ROBERTSON

User's Guide to

THE GENERAL ASSEMBLER PROGRAM (GAP)

March 1970

Charles Stark Draper Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

ACKNOWLEDGEMENT

This report was prepared under DSR Project 55-23870, sponsored by the Manned Spacecraft Center of the National Aeronautics and Space Administration through Contract NAS 9-4065 with the Charles Stark Draper Laboratory, Massachusetts Institute of Technology, Cambridge, Mass.

This report was compiled and edited by Mr. Stephen Landy of The Computer Usage Corporation under contract to the Charles Stark Draper Laboratory.

Questions should be directed to Peter Peck MIT/CSDL-7.

PREFACE

The GAP (General Assembler Program) system is a system which assembles assembler language for the AGC (Apollo Guidance Computer). In addition to doing assemblies, the GAP (AGC) system has a large section which performs tasks needed to send necessary information to the manufacturers of the AGC and to ensure the accuracy of the information sent.

The historical forerunner of the GAP (AGC) system was the YUL system. The YUL system ran on the Honeywell-1800 computer.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

Erasable memory is divided into 8 banks with 256 words per bank. Words in banks 0 - 2 are addressed directly by the address portion of the instruction word. Addressing words in banks 3 - 7 is done by relative addressing. This requires that a bank register (EBANK) be set to the proper value at execution time. The 8 hardware registers mentioned above are treated as the first 8 locations in EBANK 0.

Fixed memory is divided into 36 banks of 1024 words each. Two of these banks are directly addressable. An additional 22 banks may be addressed using 1 bank selector register (FBANK) to supplement the address information in the instruction word. The remaining 12 banks require the use of 2 registers (EBANK and SUPERBNK) in addition to the address information in the word.

1.3 AGC Language

AGC programs are written in two languages. The first is an assembly language referred to as AGC Basic or just Basic. Part of the Basic coding of each AGC program is a limited facility interpreter. This part of the program uses AGC Interpretive coding as a set of parameters to be used at execution time. Since each interpretive instruction is expanded into many basic instructions, the result is similar to the savings effected by the use of subroutines. Thus, despite the space allocated to the interpreter, a net saving in instruction storage requirements is realized. The penalty paid for saving space is greatly increased execution time. Thus, interpretive coding is used only where realtime performance is not required.

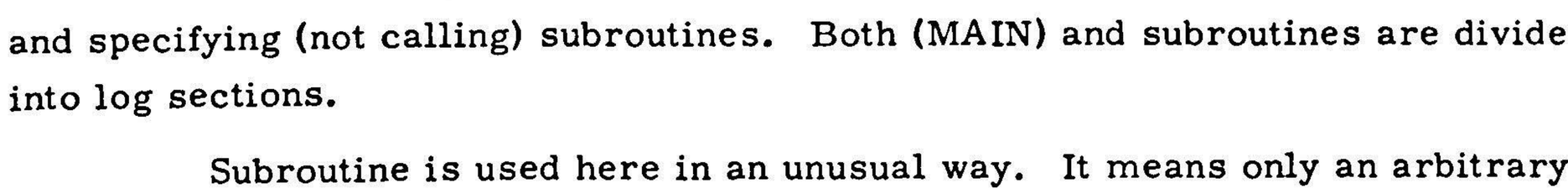
1.4 Structure of the AGC Programs

AGC programs have three levels of organization. They are (MAIN), SUBROUTINE, and Log Section. (MAIN) is essentially for remarks, bookkeeping, and specifying (not calling) subroutines. Both (MAIN) and subroutines are divided into log sections.

Subroutine is used here in an unusual way. It means only an arbitrary section of the program. It is not called in the usual sense: subroutines are used mostly to split the whole program into manageable pieces. The split has no necessary relationship to the structure of the program.

Log sections are tied more directly to the program logic, since instruction numbering is carried out within log sections.

To understand this structure, a picture is helpful. First note that the subroutines to be assembled with a given program are specified within (MAIN) by means of SUBRO cards.



Here L1, L2, S1L1, etc., denote log sections made up of a set of detail cards (AGC coding and remarks). The cards contained in the subroutines named by the SUBRO cards in the log section whose name is SUBROUTINE CALLS are simply strung onto the end of the main program. The subroutines appear in the order in which the SUBRO cards appear.

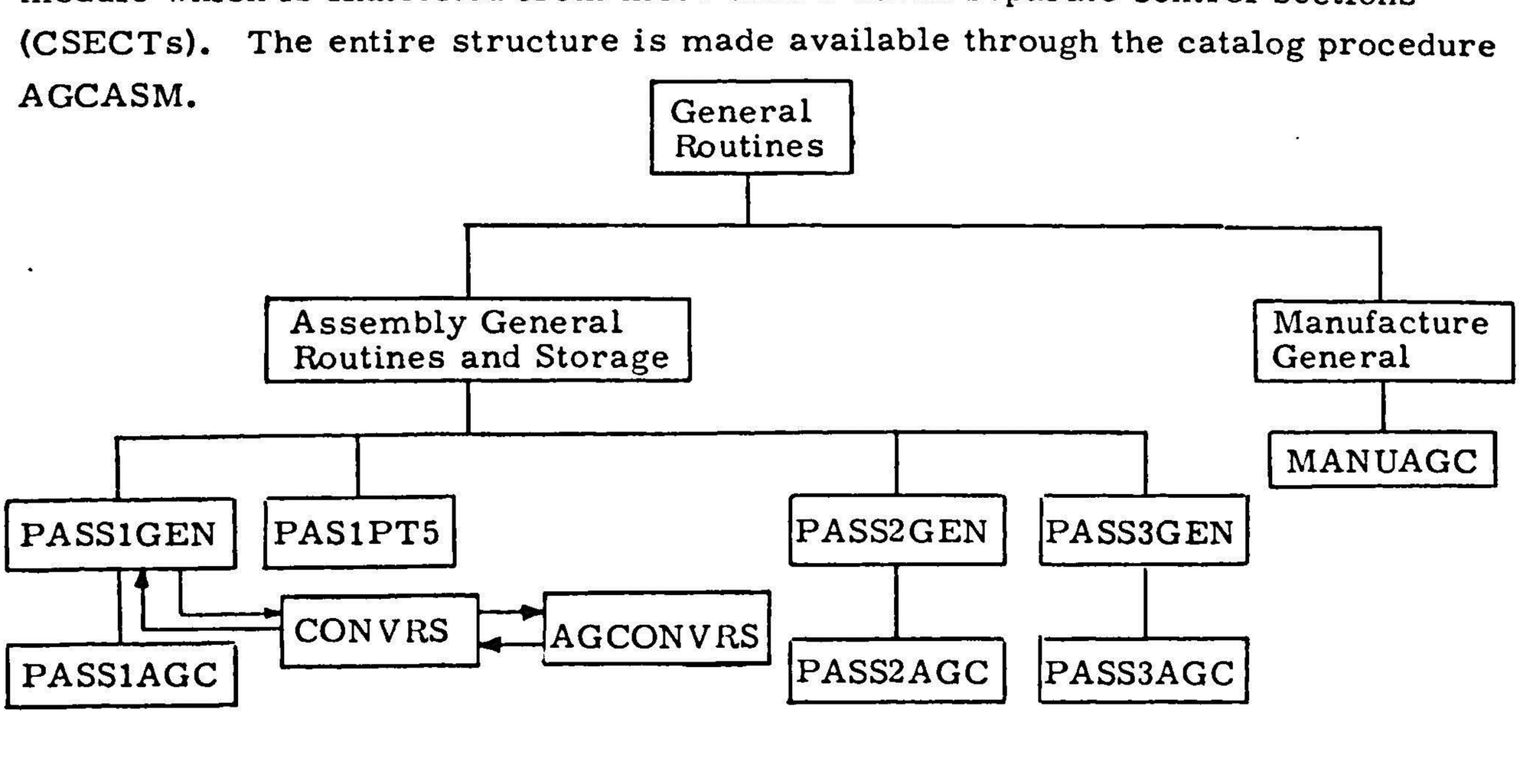
1.5 The Role of the 360/75

In order to fully appreciate the AGC and its associated languages and procedures, it is necessary to understand the central role the 360/75 plays in program development, simulation, and manufacturing. To put it most dramatically: the AGC cannot physically exist until the programer's work is done.

This work is carried out through a series of 360/75 programs. These programs create files of card images of the source deck, assemble the source program, and store the binary object module on a second disk pack, simulate the logic of the AGC with or without environment, and, when the program is finally debugged, manufacture various binary outputs to be used in wiring the AGC. The details of the use and functions of these programs will be presented later, but for now, note that, making the preposterous assumption that one had coded a complete and perfect AGC program, this is the order of work flow.

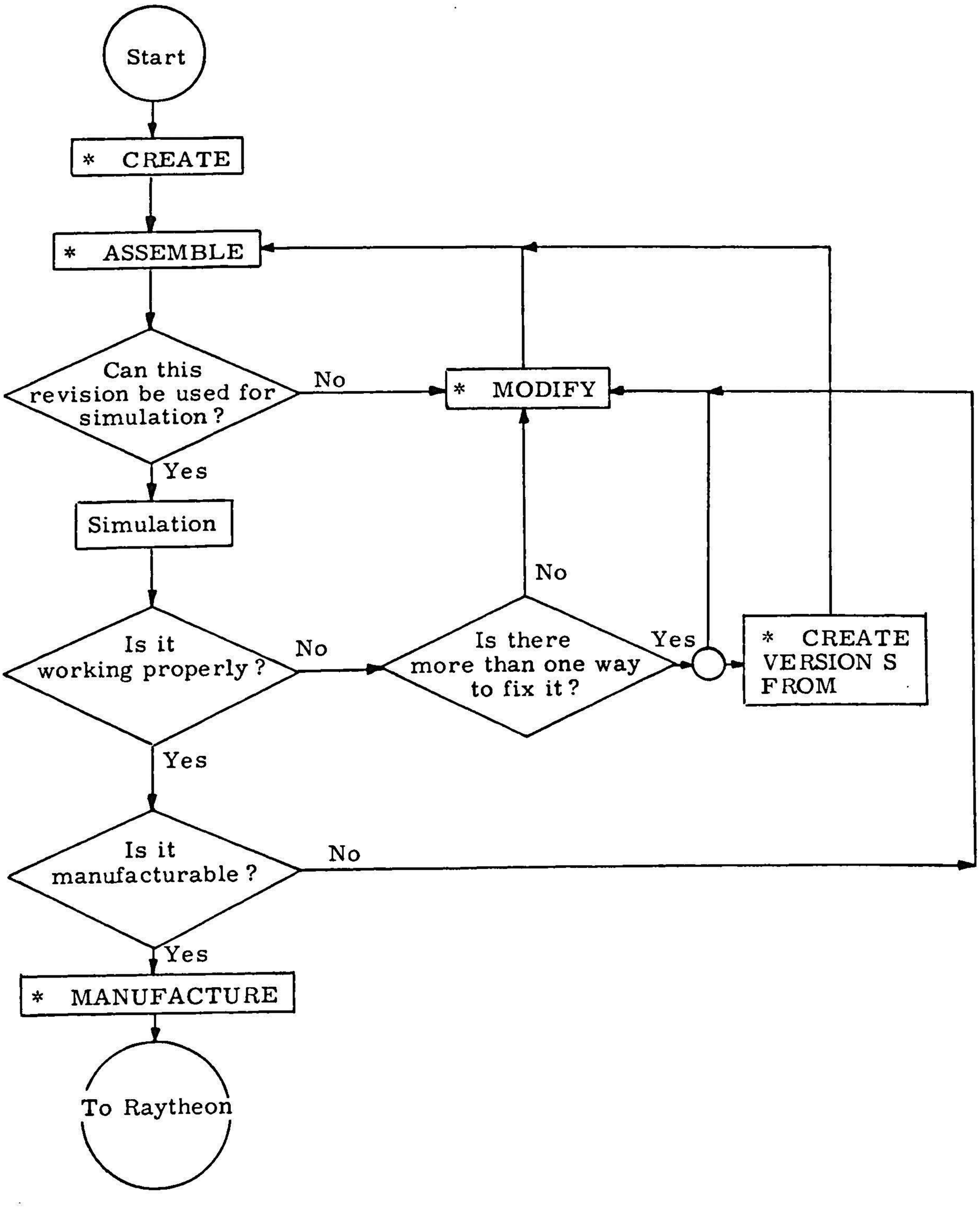
1.6 The GAP System

The GAP system may be diagramed as follows. It is an overlay load module which is linkedited from more than a dozen separate control sections (CSECTs). The entire structure is made available through the catalog procedure AGCASM.



facilitating the construction of a generalized assembler. By replacing those control sections specific to the AGC with control sections specific to other computers, one gets an assembler for that computer.

1.7 Usual Flow of Work in Developing an AGC Program



2. AGCONVRS

2.1 Introduction to AGCONVRS

AGCONVRS is an extremely sophisticated, highly user-oriented multiple file maintenance system for linearly ordered data sets. It allows the user to create files, merge additional material, delete material, renumber cards, print and punch material. All of these features leave complete histories. Thus it is not necessary to use special backup procedures to recover from user error. Sufficient features are built into the system to avoid this. All of these features are available at levels that correspond to AGC coding. That is, one creates programs, subroutines and segments, and is able to work within these at the log section level. Since the programs for the AGC contain upward of 40,000 cards, this eases numbering problems.

2.2 Filing System of AGCONVRS

This section is devoted to a characterization of the filing system of AGCONVRS in terms of principles that should direct the reader in his use of the system. It should be noted that the system is almost completely general: it can store almost any kind of card images. (Obvious exceptions are instructions to the system.)

First, some brief definitions. "Data" refers to those card images the user wishes to store for his further use. This can include programs, alphameric material of wide variety, or arbitrary combinations of these. A linearly ordered data set is one in which, given two cards from it, one can determine which card comes first. "File" has two meanings. The first is the same as that in System/360 JCL. That is, it is an area of storage unified by its labeling and addressing in JCL. The second meaning is more properly thought of as subfile. That is, it is a set of card images which occupy parts of a single JCL file. These card images are unified by the AGCONVRS program. This is rather abstract. It is perhaps less confusing to say that the program AGCONVRS has assigned to it one data set or file. This is done by the JCL with the program. AGCONVRS is then able to subdivide this storage space, in order to store sets of related card images. These sets of card images

will be PROGRAMS, SUBROUTINES, SEGMENTS, and VERSIONS. In order to avoid the necessity of trotting out this list or an "etc.," I wish to call these sets of card images by the generic name file. It should be amply clear (in the light of this warning) which meaning is appropriate. The directory is a file containing a list of names of the files stored, and an area where revision history and location and sequencing data for the records (card images) are stored. "Revision n" refers to an entire data file. It is generated by modifying revision n-1.

There are four things that characterize the system. The first is that two areas are written into during creation or updating of a file; one is the data area, the other is the directory. Second, each card image exists only once in the file. Third, once a card image is in the data file, it remains in the file. Finally, subroutines within programs, log sections within programs or subroutines, and cards within log sections are linearly ordered. Thus, the system is able to use the fact of total linear ordering to store and retrieve data. Although this scheme may be obscured by the details of the system and the proliferation of names of files, it tends to be the best arbiter of the use of the control options.

The main benefit of this kind of filing system is that very large files of data can be created and maintained (updated, modified, etc.), keeping all revisions of a given file available, without using excess disk storage space. The reason for this is that each update or version creation does not make a duplicate copy of the file, but adds only those cards which make it different from its predecessor. Of course, the system would use storage disadvantageously with relatively small files that underwent considerable revision. Even in this case, the ability to recall all prior revisions may outweigh the cost of additional storage.

AGCONVRS also has the capability to expand the storage area allocated to it for its data files. It informs the user when it does so.

2.3 Features Available to AGCONVRS Users

The next thing to discuss should be the details of the AGCONVRS filing system. This is in order to fully explain the options available to the user. Unfortunately, in order to describe the filing system, it is necessary to use the terminology of the control cards. Thus, in this section, this terminology will be introduced; the explanations will come later.

Instructions to the AGCONVRS system can take four forms. The highest level is the Director. At this level one can CREATE, MODIFY, DELETE, PRINT(*), and PUNCH(*) ones files. The next level is the subdirector FROM. This is a modifier to the director CREATE, and allows the user to add to a program's list of names, to the effect that subsequent revisions to the old and new Version are independent. The third level is the Acceptor or =LOG card. It too is used as a

modifier to director cards and specifies the log section to be modified, or (say) printed. This is necessary since directors and subdirectors refer to programs, subroutines, segments, and versions, while the numbers on the data cards are unique only within log sections. The fourth level is the merge control card. These allow the user to DELETE, CHANGE CARDNS, PRESERVE CARDNS, BEGIN (END) INSERT at the card level within log sections.

It should be noted in passing that once a file exists, it may be referred to as either a program or a subroutine, regardless of whether it was created as a program, subroutine, or version. Segments remain as segments.

Each director card refers to a file and causes the execution of a major routine of AGCONVRS. All such routines are reenterable. This means that as many director cards may be included in a job step as the user wants. AGCONVRS is part of the catalog procedure AGCASM.

2.4 Structure of AGCONVRS Files

Before proceeding with the details of the features just discussed, the filing system of AGCONVRS will be described, in the hope that it will prove useful to the reader in understanding the interactions between, and restrictions on, the various features.

It will be recalled that AGCONVRS uses two storage areas on a disk pack; one for data and one for the directory. The data area is further broken down into the linear file and the overflow file. The linear file contains the original contents of each data file. That is, it contains those card images that resulted from the cards that were included at the time of creation (creation without the subdirector FROM). The overflow file contains only the additions to the original file (both data and merge control cards). Within the linear file, all card images pertaining to a given file are in sequential order. There are no card images from other data files interleaved. The overflow file contains card images of those cards that are added during modification. It contains them in the order they are received; thus, the card images relating to a given file may not all be contiguous. The directory is similar to the overflow file in that information pertaining to a given file may be in several noncontiguous blocks.

When one originally creates a file, the card images are stored in the linear file. They are referred to by a NAME and a revision number. The name is that assigned by the user; the revision number will be one unless the user specifies otherwise. Each modify director increments the revision number by one and places the incoming cards in the overflow file. The directory is updated with the information that an additional revision exists and the address in the overflow file of the card images that were added. If one deletes revision n, the directory

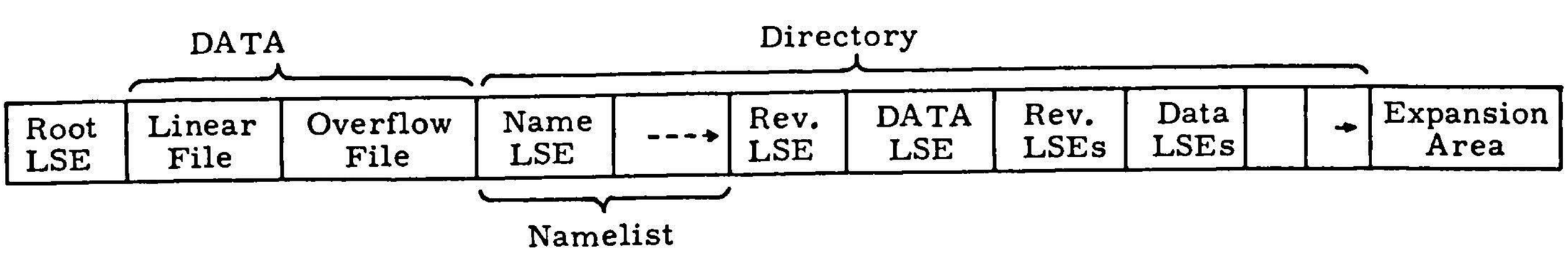
is modified so that the cards added to form revision n cannot be accessed. When the user subsequently modifies revision n-1 again, the card images for that revision are put in the overflow file in a location different from the location of those card images which made up the prior modification of revision n-1. A new entry is made in the directory as well. Thus, the data card images and the directory information that have been made obsolete remain on the disk pack until file maintenance is done. File maintenance is performed periodically to remove such items from storage. File maintenance also rewrites the linear file, incorporating the material on the overflow file. The overflow file is empty after this operation.

2.5 Structure of the Directory

The directory is considerably more complex than the data files. It contains all the information necessary to honor a request to the AGCONVRS system. It contains the name lists, revision number history, sequence number information, status information, and address information. The point is that the records themselves are disassociated from any information about themselves, thus they can be in different status with respect to different programs: this is the heart of copyless quasi-duplicate data storage.

The information is stored in four types of record areas called list structure elements (LSEs). These are root, name, revision, and data. These provide, respectively, information for initialization of the system, name and version lists with pointers to appropriate revision and data LSEs, external and internal revision number correlation and status information, and location and description of data records. The root LSE is not physically contiguous with the rest of the directory. Of necessity, it precedes the other areas of the AGCONVRS file. Each type of LSE is associated with an algorithm within AGCONVRS. The algorithm is designed to use the information in an LSE of the appropriate type. Each LSE is subdivided into a number of fields. These fields are associated, in a regular way, with the various parts of the information which the LSE contains.

The various files fit together in the following way. The first thing in the file is the root LSE. Next is the record storage area, which is split into the linear and overflow files. Third is the directory, which contains the name LSE, and then blocks which contain one revision LSE and several data LSEs. Finally, there is an expansion area. A rough pictorial representation is as follows.



Each LSE is subdivided into a number of fields. These fields are associated, in a regular way, with the various parts of the information which the LSE contains.

There is one root LSE. It has twelve fields in its basic form; optional information may be included by making modifications at the systems level. The first field identifies the file as an AGCONVRS file. The second field is the standard increment for the sequence number. In the case of the GAP system, the six digit sequence number is incremented by 100, wherever incrementation is left to the system. Of course when a sequence number is specified by the incoming cards, this overrides the system renumbering. The third field contains the number of the first column of the sequence field, two in the case of the GAP system. The fourth column contains the length of the sequence number field, six in the case of the GAP system. The fifth and sixth fields contain, respectively, pointers to the last record in the linear and overflow files. The ninth field contains a pointer to the start of the name LSE. The tenth field contains a pointer to the expansion area relative to the directory. Fields eleven and twelve contain, respectively, the original size of the linear and overflow files.

Each name LSE has either six or nine fields, depending on whether it is associated with a file under its original name or with a version. The first field contains the size of the LSE. The second field contains the internal representation of the name of the file. The third field contains two flags; the first indicates original file or version, the second gives active or deleted status. The fourth and fifth fields give, respectively, the address, relative to the directory, of the first revision and data LSEs relating to the file. If the file is not a version, the sixth field is the last and gives the external (user assigned) file name. If the name LSE refers to a version, the sixth field contains a branch index. This is used, together with the internal revision number in field seven, to find the original file, which the version is a descendant of. The eighth field contains the initial revision number assigned by the user. The ninth field contains the external version name.

Each revision LSE has four fields. The first gives the size of the LSE. The second field contains the internal revision number. The third field contains flags to indicate possible numbering errors due to mechanical failures (non user errors), delete status. The fourth field contains the actual revision number.

The set of name LSEs in the GAP system is made more complex by the fact that each log section has its own name LSE. Thus the collection of name LSEs necessary to retrieve a program has as many elements as there are log sections in the entire program.

The data LSEs associated with each revision LSE have nine fields. The first is the size of the LSE. The second field is the internal revision number of

the revision which inserted the group of records pointed to by the data LSE. The third field contains the number of records covered by this LSE. The fourth field contains two flags; the first indicates whether the records are in the linear or overflow files, the second indicates whether the records are data, control, or remarks. The fifth field contains a pointer to first record of the group, relative to the start of the section which contains the record. The sixth field contains the sequence number of the first record in the group at the time of insertion. The seventh field contains the increment to the sequence number to be used for this group of records. The eighth field contains the number of entries in the deletion list. The ninth field contains the deletion list, which is a list of the internal revision numbers of those revisions which deleted the records covered by this LSE.

As mentioned above, there are four main algorithms in AGCONVRS, each associated with one of the four types of LSEs. The start algorithm initializes the system. If the identification field of the root LSE is valid, the start algorithm sets up tables and copies the name list onto temporary direct access storage. The reason for this is to protect against machine failure, since AGCONVRS operates by modifying the directory.

The name algorithm searches the name list for a name which matches that on the director card. If it finds the name LSE it passes the location of the revision LSE's and the data LSE's to the revision and data algorithms. It also copies these LSE's onto temporary direct access storage.

If the director is a delete, the name algorithm checks for descendants. If there are none, the name LSE is flagged as being in delete status. If the name algorithm finds no match with the name on the director, it either creates a new name LSE (if the director is a create), or returns an error message (in the case of any director other than CREATE, including *CREATE/S FROM).

The revision algorithm searches the revision LSE's to find the revision number referred to on the director. If one is not found, it searches for a location to place a new revision LSE. Furthermore, during the search, the ancestral table is formed. It is a well ordered set of internal revision numbers which are associated with revisions or versions that are ancestors of the revision specified. The most recent revision subject to renumbering is also noted. If a version is involved, the revision number on the director is changed to the complete form which shows what the revision is descended from.

The data algorithm is in a sense a multiple algorithm, since it functions rather differently in case of retrieval (e.g., for the assembler), modification, or deletion of the revision. In addition during a modification or retrieval an auxiliary routine is used in the process of assigning sequence numbers to data records.

Since the assignment of sequence numbers is one of the keys to copyless quasi-duplicate data storage, it is worthwhile to explain it in some detail.

The main purpose of the auxiliary routine is to tell the calling routine which sequence number information to use. It first attempts to read the data LSE. If there are no more data LSEs, it returns to the data algorithm. If the data LSE points to stored control records (meta-remarks), then it goes on to the next LSE. If the data LSE points to data records, it then asks if any revisions in the deletion list occur in the ancestral table. If they do, it reads the next data LSE. If no entries to the deletion list occur in the ancestral table, the routine then asks if the last revision which caused renumbering was prior to the revision which created the data LSE. If this is not the case, then the routine creates an area which contains sequence number information and returns to the calling routine with a message to use the sequence number information in the area just created. When renumbering has occurred, the increment to the sequence number used is the standard one of 100, rather than whatever appears in field seven of the data LSE.

A routine which uses the data algorithm for retrieval must issue calls for one record at a time. The data algorithm reads a data LSE, and asks if the revision which created this data LSE is in the ancestral table. If not, it reads the next data LSE. If the LSE is in the ancestral table the auxiliary numbering is invoked. When the numbering routine encounters a data LSE which points to records which belong in the revision being called it sends the location and sequence number of the next record to the calling program. Only when one LSE has been exhausted does the data algorithm move on to the next LSE.

There are two basic operations which occur during file modification. Cards may be added to the file or cards may be made inaccessible to the new revision. The key to understanding both processes is to recall that the directory is actually rewritten during modification, hence data LSEs may be rewritten as two or more new LSEs.

First consider deletion of one or more card images. All cases may be reduced to the case where all card images to be deleted lie within a certain sequence number range. The range may, of course, contain only one card image. It is most fruitful to consider sequence number ranges, since a sequence number range can easily be calculated for each data LSE from the information contained in it.

The data LSEs are read, then range computed and compared with the range of the delete. If the range of the LSE contains a subset of the range of the delete, the LSE is split in two, so that the range of one of the new LSEs is contained in the range of the delete, and the intersection of the range of the other new LSE with the range of the delete is empty. The first new LSE gets the entry to the delete list. The second is left alone. If the range of the delete is a proper subset of

the range of the LSE, and the sequence number of at least one record pointed to by the LSE is included in the range of the delete, the LSE is split into two or three new LSEs, depending on whether the range of the delete is or is not coterminous with the range of the LSE. Again, it is split so that one of the new LSEs is coterminous with the range of the delete, and the range of the other one or two LSEs are disjoint from the range of the delete. The new LSE whose range is coterminous with the range of the delete receives the addition to its delete list. If no record pointed to by the LSE has a sequence number within the range of the delete, no action is taken, and the user is informed that no card images were deleted.

Addition of a card image is more complex. This is due to the ability of the system to effect (apparent) replacement of card images without being separately requested to delete the old card image first. Since cards that are being added are simply added to the overflow file in the order they are read in (provided they are in proper sequence), the address of the card image is passed to the data algorithm along with the sequence number.

Assume first that the image to be added to the file has a sequence number which does not appear in the revision being modified. Then one of three things must happen. Either there is a data LSE which is being created for the new revision to which the record may be added, or there is an active data LSE whose range contains the sequence number of the new record, or neither of the first two cases has occurred. In the first case the group size of the LSE is increased by one. In the second case the old LSE is split in two and a new LSE, containing the new record is formed between the halves of the old LSE. In the third case a new LSE is formed.

If the new record has the same sequence number as a record in the revision being modified, then conditions one and three above cannot occur. Thus either the new sequence number equals the first sequence number of the records pointed to by the LSE, or it equals the last such sequence number, or it equals an interior sequence number. In the first two instances, the old LSE is split in two, so that the record whose sequence number is matched is placed alone in an LSE. This LSE then gets an entry to its deletion list. Then a new LSE is formed in front of it to point to the new record. If the sequence number of the record matches an interior sequence number of the range of the LSE, then the LSE is split into three LSE's such that the middle one points to only the record whose sequence number in the revision being being modified is matched by the record to be added. This LSE receives an entry to its delete list. Between the first two of the three LSEs a new LSE is created which points to the record being added.

If all revisions of a member or version are being deleted, it is only necessary to flag the name LSE for a delete. This is of course preceded by a check to see that the member or version has no descendants. If only one revision

is being deleted the data LSE's are read to determine if they were inserted by this revision, or if this revision is on the deletion list of the LSE. In the former case the LSE is physically deleted from the directory. In the latter case the revision number is removed from the deletion list. This is to remove the effects of the revision on the data LSE.

3. THE ASSEMBLER

3.1 Overview

The assembler is called by the catalog procedure AGCASM. The task is specified by the director ASSEMBLE. It is a two-pass assembler. That is, it goes through a program twice, processing each card sequentially. There is an intermediary editing and bookkeeping step between passes called pass 1.5. Pass 2 is followed by a step called pass 3 which does final editing and I/O operations. Output from the assembler consists of certain on-line printout which contains information on the various program steps, an assembly listing which is printed from the GAPOUT tape, and an object module on disk pack AGCB02.

3.2 Pass 1

Pass 1 requests the program named on the ASSEMBLE director from AGCONVRS. The request is issued for one card image at a time. Pass 1 completes its processing of each card before it requests the next. Pass 1 produces 88 bytes of output from each 80 byte card image.

This output consists of a duplicate card image, 80 bytes, preceded by eight bytes of additional information. The first byte of this is the "opflag." It may take on the values 10, 20, 40, 60 in hexadecimal. These values indicate respectively numeric tag, symbolic tag, illegal location field and tag table full when encountered. The latter of course will terminate the assembly.

The second byte is "opaux." It may take on the EBCDIC characters F, T, U, B, M and O. These indicate respectively illegal sign anywhere, no room, undefined, badly defined, multiply defined and oversize symbol in address field. Some other cusses are also indicated during pass 1. Bytes three and four contain a tentative fifteen bit AGC word. If the card contained a basic op code, pass 1, through a table look-up, places the binary machine instruction in the word. If the card contained something other than a basic instruction other action must be taken. Bytes five and six contain the number of additional words reserved by that word. Bytes seven and eight contain the value of the location counter.

Pass 1 also constructs the symbol table. This is used most heavily by pass 2 to translate symbolic addresses and tags into binary memory information.

Pass 1 initiates processing of EBANK, SBANK, ERASE, MEMORY, EQUALS, and =. SUBRO is processed by placing the information in the address field in a special table. This table is referred to at the end of (MAIN). The entries are used to generate requests to AGCONVRS for the subroutines name in the address field of the SUBRO cards. If something other than a basic instruction or one of the assembler instructions discussed above is encountered, pass 1 identifies the type of statement on the card, and places this information in bytes two and three of the output word for pass 2 to use.

Pass 1 output is written on a scratch disk. It contains 88 bytes of output for each 80 byte card image input. It also contains a partially filled symbol table (definition and health), and a rough form of the memory type and availability display.

It should be noted that card images are sentence read. That is, a blank is the signal for the end of a field. This leads to some strange results in certain special cases when the syntax interpreter tries to operate. For example

```
1          25          40
61533      TIME        INSERTION OF NEW
```

would produce the error message

"uninterpretable word in director."

This is because the first three letters in the third field are INSERT, and the sentence reader thinks it has discovered an INSERT card. It then looks at field two, and sees TIME, which does not match either BEGIN or END, the two legal possibilities. Note that if by really odd circumstance BEGIN had been the tag in the tag field, the assembler would have been really led astray.

3.3 Pass 1.5

Pass 1.5 edits the output from pass 1. It does not process card images, but performs tasks which facilitate further processing of the data. It does three things. First it changes the symbol table from a threaded list to a geographic list. The purpose of this is to save time. It is faster to use a threaded list during pass 1 and a geographic list during pass 2. The time saved more than makes up for the time used in conversion. Pass 1.5 also processes the reservation and conflict tables. The two main tasks are to order the tables and delete multiple entries. Finally, pass 1.5 resolves threads of symbols and locations created by EQUALS or =. Again, this processing collects information and discards redundant material so as to form unique chains of entries.

3.4 Pass 2

Pass 2 operates on the output of pass 1.5. Thus, in addition to card images it has available the symbol table, the reservation and conflict tables, and the eight bytes of information generated by pass 1 for each line of coding.

Pass 2 completes the binary AGC word. It assigns binary equivalents of interpretive coding and completes the addresses. It also processes those assembler instructions which pass 1 could not. These are the address constant generators (1DNADR, 2BCADR, 2CADR, 2FCADR, MINUS, PLUS, ADRES, BBCON, CADR, DNCHAN, ONPTR, ECADR, FCADR, GENADR and REMADR), together with COUNT (* $$/XXXXX). Processing of BANKSUM is initiated by pass 2. Pass 2 also generates the rest of the cusses, looks up the appropriate message in a table for cusses generated in pass 1 and pass 2, and writes them into the GAPOUT tape.

Pass 2 writes all of the GAPOUT tape, with the exception of the various tables printed at the end of the listing. Pass 2 writes the binary memory information on a drum. The data is geographically ordered within paragraphs, although the paragraphs are not necessarily in geographic order.

3.5 Pass 3

Input to pass 3 is the output from pass 2. Pass 3 places the paragraphs in geographic order, and inserts proper header records. It writes the ordered and labeled object module on the disk pack AGC BO2, together with the symbol table. This is not an object module for the 360, but for the AGC. The object module which is stored could not be converted to a systems/360 load module, and thus, could not be run on the 360. The simulator (which is a systems/360 program) uses the AGC object module, together with the symbol table, as a set of input parameters to be used at execution time. The use of the symbol table allows symbolic reference to AGC locations during simulation. The AGC object module is an exact map of core of a mission peculiar version of the AGC. If an assembly is requested that already exists on AGC BO2, then all steps are done except to overwrite on AGC BO2.

4. USE OF GAP

4.1 JCL

Having at last reached the section most of my readers will turn to first, I must insert a plea to at least read the other sections while waiting for your first run to be returned. That way you will have something satisfyingly legitimate to do, and might even be better prepared to understand why your run bombed out. I do not have doubts about the accuracy of the instructions I am about to give: as a consumer of manuals, I have come to learn that meaning changes subtly (yet vitally) with the first few experiences.

No matter which facet of GAP one wishes to use, the first problem is to get on the air. As brought out earlier, this necessitates using 360/JCL. The particular JCL statements that are used depend, of course, on the nature of the job to be done.

There are eight types of JCL statements. Only four out of the eight are necessary to use the GAP system. The need for the other four is pretty well eliminated by the great flexibility of the GAP system.

The four necessary types are the job statement, the execute statement, the data definition statement, and the delimiter statement. All four have special symbols in card columns one and two. Each one has one to four fields. The fields must appear in a predetermined order, but do not need to follow a rigid format. The end of a field is indicated by one or more blank columns with the exception of the comments field which is sentence read. The first field in a JCL statement is the name field, the second is the operation field, the third is the operand field, and the fourth is the comments field. Because of the use of blanks to indicate the end of a field, the name in the name field must be contiguous with the special characters in cc. 1 and 2.

The first JCL statement is the priority card. This is peculiar to this institution: priority is usually a parameter on the JOB card. The card has the form:

```
/*PRIO NN
```

where nn ranges from 01 to 13. APOLLO work is done on priorities 02 through 09, with larger numbers indicating both greater urgency and shorter execution time. The actual system is posted throughout the Laboratory. The priority statement is punched on a special orange card. Pre-punched priority cards are available in the computer rooms.

The second JCL statement is the JOB card. It has the form

```
//JOBXNAME JOB NNNN,NAME.F PROBNAME
```

where jobxname is eight alphameric characters. (Further information on jobnames may be found on p. 14 of IBM form C28-6539-7: Job Control Language.) The jobname is followed by one or more blanks. Next is the word JOB followed by one or more blanks. This is followed by accounting information. First is a four digit account number. These are available from group leaders. Next comes the last name of the programer and his first initial, separated by a period. This name must be on the list of names that is accessible to the computer. Group Leaders are responsible for seeing that names are added. Last is the problem name. It is used to analyze machine use to plan for future needs. The only blanks that should appear are those that are indicated.

