MASSACHUSETTS INSTITUTE OF TECHNOLOGY

APOLLO

GUIDANCE AND NAVIGATION

E-2254

AUXILARY MEMORY SYSTEM FINAL REPORT ON PHASE I

by

D.J. Bowler

APRIL 1968

MIT INSTRUMENTATION LABORATORY

CAMBRIDGE 39, MASSACHUSETTS

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

APOLLO

GUIDANCE AND NAVIGATION

Approved: Eldon C Hall Date: 5/15/68
ELDON C. HALL, DIRECTOR, DIGITAL DEVEL.
APOLLO GUIDANCE AND NAVIGATION PROGRAM

Approved: David G Hoag Date: 20 May 68
DAVID G. HOAG, DIRECTOR
APOLLO GUIDANCE AND NAVIGATION PROGRAM

Approved: Ralph R. Ragan Date: 20 May 68
RALPH R. RAGAN, DEPUTY DIRECTOR
INSTRUMENTATION LABORATORY

E-2254

AUXILARY MEMORY SYSTEM
FINAL REPORT ON PHASE I

by

D.J. Bowler

APRIL 1968

MIT INSTRUMENTATION LABORATORY
CAMBRIDGE 39, MASSACHUSETTS

TABLE OF CONTENTS

SECTION 1

INTRODUCTION

The purpose of this report is to summarize the eight-man-month study on contract NAS 9-4065 (55-38100) Task 6 Phase I. This report completes the requirements of the task.

This report has been divided into three main sections: Summary of Study to Date, Decision Areas, and Recommendations for Future Work.

SECTION 2

SUMMARY OF THE STUDY TO 8 FEBRUARY 1968

The major effort to date is reported in Appendix A. This section will try to summarize that appendix to tie the AGC and the Auxiliary Memory (AM) into the Block 2-1/2 entity that they are. The Block 2-1/2 extension is appropriate because the AM has access to the bus structure of the AGC and its proposed addressing structure is an augmentation of the bank concept already in use in the AGC.

This combination can make the AGC into a more advanced and flexible processor. We shall demonstrate this flexibility in six steps. In fact, the major decisions to be made will be to decide how much of this flexibility one does want. At the conclusion of this section is a list of contributions or results of this study.

2.1 Multi-Job Multi-Task Capability

The software associated with the present AGC (job stack, VAC areas, phase table, waitlist, verb-noun structure, etc) is already quite flexible in that it is set up for multi-job, multi-task restartable operation and astronaut-computer communication (Fig. 1). The system is restartable in the sense that, if there were a primary power interruption, the AGC would gracefully shut down and then come back on without disturbing its memory and would initialize by using the phase table so that active jobs in progress would be reactivated at a predetermined point in the program. The verb-noun structure also allows the astronaut by means of the display to communicate with the auxiliary memory, tape processor, interface processor, etc.

2.2 Address and Control Flexibility

This programming flexibility, plus the addressing capability shown in Fig. 2 and 4, make this Block 2-1/2 concept attractive. Figure 2 shows that the general AGC instruction word has only three bits for an Op code but for erasable operands one can utilize two more bits (quarter codes) and for Channel instructions one could address 512 locations.

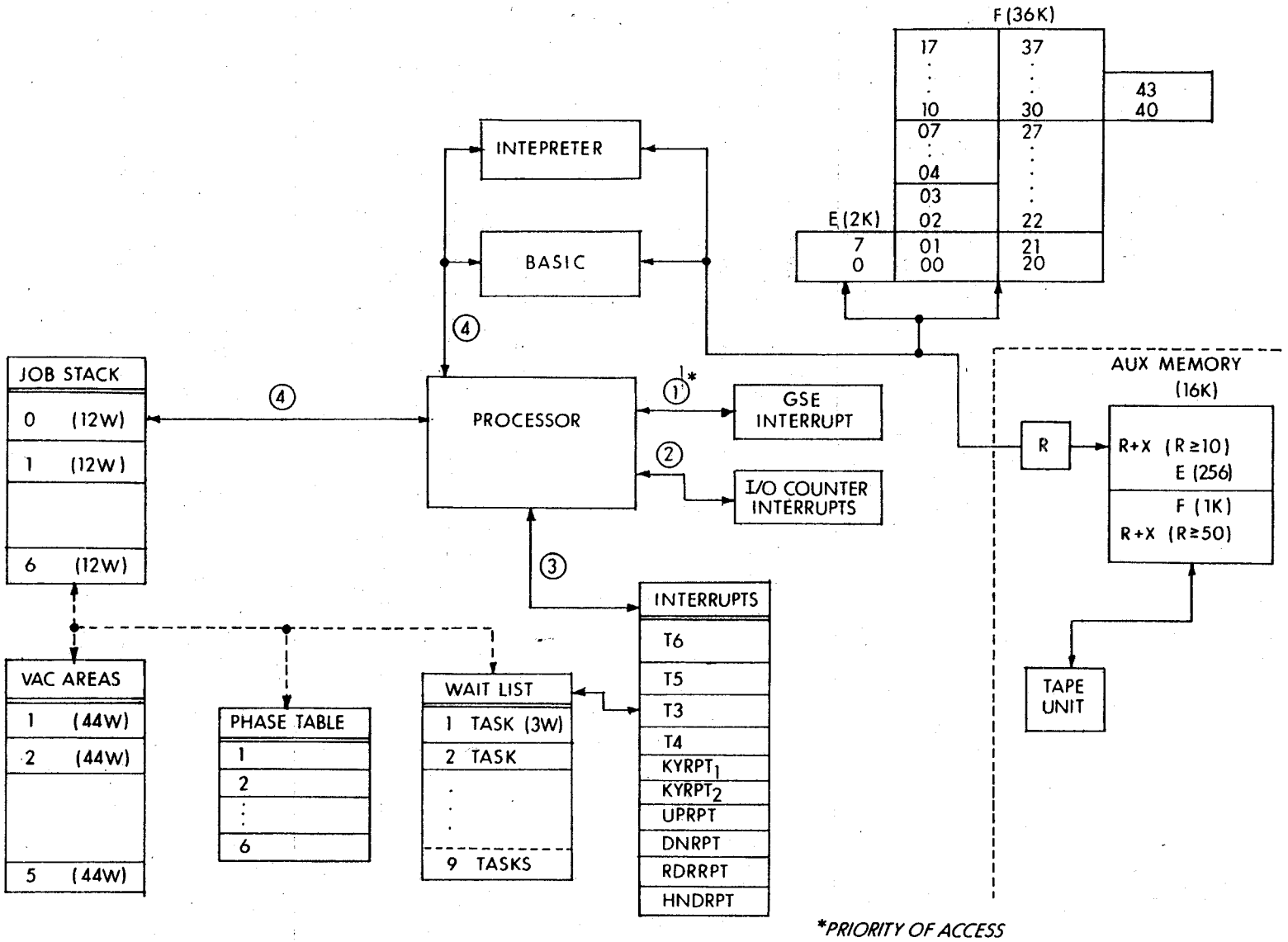


Fig. 1 AGC - Aux Memory.

AGC BASIC ADDRESSING

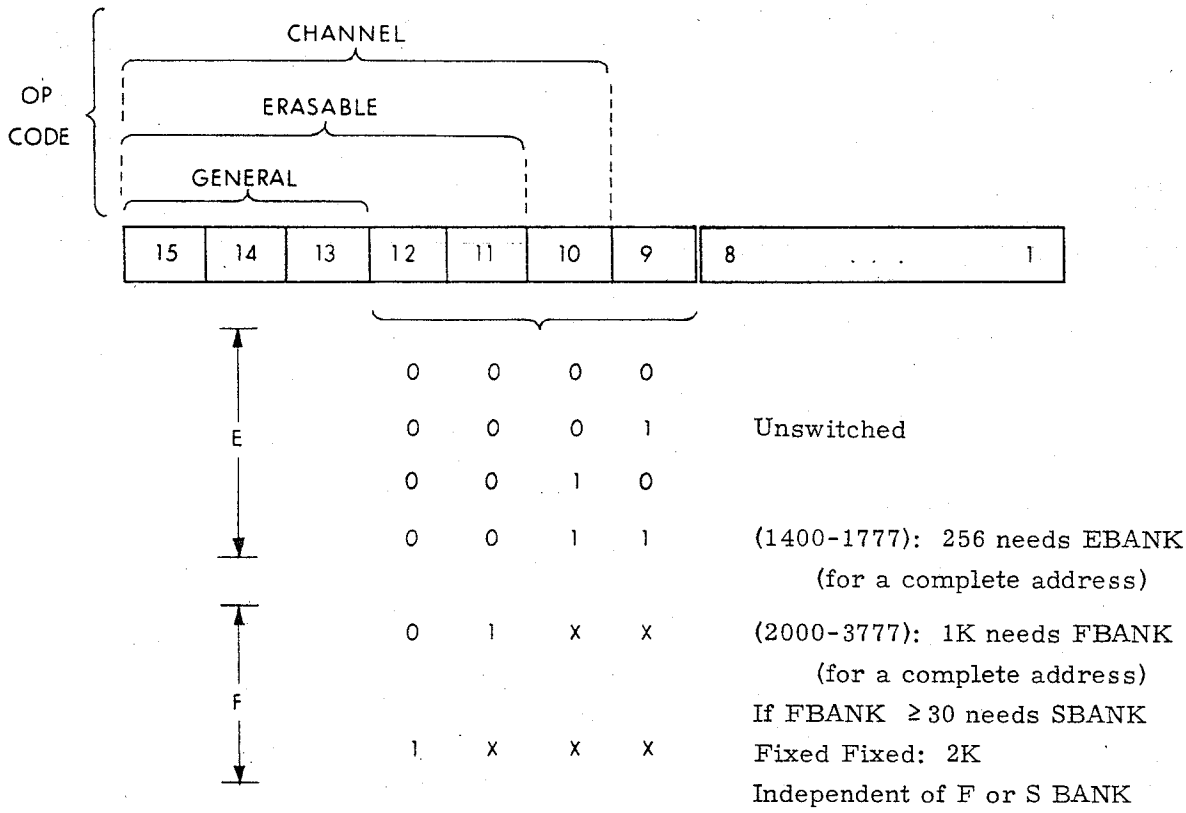
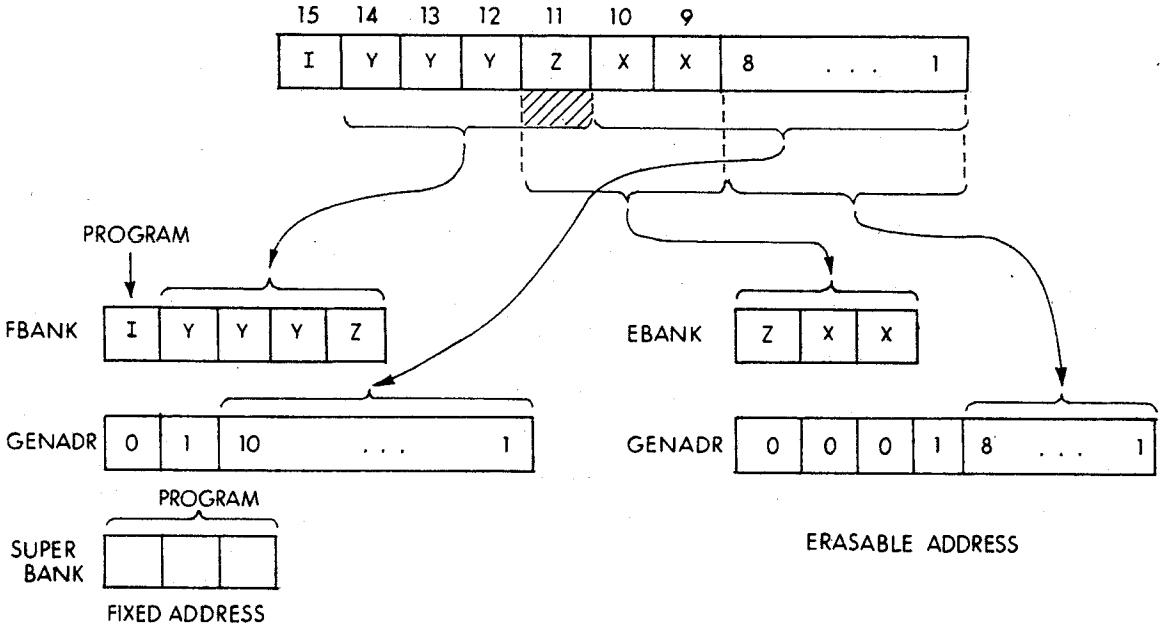


