, called the "G register," is incorporated into AGC 4. The G register is loaded from Erasable or Fixed Memory before the seventh time pulse and must be prepared for writing into the Erasable Memory before the tenth.

The Write Buses in AGC 4 communicate among the G, Central, input, and output registers, and the arithmetic unit. Since all of these have flip-flop storage, there is no need for a storage medium in the Write Buses such as the Write Latches of AGC 3. The Write Buses are simply multi-input amplifiers.

In AGC 3, the add time and parity-generation time were smaller than a number transfer time. In AGC 4, they are greater; consequently, the adder and parity circuits need input storage, whereas output storage was used in AGC 3. Three microseconds are allowed for additions and two for parity detection, including the time for writing in and reading out.

Three new instructions have been added to AGC 4: Mask, Divide, and Subtract. An extra bit is added to the three-bit operation code by causing a negative overflow to take place within an Index instruction and using the uncorrected sign bit in conjunction with the sign bit in selecting operations. Multiply, Divide, and Subtract are the three instructions which require negative overflow on Index for selection.

Table 1-1 lists some of the principal logical attributes of AGC 4 and compares these with equivalent properties of AGC 3. Certain numbers in the table, such as the numbers of counters, interrupts, and inputs and outputs, are descriptive of the guidance system interface rather than of logical limitations of the computers.

Table 1-1. Comparison of AGC 3 and AGC 4 Characteristics

Characteristic	AGC 4	AGC 3
Word Length: 16 bits (15 bits + parity)		
Number System: "1's" complement, with overflow correction		
Memory Cycle Time	11.7 μsec	Av. 19.5 μsec
Wired-in memory (Core Rope)	12,288 words	12,288 words
Erasable memory (Coincident current Ferrite)	1008 words	992 words
Normal order code	11 instructions	8 instructions
Involuntary instructions (Interrupt, Increment, Load, Start)	8 instructions	6 instructions
Add instruction time	23.4 μsec	39 μsec
Double precision Add subroutine (X + x) + (Y + y) = (Z + z)	234 μsec	≈ 1 msec
Multiply	98 μsec	634 μsec
Double precision multiply subroutine	780 μsec	≈ 4 msec
Counter incrementing	11.7 μsec	19.5 μsec
Aggregate input rate at which instructions are executed at half speed	43 Kpps	25.6 Kpps
Number of counters	20 counters	20 counters
Interrupt options	5 options	5 options
Discrete input lines (one input bit per line)	60 lines	60 lines
Discrete outputs (for displays)	18 lines	18 lines
Pulsed outputs under program control	25 lines	25 lines
Pulsed outputs not under program control	16 lines	16 lines
Telemetry: Single error correcting pulse train. Output asynchronous to AGC timing.		
Up Link: Serial input to one register, rates up to 5 words per second, asynchronous.		

This page intentionally left blank.

Chapter 2

AGC 4 SYSTEM

GENERAL ORGANIZATION

The principal structure of AGC 4 is shown in figure 2-1. The various Central registers are shown at the left of the drawing and are denoted A, Q, Z, etc. All of these are flip-flop registers, and all except B, X, Y, and P are addressable. The Central registers communicate with one another and with the rest of the computer via the Write Buses. Gating pulses for reading out of, and writing in to, the Central registers are formed in the control pulse amplifiers. For addressable registers, the gating pulses are in part dependent upon the memory selection logic.

Memory addressing is effected via the S register and the Bank register, which is used for memory bank selection. The bank register is addressable; the S register is not. Bank selection is used to choose one of ten groups of 1024 words of Fixed Memory to be the fourth of four addressable 1024 word segments of memory. This means that octal addresses 6000 to 7777 refer to one of ten groups of 1024 words, depending on the state of the Bank bits. All addresses below octal 6000 have the same meaning regardless of the state of the Bank bits. The first segment of 1024 words comprises the Central and Erasable Memories. The second and third segments are always the same two groups of fixed storage.

Whenever the Erasable or Fixed portion of memory is addressed, the information is read to the G register prior to time pulse seven. The G register (not addressable) communicates with the Write Buses in normal fashion for reading out and in one of five modes for writing in. The choice of mode depends upon the address stored in the S register. When a cycling or shifting register address is stored in S (these registers are CYR, SR, CYL, and SL), the appropriate mode is used to effect the desired editing transformation on the word being written into G for storage in the Erasable Memory; otherwise, the normal mode is used.

The S and SQ registers together add up to sixteen bits: the SQ register uses four bits, and the S register the remaining twelve bits.

Output signals are formed by logical functions of output register bits and time signals from the scaler. Input signals belong to one or both of two classes: those which appear as bits in input registers and those which request counter increments or program interrupts. Those signals belonging to the latter class operate a priority network which causes the instruction selection logic to interject an increment or interrupt sequence as

soon as permissible. The priority network also supplies to the Write Buses the counter address or the interrupt sequence address.

CLOCK AND SCALER

The oscillator for the AGC is the frequency standard for all of the APOLLO guidance and spacecraft systems. It is a 2.048 megacycle, crystal-controlled, transistor oscillator with an oven for thermal regulation. The computer uses as its clock signals four phases of the 1.024 MC square wave obtained from a binary division of the oscillator output. One more binary division produces the 512 KC signal, which serves as a synchronizing signal to the spacecraft systems clock.

The Scaler provides timing signals for the operation and synchronization of the electromechanical parts of the guidance system and for other sequential control processes with which the computer is concerned.

Figure 2-2 illustrates the organization of the clock and scaler. The 512 KC square wave is gated to the Scaler by the start-stop logic and is scaled by a factor of five to a frequency of 102.4 KC. All of its subharmonics are generated by powers of two down to 0.390625 cps, this last frequency being used to control the computer in the standby (low power) mode. Chapter 5 of this volume contains detailed information concerning the uses of the various intermediate frequencies and the method of standby operation.

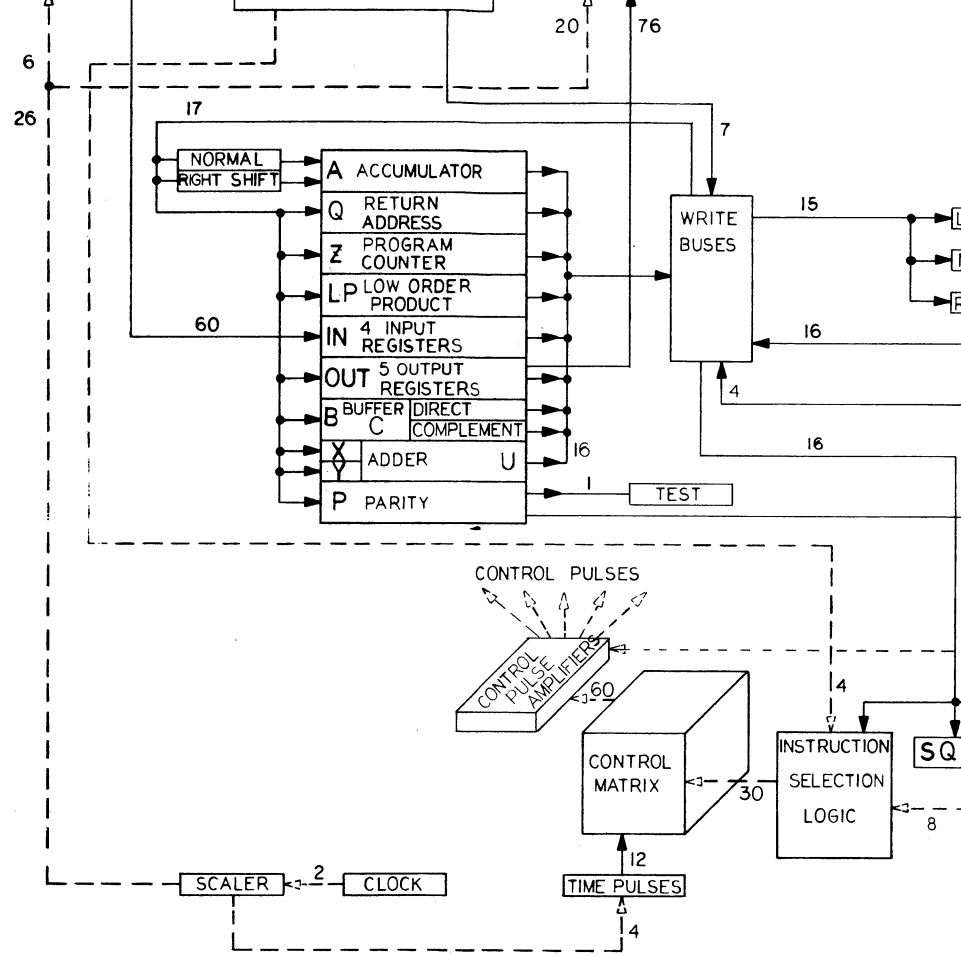
Under control of the Monitor, an external ground support device, the start-stop logic can cause the scaler to halt. This permits certain aspects of real time operation to be studied at slow speed or with manual interventions. *

The twelve time pulses are generated by means of a four-stage, gray-coded counter advanced by gated phases of the computer clock. Like the scaler, the time-pulse generator can be halted by the Monitor.

SEQUENCE GENERATOR

The operation of the sequence generator is functionally depicted in figure 2-3. Time Pulse 12 is reserved for instruction selection. If a new instruction is to be initiated, "read B" (RB) and "write SQ" (WSQ) control pulses are generated at Time 12. The operation code in the upper four bits of B is transferred into the SQ register, where it remains throughout the execution of the instruction. The stage counter is cleared by the coincidence of Time 12 and the absence of inputs to its first section. The contents of SQ and the stage counter uniquely select a subinstruction memory cycle, which will be executed in eleven steps unless a counter increment has been requested. In that case, the selected subinstruction is inhibited during any increment cycle servicing the counter.

*When the Monitor is not connected to the computer, the computer is always operating in real time, either at full speed or in the standby mode.



NOTES:

1. SOLID ARROWS DENOTE FLOW OF NUMERICAL DATA OR OF INSTRUCTION AND ADDRESS INFORMATION. DASHED ARROWS DENOTE CONTROL SIGNALS.
2. A NUMBER WRITTEN BY AN ARROW INDICATES THE NUMBER OF SIGNAL LINES REPRESENTED BY THE ARROW.

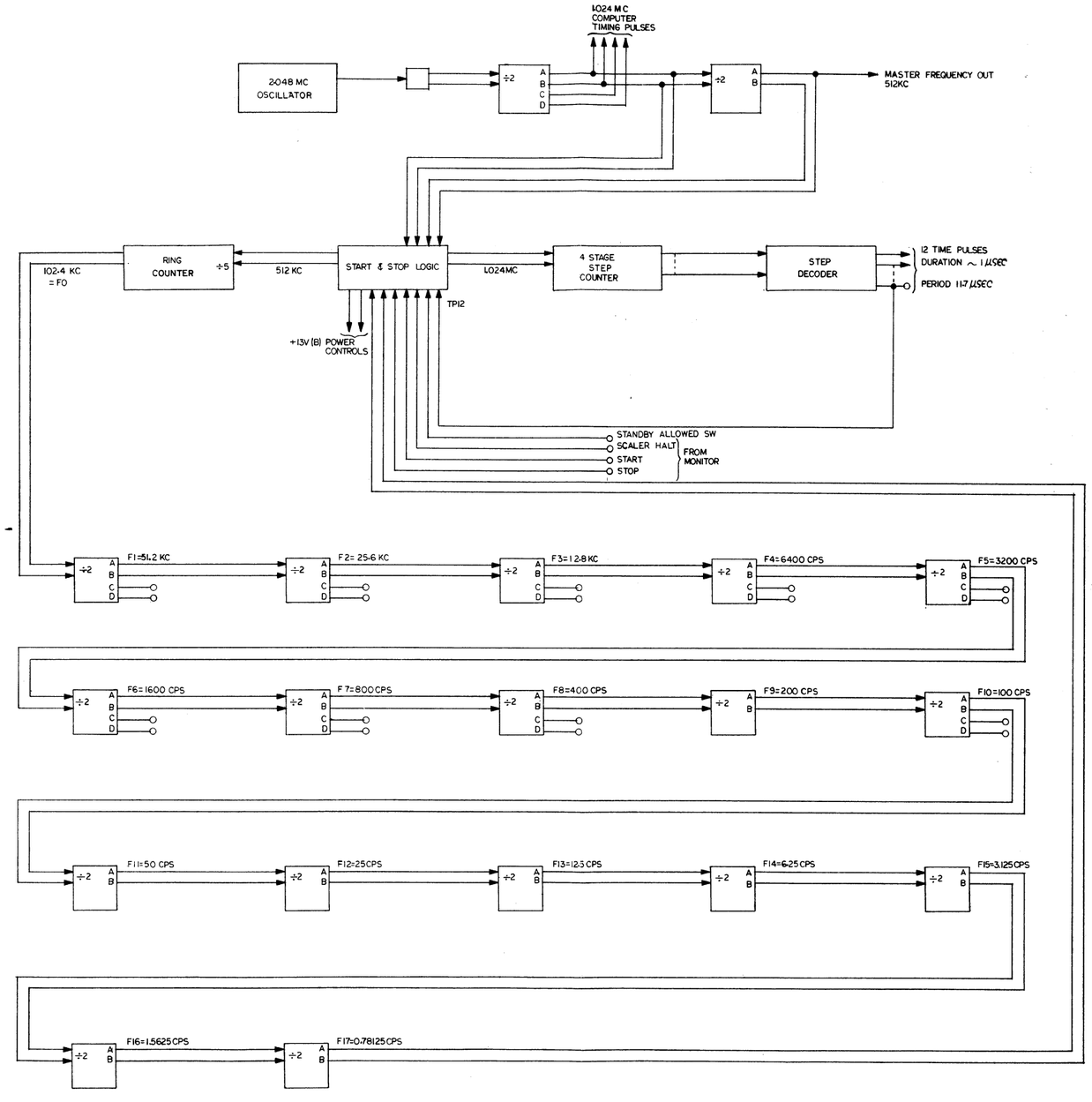


Figure 2-2. Clock and Scaler Block Diagram.

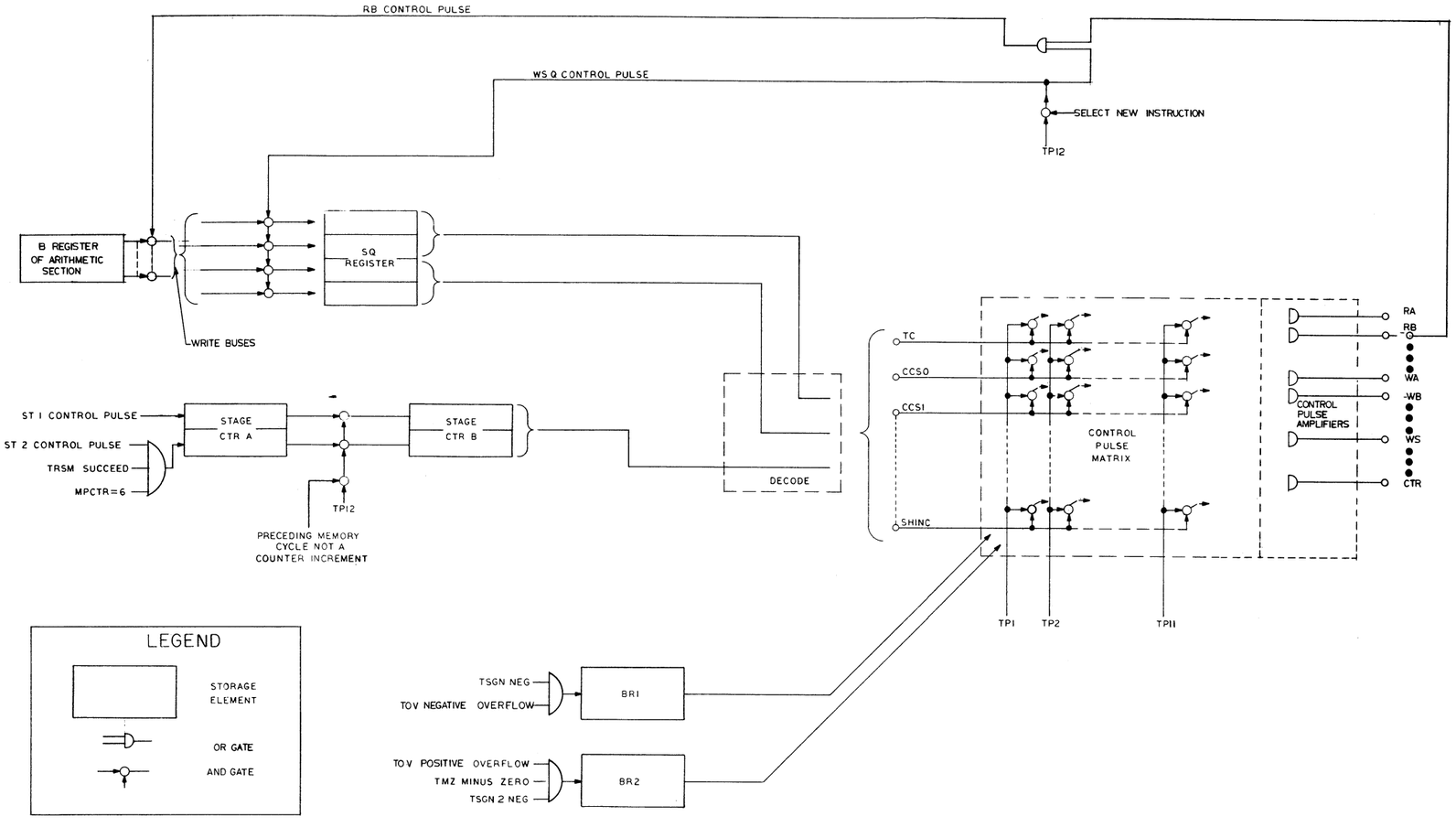


Figure 2-3. Sequence Generator Block Diagram.

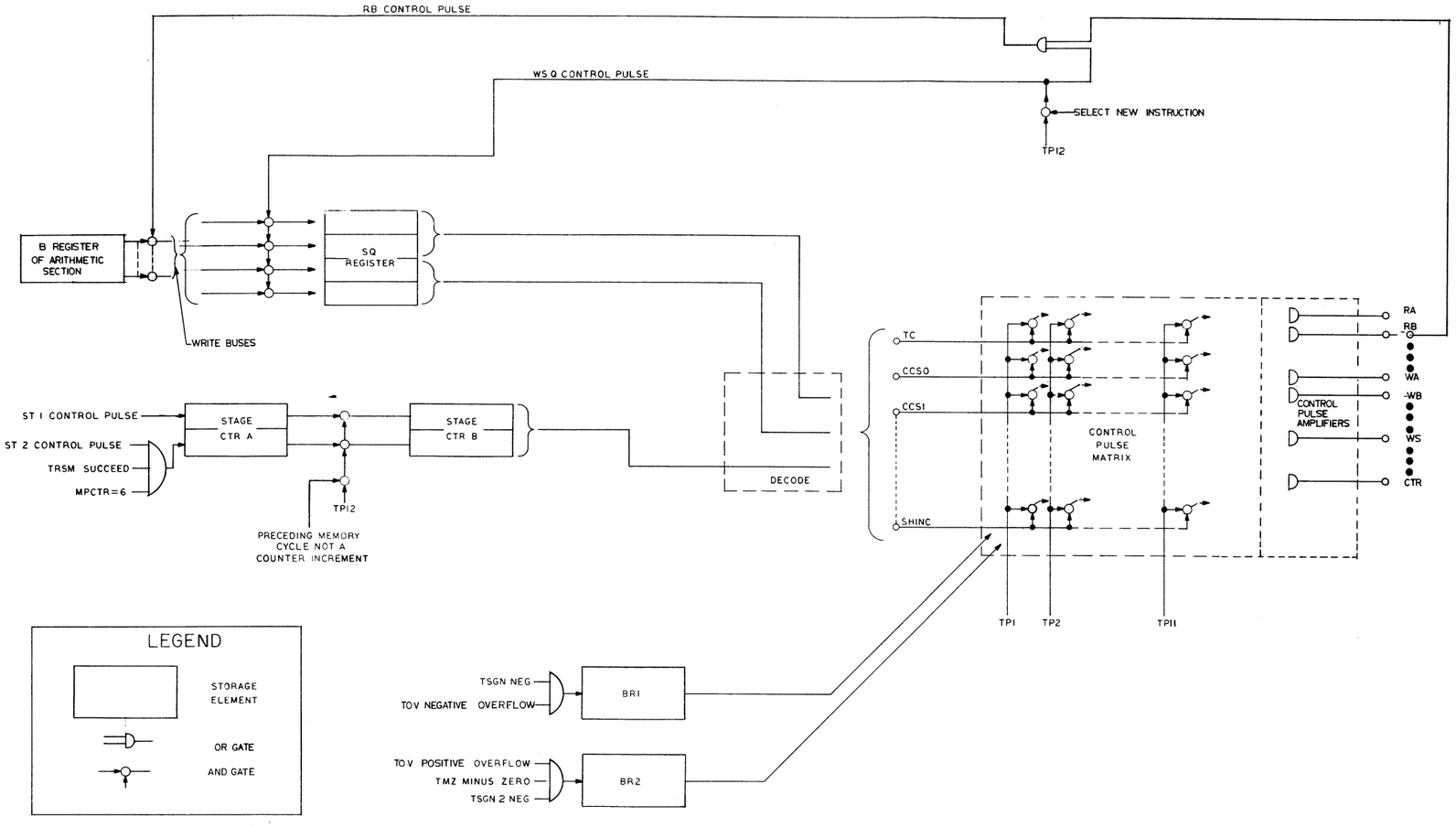


Figure 2-3. Sequence Generator Block Diagram.

The following Time 12 does not disturb SQ or the stage counter in order that the selected instruction be executed after all counter requests are fulfilled.

If an instruction consists of more than one memory cycle, as is true of all but TC and the three counter incrementing sequences, the stage counter is advanced by control pulses (ST1, ST2, TRSM, CTR) at some time during the first 11 time pulses; and a transfer to the "B" section of the stage counter occurs at Time 12.

A Control pulse is generated as the logical sum of a subset of the matrix crosspoints. A crosspoint, in turn, is the logical product of a subinstruction variable and a time pulse, and may produce as many as five control pulses by driving five different logical sum circuits.

Branching is effected by storing the result of a test in a flip-flop and using the flip-flop outputs to inhibit certain crosspoints. In some instances, two branch conditions are stored simultaneously, thereby permitting a selection from among four crosspoints at one time of one subinstruction. In figure 2-3, the four branch tests are indicated setting BR1 or BR2 for their various conditions.

CENTRAL PROCESSING SECTION

The circuits which constitute the Central registers, input and output registers, the G and S registers, and the Write Buses, are physically organized into sixteen identical, interchangeable units called "bit sticks." Each bit stick contains one bit of each register and its connections to and from the Write Buses. The sixteen bit sticks, together with a few others, constitute the Central Processing Section, about half of the micrologic part of the computer.

The AGC adder is a 16-bit parallel adder with end around carry. It operates in a modified "1's" complement system in which the sign bit is processed in two adjacent columns. The two sign columns of the sum are identical unless overflow occurs on the addition. If overflow does occur, the leftmost sign bit is the same as the original sign of the operands; and it is adopted as the sign (SG) of the sum. Thus it can be said that sign is preserved on overflow.

Some of the arithmetic and logical machine processes require preservation of the rightmost sign bit, the uncorrected sign (US). For this reason, the A, Q, Z, LP, and B registers contain columns for storage of the uncorrected sign. These registers have no parity bit position; hence, the bit stick which contains the parity positions of the G and Output registers also contains the US positions of the other Central registers and of the Write Buses.

Since words coming from Erasable or Fixed Memory contain no overflow information, all transfers from G to the Central registers duplicate the sign of G into the US position of the Write Buses.

The functions of the various Central registers are given in detail in chapter 3 of this volume. Briefly, the arithmetic is all done in the adder, which has flip-flop storage for two operand words X and Y. The correct sum of X and Y is present at the output of the sum circuit U within three microseconds of the time at which the read-in of the second operand (always X) occurs. It is also possible to add one to a positive number in X or Y by means of a "Carry In" (CI) control pulse. Three microseconds are sufficient for generating the correct sum in this case, too.

The B register is primarily for temporary storage. The C register is fictitious; C is the complement side of the B register. The A register is also used for temporary storage. It is called the accumulator and retains data from one instruction to the next. The LP register, together with the A register via right shift gates, forms a double-length shifting accumulator for multiplication.

The input and output registers are flip-flop registers whose assignments and functions are described in chapter 5.

Table 2-1 is a list of Special registers, including addressable Central registers, their octal addresses, and their physical locations.

Table 2-1. AGC 4 Special Registers

OCTAL ADDRESS	REGISTER NAME	LOCATION	OCTAL ADDRESS	REGISTER NAME	LOCATION
0000	A	Located in Bit Sticks	15	Bank	Located in Service Sticks
1	Q		16	Relint	
2	Z		17	Inhint	
3	LP		0020	CYR	Located in Erasable Memory
4	IN0		21	SR	
5	IN1		22	CYL	
6	IN2		23	SL	
7	IN3		24	ZRUPT	
0010	OUT0		25	BRUPT (RIP)	
11	OUT1		26	ARUPT	
12	OUT2		27	QRUPT	
13	OUT3				
14	OUT4 (DOWNTEL)				

MEMORY OPERATION

The signals which operate the Erasable and Fixed Memories are generated during every subinstruction (with two exceptions) as functions of the time-pulse generator, the clock, and the contents of the S and Bank select registers. If the S register contains the address of a flip-flop register, neither memory is operated; if not, then only the appropriate memory is operated. The memory is not operated during the Multiply and Divide repetitive loops, MP1 and DV1.

Chapter 4 contains further details of memory selection and timing.

PARITY TEST AND GENERATION

The purpose of the parity circuit is to ensure that the numbers stored in Erasable and Fixed Memory are read correctly. Single errors and all other odd multiple errors cause an alarm to be signalled to the computer and the display panel. The parity bit is chosen so that the number of "1's" in a word is odd. In Fixed Memory, this bit is wired in along with the rest of the information. In Erasable Memory, it must be generated each time a new word is stored. It is convenient, in fact, to generate a new parity bit each time a word is read so that any wrong parity indication will not persist. In the Special registers, no parity test is made because of the difficulty of generating the parity bit for words produced by arithmetic, logical, and editing operations.

Figure 2-4 illustrates the flow of parity information. A single circuit, called a "pyramid," is sufficient to generate and test a parity bit. If "read G" (RG) and "write P" (WP) gates are on at the same time, the pyramid produces even number indications of the incoming word, both with and without the parity bit. These two indications are called the P-15 and 1-15 indications, respectively. If the source of the word is not G, then both outputs produce a 1-15 indication.

The parity output of G has only a single destination, which is the write gate of the P register. The position in the Write Buses corresponding to the parity bit is used for the uncorrected sign, as mentioned in the Central Processing Section of this chapter, and receives the SG bit from G.

During memory cycles in which information is brought from memory, the process of parity testing is initiated by RG and WP control pulses. A later Test Parity (TP) pulse generates an alarm if an even number of "1's" was written into P.

When no new information is being written into Erasable Memory, the word just read is regenerated except for the parity bit. The latter is generated by the 1-15 indication of the pyramid and gated into the G register by a Generate Parity (GP) control pulse.

When a new word is stored, the same method is used except in the Exchange instruction. There, it is necessary to store the generated parity bit while the pyramid is used to test another incoming word. The P2 register serves that purpose. A Read P2 (RP2) control pulse produces the same result from P2 as does GP from the 1-15 indication.

Finally, for Down telemetry, it is required to store a parity bit in the OUT4 register; there is a special gate which is enabled only when OUT4 is addressed. This gate transmits the parity bit as required.

COUNTER INCREMENTING

Counters in AGC 4 are addressable registers in the Erasable Memory. The inputs to these counters are trains of pulses which are first stored in a special set of circuits called a Priority Chain. The Priority Chain is shown symbolically in figure 2-5. The cells labeled P_1, P_2, P_3, ... P_{20}, each represents a pair of flip-flops and some extra gates; each flip-flop stores one of the two possible incoming pulses (e.g., a pulse for increment or a pulse for decrement). If either flip-flop of P_i is set to "1", which means that the counter i is to be serviced, then a "1" is transmitted through gate G_i into the Counter Address Generation Network; and all other gates G_j, where $J > i$, are blocked. Gate G_i will transmit a "1" into the Counter Address Generation Network only if no other earlier counter G_k, where $k < i$, requires processing.

Should simultaneous requests be made, the counters are serviced in preassigned order, hence, the name "Priority Chain."

The cells P_i each have two outputs, such as C_i +, C_i -, or C_i +, C_iS. These outputs are "1" if the corresponding storage flip-flops are on, and if the corresponding gages G_i are not blocked.

The signals C_i +, C_i -, and C_iS are grouped into three OR gates; the outputs from these OR gates go to the Sequence Generator, where they are used in the selection of a PINC, MINC, or SHINC sequence.

When an increment is received, the address of the appropriate Erasable register is supplied through the Address Generation Network; information for the selection of the appropriate sequence is provided to the Sequence Generator; and activity relating to all counters of lower priority is inhibited temporarily. Inputs to lower priority counters are not lost, however, since they are stored in the P cells.

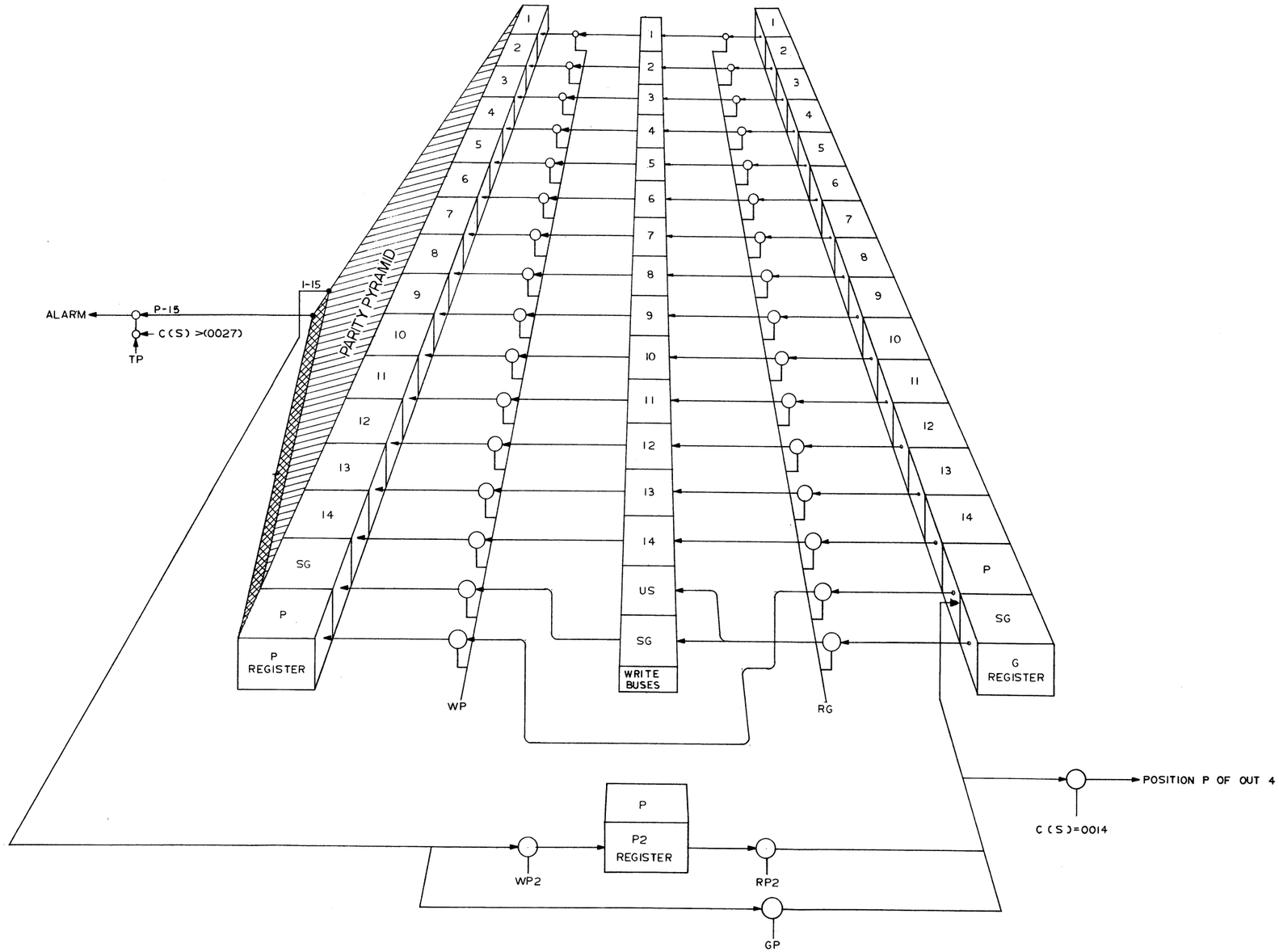


Figure 2-4. Parity Test and Generation Block Diagram.

