



Author(s)	Hogg, Robert Lee.; Glover, Dennis C.
Title	Control system programming remote computing and data display
Publisher	Monterey, California: U.S. Naval Postgraduate School
Issue Date	1963
URL	http://hdl.handle.net/10945/11864

This document was downloaded on May 14, 2015 at 07:35:58



<http://www.nps.edu/library>

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

**Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943**



<http://www.nps.edu/>

NPS ARCHIVE
1963
HOGG, R.

CONTROL SYSTEM PROGRAMMING
REMOTE COMPUTING AND DATA DISPLAY

ROBERT LEE HOGG
and
DENNIS C. GLOVER

LOAN DOC

24

CONTROL SYSTEM PROGRAMMING
REMOTE COMPUTING
and
DATA DISPLAY

by
Robert Lee Hogg
Lieutenant, United States Navy
and
Dennis C. Glover
Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of
MASTER OF SCIENCE
IN
ENGINEERING ELECTRONICS
United States Naval Postgraduate School
Monterey, California
1963

CONTROL SYSTEM PROGRAMMING
REMOTE COMPUTING
and
DATA DISPLAY

Robert Lee Hogg
and
Dennis C. Glover

CONTROL SYSTEM PROGRAMMING
REMOTE COMPUTING

and

DATA DISPLAY

by

Robert Lee Hogg

and

Dennis C. Glover

This work is accepted as fulfilling the
thesis requirement for the degree of

MASTER OF SCIENCE

IN

ENGINEERING ELECTRONICS

from the

United States Naval Postgraduate School

ABSTRACT

The increasing demands upon large digital computing facilities have necessitated the development and use of a system control program to provide for continuous automatic job processing. A comparative study of two current systems was conducted. The first, CDC FORTRAN 60, is described and its advantages and disadvantages are pointed out. This system was modified by the authors to incorporate a remote (satellite) operation in a time shared mode. The second system studied was the CDC COOP MONITOR system. This system, a more complex, sophisticated control system is likewise discussed. Modifications to correct certain disadvantages of this system are shown, along with the programming necessary to provide for remote station operation in the COOP MONITOR environment.

Several other programs which were developed by the authors to improve the computing center's capability are documented. These include; (1) A graph plotting routine utilizing CDC 1604/160 computers and a CDC 165 / CalComp Plotter, (2) A large file merge sort routine, and (3) A data display "Line Printer" routine.

The authors wish to express their appreciation to Professor Mitchell L. Cotton of the U.S. Naval Postgraduate School, and to the personnel of the school's computing center for their kind consideration during the term of this thesis.

TABLE OF CONTENTS

Section	Title	Page
1.0	Introduction	1
2.0	Fortran 60 System	4
2.1	Program Control	4
2.2	Monitor	6
2.3	Resident Routines	7
2.4	System's Characteristics	13
	2.4.1 Advantages	14
	2.4.2 Disadvantages	16
3.0	Coop Monitor System	20
3.1	Master Control System	20
3.2	Secondary Control Systems	25
3.3	System Operation	27
3.4	System's Characteristics	30
	3.4.1 Advantages	30
	3.4.2 Disadvantages	32
3.5	Alterations To Coop Monitor System	33
3.6	Specifications for Further Modifications	42
4.0	System Routines	50
4.1	Grafplot	50
4.2	Display "Line Printer Simulator"	57
4.3	Merg Sort	64
4.4	Satellite System Programming	69
	Bibliography	77

APPENDICES

- I. Coop Monitor Library Edit Routine (Libedit)
- II. 1604 Graph Plotting Routine (Grafplot)
- III. 1604 Merg-Sort Routine (Sort)
- IV. Fortran 60 Resident (FortSat)

LIST OF ILLUSTRATIONS

Figure	Page
1. BLOCK DIAGRAM FORTRAN 60 CONTROL SYSTEM	5
2. FORTRAN 60 "LOW CORE" RESIDENT ENTRIES	9
3. BLOCK DIAGRAM COOP MONITOR MCS & EXTREC	23
4. COOP MONITOR LIBRARY TAPE FORMAT	28
5. FLOW DIAGRAM COOP MONITOR SYSTEM MCS	29
6. PROGRAMMING FOR COOP MONITOR "BOOT"	36
7. FLOW DIAGRAM CDC 1604 GRAFLOT ROUTINE	53
8. OUTPUT FORMAT FOR DD65 DISPLAY	61
9. DATA DISPLAY MODEL 65 EXTERNAL FUNCTION CODES	62
10. FLOW DIAGRAM FOR CDC 1604 MERGSORT ROUTINE	65
11. FLOW DIAGRAM FOR CDC 1604 SATELLITE INTERRUPT	71
12. CDC 160-1604 SATELLITE CONTROL WORD FORMAT	72
13. LOGIC DIAGRAM FOR CDC 1607 SATELLITE STATUS LIGHTS AND PROGRAM CONTROL SELECTION	75

ENCLOSURES

1. OPERATING INSTRUCTIONS CDC 160 GRAFLOT ROUTINE
2. OPERATING INSTRUCTIONS CDC 160A GRAFLOT ROUTINE
3. OPERATING INSTRUCTIONS CDC 1604 GRAFLOT ROUTINE

ABBREVIATIONS

MCS	Master Control System
SCS	Secondary Control System
RHT	Running Hardware Table
SIO	Standard Input Output
AET	Available Equipment Table
AEDNT	Available Equipment Driver Name Table
RBL	Relocatable Binary Loader
ELT	Edit Library Tape
PLT	Prepare Library Tape
FWA	First Word Address
LWA+1	Last Word Address Plus One
I/O	Input/Output
BCD	Binary Coded Decimal
CDC	Control Data Corporation

1.0 INTRODUCTION

At present, the Naval Postgraduate School's computing center utilizes approximately 60% of its prime time processing short, 1 to 15 minute, "job shop", input programs. Practically all of these are of a scientific problem solving nature. Three different compiler/assembler control systems for processing these programs are currently in significant usage. They are the Control Data Corporation's "Fortran 60 System", the Naval Electronics Laboratory's "Neliac Compiler", and the school's own algebraic assembly routine, "Scrap". Of the above, the Fortran 60 System is the only system which incorporates a full control, automatic, job sequencing "monitor" system. Much more will be said about this "monitor" capability and the Fortran 60 System. The remaining 40% of the prime computing time is utilized for class laboratory instruction, maintenance and update of systems library tapes, data retrieval, schedule analysis, and other special job users.

The non-prime time, 1630 to 0800, is scheduled block time for the various development projects and long run programs.

This environment places increasing demands upon the computing center's capabilities. Not only is improved efficiency required but also more flexibility is in demand as the programmer's talents become more and more sophisticated.

Several tasks were undertaken in the scope of this thesis project. Towards improving over-all system efficiency, a comparative study of the present Fortran 60 System and its proposed successor, "Coop Monitor/Fortran 62 System", was conducted. As a result, an outline and a discussion of the outstanding advantages and disadvantages of each system is incorporated in this thesis.

Towards improved system flexibility, two new facilities were developed. They were the implementation of an operational remote (satellite) computing station and a system's provision for automatic graph plotting of two dimensional computer

derived arrays.

The full implementation of the remote station complex is a continuing group project with much remaining to be accomplished. As an aid to those who pursue this task, several recommendations are made and suggested methods for solving some of the problems are expounded.

Besides the work accomplished in establishing the satellite station, another aspect of the center's services was improved. This was the provision of a graph plotting system. This required the development of a plotting program for both the Fortran 60 System and a much improved and shorter Grafplot program for the Coop Monitor System. The major portion of this project was completed during the author's five week industrial tour at the Control Data Corporation, Palo Alto, California. The off-line processing of this graph plotting program was provided by rewriting, with many revisions, an existing plot drive program initially developed by the U.S. Naval Fleet Numerical Weather Facility, Monterey, California. Versions of this program were developed to operate on either the 160 or 160A CDC computers. The plotter employed is the CDC model 165/CALCOMP incremental plotter. This system has proven very effective and has enjoyed considerable usage. Although this provided an improved data output medium for the center, it has also stimulated an increased interest in utilizing the Digital Computer for project studies. Thus the center's task and effectiveness is ever increasing.

A second method of data display was provided by utilizing a new cathode ray tube, Data Display model DD65, as an "on-line" line printer. Thus it is possible to monitor the Fortran System's output listings concurrently as it is listed upon the magnetic tapes for later off-line hard copy printing.

This is a small programmed package to demonstrate a unique method of providing a "rolling page" effect on the display tube. The details of the technique are explained in section 4. It was originally planned to provide much more programming for the data display unit, however, the unexpected complexity of the Coop Monitor System limited the extent to which the authors were able to pursue this task. Modifications to the Coop Monitor System were made to couple the data display to the system. If time permits before termination of this work, the display driver routines will be completed to provide normal input/output capability between the display and the central processing computer.

Two additional projects of secondary importance were completed. They were the provision of tape control programs to enable utilization of the 1607 magnetic tape units by the CDC 160 computers when assembling programs using the CDC 160 OSAS Assembly System. The other project was the developement of a merg sort routine for large files written in 120 character BCD format. This sorting capability was a very desireable ability which had not previously been available at this installation.

2.0 FORTRAN 60 SYSTEM

The Fortran 60 System is the present fortran processor in use at the Postgraduate School. It is maintained on a system library tape, recorded in binary format, 54 words per record, see ref.21. The records are grouped sequentially such that one record or several records may constitute a program package. These packages are referred to as being "subroutines". The first such package on the system library tape is the System Control Routine and is titled "Resident". Operation of the system is as illustrated by figure 1. The figure points out the control transfer sequence which takes place after the resident program is "bootstrapped" into the 1604 computer's memory. A short verbal explanation of the diagram follows: Initial "bootstrap" procedures are executed by the operator. This reads in the Resident program from the system library. The resident program then assumes control of the computer.

2.1 Program Control

The section of resident in which control normally resides is termed "Program Control". The function of program control is to interpret control instructions which will direct the transfer of control to some secondary control package or subroutine. All such routines must eventually return control to "Program Control". Normally program control reads its instructions from the console typewriter. These control statements may be any one of the twenty routine names shown by figure 1, or one of the eight transient program names which previously may have been "called" into core (memory) by the call routine. Some of the twenty control statements have arguments. The CDC Fortran System Manual and its update revision notices

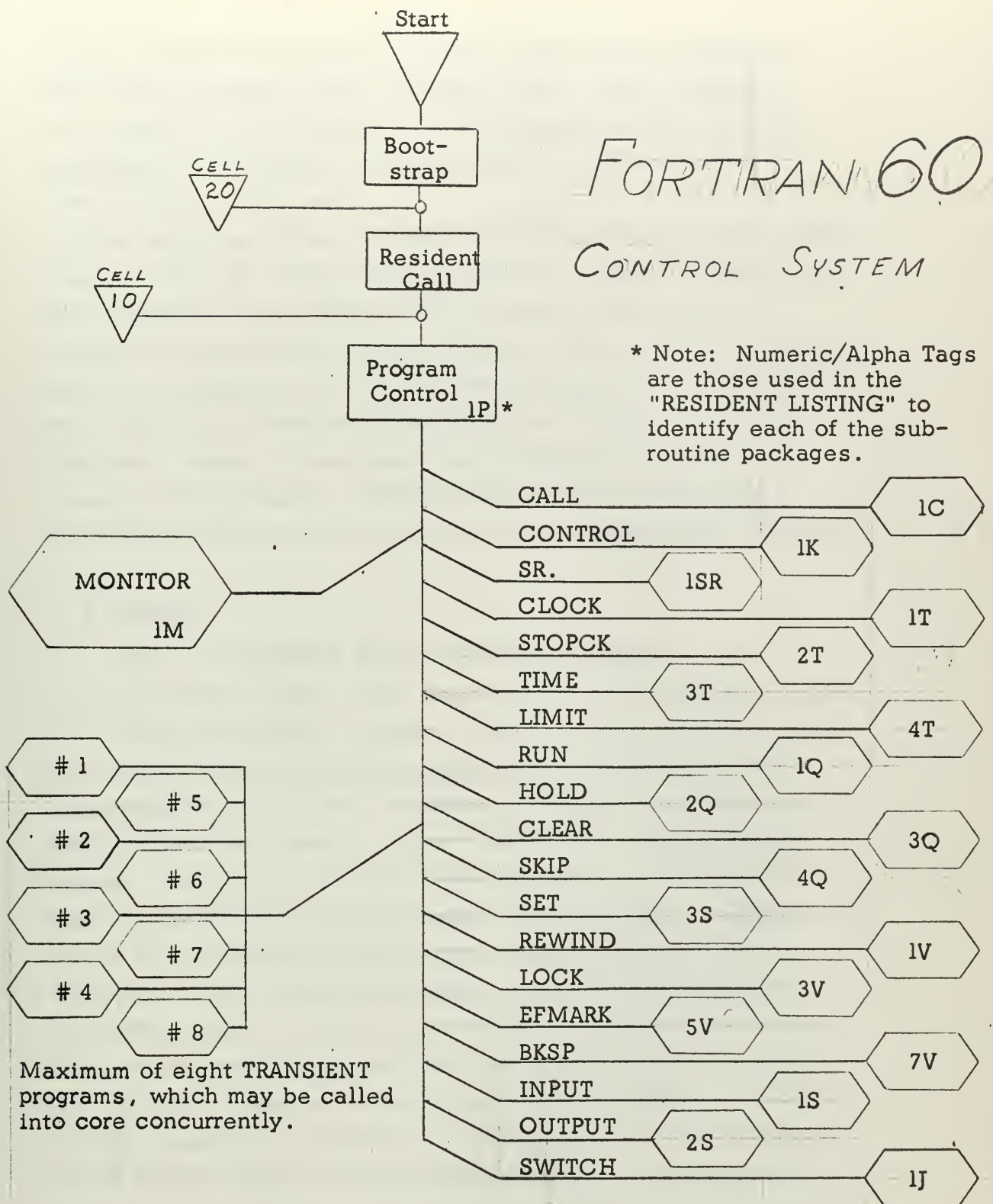


Figure 1

discuss the use of these arguments and control statements, with the exception of the statement "SR". This statement was added by the authors to provide automatic call of eight frequently used service routines such as transfer, vfylib, list, copys, verify, etc.

As mentioned before, normal operation of the control system requires that all sub-program packages, to which control may be transferred, must provide for returning control back to Program Control at the end of the directed processing sequence. Thus an examination of the resident programming, appendix IV, will find that it consists of several small programming sub-routines. Some of these are self contained and some are linked to other routines which assist in the performance of the functional task for which each basic package was designed.

2.2 Monitor

It has been pointed out that "Program Control" has a very responsible task in the organization of the system, since it is the prime control routine. The next most important and the largest of the programming packages in resident is the processor for the control statement, "Monitor" or sometimes called "Monitor Control". It is a routine to which program control can transfer control of the computer to provide automatic processing of stacked input Fortran Programs. Monitor is by far the most complex of the resident routines and yet, compared to the second generation system, Coop Monitor, it is rather simple. A full understanding of the monitor routine is best obtained by reading this section of coding in appendix IV. Briefly, operation of the monitor is as follows. Program Control receives a statement, "Monitor,3,4." This statement could have had up to seven arguments, but for this example two will be sufficient. This statement directs program control

to transfer the arguments into the monitor control routine and then to transfer control to monitor. Monitor then tests certain flags to check if recovering from a previous job execution which lost control, requiring a forced recovery entry back to monitor. This not being the case, monitor then ^wreinds all output mediums, senses for input/output conflicts, initilizes for next library routine call, builds a tape assignment table, reads from the input medium (tape 3 in this example), detects a job description record, calls in the Fortran compiler, inserts the necessary arguments into the compiler, then transfers control to the compiler for compilation of the input program. If the program is successfully compiled, control is returned to monitor and monitor then transfers control to the program just compiled, this is termed "executing the program". If the program is successfully executed, it returns control back to the monitor which then returns to process the next job on the stacked input medium.

2.3 Resident Routines

The technique of transferring control of the computer has been demonstrated and we should now discuss a technique of utilizing the resident routines which are available to non-resident programs. The organization of the system provides what are called "low core" (cells 7 thru 35), entry points. These entry points are assigned to be the permanent entries to several of the programming packages contained in resident. All of these entries are shown in figure 2. Since these cells are permanently assigned, standardization limits all usage of the resident routines by non-resident programs to enter via the appropriate low-core cell. This then allows modifications to be made on the system's "Resident" routine without influencing the non-resident routines. As illustrated by

figure 2, eight of the low-core entry routines have switches which condition on whether monitor is controlling the job sequencing. If the monitor is in effect, then these eight entries are diverted in order to trap illegal monitor usage of input/output units.

The low-core entry routines are rather self explanatory. They are mostly service routines such as read, write, equipment controllers, and message handlers.

Nearly all the routines in resident fall into one of the two groups discussed. There are a few which are not in either group. These shall now be discussed.

Two similar routines which have not been mentioned are the "Input Index Register Loader" and the "Output Index Register Loader" routines. These two serve to translate the alphabetical equipment designators into the equipment codes as listed on the first page of the "Resident List", appendix IV. These equipment codes are then loaded into index registers #1 and #2. The index registers then control the proper equipment selection for the read/write routines.

Two other routines are the "Tape Read/Write" routines. These are both tape drive routines for controlling the system's magnetic tape units. The routines are commonly used by both the BCD and Binary Read/Write routines. The tape drive routines apparently were the only equipment drivers deemed worthy of individual programming. All other equipment drivers are embedded into the main body of the Read/Write routines.

There is a routine called, "Initialize for Next Call". Its function is to reset the program bias and program counter cells, #62 and #63 respectively, prior to calling each program from the library tape. This routine is bypassed if

"LOW CORE", CELLS 7 THRU 33
 RESIDENT ROUTINE ENTRIES
 WHICH ARE AVAILABLE TO
 NON-RESIDENT PROGRAMS

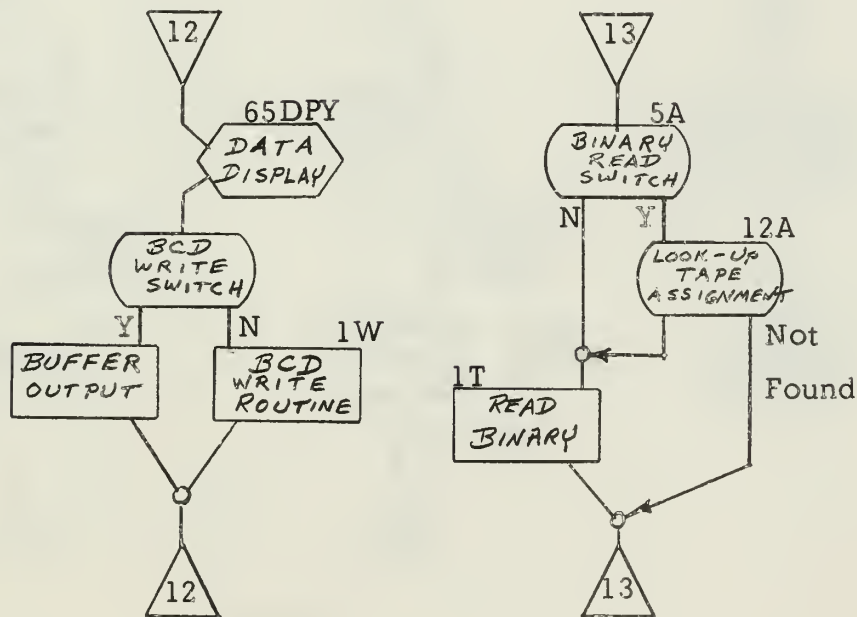
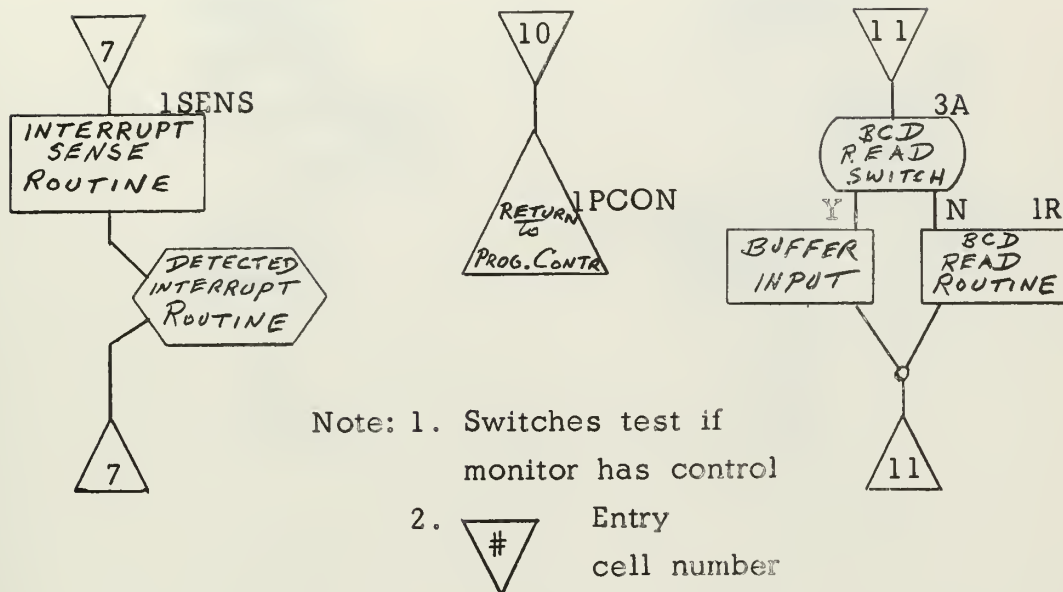


Figure 2 a

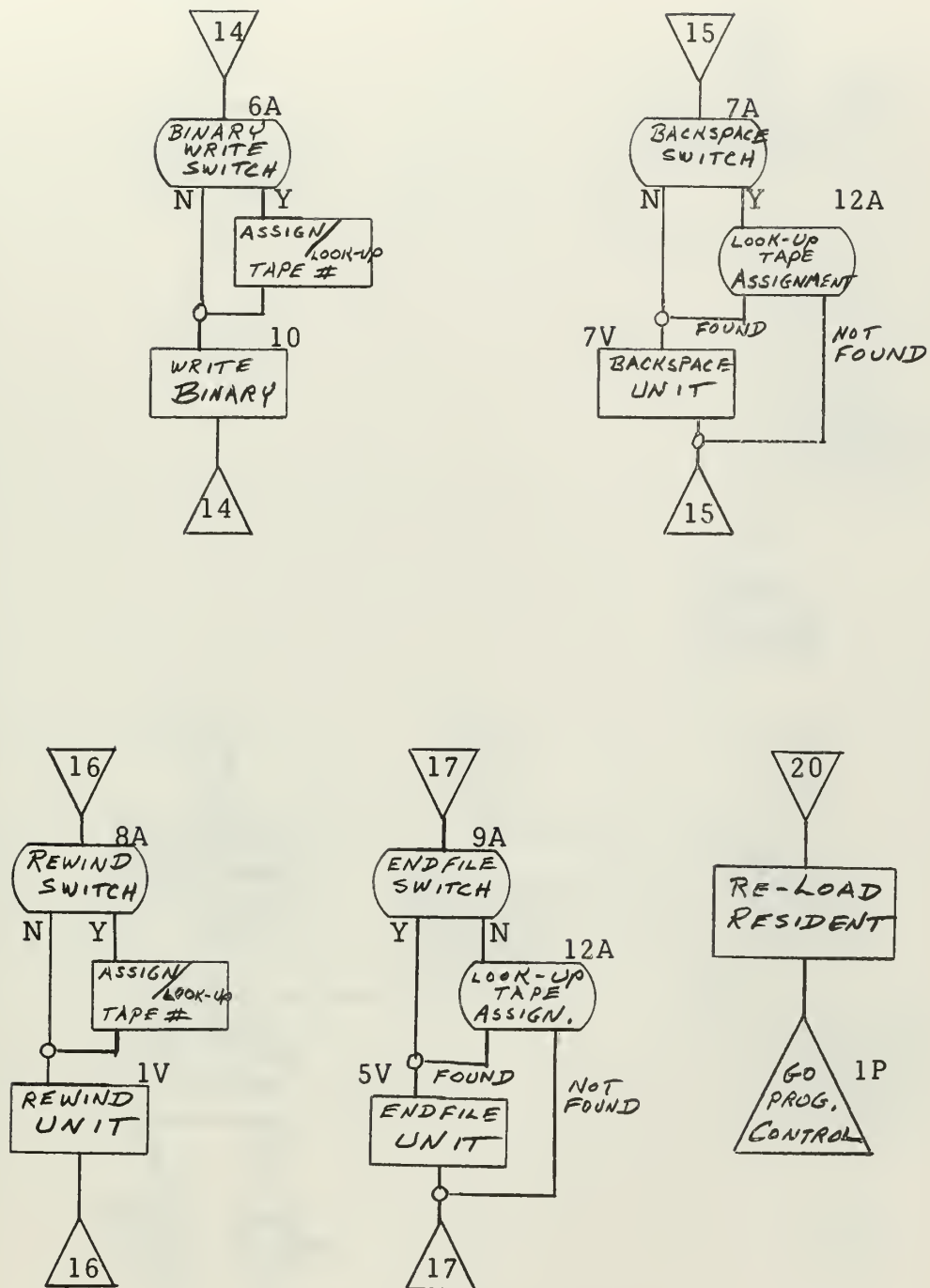


Figure 2b

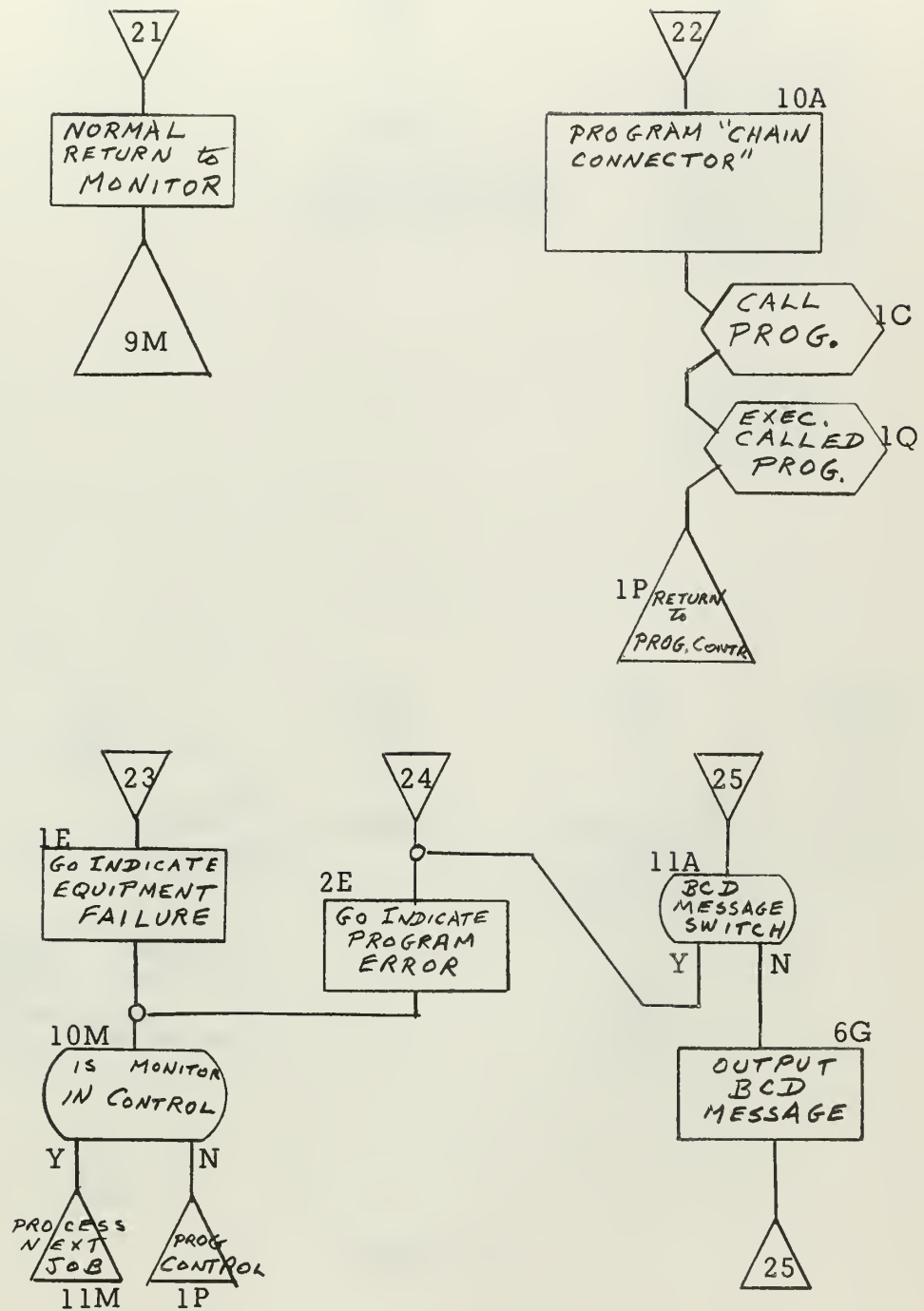


Figure 2c

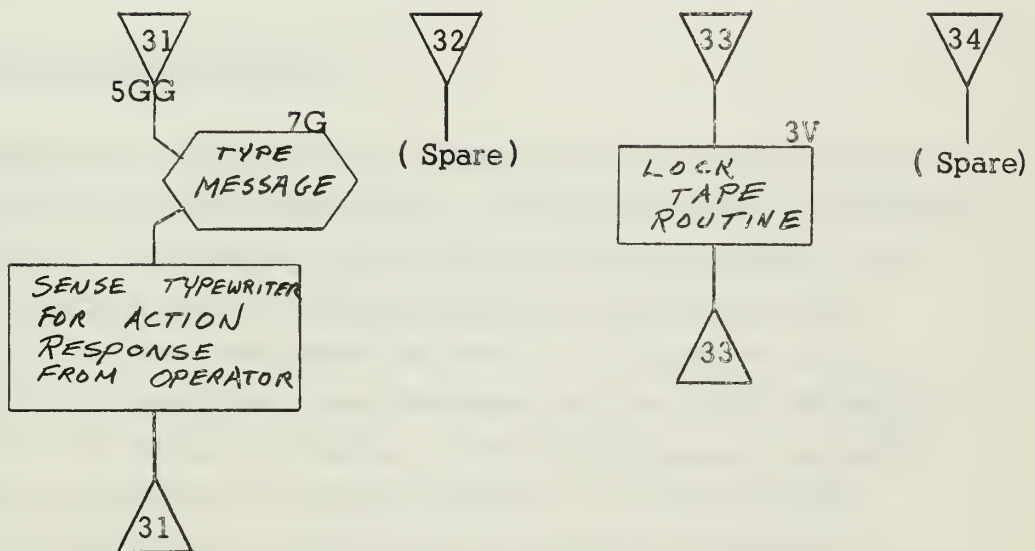
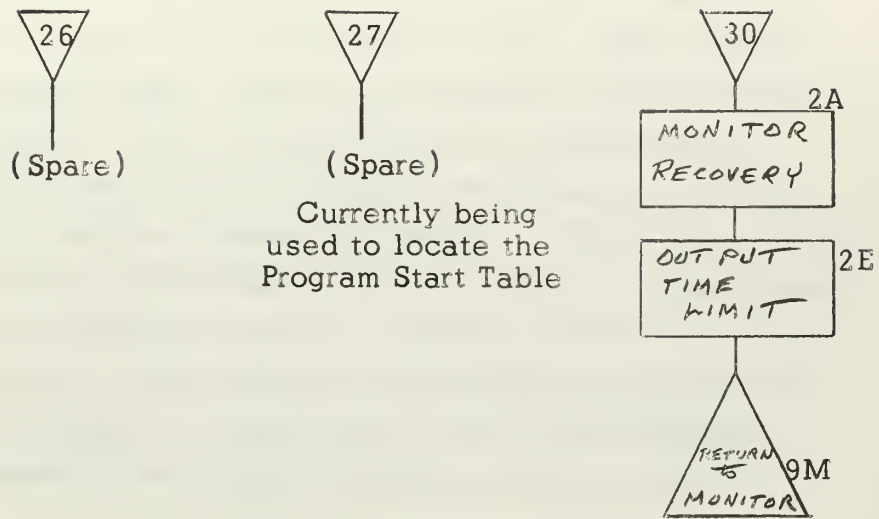


Figure 2d

a "Hold" control statement was issued immediately prior to the issue of the "Call" statement.

One final routine to be discussed is the low-core cell #22 entry routine, "Chain". Its purpose is to call a named program from a designated library medium, then execute it without packing any arguments. This provides a method of segmenting a program which is too large to fit into the available core memory along with its "Common Storage Data Arrays". Thus the segments can be called into core one at a time for execution. Each segment over-writes the previous segment. Processed data from one segment to the next is inter-changed via designated common storage. This idea is extended one step further in the COOP MONITOR system by providing both segmenting and over-lay modes of operation.

The remaining resident routines, which were added to resident by the authors, shall be discussed in section 4.2 and section 4.4. These sections concern the "Data Display Programming" and the "Satellite Station Programming".

2.4 System Characteristics

Perhaps the most significant characteristic of the Fortran 60 System is that in its original form, its Resident control programming including all equipment control, job sequencing monitor, and basic service routines occupied only 4000 octal cells of core storage. The next most important feature of the system is its simplicity of organization when compared with the COOP MONITOR system. This is due, of course, to its limited scope. For some uses this simplicity is of great importance. Later the COOP MONITOR system will be discussed, at that time we shall see

that it has many advantages over the simpler Fortran 60 System. With this comparison to make, it is necessary to list the relative advantages and disadvantages of the Fortran 60 System.

2.4.1 Advantages

- a. **SIZE;** As mentioned, the basic control programming was a modest 4000 octal cells. The authors' additional programming to provide a remote satellite computing station, and cathode ray tube data display facility increased the resident size to only 5000 octal cells. This remains a fairly acceptable figure.
- b. **FAST STACKED JOB PROCESS TIME;** The Fortran 60 monitor operates in a mode which allows re-use of the Fortran Compiler by many different jobs. This saves library reference time and hence saves process time, provided the compiler remains undamaged. The compiler compiles in memory only and uses no scratch tapes, this also leads to a very fast compilation. There are disadvantages to both these techniques which over shadow the advantages. These will be discussed as listed disadvantages.
- c. **MINIMUM INPUT / OUTPUT UNITS REQUIRED;** It is possible to operate the Fortran 60 System with only three I/O units operating. This can be a big advantage when several I/O units are inoperative or are assigned for utilization by the center's 160 computers.
- d. **OPERATOR SERVICE ROUTINES;** A well developed repertoire of operator service routines are an important advantage to a system. Fortran 60 has several such routines such as; endfile mark tapes, rewind, lock, duplicate tape, verify, list, etc. Given a little time, the COOP MONITOR System will probable incorporate several of these routines.

e. OPERATOR TRAINING TIME; At the Postgraduate School many students are given instruction, check-out, and direct access to the school's computers. As many as 250 students may participate in these laboratory periods per year. Fortran 60, with its simplicity provides a good vehicle for instructional purposes.

f. MONITOR / OPERATOR REQUIREMENTS; The Fortran 60 monitor system requires a minimum of operator intervention to process a stacked job run. Unless one of the jobs "blows the system Off-Line", ie ; manages to execute a storage or jump instruction erroneously, the system requires no operator assists. The COOP MONITOR requires considerable operator manipulation of "Jump Keys" and loading / unloading of tapes, etc. Thus the Fortran 60 System normally requires less operator supervision.

g. LIBRARY MAINTENANCE; Library maintenance and updating should not be a daily occurrence, however it is required frequently. Therefore a quick and reliable system to accomplish this task is required. Fortran 60 has three routines and a simple organization which make this an easy job. The routines are; transfer, vfylib, and list. In contrast, the COOP MONITOR System's "Libedit" routine is rather complex, is indirect via punched cards, and requires apriori knowledge concerning the current "directory" of routines on the library. It would be nice, if a directory dump routine was written and also an individual routine verify capability is required.

h. MIXED FORTRAN and SYMBOLIC STATEMENTS; Fortran 60 incorporates a fortran compatible symbolic language. Programs written for this system can inter-mix both symbolic and fortran statements on a line for line bases, if desired. For many uses such as programming shifts, logical and masking operations, searches, and other computer operations which the Fortran 60 language doesn't provide, this ability has a distinct advantage.

i. INTERRUPT ROUTINE USAGE; The only usage of the interrupt routine in the Fortran 60 System is for providing a real time clock. This would appear to be a gross neglect of such a powerful equipment capability. The advantage, here, is that the provision for a satellite station operation requires much usage of this interrupt routine, hence having a system which doesn't utilize the interrupt in some complex manner is unique.

2.4.2 Disadvantages

a. COMPILER VULNERABILITY; As mentioned under advantages, the Fortran 60 compiler is normally called into the computer's memory only when initializing monitor processing of a stacked job run. Thus there exists the possibility that one of the individual jobs may possibly damage the compiler. There after the remaining jobs of the stack are forced to be compiled by a compiler which is no longer reliable. This can result in significant loss of computer and programmer's time, since the programmer is led to believe his program was incorrect.

b. SLOW TAPE HANDLING TECHNIQUES; Two standard length tape records are employed. Standard binary records are 54 words long. Standard BCD records are 15 words long, thus providing for 120 character records. This standardization is an advantage in some instances. However, the disadvantage is the cost in time which must be paid since the tape motion must be stopped and started between each record. The COOP MONITOR System solved this problem quite well. A second detriment to tape handling is the lack of utilizing the buffer input / output channels to advantage. The only usage of the buffer capability is employed under monitor process control. In this instance, very poor usage is made since there is a complete disregard for error checking during input / output operations. This slow tape handling characteristic is a serious

one since much time is spent by the system in searching the library tape. The library tape can grow to be of considerable length due to the considerations discussed as a disadvantage due to "Multiplicity of Subroutines on the Library". COOP MONITOR solved this problem by providing long variable length library records and a directory which prevents searching the full length of the library tape for routines which are "not available".

c. PROGRAM SIZE LIMITATIONS; Since Fortran 60 doesn't utilize scratch tapes during compilation, it requires that memory must hold the resident program (4000) plus the compiler (14000) plus the compiled version of the user's program plus an undefined number of cells outside the compiler which are used by the compiler for tables and temporary storage. Thus the programmer losses at least 20,000 octal cells. This is a serious limitation in many large programs. COOP MONITOR has improved this situation, but at a considerable cost in speed, due to the necessity of re-loading the compiler for each job and also having to provide an intermediate and "Load and Go" tape.

d. INEFFICIENT SPACE COMPILATION; Fortran 60 compiles the full size of arrays and reserved space into the binary object program (ie. card images corresponding to the compiled program) at compile time. This coupled with the compilation limitation mentioned previously tends to further hinder efficient usage of memory space during compilation of programs. One advantage this does provide is that arrays and reserved spaces are initially cleared in a Fortran 60 program. This is not true of a COOP MONITOR compiled program.

e. MULTIPICITY of SUBROUTINES on the LIBRARY; Fortran 60 requires that all subroutines called by a program be available at compile time. The reason for this is that they are embedded into the compiled program as it is compiled. If this compiled program is then added to the library, the library is unnecessarily lengthened. This is due to the fact that versions of the existing subroutines are embedded in many programs. Not only does this scheme of compilation lengthen the library tape and thereby slow the system down, but it also creates very serious library maintenance problems. Thus the simple corrections or updating of a subroutine isn't so simple anymore. In order to correct a subroutine, it is necessary to re-compile all programs which use the subroutine. This is due to the embedding effect. This can get to be a big bookkeeping problem and can create a lot of work. Again the COOP MONITOR System has solved this problem.

f. NO PROVISIONS FOR MIXED BINARY / BCD INPUT PROGRAMS; Fortran 60 does not provide any method for inputting a binary program or subroutine into a stacked monitor run, if it is not already a part of the library. Thus no mixed input to the system is allowed. This requires frequent recompilation of subroutines and programs which are not incorporated onto the library tape. Again the COOP MONITOR / FORTRAN 62 System provides the solution.

g. LIMITED "DE-BUGGING" AIDS; The only automatic debugging aids incorporated into the Fortran 60 System are the compiler and arithmetic error print-outs. The selectable aids consist of a "Map Language" (see ref. #21, pg #102) listing of the compiled program, and a few dump routines which were never very effective. A magnitude improvement has been made by the COOP MONITOR System towards providing de-bugging facilities, both automatic and optional.

h. UNDESIRABLE INTERMEDIATE LANGUAGE FORMAT; The intermediate language of the Fortran 60 compiler is "Map Language". This form of computer coding can become familiar, but only a small percentage of the general users of the computer facility were ever motivated to use it. Three "languages" for coding a single compiler seems to be more than should be required. Successful utilization of the Fortran 60 System requires a knowledge of the fortran language, CDC Symbolic format, and of the intermediate language, "Map". The Fortran 62 system is a two language system and is quite acceptable except for the exclusion of the line for line inter-mix capability between fortran and symbolic coding.

i. POOR INPUT / OUTPUT FLEXIBILITY UNDER MONITOR CONTROL; The Fortran 60 System originally had a very weak tape assignment system. It would assign tapes for scratch when the units weren't even loaded. This was corrected by the authors for the school's facility. Another deficiency is the inability of the monitor to provide multiple input / output BCD units. This is desired in many programs, for instance, when cumulative data is being generated and processed over a long period, or when special table lists are to be formed by a program, etc.

j. SLOW CARD READ / PUNCH ABILITY; The school's "On-Line" card facility is a CDC 1609 system which is capable of operating at either 50 or 100 cards per minute. The Fortran 60 system provided operation only at the 50 card per minute rate. COOP MONITOR will run at the maximum speed of 100 cards per minute and compiles as it reads if the card reader is being used for input. This rate is still too slow for a general usage on-line input medium, hence it is used only as a back-up system for the IBM 1401 System which normally processes the card to tape conversion "off-line".