The second JCL statement is the execute card. This has the basic form

```
// EXEC AGCASM
```

EXEC is a command to the system to execute the catalog procedure AGCASM. The procedure consists of a set of JCL statements. These appear on the output marked by XX, rather than // or /*. The procedure specifies data sets and the devices on which they are located and contains further execute statements. These execute statements cause the execution of programs which control AGCONVRS and the various programs which make up the assembler.

If the user wishes to obtain a listing of the names of the programs whose source decks are stored under AGCONVRS, together with the revision number information necessary to trace ancestry of a program, he must code the parameter DIR=0. In order to change the maximum permissible time for a job step to run, the parameter TIME.stepname=(m,s) is coded, where m is minutes and s is seconds. There is an upper limit on the number of minutes which may be specified, but this limit is quite large. The parameter may also be coded TIME.stepname=(m): if the parameter is not coded TIME.UA=15, TIME.DIR=5, are the default options. The time limit of 2 minutes on steps 3 and 4 are set by the operating system. These limits are generous, and need not be modified. It is also possible to request that the assembly listing be directed to the on-line printer, rather than being first written on a GAPOUT tape. The parameter is

```
ASMLIST='SYSOUT=A'
```

SYSOUT=A is, of course, the standard JCL parameter setting for printed output. It is best if this option is reserved for short assemblies, such as segments. This is to optimize the use of I/O facilities.

Most programs are stored on GAP001. Some, however, are on GAP002 and GAP003. If the user's program is on either of the latter disk packs, it is necessary to code FILE=GAP00k, k = 2 or 3 as appropriate. If FILE is not coded, FILE=GAP001 is the default option.

There is one last parameter which can be useful in certain circumstances. It will be recalled that AGCONVRS is re-enterable, while the assembler is not. This means that if one wants to do more than one assembly, one must execute the catalog procedure AGCASM more than once (rather than just adding * ASSEMBLE directors, as can be done with AGCONVRS directors). The usual way to do this is to submit a separate job for each assembly. If the assemblies are truly independent, this is the best procedure. It allows higher priority and assures that the remaining jobs are not affected by the failure of one. However, if one is modifying a program several times, with assemblies of more than the final revision, it is necessary that the modifications proceed in sequence. In a system with MVT, this is not possible to insure with separate jobs without waiting for output, or issuing special instructions to the operator. Both of these methods have drawbacks and can easily be circumvented by submitting a job with multiple job steps. The set-up of a deck with multiple jobs will be illustrated later, but for now, note that it may be desirable to insure that later assemblies are attempted even if an earlier assembly terminates abnormally. This is done by coding COND=EVEN on the second and subsequent EXEC cards. This tells the system to execute the current job step even if a previous job step terminated abnormally. This seems to allow the possibility of a modification step failing, with the result that prerequisite coding would not be present when cards are later added to the file, thus giving the program a form neither anticipated nor desired. In fact, what happens is that when a modify step fails, the GAP system deletes any assembly that follows it. Since it is the GAP system, and not O/S, COND=EVEN does not affect this kind of cancellation. For the record, it is possible to have a non-crash system failure which will fool the GAP system. This is a rare enough occurrence so that for practical purposes, it can be ignored. The same conditions which cause the assembly to be omitted will cause all future modify requests to be rejected. Thus, except for one type of rare case, the system is failsafe.

The next JCL card is the data definition statement. It has the following form.

```
//IIA.SYSIN DD *
```

UA is a job step name within the catalog procedure AGCASM. SYSIN is the data set name which step UA uses to refer to the input data. As explained earlier, DD says this is a data definition card, and the * in the operand field says that the data follows in the input stream.

Following the DD card are the non-JCL cards, e.g., directors or detail cards. After the last card in the job step which is data for AGCONVRS, one places the delimiter card, /*.

4.2 Directors and Subdirectors for AGCONVRS

Once the user has gotten GAP on the air, he must specify to the GAP system which tasks he wishes performed, and which files he wants them performed on. The first step in this process is the director card. All operations in the GAP system are in the category of one or another director. In many cases, additional specification is needed or desired. In order of decreasing generality, this is provided by subdirectors, acceptors, various merge control cards and detail cards. Some of these types are used only with AGCONVRS.

The following are the valid directors for AGCONVRS, together with their necessary subdirectors.

```
*   CREATE AGC PROGRAM progname BY programer
*   CREATE REVISION n OF AGC PROGRAM progname BY programer
*  ⎧CREATE AGC VERSION progname BY programer
S  ⎩FROM REVISION n OF AGC PROGRAM progname BY programer
*  ⎧CREATE REVISION m OF AGC VERSION progname BY programer
S  ⎩FROM REVISION n OF AGC PROGRAM progname BY programer
*   MODIFY REVISION n OF AGC PROGRAM progname BY programer
*   DELETE REVISION n OF AGC PROGRAM progname BY programer
*   DELETE REVISION ALL OF AGC PROGRAM progname BY programer
*   PRINT REVISION n OF AGC PROGRAM progname BY programer
*   PUNCH REVISION n OF AGC PROGRAM progname BY programer
*   PRINT* REVISION n OF AGC PROGRAM progname BY programer
*   PUNCH* REVISION n OF AGC PROGRAM progname BY programer
```

In all cases the asterisk or S is in column one, column two is blank, and all words are separated by blanks. Words written in lower case are variables. "m" or "n" are replaced by integers, "progname" by eight or fewer alphameric characters, and "programer" by sixteen or fewer alphameric characters which may contain embedded blanks. "programmer" is the only word which may contain embedded blanks. The word PROGRAM can be arbitrarily and capriciously replaced with SUBROUTINE, SEGMENT or VERSION. The only place where use of these words is restricted is in the director CREATE. When one is creating FROM, one must create a VERSION.

Once a file is given a name at creation time, it must be referred to by that name. Conversely, once a name has been used for one file, it may not be used for another. The name of the programer, however, is written into the file each time the file is referred to. This means that each time the programer uses a different name, that name will appear in the directory listing. Prior names may be retrieved by using the PRINT* director. Thus, this feature may be used to record the programer or group responsible for the latest revision.

CREATE The CREATE director has two rather different functions. The first is to originate files of card images directly from cards. The first two forms given do this. When REVISION n is not specified, the revision number assigned is one. The second function is to make available two or more independent versions of the same file. This is done using the director CREATE together with the subdirector FROM. It is also possible to add detail cards during the creation. The cards added become part of the new version. Only one revision of the new version is made in either case. The method of adding detail cards is the same as with the director MODIFY.

MODIFY The director MODIFY is used to add, replace or delete cards within a given file. This may be done within existing log sections or entire log sections may be added or deleted. Each time the MODIFY director is used with a given file, the revision number is incremented by one. MODIFY may be applied to the last revision only.

DELETE The director DELETE can be used either to delete the latest revision of a given file or to delete all revisions of a given file. These deletions do not remove the images of detail cards from the file. They modify the directory to make the file inaccessible. If revision n of a given file has been deleted and revision n-1 updated, only the latest (chronological) revision n is available. It is not possible to delete any revision which has descendants.

PRINT,PUNCH The print and punch directors are used in the same way. They are used to obtain printed or punched output of the detail cards of a file. It is possible to restrict the output to specified log sections.

PRINT*,PUNCH* These directors will print or punch those cards that caused revision n to be different from revision n-1. This includes "programer" and merge control cards.

4.3 Merge Control Cards

Some directors allow, and some require, further specification of the task to be performed. The subdirector FROM has already been presented. This is used only with the director CREATE and so may be considered something of a special case. There is a further class of instructions; the merge control cards. The most general of these are the acceptor cards. They take the following forms.

=LOG	log section name
=LOGADD	log section name
=LOGDEL	log section name

"log section name" is a sixteen character alphameric variable which may contain embedded blanks. These cards must appear in the order the log sections appear.

=LOGDEL can be used only during modification. It is the only card necessary to delete the entire log section named. Up to ten log sections may be deleted under any one director card.

=LOGADD can be used only in modification. It must be preceded by an =LOG card with the name of the existing log section which the user wishes the new log section to follow. Otherwise, the new log section will appear as the first one in the file. Detail cards and merge control cards may be included under the =LOG card used to place the new log section, but only detail cards may be included under the =LOGADD card.

=LOG can be used with all directors except DELETE. It has three meanings. When used with PRINT, PUNCH, PRINT* or PUNCH*, it indicates a restriction of the output to those log sections named. It is used to place a new log section within a file. Finally,, when used with CREATE or MODIFY, it ordinarily specifies which log section the detail cards following it will be added to.

The use of either CREATE (with subdirector) or MODIFY together with an =LOG card primes AGCONVRS to merge cards into the file named on the director. The reason that the acceptor card is necessary is that the namelist of the directory contains "progname-log section name" pairs, and thus AGCONVRS needs both to refer to the stored data. All detail cards with proper sequence numbers following the acceptor card will be placed in the file. If the number on a detail card is not matched by the sequence number of a card image already in the log section of the file, then the card image is simply added to the file. If a card image with the same sequence number already exists in the log section, then the new card replaces the old card in the file. It does not matter if the old card and the new card have the same specification in column one: the replacement occurs in all cases. More than one =LOG card may be included under one director.

This method of adding coding, together with the DELETE merge control card described below, is sufficient for most changes to a program. The additional merge control cards do provide more power. They also allow for far worse mistakes.

There are seven further merge control cards. They are

```
1              8                  18          25
b n n n n n n b ------------- DELETE
b n n n n n n b ------------- DELETEb THROUGH b m m m m m m m
b n n n n n n b CHANGE b b b CARDNS b TO b m m m m m m m
b n n n n n n b PRESERVE b CARDNS
b n n n n n n b BEGIN b b b b INSERT
b n n n n n n b BEGIN b b b b INSERT b WITH b m m m m m m m
                  END              INSERT
```

These cards must be preceded by an =LOG card. They may be used under any MODIFY director, or under a CREATE-FROM pair. They may not be used at file creation time. They must be in proper order by sequence number. They may not be used under =LOGADD.

The DELETE card makes the specified card image inaccessible to the new and subsequent revisions. The DELETE THROUGH card performs the same operation on images nnnnnnnn through (including) mmmmmmm. It is not necessary that there exist card images with all possible sequence numbers, or even that there exist cards with the sequence numbers that specify the range. If matches are not found, this information will be printed out.

Remember that the standard sequence number increment is 100 in AGCONVRS. This means that the system "expects" only cards with sequence numbers nnnn00, but it will search for all others. Thus, the only ones it reports as non-existent are those with sequence numbers of the form nnnnnn, nnnnnn +100, and so on. It will find and delete all, however.

The increment of 100 was chosen as an aid to program modification since it leaves 99 numbers free between each pair of cards. This is usually sufficient, but not always. There are two ways to take care of the exceptional case. One may CHANGE CARDNS and then add cards in the ordinary way, or one may use the BEGIN INSERT option.

CHANGE CARDNS renumbers card nnnnnn, making it card mmmmmm. It then takes cards sequentially and numbers them mmmmmm+100, mmmmmm+200, and so on, until either the end of the log section, or another merge control card which causes or stops renumbering is encountered. Usually

the new initial number is coded mmmm. AGCONVRS interprets this as the four high order digits. Once renumbering has been performed, there is usually enough space to insert the cards.

More logical, however, is to use BEGIN INSERT. This allows the insertion of any number of new cards into the coding. In fact, the best way is to include

```
2       9          18      25
0001    CHANGE     CARDNS TO 0001
```

at the beginning of the log section. Then any number of inserts, substitutions or deletions may be made within the log section, and the numbering will retain the form 000100, 000200, ---, nnnn00, ---.

To insert any number of cards at any point in the coding, use

```
NNNNNN BEGIN      INSERT
```

The number nnnnnn may either be free or may already be assigned to a card image. In the former case, there is no problem; the cards following the insert card are placed in the file and assigned sequence numbers nnnnnn, nnnnnn+200, nnnnnn+200, ---. The cards previously in the file which had sequence numbers higher than nnnnnn are renumbered similarly, starting with the sequence number assigned the last card in the insert plus 100. If the number nnnnnn is not free, then it is necessary to consider whether or not the card with sequence number nnnnnn is to be replaced (by the first card following the insert card). If it is not to be replaced, then it is necessary that the first card following the insert card be an exact duplicate of the card with sequence number nnnnnn. If card nnnnnn is to be replaced, then of course, the first card following the insert card must be the card that it is intended to replace card nnnnnn with.

The end of the group of cards that is to be inserted at nnnnnn is marked by

```
9         18
END       INSERT
```

END INSERT does not turn off renumbering. Its purpose is to indicate that the detail cards which follow it are to be treated individually by AGCONVRS. If another insert card immediately follows the detail cards of the previous insert, an end insert card is not needed.

Cards following an insert card do not need to be numbered. Any sequence numbers will be ignored. It is necessary to have the appropriate character punched in column one (e.g., a remarks card must have the R punched).

It is also possible to force the sequence number of the first card of the insert to any desired value, provided that value is greater than the sequence number of the card the insert is to follow. This is done by coding

```
2       9          18        25
NNNNNN  BEGIN      INSERT WITH MMMMMM
```

Usually the number selected as the sequence number of the first card of the insert will have the form mmmm00.

4.4 Assembler Instructions

The assembler must be taken into consideration during three phases of program development. First, during coding, certain instructions to the assembler must be included. Their purpose is to enable the assembler to assign addresses and locations at assembly time, and to check that the programer does not violate certain addressing requirements. These instructions are local in effect, in that their influence extends at most over a log section. Often there are several such instructions within a log section. There are also assembler instructions which generate constants, reserve erasable locations, equate symbols and generate addresses. These are included with the coding in appropriate places.

The second function of the assembler which the programer must provide instructions for is the collecting of subroutines at assembly time. This is done by including cards of the form

```
2                  18        25
NNNNNN             SUBRO     SUBNAME
```

in the main section of the program. Subroutines are simply added on at the end of the main program in the order in which their SUBRO cards appeared in (MAIN). SUBRO cards may be scattered throughout the main program, since at assembly time they are used to construct a table which is then used after PASS1GEN has completely read (MAIN) to issue calls to AGCONVRS for the cards in the subroutines. Let it be emphasized again that SUBROUTINES in AGC coding are simply housekeeping devices. Thus, it is perfectly natural that the assembler simply strings the subroutines to (MAIN).

The third way in which the programer must instruct the assembler is to cause the assembler to carry out its entire set of functions. That is, the programer must make up a deck which causes the assembler program to execute. Again, the deck will consist of JCL (as specified in 4.1), plus certain directors and subdirectors peculiar to the assembler.

The various forms of the assembler director are as follows:

* ASSEMBLE REVISION n OF AGC PROGRAM progname BY programer
* ASSEMBLE REVISION n OF AGC SUBROUTINE subname BY programer
* S { ASSEMBLE REVISION n OF AGC SEGMENT segname BY programer
  S { WITH REVISION n OF AGC PROGRAM progname BY programer

In all cases, the file of card images indicated in the director card is assembled. In the case of a program with SUBRO cards, those subroutines are assembled as part of the program. The segment assembly requires that the program named on the subdirector has already been assembled, and that the object module is available on AGCB02. The segment assembly uses the symbol table of the program in order to determine definitions, conflict and so on.

By substituting # for * on a director, one can assemble directly from cards in the input stream. The input cards are not stored. The object module is stored on AGCB02 as usual.

4.5 Examples and Stray Hints

The foregoing presentation has been somewhat abstract and discrete. This section will provide examples of the use of the GAP system in the form of sample decks and the output associated with each. The idea is to indicate some of the possibilities, rather than to demonstrate exactly how to use the system. A pretty fair rule is: if this manual does not inveigh against it, or if your common sense does not tell you it is foolish, then the scheme you are contemplating will probably work, and almost certainly will not get you in serious trouble. The reason that this cavalier statement can be made is that all error conditions that are caused by improper control cards can be recovered from by using only control cards for the GAP system; no additional programs or data sets are necessary.

In general, most sequences of tasks (directors) are valid provided the individual director is valid. Further, all sequences of merge control cards are valid. It is not valid to telescope tasks, since the presence of a control card in the input stream would be interpreted as the end of the old task and the start of the new. It is not valid to include acceptor cards or merge control cards in the creation of a new file from cards. It is not valid to add or delete log sections during version creation (this is an as yet unresolved anomaly in the GAP system). It is not valid to have more than one assembly per job step. At most ten log sections may be added or deleted during any update. Other invalid usages are recognized by the system as mistakes, and some message is printed.

The output generated by a GAP job comes in two parts. The first part is the on-line output, the second is printed from the GAPOUT tape.

The on line output contains information on the person whose name appeared on the JOB card, the JCL the input deck contained together with the JCL in the catalog procedure AGCASM, and various OS/360 messages and codes. Next comes reports on the outcome of the tasks requested of the GAP system. These will include reasons why a task was impossible, and also printout of cards added or deleted, and the first hundred or so cussed lines of an assembly. It also contains a section on assembly evaluation. If a directory listing was requested this comes next, followed by the list of assembled programs resident on AGCB02 (the VTOC list). The final part of the normal on-line output is the accounting information.

The printout of the GAPOUT tape consists of a listing of the assembled program together with several tables containing information on the program. The program is divided into (MAIN) and SUBROUTINE sections, and within these into L (log) sections.

At the top of each page of the printout from the GAPOUT tape is reproduced the assembly director which requested the assembly. Also at the top of the page are the data and time the assembly was made, (MAIN) or SUBROUTINE and revision number and page. The next line contains L log name, the page number within the log section (user's page no.), and current setting of EBANK and SBANK.

Following this two line heading are the detail cards. Column 1 contains an R, A, P or blank. Columns 2-7 contain the sequence number. If there is a P or R in column 1, the card is simply printed out. If there is an A in column 1, then the card is printed starting in column 49. If column 1 is blank, then columns 9-26 contain the number of the reference to the tag in the address field, and the page of last reference to that tag. Columns 30-36 contain bank and location within bank information. Bank information is in octal, while the within bank location is in octal. Columns 39-46 contain either another address if the operand is one which is reserving more than one location, or an octal representation of the contents of the AGC word at that location. The tag field starts in column 50, the op code starts in column 59 and the address field starts in column 66. Columns 82 onward are used for remarks.

Since there are six octal positions available, and only five are necessary to represent an AGC word, some explanation is in order. For the separate representation of the operand and address portion of the word it is necessary to represent both the 4 bit operand and the 10 bit address. The minimum number of octal places for the first is two, and for the second is four. Those instructions which can be represented by three bits print out only a single place octal number, and the full address is printed out. If the instruction requires 4 bits, then the instruction is

printed out in two octal places. In this case if the address only uses 9 bits only three octal places are used, while if more than 9 bits are used for the address the op code is incremented, an apostrophe is placed in column 44, and only three octal places are printed out. Note that 2777 is the highest displacement used, so that it is impossible for the op code to be incremented by more than two.

As an example, consider these two possibilities;

```
54,003        TS
55,400        TS
```

In binary these would be

```
011   100   000   000   011
 5     4     0     0     3
011   101   100   000   000
 5     5     4     0     0
```

in both cases the op code is 0111.

Other cases are decoded analogously.

Following the actual program are several tables containing information about the program. In sequence these are the symbol table, the undefined symbol table, the unreferenced symbol table, erasable & equals cross-reference table, memory type and availability display, count data (from COUNT and COUNT* assembler operations), paragraphs generated (this includes module, side sense line and wire information), octal listing, occupied locations table and a list of the subroutines used.

The following printout is selected from several runs. It should illustrate fairly well the kinds of output to be expected. The various tables for the AGC program are not included, since they are both self-explanatory, and of no use until the programmer has a certain expertise.

JOB AC76626

******** 00019 SEPT. 2,1969 ******** HELP FOR YOUR COMPUTER PROBLEMS.

DCG STAFF MEMBERS ARE AVAILABLE TO HELP YOU WITH YOUR COMPUTER PROBLEMS:

MARTHA PENNELL	GENERAL CONSULTING CATALOGED PROCEDURES FORTRAN, PL/I, BAL LIPSVC, FORMAC, CRBF PUBLISHER	IL7-116 X821-101
DAVE LYNCH	MAC, MAC-TO-FORTRAN	IL7-116B X821-447
BETTY SILVER	CRBE, SYSTEM	IL7-121 X821-104
NANCY CLARK (MORNINGS ONLY)	MAC, FORTRAN, PLOTTING PL/I	IL7-149 C X821-585
PETE ROBERTS	LIPSVC	IL7-125 X821-191
DAVE LATIMER	CAP	IL7-121 X821-104
RUSS FANEUF	FILLIP	IL7-119 X821-151
ANN HATHAWAY	MAC, FORTRAN, CRBE	IL7-149C X821-585
BILL WHITTEMORE	PL/I, LIPSVC	IL7-149C X821-585
ECE ECUCHER	TAPE STORAGE ACCOUNTING ADMINISTRATION	IL7-151D X821-234
ERNIE SABINE	SYSTEM, PACK MAINTENANCE	IL7-129 X821-292
BARBARA DeBOER	PUBLISHER, MANUALS, MEMOS	IL7-149B X821-231
MARK KATZ	PL/I, CRBE	IL7-129 X821-292

********** **********

******** 00018 AUGUST 13, 1969 ******** DEMISE OF DCG.MORSEMAC, DCG.MORSELIB, ETC.

EFFECTIVE IMMEDIATELY DCG.MORSEMAC HAS BEEN RENAMED DCG.SYSMAC. AS OF 8/13/69 DCG.MORSELIB, DCG.MORSEOBJ, DCG.MORSEMOD HAVE BEEN RENAMED DCG.SYSLIB, DCG.SYSOBJ, AND DCG.SYSMOD. PROCEDURES 'RFMA', 'RFMAL', 'RFMUA', AND 'RFMUAL' HAVE BEEN CHANGED TO 'DCGA', 'DCGAL', 'DCGUA', AND 'DCGUAL' RESPECTIVELY.

********** **********

```
*****************************************
*                                       *
*         JOB: A076626                  *
*     PROBLEM: 0375                     *
* PROGRAMMER: LANDY.S                   *
*       ROOM: IL7  -2308     MS# 64     *
*  EXTENSION: 1260                      *
* HOME PHONE: 861-8854                  *
*                                       *
*****************************************
```

```
// EXEC AGCASM,ASMLIST='SYSOUT=A'                                          X0001000
XX  PROC  ASMLIST='UNIT=(TAPE9,,DEFER),LABEL=(,NL),VOLUME=SER=GAPOUT',     X0002000
XX            FILE=GAP001,DIR=1                                            0003000
XXUA        EXEC   PGM=MACFETCH,PARM=AGCASM,REGION=450K,TIME=15            0004000
XXMACPAC    DD DSNAME=SYS1.USERLIB,DISP=SHR                                0005000
//SYSPRINT DD SYSOUT=(A,,005)                                              0006000
X/SYSPRINT  DD SYSOUT=A                                                    0007000
XXSYSOUT    DD &ASMLIST                                                    0008000
XXSYSUDUMP  DD SYSOUT=A,SPACE=(TRK,(1,50),RLSE)                            0009000
XXSYSPUNCH  DD SYSOUT=B,DCB=(BLKSIZE=1680)                                 0010000
XXDDCARD    DD VOLUME=SER=&FILE,UNIT=2314,DSNAME=AGCPROGM,DISP=OLD         0011000
XXRFDIR  DD  UNIT=2301,SPACE=(TRK,(10,2))                                  0012000
XXRFUT1     DD UNIT=2314,SPACE=(CYL,(2,1)),DCB=BLKSIZE=7280                0013000
XXRFUT2     DD UNIT=(2314,SEP=(RFUT1)),SPACE=(TRK,(1,300))                 0014000
XXTEMPSTOR  DD UNIT=2301,SPACE=(TRK,(26))                                  0015000
XXINTERPAS   DD  DSNAME=SYS1.UT2,DISP=OLD                                  0016000
XXYULBINO1  DD UNIT=2314,VOLUME=SER=AGCB02,SPACE=(CYL,(2,1),RLSE),         X0015000
XX             DISP=(NEW,PASS)                                              0016000
XXYULBINO2  DD UNIT=2314,VOLUME=SER=AGCB02,SPACE=(CYL,(2,1),RLSE),         X0017000
XX             DISP=OLD                                                     0018000
XXYULBINTP  DD VOLUME=SER=GAPBIN,LABEL=(1,NL),                             X0019000
XX             UNIT=(TAPE9,,DEFER),DISP=(,KEEP)                             0020000
//UA.SYSIN DD *
```

09/16/69 03:22:21

```
BEGIN STEP  1: REGION=450K MAXTIME=   15.000 PGM=MACFETCH
IEF236I ALLOC. FOR A076626  UA
IEF237I MACPAC   ON 347
IEF237I SYSPRINT ON 346
IEF237I SYSOUT   ON 346
IEF237I SYSUDUMP ON 346
IEF237I SYSPUNCH ON 346
IEF237I DDCARD   ON 453
IEF237I RFDIR    ON 1C0
IEF237I RFUT1    ON 456
IEF237I RFUT2    ON 341
IEF237I TEMPSTOR ON 1C0
IEF237I INTERPAS ON 237
IEF237I YULBINO1 ON 454
IEF237I YULBINO2 ON 454
IEF237I YULBINTP ON 0C3
IEF237I SYSIN    ON 456
```

```
L       PCRSE

PC001   ***** FOLLOWING CARD IMAGES, WITHOUT 'R' AND LOC NUMBERS, WERE INPUT
R0002   ***** TO THE GAP ASSEMBLER.  ***************************************

R0003           SETLOC  REENTRY         FBANK 25
R0004           BANK                    FIND VACANT LOCATIONS IN FBANK 25
R0005           EBANK=  JMSUM           EBANK 5
R0006           COUNT*  $$/JOHN

R0007   PROGEX CA      MYEB5
R0008           TS      EBANK           SWITCH TO EBANK 5
R0009           CA      JMFOUR          A
R0010           TS      JMSUM           SUM = A
R0011           CA      JMTEN           INITIALIZATION OF LOOP COUNTER
R0012   JMLOOP  TS      JMLPCNT
R0013           CA      JMTWO           B
R0014           ADS     JMSUM           SUM = SUM + B
R0015           CCS     JMLPCNT         TEST LOOP COUNTER AND DECREMENT
R0016           TC      JMLOOP          +,  GO BACK
R0017           TC      ENDOFJOB        +0, END

R0018   JMTWO   DEC     2
R0019   JMFOUR  DEC     4
R0020   JMTEN   DEC     10
R0021   MYEB5   OCT     02400           CONSTANT FOR EBANK 5

R0022           SETLOC  2400            EBANK 5
R0023           BANK
R0024   JMSUM   ERASE   +1              RESERVE 2 LOCATIONS, CALL FIRST JMSUM
R0025   JMLPCNT EQUALS  JMSUM +1        CALL SECOND LOCATION JMPLCNT

R0026   ***** CODE RESULTING FROM THE ABOVE INPUT *****

 0027   REF  1                  25,2000                         SETLOC  REENTRY         FBANK 25
 0028                           25,2000                         BANK                    FIND VACANT LOCATIONS IN FBANK 25
 0029   REF  1                  25,1400                         EBANK=  JMSUM           EBANK 5
 0030   REF  1                                                  COUNT*  $$/JOHN

 0031   REF  1                  25,2000  3 2016 1  PROGEX      CA      MYEB5
 0032   REF  1                  25,2001  54 0003 0              TS      EBANK           SWITCH TO EBANK 5
 0033   REF  1                  25,2002  3 2014 0               CA      JMFOUR          A
 0034   REF  2  LAST    4      25,2003  55 4000 0              TS      JMSUM           SUM = A
 0035   REF  1                  25,2004  3 2015 1               CA      JMTEN           INITIALIZATION OF LOOP COUNTER
 0036   REF  1                  25,2005  55 4001 1  JMLOOP      TS      JMLPCNT
 0037   REF  1                  25,2006  3 2013 1               CA      JMTWO           B
 0038   REF  3  LAST    4      25,2007  27 4000 0              ADS     JMSUM           SUM = SUM + B
 0039   REF  2  LAST    4      25,2010  11 4001 1              CCS     JMLPCNT         TEST LOOP COUNTER AND DECREMENT
 0040   REF  1                  25,2011  0 2005 0               TC      JMLOOP          +,  GO BACK
 0041   REF  1                  25,2012  0 5045 0               TC      ENDOFJOB        +0, END

 0042                           25,2013  00002 0    JMTWO       DEC     2
```

```
0043                        25,2014   00064 0  JMFOUR   DEC     4
0044                        25,2015   00012 1  JMTEN    DEC     10
0045                        25,2016   02400 1  MYEB5    OCT     02400          CONSTANT FOR EBANK 5

0046                        E5,1400                    SETLOC  2400           EBANK 5
0047                        E5,1400                    BANK
0048                        E5,1400   E5,1401  JMSUM    ERASE   +1             RESERVE 2 LOCATIONS, CALL FIRST JMSUM
0049  REF   4  LAST    4   E5,1401             JMLPCNT  EQUALS  JMSUM +1       CALL SECOND LOCATION JMPLCNT
```

```
P0142   EXAMPLE OF SIMPLE INTERPRETIVE PROGRAM

 0143   REF  3  LAST   0   25,2000                          SETLOC  RENTRY
 0144                      25,2063                          BANK
 0145   REF  1             E5,1402                          EBANK=  INMAT1
 0146   REF  3 LAST   6 TO   7?    36    51*                COUNT*  $$/JOHN

 0147   REF  1             25,2063  C 6000 1  MXM          TC      INTPRET
 0148                      25,2064   66201 1                SETPD   SSP
 0149                      25,2065   00001 0                        OD
 0150   REF  1             25,2066   00051 0                        S1
 0151                      25,2067   00002 0                DEC     2
 0152                      25,2070   77770 1                AXT,1
 0153                      25,2071   00006 1                DEC     6
 0154                      25,2072   64743 0  MATLOOP       DLOAD*  PDDL*
 0155   REF  1             25,2073   02447 1                        INMAT2 +18D,1    Z COMPONENT
 0156   REF  2  LAST   9   25,2074   02441 1                        INMAT2 +12D,1    Y COMPONENT
 0157                      25,2075   55523 0                PDDL*   VDEF
 0158   REF  3  LAST   9   25,2076   02433 1                        INMAT2 +6D,1     X COMPONENT
 0159                      25,2077   41406 0                PUSH    PUSH
 0160                      25,2100   77641 1                DCT
 0161   REF  2  LAST   9   25,2101   02403 1                        INMAT1
 0162   REF  1             25,2102   06455 0                STORE   OUTMAT +6D,1
 0163                      25,2103   50375 0                VLOAD   DOT
 0164   REF  3  LAST   9   25,2104   02411 1                        INMAT1 +6D
 0165                      25,2105   77626 0                STADR
 0166   REF  2  LAST   9   25,2106   71314 1                STORE   OUTMAT +12D,1
 0167                      25,2107   50375 0                VLOAD   DOT
 0168   REF  4  LAST   9   25,2110   02417 1                        INMAT1 +12D
 0169                      25,2111   77626 0                STADR
 0170   REF  3  LAST   9   25,2112   71306 1                STORE   OUTMAT +18D,1
 0171                      25,2113   77500 1                TIX,1   EXIT
 0172   REF  1             25,2114   52072 0                        MATLOOP
 0173   REF  3  LAST   6   25,2115  0 5045 0  MXMEND        TC      ENDOFJOB

 0174                      E5,1400                          SETLOC  2400
 0175                      E5,1402                          BANK
 0176                      E5,1402  E5,1423   INMAT1        ERASE   +17D             FIRST INPUT MATRIX
 0177                      E5,1424  E5,1445   INMAT2        ERASE   +17D             SECOND INPUT MATRIX
 0178                      E5,1446  E5,1467   OUTMAT        ERASE   +17D             OUTPUT MATRIX
```

STATISTICAL INFORMATION FOR DATASET AGCPROGM VOLUME=GAP001 18 AUG 69 17:39:49.29 PAGE NO 1

THE NUMBER OF RECORDS IN THE LINEAR PORTION OF THE FILE IS 16186C (71% FULL)

THE NUMBER OF RECORDS IN THE OVERFLOW PORTION OF THE FILE IS 29316 (81% FULL)

THE LINEAR FILE STARTS AT BLOCK 1 AND HAS BEEN ASSIGNED 2560 BLOCKS

THE OVERFLOW FILE STARTS AT BLOCK 2501 AND HAS BEEN ASSIGNED 400 BLOCKS

THE DIRECTORY STARTS AT BLOCK 2901 AND IS USING 671 BLOCKS

THE NUMBER OF PHYSICAL EXTENTS IS CURRENTLY 1

THE BLOCK SIZE (IN BYTES) IS 7280, AND THE LOGICAL RECORD LENGTH (BYTES) IS 80

TABLE OF CONTENTS FOR DATASET AGCPROGM VOLUME=GAPOL1 18 AUG 69 17:39:49.29 PAGE NO 1

PAGE	LATEST REVISION	NAME	REMARKS
286	3	AAPCOM	VERSION OF COMANCHE REVISION 55 AUTHOR=VELLA 15:02 APR. 15,1969
288	5	AAPLUM	VERSION OF LUMINARY REVISION 96 AUTHOR=VELLA 14:36 APR. 18,1969
172	1	ADDRSCHK	AUTHOR=THE MAD LOADER
226	3	AIDJOE	VERSION OF COMAID REVISION 27 AUTHOR=TURNBULL 19:19 MAR. 12,1969
88	5	ALESKIP	VERSION OF SKIPPER REVISION 50 AUTHOR=ALEXA
90	21	ALEXA	VERSION OF COLOSSUS REVISION 224 AUTHOR=SORANT 9:20 DEC. 12,1968
87	1	ALEXADF	VERSION OF KOOLADE REVISION 57 AUTHOR=ALEXA
86	1	ALEXRASE	VERSION OF KILERASE REVISION 74 AUTHOR=ALEXA
242	13	ALIGNCOM	VERSION OF COMANCHE REVISION 51 AUTHOR=P50GROUP-BARNERT 17:17 JUNE 18,1969
407	6	ARTEMIS	VERSION OF COMANCHE REVISION 67 AUTHOR=LATONA 20:45 AUG. 15,1969
250	1	AUXERASE	VERSION OF LUMERASE REVISION 102 AUTHOR=AUXBOX 18:14 MAR. 20,1969
251	2	AUXP20S	VERSION OF LEMP20S REVISION 114 AUTHOR=AUXBOX 17:23 MAR. 24,1969
252	2	AUXP30S	VERSION OF LEMP30S REVISION 102 AUTHOR=AUXBOX 17:18 APR. 9,1969
257	1	BLIND	VERSION OF FLY REVISION 109 AUTHOR=BAT 17:23 MAR. 24,1969
244	1	BOOSTAID	VERSION OF COMAID REVISION 29 AUTHOR=ZELDIN 1:55 MAR. 19,1969
248	1	BOOSTBOS	VERSION OF CHIEFTAN REVISION 26 AUTHOR=ZELDIN 1:55 MAR. 19,1969
345	4	BOOSTCOD	VERSION OF SAVECODE REVISION 5 AUTHOR=CRAMER-ZELDIN 13:46 JUNE 10,1969
249	6	BOOSTCOM	VERSION OF COMANCHE REVISION 51 AUTHOR=CRAMER-ZELDIN 13:46 JUNE 10,1969

INTERNAL REVISION_#	FLAGS B B D	ACTUAL_REVISION_NUMBER
119		45.1(6)
121		45.2(6)
91		46
92		47
93		48
99		49
107		49.1(2)
112		49.2(2)
116		49.3(2)
122		49.4(2)
123		49.5(2)
130		49.6(2)
133		49.7(2)
151		49.8(2)
152		49.9(2)
170		49.10(2)
172		49.11(2)
192		49.12(2)
198		49.13(2)
202		49.14(2)
203		49.15(2)
207		49.15(2).1(1)
213		49.16(2)
219		49.17(2)
220		49.18(2)
221		49.19(2)