Fig. 2 AGC Basic Addressing



Because of I and Z the memory decomposes into the following:

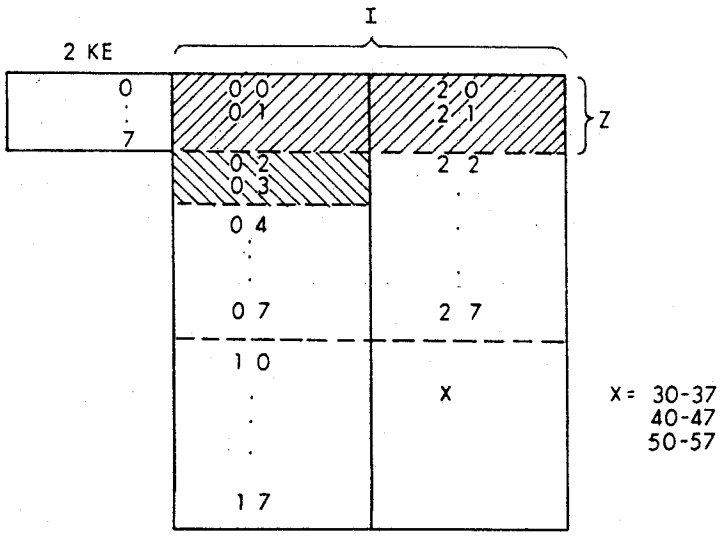
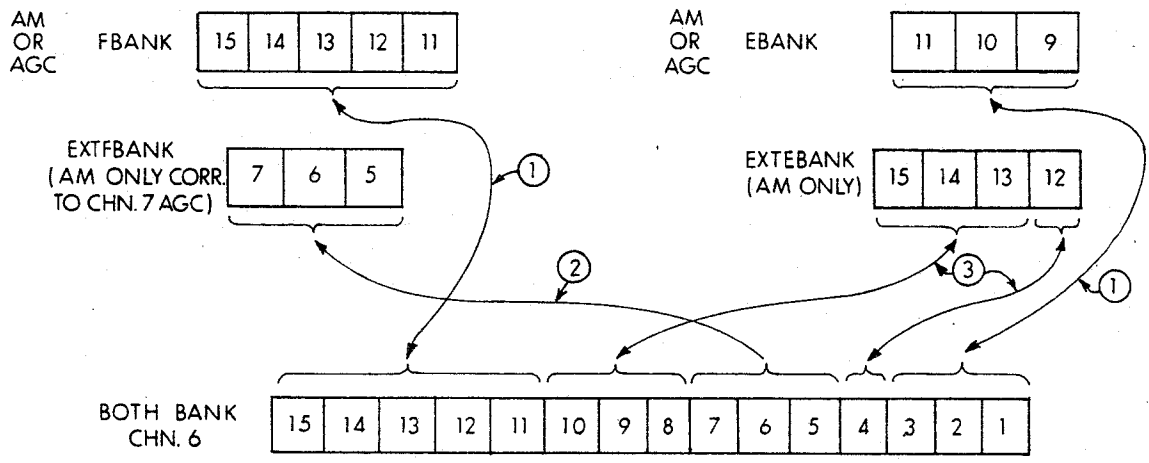


Fig. 3 Interpretive Addressing



NOTES:

① Paths already wired in AGC.

② Path recommended to be wired into AGC.

③ Bank that would exist in AM only.

Fig. 4 AGC - Aux. Memory Both Bank paths.

Presently, in the Block II AGC, there are only 21 channel addresses in use and, even though the AGC hardware limits one to 64 addresses, 43 addresses could be made available for control. Raytheon in its present hardware has only utilized 6 of the 43 channels to allow the AGC to control the AM and associated tape unit (turn off power, failure isolation, etc.). Thus, one sees that there is a great deal of control potential available.

The addressing structures in Fig. 4 show the present AGC bank structure as well as the auxiliary memory structure and indicate that the fixed addressing (F BANK + Super bank, channel 7) already can be extended by 16K (fixed banks $50\text{-}67_8$, 1024 words each) but that one would need to extend the E BANK by four bits so as to allow one to have 16K erasable ($010\text{-}107_8$ banks of 256 words each). This extension is accomplished by adding four bits in the auxiliary memory.

One of the prime results of this study was that, while adding the bank bits would allow one to utilize the 16K for instructions in basic, a relocation scheme was needed if one wanted to utilize all of the 16K for interpretive erasable (using the present interpreter) (Fig. 3).

Figure 4 also indicates that, if one wanted to conserve memory and reduce the number of instructions, it would be desirable to put the super bank setting in the Both Bank registers. It would save an EXTEND and a Write Channel 7 instruction. It only requires wiring changes to Tray A to write all three banks when one writes Both Bank (Channel 6) but, to be able to read all three with a read Both Bank, changes to modules in Tray A would be required.

2.3 Interface and Multiprocessor Capability (Modularity, Interleaving)

The AGC interface expansion capability is difficult to utilize because it requires getting into existing harnesses. The capability has been used for unmanned flights for the LM mission programmer and in the command module for the ranging unit.

The auxiliary memory would add to the present interface flexibility by providing a connector or connectors that are not presently committed to spacecraft harnesses. It would also permit additional spacecraft interface processing capability. The problem here is trying to anticipate the desired capability when future programs are in such a state of flux. There are two approaches in this situation.

1. To allow general processing capability controllable by the channel address structure just as is presently done in the control of the operation of the tape unit. That is, the channel words would contain the macroinstructions to the interface processor in the auxilary memory unit which would provide

the electronics, etc, to massage the interfaces with a minimum of AGC program involvement. This approach effectively increases the speed of the present AGC as well as its interface capability. It would require an interface adapter unit.

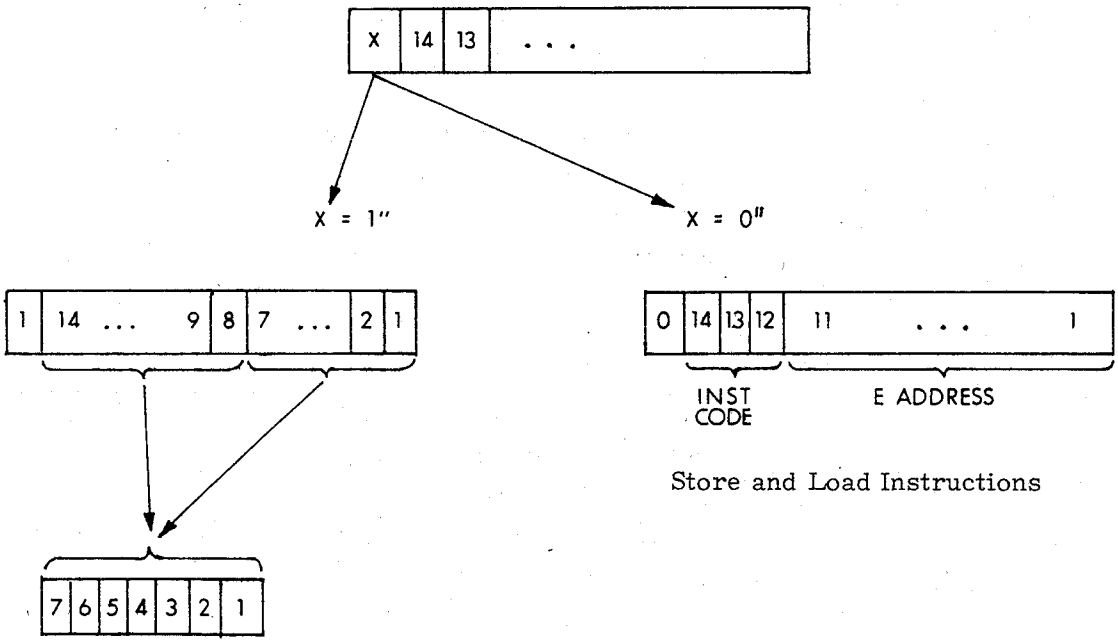
2. To design the auxiliary memory as a modular structure to which various functions could be added; capability to drive tape unit (tape processor), an interface processor, with or without specific interfaces (data adapter or not). These processors could be interleaved with the AGC processor in accessing the auxiliary memory's 16K erasable memory, again increasing the AGC's rate of through-put or processing speed. This approach also has the attractive feature of allowing one to study the multiprocessor concept in an evolutionary manner.

2.4 Relocation

Looking at Fig. 1 again, one sees that the AGC's processor has effectively two modes through which to utilize the memory - the basic and interpretive modes. As mentioned above, Appendix A studied the problem of making the 16K in the auxil-memory available to the present interpreter as erasable. The actual problem is made somewhat clearer if one looks at Fig. 3 and 5. The interpretive store and load instructions only have 11 bits of address from which the interpreter has to generate the E-bank setting and the Genadr (1400 - 1777 part of the address). Since one now has 16K more of erasable, one needs some mechanism for augmenting the 11-bit address.

Two schemes have been proposed; the software relocation approach (Craig Schulenberg) and the hardware relocation approach (Hugh Blair-Smith). Both approaches confine any segment of program to 2K erasable, 1K of which is always AGC E-BANK 0-3 in which the executive, input and output counters, VAC areas, phase table, waitlist, etc., are located, while the other 1K is variable. One can jump from one set of 1K to another by making use of the E-code number and the SEC instruction (Table 1) in the software scheme, or the relocation number (Table 2) in hardware scheme. Both these approaches are described in more detail in Appendix A.

It is felt that the hardware scheme does give one more flexibility as well as increased speed. It should also be noted that the bank structure of the AGC, plus the desirability of expanding the interpreter erasable addressing capability, led one in a natural manner to the relocation concept presently being designed into many modern computers for multiprogramming use. That is, with the present bank structures, the AGC had a relocation capability already built-in but that this capability was treated as a constraint on the programmer rather than a flexibility. One wonders if, had it



NOTE: *Next word bit 15.

Fig. 5 Interpreter instructions.

TABLE 1

FIXED AND ERASABLE ADDRESSING

AM ADDRESS	SUPER BANK ADDRESS	EXT E BANK ADDRESS	ASSOCIATED E CODE NO.*
		00-03 (AGC)	00
		04-07 (AGC)	00
00000-01777	50	10-13	04
02000-03777	51	14-17	10
04000-05777	52	20-23	14
06000-07777	53	24-27	20
10000-11777	54	30-33	24
12000-13777	55	34-37	30
14000-15777	56	40-43	34
16000-17777	57	44-47	40
20000-21777	60	50-53	44
22000-23777	61	54-57	50
24000-25777	62	60-63	54
26000-27777	63	64-67	60
30000-31777	64	70-73	64
32000-33777	65	74-77	70
34000-35777	66	100-103	74
36000-37777	67	104-107	100

*These numbers start in bit 9. The E code number added to bits 9-11 of the ECADR should give the correct extended EBANK address for interpretive instructions.