3.0 COOP MONITOR SYSTEM

The COOP MONITOR SYSTEM was developed to allow a computer facility the ability to operate with only one "library" tape and yet effectively process any type of input programming or source language. This would eliminate the need to sort incoming jobs into separate groups to be run under individual library tapes. The advantage of a single library is readily apparent when operation from a remote station is anticipated, where the operator is physically separated from the main computer and unable to change the current system library tape.

To increase COOP MONITOR'S flexibility it is organized into two basic levels of control. At the highest level is the Master Control System (MCS), which maintains the overall system coordination and supervision. At the next level is the Secondary Control System (SCS), and there may be any number of these available on the library. Each SCS is designed to perform a particular job or combination of jobs depending on the requirements of the computer facility.

A brief description will now be given covering the basic organization and operation of COOP MONITOR, for a more detailed description refer to ref.10.

3.1 Master Control System

To accomplish the desired system supervision, MCS is organized into nine functional subroutines and the Bootstrap loader (BOOT), see figure 3. BOOT and the first eight subroutines make up the "permanent" resident, with the memory space occupied by the Job Sequencer (6000 cells) released for use each time control is given to the Secondary Control System. After the SCS has finished using the subroutine

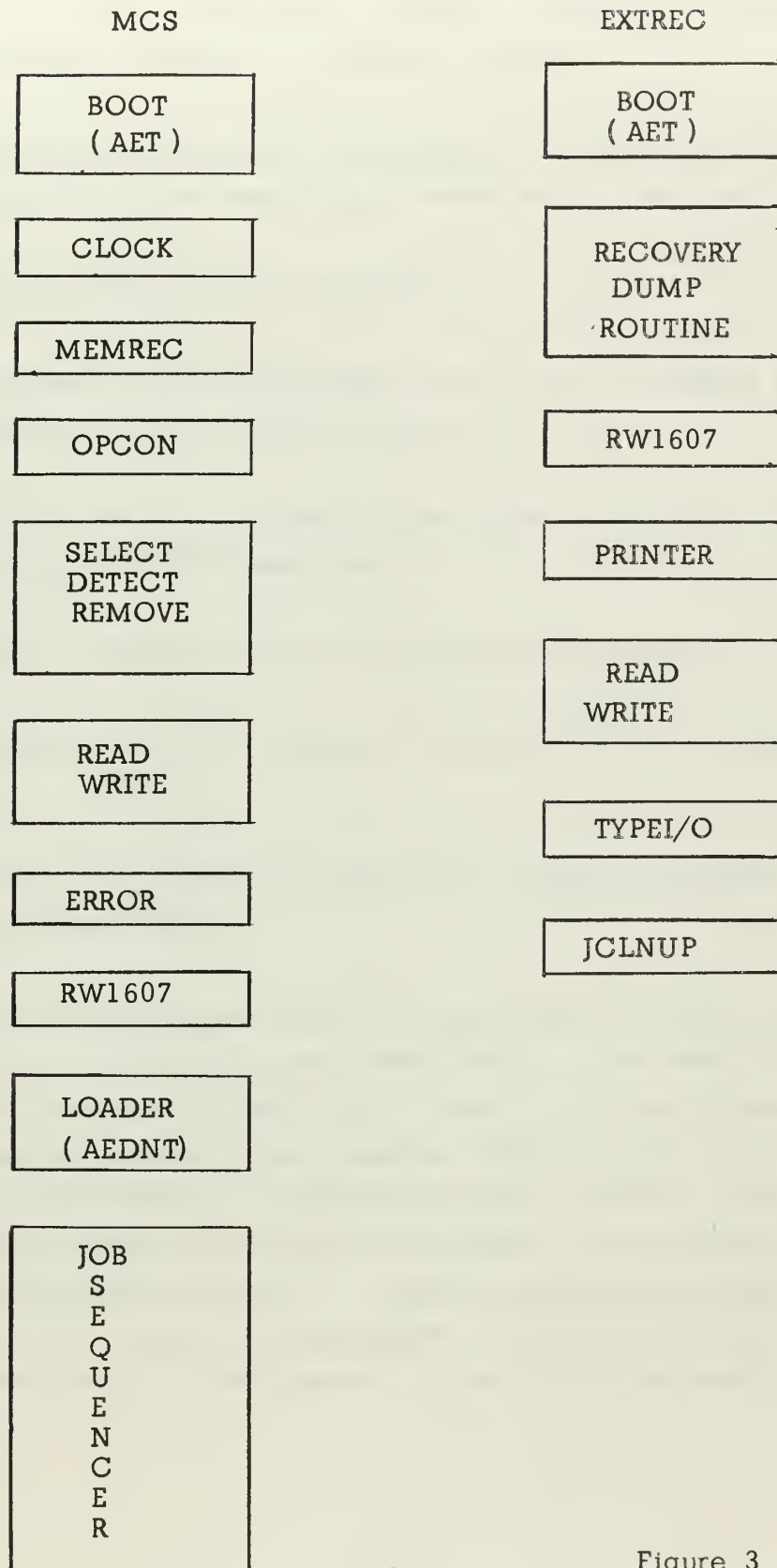
"Loader" the memory space it occupied may also be released for use by the current program. This procedure allows MCS to be very large and complex (13500 cells) and yet not unduly penalize an input program in memory space requirements. The basic functions of each subroutine are as follows:

- a. BOOT; Contains the Available Equipment Table (AET) , which enables MCS to determine the input/output units available to the computer and their intended use. This routine will load on option either MCS or the system recovery dump routine "EXTREC". BOOT also contains the MCS flags to store current job information to allow intra-job continuity.
- b. CLOCK; Allows use of the computer real time clock to set a time limit on the current job. This routine is available to the programmer to set one additional time limit and also may be used to determine elapsed time.
- c. MEMREC; Maintains a record of the limits on the one contiguous block of available memory between resident and any programs loaded into high core.
- d. OPCON; All communication between the computer operator and MCS or SCS is thru this routine. This input or output is normally via the console typewriter.
- e. INTERRUPT; This routine has three sections, SELECT, DETECT, and REMOVE. These allow for the selection and detection, or the release of any of the internal computer interrupts or on channels 1 to 6 inactive.

- f. READ/WRITE; This subroutine controls all input/output operations of the computer. Provides protection from illegal use of any unit and prevents interference between units on the same channel.
- g. ERROR; Used in the event of user program failure or the detection of illegal operation by MCS, entry into this subroutine will terminate the current job. ERROR will output the current contents of all the console registers (on the standard output medium) and the lower address of cell 7, and then determine if the user desires a memory dump.
- h. RW1607; The driver for the magnetic tape units and the only driver contained within MCS. This subroutine is the only place where tape units are selected and activated, except in BOOT for loading MCS or EXTREC.
- i. LOADER; Loads all relocatable binary programs or subroutines either from the system library or user provided input medium. This subroutine packs programs into high core memory with common block storage assigned to low core just following "LOADER".
- j. JOB SEQUENCER; Used between input jobs to set program limits, determine and load input/output drivers requested by the user, and to call and transfer control to the desired Secondary Control System (SCS). This subroutine also contains the programming to process any computer operator statements received via "OPCON".

These functional blocks are closed subroutines with multiple entry points depending on the particular function desired. These entry points are available to all routines and must be used whenever one of the above operations is necessary. This also includes other blocks within MCS, example, The JOB SEQUENCER

BLOCK DIAGRAM COOP MONITOR



must use LOADER to make a library call, and LOADER in turn must use RW1607 to actuate and read the magnetic tape unit. These subroutines operating together as a Master Control System provide the following features:

- a. Automatic, variable job sequencing, including accounting records and time and output line limits set for each job.
- b. Operator-Machine communication.
- c. Assignment of all input/output units and the loading of the necessary drivers for these units.
- d. Single linkage to all input/output drivers to protect them from interference and illegal use.
- e. Flexible linkage to the CDC 1604 interrupt features.
- f. Automatic recovery features in the event of "user" program failure.
- g. Calling and transfer of control to any desired Secondary Control System (SCS).

The assignment and control of input/output equipment by the MCS is accomplished thru three tables. The first, the Available Equipment Table (AET), lists all the input/output equipment connected to the computer. Some of these I/O units are designated as "standard" (SIO), and are assumed to be always available for use by the system. The entries in the AET (one for each I/O unit), contain the following information: Allowed use, current job assignment, standard unit or not, equipment type, and the location of the I/O driver name. For

a complete description of the AET entry format see ref.10. This table is set for each installation and is changed only as equipment status changes. A listing of the current AET is shown in figure 6c.

The second, the Available Equipment Driver Name Table (AEDNT), lists the name and entry point location for each I/O driver subroutine for controlling the units listed in the AET. The entries are two words long. The first word contains the entry point locations of the routine if it is in memory, or an exit to MCS if not in memory, see section 3.5 part D for an example of an AEDNT entry.

The third, the Running Hardware Table (RHT), lists the input/output assignments of units for the current job only. The entries are AET entries with the address of the proper driver entry point inserted and the allowed use set. Logical unit numbers are used as an index to this table, logical numbers 1-49 are available to the programmer, 0 is reserved for the system library tape and 50-63 for standard units.

3.2 Secondary Control Systems

The great flexibility possible with the COOP MONITOR is in the use of a Secondary Control System (SCS) to control the processing of a specific job. There can be any number of Secondary Control Systems which may be written by each computer facility to handle the specialized needs of its users. Thus for use at the Postgraduate School the standard "stacked job" daily production runs would be controlled by one SCS, and a satellite digital controller simulation by a different SCS, and a real time radar intercept problem by a third SCS. All of these SCS would be placed on the standard library tape and available within a few seconds after a particular one is called from the library. No modification is needed in the standard COOP MONITOR

to use a different SCS. The name of the desired SCS is given to the JOB SEQUENCER in MCS at the beginning of each job and this name will go to LOADER and the SCS will be loaded from the system library.

There are two SCS provided in the present version of COOP MONITOR, these are called "COOP" and "LIBEDIT". A brief description of each is given to illustrate the technique of tailoring a SCS to do a specific problem. The first, COOP, is responsible for the intra-job sequencing necessary to compile and execute an input job, ie; the calling of compilers or assemblers, transferring binary card images to a "load and go" tape, calling LOADER to load the "load and go" tape and finally, execute the loaded program. The present version of COOP can call two compilers, FORTRAN-62 and CODAP1 (actually an assembler), but COOP can be modified to accept any number of compiler languages. COOP also contains internal routines for user controled binary deck manipulation, loading and execution.

The second, LIBEDIT, has two subroutines, Edit Library Tape (ELT) and Prepare Library Tape (PLT). Edit Library Tape is used for minor changes in the system library such as a deletion or addition to the subroutine file or within MCS and EXTREC. This secondary control system makes a copy of the present library tape upto the point of the desired change and then inserts or deletes the proper information. For a detailed description of the steps necessary in a complete LIBEDIT, see appendix I and ref.17. Prepare Library Tape is used for major changes to the system library. The user must now supply all the information necessary to produce a new version of the system library tape, nothing is copied from the old library tape. LIBEDIT does not provide any means of verifying the portions of the new library tape that were not changed.

3.3 System Operation

A brief description of the operation of MCS can now be given, for a more detailed description see ref.10. The first record of the library tape, BOOT, see figure 4, is read into the computer by manually selecting and activating the proper magnetic tape unit. BOOT is then executed at cell #2, see figure 6, and will verify the AET table and then skip over the next record on the library tape, EXTREC, and load MCS, see figure 4. Control is now given to MCS which determines the standard input/output units (SIO), loads the necessary drivers for those units if they are not in memory, fills the unused memory space with a jump to ERROR instruction to trap user program failures, and then notifies the computer operator the system is ready to accept job assignments. The operator then either signals MCS to alter the AET or SIO assignments or to begin processing the "stacked job" on the standard input. When signalled to begin processing, MCS reads the control card from the standard input, logs the first job on the standard outputs (list and punch tapes and the console typewriter), and sets the time and output line limits for the current job, the recovery key is stored in BOOT. The input/output units needed, in addition to the standard units, by the incoming job are determined and assigned and if the proper drivers are not already in memory they are called and loaded. MCS then releases the memory space occupied by the non-permanent section of MCS (Job Sequencer) for use by the SCS or the input program. The desired secondary control system is determined and loaded into memory and MCS then transfers control to this SCS, see figure 5. When the SCS has finished processing the current job, control is returned to BOOT with an indication for a "normal" or "abnormal" job termination. For a "normal" termination BOOT reloads MCS

COOP MONITOR LIBRARY TAPE FORMAT

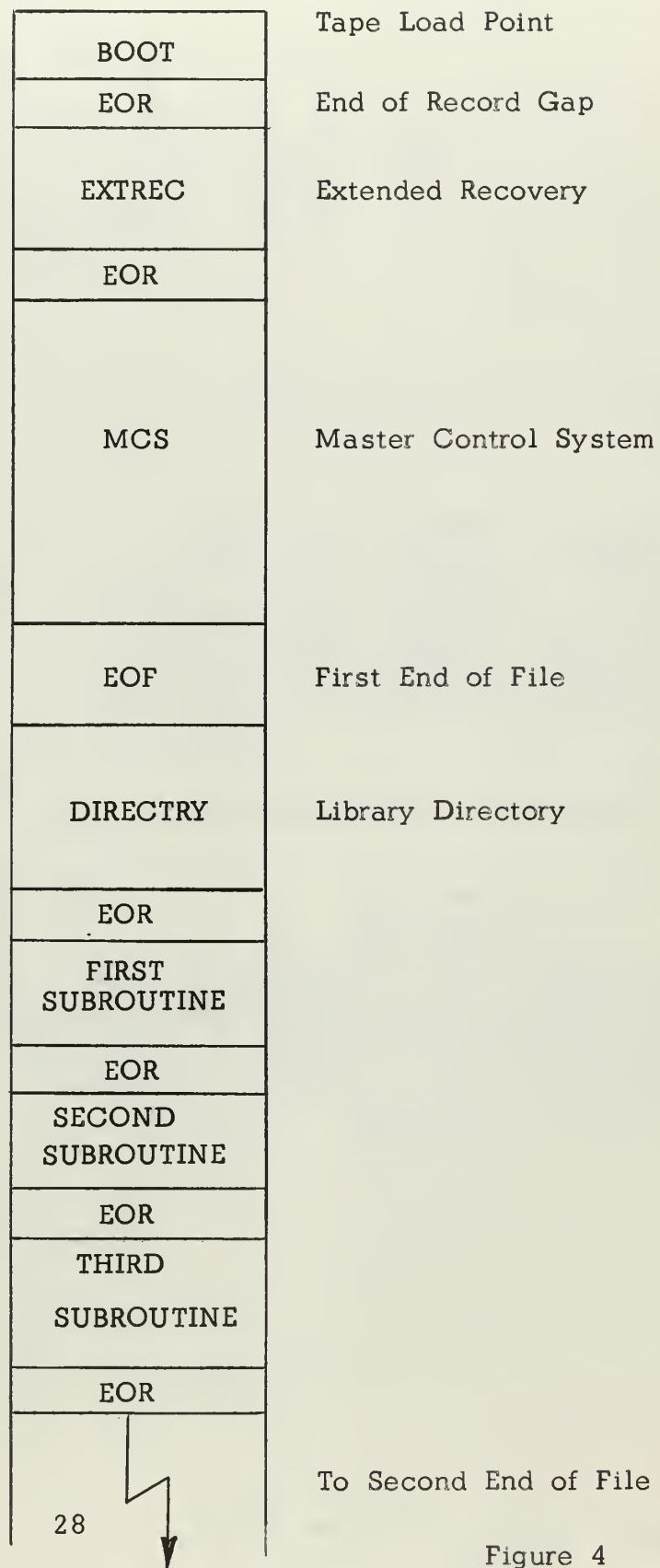
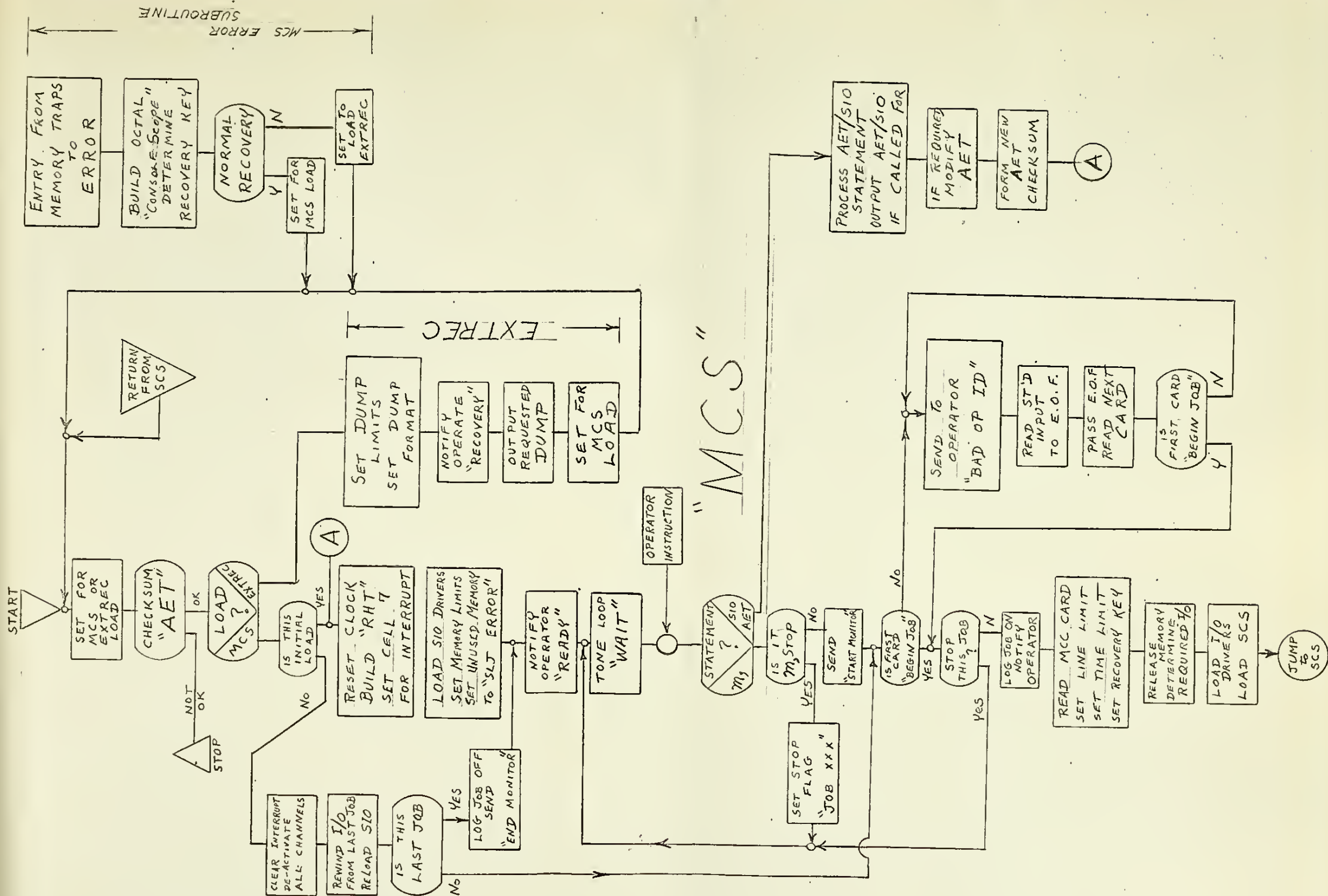


Figure 4



COOP MONITOR SYSTEM MCS
FLOW DIAGRAM

Figure 5

and if there are more jobs on the standard input, the cycle is repeated. For a "abnormal" (error) termination a "Console Scoop" (the contents of all the console registers) is prepared and sent to the standard output tape to aid the programmer in debugging efforts. The recovery key, set by the user at the start of the job, is examined to determine if a memory dump is desired, if not control is given back to BOOT for a MCS reload and the process continues as before. When a memory dump is desired control is returned to BOOT for a EXTREC load, see figure 6. Extended recovery then provides the desired memory dump on the standard output tape and returns to BOOT for a MCS reload. Again if there are more jobs on the standard input tape MCS will start the sequence over, otherwise it will terminate this run and notify the computer operator.

3.4 System's Characteristics

Once over the initial shock of working on a system of this size, the next effort was to consider COOP MONITOR'S good and bad characteristics when the special requirements of the Postgraduate School computer center were included in the system. These special requirements included the daily "stacked job" production runs of short duration, longer runs by the students during the evening, the increased use of the satellite capability of the schools computers, and the addition of a sophisticated digital display unit (Data Display Inc. DD65) as a remote computer input/output device. In this light the COOP MONITOR was determined to have the following major advantages and disadvantages:

3.4.1 Advantages

1. The system can handle stacked input jobs written for multiple compiler languages. The present version includes Fortran-62 and Codapl and the addition of the Neliac compiler and the Scrap assembler is possible.

2. Operates with multiple secondary control systems each specifically designed for a job. Present system has two, COOP and LIBEDIT and others may be added to process satellite and remote display problems.
3. Automatic, programmer controlled, debugging aids such as the "console scoop", provided in the event of program failure and the "snap" feature for those that do not fail but also do not produce correct results.
4. Automatic system recovery in the event of failure by the input program.
5. All input/output is thru one routine, therefore it is very difficult for an input program to illegally use any I/O unit. Interference between units is also minimized.
6. COOP can accept mixed, binary and source language, input programs.
7. Magnetic tape driver routine uses variable block read and write for maximum speed when controlling the system library tape. All read/write is buffered, with parity and length checking.
8. Variable standard input/output units are allowed and these can be changed before each job.
9. Multiple input/output units for each job are allowed up to the maximum available to the computer.
10. The system is capable of handling very large input programs as MCS releases memory and the compiler used is over written when the program is loaded for execution. For programs that are still too large, segmenting and overlaying are provided.

11. The compiler used, Fortran-62, has special features all its own; easy to read intermediate language (Codapl), buffer operation for input/output, and single library subroutine call for each program.

12. Compiler is reloaded for each job thus assuring a clean start.

13. The system will accept Fortran-63 when it is available.

3.4.2 Disadvantages

1. The system is very large (13500 cells) and complex, making it difficult to gain the understanding necessary to perform any system modifications.

2. Many complex (at first) control cards are necessary to process each input job.

3. Provides an unused and unsuppressable accounting information output on the paper tape punch.

4. Normal system operation requires a large number of standard input/output units to be operating. Four magnetic tape units (library, punch, input, and output), the paper tape punch for system accounting, and the console typewriter are required to satisfy the "standard" I/O requirements.

5. Inability of the computer operator to add or delete a standard input/output unit (the type of I/O unit can be changed).

6. Step by step operation by the computer operator via the console typewriter is not possible.

7. The interrupt schedule is not compatible with a real time problem solution in a micro-second time sharing mode, due to the frequent use of interrupt lockout. No provision is made for operation from a remote (satellite) station.
8. No programmed restart available for use by non-autoload CDC 1604 computers (the autoload must be simulated by manually performing the required instructions).
9. No operator service routines are available such as call, copy, transfer, verify, etc.
10. Program errors are sent only to the standard output tape requiring the user to list this tape before determining any errors.
11. The system would not output/input on the CDC 1609 card punch/read unit available at the computer facility.

3.5 Alterations To Coop Monitor System

After a basic understanding of the Coop Monitor System was obtained, it became apparent some modifications would be required before the system would be acceptable for operation at the Postgraduate School's computer facility.

The first modification attempted was the alteration and corrections needed in the cardread and cardpunch drivers to allow operation with the CDC 1609 card read/punch unit. In addition to the corrections, it was also necessary to rewire the unit's plugboard to make the read/punch format compatible with the Coop Monitor. These read/punch routines are an improvement in speed to 100 cards per minute vice the 50 cards per minute available with Fortran-60. Once the drivers were corrected, they were recompiled and the binary versions

were added to the system library tape using the secondary control system "LIBEDIT". For a complete description of the LIBEDIT procedure, and the input job format see appendix I. The correction of these subroutines provided a library tape more suited to the needs of the school computer center but still left some major deficiencies in the system that would prevent convenient operation. These were the following:

- a. The unsuppressable accounting medium being output on the paper tape punch, requiring the punch to be selected at all times. This made the paper punch unavailable to the user as an output device.
- b. No provision for a programmed restart for operation on a non-autoload computer (CDC 1604). In the event of MCS failure a manual "bootstrap" is required.
- c. Error messages from MCS "LOADER" are not available to the user except by listing the standard output medium.
- d. No provision for handling operation from a remote station.

The correction of these deficiencies all had one thing in common, they would all involve changes within the structure of MCS. Many early failures soon made it apparent changes in MCS could not be made without a complete understanding of the internal organization of MCS, and a positive knowledge of the operation of LIBEDIT. After these two programs were more thoroughly understood, the modifications resulted in the successful production of a new library tape providing these new features:

- a. Accounting medium suppressed, the paper tape punch is not selected unless the operator intends to use it as an output medium.
- b. A programmed restart is available, at cell #30, and this will simulate the "autoload" feature of the CDC 1604A except for the initial load.

- c. All error messages from MCS "LOADER" are now also output on the comment to the operator medium, normally the console typewriter.
- d. All linkages necessary (AET and AEDNT entries) for the calling and execution of a driver subroutine for the remote display (DD65) are now in MCS. Space required for this driver is reserved (1000 cells) within the "permanent" section of MCS just prior to LOADER.

These modifications now provide a library tape that should be much more convenient for the computer center to use and at the same time provide a system that will readily accept experimental drivers and subroutines for operation from the remote station. The detailed explanations and coding involved in these modifications are shown in the following sub-sections.

A. Modifications to BOOT

The postgraduate schools computer (CDC 1604) is a non autoloading machine and thus requires a manual selection and activation to read the library tape. The COOP MONITOR uses the autoloading feature anytime there is a failure of the automatic recovery routine. To avoid the necessity of this manual "bootstrap" each time this may occur a simulated autoloading was placed into BOOT starting at cell #30. This allows an "autoloading" restart when needed. At the same time the standard input/output assignments in the Available Equipment Table (AET) were altered to provide the following:

- a. Library on channel 3/4, tape unit #1
- b. Input on channel 3/4, tape unit #2
- c. Available for use, channel 3/4, tape units #3 and #4
- d. Output on channel 5/6, tape unit #1
- e. Punch on channel 5/6, tape unit #2
- f. Scratch 1 and 2 on channel 5/6, tape units #3 and #4


```
START MONITOR RUN.
BEGIN JOB 231
COOP,'S/2S,20,50000.
CODAP1,1,1,.
```

RANGE

ENTRY POINTS

IDENT	BOOT
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

FWA
00002 - LWA+1
14557

01205	READ**
01212	WRITE**
01265	SELECT**
001143	REMOVED**
001743	DETECT**
017763	GETCH**
017773	CHKSTD**
04063	LOADER**
04064	RELOC**
02743	LIBREW**
02657	CLANBCD**
02615	EXIT**
01777	ERROR**
01174	MODIRET**
00401	BCDRN**
00335	OPCOM**
02235	RECLIM**
02252	FLAGTST**
00524	MEMREC**
00217	CBF**
01742	CBF1**
01743	CBF2**
01744	CBF7**
01751	IOSELECT**
00714	MWRW600**
01743	LOK1**
06342	LMSRCH**
05745	ILMF**
01125	FCEN**
01752	EXSE**
01034	SETCLK**
00167	DATE**
00076	SIEOF**
00061	LIBPOSIT**
02732	

NO	EXTERNAL SYMBOLS	INTERNAL SYMBOLS
00002	50	1
00003	50	0
	74	3

CRIC
CENTRE

1	CHECKSUM-9
	CHECKSUM
3	400008

SET IX

TERMINATE BUFFER

ADDRESS	OPERATION	OPERAND	COMMENT
00004	ADD	00010	ACCUMULATE
00005	IJP	00004	CHECKSUM
00006	AJP	00005	
00007	SLJ	00014	CKSUMERR
00008	STA	00003	
00009	EXF	00070	70B
00010	SLJ	00023	LOAD
00011	NOP	00022	PASSONE
00012	EXF	00003	
00013	ENI	00101	101B
00014	LAC	00045	CHECKSUM-9
00015	ADD	00057	CHECKSUM
00016	IJP	00012	
00017	EXF	00070	70B
00018	EXF	00103	100B
00019	AJP	00015	LOADON
00020	SLS	00014	
00021	EXF	32011	32011B
00022	EXF	32005	32005B
00023	SIL	00003	
00024	EXF	00022	32000B
00025	QJP	00021	PASSONE
00026	SLJ	00003	*+2
00027	NOP	00003	
00028	RTJ	01777	ERROR*
00029	EXF	01777	-0
00030	EXF	32003	32000B
00031	EXF	32003	-0
00032	EXF	32000	32000B
00033	EXF	00077	EXECUTE
00034	EXF	32003	32000B
00035	EXF	32003	*+2
00036	SLJ	00026	0
00037	EXF	00077	EXECUTE
00038	EXF	00003	0
00039	SLS	00077	EXECUTE
00040	NOP	00003	
00041	EXF	00003	
00042	EXF	32015	32015B
00043	EXF	32000	32000B
00044	EXF	00002	2
00045	EXF	32003	32000B
00046	SLJ	00002	2
00047	NOP	00000	
00048	EXF	77777	
00049	EXF	77777	
00050	EXF	13201	13201B

00066	00	00000	RPRG	OCT	0
00067	00	00000	RDMPFLG	OCT	0
00070	00	00000	MARKTIME	OCT	0
00071	00	00000	LASTOT	BSS	1
00072	00	00000	OLDTIME	BSS	1
00073	37	77777	TOTIME	OCT	377777777777777777
00074	00	00000	RHT	OCT	0
00075	00	00000	REMTIME	OCT	0
00076	00	00000	LASTCELL	OCT	0
	00	00000	DATE*	EQU	LASTCELL
	00	00076	EXECUTE	EQU	LASTCELL+1

IN ROOT

All references to channel 1/2 magnetic tape units were removed (the school has no units available) and the I/O linkage for the remote display driver was inserted, see figure 6.

These modifications to BOOT altered the location of the begining of the AET and the subroutine EXTREC uses this location as an absolute address. This required recompiling EXTREC with the correct address for the AET and then placing this on the system library tape again using the routine LIBEDIT.

B. Suppression of Accounting Medium

The original version of COOP MONITOR provided for accounting information to be output on the paper tape punch. This output was of unknown format and therefore of no use to the computing center. This did demand the paper punch be selected and prevented the use of the punch as an output device. The suppression of accounting output was accomplished by taking advantage of the READ/WRITE subroutine in MCS. All I/O requests must be thru this routine and this includes any requests for output to the accounting medium, normally the paper punch. Logical number 55 is reserved for the accounting medium and to suppress the output a test for logical 55 is made each time READ/WRITE is entered. If the entry is for any logical number except 55 the request is processed. If the request is for logical 55 a no action exit is made.

This test is made by adding the following programming to the READ/WRITE section of MCS at approximately cell 1225.

INA		-55	Test for accounting
AJP	N	*+3	Jump if not accounting
INA		2	Build exit to suppress
SAU		MRW800	accounting output
SLJ		MRW800	Jump to exit

This suppresses the accounting output but still allows the paper tape punch to be used as an output medium if desired. Unless the punch is to be used as an additional output unit is need not be selected.

C. Loader Errors to the Operator

Many of the failures of programs to run under the COOP MONITOR result from errors detected by the subroutine LOADER in MCS. These errors cause messages to be written on the standard output medium and this must be listed off-line (unless the standard output is an on-line printer) before the operator can determine the cause of the error. To allow the error message to also be output on the comment-to-operator medium, normally to the typewriter, the following programming was added to the subroutine LOADER in MCS at approximately cell 6274.

RTJ	WRITE	Errors to operator	
01	53	BCD	Unit #
03	LERR	Write/Check	FWA
00	LERR+5		LWA+1
NOP		Error return	
NOP			

Loader errors are now immediately available to the computer operator as the program is being run to assist in the correction and possible rerun if desired.

D. Linkage for Remote Display

The addition of a I/O subroutine for remote operation of a digital display unit (DD65) requires that proper entry linkages be entered into MCS. This will permit the driver to be called (if not within the MCS) and executed. These linkages are in the Available Equipment Table (AET) and the Available Equipment Driver Name Table (AEDNT). Both of these links

have been provided in the present (modified) version of the library tape. In addition, a reserve area of 1000 cells has been made available within MCS (cells 3060 to 4060) to contain the programming for the display driver.

The AET entries (in BOOT) are shown in figure 6 and the AEDNT entries are located at the end of the subroutine LOADER, at approximately cell 6460 in MCS. These entries are:

BCD	DISPLAY
SEV	3070
SLJ	HOWLER

where SEV indicates the driver is within MCS
and 3070 is the entry point address of the driver

This link therefore requires the display driver to have its entry point located at cell 3070 to be executed properly. If a new entry address is desired it must be entered into the AEDNT, the AET entry need not be changed.

The final modifications made to the COOP MONITOR system were the addition of two subroutines to the library tape. These subroutines were: 1. Grafplot, to allow a graph of any two designated arrays to be drawn on an incremental plotter, see section 4.1. 2. UCSD, a new Secondary Control System written by the University of California at San Diego, which allows the use of simple control cards for running "stacked job" input for compile and execution.

3.6 Specifications for Further Modifications

The modified version of COOP MONITOR is now ready for the next step in the development of an operating remote station capability. This is the programming and insertion into MCS of a remote station display (DD65) driver, the alteration of

the system interrupt handler (SELECT, REMOVE, DETECT) to allow remote station interrupts and a subroutine to display graphical information on the remote display. The specifications for these three modifications are discussed in the following sub-sections:

A. Remote Display Driver

This subroutine would allow the display to function as the comment-to-operator medium, normally the console typewriter. One of the subroutines for this driver has been completed and is fully described in section 4.2 "line printer simulator". This subroutine will produce a line by line display on the console (DD65), that will move upward with the arrival of each new line, like a typewriter. Only the upper half of the tube face should be used for the "line printer" to allow the lower half to be used as a "status board" for computer input or output assignments and any special information for the remote station operator.

To correctly process input/output requests from COOP MONITOR (from READ/WRITE) the remote display driver must accept the standard calling sequence and correctly respond to the parameters specified. The specifications for the standard calling sequence is now given in the COOP MONITOR format:

TITLE: DISPLAY

PURPOSE: To input from or output to the remote station display (DD65) in a format compatible with the COOP MONITOR.

USAGE: Calling Sequence;

	ENA	Location interrupt routine
L	RTJ	DISPLAY
	R.M.	E.C.
L+1	F.C.	A
	I	B
L+2		Alternate Return (error)
		E
L+3		Normal Return

R.M.= Recording Mode, 1 if format is BCD 8 char/word
2 if format is 8 typ. char/word
3 if format is 1 typ. char/word

BCD 8 char/word; implies the internal format is BCD with blanks replacing typewriter characters for which there is no BCD equivalent.

8 typ. char/word; implies the internal format is 8 typewriter characters per word.

1 typ. char/word; implies the internal format is 1 typewriter character per word.

E.C.= Equipment Code with format CRXX

C= Channel number (7 for display)

R= 1 for input, 0 for output

X= Not interpreted

I= Interrupt Selection

1= With interrupt, use the address in the A register

0= No interrupt, ignore the address in the A register

F.C.= Function Code explained below.

A, B, E = Detailed under each function code.

Functions: In all cases if a parameter error is encountered E is set equal to 0 and control is transferred to the alternate return. Parameter errors are such things as channel not equal to 7, R not equal to 0 or 1, incorrect function codes, etc. E is always set to reflect the case in which the typewriter (simulated by the display) is left. This has no meaning for the display but the correct response if necessary for COOP MONITOR. The 2^6 bit of E is set to 1 for upper case.

F.C.=1

Starts transmission of data to or from an area specified by the first address (A) and the last address plus 1 (B). In some cases a conversion and/or packing/unpacking operation is required by the recording mode.

For input, R.M.=1 (BCD 8/wd)

Keyboard input with no interrupt on carriage return is selected. Characters are packed eight per word until the area from A to B-1 is filled or carriage return is sensed. If carriage return is sensed fill out the last word with blanks if necessary. Interrupt on carriage return is selected and an exit made with the number of words read in the A register.

R.M =2 (Typ 8/wd)

This mode is the same as R.M.=1, except the BCD code from the display must be converted to typewriter characters.

R.M.=3 (Typ 1/wd)

The characters are read, converted to typewriter code, and stored one per word in area A to B-1. Interrupt on carriage return remains selected.

For output, R.M.=1 (BCD 8/wd)

The information from A to B-1 is packed into the proper format for the display and output. If the message overflows a three line buffer (186 characters) the overflow is lost.

R.M.=2 (Typ 8/wd)

The information from A to B-1 is converted to BCD and packed into the display format and output.

R.M.=3 (Typ 1/wd)

Same as R.M.=2 above.

F.C.=2 Check only

Control is returned with the number of words transmitted by the last read (R=1) or write (R=0) operation. This function if needed to satisfy MCS which assumes some of the transmissions are buffered and therefore it must return to the driver a second time to determine the number of words "buffered".

F.C.=3 Read/Write with checking

Same as the operation for F.C.=1, with the number of words processed entered in the A register before returning.

F.C.=4 Sense Equipment Ready

Determine if the display is in a "ready" status, if it is not ready set E to 1 and use the alternate return. If it is ready set E to indicate the current status for the "case" and use the normal return.

The present COOP MONITOR does not use any of the provisions incorporated for F.C. greater than 4. The driver routine need only check that the function code is within the range 1 to 4. If not set E=0 (parameter error) and use the alternate return.

B. Coop Monitor Interrupt Package

Before any great amount of programming can be done to adapt COOP MONITOR for remote operation, the present interrupt package (Select, Detect, and Remove) must be modified. This subroutine can presently select and sense interrupts on channels 1 thru 6 inactive and all internal interrupts. It cannot select or sense satellite generated interrupts on channels 3/4 or 5/6 via the CDC 1607 magnetic tape units, nor can it select or sense any channel 7 interrupt.

The calling sequence for the interrupt subroutine, see ref. 10, uses the integers 1 thru 12 to set the type of interrupt to be selected and sensed. The integers 1 thru 8 are used for those interrupts to be removed. These calling parameters need to be increased to provide the following additional options.

For Select:

Type Interrupt	Action
13	Select program control for channel 3/4
14	Select program control for channel 5/6
15	Interrupt on keyboard 1 hit (DD65)
16	Interrupt on keyboard 2 hit (DD65)
17	Interrupt on radar target hit (DD65)

For Remove:

To avoid confusion use the same parameters to remove the above selections

Before any of these changes are attempted the precautions necessary for programming an interrupt package, discussed in ref. 13, must be carefully considered.

Along with these modifications to the interrupt subroutine, the main frame of the computer should be modified to allow

computer selection of interrupt lockout. This would avoid the present procedure of selecting interrupt on arithmetic fault and then forcing a fault just to place the computer in lockout mode. There is just such a routine in MCS (RLOKOUT) to perform this function. This takes 20 instructions to accomplish what could be done with one. These modifications would be similar to those described in section 4.4 to provide computer selection of remote indicator lights.

C. Data Display Graph Plot Routine

The incremental graph plotting system has been operational for approximately six months, see section 4.1. A routine to provide this graph plot capability utilizing the Data Display unit (DD65), still needs to be completed. The following specifications and suggestions should provide an outline for incorporating this "Graph Display".

1. Use the author's scaling routine to scale to units per inch.
2. Use this scale and 16 increments per inch vice the 100 per inch used in the incremental system to build a scaled integer array of coordinate points.
3. Output the axis, titling, and labeling data using the standard "Grafplot" technique.
4. Curve drawing requires extensive use of the string capability of the display in order to provide for multiple curve graphs. This is due to the relatively small, 1024 cell 24 bit, memory of the data display.
5. Memory allocation in the DD65 Display should be about as follows:
 - a. Vertical and Horizontal Axis, 8 inches long. Use vector size 64/line. Requires 40 cells.