DSNAME	SERIAL	SEQ	CREDT	EXPDT	DSO	RECFM	BLKSZ	LRECL	KEY	OP	IRKAL	IRKUS	EX	SECQU	I
AGCPROG.LUMINARY.R117	AGCB02	0001	08/15/69	08/26/69	PS	U	3008	0	0	80	40	38	1	1	C
AGCPROG.LUMINARY.R118	AGCB02	0001	08/20/69	08/22/69	PS	U	3008	0	0	80	40	38	1	1	C
AGCPROG.LUMIRIG.R069	AGCB02	0001	04/01/69	04/01/69	PS	U	3008	0	0	80	20	15	1	1	C
AGCPROG.LUMIRIG.R116	AGCB02	0001	08/12/69	08/24/69	PS	U	3008	0	0	80	20	15	1	1	C
AGCPROG.LUMPERF.R099	AGCB02	0001	06/23/69	06/23/69	PS	U	3008	0	0	80	20	15	1	1	C
AGCPROG.LUMPERF.R116	AGCB02	0001	08/12/69	08/28/69	PS	U	3008	0	0	80	20	15	1	1	C
AGCPROG.LUMYAGS1.R116	AGCB02	0001	08/14/69	08/14/69	PS	U	3008	0	0	80	20	15	1	1	C
AGCPROG.LUM69.R002	AGCB02	0001	03/29/69	05/27/69	PS	U	3008	0	0	80	40	38	1	1	C
AGCPROG.MANCHE45.R002	AGCB02	0001	03/29/69	05/19/69	PS	U	3008	0	0	80	40	39	1	1	C
AGCPROG.MANCHE51.R002	AGCB02	0001	04/24/69	04/25/69	PS	U	3008	0	0	80	41	39	1	1	C
AGCPROG.MERGCHK.R004	AGCB02	0001	05/02/69	05/02/69	PS	U	3008	0	0	80	20	1	1	1	C
AGCPROG.NAVIGATE.R014	AGCB02	0001	08/12/69	09/04/69	PS	U	3008	0	0	80	40	39	2	1	C
AGCPROG.NAVIGATE.R015	AGCB02	0001	08/20/69	08/29/69	PS	U	3008	0	0	80	41	39	1	1	C
AGCPROG.NMNCHE.R001	AGCB02	0001	07/29/69	08/21/69	PS	U	3008	0	0	80	40	38	2	1	C
AGCPROG.NMNCHE.R002	AGCB02	0001	08/21/69	09/09/69	PS	U	3008	0	0	80	40	38	1	1	C
AGCPROG.NMNCHE.R003	AGCB02	0001	09/12/69	09/15/69	PS	U	3008	0	0	80	40	38	1	1	C
AGCPROG.ORBIT.R003	AGCB02	0001	04/11/69	04/11/69	PS	U	3008	0	0	80	41	39	1	1	C
AGCPROG.PIPCRTHC.R008	AGCB02	0001	04/10/69	04/10/69	PS	U	3008	0	0	80	20	15	1	1	C
AGCPROG.PLTEST.R001	AGCB02	0001	05/06/69	05/14/69	PS	U	3008	0	0	80	40	39	1	1	C
AGCPROG.PORTA.R001	AGCB02	0001	05/17/69	05/17/69	PS	U	3008	0	0	80	20	1	1	1	C
AGCPROG.PWRDCOM.R021	AGCB02	0001	07/31/69	08/21/69	PS	U	3008	0	0	80	40	39	1	1	C
AGCPROG.PWRDCOM.R026	AGCB02	0001	08/14/69	08/20/69	PS	U	3008	0	0	80	40	39	1	1	C
AGCPROG.PWRDCOM.R027	AGCB02	0001	08/20/69	08/20/69	PS	U	3008	0	0	80	40	39	1	1	C
AGCPROG.RATEOPTC.R008	AGCB02	0001	09/09/69	09/09/69	PS	U	3008	0	0	80	40	39	2	1	C
AGCPROG.RATEOPTC.R009	AGCB02	0001	09/10/69	09/10/69	PS	U	3008	0	0	80	40	39	2	1	C
AGCPROG.RATEOPTC.R010	AGCB02	0001	09/11/69	09/15/69	PS	U	3008	0	0	80	40	39	2	1	C
AGCPROG.RESCALE2.R002	AGCB02	0001	05/13/69	07/07/69	PS	U	3008	0	0	80	40	39	1	1	C
AGCPROG.RUNEC.R004	AGCB02	0001	04/11/69	05/23/69	PS	U	3008	0	0	80	40	39	1	1	C
AGCPROG.SAVARY.R001	AGCB02	0001	06/15/69	06/15/69	PS	U	3008	0	0	80	40	36	1	1	C
AGCPROG.SAVARY.R002	AGCB02	0001	06/17/69	06/17/69	PS	U	3008	0	0	80	41	38	2	1	C
AGCPROG.SAVARY.R003	AGCB02	0001	08/05/69	08/05/69	PS	U	3008	0	0	80	41	38	1	1	C
AGCPROG.SERVICHE.R025	AGCB02	0001	07/18/69	09/04/69	PS	U	3008	0	0	80	40	38	1	1	C
AGCPROG.SERVICHE.R029	AGCB02	0001	08/14/69	08/14/69	PS	U	3008	0	0	80	41	38	1	1	C
AGCPROG.SERVICHE.R030	AGCB02	0001	09/01/69	09/12/69	PS	U	3008	0	0	80	41	38	1	1	C
AGCPROG.SOLARSPY.R017	AGCB02	0001	07/10/69	07/10/69	PS	U	3008	0	0	80	41	33	2	1	C
AGCPROG.STARTRAK.R019	AGCB02	0001	07/02/69	07/02/69	PS	U	3008	0	0	80	40	38	1	1	C
AGCPROG.STARTRAK.R021	AGCB02	0001	07/10/69	07/10/69	PS	U	3008	0	0	80	40	38	2	1	C
AGCPROG.SUNDANCE.R306	AGCB02	0001	07/31/69	08/30/69	PS	U	3008	0	0	80	41	37	1	1	C
AGCPROG.SUNDISK.282	AGCB02	0001	02/05/69	02/05/69	PS	U	3008	0	0	80	40	37	1	1	C
AGCPROG.THOR.R007	AGCB02	0001	07/03/69	07/03/69	PS	U	3008	0	0	80	40	39	1	1	C
AGCPROG.YAHOO.C05	AGCB02	0001	03/13/69	03/27/69	PS	U	3008	0	0	80	40	39	1	1	C

<<<<< END VTOC 84 DATA SETS >>>>>

```
IEF285I   SYS69259.T005811.SV000.A076626.R0000477      SYSOUT
IEF285I   VOL SER NOS= SYSIC2.
IEF285I                                                KEPT
IEF285I   VOL SER NOS= AGCB02.
END STEP  4: A076626  VTCC              SYSTEM CODE=000,USER CODE=0000          STEP TIME:   0.033 MIN   09/16/69 03:24:33


IEF284I   SYS69259.T005811.RV000.A076626.R0000458      NOT DELETED 8
IEF284I   VOL SER NOS= AGCB02 1.
END JOB: A076626   0375                              LANDY.S              PRTY=02  JOB TIME:   0.538 MIN   09/16/69 03:24:33

BASED ON  DECEMBER, 1968   ACCOUNTING, THIS JOB COST      3.14 DOLLARS.

THIS ACCOUNT BUDGET:  TB=   360.000 MIN,   TL=   193.398 MIN,   TI=     0.000 MIN.
```

5. GAP MANUFACTURING

5.1 Introduction

The manufacturing facility of the GAP system provides a means of converting binary AGC programs stored in direct access storage to a variety of forms and mediums suitable for input to certain external devices and processes. The devices and processes referred to are described in the sections concerned with specific manufacturing tasks. Additionally, GAP Manufacturing has a feature that allows two binary AGC programs to be compared and all differences listed.

Each time an AGC program is manufactured, a record is appended to a permanent data set called the Manufacturing History Data Set. This record includes the identification of the AGC program(s), the date and time of manufacture, what type of manufacturing was done, and a notation as to whether the job was successfully completed or not. In the case of a binary comparison, statistics concerning the number of words and paragraphs that differed and the number(s) of the rope module(s) affected are also included.

The entire contents of the Manufacturing History Data Set may be listed whenever desired by executing the catalogued procedure GAPPMHST. As a matter of practice, this is commonly done after each manufacture by inserting the card

```
//      EXEC      GAPPMHST
```

after the last manufacturing subdirector card in the job control deck. It will be seen that this practice is followed in most of the examples scattered throughout the text.

One other permanent data set, called the Master Deck Sequence Number Data Set, is maintained in conjunction with manufacturing. This data set consists of a single record that contains a six-digit decimal integer called the Master Deck Sequence Number. Whenever a master deck is manufactured, the current Master Deck Sequence Number is retrieved and used as a label. This number is increased by one before restoring it to the Master Deck Sequence Number Data Set. Thus every master deck is provided with a unique identification. Execution of the catalogued procedure GAPPMHST will also list the current contents of the Master Deck Sequence Number Data Set.

Special programs exist for initializing both the Manufacturing History Data Set and the Master Deck Sequence Number Data Set - see Appendix A. A typical page of the Manufacturing History Data Set Listing is shown in Appendix B, along with a list of abbreviations used.

Finally it must be noted that although the modules which perform manufacturing are link-edited to form part of the overlay load module called by the catalog procedure AGCASM, it is not traditional to perform manufactures this way. There are separate catalog procedures which call a load module designed especially for performing manufacturing tasks. This special load module is simpler than that called by AGCASM, and thus saves computer space and time.

5.2 MANUFACTURE Tasks

The following manufacturing tasks may be performed by invoking the correct procedure:

1) PUNCH SYMBOL TABLE
2) PUNCH MASTER DECK
3) PUNCH SYMBOL TABLE AND MASTER DECK
4) PUNCH 36K CORE ROPE SIMULATOR TAPE
5) COMPARE binary AGC program A
   WITH binary AGC program B
6) WRITE PORTAFAM TAPE
7) PUNCH 36K BRAID WIRING TAPE

In reference to tasks 1 through 3 above, the word "PUNCH" is a misnomer, and is retained only for historical reasons; in actuality 80-character card images are written on magnetic tape, and no cards are punched. Task 4 includes the ability to write DIGISTORE tape or magnetic tape as well as to punch paper tape; the desired output medium is specified by the appropriate subdirector card. Tasks 1 through 6 are discussed under separate headings in the following text. Task 7 is available only to members of the Digital Development Group and will not be discussed further. Questions concerning its use should be directed to the Digital Development Group (H. Robert Howie, Jr.).

Any attempt to manufacture a binary AGC program that has been flagged as a bad assembly will result in an abort in Tasks 2, 3, 6, and 7: the manufacture is allowed to proceed in Tasks 1, 4, and 5. In any event, the message !!!!!*****WARNING. WARNING. YOU ARE ATTEMPTING TO PROCESS A BAD ASSEMBLY. WARNING. WARNING. *****!!!!! is printed 100 times in the output listing and the letter "B" is inserted in the manufacturing history record.

The manufacturing output listing may contain error messages or codes. Messages and User Codes are listed in Appendix C along with the appropriate action to be taken. The presence of a System Code other than 000 usually implies that serious system problems exist; in this case a GAP system expert should be consulted (David P. Latimer).

5.3 MANUFACTURE JOB Deck

A binary AGC program is manufactured by submitting for computer run a deck that contains the following cards:

a) a JOB card

b) a JOBLIB card if the output medium is DIGISTORE tape

c) an EXEC ute card that names a catalogued procedure that invokes GAP manufacturing

d) a SYSIN card

e) a Director Card containing the name, revision number and author of the AGC program to be manufactured or compared

f) one or more Subdirector Cards that specify the manufacturing task to be performed, the output medium to be used, the number(s) of the paragraph(s) of the AGC program to be processed, or the identification of the second AGC program in a comparison task.

g) an EXEC ute card to print the Manufacturing History Data Set

h) an EXEC ute card to run the Master Deck Tape Checking and File Duplicating Program if the task is PUNCH MASTER DECK

i) an EXEC ute card to list the PORTAFAM program tape if the task is WRITE PORTAFAM TAPE

j) a job step termination card (/*)

Many examples of job decks that illustrate the use of the cards appear in the following sections.

5.4 Execution of GAP Manufacturing

The GAP System manufacturing facility is invoked by executing either of the catalogued procedures GAPMAGC or GAPMAGCA. This is done by including either of the cards

```
//      EXEC    GAPMAGC    or
//      EXEC    GAPMAGCA
```

in the job deck.

The procedure GAPMAGCA is to be used only when the task is

S PUNCH 36K CORE ROPE SIMULATOR TAPE

and the output medium is 7-TRACK MAGNETIC TAPE. In all the other cases the procedure GAPMAGC should be used.

5.5 Director Card

All tasks use a Director Card to identify the AGC program to be manufactured or compared. This card must be inserted in the job deck immediately following the SYSIN card and must precede all Subdirector Cards. The Director card has the following format:

* {MANUFACTURE / COMPARE} REVISION n OF AGC {PROGRAM / SEGMENT} m BY a

Here n represents a one to three digit revision number, m is a one to eight character program name containing no imbedded blanks, and a is a 16-character author name that may contain as many as 15 imbedded or trailing blanks. Unlike other GAP tasks, the author name must be exactly as placed in the GAP author name file, otherwise the manufacture will be aborted with user code 009. The directive COMPARE is used if a binary comparison is to be performed; MANUFACTURE is used for all other tasks. SEGMENT is used to designate AGC Erasable Programs.

Examples:

```
*    MANUFACTURE REVISION 046 OF AGC PROGRAM COMANCHE BY RESERVATION
*    COMPARE REVISION 02 OF AGC PROGRAM LUM69 BY NASA 2021112-031
*    MANUFACTURE REVISION 069 OF AGC SEGMENT LEMIRIG BY STG
```

5.6 Manufacturing Subdirectors

With the exception of COMPARE, all manufacturing tasks are specified by the first Subdirector Card following the Director Card. Each task is discussed under a separate heading in the following text.

5.6.1 Punch Symbol Table

The subdirector

S PUNCH SYMBOL TABLE

causes Manufacturing to convert the AGC Program Symbol Table to 80-character card images and write them out on magnetic tape. The following actions ensue:

a) A Symbol Table Header Card image, containing the AGC Program identification and a count of the number of following Symbol Table Card images (exclusive of the Symbol Table Trailer Card image), is generated and written on magnetic tape.

b) A Symbol Table Record is retrieved from direct access storage and checked for validity.

c) Each successive set of five symbols and their definitions is treated as follows:

1) The symbol definitions are converted from binary to six-digit zoned octal numbers.
2) The five symbols and their octal definitions are formatted into an 80-character EBCDIC card image.
3) The card image is written on magnetic tape.

d) Steps b) and c) are repeated until the Symbol Table is exhausted. Note that the last card image may contain fewer than five symbols and their definitions.

e) A Symbol Table Trailer Card image, containing the AGC Program identification, is written on the magnetic tape.

Appendix D contains a listing of a specimen Symbol Table.

If an AGC Symbol Table is manufactured by the procedure GAPMAGC, the card images appear on the magnetic tape as 80-character unblocked seven-track even parity BCD records written at a density of 556 BPI. The tape is unlabelled.

An example of a job deck to manufacture a Symbol Table is the following:

```
//COMANCHE JOB 0612,HOPE.J    COMANCHEMFR
//          EXEC     GAPMAGC
//M.SYSIN     DD          *
*         MANUFACTURE REVISION 44 OF AGC PROGRAM COMANCHE BY HOPE
S         PUNCH SYMBOL TABLE
//        EXEC       GAPPMHST
/*
```

An AGC Symbol Table may be manufactured under the procedure GAPMAGCA only if this follows the previous manufacture of a 36K Core Rope Simulator Tape on seven-track magnetic tape. The Symbol Table card images will then comprise the third file on the unlabeled seven-track magnetic tape. In this case the 80-character card images are blocked, twenty-one to a block, and written on the tape as 1680 byte (2240 tape character) odd parity EBCDIC records at a density of 800 BPI.

A job deck representative of this type of Symbol Table manufacture is shown in the section headed PUNCH 36K CORE ROPE SIMULATOR TAPE.

5.6.2 Punch Master Deck

AGC Core Rope Memories are fabricated (by the Raytheon Company) by a process whose initial input is an AGC Master Deck tape. This tape is generated by GAP Manufacturing if the subdirector

S PUNCH MASTER DECK

is present in the job deck. The AGC program named on the Director Card is converted from binary to octal and written on magnetic tape in the form of successive unblocked 80-character card images. This is done as follows:

a) A Master Deck Header Card image is generated, containing the current Master Deck Sequence Number and the date and time of manufacture.

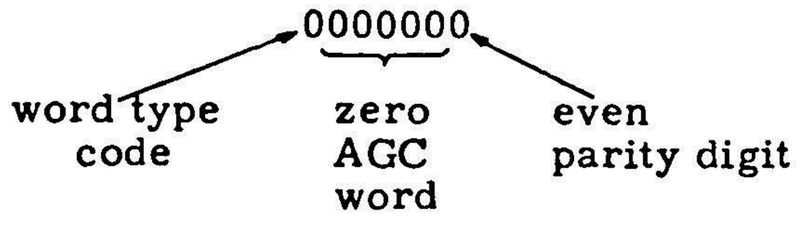
b) An AGC Program Paragraph Record is retrieved from direct access storage and checked for valid identification. The number of the paragraph is checked against the number determined from the Paragraph Presence Indicators.

c) The Paragraph Trailer Card image for the preceding paragraph (or Master Deck Header Card image if this is the first paragraph) is written on magnetic tape after the current paragraph number has been inserted.

d) A Paragraph Header Card image, composed of the Master Deck Sequence Number, the AGC program identification, the paragraph number in octal and the manufacturing location code (see Appendix E) is written on magnetic tape.

e) The 256 words of the paragraph are separated into 32 sets of eight words each. Each eight-word set is treated as follows:

1) The AGC words and their (odd) parity bits are converted from binary to (five-digit) zoned octal numbers plus parity digit. Each word is prefixed with a decimal code digit that indicates the word type (basic instruction, constant, etc. - see Appendix E). Unused AGC words are represented by five zeros, a zero (even!!) parity digit, and a zero word type code. Thus:



2) The eight octal AGC words are imbedded in an 80-character EBCDIC card image. This card image also contains the Master Deck Sequence Number, the octal address of the first AGC word relative to the first word of the paragraph, and the manufacturing location code.

3) The card image is written on magnetic tape.

f) A Paragraph Trailer Card image, containing the number of the paragraph just processed, is generated.

g) Steps b) through f) are repeated until all paragraphs have been processed.

h) The final Paragraph Trailer Card image is converted to a Master Deck Trailer Card image by inserting the words

END OF AGC MASTER DECK

This card image is written on the magnetic tape. Appendix E contains a portion of a listing of an AGC Master Deck Tape along with brief explanations of the Word Type Codes and Manufacturing Location Codes. The decimal digit that runs in column eight of the listing is used for carriage control (spacing) purposes if the listing is produced on an IBM Type 407 Document Originating Machine from a previously punched card deck.

Erasable banks (paragraphs) are treated in the same fashion as fixed paragraphs, except that the Manufacturing Location Code is replaced by the characters EBKn where "n" is the octal bank number. It should be clear that Master Deck Tapes intended for rope manufacture must not contain erasable banks.

Master Deck Tapes must be manufactured by the procedure GAPMAGC. The card images on the resulting magnetic tape appear as 80-character unblocked seven-track even parity BCD records written at a density of 556 BPI. The tape is unlabeled.

A Master Deck Tape may be listed, if necessary, by using the following job deck with the appropriate job card:

```
//LISTMDTP JOB 0612,HOPE.J  PRINT MASTER DECK
//          EXEC    GAPPMDTP
/*
```

5.6.2.1 Releases

Master Deck Tapes intended for release for rope manufacture must be produced in accordance with the specifications in

APOLLO G&N SPECIFICATION
ND 1002377 Rev.
OPERATIONAL PROCEDURE FOR GENERATION OF MASTER DECK TAPE FOR MISSION PROGRAMS.

The following is an example of a job deck to produce a Master Deck Tape that satisfies these specifications:

```
//MSTDKMFG JOB 0614,HOPE,J  MASTER DECK COM 55
//      EXEC   GAPMAGC
//M.SYSIN   DD   *
*        MANUFACTURE REVISION 055 OF AGC PROGRAM COMANCHE BY NASA 2021113-051
S        PUNCH MASTER DECK
//      EXEC      GAPPMHST
//      EXEC      GAPCKMDT
/*
```

The procedure GAPCKMDT in the above deck invokes the Master Deck File Duplicating and Tape Checking Program, a FORTRAN program which does the following:

a) The Master Deck Card images that were written by the first procedure (GAPMAGC) on the magnetic tape are copied back from (the first file of) the tape into a temporary data set in direct access storage.

b) The temporary data set is copied repeatedly to form files two, three and four on the magnetic tape.

c) The tape is rewound and all four tape files are compared successively with the temporary data set to verify the absence of copying errors.

d) Each octal AGC word in the temporary data set is checked for correct parity and all bank check sums are verified.

e) A listing of the AGC Master Deck is made.

f) A page is printed that lists all occupied paragraphs, all vacant paragraphs, the location of the check sum constant, the value of the checksum constant and the value of the checksum for each bank. A sample page is shown in Appendix E.

5.6.3 Punch Symbol Table and Master Deck

An AGC Program Symbol Table followed by a Master Deck will be written on magnetic tape if the subdirector

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

is included in the run deck. The formats of the respective card images are the same as those described in the two previous sections entitled Punch Symbol Table and Punch Master Deck.

The following is an example of a job deck:

```
//MSTANDMD JOB 0641,HOPE.J    SYMBOL TABLE & MASTER DECK
//       EXEC      GAPMAGC
//M.SYSIN     DD      *
*          MANUFACTURE REVISION 055 OF AGC PROGRAM COMANCHE BY HOPE
S          PUNCH SYMBOL TABLE AND MASTER DECK
//       EXEC      GAPPMHST
/*
```

This subdirector must not be used to produce magnetic tapes that will be used to manufacture core ropes.

5.6.4 Punch 36K Core Rope Simulator Tape

The core rope memory of an Apollo Guidance Computer (AGC) may be replaced for testing and debugging purposes in part or in toto by the erasable memory of a device called the 36K Core Rope Simulator (or AGC Monitor). The purposes and use of the AGC Monitor are discussed in the publication E-2026 "Users' Guide to the AGC Monitor (Core Rope Simulator)" by James D. Wood, September 1966, Apollo Guidance, Navigation and Control, MIT Instrumentation Laboratory, along with a brief description of the hardware.

The memory of the 36K Core Rope Simulator may be loaded with the desired AGC program from any of the following sources:

a) an eight-channel punched (paper or mylar) tape;

b) a DIGISTORE (magnetic) tape;

c) the core memory of an SDS 9300 Digital Computer (if the 36K Core Rope Simulator is tied into the Hybrid Facility).

In case c) above, the SDS 9300 Digital Computer memory must have been previously loaded with the AGC program from a seven-track magnetic tape. The (punched, DIGISTORE, seven-track magnetic) tapes referred to in the preceding are known collectively as 36K Core Rope Simulator Tapes.

36K Core Rope Simulator Paper Tapes are also used by certain Apollo subcontractors to load the memories of their Program Analyzer Consoles (PACs).

A 36K Core Rope Simulator Tape of an AGC program will be produced by GAP manufacturing if the first subdirector in the job deck is

```
S       PUNCH 36K CORE ROPE SIMULATOR TAPE
```

This subdirector must be followed by two more subdirectors (their order is inconsequential) which specify

1) the number(s) of the paragraph(s) of the AGC program to be manufactured;

2) the type of output medium to be used (paper tape, DIGISTORE, etc.).

5.6.4.1 PARAGRAPH and Medium Subdirectors

If the entire AGC program (specified on the Manufacturing Director Card) is to be converted to a 36K Core Rope Simulator Tape, the subdirector

```
S       PARAGRAPH ALL
```

must be included in the job deck. For example, a 36K Core Rope Simulator Tape of Revision 44 of AGC Program COMANCHE will result if the following deck is run:

```
//COMANCHE JOB 0341,HOPE.J  COMANCHE MFR
//      EXEC       GAPMAGC
//M.SYSIN    DD    *
*         MANUFACTURE REVISION 044 OF AGC PROGRAM COMANCHE BY NASA 2021113-011
S         PUNCH 36K CORE ROPE SIMULATOR TAPE
S         PARAGRAPH ALL
S         USE BLUE MYLAR TAPE
//      EXEC      GAPPMHST
/*
```

A single paragraph of the AGC program may be converted to a 36K Core Rope Simulator Tape by using the subdirector

```
S       PARAGRAPH Y
```

where y is one to three-digit (octal) paragraph number.

If the subdirector

```
S       PARAGRAPH Y THRU Z
```
or
```
S       PARAGRAPH Y THROUGH Z
```

is used, a 36K Core Rope Simulator Tape will be produced that includes all the paragraphs from y through z inclusive. Here y and z are both one to three-digit (octal) paragraph numbers and y must be less than z. Paragraph numbers 0-7 refer to erasable banks 0-7 respectively. In the case of the two preceding subdirectors, if either paragraph y or any of y-z are not present in the AGC program being manufactured, no 36K Core Rope Simulator Tape is produced, and the manufacturer aborts with user code 24.

It should be carefully noted that all 36K Core Rope Simulator Tapes are manufactured in units of banks (one bank contains four paragraphs) not paragraphs. Thus, if an AGC program does not use certain paragraphs, the vacant paragraphs will still be included in the 36K Core Rope Simulator Tape output if they form part of a bank that contains occupied paragraphs.

If the subdirector

```
S       PARAGRAPH Y
```

is used, one bank composed of paragraph y and three vacant paragraphs form the resulting 36K Core Rope Simulator Tape. Similarly, if the subdirector

```
S       PARAGRAPH Y THRU Z
S       PARAGRAPH Y THROUGH Z
```

is included in the job deck, the resulting 36K Core Rope Simulator Tape will include vacant paragraphs if at least one of the following is true:

1) y is not the first paragraph of a bank;

2) z is not the last paragraph of a bank.

If the 36K Core Rope Simulator Tape medium is DIGISTORE tape or seven-track magnetic tape, each vacant paragraph will be represented as if it was formed by converting 256 unused AGC words (unused AGC word is 15 zero bits plus zero even parity bit) to the correct output format. If the output medium is punched tape, a vacant paragraph will appear to have been formed by converting 256 identical AGC words, each one of which is 17777 (octal) plus zero odd parity bit (17777 = TC to the end of the bank). These same conventions also apply to any unused words at the end of a partially filled bank.

As an example of the use of the subdirector

```
S       PARAGRAPH Y
```

the following deck will generate a 36K Core Rope Simulator Tape punched into blue paper-mylar tape:

```
//COMCHEMF JOB 0250,HOPE.J  COMCHEMF
//         EXEC     GAPMAGC
//M.SYSIN   DD   *
*        MANUFACTURE REVISION 067 OF AGC PROGRAM COMANCHE BY NASA 2021113-061
S        PUNCH 36K CORE ROPE SIMULATOR TAPE
S        PARAGRAPH 102
S        USE BLUE MYLAR TAPE
//      EXEC       GAPPMHST
/*
```

The output contains Fixed Bank 14, composed of paragraphs 100, 101, 102, and 103. Paragraphs 100, 101, and 103 are punched as vacant paragraphs, each word of which is 17777 (octal) as described in the preceding paragraph. An example similar to the above which illustrates the use of the subdirector

```
          S       PARAGRAPH Y THRU Z
or        S       PARAGRAPH Y THROUGH Z
```

is the following:

```
//COMCHEMF JOB 0250,HOPE.J  COMCHEMF
//      EXEC      GAPMAGC
//M.SYSIN    DD    *
*         MANUFACTURE REVISION 067 OF AGC PROGRAM COMANCHE BY NASA 2021113-061
S         PUNCH 36K CORE ROPE SIMULATOR TAPE
S         PARAGRAPH 101 THRU 107
S         USE BLUE MYLAR TAPE
//       EXEC     GAPPMHST
/*
```

The punched output contains Fixed Banks 14 and 15, comprising paragraphs 100 through 107 inclusive. Paragraph 100 is a vacant paragraph, punched as shown in the preceding example.

If the Paragraph Subdirector Card is omitted, the default option is

```
S         PARAGRAPH ALL
```

The subdirector

```
S         USE BLUE MYLAR TAPE
```

that appears in each of the three preceding examples causes the 36K Core Rope Simulator Tape output to be punched into an eight-channel blue paper-mylar tape. If the output is to be an eight-channel punched paper, aluminum or fiber tape, use one of the following subdirectors:

```
S         USE PINK OILED PAPER TAPE
S         USE BLACK OILED PAPER TAPE
S         USE ALUMINUM TAPE
S         USE GREY FIBER TAPE
S         USE GRAY FIBER TAPE
```

If the subdirector

```
S         USE DIGISTORE TAPE
```

or

```
S         USE DIGISTOR TAPE
```

is included in the job deck, the 36K Core Rope Simulator Tape output will be written on DIGISTORE (magnetic) tape. For example, a DIGISTORE tape of the 36K Core Rope Simulator Tape output will be generated by the following job deck:

```
//COMANCHE JOB 0250,HOPE.J   COMANCHE MFR
//JOBLIB   DD   DSNAME=KTL.LIB,DISP=(SHR,PASS),VOLUME=SER=SYSLIB,UNIT=2314
//      EXEC      GAPMAGC
//M.SYSIN    DD    *
*         MANUFACTURE REVISION 044 OF AGC PROGRAM COMANCHE BY NASA 2021113-011
S         PUNCH 36K CORE ROPE SIMULATOR TAPE
S         PARAGRAPH ALL
S         USE DIGISTORE TAPE
//       EXEC     GAPPMHST
/*
```

The 36K Core Rope Simulator Tape output will be written on seven-track magnetic tape if the subdirector

```
S       USE 7 TRACK MAGNETIC TAPE
```

is included in the job deck, and if the job is run under the catalogued procedure GAPMAGCA. In this case, two files are written on the magnetic tape:

1) File one contains the program identification in the form of a single 77-character, even parity, 800 BPI, BCD record.

2) File two contains the 36K Core Rope Simulator tape output. LRECL=126, BLKSIZE=3150, therefore there are 4200 tape characters (core character = 8 bits, tape character = 6 bits) per block. The records are odd parity binary and are written at a density of 800 BPI. The following job deck will cause such a tape to be manufactured:

```
//COMANCHE JOB 0250,HOPE.J   COMANCHE MFR
// .    EXEC     GAPMAGCA
//M.SYSIN     DD     *
*       MANUFACTURE REVISION 044 OF AGC PROGRAM COMANCHE BY NASA 2021113-011
S       PUNCH 36K CORE ROPE SIMULATOR TAPE
S       PARAGRAPH ALL
S       USE 7 TRACK MAGNETIC TAPE
//      EXEC     GAPPMHST
/*
```

An example of a job deck that will result in a seven-track magnetic tape with an AGC Symbol Table as the third file (as described in the section PUNCH SYMBOL TABLE) is the following:

```
//COMANCHE JOB 0250,HOPE.J   COMANCHE MFR
//      EXEC     GAPMAGCA
//M.SYSIN   DD    *
*       MANUFACTURE REVISION 044 OF AGC PROGRAM COMANCHE BY NASA 2021113-011
S       PUNCH 36K CORE ROPE SIMULATOR TAPE
S       PARAGRAPH ALL
S       USE 7 TRACK MAGNETIC TAPE
/*
//      EXEC     GAPMAGCA
//M.SYSIN   DD    *
*       MANUFACTURE REVISION 044 OF AGC PROGRAM COMANCHE BY NASA 2021113-011
S       PUNCH SYMBOL TABLE
//      EXEC     GAPPMHST
/*
```

If the Subdirector Card specifying the output medium is omitted, the default option is

```
S       USE PINK OILED PAPER TAPE
```

The output formats of the various types of 36K Core Rope Simulator Tapes are illustrated in Appendix F. The formats are also described in the above mentioned document E-2026 "USERS' GUIDE TO THE AGC MONITOR" on page 19 (note that GAP Manufacturing does not use the formats labeled character A and character C).

When the output medium is punched tape, each bank is preceded and followed by the program identification and the bank number punched in the form of human readable characters. In this case a fixed bank occupies 436.4 inches and an erasable bank 216.8 inches of punched tape, including tape feed and the human readable identification. If the 36K Core Rope Simulator Tape output is written on DIGISTORE tape, no program identification is included and the user must rely on the computer operator to label (i.e. gummed label) the tape correctly. If the output is written on seven-track magnetic tape, the first file contains the program identification in the form described earlier in this section.

All 36K Core Rope Simulator Tapes are actually manufactured in two stages. In the first job step, the binary AGC program is converted to 36K Core Rope Simulator Tape format and stored in a temporary data set on direct access storage, along with information concerning the type of output medium and the total number of frames generated. In the second job step, the contents of the temporary data set are retrieved and written on the proper output medium (determined from the information passed along by job step one). If the output is punched, the computer operator is informed via the on-line typewriter as to the type of tape needed and the number of feet required. If the output is written on DIGISTORE tape, an on-line message is sent to the operator requesting that the tape be labeled AGCPROG.p.r, where "p" represents the program name and "r" is the three-digit revision number.

Unlike seven-track magnetic tapes, DIGISTORE tapes are not rewound when the job has been completed. Hence a whole series of AGC programs and/or segments may be converted to 36K Core Rope Simulator Tape output format and written successively on a single DIGISTORE tape.

5.6.5 Binary COMPARE

GAP Manufacturing will compare two binary AGC programs with one another and list all differences if the job deck contains a COMPARE Director Card followed by a WITH subdirector card.

The COMPARE Director Card, containing the identification of the first AGC program involved in the comparison, was described in the section headed Director Card.

5.6.5.1 Subdirector WITH

The subdirector WITH identifies the second AGC program involved in the comparison; it has the following form:

```
S        WITH 360 REVISION N OF AGC PROGRAM M BY A
```

Here, as in the case of the Director Card, n represents the revision number, m the name of the program, and a the name of the author.

The WITH subdirector must be followed by one of the three types of PARAGRAPH subdirectors discussed in the section entitled PUNCH 36K CORE ROPE SIMULATOR TAPE. The Paragraph Subdirector Card specifies the number(s) of the corresponding paragraphs from each AGC program that are to be compared. In this case, comparison takes place, paragraph by paragraph, not bank by bank, so that bank boundaries are of no concern, in contrast with 36K Core Rope Simulator Tape manufacture. For example, execution of the job deck

```
//BINCOMP  JOB 0341,HOPE.J      BIN COMPARE JOB
//      EXEC    GAPMAGC
//M.SYSIN   DD   *
*        COMPARE REVISION 097 OF AGC PROGRAM LUMINARY BY NASA 2021112-041
S        WITH 360 REVISION 099 OF AGC PROGRAM LUMINARY BY NASA 2021112-051
S        PARAGRAPH ALL
//      EXEC      GAPPMHST
/*
```

will cause all paragraphs of Revision 97 of the binary AGC program LUMINARY to be compared with the corresponding paragraphs of Revision 99 of LUMINARY and all differences to be listed.

5.6.5.2 PARAGRAPH Subdirector

When the subdirector

```
S        PARAGRAPH ALL
```

is employed and a paragraph present in one of the AGC programs is missing in the other program, a message to this effect is written in the output that includes the number of the paragraph and the identity of the program from which it is absent.

Corresponding paragraphs from each of the two AGC programs are compared word by word. If the paragraphs are identical, no output is produced. If the paragraphs differ, a page is printed that shows the following:

1) the identities of the two AGC programs;

2) the current date and time;

3) the (octal) paragraph number, (octal) bank number and manufacturing location information (rope module number, side, sense line set number and wire numbers);

4) the (octal) bank address of each paragraph word; also the corresponding (octal) address relative to the first word in the paragraph (addresses 000 through 377);

5) an equal sign for each pair of corresponding AGC words that match identically;

6) the side by side octal values and parity digits of each pair of corresponding AGC words that do not match.

When all paragraph comparisons have been completed, a final page of output is generated that tabulates:

1) the number of paragraphs that were compared;

2) the number of paragraphs that differed;

3) the total count of pairs of words that didn't match;

4) the module numbers of all core ropes that differed;

5) the number of paragraphs which did not possess counterparts with matching paragraph numbers in the opposite data set.

A typical comparison output listing is shown in Appendix G.

It is possible to compare a binary AGC program stored in direct access storage with a program on magnetic tape. It is also possible to manufacture a program stored on magnetic tape. In either case the tape is searched until the program in question is found. The catalogued procedures GAPMAGC or GAPMAGCA must be modified before any attempt is made to work with binary AGC programs on magnetic tape; consult a GAP system expert for further information on this subject.

5.6.6 Write Portafam Tape

The PORTAFAM (Portable Fixed Memory) unit is a device that is designed to replace the AGC Fixed Memory for field simulation tests and diagnostic purposes. The principal elements of the PORTAFAM system are an erasable memory, designed to replace the AGC fixed memory when testing in the field, and a magnetic tape transport. An operational description of the PORTAFAM system is given in the MIT Instrumentation Laboratory publication, E-2402, "PORTAFAM SYSTEM OPERATIONAL DESCRIPTION", by Jonathan Leavitt, July 1969.

The PORTAFAM memory is loaded with an AGC program by the magnetic tape transport from a one-half inch, seven-track, IBM format magnetic tape called, not surprisingly, a PORTAFAM tape.

PORTAFAM tapes of AGC Mission programs will be generated by GAP Manufacturing if the subdirector

```
S       WRITE PORTAFAM TAPE
```

follows the Manufacturing Director Card in the job deck.

PORTAFAM tapes are blocked tapes containing one bank of AGC data per block. The entire AGC program is repeated three times in succession on the tape to provide redundancy and insure against loss of information from damage or dropout when the tape is read.

PORTAFAM tape characteristics and formats are discussed in detail in Appendix H.