TABLE 2

BANK SELECTION FOR VARIOUS BANK-REGISTER CONTENTS

CHAN. 07 (BITS 7-5)	F-BANK (BITS 15-11)	SELECTED FIXED BANK	
X X X (X = Don't Care)	$\underbrace{F_{15}\ F_{14}}_{\neq 11}\ F_{13}\ F_{12}\ F_{11}$	$0\ F_{15}\ F_{14}\ F_{13}\ F_{12}\ F_{11}$	00-27
0 X X	$1\ \ 1\ \ F_{13}\ F_{12}\ F_{11}$	$0\ 1\ \ 1\ \ F_{13}\ F_{12}\ F_{11}$	30-37
1 0 0	$1\ \ 1\ \ 0\ \ F_{12}\ F_{11}$	$1\ 0\ \ 0\ \ 0\ \ F_{12}\ F_{11}$	40-43
$\underbrace{1\ SB_6\ SB_5}_{SB_6 \neq SB_5}$	$1\ \ 1\ \ F_{13}\ F_{12}\ F_{11}$	$1\ SB_6\ SB_5\ F_{13}\ F_{12}\ F_{11}$ + $R_{14}\ \ R_{13}\ R_{12}\ R_{11}$ = $RF_{14}\ RF_{13}\ RF_{12}\ RF_{11}$	50-67 [Relocated To ACM 00-17]

EMODE (BBANK BIT 10)	EXT. EBANK (BBANK BITS 9, 8, 4)	EBANK (BITS 3-1)	SELECTED ERASABLE BANK	
X	0 0 0	$E_3\ E_2\ E_1$	$0\ 0\ 0\ E_3\ E_2\ E_1$	00-07
0	$\underbrace{E_6\ E_5\ E_4}_{\neq 000}$	$E_3\ E_2\ E_1$	$E_6\ E_5\ E_4\ E_3\ E_2\ E_1$ + $R_{14}\ R_{13}\ R_{12}\ R_{11}\ 0\ 0$ - $0\ 0\ 1\ 0\ 0\ 0$ = $RE_6\ RE_5\ RE_4\ RE_3\ E_2\ E_1$	10-77 [Relocated To ACM 00-77]
1	$\underbrace{E_6\ E_5\ E_4}_{\neq 000}$	$0\ E_2\ E_3$	$0\ 0\ 0\ 0\ E_2\ E_1$	00-03
1	$\underbrace{E_6\ E_5\ E_4}_{\neq 000}$	$1\ E_2\ E_1$	$E_6\ E_5\ E_4\ 1\ E_2\ E_1$ + $R_{14}\ R_{13}\ R_{12}\ R_{11}\ 0\ 0$ - $0\ 0\ 1\ 0\ 0\ 0$ = $RE_6\ RE_5\ RE_4\ RE_3\ E_2\ E_1$	N4-N7 N=1(1)7 [Relocated To ACM 04-77]

been looked at in this light rather than as an akward way of increasing memory addressing capability, would present programs have been written as they now are? (A study of this type is proposed in Sec. 6.1.1). As can be seen from Fig. 2, any programs written for the AGC, once the banks are set, can address 256 words of erasable, corresponding to the EBANK setting, plus 3×256 additional words of erasable, the unswitched erasable, and 1K of fixed programming determined by the F and Super bank setting plus $2 \times 1K$ of the fixed memory. This is a total of 1K erasable and 3K of fixed.

This program segmentation and relocation capability has been discussed in Appendix A by Mr. Blair-Smith and the general aspect is discussed further here. For a present Apollo flight the entire program is assembled and, if any changes are made, the entire program is reassembled. That is, memory addresses are assigned to each instruction and to each item of data for storage. When the resulting object program is loaded into the computer (wired into the ropes, etc.), the program is placed in these assigned addresses.

It is often desirable to be able to load an object program into a different set of locations without recourse to reassembly, that is, to move the program up or down in memory, leaving it unchanged otherwise. This could happen if:

1. there were changes in the storage requirements of assembled programs that were in storage at the same time;

2. one were to have a varying set of jobs with varying storage requirements one would need the ability to relocate the jobs. This could be true in future Apollo applications where they would be all kinds of experiments, etc, that might want to be pre-processed or different autopilots for various spacecraft configurations; and

3. one might want to be able to borrow a completed program written by some other programmer and use it in combination with other program segments.

The time it takes to reassemble a complete flight program with subroutine changes like COLOSSUS on a 360 is on the order of an hour. One can see the possible cost savings as well as computer time saved by having such a capability.

This flexibility is availabe if one has the ability to relocate. Appendix A, discusses this in somewhat more detail.

In the way of an example, the astronaut could call up Program 76. The executive would search the program list to see if this program were already in the auxiliary memory and, if it were not, it would fetch it.

When the program is located on tape, the first word or two are read off, telling the executive how much memory it requires. It would then be necessary for the executive to find or make available sufficient space in the auxiliary memory. The tape control would then read the data into the auxiliary memory and the executive for the hardware scheme would store the job number, relocation number, and amount of storage or, for the software scheme, would store the job number, E code number, and the amount of storage.

2.5 Additional Functions

These following items have been considered as desirable.

1. Dispatcher

A study was made to see how one could increase the speed of the AGC. One possibility considered was to shorten the memory cycle time of the AGC. This was rejected because of the hardware impact on Tray A as well as on Tray B, and the additional impact on GSE equipment interfacing with the AGC.

Another possibility considered was to put the dispatcher part of the interpreter into hardware. If one considers Fig. 5, one sees that there are two phases associated with each interpretive instruction. First, one has to decide what kind of an instruction one has (dispatcher), then one has to execute the instruction. By putting the dispatcher into hardware, it was felt that the speed of the interpretive programs could be increased. This could also result in reducing the memory storage requirements as programs have been written in basic rather than interpretive because the interpretive program is too slow.

At present an investigation is being performed to see how much time would be saved and how much hardware it would take. A preliminary printout defining the required control pulses is available.

2. Critical-Mode Monitor

During various phases of the Apollo program, proposals have been made for monitoring the operation of the AGC during critical phases.

It could be possible to put the Auxiliary Memory in a monitor mode so that during critical phases it could set aside part of its memory to be compared with the AGC.

That is, if any particular address were read and the AGC and monitor disagreed, then, if there were no parity errors or there were one in the auxiliary memory, the auxiliary memory mode would be discontinued and one would rely on the AGC. If the AGC indicated a parity error in the AGC memory, the parity error would cause a restart during which time the AGC would be inhibited and the auxiliary memory utilized. Additional study in this area could possibly result in expanded use of this mode.

3. Restart Monitor

This function can be performed by a module that plugs onto the AGC test connector. Basically, it stores the various types of alarms in channel 77 that cause a restart and is used after a restart to indicate which alarm caused the restart.

4. Restart Capability

As is presently envisioned, the auxiliary memory would operate just as the AGC would. If a restart were generated, coding would be incorporated into channel 77 to localize it to the auxiliary memory and the programs would restart just as in the AGC. It is conceivable that the phase table might indicate that, if the program were operating out of the auxiliary memory, a sum check could be made of that part of the auxiliary memory considered fixed and, if it were found to be disturbed, the programs could be read from tape again.

A series of restarts attributable to the auxiliary memory would result in it shutting itself down until turned on again by either the ground or the astronaut.

2.6 Programming Impact

The complexity of the Apollo mission is reflected somewhat in the programs in the AGC. The job structure, associated mission analysis, and programming are all interwoven.

If one were to utilize the auxiliary memory, one would still have this complex situation to contend with but the ability to utilize the relocation ideas plus the additional storage capability, both bulk and erasable, does appear to give one additional degrees of freedom.

The addressing techniques used in the auxiliary memory are merely an extension of the bank concept with which AGC programmers are quite familiar. The relocation number is really not much different from the bank number and is treated in a similar manner.

The fact that the actual utilization of the auxiliary memory is intimately related to the program structure, the fact that the unit has direct access to the AGC memory bus, the fact that the changes recommended for the AGC itself are minor and the increased capacity provided by the AGC auxiliary memory combination, altogether motivate calling this a Block 2-1/2 system.

2.7 Advantages of the Auxiliary Memory

1. Preserves the physical integrity of the AGC.

2. Minimizes impact on spacecraft interfaces and procedures (mechanical and electrical).

3. Allows additional flexibility with regard to future missions and subsystem or spacecraft checkout.

4. Utilizes to the utmost the equipment that is already there, reducing cost-schedule impact.

5. Offers wider choice between alterable and nonalterable memory storage.

6. Allows separate procurement.

SECTION 3

SPECIFIC CONTRIBUTIONS OF THIS STUDY

1. Use of relocation as solving the interpreter problem.

2. Segmentation of programs utilizing resident system.

3. Clarification of the flexibility inherent in the AGC auxiliary memory approach.

4. Wiring changes to allow one to write channel 7 when writing BOTH BANK (channel 6).

5. Desirability of having auxiliary memory work with a nonflight-qualified tape unit as well as a flight-qualified tape unit.

6. Interleave capability of AGC, tape unit, and interface processor,

7. Inclusion of restart monitor, critical mode monitor, and hardware dispatcher to auxiliary memory.

8. Modular expansion of auxiliary memory vs integral unit.

9. Use of a simple flight harness between AGC and auxiliary memory.

SECTION 4

USE OF THE RAYTHEON-BUILT ACM PROTOTYPE

This equipment working with an AGC has been successfully demonstrated to both NASA and MIT/IL during the weeks of 17 January and 24 January 1968.

The next logical step would be to tie it in with a system and system programs. There are no conceivable reasons why such a test configuration would not work. The primary objectives in running such a test would be to eliminate any unanticipated problems that might be present and to gain experience working with the equipment and the associated programming.

Unless the equipment were to run with a PAC, one would need to build another harness and, in order to utilize the interpreter, one would have to use the software relocation scheme for the interpretive programming.

The programming effort would be quite extensive if one wanted to use the systems programs in the auxiliary memory in anything but a simple manner. A possible series of system tests would be:

1. Simple programs running out of the auxiliary memory.
2. Programs running out of the auxiliary memory would be used only as extended basic.
3. Programs being used as both extended basic and interpretive fixed.
4. Interpretive programs, fixed and erasable.

The last type of programming would require changes to the executive, etc., to allow software-type relocation program segmentation and, depending on how one wanted to, could be restart-protected, etc. Similarily, for items 2-4 we would have to modify the simulator to recognize the additional E-BANKS. This would not be a serious modification.

SECTION 5

DECISION AREAS

Decisions have to be made before one can proceed to define a procurement specification. The decisions are of three types (a) those necessary to continue the study, (b) those defining how much flexibility one wants, and (c) those having a reliability impact.

1. Addition of a wire in the AGC (Tray A) from the test connector to the write G/ gate. This wire is not necessary but it is the simplest, most-direct solution to the problem of being able to transfer the data from the auxiliary memory to the AGC G register.

2. Whether there shall be one or two harnesses used between the AGC and the auxiliary memory. Because of the required flight reliability and minimum amount of space, it is felt that there should be two harnesses, one for flight and one for test stations where one would be using core-rope simulator, computer test set, or PAC. This would allow the flight cable to have one less connector (144-pin) and a solid-state line-driver/line-receiver type of interface. This would result in a simple, clean, and, therefore, most-reliable interface.