- b. Labels, 4 DD65 words/label. 18 labels requires 72 cells.
- c. Titles, A minimum of about 200 cells.
- d. Curves, Total memory usage of above is 312 cells. This leaves 714 cells for curves. Using the string vector mode this will provide storage for approximately 2500 vectors.

6. Using only the smallest vectors this will provide for approximately 20 traverses of the display tube. Assuming that an average curve traverses the tube 4 times, it appears that 5 curves may be displayed simultaneously.

Besides an on-line graph display routine, another routine which is very desirable would provide an off-line display of the graphs which were output by the CDC 1604 program for "hard copy" incremental plotting.

4.0 SYSTEM ROUTINES

Several system routines were written for both the Fortran 60 System and the Coop Monitor System. Four of these routines, which were developed, shall now be discussed.

4.1 Grafplot Routines

The 1604 Grafplot Routine, see enclosure 3, was written for the CDC 1604 computer. It generates a magnetic tape output in the format required to drive the 160A Grafplot Routine, see enclosure 2. A version of the 160A routine was written for the 160 computer, see enclosure 1. It bypasses the interpolation option in order to operate in the smaller memory of the 160 computer.

This system will produce multiple curved graphs, each curve is limited to a maximum of 900 points. Optional auto-scaling of data points, titling and curve labeling is available. Figure 7 is a general flow diagram of the 1604 Grafplot routine. Copies of the coding for these routines are available from the CDC DATA CENTER, PALO ALTO, CALIFORNIA or from the U.S.N. Postgraduate School Computing Center.

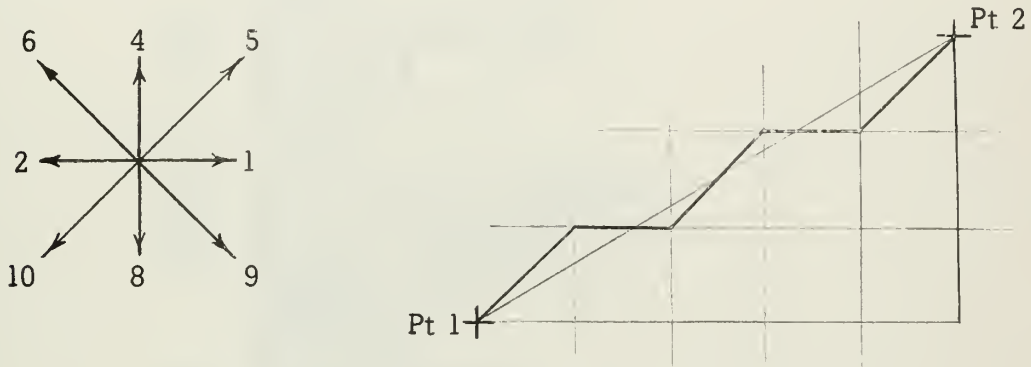
Two original techniques were developed which are of interest. They are the quantized scaling iterative procedure used in the 1604 program and the "decision rule" algorithm for providing "best fit" motion commands to an incremental plotter.

The scaling technique assumes a scale, tests for fit, then iterates to the next quantized scale and test again until the optimum available scale is found.

The two point connector algorithm, "LINE", computes the code string required to move the pen of a digital incremental X,Y plotter from an initial point (X1, Y1) to a terminal point (X2, Y2) by the "best fit" approximation to the straight line between the

points. The permitted elemental pen movement is to an adjacent point in a plane Cartesian point lattice, diagonal moves are permitted. The approximation is "best fit" in the sense that the deviation of the generated line from the true straight line never exceeds one half an increment. This is physically the best fit possible within the movement constraints imposed.

As an example of the LINE program, assume it is required to increment a pen's position with minimum deviation, along a straight line connecting point 1 and point 2. The movement codes for the CalComp Plotter are as follows;



Executing the LINE program, the following values are found;

A = 5
 B = 3
 NRLONG = 5
 NRSHORT = 3
 MOSTLY = 1
 IBOTH = 5

and the iteration scheme produces the incremental path shown.

Note, in the following fortran program, LINE, a subroutine is required to output the movement codes from the user's particular computer to the incremental plotter. Thus subroutine "PLOT (MOVE)" would output the movement code "MOVE" to the plotter. For the CalComp Plotter the NCODE instructions may be deleted, since the routine is written to require no code conversion for this equipment.


```

PROGRAM LINE ( X1, Y1, X2, Y2 )
DIMENSION NCODE ( 10 )
NCODE ( 1 ) = "user's +X movement code"
NCODE ( 2 ) = "    -X    "
NCODE ( 4 ) = "    +Y    "
NCODE ( 5 ) = "    +X+Y  "
NCODE ( 6 ) = "    -X+Y  "
NCODE ( 8 ) = "    -Y    "
NCODE ( 9 ) = "    +X-Y  "
NCODE (10) = "    -X-Y  "
A = X2 - X1
B = Y2 - Y1
IF ( A ) 200,100,100
100  ICODE = 1
    Go To 300
200  A = - A
    ICODE = 2
300  IF ( B ) 500,400,400
400  JCODE = 4
    Go To 600
500  B = - B
    JCODE = 8
600  IF ( A + B ) 1500,1500,700
700  IF ( A - B ) 800, 900,900
800  MOSTLY = JCODE
    NRLONG = B
    NRSHORT = A
    Go To 1000
900  MOSTLY = ICODE
    NRLONG = A
    NRSHORT = B
1000 IBOTH = ICODE + JCODE
    LIMIT = NRLONG
    NSWITCH = NRSHORT
    MOSTLY = NCODE ( MOSTLY )
    IBOTH = NCODE ( IBOTH )
1100 IF ( NRLONG - NSWITCH - NSWITCH ) 1200,1400,1400
1200 MOVE = IBOTH
    NSWITCH = NSWITCH - NRLONG + NRSHORT
1300 CALL PLOT ( MOVE )
    LIMIT = LIMIT - 1
    IF ( LIMIT ) 1500,1500,1100
1400 MOVE = MOSTLY
    NSWITCH = NSWITCH + NRSHORT
    Go To 1300
1500 END
    END

```

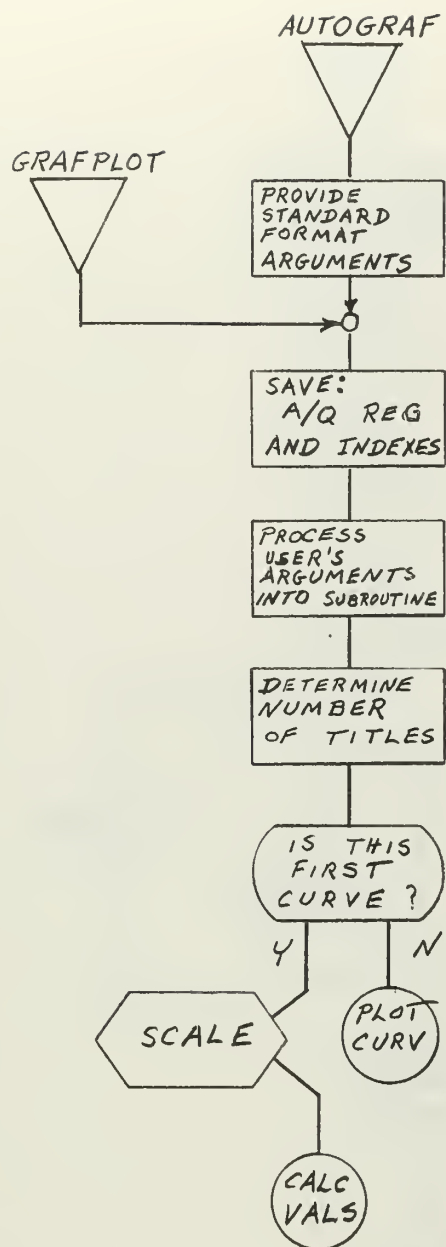


Figure 7a

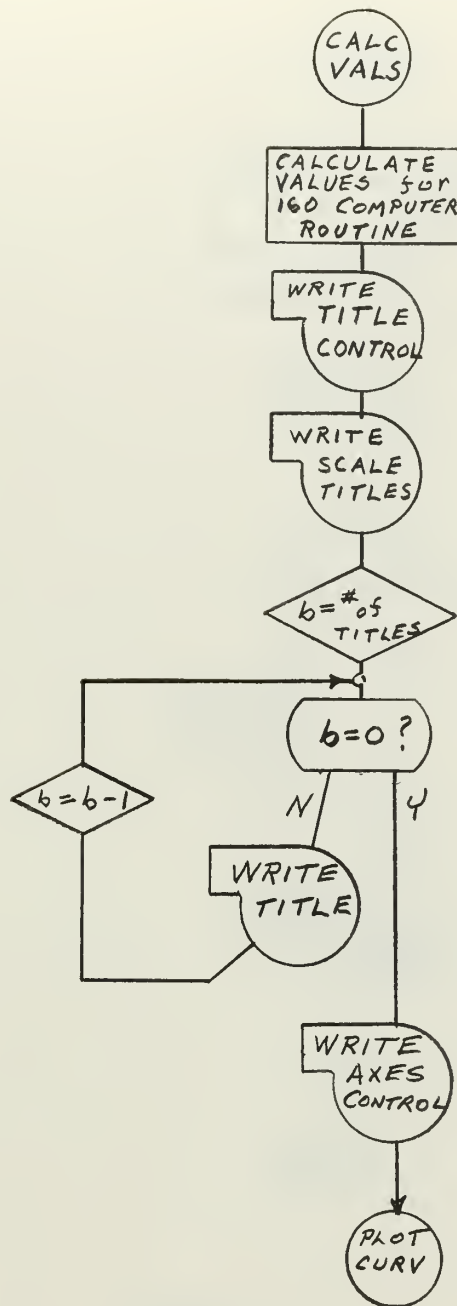


Figure 7b

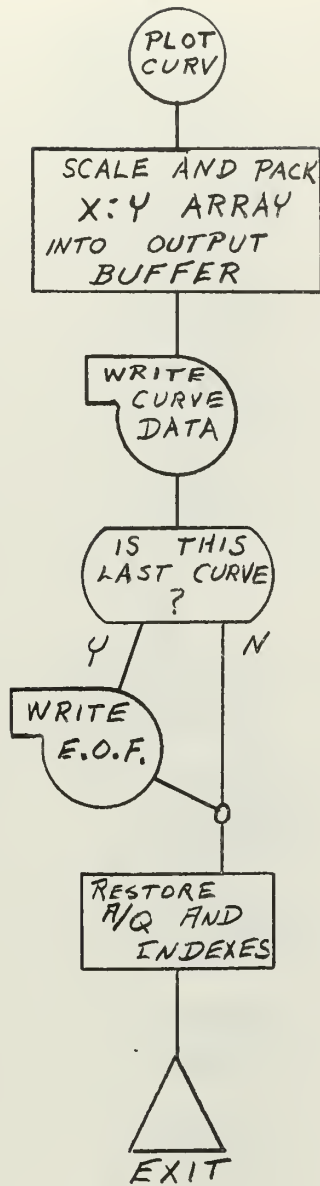


Figure 7c

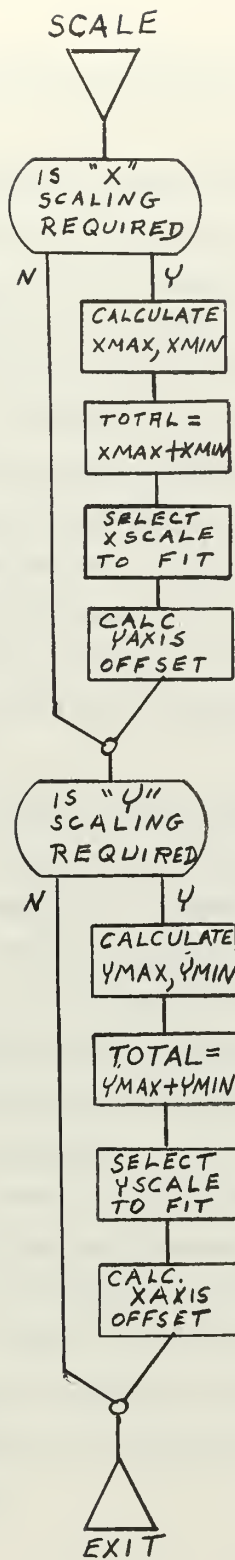


Figure 7d

4.2 Data Display "Line Printer Simulator"

The "line printer simulator" program demonstrates a very effective method of providing a continuous page listing ability. The problem was to display one full page of data on the Data Display Incorporated, Model 65, console. The data displayed was to be updated one line at a time by over-writing an old line of data in the DD65's memory. It was desired that this new line entry always occur at the bottom of the page and all other lines would precess up the page and finally disappear. An effective method to provide this listing ability was programmed and added to the Fortran 60 System. An understanding of this program requires a familiarity with the word format used to update the DD65. The DD65 has the following characteristics;

a. It has two keyboards which can be used only as output mediums to a computer. There is no connection or internal logic for providing tube display of a character when it's corresponding key is hit. This is different from a conventional typewriter, which prints it's character automatically when a key is hit. In order to display the character, it must be read by a computer, processed and the appropriate word format must be transmitted back to the display. The display logic then decodes the format of the words transmitted to its memory and displays the appropriate character/characters. There is no provision for read out from the DD65 memory by a computer. The memory serves only to store the data for display upon the cathode ray tubes. The logic cabinet of the display unit contains the circuitry required to scan its memory and display the contents onto the cathode ray tubes.

b. To read from a display keyboard, requires that a sequence of external function codes (see fig. # 9) must be executed to sense keyboard hit, select keyboard for input to the computer, then input one word via channel 7. The BCD code for the key read-in will be the lowest 6 bits for keyboard 1 characters and the lowest 9 bits if keyboard 2 was read. The process of reading the key into the computer releases the key at the DD65 console.

c. Output to the display maybe either characters or vectors. There are three sizes of characters, 32, 64, or 128 per line. There are two sizes of vectors, 64 or 128 per line. The line mentioned is the width of the tube display area, approximately 8.5 inches square.

d. A selection of either character or vector mode, a memory storage address, the X / Y display position, and a tube selection are all data which is conveyed to the display. The word which contains this information is called a designator word, see figure # 8. This designator word can also carry two characters or vectors, depending on the mode selected. Additional characters or vectors can be transmitted after the designator word, if packed in a format called "string word format", see figure # 8. These "string" word characters or vectors will be positioned relative to their preceding designator word. Thus it is the designator word that controls the tube location where the character/vector string will appear. For additional display information, see reference #19.

The line printer routine provides for displaying 32 lines of data. This requires that 32 designator words, each followed by a line of data, be stored in the DD65 memory. Once the data was in the DD65 memory, it was only necessary to selectively transmit new designator words with the appropriate new Y position indicators to move the whole line's position. Thus once the line data has been output, the 1604 is not required to retain an image of the page data in its memory, nor need any time be wasted in re-transmitting redundant information to the display.

The technique employed required that a 32 word designator table be constructed. This table provided memory addresses for data storage from cells 600 thru cell 1777 in the DD65 memory. Note, the DD65 memory cells are only 24 bits long, the dis-assembly from the 1604 computer's 48 bit word format is automatic in the DD65 logic. Note also that equipment limitations dis-allow addressing odd numbered memory locations in the DD65. The designator table also provides 32 discrete, equally spaced vertical "Y" axis locations to provide separation between the lines to be displayed.

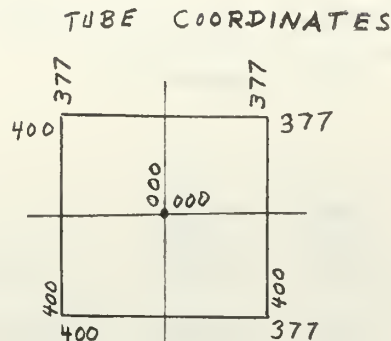
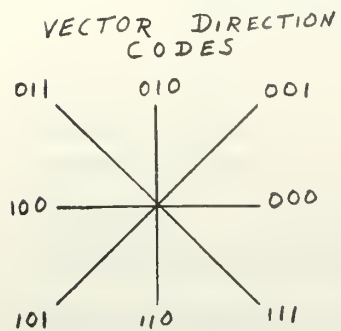
The method of up-dating the page displayed requires that all the "Y" locators in the designator words be incremented vertically one line space (20 units). A mask search is then executed to locate the designator word which contains the "Y" locator for the bottom line entry to the page. This "Y" locator is 400. Once found, the designator word is stored in the output buffer. Then the designator table is transmitted to the DD65, thus sweeping all lines vertically one space. The new line is packed into the output buffer. This buffer now contains a

designator word followed by a line of string words. These are now output via channel 7 to the display's memory. Note the channel 7 input / output instructions transmit words, from the 1604 memory in reverse sequence compared to the normal buffer channels. This requires an inverse packing order when packing the buffer for output on channel 7.

This line printer technique has also been incorporated into a typewriter page simulator for the right tube of the display. Both systems are working well.

See appendix IV for the resident coding required for the line printer package. It required only 77 cells including the designator table and a 10 cell output buffer. The designator table is listed in the appendix IV, page # 6.

CHANNEL 7 OUTPUT FORMAT for DD65 DATA DISPLAY



LFT CHR = 00
LFT VECT = 20
RT CHR = 40
RT VECT = 60

BUFF + B2 Y	[]	()	\$	<	0	+ B3 X	+
	=	≠	↑	→	{	}	0		*
	4	5	6	7	8	9	0		≥
	X	Y	Z	O	L	2	0		3
	Q	R	S	T	U	V	0		W
	J	K	L	M	N	Ø	0		P
	C	D	E	F	G	H	0		I
SIZE	INTENS	← X → POSIT		A	B	INCR.	10 BIT MEMORY ADDRESS	TUBE MODE	← Y → POSIT

0 = FIXED BRIGHT
1 = NORMAL

0 ⇒ INCREMENT CHARACTERS TO THE RIGHT
1 ⇒ DOWNWARD

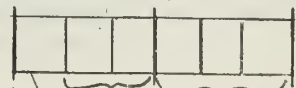
0 = CHAR.
1 = VECTOR
0 = LEFT
1 = RIGHT

SIZE CODES
00 ⇒ 128
01 ⇒ 64
10 ⇒ 32
} PER LINE

6 BIT BCD CHARACTER or VECTOR CODES

B3 →

VECTOR CODE FORMAT



SIZE 3 BIT DIRECTION CODE

0 ⇒ BLANK
1 ⇒ UNBLANK

Figure 8

10 December 1962

SELECT CODES

=====

FUNCTION =====	Select 1604 =====	Code 160 =====	Computer Input Word Bit Format =====
Select Keyboard 1 for Input	¹ C7140	7140	000000FFFFFF
Keyboard 2 for Input	C7120	7120	000VVVFFFFFF
Select Track Ball "'X'"	C7102	7102	000XXXXXXXXX
Track Ball "'Y'"	C7104	7104	000YYYYYYYYY
Select Range Switch	C7110	7110	00000000RRR
Select 160 Status Response	² (C7040)	7040	
Select Memory Update;			
from 1604	C7010	(7010)	
from 160	(C7020)	7020	
Select Radar Target Data;			
to 1604	C7002	(7002)	
to 160	(C7004)	7004	
to Auxiliary Equip.	C7001	7001	
Select Interrupt on;			
Keyboard 1 Key Hit	C7105	(7105)	
Keyboard 2 Key Hit	C7103	(7103)	
Radar Pulse Hit	C7141	(7141)	
Release Interrupt Request	C7111	(7111)	
Remove All Interrupt Selects	C7121	(7121)	

Notes 1. "'C'" Indicates 1604 data channel number. (Normally, C=7)
 ===== 2. () Indicates code disabled selectively on "'160 or 1604
 ONLY'" mode switch position.

Figure 9a

SENSE CODES

=====

CONDITION

=====

1604 Full Exit Sense Code

=====

160 Status Response

=====

Keyboard 1 Hit	C7172	xxx2
Keyboard 2 Hit	C7175	xxx1
Keyboard 1 NOT Hit	C7173	
Keyboard 2 NOT Hit	C7174	
Tab Hit		xxlx
Tab NOT Hit	C7157	
Carriage Return Hit		xxx4
Carriage Return NOT Hit	C7166	
Keyboard 1 Selected		xx4x
Keyboard 2 Selected		4xxx
Keyboard 1 NOT Selected	C7167	
Keyboard 2 NOT Selected	C7037	
NOT DD-65 Interrupt	C7156	
DD-65 from 160 Selected	C7020	2xxx
DD-65 from 1604 Selected	C7010	1xxx
Radar to 160 Selected	C7004	x4xx
Radar to 1604 Selected	C7002	xx2x
Radar to Auxiliary Equipment		
NOT Selected	C7001	x1xx
Radar Target Present	C7136	xx2x
Radar Target NOT Present	C7137	

.....

Figure 9b

4.3 Merg Sort

The SORT Routine, see appendix III, was written to improve the efficiency and reliability of sorting using the 1604 computer vice the IBM mechanical card sorter. Another motivation was to provide the capability of sorting 120 character records which could not be easily sorted using 80 character cards.

The SORT Routine can sort on variable length column fields from 1 to 120 columns. The present routine divides the field to be sorted into 7 character fields, then multiple entries are made to a subroutine, SORT7, which actually performs the sorting operation. SORT7 divides the computer's memory into three equal bins, ABIN, BBIN, and CBIN. These are each 20,000 octal cells long. SORT7 requires an input tape written in 120 character records with an "End of File Mark" terminating the file to be sorted. Upon entry to SORT7, one of its subroutines, Alsinp, reads 1000 octal cards into ABIN. As each card is read, a sorting key is formed from the column field designated and this key is stored in the cell preceeding the input record image from which it was generated. The key occupies one cell, the record image occupies 17 octal cells, thus each input record requires 20 octal memory cells. Multiply 20 octal by 1000 octal to obtain the arbitrary 20,000 octal bin size which was chosen as the basic block size. Alsinp always checks for an E.O.F. mark and sets a flag to signal the rest of the routine that the last block has been read. After Alsinp has read 1000 octal records or encountered an E.O.F. mark, then an entry is made to a subroutine, Rlhmerg, which performs an internal merg sort by oscillating between ABIN and BBIN, sorting on the keys. The internal merg sort terminates with the sorted block in CBIN. If this was the first block, it is immediately written onto one of the two scratch tapes and Alsinp inputs another block. If this was not the first block, then a subroutine, Hexsort, is entered to perform a merg of the new block

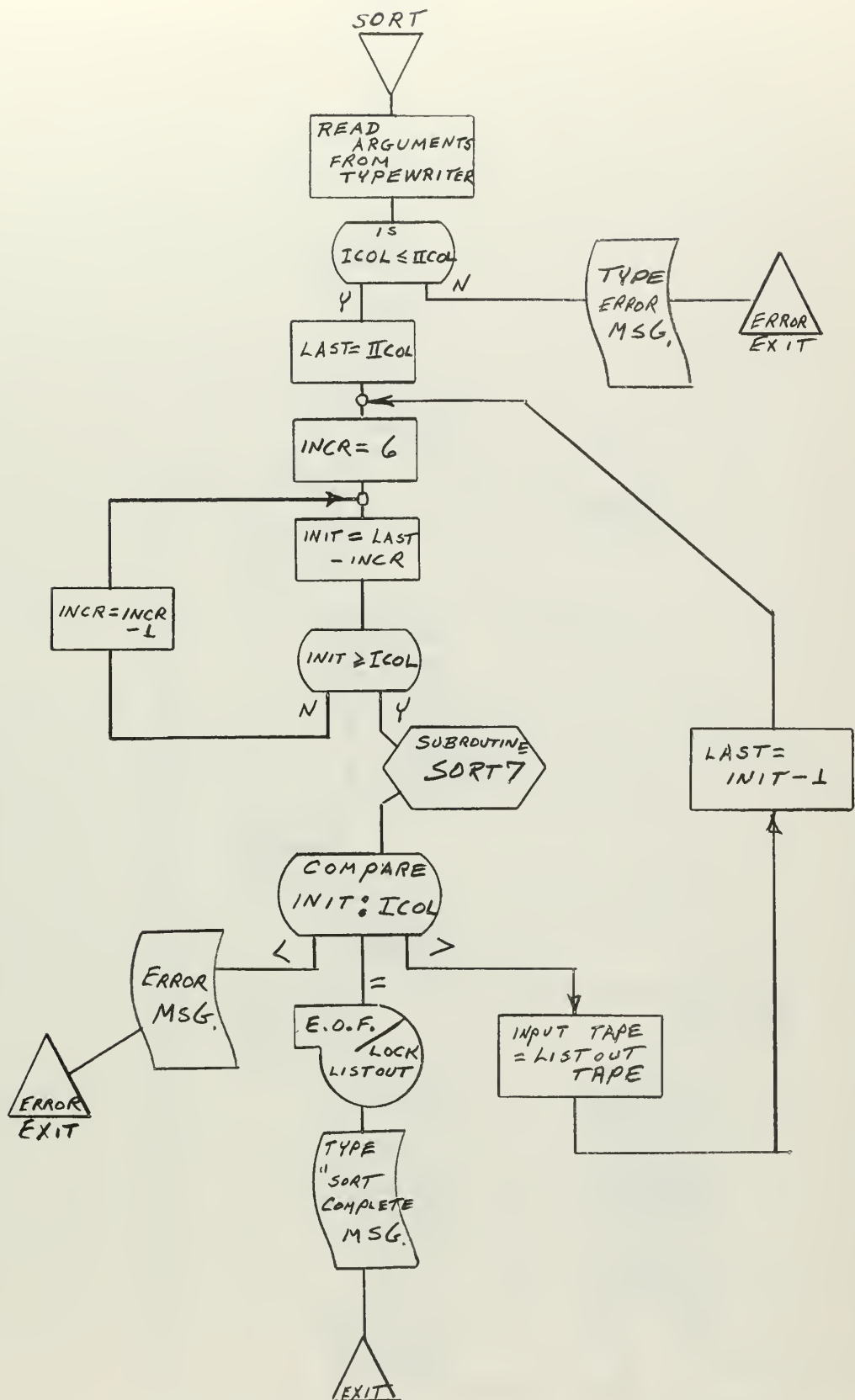


Figure 10a

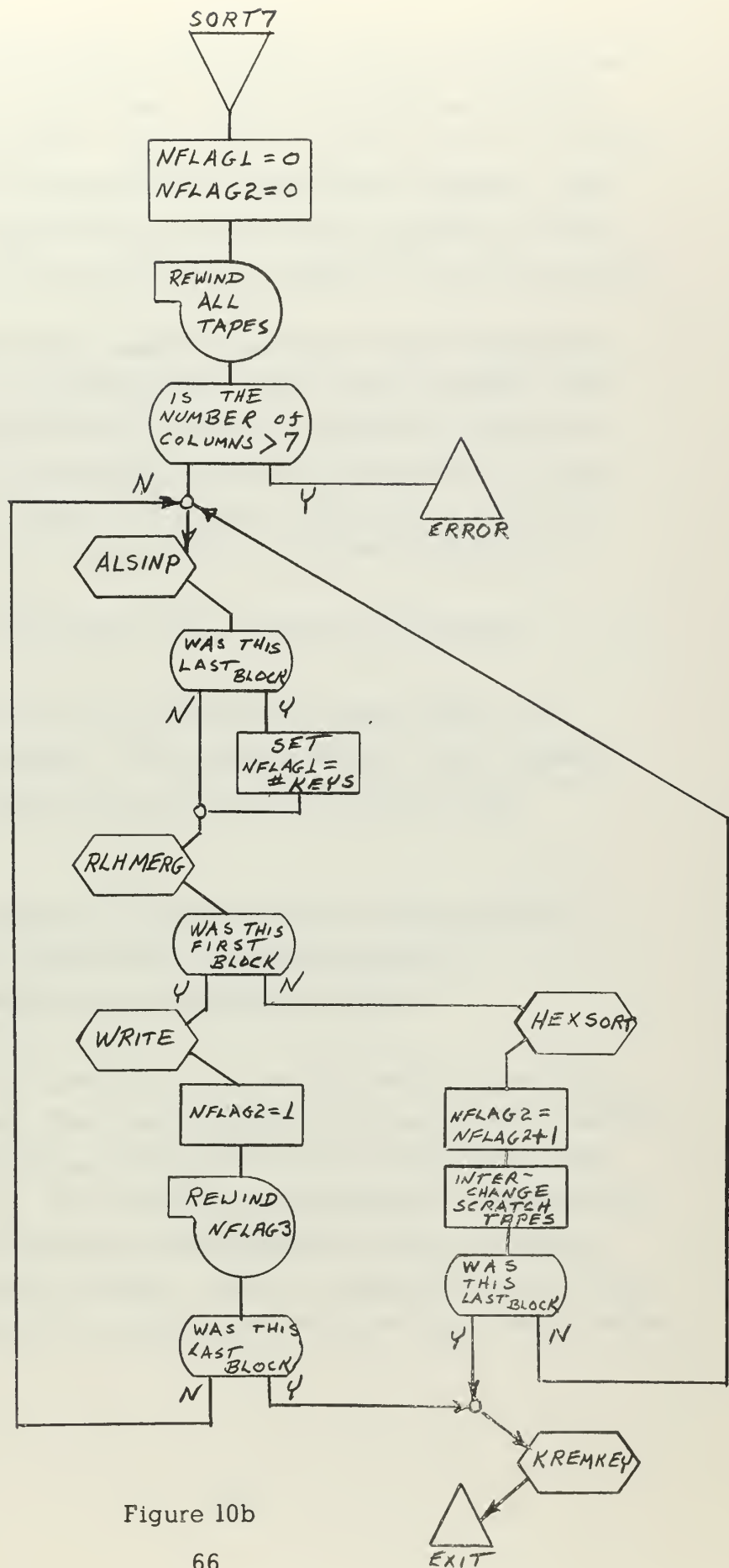


Figure 10b

into the file of previously sorted blocks which are accumulated externally on a scratch tape. This cycle continues until an E.O.F. is encountered, then after the external merge is complete, the scratch tape is rewound and a subroutine, Kremkey, reads in the ordered blocks from the scratch tape, removes the keys and writes out the output file in an ordered, 120 character record length file.

If additional fields remain to be sorted, this output file is designated the input and another entry is made to SORT7 with the new character field's initial and last column numbers as arguments. This procedure continues until the total sort field is in order and listed on the output. The output is then rewound to lockout and the operator is informed that the job is finished.

Several improvements to this routine are suggested.

- a. Build and sort on a two cell key, if large fields are to be sorted frequently. This would provide a better balance between the internal sorting time and the external tape handling time.
- b. Provide for sorting on split fields, ie. non-consecutive column fields. This could be provided by forming the sort key piece-wise from specific columns.
- c. Use a double entry key / key location table for performing the internal sort. This would eliminate the necessity of carrying the input record image around with the key each time the key order is changed. The sort could be performed by ordering only the keys and maintaining correspondence in the key location table as the keys are sorted. This key location table could then be used to perform the internal ordering of the input record images.

This scheme would improve the internal sorting speed and reliability since it would require fewer internal data transfers.

d. The last and probably the most significant improvement which is indicated, would be to utilize two more scratch tapes. This would improve the sort routine speed by nearly 50%. This is due to the rewind time required by the tape drives and their inability to read in the reverse direction. At present, much time is wasted waiting on the two scratch tapes to rewind in order that the next internal block can be merged with the external ordered file. A suggested method to improve the external sort would be to alternate the merging of the internal block between two sets of scratch tapes similar to the set presently employed. This would increase the rewind time available for each set and thus decrease the idle time mentioned previously. After Hexsort has finished merging the last block, then Kremkey would have to operate differently than at present. Kremkey would have to read one block from the ordered tape of each scratch tape set, then perform a final merge comparison similar to the Hexsort merge routine. Instead of forming an ordered list in memory, this final merge could be written out as the final ordered, 120 character listable output. Before any other modification is considered, this last suggestion should be given priority "one". At present it remains a gross deficiency against high speed sorting using this routine.

4.4 Satellite System Programming

There are two basic operations necessary to establish an adequate satellite system. These are the ability to exchange data and to selectively execute a pre-stored program when directed by another member of the satellite system. The present installation proposed to couple the CDC 160 to the CDC 1604 Computer. The CDC 160 Computer does not provide an interrupt mode, therefore only the 1604 interrupt mode was available to be employed.

The desirable characteristics of the basic 1604 satellite interrupt program are:

- a. Reliability
- b. Flexibility
- c. Speed
- d. Effective Employment of the Communication Flags
- e. Minimize Memory Requirements

The problems encountered in designing the basic 1604 interrupt routine shall now be discussed. The 1604 Computer is unable to prevent the 160 Computer from entering the interrupt auxiliary routine which is wired into the 1604. For this reason, a protective "lock-out" scheme was required to prevent interference with the 1604 equipment handling routines. Specifically, there exists a critical time interval between equipment selection and activation during which an interrupt from a satellite station would be disastrous to the main program. The technique developed to prevent this, is as follows: All 1604 equipment handling routines were modified to turn on communication flag I (see ref. # 2) in advance of any equipment selection. Communication flag I is then cleared after the selected equipment is activated. Using this scheme it is

then required that the 160 satellite computer inspect communication flag I prior to executing an interrupt. If flag I is set, then the 160 computer enters a "wait loop" until flag I is cleared by the 1604 Computer. Thus through the use of communication flag I, the 1604 computer is, in effect, capable of "locking out" the 160 satellite computer from an interrupt request. This was a desirable feature not presently provided in the hardware design.

The next problem in the design of the interrupt routine was how to indicate to the 1604 computer, the nature of the task requested by the 160 computer. The method used requires that each interrupt shall immediately trigger a 48 bit word, data transfer to the 1604. The format of this word, designated as the "control word", is outlined in detail in figure # 12. After receiving this "control word" the 1604 interprets the operation to be performed. There are four tasks which can be directed by the control word, they are:

1. 1604 INPUT DATA BLOCK
2. 1604 OUTPUT DATA BLOCK
3. PACK ARGUMENT ADDRESSES INTO A SUBROUTINE AND THEN EXECUTE THAT SUBROUTINE.
4. LOAD "A" REGISTER WITH A 21 BIT ARGUMENT FROM THE CONTROL WORD AND THEN RETURN JUMP TO EXECUTE THE SUBROUTINE.

A flow diagram of this 1604 satellite interrupt routine is provided in figure # 11.

One additional problem remains to be discussed, that is storage allocation for the programming required by the satellite system. This was acquired by setting the resident bias level on the Fortran 60 System to 5000 octal, vice the normal 4000.

1604 INTERRUPT ROUTINE

FLOW DIAGRAM

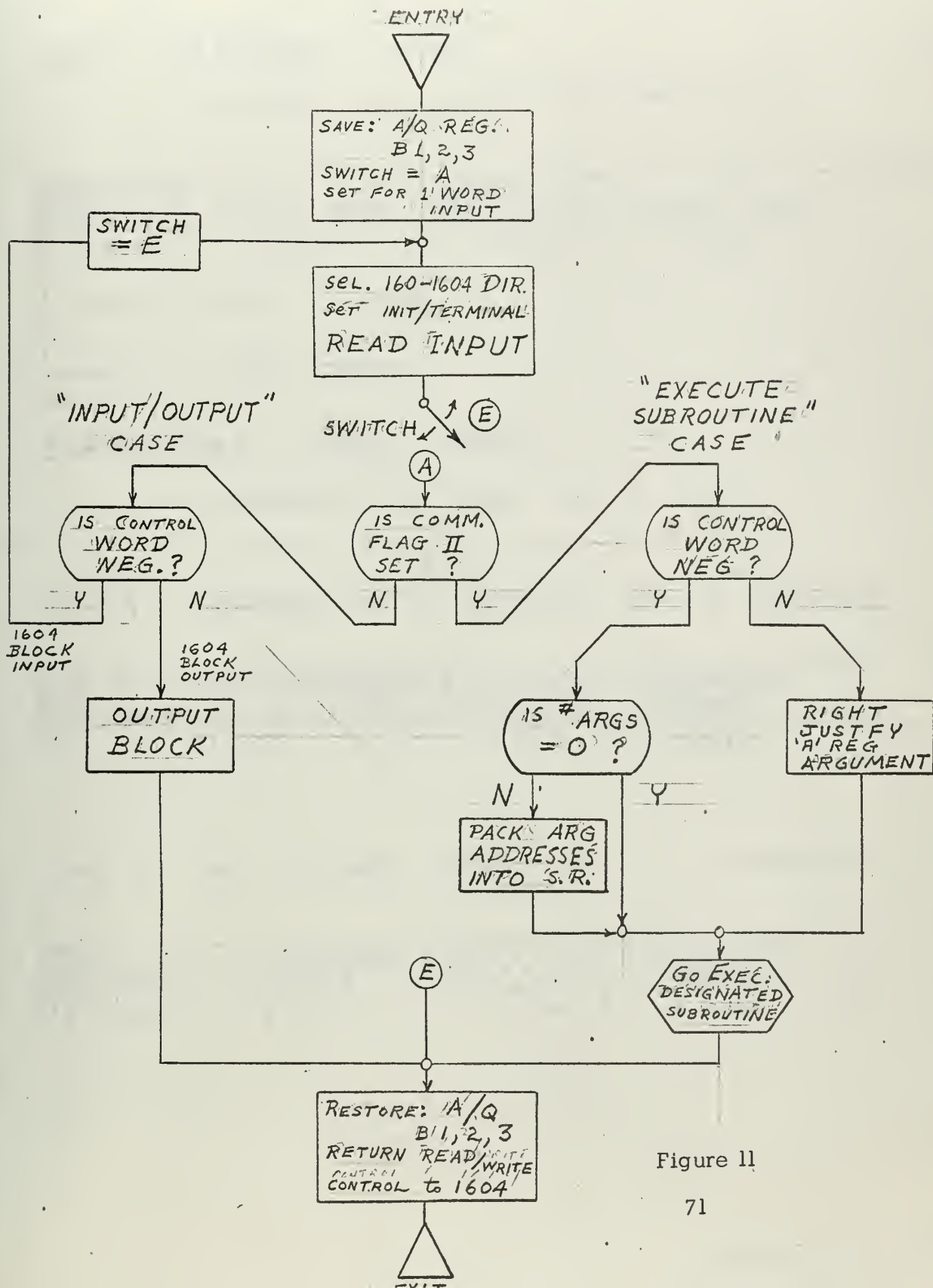


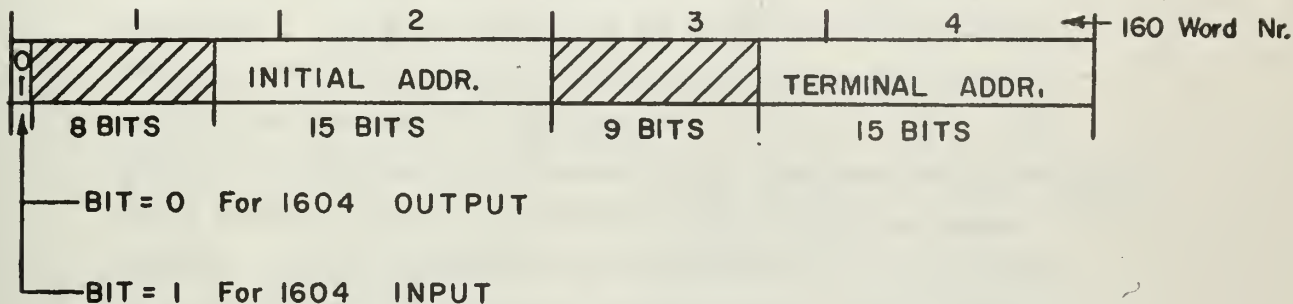
Figure 11

160-1604 SATELLITE

CONTROL WORD FORMAT

DATA TRANSFER MODE

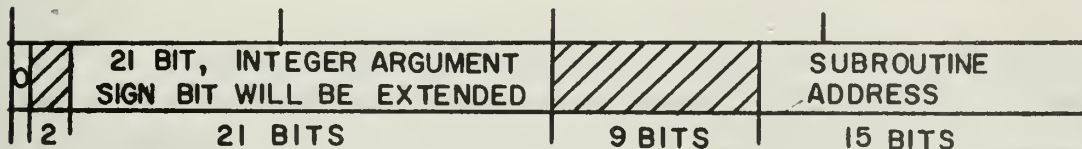
INTERRUPT WITH COMM FLAG II "NOT" SET



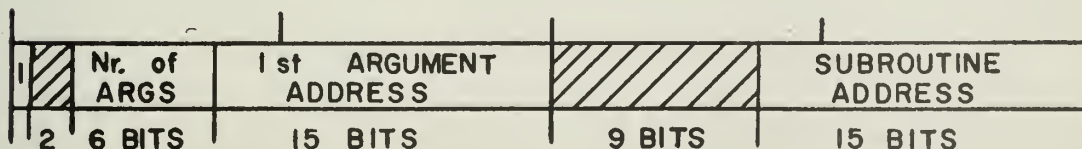
SUBROUTINE JUMP MODE

INTERRUPT WITH COMM FLAG II "SET"

TYPE I, ARGUMENT IN "A" REGISTER, RTJ To SUBROUTINE



TYPE II, PACK ARGUMENT ADDRESS, RTJ To SUBROUTINE



This provided the memory space needed to store the satellite programming plus an additional "buffer" area for receiving transient subroutines from the 160 computer. These subroutines may be executed by an interrupt entry to the 1604 computer via the satellite interrupt system.