As an example, running the following job deck will cause a PORTAFAM tape containing Revision 69 of the AGC Mission program LUMINARY to be written:

```
//MANUPFAM JOB 1024,WARD.P       PORTAFAM
//      EXEC     GAPMAGC
//M.SYSIN     DD     *
*       MANUFACTURE REVISION 069 OF AGC PROGRAM LUMINARY BY APOLLO
S       WRITE PORTAFAM TAPE
//      EXEC     GAPPMHST
//      EXEC    GAPPPFTP
/*
```

In the above, the catalogued procedure GAPPPFTP generates a hexadecimal dump of the PORTAFAM tape written in the first job step. Specimen pages of a PORTAFAM Tape Dump are shown in Appendix I along with instructions for reading the dump.

Catalogued Procedures

All catalogued procedures mentioned in this document are listed in Appendix J.

6. UTILITIES AND BACKUP

6.1 GAPCOMPR

The original purpose of GAPCOMPR was to insure an orderly and accurate transition from the H-1800 to the IBM 360/75 when the latter was installed in the spring of 1968. The program compares symbolic files in a point by point sequential manner. In order for it to work properly the two files being compared should have the same structure down to the log section level. GAPCOMPR matches program names, subroutine names and log section names before attempting to compare card images. This matching procedure is not flexible; if for example the order of two log sections were interchanged in one of the programs, at best one of the two would be compared at the detail card level. If there were other log sections intervening between the two that were interchanged, then these also would not be compared. Furthermore, the program compares the entire card, so that even if the coding were the same in the two programs, if the numbering schemes are different, all coding will be printed out as mismatches.

GAPCOMPR proceeds in two steps. The first step copies one of the two files onto magnetic tape. The second step actually compares the file on the tape with a specified file on the disk pack.

GAPCOMPR is a catalog procedure. Thus all that is necessary to perform a comparison is to submit the appropriate JCL cards and control cards. After the ordinary JOB card these are:

```
// EXEC GAPCOMPR,(,FILE=GAPOON)
//TAPE.SYSIN DD *
     PROGRAM PROGNAME N PROGRAMMER
/*
//COMPR.SYSIN DD *
     PROGNAME (TAPE) PROGNAME (DISK) N (DISK)
```

The cards preceding the /* cause the program(s) named to be copied onto tape. n is the revision number. The remaining cards cause the comparison. Theoretically one can make as many comparisons as one wishes: the practical limit is the number of programs which will fit on one reel of tape.

6.2 Disk Deletion

When an assembly of an AGC program is made, it is stored on a disk pack by the name of AGCBO2. This disk pack fills up quite often, since no provision is made to avoid quasi-duplicate files. Thus while a hundred revisions of an AGC program may occupy only fifty percent more space than the original in symbolic form, the assembled programs would take up one hundred times the space of one assembly. Generally it is not necessary to have more than a few revisions out of several hundred assembled at one time (latest revision, working program and the like). So, in order to fit everything on one disk, periodic deletions of unused revisions are made. The assemblies may be recreated at any time from the symbolic records.

There are two parts to removing a program from AGCBO2. The first is to remove the actual stored program. The second is to remove the catalogue information which was used to locate the assembled program. After the usual JOB card code:

```
// EXEC PGM=IEHPROGM
//VOLUME01 DD VOLUME=SER=AGCBO2,UNIT=2314,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
   UNCATLG DSNAME=AGCPROG.PROGNAME.RNNN
   SCRATCH DSNAME=AGCPROG.PROGNAME.RNNN,PURGE,VOL=2314=AGCBO2
/*
```

As many programs may be deleted in one step as the user wants. The order of the control cards (UNCATLG and SCRATCH) is immaterial.

6.3 AGCTAPE

AGCTAPE is a procedure for copying AGC symbolic programs from GAP disk files onto magnetic tapes. The first step of the procedure labels the tape and the second tape copies the program. The first step may be omitted if the tape is already labeled.

The program can be instructed to write a CREATE director card preceding the program. This allows the tape to be used as input to AGCONVRS to create a file. The user can change the name, revision number or author of the file as it exists on the tape.

A program can be "frozen" by copying the main portion onto tape with a CREATE heading it, and copying its subroutines following it without CREATE directors preceding them. When the program is then recopied onto a GAP file it will include its subroutines within a single file.

In order to label and copy a file the following JCL is used

```
// EXEC AGCTAPE,TAPE=TAPENAME(,FILE=GAPOON),PGMNAME=IEHINITT
//LABEL.SYSIN DD *
TVOL  INITT  SER=PROGNAME,OWNER=GAPSYS,NUMBTAPE=1,DISP=REWIND
/*
//PROGM.SYSIN DD *
```

n=1,2,3,4. 1 is the default option. PGMNAME=IEHINITT is coded only when the tape is to be labeled.

Following the JCL it is necessary to include a control card. These are:

```
*        WITH DIRECTOR CARD
*        WITHOUT DIRECTOR CARD
```

These instruct the program whether or not to precede the file by a CREATE director. Following the control cards are the data cards. There is one data card for each program, subroutine or segment to be copied. The form is

```
PROGRAM
SUBROUTINE      name nnn programmer newname newnnn
SEGMENT
```

There must be at least one initial blank. There may not be embedded blanks, but a break character ("_") on a data card is replaced by a blank on the director written to head the file. Renaming and renumbering are idependently optional. Whatever is coded for programmer will appear on the director card.

6.4 Backup and Recovery Procedures for GAP Files

GAP data sets are exposed to minimal risk of destruction from machine failure. Nevertheless there are times during which they are vulnerable. For this reason regular and systematic backup procedures are necessary. The system chosen for backup depends on periodic copying of the entire file onto magnetic tape. There are additional procedural safeguards: four backups are made before the first is written over, and cards that went into an update of a major Apollo program are kept as an update record. Recovery is by copying the latest tape version of the file onto the disk and updating with cards.

There are two catalog procedures which have been written to implement the backup. AGCBKUP copies a GAP file onto two reels of magnetic tape. AGCRECVR restores the GAP file from tape. It is necessary that the tapes have standard labels; for this reason program IEHINITT must be run prior to the first run of AGCBKUP with each pair of tapes to insure standard labeling. Once a tape has been labeled it is not necessary to label it again unless its name is to be changed. Backup on GAP001 should be done twice a week: GAP002 and GAP003 should be backed up after each addition to the file.

To label a pair of tapes and copy the GAP file use the following deck.

```
// EXEC PGM=IEHINITT
//SYSPRINT DD SYSOUT=A
//TVOL DD UNIT=(TAPE9,1,DEFER)
//SYSIN DD *
TVOL INITT SER=TAPENAME,OWNER=GAPSYS,DISP=REWIND
TVOL INITT SER=TAPETWO,OWNER=GAPSYS,DISP=REWIND
/*
// EXEC AGCBKUP,TAPEA=TAPENAME,TAPEB=TAPETWO,FILE=GAPOON
/*
```

The cards preceding the first delimiter card cause the tapes to be labeled. Currently there are eight tapes that have been labeled, enough for four cycles of backup. Unless it becomes necessary to relabel or replace and label one or more of these tapes, it is only necessary to submit the card which causes the execution of the catalog procedure AGCBKUP. n can take on the values 1-4. FILE=GAP001 is the default option if FILE is not coded.

The eight tapes used are labeled GP1BK1, GP1BK2, ---, GP1BK8. They are used two at a time for the first four backups, and then the first two are used over again for the fifth backup, and so on.

AGCBKUP will print a message to indicate whether or not the procedure was successfully completed. If failure was caused by a write error rerun using a different tape. If failure was caused by a read error (error on the disk), it will be necessary to restore the GAP file from an earlier backup.

To recover a GAP file use

```
// EXEC GAPRECVR,TAPEA=TAPENAME,TAPEB=TAPETWO,FILE=GAPOON
```

All parameters must be coded.

6.5 TABSIM

TABSIM is a general purpose card-handling program written to operate on an IBM 360 Model 20 computer. TABSIM accomplishes the usual functions of the IBM 519 reproducing punch, the IBM 557 interpreter and the IBM 407 tabulating machine. It can accomplish these functions one at a time or several in one pass. TABSIM can accept input cards in either BCD mode or EBCDIC mode and it will convert from BCD to EBCDIC if so desired.

Punching speed ranges from about 90 cards per minute when punching all 80 columns per card to about 250 cards per minute when punching only the first 10 columns per card. Interpreting can take place simultaneously with card punching with only about a 25% decrease in punching speed. Listing speed is about 300 lines per minute.

A control card determines which tasks are to be performed in a particular run. Every run requires a control card and certain runs also require parameter cards. For general tasks such as listing, reproducing, and interpreting, TABSIM requires only a control card, and the various control cards for performing these tasks are available at the IBM 360 Model 20 console. In order to use TABSIM to perform these general tasks, it is necessary to read only Section 2 (Operating Procedures) of this guide. For more specialized tasks such as reproducing and gangpunching, numbering, reproducing with rearranging, or reproducing with numbering, it is necessary for the user to prepare the control card and the parameter cards, if any are needed. The methods for doing this are explained in Section 1 of this guide.

6.5.1 Section 1 - Preparing Control and Parameter Cards

Every run must have a control card. Runs that gangpunch, reproduce with rearranging, or reproduce only part of each card, also require up to four parameter cards. A control card must have the following punches in the first 4 columns.

Column	1	2	3	4
	12	0	12	6
	0		2	9
	7		9	
	9			

It is necessary to use the multi-punch key in order to punch the code for columns 1, 3, and 4.

The operation codes, specifying the tasks for a particular run, are punched in columns 10-17 of the control card. TABSIM can perform the following tasks: reproducing, gangpunching, numbering, listing, and interpreting, either one task per run or any combination simultaneously during the same run. TABSIM can also be used for comparing two decks of cards, but when TABSIM is used for comparing, no other tasks can be performed during the same run. If the user specifies both punching and interpreting, it is the output deck that is interpreted, but if the user specifies only interpreting, the input deck is interpreted. When both punching and listing are specified, the output cards are listed. If only a listing is specified, the input cards are listed.

The 2560 card reader/punch on the IBM 360 Model 20 has two feed hoppers. When using TABSIM all card reading is done through the primary feed and all card punching is done through the secondary feed with one exception. That is, when TABSIM is used for comparing two card decks, then one card deck is read through the primary feed and the other is read through the secondary feed. Therefore, the control card and parameter cards are always read through the primary feed, the input deck when listing reproducing or interpreting is read through the primary feed and when gangpunching or numbering an existing deck, this deck is read through the secondary feed.

On the next page is a summary of the operation codes and their appropriate columns in the control card. These are explained in more detail in the following pages.

Column	Operation Code	Task
10	8	80x80 Listing
10	M	MAC Listing
10	A	360 BAL Listing
11	blank or 1	Listing is single spaced.
11	2,3,4,...9	Listing is spaced this number of lines between print lines.
12	R	80x80 reproducing
12	G	80x80 gangpunching
12	P	Reproducing and gangpunching with a parameter card to determine which columns are reproduced, which are gangpunched and which are left alone.
12	F	Reproducing and gangpunching with column switching. The first parameter card determines which columns are reproduced, which are gangpunched and which are left alone. Two additional parameter cards determine the column switching.
13	N	Numbering
13	R	Numbering, with the card number interpreted on the right side between row 11 and row 12 on the output cards
13	L	Numbering, with the card number interpreted on the left side between row 11 and row 12 on the output cards
14	I	Interpreting
15	B	The input deck is assumed to be in BCD mode. If punching is requested, the cards are punched in EBCDIC mode.
16	C	The output cards are checked to be certain all columns are blank prior to punching.
17	C	Comparing with listing
17	S	Comparing - The run will stop when there is an illegal compare.

6.5.2 Card Listing

Column 10 of the control card is used to specify the type of listing desired, if any. Column 11 is used to control vertical spacing of the listing. The following are the legal operation codes for column 10.

blank No listing

8 Standard 80x80 listing with no special features.

M For listing a MAC program or MAC data. The listing is formated the same as using the "Honeywell Formats 407 Board" with all switches down on the 407.

A For listing a 360 BAL (Basic Assembly Language) program. This type of listing has the following special features. If columns 10-14 contain TITLE, the card image is not printed in the listing but the printer skips to head of form, and the card image is printed as a heading at the top of this and each succeeding page of the listing. If columns 10-14 contain SPACE, the card image is not printed but the printer is spaced the number of lines specified in column 16. If columns 10-14 contain EJECT, the card image is not printed but the printer skips to head of form.

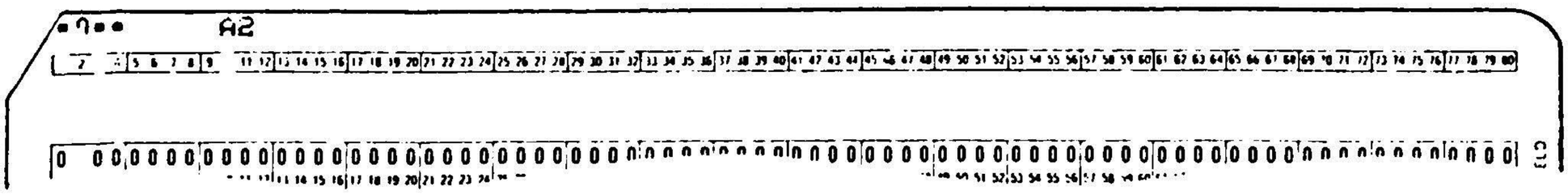
The following are the legal operation codes for column 11.

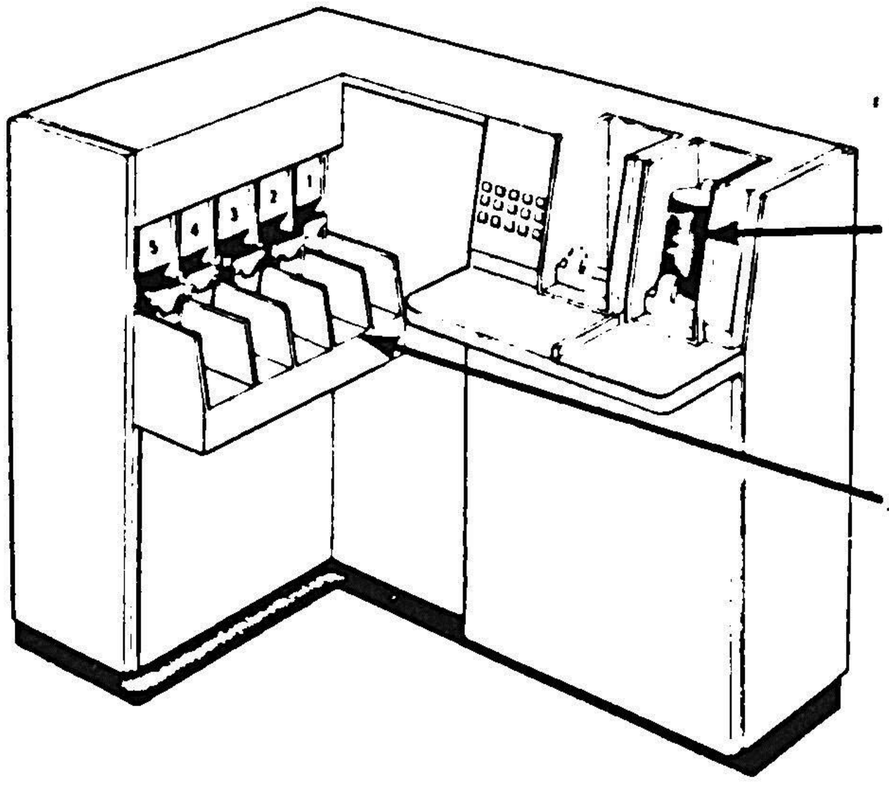
blank or 1 The listing is single spaced.

2,3,...9 The listing is spaced this number of lines between print lines.

Example

To list a 360 BAL program with double spacing, the control card should be as follows:





The program deck preceded by the control card and followed by at least 6 blank cards should be placed in the primary feed of the 2560 card reader/punch.

After the cards are read they will be stacked in stacker 2.

6.5.3 Reproducing and Gangpunching

TABSIM can be used to perform any of the usual tasks done on the IBM 519 reproducing punch. Like the 519, TABSIM can switch columns while reproducing. In other words, it can take data from any column of the input card and punch it in a different column of the output card. The format of the output cards is determined by parameter cards when using TABSIM rather then by a wired board when using the 519. Column 12 of the control card determines whether any reproducing and/or gangpunching takes place and if so what type. The following are the legal operation codes for column 12.

blank No reproducing or gangpunching

R Standard 80x80 reproducing. No parameter cards are needed with this operation code. This is exactly like using the 80x80 reproducing board on the 519.

G 80x80 gangpunching. One parameter card is needed with this operation code. All the output cards are punched identically like this parameter card.

P Reproducing and/or gangpunching. Either one or two parameter cards are needed with this operation code. The first parameter card must be an R/G card. This card shows which columns are to be punched by reproducing, which are to be punched by gangpunching and which are to be left alone. Each column of this card must contain a blank, an R, or a G. All columns that are blank are left alone. All columns that contain an R are reproduced and all columns that contain a G are gangpunched. If any column of the R/G parameter card contains a G, a second parameter card is needed. This card contains the data in the appropriate columns which is to be gangpunched into the output cards. This operation code is exactly like using the General Purpose Board on the 519.

F Reproducing and/or gangpunching with column switching. Either three or four parameter cards are needed with this operation code. The first one or two parameter cards are the same as the parameter cards described in the previous operation code "P". The last two parameter cards are rearranger cards. These 2 cards define the column switching. They specify which input column corresponds to each output column. The following example will explain how this is done.

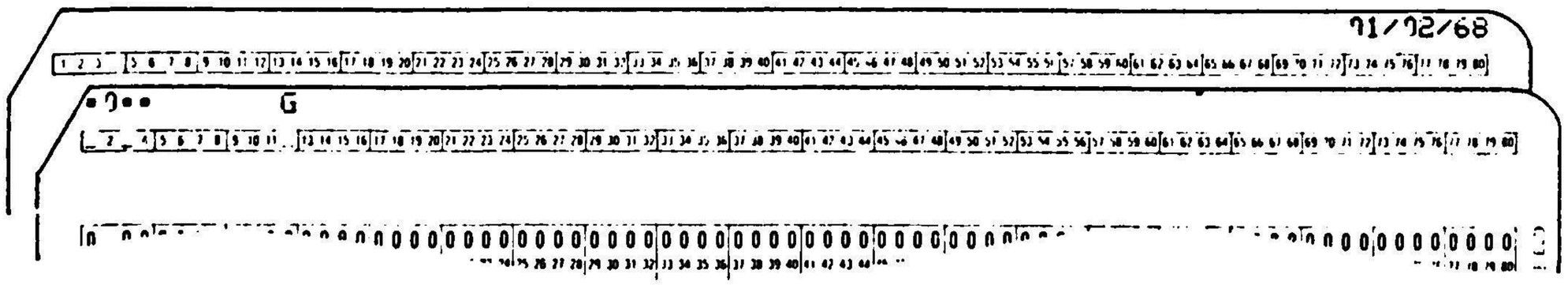
To cause column 10 of the output cards to be reproduced from column 25 of the input column, punch an "R" in column 10 of the R/G parameter card. Also punch "2" into column 10 of the first

rearranger card and "5" into column 10 of the second rearranger card.

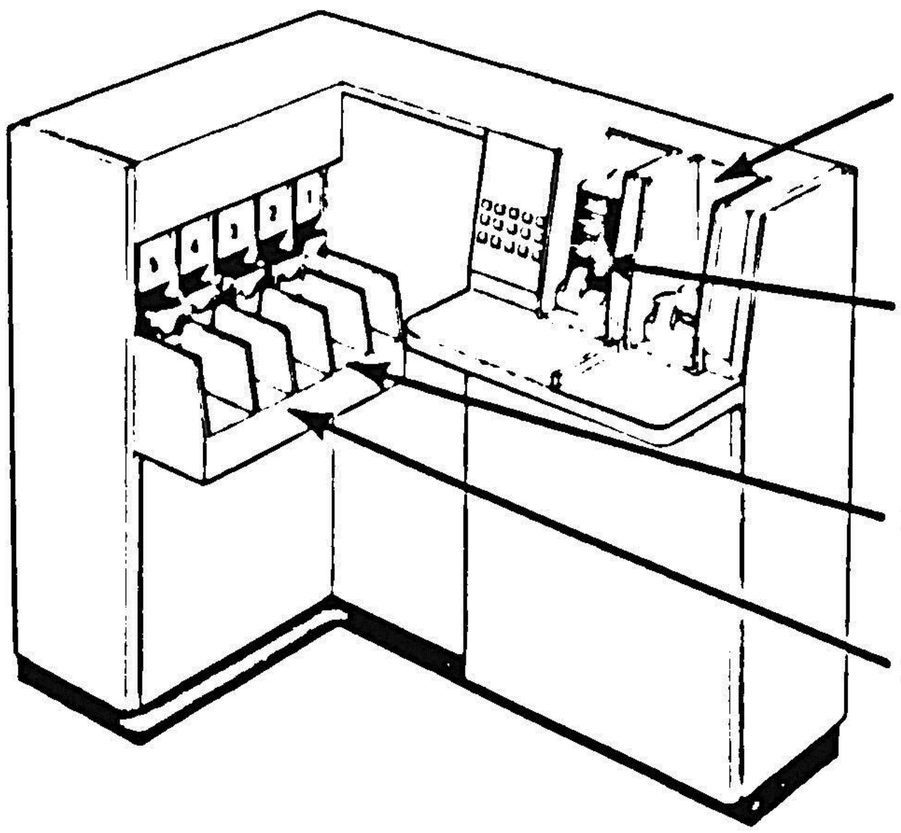
Hence for each column of the output card that is reproduced, the first rearranger card contains in that column, the "tens" digits of the input column and the second rearranger card contains in that column the "ones" digit of the input column. If columns 1-9 are input columns, the "tens" digits would be zero. If the input column and the output column are the same, for example, if column 15 of the output cards is reproduced from column 15 of the input cards, it is not necessary to punch the number of the input column in the rearranger cards. If for any column there is an R in the R/G card and that column on the rearranger cards is blank, TABSIM assumes the input column and the output column are the same.

Example 1

In order to gangpunch 01/02/68 in columns 73-80 of an existing deck, the control card and parameter card would be:



The cards should be placed in the 2560 card reader/punch as follows:



Primary Feed - Place here the control card and parameter card followed by at least 3 blank cards.

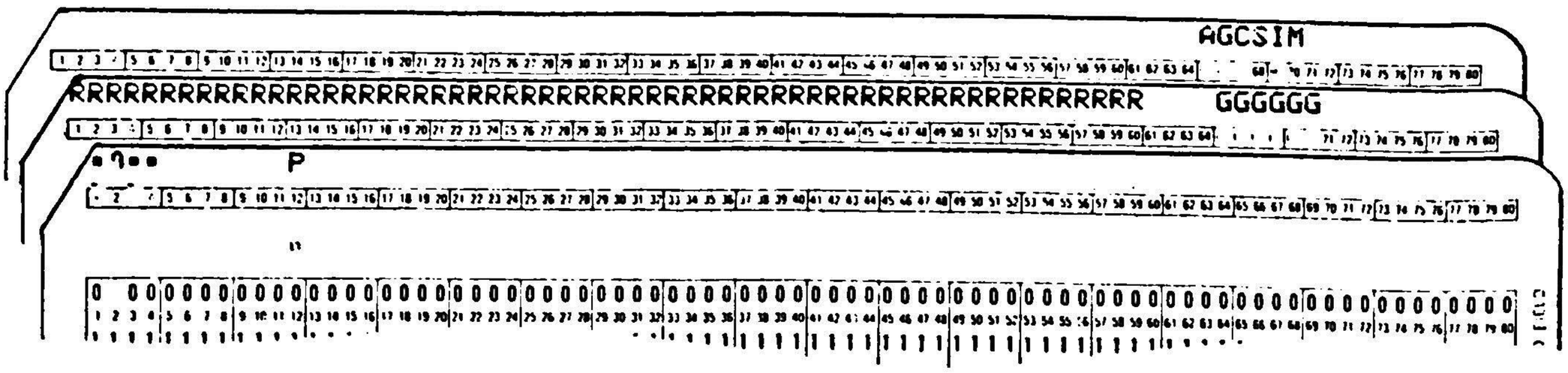
Secondary Feed - Place here the existing deck which is to be gangpunched, followed by at least 3 blank cards.

Stacker 2 will contain the cards from the primary feed.

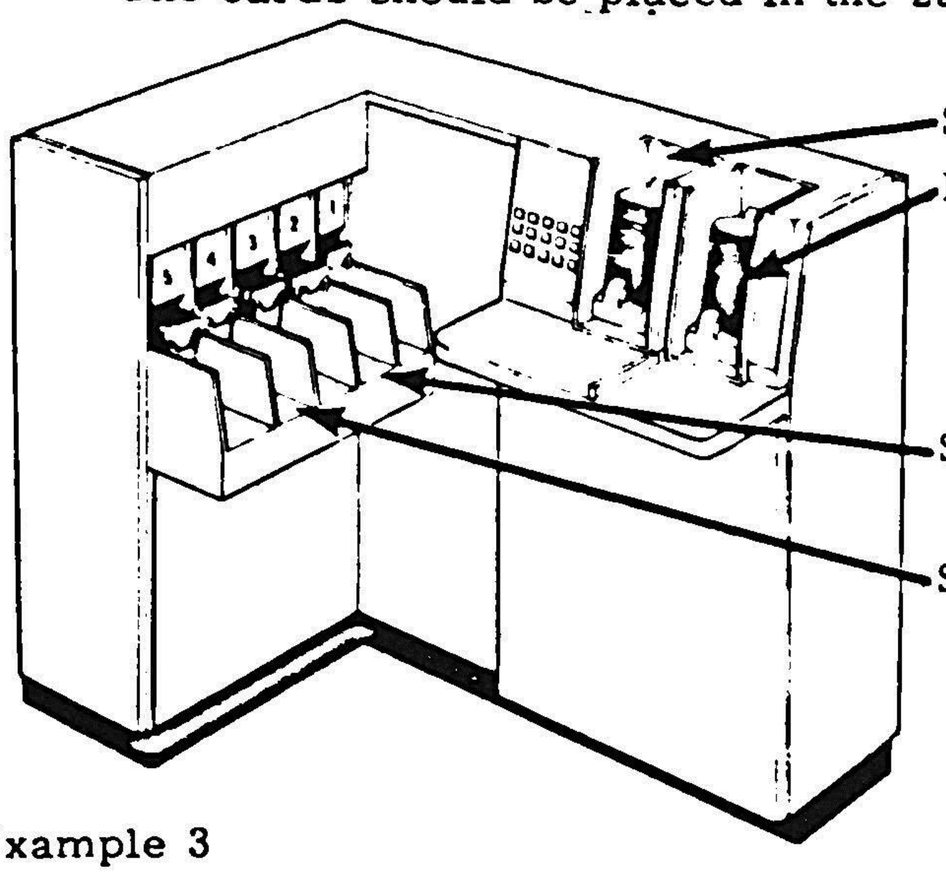
Stacker 4 will contain the cards from the secondary feed.

Example 2

In order to reproduce columns 1-60 of a card deck and gangpunch AGCSIM in columns 65-70 of the output deck, the control card and parameter cards would be:



The cards should be placed in the 2560 card reader/punch as follows:



Secondary Feed - Place blank cards here.

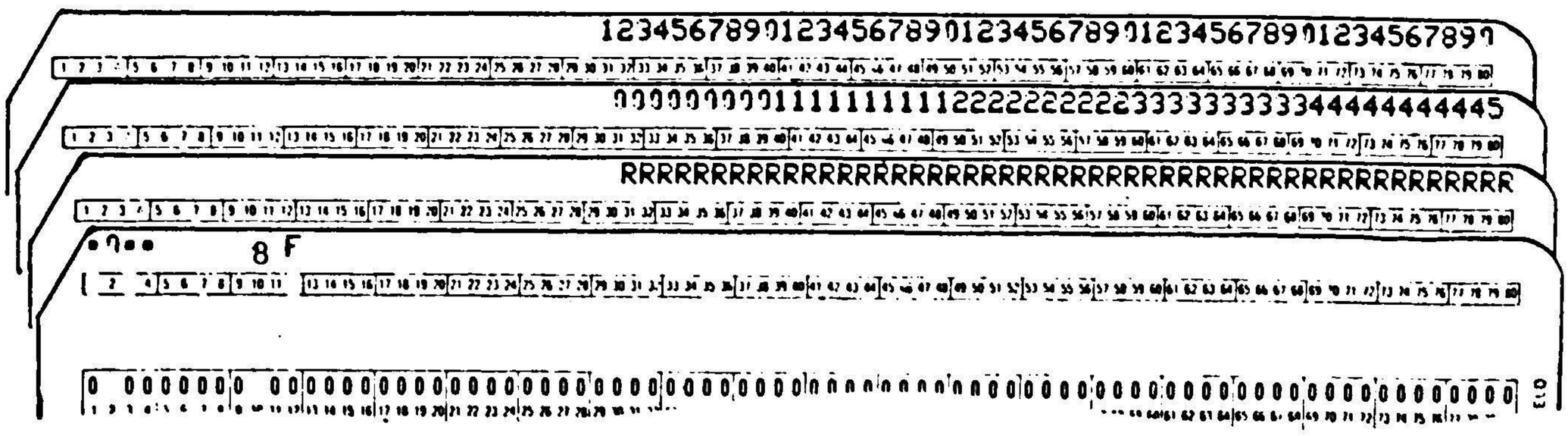
Primary Feed - Place here, the control card, parameter cards, and the input deck followed by at least 6 blank cards.

Stacker 2 will contain the cards from the primary feed.

Stacker 4 will contain the cards from the secondary feed.

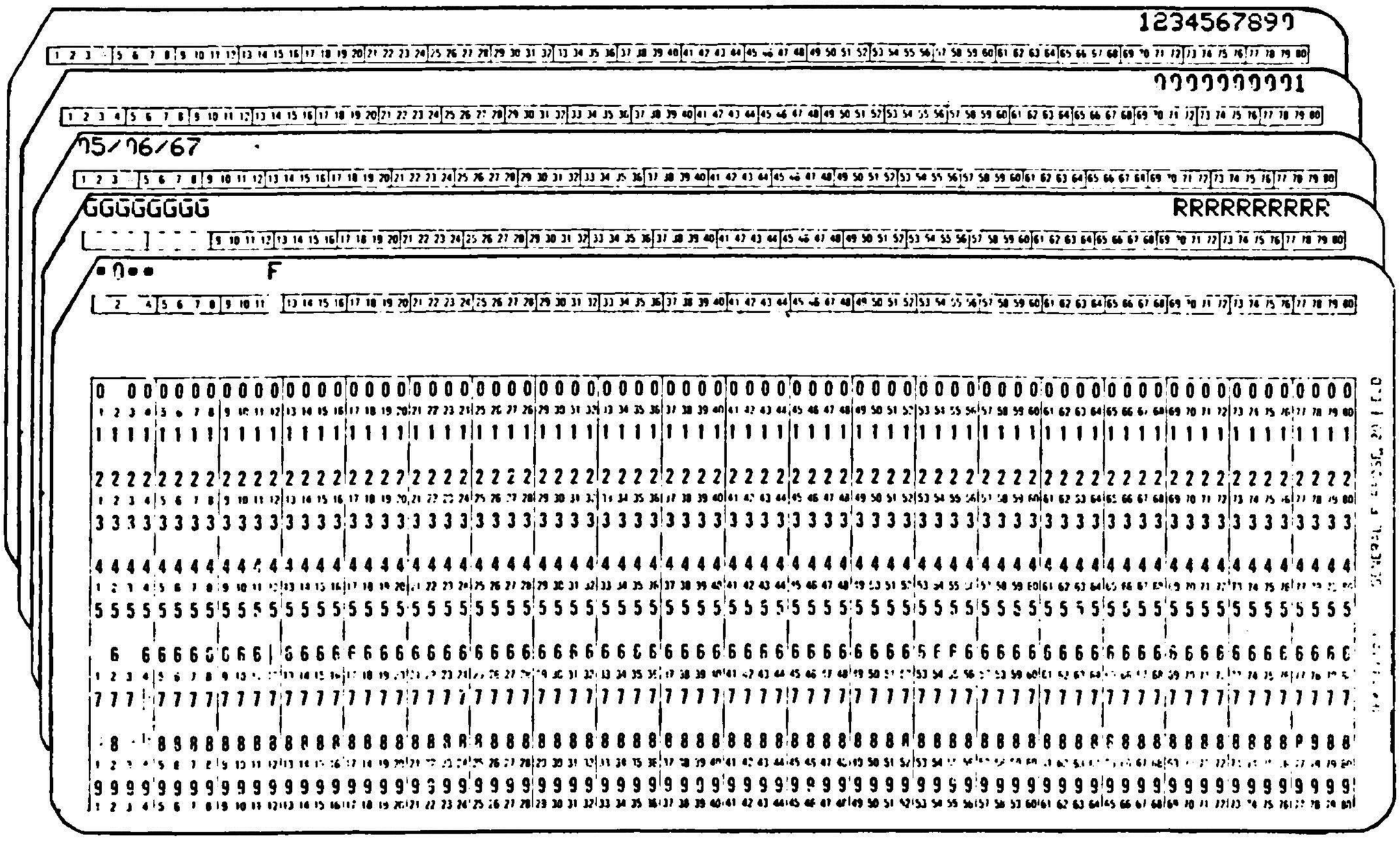
Example 3

In order to reproduce columns 1-50 of a card deck into columns 31-80 of the output deck and get a 80x80 listing of the output cards, the control card and parameter cards would be:



Example 4

In order to reproduce columns 1-10 of a card deck in columns 70-79 of the output deck and gangpunch 05/06/67 in columns 1-8 of the output deck, the control card and parameter cards would be:



6.5.4 Numbering

TABSIM can either number along with reproducing and gangpunching or number what is presumed to be an existing deck. Numbering can start in any column and extend for up to nine columns. Any decimal numbers can be used for the initial value and for the increment. Column 13 of the control card determines whether any numbering takes place. The acceptable operation codes for column 13 are as follows:

blank	No numbering
N	Numbering
R	Numbering, with the card numbers interpreted on the right side between row 11 and row 12 on the output cards
L	Numbering, with the card numbers interpreted on the left side between row 11 and row 12 on the output cards

The user can either state explicitly the starting column for the numbering, the extent of the numbering field, the initial value and the increment, or he can use a keyword which implies a specific column, initial value and increment . If a keyword is used it must be punched in columns 20-22 of the control card. The keywords available and their implied formats are as follows:

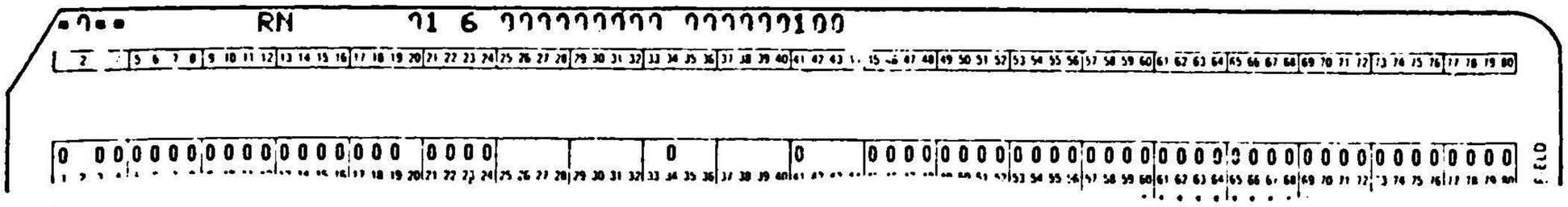
Keyword	Initial Value	Increment	Columns
MAC	000 000 000	000 000 100	2-7
360	000 000 000	000 000 100	73-80

If the user desires to state the columns, initial value, and increment explicitly, this information is punched into the control card as follows:

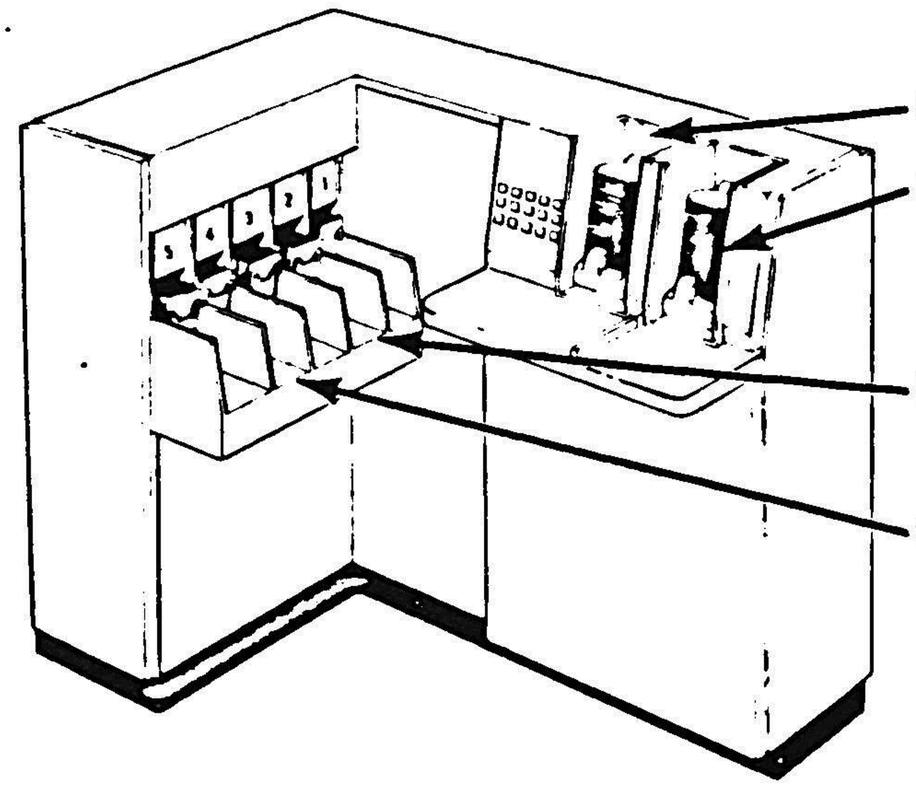
Columns	
20-21	Starting column for numbering (Punching as 01,02,...09,10,...80)
23	Size of the numbering field (Any size up to 9 columns is legal)
25-33	Initial Value (If the numbering field is less than 9 columns, the initial value should be right-justified with any remaining columns to the left filled with zeros.)
35-43	Increment (If the numbering field is less than 9 columns, the increment should be right justified with any remaining columns to the left filled with zeros.)