3. Software or hardware type of relocation. This is discussed in the body of the report.

4. Adding nine wires to Tray A to allow one to write channel 7 when one writes BOTH BANK (Channel 6).

5. Expand tape driver to utilize both a nonflight-qualified IBM compatible tape unit as well as a flight-qualified tape unit. The amount of actual space-craft operating time of any hardware like the auxiliary memory, AGC, etc., is usually an insignificant (time-wise) portion of the actual operating life of the equipment. Realizing this, and that there is a finite life-time associated with the flight-tape unit, it is felt that, by having the auxiliary memory electronics able to drive a nonflight-qualified unit, one would maximize the flight reliability of the flight unit and yet utilize the full capability and use-fulness of the auxiliary AGC - Block 2-1/2 combination. One may need only to modify the tape unit itself.

6. Inclusion of restart monitors.

7. Inclusion of critical-mode monitor.

8. Addition of hardware dispatcher. This item is still under evaluation and a first look indicates it could require a great deal of associated electronics. This study should be continued until a decision can be made.

9. A modular or integrated approach to packaging.

SECTION 6

FUTURE AUXILIARY MEMORY WORK

6.1 Logic Improvements

1. Analyze a typical lunar mission with regard to program segmentation, utilizing relocation concept.
2. Delineate changes to mission software, executive, pinball, etc.
3. Delineate changes to associated software, YUL, simulation, etc.
4. Continue investigation of restart capability, reliability impact, and EMI objectives.
5. Define interface processor structure (tape, expanded interface).
6. Define hardware necessary for loading and checking ATM.
7. Develop interleave capability of auxiliary memory with AGC, tape, and interface processor.
8. Continue study of hardware dispatcher.
9. Define hardware relocation electronics.
10. Develop program manual.
11. Define software for flight test.
12. Define interconnect cables between AGC and auxiliary memory.
13. Develop purchase specification.
14. Study modular design of auxiliary memory (provide capability for expansion).

6.2 Raytheon Equipment

1. Unmodified
   a. Build harness to operate with core-rope simulator.
   b. Study interface signals.
   c. Mate with a G/N system to study system and program compatibility.

2. Modify

a. Build harness to operate with core-rope simulator.

b. Modify electronics to have hardware relocation capability.

c. Adapt a faster memory to permit interleaving with the AGC and tape unit.

d. Modify tape electronics to run with either flight-tape unit or a commercial unit.

e. Mate with G/N system to study system and program compatibility.

APPENDIX A

Transcript of Apollo Project Memo No. 1818

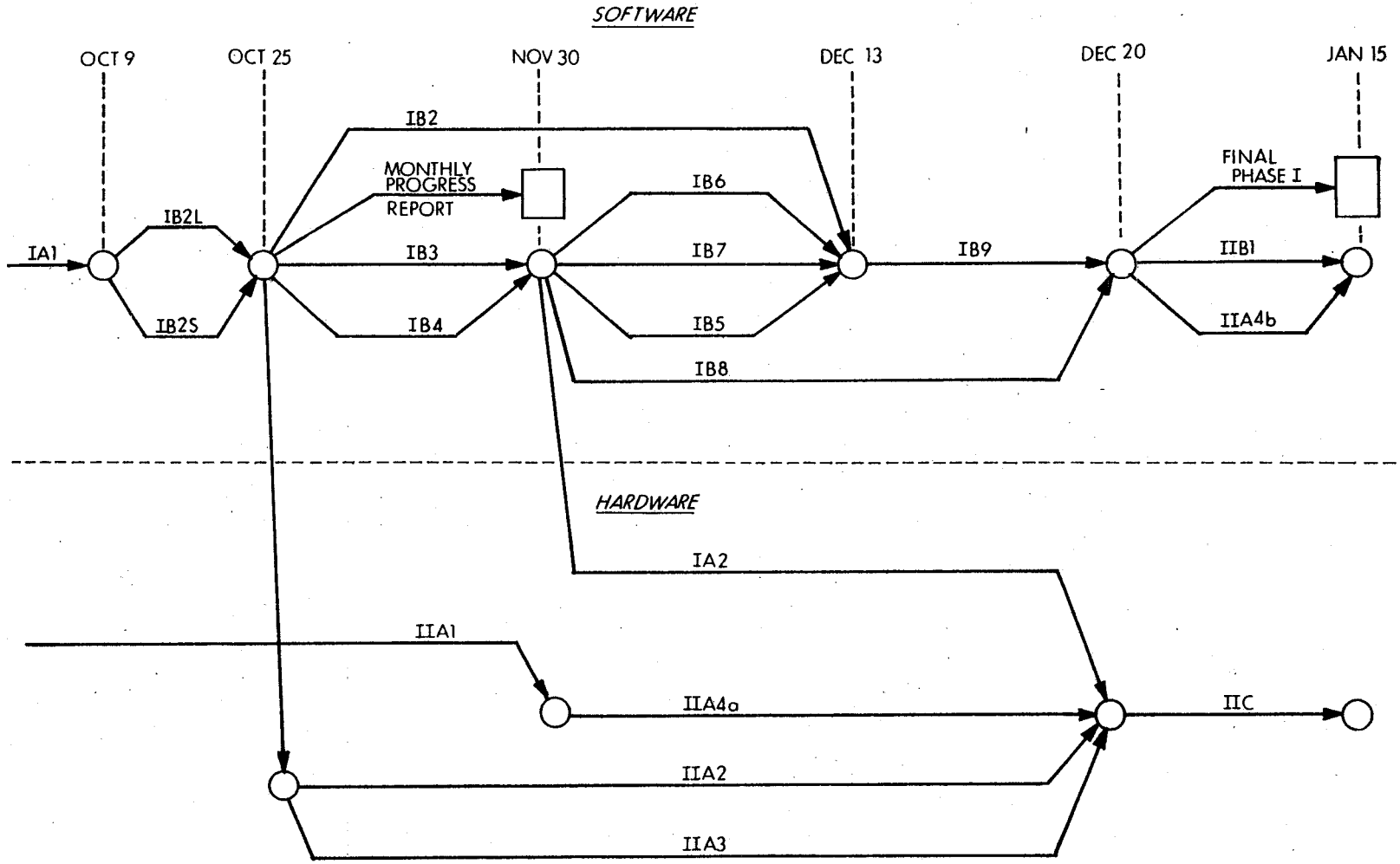
Introduction

The purpose of this report is to summarize work done to date on the Auxiliary Memory. A schedule and list of objectives are contained on pages 32 and 33. Because of main-line commitments of many of the programmers, this program has been carried on at a fairly low level of effort.

The following people have contributed to this study.

H. Blair-Smith	relocation approach
C. Schulenberg	software approach
C. Muntz	interpreter
E.C. Hall	
D.J. Bowler	
T. Lawton	Consultant
G. Cox	
F. Gauntt	
R. Crisp	
A. Laats	

This report covers the period June 12th to November 20, 1967 for Contract NAS 9-4065, Task Order #31, Task Statement #6.


SOFTWARE
OCT 9
OCT 25
NOV 30
DEC 13
DEC 20
JAN 15
IB2
MONTHLY PROGRESS REPORT
FINAL PHASE I
IB6
IB2L
IA1
IB3
IB7
IB9
IIB1
IB2S
IB4
IB5
IIA4b
IB8
HARDWARE
IA2
IIA1
IIA4a
IIC
IIA2
IIA3

AM Schedule.

AUXILIARY MEMORY SYSTEM

TASK I

I. Phase I Tasks

A. Hardware

1. Reliability impact of AGC/AM system integration.
2. Hardware for loading and checking the ATM.

B. Software

1. AM system hardware and software design constraints.
2. Analyze typical lunar mission with regard to memory allocation.
3. Changes to Executive, Pinball, etc.
4. Identify changes to YUL and SIMULATION.
5. Investigation program restart capability.
6. Identify potential applications.
7. Identify crew operational requirements.
8. Define software for inflight test.
9. Develop preliminary program manual.

TASK II

Phase II Tasks

A. Hardware

1. AGC/AM electrical interface compatibility and functional characteristics.
2. AGC/AM mechanical interface compatibility.
3. Investigation EMI, power, expanded I/O capability.
4. Fabricate support test equipment.
   a) Interface cables.
   b) Tapes

B. Software

1. Modify existing Apollo G/N computer program.

C. Generate Specification

COST

Summary

The main efforts to date have been directed toward utilizing the additional storage capability available in the AM in such a way so as to be compatible with present Apollo programming efforts.

In particular, the present interpreter has been studied to see what restrictions would be placed on the AM. The main restriction that the interpreter presents is that it can only address 2K of erasable. The problem of how to expand this capability to the 16K in the Auxiliary Memory has been studied and two approaches (models) have been investigated. In fact, this reporting period has been spent mainly on this problem.

The two models are the overlay model and the extended-bit model; both are described in parts 2 and 3. With regard to the extended-bit model, two approaches, called the software and hardware approaches, have been analyzed.

The results contained in this report are a fairly detailed description of these two approaches to the extended-bit model. It is planned to take these approaches and to analyze their impacts on the rest of the programs. The actual segmentation of a mission into its separate parts is also planned and the program COLOSSUS is being used as a reference. Application to possible AAP missions is also being kept in mind but, because of the lack of definitive material, this is being done only in a general sense.

Specifically, the following items from the phase tasks are covered here - reference to pages 32 and 33.

IA1. D.D. Memo #381 (Appendix B) essentially describes the effort put forth in this area.

IB2. Two hardware models (overlay and extended-bit) have been proposed and examined with regard to their impacts on the interpreter and their compatibility with regard to a command module lunar mission program COLOSSUS. Another software model, which could require a new interpreter, allowing one to program in a language like Fortran, or Mac, was eliminated from the present study because of its impact on the present programming investment.

The present results indicate that the extended-bit approach would be the better model.

IB3 & IB4 Will use the extended-bit model as a basis for their studies and are being studied concurrently.

The rest of the report will be divided into four parts so as to describe the results to date. Part A-1 will be on the interpreter, Part A-2 on the overlay model, Part A-3 on the extended-bit model, the software, and the hardware approaches, and Part A-4 on COLOSSUS.

The hardware tasks shown below the dotted line on the AM schedule have, for the most part, not been performed. The purpose of showing them is to indicate when these tasks should have logically been phased into the program.

A.1 Interpreter

The purpose of this part is to indicate the limitations that the present interpreter imposes on the AM. The two models discussed later do get around these limitations.

More detailed discussions of the interpreter can be found in Reference 1-3. As far as this report is concerned, the interpreter has 3 classes of instruction addresses.

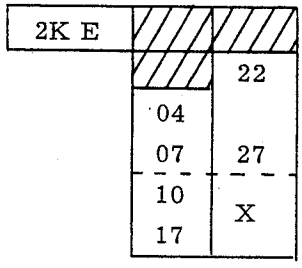
1. Erasable-store - characterized by 11 bits.
2. Branching - characterized by 15 bits.
3. All others - characterized by 14 bits.

Since the AM could have 16K of read, write, core memory, it would be desirable that some of this could be used to augment the AGC 2K erasable. To do so and not seriously affect the interpreter, one has to look at the constraints the interpreter imposes.

The interpreter store instructions are made out of 15 bits, the 15th bit indicating it is a store instruction, bits 14-12 indicating the type of store instruction, and bits 11-1 the address. Thus, unless one reduces the number of instructions from 8 to 4, one can't increase the number of E-Memory addresses.

The fact that most other operands have only 14 bits for the address, limits one to banks 02 to 17 or 22 to 37, called low and high memory, respectively. Because erasable addresses take up 11 bits, one loses the first two banks of each group corresponding to bit 11. The fact that bit 15 of the FBANK is saved on entering the interpreter means one has to stay within the high or low memory. Also, since the superbanks are not changed, one has to stay out of banks 2 and 3 of FIXED FIXED.

In summary, the interpreter decomposes the AGC memory into sections shown below.



When X = 30 - 37
40 - 47
50 - 57 etc.