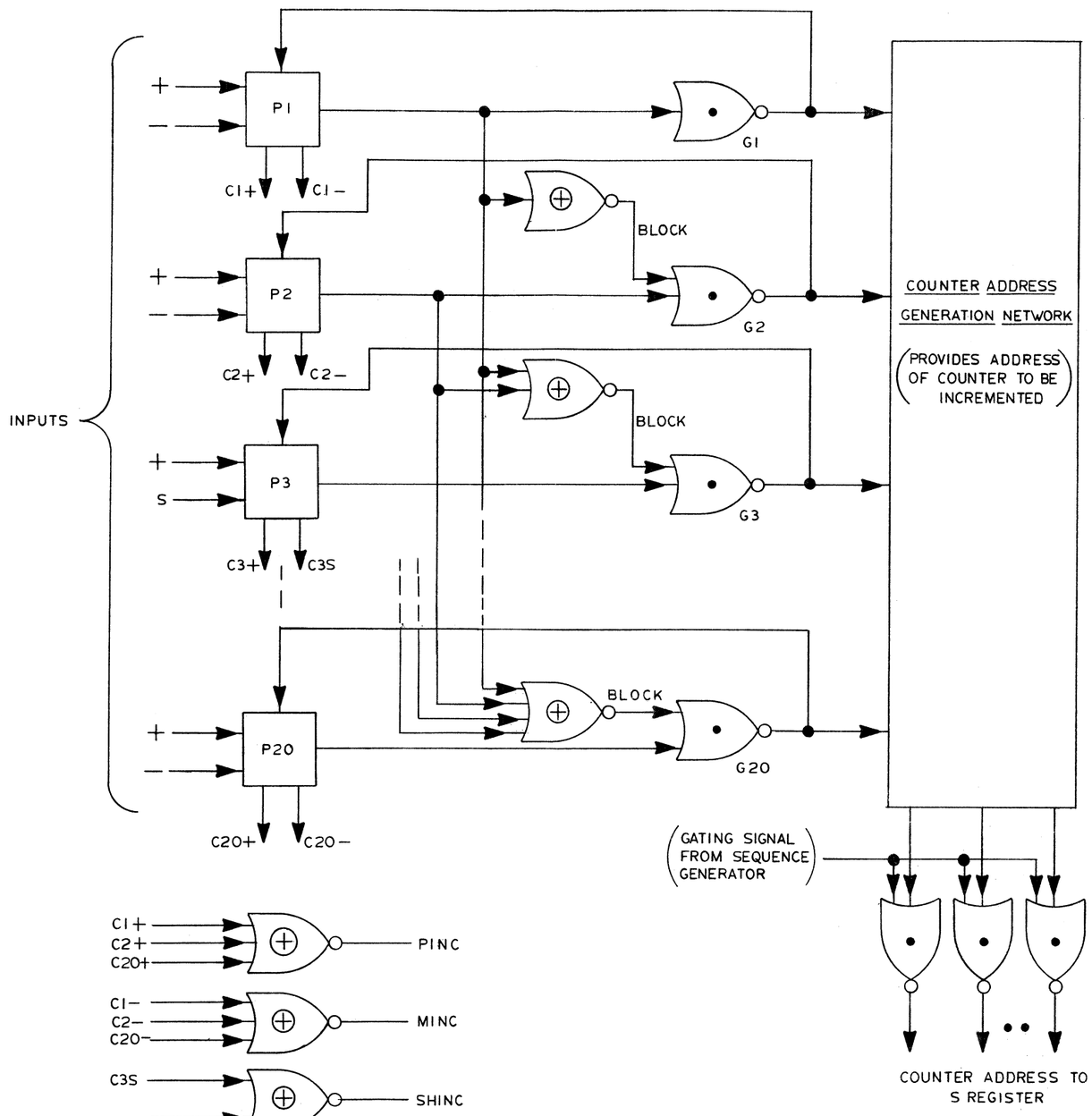


Figure 2-5. Priority System for Counters.

When the Sequence Generator reaches the end of a memory cycle, a test is made to see if any of PINC, MINC, or SHINC are requested. If so, the outputs of all P cells are temporarily (3 μsec) prevented from changing state; and the counter address and proper sequence are recorded in the address register S and in the Sequence Generator, respectively. Blocking the changes of state of outputs of all P cells is necessary in order to prevent an ill-timed input pulse from changing a counter address in the middle of the sampling period. Such an incoming pulse is not lost, however, since there are holding flip-flops before the blocking gate inside the P cells.

The counter to be serviced is then incremented after the end of a memory cycle, essentially by inserting an incrementing memory cycle before proceeding with the next "normal" cycle called for by the instruction being executed.

If more than one counter input is to be serviced, the counters are incremented in order of descending priority until there are no further increments to be made. The same test made at the end of "normal" memory cycles is made at the end of an incrementing cycle. The only thing not "normal" about an incrementing cycle is that it was not called for by an instruction.

A detailed description of the steps of PINC, MINC, and SHINC are found in chapter 3.

While normal computations proceed at half speed, AGC 4 is capable of an aggregate counting rate of about 42 Kpps. The relationship between counting rate and computing speed is an inverse one; if the aggregate input rate is about 21 Kpps, the computing speed is 75% of maximum.

It is difficult to speak of maximum counting rate for counters in an absolute sense. If there are no other counters receiving inputs, then a given counter will lose no counts if its inputs are below 80 Kpps. It is also true that no pulses will be lost if all counters receive simultaneous inputs which are at least 240 μsec apart (about 4 Kpps). In both of the above cases, however, no computing other than counting would be done, since the end of every memory cycle would find a new request for servicing a counter.

If a counter input circuit P receives simultaneous plus and minus inputs, a single incrementing cycle occurs which increments the counter by zero; i.e., there is no net change in the contents of that counter.

A 20-counter system requires about 250 NOR gates, including those necessary for generating the address of the counter.

PROGRAM INTERRUPTION

The normal sequence of instructions can be interrupted by a certain kind of signals called RUPT's (for interrupts). These signals are derived from certain other signals, which go to INBITS, or by the overflow of some counters. (See chapter 5.) The Priority Chain associated with RUPT's is much the same as that of the counters, but the resulting actions are more complicated for RUPT's than for increments.

A RUPT causes a transfer of control to some prespecified program. At the end of that program, a RSM (for resume) sequence takes place, which sets the AGC arithmetic and control units back to the exact state they were in at the time of interruption. To do this, the RUPT sequence must preserve, and store in a safe place, the contents of B and Z; the interrupting program further preserves A and Q.

Interrupts themselves cannot be interrupted. If an interrupting program is in progress, and a RUPT signal comes in at this time, the new request is preserved and acted upon only after the original interrupting program has ended and the RSM sequence has taken place.

It is sometimes desirable to inhibit interruptions for a brief period. This can be done by the instruction INDEX INHINT. The address INHINT (octal 0017) is interpreted as meaning that RUPT signals should not be processed until further notice. "Further notice" is defined as the instruction INDEX RELINT (for Release Interrupt). Address RELINT is octal 0016. The inhibition of interrupts is used sometimes when dealing with input and output registers.

WAITLIST AND EXECUTIVE PROGRAMS*

One of the most important problems in general purpose control computers is that of starting and stopping programs for which real time is a variable. The WAITLIST and EXECUTIVE programs should, for this reason, be considered to be an integral part of AGC 4. These programs do not presently exist for AGC 4, but they do exist for AGC 3; the differences between the two versions are expected to be minor.

The descriptions given below, although superficial, are included in order to give some idea as to the nature of such programs.

*Both WAITLIST and EXECUTIVE programs were first written by J. H. Laning, Jr., for the Mod 3C Computer, later AGC 3.

WAITLIST

This program allows a desired subroutine, called "task," to be entered at a specified future time. The format for such requests is as follows:

L	XCH	ΔT
L + 1	TC	WAITLIST
L + 2	TASKADDRESS	

The contents of register ΔT is a time interval Δt, the time from "now" at which the desired task is to be executed by means of an interrupt. The contents of L + 2, "TASKADDRESS," is the fourteen-bit address of the desired task. As a matter of convention, "tasks" refer to programs executed by WAITLIST, while "jobs" refer to programs under control of the EXECUTIVE, described in the next section. After making the request, the program continues at L + 3.

There may be up to seven different task requests waiting to be processed at one time. If two of these tasks should happen to request the same time of execution, then the first one requested is executed first; and the second one as soon as the first task is completed. The end of a task is signalled by the instruction TC TASKOVER. The program TASKOVER sees to it that any additional tasks requested for that time are executed before resuming the interrupted program.

The WAITLIST routine makes use of the Time 3 counter, which is incremented at the rate of 100 pps. A task request essentially results in presetting Time 3 so that it will overflow at the proper time. When Time 3 overflows, it triggers the T3RUPT interrupt. The interrupt program then determines which task is called for at this time and takes appropriate action, after which it resumes the original at the point of interruption. A common type of action is to enter a job request into the executive system of routines (of which EXECUTIVE and WAITLIST are a part). This last is done to avoid lengthy interrupting programs which, in turn, prevent other interrupting signals from being serviced.

A task request may be self-perpetuating. After a task is done, but before transferring to TASKOVER, a new request is made for the same task to be executed Δt seconds ahead.

EXECUTIVE

This program will arrange the order of processing of up to seven independent programs, each of which has a preassigned priority ranking. The AGC 4 programs will have certain points at which the EXECUTIVE can suspend that program's computations and initiate processing of a program of higher priority. When the higher priority job is

done, the interrupted job is resumed. Unlike the "hardware" priority circuits, the EXECUTIVE will allow for the interruption of interrupted programs. To initiate a job, programs FINDVAC or NOVAC are used in the interrupt mode or with interrupt inhibited. FINDVAC is used if the job desired is written in interpretive language or uses a vector accumulator; NOVAC is used if the job requested does not use a vector accumulator and is in basic language.

Entry into the EXECUTIVE is via the sequence:

L	XCH	PRIO
L + 1	TC	FINDVAC
L + 1	TC	NOVAC
L + 2	JOBADDRESS	

c(PRIO) is a priority designation for the job, and c(L + 2) = JOBADDRESS is the fourteen-bit address of the requested job. After making the request, the program continues at L + 3.

To terminate a job, it should end with the instruction TC ENDOFJOB if the job written in was basic AGC language, or with the interpretive instruction RTB ENDJOB1 if the job was written in interpretive. "RTB" stands for "Return to Basic."

All AGC programs which qualify as jobs have, at suitable points, the instructions:

L	CCS	NEWJOB	
L + 1	TC	CHANG1	(if present job is basic)
L + 1	TC	CHANG2	(if present job is interpretive)

This pair of instructions allows the EXECUTIVE to replace the present job with one of higher priority, if one exists.

The time required for the EXECUTIVE for AGC 4 is not known exactly at this time. Assuming, however, that the AGC 4 program is like that for AGC 3, then the speeds are approximately as follows:

	AGC 3	AGC 4
NOVAC	(27 + 7P) I	2(27 + 7P) MCT
FINDVAC	(40 + 7P) I	2(40 + 7P) MCT
CHANG2	50 I	100 MCT
ENDJOB1	(91 + 11P) I	2(91 + 11P) MCT

where: P is the number of jobs outstanding.
I is the AGC 3 instruction time (40 μsec).
MCT is the AGC 4 Memory cycle time (12 μsec).

Chapter 3

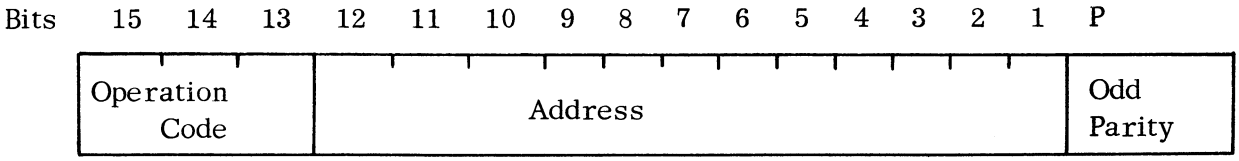
AGC 4 INSTRUCTIONS

GENERAL REMARKS

There are eighteen sequences in the repertoire of AGC 4: eight basic instructions, three extracode instructions, four unprogrammed sequences, a special case of one of the basic instructions (resume), and two sequences that operate under control of the system test equipment. The last two are of no concern to programmers and are mentioned here only for completeness.

Each sequence consists of one or more subsequences lasting one memory cycle time (MCT). 1 MCT is nominally 12 μsec; actually it is a little less.

The format of an instruction word is:



In the notation of this memo, L is the address of the location from which the instruction is taken. For any register or location K, c(K) denotes the contents of K; b(K) denotes initial ("b" is for "before") contents of K wherever the distinction clarifies the discussion. The results of addressing special registers referenced explicitly by instructions are given in the "Special Cases: Implied Address Codes" section of this chapter.

Table 3-1 shows the special registers, excluding unaddressable central registers. A, Q, Z, and LP figure prominently as arithmetic or control registers. It should be noted that an initial condition for an instruction in L is bits 12-1 of C(Z) = L + 1. Normally bits 16-13 of C(Z) are zero. Table 3-2 shows how the registers are written into. Note particularly that A has two independent sign bits. When a word is brought into A from Fixed or Erasable Memory, its sign bit goes into both positions. When C(A) is stored in Erasable Memory, bit 16 is stored as the sign. The quantity stored is said to be overflow-corrected because, if bits 16 and 15 of C(A) had become different due to an arithmetic overflow, the stored sign is that of the operands of the overflowing addition. One implication of these facts is that overflow is not so much an event as it is a class of quantity retainable in a special register; the descriptions in the next section treat it accordingly.

Table 3-1. AGC 4 Special Registers

OCTAL ADDRESS	REGISTER NAME		OCTAL ADDRESS	REGISTER NAME	
0000	A	Flip-Flop Registers	0020	CYR	(In Erasable)
1	Q		21	SR	
2	Z		22	CYL	
3	LP		23	SL	
4	IN0		24	ZRUPT	
5	IN1		25	BRUPT (RIP)	
6	IN2		26	ARUPT	
7	IN3		27	QRUPT	
0010	OUT0		0030-56	COUNTERS	
11	OUT1				
12	OUT2				
13	OUT3				
14	OUT4				
15	Bank				
16	Relint	No bits in these registers			
17	Inhint				

Table 3-2. How Registers are Written Into in AGC 4

Write Pulses	REG.	REGISTER BIT POSITION 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Comments
WA W0	A	SG	US	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
WALP	A	SG	SG	US	14	13	12	11	10	9	8	7	6	5	4	3	2	
WQ W1	Q	SG	US	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
WZ W2	Z	SG	US	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
WLP W3	LP	1	1	-	14	13	12	11	10	9	8	7	6	5	4	3	2	
WALP	LP	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	
WB	B	SG	US	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
W10-W14	OUT0-OUT4	SG	P	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
SBE WG SBF	G	SG	P	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Memory into G
W20	G	1	-	SG	14	13	12	11	10	9	8	7	6	5	4	3	2	Information into CYR
W21	G	SG	-	SG	14	13	12	11	10	9	8	7	6	5	4	3	2	Information into SR
W22	G	14	-	13	12	11	10	9	8	7	6	5	4	3	2	1	SG	Information into CYL
W23	G	SG	-	13	12	11	10	9	8	7	6	5	4	3	2	1	SG	Information into SL
WS	S					12	11	10	9	8	7	6	5	4	3	2	1	
WX	X	SG	US	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
WY	Y	SG	US	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
WP1	P1	SG	P	14	13	12	11	10	9	8	7	6	5	4	3	2	1	For Test Parity
WP2	P2	SG	-	14	13	12	11	10	9	8	7	6	5	4	3	2	1	For Generate Parity
WBK	BNK	-	-	14	13	12	11	-	-	-	-	-	-	-	-	-	-	
WSQ	SQ	SG	US	14	13	-	-	-	-	-	-	-	-	-	-	-	-	

Table entries show which Write Latch connects to that Register Bit Position

P = Parity
SG = Sign
US = Uncorrected Sign

BASIC INSTRUCTIONS

These eight instructions are called basic because, unlike the extracodes, they are completely specified by an instruction word. The code number accompanying each description is the octal digit in bits 15-13 of the instruction word.

Code 0. L: TC K Transfer Control 1 MCT

Set C(Q) = TC L + 1;

Take next instruction from K, and proceed from there.

Code 1. L: CCS K Count, Compare, and Skip 2 MCT

Set C(A) = DABS [C(K)] , where the DABS function (Diminished Absolute Value) of an integer α is defined as =

$$\text{DABS}(\alpha) = \begin{matrix} |\alpha| - 1, \text{ if } |\alpha| > 1 \\ +0, \text{ if } |\alpha| \leq 1; \end{matrix}$$

Set C(K) = b(K);

Take next instruction from L + n, and proceed from there, where

n = 1 if C(K) > 0

n = 2 if C(K) = +0

n = 3 if C(K) < 0

n = 4 if C(K) = -0.

Remarks: K may not be in Fixed Memory; a parity error and wrong answer would result. This is no restriction because nothing is gained by testing the contents of Fixed Memory.

If K is a flip-flop register, the sign bit (position 16) is tested.

Note that for any quantity α, $\alpha - \alpha$ = -0 in "1's" complement notation.

Code 2. L: INDEX K (K $\neq$ 0025) Index 2 MCT

Use the sum of C(L + 1) + C(K) as the next instruction just as if that sum had been taken from L + 1; set C(K) = b(K).

Final C(L + 1) = b(L + 1).

Remarks: One of eight operation codes distinct from the basic set results if the sum contains an overflow. Three of these (the extracodes) are used; interrupt cannot occur at the end of an INDEX that overflows.

Code 2. L: RESUME (= INDEX 25) Resume 2 MCT

Set C(Z) = C(24);

Enable further interrupts.

Use C(25) as the next instruction.

Remarks: This brings about resumption of an interrupted program because the interrupt sequence stores in 24 (ZRUPT) the contents of Z at the time of interruption and in 25 (BRUPT) the contents of B at that time. Interrupt occurs between instructions, at which time C(B) is the upcoming instruction and C (Z) is $L' + 1$, where L' is the address of the source of C(B). Notice that RESUME is INDEX 25 and not TC 25.

Code 3. L: XCH K Exchange 2 MCT

Set C(A) = b(K);

Set C(K) = b(A) unless K is in fixed memory;

Take next instruction from L + 1.

Code 4. L: CS K Clear and Subtract 2 MCT

Set C(A) = -C(K);

Set C(K) = b(K);

Take next instruction from L + 1.

Remarks: The minus sign, here and elsewhere, indicates "1's" complement.

Code 5. L: TS K Transfer to Storage 2 MCT

Set C(K) = b(A);

If b(A) contains no overflow, C(A) = b(A);

Take next instruction from L + 1;

If b(A) contains overflow, take next instruction from L + 2 and set

C(A) = 000001 for positive overflow, or

C(A) = 177776 for negative overflow

(The leftmost digit is binary, representing the sign bit. The uncorrected sign bit is accounted for in the digit second from left.

Remarks: The "skip on overflow" feature of TS is used chiefly in double precision arithmetic.

Code 6. L: AD K Add 2 or 3 MCT

Set C(A) = b(A) + C(K);

Set C(K) = b(K);

Take the next instruction from L + 1.

Remarks: If C(A) contains positive (negative) overflow after the addition, the overflow counter OVCTR (octal address 0034) is incremented (decremented) by one, which accounts for the third MCT. Also, an interrupt can never occur while C(A) contains an overflow. The effect on OVCTR is an explicit property of the AD and SU instructions and not of overflows in general.

Code 7. L: MASK K Mask 2 MCT

Set C(A) = b(A) $\wedge$ C(K);

C(K) = b(K)

Remarks: "$\wedge$" is the Boolean operator AND applied in each bit position.

EXTRACODE INSTRUCTIONS

These three instructions are formed by overflow of the addition in INDEX. They cannot be completely stored in memory, but conventional octal codes are stipulated, assuming a conventional way of causing the overflow. If no address modification is desired, this will be done by an EXTEND instruction which is simply INDEX 5777. The programmer is responsible for supplying in 5777 the octal constant 47777. If address modification of an extracode instruction is desired, the technique is to index by the sum of octal 47777 and the modifying quantity. This implies that, if an argument address is used several times to modify extracode instruction addresses, indexing of extracodes is virtually free. (See the examples of double-precision multiplication.)

In the following descriptions, the conventional octal code is shown as the code number; but it is understood that the code does not have the meaning shown unless it is preceded by the proper INDEX 5777 instruction.

Code 4. L: MP K Multiply 10 MCT

Set C(A, LP) = $b'(A) \times C(K)$;

Take next instruction from L + 1.

Remarks: $b'(A)$ is the overflow-corrected form of b(A); that is, overflow in b(A) is ignored. The high-order product in A and the low-order product in LP (address 0003) agree in sign, which is determined strictly by the sign bits of the operands, even if one or both operands is zero. C(K) = b(K) except when K = A or K = LP.

Code 5. L: DV K Divide 18 MCT

Set C(A) = b (A) / C(K);

Set C(Q) = - |Remainder| ;

Set C(LP) > 0 if quotient is positive,

< 0 if quotient is negative;

Take next instruction from L + 1.

Remarks: |b(A)| should be less than |C(K)| , although equal magnitudes are permissible if used very carefully — |C(A)| = 37777 and C(Q) = - |C(K)| in this case. If |b(A)| > |C(K)| , |C(A)| = 37777 and C(Q) is meaningless. Therefore, b(A) must not contain an overflow. C(K) = b(K) except when K = A, Q, or LP.

Code 6. L: SU K Subtract 4 or 5 MCT

Set C(A) = b(A) - C(K);

Set C(K) = b(K);

Take next instruction from L + 1.

Remarks: This instruction is identical to AD in all other respects.

SPECIAL CASES; IMPLIED ADDRESS CODES

The descriptions in the foregoing "Basic Instructions" and "Extracode Instructions" sections assume that the address K is in nonediting Erasable or in Fixed Memory, except as noted. The phrase "Set C(K) = b(K)" appears in most of the descriptions as a shorthand indication that C(K) remains unchanged unless:

1. K = LP, CYR, SR, CYL, or SL, in which case b(K) is edited by one of the editing functions shown in table 3-2 to produce the final C(K).

2. K is the address of one of the central registers always used by the instruction. It is not obvious in this case how the instruction behaves.

There is an exception to Rule 1 in MASK, MP, and DV, as stated in their descriptions. Also, in the very unlikely case that an instruction is taken from an editing register, Rule 1 holds for the instruction except for the instruction executed after a CCS. The function of this section is to state the implications of Rule 2 for each instruction.

The following can also be considered as a list of special instructions. The mnemonic of the instruction is on the left, and the means for getting it executed is explained on the right-hand text.

RESUME, which is done by INDEX 25; this is a unique combination because it actually performs a special pulse sequence during the second MCT; and

EXTEND, done by INDEX 5777.

Three more are defined which do not have to do with the consequences of Rule 2:

INHINT, done by INDEX 17, which inhibits interrupt until further notice;

RELINT, done by INDEX 16, which enables or "releases" interrupt (no storage is associated with addresses 16 and 17, which behave as if they always contained +0, so that neither INHINT nor RELINT affects the next instruction); and

XAQ, done by TC A, the mnemonic being "Execute C(A) using Q." The action is straightforward, but it appears to do an out-of-line execution of C(A) because Q follows A in the address pattern. In nearly all cases, it is better to write

INDEX	A
0	0

an important exception being the case that the instruction in A is a TC.

The remaining special actions are introduced as they come up in the following paragraphs, which discuss the consequences of Rule 2.

TC uses Z and Q. TC Z is not useful. TC Q is useful and is represented by the mnemonic code RETURN. Let b(Q) = TC K, then the action of TC Q is, in 2 MCT:

Set C(Q) = TC 2;

Take next instruction from K, and proceed from there.

Remarks: Not useful if b(Q) is not a TC; see cautionary note under TCAA (TS Z).

CCS uses Z and A. CCS Z might be useful; its action is, in 2 MCT:

Set C(A) = TC L;

Take next instruction from L + 1.

CCS A is definitely useful; its action is, in 2 MCT:

Set C(A) = DABS $\left[b(A)\right]$;

Take next instruction from L + n, and proceed from there,

where n = 1 if b(A) > 0,

n = 2 if b(A) = +0,

n = 3 if b(A) < 0,

n = 4 if b(A) = -0.

Remarks: Bit 16 is regarded as the sign. The DABS function is computed on the full 15 bits and sign of b(A), so that C(A) contains an overflow if $\left|b(A)\right| \geqq 40001$.

INDEX uses Z. INDEX Z might be useful for a small program to operate anywhere in Erasable Memory; its action is, in 2 MCT:

Use the sum of C(L + 1) + L + 1 as the next instruction just as if that sum had been taken from L + 1.

XCH uses Z and A. XCH Z might be useful. Let b(A) = TC K; then the action of XCH Z is, in 2 MCT:

Set C(A) = TC L + 1;

Take next instruction from K, and proceed from there.

Remarks: The effect is the same if b(A) is not a TC; but see cautionary note under TCAA (TS Z).

XCH A (mnemonic NOOP) is useful and its action is, in 2 MCT:

Take next instruction from L + 1.

Remarks: This is used for no operation whenever it is desired to save C(Q). Otherwise TC L + 1 is faster.

CS uses Z and A. CS Z might be useful; its action is, in 2 MCT:

Set C(A) = -(TC L + 1);

Take next instruction from L + 1.

CS A (Mnemonic COM) is definitely useful and its action is, in 2 MCT:

Set C(A) = -b(A);

Take next instruction from L + 1.

Remarks: The full 15 bits and sign are complemented.

TS uses Z and A. TS Z is useful and is represented by the mnemonic code TCAA (transfer control to address in A). Let b(A) = OP K, where OP is any four bits; then the action is, in 2 MCT:

If b(A) contains no overflow, C(A) = b(A);

If b(A) contains positive overflow, C(A) = 000001;

If b(A) contains negative overflow, C(A) = 177776;

Take next instruction from K and proceed from there.

Remarks: Assuming that no explicit manipulation of C(Z) intervenes, the next TC executed (from, say, location L′) will set C(Q) = OP L + 1, which in general is not a proper return address (see TC Q).

CAUTION: The first TC following a TCAA must not be a subroutine call. There is no problem if OP = 0000 (binary), but in this case XAQ is as fast and is preferred unless it is desired to save C(Q).

TS A is also useful and is represented by the mnemonic code OVSK (overflow skip). Its action is, in 2 MCT:

If C(A) contains no overflow, take next instruction from L + 1;

If C(A) contains positive or negative overflow, take next instruction from L + 2, and proceed from there.

AD uses Z and A. AD Z might be useful; its action is, in 2 or 3 MCT:

Set C(A) = b(A) + L + 1;

Take next instruction from L + 1.

Remarks: (Same as those under AD in the previous "Basic Instructions" section.)

AD A is definitely useful; its mnemonic is DOUBLE; its action is, in 2 or 3 MCT:

Set C(A) = b(A) + b(A);

Take next instruction from L + 1.

Remarks: (Same as those under AD in the previous "Basic Instructions" section.)

MASK uses Z and A. MASK Z is not useful. MASK A is equivalent to XCH A.

MP uses Z, A, and LP. MP Z is not useful. MP A is useful; its mnemonic is SQUARE, and its action is, in 10 MCT (including 2 MCT for the EXTEND instruction):

Set C(A, LP) = $b'(A) \times b'(A)$;

Take next instruction from L + 1.

Remarks: $b'(A)$ is the overflow-corrected form of b(A); that is, overflow in b(A) is ignored.

MP LP might be useful; its action is, in 10 MCT:

Set C(A, LP) = $b'(A) \times b(LP)$;

Take next instruction from L + 1.

DV uses Z, A, Q, and LP. DV Z and DV A are not useful. DV Q might be useful; its action is, in 18 MCT:

Set C(A) = b(A) / b(Q);
Set C(Q) = -|Remainder| ;
Set C(LP) > 0 if quotient is positive, or
< 0 if quotient is negative;

Take next instruction from L + 1.

SU uses Z and A. SU Z might be useful, its action is, in 4 or 5 MCT:

Set C(A) = b(A) - L - 1;

Take next instruction from L + 1.

Remarks: (Same as those under AD in previous "Basic Instructions" section.)

SU A is not useful.

Summary of Consequences of Rule 2:

1. With the exception of MP, addressing of a flip-flop register causes the full 15 bits and sign to be used as the operand.

2. Addressing registers, whether or not they are centrals used by the instruction, usually results in straightforward operation, except as follows:

   TCAA (TS Z) yields the ±1 overflow indicator in A but always transfers control to the location whose address is in bits 12-1 of C(A); that is, the skipping function is lacking.

   OVSK (TS A) skips if C(A) contains an overflow but leaves C(A) as it was.

   The complete set of mnemonic codes for which the address is implied is:

RESUME = INDEX 25		NOOP = XCH A	
EXTEND = INDEX 5777		COM = CS A	
INHINT = INDEX 17		TCAA = TS Z	
RELINT = INDEX 16		OVSK = TS A	
XAQ = TC A		DOUBLE = AD A	
RETURN = TC Q		SQUARE = MP A	

There are three mnemonic operation codes (not implied address) that may be used for clarity in place of the standard codes:

TCR = TC, indicating a subroutine call (Transfer Control, intending to Return);

CAF = XCH, indicating that an XCH addressed to Fixed acts like a Clear and Add;

OVIND = TS, indicating that a TS addressed to Fixed skips and sets C(A) = ±1 on overflow but changes no memory location.

SEQUENCES NOT UNDER PROGRAM CONTROL

There are four sequences whose execution is specified not by program but by the state of the priority chain, which is altered by events external to the computer and some internal events, such as overflow of a counter.

INTERRUPT 3 MCT

Set C(ZRUPT) = b(Z);

Set C(BRUPT) = b(B);

Reset priority signal, and inhibit interrupt until further notice;

Take next instruction from 2000 + 4n (n = 0, 1, 2, ---), where n is the interrupt type number; and proceed from there.

Remarks: At the end of each instruction, C(B) is the next instruction. An interrupt may occur at the end of any instruction except in the following circumstances:

1. An interrupting program is in progress;
2. An INHINT instruction has occurred, but no subsequent RELINT;
3. C(B) contains overflow (as from INDEX);
4. C(A) contains overflow.

The standard coding at 2000 + 4n is:

```
TS      ARUPT
XCH     Q
XCH     QRUPT
TC      RUPTPRGN
```

By convention, ARUPT = 26 and QRUPT = 27.

Note that an interrupt causes any overflow in C(Q) to be lost.

COUNTER INCREMENT 1 MCT

Set C(J) = b(J) + 1.

COUNTER DECREMENT 1 MCT

Set C(J) = b(J) -1.

COUNTER SHIFT 1 MCT

Set C(J) = b(J) + b(J).

Remarks: J is the address of the counter corresponding to the priority signal, $30 \leqq J \leqq 56$. For some counters, an overflow sets up an interrupt priority signal. Any memory cycle time may be preempted by the priority chain for a counter operation.

PULSE SEQUENCES

The skeleton upon which the AGC 4 sequences are built is the memory cycle, which occurs every MCT except in the Multiply and Divide loops and whenever a flip-flop register is addressed. The events of two different types of memory cycles are shown in figure 3-1, with the division into 1-μsec time periods called Time 1 through Time 12. The address selection register is written into at Time 1; if the address is 2000 or greater, the Rope manipulations shown at the left begin. If the address is in the range 20 to 1777, the Erasable Memory manipulations shown at the right begin. A strobe writes data from the sense lines of its memory into the memory buffer register G, which is cleared beforehand. It can be seen that information from Fixed Memory becomes available early in Time 7, and from Erasable early in Time 6. G accepts information to be stored in Erasable before Time 10 or by special early pulses in Time 10, and supplies it to the Z Inhibit Drive (ZID) during Times 10-12.