Example 1

In order to reproduce a card deck and number the output deck in columns 1-6, with an initial value of zero and an increment of 100, the control card would be:



The cards should be placed in the 2560 card reader/punch as follows:



Secondary Feed - Place blank cards here.

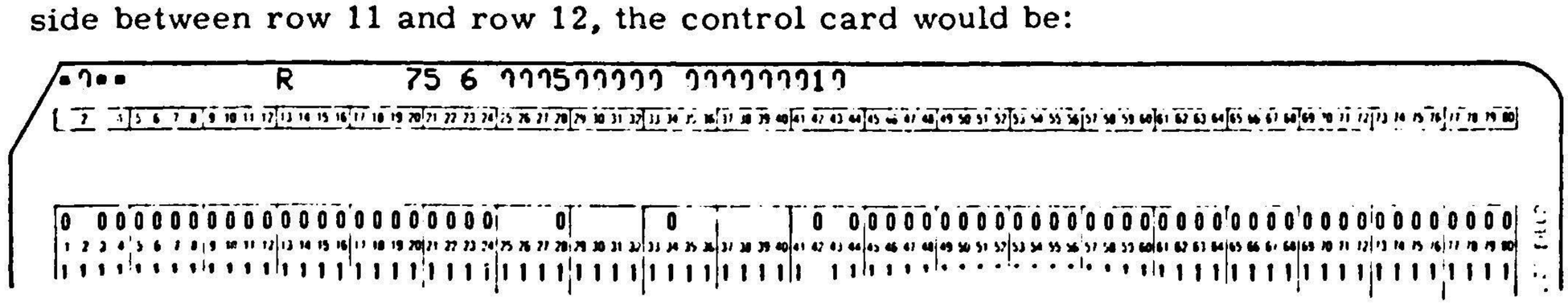
Primary Feed - Place here the control card and the input deck followed by at least 6 blank cards.

Stacker 2 will contain the cards from the primary feed.

Stacker 4 will contain the cards from the secondary feed.

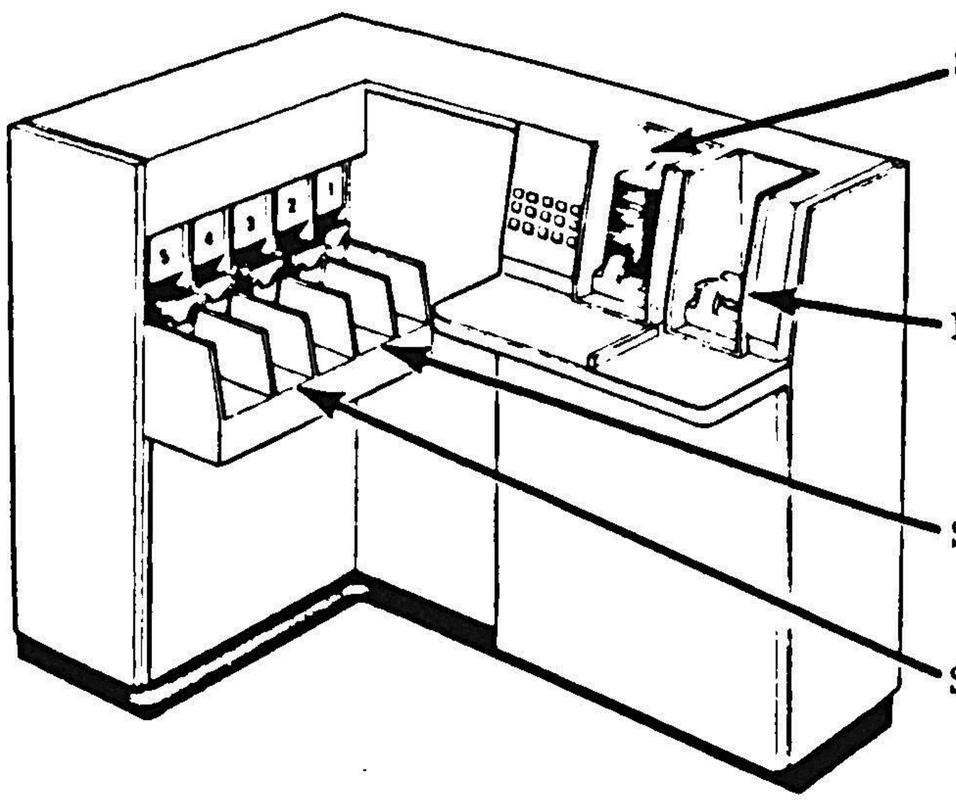
Example 2

In order to number an existing card deck in columns 75-80 with an initial value of 500,000 and an increment of 10 and to interpret the card numbers only, on the right side between row 11 and row 12, the control card would be:



Example 2 (continued)

The cards should be placed in the 2560 card reader/punch as follows:



Secondary Feed - Place here the existing deck which is to be numbered and interpreted followed by at least 3 blank cards.

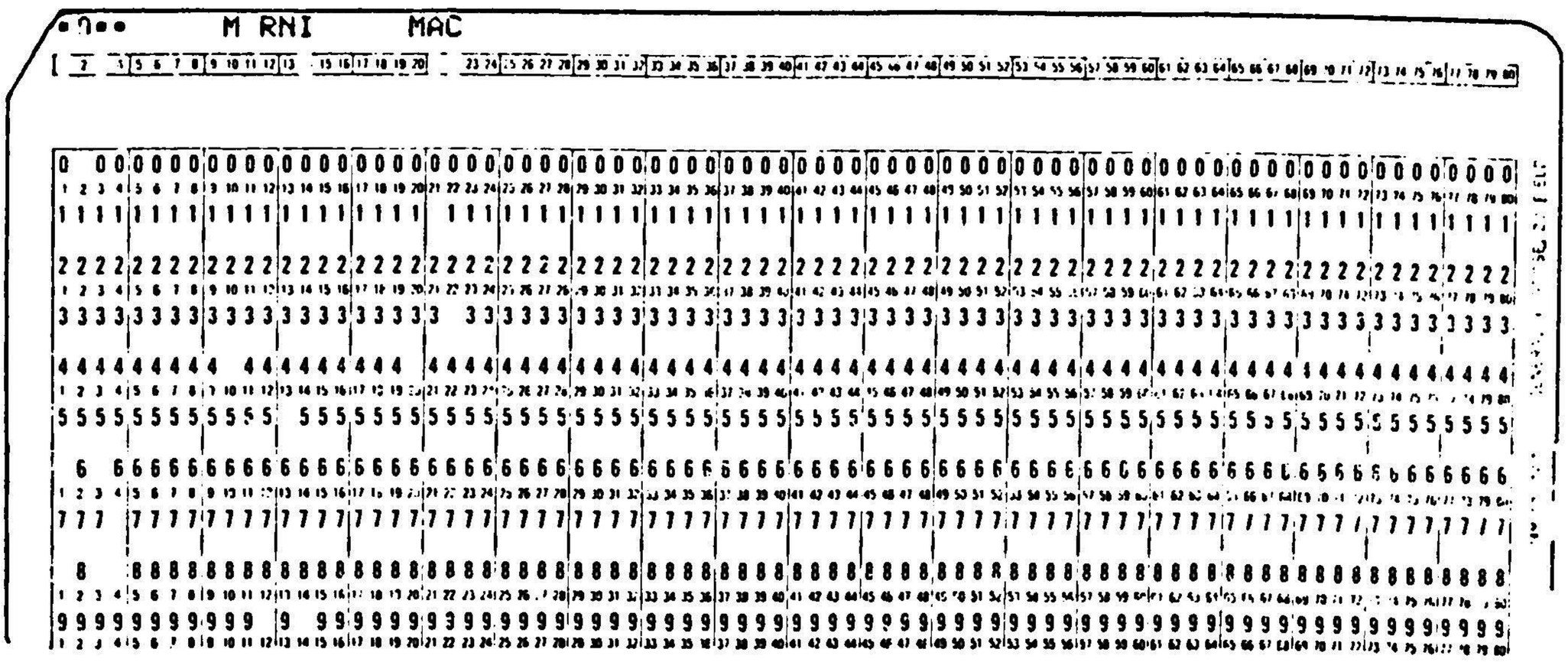
Primary Feed - Place here the control card followed by at least 3 blank cards.

Stacker 2 will contain the cards from the primary feed.

Stacker 4 will contain the cards from the secondary feed.

Example 3

In order to reproduce and list a MAC program deck and to interpret, and number the output deck in columns 2-7 with an initial value of zero and an increment of 100, the control card would be:



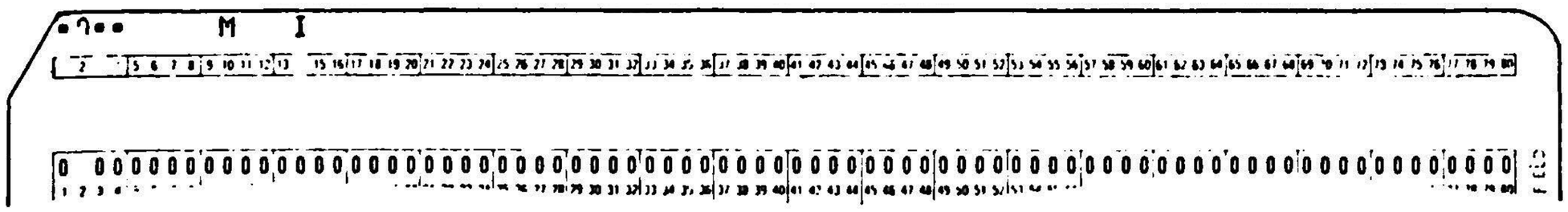
6.5.5 Interpreting

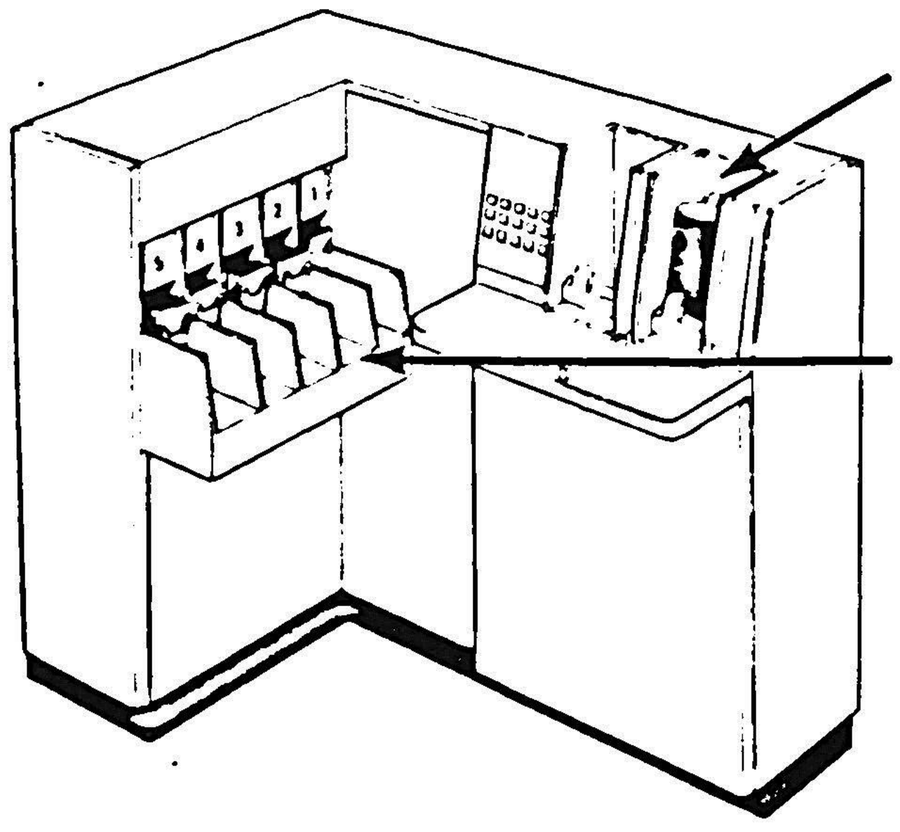
TABSIM can either punch and interpret a deck of cards or just interpret an existing deck. Column 14 of the control card determines whether standard interpreting takes place. Standard interpreting means that the first 64 columns of the card are interpreted above row 12 of the card and that columns 65-80 are interpreted right justified between row 11 and row 12 of the card. If column 14 contains an I and punching is specified (either column 12 or 13 is non-blank), the output deck is interpreted. If column 14 contains an I and no punching is specified (column 12 and 13 are both blank), the input is interpreted. If column 14 is blank, there is no standard interpreting.

There is one other type of interpreting available with TABSIM. If column 13 contains an R or an L (specifying numbering), the card number only is interpreted. The number is interpreted between row 11 and row 12 of the output cards either right justified or left justified depending on whether column 13 contains an R or an L.

Example 1

In order to interpret and list a MAC program deck, the control card would be:



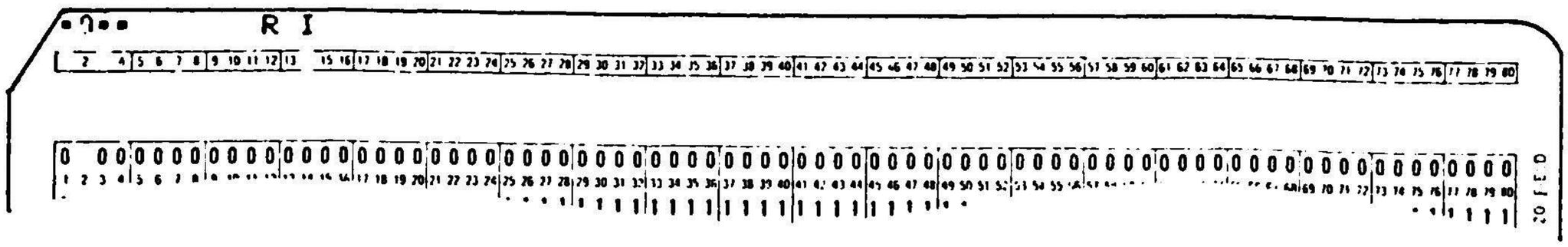


The program deck preceded by the control card and followed by at least 3 blank cards should be placed in the primary feed of the 2560 card reader/punch.

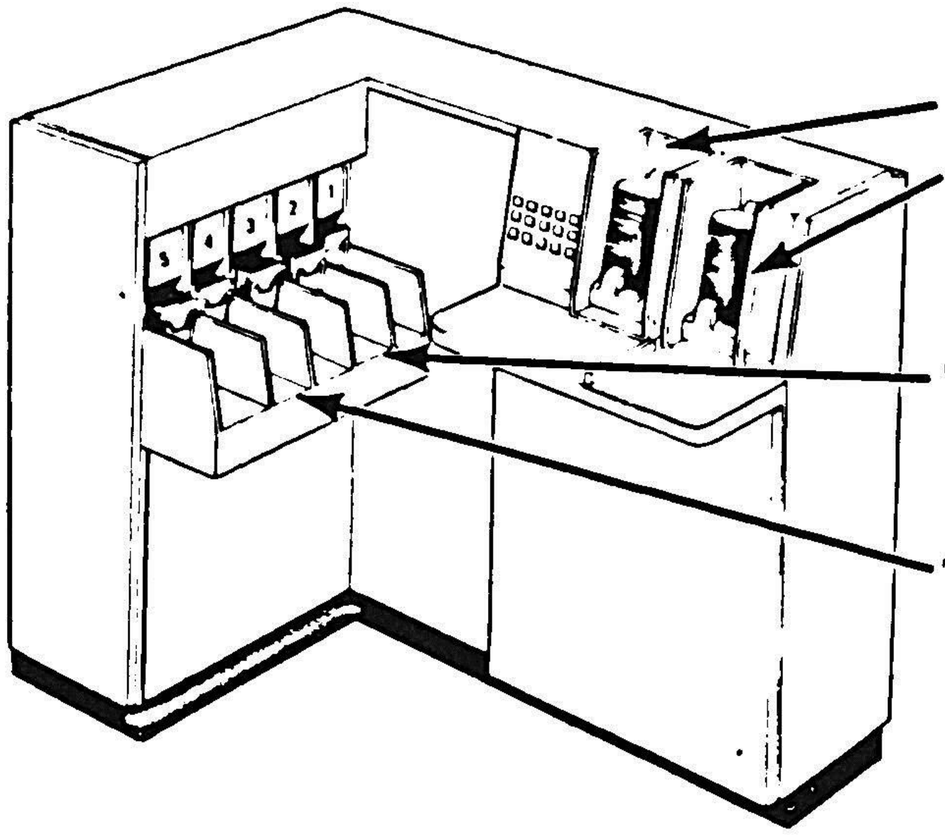
After the cards are read and interpreted they will be stacked in stacker 2.

Example 2

In order to reproduce a card deck, and interpret the output deck, the control card should be:



The cards should be placed in the 2560 card reader/punch as follows:



Secondary Feed - Place blank cards here.

Primary Feed - Place here the control card and the input deck followed by at least 6 blank cards.

The control card and input deck will be stacked in stacker 2 after they are read.

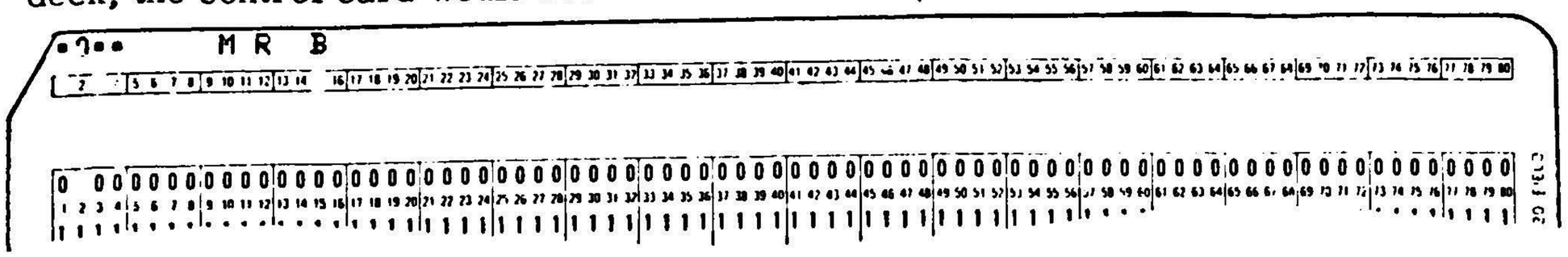
The output cards, after they are punched and interpreted will be stacked in stacker 4.

6.5.6 Mode of Input

TABSIM will accept input in either EBCDIC or BCD mode. Column 15 of the control card determines which mode. If column 15 is blank, TABSIM assumes the input deck is in EBCDIC mode. If column 15 contains a B, TABSIM assumes the input deck is in BCD mode and it converts each card image to EBCDIC before processing . Therefore, to list or interpret a card deck which is in BCD mode, the user must punch a B in column 15 of the control card. To reproduce a BCD card deck and have the output deck in BCD also, the user must leave column 15 blank. To reproduce and convert a BCD deck to EBCDIC column 15 must contain a B.

Example

In order to reproduce, converting from BCD to EBCDIC and list a MAC program deck, the control card would be:



6.5.7 Checking Output Cards Prior to Punching

TABSIM can check output cards to be sure all columns are blank, prior to punching. The card reader/punch on the IBM 360 Model 20 is capable of reading cards prior to punching them, which makes this feature of TABSIM possible. Column 16 of the control card determines if any checking takes place. If column 16 contains a C, all output cards are checked before punching. If a card is found that has a non-blank column, the card is ejected into stacker 3 and the data to be punched into that card is punched into the next blank card. If the user specifies 80x80 reproducing by putting an R in column 12 of the control card, this checking feature is turned on automatically.

6.5.8 Comparing

TABSIM can be used to compare two decks of cards. All 80 columns of the cards are compared . TABSIM can either compare two decks of cards and stop when it finds two cards that are not identical or compare and list the two decks and note on the listing when it finds two cards that are not the same. One deck is read through the primary feed and the other deck is read through the secondary feed. The control card must always be the first card of the deck read through the primary feed. Column 17 of the control card determines whether any comparing takes place. The acceptable operation codes for column 17 are as follows:

blank — No comparing

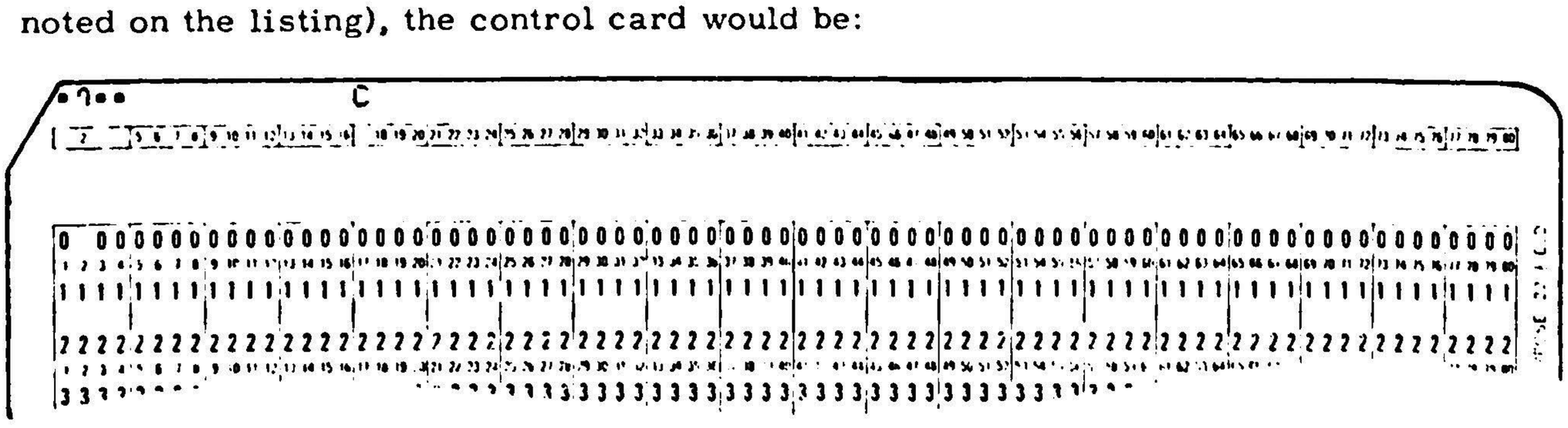
C — TABSIM reads and compares two card decks. If two cards are identical, the card image is printed once on the printer. If the two cards are not identical both card images are printed with the word "ERROR" printed to the right of each card image. The card read though the primary feed is printed first.

S — TABSIM reads and compares two decks of cards. If two cards are not identical, the computer stops. The non- matching card from the primary feed will be in stacker 1 and the non-matching card from the secondary feed will be in stacker 5. Before restarting the computer, the card reader/punch must be run out and the bad card fixed.

If TABSIM is used for comparing it can not interpret or punch during the run. Columns 10-16 of the control card must be blank.

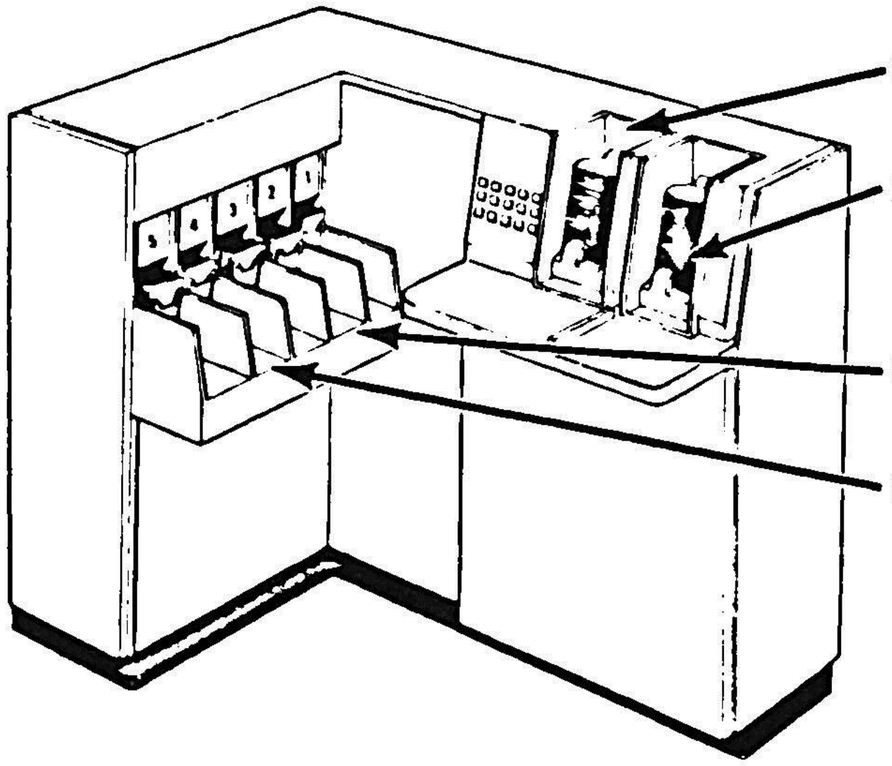
Example

To compare two card decks using the list feature (non-matching cards are noted on the listing), the control card would be:



Example (continued)

The cards should be placed in the 2560 card reader/punch as follows:

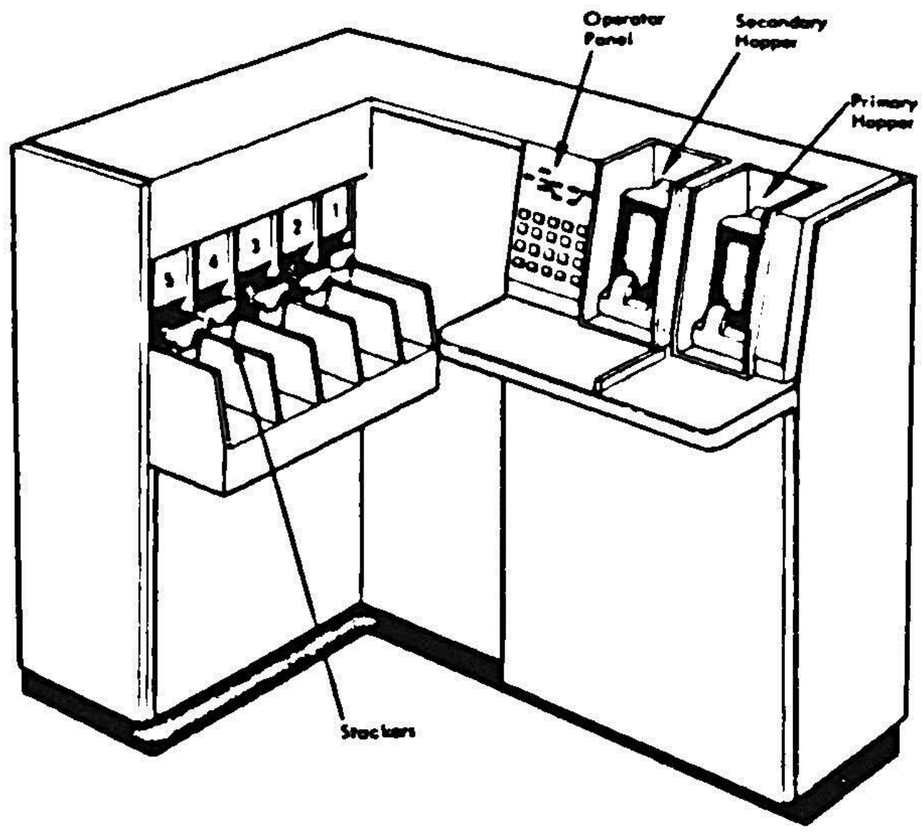


Secondary Feed - Place one card deck here followed by at least 3 blank cards.

Primary Feed - Place here the other card deck preceded by the control card and followed by at least 3 blank cards.

Stacker 2 will contain the cards from the primary feed.

Stacker 4 will contain the cards from the secondary feed.



The 2560 card reader/punch on the IBM 360 Model 20 computer has two feed hoppers. The computer is capable of reading or punching cards from either feed hopper. When using the program TABSIM, all cards to be punched must be placed in the secondary feed hopper and all cards to be read must be placed in the primary feed hopper except when using TABSIM for comparing two card decks. In this case one deck is read through the primary feed hopper and the other is read through the secondary feed hopper. Therefore, the control card and the parameter cards, if any, are always placed in the primary feed hopper.

The program TABSIM should be loaded in the IBM 360 Model 20 computer at all times. To start the program, proceed as follows:

1. Clear the card feed path of the 2560 multi-function card machine by pressing the nonprocess-runout (NPRO) key.

2. a. If the run is going to read an input deck and do no punching (when interpreting or listing), place the input deck preceeded by the control card and followed by at least three blank cards in the primary feed hopper of the 2560.

   b. If the run is going to read an input deck and punch an output deck (when reproducing, reproducing and numbering, reproducing and interpreting etc.) place the input deck preceded by the control card and the parameter cards, if any, in the primary feed of the 2560. Place a deck of blank cards in the secondary feed of the 2560.

   c. If the run is going to gangpunch or number an existing deck, place the control card and the parameter card, if any, followed by at least three blank cards in the primary feed of the 2560. Place the existing deck to be numbered or gangpunched , followed by at least three blank cards in the secondary feed of the 2560.

d. If the run is going to compare two card decks, place one deck preceded by the control card in the primary feed of the 2560. Place the other deck in the secondary feed of the 2560. Put at least three blank cards at the end of each deck.

3. If the run is going to print, press the start key on the printer.

4. Press the start key on the 2560 card reader/punch.

5. Press the System reset key and the Load key on the console. (As soon as you press the load key the program should start processing.)

If during the run, the 2560 card reader/punch runs out of cards, the computer will stop. To restart, fill the hopper that is empty and press the start key on the console.

At the completion of the run, the cards read through the primary feed, will be in stacker 2 and the cards read through the secondary feed will be in stacker 4. Destroy any cards that are in stacker 3. These are cards that were punched incorrectly and have been repunched.

If the stop light comes on before the completion of the run, one of the following error conditions may exist. The various error conditions are distinguished by displaying different hexadecimal characters in the E, S, T and R registers.

0 0 0 1 - This indicates an illegal control card.

0 1 1 1 - This indicates that the program has found two non-matching cards while comparing. The non-matching card from the primary feed will be in stacker 1 and the non-matching card from the secondary feed will be in stacker 5.

To Restart

1. Correct whichever card that is wrong.

2. Runout the 2560.

3. Reload the cards from stacker 1 into the primary feed and reload the cards from stacker 5 into the secondary feed.

4. Press Start on the 2560.

5. Press Start on the console.

0 F F 1 - This indicates a primary feed reader check. The card that caused the reader check should be in stacker 1.

To Restart

1. Check the card in stacker 1 to be sure that all columns contain legal punches.
2. Runout the 2560.
3. Reload the cards from stacker 1 into the primary feed and reload the cards from stacker 5 into the secondary feed.
4. Press Start on the 2560.
5. Press Start on the console.

0 F F 2 - This indicates a secondary feed reader check. The card that caused the reader check should be in stacker 5.

To Restart

1. Check the card in stacker 5 to be sure that all columns contain legal punches.
2. Runout the 2560 card reader/punch.
3. Reload the cards from stacker 1 into the primary feed and reload the cards from stacker 5 into the secondary feed.
4. Press Start on the 2560.
5. Press Start on the console.

0 F F F - This indicates a feed check.

To restart

1. Try to runout the 2560. If there are still cards in the card path of the 2560 they will have to be taken out manually. Do not try to do this without the help of a 360 computer operator.
2. a. If the run was just listing or interpreting, reload in the primary feed any cards from the card path that were not yet listed or interpreted.

   b. If the run was reading an input deck and punching an output deck, reload in the primary feed any card from the input deck that a output card has not been punched for. The last one or two cards in stacker 2 may have to be reloaded.

   c. If the run was comparing two decks, reload any cards that were in the card path in their proper feed. Be sure that the first cards in each feed are cards that you want compared to each other.

d. If the run was gangpunching or numbering an existing deck, reload in the secondary feed any cards in the card path that were not yet punched.

3. Press Start on the 2560.
4. Press Start on the console.

APPENDIX A

Listings of programs to initialize the manufacturing history data set and the master deck sequence number data set.

a) Program to initialize manufacturing history data set.

```
//GAP JOB 0375,HOPE,J    INITIALIZE MFG HISTORY
// EXEC GAPPMHST
/*
// EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//VOLUME1 DD VOLUME=SER=DCG003,UNIT=2314,DISP=OLD
//SYSIN DD *
        SCRATCH DSNAME=GAP.AGCMANU.HISTORY,VOL=2314=DCG003,PURGE
/*
// EXEC ASMFCLG
//ASM.SYSIN DD *
INMH     TITLE 'PROGRAM TO INITIALIZE MANUFACTURING HISTORY DATA SET.'
DEFNMHST CSECT
INITMHST SAVE  (14,12),,*
         BALR  12,0
         SPACE 2
         USING *,12
         SPACE 2
         ST    13,SAVE13
         OPEN  (MHISTDCB,(OUTPUT,DISP))
         PUT   MHISTDCB,RECORD0
         CLOSE (MHISTDCB)
         L     13,SAVE13
         RETURN (14,12),RC=0
         SPACE
         PRINT DATA
         SPACE 2
* INITIAL MANUFACTURING HISTORY DATA SET RECORD.
RECORD0  DC    CL120'                                              DATE OF INIC
               TIALIZATION - AUG.   6, 1969'
SAVE13   DS    F
         EJECT
* MANUFACTURING HISTORY DATA CONTROL BLOCK.
* QUEUED SEQUENTIAL ACCESS METHOD. MOVE-MODE PUT.
MHISTDCB DCB   DSORG=PS,                                           C
               MACRF=(PM),                                         C
               DDNAME=MFGHIST,                                     C
               DEVD=DA,                                            C
               OPTCD=W,                                            C
               RECFM=FB,                                           C
               LRECL=120,                                          C
               BLKSIZE=120,                                        C
               BFTEK=S,                                            C
               BUFNO=1,                                            C
               BFALN=D,                                            C
               ERROPT=ABE,                                         C
               SYNAD=HSTERRAD
         SPACE 2
HSTERRAD ABEND 777
         END   INITMHST
/*
//GO.MFGHIST DD DSNAME=GAP.AGCMANU.HISTORY,UNIT=2314,VOLUME=SER=DCG003,C
//              DISP=(,KEEP),SPACE=(TRK,(25,5))
//         EXEC GAPPMHST
/*
```

Notes:

1) The date field on the continuation card following the card labeled RECORDO should be changed to the date on which the program will be run. Do not shift or otherwise alter the apostrophe following the date.