A.2 Overlay Model

The essence of this model is that it would work in a manner similar to that of the present core-rope simulator. That is, one would assign the 16 (1K) memory banks of the AM to 16K of the AGC in such a way that, if one addressed one of the words in these banks, the information would come from the AGC or the AM, depending as to whether a particular bit in a channel were set or not set.

By having the ability to read from tape to any of the AM banks, one could effectively put any program into any of the 16 banks of the AM and use the data as if it were within the address field of the AGC, either erasable or rope.

The interpreter store instructions would not need to be changed, since at any one time there would be a unique address for it, as the AM channel bit would determine whether the word was in the AM or AGC.

The main reasons that this model is felt to be poorer are:

1. For Apollo-type missions there are too many routines in banks 00-27 that are used too often to be overlaid.

2. The problem of interweaving these various programs for either Apollo and AAP missions looks quite complex.

3. If the total storage is limited, the mission sequencing from tape storage may be more difficult. That is, the case of the overlay scheme would not add to the amount of memory available and would, in fact, decrease it since a much more sophisticated executive program would be required in order to locate the various program segments from tape into the available memory banks.

At best, it could make 16K of the 36K AGC's fixed appear to be erasable in the sense that the same addresses could contain different programs but it wouldn't be erasable in the sense that they could be modified by the AGC.

The main advantages the overlay technique would have are:

1. Would require minimum change to the present interpreter.

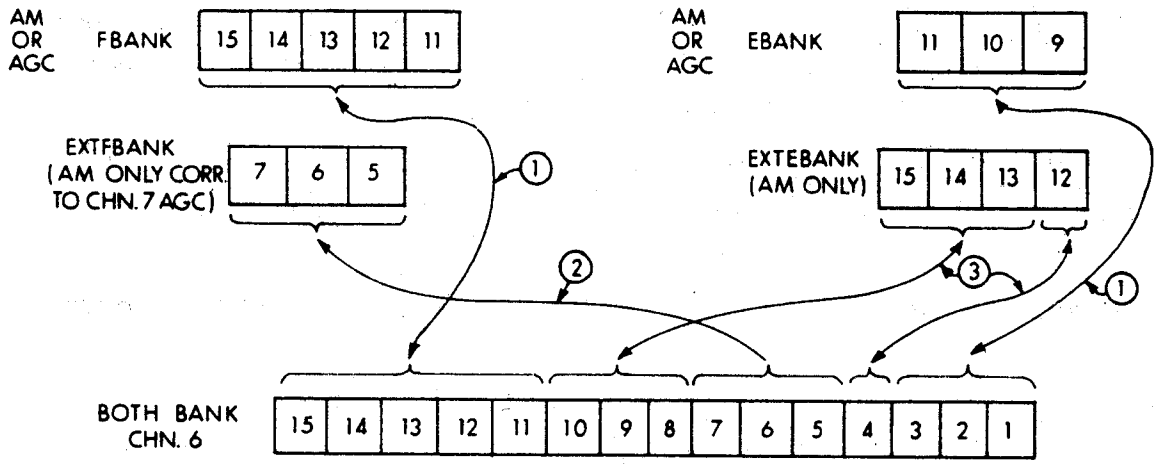
2. If there were a bug observed in a program in a rope, the program could be overlaid.

A.3 Extended-Bit Model

This model is the same as that being used in the RFQ responded to earilier this year by Raytheon and IBM. Essentially, one adds erasable addressing capability by adding bits to the EBANK register. Since this is impossible to do in the AGC without physically changing it, these EBANK bits are provided in the AM, along with the associated memory locations.

Similarly, the AM can provide more memory for fixed addresses, since the addressing capability already exists in the super bank register (channel 7) of the AGC.

Thus, in this model, one has the address banks shown in Fig. A. 1.



NOTES:

① Paths already wired in AGC.

② Path recommended to be wired into AGC.

③ Bank that would exist in AM only.

Fig. A-1 AGC - Aux. Memory Both Bank Paths.

Essentially, all the bank-memory capability can be handled by one 15-bit word. If one reads or writes into the BBANK of the AM, then one essentially modifies all the AM memory register banks shown in Fig. A. 1.

The AM will be setup so that, if one reads or writes the following AGC memory address banks, the Auxiliary Memory's corresponding register will change.

AGC	AM
FBANK	AMFBANK
EBANK	AMEBANK and AMEXTEBANK
Super Bank (Chn. 7)	AMEXTFBANK
BBANK	AMBBANK ∴ all the banks above.

On the read cycle, the AM would not have to put outputs on the lines for which the AGC would have information but could check what the AGC puts out to see if it agrees with what the AM has.

By adding a few wires the AGC could be changed so that, when one writes BBANK (address 0006), channel 7 gets written into also. This could save some instructions in the present program and is a desirable change.

The next two sections discuss the two extended-memory approaches that are presently being studies. Both allow the AM Memory to be used as either expanded-erasable or fixed in the interpreter mode.

The advantage of the extended-bit concept over the overlay is that it effectively increases the amount of program a available at any one time and minimizes the time to access it. One can address 2K - 18K erasable and/or 36K - 52K of fixed in basic or interpretive without having to reload from tape. This allows a great deal of flexibility.

A.3.1 Software Solution to the Interpreter Problem

1. Basic Programming

Table A.1 shows the addressing scheme for utilizing the Auxiliary Memory. Each memory location can be addressed as a fixed-address bank 50-67 or as an erasable-address bank 10-107.

The AM will normally inhibit any AGC memory cycle addressing fixed banks 50-67 or erasable banks 10-107.

There is a constraint on the programmers in that, if they address erasable banks 10, 20, 30, 40, 50, 60, 70, 100 words 0-7, the AGC will regard them as special and central addresses and the AM cannot inhibit the AGC special and central locations.

Each location in the auxilairy memory will have two addresses as shown in Table A.1.

TABLE A.1

FIXED AND ERASABLE ADDRESSING

AM ADDRESS	SUPER BANK ADDRESS	EXT E BANK ADDRESS	ASSOCIATED E CODE NO.*
		00-03 (AGC)	00
		04-07 (AGC)	00
00000-01777	50	10-13	04
02000-03777	51	14-17	10
04000-05777	52	20-23	14
06000-07777	53	24-27	20
10000-11777	54	30-33	24
12000-13777	55	34-37	30
14000-15777	56	40-43	34
16000-17777	57	44-47	40
20000-21777	60	50-53	44
22000-23777	61	54-57	50
24000-25777	62	60-63	54
26000-27777	63	64-67	60
30000-31777	64	70-73	64
32000-33777	65	74-77	70
34000-35777	66	100-103	74
36000-37777	67	104-107	100

*These numbers start in bit 9. The E code number added to bits 9-11 of the ECADR should give the correct extended EBANK address for interpretive instructions.

b. Interpretive Mode

Since one is restricted to an 11-bit address (2K) for erasable locations, all addresses 0000 to 1777 are assigned to banks 0-3 in the AGC, addresses 2000 to 3777 will be either in the AGC of the AM, depending on the extended EBANK address. (Refer to Table A. 1). The E code is the number that needs to be added to the 11-bit interpreter address to give the correct extended EBANK address.

There are several places in the present interpreter program that have to be modified so that, if the address is 2000-3777, this E-code number must be added in. The interpreter programs using these addresses will be slowed down by these additional operations.

An interpreter instruction SEC (Set E Code) is also needed to facilitate the programming. It essentially puts the correct number in the E-code address and is required any time one is going to refer to any other 1K of erasable. An idea of how to program with the SEC instruction is shown below.

```
TC INTPRET              Sets E Code = 00
DLOAD
                  X     X ϵ Bank 03
STORE             Y     Y ϵ Bank 07
SEC               DLOAD
                  27    Sets E Code = 20
                  Z     Z ϵ Bank 26
STORE             K     K ϵ Bank 24
DLOAD             DMP
                  B     B ϵ Bank 03
                  C     C ϵ Bank 25
PDDL              DMP
                  D     D ϵ Bank 02
                  E     E ϵ Bank 25
SEC
                  21    Set E Code = 14
STORE             L     L ϵ Bank 22
```

The advantage of the software solution to the interpreter problem is that it can be used with the hardware that Raytheon is presently making. It also requires less hardware than the hardware solution in the next section. Its disadvantages are that it would run slower and present a much more complicated executive program.

A.3.2.1 Hardware Solution to the Interpreter Problem

By Hugh Blair-Smith

This approach is essentially contained in Table A.2. The fixed addressing is the same as in the present AGC with the AM taking over super banks 50 to 67. However, the first word in AGC fixed bank 50 will correspond to AM address 00000 or the first word of AM bank 00.

As in Section A.3.1, each word in the AM can be addressed as a fixed location or as an erasable location. Tables A.1 and A.3 show the relationship between AM addresses, fixed, erasable, and pseudo addresses.

The only difference between basic and interpretive addressing is that a one is set on bit 10 of the extended AM BBANK when one does interpretive programming. The reason for this is so that the hardware would know which 2K of erasable the interpreter is refering to. That is, just as in the previous section, the first 1K always refers to AGC erasable banks 0-3 and the second 1K depends on the extended-bank address. If one looks at Table A-2, one notices that the programmer can only address 8K (32 EBANKS) in interpretive. ($E_3 = 1$, E_4, E_5, E_6 not all 0, E_1 E_2 arbitrary). However, due to the relocatability of the segments, 15K can be used. That is, it is possible to make use of a relocation register that would allow one to relocate program segments in steps of 1K. This register is described in Table A.2 and Fig. A.2.

The hardware approach will also make use of a SEC instruction to set the extended-EBANK bits (4, 5, 6) when one wants to use more than 2K of interpretive erasable. (Switch EBANKS in interpretive.) Nothing else in the interpreter will have to change. The advantage to this approach is that it lends itself to being assembled in a natural manner (this is discussed next) and it does not lengthen any of the interpretive instructions as does the software approach.

A.3.2.2 Assembly and Programming Considerations for Tape-Loaded Memory

In a wired-program computer, there is strict one-to-one correspondence between program and memory: a program can occupy only the memory allocated to it by the assembly process, and no other program can occupy any of that memory. When programs are executed from an erasable memory that is loadable in part or in whole from a backing store like a tape, neither statement is true; a given program may be loaded into different parts of the memory on different occasions, and its place in memory may at other times be occupied by a totally unrelated program. In any such system there must be a program that keeps its one-to-one correspondence so as to be available to all the other programs at all times (it must at the very least do the

TABLE A. 2

BANK SELECTION FOR VARIOUS BANK-REGISTER CONTENTS

CHAN. 07 (BITS 7-5)	F-BANK (BITS 15-11)	SELECTED FIXED BANK	
X X X (X = Don't Care)	$\underbrace{F_{15}\ F_{14}}_{\neq 11}\ F_{13}\ F_{12}\ F_{11}$	$0\ F_{15}\ F_{14}\ F_{13}\ F_{12}\ F_{11}$	00-27
0 X X	1 1 $F_{13}\ F_{12}\ F_{11}$	$0\ 1\ 1\ F_{13}\ F_{12}\ F_{11}$	30-37
1 0 0	1 1 0 $F_{12}\ F_{11}$	$1\ 0\ 0\ 0\ F_{12}\ F_{11}$	40-43
$\underbrace{1\ SB_6\ SB_5}_{SB_6 \neq SB_5}$	1 1 $F_{13}\ F_{12}\ F_{11}$	$1\ SB_6\ SB_5\ F_{13}\ F_{12}\ F_{11}$ $+\ R_{14}\ R_{13}\ R_{12}\ R_{11}$ $RF_{14}\ RF_{13}\ RF_{12}\ RF_{11}$	50-67 [Relocated To ACM 00-17]