The control pulses that form the micro-operations of the sequences are listed in table 3-4 at the end of this chapter.

As many as five pulses can occur simultaneously. Table 3-5 (at the end of this chapter) shows the subsequences of twelve times (1 MCT) each. The subsequences that make up a sequence are selected in turn by changes in a 2-bit stage counter, which starts at 00 for instructions and 01 for interrupt, and is stepped explicitly by the pulses ST1 and ST2 as they occur in subsequences. Each subsequence has a name consisting of a mnemonic abbreviation and a suffix (0, 1, 2, or 3) showing the state of the stage counter when it is selected. Where no suffix is given, 0 is understood.

Flowcharts of the sequences are given in table 3-6 (at the end of this chapter), Parts 1 through 13. These are simplified to show activities of central importance to the results of each sequence, omitting parity procedures and the application or not of Rule 1 in the "Special Cases: Implied Address Codes" section. Explanatory remarks on the flowcharts comprise the bulk of this discussion.

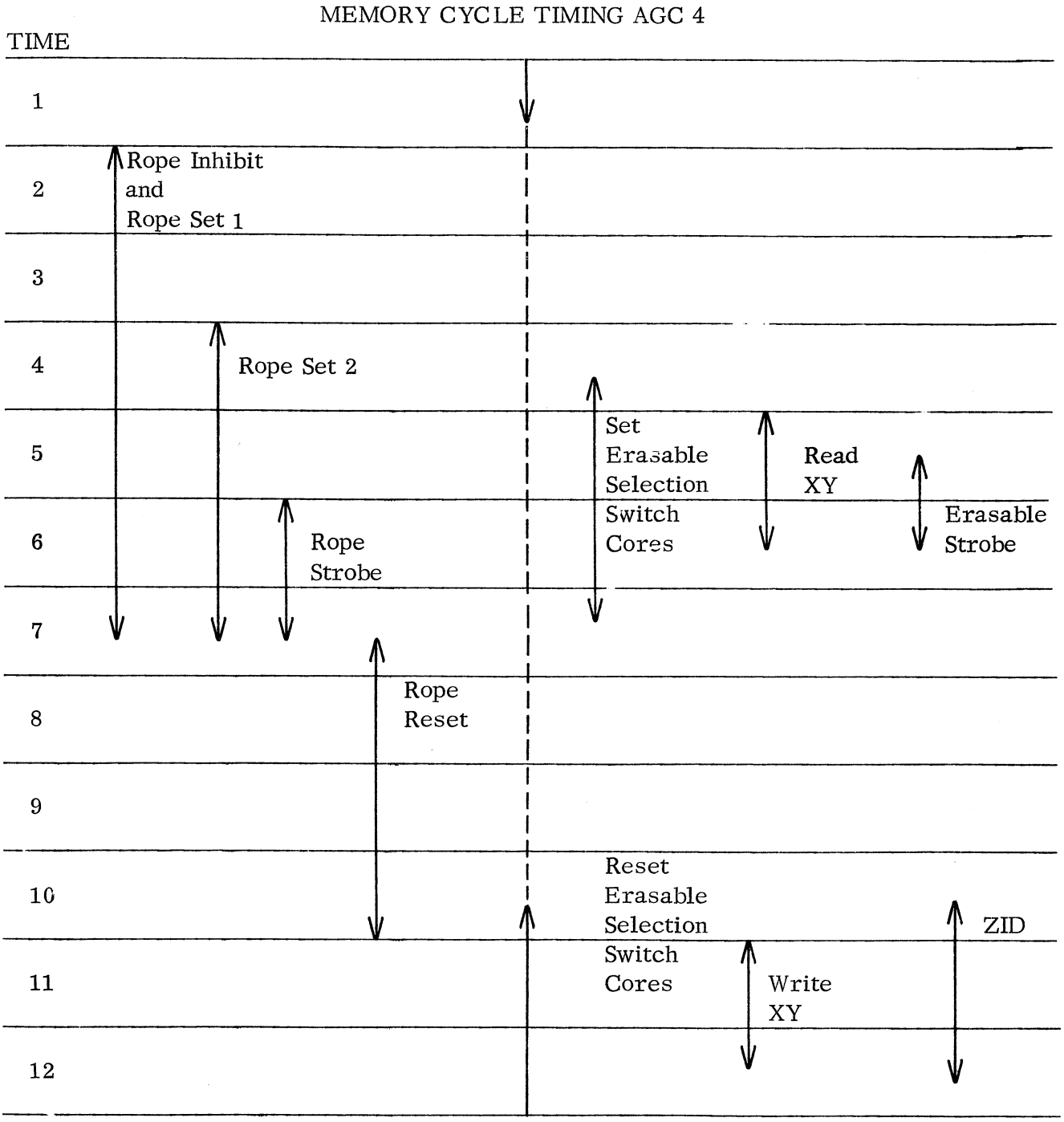


Figure 3-1. APOLLO Guidance Computer 4, Memory Cycle Timing.

Notation: INHINT 1 denotes that interrupt is prevented from occurring at the end of the current instruction; "V" denotes superposition (Boolean inclusive OR); INHINT denotes the inhibition of interrupt until further notice; other notation is that of the "Basic Instructions" and "Extracode Instructions" sections, except that "C(n) ← " is used instead of "set C(n) = ."

Boundary conditions: At the beginning of the instruction in L, C(Z) = L + 1. After Time 11 of the last subsequence in each sequence, C(B) is the next instruction. If no unprogrammed sequences are commanded, Time 12 selects the next instruction by writing C(B) into the sequence selection register SQ.

Table 3-6, Part 1: TC requires only 1 MCT because its only operation on memory is to fetch the instruction to which control is being transferred. Time 7 sets up that instruction, Time 8 the return address, and Time 10 the initial condition in Z for C(K).

Table 3-6, Part 2: Time 2 of CCS0 sets up C(Z) for the skip increment in Time 8. The argument is fetched from G at Time 6 because of the amount of work to be done in the remainder of MCT 1. This accounts for the restriction that K not be in Fixed Memory. The chief constraint on timing in CCS0 and a few other subsequences is carry propagation time in the adder; the sum is not available until the third μsec after the second addend is entered. Time 10 of CCS0 and Times 5, 8, and 10 of CCS1 compute the DABS function. An overflow check is made in Time 10 of CCS1 because b(A) could contain overflow.

Table 3-6, Part 3, shows how INDEX and RESUME share the same first MCT. Time 7 of NDX0 either sets up the indexing quantity (INDEX) or restores C(B) from BRUPT (RESUME), the semantic distinction being made by which subsequence is chosen as MCT 2. Times 6, 8, and 11 of NDX 1 perform the indexing; the overflow check is made because an extracode instruction cannot be preserved in BRUPT. Time 7 of RSM sets up the initial condition in Z for the instruction restored to B.

Table 3-6, Part 4, shows the method of XCH; the overflow check is made for the case of NOOP (XCH A) because C(A) never contains an overflow when filled from memory.

Table 3-6, Part 5, shows the method of CS; the overflow check is made for the case of COM (CS A).

Table 3-6, Part 6, shows the method of TS; the overflow check in Time 10 of TS is made for the case of OVSKP (TS A). Observe also the implications of K = Z.

Table 3-6, Part 7, shows the method of AD. The incrementing or decrementing of OVCTR occurs between the two normal MCT's of AD because a counter operation can preempt any MCT.

Table 3-6, Part 8, shows how MASK accomplishes its function using De Morgan's law $A \wedge B = -[(-A) \vee (-B)]$. Notice how MASK, like MP and DV, but unlike all other instructions, does not change the contents of an editing register when referencing it. This is done by omitting RB WG WSC at Time 9 and is intended as a useful property of MASK.

Table 3-6, Part 9, depicts the MP sequence, shows the prologue of 1 MCT, and the 6-MCT loop and 1-MCT epilogue. Broadly, the method is as follows:

1. Generate in B a number with the magnitude of the multiplicand [C(K)] and the sign of the product.

2. Generate in bits 13-1 and 16 of LP (in LP, bits 16 and 15 always agree) the right-cycled absolute value of the multiplier [b(A)], and generate in bit 14 of LP the sign of the product. (Note that this quantity is not the same as a right-cycled version of a number having the magnitude of the multiplier and the sign of the product.)

3. Generate in A a zero with the sign of the product.

4. Examine bit 16 of LP and rewrite LP, causing bits 16 and 15 to be replaced from bit 1, bit 14 to be cleared, and former bits 14-2 to be shifted right 1. If zero, shift right C(A) into A and bit 14 of LP, extending the sign in A; if one, shift right C(B) into A and bit 14 of LP, extending the sign in A.

5. Repeat step 4 thirteen times, except; if the examined bit from LP is one, shift right b(A) + C(B) into A and LP, extending the sign. The overall effect is that the partially formed product and the absolute value of the multiplier migrate to the right in a double-length shift register A and LP, separated by the product sign injected in step 2, while each multiplier bit is discarded after examination. The injected product sign ends up in bits 16 and 15 of LP; thus, both halves of the product have correct magnitude and sign.

Now refer to table 3-6, Part 9a, for the events of the prologue. The sign test in Time 2 enables the generation of the absolute value of the multiplier and, at the same time, chooses whether to use the multiplicand or its complement. Thus, the sign of the product is known even though the original sign of the multiplicand is never examined. The multiplier magnitude is stored in LP in Time 4 and retrieved in Time 5 for testing in Time 10. Times 7-9 perform the function of step 1 above and test the sign of the product. In Time 10, the product sign is injected into LP to the immediate left of the multiplier and the first (originally rightmost) multiplier bit is tested. Time 11 performs the remaining functions of steps 3 and 4 above. MP0 performs all of steps 1 through 4, although not in the same order.

The loop subsequence MP1 (table 3-6, Part 9b, top) performs twelve of the thirteen iterations of step 5 at the rate of two per MCT. The two iterations within each MCT are slightly overlapped to make use of the adder carry propagation time. C(A) is moved to the adder in Time 1 to make room in A and set up the possible addition. A multiplier bit is tested in Time 2, and the addition, if required. is begun in Time 3. The multiplier (with sign of product and partial lower product) is shifted into LP in Time 4. C(LP) is tested in Time 5 but not moved because it lacks bit 14, which is supplied in Time 6 as the sum (or b(A)) is shifted into A and LP. The action of Times 7 and 8 is analogous to that of Times 1 and 3, and that of Times 9 and 10 to that of Times 2 and 4 but without the bit test, which was overlapped into the preceding iteration. The Multiply Loop Counter is decremented and tested in Time 10 to determine the successor to Times 11 and 12. The sum from the second iteration of the loop is shifted into A and bit 14 of LP in Time 11.

In the epilogue subsequence MP3 (table 3-6, Part 9b), the last iteration of step 5 is interleaved with fetching the next instruction, which is initiated in Time 1. C(LP) is tested in Time 2 but not moved because there is no place for it to go. The adder remains busy until Time 4, when the initial condition in Z is set up for the next instruction. The final addition is begun in Times 5 and 6, leaving room for the next instruction in B in Time 7. The final lower product with its sign bit, but lacking bit 14, is formed during Times 8 and 10; and the upper product with that bit is shifted into place in Time 11.

Table 3-6, Part 10, shows the logic of the prologue, loop, and epilogue of the DV sequence. Broadly, the method is as follows:

1. Generate in bits 16 and 15 of LP the sign of the quotient.

2. Generate in Q the negative absolute value of the dividend (b(A)).

3. Generate in A the absolute value of the divisor (C(K)).

4. Set up 00001 in B, representing -37777 left-cycled.

5. Generate -2 |dividend| in Q and -2 |dividend| + |divisor| in the adder. If the difference is zero or negative, accept it in Q as the new dividend and generate a quotient bit = 1 by leaving C(B) alone; otherwise, retain C(Q) as the next dividend and generate a quotient bit = 0 by injecting a sign bit into B.

6. Cycle C(B) left. If the sign bit of the C(B) is zero, go back to step 5; otherwise, the marker bit placed in B in step 4 has arrived in the sign position; and C(B) = - |quotient| , C(Q) = - |remainder| . Now test the sign of C(LP), and put the quotient with correct sign in A.

7. Use the standard instruction-fetch subsequence.

Now refer to the top of Part 10 in table 3-6 for the events of the prologue. Times 2, 4, and 6 produce - |dividend| in Q, and Time 5 sets up the sign bits of C(LP) to the opposite of the sign of the dividend. Times 7 and 8 set up the divisor in A and test its sign. If positive, C(A) is left alone and C(LP) is read and restored to rotate the opposite sign into position. (The injection of a low-order 1 ensures that the final C(LP) is nonzero, facilitating subsequent testing by CCS if desired.) If the divisor is negative, C(A) is set to the negative of the divisor. Thus, in all cases, C(A) = |divisor| after Time 10. Time 11 sets up in B the marker bit, which will also be the legitimate sign of - |quotient| .

The operation of the loop DV1 depends on some peculiar properties of the editing function in AGC 4. If S contains the address of one of the four Erasable editing registers, data passing from a central register from G is transformed according to the definition of the editing register so that G is actually the editing register. Since S contains 0022 during DV1, G is a cycle-left register during DV1. G contains only one sign position, which is fed normally by write line 16. In this case, it is fed by write line 14, bit 1 being fed by write line 16.

In Time 2, C(Q) is cycled left into G. The result, with a sign bit superimposed, is sent to Q and the adder in Time 3. This last is twice b(Q) because cycling left is equivalent to doubling except that the sign bit is changed if the argument magnitude is at least 1/2. b(Q) is known to be negative; hence, supplying the sign bit remedies the defect and can return a sum with correct overflow. The main subtraction of the loop is begun in Time 4. The sign test on C(LP) in Time 5 has no effect until Time 10 is executed for the 14th time. Time 7 decides the value of the quotient bit. If the result of the subtraction is negative, it becomes the new dividend in Q; otherwise, C(Q) is unchanged. In Time 9, the quotient bit is injected into B. Since each bit in C(B) except the marker that becomes the sign is initially a 0 (which has a value of 1 in a negative number), a bit is injected if the true quotient bit is 0. Times 9 and 10 cycle C(B) left, and the sign test in Time 10 tests for the marker bit. The quotient with correct sign is stored in A in Time 11 of the 14th and last execution of DV1, the choice of sign being governed by the sign test on C(LP) in Time 5.

Table 3-6, Part 11, shows the method of SU. It is identical with AD except for the negative sign on C(K) in Times 8 and 11 of SU.

Table 3-6, Part 12, shows the actions of the interrupt sequence. The C(B) stored in BRUPT is the instruction that was about to be executed, and the C(Z) is one greater than the location of that instruction.

Table 3-6, Part 13, shows the counter operations. C(CTR) can be fetched in Time 6 because the counters are in Erasable. The sum is not stored until Time 10 because of the time required to compute its parity; actually, it is read out 250 nsec earlier than normal.

COMPARATIVE SPEED CHART

Table 3-3 is a chart comparing running times of the AGC 4 sequences with those of AGC 3. The times quoted are actual rather than nominal, the units being:

	Nominal	Actual
Memory Cycle Time, AGC 4	12 μsec	11.72 μsec
Instruction Time, AGC 3	40 μsec	39.06 μsec

It is assumed that, on the average, address modification of AGC 4 extracode instructions costs no appreciable time.

Table 3-3. Comparison of Running Times of AGC 4 Sequences with AGC 3 Sequences

Operation	Op. Code	Time, MCT	Time, μsec	AGC 3 time, μsec	Speed factor
Transfer Control	TC	1	11.72	39.06	3.33
Count, Compare, and Skip	CCS	2	23.44	78.13	3.33
Index	INDEX	2	23.44	39.06	1.67
Exchange	XCH	2	23.44	39.06	1.67
Clear and Subtract	CS	2	23.44	39.06	1.67
Transfer to Storage	TS	2	23.44	39.06	1.67
Add (no overflow)	AD	2	23.44	39.06	1.67
Add (overflow)	AD	3	35.16	58.59	1.67
Mask	MASK	2	23.44	117.19*	5.00
Multiply (not indexed)	MP	10	117.19	625.00	5.33
Multiply (indexed)	MP	10	117.19	664.06	5.67
Subtract (not indexed, no overflow)	SU	4	46.88	117.19*	2.50
Subtract (not indexed, overflow)	SU	5	58.59	136.72*	2.33
Subtract (indexed, no overflow)	SU	4	46.88	156.25*	3.33
Subtract (indexed, overflow)	SU	5	58.59	175.78*	3.00
Divide (not indexed)	DV	18	210.94	9062.50*	42.96
Divide (indexed)	DV	18	210.94	9101.56*	43.15
Interrupt	-	3	35.16	39.06	1.11
Resume	-	2	23.44	78.13	3.33
Counter Increment	-	1	11.72	19.53	1.67
Counter Decrement	-	1	11.72	19.53	1.67
Counter Shift	-	1	11.72	19.53	1.67

*Not an AGC 3 instruction; time is for equivalent coding.

EXAMPLES OF AGC 4 PROGRAMMING

Examples 1, 2, and 3 are simplified versions of the standard double-precision arithmetic subroutines for AGC 4. They are for illustration only and are not grammatically complete, since some obvious symbols like ONE, ZERO, A, etc., are not explicitly defined. The following is a set of exercises to aid a close study of the subroutines.

EXERCISES ON THE EXAMPLES

1. Multiple precision numbers are often easier to deal with if their component words agree in sign. It is not easy, however, to keep them that way.

   a. For each of three kinds of operation in the examples (since we are dealing with signed numbers, addition and subtraction are considered the same kind of operation throughout these exercises), show that the component words of the double (triple for DMP) precision result can differ in sign when the component words of each operand agree in sign. The signs of the two operands may differ (except for DDV).

   b. Write an AGC 4 program to convert a double precision number, whose component signs may or may not differ, into the equivalent double precision number whose component signs agree.

   c. Do the same for a triple precision number.

2. Assume, as was done in the examples, that the possibility of net overflow in addition is to be ignored because only checked-out programs go into orbit. Consider the following two routines for double precision addition (the address symbols have the same meaning as in the examples):

```
DAD1    XCH     MPAC          DAD 2   XCH     MPAC
        INDEX   ADDRWD                TS      OVCTR
        AD      0                     XCH     MPAC + 1
        XCH     OVCTR                 INDEX   ADDRWD
        XCH     MPAC + 1              AD      1
        INDEX   ADDRWD                XCH     MPAC + 1
        AD      1                     XCH     OVCTR
        XCH     MPAC + 1              INDEX   ADDRWD
        XCH     OVCTR                 AD      0
        TS      MPAC                  XCH     MPAC
```

   a. In what special circumstance does each routine give a grossly wrong result?

   b. What does each routine do with the operands that make the other routine fail?

c. Is there a pair of operands that makes both routines fail?

d. Show that the routine in Example 1 is proof against these failures.

3. There is no facility in AGC 4 for intelligent handling of overflow out of OVCTR.

a. Show that this problem does not arise in the double precision multiply routine in Example 2.

b. Rewrite the routine using the skipping and overflow-indicating features of the TS instruction and without any use of OVCTR.

4. Analysis of double-precision arithmetic is often aided by thinking of double-precision numbers as two-digit fractions in radix-16384 notation (for AGC 4). This is especially true for double-precision division.

a. Derive the approximation used in Example 3 for double-precision division.

b. Derive an expression for the error of that approximation.

c. Show that the routine in Example 3 works when the high-order words of the operands are equal.

d. Show the necessity of the restriction assumed in the introductory remarks to Example 3.

EXAMPLES OF AGC 4 PROGRAMMING EXAMPLE 1

DOUBLE PRECISION ADD AND SUBTRACT SUBROUTINES. THE MULTIPLE PRECISION ACCUMULATOR, MPAC THROUGH MPAC +2, IS THE SOURCE OF ONE OPERAND IN ALL THESE EXAMPLES AND IS THE DESTINATION OF THE RESULT. IN THESE EXAMPLES, ASSUME THAT C(MPAC +2) IS INITIALLY ZERO, AND THAT ADDRWD INITIALLY CONTAINS THE ADDRESS N, WHERE THE OTHER OPERAND IS IN N AND N +1. FOR ADDITION AND SUBTRACTION, C(N, N +1) IS TO BE ADDED TO OR SUBTRACTED FROM C(MPAC, MPAC +1). THE TIME ESTIMATE ASSUMES NO OVERFLOWS.

```
DSU     CAF     EXTENDER    MODIFY C(ADDRWD) SO AS TO CHANGE
        AD      ADDRWD      EACH INDEXED AD TO SU WITHOUT
        TS      ADDRWD      CHANGING THE ADDRESS MODIFICATION.
```

THE SYMBOL ..EXTENDER.. STANDS FOR THE ADDRESS (5777) OF THE LOCATION IN FIXED MEMORY WHICH BY CONVENTION CONTAINS THE OCTAL CONSTANT 47777. THIS CONVENTION FIXES THE DEFINITION OF THE MNEMONIC OP CODE ..EXTEND..

```
DAD     XCH     MPAC +1     C(OVCTR) MAY CHANGE IN THIS SUB-
        INDEX   ADDRWD      ROUTINE, BUT WE DO NOT CARE.
        AD      1           ADD OR SUBTRACT C(N +1).
        TS      MPAC +1     STORE RESULT, SKIP ON OVERFLOW.
        CAF     ZERO        OTHERWISE MAKE INTERWORD CARRY = 0.

        AD      MPAC        PROPAGATE CARRY OF +1, -1, OR 0.
        INDEX   ADDRWD
        AD      0           ADD OR SUBTRACT C(N).
        XCH     MPAC        STORE RESULT THUS TO IGNORE OVERFLOW.
        RETURN              RUNNING TIMES- DAD 20 MCT, DSU 26 MCT.
```

EXAMPLES OF AGC 4 PROGRAMMING EXAMPLE 2

DOUBLE PRECISION MULTIPLY SUBROUTINE. HERE WE STORE IN MPAC THROUGH MPAC +2 THE PRODUCT OF B(MPAC, MPAC +1) AND C(N, N +1) , CARRIED OUT TO TRIPLE PRECISION. WE ABBREVIATE THE OPERANDS AS M0, M1 AND N0, N1. THE NOTATIONS U(MXNY) AND L(MXNY) MEAN UPPER WORD OF THE PRODUCT MXNY AND LOWER WORD, RESPECTIVELY.

```
DMP     CAF     EXTENDER    CHANGE C(ADDRWD) SO THAT THE INDEX
        AD      ADDRWD      OPERATIONS OVERFLOW FOR THE EXTRA-
        TS      ADDRWD      CODE MP.

        XCH     MPAC +1
        TS      OVCTR       M1 TO OVCTR.
        INDEX   ADDRWD
        MP      1           M1N1. ONLY U(M1N1) IS TO BE KEPT.
        XCH     OVCTR       U(M1N1) TO OVCTR, M1 TO A.

        INDEX   ADDRWD
        MP      0           M1N0.
        XCH     OVCTR       U(M1N0) TO OVCTR, U(M1N1) TO A.
        AD      LP          MAYBE INCREMENT U(M1N0) IN OVCTR.
        XCH     MPAC        L(M1N0)+U(M1N1) TO MPAC, M0 TO A.
```

..INCREMENT.., IN THIS SUBROUTINE, MEANS INCREMENT OR DECREMENT ACCORDING TO THE SIGN OF THE OVERFLOW.

```
        TS      MPAC +2     M0 TO MPAC +2.
        INDEX   ADDRWD
        MP      1           M0N1.
        XCH     OVCTR       U(M0N1) TO OVCTR, U(M1N0) TO A.
        XCH     MPAC        U(M1N0) TO MPAC, L(M1N0)+U(M1N1) TO A.
        AD      LP          MAYBE INCREMENT U(M0N1) IN OVCTR.
        XCH     MPAC +2     C(MPAC +2) = L(M0N1+(L(M1N0)+U(M1N1).

        INDEX   ADDRWD
        MP      0           M0N0.
        XCH     OVCTR       U(M0N0) TO OVCTR, U(M0N1) TO A.
        AD      MPAC        MAYBE INCREMENT U(M0N0) IN OVCTR.
        AD      LP          DITTO.
        XCH     MPAC +1     C(MPAC +1) = L(M0N0)+U(M1N0)+U(M0N1).
        XCH     OVCTR
        TS      MPAC        C(MPAC) = U(M0N0).
        RETURN              RUNNING TIME 83 MCT AVERAGE.
```

EXAMPLES OF AGC 4 PROGRAMMING EXAMPLE 3

DOUBLE PRECISION DIVIDE SUBROUTINE. USING THE NOTATION OF THE MULTIPLY SUBROUTINE, ASSUME THAT M0, M1, N0, AND N1 ARE ALL POSITIVE, THAT N0, N1 IS GREATER THAN M0, M1, AND THAT N0 IS GREATER THAN 1/2 (ASSUMING THE BINARY POINT TO LIE BETWEEN THE SIGN AND BIT 14). AN ACTUAL DIVIDE SUBROUTINE MUST MAKE THOSE THINGS TRUE BEFORE DOING THE PART SHOWN HERE. THE QUOTIENT OF (M0, M1)/(N0, N1) IS DELIVERED TO MPAC AND MPAC +1. USING THE NOTATIONS Q(MX/NY) AND R(MX/NY) FOR THE QUOTIENT OF MX/NY AND THE REMAINDER, RESPECTIVELY, DIVISION IS ACHIEVED BY MEANS OF THE APPROXIMATION

$$\frac{M0+SM1}{N0+SN1} = Q\left(\frac{M0}{N0}\right) + \frac{S}{N0}\left(M1 + R\left(\frac{M0}{N0}\right) - Q\left(\frac{M0}{N0}\right) N1\right), \text{ WHERE } S = \frac{1}{16384} = 2^{-14} .$$

```
DDV     CAF     EXTENDER    CHANGE C(ADDRWD) SO THAT THE INDEX
        AD      ADDRWD      OPERATIONS OVERFLOW FOR THE EXTRA-
        TS      ADDRWD      CODES.

        XCH     Q
        XCH     MPAC        RETURN ADDRESS TO MPAC, M0 TO A.
        INDEX   ADDRWD
        DV      0           Q(M0/N0) TO A, -ABS(R(M0/N0)) TO Q.
        TS      OVCTR       Q(M0/N0) TO OVCTR.

        INDEX   ADDRWD
        MP      1           Q(M0/N0)N1.
        AD      Q           -R(M0/N0)+Q(M0/N0)N1.
        COM                 R(M0/N0)-Q(M0/N0)N1.
        AD      MPAC +1     M1+R(M0/N0)-Q(M0/N0)N1.  MAYBE INCREMENT
        OVSK                Q(M0/N0) IN OVCTR.  SKIP ON OVERFLOW.
        TC      +3          BRANCH IF NO OVERFLOW.
```

EXAMPLES OF AGC 4 PROGRAMMING EXAMPLE 3

DOUBLE PRECISION DIVIDE SUBROUTINE (CONTINUED).

```
        INDEX   ADDRWD      REDUCING C(A) BY THE DIVISOR (N0)
        SU      0           GUARANTEES NO OVERFLOW NOW.
+3      TS      MPAC +1     TENTATIVE DIVIDEND TO MPAC +1.
        CCS     A
        AD      ONE         COMPUTE
        TC      +2            ABSOLUTE
        AD      ONE             VALUE.
+2      INDEX   ADDRWD
        SU      0           COMPARE ABSOLUTE VALUE OF TENTATIVE
        CCS     A           DIVIDEND WITH THE DIVISOR, N0.
        AD      ONE         GREATER, SO MUST USE REDUCED DIVIDEND.
        TC      +2          THAT CCS NEVER COMES HERE.

        TC      +4          LESS, SO DIVIDEND IS OK.

+2      XCH     MPAC +1     EQUAL, SO MUST USE ZERO.
        DOUBLE              INCREMENT OR DECREMENT OVCTR.
        CCS     A           RECALL SIGN OF DIVIDEND.
+4      XCH     MPAC +1     POSITIVE.
        TC      +2

        CS      MPAC +1     NEGATIVE.

+2      INDEX   ADDRWD
        DV      0           (M1+R(M0/N0)-Q(M0/N0)N1)/N0
        TS      MPAC +1     IS THE LOWER QUOTIENT.
        XCH     OVCTR
        XCH     MPAC        STORE ADJUSTED UPPER QUOTIENT.
        TC      A           RETURN.  RUNNING TIME 96 MCT AVERAGE.
```

Table 3-4. Control Pulses in AGC 4

PULSE NAME	MNEMONIC FOR:	TIMING (N = NORMAL)	ACTION
CI	Carry In	N	Force carry-in to adder stage 1.
CLG *	Clear G	Full μsec	Reset G.
CTR	Loop Counter	N	1. Increment loop counter. 2. Set stage 2 flip-flop next Time 12 if loop counter goes to six.
GP	Generate Parity		1. Reset the parity bit of the G register, and write the 1-15 output of the parity pyramid into the parity bit of the G register. 2. If c(S) is 0014, reset the parity bit of the OUT 4 register; and write the 1-15 output of the parity pyramid into the parity bit of the OUT 4 register.
KRPT	Knock down Rupt priority	N	Reset the current interrupt priority flip-flop.
NISQ	New instruction to the SQ register	N	Set flip-flop to cause transfer from B to SQ next Time 12.
RA	Read A	N	
RB	Read B	N	
RB14	Read Bit 14	N	Octal 20000 to Write Buses.
RC	Read C	N	
RG	Read G	N	Read G, placing the Sign bit into the SG and US positions of the Write Buses.
RG *	Read G	Full μsec	As above.

Table 3-4. Control Pulses in AGC 4 (Continued)

PULSE NAME	MNEMONIC FOR:	TIMING (N = NORMAL)	ACTION
RLP	Read LP	N	
RP2	Read Parity two	N	1. Reset the parity bit of the G register, and write the contents of P2 into the parity bit of the G register. 2. If c(S) is 0014, reset the parity bit of the OUT 4 register; and write the contents of P2 into the parity bit of the OUT 4 register.
RQ	Read Q	N	
RRPA	Read RUPT Address	N	Read current interrupt address to Write Buses.
RSB	Read Sign Bit	N	Octal 100000 to Write Buses (US unaffected).
RSC	Read Special and Central	N	Read addressed flip-flop register.
RSCT	Read Selected Counter address	N	Read current counter address to Write Buses.
RU	Read sum	N	
RU *	Read sum	Full μsec	
RZ	Read Z	N	
R1	Read one	N	Octal 00001 to Write Buses.
R1C	Read one Complemented	N	Octal 177776 to Write Buses (US = 1).
R2	Read 2	N	Octal 00002 to Write Buses.
R22	Read 22	N	Octal 00022 to Write Buses.
R24	Read 24	N	Octal 00024 to Write Buses.

Table 3-4. Control Pulses in AGC 4 (Continued)

PULSE NAME	MNEMONIC FOR:	TIMING (N = NORMAL)	ACTION
ST1	Stage 1	N	Set stage 1 flip-flop next Time 12.
ST2	Stage 2	N	Set stage 2 flip-flop next Time 12.
TMZ	Test for Minus Zero	N	Set Branch 2 flip-flop if Write Buses are at minus zero.
TOV	Test for Overflow	N	1. Set Branch 2 flip-flop if positive overflow. 2. Set Branch 1 flip-flop if negative overflow.
TP	Test Parity	N	Generate alarm if P-15 output of parity pyramid is "1."
TRSM	Test for Resume	N	Set stage 2 flip-flop next Time 12 if c (S) = 00025.
TSGN	Test Sign	N	Set Branch 1 flip-flop if sign bit is "1."
TSGN 2	Test Sign	N	Set Branch 2 flip-flop if sign bit is "1."
WA	Write A	N	Reset and write A.
WALP	Write A and LP	N	1. Reset A and write A via right-shift write gates. 2. Reset bit 14 of LP, and write bit 1 into bit 14 of LP.
WB	Write B	N	Reset and write B.