2) If this program aborts with a User Code 777, a permanent write error was encountered in the attempt to initialize the Manufacturing History Data Set; consult a GAP System expert.

b) Program to initialize master deck sequence number data set.

```
//GAP      JOB 0375,HOPE,J          INITIALIZE MDSEQNO
// EXEC ASMFCLG
//ASM.SYSIN DD *
IMDN     TITLE 'PROGRAM TO INITIALIZE MASTER DECK SEQUENCE NUMBER.'
DEFNMDSN CSECT
INITMDSN SAVE  (14,12),,*
         BALR  12,0
         SPACE 2
         USING *,12
         SPACE 2
         ST    13,SAVE13
         OPEN  (MDSQNDCB,(OUTPUT,DISP))
         PUT   MDSQNDCB,RECORDO
         CLOSE (MDSQNDCB)
         L     13,SAVE13
         RETURN (14,12),RC=0
         SPACE
         PRINT DATA
         SPACE 2
* INITIAL MASTER DECK SEQUENCE NUMBER DATA SET RECORD.
* MASTER DECK SEQUENCE NUMBER STARTS AT RECORDO+77.
RECORDO  DC    CL120'                                                THE NEXT MASTC
               ER DECK SEQUENCE NUMBER IS 010012.'
SAVE13   DS    F
         EJECT
* MASTER DECK SEQUENCE NUMBER DATA CONTROL BLOCK.
* QUEUED SEQUENTIAL ACCESS METHOD, MOVE-MODE PUT.
MDSQNDCB DCB   DSORG=PS,                                             C
               MACRF=(PM),                                           C
               DDNAME=MSTDKSN,                                       C
               DEVD=DA,                                              C
               OPTCD=W,                                              C
               RECFM=FB,                                             C
               LRECL=120,                                            C
               BLKSIZE=120,                                          C
               BFTEK=S,                                              C
               BUFNO=1,                                              C
               BFALN=D,                                              C
               EROPT=ABE,                                            C
               SYNAD=MDNERRAD
         SPACE 2
MDNERRAD ABEND 666
         END   INITMDSN
/*
//GO.MSTDKSN DD DSNAME=GAP.AGCMANU.MDSEQNO,UNIT=2314,VOLUME=SER=DCG003,C
//              DISP=(,KEEP),SPACE=(TRK,(1,1))
/*
```

Notes:

1) The six-character numerical field on the continuation card following the card labeled RECORDO should be changed to the six-digit sequence number of the first master deck to be manufactured after this program has been run. This number must always contain six digits and must occupy the same columns as the number shown in the above program listing.

2) If the program aborts with a User Code 666, a permanent write error was encountered in the attempt to initialize the Master Deck Sequence Number Data Set; consult a GAP System expert.

APPENDIX B

Sample page of listing of manufacturing history data set.

AGC MANUFACTURING HISTORY

```
15:42 MAR. 17, 1969 REV 5 OF PROGRAM STARTRAK BY STG                    B36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
23:13 MAR. 17, 1969 REV 6 OF PROGRAM STARTRAK BY STG                     36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
19:47 MAR. 18, 1969 REV 89 OF PROGRAM LUMINARY BY 504GROUP              B36KCRS MAGTP TASK SUCCESSFULLY COMPLETED
12:40 MAR. 19, 1969 REV 51 OF PROGRAM COMANCHE BY NASA 2021113-031       MD#C10064     TASK SUCCESSFULLY COMPLETED
13:54 MAR. 19, 1969 REV 51 OF PROGRAM COMANCHE BY NASA 2021113-031       MD#C10065     TASK SUCCESSFULLY COMPLETED
11:44 MAR. 20, 1969 REV 7 OF PROGRAM STARTRAK BY STG                     36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
10:15 MAR. 21, 1969 REV 89 OF PROGRAM LUMINARY BY 504GROUP              B36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
 6:32 MAR. 22, 1969 REV 89 OF PROGRAM LUMINARY BY 504GROUP              B36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
10:16 MAR. 24, 1969 REV 9? OF PROGRAM LUMINARY BY 504GROUP              B36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
10:40 MAR. 24, 1969 REV 45 OF PROGRAM COMANCHE BY NASA 2021113-021       (360 ) COMPARED WITH
                    REV 2 OF PROGRAM QUALITY BY NON EXISTENT-ED         (360 ) DIFFERENCES:  96P,  5647W,M#S 123456,   OUM
10:42 MAR. 24, 1969 REV 51 OF PROGRAM COMANCHE BY NASA 2021113-031       (360 ) COMPARED WITH
                    REV 2 OF PROGRAM QUALITY BY NON EXISTENT-ED         (360 ) DIFFERENCES:   4P,    77W,M#S  2    ,   OUM
17:41 MAR. 24, 1969 REV 3 OF PROGRAM RUINED BY SEX AND VIOLENCE          36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
21:26 MAR. 24, 1969 REV 3 OF PROGRAM TAGLTEST BY LATIMER                B36KCRS BLK 0 TASK SUCCESSFULLY COMPLETED
21:28 MAR. 24, 1969 REV 4 OF PROGRAM YAHOO BY TURNBULL                   36KCRS MAGTP TASK SUCCESSFULLY COMPLETED
21:30 MAR. 24, 1969 REV 69 OF PROGRAM LUMINARY BY NASA 2021112-011       36KCRS MAGTP TASK SUCCESSFULLY COMPLETED
21:32 MAR. 24, 1969 REV 4 OF PROGRAM YAHOO BY TURNBULL                   MD#010066     TASK SUCCESSFULLY COMPLETED
21:34 MAR. 24, 1969 REV 69 OF PROGRAM LUMINARY BY NASA 2021112-011       MD#C10067     TASK SUCCESSFULLY COMPLETED
21:41 MAR. 24, 1969 REV 9? OF PROGRAM LUMINARY BY 504GROUP              B36KCRS MAGTP TASK SUCCESSFULLY COMPLETED
22:40 MAR. 24, 1969 REV 3 OF PROGRAM TAGLTEST BY LATIMER                B36KCRS BLK 0 TASK SUCCESSFULLY COMPLETED
22:42 MAR. 24, 1969 REV 4 OF PROGRAM YAHOO BY TURNBULL                   36KCRS MAGTP TASK SUCCESSFULLY COMPLETED
22:44 MAR. 24, 1969 REV 69 OF PROGRAM LUMINARY BY NASA 2021112-011       36KCRS MAGTP TASK SUCCESSFULLY COMPLETED
22:45 MAR. 24, 1969 REV 4 OF PROGRAM YAHOO BY TURNBULL                   MD#C10068     TASK SUCCESSFULLY COMPLETED
22:47 MAR. 24, 1969 REV 69 OF PROGRAM LUMINARY BY NASA 2021112-011       MD#C10069     TASK SUCCESSFULLY COMPLETED
 8:00 MAR. 25, 1969 REV 91 OF PROGRAM LUMINARY BY 504GROUP              B36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
10:17 MAR. 25, 1969 REV 91 OF PROGRAM LUMINARY BY 504GROUP              B36KCRS DGSTR NON-PROGRAM ABORT BEFORE COMPLETION
11:08 MAR. 25, 1969 REV 91 OF PROGRAM LUMINARY BY 504GROUP              B36KCRS MAGTP NON-PROGRAM ABORT BEFORE COMPLETION
11:54 MAR. 25, 1969 REV 91 OF PROGRAM LUMINARY BY 504GROUP              B36KCRS MAGTP TASK SUCCESSFULLY COMPLETED
12:13 MAR. 25, 1969 REV 91 OF PROGRAM LUMINARY BY 504GROUP              B36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
13:49 MAR. 25, 1969 REV 51 OF PROGRAM COMANCHE BY NASA 2021113-031       36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
11:14 MAR. 26, 1969 REV 51 OF PROGRAM COMANCHE BY NASA 2021113-031       36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
17:51 MAR. 26, 1969 REV 92 OF PROGRAM LUMINARY BY NASA 2021112-021       36KCRS MAGTP TASK SUCCESSFULLY COMPLETED
20:05 MAR. 26, 1969 REV 92 OF PROGRAM LUMINARY BY NASA 2021112-021      36KCRS DGSTR NON-PROGRAM ABORT BEFORE COMPLETION
20:18 MAR. 26, 1969 REV 92 OF PROGRAM LUMINARY BY NASA 2021112-021      36KCRS DGSTR NON-PROGRAM ABORT BEFORE COMPLETION
20:21 MAR. 26, 1969 REV 92 OF PROGRAM LUMINARY BY NASA 2021112-021      36KCRS DGSTR NON-PROGRAM ABORT BEFORE COMPLETION
20:26 MAR. 26, 1969 REV 92 OF PROGRAM LUMINARY BY NASA 2021112-021      36KCRS DGSTR NON-PROGRAM ABORT BEFORE COMPLETION
 6:19 MAR. 27, 1969 REV 92 OF PROGRAM LUMINARY BY NASA 2021112-021      36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
 8:58 MAR. 27, 1969 REV 92 OF PROGRAM LUMINARY BY NASA 2021112-021      36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
18:18 MAR. 27, 1969 REV 6 OF PROGRAM ONEIF BY LAND(WEINSTEIN)           B36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
18:43 MAR. 27, 1969 REV 5 OF PROGRAM YAHOO BY TURNBULL                   36KCRS MAGTP TASK SUCCESSFULLY COMPLETED
16:19 MAR. 28, 1969 REV 92 OF PROGRAM LUMINARY BY NASA 2021112-021       36KCRS BLU M TASK SUCCESSFULLY COMPLETED
18:30 MAR. 28, 1969 REV 51 OF PROGRAM COMANCHE BY NASA 2021113-031      (360 ) COMP  NON-PROGRAM ABORT BEFORE COMPLETION
20:57 MAR. 28, 1969 REV 69 OF PROGRAM LUMINARY BY NASA 2021112-011      PORTAFAM      TASK SUCCESSFULLY COMPLETED
21:21 MAR. 28, 1969 REV 51 OF PROGRAM COMANCHE BY NASA 2021113-031      (360 ) COMP  NON-PROGRAM ABORT BEFORE COMPLETION
 8:11 MAR. 29, 1969 REV 51 OF PROGRAM COMANCHE BY NASA 2021113-031      (360 ) COMP  NON-PROGRAM ABORT BEFORE COMPLETION
 8:15 MAR. 29, 1969 REV 51 OF PROGRAM COMANCHE BY NASA 2021113-031      (360 ) COMP  NON-PROGRAM ABORT BEFORE COMPLETION
13:00 MAR. 31, 1969 REV 92 OF SEGMENT LUMAGS BY STG                     36KCRS DGSTR TASK SUCCESSFULLY COMPLETED
14:44 MAR. 31, 1969 REV 2 OF PROGRAM LUM69 BY NASA 2021112-031          (360 ) COMPARED WITH
```

ABBREVIATIONS EMPLOYED:

a) Manufacturing Task Codes:

ST - Symbol Table

MD#nnnnnn - Master Deck Number nnnnnn where nnnnnn is a six-digit Master Deck Sequence Number.

ST&MD#nnnnnn - Symbol Table and Master Deck Number nnnnnn where nnnnnn is a six-digit Master Deck Sequence Number.

36KCRS - 36K Core Rope Simulator Tape

36KBRD - 36K Braid Wiring Tape

PORTAFAM - PORTAFAM Tape

b) Output Media Codes:

PNK O - Pink oiled paper tape
BLK O - Black oiled paper tape
BLU M - Blue paper-MYLAR tape
GRY F - Grey fiber tape
ALUM - Aluminum tape
DGSTR - DIGISTORE (magnetic) tape
MAGTP - Seven track magnetic tape

c) Comparison Task Codes:

COMP - Compare. (This abbreviation occurs only when the comparison job was aborted.)

nnnP - Total number of paragraphs that differed, where nnn is a one to three digit integer.

nnnnnW - Total number of words that differed, where nnnnn is a one to five digit integer.

M#Sddddddd - The number(s), (from one to six) of the Core Rope Module(s) that contained differing words; here a "d" represents a blank or a module number (1 for the first "d", 2 for the second "d", up to 6 for the sixth "d"). For example M#S 1 4 6 means that Core Rope Modules, one, four, and six contained words that differed in the comparison.

nnnUM - Total number of paragraphs found which did not possess counterparts with matching paragraph numbers in the other data set. nnn is a one to three digit integer.

(360) - Machine on which program was assembled. (This will always be an IBM System 360 machine for the forseeable future.)

Notes:

1) Columns 1 through 19 contain the time and date of manufacture.

2) Columns 21 through 67 contain the identification of the AGC program that was manufactured.

3) Column 69 contains the letter B if the AGC program that was manufactured was flagged as bad when it was assembled.

4) Columns 70 through 81 contain the abbreviated Manufacturing Task Code and/or Output Medium Code.

5) Columns 83 through 119 contain the message

NON-PROGRAM ABORT BEFORE COMPLETION

if the manufacture failed and was aborted for any reason. If the manufacture was successful, the message

TASK SUCCESSFULLY COMPLETED

is displayed.

6) When the task is COMPARE, the Manufacturing History includes a second line which contains the identification of the second AGC program in the comparison and the Comparison Task Codes.

APPENDIX C

USER COMPLETION CODES AND MESSAGES

Return Codes Set in GAP Manufacture AGC Control Section.

User Code	Reason for Termination	User Action
0001	Manufacturing Communication Table does not contain "AGC " for computer name	Consult GAP System Expert
0002	First (control) record identification character is not X'80'.	" "
0003	Input data set is not an AGC program	" "
0004	"AUTHMEMO" field of first record is incorrect	" "
0005	Revision number in first record differs from revision number in Manufacturing Communication Table.	" "
0006	Program name in first record differs from program name in Manufacturing Communication Table.	" "
0007	Program name field in first record is blank.	" "
0008	Author's name field in first record is blank.	" "
0009	Author's name in first record differs from author's name in Manufacturing Communication Table.	Punch correct 16-character author's name in Mfg Director Card and rerun job.
0010	Second (control) record identification character is not X'c0'.	Consult GAP System Expert
0011	Second record identification field (program name, computer type, device name character, revision number) differs from program data set identification.	" "
0013	Symbol table record identification character is not X'00'.	" "

User Code	Reason for Termination	User Action
0014	Symbol table record contains incorrect program name.	Consult GAP System Expert.
0015	"SYMBOLS " field of a symbol table record is incorrect.	" "
0016	An entire paragraph record of fixed switchable memory contains only unused AGC words.	" "
0017	An AGC word contains the characters "BADW" (bad word).	" "
0018	An AGC word contains the characters "CONF" (conflict).	" "
0019	Paragraph record identification character is not X'20'.	" "
0020	Paragraph record is missing or out of sequence.	" "
0021	Paragraph record contains incorrect program name.	" "
0022	Paragraph number in record differs from the paragraph number specified in the Paragraph Presence Indicators.	" "
0023	No manufacturing task has been specified. This error can be caused only by a machine failure or an error in YUL GENERAL.	" "
0024	A paragraph requested for manufacture (specified in the Output Paragraph Presence Indicators) is missing from the input data set.	" "
0025	Error in Compare Task; the first data set named in the Manufacturing Communication Table was not assembled on the IBM 360 Model 75.	" "
0055	An attempt to find the location of the data set to be manufactured failed because of a LOCATE error.	Consult GAP System Expert. Make sure the AGC program has been assembled and catalogued.
0056	Data set is contained on more than one volume.	Consult GAP System Expert.

User Code	Reason for Termination	User Action
0057	Either (1) the AGC data set to be manufactured has been located on a device other than the assembly disk AGCB02 but no DD card describing this device is present in the catalogued procedure GAPMAGC (or GAPMAGCA); or (2) the AGC data set is on magnetic tape but the AGCTAP DD cards were not appended to the catalogued procedure GAPMAGC (or GAPMAGCA). The error message "THE AGC ASSEMBLY AGCPROG.dsname. revision number THAT YOU ARE ATTEMPTING TO PROCESS COULD NOT BE FOUND ON THE ASSEMBLY DISK" is printed on the job output.	Consult GAP System Expert.
0058	Input AGC data set failed to open properly and the associated DCB was not successfully completed.	" "
0123	This is the normal return code from the first job step if the manufacturing task was PUNCH 36K CORE ROPE SIMULATOR TAPE.	No action required.
0124	This is the normal return code from the first job step if the manufacturing task was PUNCH 36K BRAID WIRING TAPE.	" "
0999	The end-of-file of an input data set was read and the end-of-data-set exit of the corresponding DCB was taken.	Consult GAP System Expert.
1000	A record of an input data set was found to be unreadable and the SYNAD exit of the corresponding DCB was taken. The error message "UNREADABLE DATA SET RECORD. RERUN THIS JOB. IF ERROR PERSISTS INFORM ASSEMBLY CONTROL OF THE PROBLEM." is printed on the job output.	Rerun the job. Consult GAP System expert if the failure is repeated.
1999	An attempt to write a record on the PORTAFAM output magnetic tape resulted in a permanent I/O error and the SYNAD exit of the PORTAFAM DCB was taken.	Rerun the job with another magnetic tape. If error persists, consult GAP System Expert.
2000	An attempt to write a Manufacturing History Record resulted in a permanent I/O error. The SYNAD exit of the Manufacturing History DCB was taken.	Consult GAP System Expert.

User Code	Reason for Termination	User Action
2999	An end-of-file was encountered when an attempt was made to read the Master Deck Sequence Number data set. The EODAD exit of the Master Deck Sequence Number DCB was taken.	Consult GAP System Expert.
3000	An attempt to read or rewrite the Master Deck Sequence Number data set record resulted in a permanent I/O error. The SYNAD exit of the Master Deck Sequence Number DCB was taken.	" "
	The message "|||||***WARNING. WARNING. YOU ARE ATTEMPTING TO PROCESS A BAD ASSEMBLY. WARNING. WARNING.***||||| " is printed in the job listing if an attempt is made to process an AGC program that was flagged as a bad assembly by the GAP Assembler. If the manufacturing task was PUNCH MASTER DECK, PUNCH 36K BRAID WIRING TAPE, or WRITE PORTAFAM TAPE, the job was aborted.	User should heed warning and be fully conscious of what he is doing. In case the warning was unexpected, a GAP System Expert should be consulted.
	The message "MANU. INPUT PROG. SYSTEM/ 360. RESIDES ON DIRECT ACCESS DEVICE." is printed in the job listing if the AGC program data set has been successfully located on the assembly disk and opened. If the task is COMPARE the message is repeated for the second data set involved in the comparison if it is successfully located and opened. If the AGC data set was assembled and stored on magnetic tape, the words "DIRECT ACCESS DEVICE" in the message are replaced by "360 TAPE". If the Device Name character in the first record of the data set is not recognized by GAP Manufacturing the message "MANU. INPUT PROG. DEVICE NAME NOT FOUND." is printed in the job listing.	None required, except that if the "DEVICE NAME NOT FOUND." message is printed, a GAP System Expert should be informed.
	In each case when the job is aborted because of an error, the computer operator gets the console message "GAP MANUFACTURING ABORT".	

Return Codes Set in GAP Manufacture General Control Section

User Code	Reason for Termination	User Action
1998	An attempt to write a record in the card output data set (Symbol Table or Master Deck card images on magnetic tape) has resulted in a permanent I/O error. The SYNAD exit of the card output data set DCB (PUNCHDCB) was taken.	Rerun job with fresh reel of magnetic tape. If error persists, consult GAP System Expert.
1999	An attempt to write a record in the paper tape output temporary data set (36K Core Rope Simulator Tape buffer output on direct access device) resulted in a permanent I/O error. The SYNAD exit of the paper tape output DCB (PAPERDCB) was taken.	Consult a GAP System Expert.

Return Codes Set in GAP Manufacture Paper Tape Punch Program Control Section

User Code	Reason for Termination	User Action
1001	End-of-data-set exit (EODAD) of the input data set DCB (PAPERDCB) was taken.	Consult GAP System Expert.
1002	Permanent read error encountered while reading the input data set. The SYNAD exit of the input data set DCB (PAPERDCB) was taken.	" "
1997	An attempt to write a record on magnetic tape (36K Core Rope Simulator Tape output on magnetic tape) resulted in a permanent I/O error. The SYNAD exit of the magnetic tape output DCB (MAGTPDCB) was taken.	Rerun job with fresh reel of magnetic tape. If error persists, consult GAP System Expert.
2000	A permanent I/O error was encountered while reading or writing in the Manufacturing History Data set. The SYNAD exit of the Manufacturing History DCB (MHISTDCB) was taken.	Consult GAP System Expert.
3333	Unable to properly open paper tape DCB in on-line paper tape punch utility.	" "

User Code	Reason for Termination	User Action
3344	An uncorrectable paper tape punch error has occurred. The operator also receives the console message "GAP MANUFACTURING ABORT. UNCORRECTABLE PAPER PUNCH ERROR."	Rerun job. If error recurs, consult GAP System Expert. IBM customer engineer should be informed if hardware trouble is suspected.
3355	An uncorrectable DIGISTORE error has occurred. The operator also receives the console message "GAP MANUFACTURING ABORT. UNCORRECTABLE DIGISTOR ERROR."	Rerun job. If error recurs, consult GAP System Expert. DIGISTORE hardware expert (Robert Pinckney) should be notified if hardware trouble is suspected.
3366	This error return occurs in a portion of the program that will read the DIGISTORE tape backwards and check it against the records from which it was written. Since this section of the program is not used at the present time, this error should never occur.	Consult GAP System Expert.
	The operator may receive the console message "PRESS DIGISTOR PERMIT BUTTON. TYPE "GO" WHEN READY."	Operator must follow instructions in message.

Error Messages Written in Job Output Listing by Master Deck File Duplicating and Tape Checking Program.

User Code	Message and Reason for Termination	User Action
1000	"PERMANENT I/O ERROR IN MASTER DECK TAPE FILE f. ERROR RECORD IS SHOWN BELOW. eighty character error record printed in EBCDIC form", where f is the number of the magnetic tape file that contains the error record ($1 \leq f \leq 4$). This message is printed if an attempt to read a card image record from the Master Deck Tape results in a permanent I/O error.	Rerun the job with a fresh reel of magnetic tape. If error persists, consult a GAP System Expert.

User Code	Message and Reason for Termination	User Action
1010	"PERMANENT I/O ERROR IN MASTER DECK DISK FILE. ERROR RECORD IS SHOWN BELOW. eighty character error record printed in EBCDIC form". This message is printed if an attempt to read a card image record from the Master Deck data set stored as a temporary file on a 2314 disk results in a permanent I/O error.	Rerun the job. If error persists, consult GAP System Expert. If hardware trouble is suspected inform an IBM customer engineer.
1020	"ERROR. THIS MASTER DECK CONTAINS AN ERASABLE BANK." Master Deck tapes manufactured for release must not contain erasable banks.	Inform Assembly Control of the situation.
1030	"ERROR. BAD AGC WORD PARITY IN MASTER DECK. BANK ADDRESS = bank address BANK NUMBER = bank number PARAGRAPH ADDRESS = paragraph address PARAGRAPH NUMBER = paragraph number AGC WORD = AGC word GIVEN PARITY = parity bit given with above AGC word THE ABOVE NUMBERS ARE HEXADECIMAL." This message is printed if a (used) AGC word is detected whose associated parity bit results in an even word parity.	Inform Assembly Control of the problem.
1040	"ERROR IN CHECKSUM ARITHMETIC." This error is caused by a faulty program branch in the checksum recomputation (branch is in statement following statement 140). The error should never occur.	If this error occurs, it implies that something is wrong with the Tape Checking Program. Consult GAP System Expert.
1050	"ERROR. AGC BANK CHECKSUM IS INCORRECT. BANK ADDRESS = bank address BANK NUMBER = bank number PARAGRAPH ADDRESS = paragraph address PARAGRAPH NUMBER = paragraph number AGC CHECKSUM WORD = AGC bank checksum word COMPUTED CHECKSUM = computed bank checksum THE ABOVE NUMBERS ARE HEXADECIMAL." This message is printed if the computed AGC bank checksum is not equal to either the AGC bank number or its one's complement.	Inform Assembly Control of the problem.

User Code	Message and Reason for Termination	User Action
1060	"ERROR. NON-ZERO WORD FOUND IN BANK AFTER CHECKSUM WORD. BANK NUMBER = bank number PARAGRAPH ADDRESS = paragraph address PARAGRAPH NUMBER = paragraph number NON-ZERO AGC WORD = (used) AGC word THE ABOVE NUMBERS ARE HEXADECIMAL." The Tape Checking Program looks for non-zero words following the AGC bank checksum word in the remainder of the paragraph. If a non-zero word is found the above message is printed in the job output.	Inform Assembly Control of the problem.
1070	"ERROR. COMPARISON FAILURE BETWEEN MASTER DECK TAPE FILE f AND DISK FILE. TAPE RECORD IS SHOWN BELOW, FOLLOWED BY DISK RECORD. eighty character tape record printed in EBCDIC form eighty character disk record printed in EBCDIC form", where f is the number of the magnetic tape file that contains the suspect record (1≦f≦4). The Tape Checking Program checks each record of each Master Deck tape file by reading the record from the magnetic tape and comparing the record with the corresponding record read from the temporary Master Deck data set file stored on the 2314 disk. If the two records differ, the above message is printed in the job output.	Rerun the job. If error persists, consult a GAP System Expert. If hardware trouble is suspected inform an IBM customer engineer.

AGC Program Identification

Symbol Table Header Card Image

Count of number of Symbol Table Card Images (decimal)

SYMBOL TABLE 0046 AGC REVISION 0 OF PROGRAM MANUTASK BY LATIMER

004155	-0CHK	004166	-1CHK	000000	A	006656	ADDCHK
030060	ADRS+1	006440	ADRSBZMF	006441	ADRSDCA	030061	ADRSDV1
006442	ADRSQXCH	004054	ADRS1	004055	ADRS2	004056	ADRS3
004057	ADRS4	004060	ADRS5	004061	ADRS6	004062	ADRS7
000010	ARUPT	004774	AUGCHK	002407	BANK0-1	000006	BB
000016	BBRUPT	004063	BNKINIT	002410	BNKNMBR	000017	BRUPT
005153	BZFCHK	005175	BZMFCHK	004212	CCSCHK	000013	CHAN13
000014	CHAN14	000003	CHAN3	000004	CHAN4	004623	CNTINU
000022	CYL	006713	CYLCHK	006724	CYLCNTDN	006717	CYLLOOP
000020	CYR	006672	CYRCHK	006703	CYRCNTDN	030042	CYRCON
006676	CYRLOOP	004434	DAS++	004450	DAS++DV	004466	DAS+--+
004506	DAS--	004522	DAS--DV	005023	DIMCHK	005362	DV1++
005376	DV1+-	005405	DV1-+	005412	DV1--	030056	DV1CON
005422	DV2++	005435	DV2+-	005445	DV2-+	005451	DV2--
030057	DV2CON	005464	DV3++	005472	DV3+-	005501	DV3-+
005505	DV3--	005515	DV4++	005527	DV4+-	005537	DV4-+
005545	DV4--	005561	DV5++	005571	DV5+-	005602	DV5-+
005614	DV5--	005631	DV6++	005645	DV6+-	005653	DV6-+
005663	DV6--	005673	DV7++	005704	DV7+-	005712	DV7-+
005721	DV7--	005730	DV8++	005742	DV8+-	005752	DV8-+
005762	DV8--	000003	EB	002413	EBKEEP	002412	EBSTART
000023	EDOP	006747	EDOPCHK	006753	EDOPLOOP	006726	ENDCYL
006705	ENDCYR	007002	ENDEDOP	002414	EOPTION	004147	ERRORS
003730	ETADRS	000004	FB	000770	FLAG	005773	IN-OUT1
006306	IN-OUT2	002417	KEEP1	002420	KEEP2	002421	KEEP3
002422	KEEP4	002423	KEEP5	002424	KEEP6	002425	KEEP7
000001	L	000011	LRUPT	005220	MP1++	005225	MP1+-
005231	MP1-+	005235	MP1--	005244	MP2++	005251	MP2+-
005255	MP2-+	005261	MP2--	005270	MP3++	005275	MP3+-
005302	MP3-+	005307	MP3--	005314	MP4++	005321	MP4+-
005325	MP4-+	005332	MP4--	005337	MP5++	005345	MP5+-
005351	MP5-+	005356	MP5--	004251	MSKCHK	002401	NDX+MAX
002400	NDX+0	006544	NDXAUG	006451	NDXBZMF	004571	NDXCHK
006455	NDXDCA	006553	NDXDIM	006503	NDXDV	006562	NDXINOUT
002402	NDXKEEP1	002403	NDXKEEP2	002404	NDXKEEP3	006524	NDXMSU
006533	NDXOXCH	002405	NDXSELF1	002406	NDXSELF2	000054	OCSCDU
000050	OCXCDU	000051	OCYCDU	000002	Q	002416	QADRES
002415	QKEEP	004751	QNMBR	000012	QRUPT	004756	QXCHCON1
004757	QXCHCON2	006016	RANDCHK	006041	RANDOV	006323	RAND2
002411	RMMBR	006123	RORCHK	006154	RORDV	006353	ROR2
006245	RXORCHK	006276	RXORDV	006417	RXOR2	030035	S+MAX
030017	S+ZERO	030020	S+1	030021	S+2	030027	S+2B
030022	S+3	030023	S+4	030024	S+5	030025	S+6
030026	S+7	030036	S-MAX	030055	S-ZERO	030054	S-1
030045	S-14	030044	S-15	030053	S-2	030052	S-3
030051	S-4	030050	S-5	030047	S-6	030046	S-7
030000	SBIT1	030011	SBIT10	030012	SBIT11	030013	SBIT12
030014	SBIT13	030015	SBIT14	030016	SBIT15	030001	SBIT2
030002	SBIT3	030003	SBIT4	030004	SBIT5	030005	SBIT6
030006	SBIT7	030007	SBIT8	030010	SBIT9	004611	SELF1
004617	SELF2	004673	SELF3	030040	SEVEN	030037	SINOUT1
030041	SINOUT2	030043	SINOUT3	030034	SODD	000021	SR
006732	SRCHK	006745	SRCNTDN	006736	SRLOOP	004177	START
004704	STRTXTRA	030032	S11BITS	030033	S13BITS	030030	S6BITS
030031	S7BITS	004204	TCCHK	000027	TIME4	000031	TIME6
006054	WANDCHK	006104	WANDOF	006334	WAND2	006170	WORCHK
006223	WORDV	006375	WOR2	006443	XTRANDX	000005	Z
000015	ZRUPT						

Symbol Table Card Images

Symbol Table Trailer Card Image

END OF SYMBOL TABLE REVISION 0 OF PROGRAM MANUTASK BY LATIMER

LISTING OF AGC MASTER DECK

Master Deck Header Card Image
Paragraph Header Card Image

```
L0100864AGC  MASTER DECK     TIME 22.52     1ST PARAGRAPH 010          MAY  14, 1969
R0100862REVISION 99 OF PROGRAM LUMINARY BY NASA 2021112-051   PARAGRAPH 010#1A05
 010086  0000040 0340541 0560061 0126670 0520110 0000061 0340560 0520060 0001A05
 010086  0520110 0000061 0312750 0520060 0520110 0340571 0560061 0134070 0101A05
 010086  0520110 0340641 0560061 0120001 0520110 0340600 0560061 0132741 0201A05
0100862  0520110 0340611 0560061 0123320 0520110 0340600 0560061 0133170 0301A05
 010086  0520110 0340621 0560061 0135060 0520110 0340630 0560061 0131501 0401A05
 010086  0520110 0340650 0560061 0122751 8121030 8020650 8361060 8021031 0501A05
 010086  8101001 8161070 8121000 8521001 8141060 8221021 8040251 8100030 0601A05
0100862  8140310 8200330 8240171 8300361 8340341 8400231 8440351 8500370 0701A05
 010086  8540000 8600001 0000040 0000061 0000040 0540011 0000061 0060040 1001A05
 010086  0000061 0141151 0000061 0000040 0540011 0447330 0600010 0000061 1101A05
 010086  0141030 0000061 0000031 0000020 0341441 0710221 0000061 0141311 1201A05
0100862  0041320 0310201 0540231 0300230 0046161 8623371 0347551 0550130 1301A05
 010086  0310220 0042551 0041430 0051550 8376000 0043641 0410411 0550130 1401A05
 010086  0046161 8623371 0046350 8620021 0220070 0541230 0347361 0710211 1501A05
 010086  0610120 0100000 0000020 0347531 0600020 0550370 0230220 0042040 1601A05
0100862  0342010 0560061 0000061 0040071 0550400 0342010 0000061 0010071 1701A05
 010086  0020000 8621010 0530400 0051650 0347351 0550211 0000020 0220020 2001A05
 010086  0042200 0042241 0300010 0750120 0600040 0550421 0051330 0051550 2101A05
 010086  0110421 0042271 0000020 0042271 0110430 0042271 0000020 0056520 2201A05
0100862  8012061 0342010 0000061 0010071 0342421 0550130 0443601 0547771 2301A05
 010086  0046350 8614501 8000421 0342010 0000061 0010071 0342540 0550130 2401A05
 010086  0443601 0547771 0046350 8614421 8000411 0747571 0541230 0347361 2501A05
 010086  0710211 0610120 0100000 0000020 0240020 0101230 0142701 0000020 2601A05
0100862  0220020 0343021 0560061 0000061 0040071 0521310 0342010 0000061 2701A05
 010086  0010071 0035051 8601011 0550171 0540030 0743570 0650070 0541450 3001A05
 010086  0000020 0310170 0143040 0540030 0743570 0650070 0000020 8000160 3101A05
 010086  8000111 8000040 0540201 0400201 0400201 0400201 0400201 0560200 3201A05
0100962  0000020 0540220 0400220 0400220 0400220 0400220 0560221 0000020 3301A05
 010086  0600001 0600001 0600001 0600001 0600001 0000020 8000370 8017400 3401A05
 010086  8760000 0050721 0052030 0052611 0051050 8300001 8038770 8003771 3501A05
 010086  8000230 8000211 8000250 8000121 0347450 0000061 0050111 0000020 3601A05
0100862  0447451 0000061 0030111 0000020 0347471 0000061 0050111 0000020 3701A05
```

Paragraph Trailer Card Image

```
R0100868        END OF PARAGRAPH 010      NEXT PARAGRAPH 011
```

LISTING OF AGC MASTER DECK

```
$0100862REVISION 99 OF PROGRAM LUMINARY BY NASA 2021112-051   PARAGRAPH 011#1B05
 010086  0000061 0300250 0520140 0000020 0000061 0301560 0201561 0601541 0001B05
 010086  0261540 0540071 0000020 054162C C000020 0541351 0100000 0301350 0101B05
 010086  0173100 0144170 0447550 0541540 0541551 0541561 0000020 034746C 0201B05
 0100862 0000061 0050111 0000020 0447461 0000061 0030111 00C0020 0444440 0301B05
 010086  0600020 0600040 0046350 8106470 8020030 0220070 0541230 0347361 0401B05
 010086  0710211 0610120 0100000 0144550 C141641 0240020 0144371 0560020 0501B05
 010086  0541441 044736C 0000040 0710211 0550211 0110430 0044700 0044730 0601B05
 0100862 0347551 0570431 0051371 0000031 0447470 0000061 0030111 0347551 0701B05
 010086  0550121 0001440 0560020 0541441 0110430 0045071 0044730 0347551 1001B05
 010086  0550121 0001440 0000061 0645151 0145160 0400000 0500020 0600001 1101B05
 010086  0000061 0667410 0167370 0000061 0500020 0500001 0300010 0240020 1201B05
 0100862 0520621 0220020 0347400 0701100 0100000 0145501 0300610 0045120 1301B05
 010086  8661611 0000010 0345621 0600620 0045120 8644200 0000010 0045600 1401B05
 010086  0345630 0600620 0045120 8656730 0000010 0300610 0045120 8516151 1501B05
 010086  0500010 0000010 8007650 8164501 0347531 0713031 0100000 0000020 1601B05
 0100862 0347441 0540011 0346151 0701100 0000061 0146010 0347501 0701100 1701B05
 010086  0100000 0300010 0610360 0700011 0000061 0167421 0310360 0000061 2001B05
 010086  0060010 0747330 0647351 0550361 0000020 8101020 0521340 0500020 2101B05
 010086  0300001 0240020 0540011 0220040 0750120 0560020 0521340 0500020 2201B05
 0100862 0020000 0561341 0560040 0561341 0001330 0560020 0500001 0300001 2301B05
 010086  0540041 0750120 0560020 0500020 0120001 0350121 0701331 0601341 2401B05
 010086  0000020 0541351 0560040 0000061 0040071 0561350 0750120 0560010 2501B05
 010086  0000040 0000061 0010071 0500010 0320000 0561350 0000061 0010071 2601B05
 0100862 0000031 0540041 0301350 0000020 0520731 0500020 0300001 0240020 2701B05
 010086  0540011 0220040 0750120 0560020 0520731 0500020 0020000 0560730 3001B05
 010086  0560040 0560730 0000721 0541640 0347441 0540231 0300061 0541651 3101B05
 010086  0500020 0300001 0540041 0750120 0560020 0561641 0146430 0000061 3201B05
 0100862 0010071 0000020 8377771 8377771 8577771 8400000 8200000 8100000 3301B05
 010086  8040000 8020000 8010000 8004000 8002000 8001000 8000400 8000200 3401B05
 010086  8000100 8000040 8000020 8000010 8777770 8000001 8000051 8000070 3501B05
 010086  8000130 8000150 8000171 8000301 8000331 8000351 8000320 8000450 3601B05
 0100862 8000460 8000501 8000551 8000601 8000620 8001201 8001401 8001440 3701B05
$0100868       END OF PARAGRAPH 011      NEXT PARAGRAPH 012
```

Master Deck Sequence Number

Carriage Control Character for IBM 407 listing purposes

000 thru 377 are the octal addresses of the first AGC words of each card image relative to the first AGC word of the paragraph

Mfg Location Code

```
R01C0862REVISION 99 OF PROGRAM LUMINARY BY NASA 2021112-051   PARAGRAPH 237#6B12
 C10086  0540011 0240010 0500001 0520011 0513770 0400011 0513770 0600001 0006B12
 C10086  0032731 0313600 0000061 0134370 0000061 0513770 0400011 0513770 0106B12
 010086  0520011 0513770 0400000 0513770 0600010 0032731 0313600 0000061 0206B12
 0100862 0134370 0000061 0313761 0513770 0520011 0347551 0553601 0000031 0306B12
 010086  0033320 0313740 0540030 0253770 0413771 0613731 0000061 0134510 0406B12
 010086  0033670 0113721 0033620 0253720 0300031 0647430 0540030 0632431 0506B12
 010086  0000061 0133550 0100030 0033501 0350070 0540030 0347711 0553721 0606B12
 0100862 0647501 0500001 0400000 0113721 0034671 0332410 0540201 0540220 0706B12
 010086  0540210 0540231 0600200 0600221 0600211 0600230 0632420 0032731 1006B12
 010086  0600200 0600221 0600211 0600230 0647531 0032731 0253671 0033010 1106B12
 010086  0347540 0553760 0347551 0553741 0347531 0553771 0347551 0553711 1206B12
 0100862 0553730 0347531 0553750 0313740 0540011 0743501 0613731 0046511 1306B12
 010086  0035631 0647411 0035741 0400000 0553771 0000061 0135511 0347400 1406B12
 010086  0035520 0350200 0553730 0347551 0553711 0347531 0553750 0513731 1506B12
 010086  0300001 0035631 0035741 0553721 0613710 0553711 0347551 0613710 1606B12
 0100862 0553711 0413721 0613731 0000020 0220001 0313731 0750120 0643500 1706B12
 010086  0000061 0136701 0113750 0036061 0036061 0036700 0100011 0036160 2006B12
 010086  0036160 0036160 0113750 0036171 0377471 0036171 0347531 0553750 2106B12
 010086  0113760 0100671 0051220 0036250 0033320 0253731 0113771 0035331 2206B12
 0100862 0035331 0035570 0035570 0413741 0637231 0000061 0131310 0313740 2306B12
 010086  0647411 0553741 0036460 0347351 0273741 0036620 0743501 0000061 2406B12
 010086  0136600 0632401 0000061 0136550 0036620 0332360 0273741 0036620 2506B12
 010086  0632371 0273741 0113771 0035250 0347531 0035430 0347450 0035250 2606B12
 0100862 0313740 0743501 0043311 0540011 0313740 0743570 0000061 0137051 2706B12
 010086  0540210 0300010 0747571 0600211 0540011 0313761 0000061 0137111 3006B12
 010086  0031131 0113711 0037140 0037151 0647531 0553711 0400011 0613710 3106B12
 010086  0677471 0032731 0036331 8661C00 0347551 0137331 0347531 0137331 3206B12
 0100862 0347520 0137331 0362451 0551660 0020761 0310110 0000061 0137421 3306B12
 010086  0046350 8117621 0310110 0551710 0311661 0551720 0347531 0551731 3406B12
 010086  0046350 8114011 8037521 8037530 8672331 0000000 0000000 0000000 3506B12
 010086  0000000 0000000 0000000 0000000 0000000 0000000 0000000 0000000 3606B12
 0100862 0000000 0000000 0000000 0000000 0000000 0000000 0000000 0000000 3706B12
```

Master Deck Trailer Card Image → R0100868 END OF PARAGRAPH 237 END OF AGC MASTER DECK

Notes:

1) Word 354 (8672331) is the Bank Checksum Constant.