EMODE (BBANK BIT 10)	EXT. EBANK (BBANK BITS 9, 8, 4)	EBANK (BITS 3-1)	SELECTED ERASABLE BANK	
X	0 0 0	$E_3\ E_2\ E_1$	$0\ 0\ 0\ E_3\ E_2\ E_1$	00-07
0	$\underbrace{E_6\ E_5\ E_4}_{\neq 000}$	$E_3\ E_2\ E_1$	$E_6\ E_5\ E_4\ E_3\ E_2\ E_1$ $+R_{14}\ R_{13}\ R_{12}\ R_{11}\ 0\ 0$ $-0\ 0\ 1\ 0\ 0\ 0$ $RE_6\ RE_5\ RE_4\ RE_3\ E_2\ E_1$	10-77 [Relocated To ACM 00-77]
1	$\underbrace{E_6\ E_5\ E_4}_{\neq 000}$	$0\ E_2\ E_3$	$0\ 0\ 0\ 0\ E_2\ E_1$	00-03
1	$\underbrace{E_6\ E_5\ E_4}_{\neq 000}$	$1\ E_2\ E_1$	$E_6\ E_5\ E_4\ 1\ E_2\ E_1$ $+R_{14}\ R_{13}\ R_{12}\ R_{11}\ 0\ 0$ $-0\ 0\ 1\ 0\ 0\ 0$ $RE_6\ RE_5\ RE_4\ RE_3\ E_2\ E_1$	N4-N7 N=1(1)7 [Relocated To ACM 04-77]

TABLE A.3

TABLE OF CORRESPONDENCE:

PSEUDO-ADDRESSES, ACM E-BANKS, AND ACM FBANKS

SUPER-BANK 5			SUPER-BANK 6		
PSEUDO-ADDRESS	E-BANK	F-BANK	PSEUDO-ADDRESS	E-BANK	F-BANK
130000-130377	10	50	150000-150377	50	60
130400-130777	11		150400-150777	51	
131000-131377	12		151000-151377	52	
131400-131777	13		151400-151777	53	
132000-132377	14	51	152000-152377	54	61
132400-132777	15		152400-152777	55	
133000-133377	16		153000-153377	56	
133400-133777	17		153400-153777	57	
134000-134377	20	52	154000-154377	60	62
134400-134777	21		154400-154777	61	
135000-135377	22		155000-155377	62	
135400-135777	23		155400-155777	63	
136000-136377	24	53	156000-156377	64	63
136400-136777	25		156400-156777	65	
137000-137377	26		157000-157377	66	
137400-137777	27		157400-157777	67	
140000-140377	30	54	160000-160377	70	64
140400-140777	31		160400-160777	71	
141000-141377	32		161000-161377	72	
141400-141777	33		161400-161777	73	
142000-142377	34	55	162000-162377	74	65
142400-142777	35		162400-162777	75	
143000-143377	36		163000-163377	76	
143400-143777	37		163400-163777	77	
144000-144377	40	56	164000-164377	100*	66
144400-144777	41		164400-164777	101*	
145000-145377	42		165000-165377	102*	
145400-145777	43		165400-165777	103*	
146000-146377	44	57	166000-166377	104*	67
146400-146777	45		166400-166777	105*	
147000-147377	46		167000-167377	106*	
147400-147777	47		167400-167777	107*	

*Not addressable. Reached through relocation only.

TABLE A.4

ACM FLIP-FLOP REGISTERS FOR ADDRESS SELECTION

FBC	Fixed-Bank Cycle	(= WL12/. WL11 . WS . T01)
EBC	Erasable-Bank Cycle	(= WL12/. WL11/. WL10 . WL09 . WS . T01)

DS10	DS09	DS08	DS07	DS06	DS05	DS04	DS03	DS02	DS01

Duplicate S Register

DSB07	DSB06	DSB05	Duplicate Super-Bank Register

ASBM	Super-Bank Mode (=WL15 . WL14 . WFB)

DF13	DF12	DF11	Duplicate FBANK Register

EMODE	Erasable Mode (AGC-ACM Split) (= WL10 . WEB)

EB6	EB5	EB4	Extended EBANK Register

R14	R13	R12	R11	Relocation Register (From WL04 - WL01?)

DE3	DE2	DE1	Duplicate EBANK Register

AGC ACCESS TO REGISTERS (NO AGC CHANGE)

0	0	0	0	E_3	E_2	E_1	0	0	0	0	0	0	0	EBANK	(0003)
F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	0	0	0	0	0	0	0	0	0	FBANK	(0004)
F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	E_M	E_6	E_5	0	0	0	E_4	E_3	E_2	E_1	BBANK	(0006)
0	0	0	0	0	0	0	0	SB_7	SB_6	SB_5	0	0	0	0	FEXT	(CH 07)
0	0	0	0	0	0	0	0	0	0	0	R_4	R_3	R_2	R_1	RELO-CATE	(CH??)

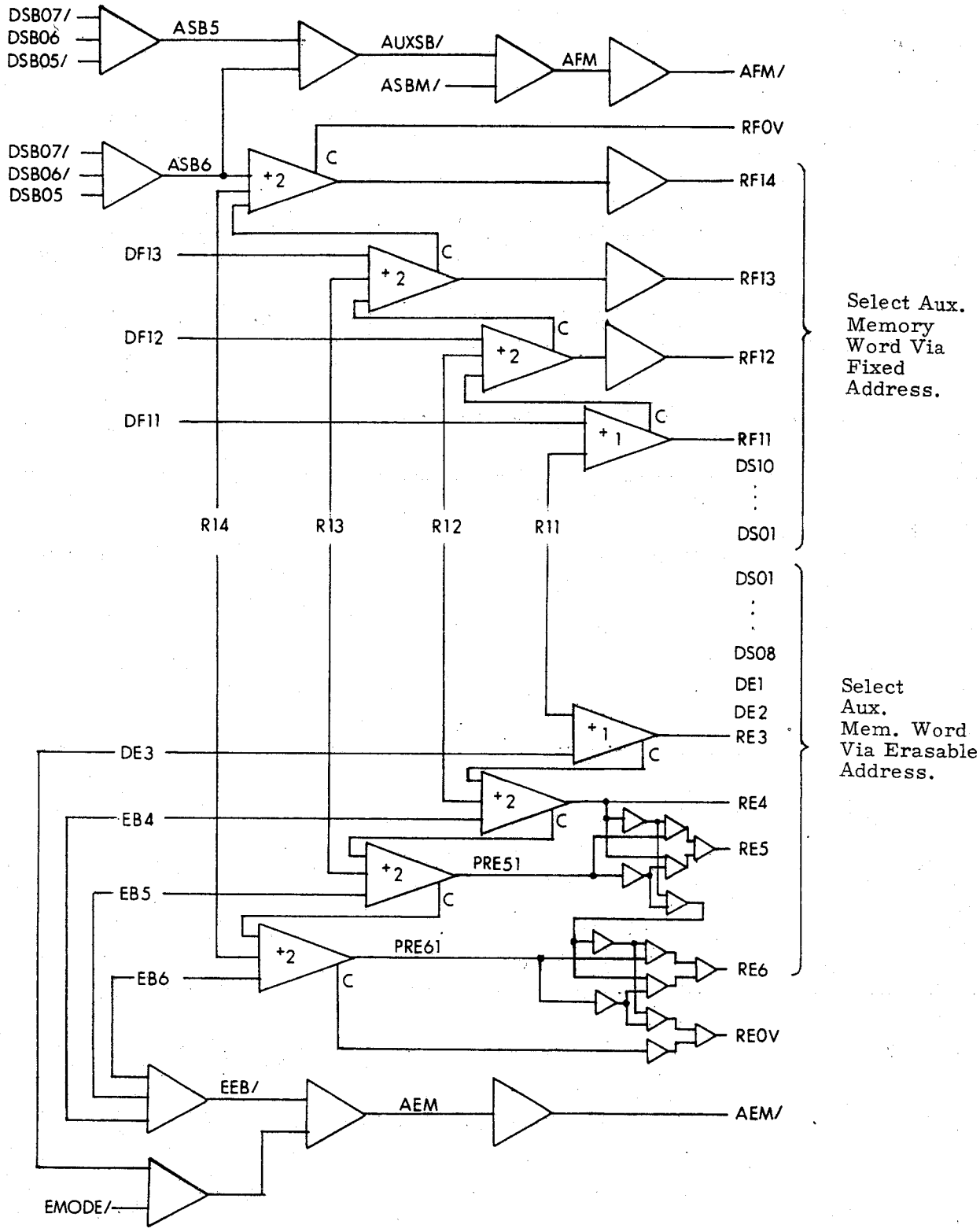
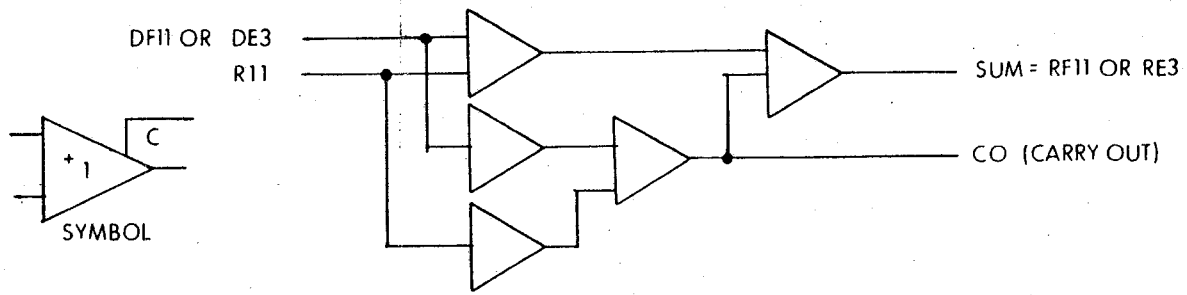
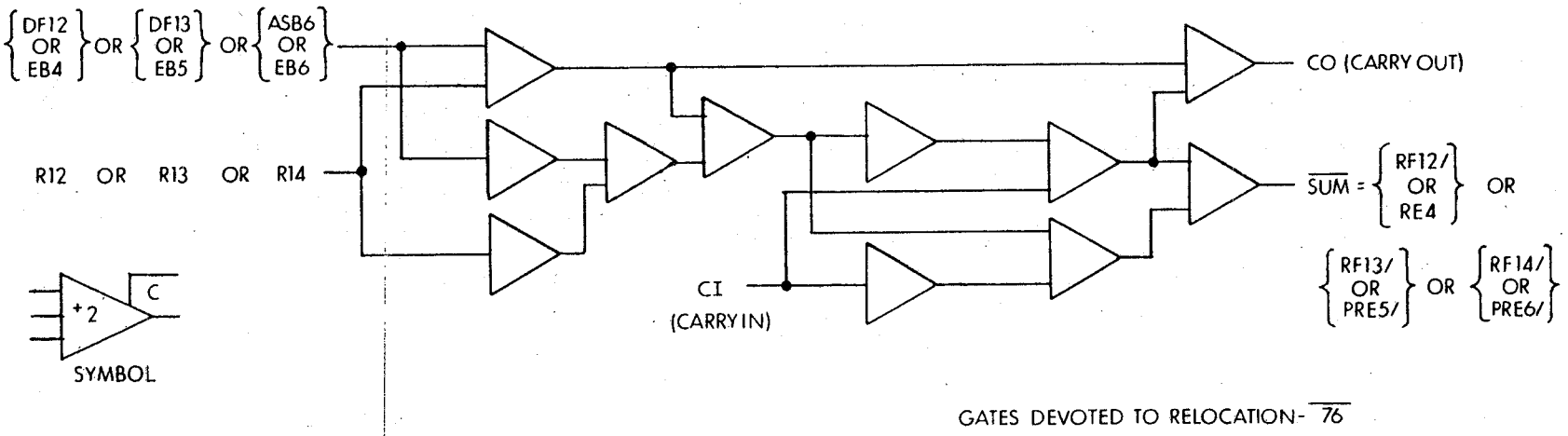


Fig. A.2 ACM addressing logic with type 1 and type 2 adder stages, and logic to make EBANK 10 start at beginning of FBANK 50.