Table 3-4. Control Pulses in AGC 4 (Continued)

PULSE NAME	MNEMONIC FOR:	TIMING (N = NORMAL)	ACTION
WG	Write G	N	Reset all of G except bit P, and write via normal or shifting or cycling write gates as dictated by contents of S.
WG *	Write G	Full μsec	Write G (do not reset) via normal write gates, and write into the parity bit of G from the 1-15 output of the parity pyramid.
WLP	Write LP	N	Reset and write LP.
WOVC	Write Overflow Counter	N	Direct overflow to OVCTR priority inputs.
WOVI	Write Overflow RUPT Inhibit	N	Inhibit interrupt after this instruction if overflow indication on Write Buses.
WOVR	Write Overflow	N	1. Direct overflow to priority inputs. 2. Reset current counter priority flip-flop.
WP	Write P	N	Reset and write P.
WP *	Write P	Last 3/4 μsec	Write P (do not reset).
WP2	Write P2	N	Reset and write P2 from 1-15 output of parity pyramid.
WQ	Write Q	N	Reset and write Q.
WS	Write S	N	Reset and write S.

Table 3-4. Control Pulses in AGC 4 (Continued)

PULSE NAME	MNEMONIC FOR:	TIMING (N = NORMAL)	ACTION
WSC	Write Special and Central	N	Reset and write addressed flip-flop register.
WX	Write X	N	Write X (do not reset).
WX *	Write X	Full μsec	As above.
WY	Write Y	N	1. Reset X. 2. Reset and write Y. 3. Reset CI flip-flop.
WY *	Write Y	Full μsec	Write Y (do not reset).
WZ	Write Z	N	Reset and write Z.

Table 3-5. Control Pulse Sequences in AGC 4

Wherever a line of sequence appears more than once, a notation such as 01, 10, 0X, etc. appears beside the line number. These are not control pulses but represent the state of branch flip-flops 1 and 2, mentioned in the definitions in table 3-1 of the control pulses TSGN, TSGN2, TOV, and TMZ. "X" indicates "don't care."

TC

1. RB WY WS CI
2.
3. WG
4. RA WOVI
5.
6.
7. RG RSC WB WP
8. RZ WQ GP TP
9. RB WSC WG
10. RU WZ
11. NISQ

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

CCS0

1. RB WS
2. RZ WY
3. WG
4.
5.
6. RG RSC WB TSGN WP
7. 0X: RC TMZ
7. 1X: RB TMZ
8. 00: GP TP
8. 01: R1 WX GP TP
8. 10: R2 WX GP TP
8. 11: R1 R2 WX GP TP
9. RB WSC WG
10. 00: RC WA
10. 01: WA R1C
10. 10: RB WA
10. 11: WA R1C
11. RU ST1 WZ

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

CCS1

1. RZ WY WS CI
2.
3. WG
4. RU WZ
5. RA WY CI
6.
7. RG RSC WB WP
8. RU WB GP TP
9.
10. RC WA WOVI
11. RG RSC WB NISQ

NDX0

1. RB WS
2.
3. WG
4. RA WOVI
5.
6.
7. RG RSC WB WP
8. GP TP
9. RB WSC WG
10. TRSM
11. ST1

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

NDX1

1. RZ WY WS CI
2.
3. WG
4. RU WZ
5.
6. RB WY
7. RG RSC WB WP
8. RB WX GP TP
9. RB WSC WG
10.
11. RU WB WOVI NISQ

RSM

1. R24 WS
2.
3. WG
4.
5.
6.
7. RG WZ
8.
9.
10.
11. NISQ

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

XCH

1. RB WS
2. RA WP
3. WG
4. WP2
5.
6.
7. RG RSC WB WP
8. GP TP
9. RA WSC WG RP2
10. RB WA WOVI
11. ST2

CS

1. RB WS
2.
3. WG
4.
5.
6.
7. RG RSC WB WP
8. GP TP
9. RB WSC WG
10. RC WA WOVI
11. ST2

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

TS

1. RB WS
2. RA WB TOV WP
3. WG
4. 00:
4. 01: RZ WY CI (overflow)
4. 10: RZ WY CI (underflow)
5. 00:
5. 01: R1 WA
5. 10: WA R1C
6.
7. 00:
7. 01: RU WZ
7. 10: RU WZ
8. GP
9. RB WSC WG
10. RA WOVI
11. ST2

AD

1. RB WS
2. RA WY
3. WG
4.
5.
6.
7. RG RSC WB WP
8. RB WX GP TP
9. RB WSC WG
10.
11. RU WA WOVC ST2 WOVI

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

MASK

1. RB WS
2. RA WB
3. WG
4. RC WY
5.
6.
7. RG RSC WB WP
8. RU RC WA GP TP
9.
10. RA WB
11. RC WA ST2 WOVI

MP0

1. RB WS
2. RA WB TSGN
3. RSC WG
4. 0X: RB WLP
4. 1X: RC WLP
5. RLP WA
6.
7. 0X: RG WY WP
7. 1X: RG WB WP
8. 0X: GP TP
8. 1X: RC WY GP TP
9. RU WB TSGN2
10. X0: RA WLP TSGN
10. X1: RA RB14 WLP TSGN

(continued on next page)

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

MP0 (Continued)

11. 00: ST1 WALP
11. 01: R1 ST1 WALP R1C
11. 10: RU ST1 WALP
11. 11: RU ST1 WALP

MP1

1. RA WY
2. RLP WA TSGN
3. 0X:
3. 1X: RB WX
4. RA WLP
5. RLP TSGN
6. RU WALP
7. RA WY
8. 0X:
8. 1X: RB WX
9. RLP WA
10. RA WLP CTR
11. RU ST1 WALP

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

MP3

1. RZ WY WS CI
2. RLP TSGN
3. WG
4. RU WZ
5. RA WY
6. 0X:
6. 1X: RB WX
7. RG RSC WB WP
8. RLP WA GP TP
9. RB WSC WG
10. RA WLP
11. RU WALP NISQ

DV0

1. RB WS
2. RA WB TSGN
3. RSC WG
4. 0X: RC WA
4. 1X:
5. 0X: R1 WLP
5. 1X: R2 WLP
6. RA WQ
7. RG WB TSGN WP
8. RB WA GP TP
9. 0X: RLP R2 WB
9. 1X:
10. 0X: RB WLP
10. 1X: RC WA
11. R1 ST1 WB

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

DV1

1. R22 WS
2. RQ WG
3. RG WQ WY RSB
4. RA WX
5. RLP TSGN2
6.
7. RU TSGN
8. 0X:
8. 1X: RU WQ
9. 0X: RB RSB WG
9. 1X: RB WG
10. RG WB TSGN
11. 0X: ST1
11. 10: RC WA ST2
11. 11: RB WA ST2

SU

1. RB WS
2. RA WY
3. WG
4.
5.
6.
7. RG RSC WB WP
8. RC WX GP TP
9. RB WSC WG
10.
11. RU WA WOVC ST2 WOVI

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

RUPT1

1. R24 WY WS CI
2.
3. WG
4.
5.
6.
7.
8.
9. RZ WG
10. RU WZ
11. ST1 ST2

RUPT3

1. RZ WS
2. RRPA WZ
3. RZ KRPT WG
4.
5.
6.
7.
8.
9. RB WSC WG
10.
11. ST2

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

STD2

1. RZ WY WS CI
2.
3. WG
4. RU WZ
5.
6.
7. RG RSC WB WP
8. GP TP
9. RB WSC WG
10.
11. NISQ

PINC

1. WS RSCT
2.
3. WG
4. R1 WY
5.
6. RG* WX* WP
7. TP
8. WP
9. RU* CLG* WP*
10. RU* WG* WOVR
11.

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

MINC

1. WS RSCT
2.
3. WG
4. WY R1C
5.
6. RG* WX* WP
7. TP
8. WP
9. RU* CLG* WP*
10. RU* WG* WOVR
11.

SHINC

1. WS RSCT
2.
3. WG
4. WY
5.
6. RG* WY* WX* WP
7. TP
8. WP
9. RU* CLG* WP*
10. RU* WG* WOVR
11.

Table 3-5. Control Pulse Sequences in AGC 4 (Continued)

OINC

1. WS
2.
3. WG
4.
5.
6.
7. RG RSC
8.
9.
10.
11.

LINC

1. WS
2.
3. WG
4.
5.
6.
7. WG WP WSC
8. GP
9.
10.
11.

This page intentionally left blank.

Table 3-6. Flow Charts of Pulse Sequences in AGC 4
Part 1

L TC K — 1 MCT — Code 0

MCT 1	1.	Select address K; c(U) ← K+1
(TC)	4.	INHINT1 on overflow in c(A)
	7.	c(B) ← c(K)
	8.	c(Q) ← b(Z) = TC L+1
	10.	c(Z) ← c(U) = K+1

Part 2

L CCS K — 2 MCT — Code 1

MCT 1	1.	Select address K (K may NOT be in fixed memory)
(CCS0)	2.	c(U) ← b(Z) = L+1
	6.	c(B) ← c(K); test sign of c(B)

\+ → 7. Test c(B) vs. -0

≠ → 8. 10. c(A) ← -c(B)

= → 8. c(U) ← c(U)+1; 10. c(A) ← -1

− → 7. Test c(B) vs. -0

≠ → 8. c(U) ← c(U)+2; 10. c(A) ← c(B)

= → 8. c(U) ← c(U)+3; 10. c(A) ← -1

11. c(Z) ← c(U) = M = L+n, where n = 1, 2, 3, or 4

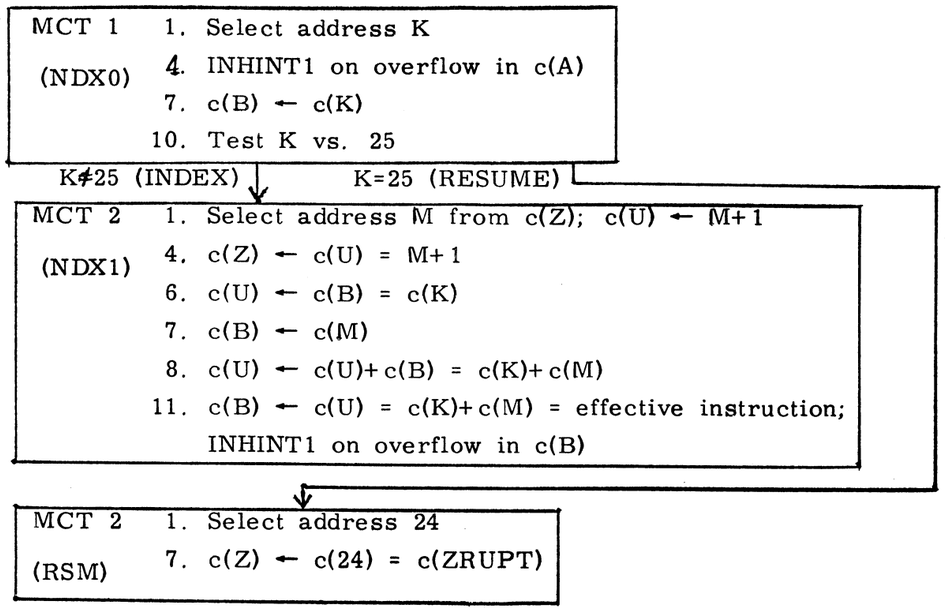
MCT 2	1.	Select address M; c(U) ← M+1
(CCS1)	4	c(Z) ← c(U) = M+1
	5.	c(U) ← c(A)+1
	8.	c(B) ← c(U)
	10.	c(A) ← -c(B) = DABS(c(K))*; INHINT1 on overflow in c(A)
	11.	c(B) ← c(M)

* The DABS (diminished absolute value) function of an integer α is defined as $|\alpha|-1$ if $|\alpha| > 1$, or

+0 if $|\alpha| = 0$ or 1.

Table 3-6. Flow Charts of Pulse Sequences in AGC 4 (Continued)
Part 3

L INDEX K (or RESUME) 2 MCT **Code 2**



Part 4

L XCH K 2 MCT **Code 3**

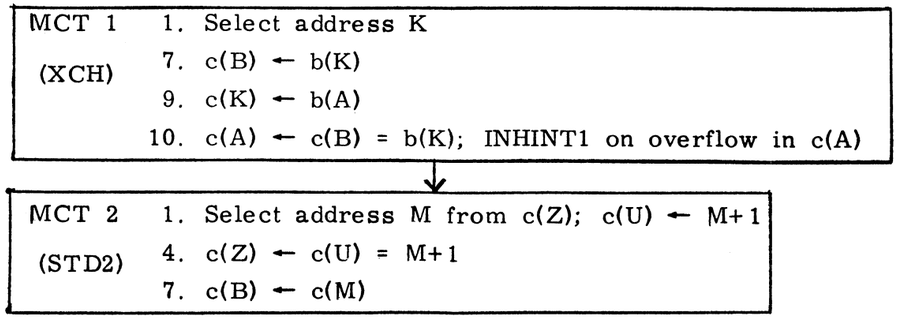
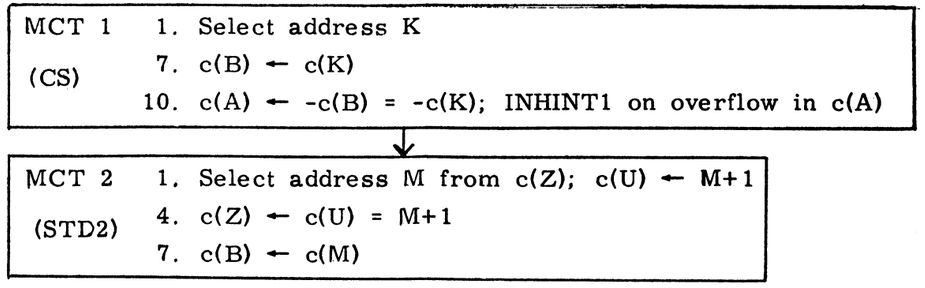


Table 3-6. Flow Charts of Pulse Sequences in AGC 4 (Continued)
Part 5

L CS K — 2 MCT — Code 4



Part 6

L TS K — 2 MCT — Code 5

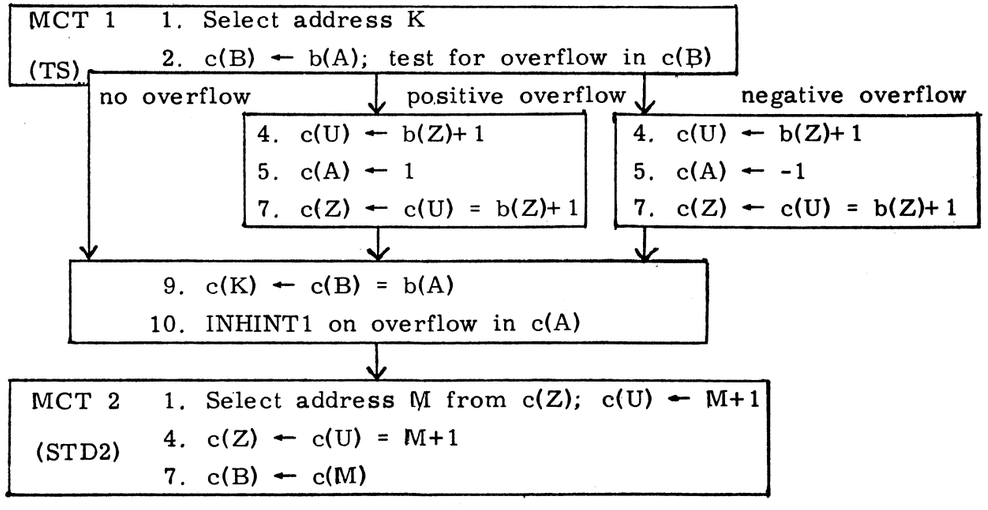
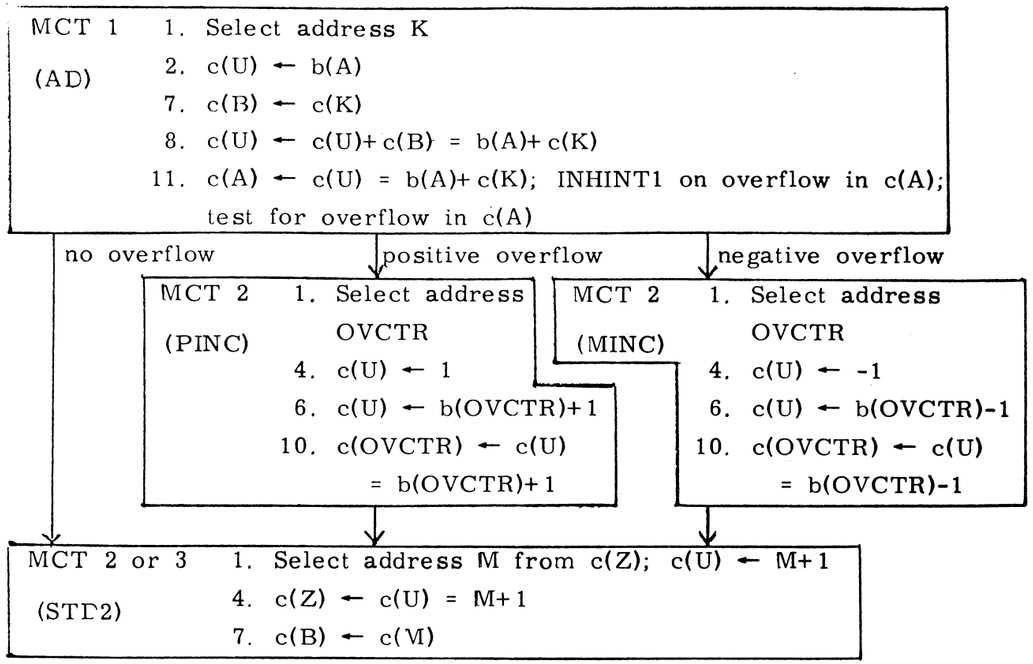


Table 3-6. Flow Charts of Pulse Sequences in AGC 4 (Continued)

Part 7

L AD K — 2 or 3 MCT — Code 6



Part 8

L MASK K — 2 MCT — Code 7

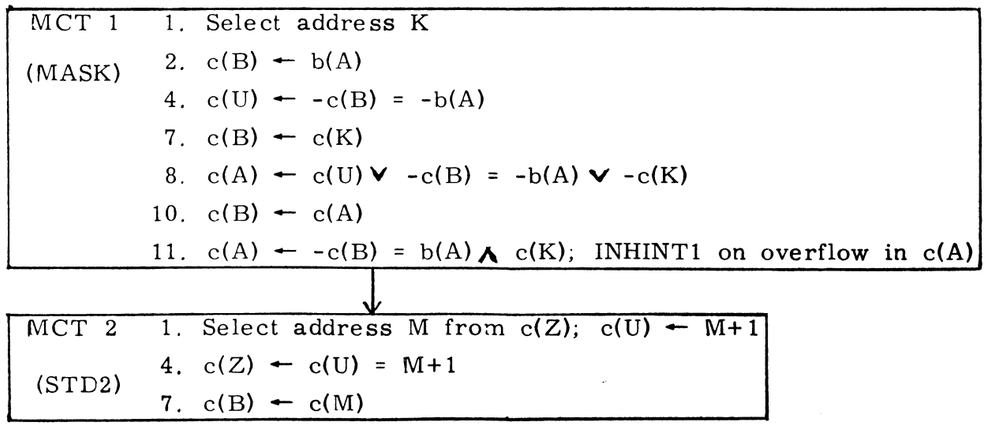
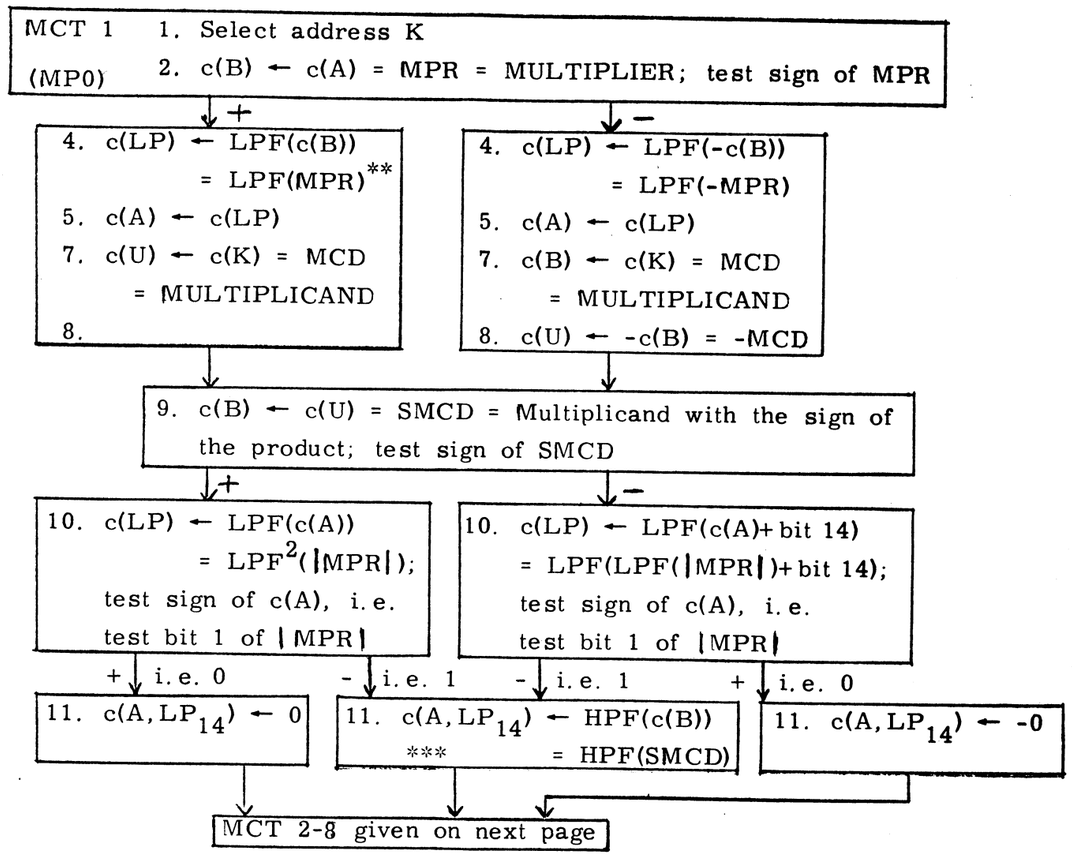


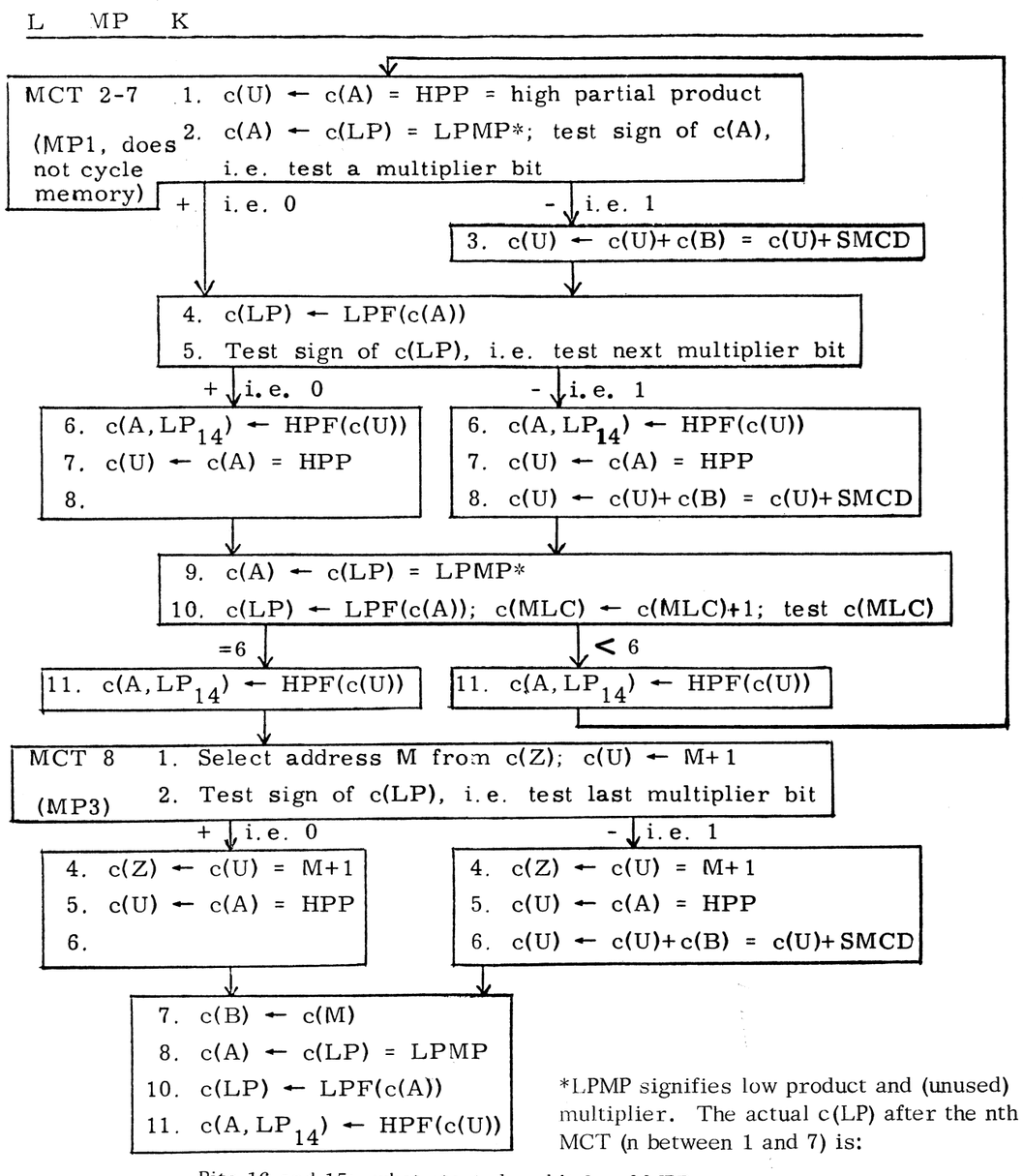
Table 3-6. Flow Charts of Pulse Sequences in AGC 4 (Continued)
Part 9a

L MP K * 8 MCT Conventional code 4



*The multiply loop counter (MLC) is reset to zero between multiplications.
**The notation LPF stands for the editing function performed by writing into LP, that is, cycle right 1, clearing bit 14.
***The notation HPF (high product function) stands for the editing function performed by the control pulse WALP, that is, shift right 1 into A and into bit 14 of LP. See Table 3-2 for details.

Table 3-6. Flow Charts of Pulse Sequences in AGC 4 (Continued)
Part 9b



*LPMP signifies low product and (unused) multiplier. The actual c(LP) after the nth MCT (n between 1 and 7) is:

Bits 16 and 15: what started as bit 2n of MPR,
Bits 14 through (16-2n): 2n-1 low-order bits of product,
Bit (15-2n): sign of product, as supplied in line 10 of MCT 1,
Bits (14-2n) through 1: 14-2n high-order unused multiplier bits.

Table 3-6. Flow Charts of Pulse Sequences in AGC 4 (Continued)
Part 10

L DV K — 16 MCT — Conventional code 5

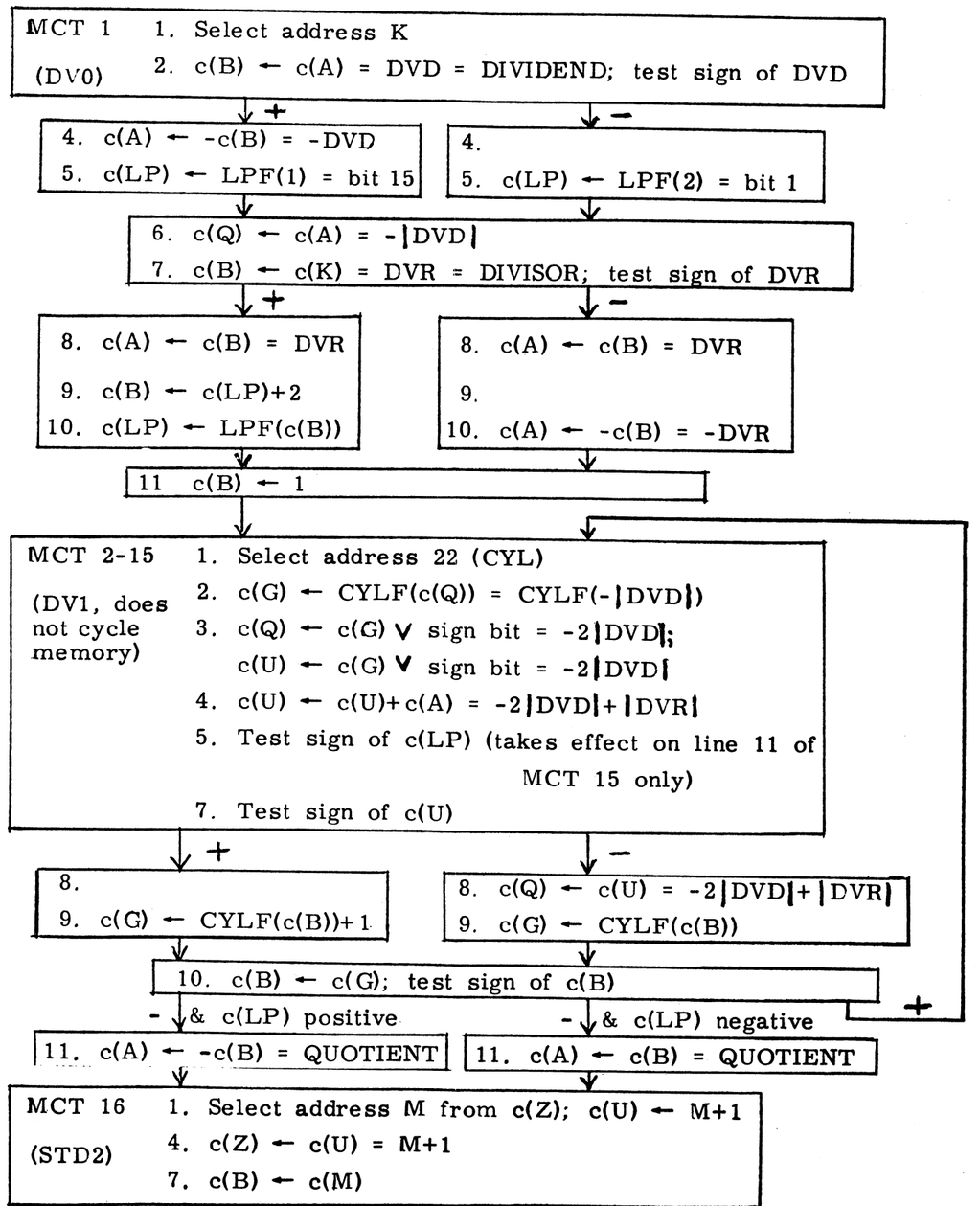


Table 3-6. Flow Charts of Pulse Sequences in AGC 4 (Continued)
Part 11

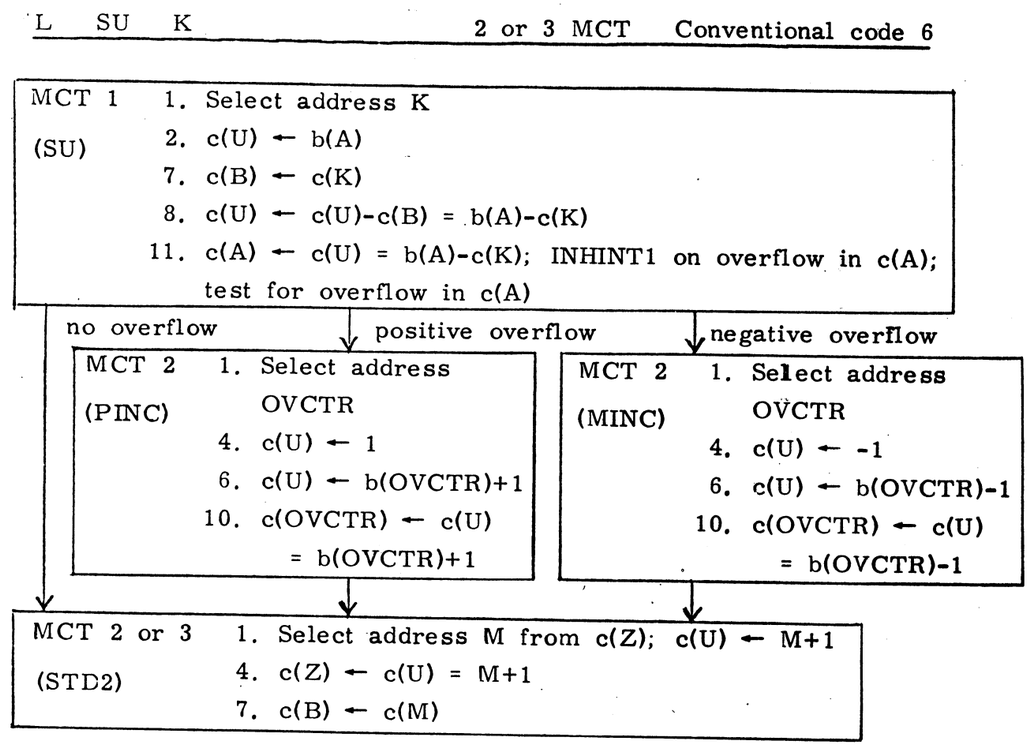


Table 3-6. Flow Charts of Pulse Sequences in AGC 4 (Continued)
Part 12

INTERRUPT 3 MCT

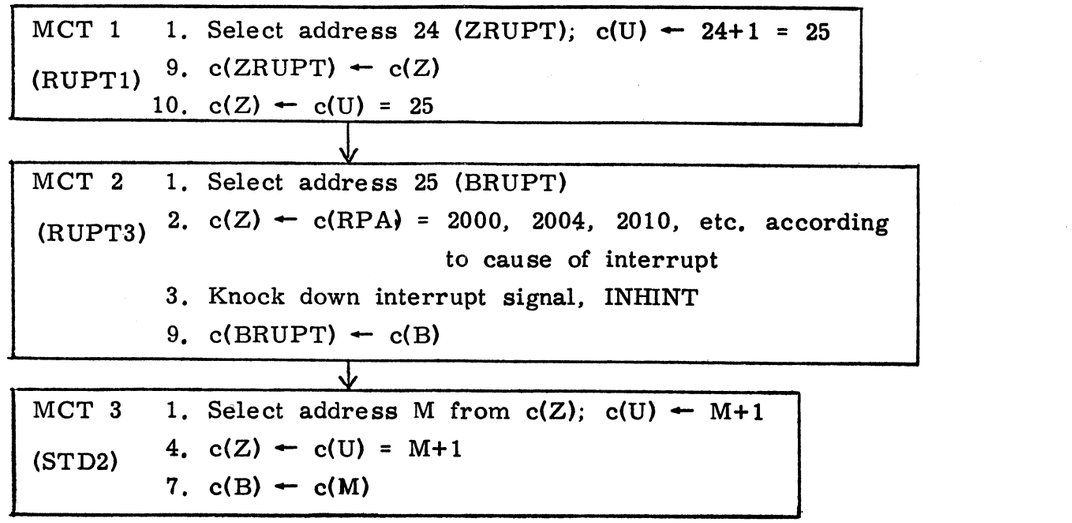


Table 3-6. Flow Charts of Pulse Sequences in AGC 4 (Continued)

Part 13a

COUNTER INCREMENT 1 MCT

MCT 1 (PINC)	1. Select address of counter from SCT 4. c(U) ← 1 6. c(U) ← c(U)+b(CTR) 10. c(CTR) ← c(U) = b(CTR)+1; for some counters, request an interrupt if overflow in c(CTR)

Part 13b

COUNTER DECREMENT 1 MCT

MCT 1 (MINC)	1. Select address of counter from SCT 4. c(U) ← -1 6. c(U) ← c(U)+b(OTR) 10. c(CTR) ← c(U) = b(CTR)-1; for some counters, request an interrupt if overflow in c(U)

Part 13c

COUNTER SHIFT 1 MCT

MCT 1 (SHINC)	1. Select address of counter from SCT 4. c(U) ← 0 6. c(U) ← b(CTR)+b(CTR) 10. c(CTR) ← c(U) = 2b(CTR); request an interrupt if overflow in c(U)

MISCELLANEOUS

There are a few special sequences that operate under control of the system test equipment and do not concern the programmer.