The use of the 160 as a satellite station requires that the modified library tape, FORTSAT, be in use by the 1604 computer. Also, the 1604 must be operating under monitor control in order to utilize a remote monitor process routine which was developed by the authors' (see ref. # 22). The physical separation of the satellite 160 computer station required some means of sensing the above conditions at the remote station. Visual indicators were selected as the best method of implementing the required status sensors. The console of the 160 Computer contains many unused indicator lights, two of these were selected to be used as status indicators. The first light, designated # 1 or FORTSAT, is a blue-green background field light at the left end of the "P" register. The second light, designated # 2 or MONITOR, is the next blue-green background field light to the right of the # 1 light.

Manual control of these lights at the console of the 1604, was considered an unnecessary burden to the operator. Computer selection would be more convenient and the status indications more current in the reflection of the actual status of the 1604 Computer operations.

To enable computer control of the indicator lights, two new select codes (on - off pairs) for the 1604 were required. As all satellite selects are made in the 1607 Magentic Tape Logic Cabinets, modification of these units was necessary. This modification would allow the 1607 to accept and decode the new indicator light codes. The two codes selected were;

LIGHT # 1	(FORTSAT)	"ON"	53511
		"OFF"	63511
LIGHT # 2	(MONITOR)	"ON"	53544
		"OFF"	63544

The select code format is as follows:

First digit;	Channel number
Second digit;	Cabinet number
Third digit;	Group code, 5 for satellite
Fourth & Fifth;	Condition code

The two codes selected for the indicator lights contain the false cabinet number 3. This was necessary because the 1607 (cabinet # 2) does not uniquely decode the last two digits of any select code. Therefore, to prevent interference with any of the existing codes, a false 3 was used.

To avoid the need for manual selection of "program control" mode of operation for the 1607 Tape Cabinets, when enabling the satellite station, this selection was wired to be controlled by light # 1 and the master clear switch. Initial turn on of light # 1 selects "program control" and the Master Clear disables both lights and removes the program control selection. The logic diagrams for the modifications made to the 1607 Tape Cabinets are shown by figure # 13. Two relays were added to the 160 to provide switching of the two lights provided.

NOT "160 ONLY"

2 C 2 2 A - 4

A 9 9 9

1B22

2B37A

31 CARD
ADDED

A 9 9 3

1604 ONLY

PROVISION for EXF CODE CONTROL

of "PROGRAM CONTROL MODE"

CDC 1604 CABINET MODIFICATION

1C35-11
A 220

1B25-10

Q005

Figure 13b

BIBLIOGRAPHY

1. CDC MANUAL, Pub# 025, INPUT-OUTPUT SPECIFICATIONS
2. CDC MANUAL, Pub# 187, PROGRAMMING MANUAL FOR
CONTROL DATA SATELLITE COMPUTER SYSTEM
3. CDC MANUAL, Pub# 023, 160 COMPUTER PROGRAMMING
MANUAL
4. CDC MANUAL, Pub# 167, 1604 COMPUTER PROGRAMMING
MANUAL
5. CDC MANUAL, Pub# 032, 1604 MAINTENANCE MANUAL , Vol 2
6. CDC MANUAL, Pub# 037, 1607 DESCRIPTION AND
OPERATION, Vol 1
7. CDC MANUAL, None , MAGNETIC TAPE UNIT (1607)
CIRCUIT DIAGRAMS
8. CDC MANUAL, Pub# 069, 160 MAINTENANCE MANUAL, Vol 2
9. CDC MANUAL, Pub# 509, COOP MONITOR/OPERATOR'S GUIDE
10. CDC MANUAL, Pub# 508, COOP MONITOR/PROGRAMMERS
GUIDE
11. CDC MANUAL, Pub# 506, FORTRAN 62/REFERENCE MANUAL
12. CDC MANUAL, Pub# None, CODAP1/REFERENCE MANUAL
13. CDC MANUAL, Pub# None, SPECIFICATIONS FOR THE
COOP MONITOR SYSTEM
14. CDC BULLETIN, PSB-AS06621, GUIDE FOR CONVERTING
FORTRAN 60 PROGRAMS TO FORTRAN 62 PROGRAMS
15. CDC BULLETIN, PSB-AE01, COOP MONITOR USAGE
16. CDC BULLETIN, PSB-AE02, COOP MONITOR MANUALS
SUPPLEMENT
17. CDC BULLETIN, PSB-AE04, COOP MONITOR LIBEDIT ROUTINE
18. CDC BULLETIN, Pub# 516, COOP MONITOR/LIBRARY
SUBROUTINES

19. C.G. LAWSON, AN INTEGRATED DISPLAY AND CONTROL SYSTEM FOR MAN-MACHINE COMMUNICATION, U.S. NAVAL POSTGRADUATE SCHOOL M.S. THESIS, 1962
20. R.P. WARRICK, DESIGN OF AN AUTOMATIC MULTI-PROCESSING SYSTEM, U.S. NAVAL POSTGRADUATE SCHOOL M.S. THESIS, 1962
21. CDC MANUAL, Pub# 087A, FORTRAN 60 SYSTEM MANUAL
22. R.L. HOGG/D.C. GLOVER, PROGRAM CONTROL SYSTEM FOR A SATELLITE MODE COMPUTER COMPLEX, U.S. NAVAL POSTGRADUATE SCHOOL REPORT # TR-DCL-62-11/13, 21 DECEMBER 1962

APPENDICES

APPENDIX I

LIBEDIT PROCEDURE

There are two publications available to assist a user of COOP MONITOR in performing a successful library preparation or modification. These are ref.17 and 10. The following was obtained from these two sources together with additional remarks by the authors to aid in the understanding of the procedure.

A. Library Tape Format

The library tape is composed of two files. The first file contains three records in absolute binary format. These are:

- a. Bootstrap routine (BOOT)
- b. Recovery routine (EXTREC)
- c. Master Control routine (MCS)

The second file contains the subroutine library and starts with the "directory". This is followed by the library subroutines each consisting of one record (the system can accept subroutines with more than one record but the present library has none) composed of contiguous relocatable binary card images. Each subroutine is identified and defined in the directory by the following.

A subroutine name

Number of records in the subroutine (normally 1)

Number of common blocks assigned

Names of those common blocks

Length of each common block

Range of the subroutine (FWA and LWA+1)

Number of entry points to the subroutine

Entry point names

Number of external symbols referenced

External symbol names

The order in which the subroutines are identified in the directory must be the same as the order of appearance of the subroutines in the file. The identifying name, each entry point name, each external symbol and each common block name must have the following properties:

- a. Eight BCD characters in length, no imbedded blanks
- b. Leading character not digital or blank, except common block

The first library subroutine defined in the directory is the Master Control System called "Resident". The length of Resident and all named entry points are defined. External references are not made within the resident program. There are no entries in the directory for the subroutine EXTREC.

The recommended layout of the library subroutine file is:

1. Directory (this must be first)
2. I/O drivers for standard medium
3. All other I/O drivers
4. Secondary Control Systems (SCS)
5. Any other subroutines
6. SNAP (selective dump routine)
7. MAP (produces a memory map of loaded programs)
8. Systems such as FORTRAN, CODAP1, ALGOL, etc.

This layout facilitates one pass loading of library subroutines. The sequence for loading library subroutines will be:

1. I/O drivers for the standard mediums.
2. I/O drivers for the current job.
3. The desired secondary control system.

B. Directory Format

The directory is a set of 8 different types of cards for each subroutine in file 2 of the library tape. These must be ordered in the directory deck in the same sequence as the routines appear

in file 2. Care must be exercised to assure the directory reflects the exact composition of the subroutine file, otherwise errors will result. Each set of directory cards must contain:

1. A card with the subroutine name, left-justified, in columns 1-8.
2. A card with a 16 digit octal integer, left-justified, in columns 1-16. This has the format:

000KKKKK000RRRRR

where 000 is zero

KKKKK is the number of common blocks required by the subroutine

RRRRR is the number of records the subroutine occupies on the library tape (usually 1)

3. If KKKKK is not zero, there must be a number of cards equal to KKKKK containing the name of each common block. The name is punched, left-justified, in columns 1-8. The card may be blank if no name for the common block is given.
4. If KKKKK is not zero, there must be a number of cards required to define the lengths of the common blocks. These cards contain octal integers separated by commas. The integers specify the length of the common blocks named by the cards in #3 and the lengths must be in the same sequence as the names. Any number of integers, more than zero may appear on a single card. The last integer on the card must be followed by a blank. The number of cards required depends on the number of common blocks and the packing of lengths.
5. A card with a 16 digit octal integer, left-justified, in columns 1-16. This has the format:

PPPPFFFFXXLLLLL

where PPP is the number of entry points to the subroutine (must not be zero)

FFFFF is the relative first word address of the subroutine (FWA)

XXX is the number of external symbols referenced
by the subroutine

LLLLL is the relative last word address plus 1
of the subroutine (LWA+1)

6. The number of cards equal to PPP, each containing an entry point name, left-justified, in columns 1-8. There must be at least one such name.

7. The number of cards required to define the entry point addresses. These cards contain octal integers of 1 to 5 digits separated by commas. The integers specify the relative address of the entry points named on the cards in 6. The entry point addresses must be in the same sequence as their corresponding entry point names.

8. The number of cards equal to XXX in 5. If this is greater than zero, each card just contains the external symbol name, left-justified, in columns 1-8. These external symbol names must be in the same order as they were defined during the assembly or compilation of the subroutine.

The sets of directory cards are preceded by a card with "DIRECTRY" punched in columns 1-8 and followed by a card with "7777777777777777" punched in columns 1-16.

C. Edit Input

The input data for editing a library tape is composed of control cards, directory cards, and program information. The program information is in the form of relocatable binary cards and this binary format must be thoroughly understood by the user to perform a successful LIBEDIT. Appendix III of ref.10 gives a complete description of this binary card format.

The input data must be on the standard input unit along with the required MCS control information ("begin job" and "mcs" cards). The next card of the edit input deck must

LIBEDIT INPUT DECK

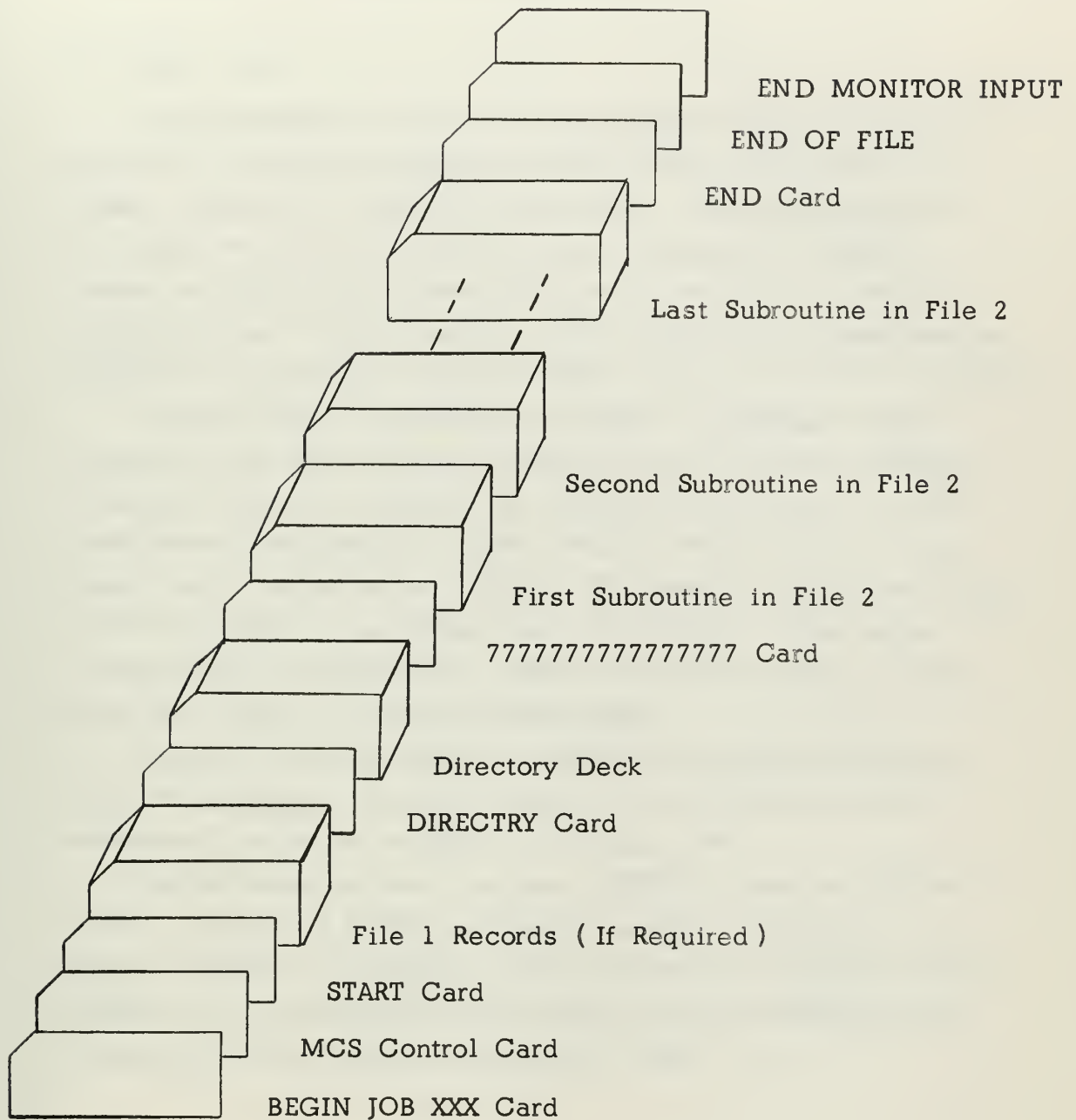


Figure A.2 Edit Input Deck

be a START card. This card contains the word "START" in columns 10-14 to produce a library tape of the same format as the old, see ref.17 for other formats possible. The succeeding cards in the input deck are determined by the type of edit and the file of the library affected.

D. File 1 Edit

The information in file 1 is in absolute binary format, and the file is divided into three records. Each record must be added, replaced or deleted as an entity. The information to be added or replaced must have been prepared by CODAP1 using the ORG pseudo instruction and it must be a complete, absolute binary format program. The complete deck produced by CODAP1 is included (only the "begin job" card need be removed).

The edit input is composed of decks of absolute binary records, each preceded by a control card. This card contains a decimal integer in column 10, followed by a blank, R, or D. The integer specifies the record position within file 1 of the binary card images which follow. The blank specifies the record is to be added, R to replace, and D to delete. The decks composing the records must appear in the same order in which they will appear on the new library tape.

If any of the entry points in MCS are altered by this edit then the complete directory (with the new MCS entry point addresses) must be included in the edit input.

The Extended Recovery routine (EXTREC) uses the AET and flags contained in BOOT as absolute address locations. If these locations are changed, then they must be changed within EXTREC (by recompiling) and including the new EXTREC deck, between BOOT and MCS, in the file 1 edit.

E. File 2 Edit

The information in file 2 is in several records in relocatable binary format. The first record is the directory. A new version of the directory need not be included in the edit input deck when the changes to file 2 are replacements which do not affect record lengths or entry points. In all other cases the complete directory must be included.

The first record of each subroutine (normally there is only one record) must be composed of relocatable binary card images produced by CODAP1. For edits involving subroutines of more than one record see ref.17. Each record must be preceded by a control card containing the subroutine name, left-justified, in columns 1-8, a blank in column 9, and a decimal integer starting in column 10 followed by a blank, R, or D. This integer specifies the position of this record within the records of this subroutine (usually 1). The blank specifies that the record is to be added; R specifies it is to replace an existing one, D specifies it is to be deleted. If a D appears, a binary deck for that record must not appear in the edit deck.

The edit deck, see figure A.2, must be terminated by a card with the word END in columns 10-12. To finish the input job an "END OF FILE" and a "END MONITOR INPUT" are necessary.

F. Operation

The library edit program is called from the library as the Secondary Control System by a MCS control card. The edit input is read from the standard input unit; a new library tape is written on the logical unit #13; operating information is

written on the standard output unit and the comment to operator unit.

For an edit operation the MCS control record (card) might appear:

⁷ELT, MGH, 34000, O/13, 5, 1000.
₉

During operation of the program, a listing is produced on the standard output unit. This listing contains each control card as it appears in the edit deck, all directory cards, and a message PREV REC X for each record transferred from the existing library to the new tape. The completion of the operation is indicated by the message EDIT COMPLETED. A sample edit listing is shown in figure A.1. Error diagnostics are written on the standard output unit and the comment to operator medium as errors are detected. These diagnostics are explained in table 1.

```

START MONITOR RUN.
BEGIN JOB 234
ELT,222,MGH,O/13,5,1000.
                                START
                                1R
PREV.                          2
                                3R
                                DIRECTRY
                                RESIDENT
                                0000000000000000
                                0430007700014557
                                READ*
                                WRITE*
                                SELECT*
                                .
                                .
                                .
                                1205
                                1212
                                655
                                .
                                .
                                .
                                TYPEI/O
                                0000000000000000
                                00100000000300277
                                TYPEI/O
                                0
                                CBF1*
                                CBF2*
                                SELECT*
                                CARDREAD
                                00000000000000001
                                00100000000200466
                                .
                                .
                                .
                                WRITE*
                                LOADER*
                                7777777777777777
PREV. TYPEI/O                  1
                                CARDREAD 1R
PREV. PTI                      1
PREV. PTLOG                    1
PREV. BFT                      1
                                .
                                .
                                .
PREV. MAP                      1
PREV. LIBEDIT                  1
PREV. COPY*                    1
EDIT COMPLETED

```

Figure A.1
Sample Edit Listing

TABLE 1 LIBRARY EDIT ERROR DIAGNOSTICS

CKSUM ERROR XXXXXXXX LTN-50

An input column binary card checksum detected to be in error. XXXXXXXX will be the record number if the error is detected during a file 1 edit; or it will be the name of the subroutine if the error is detected during a file 2 edit.

DIRECTRY ERROR

An error has occurred in the directory.

DIROFLO ERROR XXXXXXXX

One of the eight directory tables requires more than 1/8 of available core. XXXXXXXX will be the name of the routine.

EDITINP ERROR XXXXXXXX

The order of the attempted edit is incorrect or incompatible with the directory. XXXXXXXX will be the name of the routine.

END OF TAPE

An end of tape mark is encountered during writing of the new library tape.

EPT ERROR

Two routines in the directory have the same entry point names.

OFLCORE ERROR XXXXXXXX

Available core is not large enough to allow forming a record. XXXXXXXX will be the record number if the error occurs during a file 1 edit; or it will be the name of the routine if the error occurs during a file 2 edit.

RANGE ERROR XXXXXXXX

An inconsistency exists between the IDC and RBD cards of a file 2 binary deck. XXXXXXXX will be the name of the routine.

READING ERROR XXXXXXXX LTN-YY

A. A persistent (not cleared after three tries) parity error or illegal BCD character is encountered when reading the library tape or the standard input unit.

B. An illegal or impossible number is detected on an edit control card. XXXXXXXX will be the record number if the error is detected during a file 1 edit; or it will be the name of the routine if the error is detected during a file 2 edit.

YY will be the number of the logical unit number on which the error was detected. 00 for the library or 50 for the standard input unit.

WRITING ERROR XXXXXXXX LTN-13

A persistent (not cleared after three tries) parity error is encountered when writing the new library tape. XXXXXXXX will be the number of the record if the error is detected during a file 1 edit; or it will be the name of the routine if the error is detected during a file 2 edit.

RANGE

ENTRY POINTS

EXTERNAL SYMBOLS

IDENT

GRAF PLOT

- LWA+1
00000 02123
00022 GRAFPLOT
00003 AUTOGRAF
00001 FLOATF
00002 XTOI
00003 XINTF
00004 ERROR
00005 BUFOUT
00006 INGOUT
00007 GOUT
00010 ENDGOUT
00011 WREOF
00012 IFUNIT
00013 BKSP
00014 WRITE*
00015 ERROR*

GRAF PLOT - AUTOGRAF
FORTRAN COMPATIBLE SUBROUTINE FOR PLOTTING DATA
R.L. HOGG / D.C. GLOVER 1 APRIL 1963
J.S. NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

INPUT PARAMETERS FOR GRAFPLOT ENTRY

1. MODCURV = 0 FIRST AND LAST CURVE FOR THIS GRAPH
1 FIRST CURVE OF MANY
2 INTERMEDIATE CURVE
3 FINAL CURVE FOR THIS GRAPH
2. NUMPTS = NUMBER OF POINTS-MUST BE LESS OR
3. X = LOCATION OF X ARRAY
4. Y = LOCATION OF Y ARRAY
5. LABEL = 4 CHARACTER BCD LABEL FOR THIS CURVE
= -0 WILL CAUSE AUTOMATIC SEQ. UP TO 8
6. INTERPS = 0 NO INTERPOLATION BETWEEN INPUT POINTS
= 1 THRU 7-INTERPOLATE 1 TO 7 INTERMED
7. SFX = 0 PROGRAM WILL FIND BEST SCALE FACTOR
= A SCALE FACTOR FOR X, UNITS/INCH
8. SFY = 0 PROGRAM WILL FIND BEST SCALE FACTOR
= B SCALE FACTOR FOR Y, UNITS/INCH
9. IXOFFS = NO. OF INCHES TO OFFSET X AXIS

```

10. IVOFFS = NO. OF INCHES TO OFFSET Y AXIS
11. IHWIDE = NO. OF INCHES FOR X AXIS, MAX=9
12. IHIGH  = NO. OF INCHES FOR Y AXIS, MAX=24
13. MODE   = 0 AUTO CAL. BEST OFFSET OF AXES
              1 LOCATE AXES CENTER LEFT
              2 LOCATE AXES LOWER LEFT
              3 LOCATE AXES UPPER LEFT
              4 LOCATE AXES UPPER RIGHT
              5 LOCATE AXES LOWER RIGHT

```

```

14. TITLESIZ= 1-77      FIXED POINT MULT. FOR
                        LETTER SIZE. BASIC SIZE
                        IS .1 INCHES

```

```

15. TITLE1 = ADDRESS OF 1ST TITLE ARRAY
16. TITLE2 = ADDRESS OF 2ND TITLE ARRAY
              * 0 TO 2048 TITLES MAY BE USED
              *

```

INPUT PARAMETERS FOR AUTOGRAF ENTRY

```

1. MODCURV
2. NUMPTS
3. X Y
4. TITLE1
5. * *

```

0 TO 2048 TITLES MAY BE USED

```

ENTRY
EXT
EXT
EXT
EXT
EXT
EXT
EXT
EXT
EXT
EXT
EXT
EXT
EXT
EXT
EXT

```

```

00022+
00003+

```

```

GRAFPLT
AUTOGRAF
FLOATF
XTOI
XINTF
ERROR
RUFOUT
INGOUT
GOUT
ENDGOUT
WREOF
IFUNIT
BKSP
WRITE*
ERROR*

```

```

00001
00002
00003
00004
00005
00006
00007
00010
00011
00012
00013
00014
00015

```

**MODCURV
 NUMPTS
 AUTOGRAF
 24
 **+2
 +1
 ARGR2
 **ARGU1
 MODCURV
 NUMPTS
 GRAFPLOT

4

1

ZRO
 STA
 STQ
 LDA
 ARS
 SAU
 INA
 SAL
 LDA
 STA
 LDA
 LDQ
 SLJ
 OCT

AUTOGRAF

77777 +
 01110 +
 01111 +
 00003 +
 00030 +
 00004 +
 00001 +
 00016 +
 77777 +
 00007 +
 01110 +
 01111 +
 00022 +
 00000
 00003
 00003
 00015 +
 01014 +
 01014 +
 01014 +
 01014 +
 01014 +
 01022 +
 01022 +
 01014 +
 01017 +
 77777
 77777
 00000
 00000
 77777
 77777
 77777
 77777
 77777
 77777

ARGU1

ALL7S

=0
 =0
 =0
 =0
 =9
 =9
 =0
 =2
 -0

ALL7S

ARGR2

PLOTEXIT

OCT
 OCT
 ENI
 ENI
 ENI
 ENI
 ENI

1 2 3 4 5 6

GRAFPLOT

SLJ
 NOP
 SIL
 SIL
 SIL
 SIL
 SIL
 SAU

77777
 00003
 00017 +
 00017 +
 00023 +
 00023 +
 00021 +
 00021 +
 00027 +
 00003

75
 50
 56
 56
 57
 56
 57
 60

00022+
 00023+
 00024+
 00025+
 00026+

**
 PLOTEXIT
 PLOTEXIT+1
 PLOTEXIT+1
 PLOTEXIT+2
 PLOTEXIT+2
 **+1

1 2 3 4 5 6

00060+	51	1	0001	INI	1	1	1/2ADDR
00061+	51	1	01120+	SIL	1	4	
00062+	12	1	00004	INI	1	0	HOWMANY
00063+	22	1	00000	LDA	P	0	HOWMANY
00064+	22	1	00124+	AJP	N	0	SET4AUTO
00065+	75	0	00124+	SLJ	1	0	
00066+	51	1	00105+	INI	1	0	SFYADDR
00067+	12	1	00000	LDA	1	24	SFXADDR
00068+	61	0	01115+	ALS	1	0	
00069+	05	0	00030	SAL	1	0	
00070+	61	0	01114+	INI	1	0	
00071+	12	1	00000	LDA	1	0	
00072+	20	0	00071+	SAU	1	0	
00073+	12	0	00030	ALS	1	0	
00074+	60	0	00070+	SAU	1	0	
00075+	60	0	77777+	SAU	1	0	
00076+	20	0	01075+	STA	1	0	
00077+	12	0	77777+	LDA	1	0	
00100+	20	0	01076+	INI	1	0	
00101+	51	1	00001	LDA	1	0	
00102+	12	0	00000	SAU	1	0	
00103+	60	0	00076+	ALS	1	0	
00104+	50	0	00030	SAU	1	0	
00105+	12	0	00075+	SAU	1	0	
00106+	20	0	00000	LDA	1	0	
00107+	12	0	77777+	STA	1	0	
	20	0	01077+	LDA	1	0	
	51	1	77777+	STA	1	0	
	12	0	01115+	INI	1	0	
	60	0	00001	LDA	1	0	
	50	0	00000	SAU	1	0	
	12	0	00102+	ALS	1	0	
	20	0	01067+	SAL	1	0	
	51	1	77777+	LDA	1	0	
	12	0	01117+	STA	1	0	
	60	0	00001	INI	1	0	
	50	0	00000	SIL	1	0	
	12	0	00120+	LDA	1	0	
	20	0	00000	AJP	1	0	
	51	1	00110+	AJP	1	0	
	12	0	00110+	INI	1	0	
	60	0	00000	LDA	1	0	
	50	0	00001	SCL	1	0	
	12	0	00000	STA	1	0	
	20	0	01050+		1	0	
	51	1	01120+		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1	0	
	51	1	00000		1	0	
	12	0	00000		1	0	
	60	0	00000		1	0	
	50	0	00000		1	0	
	12	0	00000		1	0	
	20	0	00000		1		

00110+	12	0003	01077+	LDA			IWIDE
00111+	15	0003	01022+	SUB			=9
00112+	22	0003	00113+	AJP		Z M	*+2
00113+	22	0000	00113+	AJP			*+2
00114+	12	0000	01077+	LDA			=9
00115+	15	0000	01115+	LDA			IWIDE
00116+	22	0000	01025+	SUB		Z M	IHIGH
00117+	22	0003	00115+	AJP			=20
00120+	20	0004	00115+	AJP			*+2
00121+	75	0004	01025+	LDA			*+2
00122+	50	0004	01115+	STA			=20
00123+	50	0004	01077+	ENA			IHIGH
00124+	50	0001	X00001	RTJ			IWIDE
00125+	75	0000	X00004	RTJ			FLOATF
00126+	50	0000	00003				ERROR
00127+	20	0000	01065+	STA			WIDTH
00128+	12	0000	01123+	ENI			IHIGH
00129+	20	0000	01173+	LDA			FLOATF
00130+	60	0000	00125+	SAU			ERROR
00131+	50	0000	00003				HEIGHT
00132+	12	0000	77777+			1	0
00133+	20	0000	01171+	STA			T1/2ADDR
00134+	16	0000	01047+	ENI			ITEMP
00135+	44	0000	01171+	LDA			*+1
00136+	22	0001	00132+	STA			
	51	0000	00002	LDA			
	75	0000	01173+	STA			
	06	0000	00125+	LDQ			
	44	0000	00033	LDL			
	22	0001	01171+	AJP			
	72	0001	00135+	INI			
	51	0001	01173+	RAO			
	50	0000	00001	SLJ			
	57	0000	00003	QLS			
	12	0000	01154+	LDL			
	11	0000	01173+	AJP			
	60	0001	00001	RAO			
		00022+	00022+	INI			
				SIL			
				LDA			
				INA			
				SAU			

TEST IF FIRST CURVE ENTRY, THIS GRAPH

00137+	12	00	01110+	+	LDA	MODCURV
00140+	11	77	7775		INA	-2
00141+	22	00	340+		AJP	PLOT CURV
00142+	50	00	0000	+		SCALE
00143+	75	00	0435	+	RTJ	
00144+	50	00	0000			XSCALE
00145+	12	00	1102	+	LDA	X/YSCALE
00146+	20	00	1072	+	STA	ENCODE
00147+	75	00	0717	+	RTJ	
00148+	50	00	0000			ITEMP
00149+	12	00	1170	+	LDA	XSCLAB1
00150+	20	00	1104	+	STA	ITEMP+1
00151+	20	00	1171	+	LDA	XSCLAB2
00152+	16	00	1105	+	STA	MK1
00153+	44	00	1044	+	LDQ	ITEMP
00154+	05	00	1170	+	LDL	12
00155+	20	00	0014		ALS	LABELX
00156+	12	00	1103	+	STA	YSCALE
00157+	20	00	1107	+	LDA	X/YSCALE
00158+	75	00	1072	+	STA	ENCODE
00159+	50	00	0717	+	RTJ	
00160+	12	00	0000			ITEMP
00161+	20	00	1170	+	LDA	YSCLAB1
00162+	20	00	1106	+	STA	ITEMP+1
00163+	20	00	1171	+	LDA	YSCLAB2
00164+	16	00	1104	+	LDQ	MK1
00165+	44	00	1170	+	LDL	ITEMP
00166+	05	00	0014		ALS	12
00167+	20	00	1101	+	STA	LABELY
00168+	10	00	0000		ENA	0
00169+	20	00	1136	+	STA	IH
00170+	20	00	1137	+	STA	JV
00171+	12	00	1075	+	LDA	IXOFFS
00172+	24	00	1033	+	MUI	=100
00173+	20	00	1140	+	STA	JH
00174+	15	00	1026	+	SUB	=25
00175+	20	00	1141	+	LDA	JHL
00176+	12	00	1077	+	MUI	IWIDE
00177+	24	00	1033	+	STA	=100
00178+	20	00	1142	+	STA	LH
00179+	14	00	1136	+	ADD	IH
00180+	15	00	1021	+	SUB	=8
00181+	20	00	1143	+	STA	IHL
00182+	13	00	1033	+	LAC	=100
00183+	20	00	1144	+	STA	INCRJVL
00184+	20	00	1122	+	STA	IWIDE
00185+	12	00	1077	+	LDA	IYOFFS
00186+	15	00	1075	+	SUB	

NOT 1ST, GO PLOT CURVE
GO TEST/PERFORM SCALING

00170+	24	0	01100+	MUI	LABELX
00171+	13	0	01123+	STA	IVH
00172+	20	0	01103+	LAC	LABELX
00173+	12	0	01124+	STA	INCRVH
00174+	11	0	01077+	LDA	IWIDE
00175+	20	0	00001	INA	I
00176+	10	0	01125+	STA	NH
00177+	20	0	00204	ENA	204B
00178+	12	0	01150+	STA	LSHOH
00179+	24	0	01076+	LDA	IYOFFS
00180+	20	0	01033+	MUI	=100
00181+	20	0	01145+	STA	IV
00182+	12	0	01116+	LDA	IHIGH
00183+	24	0	01033+	MUI	=100
00184+	20	0	01145+	STA	LV
00185+	12	0	01076+	LDA	IYOFFS
00186+	24	0	01033+	MUI	=100
00187+	15	0	01031+	SUB	=70
00188+	20	0	01126+	STA	IVL
00189+	12	0	01146+	LDA	LV
00190+	11	0	00012	INA	IO
00191+	20	0	01127+	STA	JVL
00192+	15	0	01116+	LDA	IHIGH
00193+	24	0	01075+	SUB	IXOFFS
00194+	20	0	01101+	MUI	LABELY
00195+	13	0	01130+	STA	IVV
00196+	20	0	01101+	LAC	LABELY
00197+	12	0	01131+	STA	INCRVV
00198+	11	0	01116+	LDA	IHIGH
00199+	20	0	00001	INA	I
00200+	10	0	01132+	STA	NV
00201+	20	0	00204	ENA	204B
00202+	20	0	01133+	STA	LSVOV
00203+	12	0	01136+	LDA	IH
00204+	20	0	01134+	STA	IT
00205+	13	0	01053+	LAC	LOWMARG
00206+	24	0	01033+	MUI	=100
00207+	20	0	01135+	STA	JT
00208+	10	0	00004	ENA	4
00209+	20	0	01147+	STA	ITOR
00210+	12	0	01154+	LDA	KTITLE
00211+	11	0	00002	INA	2
00212+	20	0	01154+	STA	KTITLE
00213+	20	0			
00214+	20	0			
00215+	16	0	01045+	LDQ	WRITE TITLE CONTROL RECORD
00216+	44	0	01134+	LDL	MK2
	05	0	0001+	ALS	IT
					12

00217+	43	0	01135+	SSU	JT
00220+	05	0	00014	ALS	12
00221+	43	0	01147+	SSU	ITOR
00222+	05	0	00014	ALS	12
00223+	43	0	01154+	SSU	KTITLE
00224+	20	0	01170+	STA	ITEMP
00225+	50	0	00000		=1
00226+	10	0	01015+	ENA	=48
	04	0	01027+	ENQ	BUFOUT
	75	4	X00005	RTJ	
	50	0	00000		
	00	0	01170+	ZRO	ITEMP
	00	0	01170+	ZRO	ITEMP
	75	4	X00004	RTJ	ERROR
	50	0	00000		
	75	4	00737+	RTJ	TPCHK
	50	0	00000		

WRITE OUT AXIS SCALE TITLES

00227+	12	0	01056+	LDA	SCALEX1
00230+	16	0	01044+	LDQ	MK1
00231+	06	0	00006	QLS	6
00232+	43	0	01054+	SSU	SCALSIZ
00233+	20	0	01170+	STA	ITEMP
00234+	20	0	01057+	LDA	SCALEX2
00235+	20	0	01171+	STA	ITEMP+1
00236+	12	0	01104+	LDA	XSCLAB1
00237+	20	0	01172+	STA	ITEMP+2
00238+	12	0	01105+	LDA	XSCLAB2
00239+	20	0	01173+	STA	ITEMP+3
00240+	50	0	00000		=1
00241+	10	0	01015+	ENA	=48
00242+	04	0	01027+	ENQ	BUFOUT
00243+	75	4	X00005	RTJ	
00244+	50	0	00000		
00245+	00	0	01170+	ZRO	ITEMP
	00	0	01173+	ZRO	ITEMP+3
	75	4	X00004	RTJ	ERROR
	50	0	00000		
	75	4	00737+	RTJ	TPCHK
	50	0	00000		
	12	0	01060+	LDA	SCALEY1
	16	0	01044+	LDQ	MK1
	06	0	00006	QLS	6
	43	0	01054+	SSU	SCALSIZ
	20	0	01057+	STA	ITEMP+10
	12	0	01061+	LDA	SCALEY2
	20	0	01203+	STA	ITEMP+11

00246+	12	0	01105+	LDA	YSCLAB1
	20	0	01204+	STA	ITEMP+12
00247+	12	0	01107+	LDA	YSCLAB2
	20	0	01205+	STA	ITEMP+13
00250+	50	0	00000		
	10	0	01015+	ENA	=1
00251+	04	0	01027+	ENQ	=48
	75	4	X00005	RTJ	BUFOUT
00252+	50	0	00000		
	00	0	01202+	ZRO	ITEMP+10
00253+	00	0	01203+	ZRO	ITEMP+13
	75	4	X00004	RTJ	ERROR
00254+	50	0	00000		
	75	4	00737+	RTJ	TPCHK
	50	0	00000		

LOOP TO WRITE OUT TITLES

00255+	53	1	0115+	LIL	1	KTITLE
	51	1	77775	INI	1	-2
00256+	12	0	01051+	LDA		CONTROL2
	20	0	01052+	STA		CONTR1
00257+	12	0	01120+	LDA		T1/2ADDR
	60	0	00260+	SAU		**1
00260+	12	0	77777	LDA		**
	05	0	00033	ALS		24
00261+	60	0	00265+	SAU		STRITILE
	50	0	00000			
00262+	55	1	00263+	IJP	1	*+1
	75	0	00303+	SLJ	4	CONTROL
00263+	50	0	00024	ENI		20
	10	0	00000	ENA		0
00264+	20	4	01207+	STA	4	STORE
	55	4	00264+	IJP	4	KLEER
00265+	50	2	00000	ENI	2	0
	12	0	01117+	LDA		TITLESIZ
00266+	16	2	77777	LDQ	2	**
	07	2	00052	LLS	2	42
00267+	20	0	01207+	STA	2	STORE
	07	0	00005	LLS		6
00270+	20	0	01225+	STA		STORE+14
	50	0	00000			
00271+	12	0	01117+	LDA		TITLESIZ
	24	0	01021+	MUI		=8
00272+	70	0	01225+	RAD		STORE+15
	15	0	01033+	SUB		=100
00273+	22	0	00275+	AJP	P	NOMORR
	12	0	01225+	LDA		STORE+14
00274+	54	2	00011	ISK	2	9

00324+	43	0	01126+	SSU	IVL
00325+	43	0	00014+	SSU	12 JVL
00326+	43	0	00014+	ALS	12 INCR JVL
00327+	20	0	01122+	SSU	ITEMP+3
00328+	44	0	01130+	STA	IVV
00329+	44	0	00014+	LDL	12 INCR VV
00330+	43	0	01131+	SSU	12 NV
00331+	43	0	00014+	SSU	12 LSV OV
00332+	43	0	01132+	SSU	ITEMP+4
00333+	20	0	00014+	SSU	=1
00334+	50	0	01174+	STA	=48
00335+	10	0	00000	ENA	BUFOUT
00336+	04	0	01015+	ENQ	ITEMP
00337+	75	4	01027+	RTJ	ITEMP+4
			X00005		ERROR
			00000		TPCHK
			01173+	ZRO	
			01174+	ZRO	
			X00004	RTJ	
			00000		
			00737+	RTJ	
			00000		

1

WRITE CURVE DATA RECORD

00340+	10	0	01210+	ENA	STORE+1
00341+	41	0	01050+	SCL	CLEARMK
00342+	61	0	01153+	SAL	BUFAD
00343+	12	0	01051+	LDA	CONTR1
00344+	20	0	01052+	STA	CONTR2
00345+	12	0	01121+	LDA	INTERPS
00346+	20	0	01207+	STA	STORE
00347+	50	3	00000	ENI	0
00348+	12	3	77777	LDA	0 *
00349+	32	0	01034+	FMU	=100.0
00350+	33	0	01102+	FDV	XSCALE
00351+	20	0	01173+	STA	ITEMP
00352+	10	0	01170+	ENA	ITEMP
	75	4	X00003	RTJ	XINTF
	75	4	X00004	RTJ	ERROR
	50	0	00000		IV
	14	0	01145+	ADD	*+1
	50	0	00000		0
	22	0	00352+	AJP	IPT
	10	0	00000	ENA	
	20	0	01151+	STA	