Type 1. No carry in. Quantity 2 at 5 gates 10



Type 2. With carry in. Quantity 6 at 11 gates 66



Conclusion: 1 special chip for each adder stage would be a good economy.

Fig. A.3 All-Nor-Gate adder stages for auxiliary memory relocation.

dynamic memory allocation described above). This program is variously called a monitor, an executive, an operating system, etc., or (most descriptively) a resident system. In the AGC-AM setup, the program in the 36 banks of rope memory constitutes the resident system as well as the high-priority mission and utility routines.

In AGC practice, we have come to use the word "program" to mean a whole 36K-word ropeful, implying that every program is completely independent of every other program. Since a program that runs in the ACM is relocatable and depends heavily on the AGC program for interpreter, executive, waitlist, and many other services, it ought to be categorized by a different name, such as "segment". The definition of a segment, then, consists of the following properties:

1. A segment is a collection of instructions, constants, and variables that form a functional unit, such as a verb or a group of closely related verbs; all words and variables dedicated to the segment are located in the ACM.

2. The size of a segment is measured in 1024-word banks (F-banks); thus, no ACM bank contains protions of more than one segment -- all this fits with the principle of relocation in increments of one F-bank.

3. Every segment runs as if it had the ACM all to itself, and does not depend on the simultaneous presence of any other segment.

4. Every segment may use the AGC program and variables as needed.

5. A segment is assembled and manufactured as a unit, stored on the ATM as a unit, and loaded into the ACM as a unit.

The consequences of these properties are worked out in the following sections on various aspects of programming segments. Where it makes a difference, which is not at all often, the assumption is made that address relocation is by hardware.

A. Restriction

The E-mode concept of basic and interpretive accesses to ACM E-banks requires unrelocated addresses for interpretive access to be in AGC E-banks 0-7 or in ACM E-banks 14-17, 24-27, 34-37, 44-47, 54-57, 64-67, or 74-77. This last expression will be abbreviated hereinafter as "n4-n7, n = 1(1)7". Relocation allows interpretive access to any of ACM E-banks 14-107, excluding only 10-13 because relocation doesn't go backwards and probably shouldn't be allowed to wrap around the end of ACM memory.

B. Assembly Rules

Every segment is designed to run with a particular program (property 4), so it should indicate to the assembler the name of that program. This can be done by requiring in every segment's assembly language a card of the form:

(location)	(op code)	(address)
SUNSHINE	SEGNUM	10D

which also gives the identifying number by which the segment is known to the program, or at any rate to the program-segments system (aggregate). In this case, the segment in which that card appears is segment #10 of the program SUNSHINE, and it will bear that number as identification on the AM tape. The uses of the identifying number will be gone into more thoroughly in the section on operation. In any case, these numbers will be important, there will be a lot of them, and nobody wants to remember a mess of old numbers anyway, so programmers should be allowed to generate them by writing segment names instead. For this purpose, an op code SEGID may be defined as follows:

(location)	(op code)	(address)
	SEGID	S4BHOUSE

which forms a single-precision constant whose value is the segment number given in the SEGNUM card in segment S4BHOUSE. In effect, the segment number is the definition of a "symbol", the segment's name, which is available to the program declared in the segment's SEGNUM card, and to all other segments that also declare that program. In short, an aggregate consists of one program and any number of segments; each segment has one card that names the program and assigns a unique number to the segment; and any member of the aggregate can contain SEGID constants that identify any segment in the aggregate. In programming terms (since the primary use of the SEGID constant will be to call for execution of a segment) this means that you can call any segment from any part of the aggregate without knowing its identifying number. Only the Segment I. D. Number Subcommittee (you think I'm kidding?) will know or care. This arrangement also means that you cannot accidentally call from one aggregate a segment that belongs to another.

A good percentage of the concepts of segments and aggregates and their relationship arises directly from two plain facts: (a) sooner or later there will be lots of segments, and (b) no assembler can or should attempt to assemble the whole aggregate at once. The fundamental idea of these concepts is

to make the most out of the natural segmentation points in the problems to be solved, or, failing that, in the enforced limitations of the size of AGC/AM memory. It follows from the properties of a segment that it be assembled without any regard for symbol definitions or memory allocations in other segments of whatever aggregate, but with complete knowledge of symbol definitions in the AGC program that it goes with. This need may be readily mechanized by making the assembly-of-segments a operation distinct from the assembly-of-program, by requiring that the program indicated on the SEGNUM card had been assembled before a segment is assembled, and by using the entire symbol table of the program to initialize the symbol table of the segment. Example:

```
Y   ASSEMBLE NEW AGC SEGMENT S4BHOUSE BY ROCKET DWELLER
L   BIOSPHERE MONITOR ROUTINES FOR USED S4B
 0001 SUNSHINE 10D
etc., etc.
```

would (a) reject if there were no AGC program SUNSHINE, (b) have all the symbols and definitions from the latest revision of SUNSHINE in the symbol table of S4BHOUSE, (c) cause the card

```
nnnn        SEGID        S4HOUSE
```

to be assembled as 00012_8 in subsequent assemblies of SUNSHINE and all its segments, and (d) generate a binary record that specified which revision of SUNSHINE was used, and included only the words generated from S4BHOUSE coding. The assembler should also take note of the fact that a revision of the program makes all its segments obsolete, in general.

Memory allocation in segment assemblies is made easy and straightforward by hardware relocation. The principle that relocation is by the same amount for fixed and erasable addresses, together with segment properties 1 and 2, suggest that the instructions, constant, and variables of a segment should be bunched into a contiguous region of memory starting at the beginning of bank 50 for optimum relocatability. For instance, if S4BHOUSE needs 2 "fixed banks" and 4 "erasable banks", and if it contains interpretive coding that refers to the erasable, then the instructions and constants should be allocated to banks 50 and 52, and the variables to E-banks 14-17 (replacing bank 51). This pattern may in fact be operated out of 14 different places in the ACM: 50-52, 51-53, 52-54, ..., 65-67. An exception to the bank-50 rule occurs if a segment with interpretive coding takes up less than 1024 words, variables included. If it were started in bank 50, the restriction would make it take a second 1024-word bank for its erasable, so it should all be allocated to bank 51.

The interpretive mode of access to ACM erasable requires another assembler facility to fill in bit 10 of BBCON words and to let the programmer declare which access mode he is using in the various parts of the program. The EBANK = declaration is well suited to this purpose, if one feature is added: the location field must be either blank (as at present), meaning basic mode, or contain PRETMODE, meaning interpretive mode. In one-shot EBANK = declarations, this would determine bit 10 of BBCON words, and in standing declarations it would govern the checking of references to ACM erasable.

C. Manufacturing and Simulation

Manufacturing output from a segment probably should carry as part of the identification the name and revision number of its program, and an indication as to whether it is obsolete. A segment should be considered manufacturable or not by the same criteria as a program, mostly to keep the manufacturing process from having to make decisions that the assembler couldn't on the basis of information given by the programmer. The minimum set of manufacturing formats for segments is:

```
S       PUNCH MASTER DECK
S       PUNCH SYMBOL TABLE
S       PUNCH SYMBOL TABLE AND MASTER DECK
```

which brings up the question: what exactly is a segment's symbol table?; are they its own symbols or those of its program as well? The symbol table in the binary record is ultimately used only in simulations, which must use the proper revision of the program as well, so the only symbols a segment needs in its binary record are the ones defined in its own coding. Another manufacturing function that might pay off in reliability is writing the AM tape directly through special hardware, but this could come later.

Segment property 3 suggests that any segment can be at least partially tested by itself in the company only of its program. Therefore, it is meaningful to speak of simulating a segment, if the simulator collects the proper revision of the program for the run. Simultaneous testing of two or more segments should be done by a program simulation, which must collect non-obsolete segments on request, and account for the action of the AM tape.

D. Operation

The concept of relocation and the properties of segments set down here rest on the assumption that the ACM will be occupied at certain times by unrelated segments doing unrelated things for jobs with various priorities, but they are by no means limited to that assumption. In the simplest case, every segment might be a whole 16K ACM-full, corresponding perhaps to a particular mission phase. Relocation would not be used in such a case, but at least it would do no harm.

This is a good place to spell out the meaning of segment property 3 a little more thoroughly. It reads in part, "Every segment ... does not depend on the simultaneous presence of any other segment". This is not to say that a segment is unaware of the existence of any other segments, or even that the AGC/AM system cannot benefit from the simultaneous presence of two segments, but only that the interface between segments be constructed so as to depend on simultaneous presence. For instance, imagine that segment S4BHOUSE, which controls the biospheric parameters in a spent booster-turned-laboratory, requires Verb 97, "Initiate Sewage Treatment". If Verb 97 serves no other segment than S4BHOUSE then it ought to be part of that segment, assuming that the ACM is big enough for both. Now suppose that another segment, EXOFAUNA, has been introduced to run the biosphere of a cage for such captured animals as the Amorphous Ringwraith from the dark ring of Saturn; it too needs Verb 97 (presumably). In this case, it is likely that overall efficiency will be served by giving Verb 97 a segment of its own named DEERISLE, which will be called as a subroutine by S4BHOUSE and EXOFAUNA from time to time. If DEERISLE is present in the ACM at the time of a call, it can be used at once. If it is absent but there is a vacancy of sufficient size, the intersegment communication routine (ISCR) must fetch DEERISLE from the tape before passing control to it. If there is not a vacancy of sufficient size, something has got to go. In particular, if S4BHOUSE fills all 16K, or if its continuing operations has a lower priority than all other present segments and the call to DEERISLE, it must be overlaid by DEERISLE and restored from tape afterward. In short, any segment may call or otherwise converse with any other, but in the worst case, it must be prepared to get out of the ACM while the other segment is active.

Given some idea of the operational worst case, it is possible to set down some positive statements about segment interfaces, to supplement the rather negative statement in property 3. First, the modes of conversation should be a fairly simple set like GO TO, CALL, and DATA CALL. Second, these modes should correspond to entries into an extended interbank communication

routine (ISCR) as mentioned in the preceding paragraph. These same entries would be used by the AGC program to communicate with a segment, but not the other way around; every segment may use AGC coding and erasable as freely as the AGC program itself. When the ISCR is called by any part of the aggregate, it needs the segment identification (a SEGID constant), an address relative to the beginning of the segment, and a priority for the conversation being requested. Example:

```
TC          ISCRGOTO
SEGID       S4BHOUSE
OCT         24003       COMBINED PRIORITY AND REL. ADDR.
```

where the beginning of segment S4BHOUSE contains:

```
SUNSHINE  SEGNUM  10D

          BANK    50
          SEGID   S4BHOUSE
          EBANK=  1400        SPECIAL: SETS UP NO. OF BANKS AFTER
          BBCON   50          FIRST: 60123 MEANS BANK 50 + 3 MORE.
S4BNTRY1  TCF     ROUTINE1    TRANSFER VECTOR: ALL ENTRY
S4BNTRY2  TCF     ROUTINE2    POINTS ARE PACKED HERE.
```

The call to ISCRGOTO will pass control unconditionally to ROUTINE2 of S4BHOUSE by way of S4BNTRY2 (word 3 of S4BHOUSE).