Chapter 4

MEMORY

GENERAL CHARACTERISTICS

The bulk of the AGC 4 memory is divided into two classes: a wired-in or Fixed Memory of 12, 288 words for storage of programs, constants, and tables, and an Erasable Memory of 1008 words for storage of intermediate results and auxiliary program information.

It is necessary to have the two kinds of memories because accidents of input-output can cause some of the contents of Erasable to be altered without the Erasable Memory being at fault in any way. The requirements for Fixed Memory are: that it be permanent, compact, and insensitive to accidents of input-output. In addition, Fixed Memory must not be damaged by failures in other parts of the computer. Other requirements are low power consumption, high reliability, adequate margins, and environmental tolerances.

The density requirement for Erasable Memory is less severe, since fewer words are used. High reliability, low power and adequate tolerances, however, are just as important in the Erasable Memory as in the Fixed Memory.

The Fixed Memory consists primarily of three Magnetic Core Ropes, each of which is composed of cores, diodes, resistors, and wire. The information is stored geometrically and is not electrically alterable. The density of Rope storage compares well with all other current nondestructive memories with the exception of some magnetic drums. Including all of the selection, driving, and sensing equipment, and plug connectors for replacement of 1024 word segments, the density of the AGC 4 Fixed Memory is about 800 bits per cubic inch.

The power consumption is moderate to low. The energy required per access is about 200 microwatt seconds. At full computing rate, this requires about 17 watts of power. The memory consists of 3072 cores, 25 current switching circuits, 768 diodes, 768 resistors, wire, and 16 sense amplifiers, and is potentially extremely reliable. Tolerances to voltage margins and component value changes are excellent, since any or all drive currents may vary by ±25% without impairing the function of the memory. The environmental limits are those of the component parts, which are adequate for the currently projected environment.

The Erasable Memory uses a coincident-current ferrite-core array with a core density of about 4000 cores per cubic inch. Including selection, driving, and sensing equipment and plug connectors, the bit density of the memory is about 500 per cubic inch. Power consumption is almost exactly the same as in the Fixed Memory, with a possibility of its being 40% less with the production of a new ferrite material. The memory consists of 16,384 ferrite cores, 30 current switching circuits, 128 diodes, plus sense amplifiers. It has a somewhat lower potential reliability than the Fixed Memory. Its tolerances are not as high as the Fixed Memory, but, because the current limiting circuits have nearly the same temperature coefficient as the coercive force of the ferrite material, the Erasable Memory can be expected to operate over an adequate range of temperature. The critical circuits are the four current limiting circuits and the sixteen sense amplifiers.

FIXED MEMORY ORGANIZATION

The principle of the Core Rope Memory is to cause one of many cores to switch and, later, to switch back to its original state. While the core is switching, it induces a voltage in all sense lines which pass through it and none in those lines which do not.

In AGC 4, the 12,288 words of Fixed Memory are divided into three separate physical groups called "ropes." Each rope stores 4096 words in 1024 cores. Words are 16 bits long; there are four words per core; hence, there must be 4 times 16 or 64 sense lines in each rope.

For reasons having to do with signal to noise ratios, it is not desired to have sense lines be long enough to thread 1024 cores. Consequently, each rope actually contains 128 different sense lines, each long enough to thread 512 cores. Since 16 sense lines make up a word, it is natural to group the sense lines that way and to say that a rope, with its 128 sense lines, has eight groups of 16 sense lines, called "strands." Since each strand is 512 cores long, there are 512 words per strand. Eight strands make one rope or 4096 words.

It is a peculiarity of the AGC that the 512 words in a strand do not all have consecutive addresses. This is because ropes are made up of four pluggable modules called "quarter ropes," which contain 256 cores each. Two quarter ropes together make up a 512-core circuit containing four strands. Machine addresses are made consecutive within a quarter rope; thus, it is convenient to speak of "substrands," which are half strands, each 256 cores long. A quarter rope contains four substrands; addressing is consecutive within a substrand. Figure 4-1 illustrates this hierarchy of rope suborganization.

Each rope contains a set of ten pairs of selection wires called "inhibit lines", which, together with a set line and a reset line, select one core in the rope and cause it to switch. The current drivers for these lines are common for the three ropes, but no more than one of the three can operate at one time. The selection between ropes is effected by three high-current gates, labeled RPG1, RPG2, RPG3, as shown in figure 4-2.

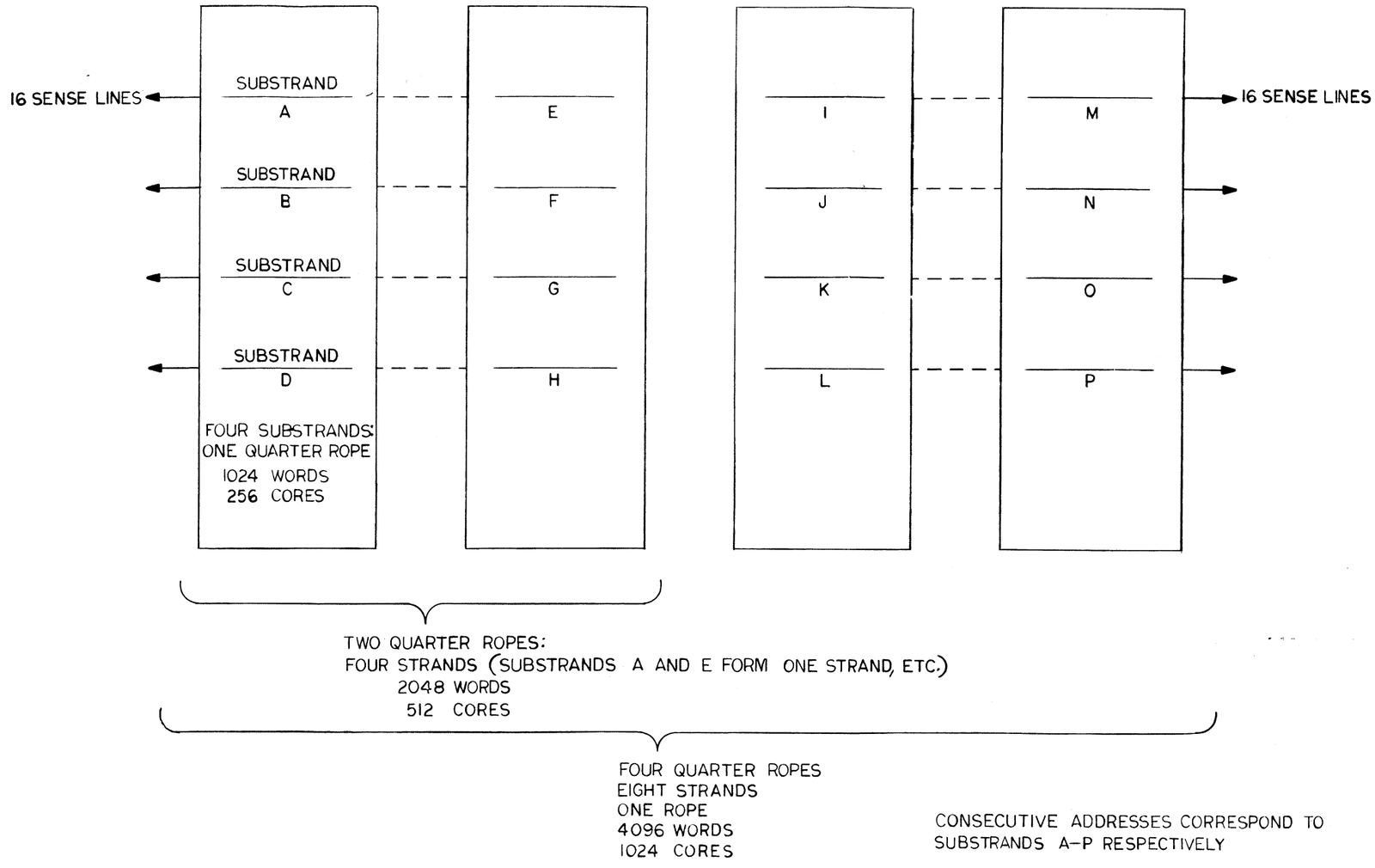


Figure 4-1. Rope Suborganization Diagram.

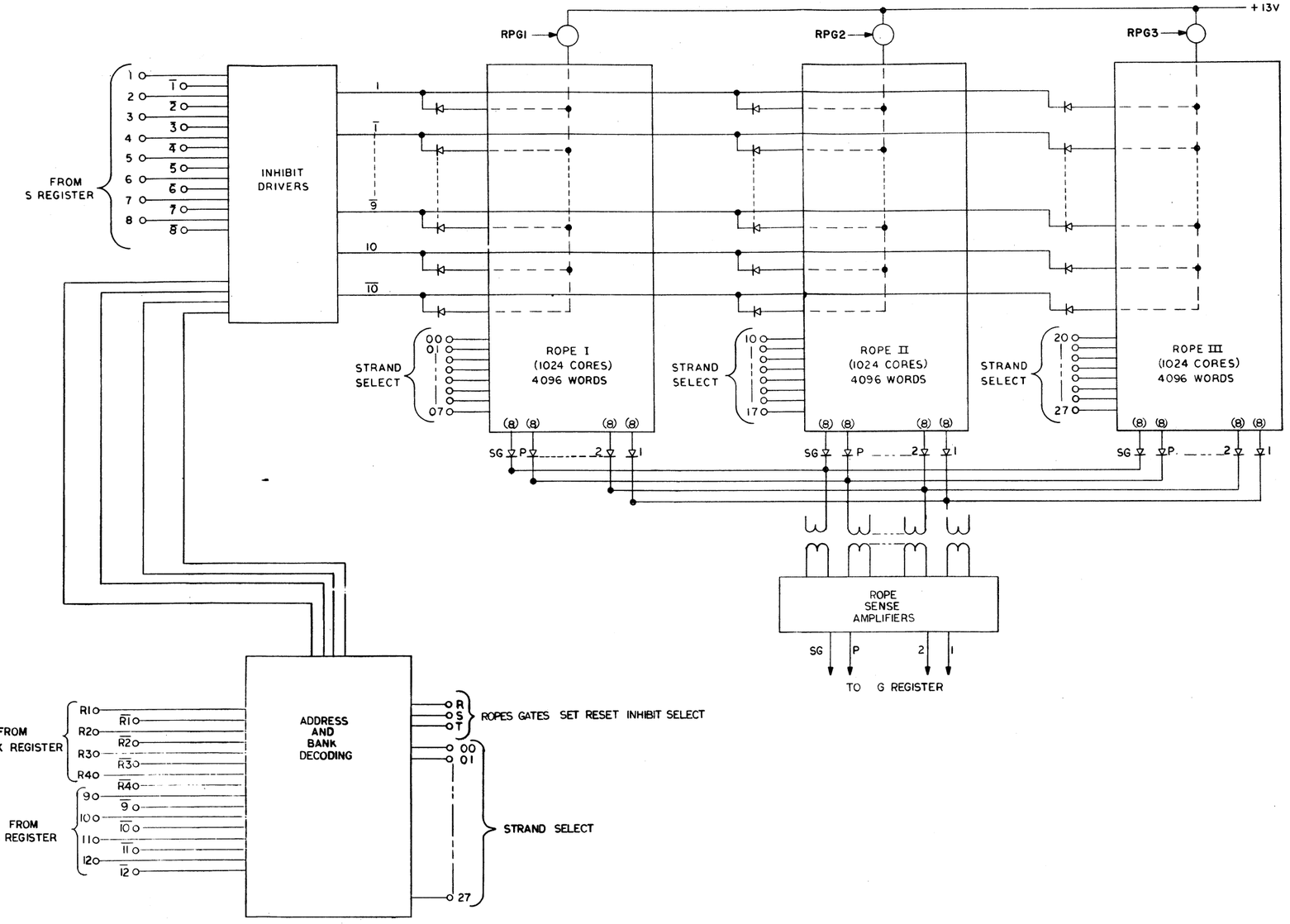


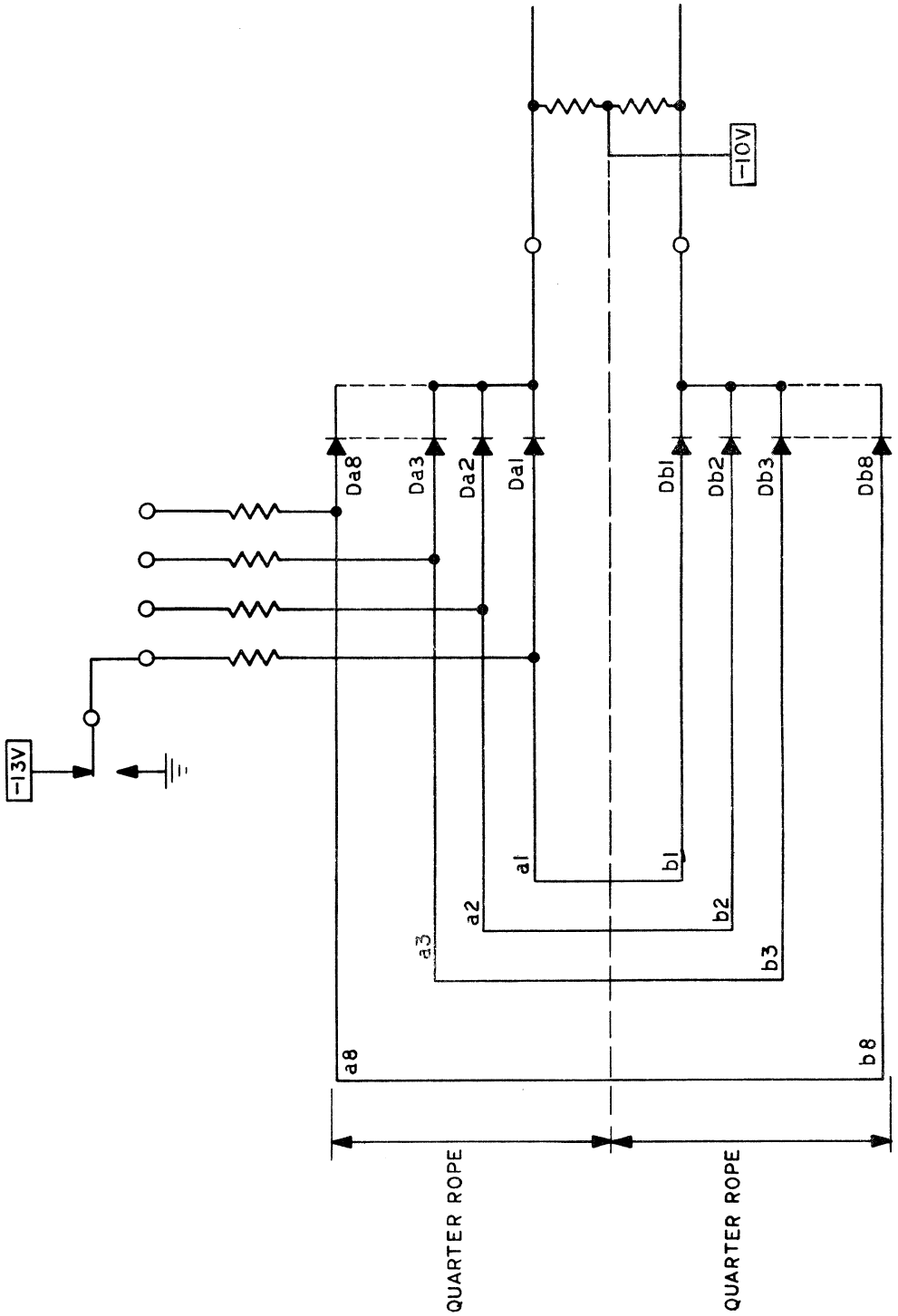
Figure 4-2. Rope Selection System.

Since there are eight separate sets of sense lines, or strands, in each rope, there are in all twenty-four strands among which to select. The selection mechanism is a pair of diodes in series with each sense line that block the sense signal unless forward biased. See figure 4-3. Twenty-four, separate, strand-selection signals, labeled SD0 through SD23, are generated in the decoding logic for operating these diode gates.

The addresses within a substrand are consecutive and are directly controlled by bits 1-8 of the S register via inhibit lines 1-8 and $\overline{1}$-$\overline{8}$. The substrands within a quarter rope are consecutive and are selected by bits 9 and 10 of the S register via the sense-line selection circuits. Quarter ropes are consecutive within a rope and are selected by logical functions of the Bank register bits R1-R4, and of bits 11 and 12 of the S register via inhibit lines 9, 10, $\overline{9}$, and $\overline{10}$. The same function that controls inhibit lines 10 and $\overline{10}$ also selects between the two groups of strands threading the first two and last two quarter ropes. The three, high-current, rope gates (RPG1, RPG2, RPG3) are also selected by logical functions of the Bank register bits and of bits 11 and 12 of the S register.

The purpose of the Bank register, as mentioned in chapter 1, is to modify the selection of a word when bits 11 and 12 of S are both "1". The required transformations are given in table 4-1. The address-decoding network selects the proper quarter rope and strand set, and the bits of lower order than 11 operate in the normal way to select substrand and word. All of the selection switching functions are given in table 4-2.

The Fixed-Memory timing cycle is shown in figure 4-4. The earliest time at which the information can safely be assumed to be in the G register is Time 7.



A. SELECTED SENSE LINES HAVE BOTH DIODES FORWARD BIASED.
B. UNSELECTED LINES HAVE BOTH DIODES BACK BIASED.
C. THERE SHOULD BE ONLY ONE SELECTED SENSE LINE AT ANY ONE TIME.

FIG 3
ROPE SENSE
LINE SELECTION

Table 4-1. AGC 4, Bank Substitution

CONTENTS OF BANK REGISTER				LOCATION OF OCTAL ADDRESSES 6000-7777		INTERPRETIVE OCTAL ADDRESSES
R1	R2	R3	R4	ROPE	QUARTER	
0	0	0	0	1	4	6000-7777
0	0	0	1	1	4	
0	0	1	0	1	4	
0	0	1	1	1	4	
0	1	0	0	1	1	10000-11777
0	1	0	1	2	2	12000-13777
0	1	1	0	2	3	14000-15777
0	1	1	1	2	4	16000-17777
1	0	0	0	2	1	20000-21777
1	0	0	1	3	2	22000-23777
1	0	1	0	3	3	24000-25777
1	0	1	1	3	4	26000-27777
1	1	0	0	3	1	30000-31777
1	1	0	1	-		NOT USED
1	1	1	0	-		
1	1	1	1	-		

Table 4-2. AGC 4, Rope Switching Functions

DEFINITIONS

Input Variables:

S register bits: S1 through S12;

Bank bits: R1, R2, R3, R4.

Output Functions:

RPGk: Rope gates (one of three; k = 1, 2, 3)

Fi: Core Selection (one of 3072, in three groups of 1024 each)

SDj: Strand Selection (one of 24)

Alarm: If the wrong Bank bit combination is used.

Intermediate Functions:

$$IL9 = S11\ \overline{S12} + S11\ S12\ \left[\overline{R1}\ \overline{R2} + R4(\overline{R1}\ R2 + R1\ \overline{R2})\right]$$

$$IL10 = \overline{S11}\ S12 + S11\ S12\ \left[\overline{R1}\ \overline{R2} + R3(\overline{R1}\ R2 + R1\ \overline{R2})\right]$$

Functions:

$$RPG1 = S11\ \overline{S12} + \overline{S11}\ S12 + S11\ S12\ (\overline{R1}\ \overline{R2} + \overline{R1}\ R2\ \overline{R3}\ \overline{R4})$$

$$RPG2 = S11\ S12\ \left[\overline{R1}\ R2\ (R3 + R4) + R1\ \overline{R2}\ \overline{R3}\ \overline{R4}\right]$$

$$RPG3 = S11\ S12\ \left[R1\ \overline{R2}(R3 + R4) + R1\ R2\ \overline{R3}\ \overline{R4}\right]$$

$$Alarm = S11\ S12\ R1\ R2\ (R3 + R4)$$

$$Fi = fi(S1, S2, S3, S4, S5, S6, S7, S8, IL9, IL10)RPGk.$$

The function fi is the AND of all the named variables or their inverses. The variable RPGk selects one of three groups of cores, while fi selects one of 1024 cores for all three groups.

Each SDj is the AND of (IL10) (S10) (S9) or their inverses, and RPGk, as follows:

Table 4-2. AGC 4, Rope Switching Functions (Continued)

		IL10	S10	S9	RPGk
SD	0	0	0	0	RPG1
	1	0	0	1	RPG1
	2	0	1	0	RPG1
	3	0	1	1	RPG1
	4	1	0	0	RPG1
	5	1	0	1	RPG1
	6	1	1	0	RPG1
	7	1	1	1	RPG1
	8	0	0	0	RPG2
	9	0	0	1	RPG2
	10	0	1	0	RPG2
	11	0	1	1	RPG2
	12	1	0	0	RPG2
	13	1	0	1	RPG2
	14	1	1	0	RPG2
	15	1	1	1	RPG2
	16	0	0	0	RPG3
	17	0	0	1	RPG3
	18	0	1	0	RPG3
	19	0	1	1	RPG3
	20	1	0	0	RPG3
	21	1	0	1	RPG3
	22	1	1	0	RPG3
	23	1	1	1	RPG3

e.g., SD0 = $(\overline{IL10})\ (\overline{S10})\ \overline{(S9)}\ (RPG1)$

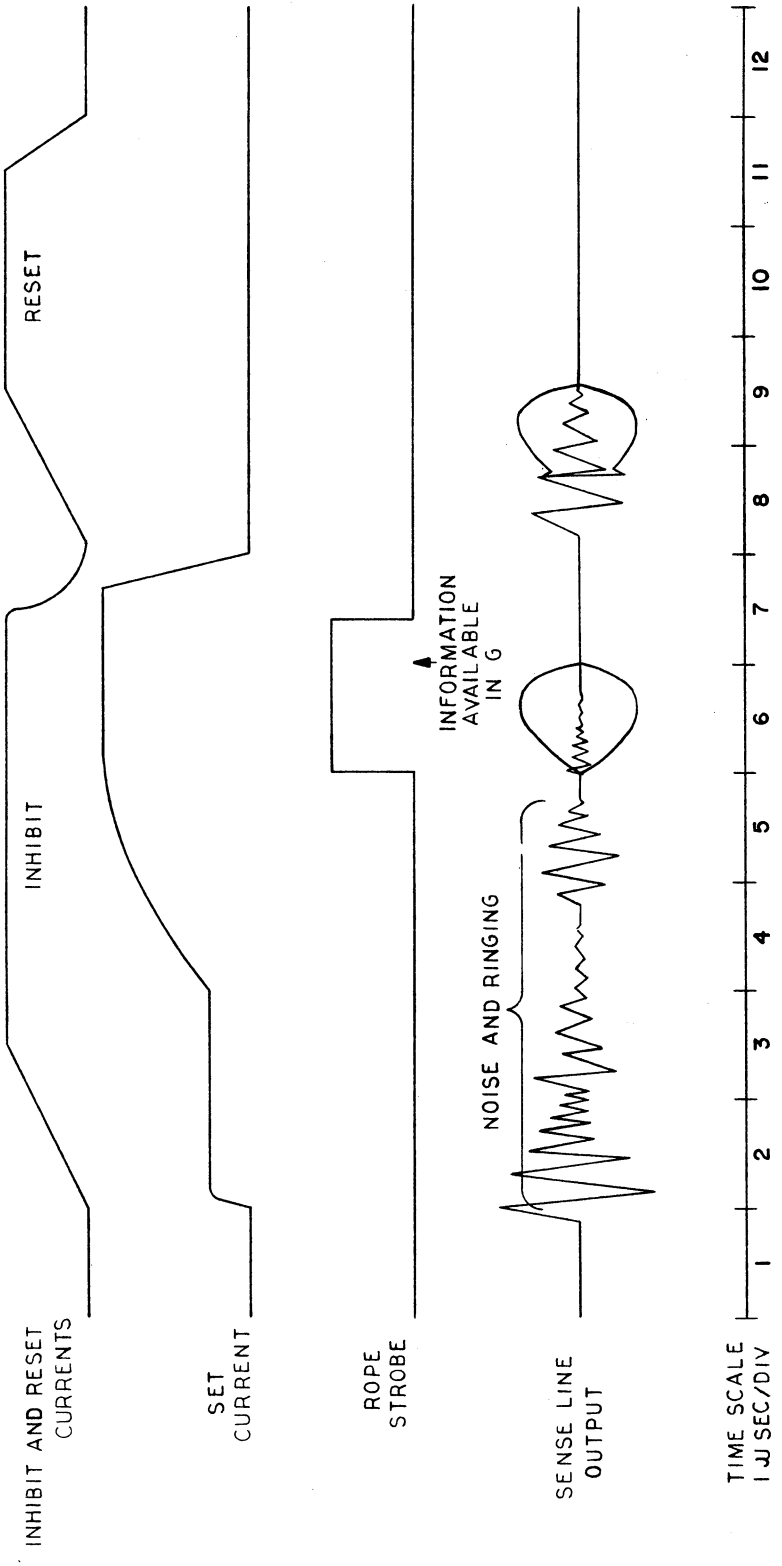


Figure 4-4. Fixed Memory Timing.

ERASABLE MEMORY ORGANIZATION

The 1008 words of Erasable Memory are contained in a coincident-current core array of 1024-word capacity with associated circuitry for selection, operation, and sensing. The selection of one out of 1024 addresses is effected by making two selections of one drive line out of thirty-two. Standard nomenclature for coincident-current memories is that the two sets of drive lines are called the X and Y sets, respectively. The third set of drive lines, which governs the writing of "1's" and "0's," is called the Z set.

The Erasable Memory system is illustrated in figure 4-6. The X and Y selections are each made by a current-steering network which is itself organized on the principle of coincident selection. Eight X-Top and four X-Bottom Current Switches steer current through one of the thirty-two X lines. The X-Top selection is governed by the three lowest order address bits in the S register. The next two bits govern X-Bottom. The next three govern Y-Top, and the next two Y-Bottom. Thus, ten bits select among 1024 words, as required.

The current signals which operate the Erasable Memory are suppressed by the control logic when the address in S is a Fixed Memory address, or a Central or In-Out register address (Octal 17 or lower).

The timing of the Erasable Memory cycle is shown in figure 4-5. It is similar to conventional coincident-current memory cycles but with ample time between read and write current pulses to allow full recovery of the magnetic-core current-switching circuits. Erasable Memory information can be assumed to be available in the G register at Time Pulse 6. This fact is used for the CCS and counter incrementing instructions.

Chapter 4 REFERENCES

MIT/IL Report R-276 — Design Principles for a General Control Computer by R. Alonso and J. H. Laning, Jr.