00403+	07	00030		LLS	24	ST1007
00404+	27	00041+	+	QJPS	48	NFLAG
00405+	16	01036+	+	LDQS	24	BUFAD
00406+	07	00030	+	LLS	24	BUFAD
00407+	22	01153+	+	RAD	0	LABEL
00408+	72	01153+	+	NDP	24	104B
00409+	50	00000		ENQ	36	
00410+	10	00000	+	ENQ	24	BUFAD
00411+	16	01155+	+	LLS	36	INITERM
00412+	07	00030		ENQ	24	ST1008
00413+	04	00104		QLS	36	INFLAG
00414+	06	00044		LLS	24	LABEL
00415+	07	00030	+	LLS	24	BUFAD
00416+	20	01153+	+	STA	36	BUFAD
00417+	12	01153+	+	LDA	24	BUFAD
00418+	61	00423+	+	SAL	36	INITERM
00419+	75	00421+	+	SLJ	24	ST1007
00420+	12	01037+	+	LDQS	24	BUFAD
00421+	16	01155+	+	LLS	36	BUFAD
00422+	07	00030		STA	24	BUFAD
00423+	22	01153+	+	RAD	36	INITERM
00424+	10	00104		ENQ	24	BUFAD
00425+	05	00044		ALS	36	BUFAD
00426+	20	01153+	+	STA	24	INITERM
00427+	12	00423+	+	LDA	36	BUFAD
00428+	61	01015+	+	SAL	24	INITERM
00429+	10	01027+	+	ENQ	1	=48
00430+	04	00005		RTJ	=48	BUFOUT
00431+	75	00000		ZRO		STORE
00432+	50	01207+	+	ZRO	**	ERROR
00433+	00	77777	+	RTJ	TPCHK	
00434+	00	00004		RTJ	MODCURV	
00435+	04	00000		LDA	**3	
00436+	04	00737+	+	AJP	**2	
00437+	50	00000		AJP	-3	
00438+	12	01110+	+	INA	**4	
00439+	22	00431+	+	AJP	=48	
00440+	22	00431+	+	ENQ	WEOF	
00441+	50	00000		RTJ	ERROR	
00442+	11	77774	+	LDA	LABELCEL	
00443+	22	00434+	+			
00444+	10	01027+	+			
00445+	75	000011				
00446+	50	000004				
00447+	12	00000				
00448+	00	01042+	+			

00434+	76	1	00434+	00434+	SLS	1	*+1	
00435+	75	0	00435+	00017+	SLJ		PLTEXIT	
00436+	75	0	00436+	00000	SLJ		**SFXADDR	
00437+	12	0	00437+	77777+	LDA		*+1	
00440+	60	0	00440+	00437+	SAU			
00441+	50	0	00441+	00000				
00442+	12	0	00442+	77777+	LDA	N	**XSCALE	
00443+	20	0	00443+	01102+	STA		*+3	
00444+	22	1	00444+	00443+	AJP		SCALEX	
00445+	75	0	00445+	00451+	RTJ		XSCALE	
00446+	10	0	00446+	01102+	ENA		IYOFFS	
00447+	04	0	00447+	01075+	ENQ		ALLZEROS	
00450+	75	4	00450+	00732+	RTJ			
00451+	50	0	00451+	00000				
00452+	12	0	00452+	01115+	LDA		SFYADDR	
00453+	60	0	00453+	00444+	SAU		*+1	
00454+	20	0	00454+	77777+	LDA	N	**YSCALE	
00455+	22	1	00455+	01103+	STA		*+3	
00456+	75	0	00456+	00450+	AJP		SCALEY	
00457+	10	0	00457+	00475+	RTJ		XSCALE	
00460+	04	0	00460+	01103+	ENA		IYOFFS	
00461+	75	4	00461+	01075+	ENQ		ALLZEROS	
00462+	50	0	00462+	00732+	RTJ			
00463+	12	0	00463+	00000				
	20	0		00435+	SLJ		SCALE	
	75	0		00000				
				77777+	SLJ			
				01112+	SLJ		**XIADDR	
				01111+	LDA		NUMPTS	
				00577+	LDQ		MAX/MIN	
				01156+	RTJ		XMAX	
				01161+	STA	P	XMIN	
				00455+	STQ		*+1	
				00000	AJP		0	
				01157+	ENA		AXMAX	
				01160+	STA		AMAX	
				01161+	STA		XMIN	
				00000	LDA			
				00460+		M	*+1	
				77777+	AJP		-0	
				00015+	ENA		ALL7S	
				01162+	SCM		AXMIN	
				01163+	STA		AMIN	
				01064+	STA		WIDTH	
				01063+	LDA		LENGTH	
				77777+	STA		-0	
				01040+	ENA		X/YFLAG	
				00652+	STA		GETTOTAL	

SET ABSOLUTE
XMAX OR
(=0 IF XMAX IS -)

SET
ABSOLUTE
XMIN OR
(=0 IF XMIN IS +)

GO SET
AXMAX/AXMIN/XTOTAL
IN ACCORDANCE
WITH MODE SELECTED

00464+	12	0	01062+	+	LDA	N	TOTAL	
00465+	25	1	00466+	+	AJP		*+2	
00466+	50	0	00451+	+	SLJ		SCALEX	
00467+	52	1	00000		LIU	1	SCALEX	BUILD
00470+	56	1	00002		INI	1	2	NORMAL
00471+	75	1	00451+	+	SIU	1	SCALEX/Y	RETURN
00472+	12	4	00521+	+	RTJ		X/YSCALE	GO SCALE X
00473+	20	0	01072+	+	LDA		XSCALE	
00474+	25	0	01102+	+	STA		LENGTH	
00475+	50	0	01063+	+	LDA		WIDTH	
00476+	12	0	01064+	+	STA		CALCOFFS	
00477+	25	4	00622+	+	RTJ		IOFFS	
00478+	50	0	00000		LDA		IYOFFS	
00479+	12	0	01074+	+	SLJ		SCALEX	
00480+	25	0	01076+	+				
00481+	50	0	00451+	+				
00482+	12	0	00000		SLJ		**	
00483+	25	0	77777		LDA		YIADDR	
00484+	50	0	01113+	+	LDQ		NUMPTS	
00485+	12	0	01111+	+	RTJ		MAX/MIN	
00486+	25	4	00577+	+	STA		YMAX	
00487+	50	0	01164+	+	STQ		YMIN	
00488+	12	0	01165+	+	AJP	P	*+1	
00489+	25	0	00501+	+	ENA		0	
00490+	50	0	00000		STA		AYMAX	SET ABSOLUTE
00491+	12	0	01165+	+	STA		AMAX	YMAX OR
00492+	25	0	01166+	+	LDA		YMIN	(=0 IF YMAX IS -)
00493+	50	0	00000					SET
00494+	12	0	00504+	+	AJP	M	*+1	
00495+	25	3	77777		ENA		-0	
00496+	50	0	00015+	+	SCM		ALL7S	ABSOLUTE
00497+	12	0	01167+	+	STA		AYMIN	YMIN OR
00498+	25	0	01163+	+	STA		AMIN	(=0 IF YMIN IS +)
00499+	50	0	01065+	+	LDA		HEIGHT	
00500+	12	0	01063+	+	STA		LENGTH	
00501+	25	0	00000		ENA		+0	
00502+	50	0	01040+	+	STA		X/YFLAG	
00503+	12	0	00652+	+	RTJ		GETTOTAL	
00504+	25	4	01062+	+	LDA	N	TOTAL	
00505+	50	0	00512+	+	AJP		*+2	
00506+	12	1	00475+	+	SLJ		SCALEY	SET FOR Y SCALING
00507+	25	0	00000					
00508+	50	0	00475+	+	LIU	1	SCALEY	
00509+	12	1	00002		INI	1	2	BUILD
00510+	25	1	00475+	+	SIU	1	SCALEX/Y	NORMAL
00511+	50	4	00521+	+	RTJ		SCALEY	RETURN
00512+	12	0	01072+	+	LDA		X/YSCALE	GO SCALE Y
00513+	25	1						
00514+	50	0						

00515+	20	0	01103+	STA	YSCALE	
00516+	12	0	01063+	LDA	LENGTH	
00517+	20	0	01065+	STA	HEIGHT	
00520+	75	0	00622+	RTJ	CALCOFFS	
00521+	50	0	00000			
00522+	12	0	01074+	LDA	IOFFS	
00523+	20	0	01075+	STA	IXOFFS	
00524+	75	0	00475+	SLJ	SCALEY	
00525+	50	0	77777			
00526+	10	0	00001	SLJ	**	EXIT
00527+	20	0	01041+	ENA	1	
00528+	50	0	01066+	STA	O/UFLAG	
00529+	20	0	00000	STA	EXPONENT	
00530+	50	0	00574+	ENI	0	
00531+	12	0	01073+	LDA	QUANTAB	
00532+	20	0	00000	STA	QUANTA	
00533+	50	0	01024+	ENA	=10.0	
00534+	10	0	01065+	ENQ	EXPONENT	
00535+	20	0	00002	RTJ	XTOI	
00536+	75	0	00000		ERROR	
00537+	50	0	00000	RTJ		
00538+	12	0	01070	FMU	QUANTA	
00539+	20	0	01071+	STA	TEMPSCAL	
00540+	50	0	01063+	FSB	LENGTH	
00541+	31	0	01062+	FAP	TOTAL	
00542+	22	0	00534+	AJP	HITSCALE	
00543+	22	0	00535+	AJP	UNDRTEST	
00544+	75	0	00542+	SLJ	OVERTEST	
00545+	50	0	00000			
00546+	12	0	01071+	LDA	TEMPSCAL	
00547+	20	0	01072+	STA	X/YSCALE	
00548+	75	0	00521+	SLJ	SCALEX/Y	
00549+	50	0	00000			
00550+	12	0	01041+	LDA	O/UFLAG	
00551+	22	0	00550+	AJP	URESSCALE	
00552+	36	0	01041+	SSK	O/UFLAG	
00553+	75	0	00551+	SLJ	UP1/OUT	
00554+	50	0	00562+	RTJ	INCQUANT	
00555+	12	0	00000			
00556+	20	0	00525+	SLJ	TESTSCAL	
00557+	75	0	00000			
00558+	50	0	01041+	LDA	O/UFLAG	
00559+	12	0	00552+	AJP	URESSCALE	
00560+	22	0	01041+	SSK	O/UFLAG	
00561+	36	0	00545+	SLJ	LOWERSCL	
00562+	75	0	01071+	LDA	TEMPSCAL	
00563+	50	0	01072+	STA	X/YSCALE	

RESCALE IF FIRST PASS

O/UFLAG=+ IF OVER BEFORE
- IF UNDER BEFORE
INCREASE SCALE

00545+	75	0	00521+	SLJ	SCALEX/Y
00546+	50	0	00003	RTJ	DECQUANT
00547+	50	4	00566+		
00550+	75	0	00000	SLJ	TESTSCAL
00551+	50	0	00525+		
00552+	10	0	77777	URESCALE	-0
00553+	20	0	01041+	ENA	0/UFLAG
00554+	75	0	00525+	STA	TESTSCAL
00555+	50	0	00003	SLJ	
00556+	75	0	01041+	ENA	0
00557+	50	0	00525+	STA	0/UFLAG
00560+	32	0	00003	SLJ	TESTSCAL
00561+	75	0	01041+		INCQUANT
00562+	50	0	00525+	RTJ	=10.0
00563+	52	0	00003	ENA	EXPONENT
00564+	50	0	01024+	ENQ	XTOI
00565+	54	0	01066+	RTJ	ERROR
00566+	72	0	X00002		
00567+	52	0	00000	RTJ	QUANTA
00570+	50	4	X00004		X/YSCALE
00571+	50	0	00003	FMU	SCALEX/Y
00572+	50	0	01073+	STA	**
00573+	20	0	01072+	SLJ	* GOTEST+1
00574+	20	0	00521+		
00575+	20	0	00003	INCQUANT	2
		3	77777	SLJ	GOTEST
		3	00562+	LIU	EXPONENT
		3	00573+	SIU	GOTEST
		0	00003		**
		2	00002	ISK	* GOTEST+1
		0	00572+	SLJ	GOTEST
		0	01066+	RAO	EXPONENT
		0	00572+	SLJ	GOTEST
		3	77777	SLJ	**
		3	00566+	LIU	* GOTEST+1
		3	00573+	SIU	GOTEST
		0	00003		EXPONENT
		2	00572+	IJP	2
		0	00000		QUANTAB
		0	01066+	RSQ	QUANTA
		2	00002	ENI	**
		0	00574+	LDA	1.0
		0	01073+	STA	2.0
		0	77777	SLJ	
		0	00003	DEC	
		0	14003	DEC	
		0	00003		
		0	24003		

00576+	00	0	0000		DEC		5.0	
00577+	20	0	35000		SLJ		**RESTOR	
00600+	00	0	00000		SIU		SET1	
00601+	75	0	77777		SAL	MAX/MIN	OPERATE	
00602+	56	0	00617+		SAL		SET2	
00603+	60	0	00607+		SAL		NOTBGR	
00604+	61	0	00610+		SAL		SET3	
00605+	61	0	00612+		SAL		48	
00606+	61	0	00613+		LLS		-1	
00607+	61	0	00060		INA		TESTING	
00608+	07	0	77775		SAU		0	
00609+	11	0	00615+		ENI			
00610+	60	0	00000		LDA	SET1	**MAXARG	
00611+	50	0	00000		STA		MINARG	
00612+	12	0	77777		STA			
00613+	20	0	00620+		LDA	OPERATE	MAXARG	
00614+	30	0	00621+		FSB		**NOTBGR	
00615+	12	0	00620		AJP	SET2	**MAXARG	
00616+	22	0	77777		LDA		TESTING	
00617+	20	0	00612+		STA	NOTBGR	MINARG	
00618+	75	0	00620		SLJ		**TESTING	
00619+	31	0	77777		FSB	SET3	**MINARG	
00620+	22	0	00615+		AJP		0	
00621+	12	0	77777		LDA	TESTING	**MAX/MIN	
00622+	50	0	00621+		STA		1	
00623+	20	0	00000		ENI		1	
00624+	54	0	77777		ISK			
00625+	75	0	00607+		SLJ	RESTOR		
00626+	16	0	00620+		LDA	MAXARG		
	50	0	77777		ENI	MINARG		
	75	0	00577+		SLJ	BSS		
					BSS			
00622+	75	0	77777		SLJ	CALCOFFS	**MODEADDR	
00623+	12	0	01067+		LDA		MODE0	
00624+	22	0	00624+		AJP		0	
00625+	10	0	00000		ENA	MODE0	MODE1	
00626+	22	0	00632+		AJP		AMIN	
	12	0	01163+		LDA		X/YSCALE	
	33	0	01072+		FAD		=1.0	
	30	0	01015+		FAD	CONVERT	TEMP	
	20	0	01073+		STA			

Q TO A

SET NUMBER IN ARRAY

PRIME ROUTINE

LOOP

JUMP IF POSITIVE

NEW MAXARG

JUMP IF NEGATIVE

NEW MINARG

TEST FOR ALL ARGUMENTS

EXIT

00627+	10	0	01073+	INA	TEMP
00630+	75	4	x00003	RTJ	XINTF
00631+	50	0	00000	RTJ	ERROR
00632+	75	0	000004		
00633+	20	0	00000	STA	IOFFS
00634+	75	0	01074+	SLJ	CALCOFFS
00635+	11	0	00622+	INA	-1
00636+	22	0	77775	AJP	MODE2
00637+	33	0	00635+	LDA	LENGTH
00638+	75	0	01063+	FDV	=2.0
00639+	50	0	01023+	SLJ	CONVERT
00640+	11	0	00625+		
00641+	22	0	00000	INA	-1
00642+	75	0	77775	AJP	MODE3
00643+	10	0	00637+	ENA	CONVERT
00644+	75	0	00000	SLJ	-1
00645+	22	0	00626+	INA	MODE4
00646+	36	0	77775	SSK	X/YFLAG
00647+	75	0	01043+	SLJ	SETMOD3X
00648+	10	0	00642+	ENA	CONVERT
00649+	75	0	00000	SLJ	LENGTH
00650+	22	0	00625+	LDA	CONVERT
00651+	11	0	01063+	SLJ	-1
00652+	75	0	77775	INA	MODE5
00653+	10	0	00645+	AJP	LENGTH
00654+	75	0	01063+	LDA	CONVERT
00655+	22	0	00626+	SLJ	-1
00656+	36	0	77775	INA	MODERR
	75	0	00651+	AJP	X/YFLAG
	10	0	01043+	SSK	SETMOD5X
	75	0	00650+	SLJ	LENGTH
	22	0	01063+	SLJ	CONVERT
	11	0	00625+	ENA	CONVERT
	75	0	00000	SLJ	CONVERT
	10	0	00626+	ENA	MODE0
	75	0	00000	SLJ	**
	22	0	00627+	SLJ	MODEADDR
	75	0	77777	LDA	MODE0CK
	10	0	01067+	AJP	MODE1CK
	75	2	00654+	ENA	AMAX
	22	0	00000	AJP	AMIN
	10	0	00660+	LDA	TOTAL
	75	0	01160+	FAD	LENGTH
	22	0	01163+	STA	=1.0
	30	0	01062+	LDA	
	20	0	01063+	FSB	
	12	0	01015+		
	31	0	01015		

00657+	00	00652+	MODE1CK	STA SLJ	LENGTH GETTOTAL
00660+	11	77775		INA	-1
00661+	12	00665+		AJP	MODE2CK
00662+	31	01163+	+	FDB	AMAX
00663+	22	00664+		AJP	AMIN
00664+	30	01160+		LDA	*+2
00665+	75	01163+	+	FAD	AMAX
00666+	12	00665+		SLJ	*+2
00667+	30	01163+	+	LDA	AMIN
00670+	25	01163+	+	FAD	AMIN
00671+	75	01062+	+	STA	TOTAL
00672+	11	00652+	MODE2CK	SLJ	GETTOTAL
00673+	22	77775		INA	-1
00674+	10	00672+		AJP	MODE3CK
00675+	20	00003		ENA	0
00676+	12	01163+		STA	AMIN
00677+	75	01163+	+	LDA	AMAX
00678+	50	01062+	+	STA	TOTAL
00679+	11	00652+	MODE3CK	SLJ	GETTOTAL
00680+	22	77775		INA	-1
00681+	36	00702+		AJP	MODE4CK
00682+	75	01043	+	SSK	X/YFLAG
00683+	10	00677+	XMODE3	SLJ	YMODE3
00684+	20	00003		ENA	0
00685+	12	01163+		STA	AMIN
00686+	75	01163+	+	LDA	AMAX
00687+	50	01062+	+	STA	TOTAL
00688+	10	00652+	YMODE3	SLJ	GETTOTAL
00689+	20	00003		ENA	0
00690+	12	00003		STA	AMAX
00691+	75	01163+		LDA	AMIN
00692+	11	01062+	MODE4CK	STA	TOTAL
00693+	22	00652+		SLJ	GETTOTAL
00694+	10	00003		INA	-1
00695+	20	77775		AJP	MODE5CK
00696+	75	00705+		ENA	0
00697+	11	00003		STA	AMAX
00698+	22	01163+	+	LDA	AMIN
00699+	10	01163+	+	STA	TOTAL
00700+	20	01062+	+	SLJ	GETTOTAL
00701+	75	00652+		INA	-1
00702+	11	00003		AJP	MODE5CK
00703+	22	77775		ENA	0
00704+	10	00705+		STA	AMAX
00705+	20	00003		LDA	AMIN
00706+	75	01163+	+	STA	TOTAL
00707+	11	01062+	+	SLJ	GETTOTAL
	22	00652+	MODE5CK	INA	-1
	36	00003		AJP	MODERROR
		77775		SSK	X/YFLAG
		00715+			
		01043			

00710+	75	0	00713+	00003	SLJ	YMODE5	YMODE5
00711+	10	0	00714+	00003	ENA		0
00712+	20	0	00715+	01163	STA		AMAX
00713+	12	0	00716+	01163	LDA		AMIN
00714+	20	0	00717+	01062	STA		TOTAL
00715+	75	0	00720+	00652	SLJ		GETTOTAL
00716+	50	0	00721+	00000		YMODE5	0
00717+	10	0	00722+	00003	ENA		AMIN
00720+	75	0	00723+	00654	STA		AMAX
00721+	50	0	00724+	77777	LDA		TOTAL
00722+	10	0	00725+	00000	SLJ		GETTOTAL
00723+	75	0	00726+	01033		MODEERROR	0
00724+	50	0	00727+	01055	ENA		MODEOCK
00725+	10	0	00730+	01055	ENQ		**
00726+	75	0	00731+	X00005	RTJ		=63
00727+	50	4	00732+	00000			F.20
00730+	75	0	00733+	01173	ZRO		INGOUT
00731+	50	0	00734+	01023	ZRO		ITEMP
00732+	10	0	00735+	01023	ZRO		=10
00733+	75	0	00736+	X00004	RTJ		ERROR
00734+	50	4	00737+	00000			GOUT
00735+	10	0		00000	RTJ		
00736+	75	0		X00007			=0
00737+	50	4		00003	ZRO		X/YSCALE
	75	0		01014	ZRO		ERROR
	50	0		01072	RTJ		ENDGOUT
	75	0		X00004			ERROR
	50	4		00003	SLJ		ENCODE
	75	0		00717			**+2
	50	0		00000	SLJ		48
	75	0		77777	SAL		**+2
	61	0		00734	LLS		=1.0
	75	0		00063			**
	61	0		00735	SLJ		**
	12	0		01015	SAL		ALLZEROS
	20	0		77777	LDA		
	12	0		01015	STA		
	20	0		77777	LDA		
	75	0		00732	SLJ		
	50	0		00003			
	75	0		77777	SLJ		
	56	1		00765	SIU		

CHECK TAPE WRITE

00740+	52	1	00737+		LIU	1	TPCHK		
00741+	51	1	77774		INI	1	-3		
00742+	20	1	00000		LDA	1	0		
00743+	10	0	00753+	TP1	STA		TPPAR2		
00744+	75	0	00060	+	ENA		48		
00745+	22	0	X00012		RTJ	3	IFUNIT		
00746+	22	0	00742+	+	SCL	0	IFMSK		
00747+	22	0	00765+		AJP		TP1		
00750+	75	0	00001		AJP		TPE		
00751+	10	0	00756+		ALS	3	1		
00752+	04	0	00767+		AJP		TPEOT		
00753+	75	0	00765+		RAO	0	TPCTR		
00754+	50	0	00000		AJP		TPE		
00755+	75	0	X00013	+	ENA		=48		
00756+	50	0	X00015		RTJ		BKSP		
00757+	05	0	01027+		RTJ		ERROR*		
00758+	75	0	00000		ENA		=1		
00759+	50	0	01027+	TPPAR1	ENQ		=48		
00760+	75	0	X00005		RTJ		BUFOUT		
00761+	50	0	77777	TPPAR2	ZRO		**		
00762+	75	0	77777		ZRO		**		
00763+	50	0	X00015		RTJ		ERROR*		
00764+	75	0	00000		SLJ		TP1		
00765+	50	0	00742+	+					
00766+	10	0	00000		ALS				
00767+	75	0	00001	TPEOT	AJP	3	1		
00768+	22	0	00761+		ENA		TPEOTP		
00769+	75	0	01027+		RTJ		=48		
00770+	50	0	00771+	+	SLJ		NEWTAPE		
	75	0	00765+				TPE		
	50	0	00000		ENA		=48		
	75	0	X00013	+	RTJ		BKSP		
	50	0	X00015		RTJ		ERROR*		
	75	0	00000		ENA		=48		
	50	0	01027+	TPEOTP	RTJ		NEWTAPE		
	75	0	00771+	+	SLJ		TPPAR1		
	50	0	00751+	+					
	75	0	00000		ENA		**		
	50	1	77777	TPE	ENI	1	-3		
	10	0	77774		ENA		TPCTR		
	20	0	00767+	+	STA		TPCHK		
	75	0	00737+		SLJ		-3		
	77	7	77777	TPCTR	OCT				
	07	7	77774						
	07	7	77777	IFMSK	OCT				
							0777777777777777		

CAL IFUNIT TO CHECK WRITE

ZRO. O.K., OUT

BIT 47 NEG.=EOT
PARITY ERROR

TRY TO REWRITE 3 TIMES

IF NOT CORRECTED,
THEN IGNORE IT

REWRITE, TRY 3 TIMES

EOT

LAST RECORD OK

GET NEW TAPE
OUT

EOT WITH BAD PARITY

GET NEW TAPE
REWRITE LAST RECORD

EXIT

RESET COUNTER TO TRY 3

01020+	20	0	24000	=2.0	DEC	2.0
01021+	00	0	00000	=8	DEC	8
01022+	00	0	00010	=9	DEC	9
01023+	00	0	00011	=10	DEC	10
01024+	00	0	00012	=10.0	DEC	10.0
01025+	20	0	45000	=20	DEC	20
01026+	00	0	00000	=25	DEC	25
01027+	00	0	00024	=48	DEC	48
01027+	00	0	00031	=63	DEC	63
01030+	00	0	00060	=70	DEC	70
01031+	00	0	00077	=80	DEC	80
01032+	00	0	00106	=100	DEC	100
01033+	00	0	00120	=100.0	DEC	100.0
01034+	00	0	00144	=900	DEC	900
01035+	20	0	76200	NFLAG	OCT	3777377700000000
01035+	00	0	00000	INFLAG	OCT	37773777
01036+	00	0	01604	X/YFLAG	BSS	1
01037+	37	0	73777	O/UFLAG	BSS	1
01040+	00	0	00000	LABELCEL	BCD	112345678
01041+	01	0	20304	LABELCYC	OCT	0
01042+	05	0	60710	MK1	OCT	0077000000000000
01043+	00	0	00000	MK2	OCT	7777
01044+	00	0	70000	MASK	OCT	7777777700000000
01045+	00	0	00000	LOWMASK	OCT	77700000
01046+	77	0	77777	CLEARMK	OCT	7777777777700000
01047+	00	0	00000	CONTROL2	OCT	5252525252525252
01050+	77	7	77777			
01051+	77	7	00000			
01051+	52	5	25252			

01052+	52	CONTR1	OCT	5252525252525252
01053+	52	LOWMARG	DEC	1
01054+	00	SCALSIZ	OCT	0200000000000000
01055+	02	F.20	BCD	1(E10.2)
01056+	34	SCALEX1	BCD	1 X AXIS
01057+	73	SCALEX2	BCD	1SCALE =
01060+	20	SCALEY1	BCD	1 Y AXIS
01061+	22	SCALEY2	BCD	1SCALE =
01062+	65	TOTAL	OCT	0
01063+	00	LENGTH	OCT	0
01064+	00	WIDTH	OCT	0
01065+	00	HEIGHT	OCT	0
01066+	00	EXPONENT	OCT	0
01067+	00	MODEADDR	OCT	0
01070+	00	QUANTA	OCT	0
01071+	00	TEMPSCAL	OCT	0
01072+	00	X/YSCALE	OCT	0
01073+	00	TEMP	OCT	0
01074+	00	IOFFS	OCT	0
01075+	00	IXOFFS	OCT	0
01076+	00	IYOFFS	OCT	0
01077+	00	IWIDE	OCT	0
01100+	00	LABELX	OCT	0
01101+	00	LABELY	OCT	0

01102+	00	00000	XSCALE	OCT	0
01103+	00	00000	YSCALE	OCT	0
01104+	00	00000	XSLAB1	OCT	0
01105+	00	00000	XSLAB2	OCT	0
01106+	00	00000	YSLAB1	OCT	0
01107+	00	00000	YSLAB2	OCT	0
01110+	00	00000	MODCURV	OCT	0
01111+	00	00000	NUMPTS	OCT	0
01112+	00	00000	X1ADDR	OCT	0
01113+	00	00000	Y1ADDR	OCT	0
01114+	00	00000	SFXADDR	OCT	0
01115+	00	00000	SFYADDR	OCT	0
01116+	00	00000	IHIGH	OCT	0
01117+	00	00000	TITLESIZ	OCT	0
01120+	00	00000	T1/2ADDR	OCT	0
01121+	00	00000	INTERPS	OCT	0
01122+	00	00000	INCRJVL	OCT	0
01123+	00	00000	IVH	OCT	0
01124+	00	00000	INCRVH	OCT	0
01125+	00	00000	NH	OCT	0
01126+	00	00000	IVL	OCT	0
01127+	00	00000	JVL	OCT	0
01130+	00	00000	IVV	OCT	0
01131+	00	00000	INCRVV	OCT	0
01132+	00	00000	NV	OCT	0

01133+	00	00000	LSVOV	OCT	0
01134+	00	00000	IT	OCT	0
01135+	00	00000	JT	OCT	0
01136+	00	00000	IH	OCT	0
01137+	00	00000	JV	OCT	0
01140+	00	00000	JH	OCT	0
01141+	00	00000	JHL	OCT	0
01142+	00	00000	LH	OCT	0
01143+	00	00000	IHL	OCT	0
01144+	00	00000	INCR IHL	OCT	0
01145+	00	00000	IV	OCT	0
01146+	00	00000	LV	OCT	0
01147+	00	00000	ITOR	OCT	0
01150+	00	00000	LSHOH	OCT	0
01151+	00	00000	IPT	OCT	0
01152+	00	00000	JPT	OCT	0
01153+	00	00000	BUFAD	OCT	0
01154+	00	00000	KTITLE	OCT	0
01155+	00	00000	LABEL	OCT	0
01156+	00	00000	XMAX	OCT	0
01157+	00	00000	AXMAX	OCT	0
01160+	00	00000	AMAX	OCT	0
01161+	00	00000	XMIN	OCT	0
01162+	00	00000	AXMIN	OCT	0


```

C      IICOL = HIGHEST NR. COLUMN OF FIELD, MAX FIELD WIDTH = 7
C      NFLAG1 = LAST BLOCK KEY WORD COUNT
C      NFLAG2 = RUNNING TOTAL OF NR. OF BLOCKS ON MERGED TAPE
C      NFLAG3 = TAPE NR. WHICH HOLDS MERGED FILE
C      NFLAG4 = TAPE NR. ON WHICH TO ORDER NEXT MERG
      SLJ(400)
      NFLAG1 = -0
      NFLAG2 = 0
      REWIND INPUT
      REWIND NFLAG3
      REWIND NFLAG4
      REWIND LISTOUT
      NRCOLS=IICOL-IICOL+1
      IF(NRCOLS-7) 100,100,99
      99 STOP
      100 CALL ALSINP(IICOL,IICOL,INPUT)
      ALS(33)
      ARS(33)
      STA(NFLAG1)
      AJP3(200)
      AJP(260)
      200 CALL RLHMERG(NFLAG1)
      IF(NFLAG2) 250,250,300
      250 CALL WRITE(NFLAG3,37000B,57000B,1)
      NFLAG2 = 1
      REWIND NFLAG3
      LDA(NFLAG1)
      AJP3(100)
      260 SLJ(400)
      NFLAG2=NFLAG2+1
      300 SLJ(400)
      CALL HEXSORT(NFLAG1,NFLAG2,NFLAG3,NFLAG4)
      NFLAG2 = NFLAG2 + 1
      ITEM3 = NFLAG3
      NFLAG3 = NFLAG4
      NFLAG4=ITEM3
      REWIND NFLAG3
      REWIND NFLAG4
      LDA(NFLAG1)
      AJP3(100)
      400 REWIND INPUT
      REWIND NFLAG4
      CALL KREMKEY(NFLAG1,NFLAG2,NFLAG3,LISTOUT)
      END

      MACHINE RDMT(1ARG,2ARG,3ARG,4ARG,5ARG,6ARG)

```

```

* NOTE G = ADDRESS OF ENTRY TO TYPEWRITER SENSE ROUTINE
* P = ADDRESS OF PROGRAM CONTROL ROUTINE JUMP ENTRY
* THIS ROUTINE IS FOR PROGRAM CALL

```

LOC(G=31, P=10)

* TYPEWRITER MESSAGE CODE TABLE

```

CON(T6 =4504041220302204B,
*   T8 =1212C31200000000B,
*   T10=1530121401250420B,
CON(T12=4504041220302204B,
*   T14=1411200000000000B,
*   T40=4504040130152004B,
*   T42=0414062214163001B,
T7 =1120061301050420B,
T9 =4504041220302204B,
T11=1212031200000000B,
T13=2006220403260426B,
T15=4504043112140120B,
T41=3406140104060301B,
T43=2022420000000000B)

```

* TAPE UNIT ASSIGNMENT TABLE
ASSIGNMENTS FOR CHANNEL 3/4 LIBRARY TAPE

```

CON(K0 =0, K1 =332010B, K2 =332020B,
*   K3 =332030B, K4 =332040B, K5 =552010B,
*   K6 =552020B, K7 =552030B, K8 =552040B)

```

* ASSORTED CONSTANTS

```
CON(R4 =73737373737373B, R16=0070000000000000B)
```

* TAPE READING ROUTINE

```

1X SLJ(N)
1ARG ZRO(O)
2ARG ZRO(O)
3ARG ZRO(O)
4ARG ZRO(O)
5ARG ZRO(O)
6ARG ZRO(O)
LDA(5ARG)
LDA7(6ARG)
SIUT(23X)
LDA7(1ARG)
SIUT(20X)
ENA(T40)
SLJ(P)
INA(11B)
LIL1(K0)
ADD7(4ARG)
SCL(777B)
SAL(5X)

SLJ(L+7)
ZRO(O)
ZRO(O)
ZRO(O)
ZRO(O)
ZRO(O)
ZRO(O)
SAU(22X)
SAL(23X)
ENI(O)
INA(-11B)
AJP3(L+3)
SLJ4(3)
ZRO(O)
STA(K0)
LDA1(K0)
SAU(4X)
SAL(4X)
SAL(11X)

```

```

. EXIT/ENTRY
. TAPE UNIT NUMBER ARG.
. INITIAL ADDRESS
. TERMINAL ADDRESS
. MODE, 1 = BINARY, 2 = BCD
. ENDFILE EXIT ADDRESS LOCATOR

. CHECK FOR TU ASSIGNMENT
. GO PRINT ERROR

. SET TU CONTROL CODES
. INITIALIZE

```

00970
00980
00990
01000
01010
01020
01030
01040
01050
01060
01070
01080
01090
01100
01110
01120
01130
01140
01150
01160
01170
01180
01190
01200
01210
01220
01230
01240
01250
01260
01270
01280
01290
01300
01310
01320
01330
01340
01350
01360
01370
01380
01390
01400
01410
01420
01430
01440
01450


```

INA(3)
INA(2)
INA(1)
INA(1)
LRS(18)
LRS(3)
LRS(3)
LRS(39)
LRS(KO)
SSU(KO)
LDA7(2ARG)
LDA(6X)
LDA(6X)
LDA7(3ARG)
SAU(6X)
ENI2(4)
EXF(N)
EXF(N)
EXF(N)
SUB(R4)
INI1(-1)
INI1(1)
EXF7(N)
EXF7(N)
EXF7(N)
QJP1(19X)
SLJ(19X)
IJP2(L+2)
IJP2(L+1)
AJP(3X)
EXF(N)
LIL1(N)
ENA(T6)
ENA(T9)
ENA(T12)
SLJ4(G)
AJP(19X)
SLJ(11X)
SLJ4(G)
AJP(3X)
SLJ(11X)
AJP(11X)
ENI2(N)
ENI1(N)
LIL2(L-2)
SIL2(N)
ENI1(N)
END

```

2X
3X
4X
5X
6X

7X
8X
9X

10X

11X
12X
13X
14X
15X
17X

18X

19X
20X
21X
22X
23X

```

SAU(8X)
SAU(9X)
SAU(11X)
SAU(7X)
ENA(0)
SAL(2X)
SAU(12X)
LDQ(RT5)
LDA(5X)
STA(5X)
SSU(KO)
ENI(0)
SAL(N)
SIU2(19X)
LIL1(N)
EXF7(N)
EXF7(N)
LDA7(2ARG)
LRS(48)
LDA1(Z)
SUB(R4)
SLJ(21X)
SLJ(L+4)
SLJ(10X)
AJP(3X)
ZRO(0)
SLJ(14X)
SLJ(13X)
QJP(3X)
EXF7(N)
SLJ(4X)
SLJ(17X)
SLJ(18X)
ZRO(0)
AJP3(3X)
ZRO(0)
AJP3(3X)
ZRO(0)
AJP3(3X)
ZRO(0)
ENI(0)
SLJ(1X)
INI2(-1)
ENI2(512)
SLJ(N)

```

SELECT UNIT
ACTIVATE CHANNEL

SENSE ENDFILE
SENSE PARITY ERROR
SENSE LENGTH ERROR
SENSE FOR BAD RECORD
EXIT
SENSE FIVE PARITY ERRORS
SENSE FIVE LENGTH ERRORS
SENSE FOR BAD RECORD
BACKSPACE
RETURN TO REREAD
SET TO INDICATE TROUBLE

TROUBLE EXIT
SENSE ACTION TO TAKE

TROUBLE EXIT
SENSE ACTION TO TAKE

RE-STORE INDEXES

ENDFILE DETECTED/EXIT

01460
01470
01480
01490
01500
01510
01520
01530
01540
01550
01560
01570
01580
01590
01600
01610
01620
01630
01640
01650
01660
01670
01680
01690
01700
01710
01720
01730
01740
01750
01760
01770
01780
01790
01800
01810
01820
01830
01840
01850
01860
01870
01880
01890
01900
01910
01920
01930
01940

[illegible]

Address	Operation	Comment
000000	CON(ORD40=1717171717171717B,	ORD41=1616161616161616B)
000001	CON(ORD42=1515151515151515B,	ORD43=1414141414141414B)
000002	CON(ORD44=1313131313131313B,	ORD45=1212121212121212B)
000003	CON(ORD46=1111111111111111B,	ORD47=1010101010101010B)
000004	CON(ORD50=0707070707070707B,	ORD51=0606060606060606B)
000005	CON(ORD52=0505050505050505B,	ORD53=0404040404040404B)
000006	CON(ORD54=0303030303030303B,	ORD55=0202020202020202B)
000007	CON(ORD56=0101010101010101B,	ORD57=0000000000000000B)
000008	ENA(1EFX)	STA(ENDFILE)
000009	ENA(-0)	STA(SWITCH)
000010	ENA(SLK1)	STA(SLK1)
000011	INA(1)	STA(INIT)
000012	SAU(2WHL)	SAU(2TRU)
000013	INA(17B)	STA(1TERM)
000014	SIU2(2STI)	ENI2(1)
000015	LDA(IICOL)	SUB(IICOL)
000016	SAU(2CNV)	SIL6(2STI)
000017	INA(1)	MUI(6)
000018	SAL(2CKK)	ENQ(-0)
000019	ENA(0)	LLS(N)
000020	STQ(MSK2)	ENQ(0)
000021	LDA(ICOL)	DVI(8)
000022	QJP1(L+1)	STQ(SWITCH)
000023	STA(TMP1)	LDA(IICOL)
000024	ENQ(0)	STQ(TMP2)
000025	DVI(8)	AJP1(1SPL)
000026	SUB(TMP1)	SLJ(1TRU)
000027	SKK(SWITCH)	SUB(TMP2)
000028	ENA(8)	SAL(3WHL)
000029	MUI(6)	ADD(TMP1)
000030	ENA(N)	ENI(0)
000031	SAU(3WHL)	CALL RDMT(INPUT,INIT,1TERM,2,SLK1,ENDFILE)
000032	CALL RDMT(INPUT,INIT,1TERM,2,SLK1,ENDFILE)	ARS(N)
000033	LDA(N)	SLJ4(1CNV)
000034	SLJ4(1CNV)	SLJ(2WHL)
000035	LDA(TMP2)	ENA(8)
000036	ENA(8)	SLJ(1WHL)
000037	SLJ(1WHL)	ENA(6)
000038	ENA(6)	SAU(4TRU)
000039	SAU(4TRU)	ENA(N)
000040	ENA(N)	SAU(3TRU)
000041	SAU(3TRU)	ENI(0)
000042	SAL(3TRU)	CALL RDMT(INPUT,INIT,1TERM,2,SLK1,ENDFILE)
000043	CALL RDMT(INPUT,INIT,1TERM,2,SLK1,ENDFILE)	LQ(N)
000044	LQ(N)	SLJ4(1CNV)
000045	SLJ(2TRU)	