2) Words 355 through 377 are unused AGC words. (Even parity!!)

3) Each seven-digit AGC word group is composed as follows:
   a) The first (decimal) digit is a Word Type Code;
   b) The next five (octal) digits comprise the AGC word itself;
   c) The last digit is the AGC word parity (odd);

4) The symbol # in the Paragraph Header Card Image will be printed as an = sign if a BCD listing of the Master Deck Tape is made (on, say, an IBM 1401 computer);

5) If the paragraph is an erasable bank, the Manufacturing Location Code field contains the characters EBKb, where b is the (octal) bank number.

Manufacturing Location Code

The Manufacturing Location Code is written in the form xyz.

x represents the Core Rope Module Number, a digit from 1 to 6.

y stands for either of the letters A or B specifying side A or side B of the Core Rope Module.

z is the two-digit Sense Line Set Number which ranges from 01 to 12 (decimal).

For example, 1B05 means Rope Module 1, Side B, Sense Line Set 05.

AGC Word Type Codes

0 A basic instruction, but not complemented or leftover.

9 Interpretive operation code word.

8 A constant or a complemented or leftover basic instruction

4 Same as for 0. Instruction's address refers to an unused location in fixed-fixed memory.

REVISION 99 OF PROGRAM LUMINARY BY NASA 2021112-051

BANK	PARAGRAPHS OCCUPIED	VACANT	CHECKSUM LOCATION	CONSTANT VALUE	CHECKSUM
02(FF)	010,011,012,013		5777	67044 0	00002
03(FF)	014,015,016,017		7760	61751 0	00003
00	020,021,022,023		3776	77716 1	77777
01	024,025,026,027		3777	55151 1	00001
04	040,041,042,043		3773	12532 0	77773
05	044,045,046,047		3774	44523 0	00005
06	050,051,052,053		3766	44057 0	00006
07	054,055,056,057		3775	66702 1	00007
10	060,061,062,063		3777	56105 0	00010
11	064,065,066,067		3777	43373 1	00011
12	070,071,072,073		3777	73411 1	00012
13	074,075,076,077		3771	21126 1	77764
14	100,101,102,103		3776	56616 0	00014
15	104,105,106,107		3777	47112 0	00015
16	110,111,112,113		3774	10415 0	77761
17	114,115,116,117		3746	64555 0	00017
20	120,121,122,123		3774	41116 1	00020
21	124,125,126,127		3772	67517 0	00021
22	130,131,132,133		3775	00471 0	77755
23	134,135,136,137		3771	57135 1	00023
24	140,141,142,143		3777	76466 1	00024
25	144,145,146,147		3772	10540 1	77752
26	150,151,152,153		3777	73435 1	00026
27	154,155,156,157		3774	45310 1	00027
30	160,161,162,163		3777	71361 0	00030
31	164,165,166,167		3763	53641 1	00031
32	170,171,172,173		3775	71663 1	00032
33	174,175,176,177		3777	07450 1	77744
34	200,201,202,203		3777	60264 1	00034
35	204,205,206,207		3777	44737 1	00035
36	210,211,212,213		3767	53320 0	00036
37	214,215,216,217		3776	47535 1	00037
40	220,221,222,223		3715	70762 0	00040
41	224,225,226,227		3733	72012 1	00041
42	230,231,232,233		3774	56666 1	00042
43	234,235,236,237		3756	67233 1	00043

NOTE: A bank checksum must always equal the bank number or the one's complement of the bank number.

APPENDIX F

Output Formats of 36K Core Rope Simulator Tapes

The following notation is used:

1) An Erasable Bank number is represented in binary as $E_0E_1E_2$; where each one of the E's is a 0 or a 1. For example, Erasable Bank 6 is 110 in binary.

2) A Fixed Bank Number is specified in binary by the Fixed Extension Bits $x_0x_1x_2$ and the Fixed Bank Bits $F_0F_1F_2F_3F_4$, where each of the x's and F's is a 0 or 1. The Fixed Extension Bits 000 are used for Fixed Banks 0 through 37 (octal). Banks 40 through 43 (octal) use the Fixed Extension Bits 100 and Fixed Bank Bits 11000 through 11011 respectively. For example, Bank 32 is written

   000 (Fixed Extension Bits) 11010 (Fixed Bank Bits);

   Bank 42 becomes

   100 (Fixed Extension Bits) 11010 (Fixed Bank Bits).

   In any case, for Banks 40 (octal) or greater, $x_0x_1x_2$ is formed from the <u>first</u> three bits of the Bank Number; F_0F_1 is set to 11 (binary) and $F_2F_3F_4$ is formed from the <u>last</u> three bits of the Bank Number.

3) An AGC word is written in binary as

   $$B_0B_1B_2B_3B_4B_5B_6B_7B_8B_9B_{10}B_{11}B_{12}B_{13}B_{14}W$$

   where W and each of the B's is a 0 or a 1. W is the (odd) AGC word parity bit.

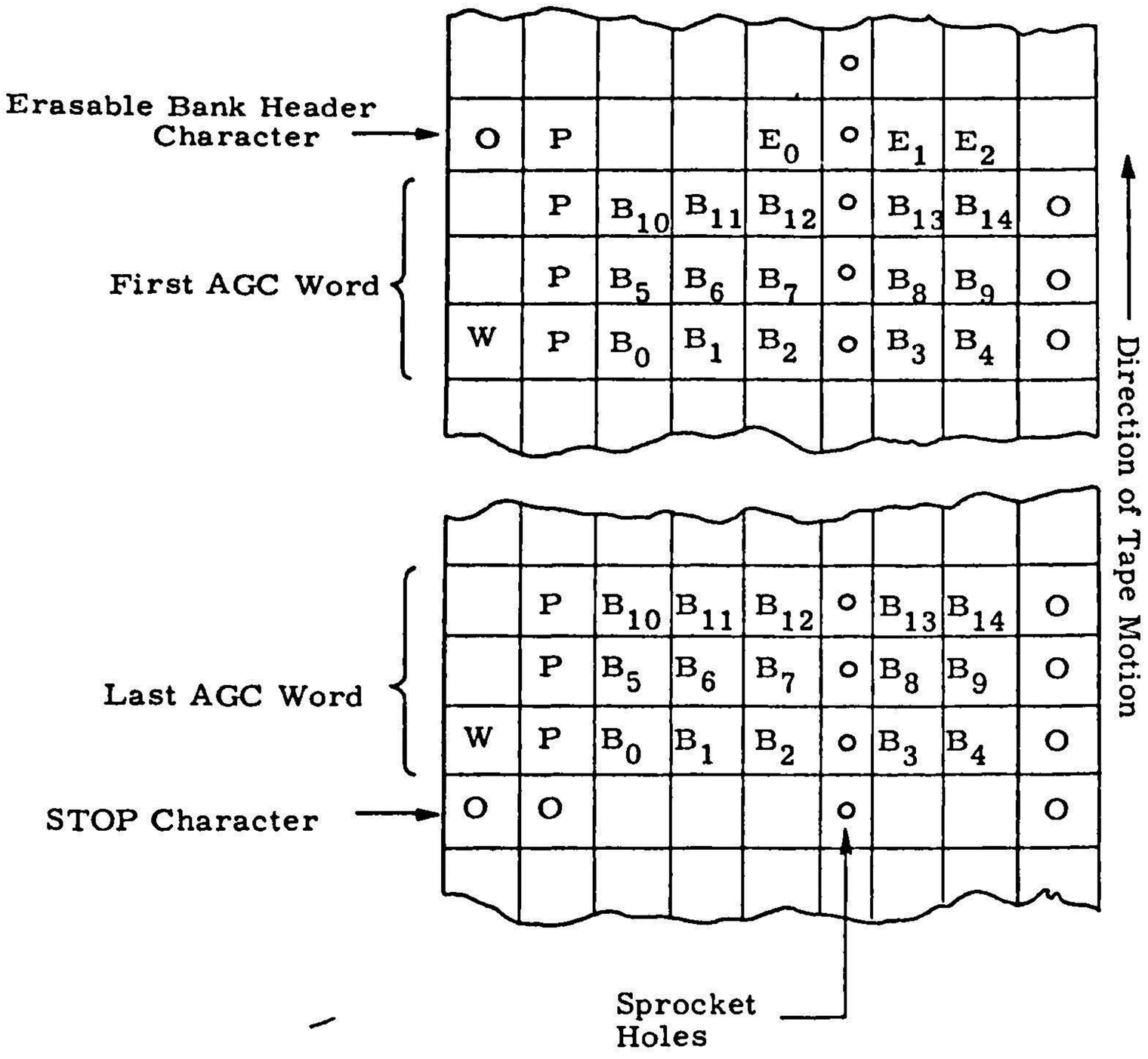
4) An <u>unused</u> AGC word that will appear on DIGISTORE tape or seven-track magnetic tape has the binary form

   000000000000000 (even parity!)

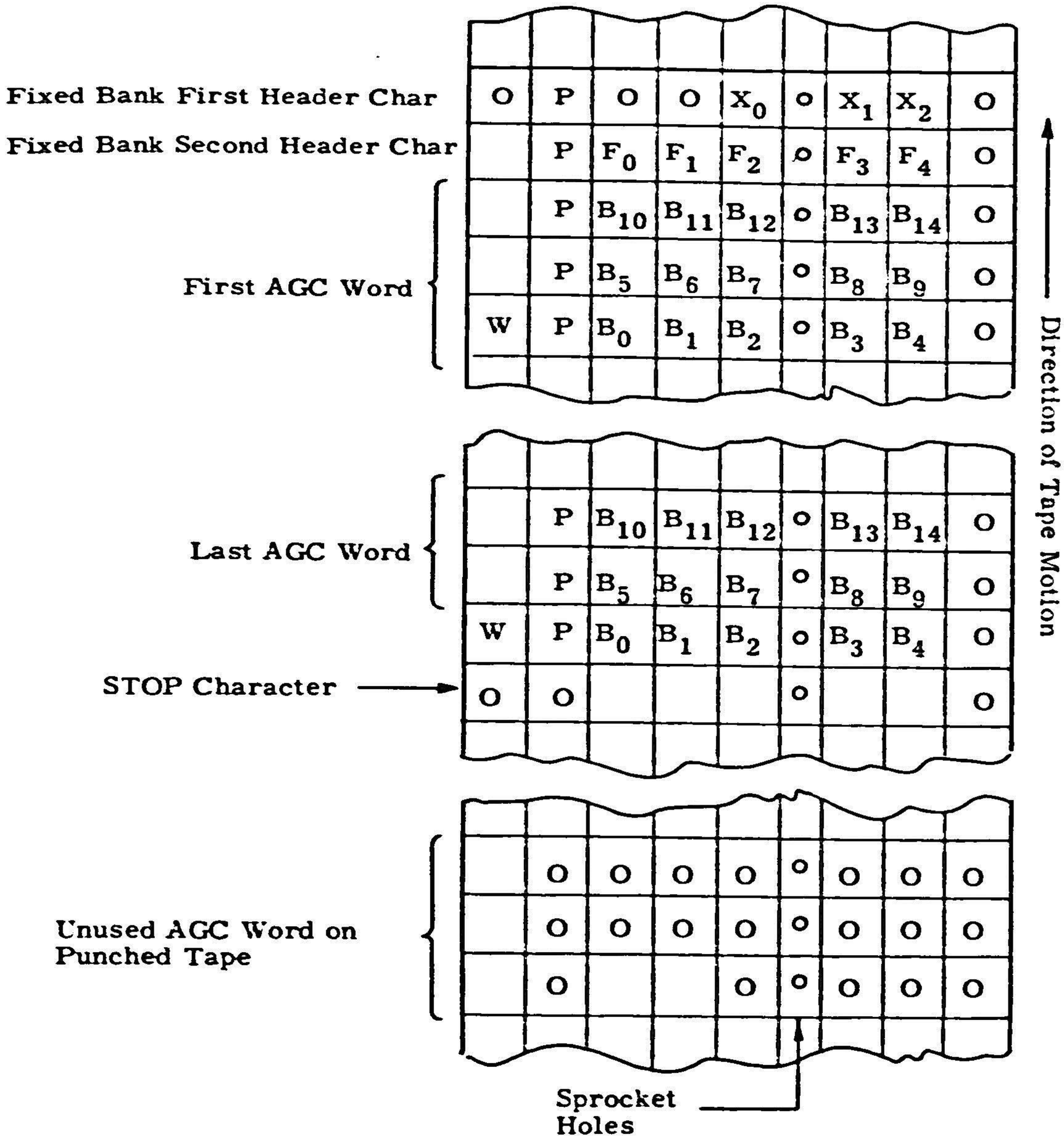
   An <u>unused</u> AGC word that will appear on <u>punched</u> tape is converted to the binary form 001111111111110 (odd parity).

Punched Output Formats

1) Erasable Bank



2) Fixed Bank


Fixed Bank First Header Char
Fixed Bank Second Header Char
First AGC Word
Last AGC Word
STOP Character
Unused AGC Word on Punched Tape
Sprocket Holes
Direction of Tape Motion

Notes:

a) 0 depicts the sprocket hole.

b) O represents a hole punched in the tape (binary 1).

c) A binary 1 is represented by punching a hole in the tape at the given position, a binary O by <u>not</u> punching the tape.

d) P stands for the (odd) parity bit for each eight-bit paper tape frame

e) <u>Each</u> Erasable or Fixed Bank is preceded and followed by the AGC program identification and Bank Number punched into the tape in the form of human readable alphanumeric characters.

DIGISTORE Output Format

DIGISTORE tape formats resemble punched tape output formats, except that;

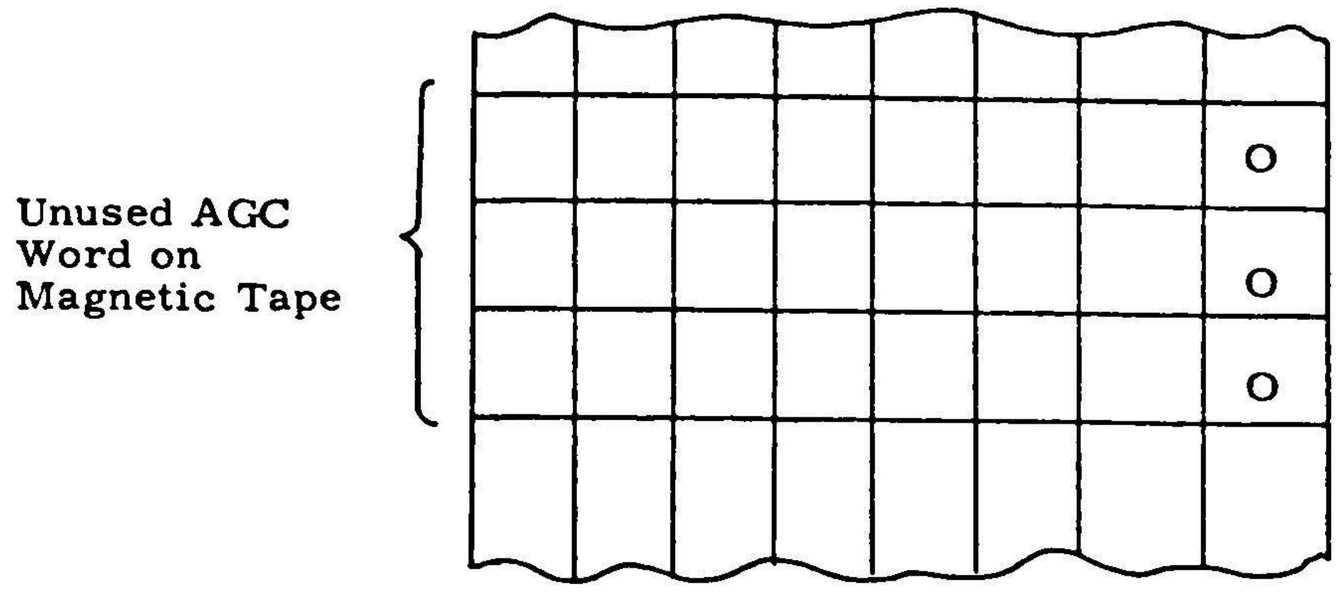
a) There is no sprocket hole that appears in the body of each eight-bit frame.

b) O represents a magnetized spot rather than a punch.

c) A binary one is represented by a magnetized spot.

d) No AGC program identification and Bank Number precedes or follows any bank written on the tape.

e)



Seven-track Magnetic Tape Output Formats

Seven-track magnetic tape output formats are similar to DIGISTORE tape formats. The principal differences are:

a) A non-return-to-zero (NRZ) magnetic tape recording scheme is used.

b) Each group of three eight-bit characters is split and formed into a group of four six-bit characters which are then written on the magnetic tape.

c) The seventh (c) track of the magnetic tape is the standard IBM tape (odd) parity channel.

APPENDIX G

Sample of Binary Comparison Output

COMPARISON OF REVISION 2 OF PROGRAM MANCHE45 BY NASA 2021113-041 (360) APR. 10, 1969
WITH REVISION 45 OF PROGRAM COMANCHE BY NASA 2021113-021 (360) TIME 6:25

BANK 11 PARAGRAPH 066 ROPE MODULE 2, SIDE A, SENSE LINE SET 08 (WIRES 113-128)

BANK ADDRESS	0	1	2	3	4	5	6	7	PARA ADDR
11.3000	=	=	=	=	=	=	=	=	000
11.3010	=	=	=	=	=	=	=	=	010
11.3020	=	=	=	=	=	=	=	=	020
11.3030	=	=	=	=	=	=	=	=	030
11.3040	=	=	=	=	=	=	=	=	040
11.3050	=	=	=	=	=	=	=	=	050
11.3060	=	=	=	=	=	=	=	=	060
11.3070	=	=	=	=	=	=	=	=	070
11.3100	=	=	=	=	=	=	=	=	100
11.3110	=	=	=	=	=	=	=	=	110
11.3120	=	=	=	=	=	=	=	=	120
11.3130	401610 777511	=	231331 000331	=	=	=	536700 430701	=	130
11.3140	201531 003430	400550 232131	022020 635450	227621 000171	362021 635250	277701 000211	777161 652151	412060 000031	140
11.3150	277550 453521	605250 003031	442151 415250	000111 000230	116130 653610	412050 000250	000171 453161	277550 237460	150
11.3160	565610 524051	000171 277550	777251 523611	432250 021520	116130 725610	000111 777251	140111 000171	000171 632050	160
11.3170	412050 000211	000230 021521	277550 742350	433521 000250	603250 533320	023101 777251	000501 413010	672061 000501	170
11.3200	017750 000010	742710 742050	532571 277530	570011 532571	702570 576050	201460 000331	653250 777541	015170 021501	200
11.3210	277550 776001	451560 232121	021501 532571	553441 201531	400550 022020	=	227621 776001	022020 227621	210
11.3220	=	=	=	=	=	=	=	=	220
11.3230	=	=	=	=	=	=	=	=	230
11.3240	=	=	=	=	=	=	=	=	240
11.3250	=	=	=	=	=	=	=	=	250
11.3260	=	=	=	=	=	=	=	=	260
11.3270	=	=	=	=	=	=	=	=	270
11.3300	=	=	=	=	=	=	=	=	300
11.3310	=	=	=	=	=	=	=	=	310
11.3320	=	=	=	=	=	=	=	=	320
11.3330	=	=	=	=	=	=	=	=	330
11.3340	=	=	=	=	=	=	=	=	340
11.3350	=	=	=	=	=	=	=	=	350
11.3360	=	=	=	=	=	=	=	=	360
11.3370	=	=	=	=	=	=	=	=	370

Notes:

1) In a typical word pair, such as 401610 777611, the first word (401610) comes from Revision 2 of MANCHE45 and the second word (777611) from Revision 45 of COMANCHE.

2) The last (sixth) digit of each word is the parity digit.

3) To clarify the process of reading addresses, observe the following:

   a) the (unequal) AGC word pair 231331 000331 is located in Bank 11, Bank Address 3132, Relative Paragraph Address 132;

   b) the word pair 536700 430701 is in Bank 11, Bank Address 3136, Relative Paragraph Address 136.

4) See Appendix B for a discussion of the Comparison Statistics, paritcularly in reference to the Rope Module Number(s) field.

COMPARISON OF REVISION 2 OF PROGRAM MANCHE45 BY NASA 2021113-041 (360) APR. 10, 1969
WITH REVISION 45 OF PROGRAM COMANCHE BY NASA 2021113-021 (360) TIME 0125

BANK 11 PARAGRAPH 067 ADDR MODULE 2. SIDE B. SENSE LINE SET 08 (WIRES 113-128)

BANK ADDRESS	0	1	2	3	4	5	6	7	PARA ADDR
11,3400	=	=	=	=	=	=	=	=	000
11,3410	=	=	=	=	=	=	=	=	010
11,3420	=	=	=	=	=	=	=	=	020
11,3430	=	=	=	=	=	=	=	=	030
11,3440	=	=	=	=	=	=	=	=	040
11,3450	=	=	=	=	=	=	=	=	050
11,3460	=	=	=	=	=	=	=	=	060
11,3470	=	=	=	=	=	=	=	=	070
11,3500	=	=	=	=	=	=	=	=	100
11,3510	=	=	=	=	=	=	=	=	110
11,3520	=	=	=	=	=	=	=	=	120
11,3530	=	=	=	=	=	=	=	=	130
11,3540	=	=	=	=	=	=	=	=	140
11,3550	=	=	=	=	=	=	=	=	150
11,3560	=	=	=	=	=	=	=	=	160
11,3570	=	=	=	=	=	=	=	=	170
11,3600	=	=	=	=	=	=	=	=	200
11,3610	=	=	=	=	=	=	=	=	210
11,3620	=	=	=	=	=	=	=	=	220
11,3630	=	=	=	=	=	=	=	=	230
11,3640	=	=	=	=	=	=	=	=	240
11,3650	=	=	=	=	=	=	=	=	250
11,3660	=	=	=	=	=	=	=	=	260
11,3670	=	=	=	=	=	=	=	=	270
11,3700	=	=	=	=	=	=	=	=	300
11,3710	=	=	=	=	=	=	=	=	310
11,3720	=	=	=	=	=	=	=	=	320
11,3730	=	=	=	=	=	=	=	=	330
11,3740	=	=	=	=	=	=	=	=	340
11,3750	=	=	=	=	=	=	=	=	350
11,3760	=	=	=	=	=	=	=	=	360
11,3770	=	=	=	=	=	=	=	660531 646610	370

COMPARISON OF REVISION 2 OF PROGRAM MANCHE45 BY NASA 2021113-041 (360) APR. 10, 1969
WITH REVISION 45 OF PROGRAM COMANCHE BY NASA 2021113-021 (360) TIME 6:25

BANK 12 PARAGRAPH 073 ROPE MODULE 2, SIDE B, SENSE LINE SET 10 (WIRES 145-160)

BANK ADDRESS	0	1	2	3	4	5	6	7	PARA ADDR
12,3400	=	=	=	=	=	=	=	=	000
12,3410	=	=	=	=	=	=	=	=	010
12,3420	=	=	=	=	=	=	=	=	020
12,3430	=	=	=	=	=	=	=	=	030
12,3440	=	=	=	=	=	=	=	=	040
12,3450	=	=	=	=	=	=	=	=	050
12,3460	=	=	=	=	=	=	=	=	060
12,3470	=	=	=	=	=	=	=	=	070
12,3500	=	=	=	=	=	=	=	=	100
12,3510	=	=	=	=	=	=	=	=	110
12,3520	=	=	=	=	=	=	=	=	120
12,3530	=	=	=	=	=	=	=	=	130
12,3540	=	=	=	=	=	=	=	=	140
12,3550	=	=	=	=	=	=	=	=	150
12,3560	=	=	=	=	=	=	=	=	160
12,3570	=	=	=	=	=	=	=	=	170
12,3600	=	=	=	=	=	=	=	=	200
12,3610	=	=	=	=	=	=	=	=	210
12,3620	=	=	=	=	=	=	=	=	220
12,3630	=	=	=	=	=	=	=	=	230
12,3640	=	=	=	=	=	=	=	=	240
12,3650	=	=	=	=	=	=	=	=	250
12,3660	=	=	=	=	=	=	=	=	260
12,3670	=	=	=	=	=	=	=	=	270
12,3700	=	=	=	=	=	=	=	=	300
12,3710	=	=	=	=	=	=	=	=	310
12,3720	=	=	=	=	=	=	=	=	320
12,3730	=	=	=	=	=	=	=	=	330
12,3740	=	=	=	=	=	=	=	=	340
12,3750	=	=	=	=	635250 037541	000211 037550	776250 512150	742050 000000	350
12,3760	277550 000000	000171 003300	653320 003000	000171 000000	433420 000000	000031 000000	140031 000000	000211 000000	360
12,3770	443421 000000	000051 003300	033051 000000	743351 000030	017741 000000	435251 000000	000230 000000	357011 000000	370

COMPARISON OF REVISION 2 OF PROGRAM MANCHE45 BY NASA 2021113-041 (360) APR. 10, 1969
WITH REVISION 45 OF PROGRAM COMANCHE BY NASA 2021113-021 (360) TIME 6:25

BANK 13 PARAGRAPH 077 ROPE MODULE 2, SIDE B, SENSE LINE SET 12 (WIRES 177-1921

BANK ADDRESS	0	1	2	3	4	5	6	7	PARA ADDR
13,3400	=	=	=	=	=	=	=	=	000
13,3410	=	=	=	=	=	=	=	=	010
13,3420	=	=	=	=	=	=	=	=	020
13,3430	=	=	=	=	=	=	=	=	030
13,3440	=	=	=	=	=	=	=	=	040
13,3450	=	=	=	=	=	=	=	=	050
13,3460	=	=	=	=	=	=	=	=	060
13,3470	=	=	=	=	=	=	=	=	070
13,3500	=	=	=	=	=	=	=	=	100
13,3510	=	=	=	=	=	=	=	=	110
13,3520	=	=	=	=	=	=	=	=	120
13,3530	=	=	=	=	=	=	=	=	130
13,3540	=	=	=	=	=	=	=	=	140
13,3550	=	=	=	=	=	=	=	=	150
13,3560	=	=	=	=	=	=	=	=	160
13,3570	=	=	=	=	=	=	=	=	170
13,3600	=	=	=	=	=	=	=	=	200
13,3610	=	=	=	=	=	=	=	=	210
13,3620	=	=	=	=	=	=	=	=	220
13,3630	=	=	=	=	=	=	=	=	230
13,3640	=	=	=	=	=	=	=	=	240
13,3650	=	=	=	=	=	=	=	=	250
13,3660	=	=	=	=	=	=	=	=	260
13,3670	=	=	=	=	=	=	=	=	270
13,3700	=	=	=	=	=	=	=	=	300
13,3710	=	=	=	=	=	=	=	=	310
13,3720	=	=	=	=	=	=	=	=	320
13,3730	=	=	=	=	=	=	=	=	330
13,3740	=	=	776516 000001	762370 000001	=	=	=	=	340
13,3750	=	=	=	=	=	=	=	=	350
13,3760	=	=	=	=	=	=	=	=	360
13,3770	712140 037701	000436 037713	232200 747701	000171 [illegible]	000000 257540	000000 037761	000000 655271	000000	370

COMPARISON OF REVISION 2 OF PROGRAM COMANCHE45 BY NASA 2021113-041 (360) APR. 10, 1969
WITH REVISION 45 OF PROGRAM COMANCHE BY NASA 2021113-021 (360) TIME 6:25

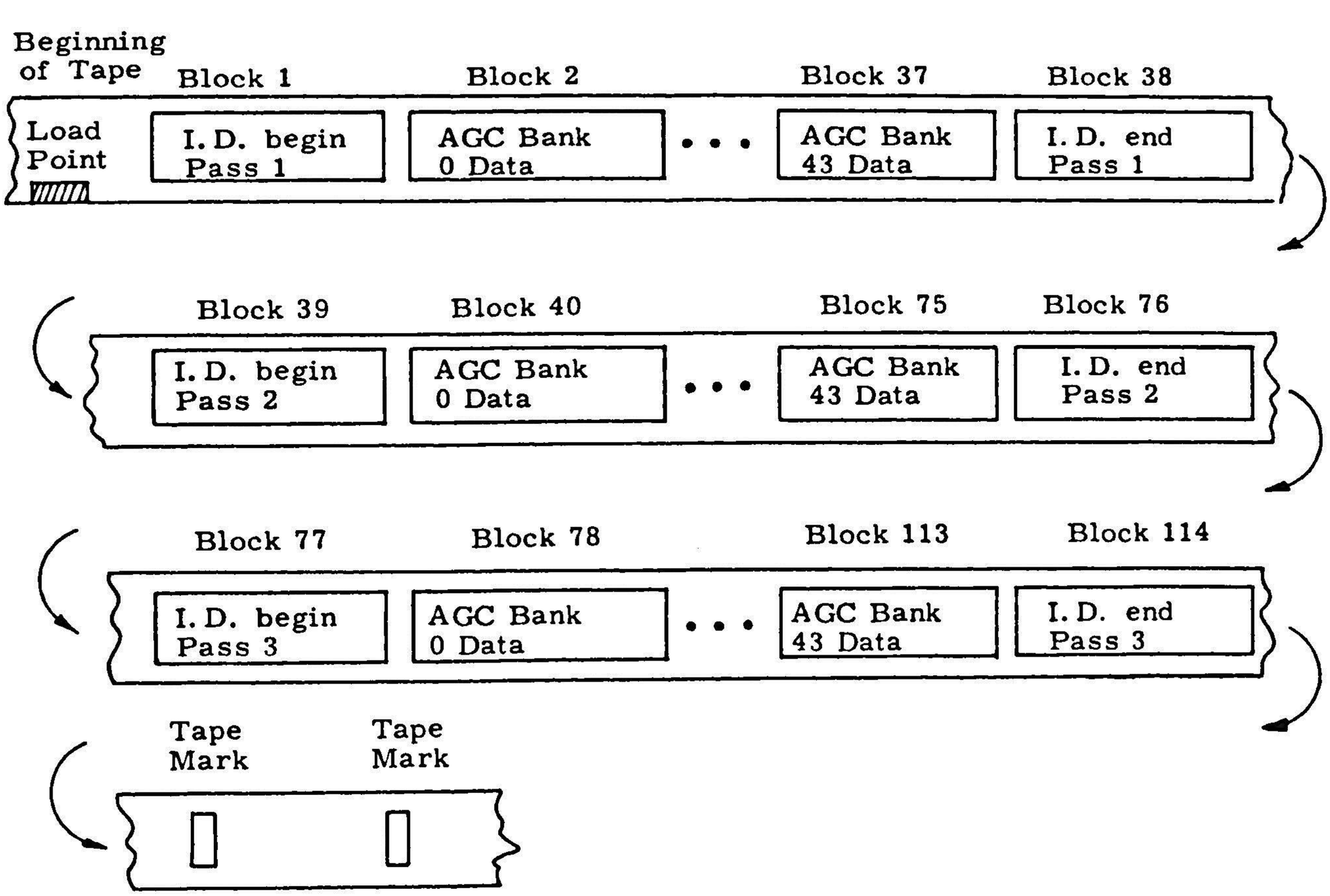
SUMMARY OF COMPARISON STATISTICS

144 PARAGRAPHS WERE COMPARED. 6 PARAGRAPHS DIFFERED. 61 WORDS DIFFERED. ROPE MODULE NUMBER(S) 2 DIFFERED.
0 PARAGRAPHS WERE FOUND WHICH DID NOT POSSESS COUNTERPARTS WITH MATCHING PARAGRAPH NUMBERS IN THE OPPOSITE DATA SET.

APPENDIX H

PORTAFAM Magnetic Tape Formats

Characteristics: Seven-track, half-inch magnetic tape recorded in standard IBM NRZI format at a density of 556 (odd parity) frames per inch. Each block of data is 4100 frames (equals 3075 bytes) long and contains a single AGC bank. Each of the identification and pass number blocks is 48 frames (equals 36 bytes) in length. The tape organization of an AGC Mission Program is shown in the following figure:



Only one tape mark (plus erase gap) is written in the above if the AGC Mission Program is followed by a Special Program.

Special programs have the same tape organization as Mission programs, except that the identification and pass number blocks contain a different Program Type Code; see the Status Block Codes below.