MIT/IL Report E-1158 — Erasable Store Mod 3C by D. Shansky

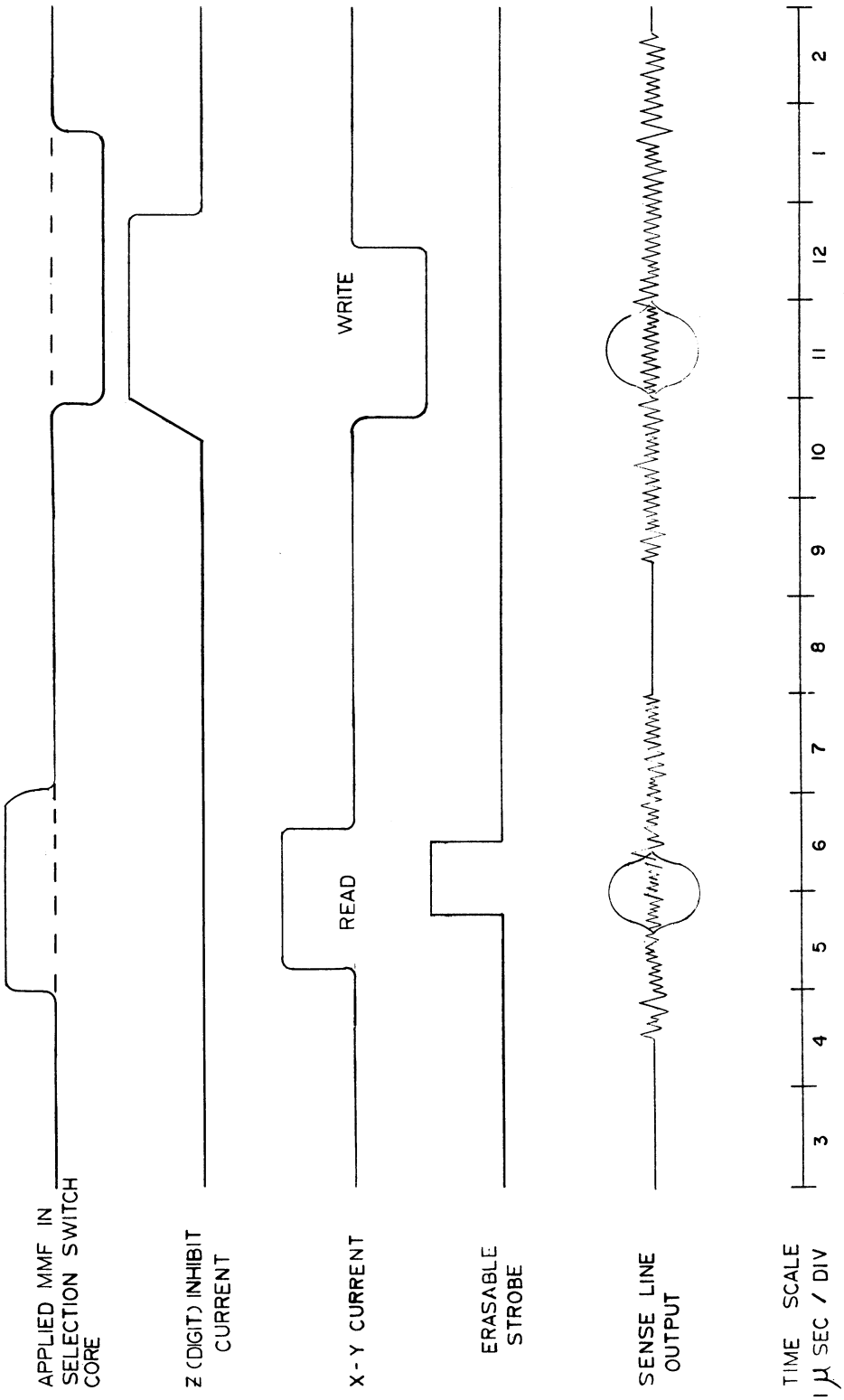
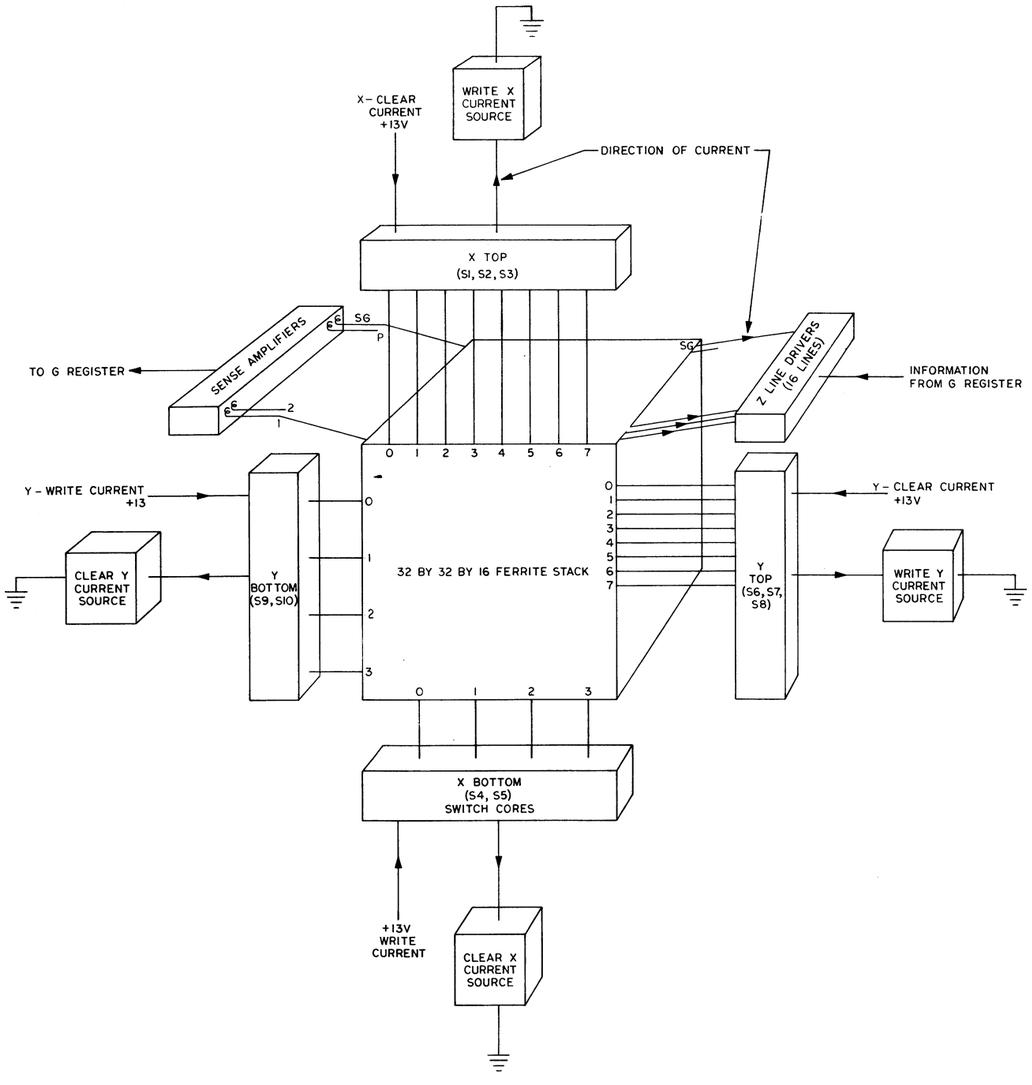
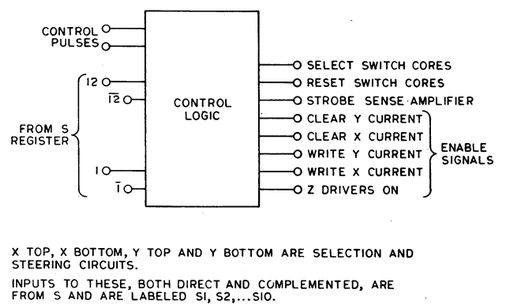


Figure 4-5. Erasable Memory Timing.





X TOP, X BOTTOM, Y TOP AND Y BOTTOM ARE SELECTION AND STEERING CIRCUITS.

INPUTS TO THESE, BOTH DIRECT AND COMPLEMENTED, ARE FROM S AND ARE LABELED S1, S2,...S10.

Figure 4-6. Erasable Memory System.

Chapter 5

IN-OUT

GENERAL

The relationship between AGC 4 and the system to be controlled is shown in figure 5-1. The breakdown into subsystems is functional rather than physical. For the purposes of this chapter, both the AGC Power Supply and the Keyboard and Display (DSKY) sections are considered as part of the external system, rather than as part of the computer. The numbers on each path refer to the number of signals in that path. A complete list of signals can be found in table 5-1 at the end of this chapter.

TYPES OF INPUTS AND OUTPUTS

There are three broad categories of input signals: Discrete signals, Incremental signals, and Power connections.

Discrete Signals are inputs to bits of the various IN x (x = 0, 1, 2, 3) registers (INBITS) under program control. Some of these signals also cause an interrupt, which results in the transfer of control to a program which then examines the contents of the appropriate IN x register. These signals may be either pulse trains or contacts on switches or relays.

Incremental Signals are inputs to a Priority System which controls the servicing of counter registers in the Erasable Memory. The input signals cause the contents of counters to be incremented by a low order plus or minus "1" or to be shifted left by one place. Each incoming signal is associated uniquely with a specific counter and causes a specific action, such as plus increment (PINC). For example, the PIPA X accelerometer sends two signals into the AGC, called PIPA X + and PIPA X -. These signals cause, respectively, a PINC and a MINC (minus increment) sequence to happen to the contents of register 0044. See list of AGC counter assignments in table 5-1 at the end of this chapter.

Power Connections are not signals, strictly speaking, but can be treated as such with some gain in the ease of classification of inputs and outputs. For this reason, they are listed in the various Input and Output signal lists.

The output signals are divided into four categories: Controlled Pulse Rates, Uncontrolled Pulse Rates, Levels (to DSKY only), and Contacts.

Controlled Pulse Rates are pulse rates which may be turned on and off under program control. These pulses are transformer coupled.

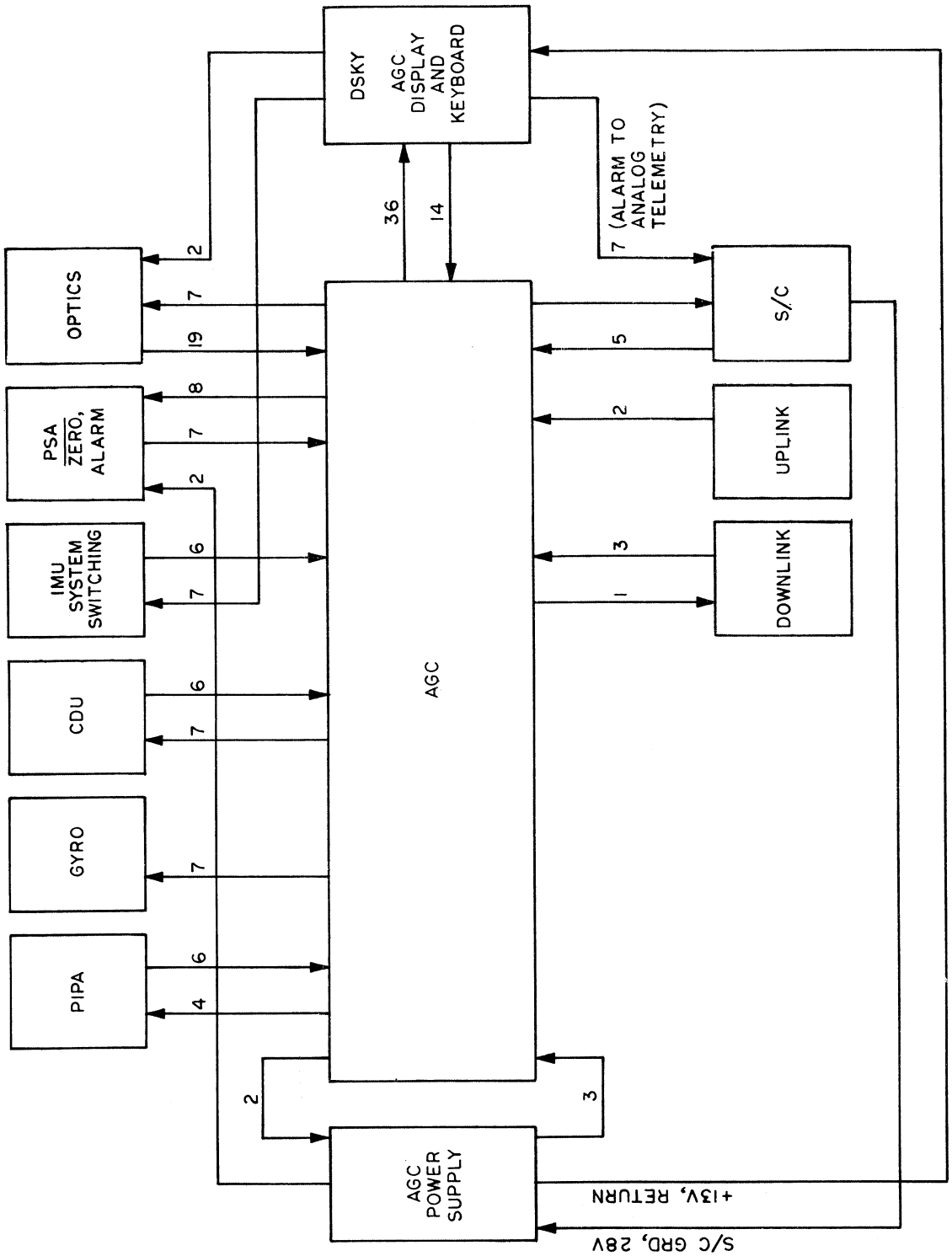


Figure 5-1. In-Out Functional Breakdown.

Uncontrolled Pulse Rates refer to pulse rates which are not under program control, such as the clock signals to the PSA subsystem; they have an asterisk (*) before their name in the AGC output list. (See table 5-1F.) These pulses are also transformer coupled.

Levels refer to DC coupled signals; they exist only in connection with the AGC's own DSKY.

Contacts refer to relay contacts, where the relays are controlled by the AGC. These contacts preserve DC isolation between the AGC and the rest of the system.

Taken together, the AGC and DSKY preserve DC isolation for both inputs and outputs. The power supply is a DC to DC converter which also preserves DC isolation.

METHOD FOR GENERATING OUTPUT RATES

Previously, the AGC output rates were true rates in the sense that various disjoint pulse rates were mixed under control of certain OUTBITS, and the output rate was the sum of the fundamentals. This method, known as the Rate Multiplier Method, made it somewhat cumbersome to send an exact number of pulses to a device. The new scheme sends a burst of pulses to the intended destination, which, while it lasts, is at a fixed rate. To generate a rate-like output, it is necessary to repeat these bursts at fixed intervals. Typically, bursts lasting up to 0.08 seconds can be repeated every 0.24 seconds. For example, the three CDU's may be impulsed sequentially with the above schedule.

The new "rate" method saves about 25 OUTBITS over the number needed by the Rate Multiplier Method; it also simplifies programs for positioning rather than for rate-driving various devices.

The disadvantage is the loss of smoothness of the outgoing rates; instead of having a smooth pulse rate, there is now a burst of K pulses repeated at regular intervals. Another disadvantage is that maximum rate to a device varies inversely with the number of devices which time-share that pulse source.

The new scheme works as follows: OUTBITS 8 through 15 of register OUT 2 (= W2) are reserved for generating Rate I, and OUTBITS 1 through 5 of OUT 2 are used for generating Rate II.

The OUTBIT assignments for Rate I are as follows:

W2,	Bit 15	- Rate I out.
	14	+ Rate I out.
	13	1600 pps to CDU Z. Reset is at 1600 pps.

12	1600 pps to CDU Y.
11	1600 pps to CDU X.
10	1600 pps to GYRO Z. Reset is at 1600 pps.
9	1600 pps to GYRO Y.
8	1600 pps to GYRO X.

To send a positive rate to CDU X, for example, a "1" is required in both OUTBITS 14 and 11. The reason for requiring two OUTBITS to be set is to minimize the risk of accidentally sending unintended pulses because of program malfunctions.

Prior to sending pulses to CDU X, a number 1-K is placed in counter OUTCR I. K is such that OUTCR I will overflow after K pulses have been sent. When OUTCR I overflows, all the OUTBITS associated with Rate I are knocked down to "0," thus preventing any further pulses from being sent out.

The three CDU's may be controlled in sequence by using the Display routine, which is entered every 80 msec. This routine can be made so as to write into OUTCR I the number of pulses to be sent, and to select the device. This process results in sending bursts of pulses to all three CDU's twelve times each second, or four times each second for each CDU. (It is assumed that no gyros are driven.)

Rate II is controlled by OUTBITS 1-5 of OUT 2. Bit assignments are:

W2, Bit 5	- Rate II out.
4	+ Rate II out.
3	Thrust. Reset is at 1600 pps.
2	Optics Y (1600 pps). Reset is at 1600 pps.
1	Optics X (1600 pps).

As with Rate I, the OUTBITS which control Rate II are reset to "0" when OUTCR II overflows.

Rates I and II may be used simultaneously and independently.

The overflows of OUTCR I and OUTCR II do not cause interrupts.

CONVERSION OF PULSES INTO ANALOG SIGNALS

The method for conversion of numbers of pulses into analog signals is as follows: A SET signal (the output rate) sets an external flip-flop F. A RST (for reset) line resets F. The time relation between any possible SET and RST is accurately controlled by the

AGC Scaler. The output of F can then be used to charge a condenser with accurate quanta of charge; the voltage across the condenser can be filtered; and the output is then a voltage proportional to the rate of occurrence of SET pulses. In the AGE system, the DC signals are chopped and converted to 800-cps signals. Figure 5-2 shows the timing of SET and RST. Figure 5-3 shows a typical D-A system. The output voltage V_{out} is an 800-cps signal. The SET pulses occur at an average rate of S pulses per second, and K is a scaling constant.

UPLINK

Information may be entered serially into the AGC by way of the UPLINK counter (octal address 0041). This counter has two input lines: a pulse on line "UPLINK SHIFT" results in shifting the contents of the counter left by one position; a pulse on line "UPLINK +" results in a similar shift, and then in placing a "1" in the least significant digit of the word in the counter. The input pulses need not be synchronized with the AGC clock.

A word is entered by sending a string of 16 pulses, the first of which is a "1." The sixteenth pulse sent results in shifting the first "1" out of the register, i.e., the register overflows. The overflow generates an interrupt (UPRUPT) which, in turn, causes a program to pick the word out of the UPLINK register and reset it to all "0's."

The UPRUPT interrupt can be such that every word entered into UPLINK is treated as if it were a word coming from the Keyboard. In this way, the Ground Support Equipment (GSE) can be used to operate the AGC by means of a keyboard identical to the one on board the craft, which has an additional parallel-to-serial conversion device. The serial nature of the information into UPLINK reduces the number of connections from the ground to the AGC to two.

DOWNLINK

The present telemetry plans call for a single PCM channel with a bit frequency of 64 KC. The following are definitions of various terms:

GROUP: 1 second worth of pulses (= 64×10^3 bits).

FRAME: 20 msec worth of pulses (= 1280 bits).
1 GROUP = 50 FRAMES.

WORD: 125 μsec worth of pulses (= 8 bits).
1 FRAME = 160 WORDS.
(This kind of WORD is different from an AGC word).

BIT: 1 pulse.
1 WORD = 8 BITS.

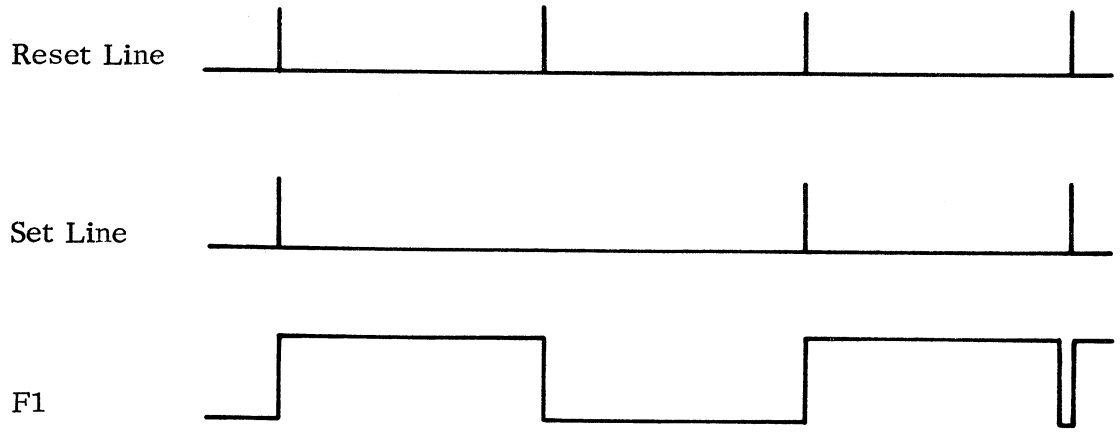


Figure 5-2. Timing for D-A Converter.

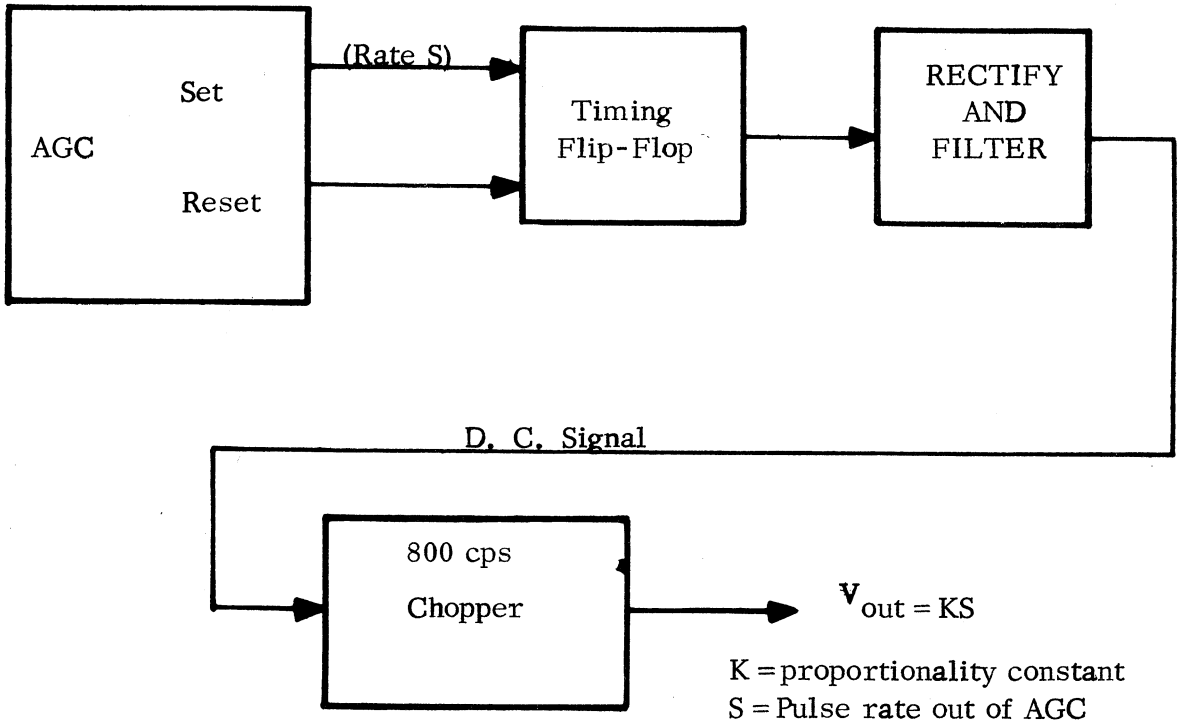


Figure 5-3. Typical D-A Converter System.

The interface between the Telemetry Programmer and the AGC is illustrated in figure 5-4, which shows three signal lines coming from the Telemetry Programmer to the AGC and one line from the AGC to the Transmitter.

The Programmer sends a START pulse to the AGC every other Frame. The AGC then feeds pulses into the Transmitter input in synchronization with the BITSYNC pulses. This activity continues until an ENDPULSE comes from the Programmer, at which time the AGC stops sending pulses to the Transmitter. The time relationships are shown in figure 5-5.

The Transmission period allotted to the AGC under this system is five consecutive Telemetry Words per alternate Frame. In AGC terminology, each five words constitute a Transmission and corresponds to a single sixteen-bit AGC word sent twice (for error detection and correction), and an eight-bit identifying code. This identifying code tells if the Transmitted AGC word is the first or second of a pair under control of OUTBIT 9 of OUT 1. This bit is set to "0" for identifying the first word of a pair, and to "1" for identifying the second word.

Figure 5-6 shows the overall system in terms of AND and OR gates. The AGC word about to be transmitted is placed in OUT 4. The five-stage counter is assumed to be in state 00000. When the START pulse appears, flip-flop F1 is turned on, which in turn enables AND gate G2. The output of the scanning network is then transmitted through G4 into G5. If the START pulse and BITSYNC are coincident, then G5 will be enabled at this time; and a pulse (or no pulse) will be sent to the transmitter. The five-stage counter is advanced when BITSYNC goes to "0." At the following BITSYNC pulse, the scanning network connects to the transmitter the next output bit to be sent.

This process continues until the five-step counter overflows, which is after thirty-two BITSYNC pulses have occurred. When the overflow pulse occurs, F1 is turned off; and F2 is turned on, which causes G2 to be disabled. The output of the scanning network is no longer connected to the transmitter. The BITSYNC pulses will now be routed to the transmitter if the WORD ORDER bit is "1." The BITSYNC pulses are then transmitted until ENDPULSE turns F2 off. If the WORD ORDER bit was "0," then no further pulses are sent to the transmitter.

The ENDPULSE is also used as a signal to lead a new word into OUT 4 by means of the DSRUPT interrupt option. This interrupt option is also used for periodic updating of the displays. The time between an ENDPULSE and the next START is almost 40 msec, which is ample for loading the appropriate word in OUT 4. See figure 5-5.

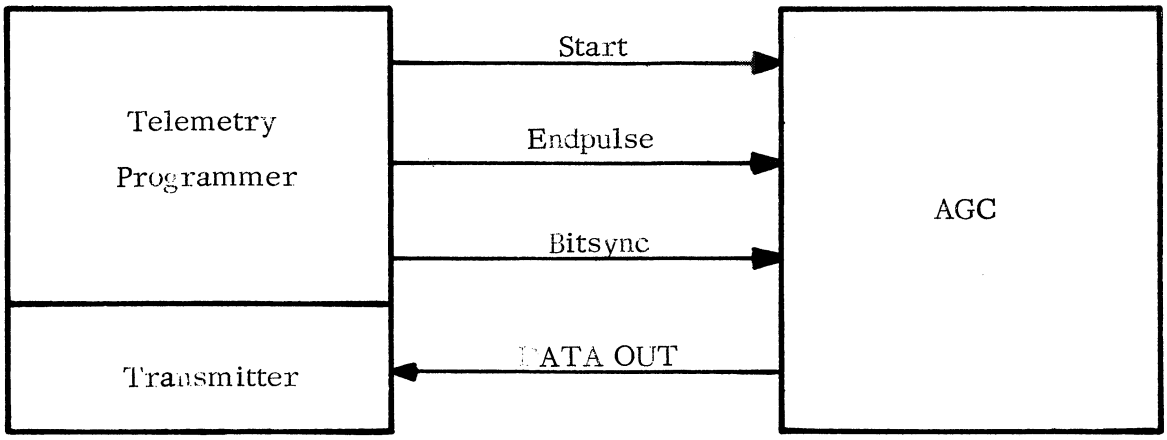


Figure 5-4. Telemetry Interface.

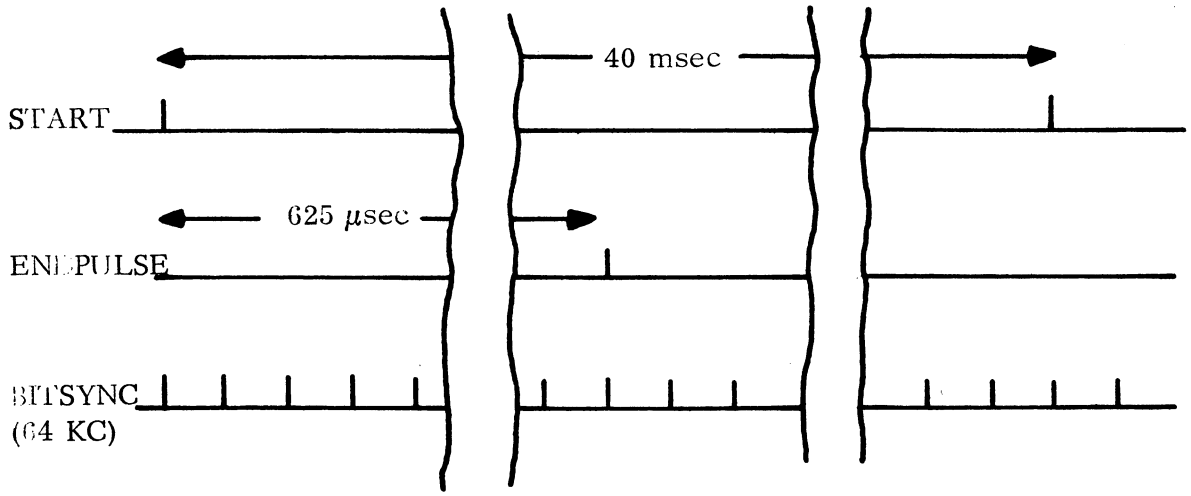


Figure 5-5. Telemetry Timing.

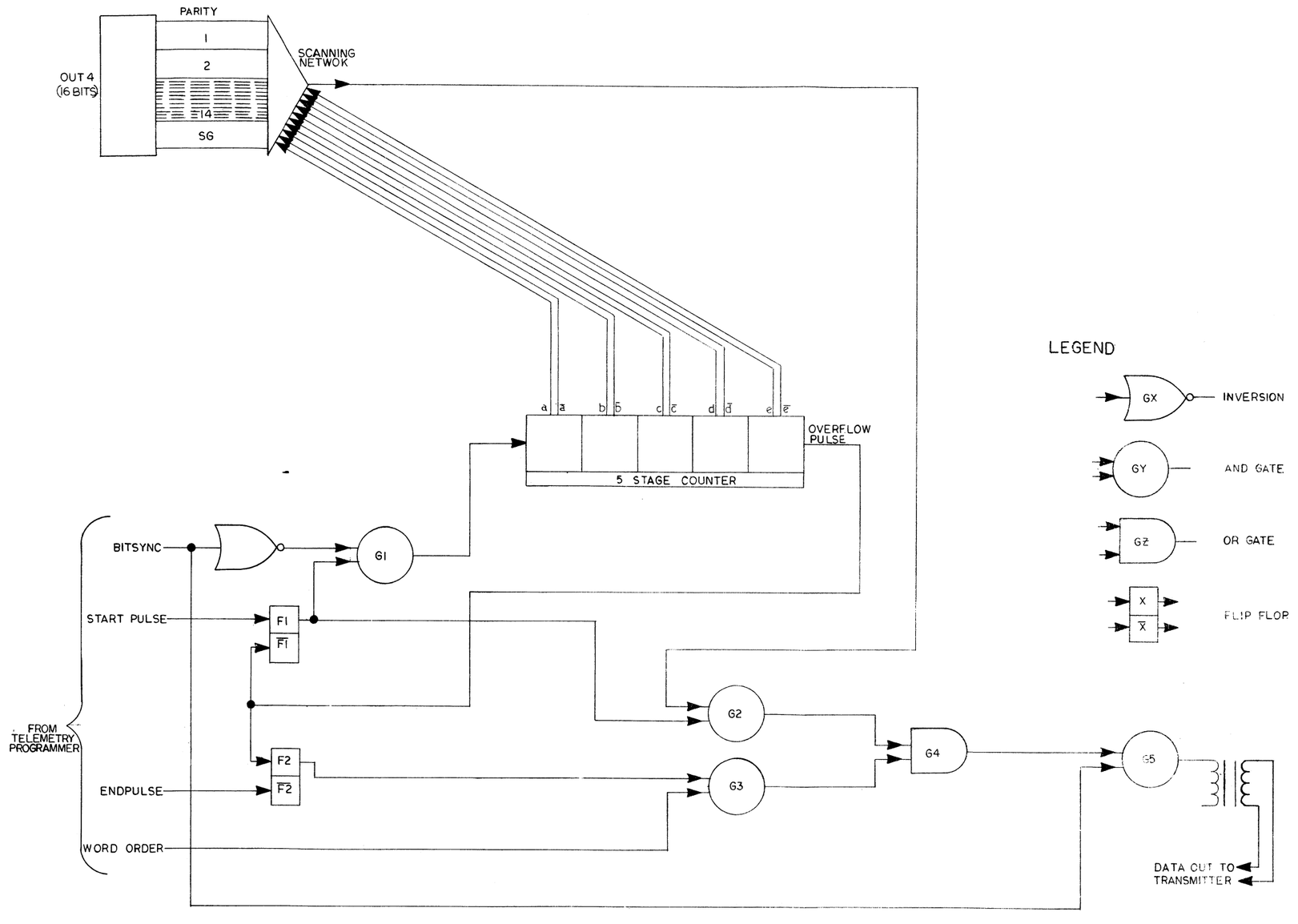


Figure 5-6. Telemetry System.

The encoding of DOWNLINK information consists of twice sending a word and its correct parity. This code is inefficient in the sense of information theory but extremely economical to implement in the context of the AGC 4 organization; it provides double error detection and single error correction. It is a property of this code that every string of sixteen bits must have correct parity (there are sixteen such strings in a message of thirty two bits. If a single bit is wrong, there will be at least one string of sixteen bits whose parity is correct, and at least one string whose parity is incorrect.

If two bits are wrong, then there will be either one or two 16-bit sequences with incorrect parity.

As a practical matter, error detection and correction can be done as follows: the 32-bit message is divided into two halves of 16 bits each. The following cases exist:

Case I: The words agree bit by bit, and both words have correct parity. The message is then assumed to have been transmitted correctly.