LIL6(VA+7)	ENI(0)	SET B6=TOTAL NR.	03420
LDA(KA4)	STA6(A3 IN)	MODIFY	03430
INI6(20B)	SIL6(VA+10B)	EXCESSIVE	03440
LDA(VA+10B)	SUB(KA2)	KEYS IN	03450
AJP3(18B+2)	ENI(0)	BLOCK	03460
CYCLE			03470
LDA(VA+2)	SAL(2BJ)	SET FIRST ADDRESS THIS CYCLE	03480
LDA(VA+1)	STA(VA+5)	SET FIRSTL=ALPHA	03490
LDA(VA+3)	INA(7776B)	GENERATE ORDSIZE - 1	03500
SAU(4BF)	SAU(4BV)	SET(ORDSIZE) IN STOR, RRJNOUT	03510
SAU(4BG)	SAU(4BV)	SET(ORDSIZE) IN STORL AND LRUNOUT	03520
PRE MERG			03530
LDA(VA+5)	SAU(1BE)	SET FIRSTL ADDRESS IN MERG	03540
ADD(VA+3)	STA(VA+6)	GENERATE FIRSTR ADDRESS	03550
SAL(1BE)	LQ2(VA+6)	SET FIRSTR ADDRESS IN MERG	03560
MERG			03570
LDA1(N)	SUB2(N)	SUBTRACT CURRENT KEYR FROM KEYL	03580
AJP(1BF)	ENI(0)	IF DIFF IS ZERO, GO STORL	03590
AJP2(1BG)	SLJ(1BF)	IF DIFF IS (+) GO TO STORR,	03600
		IF (-) GO TO STORL	03610
STORL			03620
LDA(VA+5)	INA1(0)	GENERATE ADDRESS OF KEYL	03630
SAU(2BJ)	SLJ4(13J)	SET ADDR OF KEYL IN (STORE),	03640
		RETURN JUMP TO (STORE)	03650
INI1(17B)	ENI(0)	INCREASE INDEX-1 BY 17B	03660
ISK1(N)	SLJ(1BE)	IF B1=ORDSIZE-1, ZERO B1, EXIT	03670
		IF NOT, B1 = B1 + 1, GO TO MERG	03680
SLJ(1BM)	ZRO(0)	COMPLETE STORAGE OF RIGHT LIST	03690
STORR			03700
LDA(VA+6)	INA2(0)	GENERATE ADDRESS OF KEYR	03710
SAU(2BJ)	SLJ4(13J)	SET ADDR KEYR IN (STORE) RJ(STORE)	03720
INI2(17B)	ENI(0)	INCREASE B2, 17B	03730
ISK2(N)	SLJ(1BE)	IF B2=ORDSIZE-1, ZERO B2	03740
		IF NOT, B2=B2+1, GO TO MERG	03750
SLJ(1BN)	ZRO(0)	COMPLETE STORAGE LEFT LIST	03760
NEW PAIR			03770
LDA(VA+3)	ADD(VA+3)	GENERATE PAIR SIZE	03780
ADD(VA+5)	STA(VA+5)	GEN/SET ADDR OF FIRSTL	03790
ENI1(0)	ENI2(0)	ZERO B1 AND B2	03800
SLJ(1BD)	ENI(0)	GO TO PRE-MERG	03810
RE-CYCL			03820
LDA(VA+1)	ADD(VA+4)	ALPHA= ALPHA (+) OR (-) 20,000B	03830
STA(VA+1)	LAC(VA+4)	COMPLIMENT (+) OR (-) 20,000B	03840
STA(VA+4)	RAD(VA+2)	BETA = BETA (+) OR (-) 20,000B	03850
LDA(VA+3)	RAD(VA+3)	DOUBLE ORDSIZE	03860
ISK5(10B)	SLJ(1BC)	EXIT IF MERG COMPLETE	03870
		IF NOT GO TO CYCLE	03880
SLJ(1BK)	ZRO(0)	GO TO (EXIT)	03890
STORE			03900

1BJ	SLJ(N)	ENI3(O)	ENTRY/EXIT, ZERO B3	03910
2BJ	LDA3(N)	STA4(N)	STORE KEY AND	03920
	INI4(1)	ENI(O)	TRAILER WORDS	03930
	ISK3(17B)	SLJ(1BJ+1)	AT END OF MERGED LIST	03940
	INI4(77776B)	ENI(O)	INCREASE B1 BY (-1)	03950
	ISK4(17777B)	SLJ(1BJ)	IF BLC PROCESS COMPLETE GO TO	03960
*			(RE-CYCLE) IF NOT COMPLETE IT	03970
	ENI1(O)	ENI2(O)	ZERO B1, B2	03980
	SLJ(1BI)	ZRO(O)	JUMP TO RE-CYCL	03990
* 1BK	EXIT			04000
	LDA(VA)	AJP2(13L)	NFLAG1=-0, IF NOT GOTO(FINAL EXIT)	04010
	LIU1(RA)	LIL2(RA)	RESET	04020
	LIU3(RA+1)	LIL4(RA+1)	ALL B - BOXES	04030
	LIU5(RA+2)	LIL6(RA+2)	EXIT FROM S.R. RLHMERG	04040
	SLJ(1AA)	ZRO(O)		04050
* 1BL	FINAL EXIT			04060
	LDA(KA2)	SUB(VA+7)	DETERMINE NR. OF NULL CELLS	04070
	STA(VA+7)	LIL6(VA+7)	SET B6=NR. OF NULL CELLS	04080
	ENI1(O)	ENI(O)	SET B1=0	04090
	LDA6(BBIN)	ENI(O)	RELOCATE VALID CELLS	04100
	STAT(BBIN)	INI1(1)	TO LOWER STORAGE AT 30,000B	04110
	ISK6(17777B)	SLJ(1BL+3)	WHEN COMPLETE, JUMP TO RESTORE	04120
	LDA(VA)	SLJ(1BK+1)	SECTION OF (EXIT)	04130
* 1BM	RRUNOUT			04140
	LDA(VA+6)	INA2(O)	GENERATE ADDRESS OF KEYR	04150
	SAU(2BJ)	SLJ4(13J)	SET ADDR KEYR IN (STORE)RJ(STORE)	04160
4BM	INI2(17B)	ENI(O)	INCREASE B2, 17B	04170
	ISK2(O)	SLJ(1BM)	IF B2=ORDSIZE-1, ZERO B2	04180
	SLJ(1BH)	ENI(O)	IF NOT, B2=B2+1, GO TO NEW PAIR	04190
* 1BN	LRUNOUT			04200
	LDA(VA+5)	INA1(O)	GENERATE ADDRESS OF KEYL	04210
	SAU(2BJ)	SLJ4(1BJ)	SET ADDR OF KEYL IN (STORE),	04220
*			RETURN JUMP TO (STORE)	04230
4BN	INI1(17B)	ENI(O)	INCREASE INDEX 1 BY 17B	04240
	ISK1(O)	SLJ(1BN)	IF B1=ORDSIZE-1, ZERO B1, EXIT	04250
	SLJ(1BH)	ZRO(O)	IF NOT, B1 = B1 + 1 GOTO NEW PAIR	04260
	END			04270
				04280
				04290
				04300
				04310
				04320
				04330
				04340
				04350
				04360
				04370
				04380
				04390

SUBROUTINE HEXSORT(NFLAG1,NFLAG2,NFLAG3,NFLAG4)	
LOC(ABIN=17000, BBIN=37000, CBIN=57000)	
CON(TWENTY=20B)	
SIU1(STOR1)	SAVE B - BOXES
SIU3(STOR2)	SET B-BOXES
SIU6(STOR3)	TO ZERO
ENI1(1)	
ENI3(O)	
ENI6(1)	
SAU(2MERS)	PRIME SUBROUTINE

LDA(NFLAG2)	SAU(1C)	.	04400
SAU(1TEST)	ENI(0)	.	04410
SLJ4(1RD)	ENI(0)	.	04420
LDA(NFLAG1)	AJP3(1A)	INSPECT IF LAST BLOCK	04430
MUI(TWENTY)	SAU(2MERG)	SET LAST BLOCK PARAMETER	04440
INI1(-1)	ENI(0)	MERG ONE KEY	04450
LDA1(ABIN)	SUB2(BBIN)	.	04460
AJP(1ASML)	ENI(0)	AND TRAILER	04470
AJP2(1BSML)	SLJ(1ASML)	IS BBIN EMPTY	04480
ISK2(20000B)	SLJ(1CONT)	IS ABIN EMPTY	04490
ISK1(20000B)	SLJ4(1DUMP)	IS READ TAPE EMPTY	04500
ISK6(N)	SLJ(1PROC)	IS SERVICE CBIN PRIOR TO EXIT	04510
SLJ(2TEST)	ENI(0)	PROCESS OUT	04520
SLJ4(1RD)	ENI(0)	REST OF	04530
ENI1(1)	SLJ4(1DUMP)	READ TAPE	04540
SLJ(1C)	ENI(0)	.	04550
INI2(-1)	ENI(0)	.	04560
ISK1(20000B)	SLJ(1A)	IS ABIN EMPTY	04570
ISK6(N)	SLJ(1READ)	IS READ TAPE EMPTY	04580
SLJ4(2DUMP)	ZRD(0)	.	04590
LDA(NFLAG1)	AJP3(1END)	IS BBIN LAST BLOCK	04600
AJP(1END)	INA(-1000B)	WAS LAST BLOCK, FULL SIZE	04610
SLJ(1END)	SLJ4(1WRT)	.	04620
SLJ(N)	ENI(0)	CBIN STORAGE	04630
LDA5(N)	ENI(0)	ROUTINE	04640
INI3(1)	STA3(CBIN)	AND AUTOMATIC	04650
ISK5(17B)	ENI(0)	WRITE-OUT	04660
INI3(-1)	SLJ(20TC3+1)	WHEN FULL	04670
ISK3(17777B)	ENI(0)	.	04680
SLJ4(1WRT)	SLJ(20TCB)	.	04690
SLJ(20TCB)	ENI(0)	.	04700
SLJ(N)	ENI(0)	READ ROUTINE	04710
CALL READ (NFLAG3,17000B,37000B,1)	ENI(0)	.	04720
SLJ(1RD)	ENI(0)	WRITE ROUTINE	04730
SLJ(N)	ENI(0)	.	04740
CALL WRITE (NFLAG4,57000B,77000B,1)	ENI(0)	.	04750
SLJ(1WRT)	ENI(0)	TRANSFER KEY	04760
ENAI(ABIN)	SAU(20TC3+1)	FROM ABIN	04770
SLJ4(20TCR)	ENI(0)	TO CBIN	04780
INI1(20B)	SLJ(2MERG)	TRANSFER KEY	04790
ENAI(20TCR)	SAU(20TC3+1)	FROM BBIN	04800
SLJ4(20TCB)	ENI(0)	.	04810
SLJ(20B)	SLJ(2MERG)	TRANSFER	04820
SLJ(N)	INI1(-1)	REMAINING	04830
ENAI(ABIN)	SAU(20TC3+1)	KEYS	04840
SLJ4(20TCB)	ENI(0)	FROM	04850
INI1(17B)	ENI(0)	ABIN TO	04860
ISK1(17777B)	SLJ(1DUMP+1)	CBIN	04870
			04880

```

2DUMP      SLJ(1DUMP)
            SLJ(N)
            ENA2(RBIN)
            SLJ4(20TCB)
            INI2(17H)
            ISK2(17777B)
            SLJ(2DUMP)
            SLJ4(1RD)
            ENI(1)
            LIU1(STOR1)
            LIU3(STOR2)
            LIU6(STOR3)
            END

1READ      SLJ(1A)
            LIL2(STOR1)
            LIL5(STOR2)
            ENI(O)

1END

```

```

. TRANSFER
. REMAINING
. KEYS FROM
. 8BIN TO
. CBIN
.
. NON-RETURN JUMP
. READ ROUTINE
.
.
.

```

SUBROUTINE KREMKEY(NFLAG1,NFLAG2,NFLAG3,LISTOUT)

```

REWIND LISTOUT
REWIND NFLAG3
NFLAG22=NFLAG2
NFLAG22=NFLAG22-1
IF(NFLAG22) 200,200,100
CALL READ(NFLAG3,17000B,37000B,1)
DO 10 I=1,512
INIT = 16761B + 20B*I
NTERM = 17000B + 20B*I
CALL WRITE(LISTOUT,INIT,NTERM,2)
CONTINUE
GO TO 50
IF(NFLAG1) 20,20,201
CALL READ(NFLAG3,17000B,37000B,1)
DO 20 I=1,NFLAG1
INIT = 16761B+20B*I
NTERM = 17000B+20B*I
CALL WRITE(LISTOUT,INIT,NTERM,2)
CONTINUE
REWIND NFLAG3
END FILE LISTOUT
END

```

MACHINE READ(1ARG,2ARG,3ARG,4ARG)

* NOTE G = ADDRESS OF ENTRY TO TYPEWRITER SENSE ROUTINE
 * P = ADDRESS OF PROGRAM CONTROL ROUTINE JUMP ENTRY
 * THIS ROUTINE IS FOR PROGRAM CALL

LOC(G=31, P=10)

04890
 04900
 04910
 04920
 04930
 04940
 04950
 04960
 04970
 04980
 04990
 05000
 05010
 05020
 05030
 05040
 05050
 05060
 05070
 05080
 05090
 05100
 05110
 05120
 05130
 05140
 05150
 05160
 05170
 05180
 05190
 05200
 05210
 05220
 05230
 05240
 05250
 05260
 05270
 05280
 05290
 05300
 05310
 05320
 05330
 05340
 05350
 05360
 05370

* TYPEWRITER MESSAGE CODE TABLE

```

CON( T6 =4504041220302204B,
*      T8 =1212031200000000B,
*      T10=1530121401250420B,
CON( T12=4504041220302204B,
*      T14=1411200000000000B,
*      T40=4504040130152004B,
*      T42=0414062214163001B,

```

* TAPE UNIT ASSIGNMENT TABLE
* ASSIGNMENTS FOR CHANNEL 3/4 LIBRARY TAPE

```

CON(K0 =0, K1 =332010B, K2 =332020B,
*      K3 =332030B, K4 =332040B, K5 =552010B,
*      K6 =552020B, K7 =552030B, K8 =552040B)

```

* ASSORTED CONSTANTS

```

CON(R4 =73737373737373B, R16=0070000000000000B)

```

* TAPE READING ROUTINE

```

1X      SLJ(N)
1ARG    ZRO(0)
2ARG    ZRO(0)
3ARG    ZRO(0)
4ARG    ZRO(0)
        LDAT(1ARG)
        SIU1(19X)
        ENA( T40)
        SLJ( P)
        INA(11B)
        LIL1(K0)
        ADD7(4ARG)
        SCL(777B)
        SAL(5X)
        INA(3)
        INA(2)
        INA(1)
        INA(1)
        LRS(18)
        LLS(3)
        SAL(3X)

        SLJ(L+5)
        ZRO(0)
        ZRO(0)
        ZRO(0)
        ZRO(0)
        INA(-11B)
        AJP3(L+3)
        SLJ4( 5)
        ZRO(0)
        STA(K0)
        LDA1(K0)
        SAU(4X)
        SAL(4X)
        SAL(11X)
        SAU(8X)
        SAU(9X)
        SAU(11X)
        SAU(7X)
        ENA(0)
        SAL(2X)
        SAU(12X)

```

```

* EXIT/ENTRY NUMBER ARG.
* TAPE UNIT ADDRESS
* INITIAL ADDRESS
* TERMINAL ADDRESS
* MODE, 1 = BINARY, 2 = BCD
* CHECK FOR TU ASSIGNMENT
* GO PRINT ERROR
*
* SET TU CONTROL CODES
* INITIALIZE

```

05380
05390
05400
05410
05420
05430
05440
05450
05460
05470
05480
05490
05500
05510
05520
05530
05540
05550
05560
05570
05580
05590
05600
05610
05620
05630
05640
05650
05660
05670
05680
05690
05700
05710
05720
05730
05740
05750
05760
05770
05780
05790
05800
05810
05820
05830
05840
05850
05860

* TYPEWRITER MESSAGE CODE TABLE

[illegible]

* TAPE UNIT ASSIGNMENT TABLE
* ASSIGNMENTS FOR CHANNEL 3/4 LIBRARY TAPE

CON(K0 = 0,	K1 = 442010B,	K2 = 442020B,
K3 = 442030B,	K4 = 442040B,	K5 = 662010B,
K6 = 662020B,	K7 = 662030B,	K8 = 662040B)

** ASSORTED CONSTANTS

CON(R4 =7373737373737373B, R16=0070000000000000B)

* TAPE WRITING ROUTINE

```

1Y
1ARG
2ARG
3ARG
4ARG
SLJ(N)
ZRO(O)
ZRO(O)
ZRO(O)
ZRO(O)
LDA7(1ARG)
SIUT(21Y)
SLJ(P)
ENA(T40)
INA(11B)
LIL(KO)
ADD7(4ARG)
SAU(9Y)
SAL(2Y)
SAL(8Y)
INA(3)
SAU(11Y)
SAU(6Y)
INA(1)
SAU(7Y)
LRS(15)
ENI(O)

```

SLJ(L+5)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
INA(-11B)
AJP3(L+3)
ZRO(0)
ZRO(4) G)
STA(K0)
LDA1(K0)
SAU(2Y)
SCL(777B)
SAL(4Y)
SAL(10Y)
SAU(5Y)
INA(2)
SAU(12Y)
SAU(8Y)
SAU(13Y)
SAL(1YB)
ENQ(0)

```

EXIT/ENTRY
TAPE UNIT NUMBER ARG.
INITIAL ADDRESS
TERMINAL ADDRESS
MODE, 1 = BINARY, 2 = BCD
CHECK FOR TU ASSIGNMENT
GO PRINT ERROR
SET TU CONTROL CODES
INITIALIZE

```

[illegible]

1YB	LLS(39)	LDQ(R15)
	SSU(KO)	LDA(4Y)
	LDA(10Y)	SSU(KO)
	LDA7(3ARG)	SIL2(21Y)
	INA(-1)	SAL(N)
	SAU(15YB)	SAU(14YB)
	LDA7(2ARG)	SAL(20YB)
	SAU(10Y)	SAU(4Y)
	SAL(15YA)	SAU(14YA)
2Y	EXF(N)	SAL(20YA)
4Y	EXF7(N)	EXF7(N)
5Y	EXF7(N)	SLJ(17Y)
6Y	EXF7(N)	SLJ(19Y)
7Y	EXF7(N)	SLJ(21Y)
	SLJ(16Y)	ZRO(O)
8Y	EXF(N)	EXF7(N)
9Y	EXF(N)	SLJ(13YA)
10Y	EXF(N)	EXF7(N)
11Y	EXF7(N)	SLJ(20YA)
12Y	EXF7(N)	SLJ(20YA)
13Y	EXF7(N)	SLJ(20YA)
	SLJ(16Y)	ZRO(O)
13YA	IJP2(L+1)	SLJ(2Y)
14YA	LDA(N)	STA(T)
14YB	LDA(N)	STA(T+1)
15YA	LDA(R4)	STA(N)
15YB	STA(N)	SLJ(10Y)
16Y	ENA(T21)	SLJ4(3)
	AJP(2Y)	ENI2(O)
17Y	AJP3(21Y)	SLJ(8Y)
18Y	ENA(T18)	SLJ4(3)
	AJP3(21Y)	ENI2(1)
	AJP(8Y)	ENI2(O)
19Y	SLJ(8Y)	ZRO(O)
	ENA(T15)	SLJ4(3)
20YA	SLJ(18Y)	ZRO(O)
20YB	LDA(T)	STA(N)
	LDA(N)	STA(N)
21Y	SLJ(2Y)	ZRO(O)
	ENI1(N)	ENI2(N)
	SLJ(1Y)	ZRO(O)
	END	

INSERT CHANNEL SELECTION

SELECT UNIT
 ACTIVE CHANNEL
 SENSE PARITY ERROR
 SENSE LENGTH ERROR
 SENSE END OF TAPE

BACKSPACE
 SELECT UNIT
 WRITE BAD RECORD INDICATION
 SENSE PARITY ERROR
 SENSE LENGTH ERROR
 SENSE END OF TAPE

SET BAD RECORD INDICATION

TROUBLE EXIT
 SENSE ACTION TO TAKE

TROUBLE EXIT
 SENSE ACTION TO TAKE

TROUBLE EXIT

RESET INIT AND FINAL CELLS

RE-STORE INDEXES

06850
 06860
 06870
 06880
 06890
 06900
 06910
 06920
 06930
 06940
 06950
 06960
 06970
 06980
 06990
 07000
 07010
 07020
 06690
 07040
 07050
 07060
 07070
 07080
 06790
 07090
 07100
 07110
 06780
 07140
 07150
 07160
 07170
 07180
 07190
 07200
 07210
 07220
 07230
 07240
 07250
 07260
 07270
 07280
 07290

* FORTSAT * FORTSAT * FORTSAT * FORTSAT * FORTSAT * FORTSAT * FORTSAT *

* USNPS FORTRAN SYSTEM RESIDENT PROGRAM 25 SEPT 1962

* FORTSAT IS A 5K BIAS VERSION OF FORTRAN SIXTY
* WITH USNPGS SATELLITE SYSTEM IMPLEMENTATION

* (CDC VERSION 15 MARCH 1962 -- MODIFIED)
* LAST UPDATE -- 20 APRIL 1963
* R.L. HOGG / D. C. GLOVER

I/O DESIGNATORS

T - KEYBOARD OR TYPEWRITER (01)
F - FLEX PAPER TAPE (02)
P - BCD PAPER TAPE (INPUT) (03)
C - 521 READER (10) PUNCH (12)
1 - TU1 CH 3-4 (21)
2 - TU2 CH 3-4 (22)
3 - TU3 CH 3-4 (23)
4 - TU4 CH 3-4 (24)
5 - TU5 CH 5-6 (31)
6 - TU6 CH 5-6 (32)
7 - TU7 CH 5-6 (33)
8 - TU8 CH 5-6 (34)
9 - TYPEWRITER (01)
P - TYPEWRITER (01)
M - MEMORY (06)

RESIDENT ROUTINES (NO CALL REQUIRED)

CALL -	(TU NO.)	(UP TO EIGHT PROGRAM NAMES)
CONTROL -	(CONTROL MEDIUM)	(CONTROL STATEMENT OUTPUT MEDIUM)
REWIND -	(TU NO.)	
BKSP -	(TU NO.)	
TIME -	(NO ARGS)	TYPES OUT TIME
LIMIT -	(MINUTES)	(SECONDS)
CLOCK -	(NO ARGS)	STARTS CLOCK
EFMARK -	(TU NO.)	THIS IS 17 CODE
RUN -	(NO ARGS)	EXECUTES LAST PROGRAM CALLED
HOLD -	(NO ARGS)	EFFECTIVE FOR NEXT CALL ONLY

```

* * * * *
CLEAR - (NO ARGS) CLEARS MEMORY FROM
RESIDENT BIAS TO CELL 77776
SWITCH - (UP TO EIGHT SETTINGS) NOTE - FIRST THREE
ARE ORDERED WITH PHYSICAL SETTING
OF CONSOLE KEYS
INPUT - (MNEMONIC EQPT CODE) (BIOCTAL EQPT CODE)
OUTPUT - (MNEMONIC EQPT CODE) (BIOCTAL EQPT CODE)
STOPCK - (NO ARGS)
SET - (STARTING ADDRESS M-FILE) (RECORD SIZE
IN WORDS)
SKIP - (MEDIUM) (I.D. CODE)
LOCK - (TU NO.)
MONITOR - (INPUT) (OUTPUT) (MODE) (TIME LIMIT)
SR - (ALT OUTPUT) (SOURCE LISTING) (MAP LISTING)
* * * * *
(NO ARGS) THIS IS A SERVICE ROUTINE CALL
WHICH CALLS THE FOLLOWING LIB ROUTINES,
COPIES, TRANSFER, VFYLIB, LIST,
TROUBLE, READMT, WRITEMT, LIBFIX

```

BOOTSTRAP FROM CHANNEL 3-4
CURRENT LIBRARY LISTING FOLLOWS RESIDENT LISTING

S1 MACHINE RESIDENT

```

* RSV(I=11, BUF65=9, SAT3UF=1)
LOC(Z=0, J=26, C=60, R=100, W=300, M=531, S=5000)
TO CHANGE RESIDENT BIAS, SET S=NEW VALUE

```

* BCD CODE TABLE

```

CON(F0 =128, F1 =018, F2 =028, F3 =038, F4 =048, F5 =058, F6 =068,
* F7 =078, F8 =108, F9 =118, F10 =128, F11 =138, F12 =148, F13 =158, F14 =168,
* F15 =178, F16 =188, F17 =198, F18 =208, F19 =218, F20 =228, F21 =238, F22 =248, F23 =258, F24 =268,
* F25 =278, F26 =288, F27 =298, F28 =308, F29 =318, F30 =328, F31 =338, F32 =348, F33 =358, F34 =368,
* F35 =378, F36 =388, F37 =398, F38 =408, F39 =418, F40 =428, F41 =438, F42 =448, F43 =458, F44 =468,
* F45 =478, F46 =488, F47 =498, F48 =508, F49 =518, F50 =528, F51 =538, F52 =548, F53 =558, F54 =568, F55 =578,
* F56 =588, F57 =598, F58 =608, F59 =618, F60 =628, F61 =638, F62 =648, F63 =658, F64 =668, F65 =678,
* F66 =688, F67 =698, F68 =708, F69 =718, F70 =728, F71 =738, F72 =748, F73 =758, F74 =768, F75 =778,
* F76 =788, F77 =798, F78 =808, F79 =818, F80 =828, F81 =838, F82 =848, F83 =858, F84 =868, F85 =878,
* F86 =888, F87 =898, F88 =908, F89 =918, F90 =928, F91 =938, F92 =948, F93 =958, F94 =968, F95 =978,
* F96 =988, F97 =998, F98 =1008, F99 =1018, F100 =1028, F101 =1038, F102 =1048, F103 =1058, F104 =1068,
* F105 =1078, F106 =1088, F107 =1098, F108 =1108, F109 =1118, F110 =1128, F111 =1138, F112 =1148, F113 =1158,
* F114 =1168, F115 =1178, F116 =1188, F117 =1198, F118 =1208, F119 =1218, F120 =1228, F121 =1238, F122 =1248,
* F123 =1258, F124 =1268, F125 =1278, F126 =1288, F127 =1298, F128 =1308, F129 =1318, F130 =1328, F131 =1338,
* F132 =1348, F133 =1358, F134 =1368, F135 =1378, F136 =1388, F137 =1398, F138 =1408, F139 =1418, F140 =1428,
* F141 =1438, F142 =1448, F143 =1458, F144 =1468, F145 =1478, F146 =1488, F147 =1498, F148 =1508, F149 =1518,
* F150 =1528, F151 =1538, F152 =1548, F153 =1558, F154 =1568, F155 =1578, F156 =1588, F157 =1598, F158 =1608,
* F159 =1618, F160 =1628, F161 =1638, F162 =1648, F163 =1658, F164 =1668, F165 =1678, F166 =1688, F167 =1698,
* F168 =1708, F169 =1718, F170 =1728, F171 =1738, F172 =1748, F173 =1758, F174 =1768, F175 =1778, F176 =1788,
* F177 =1798, F178 =1808, F179 =1818, F180 =1828, F181 =1838, F182 =1848, F183 =1858, F184 =1868, F185 =1878,
* F186 =1888, F187 =1898, F188 =1908, F189 =1918, F190 =1928, F191 =1938, F192 =1948, F193 =1958, F194 =1968,
* F195 =1978, F196 =1988, F197 =1998, F198 =2008, F199 =2018, F200 =2028, F201 =2038, F202 =2048, F203 =2058,
* F204 =2068, F205 =2078, F206 =2088, F207 =2098, F208 =2108, F209 =2118, F210 =2128, F211 =2138, F212 =2148,
* F213 =2158, F214 =2168, F215 =2178, F216 =2188, F217 =2198, F218 =2208, F219 =2218, F220 =2228, F221 =2238,
* F222 =2248, F223 =2258, F224 =2268, F225 =2278, F226 =2288, F227 =2298, F228 =2308, F229 =2318, F230 =2328,
* F231 =2338, F232 =2348, F233 =2358, F234 =2368, F235 =2378, F236 =2388, F237 =2398, F238 =2408, F239 =2418,
* F240 =2428, F241 =2438, F242 =2448, F243 =2458, F244 =2468, F245 =2478, F246 =2488, F247 =2498, F248 =2508,
* F249 =2518, F250 =2528, F251 =2538, F252 =2548, F253 =2558, F254 =2568, F255 =2578, F256 =2588, F257 =2598,
* F258 =2608, F259 =2618, F260 =2628, F261 =2638, F262 =2648, F263 =2658, F264 =2668, F265 =2678, F266 =2688,
* F267 =2698, F268 =2708, F269 =2718, F270 =2728, F271 =2738, F272 =2748, F273 =2758, F274 =2768, F275 =2778,
* F276 =2788, F277 =2798, F278 =2808, F279 =2818, F280 =2828, F281 =2838, F282 =2848, F283 =2858, F284 =2868,
* F285 =2878, F286 =2888, F287 =2898, F288 =2908, F289 =2918, F290 =2928, F291 =2938, F292 =2948, F293 =2958,
* F294 =2968, F295 =2978, F296 =2988, F297 =2998, F298 =3008, F299 =3018, F300 =3028, F301 =3038, F302 =3048,
* F303 =3058, F304 =3068, F305 =3078, F306 =3088, F307 =3098, F308 =3108, F309 =3118, F310 =3128, F311 =3138,
* F312 =3148, F313 =3158, F314 =3168, F315 =3178, F316 =3188, F317 =3198, F318 =3208, F319 =3218, F320 =3228,
* F321 =3238, F322 =3248, F323 =3258, F324 =3268, F325 =3278, F326 =3288, F327 =3298, F328 =3308, F329 =3318,
* F330 =3328, F331 =3338, F332 =3348, F333 =3358, F334 =3368, F335 =3378, F336 =3388, F337 =3398, F338 =3408,
* F339 =3418, F340 =3428, F341 =3438, F342 =3448, F343 =3458, F344 =3468, F345 =3478, F346 =3488, F347 =3498,
* F348 =3508, F349 =3518, F350 =3528, F351 =3538, F352 =3548, F353 =3558, F354 =3568, F355 =3578, F356 =3588,
* F357 =3598, F358 =3608, F359 =3618, F360 =3628, F361 =3638, F362 =3648, F363 =3658, F364 =3668, F365 =3678,
* F366 =3688, F367 =3698, F368 =3708, F369 =3718, F370 =3728, F371 =3738, F372 =3748, F373 =3758, F374 =3768,
* F375 =3778, F376 =3788, F377 =3798, F378 =3808, F379 =3818, F380 =3828, F381 =3838, F382 =3848, F383 =3858,
* F384 =3868, F385 =3878, F386 =3888, F387 =3898, F388 =3908, F389 =3918, F390 =3928, F391 =3938, F392 =3948,
* F393 =3958, F394 =3968, F395 =3978, F396 =3988, F397 =3998, F398 =4008, F399 =4018, F400 =4028, F401 =4038,
* F402 =4048, F403 =4058, F404 =4068, F405 =4078, F406 =4088, F407 =4098, F408 =4108, F409 =4118, F410 =4128,
* F411 =4138, F412 =4148, F413 =4158, F414 =4168, F415 =4178, F416 =4188, F417 =4198, F418 =4208, F419 =4218,
* F420 =4228, F421 =4238, F422 =4248, F423 =4258, F424 =4268, F425 =4278, F426 =4288, F427 =4298, F428 =4308,
* F429 =4318, F430 =4328, F431 =4338, F432 =4348, F433 =4358, F434 =4368, F435 =4378, F436 =4388, F437 =4398,
* F438 =4408, F439 =4418, F440 =4428, F441 =4438, F442 =4448, F443 =4458, F444 =4468, F445 =4478, F446 =4488,
* F447 =4498, F448 =4508, F449 =4518, F450 =4528, F451 =4538, F452 =4548, F453 =4558, F454 =4568, F455 =4578,
* F456 =4588, F457 =4598, F458 =4608, F459 =4618, F460 =4628, F461 =4638, F462 =4648, F463 =4658, F464 =4668,
* F465 =4678, F466 =4688, F467 =4698, F468 =4708, F469 =4718, F470 =4728, F471 =4738, F472 =4748, F473 =4758,
* F474 =4768, F475 =4778, F476 =4788, F477 =4798, F478 =4808, F479 =4818, F480 =4828, F481 =4838, F482 =4848,
* F483 =4858, F484 =4868, F485 =4878, F486 =4888, F487 =4898, F488 =4908, F489 =4918, F490 =4928, F491 =4938,
* F492 =4948, F493 =4958, F494 =4968, F495 =4978, F496 =4988, F497 =4998, F498 =5008, F499 =5018, F500 =5028,
* F501 =5038, F502 =5048, F503 =5058, F504 =5068, F505 =5078, F506 =5088, F507 =5098, F508 =5108, F509 =5118,
* F510 =5128, F511 =5138, F512 =5148, F513 =5158, F514 =5168, F515 =5178, F516 =5188, F517 =5198, F518 =5208,
* F519 =5218, F520 =5228, F521 =5238, F522 =5248, F523 =5258, F524 =5268, F525 =5278, F526 =5288, F527 =5298,
* F528 =5308, F529 =5318, F530 =5328, F531 =5338, F532 =5348, F533 =5358, F534 =5368, F535 =5378, F536 =5388,
* F537 =5398, F538 =5408, F539 =5418, F540 =5428, F541 =5438, F542 =5448, F543 =5458, F544 =5468, F545 =5478,
* F546 =5488, F547 =5498, F548 =5508, F549 =5518, F550 =5528, F551 =5538, F552 =5548, F553 =5558, F554 =5568,
* F555 =5578, F556 =5588, F557 =5598, F558 =5608, F559 =5618, F560 =5628, F561 =5638, F562 =5648, F563 =5658,
* F564 =5668, F565 =5678, F566 =5688, F567 =5698, F568 =5708, F569 =5718, F570 =5728, F571 =5738, F572 =5748,
* F573 =5758, F574 =5768, F575 =5778, F576 =5788, F577 =5798, F578 =5808, F579 =5818, F580 =5828, F581 =5838,
* F582 =5848, F583 =5858, F584 =5868, F585 =5878, F586 =5888, F587 =5898, F588 =5908, F589 =5918, F590 =5928,
* F591 =5938, F592 =5948, F593 =5958, F594 =5968, F595 =5978, F596 =5988, F597 =5998, F598 =6008, F599 =6018,
* F600 =6028, F601 =6038, F602 =6048, F603 =6058, F604 =6068, F605 =6078, F606 =6088, F607 =6098, F608 =6108,
* F609 =6118, F610 =6128, F611 =6138, F612 =6148, F613 =6158, F614 =6168, F615 =6178, F616 =6188, F617 =6198,
* F618 =6208, F619 =6218, F620 =6228, F621 =6238, F622 =6248, F623 =6258, F624 =6268, F625 =6278, F626 =6288,
* F627 =6298, F628 =6308, F629 =6318, F630 =6328, F631 =6338, F632 =6348, F633 =6358, F634 =6368, F635 =6378,
* F636 =6388, F637 =6398, F638 =6408, F639 =6418, F640 =6428, F641 =6438, F642 =6448, F643 =6458, F644 =6468,
* F645 =6478, F646 =6488, F647 =6498, F648 =6508, F649 =6518, F650 =6528, F651 =6538, F652 =6548, F653 =6558,
* F654 =6568, F655 =6578, F656 =6588, F657 =6598, F658 =6608, F659 =6618, F660 =6628, F661 =6638, F662 =6648,
* F663 =6658, F664 =6668, F665 =6678, F666 =6688, F667 =6698, F668 =6708, F669 =6718, F670 =6728, F671 =6738,
* F672 =6748, F673 =6758, F674 =6768, F675 =6778, F676 =6788, F677 =6798, F678 =6808, F679 =6818, F680 =6828,
* F681 =6838, F682 =6848, F683 =6858, F684 =6868, F685 =6878, F686 =6888, F687 =6898, F688 =6908, F689 =6918,
* F690 =6928, F691 =6938, F692 =6948, F693 =6958, F694 =6968, F695 =6978, F696 =6988, F697 =6998, F698 =7008,
* F699 =7018, F700 =7028, F701 =7038, F702 =7048, F703 =7058, F704 =7068, F705 =7078, F706 =7088, F707 =7098,
* F708 =7108, F709 =7118, F710 =7128, F711 =7138, F712 =7148, F713 =7158, F714 =7168, F715 =7178, F716 =7188,
* F717 =7198, F718 =7208, F719 =7218, F720 =7228, F721 =7238, F722 =7248, F723 =7258, F724 =7268, F725 =7278,
* F726 =7288, F727 =7298, F728 =7308, F729 =7318, F730 =7328, F731 =7338, F732 =7348, F733 =7358, F734 =7368,
* F735 =7378, F736 =7388, F737 =7398, F738 =7408, F739 =7418, F740 =7428, F741 =7438, F742 =7448, F743 =7458,
* F744 =7468, F745 =7478, F746 =7488, F747 =7498, F748 =7508, F749 =7518, F750 =7528, F751 =7538, F752 =7548,
* F753 =7558, F754 =7568, F755 =7578, F756 =7588, F757 =7598, F758 =7608, F759 =7618, F760 =7628, F761 =7638,
* F762 =7648, F763 =7658, F764 =7668, F765 =7678, F766 =7688, F767 =7698, F768 =7708, F769 =7718, F770 =7728,
* F771 =7738, F772 =7748, F773 =7758, F774 =7768, F775 =7778, F776 =7788, F777 =7798, F778 =7808, F779 =7818,
* F780 =7828, F781 =7838, F782 =7848, F783 =7858, F784 =7868, F785 =7878, F786 =7888, F787 =7898, F788 =7908,
* F789 =7918, F790 =7928, F791 =7938, F792 =7948, F793 =7958, F794 =7968, F795 =7978, F796 =7988, F797 =7998,
* F798 =8008, F799 =8018, F800 =8028, F801 =8038, F802 =8048, F803 =8058, F804 =8068, F805 =8078, F806 =8088,
* F807 =8098, F808 =8108, F809 =8118, F810 =8128, F811 =8138, F812 =8148, F813 =8158, F814 =8168, F815 =8178,
* F816 =8188, F817 =8198, F818 =8208, F819 =8218, F820 =8228, F821 =8238, F822 =8248, F823 =8258, F824 =8268,
* F825 =8278, F826 =8288, F827 =8298, F828 =8308, F829 =8318, F830 =8328, F831 =8338, F832 =8348, F833 =8358,
* F834 =8368, F835 =8378, F836 =8388, F837 =8398, F838 =8408, F839 =8418, F840 =8428, F841 =8438, F842 =8448,
* F843 =8458, F844 =8468, F845 =8478, F846 =8488, F847 =8498, F848 =8508, F849 =8518, F850 =8528, F851 =8538,
* F852 =8548, F853 =8558, F854 =8568, F855 =8578, F856 =8588, F857 =8598, F858 =8608, F859 =8618, F860 =8628,
* F861 =8638, F862 =8648, F863 =8658, F864 =8668, F865 =8678, F866 =8688, F867 =8698, F868 =8708, F869 =8718,
* F870 =8728, F871 =8738, F872 =8748, F873 =8758, F874 =8768, F875 =8778, F876 =8788, F877 =8798, F878 =8808,
* F879 =8818, F880 =8828, F881 =8838, F882 =8848, F883 =8858, F884 =8868, F885 =8878, F886 =8888, F887 =8898,
* F888 =8908, F889 =8918, F890 =8928, F891 =8938, F892 =8948, F893 =8958, F894 =8968, F895 =8978, F896 =8988,
* F897 =8998, F898 =9008, F899 =9018, F900 =9028, F901 =9038, F902 =9048, F903 =9058, F904 =9068, F905 =9078,
* F906 =9088, F907 =9098, F908 =9108, F909 =9118, F910 =9128, F911 =9138, F912 =9148, F913 =9158, F914 =9168,
* F915 =9178, F916 =9188, F917 =9198, F918 =9208, F919 =9218, F920 =9228, F921 =9238, F922 =9248, F923 =9258,
* F924 =9268, F925 =9278, F926 =9288, F927 =9298, F928 =9308, F929 =9318, F930 =9328, F931 =9338, F932 =9348,
* F933 =9358, F934 =9368, F935 =9378, F936 =9388, F937 =9398, F938 =9408, F939 =9418, F940 =9428, F941 =9438,
* F942 =9448, F943 =9458, F944 =9468, F945 =9478, F946 =9488, F947 =9498, F948 =9508, F949 =9518, F950 =9528,
* F951 =9538, F952 =9548, F953 =9558, F954 =9568, F955 =9578, F956 =9588, F957 =9598, F958 =9608, F959 =9618,
* F960 =9628, F961 =9638, F962 =9648, F963 =9658, F964 =9668, F965 =9678, F966 =9688, F967 =9698, F968 =9708,
* F969 =9718, F970 =9728, F971 =9738, F972 =9748, F973 =9758, F974 =9768, F975 =9778, F976 =9788, F977 =9798,
* F978 =9808, F979 =9818, F980 =9828, F981 =9838, F982 =9848, F983 =9858, F984 =9868, F985 =9878, F986 =9888,
* F987 =9898, F988 =9908, F989 =9918, F990 =9928, F991 =9938, F992 =9948, F993 =9958, F994 =9968, F995 =9978,
* F996 =9988, F997 =9998, F998 =10008, F999 =10018, F1000 =10028, F1001 =10038, F1002 =10048, F1003 =10058,
* F1004 =10068, F1005 =10078, F1006 =10088, F1007 =10098, F1008 =10108, F1009 =10118, F1010 =10128, F1011 =10138,
* F1012 =10148, F1013 =10158, F1014 =10168, F1015 =10178, F1016 =10188, F1017 =10198, F1018 =10208, F1019 =10218,
* F1020 =10228, F1021 =10238, F1022 =10248, F1023 =10258, F1024 =10268, F1025 =10278, F1026 =10288, F1027 =10298,
* F1028 =10308, F1029 =10318, F1030 =10328, F1031 =10338, F1032 =10348, F1033 =10358, F1034 =10368, F1035 =10378,
* F1036 =10388, F1037 =10398, F1038 =10408, F1039 =10418, F1040 =10428, F1041 =10438, F1042 =10448, F1043 =10458,
* F1044 =10468, F1045 =10478, F1046 =10488, F1047 =10498, F1048 =10508, F1049 =10518, F1050 =10528, F1051 =10538,
* F1052 =10548, F1053 =10558, F1054 =10568, F1055 =10578, F1056 =10588, F1057 =10598, F1058 =10608, F1059 =10618,
* F1060 =10628, F1061 =10638, F1062 =10648, F1063 =10658, F1064 =10668, F1065 =10678, F1066 =10688, F1067 =10698,
* F1068 =10708, F1069 =10718, F1070 =10728, F1071 =10738, F1072 =10748, F1073 =10758, F1074 =10768, F1075 =10778,
* F1076 =10788, F1077 =10798, F1078 =10808, F1079 =10818, F1080 =10828, F1081 =10838, F1082 =10848, F1083 =10858,
* F1084 =10868, F1085 =10878, F1086 =10888, F1087 =10898, F1088 =10908, F1089 =10918, F1090 =10928, F1091 =10938,
* F1092 =10948, F1093 =10958, F1094 =10968, F1095 =10978, F1096 =10988, F1097 =10998, F1098 =11008, F1099 =11018,
* F1100 =11028, F1101 =11038, F1102 =11048, F1103 =11058, F1104 =11068, F1105 =11078, F1106 =11088, F1107 =11098,
* F1108 =11108, F1109 =11118, F1110 =11128, F1111 =11138, F1112 =11148, F1113 =11158, F1114 =11168, F1115 =11178,
* F1116 =11188, F1117 =11198, F1118 =11208, F1119 =11218, F1120 =11228, F1121 =11238, F1122 =11248, F1123 =11258,
* F1124 =11268, F1125 =11278, F1126 =11288, F1127 =11298, F1128 =11308, F1129 =11318, F1130 =11328, F1131 =11338,
* F1132 =11348, F1133 =11358, F1134 =11368, F1135 =11378, F1136 =11388, F1137 =11398, F1138 =11408, F1139 =11418,
* F1140 =11428, F1141 =11438, F1142 =11448, F1143 =11458, F1144 =11468, F1145 =11478, F1146
```