The general part of the ISCR, common to all call modes, can then consult its table showing which segments are present, and if the desired one is absent, seek it on the tape and make the executive find another job to do in the meantime. A segment on tape must contain some bookkeeping information in its first few words, including its own identification, its nominal base address (probably bank 50 to 51), and its size in banks. This information can be compared against another ISCR table that shows which banks are occupied by active segments. After the ISCR finds or makes a place to read in the segment, it enters the segment ID and the relocation amount into its present-segment table. When the time comes to activate the segment (which is right away if it was present to begin with), the ISCR installs the relocation amount in the relocation register and passes control to the indicated address in the segment, if the mode is GO TO or CALL, or to the DATACALL routine in the rope. Furthermore, if the mode is CALL or DATA CALL, the return address in the calling segment or program is saved in the present-segment table so as to correspond to the called segment.

All these procedures are designed to meet the worst case of AM usage, in which an occasionally overcrowded ACM is being used by several independent jobs or tasks, so it may be possible to implement a subset for a realistic situation. What matters is that the AM hardware be prepared for the worst case, because if it isn't, the design will doubtless be frozen when the worst case does arrive -- of AGC Block II.

A.4 COLOSSUS

This program is not a complete lunar mission program yet but it is the only one available to this study. The Fixed Memory map for the program in its present state is shown in Fig. A.2.

With regard to the Fixed Memory map it appears that there would be no point in overlaying the first two modules, banks 00-13 (there are 6 banks to a module). The next module, module 3, is mostly autopilot; modules 4 and 5 look like they could be overlayed while module 6 is mostly pinball.

Thus, we effectively have 2 rope modules (4, 5) or 12 banks to utilize in the overlay model which is slightly less capable than the present contemplated in the AM.

However, it is felt that the overlay model would require excessive effort to sequence the programs so that they could be interwoven in a non-interfering manner. (See Section A.2.)

The extended-bit concept could essentially allow one more memory computer (36K rope and 16K AM) and the penalty is the time it could take one to modify addresses so as to be extended.

The study of COLOSSUS is only beginning and will be continued using the extended-bit concept as a model.

REFERENCES

Ref. 1 R-489 Users Guide to the Block 2 AGC/LGC Interpreter, C. Muntz.

Ref. 2 Apollo Guidance Computer Information Series Issue 6B - Raytheon.

Ref. 3 E-2052 AGC-4 Basic Training Manual, B. Savage and A. Drake.

00 Interpreter Delayjob	10 Displays Phase Maint S40. 2, 3 GEOM	17 DAP	26 ENTRY Lunar Rot Ephem S61's P60's	35 P22, P23 P30 P34, P35 R31 Module #5 ↑
01 Interpreter Executive Waitlist Restart Tables & Routines	11 recons Orbital Intvel	20 DAP S40's S62. 1, 2	27 TFF UPDATE VECPT R60, R62 R30 P05, P06 Module #4 ↑	36 P20's MEASINC
02+ 03 FIX-FIX	12 CONICS	21 DAP Module #3 ↑	30 Rend. Param TPI P32, P33 P77	37 Servicer Bodyatt P20, P21 P31 R21 R61
04 Verb37 + P00 EXTVERBS PINBALL	13 RTB's INT INIT LAT LONG R32 MIDGIM R53 Module #2 ↑	22 Kalcmanu S401s RTB's	31 P15 Cislunar Landmarks P37, P38, P39	40 PINBALL R33
05 FRESH START and RESTART DOWNTELEM Module #1 ↑	14 STARTAB R50, R51 R55 P50's	23 Inflight Powflight Comgeom Periapo S30's	32 RTE	41 PINBALL
06 IMUCOMP T4RUPT	15 P52 R52 Entrydap CMDAPON	24 R03 R23 P40's	33 P01 P02 P03	42 PINBALL R05
07 SXTMARK IMODESW KEYRUPT R02 S62, 8	16 DAP S40's GEOM	25 ENTRY	34 P11, Boost R22 R61. 1 Prelaunch	43 SELFCHEC Module #6 ↑

Fig. A. 4 COLOSSUS Memory Map

APPENDIX B

Transcript of DD Memo 381
Reliability Aspects of Auxiliary Memory

There are several operational configurations of the AM and the purpose of this note is to examine them from a general reliability viewpoint.

The configuration that will be discussed are:

1. AGC
2. AGC + AM
3. AGC + AM (perfect disable)
4. AGC + AM (2 redundant memories)
5. AGC + AM (1 redundant memory

B.1 AGC

The actual failure rate of the AGC varies, depending on the source. If one uses the information in R-524*, the contribution to that failure rate due to the E-memory is 13%, that due to the fixed memory is 16%, and the contribution of both is 29% (i.e., failure rate would be decreased by 29% with perfect memories).

The mean-time-to-failure for the AM has been estimated to be on the order of 6400 hours and data on the AGC (5,200 - 9,200 hours)** indicates that, if one assumed a MTBF of 6,400 hours, one would not be far out of line.

B.2 AGC + AM

If one used the AGC and the AM without regard to where one placed critical programs, etc., the mean-time-to-failure would be just one-half that of each one separately, since we are assuming they are each 6400 hours.

Therefore, MTBF of both is 3200 hours.

B.3 AGC + AM (Perfect Disable)

If one assumed that all the critical programs would be put in the AGC and that the AM were capable of disabling itself perfectly, the MTFB would be that of the AGC alone (6400 hours).

---

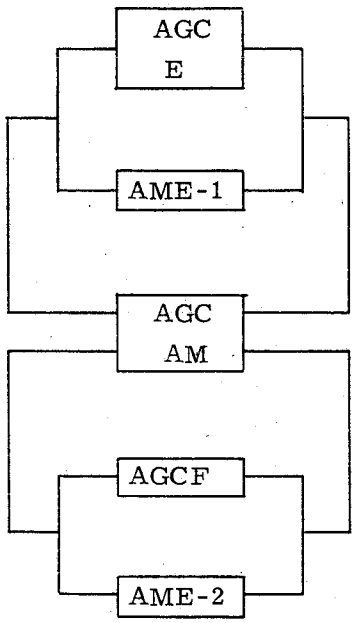
*R-524, Apollo Staff, Quarterly Reliability and Quality Assurance Progress Report, December 1965 (C).

**Apollo Guidance Computer Reliability Assessment Report No. 24, January 1967, George Mayo and Eldon Hall, (U).

B. 4 AGC + AM (2 Redundant Memories)

If one wanted to set up the AM so as to have the critical programs in both units and to provide the electronics necessary to decide which memory were bad, one could gain some additional reliability.

Treating the memories as simply redundant, one has the following model.



$$Rel = Rel\text{-}1 \cdot Rel\text{-}2 \cdot Rel\text{-}E \cdot Rel\text{-}F \qquad (B\text{-}1)$$

Rel-1 = Rel AGC with perfect E and F memories.
Rel-2 = Rel AM with perfect memories.
Rel-E = Rel redundant E memories.
Rel-F = Rel redundant F memory.

If one assumes that the E memory is just as reliabile as the F memory (this notion of reliability obviously disregards susceptibility to transients and restart capability) and that the AM memory is just as reliable as the AGC, one gets

$$Rel\text{-}E = RE = RF = E^2 + 2\,(1 - E)\,E \qquad (B\text{-}2)$$

where E is the reliability of the nonredundant E memory. For this calculation it was assumed that

$$E = e^{\frac{-.15t}{6400}} \tag{B-3}$$

$$Rel = Rel\text{-}1 \cdot Rel\text{-}2 \cdot E^2 \cdot F^2 \cdot \left[1 + 2\left(\frac{1}{E} - 1\right)\right]\left[1 + 2\left(\frac{1}{F} - 1\right)\right] \tag{B-4}$$

but since E = F and $Rel\text{-}1 \cdot Rel\text{-}2 \cdot E^2 \cdot F^2 = Rel\ AM \cdot AGC$

$$Rel = e^{\left(\frac{-t}{3200}\right)} \cdot \left[1 + 2\left(e^{\left(\frac{-.15t}{6400}\right)} - 1\right)\right]^2 \tag{B-5}$$

Thus, to find the improvement in the MTBF, one has to solve the following equation for t.

$$e^{-1} = e^{\left(-\frac{t}{3200}\right)}\left[1 + 2\left(e^{\left(\frac{-.15t}{6400}\right)} - 1\right)\right]^2 \tag{B-6}$$

By successive approximations one gets t = 1.39 · 3200 hours. For perfect memories one would have $\left(1 + \frac{.3}{.7}\right)(3200) = 1.43\,(3200)$. That is, the mean-time-to-failure has been increased by approximately 39%.

B.5 AGC + AM (1 Redundant Memory)

This case is discussed because it might be difficult to store and protect both the erasable and the fixed memories.

The equation here is the same as (B-5) above except the last term is not squared. Thus, rewriting (B-5) we have (B-6').

$$e^{-1} = e^{\left(\frac{-t}{3200}\right)}\left[1 + 2\left(e^{\left(\frac{-.15t}{6400}\right)} - 1\right)\right] \tag{B-6'}$$

The solution for t is t = 1.17 × 3200 hours. For a perfect erasable or fixed memory, one would have $t = \left(1 + \frac{.15}{.85}\right) 3200 = 1.177 \times 3200$ hours. These results indicate that one should explore further the problems in trying to use the AM as a redundant memory.

Remarks

1. As part of the preliminary study of the AGC/AM system, consideration has been given to the optimization of system reliability. One method of increasing the reliability is the sum checking of the contents of the ACM after transfer of program from the ATM and prior to use by the AGC. This sum check would significantly increase one's confidence in the quality of the programs stored in the ACM.

2. Another consideration is that of what is probably the "worst case" hard failure of the ACM. This failure is the event that consists of a write line clamped to a "one" by the ACM, jamming the AGC, and that the ACM is unable to amputate itself from the AGC. This situation could be handled automatically or manually, enabling the AGC to run independently of the ACM.

3. The electrical reliability of the AGC-AM interface can be substantially improved by placing line drivers and receivers at the AGC interface in the harness connector.

This would simplify the interface module design (one less connector) and increase the noise rejection capability. Also, one could be working with standard gates and packaging versus the diode resistor networks presently envisaged.

The necessity of being able to operate with Core Rope Simulator or Pack would require a special test cable which would not need to be flight qualified.

4. It is also desirable to include the failure-detection capability now handled by the channel 77 failure monitor in AM.

DISTRIBUTION LIST

Internal

R. Alonso
R. Battin
B. Barrat
Hugh Blair-Smith
D. Bowler (10)
E. Copps
G. Cox
R. Crisp
J. Dahlen
P. Felleman
R. Filene
F. Gaunt
K. Glick
E. Grace
A. Green
E. Hall
D. Hanley
A. Hopkins
D. Hoag
L. B. Johnson
J. Kernan
J. Kingston*
A. Kosmala
A. Laats
D. Lickly
R. Lones
F. Martin
J. E. Miller
J. S. Miller
C. Muntz
J. Nevins
R. Ragan
W. Schmidt (MIT/MSC)
C. Schulenberg
N. Sears
D. Shansky
R. Weatherbee
W. Widnall
C. Wieser
Apollo Library (2)
MIT/IL Library (6)

*Letter of Transmittal Only.

External

National Aeronautics and Space Administration (13)
Manned Spacecraft Center
Houston, Texas 77058
ATTN: C. Brady EG25
T. Chambers EG25
R. Chilton EG
L. Danewood BG55
P. Ebersol EG26
C. Frasier EG44
R. Gardner EG
R. Harland PP7
M. Holley EG44
H. Howard EG44
W. Kelly PP7
G. Xenakis EG

National Aeronautics and Space Administration (1)
600 Independence Avenue SW
Washington, D.C. 20546
ATTN: P. Schrock

NASA/RASPO at MIT/IL EG442 (1)