Block 1 has the following format:

	C	B	A	8	4	2	1	Seven Track Tape Channel
48 frames total	P	0	0	A_0	A_1	A_2	A_3	13 (EBCDIC) character AGC program identification
	P	0	0	A_4	A_5	A_6	A_7	
				⋮				
	P	0	0	M_0	M_1	M_2	M_3	
	P	0	0	M_4	M_5	M_6	M_7	
	P	0	0	0	1	0	0	EBCDIC blank character
	P	0	0	0	0	0	0	
Status Subblock	P	1	0	0	0	0	1	Begin Pass 1 of Mission Program (Two-Bit Pass number)
	P	0	0	0	1	0	0	
	P	0	0	0	0	0	0	
	P	0	0	0	0	0	0	
				⋮				
	P	1	0	0	0	0	1	The preceding four frames repeated 4 more times.
	P	0	0	0	1	0	0	
	P	0	0	0	0	0	0	
	P	0	0	0	0	0	0	

P represents the odd frame parity bit that is generated by the IBM tape control hardware. $A_0, \ldots, A_7$ through $M_0, \ldots, M_7$ represent the bits of 13 (eight-bit) EBCDIC characters that specify the AGC program name and revision number in the form n.Rrrr; n is the one to eight character program name and rrr is the three character revision number. For example, Revision 2 of AGC program LUM69 would be written as LUM69.R002bbb where bbb represents the three blanks that are necessary to pad out the identification to 13 characters. This particular program identification would appear on tape as follows:

C	B	A	8	4	2	1	Seven Track Tape Channel
P	0	0	1	1	0	1	L
P	0	0	0	0	1	1	
P	0	0	1	1	1	0	U
P	0	0	0	1	0	0	
P	0	0	1	1	0	1	M
P	0	0	0	1	0	0	
P	0	0	1	1	1	1	6
P	0	0	0	1	1	0	
P	0	0	1	1	1	1	9
P	0	0	1	0	0	1	
P	0	0	0	1	0	0	•
P	0	0	1	0	1	1	
P	0	0	1	1	0	1	R
P	0	0	1	0	0	1	
P	0	0	1	1	1	1	0
P	0	0	0	0	0	0	
P	0	0	1	1	1	1	0
P	0	0	0	0	0	0	
P	0	0	1	1	1	1	2
P	0	0	0	0	1	0	
P	0	0	0	1	0	0	blank
P	0	0	0	0	0	0	
P	0	0	0	1	0	0	blank
P	0	0	0	0	0	0	
P	0	0	0	1	0	0	blank
P	0	0	0	0	0	0	
•							
•							
•							

Blocks 2 through 37, 40 through 75, and 78 through 113 (AGC Bank Data Blocks) have the following format:

	C	B	A	8	4	2	1	Seven Track Tape Channel
4-frame Bank Number at beginning of block ranging from 0 through 43 (octal)	P	1	0	B_0	B_1	B_2	B_3	Bank number (binary) B_5 is the least significant bit
	P	1	1	B_4	B_5	0	0	
	P	1	1	0	0	0	0	
	P	1	1	0	0	0	0	
1024 AGC words in sequential address order (equals 4096 frames)	P	0	1	W	X_0	X_1	X_2	First <u>used</u> AGC word. X_{14} is the least significant bit
	P	0	0	X_3	X_4	X_5	X_6	
	P	0	0	X_7	X_8	X_9	X_{10}	
	P	0	0	X_{11}	X_{12}	X_{13}	X_{14}	
	⋮			⋮				
	P	0	1	W	X_0	X_1	X_2	last <u>used</u> AGC word
	P	0	0	X_3	X_4	X_5	X_6	
	P	0	0	X_7	X_8	X_9	X_{10}	
	P	0	0	X_{11}	X_{12}	X_{13}	X_{14}	

In the preceding, $B_0B_1B_2B_3B_4B_5$ is the six-bit binary Bank Number, W is the AGC word (odd) parity bit, and $x_0x_1x_2.....X_{13}x_{14}$ represents a 15-bit used AGC word. An unused AGC word has the following format:

C	B	A	8	4	2	1	Seven Track Tape Channel
P	0	1	0	0	0	0	<u>Unused</u> AGC word (4 frames)
P	1	1	0	0	0	0	
P	1	1	0	0	0	0	
P	1	1	0	0	0	0	

Block 38 has the following form:

	C	B	A	8	4	2	1	Seven Track Tape Channel
48 frames total	P	0	0	A_0	A_1	A_2	A_3	13 (EBCDIC) character AGC program identification
	P	0	0	A_4	A_5	A_6	A_7	
				•				
				•				
				•				
	P	0	0	M_0	M_1	M_2	M_3	
	P	0	0	M_4	M_5	M_6	M_7	
	P	0	0	0	1	0	0	EBCDIC blank character
	P	0	0	0	0	0	0	
Status Subblock	P	1	0	1	0	0	1	End pass 1 of Mission Program (two-bit pass number)
	P	0	0	0	1	0	0	
	P	0	0	0	0	0	0	
	P	0	0	0	0	0	0	
				•				
				•				
				•				
	P	1	0	1	0	0	1	The preceding four frames repeated four more times
	P	0	0	0	1	0	0	
	P	0	0	0	0	0	0	
	P	0	0	0	0	0	0	

The notation is the same as that in Block 1.

Blocks 38 and 77 differ from Block 1 only in that their two-bit pass numbers are 10 and 11 respectively. Similarly Blocks 76 and 114 differ from Block 38 only in their respective pass numbers.

The general form of a status subblock is as follows:

	C	B	A	8	4	2	1	Seven Track Tape Channel
Status Subblock (20 frames total)	P	1	0	E	T_0	T_1	T_2	General format for Begin or End Pass number blocks
	P	0	0	P_0	P_1	0	0	
	P	0	0	0	0	0	0	
	P	0	0	0	0	0	0	
				•				
				•				
				•				
	P	1	0	E	T_0	T_1	T_2	The preceding four frames repeated four more times.
	P	0	0	P_0	P_1	0	0	
	P	0	0	0	0	0	0	
	P	0	0	0	0	0	0	

The symbols in the preceding diagram are defined as follows:

E = 0, Begin pass P_0 P_1

E = 1, End pass P_0 P_1

T_0	T_1	T_2	Three-bit Program Type Code
0	0	0	Trace Dump
0	0	1	Mission Program
0	1	0	Special Program Number 1

No others are used presently.

P_0	P_1	Two-bit Pass Number
0	0	Not used
0	1	Pass 1
1	0	Pass 2
1	1	Pass 3

IMPORTANT NOTE: PORTAFAM tapes that are to be used in the field must be written on type 3M777 magnetic tape.

ABOUT THE MAGNETIC TAPE UNIT

The need for a portable and highly reliable digital magnetic tape recorder/reproducer is an out-growth of an effort to simplify and economize field simulation tests and diagnostics of fixed memory mission programs for the Apollo Guidance Computer (AGC). The tape recorder is to furnish bulk storage in a compact system called PORTAFAM (Portable Fixed Memory). Naturally, the recorder is a key link in the PORTAFAM system. The relationship of the PORTAFAM system to the AGC is further explained.

The larger portion of the AGC memory system is fixed (read-only) memory. Each AGC fixed memory is a specialized and custom made assembly which, by intent, cannot have its stored programs altered once the memory has become a part of the AGC system. When Apollo mission requirements call for a change which affects the fixed memory portion of the AGC, in addition to a new mission program requirement, a new fixed memory must be fabricated. Eventually, the memory must undergo extensive field simulation testing. Should these tests reveal a non-trivial fixed memory deficiency, a great deal of time, money and effort are expended in detecting and correcting the source of the problem.

The PORTAFAM system is simply a design and diagnostic aid which will simplify and economize:

1. The fixed memory stored program field simulation tests.
2. The diagnostics in case an error is detected.

PORTAFAM will accomplish these objectives, respectively, as follows:

1. The AGC fixed memory is simulated with an erasable memory which can be loaded with one or more stored programs from magnetic tape. This memory temporarily replaces the AGC read-only memory until the stored programs are verified by field simulation tests.
2. A portion of the PORTAFAM erasable memory, called Trace Memory, in excess of AGC (fixed memory) requirements is reserved for use as a recirculating push-down list for the contents of recently accessed

AGC (fixed memory) locations. In the case of an error, the Trace Monitor, i.e., the Trace Memory and its control circuits, stores the current contents of the Trace Memory list onto magnetic tape for later processing.

The requirement for portability and compactness arises due to the limited available space inside the Apollo space capsule where the AGC simulation tests and diagnostics will be conducted. The PORTAFAM system is thus considered ground support equipment, and while it will not be subject to the rigid environmental specifications of a flyable system, it will encounter field system abuses.

IBM format and reel upward compatibility* of the magnetic tape recorder is required for convenience and economy reasons. Source data at the computer facility which assembles new mission programs can be readily written on magnetic tape with IBM compatible tape transports using a loaded supply reel from the PORTAFAM system. The recorded tape and reel are then sent to the field test site. No intermediate processes are required.

The choice of the seven track, 0.5 inch, NRZI version IBM format is based on the current availability of compatible tape transports at the various computer sites likely to be used for handling PORTAFAM tapes. Of the three different recording densities available for seven track format (200, 556 or 800), 556 bits per inch recording density appears to be a reasonable choice in terms of tape utilization and data reliability for this application.

A particular type of magnetic tape is specified (3M777) in order to standardize on a tape/oxide thickness, thereby obtaining a consistent basis for tape length for a given reel size and permitting the recording head saturation current to be adjusted to an optimum in the write amplifiers.

The generation of the Longitudinal Redundancy Check (LRC) and Vertical Redundancy Check (VRC) bits as well as the reaction control logic to the LOAD POINT and END-OF-TAPE sensing is performed external to the magnetic tape unit. This is intended to provide an added measure of reliability of data transfer to, and flexibility in operation of, the magnetic tape unit.

*Upward compatibility in this case implies that the tape format and reels used on the portable magnetic tape recorder are fully compatible with an IBM system tape transport, but that the normal start/stop distance and reel diameter of the IBM transport are not fully compatible with the portable recorder.

APPENDIX I

Speciman pages of a PORTAFAM Tape Dump - from Revision 67 of AGC
Program COMANCHE by NASA 2021113-061

PORTAFAM TAPE DUMP

```
19000006 18020F0E 15020F0F 10040001 180A0100 1D0A0F00 18000003 18060F04 13020F0C 15090003 120A0F0F 1C020F0F
1E020F08 18000006 19070303 18070001 19020F0A 18060F0C 120A0F0A 18000003 160A0103 15080003 16060A0D 18000006
19060F07 11000003 10060F02 180A0305 15080003 180A0207 1D0A0F0A 1E0A0108 1D000000 14000000 19020F0A 10070401
13060A08 1D080100 15080102 15080101 1D080103 16000100 1E000102 1E000101 16000103 1E060A0C 18060C05 15000100
1E000102 1E000101 15000103 1E0A0108 18060C05 1A0A0F07 10060C08 130A010C 150A0F0E 180A010D 1D0A0F0C 180A0108
1D0A0F0F 180A010D 190A0F09 150A0F08 180A010B 150A0F0D 13020F0C 10080001 17080F04 1E020F08 10090009 10070709
1E0A0101 18070806 14000000 1D0A0F0F 18000006 11070703 130A0100 10070704 130A030E 150A0F0B 180A010D 190A0F09
180A0108 150A0F0D 19020F08 18000000 19070709 18070806 190A0F0A 16020F09 1D0A0F09 180A010D 16020F09 1D0A0F09
1C020F0A 1E020F0B 10000002 1A040000 18020F0B 1F0A0308 1E080F04 18000006 11070C02 11020F0D 10070900 10070900
18070C02 19000001 13070908 18070908 18070908 11020F0D 10070909 130F0C0E 10070909 180A010B 150A0F0D 11020F0E
19000307 100A0800 1007090F 18060E04 1A0A0F0B 19020F0F 13070605 10070605 18070709 18070709 1C020F0C 16070D0D
18000006 19060008 13020F0C 1E0A0101 190A0F0C 18070800 130A000D 1A0E0F0C 18070B0C 17080F04 18000006 11070B0A
1E060A0A 18000006 19070807 18070B0C 13060A08 1A0E0F0C 18070B0C 1E060A09 1A0E0F0C 19020F0F 1007050F 180A0108
18070600 130A0105 1007050F 13020F0C 17080F04 18080E05 10080001 13020F0C 17080F0B 18000006 19070C0F 15080101
13000001 1F0A010F 1E000101 10080001 18020F0E 18000006 11070D03 18050F0A 19020F09 18070D06 10070D07 1E0A3108
1D0A0F09 1C000001 15020F09 150F0C0E 18060C05 10070A05 160C0400 180A010D 11070E05 180A0108 11070E05 130A010A
11070E05 130C080C 10080D09 18040308 13020009 18000006 19070E0B 10070F03 13020009 10080C01 13000D09 10080C02
180A0108 15080C03 19090C0D 150E0F0C 14000401 170A0104 11000000 19070F0C 1C0A010A 1F020009 11000300 19070F0C
10000002 10090C0D 190F0E0D 18070F0E 1608000F*
0C030D06 0D040C01 0D050C03 0C080C05 04080309 0F000F06 0F070400 29040000 29040000 29040000 29040000 29040000*
0C030D05 0D040C01 0D050C03 0C080C05 04080309 0F000F06 0F070400 21080000 21080000 21080000 21080000 21080000*
20303030 11060807 11070507 1907040E 19070807 11070809 1906070C 1904040E 113F090F 1106090A 19060103 1106070E
11060A05 190C0D03 1906080C 19060A07 130C0809 1F000100 15080101 19000100 19040401 18000104 1D000101 130A000E
1D080500 19000100 13040208 110C0107 13000500 18000006 1F00060E 1D08060E 13000500 18000006 1700060C 1004060D
13000500 18000006 1F000001 1A00060E 110C0108 1300060E 18000006 1F000500 150C060D 18000006 1F000500 150C060D
16000001 1F000000 1D08060F 11040304 1A0C060D 180A010D 1D08060E 1D0C060C 18000006 1F00050D 1A000600 10000002
13000500 18000006 1F00060D 1D08060D 150C0001 19040300 18000101 1D08050D 18000006 1300060E 1A00060E 1E00060C
1E00060C 1508060C 19040408 10080501 19000500 19040402 19000100 180E0405 110C0108 110C0108 130A010F 1F000100
1D080500 19000100 13040605 1800070E 15000500 130A000E 1D08050D 18040308 1D040600 1D040700 1D04060D 18040308
1D04060D 15040702 13040600 18040308 110E0F07 1D08050D 18000006 1300060D 1A00060D 18000006 11040608 10000F06
18000006 13000700 14000700 18000006 19040701 10000F03 18000006 18000702 12000702 18000006 19040707 10000F00
19000500 11040604 110C0108 1D08050D 180D080A 1904070F 1104080A 100E0906 18000600 19040807 12080500 18000006
1300060E 1A00060E 1E00060C 120C060C 1E000000 15080000 19040801 1C000500 110D070A 17000400 11000000 11040904
180A0102 17000400 11000000 180E0405 110C0108 1D08050D 180A0104 18000006 17000400 1F0C080C 1D000000 11040908
11040D0A 19040E02 19040D0E 19000703 1104080F 1104080F 13000500 16070D08 18000006 1E040507 160F0C0E 1D080500
180A010D 1D080001 190C060C 150C060D 18040300 1A00060D 1D0C060F 150C0700 18040B0A 1A000700 190C0701 190C0702
18040B0A 12000702 19000500 1D08050D 11040A02 18090804 110C0108 1E000000 1D08060E 180A010D 150C0001 19000002
13000500 16070D08 18000006 16040D02 160F0C0E 1D08050D 180A0100 1D0C060C 150C0600 1D08060E 19000500 19080500
10040C00 1A050A08 130A0102 1700040E 11000300 180E0405 110C0108 15000500 130A000E 1D08050D 180A0102 1700040E
11000000 11040109 19040108 1C00050D 16040506 1D08050D 1104090E 10040506 16000500 14000000 1D08050D 19000703
11040E06 11040E06 19040605 1400040E 18000006 1F0A0106 1D080100 11040403 140A010B 1D08050E 1508050F 15080600
19000508 1105040E 19040F03 11050509 1D08060F 100E0906 11000600 1105000C 19040F09 19050008 1D0C0509 150C0508
150C0600 1D0C060C 19000508 11050102 19050001 1905000E 14000600 18000006 1E050005 1208050E 130A0008 1508060C
10050908 180A0108 19080501 180C0108 1208050E 14000509 19050002 18000005 14000509 15040509 1208050E 11000600
1905010F 19050106 11050108 19000600 1905010F 110C0108 11050103 110C0108 18000006 1C00060D 1D04060D 1208050E
1400060C 160F0C0E 15000508 11000000 19050405 16000001 19050206 130A000E 1E000000 16000600 10080600 180A0100
160A0008 120C060C 130A000E 1F000000 1E000509 15080509 180A010D 150A0008 1A0C0508 14000600 16000508 11000000
19050405 10000508 11050005 15080600 1C000600 1E000509 19000006 1F050005 19050405 18000006 1A08050F 19000006
18000509 12000509 13000508 1F000000 15080000 19050400 10040600 1D00050F 10050705 1D08060E 110C0108 11000000
11050102 14000509 18000006 1E050102 130A000E 1E000000 120C0500 180A010D 10080508 19040F03 11000000 1905000E
18000509 18000006 15050008 1C0A000E 19050504 1A040101 18000006 1F0A000E 150C0001 1E000101 150C0001 19050709
1A000001 1A000001 1A000001 1A000001 1A000001 1A000001 1A000001 1A000001 1A000001 1A000001 1A000001 1A000001
1A000001 1D040600 11000600 10000600 19050402 1D040600 18000006 19000508 1D040600 14000600 18000006 17000509
16000600 15080000 19050806 18000006 16000508 1A08060C 11050808 18000006 1E050900 18000006 16000508 18000006
11050800 18000006 15050904 1A08060C 11050905 18000006 19050908 18000005 120C060C 16000508 12040007 18000006
19000508 10080600 19000500 10000002 10000002 10000002 18000006 1C00060D 1D04060D 180A0100 10000002 14000600
16000508 18000006 19050A07 11050708 130A0008 1508060C 14000509 16000600 19050904 140A010B 1508050F 1D080507
```

```
10060008 15040509 180F0408 15040509 19000508 11050D01 11050B07 19050C0D 1D0C0509 150C0508 150C060D 1D0C060C
18000006 1105080F 11050005 150C0700 1D0C060F 18000006 11050C03 11050005 1D0C0702 1D0C0701 18000006 19050C08
11050005 19000508 11050001 11050005 19050C0D 11050005 18000006 14000509 15040509 12080507 18000006 18000509
1504050C 19050D0A 18000006 1A08050F 18000006 18000509 12000509 13000508 1E000000 15080000 19050005 18050E08
1D040700 1D04060D 1D040700 18050E08 15040702 1D04060D 15040702 18050E08 110E0F07 13000507 1D08050E 1100060C
19050F07 11050E0E 11050F03 19000600 19050F07 10000002 11050F03 10000002 18000006 1C00060D 1D04060D 1208050E
140A0108 15080600 1400060C 16000508 11000000 19060C04 11050F0F 11050005 15080600 1C00060D 1E00050C 18000006
1E050005 1D04060D 1D00050F 19050705 13050406 150C0002 1D04060D 180F0408 1D04060D 1D040700 180F0408 1D040700
15040702 180F040B 15040702 18000000 10060008 18CF0401 180A010D 150C0501 1D080601 18060C0F 13000601 150C0501
18000006 1106010E 11050005 18000006 1300060D 1D000500 1D040203 10060E03 1100060C 19060209 1D080001 1D000500
1D040205 11050005 1C080008 16000500 11000000 14000000 19060508 11060305 140A0201 1D08050D 1800060C 1D080001
180A010D 1906040A 19000500 1106030C 1C040104 1D08050D 18000006 1300060D 1906040A 14000000 1D08050D 14000000
1D000000 130A000E 1D080508 18000006 1F000600 150C0508 18000006 1700060C 150C0001 16000508 150C0001 1D000500
1D040205 140A010B 150806C0 15040503 1D040600 15040509 18060609 15040505 1D04060D 1D040700 18060609 1D040507
1D04060D 15040702 18060609 110E0F07 1D080500 180A010D 1D0C0503 150C0502 1D0C0505 150C0504 150C0507 1D0C0506
1C000500 1D000000 130A000E 18000006 1700060C 19060302 1100060C 11060708 1906060D 11060702 1900060D 11060708
10000002 11060702 10000002 140A010D 1D08050E 18C00006 1C00060D 1500050D 11050704 1D08050E 1D04060D 1500050D
11050704 18060C00 110C0108 19000703 11060906 11060906 18060C0F 12040703 18000006 1300060D 1D000500 1D040203
10060E03 19000500 11060808 110C0108 16070D08 18000006 16060901 12040007 1A04040E 19040C03 1500050D 130A000D
1D080500 180A010D 1904010F 18000804 110C0108 110CC108 110F090F 1C0A0109 1A0C0706 18000006 1D000000 18000003
1D040700 18000006 1500070C 13000001 15040702 110D0308 18060C0F 110E0C01 18000006 1300060D 15000706 1D040001
15000703 180C0808 1A0C0706 19000703 1906080A 110C0108 18000006 13000700 15000706 15030F0D 18000006 18000702
15000706 1D030F0F 110C0108 1300060E 190D0503 1D000500 1300020A 1D08040F 110D0901 1300060D 18000006 17000000
1D08060E 180A010D 150C060D 18000006 1700060C 1A000001 1A00060E 1D0C060C 18000006 17000000 1A00060D 10000002
18000006 1A04050F 18060C00 1D040700 1D04060D 15040509 1300060E 1508050A 18060C00 1D04060E 1200050A 1E00060C
16000508 1D080508 190600CF 1D080501 15040702 1D04C60D 18060C00 190E060C 180A010D 1D08050D 1100060C 11070004
19060E09 19060F08 150C060E 150C060D 1508060C 130A010F 1D08050D 1100060C 11070004 19060F02 19060F0E 150C060D
1508060C 130A010F 1A0C0500 1100060C 11070004 10000002 19060F0E 1107020A 11000000 11070002 1900060D 180A010D
1107020A 11070002 1107020A 18080D05 12020C02 1E050204 18000006 1607020D 1D04060D 1A040101 18000006 1F0A000E
1D04060D 1D0C0101 1A0C060D 13040C0C 18000006 1700060C 16050706 1D080508 1800060C 12040007 18000006 19000508
18000006 1F0A000E 1A0C0508 18000006 1F0A0005 1D04060D 18C00006 19000508 15080509 180A010D 150C0001 18000006
19000508 1D080001 18000509 1A00060D 19000006 1107020C 13CA000B 1508060C 1D08060D 10000002 1E0A000F 18000006
16070401 1D04060D 1A040101 18000006 1F0A000E 1D04060D 1D0C0101 1A0C060D 13060007 18000006 1700060C 16040808
19070102 18000006 1300060E 1A00060E 1E00060C 120C060C 12080500 18000006 1300060E 1A00060E 1E00060C 120C060C
1E000000 15080102 11000102 11000102 1907030C 1107000E 11070307 180D0B0A 11070502 19070505 19070505 18000006
1C00060D 1D04060D 180A000F 120C060C 1D040600 1A000001 15080000 1107050D 18000006 1C000001 1D04060D 1800060C
1E000000 1D080001 1907060A 1D000000 130A000D 1F000000 18000006 1E00060C 1508060C 1C00060D 1D08060D 18000006
13000600 15040500 1806CC00 180E0709 18000003 110902C1 180F0D05 15060A08 1E010F04 19040606 1A020B0E 1F0B0306
17030309 1000090E 13050F0F 13050308 100E0200 18000006 1300060E 1A00060E 1E00060C 120C060C 18000006 1300060E
1A00060E 1E00060C 120C060C 110C0108 18070908 1107080A 13070C0A 1D08050E 180D0B0A 11070904 11070006 18000006
1C00060D 1D04060D 18070D09 150C050E 1508050F 1C0A000E 1E00060C 11000300 11070D00 19070C06 11070A01 1900060D
180A010D 1107090E 11070A01 1D08060D 1508060C 1000050E 18000006 1C00060D 160A000E 1D04060D 1504050C 10060E03
19000500 11070C08 1D040600 1504050C 1D040600 180E0709 180C0006 11060A00 12070908 17090001 1C0F0F0D 10050200
1200090F 1F0A0307 1F01CF02 1D070105 1A0D0E0E 1708010F 1D0F0A00 10060206 100E0603 1F0D0A0C 14040A04 13050308
100E020D 1000050F 18000006 1C00060D 160A000E 1D04060D 18C0050F 18000306 1C00060D 1E0A000F 1D04060D 110C0108
1D000000 130A000E 1D08050D 1804030B 19070A09 18000006 1107090E 10080B04 18020C01 180A010D 1107090E 180A000F
1907090F 170F0F03 11070C01 10000004 1D080002 130C080C 1D080308 1D000000 18020601 18000006 11070E06 19000308
19070000 180F0E0D 13020404 13070F06 1D080301 13000004 16000308 1D080001 18070F07 190A0B09 10090D05 15000308
150A0601 100A0809 180A010D 19000006 100F0601 180A080D 100A0F00 10070E08 18070F01 18070F08 10070F09 1F0C0F0E
10303030 10303030 10303030 10303030 10303030*
20343030 18000303 10000008 18000303 18000104 18000303 10000203 18000303 18000407 10000A02 10000B00 18000D0E
10000E06 18000708 1F090608 1C080C09 11000000 10050307 13080306 11000300 1004030C 1D0C0306 11000000 10040E0E
170C0306 18050B04 170A0600 120C0303 18CA0000 18050A06 120C0303 11000000 10050908 1F000307 19080000 1805070E
16040305 11000000 10040F0F 150C0307 12000000 18050401 18080307 18000000 18000000 18000000 1A040000 1804040F
17000305 1A040000 18040A08 17000303 1A040000 19040106 17000303 170C0E07 170A0009 1A0F0C08 170F0F0F 170A0807
1A0F0C08 140F0F0F 18050500 15000307 170F0F0F 170A0807 1A0F0C08 10080B04 17080203 120F0C09 170F0F0F 170A0807
1A0F0C08 18090C00 1F0A0803 1A0F0C08 18000208 170A04C0 120F0C09 18000A00 1F0A0304 120F0C09 18010F04 1F0A0705
```

Notes:

1) The symbol * indicates the end of a tape block.

2) To read the dump:
   a) convert each even-numbered pair of hexadecimal characters to binary:
   b) discard the two most significant bits, leaving the six bits of the corresponding tape character.

3) The formats of all blocks are shown in Appendix H

4) Example:

The last word of the first specimen page of the dump (Page 055) is 1DO80507. On conversion to binary this becomes

HEXADECIMAL	BINARY
1D	00011101
08	00001000
05	00000101
07	00000111

Discarding the two most significant bits of each binary character leaves the four six-bit tape frames

C	B	A	8	4	2	1	Seven Track Tape Channel
P	0	1	1	1	0	1	
P	0	0	1	0	0	0	
P	0	0	0	1	0	1	
P	0	0	0	1	1	1	

According to Appendix H this represents the binary AGC word 10110000101011 1, which is the octal AGC word 54127 1.

APPENDIX J

Listings of Catalogued Procedures

Procedure GAPCKMDT

```
//CK         EXEC PGM=MACFETCH,PARM='MDTPCK',REGION=52K
//MACPAC     DD DSNAME=SYS1.USERLIB,DISP=SHR
//FT06F001   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1596)
//FT08F001   DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=80,DEN=1,TRTCH=ET),      C
//              LABEL=(1,BLP),UNIT=TAPE7,VOLUME=SER=MASTDK,DISP=OLD,    C
//              DSNAME=AGC.MASTER.DECK.TAPE
//FT08F002   DD DCB=(*.FT08F001),DISP=(,KEEP),VOLUME=REF=*.FT08F001,    C
//              LABEL=(2,BLP),DSNAME=COPY01
//FT08F003   DD DCB=(*.FT08F001),DISP=(,KEEP),VOLUME=REF=*.FT08F001,    C
//              LABEL=(3,BLP),DSNAME=COPY02
//FT08F004   DD DCB=(*.FT08F001),DISP=(,KEEP),VOLUME=REF=*.FT08F001,    C
//              LABEL=(4,BLP),DSNAME=COPY03
//FT09F001   DD UNIT=2314,SPACE=(80,(5000,10),RLSE,,ROUND)
```

Procedure GAPMAGC

```
//M        EXEC PGM=MACFETCH,PARM='AGCMFG',REGION=180K
//MACPAC     DD DSNAME=SYS1.USERLIB,DISP=SHR
//SYSOUT     DD SYSOUT=A
//SYSPRINT   DD SYSOUT=A
//AGCB02     DD DISP=SHR,UNIT=2314,VOLUME=SER=AGCB02
//AGCB02B    DD DISP=SHR,UNIT=2314,VOLUME=SER=AGCB02
//MASTDKSN   DD DSNAME=GAP.AGCMANU.MDSEQNO,DISP=(OLD,KEEP)
//MANUHIST   DD DSNAME=GAP.AGCMANU.HISTORY,DISP=(MOD,KEEP)
//PUNCHDCB   DD DCB=(LRECL=80,BLKSIZE=80,DEN=1,TRTCH=ET),DISP=(,KEEP),   C
//              UNIT=(TAPE7,,DEFER),VOLUME=SER=MASTDK,LABEL=(,BLP),      C
//              DSNAME=AGC.MASTER.DECK.TAPE
//PFAMTAPE   DD DCB=(BLKSIZE=3075,DEN=1,TRTCH=C),VOLUME=SER=PORTFM,      C
//              DISP=(,KEEP),LABEL=(,BLP),UNIT=AFF=PUNCHDCB,             C
//              DSNAME=AGC.PORTAFAM.TAPE
//AGCPUNCH   DD DISP=(,PASS),UNIT=2314,SPACE=(TRK,(26,5))
//P        EXEC PGM=MACFETCH,PARM='AGCPCH',COND=(0,EQ,M),REGION=52K
//MACPAC     DD DSNAME=SYS1.USERLIB,DISP=SHR
//AGCPUNCH   DD DSNAME=*.M.AGCPUNCH,DISP=(OLD,DELETE)
//MANUHIST   DD DSNAME=GAP.AGCMANU.HISTORY,DISP=(OLD,KEEP)
//FRCMAGTP   DD DCB=(LRECL=77,BLKSIZE=77,DEN=2,TRTCH=ET),DISP=(,KEEP),   C
//              VOLUME=(PRIVATE,RETAIN,SER=36KCRS),LABEL=(1,BLP),        C
//              UNIT=(TAPE7,,DEFER),DSNAME=AGC.CRS.FILE1
//BRCMAGTP   DD DCB=(LRECL=126,BLKSIZE=3150,DEN=2,TRTCH=C),DISP=(,KEEP),C
//              LABEL=(2,BLP),VOLUME=REF=*.FRCMAGTP,DSNAME=AGC.CRS.FILE2
//PAPTPPCH   DD UNIT=1012
//DIGISTOR   DD UNIT=031
```

Procedure GAPMAGCA

```
//M          EXEC PGM=MACFETCH,PARM='AGCMFG',REGION=180K
//MACPAC       DD DSNAME=SYS1.USERLIB,DISP=SHR
//SYSOUT       DD SYSOUT=A
//SYSPRINT     DD SYSOUT=A
//AGCB02       DD DISP=SHR,UNIT=2314,VOLUME=SER=AGCB02
//AGCB02P      DD DISP=SHR,UNIT=2314,VOLUME=SER=AGCB02
//MASTDKSN     DD DSNAME=GAP.AGCMANU.MDSEQNO,DISP=(OLD,KEEP)
//MANUHIST     DD DSNAME=GAP.AGCMANU.HISTORY,DISP=(MOD,KEEP)
//PUNCHDCB     DD DCB=(LRECL=80,BLKSIZE=1680,DEN=2,TRTCH=C),DISP=(,KEEP),  C
//                VOLUME=(PRIVATE,RETAIN,SER=36KCRS),LABEL=(3,BLP),        C
//                UNIT=(TAPE7,,DEFER),DSNAME=AGC.CRS.FILE3
//PFAMTAPE     DD DCB=(BLKSIZE=3075,DEN=1,TRTCH=C),VOLUME=SER=PORTFM,       C
//                DISP=(,KEEP),LABEL=(,BLP),UNIT=AFF=PUNCHDCB,             C
//                DSNAME=AGC.PORTAFAM.TAPE
//AGCPUNCH     DD DISP=(,PASS),UNIT=2314,SPACE=(TRK,(26,5))
//P          EXEC PGM=MACFETCH,PARM='AGCPCH',COND=(0,EQ,M),REGION=52K
//MACPAC       DD DSNAME=SYS1.USERLIB,DISP=SHR
//AGCPUNCH     DD DSNAME=*.M.AGCPUNCH,DISP=(OLD,DELETE)
//MANUHIST     DD DSNAME=GAP.AGCMANU.HISTORY,DISP=(OLD,KEEP)
//BRCMAGTP     DD DCB=(LRECL=77,BLKSIZE=77,DEN=2,TRTCH=ET),DISP=(,KEEP),    C
//                VOLUME=(PRIVATE,RETAIN,SER=36KCRS),LABEL=(1,BLP),        C
//                UNIT=(TAPE7,,DEFER),DSNAME=AGC.CRS.FILE1
//BRCMAGTP     DD DCB=(LRECL=126,BLKSIZE=3150,DEN=2,TRTCH=C),DISP=(,KEEP),C
//                VOLUME=(PRIVATE,RETAIN,SER=36KCRS),LABEL=(2,BLP),        C
//                UNIT=(TAPE7,,DEFER),DSNAME=AGC.CRS.FILE2
//PAPTPPCH     DD UNIT=1012
//DIGISTOR     DD UNIT=031
```

Procedure GAPPMHST

```
//HS        EXEC PGM=IEBPTPCH,REGION=52K
//SYSPRINT   DD SYSOUT=A
//SYSUT1     DD DSNAME=GAP.AGCMANU.HISTORY,DISP=(OLD,KEEP)
//SYSUT2     DD SYSOUT=A
//SYSIN      DD DSNAME=SYS1.PROCLIB(GAPPMHSU),DISP=SHR
//SN        EXEC PGM=IEBPTPCH,REGION=52K
//SYSPRINT   DD SYSOUT=A
//SYSUT1     DD DSNAME=GAP.AGCMANU.MDSEQNO,DISP=(OLD,KEEP)
//SYSUT2     DD SYSOUT=A
//SYSIN      DD DSNAME=SYS1.PROCLIB(GAPPMHSV),DISP=SHR
```

(GAPPMHSU)

```
PRINT MAXLINE=52,MAXFLDS=1
TITLE ITEM=('AGC MANUFACTURING HISTORY',48)
TITLE ITEM=('    ',48)
RECORD FIELD=(120,1,,1)
```

(GAPPMHSV)

```
PRINT MAXLINE=52,MAXFLDS=1
TITLE ITEM=('AGC  MASTER DECK SEQUENCE NUMBER',45)
TITLE ITEM=('    ',45)
RECORD FIELD=(120,1,,1)
```

Procedure GAPPPFTP

```
//PF        EXEC  PGM=IEBPTPCH,REGION=52K
//SYSPRINT    DD  SYSOUT=A
//SYSUT1      DD  DCB=(RECFM=U,BLKSIZE=4100,DEN=1),VOLUME=SER=PORTFM,      C
//                DISP=(OLD,KEEP),UNIT=(TAPE7,,DEFER),LABEL=(,BLP),        C
//                DSNAME=AGC.PORTAFAM.TAPE
//SYSUT2      DD  SYSOUT=A
//SYSIN       DD  DSNAME=SYS1.PROCLIB(GAPPPFTQ),DISP=SHR
```

(GAPPPFTQ)

```
 PRINT TOTCONV=XE
 TITLE ITEM=('PORTAFAM TAPE DUMP',49)
 TITLE ITEM=('    ',49)
```

Procedure GAPPMDTP

```
//MD        EXEC PGM=IEBPTPCH,REGION=52K
//SYSPRINT   DD SYSOUT=A
//SYSUT1     DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=80,DEN=1,TRTCH=ET),      C
//             DISP=(OLD,KEEP),UNIT=TAPE7,VOLUME=SER=MASTDK,            C
//             LABEL=(,BLP),DSNAME=AGC.MASTER.DECK.TAPE
//SYSUT2     DD SYSOUT=A
//SYSIN      DD DSNAME=SYS1.PROCLIB(GAPPMDTQ),DISP=SHR
```

(GAPPMDTQ)

```
  PRINT MAXLINE=37,MAXFLDS=1
  TITLE ITEM=('LISTING OF AGC MASTER DECK TAPE',45)
  TITLE ITEM=('    ',45)
  RECORD FIELD=(80,1,,20)
```

Notes:

1) In a typical sord pair, such as 401610 777611, the first word (401610) comes from Revision 2 of MANCHE45 and the second word (777611) from Revision 45 of COMANCHE.

2) The last (sixth) digit of each word is the parity digit.

3) To clarify the process of reading addresses, observe the following:

   a) the (unequal) AGC word pair 231331 000331 is located in Bank 11, Bank Address 3132, Relative Paragraph Address 132;

   b) the word pair 536700 430701 is in Bank 11, Bank Address 3136, Relative Paragraph Address 136.

4) See Appendix B for a discussion of the Comparison Statistics, particularly in reference to the Rope Module Number(s) field.

APPENDIX K

Documents Pertinent to GAP Manufacturing

1) Operational Procedure for Generation of Master Deck Tape for Mission Programs

   Apollo G&N Specification ND 1002377
   Date - 6/68
   Author: M.G. Murley

2) Release of Master Deck Tapes for Manufacture of Core Ropes

   Apollo Project Memorandum
   Author: M.G. Murley

3) Users' Guide to the AGC Monitor (Core Rope Simulator)

   Apollo Guidance, Navigation and Control, MIT Instrumentation Laboratory, Publication E-2026
   Date - 9/66
   Author: James D. Wood

4) Users' Guide for the MIT/IL 360/Digistor I/O Routine

   MIT Instrumentation Laboratory, DCG Memo #44
   Date - 9/68
   Author: E. Sabine

5) Portafam System Operational Description

   Apollo Guidance and Navigation, MIT Instrumentation Laboratory, Publication E-2402
   Date - 7/69
   Author: J. Leavitt