* * *

1

✱

* * * * *

6TH CHAR.	7TH CHAR.	8TH CHAR.	=	TYPE	LOWER	UPPER	CASE	INPUT	CONVERSION
CHAR.	CHAR.	CHAR.		FLEX	LOWER	UPPER	CASE	INPUT	CONVERSION
CON(K0	77467756	77467756	22020202	02020202	02020202	02020202	1	07400742	32323232
K2	=00700070	02202020	22020202	02020202	02020202	02020202	K3	=00640066	46464646
K4	=00620062	02202020	45454545	45454545	45454545	45454545	K5	=00660066	70707070
K6	=00720072	02202020	45454545	45454545	45454545	45454545	K7	=00600060	44444444
K10	=00330033	02202020	02020202	02020202	02020202	02020202	K11	1=00370037	44343434
K12	=00560056	17171717	15151515	15151515	15151515	15151515	K13	=00370037	44343434
K14	=00520052	05202020	17171717	17171717	17171717	17171717	K15	=00027742	67676767
K16	=77467756	66363636	36363636	36363636	36363636	36363636	K17	=77467756	52525252
CON(K20	=00040004	04656565	56565656	56565656	56565656	56565656	K21	1=00440044	31313131
K22	=00240024	04646464	64646464	64646464	64646464	64646464	K23	=00010001	62626262
K24	=00340034	02222222	22222222	22222222	22222222	22222222	K25	=00170017	30303030
K26	=00310031	06666666	66666666	66666666	66666666	66666666	K27	=00270027	32727272
K30	=00250025	61616161	16161616	16161616	16161616	16161616	K31	1=00210021	26262626
K32	=77467756	41414141	14141414	14141414	14141414	14141414	K33	=00400040	65050505
K34	=77467756	42424242	24242424	24242424	24242424	24242424	K35	=77467756	34111111
K36	=77467756	43333333	33333333	33333333	33333333	33333333	K37	=77467756	34111111
CON(K40	=00520052	62737373	73737373	73737373	73737373	73737373	K41	1=00320032	20202020
K42	=00360036	72202154	21542154	21542154	21542154	21542154	K43	=00110011	20202020
K44	=00070007	22060603	06030603	06030603	06030603	06030603	K45	=00060006	20202020
K46	=00030003	52020532	05320532	05320532	05320532	05320532	K47	=00150015	20202020
K50	=00350035	62204034	04034034	04034034	04034034	04034034	K51	=00120012	20202020
K52	=77467756	22020202	02020202	02020202	02020202	02020202	K53	=77667756	20202020
K54	=77467756	22020202	02020202	02020202	02020202	02020202	K55	=77667756	20202020
K56	=77467756	22020202	02020202	02020202	02020202	02020202	K57	=77467756	20202020
CON(K60	=00467746	23504033	04033033	04033033	04033033	04033033	K61	=00300030	20202020
K62	=00220022	33030303	03030303	03030303	03030303	03030303	K63	=00160016	20202020
K64	=00260026	22020202	02020202	02020202	02020202	02020202	K65	=00200020	20202020
K66	=00260026	22020202	02020202	02020202	02020202	02020202	K67	=00130013	20202020
K70	=00050005	22020202	02020202	02020202	02020202	02020202	K71	=00140014	20202020
K72	=77467756	45454545	45454545	45454545	45454545	45454545	K73	=00420042	20202020
K74	=77560054	62020202	02020202	02020202	02020202	02020202	K75	=77467756	20202020
K76	=77467756	62020202	02020202	02020202	02020202	02020202	K77	=77	

```

CON(P0 = 4446457123465100B,
      P2 = 7145472423000000B,
      P4 = 63346451464300B,
      P6 = 636143300000000B,
      P10 = 51244500000000B,
      P12 = 5124456151000000B,
      P16 = 22652300000000B,
      P20 = 437147123000000B,

```

P1	=	56243	56472	56400	0000B
P3	=	46223	46472	46400	0000B
P5	=	22251	22647	22600	0000B
P7	=	22251	22647	22600	0000B
P11	=	22251	22647	22600	0000B
P13	=	70462	71465	70000	0000B
P15	=	22371	24650	22300	0000B
P21	=	51652	52674	51600	0000B

[illegible][illegible]

* BCD MESSAGE CODE TABLE

```

CON(EE240=751466751614420B,
      EE240=456400000000000B,
      EE240=436124712300000B,
      EE240=436162436540000B,
      EE240=000000000000000B,
      EE240=0564207145472745B,
      EE240=2342046520461448B,
      EE240=2371446204614530B,
      EE240=232420000000000B,
      EE240=2042265634645522B,
      EE240=2022656346455422B)
CON(EE124=0=2342046520461448B,
      EE124=0=2371446204614530B,
      EE124=0=232420000000000B,
      EE124=0=2042265634645522B,
      EE124=0=2022656346455422B)

```

[illegible]

* TYPEWRITER MESSAGE CODE TABLE

CON(T)	T0	T2	T4	T6	T8	T10	T12	T14	T16	T18	T22
CON(T0)	=450404	160520	1636B								
T2	=031220	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
T4	=122030	212200	402030	220121	2048B						
T6	=450404	120000	000000	000000	000000	000000	000000	000000	000000	000000	000000
T8	=121203	411200	000000	000000	000000	000000	000000	000000	000000	000000	000000
T10	=153012	114012	504208B								
T12	=450404	120000	000000	000000	000000	000000	000000	000000	000000	000000	000000
T14	=141120	061301	05348B								
T16	=041203	112140	01208B								
T18	=450404	120000	000000	000000	000000	000000	000000	000000	000000	000000	000000
T22	=042012	220403	60348B								

[illegible]

★ SERVICE ROUTINES CALLED BY SR

```

* * *
CON(COPYS=6346473022000000, TRANSFER=2351614522666551B,
VFYLIB=2566304371620000B, LIST=4371222300000000B,
TROUBLE=2351462462436500B, READMT=5165616444230000B,
WRITEMT=2651712365442300B, LIBFIX=4371626671270000B)

```

* 160 CALL ROUTINE CONSTANTS

```

CON(IDENT=000000001231111B). RS-022 IDENT
CON(IDEN2=000000001231122B). OSAP IDENT
CON(IDEN3=000000001231133B). OSAS IDENT
CON(IDEN4=000000001231144B). OSAS LOADER, A4.01 IDENT
CON(IDEN5=000000001231155B). OSAP LOADER
CON(E=0, E=0, E=0, E=0)
CON(SIZE=77B, SIZE2=773B, SIZE3=1214B, SIZE4=77B, SIZE5=23B)
CON(E=0, E=0, E=0, E=0)
CON(TAPE1=000000000332011B)
CON(BLOCKS=700B). LIMIT ON 160 LIB CALL BLOCK READ BEFORE EXIT

```

* TAPE CHANNEL ASSIGNMENT TABLE

```

CON(U0 =0033200000042000B, U1 =0055200000062000B)

```

* ASSORTED CONSTANTS

```

CON(R0 =01, R1 =10, R2 =100, R3 =6646512351614540B,
* R4 =7373737373737373B, R5 =4500000000000000B,
* R6 =2020202020202020B, R7 =7760000000000000B,
* R10=6545646671436500B, R11=0000000100000000B,
* R12=3777777777777777B, R13=0351244513000000B,
* R14=0041254400000000B, R15=6646512362714500B,
* R16=0070000000000000B, R17=5165446151420000B)

```

* DD65 PRINTER SIMULATOR CONSTANTS

```

CON(DDSWT=5252525252525252B)
CON(ND65R=1000B)

```

* PSUEDO PRINTER DRUM DESIGNATOR TABLE

```

CON(ITAB=34000000006001400B,
1 LTAB=3400000006241420B,
2 LTAB=3400000006501440B,
3 LTAB=3400000006741460B,
4 LTAB=3400000007201500B,
5 LTAB=3400000007441520B,
6 LTAB=3400000007701540B,
7 LTAB=340000010141560B)
CON(JTAB=340000010401600B,
1 LTAB=340000010641620B,
2 LTAB=340000011101640B,

```

```

3  TAB=3400000011341660B,
4  LTAB=3400000011601700B,
5  LTAB=3400000012041720B,
6  LTAB=3400000012301740B,
7  LTAB=3400000012541760B)
  CON(KTAB=3400000013001000B,
1  LTAB=3400000013241020B,
2  LTAB=3400000013501040B,
3  LTAB=3400000013741060B,
4  LTAB=3400000014201100B,
5  LTAB=3400000014441120B,
6  LTAB=3400000014701140B,
7  LTAB=3400000015141160B)
  CON(MTAB=3400000015401200B,
1  LTAB=3400000015641220B,
2  LTAB=3400000016101240B,
3  LTAB=3400000016341260B,
4  LTAB=3400000016601300B,
5  LTAB=3400000017041320B,
6  LTAB=3400000017301340B,
7  LTAB=3400000017541360B)

```

* INITIAL START/EXITS AND ENTRIES TO RESIDENT ROUTINES

SLJ(N)	SLJ(L)	INTERRUPT EXIT/ENTRY	00000
SLJ(Z+40B)	ZRO(O)	JUMP TO CONTINUE LOAD	00001
SLJ(N)	SLJ(3A)	BCD READ EXIT/ENTRY	00002
SLJ(N)	SLJ(65DPY)	BCD WRITE EXIT/ENTRY	00003
SLJ(N)	SLJ(5A)	BIN READ EXIT/ENTRY	00004
SLJ(N)	SLJ(6A)	BIN WRITE EXIT/ENTRY	00005
SLJ(N)	SLJ(7A)	BACKSPACE EXIT/ENTRY	00006
SLJ(N)	SLJ(8A)	REWIND EXIT/ENTRY	00007
SLJ(N)	SLJ(9A)	ENDFILE EXIT/ENTRY	00010
EXF(62540B)	SLJ(Z+40B)	JUMP TO CONTINUE LOAD/RELOAD	00012
SLJ(N)	SLJ(9M)	NORMAL RETURN TO MONITOR	00013
SLJ(N)	SLJ(10A)	GO TO CHAIN CONNECTOR	00014
SLJ(N)	SLJ(1E)	GO TO INDICATE EQUIPMENT FAILURE	00015
SLJ(N)	SLJ(2E)	GO TO INDICATE PROGRAMMING ERROR	00016
SLJ(N)	SLJ(11A)	BCD MESSAGE EXIT/ENTRY	00017
SLJ(N)	SLJ(N)	SPARE OF PROGRAM START TABLE	00020
ZRO(O)	ZRO(OZ)	ADDR TO RESUME MONITOR CONTROL	00021
SLJ(2A)	ZRO(O)	JUMP TO RESUME MONITOR CONTROL	00022
SLJ(N)	SLJ(5GG)	TYPE MESSAGE/SENSE FOR ACTION	00023
SLJ(N)	SLJ(N)	SPARE	00024
SLJ(N)	SLJ(3VV)	GO LOCK TAPE	00025
SLJ(N)	SLJ(N)	SPARE	00026
EXF(32011B)	EXF(32005B)	REWIND LIBRARY TAPE	00027

ENI1(0)	EXF7(32000B)	•	REREAD FIRST RESIDENT BLOCK	00030
EXF3(Z)	EXF7(32000B)	•	SET JUMP TO PROGRAM CONTROL	00031
ENA(1PCON)	SAU(Z+10B)	•	SET JUMP TO RELOAD RESIDENT	00032
ENA(Z+35B)	SAL(Z+20B)	•		00033
ENI2(48)	ENI3(12)	•		00034
ENA(R+54)	STA(Z+3)	•		00035
EXF3(R)	EXF7(32000B)	•	READ RESIDENT BLOCK	00036
EXF3(R+54)	ENQ(0)	•		00037
EXF7(32003B)	SLJ(Z+35B)	•	SENSE PARITY ERROR	00040
EXF7(32005B)	SLJ(Z+35B)	•	SENSE LENGTH ERROR	00041
LDA1(R+6)	STA2(Z)	•	TRANSFER WORD TO RESIDENT AREA	00042
ISKF2(T+11)	SLJ(Z+53B)	•	SENSE END OF RESIDENT	00043
ISKF(32005B)	SLJ(Z+55B)	•		00044
ISK1(47)	SLJ(Z+50B)	•		00045
SLJ(Z+44B)	ZRO(0)	•	SENSE END OF BLOCK	00046
STQ3(Z+40B)	IJP3(Z+55B)	•	GO READ NEXT BLOCK	00047
STQ(Z+55B)	STQ(Z+56B)	•	CLEAR SIMULATION LOCATIONS	00050
STQ(Z)	SLJ(1PCON)	•	EXIT TO PROGRAM CONTROL	00051
ZRO(0)	ZRO(0)	•		00052
ZRO(0)	ZRO(0)	•		00053
ZRO(0)	ZRO(0)	•		00054
ZRO(0)	ZRO(0)	•		00055
ZRO(0)	ZRO(0)	•		00056
ZRO(0)	ZRO(0)	•		00057

* CONTROL INFORMATION AND MONITOR TAPE ASSIGNMENT TABLE

ZRO(0)	ZRO(0)	•	END OF FORTRAN MAP FILE	00060
ZRO(0)	ZRO(0)	•	START OF FORTRAN MAP FILE	00061
ZRO(0)	ZRO(S)	•	INITIAL PROGRAM BIAS	00062
ZRO(0)	ZRO(S)	•	CURRENT PROGRAM BIAS	00063
ZRO(0)	ZRO(J)	•	INITIAL NUMBER OF PROGRAMS	00064
ZRO(0)	ZRO(J)	•	CURRENT NUMBER OF PROGRAMS	00065
ZRO(0)	ZRO1(PO)	•	ADDRESS OF PROGRAM-NAME TABLE	00066
ZRO(0)	ZRO1(OZ)	•	ADDRESS OF PROGRAM-START TABLE	00067
ZRO(0)	ZRO(1)	•	MONITOR TAPE-ASSIGNMENT TABLE	00070
ZRO(0)	ZRO(2)	•		00071
ZRO(0)	ZRO(3)	•		00072
ZRO(0)	ZRO(4)	•		00073
ZRO(0)	ZRO(5)	•		00074
ZRO(0)	ZRO(6)	•		00075
ZRO(0)	ZRO(7)	•		00076
ZRO(0)	ZRO(8)	•		00077

* READ BUFFER

ZRO(0)	•	READ BUFFER	00100
--------	---	-------------	-------

.....

[illegible]

* WRITE BUFFER

ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0)
ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0) ZRQ(0)

ZRQ(ZRQ(ZRQ(ZRQ(ZRQ(ZRQ(ZRQ(ZRQ(ZRQ(ZRQ(

(O)(O)(O)(O)(O)(O)(O)(O)(O)(O)

WRITE BUFFER

• • • • •

[illegible]

[illegible][illegible]

[illegible]

)
 (0)
 ZRO
 ZRO
 ZRO
 ZRO
 ZRO
 ZRO

ZRO (0)
ZRO (0)
ZRO (0)
ZRO (0)
ZRO (0)
ZRO (0)

.....

• • • • •

1234567
3333333
5555555
0000000
0000000

123456701234567
1111111222222222
6666666666666666
0000000000000000
0000000000000000

STAKE TRANSIENT PROGRAM STARTS

[illegible]

006667	PRINT MEDIUM DESIGNATOR CELL
006670	SATELLITE MONITOR JOB SWITCH
006701	160 LIBRARY CALL LOCK-OUT
006722	TRIGGER ROUTINE
006733	FOR SATELLITE MONITOR
006745	160 SAT LIBRARY CALL ENTRY
006776	RELEASE DIRECT / TEST NO DD65
006777	TAKE READ/WRITE CONTROL
007000	SEL DD65
007001	SENSE DD65 AVAILABLE
007002	SENSE DD65 CORE
007003	CLEAR DD65
007004	LOOP UNTIL CLEAR
007005	GO TO PROGRAM CONTROL
007006	

ZRO(0)	ZRO(0)
ZRO(0)	ZRO(0)
ZRO(0)	ZRO(0)
ZRO(0)	ZRO(G47)
SEV7(77777B)	SEV7(77777B)
SEV7(77777B)	SEV7(77777B)
ZRO(0)	ZRO(0)
SLJ(N)	STA(OSWT)
SLJ(L-1)	ZRO(0)
SLJ(N)	SLJ(150CL)
EXF(62500B)	SLJ(L+6)
EXF(52502B)	EXF(62502B)
EXF(77010B)	ISK1(1000B)
EXF7(77010B)	SLJ(15)
ENI(0)	OUT(Z+40B)
ENI(1000B)	SLJ(L-1)
EXF(62560B)	SLJ(15)

*** TIME ADVANCE ROUTINE**

•	CLEAR FAULT	00707
•	SENSE ONE--SECOND OVERFLOW	00710
•	RESET ONE--SECOND COUNT	00711
•	INCREASE TIME AT 50	00712
•	SENSE LIMIT AT 57	00713
•	STOP THROUGH 7	00714
•	SET TO CLOCK	00715
•	SET TO ISSUE LIMIT INDICATION	00716
•	EXIT THROUGH 7	00717
•	REWIND LIBRARY TAPE	00720
•		00721
•	SET TO INDICATE LIMIT	00722
•	SENSE NO MONITOR CONTROL	00723
•	TYPE LIMIT INDICATION	00724
•	RETURN TO MONITOR	00725

```

1A
EXF(70B)
SSK(Z)
LDA(R12)
RAO(Z+50B)
AJP2(L+2)
SLJ(Z+7)
EXF(200B)
SAU(Z+7)
STA(Z+57B)
ENA(1)
SLJ4(1V)
ENA(E16)
SSK(2M)
SLJ4(6G)
EXF(1000B)

2A
STA(T+8)
SLJ(Z+7)
STA(Z)
SUB(Z+57B)
LDA(T+8)
ZRO(O)
ENA(L+3)
LDA(R11)
SLJ(Z+7)
SAL(2V)
ZRO(O)
ENI(O)
SLJ(2E)
ZRO(O)
SLJ(9M)

```

* BCD READ SWITCH

•	SENSE FULL MONITOR CONTROL	00726
•	SENSE PARTIAL MONITOR CONTROL	00727
•	GO READ BCD RECORD	00730
•		00731
•	SENSE CARD REQUEST	00732
•	SET TO READ CARD IMAGE	00733
•		00734
•	SENSE STACKED-JOB TAPE	00735
•	SENSE FIRST OR SECOND BANK	00736

3A

SSK(3M)	SLJ(L+6)
SSK(2M)	SLJ(L+3)
SLJ4(1R)	ZRO(0)
SLJJ(3AEX)	ZRO(0)
EQS(63B)	SLJ(L-2)
LDA(2M)	SLJ(L-3)
LDA(2M)	SEN2(0)
LDS(9)	SLJ(L+19)
IHS(5)	SLJ(L+2)

INA(3200B)
 INA(5174B)
 ALS(3)
 SAU(L+9)
 LDQ(R6)
 ENI(9)
 LDA1(M+39)
 IJP1(L-1)
 STQ1(R+10)
 EXF(62540B)
 ENQ(M+49)
 STQ(Z+5)
 EXF(N)
 EXF3(M+39)
 EXF(N)
 EXF5(M+39)
 ENI(0)
 EQS(63B)
 ENI(23)
 LDA1(M+39)
 IJP1(L-1)
 EXF(62540B)
 LDA1(R)
 AJP1(L+2)
 STA1(R)
 IJP1(L-3)
 ENQ(M+63)
 ENA(14001B)
 SAU(L+1)
 ENI(0)
 EXF(N)
 ENI(0)
 EXF(62560B)
 STA(1ERSW)

3AEX

* BCD WRITE SWITCH

4A
 SSK(3M)
 SSK(2M)
 EQS(63B)
 LDQ(G63)
 INA(-12B)
 ENA(63B)
 SLJ4(1W)
 SLJ(4AEX)
 EQS(2M)
 THS(9)
 ENA(27B)
 EQS(63B)

SLJ(L+2)
 ENI2(1)
 INA(2)
 SAU(L+11)
 EXF7(31B)
 EXF7(51B)
 STA1(R)
 ENI(4)
 IJP1(L)
 SLJ4(3G)
 STQ(Z+3)
 IJP2(L+3)
 EXF7(32000B)
 SLJ(3AEX)
 EXF7(52000B)
 EXF7(52000B)
 SLJ(3AEX)
 SLJ(L-24)
 EXF7(11B)
 STA1(R+80)
 SLJ4(3R)
 ENI1(79)
 INA(-14B)
 ENA(40B)
 ENI(0)
 ENI(0)
 STQ(Z+1)
 ADD(T+11)
 EXF7(24004B)
 EXF1(M+39)
 EXF7(24004B)
 ENA(0)
 SLJ(Z+11B)

SLJ(L+11)
 SLJ(L+7)
 SLJ(L+4)
 LDQ(77B)
 AJP(L+12)
 ENI(0)
 ZRO(0)
 ZRO(0)
 SLJ(L-2)
 SLJ(L-3)
 SLJ(L-4)
 SLJ(L+13)

00737
 00740
 00741
 00742
 00743
 00744
 00745
 00746
 00747
 00750
 00751
 00752
 00753
 00754
 00755
 00756
 00757
 00760
 00761
 00762
 00763
 00764
 00765
 00766
 00767
 00770
 00771
 00772
 00773
 00774
 00775
 00776
 00777

SET TO SELECT TAPE UNIT
 SENSE CHANNELS INACTIVE

TRANSFER FROM BUFFER

DISASSEMBLE READ BUFFER

READ NEXT BLOCK TO BUFFER

SENSE STACKED-JOB DECK
 SENSE CHANNEL INACTIVE
 TRANSFER FROM BUFFER
 GO PROCESS CARD IMAGE

SENSE ALTERNATE DASH

READ NEXT CARD TO BUFFER

RELEASE COMM FLAG 1
 CLEAR ERROR FLAG / EXIT VIA 7

01000
 01001
 01002
 01003
 01004
 01005
 01006
 01007
 01010
 01011
 01012
 01013

SENSE FULL MONITOR CONTROL
 SENSE PARTIAL MONITOR CONTROL
 SENSE CARD REQUEST

SENSE CARD-PUNCH REASSIGNMENT

GO WRITE BCD RECORD

SENSE STACKED-JOB MEDIUM

SENSE CARD REQUEST

LDA(6M)
 INA(-63B)
 ENA(0)
 LDA(6M)
 SLJ4(3W)
 ENI1(23)
 LDA1(W)
 IJP1(L-1)
 ENA(M+39)
 EXF(24004B)
 EXF2(M+15)
 ENI(0)
 LDA(3M)
 THS(9)
 THS(5)
 INA(4200B)
 INA(6174B)
 ALS(3)
 SAU(L+7)
 EXF(62540B)
 ENI1(14)
 ENQ(M+15)
 LDA1(W)
 IJP1(L-1)
 STQ(Z+6)
 EXF(N)
 EXF4(M)
 EXF(N)
 EXF6(M)
 EXF(62560B)
 SSK(OSWT)
 SLJ(Z+12B)
 SSK(1ERSW)
 LDA(3M)
 AJP(Z+12B)
 SAU(1W)

4AEX

* BIN READ SWITCH

5A SSK(2M)
 SLJ4(11)
 SLJ(Z+13B)

* BIN WRITE SWITCH

6A SSK(2M)
 SLJ4(10)

AJP(L+12)
 AJP(L+3)
 SLJ4(8G)
 SLJ(L-9)
 ZRO(0)
 EXF7(21B)
 STA1(M+15)
 SLJ4(1FLAG)
 EXF7(14004B)
 SAL(Z+2)
 EXF7(14004B)
 SLJ(4AEX)
 ENI2(0)
 SLJ(L-19)
 SLJ(L+2)
 SLJ(L+2)
 ENI2(1)
 INA(2)
 SAU(L+9)
 SLJ4(4G)
 EXF7(41B)
 EXF7(51B)
 STA1(M)
 STQ(Z+4)
 IJP2(L+3)
 EXF7(+2000B)
 SLJ(4AEX)
 EXF7(52000B)
 EXF7(62000B)
 ENI(0)
 SLJ(L+2)
 ZRO(0)
 SLJ(Z+12B)
 INA(-56B)
 ENA(Z+12B)
 SLJ(7W)

SLJ4(12A)
 ZRO(0)
 ZRO(0)

SLJ4(3G)
 ZRO(0)

SENSE PUNCH-CARD MEDIUM
 SENSE ON-LINE PUNCHING
 SENSE ALL CHANNELS INACTIVE

GO FORM CARD IMAGE
 SENSE CHANNEL INACTIVE
 TRANSFER TO BUFFER

PUNCH NEXT CARD FROM BUFFER

SENSE LISTABLE-OUTPUT TAPE
 SENSE FIRST OR SECOND BANK

SET TO SELECT TAPE UNIT
 ASSEMBLE WRITE BUFFER
 SENSE CHANNELS INACTIVE

TRANSFER TO BUFFER

WRITE NEXT BLOCK FROM BUFFER

ERRORS TO SATELLITE ROUTINE

SENSE NO MONITOR CONTROL
 GO READ BINARY RECORD
 EXIT THROUGH 13

SENSE NO MONITOR CONTROL
 GO WRITE BINARY RECORD

01014
 01015
 01016
 01017
 01020
 01021
 01022
 01023
 01024
 01025
 01026
 01027
 01030
 01031
 01032
 01033
 01034
 01035
 01036
 01037
 01040
 01041
 01042
 01043
 01044
 01045
 01046
 01047
 01050
 01051
 01052
 01053
 01054
 01055
 01056
 01057

01060
 01061
 01062

01063
 01064

SLJ(Z+14B)	ZRO(0)	• EXIT THROUGH 14	01065
* BACKSPACE SWITCH			
7A SSK(2M) SAL(8V) SLJ(Z+15B)	SLJ4(12A) SLJ4(7V) ZRO(0)	• SENSE NO MONITOR CONTROL • BACKSPACE • EXIT THROUGH 15	01066 01067 01070
* REWIND SWITCH			
8A SSK(2M) SAL(2V) SLJ(Z+16B)	SLJ4(8G) SLJ4(1V) ZRO(0)	• SENSE NO MONITOR CONTROL • REWIND • EXIT THROUGH 16	01071 01072 01073
* ENDFILE SWITCH			
9A SSK(2M) SAL(6V) SLJ(Z+17B)	SLJ4(12A) SLJ4(5V) ZRO(0)	• SENSE NO MONITOR CONTROL • ENDFILE • EXIT THROUGH 17	01074 01075 01076
* CHAIN CONNECTOR			
10A SAL(L+1) SAL(L+1) STA(3C) STA(2C) SLJ(1Q)	INA(1) LDA7(N) LDA7(N) SLJ4(1C) ZRO(0)	• SET TO PICK ARGUMENTS • TRANSFER ARGUMENTS • CALL NEXT LINK • GO EXECUTE ASSEMBLED PROGRAM	01077 01100 01101 01102 01103
* BCD MESSAGE SWITCH			
11A SSK(2M) SLJ4(6G) SLJ(Z+25B) SAU(L+1) LDA(N) AJP(Z+25B) SLJ(2E)	SLJ(L+3) ZRO(0) ZRO(0) SAL(L+2) SCM(R3) ENA(N) ZRO(0)	• SENSE NO MONITOR CONTROL • GO OUTPUT BCD MESSAGE • EXIT THROUGH 25 • SENSE FORTRAN-E • EXIT TO STACK MESSAGE	01104 01105 01106 01107 01110 01111 01112
* TAPE ASSIGNMENT CHECK ROUTINE			
12A SLJ(N) EQS(1) SLJ(12A)	ENI(3) SLJ(L+2) ZRO(0)	• SENSE LIBRARY TAPE	01113 01114 01115

ALS(24)
MEQ1(C+8)
ALS(24)
SLJ(L-6)
LIU1(L-7)
SIU1(L-8)

LQC(77777B)
SLJ(L+3)
SLJ4(8G)
ZRO(0)
INI1(1B)
SLJ(L-8)

01116
01117
01120
01121
01122
01123

* CALL ROUTINE

1C SLJ(N)
2C ZRO(0)
3C ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
LDQ(4M)
LDA(2C)
SLJ4(1V)
LDA(C+4)
LDA(C+2)
ENI1(7)
LDA1(3C)
LRS(6)
STQ1(3C)
LDA(2C)
LDA(R)
LDQ(3C)
IJP5(L+1)
QJPI(L+1)
SCM(R10)
LDA(R)
EQS1(3C)
LIL3(C+5)
LDQ(R+1)
STA(T)
LDL(77777B)
ADD(C+3)
AJP2(8C)
QLS(9)
LDL(777B)
ADD(C+3)
QLS(15)
ADD(C+3)
LDA(C+3)

SLJ(L+10)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
ZRO(0)
QJPI(L+3)
SAL(2V)
ZRO(0)
STA(C+5)
STA(C+3)
ENI5(1)
ENQ(0)
AJP1(L-2)
IJP1(L-2)
SLJ4(11)
AJP(L-1)
ENI(0)
SLJ(L+2)
STA(3)
AJP(9C)
ENI1(3)
SLJ(4C)
STA3(P0)
LDL(77777B)
QRS(24)
ADD(T)
SUB(77777B)
LDQ(R+1)
ENI(0)
ALS(24)
STA3(3Z)
LDL(77777B)
STA(T+7)
STA(T+3)

EXIT/ENTRY
LIBRARY MEDIUM
PROGRAM-NAME LIST

SENSE MONITOR CONTROL
REWIND LIBRARY MEDIUM
CLEAR TRANSIENT PROGRAMS
RESET PROGRAM BIAS

POSITION NAMES
LOOP
READ BINARY BLOCK

SENSE NO PROGRAM-NAME LIST
SENSE END OF LIBRARY

LOOP TO CALLED NAME
STORE PROGRAM NAME
PROCESS PARAMETERS

SENSE OVERFLOW OF MEMORY

01124
01125
01126
01127
01130
01131
01132
01133
01134
01135
01136
01137
01140
01141
01142
01143
01144
01145
01146
01147
01150
01151
01152
01153
01154
01155
01156
01157
01160
01161
01162
01163
01164
01165
01166
01167
01170
01171
01172

1E	SAU(L+1)	SLJ4(5G)	INDICATE SYSTEM FAILURE	01245
	ENA(N)	LDQ(3M)		01246
2E	QJP2(2E)	SLJ(10M)	SENSE FULL MONITOR CONTROL	01247
	AJP(10M)	LDQ(4K)	SENSE ERROR EXIT FROM COMPILER	01250
	QJP4(6G)	SLJ4(3E)	GO OUTPUT ERROR MESSAGE	01251
	EXF(1000B)	SLJ(10M)	EXIT TO MONITOR	01252

* OUTPUT PACKED BCD MESSAGE

3E	SLJ(N)	SAU(L+2)	EXIT/ENTRY CHANNELS INACTIVE	01253
	ENA(O)	SLJ4(3G)	GO SENSE	01254
	ENI1(N)	ENI2(O)	STRING OUT BCD CODE	01255
	ENA(20B)	STA(W)		01256
	ENA(O)	ENI3(O)		01257
	LDQ1(2)	LLS(6)		01260
	AJP(L+4)	STA2(W+1)		01261
	ENA(O)	INI2(1)		01262
	ISK3(7)	SLJ(L+6)		01263
	INI1(1)	SLJ(L-4)		01264
	ENA(20B)	STA2(W+1)		01265
	ISK2(119)	SLJ(L-1)	OUTPUT MESSAGE	01266
	LDA(4K)	SLJ4(1W)	EXIT	01270
	EXF(1000B)	SLJ(3E)		01271
	LLS(6)	SLJ(L-8)		

* INPUT INDEX REGISTER LOADER

1G	SLJ(N)	STA(T)	EXIT/ENTRY	01272
	LIL1(T)	LDQ1(30)	TABLE ITEMS TO INDEX REGISTERS	01273
	QRS(6)	LDL(73)		01274
	SAL(L+2)	QRS(3)		01275
	LDL(78)	SAU(L+1)		01276
	ENI1(N)	ENI2(V)		01277
	SLJ(L-6)	ZRD(O)	EXIT	01300

* OUTPUT INDEX REGISTER LOADER

2G	SLJ(N)	STA(T)	EXIT/ENTRY	01301
	LIL1(T)	LDQ1(30)	TABLE ITEMS TO INDEX REGISTERS	01302
	LDL(78)	SAL(L+2)		01303
	QRS(3)	LDL(73)		01304
	SAU(L+1)	ENI2(V)		01305
	ENI1(N)	SLJ(L-5)	EXIT	01306

* DISASSEMBLE READ BUFFER

3G
 SLJ(N)
 ENA(R+112)
 ENI2(14)
 LDQ2(R)
 ENA(O)
 EQS(14B)
 ENA(40B)
 ENI(O)
 ISK1(7)
 ENA(8)
 IJP2(L-7)
 ENA(O)
 SLJ(3G)
 SIL2(L+10)
 SAL(L+6)
 ENI(O)
 ENI1(O)
 LLS(6)
 SLJ(L+2)
 ENI(O)
 STA1(N)
 SLJ(L-4)
 RSB(L-2)
 ENI2(N)
 STA(R+120)
 ZRO(O)

EXIT/ENTRY
 DISASSEMBLE CHARACTERS
 SENSE ALTERNATE DASH
 STORE CHARACTER
 LOOP
 EXIT

01307
 01310
 01311
 01312
 01313
 01314
 01315
 01316
 01317
 01320
 01321
 01322
 01323

* ASSEMBLE WRITE BUFFER

4G
 SLJ(N)
 ENA(W)
 LDA1(W)
 ADD1(W+1)
 ADD1(W+2)
 ADD1(W+3)
 ADD1(W+4)
 ADD1(W+5)
 ADD1(W+6)
 ADD1(W+7)
 INI1(7)
 ISK1(119)
 SLJ(4G)
 ENI1(O)
 SAL(L+8)
 ALS(6)
 ALS(6)
 ALS(6)
 ALS(6)
 ALS(6)
 ALS(6)
 STA(N)
 RAO(L-1)
 SLJ(L-9)
 ZRO(O)

EXIT/ENTRY
 ASSEMBLE CHARACTERS
 STORE WORD
 LOOP
 EXIT

01324
 01325
 01326
 01327
 01330
 01331
 01332
 01333
 01334
 01335
 01336
 01337
 01340

* SENSE TYPEWRITER FOR ACTION

5G
 SLJ(N)
 ENA(I+1)
 SAL(Z+1)
 EXF(11100B)
 EXF1(T)
 EXF7(11100B)
 EXF1(T+1)
 EXF7(11B)
 LDA(T)
 INA(-4)
 INA(-36B)
 INA(13B)
 ENA(-1)
 SLJ4(7G)
 SLJ4(1FLAG)
 EXF7(11B)
 ENA(1)
 ISK1(100B)
 SLJ(L+2)
 SLJ(53EX)
 SLJ(L-2)
 EXF(10000B)
 AJP(53EX)
 AJP(13M)
 AJP1(L-8)
 ENI(O)

EXIT/ENTRY
 SENSE CARRIAGE RETURN
 SENSE CHARACTER
 SENSE SPACE
 SENSE PERIOD
 SENSE X

01341
 01342
 01343
 01344
 01345
 01346
 01347
 01350
 01351
 01352
 01353
 01354
 01355


```

5GEX  EXF(62560B)  SLJ(53)  01356
5GG   SLJ4(5G)      ZRO(0)    01357
      SLJ(Z+31B)    ZRO(0)    01360

```

* BCD TYPEWRITER OUTPUT

```

6G   SLJ(N)          SAL(L+3)  01361
      ENA(R5)         SLJ4(7G)  01362
      EXF(62540B)    EXF7(21B)  01363
      EXF(21110B)    LDA(N)     01364
      ENI(0)          STA(T+1)   01365
      ENI2(7)         ENI(0)     01366
      LDQ(T+1)        QLS(6)     01367
      STQ(T+1)        LDL(77B)   01370
      ENQ(0)          AJP(6SEX)   01371
      STA(T)          LIL1(T)    01372
      LDQ1(K0)        QLS(6)     01373
      LDL(77B)        AJP1(L+4)  01374
      EXF7(11141B)   SLJ(L+6)   01375
      ENA(57B)        STA(T)     01376
      SLJ(L+3)        ZRO(0)     01377
      EXF7(11140B)   SLJ(L+3)   01400
      ENA(47B)        STA(T)     01401
      EXF2(T)         EXF7(21B)  01402
      QLS(6)          STQ(T)     01403
      EXF2(T)         EXF7(21B)  01404
      ENI(0)          SLJ4(1DLAY) 01405
      IJP2(L-15)     RAO(L-18)  01406
      SLJ(L-19)      ZRO(0)     01407
      EXF(62560B)    SLJ(63)    01410

```

* COMM FLAG1 SET AND DELAY ROUTINE

```

1FLAG SLJ(N)          SET COMM FLAG1 01411
      SIU1(L+4)       GO DELAY       01412
      SLJ(L-2)        RETURN        01413

```

* TYPEWRITER SLOW-DOWN LOOP

```

1DLAY SLJ(N)          EXIT/ENTRY    01414
      ENI1(0)         DELAY         01415
      ENI1(N)         RE-STORE INDEX / GO EXIT 01416

```

* TYPEWRITER PRINT ROUTINE

92

SENJ(N)
SENJ(O)
SENAL(L+5)
SENAL(T+1)
EXXF7(11418)
ENAF(578)
ENXF2(T)
EENXI(T)
QLS(6)
APSP(7GEX)
AXF2(T)
ENI(O)
IUPJ(L-4)
SLPJ(L-6)
SLXF(625608)

EXF7(2 1B)
SLJ4(1 FLAG)
EXF(21 1 10B)
SAL(Z+2)
SLJ(L+3)
STA(T)
EXF7(2 1B)
LDQ(N)
LDL(77B)
STA(T)
EXF7(2 1B)
SLJ4(1 DLAY)
RAO(L-5)
SLR(O)
SLJ(73)

EXIT/ENTRY	01417
GO SET COMM FLAG1	01420
	01421
	01422
SENSE CARRIAGE UP	01423
	01424
SHIFT CARRIAGE DOWN	01425
PICK NEXT WORD	01426
	01427
SENSE END OF PRINT CODE	01430
PRINT CHARACTER	01431
GO TO SLOW DOWN TYPEWRITE	01432
SENSE EIGHT CHARACTERS	01433
LOOP	01434
	01435