Case II: The words disagree in a single bit, and one of them has incorrect parity. The 16-bit word with correct parity is then taken as the correct message.

Case III: The words either disagree in two or more places, or both words fail parity, or both.
A multiple error is assumed. Neither word is the correct one, and the message is rejected.

DISPLAY AND KEYBOARD (DSKY)

A tentative Keyboard and Display arrangement is shown in figure 5-7. The keyboard has sixteen keys, labeled as shown in this figure. These keys connect to six lines which go to the first six bits of register IN 0. See INBIT assignments in table 5-1 at end of this chapter. The first five lines connect to the switches so as to provide a separate code for each key; the sixth line is an activity line. All the keys energize the activity line which, in turn, causes an interrupt (KEYRUPT).

The KEYRUPT interrupt transfers control to a program which examines the contents of IN 0, after allowing about 40 msec for contact-bounce to die down. The character is then usually displayed and stored for further processing. The Keyboard and Display programs will be presented in detail in future reports.

In addition to the keys from the keyboard, there is a single key labeled MARK, which is located in the optics panel. This key also causes a KEYRUPT and ultimately results in the simultaneous recording of the platform and sextant angles.

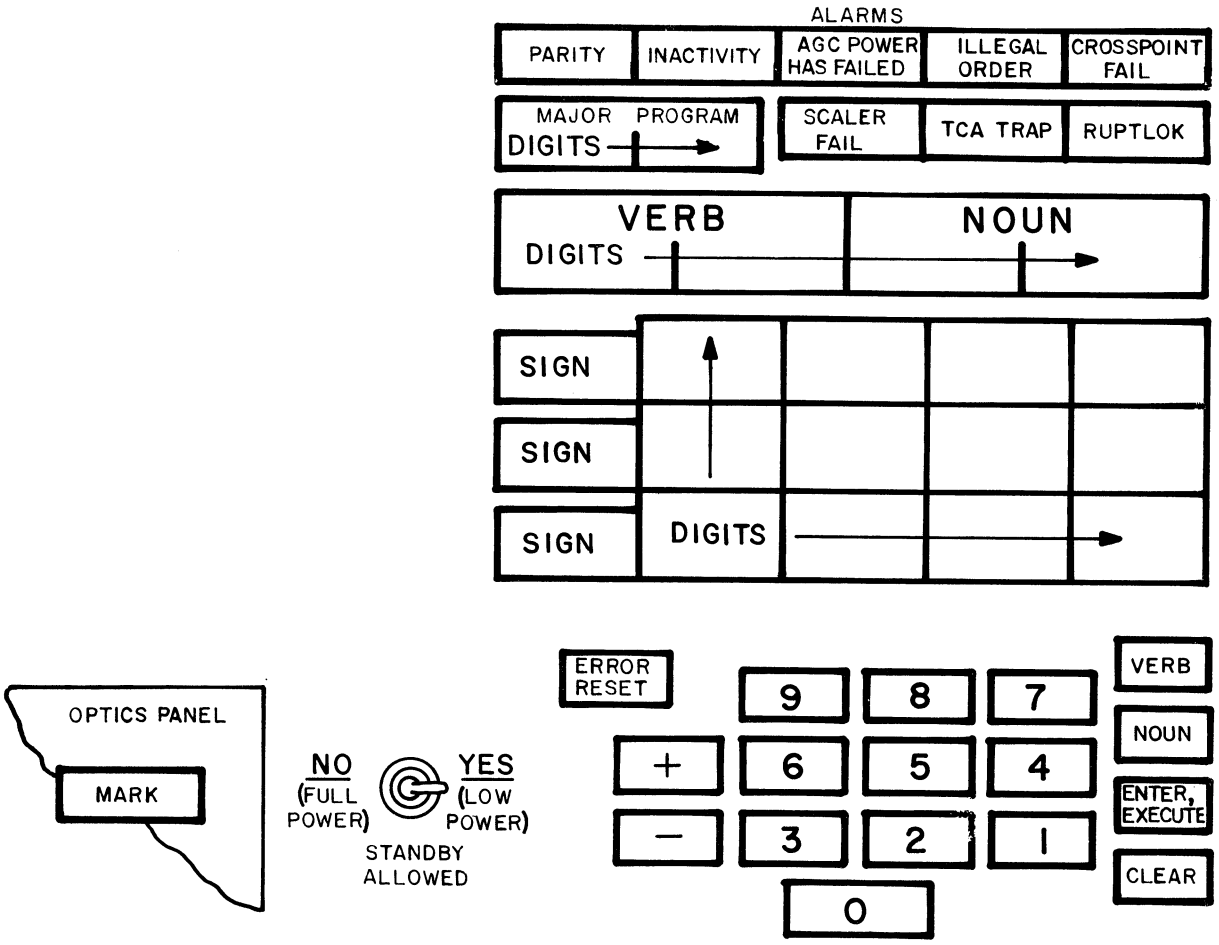


FIG. 7
AGC KEYBOARD & DISPLAY

Figure 5-7. Keyboard and Display.

The Display system is presently based on a magnetic-latching, relay-crosspoint system shown is figures 5-8 and 5-9. The tweleve lowest order bits of OUT 0 select which relays, from among twelve, are to be turned on or off (RLYBIT). The top three bits of OUT 0 (RLYWD) and the bottom three bits of OUT 1 (RLYSYS) select the group of twelve relays. The details of a crosspoint are shown in figure 5-9. Upon selection of a RLYWD, all twelve relays of a group are driven to their off state by means of a ground at point Ⓐ. After a few milliseconds, the ground at Ⓐ disappears; and those relays which are connected to energized RLYBIT lines are turned on. Relays in other RLYWD groups are unaffected.

Eighteen decimal digits and three signs are displayed in electroluminescent panels. Sign displays require two relays (since no sign at all is the third "sign" symbol). Decimal digits are formed in seven segments, which are logically interdependent and can be driven with five relays. (Four binary variables are sufficient for the ten states, but the decoding equipment is more costly than if the fifth variable had been included.)

The alarm lights shown in figure 5-7 are driven from sources independent of the display system discussed here.

STANDBY OPERATION

It is expected that the only requirement on the AGC during midcourse is keeping time, a condition which obtains for the great majority of the mission. It is possible to turn off a substantial part of the AGC during midcourse (Standby condition) and thereby save something of the order of 5 to 10 KWhr of the energy needed for continuous operation.

The AGC 4 power system is shown in figure 5-10. There are three power lines associated with the AGC, one of which, +3V(B), may be turned on and off under control of a signal from the AGC. A switch, labeled "Standby Allowed," determines whether or not the +3V(B) supply is to be turned off. There are further conditions on the control of +3V(B), which will be discussed in this section.

Figure 5-11 shows the relative timing of various events, and figure 5-12 shows some of the logical relationships involved. The +3V(B) supply is controlled directly by a flip-flop R1. As long as R1 = 0, a constant stream of 102.4 KC pulses is sent to the AGC power supply, which inhibits the generation of +3V(B). A second flip-flop, R2, is used to wait until +3V(B) has risen to its final value. As long as R2 = 0, various key flip-flops are held in their reset state and the Time Pulse generating counter is held to Time 12 and not allowed to advance.

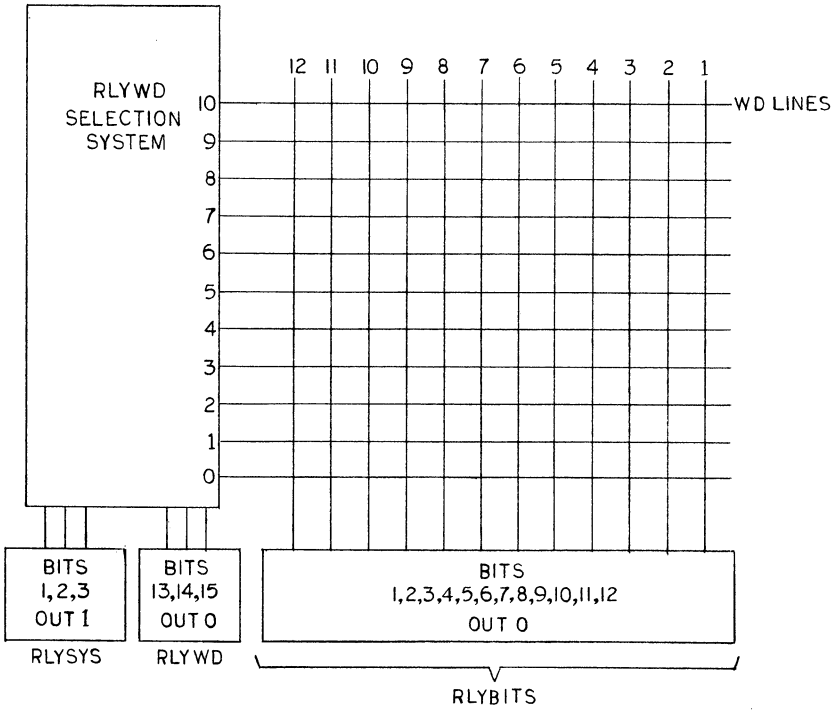


Figure 5-8. Relay Crosspoint System.

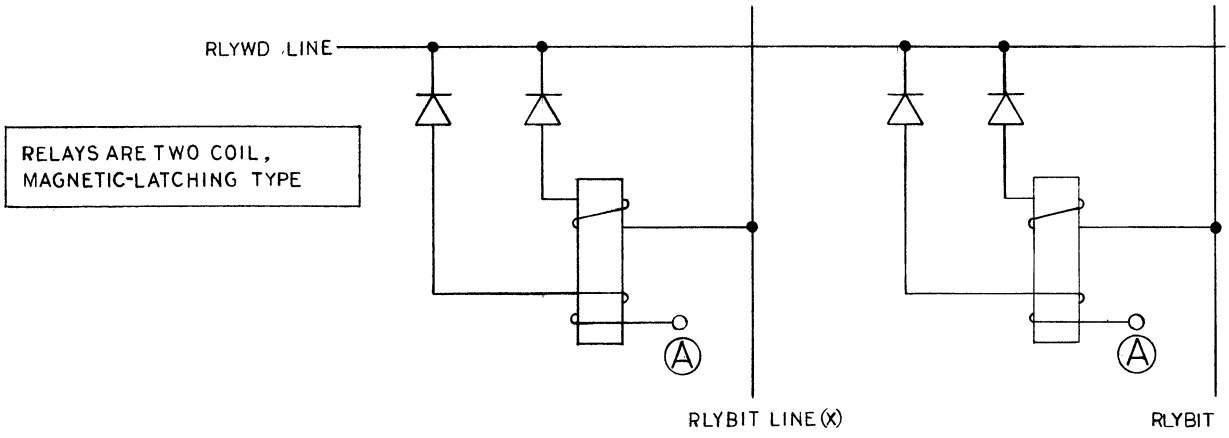


Figure 5-9. Detail of a Relay Crosspoint.

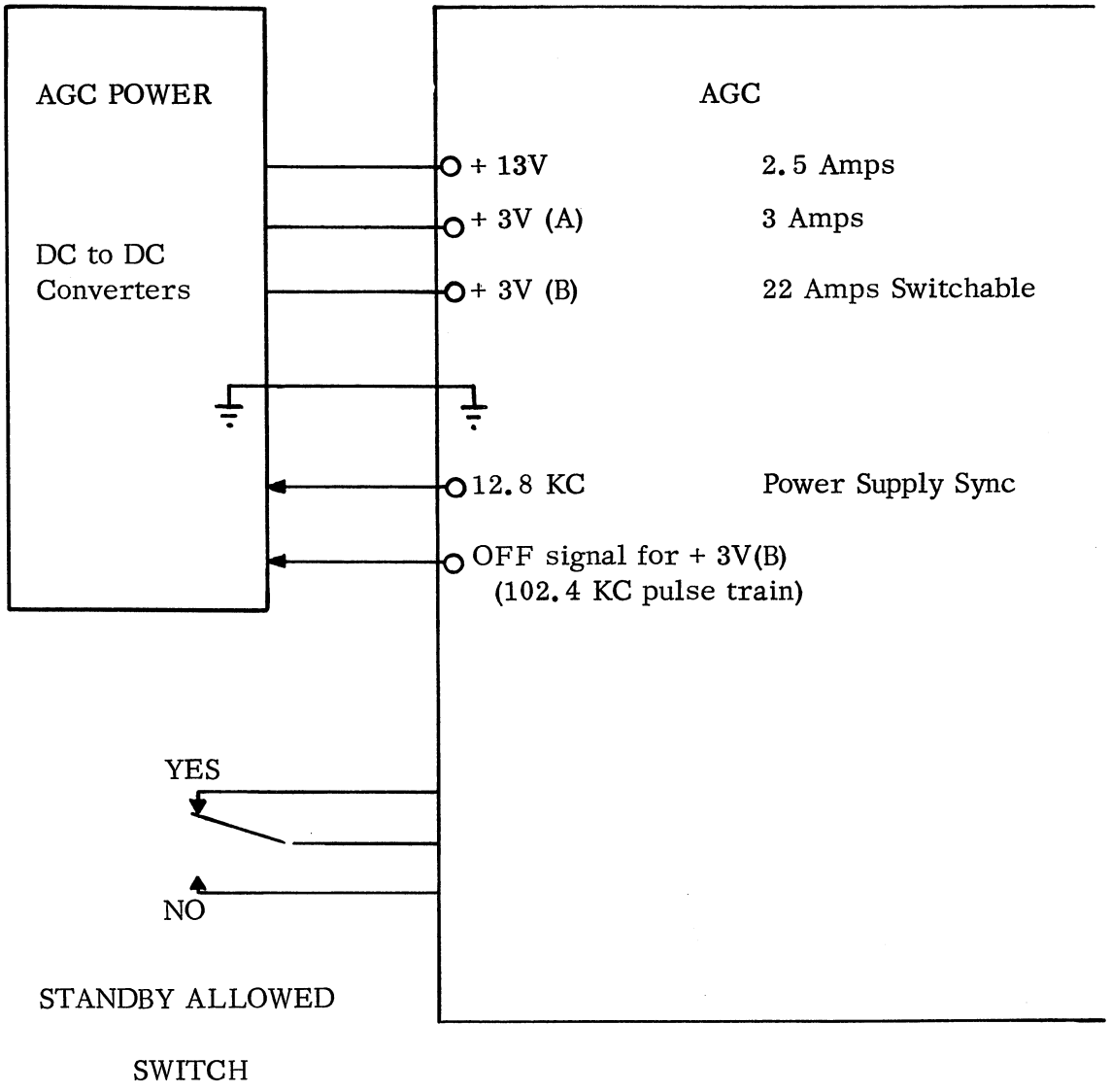


Figure 5-10. Power System.

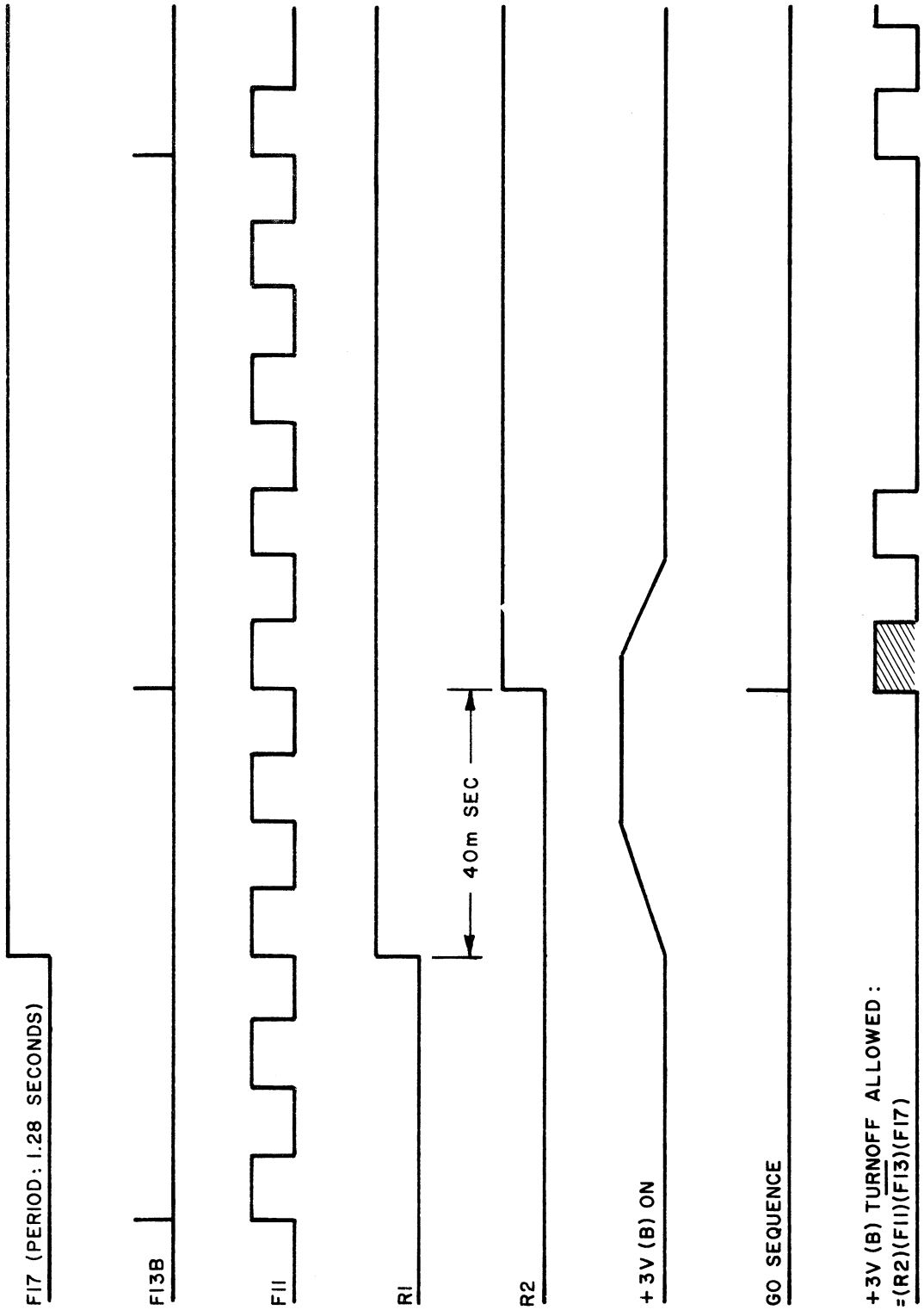
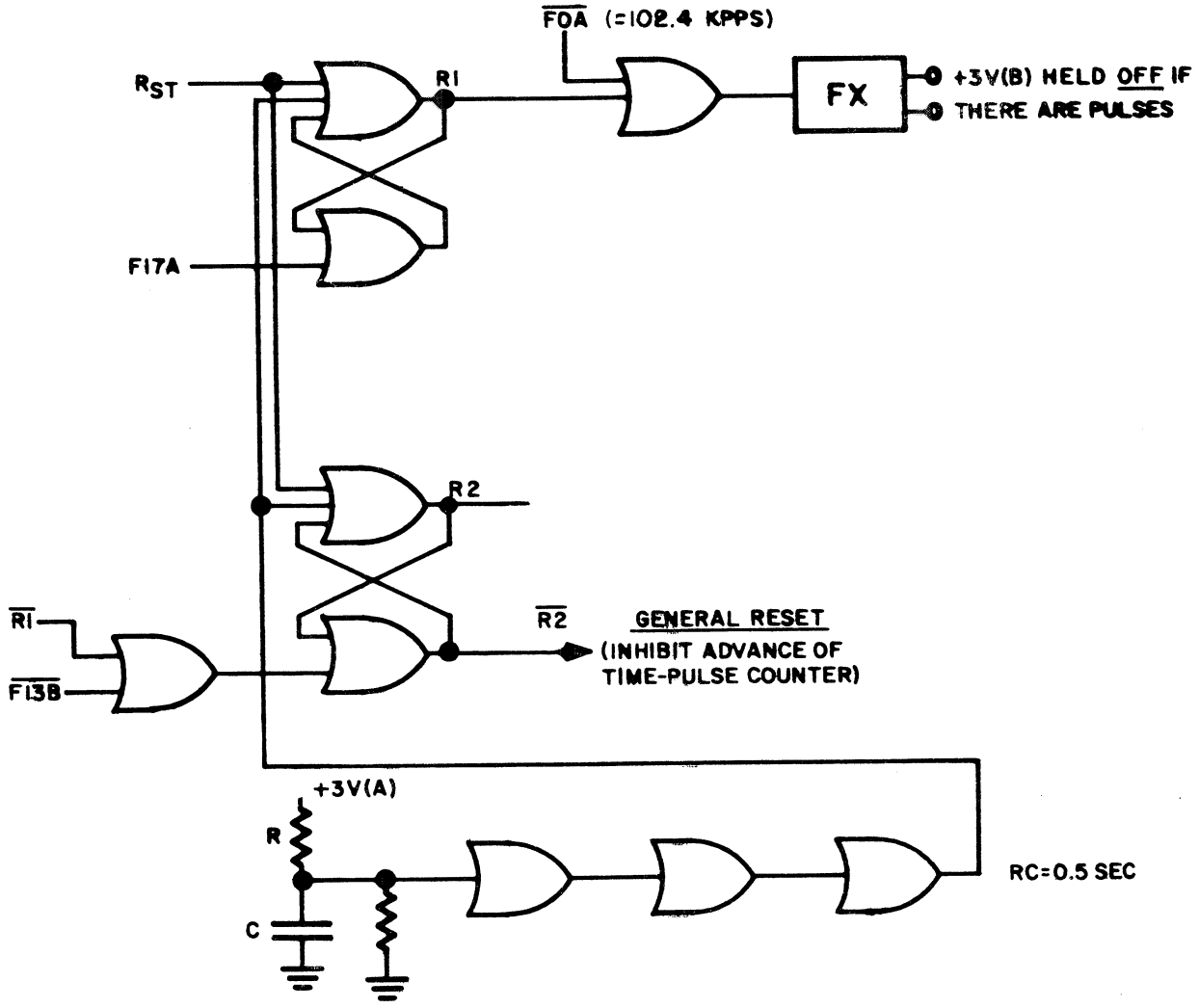


Figure 5-11. Timing for Power Switching, Detailed View.



INPUT VARIABLES

F17; F17A; F13; F13B; F11
(W1, B8) = Bit 8, OUT 1
"STANDBY ALLOWED" TOGGLE SWITCH = S. A.

FUNCTIONS

SET R1 = F17A
RST R1, R2 = (W1, B8) (S. A.)
(F17) (F13) (F11)
SET R2 = (R1) (F13B)

All Gates Connected to +3V (A)

Figure 5-12. Logic for POWOF.

A fundamental problem in the orderly turn-on of the +3V(B) supply is the uncertainty of the final state of all those flip-flops that use that supply. A priori, either state is equally likely; and some combinations of states may do damage by destroying information in Erasable Memory, or by causing unwanted outputs from the AGC, or by causing undue delays in the updating of the Time 1 counter. The following, then, are flip-flops whose states must be controlled upon turn-on of +3V(B):

1. All bits of OUT 0, OUT 1, OUT 2.

2. All alarm circuits must be inhibited.

3. All priority interrupt options must be cleared so that there are no priority requests outstanding.

4. The Time Pulse generating counter must be kept at Time 12.

5. The SQ complex must be forced to select the GO sequence.

6. All memory current drivers must be held off for the duration of the condition $(R1)\,(\overline{R2}) = 1$.

The sequence of events is as follows: the scaler pulse F17A, which occurs once every 1.28 seconds, sets R1, thereby allowing +3V(B) to go on. The rise time of +3V(B) is assumed to be less than 40 msec. Flip-flop R2 is set 40 msec after R1 is turned on, thereby unblocking the advance pulses for the Time Pulse generator. All the key flip-flops have been properly set by $\overline{R2} = 1$; in particular, the SQ complex, which determines which control pulse sequence is to be executed, had been forced to select the GO sequence. The first sequence to be executed is therefore GO, which is a Transfer Control to a predesignated starting address. The program in that location is shown in figure 5-13 in block diagram form.

TIME 1, the time counter, is incremented by 2^7 (here 2^0 is the low order bit), and the six low-order bits are made "0." This last action synchronizes the TIME 1 counter with the scaler and eliminates uncertainties which could otherwise arise from not knowing or controlling the exact time of the last +3V(B) turn-off. Any overflows of TIME 1 are added to the low-order-bit position of TIME 2.

After indexing and incrementing TIME 1, a light could be made to blink, thus giving the operator some indication of a functioning AGC.

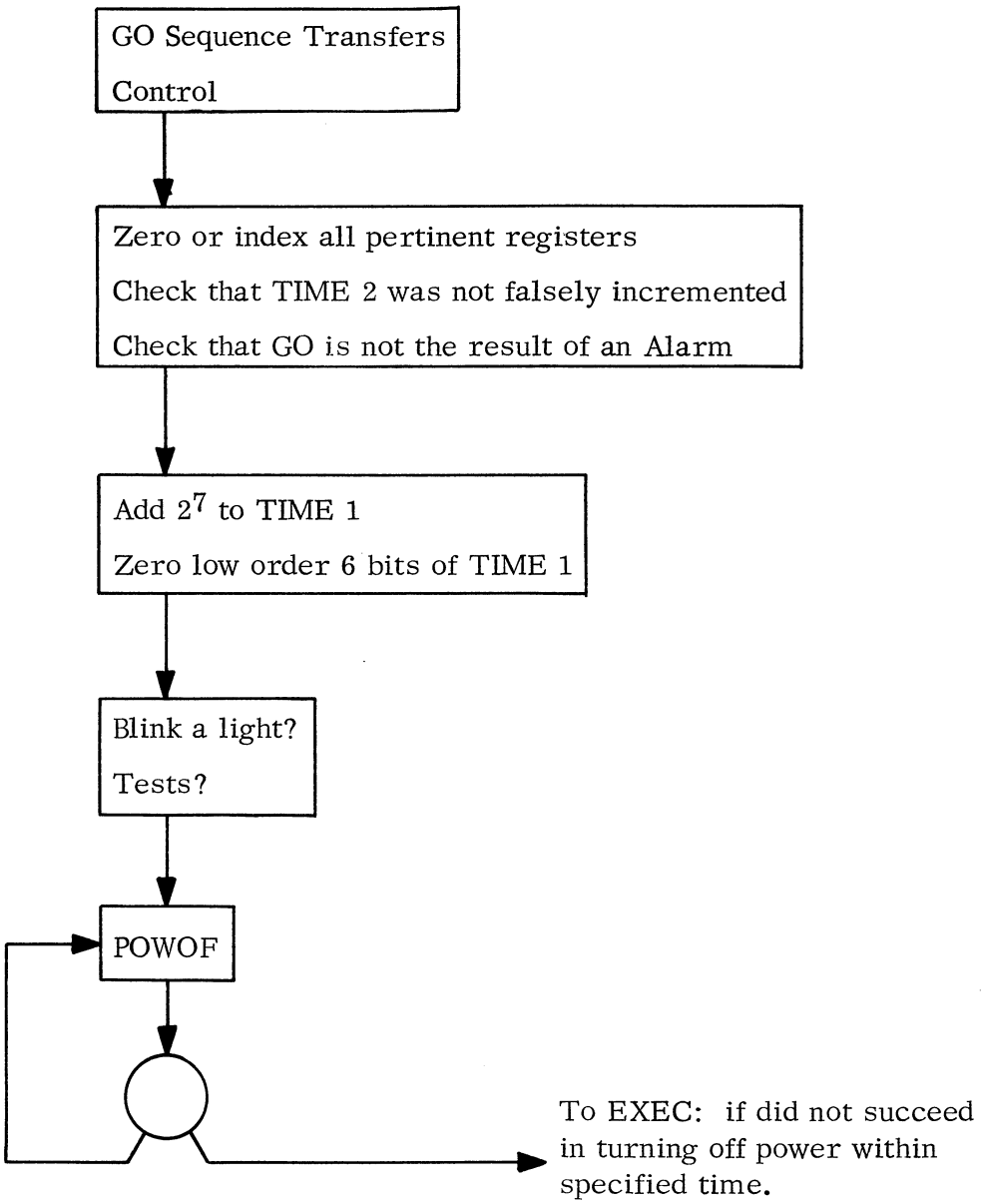


Figure 5-13. Power Off Program.

The last action taken is the program POWOF, which is as follows:

POWOF		XCH	ONE	
		AD	TIME 1	c(TIME 1) = To
		TS	TEMP 1	To + 1 ⟶ TEMP 1
	+3	XCH	OFBIT	Put "1" in Standby Outbit, (Bit 8 of OUT 1).
	+4	TS	OUT 1	See Text
		CS	TIME 1	
		AD	TEMP 1	(To + 1) -T
		CCS	A	
		TC	-5	if To + 1 > T
		BLANK		
	+10	TC	EXEC	if To + 1 = T
EXEC		XCH	PRIO	Bring priority of Endtask
		TC	NOVAC	
		TC	TASKADDR	Name of Endtask

The logical conditions on turning +3V(B) off are such that they can only be met during the 10 msec interval shown shaded in figure 5-11. The POWOF program attempts to turn off +3V(B) by setting to "1" the proper outbit. If the conditions are met, then flip-flops R1 and R2 are reset; generation of Time Steps is thereby blocked; Parity Alarm inhibited, and no instruction is executed following the TS OUT 1 on POWOF +4. If the conditions are not met for +3V(B) to be turned off, then a test is made to see if more than 10 msec have passed since incrementing TIME 1. If less than 10 msec have passed, the program again tries to turn +3V(B) off. If more than 10 msec have passed and +3V(B) is still on, control is transferred to the EXECUTIVE routine, by means of the three word program at the end. The task requested is ENDTASK, which is the lowest priority task of all and which may have provisions for attempting to turn off +3V(B) at some other time.

Failure to turn off +3V(B) need not mean a computer failure; it may be the result of processing interrupts.

An alternative to the above Program is to transfer control directly to the EXECUTIVE, requesting ENDTASK. This would be done after incrementing TIME 1 and before attempting POWOF, which would be, in this case, a part of the ENDTASK. This method makes sure that any other requests to the EXECUTIVE are processed.

A further problem exists when turning on all power supplies. There may be difficulties if the choice of an initial program is not rigidly controlled. A particularly easy choice is to arrange matters so that turning on all supplies has the same effect as only turning on +3V(B). It is necessary that +3V(A) have a rise time several times shorter than that of +3V(B), and that flip-flops R1 and R2 first come on in the state R1 = 0, R2 = 0, i.e., in Standby. This is accomplished by the RC network shown in figure 5-12.

Still another problem comes about because of the need to distinguish between leaving the Standby condition and a GO sequence which was triggered by an alarm. Alarms have flip-flops on +3V(A) which, in turn, are inputs to certain inbits. If the GO sequence is the result of an alarm, then one or more of the alarm inbits will become "1" immediately after being reset to "0" by the program.

ALARMS AND CHECKS

AGC 4 has several failure-detecting circuits which monitor a substantial part of the computer. Their purpose is not only to detect errors and to aid diagnosis and repair during flight operation but also to gather evidence as to the individual AGC's reliability during the year (or so) of preflight testing which will probably take place. Computers as complex as the AGC can sometimes give wrong answers in the absence of any component failures. Checking by means of sample problems is not adequate, for this does not provide indication of transient failures during nonsample problems.

When an alarm is triggered, the corresponding alarm light is turned on. In all cases the computer keeps working and is not stopped. First, the operator can press the "Alarm Reset" key and see if the error light goes off and stays off. This method seems best at this time for protecion against transient failures. If a component is damaged, the alarm will be triggered again shortly. The fact that an alarm did occur naturally makes one suspect the correctness of the present computations, and the operator may then call for self-checking tests, if possible.

The cost of all checking circuits may be estimated at 2% to 5% of the size of AGC 4, excluding checking programs but including the parity bit lines of both Fixed and Erasable Memories.

The various alarm functions are: Parity, Inactivity, Crosspoint Fail, Scaler Fail, TC Trap, RUPTLOCK, and AGC Power Fail.

1. Parity

This probably is the most powerful of all alarm functions. Its purpose is to check all transfers of data from either Fixed or Erasable Memory to the Arithmetic Unit. Every word has a parity bit in it, which is either "0" or "1," in order to make odd the total

number of "1's" in the word. Every word, together with its parity bit, must have at least one "1," and at least one "0." This system provides protection against a completely inactive memory, since at least one "1" must be present.

It has been the experience of the authors that most malfunctions result eventually in a parity failure indication. The word "eventually" refers to a time interval of the order of a second or less.

2. Inactivity

This alarm function detects such malfunctions as oscillator failure or 512 KC reference frequency failure. The alarm is triggered by the absence of high frequency voltage at the alarm circuit.

3. Crosspoint Fail

The Control pulses are generated by the gates at the crosspoints of a matrix, one side of which is the Time Pulse generator and the other side of which is the SQ register and Stage counter. (See Section C on Sequence generator.) The Crosspoint-Fail alarm checks that at least one crosspoint is active at any one time.

4. Scaler Fail

The scaler can be checked by verifying that one of its lowest frequency (about 0.7 cps) pulses has occurred within the last three seconds.

5. TC Trap

If the instruction in location L is TC L, then the computer enteres into an endless loop characterized by the fact that bits 13, 14, and Sg are always "0." The TC Trap alarm and corresponding light are triggered if pulses in those bit positions are absent for longer than 10 msec, excluding those "1's" due to counter activity. If this alarm is triggered, the GO sequence is selected.

The TC Trap alarm will also be triggered if the counter priority chain fails in such a way as to cause endless counting with no instructions being executed.

6. RUPTLOCK

This alarm and alarm light are triggered if the AGC 4 is in the interrupted state for longer than 20 msec. If this is the case, a GO sequence is selected.

7. AGC Power Fail

There are three power supplies for AGC 4. They are a +13V, and two +3V supplies, (A) and (B). Supply +3V(B) can be turned on and off by the computer, as explained in the description of Standby operation. These three supplies will be monitored by circuits, not yet designed in detail, which will detect the absence of proper voltage and trigger the alarm light.

Table 5-1. AGC 4, Complete Signals

The breakdown into the various categories is as follows:

INPUTS:

A. Input Signal List: This is intended as the source for various other tables, and it is meant to be a complete listing of every signal coming into the AGC.

B. Input Summary: The input signals are grouped by subsystems.

C. INBIT Assignments: Some of the input signals come into INBITS. Other INBITS come from signals originating within the AGC.

D. Counter Assignments: Address and use of counters.

E. Interrupt (RUPT) Assignments: RUPTS are originated by both external and internal signals.

OUTPUTS:

F. Output Signal List: A complete listing of all signals which go out of the AGC.

G. Output Summary: Signals grouped by subsystems.

H. OUTBIT Assignments:

I. Scaler Nomenclature: The timepiece for the AGC is the Clock and Scaler section, which generates all the timing signals. This section defines various internal signals.

J. Timing Specifications: Many of the output signals can be defined in terms of the Scaler frequencies defined in E.

K. In-Out System Scaling:

Table 5-1. AGC 4, Complete Signals (Continued)

A. INPUT SIGNAL LIST

Legends and Symbols

Electrical Character of Interface.
C = Contact on switch or relay inside AGC or DSKY.
CC = Contact on switch or relay outside AGC.
X = Transformer within AGC.

RUPT: Interrupt associated with that input signal, if any.
INBIT: Input bit assoicated with that input signal, if any. The register is identified by v, the bit by B.
CNTR: Counter address, in octal, if any.

Each entry represents a signal, not necessarily a wire.

Totals: 23 C; 1 CC; 40 X.

SIGNAL	ELECTRICAL INTERFACE	RUPT	INBIT v. B	COUNTER ADDRESS
Mark	CC	5	0, 15	
"STANDBY ALLOWED" SW	C	5	0, 14	
Keyboard Active	C		0, 6	
Key 5	C		0, 5	
Key 4	C		0, 4	
Key 3	C		0, 3	
Key 2	C		0, 2	
Key 1	C		0, 1	
OR of C1, 2, 3, 4, 10	C		1, 5	
OR of C5, 6, 7, 8, 9	C		1, 6	
OR of C11, 12	C		1, 7	
Bad Boost	X	2	2, 14	
Abort	X	2	2, 13	
IMU Fail	X	2	2, 12	
PIPA Fail	X	2	2, 11	
CDU Fail	X	2	2, 10	
OPT Fail	X	2	2, 9	
TRKR Fail	X	2	2, 8	
Photocell +	X		2, 5	
Photocell -	X		2, 4	
OR Zero, TRKR	X		2, 3	
OR Zero, OPT	X		2, 2	
OR Zero, CDU	X		2, 1	

Table 5-1. AGC 4, Complete Signals (Continued)

SIGNAL		ELECTRICAL INTERFACE	RUPT	INBIT v. B	COUNTER ADDRESS
Lift-off		X		3, 15	
Mode 1 S/C		C		3, 14	
Mode 2 S/C		C		3, 13	
Mode 1 OPT		C		3, 10	
Mode 2 OPT		C		3, 9	
Mode 3 OPT		C		3, 8	
Zero OPT		C		3, 7	
Error Light Reset		C		-	
TRNSW		C		3, 6	
ATTSW		C		3, 5	
K4		C		3, 4	
K3		C		3, 3	
K2		C		3, 2	
K1		C		3, 1	
PIPA	X+	X			44
	X-	X			44
	Y+	X			45
	Y-	X			45
	Z+	X			46
	Z-	X			46
CDU	X+	X			47
	X-	X			47
	Y+	X			50
	Y-	X			50
	Z+	X			51
	Z-	X			51
OPT	X+	X			52
	X-	X			52
	Y+	X			53
	Y-	X			53
TRKR	X+	X			54
	X-	X			54
	Y+	X			55
	Y-	X			55
	R+	X			56
	R Shift	X			56
LNK	UPSYNC	X			41
	DATA IN	X			41
	BITSYNC	X			
	START	X			
	ENDPULSE	X			

Table 5-1. AGC 4, Complete Signals (Continued)

B. INPUT SUMMARY

Legends and Symbols

N: Number of signals.
X: Transformer inside AGC.
C: Contact inside AGC or DSKY.
CC: Contact outside AGC or DSKY.

SUBSYSTEM	N	E	DEVICE	ENUMERATION, COMMENTS
IMU	6	X	PIPA	3 Counters, ±
	6	CC	Controls	Relays K1, K2, K3, K4; Switches ATTSW, TRNSW
CDU	6	X	Encoders	3 Counters, ±
OPTICS	4	X	SXT or SCT	2 Counters
	6	X	TRKR	3 Counters
	1	CC	Controls	Command to Zero encoders
	5	CC	Controls	3 Mode Selection, 2 Photocell detectors
	1	CC	MARK SW	MARK pushbutton
LNK	3	X	Programmer	START, STOP, AND BITSYNC signals from DOWNTEL Programmer
	2	X	Receiver	UPSYNC and DATA lines from onboard receiver. To UPLINK Counter (TRKR Range in LEM)
PSA PWR	5	X		Failure bits for PIPA IMU CDU OPT Tracking
	3	X		Zero indicators for CDU OPTICS Tracking
AGC PWR	4	GND		AGC Power, including switchable +3V(B) supply
		+13V		
		+3V(A)		
		+3V(B)		

Table 5-1. AGC 4, Complete Signals (Continued)

SUBSYSTEM	N	E	DEVICE	ENUMERATION, COMMENTS
S/C	1	X		Bad Boost Signal
	1	X		Abort Signal
	2	CC		Mode Select
	1	X		Lift-off Signal
DSKY (Displays and Keyboard)	10	C	Keyboard (or tape, in factory tests)	8 Data lines, 1 Activity line, 1 Error Light Reset
	3	C		3 OR of C relays
	1	C	SW	"Error Light Reset" SW
	1	C	SW	"Standby Allowed" SW

Table 5-1. AGC 4, Complete Signals (Continued)

C. INBIT ASSIGNMENTS

REGISTER	BIT ASSIGNMENT		COMMENTS
IN 0 (= V0)	Bit 1	KEY 1	All bits of this register are normally zero.
	2	KEY 2	
	3	KEY 3	
	4	KEY 4	
	5	KEY 5	
	6	KEY ACTIVE (KEYRUPT)	
	7	Not Used	
	8	Not Used	
	9	Not Used	
	10	Not Used	
	11	Not Used	
	12	Not Used	
	13	Not Used	
	14	"Standby Allowed" SW	
	15	MARK (KEYRUPT)	
IN 1 (= V1)	Bit 1	1600 pps	
	2	800 pps	
	3	400 pps	
	4	200 pps	
	5	OR C1, 2, 3, 4, 10	IMU Relay Drivers
	6	OR C5, 6, 7, 8, 9	S/C Alarm lights
	7	OR C11, 12	OPT Relay Drivers
	8	Not Used	
	9	Not Used	
	10	Not Used	
	11	PARITY	
	12	CROSSPOINT FAIL	
	13	SR INACTIVE (GO)	AGC Alarms
	14	RUPTLOCK (GO)	
	15	TCA Trap (GO)	

Table 5-1. AGC 4, Complete Signals (Continued)

REGISTER	BIT ASSIGNMENT		COMMENTS
IN 2 (= V2)	Bit 1	OR CDU $\overline{\text{Zero}}$	
	2	OR SXT Zero	
	3	OR TRACKER $\overline{\text{Zero}}$	
	4	Photocell -	
	5	Photocell +	
	6	Not Used	
	7	Not Used	
	8	Tracker Fail (RUPT)	
	9	OPTICS Fail (RUPT2)	
	10	CDU Fail (RUPT2)	
	11	ACCEL Fail (RUPT2)	
	12	IMU Fail (RUPT2)	
	13	ABORT (RUPT2)	
	14	BAD BOOST (RUPT2)	
	15	AGC Power Fail (RUPT2)	
IN 3 (= V3)	Bit 1	K1 Zero CDU SW	
	2	K2	
	3	K3	
	4	K4	
	5	ATTSW	
	6	TRNSW	
	7	ZERO OPT	
	8	MODE SELECT OPT	
	9	MODE SELECT OPT	
	10	MODE SELECT OPT	
	11	Not Used	
	12	Not Used	
	13	MODE SELECT S/C	
	14	MODE SELECT S/C	
	15	Lift-off	

Table 5-1. AGC 4, Complete Signals (Continued)

D. COUNTER ASSIGNMENTS

OCTAL ADDRESS	NAME	NOTE	INPUTS	OUTPUT	OUTPUT GOES TO
0034	OVCTR	1	+, -		
35	TIME 1	2	+	OVFLOW	TIME 2
36	TIME 2				
37	TIME 3	3	+	OVFLOW	WAITRUPT
40	TIME 4	4	+	OVFLOW	DISRUPT
41	UPLINK	5	+, Shift	OVFLOW	UPRUPT
42	OUTCR I	6	+	OVFLOW	K. O. OUT 3
43	OUTCR II		+	OVFLOW	K. O. OUT 4
44	PIPA X		+, -		
45	PIPA Y		+, -		
46	PIPA Z		+, -		
47	CDU X (*)		+, -		
50	CDU Y (*)		+, -		
51	CDU Z (*)		+, -		
52	OPT X (*)		+, -		
53	OPT Y (*)		+, -		
54	TRKR X (*)		+, -		
55	TRKR Y (*)		+, -		
56	TRKR R		+, Shift		

(*) No sign correction upon overflow or underflow. Overflows not used. (Cf DD Memo #29)

Table 5-1. AGC 4, Complete Signals (Continued)

Notes on Counters

1. OVCTR

   This register is incremented (decremented) if the ADx instruction results in an overflow (underflow). Interrupting programs which do ADx's must take care to leave this register in its original state, either by preserving its contents or guaranteeing no overflows.

2. TIME 1, TIME 2

   The AGC's timepiece counts time continuously from some arbitrary starting point. Units of TIME 1 are worth 10 msec. Inputs to TIME 1 come from F10A (=100 pps) for TIME 1.

3. TIME 3

   This register is used as a general purpose timer, and causes interrupts at specified future times. The WAITLIST program then executes the appropriate action. Units are worth 10 msec.

4. TIME 4

   Has essentially the same function as TIME 3, but is concerned only with the display programs. Units are worth 10 msec.

5. UPLINK

   This counter is used as a serial-to-parallel converter. When bits come in from the outside world, the contents of UPLINK are either shifted or incremented (bit is "1"). The first bit is always a "1", so that UPLINK will overflow when sixteen bits have been sent in. This overflow is used to detect the end of a loading operation. UPLINK may also be used in the LEM for accepting a serial transmission of range data from a tracking device (Radar or Laser).

6. OUTCR I (was RATECNTR in AGC 3)

   This counter counts the OR of all the set pulses which go to the CDU's and the gyros. See Section C on Output Rates.

   OUTCR II

   See Section C on Output Rates.

Table 5-1. AGC 4, Complete Signals (Continued)

E. INTERRUPT (RUPT) ASSIGNMENTS

1. T3 RUPT
Inputs: Overflow of TIME 3.

Programs: WAITLIST.

2. ERRUPT
Inputs: Error signals from AGC POWER FAIL
BAD BOOST
ABORT
IMU FAIL
ACCEL FAIL
CDU FAIL
OPT FAIL
TRKR FAIL.

Programs: Alarms, PIPA scale changes, abort.

3. DSRUPT
Inputs: Telemetry "End Pulse" from NAA's telemetry programmer; TIME 4 overflows.

Programs: Displays and DOWNLINK, for loading OUT 5 for use in telemetry. Also possibly for generating output rates.

Status: Moderately well defined, both inputs and programs.

4. KEYRUPT
Inputs: Activity signal from either MARK, keyboards or Tape reader.

Programs: Keyboard input programs.

5. UPRUPT
Inputs: UPLINK overflow.

Programs: Same as KEYRUPT, plus option for accepting LEM Range data for TRACKER.

Table 5-1. AGC 4, Complete Signals (Continued)

F. OUTPUT SIGNAL LIST

Legends and Symbols

X: Transformer in AGC.
C: Contact in AGC.
N: Collector from NPN, in AGC.
*: "Involuntary" signals, not under program control.

Totals: 41 X; 22 C; 34 N.

		SIGNAL	ELECTRICAL INTERFACE
IMU		* PIPA Set	X
		* PIPA Rst	X
		* PIPA Clock	X
		PIPA Fine Compensation	X
		GYRO X+	X
		GYRO X-	X
		GYRO Y+	X
		GYRO Y-	X
		GYRO Z+	X
		GYRO Z-	X
		* GYRO Rst	X
		Relay Driver C1	C
		Relay Driver C2	C
	(Located in DSKY)	Relay Driver C3	C
		Relay Driver C4	C
		Relay Driver C10	C
		Relay Driver C10 bar	C
CDU		CDU X+	X
		CDU X-	X
		CDU Y+	X
		CDU Y-	X
		CDU Z+	X
		CDU Z-	X
		* CDU Rst	X
OPT (LEM Controls)		OPT X+	X
		OPT X-	X
		OPT Y+	X
		OPT Y-	X
		* OPT Rst	X
	(Located in DSKY)	Relay Driver C11	C
		Relay Driver C12	C

Table 5-1. AGC 4, Complete Signals (Continued)

	SIGNAL	ELECTRICAL INTERFACE
LNK	DATA OUT (to DOWNTEL)	X
	* AGC Temp	C
	Parity Fail	C
	TCA Trap	C
AGC Alarms	Ruptlock	C
(Contacts located	* Scaler Fail	C
in DSKY)	* Inactivity	C
	* Crosspoint Fail	C
PSA PWR	* 800 Set	X
(Power	* 800 Rst	X
Supplies)	* 3200 Set	X
	* 3200 Rst	X
	* 12.8 KC Set	X
	* 25.6 KC Set	X
	* 25.6 KC Rst	X
	* 102.4 KC Set	X
(Relay located	P1 Relay, Contact A	C
in AGC PWR)	Contact B	C
AGC PWR	102.4 KC ON-OFF	X
	* 12.8 KC PWR Sync	X
S/C	Thrust +	X
	Thrust -	X
	* Thrust Rst	X
	Clock Start (102.4 KC)	X
	Engine Start (102.4 KC)	X
	Engine Cut-off (102.4 KC)	X
	* 512 KC Master Frequency	X
	Alarm Light Driver C5	C
	Alarm Light Driver C6	C
(Located in	Alarm Light Driver C7	C
DSKY)	Alarm Light Driver C8	C
	Alarm Light Driver C9	C

Table 5-1. AGC 4, Complete Signals (Continued)

	SIGNAL		ELECTRICAL INTERFACE
DSKY	OUTBITS:	w0, B1	N
		w0, B2	N
		w0, B3	N
		w0, B4	N
		w0, B5	N
	Rlybits	w0, B6	N
		w0, B7	N
		w0, B8	N
		w0, B9	N
		w0, B10	N
		w0, B11	N
		w0, B12	N
		w0, B13	N
	Rlwd	w0, B14	N
		w0, B15	N
		w1, B1	N
	Rlysys	w1, B2	N
		w1, B3	N
	Timing	* 25 A	N
	Signals	* 25 B	N
		* 50 A	N
	Parity Alarm		N
	Inactivity (SR)		N
	AGC Power Failed		N
	Crosspoint Fail		N
	RUPTLOCK Fail		N
	TCA Trap (Counters hung)		N
	Scaler Fail		N
	Spare 1		N
	Spare 2		N
	Spare 3		N
	+13 v supply		-
	250 v, 800 cps		-

Table 5-1. AGC 4, Complete Signals (Continued)

G. OUTPUT SUMMARY

Legends and Symbols

N: Number of Signals.
X: Transformer inside AGC.
C: Contact inside AGC or DSKY.
P: Collector from NPN, in AGC.

SUBSYSTEM	N	E	DEVICE	ENUMERATION, COMMENTS
IMU	3	X	PIPA	Set, Rst, Clock
	7	X	GYRO	3 ± Rates, 1 Rst
	6	C	Controls	Relay drivers C1, C2, C3, C4, C10, $\overline{C10}$
CDU	7	X	D/A Converter	3 ± Rates, 1 Rst
OPT	7	X	SCT, SCT, or TRKR (LEM)	3 ± Rates, 1 Rst
	2	C	Controls	Relay Drivers C11, C12
LNK	1	X	PCM	1 Output bits for DOWNTEL
	7	C	AGC Alarm	6 AGC Alarm and Temp.
PSA PWR (Power Supplies)	2	X	800 pps	1 Set, 1 Rst
	2	X	3200 pps	1 Set, 1 Rst
	1	X	12.8 Kpps	1 Set
	2	X	25.6 Kpps	1 Set, 1 Rst
	1	X	102.4 Kpps	1 Set
	2	C	P1 Relay	AGC Power Fail Indications
AGC PWR	2	X	102.4 KC Set, 12.8 KC	For controlling the +3(B) switchable supply and sync.
S/C	3	X	Thrust (LEM)	1 ± Rate, 1 Rst; for controlling variable Thrust in LEM
	3	X	Engine Controls	Engine Sequencer Start (102.4 Kpps, 3 μsec) Engine Start (102.4 Kpps, 3 μsec) Engine Cut-off (102.4 Kpps 3 μsec)
	1	X	Clock	512.0 KC square wave Master Frequency for System
	5	C		C5, C6, C7, C8, C9 to Alarm and Status Lights

Table 5-1. AGC 4, Complete Signals (Continued)

SUBSYSTEM	N	E	DEVICE	ENUMERATION, COMMENTS
DSKY (Displays and Keyboard)	18	P	Relay System	12 for Rlybit 3 for Rlywd 3 for Rlysys
	3	P	Timing Signals	Timing for Relay Selection System
	3	C	Power +3, +13	AGC Power
	2	C	250 v, 800 cps	Power for E-L Panels
	3	P	Keyboard	
	10	P	Alarm Lights	

Table 5-1. AGC 4, Complete Signals (Continued)

H. OUTBIT ASSIGNMENTS

REGISTER	BIT ASSIGNMENT		
OUT 0 (= W0)	BIT	1	Rlybit 1
		2	Rlybit 2
		3	Rlybit 3
		4	Rlybit 4
		5	Rlybit 5
		6	Rlybit 6
		7	Rlybit 7
		8	Rlybit 8
		9	Rlybit 9
		10	Rlybit 10
		11	Rlybit 11
		12	Rlybit 12
		13	Rlywd 1
		14	Rlywd 2
		15	Rlywd 3
OUT 1 (= W1)	Bit	1	Rlysys 1
		2	Rlysys 2
		3	Rlysys 3
		4	Tape Select } Factory Testing only
		5	Tape Clutch }
		6	Block UPLINK
		7	Not Used
		8	Slow Speed, STANDBY
		9	Word Order Telem
		10	Block Endpulse
		11	PIPA SCALE FINE
		12	PIPA SCALE EMERG
		13	ENG START
		14	START CLOCK
		15	ENG OUT-OFF

Table 5-1. AGC 4, Complete Signals (Continued)

REGISTER	BIT ASSIGNMENT		
OUT 2	Bit 1	OPT X	
(= W2)	2	OPT Y	
	3	THRUST	RATE II
	4	+	
	5	-	
	6	Not Used	
	7	Not Used	
	8	GYR X	
	9	GYR Y	
	10	GYR Z	
	11	CDU X	RATE I
	12	CDU Y	
	13	CDU Z	
	14	+	
	15	-	
OUT 3	Bit 1	Not Used	
(= W3)	2	Not Used	
	3	Not Used	
	4	Not Used	
	5	Not Used	
	6	Not Used	
	7	Not Used	
	8	Not Used	SPARE
	9	Not Used	
	10	Not Used	
	11	Not Used	
	12	Not Used	
	13	Not Used	
	14	Not Used	
	15	Not Used	
OUT 4	(16 bits)		
(= W4)	Word to be sent via DOWNLINK is placed here, with correct parity.		

Table 5-1. AGC 4, Complete Signals (Continued)

I. SCALER NOMENCLATURE

In order to present the input and output timing specifications, it is convenient to list the names of the various scaler signals, and some of their properties. These output signals can then be described in terms of the scaler signals.

Signals FxA, FxB, FxC, FxD represent pulses 90° (electrical) out of phase. A leads C (not B) by 90° ; C leads B by 90° ; B leads D by 90° . This system holds for signals F2A, B, C, D through F12A, B, C, D.

Signals Fx, without a letter following, mean square waves.

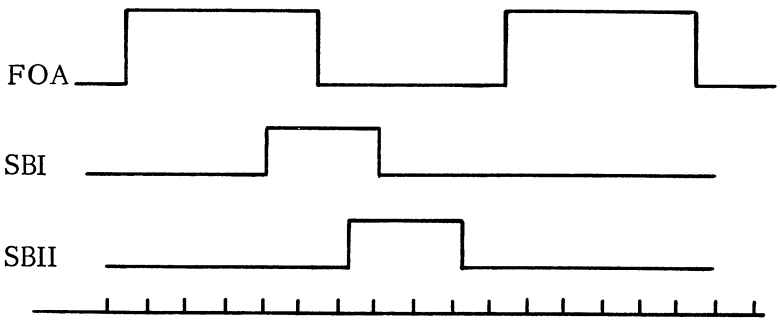
SIGNAL NAME	FREQUENCY	DURATION μSEC	PHASE (LEADING EDGES)	COMMENTS
FOA	102.4 KC	4.9		Defines phasing.
FOB	102.4 KC	4.9	$\overline{\text{FOA}}$	
F1A	51.2 KC	9.75	FOA. $\overline{\text{F1B}}$	In phase with every other FOA.
F1B	51.2 KC	9.75	FOA. $\overline{\text{F1A}}$	
F2	25.6 KC	39.2	F1	F2 is in phase with every other F1.
F2A	25.6 KC	9.75	F1A. $\overline{\text{F2B}}$	
F2B	25.6 KC	9.75	F1A. $\overline{\text{F2A}}$	
F2C	25.6 KC	9.75	F1B. $\overline{\text{F2D}}$	
F2D	25.6 KC	9.75	F1B. $\overline{\text{F2C}}$	
F3	12.8 KC	78.1	F2	
F3A	12.8 KC	9.75	F2A. $\overline{\text{F3B}}$	
F3B	12.8 KC	9.75	F2A. $\overline{\text{F3A}}$	
F3C	12.8 KC	9.75	F2B. $\overline{\text{F3D}}$	
F3D	12.8 KC	9.75	F2B. $\overline{\text{F3C}}$	
F4	6.4 KC	156.2	F3	
F4A	6.4 KC	9.75	F3A. $\overline{\text{F4B}}$	
etc.				
F5	3.2 KC	312.5	F4	
F5A	3.2 KC	9.75	F4A. $\overline{\text{F5B}}$	
etc.				
F6	1.6 KC	625	F5	
F6A	1.6 KC	9.75	F5A. $\overline{\text{F6B}}$	
etc.				
F7	800 pps	1250.00	F6	
F7A	800 pps	9.75	F6A. $\overline{\text{F7B}}$	
etc.				
F8	400 pps	2500.00	F7	
F8A	400 pps	9.75	F7A. $\overline{\text{F8B}}$	
etc.				

Table 5-1. AGC 4, Complete Signals (Continued)

SIGNAL NAME	FREQUENCY	DURATION μSEC	PHASE (LEADING EDGES)	COMMENTS
F9	200 pps	5000.00	F8	
F9A	200 pps	9.75	F8A. $\overline{F9B}$	
etc.				
F10	100 pps	10000.00	F9	
F10A	100 pps	9.75	F9A. $\overline{F10B}$	
etc.				
F11	50 pps	20000.00	F10	
F11A	50 pps	9.75	F10A. $\overline{F11B}$	
etc.				
F12	25 pps	40000.00	F11	
F12A	25 pps	9.75	F11A. $\overline{F12B}$	
etc.				
F13	12.5 pps	80000.00	F12	
F13A	12.5 pps	9.75	F12A. $\overline{F13B}$	
etc.				
F14	6.3 pps	160000.00	F13	
F14A	6.3 pps	9.75	F13A. $\overline{F14B}$	
etc.				
F15	3.2 pps	320000.00	F14	
F15A	3.2 pps	9.75	F14A. $\overline{F15B}$	
etc.				
F16	1.6 pps	640000.00	F15	
F16A	1.6 pps	9.75	F15A. $\overline{F16B}$	
etc.				
F17	.8 pps	1280000.00	F16	
F17A	.8 pps	9.75	F16A. $\overline{F17B}$	
etc.				
F18	.4 pps	2560000.00	F17	
F18A	.4 pps	9.75	F17A. $\overline{F18B}$	
etc.				

Table 5-1. AGC 4, Complete Signals (Continued)

SIGNAL NAME	FREQUENCY	DURATION μSEC	PHASE (LEADING EDGES)	COMMENTS
SBI (STROBE I)	102.4 KC	2.9	FOA + 4 μsec	Leading edge of SBI is delayed about 4 μsec relative to leading edge of FOA.
SBII (STROBE II)	102.4 KC	2.9	FOB + 1 μsec	Leading edge of SBII follows the leading edge of SBI by 2 μsec, and follows the leading edge of FOB by 1 μsec.



Time scale: 1 μsec per interval

Table 5-I. AGC 4, Complete Signals (Continued)

J. TIMING SPECIFICATIONS

Each of the output signals listed below has a corresponding Timing Function and ON-OFF function. The variables of the Timing Function are defined under "Scaler Nomenclature." The ON-OFF function variables are the OUTBITS (wk, Bj), where k is the output register, and j the output bit.

SIGNAL NAME	TIMING FUNCTION	ON-OFF FUNCTION	
PIPA Set	(F4D) (F5) (F6) (SBI)	1	
PIPA Rst Normal	(F4D) (F5) (F6) (SBI)	(W4, B12).	(W4, B11)
PIPA Rst Fine	(F4C) (F5) (SBI)	(W4, B12).	(W4, B11)
PIPA Rst Emergency	no reset pulse	(W4, B12)	
PIPA Clock	(F3B) (SBII)	1	
GYRO X+	(F6B) (SBII)	(W2, B14).	(W2, B9)
GYRO X-	(F6B) (SBII)	(W2, B13).	(W2, B9)
GYRO Y+	(F6B) (SBII)	(W2, B14).	(W2, B8)
GYRO Y-	(F6B) (SBII)	(W2, B13).	(W2, B8)
GYRO Z+	(F6B) (SBII)	(W2, B14).	(W2, B7)
GYRO Z-	(F6B) (SBII)	(W2, B13).	(W2, B7)
GYRO Rst	(F6B) (SBI)	1	
CDU X+	(F6B) (SBII)	(W2, B14).	(W2, B12)
CDU X-	(F6B) (SBII)	(W2, B13).	(W2, B12)
CDU Y+	(F6B) (SBII)	(W2, B14).	(W2, B11)
CDU Y-	(F6B) (SBII)	(W2, B13).	(W2, B11)
CDU Z+	(F6B) (SBII)	(W2, B14).	(W2, B10)
CDU Z-	(F6B) (SBII)	(W2, B13).	(W2, B10)
CDU Rst	(F6B) (SBI)	1	
OPT X+	(F6B) (SBII)	(W3, B14).	(W3, B12)
OPT X-	(F6B) (SBII)	(W3, B13).	(W3, B12)
OPT Y+	(F6B) (SBII)	(W3, B14).	(W3, B11)
OPT Y-	(F6B) (SBII)	(W3, B13).	(W3, B11)
OPT Rst	(F6B) (SBI)	1	
PWR 800 Set	(F7A) (SBI)	1	
PWR 800 Rst	(F7B) (SBI)	1	
PWR 3200 Set	(F5A) (SBI)	1	
PWR 3200 Rst	(F5B) (SBI)	1	
PWR 12.8 KC Set	(F3A) (SBI)	1	
PWR 25.6 KC Set	(F2A) (SBI)	1	
PWR 25.6 KC Rst	(F2B) (SBII)	1	
PWR 102.4 KC Set	(FOA) (SBII)	1	

Table 5-1. AGC 4, Complete Signals (Continued)

SIGNAL NAME	TIMING FUNCTION	ON-OFF FUNCTION
AGC PWR 102.4 KC Set	(FOA) (SBII)	See "Power Switching" Section.
AGC 102.4 KC Rst	(FOB) (SBII)	1
S/C Thrust +	(F6B) (SBII)	(W3, B14). (W3, B9)
S/C Thrust -	(F6B) (SBII)	(W3, B13). (W3, B9)
S/C Thrust Rst	(F6B) (SBI)	1
S/C Clock Start	(SBI)	(W4, B15)
S/C Eng. Start	(SBI)	(W4, B13)
S/C Eng. Cut-off	(SBI)	(W4, B13)

Table 5-1. AGC 4, Complete Signals (Continued)

K. IN-OUT SYSTEM SCALING
(based on Hoag's AGANI, pg. 490)

IN	COUNTER	MAXIMUM RATE, pps	NOMINAL SCALES
	UPLINK	80	N. A.
	PIPA X 1	1600	Normal: 5.22 cm/sec/pulse.
	PIPA Y	1600	Fine: 1.68 cm/sec/pulse.
	PIPA Z	1600	Emergency: 9.34 cm/sec/pulse.
	CDU X	1600	2^{15} bits/rev = 0.1918 mr/pulse.
	CDU Y	1600	
	CDU Z	1600	
	SXT X 2	3600	2^{17} bits/rev, line of sight = .0479 mr/pulse.
	SXT Y	1600	2^{15} bits/rev, line of sight = .1918 mr/pulse.
	SXT Z 3	(1600)	
	SCT X	1600	2^{15} bits/rev = .1918 mr/pulse.
	SCT Y	1600	

1. $\text{FINE} = \frac{\text{NORMAL}}{4(1 + \Delta 2)}$; $\Delta 2 = +.55$.

$\text{EMERGENCY} = \frac{2(\text{NORMAL})}{(1 + \Delta 1)}$; $\Delta 1 = -.105$.

2. Assume SXT X is the Star Trunnion axis.
3. There may only be two axes to the SXT, X, and Y.

Table 5-1. AGC 4, Complete Signals (Continued)

OUT	COUNTER	SCALE	RATE CORRESPONDING TO 1600 pps
	CDU X CDU Y CDU Z	2^{15} pulses/rev .1918 mr/pulse	17.6 deg/sec.
	GYRO X GYRO Y GYRO Z	2^{19} pulses/rev .01198 mr/pulse	1.1 deg/sec.
	OPTICS X OPTICS Y OPTICS Z	NOT YET ASSIGNED	---
	THRUST CONTROL	NOT YET ASSIGNED	---

This page intentionally left blank.