* ASSIGN TAPE MEDIUM AND/OR SENSE ALL CHANNELS INACTIVE

86

SQS(1)
SLSJ(8G)
SQSK(3M)
SENA(27B)
LQC(7777B)
ALMEQ1(C+8)
LDA1(C+8)
STA(T)
LDL1(C+8)
ISK1(7)
ENAK(27B)
LDA(T)
EXFF7(11B)
EXFF7(21B)
EXFF7(31B)
EXFF7(41B)
EXFF7(51B)
EXFL(61B)
SCL(T+1)

AJP(L+15)
SLJ(L+2)
ZRO(0)
SLJ(L+3)
SLJ(L-2)
SLJ(85)
SIQ(T+1)
ENI(8)
SLJ(L+2)
SLJ(L+6)
ENI(0)
AJP(L+3)
SLJ(L-1)
SLJ(85)
RAD(L+8)
SLJ(L)
SLJ(L)
SLJ(L)
SLJ(L)
SLJ(L)
SLJ(85)

EXIT	01436
SENSE LIBRARY TAPE	01437
	01440
SENSE FULL MONITOR CONTROL	01441
SENSE STACKED-JOB MEDIUM	01442
	01443
	01444
	01445
SENSE PREVIOUS ASSIGNMENT	01446
NUMBER TO ACCUMULATOR	01447
	01450
SEARCH FOR FREE TAPE	01451
	01452
	01453
NUMBER TO TABLE AND ACCUMULATOR	01454
SENSE ALL CHANNELS INACTIVE	01455
	01456
	01457
	01460
	01461
	01462
EXIT	01463

* READ BINARY RECORD

11
SLJ(N)
EXF(63511B)
AJP1(L+1)

STA(1SWT)
INA(-2020B)
INA(63B)

•	EXIT/ENTRY	01464
•		01465
•	SENSE CARD REFERENCE	01466

```

INA(2020B)
IJP1(31)
ENA(0)
STA1(R)
LDA(R10)
SLJ(11)
ENA2(-5)
LDA(55)
SAU(L+2)
LDQ7(6S)
ISK1(N)
INA(1)
SLJ(11)
IJP1(L+1)
ENA(10X)
LDA(10)
IJP1(L+1)
IJP1(51)
STA(T)
SAU(2X)
ENA2(0)
ADD(T)
SLJ4(1X)
SLJ(11)
ENA(14002B)
IJP2(L+1)
IJP2(51)
ENA(R)
SAU(L+2)
EXF(N)
IJP1(L-3)
ENA(R+54)
ENI2(4)
LDA1(R)
ADL(R7)
AJP1(L+2)
AJP(L+1)
STA1(R)
INI1(1)
RAD(L-7)
AJP1(L-8)
ADD2(R)
EXF(70H)
ENI1(70)
LDQ1(R+1)
LLS(40)
AJP(41EX)
ENA(T0)

```

21

31

41

```

SLJ4(1G)
IJP2(21)
ENI1(53)
IJP1(-)
STA(R)
ZRO(0)
AJP1(51)
INA(-1)
ENI(0)
STQ1(R)
SLJ(L-1)
RAD(6S)
ZRO(0)
SLJ(41)
SAL(9X)
ENI(0)
LDA(UC)
ARS(2+)
ENA(R+54)
SAU(6X)
ALS(3)
INA(1)
ZRO(0)
ZRO(0)
SLJ4(1 FLAG)
INA(-1)
SAU(L+3)
ENI1(2)
INA(24)
SAL(Z+1)
EXF7(24004B)
ENI(0)
SAL(L+1)
LDQ(N)
QLS(8)
ENI(0)
LDL(R7)
LDA(-3)
ENI(0)
IJP2(L-5)
ENA1(-55)
ENI2(53)
IJP2(L)
IJP2(T)
LDA1(R)
QLS(8)
SCM(T)
ENI(0)
SLJ4(5G)

```

```

GET NUMERIC DESIGNATOR
SENSE NO MEDIUM REFERENCE
ENDFILE BLOCK TO BUFFER

EXIT
SENSE MEMORY REFERENCE
READ MEMORY RECORD

EXIT
SENSE TAPE REFERENCE
READ TAPE RECORD

EXIT
SENSE CARD REFERENCE

READ BINARY CARD GROUP

READ CARD
LOOP
ASSEMBLE EXTRA BITS

SENSE FOR MINUS ZERO

LOOP
GET CHECK SUM

CHECK FOR CORRECT READ
TROUBLE EXIT

```

01467
01470
01471
01472
01473
01474
01475
01476
01477
01500
01501
01502
01503
01504
01505
01506
01507
01510
01511
01512
01513
01514
01515
01516
01517
01520
01521
01522
01523
01524
01525
01526
01527
01530
01531
01532
01533
01534
01535
01536
01537
01540
01541
01542
01543
01544
01545
01546
01547

```

4IEX      AJP(4IEX)      SLJ(L-22)      01550
5I         EXF(62560B)   SLJ(1I)       01551
          ENA(E9)        SLJ(1E)       01552

```

```

      . SENSE FOR ACTION TO TAKE
      . ERROR EXIT

```

* SWITCH SELECTION ROUTINE

```

1J      SLJ(N)          SLJ(L+7)      01553
        ZRO(0)          ZRO(0)       01554
        ZRO(0)          ZRO(0)       01555
        ZRO(0)          ZRO(0)       01556
        ZRO(0)          ZRO(0)       01557
        ZRO(0)          ZRO(0)       01560
        ZRO(0)          ZRO(0)       01561
        ENA(0)          ZRO(0)       01562
        STA1(Z+51B)     ENI(5)       01563
        LDA1(L-8)       IJP1(L)      01564
        EQS2(F1)        ENI2(5)     01565
        ENA(-0)         SLJ(L+3)     01566
        INI1(1)         STA2(Z+51B)  01567
        ENA(0)          SLJ(L-3)     01570
        STA1(L-13)      ENI1(5)     01571
        SLJ(L-15)       IJP1(L)     01572
        ZRO(0)          ZRO(0)

```

```

      . EXIT/ENTRY
      . ARGUMENTS

```

```

      . CLEAR SIMULATION LOCATIONS
      . SENSE SWITCH SELECTION

```

```

      . CLEAR ARGUMENTS
      . EXIT

```

* CONTROL INITIALIZER

```

1K      SLJ(N)          SLJ(L)       01573
2K      ZRO(0)          ZRO(0)       01574
3K      ZRO(0)          ZRO(0)       01575
4K      ZRO(0)          ZRO(0)

```

```

      . EXIT/ENTRY
      . CONTROL MEDIUM
      . CONTROL STATEMENT OUTPUT MEDIUM
      . ERROR OUTPUT MEDIUM

```

* MONITOR ROUTINE

```

1M      ENI(0)          SLJ(L+8)     01577
2M      ZRO(0)          ZRO(0)       01600
3M      ZRO(0)          ZRO(0)       01601
4M      ZRO(0)          ZRO(0)       01602
5M      ZRO(0)          ZRO(0)       01603
6M      ZRO(0)          ZRO(0)       01604
7M      ZRO(0)          ZRO(0)       01605
8M      ZRO(0)          ZRO(0)       01606
        LDA(Z+47B)      AJP(L+4)     01607
        LDA(2M)         SAL(8V)     01610
        SLJ4(7V)        ZRO(0)      01611
        SLJ(L+5)        ZRO(0)      01612
        LDA(3M)         SAL(2V)     01613
        SLJ4(1V)        ZRO(0)     01614

```

```

      . ENTRY
      . STACKED-JOB MEDIUM
      . LISTABLE-OUTPUT MEDIUM
      . COMPILE-MODE INDICATOR
      . JOB-TIME LIMIT
      . PUNCH-CARD MEDIUM
      . SOURCE-LANGUAGE INDICATOR
      . INTERMEDIATE-LANGUAGE INDICATOR
      . SENSE RECOVERY INDICATOR
      . BACKSPACE STACKED-JOB MEDIUM
      . REWIND LISTABLE-OUTPUT MEDIUM

```



```

LDA(6M)
SLJ4(1V)
LDA(3M)
AJP(1P)
INA(-1)
LDA(2M)
THS(9)
SUB(3M)
LDA(2M)
AJP(1P)
ENI6(1)
LDA(2K)
LDA(3M)
LDA(2M)
EQS(63B)
EXF(62540B)
SIL2(1+11)
ENA2(14001B)
STQ(Z+1)
EXF1(M+39)
THS(9)
SLJ4(1R)
LNA(M+39)
LDA3(R)
LDA3(R+1)
ADD3(R+2)
ADD3(R+3)
ADD3(R+4)
ADD3(R+5)
ADD3(R+6)
ADD3(R+7)
INI3(7)
ISK3(79)
SIL6(2C)
LDA(4M)
LDA(R15)
EXF(62560B)
SIL6(S+2)
STA(3C)
LDQ(2M)
EXF(62560B)
LDQ(2M)
ENI6(1)
EXF(62560B)
ENA(-0)
ENI1(7)
SAU2(C+8)
LDA(3M)

```

```

SAL(2V)
ZRO(0)
INA(-1)
LDA(6M)
AJP(1P)
EXF(53544B)
SLJ(1P+4)
AJP(1P)
SUB(6M)
ENI(0)
SLJ4(10C)
STA(T+9)
AJP(11M)
ENI3(0)
SLJ(L+6)
SLJ4(1G)
ENQ(M+63)
SAL(L+1)
EXF(N)
EXF7(24004B)
SLJ(L+13)
ZRO(0)
SAL(L+8)
ALS(6)
ALS(6)
ALS(6)
ALS(6)
ALS(6)
ALS(6)
ALS(6)
STA(N)
RAO(L-1)
SLJ(L-9)
SIL6(3)
AJP1(11M)
STA(3C)
SLJ4(1C)
ENA(0)
SLJ(11M)
QJP(2P)
EXF(53544B)
QJP(1P)
SLJ4(3T)
SLJ4(1J)
EXF(53511B)
STA(15WT)
ENA(0)
IJP2(-)
AJP(L+2)

```

```

REWIND PUNCH-CARD MEDIUM
SENSE INPUT-OUTPUT CONFLICTS
TURN ON REMOTE MONITOR LIGHT
GO INITIALIZE FOR NEXT CALL
SENSE PARTIAL MONITOR CONTROL
SENSE STACKED-JOB DECK
GET NUMERIC DESIGNATOR
READ BCD RECORD
SENSE STACKED-JOB TAPE
GO READ BCD RECORD
ASSEMBLE READ BUFFER
TRANSFER WORD TO MONITOR BUFFER
SENSE TEN WORDS TRANSFERRED
SENSE INTERMEDIATE COMPILE TAPE
LOAD COMPILER
SUCCESS RETURN
RELEASE COMM FLAG 1
ERROR RETURN
GO OUTPUT JOB-TIME
GO RESET SWITCH SIMULATORS
RE-SET 160 LIB CALL LOCK-OUT
RESET TAPE-ASSIGNMENT TABLE

```



```

STA(4K)
EQS1(C+9)
ENA(-0)
LDA(2M)
EQS1(C+9)
ENA(-0)
LDA(6M)
EQS1(C+9)
ENA(-0)
ENA(0)
ENA(32001B)
SAU(L+4)
ENA(10B)
ENA(20)
RAO(CANCEL)
EXF(32001B)
ENA(77777B)
LIL2(CANCEL)
ISK1(4B)
ENA(52001B)
LIL2(CANCEL)
ISK2(10B)
ENA(-0)
ENA(32001B)
EXF(70B)
STA(Z)
LDQ(4M)
LIL1(4M)
STA(Z+50B)
EXF(1000B)
QJP(L+2)
STA(6T)
ENI1(119)
LDA1(R)
IJP1(L-1)
ADD(R+1)
AJP(L+2)
SLJ(L-5)
SSK(0SWT)
ENA(-0)
STA(8M)
LDA(R+5)
AJP1(L+1)
LDA(R+6)
AJP1(L+1)
ENA(23B)
SIL6(2C)
LDA(3M)
LDA(3K)

```

```

ENI(0)
SLJ(L+2)
SAU1(C+9)
ENI1(7)
SLJ(L+2)
SAU1(C+9)
ENI1(7)
SLJ(L+2)
SAU1(C+9)
STA(CANCEL)
SAL(L+3)
ENI1(1)
RAD(L+1)
EXF(32001B)
ISK2(3000B)
SLJ(L+3)
ENI(0)
SAU2(C+7)
SLJ(L-6)
SAU(L-9)
ENI(0)
SLJ(L-11)
SAU(C+15)
SAU(L-13)
LDA(R12)
ENA(0)
QJP(L+2)
SAU1(C+7)
EXF(100B)
LDQ(5M)
SLJ4(5T)
SLJ4(4T)
ENQ(20B)
STA1(W+1)
STQ(W)
INA(-166B)
SLJ4(Z+11B)
ZRO(0)
SLJ4(1SWTH)
STA(7M)
ENI(0)
INA(-54B)
RAO(7M)
INA(-54B)
RAO(8M)
SLJ4(1W)
SIL6(W)
AJP1(L+3)
SLJ4(1W)

```

```

SENSE LISTABLE-OUTPUT TAPE
CANCEL LISTABLE-OUTPUT TAPE
SENSE STACKED-JOB TAPE
CANCEL STACKED-JOB TAPE
SENSE PUNCH-CARD TAPE
CANCEL PUNCH-CARD TAPE
CANCEL UNASSIGNED UNITS

```

```

CANCEL GRAPH OUTPUT TAPE
RESET TIME
INSURE INTERMEDIATE TAPE RESET
SENSE JOB-TIME LIMIT
SET JOB-TIME LIMIT
TRANSFER READ TO WRITE BUFFER
SENSE JOB-DESCRIPTION RECORD
RETURN FOR NEXT RECORD
GO PROCESS SATELLITE JOB
CLEAR SOURCE INDICATOR
CLEAR MAP INDICATOR
SENSE DESIRE SOURCE LIST
SET SOURCE INDICATOR
SENSE DESIRE MAP LIST
SET MAP INDICATOR
TYPE JOB-DESCRIPTION RECORD
SENSE JOB--INCORPORATED CONTROL

```

```

01676
01677
01700
01701
01702
01703
01704
01705
01706
01707
01710
01711
01712
01713
01714
01715
01716
01717
01720
01721
01722
01723
01724
01725
01726
01727
01730
01731
01732
01733
01734
01735
01736
01737
01740
01741
01742
01743
01744
01745
01746
01747
01750
01751
01752
01753
01754
01755
01756

```

```

SLJ(L+2)
SLJ4(Z+12B)
LDA(6M)
ENA(0)
ENI(1119)
LDQ1(R)
IJP1(L-1)
LDA(R+2)
ADD(R+3)
AJP1(13M)
INA(-64B)
EXF(101B)
LDA(3M)
SLJ4(5V)
SLJ4(6M)
SLJ4(5V)
LDA(3M)
THS(9)
SLJ4(3V)
LDA(6M)
THS(9)
SLJ4(3V)
ENA(-0)
LDA(T+9)
ENA(1)
SLJ4(1V)
LDA(R+5)
EQS(65B)
STA(R)
SLJ(3P)
LDA(4M)
LDA(3M)
LDA(2M)
ENA(-0)
STA(6M)
EXF(70B)
LDA(7M)
LDA(8M)
SLJ4(S)
ENA(-0)
STA(1SWT)
EXF(70B)
EXF(70B)
EXF(70B)
LDA(R15)
SLJ4(1C)
SIL6(S+2)
STA(S+3)
EQS1(C+9)
SAU1(C+9)

```

12M

13M

14M

15M

```

ZRO(0)
ZRO(0)
AJP(12M)
SLJ4(3G)
LDA(6M)
STQ1(W)
SLJ4(1W)
ALS(6)
INA(-6545B)
LDA(R+4)
AJP1(13M)
EXF(2000B)
SAL(6V)
ZRO(0)
ZRO(0)
ZRO(0)
SAL(4V)
SLJ(L+2)
ZRO(0)
SAL(4V)
SLJ(L+2)
ZRO(0)
STA(2M)
STA(2M)
SAL(2V)
ZRO(0)
INA(31B)
SLJ(2P)
STA(R+1)
ZRO(0)
AJP1(15M)
AJP1(14M)
STA(2M)
STA(3M)
ENI(0)
SLJ(2P)
STA(S+4)
STA(S+6)
ZRO(0)
SAU(C+8)
EXF(53511B)
SLJ4(S+14000B)
SLJ(9M)
STA(3M)
ZRO(0)
LDA(4M)
ENI(7)
SLJ(1P)
ENI(0)

```

```

STACK JOB-DESCRIPTION RECORD
SENSE PUNCH-CARD MEDIUM
SENSE ALL CHANNELS INACTIVE
TRANSFER READ TO WRITE BUFFER
STACK JOB-DESCRIPTION RECORD
SENSE END OF JOB-STACK
STOP CLOCK
ENDFILE LISTABLE-OUTPUT MEDIUM
ENDFILE PUNCH-CARD MEDIUM
SENSE LISTABLE-OUTPUT TAPE
REWIND LISTABLE-OUTPUT TAPE
SENSE PUNCH-CARD TAPE
REWIND PUNCH-CARD TAPE
CANCEL MONITOR CONTROL
REWIND LIBRARY TAPE
SENSE SWITCH-BACK TO MONITOR
GO TO EXECUTE MONITOR
SENSE INTERMEDIATE COMPILE TAPE
SENSE JOB-INCORPORATED CONTROL
RETURN FOR CONTROL STATEMENT
RUN IN FORTRAN-TO-MEMORY MODE
COMPILE TO MEMORY
CANCEL LIBRARY TAPE
RE-SET 160 LIB CALL LOCK-OUT
EXECUTE OBJECT PROGRAM
RETURN FOR NEXT JOB
RUN IN FORTRAN-TO-TAPE MODE
LOAD COMPILER
CANCEL INTERMEDIATE TAPE

```

01757
01760
01761
01762
01763
01764
01765
01766
01767
01770
01771
01772
01773
01774
01775
01776
01777
02000
02001
02002
02003
02004
02005
02006
02007
02010
02011
02012
02013
02014
02015
02016
02017
02020
02021
02022
02023
02024
02025
02026
02027
02030
02031
02032
02033
02034
02035
02036
02037

SLJ4(Z+16B)
LDA(7M)
LDA(8M)
SLJ4(S)
LIL6(4M)
ENA6(O)
ENA(O)
STA(3C)
ENA(-O)
STA(1SWT)
EXF(7OB)
EXF(7OB)

ZRO(O)
STA(S+4)
STA(S+6)
ZRO(O)
SIL6(2C)
SLJ4(Z+16B)
SAU6(C+7)
SLJ4(1C)
SAU(C+8)
EXF(53511B)
SLJ4(S)
SLJ(9M)

REWIND INTERMEDIATE TAPE
COMPILE TO TAPE
REWIND INTERMEDIATE TAPE
RELEASE INTERMEDIATE TAPE
CALL ORJECT PROGRAM
CANCEL LIBRARY TAPE
RE-SET 160 LIB CALL LOCK-OUT
EXECUTE OBJECT PROGRAM
RETURN FOR NEXT JOB

* WRITE BINARY RECORD

10 SLJ(N)
AJP1(L+1)
INA(2020B)

20 SLJ(10)
ENA2(-5)
LDA(5S)
SAU(L+2)
LDQ1(W)
ISK1(N)
RAD(6S)
IJP1(L+1)
LDA(U1)
IJP1(L+1)
IJP1(10)
STA(T+10)
SAU(3Y)
ALS(3)
ADD(T+10)
SLJ(10)
ENA2(-2)
AJP1(50)
ENI2(O)
ENA(W+24)
EXF(24004B)
STA2(W+54)
ENI2(O)
ENI1(71)
ADD2(W)
EXF(7OB)
INI1(-1)
LLS(8)

40 SLJ4(N)
AJP1(L+1)
INA(2020B)
IJP1(30)
SLJ(10)
ENA2(-5)
LDA(5S)
SAU(L+2)
LDQ1(W)
ISK1(N)
RAD(6S)
IJP1(L+1)
LDA(U1)
IJP1(L+1)
IJP1(10)
STA(T+10)
SAU(3Y)
ALS(3)
ADD(T+10)
SLJ(10)
ENA2(-2)
AJP1(50)
ENI2(O)
ENA(W+24)
EXF(24004B)
STA2(W+54)
ENI2(O)
ENI1(71)
ADD2(W)
EXF(7OB)
INI1(-1)
LLS(8)

INA(-2020B)
ENA(63B)
SLJ4(2G)
IJP2(20)
ZRO(O)
AJP1(50)
INA(-1)
INA(1)
STQ(5S)
SLJ(L-1)
SLJ(13)
SLJ(43)
ENI(O)
LDA(U3)
ENI(O)
ENA(W+54)
ENA2(O)
INA(1)
SLJ4(1Y)
ZRO(O)
SLJ4(1 FLAG)
EXF7(24004B)
ISK2(1000B)
ENI2(17)
SAL(Z+2)
ENA(O)
IJP2(L)
ISK2(1000B)
ENI2(53)
IJP2(-)
STQ1(W)
ENQ(-3)
STQ1(W)

EXIT/ENTRY
SENSE CARD REFERENCE
GET NUMERIC DESIGNATOR
EXIT
SENSE MEMORY REFERENCE
WRITE MEMORY RECORD
EXIT
SENSE TAPE REFERENCE
WRITE TAPE RECORD
EXIT
SENSE CARD REFERENCE
SENSE CARD REFERENCE
DELAY
PUNCH BINARY CARD GROUP
PUNCH CARD ONE
CLEAR
DELAY
GET CHECK SUM
STORE LOW ORDER PART
STORE HIGH ORDER PART

```

ENA(W+48)
EXF(24004B)
EXF2(W+24)
ENA(W)
ENI2(O)
ENI2(O)
LDQ2(N)
ISK2(4)
STA1(W+54)
ALS(15)
LDA1(W+48)
ISK1(10)
ISK1(23)
ENA(M+39)
EXF(24004B)
EXF2(M+15)
EXF(62560B)
ENA(E12)

```

60
50

```

EXF7(24004B)
SAL(Z+2)
ENI1(O)
SAU(L+3)
ISK2(1000B)
ENA(O)
LLS(8)
SLJ(L-1)
ENA(5000B)
RAD(L-3)
SLJ(L-5)
STA1(M+15)
SLJ(L-1)
EXF7(24004B)
SAL(Z+2)
EXF7(24004B)
SLJ(13)
SLJ(13)

```

PUNCH CARD TWO
ASSEMBLE EXTRA BITS
DELAY

TRANSFER TO BUFFER

PUNCH CARD THREE
ERROR EXIT

* PROGRAM CONTROL ROUTINE

```

1P  ENA(O)
2P  EXF(53511B)
    ENA(-O)
    LDA(2K)
    EXF(2000B)
    ENA(O)
    STA2(R)
    ENA(T+1)
    EXF7(11101B)
    EXF7(11B)
    LIL1(T)
    EXF7(1141B)
    QLS(30)
    STA2(R)
    INI2(1)
    EXF(1000B)
    STA(4K)
    ENA(20B)
    ISK2(120)
    LDA(R11)
    ENA(-O)
    STA1(2M)
    SLJ(L+2)
    LDA(2K)
    ENI1(71)
    LDA1(R)

```

3P
4P

```

STA(25)
EXF(63544B)
STA(15WT)
AJP1(4P)
EXF(11100B)
ENI2(120)
IJP2(L)
SAL(Z+1)
ISK1(100B)
SLJ(2P)
SLJ(L-1)
LDQ1(KO)
QLS(6)
LDL(77B)
LDA(-73B)
AJP3(L-7)
STA(3K)
ENI(O)
STA2(R)
SLJ(L-1)
STA(Z+57B)
ENI1(5)
IJP1(L)
ZRO(O)
SLJ4(1R)
ENI2(47)
STA1(W)

```

SET CONTROL AT TYPEWRITER
FORTRAN ON / MONITOR OFF
RE-SET 160 LIB CALL LOCK-OUT
SENSE TYPEWRITER CONTROL
SELECT TYPEWRITER
CLEAR BUFFER
READ CHARACTER
SENSE CARRIAGE RETURN
SENSE CHARACTER
TABLE LOOKUP
SHIFT ON LOWER CASE
LOOP UNTIL PERIOD OR OPEN PAREN
NULLIFY LIMIT AT 57
CANCEL MONITOR CONTROL
READ CONTROL STATEMENT
COPY BUFFER

02115
02116
02117
02120
02121
02122
02123
02124
02125
02126
02127
02130
02131
02132
02133
02134
02135
02136

02137
02140
02141
02142
02143
02144
02145
02146
02147
02150
02151
02152
02153
02154
02155
02156
02157
02160
02161
02162
02163
02164
02165
02166
02167
02170
02171

IJP1(L-1)
 STA2(W+72)
 LDA(3K)
 ENA(73B)
 EQS(R)
 EQS(R+1)
 SLJ(10M)
 LDA4(R)
 AJP1(L+1)
 LDA4(R)
 EQS1(F12)
 ENA(O)
 LDA4(R)
 EQS1(FO)
 LDA(T)
 ADD4(R)
 INI4(1)
 SUB(20B)
 LDA(T)
 LRS(6)
 LDL(-OB)
 EQS3(PO)
 ENA3(-7)
 SSK(2M)
 LDA3(OZ)
 SAL(7P)
 STA(8P)
 EXF(7OB)
 EQS(R17)
 ENI1(79)
 LDQ1(R)
 IJP1(L-1)
 ENA(R5)
 SLJ(2P)
 ENA(E0)
 SLJ(N)
 ZRO(O)
 ZRO(O)
 LDA4(R)
 INA(-33B)
 INA(-1)
 INA(-37B)
 SLJ(-1)
 INI4(1P)
 INI4(1)
 SUB(20B)
 ENA(O)
 LDA4(R)

5P

6P
 7P
 8P
 9P

ENA(20B)
 IJP2()
 AJP5(1W)
 ENI4(3)
 SLJ(L+3)
 SLJ(4P)
 ZRO(O)
 INA(-54B)
 INI4(1)
 ENI1(26)
 SLJ(9M)
 STA(T)
 ENI1(36)
 SLJ(L+4)
 ALS(6)
 STA(T)
 SLJ(L-4)
 AJP(L-1)
 ENQ(O)
 AJP1(L)
 LIL3(C+5)
 SLJ(5P)
 AJP2(L+2)
 SLJ(2P)
 SAL(L+3)
 ARS(24)
 SLJ4(6P)
 SLJ4(N)
 SLJ(2P)
 SLJ(L+6)
 ENA(23B)
 STQ1(W)
 SLJ4(1W)
 SLJ4(5G)
 ZRO(O)
 SLJ(1E)
 SLJ(L+3)
 ZRO(O)
 ZRO(O)
 AJP(11P)
 AJP(L+5)
 AJP(L+4)
 AJP(6P)
 AJP(6P)
 ZRO(O)
 LDA4(R)
 AJP(L-1)
 STA(T)
 ENI1(3)

02172
 02173
 02174
 02175
 02176
 02177
 02200
 02201
 02202
 02203
 02204
 02205
 02206
 02207
 02210
 02211
 02212
 02213
 02214
 02215
 02216
 02217
 02220
 02221
 02222
 02223
 02224
 02225
 02226
 02227
 02230
 02231
 02232
 02233
 02234
 02235
 02236
 02237
 02240
 02241
 02242
 02243
 02244
 02245
 02246
 02247
 02250
 02251
 02252

.....

OUTPUT STATEMENT IF INDICATED

SENSE DOUBLE PERIOD

SENSE ASTERISK

SENSE LETTER

ASSEMBLE NAME

POSITION NAME

SEARCH CALLED PROGRAMS

SENSE MONITOR-RESTRICTED GROUP

SENSE NO MONITOR CONTROL

EXECUTE PROGRAM

RETURN FOR CONTROL STATEMENT

SENSE REMARK

TYPE REMARK

RETURN FOR CONTROL STATEMENT

ERROR EXIT

PROCESS ARGUMENTS

NEXT ARGUMENT ADDRESS

NUMBER OF ARGUMENTS

SENSE COMMA OR OPEN PAREN

SENSE PERIOD OR CLOSED PAREN

SKIP SPACES

CLEAR ASSEMBLY

.....

	SENSE OCTAL NUMBER
•	02253
•	02254
•	02255
•	02256
•	02257
•	02260
•	02261
•	02262
•	02263
•	02264
•	02265
•	02266
•	02267
•	02270
•	02271
•	02272

	ASSEMBLE NUMBER
•	02253
•	02254
•	02255
•	02256
•	02257
•	02260
•	02261
•	02262
•	02263
•	02264
•	02265
•	02266
•	02267
•	02270
•	02271
•	02272

	ASSEMBLE NAME
•	02253
•	02254
•	02255
•	02256
•	02257
•	02260
•	02261
•	02262
•	02263
•	02264
•	02265
•	02266
•	02267
•	02270
•	02271
•	02272

	SENSE ARGUMENT LIMIT
•	02253
•	02254
•	02255
•	02256
•	02257
•	02260
•	02261
•	02262
•	02263
•	02264
•	02265
•	02266
•	02267
•	02270
•	02271
•	02272

	ERROR EXIT
•	02253
•	02254
•	02255
•	02256
•	02257
•	02260
•	02261
•	02262
•	02263
•	02264
•	02265
•	02266
•	02267
•	02270
•	02271
•	02272

EQS1(F0)	SLJ(L+6)
LDA4(R)	ENI1(8)
EQS1(F0)	SLJ(10P)
LDA(T)	ALS(3)
INI1(O)	STA(T)
INI4(1)	SLJ(L-4)
LDA4(R)	ENI1(36)
EQS1(F0)	SLJ(L+4)
LDA(T)	ALS(6)
ADD4(R)	STA(T)
INI4(1)	SLJ(L-4)
SUB(208)	AJP(L-1)
RAD(7P)	RSO(8)
AJP3(11P)	LDA(T)
STA7(7P)	SLJ(9P)
ENA(E18)	SLJ(1E)

10P

11P

* EXECUTE LAST ASSEMBLED PROGRAM

1Q	ENI(O)	LIL(C+5)	ENTRY	02273
	INI1(-1)	LDA1(CZ)		02274
	SAU(L+1)	ENI(O)		02275
	SLJ4(N)	ZRO(O)	EXECUTE PROGRAM	02276
	SLJ(9M)	ZRO(O)	RETURN TO MONITOR	02277

* HOLD ROUTINE

2Q	SLJ(N)	LDA(C+5)	EXIT/ENTRY	02300
	STA(C+4)	LDA(C+3)		02301
	STA(C+2)	ENA(1)		02302
	STA(Z+47B)	SLJ(L-3)	EXIT	02303

* CLEAR ROUTINE

3Q	SLJ(N)	SLJ4(10C)	EXIT/ENTRY	02304
	ENA(O)	STA1(S)		02305
	ISK1(777768)	SLJ(L-1)		02306
	SLJ(L-3)	ZRO(O)	EXIT	02307

* SKIP TO NEXT OR SPECIFIED JOB

4Q	SLJ(N)	SLJ(L+3)	EXIT/ENTRY	02310
	ZRO(O)	ZRO(O)	MEDIUM	02311
	ZRO(O)	ZRO(O)	CHARACTERS IDENTIFYING JOB	02312
	LDA(L-2)	SLJ4(1R)	READ RECORD	02313

LDA(R) LDA(-1668)
 INA(L-4) LDA(L-4)
 ENI(0) ENI(0)
 EQS(L-8) EQS(L-8)
 ENA(0) ENA(0)
 SLJ(L-12) SLJ(L-12)
 ISK(17) ISK(17)
 SLJ(L-11) SLJ(L-11)

ADD(R+1) ADD(R+1)
 AJP3(L-2) AJP3(L-2)
 ENQ(778) ENQ(778)
 ENI(0) ENI(0)
 ADL1(R+2) ADL1(R+2)
 SLJ(L+3) SLJ(L+3)
 STA(L-9) STA(L-9)
 ZRO(0) ZRO(0)
 SLJ(L-4) SLJ(L-4)
 ZRO(0) ZRO(0)

02314
 02315
 02316
 02317
 02320
 02321
 02322
 02323
 02324
 02325
 02326

* BCD READ ROUTINE

1R SLJ(N)
 IJP1(L+10)
 ENA(0)
 STA1(R+1)
 STA1(R)
 STANA(738)
 STA(R+1)
 IJP2(L+1)
 IJP2(L+1)
 ENA2(-3)
 SLJ(5R)
 IJP1(9R)
 ENI(0)
 IJP2(L+1)
 SAU(L+1)
 EXF(N)
 ENA(R+104)
 EXF1(R+80)
 EXF(62560B)
 ENI(0)
 LDA1(R)
 EQS2(F0)
 IJP1(L-2)
 EQS(14B)
 ENA(40B)
 SLJ(L-3)
 ENA(T3)
 AJP(1R)
 SLJ(N)
 STA2(0)
 ENA(60B)
 ENA(40B)
 ENA(20B)

SLJ4(1G)
 IJP2(L+6)
 ENI1(119)
 ENA(20B)
 IJP1(L)
 STA(R)
 SLJ(1R)
 SLJ(6R)
 SLJ(7R)
 AJP(8R)
 ZRO(0)
 ENI(0)
 ENA(14002B)
 INA(-1)
 IJP2(5R)
 SLJ4(1FLAG)
 SAL(Z+1)
 EXF7(14004B)
 SLJ4(3R)
 ENI1(79)
 ENI2(+7)
 SLJ(L+2)
 SLJ(1R)
 SLJ(L+3)
 STA1(R)
 ZRO(0)
 SLJ4(5G)
 SLJ(23)
 ENI1(80)
 ENI2(79)
 IJP2(L)
 SLJ4(4R)
 SLJ4(4R)
 SLJ4(4R)

02327
 02330
 02331
 02332
 02333
 02334
 02335
 02336
 02337
 02340
 02341
 02342
 02343
 02344
 02345
 02346
 02347
 02350
 02351
 02352
 02353
 02354
 02355
 02356
 02357
 02360
 02361
 02362
 02363
 02364
 02365
 02366
 02367
 02370

EXIT/ENTRY
 SENSE NO MEDIUM REFERENCE
 ENDFILE BLOCK TO BUFFER
 ENTER DOUBLE PERIOD
 EXIT
 SENSE TYPEWRITER INPUT
 SENSE PAPER TAPE INPUT
 SENSE MEMORY INPUT
 SENSE MAGNETIC TAPE INPUT
 SENSE CARD INPUT
 PROCESS CARD READER
 READ CARD
 GO PROCESS CARD IMAGE
 SENSE VALID CODES
 EXIT
 SENSE ALTERNATE DASH
 TROUBLE EXIT
 SENSE FOR ACTION TO TAKE
 PROCESS CARD IMAGE
 CLEAR BUFFER
 PROCESS ROWS

02327
 02330
 02331
 02332
 02333
 02334
 02335
 02336
 02337
 02340
 02341
 02342
 02343
 02344
 02345
 02346
 02347
 02350
 02351
 02352
 02353
 02354
 02355
 02356
 02357
 02360
 02361
 02362
 02363
 02364
 02365
 02366
 02367
 02370

```

ENA(1)
ENA(2)
ENA(3)
ENA(4)
ENA(5)
ENA(6)
ENA(7)
ENA(8)
ENA(9)
ENA(20B)
EQS1(R)
ENA(12B)
ENA(20B)
ENA(O)
EQS1(R)
ENA(20B)
ENA(O)
ENI1(39)
STAI(R+80)
ENA(O)
SLJ(3R)
SLJ(N)
LDA1(R)
STA1(R)
SSH1(R)
LDA(T)
ENI(O)
LDA1(R+1)
STA1(R+1)
SSH1(R+1)
LDA(T)
ENI(O)
INI1(2)
ENA(E9)
EXF(11100B)
ENA(T+1)
ENI2(O)
EXF1(T)
STA1(R+1)
STA1(R)
EXF1(T)
EXF7(11101B)
EXF7(11B)
EXF1(T)
EXF7(11141B)
QLS(30)
STA2(R)
ISK2(119)
EXF1(T+1)

```

```

SLJ4(4R)
SLJ4(4R)
SLJ4(4R)
SLJ4(4R)
SLJ4(4R)
SLJ4(4R)
SLJ4(4R)
ENI1(80)
SLJ(L+3)
STAI(R)
SLJ(L-2)
ENI1(80)
SLJ(L+3)
STAI(R)
SLJ(L-2)
ENA(20B)
IJP1(L)
STAI(R+120)
ZRO(O)
STA(T)
ALS(8)
ENI2(39)
SLJ(L+2)
RAD2(3)
IJP2(L-2)
ALS(8)
ENI2(39)
SLJ(L+2)
RAD2(3+40)
IJP2(L-2)
SLJ(4R)
SLJ(1E)
SLJ4(1FLAG)
SAL(Z+1)
ENI1(119)
ENA(O)
ENA(20B)
IJP1(L)
ISK1(100B)
SLJ(L+7)
SLJ(L-1)
LDQ1(KO)
QLS(6)
LDL(77B)
ENI(O)
SLJ(L-7)
EXF(11100B)

```

4R

5R
6R

```

CORRECT ZERO CODES
LOOP
CORRECT BLANK CODES
LOOP
FILL OUT BLANKS
RETURN
ROW PROCESSOR

RETURN EXIT
ERROR PROCESS TYPEWRITER

CLEAR BUFFER
SENSE CARRIAGE RETURN
LOOP TO CHARACTERS
TABLE LOOKUP
SHIFT ON LOWER CASE
STORE IN BUFFER
LOOP

```

```

02371
02372
02373
02374
02375
02376
02377
02400
02401
02402
02403
02404
02405
02406
02407
02410
02411
02412
02413
02414
02415
02416
02417
02420
02421
02422
02423
02424
02425
02426
02427
02430
02431
02432
02433
02434
02435
02436
02437
02440
02441
02442
02443
02444
02445
02446
02447
02450
02451

```



```

SLJ4(1X)      ZRO(0)
SLJ4(3G)      ZRO(0)
SLJ(1R)       ZRO(0)
:             :
: DISASSEMBLE READ BUFFER
: EXIT
:
02533
02534
02535

```

* INPUT MEDIUM DESIGNATOR

```

1S      SLJ(N)      SLJ(L+3)
        ZRO(0)      ZRO(0)
        ZRO(0)      ZRO(0)
        LIL1(L-2)   ENQ(77B)
        LDL(L-2)    ALS(6)
        ENQ(70077B) SSU1(G0)
        STA1(G0)    SLJ(1S)
:
: EXIT/ENTRY
: SYMBOL
: DEVICE
:
02536
02537
02540
02541
02542
02543
02544

```

* OUTPUT MEDIUM DESIGNATOR

```

2S      SLJ(N)      SLJ(L+3)
        ZRO(0)      ZRO(0)
        ZRO(0)      ZRO(0)
        LIL1(L-2)   ENQ(77B)
        LDL(L-2)    ENQ(77700B)
        SSU1(G0)    STA1(G0)
        SLJ(2S)     ZRO(0)
:
: EXIT/ENTRY
: SYMBOL
: DEVICE
:
02545
02546
02547
02550
02551
02552
02553

```

* SET START OF MEMORY FILE

```

3S      SLJ(N)      SLJ(L+4)
4S      ZRO(0)      ZRO(S)
5S      ZRO(0)      ZRO(0)
6S      ZRO(0)      ZRO1(S)
        LDA(4S)     SAL(6S)
        LNA(5S)     AJP1(3S)
        ENA(1S)     STA(5S)
        SLJ(3S)     ZRO(0)
:
: EXIT/ENTRY
: START OF MEMORY FILE
: MEMORY RECORD LENGTH
: END OF MEMORY FILE
: SAVE START OF MEMORY FILE
: SENSE RECORD LENGTH TO FIFTEEN
: SET RECORD LENGTH TO FIFTEEN
: EXIT
:
02554
02555
02555
02556
02557
02560
02561
02562
02563

```

* CLOCK ROUTINE

```

1T      SLJ(N)      LDA(R11)
        STA(Z+57B)  LDA(R12)
        STA(Z)       ENA(0)
        STA(Z+50B)  EXF(100B)
        EXF(1000B)  SLJ(1T)
:
: EXIT/ENTRY
: RESET LIMIT AT 57
: SET ONE-SECOND COUNT
: RESET TIME AT 50
: EXIT
:
02564
02565
02566
02567
02570

```

* STOP CLOCK ROUTINE


```

2T      SLJ(N)
        EXF(2000B)
        EXF(101B)
        SLJ(2T)
        . EXIT/ENTRY
        . EXIT
02571
02572

```

* TIME ROUTINE

```

3T      SLJ(N)
        LDA(Z+50B)
        DVI(60)
        SLJ4(L+8)
        ADD(R13)
        LDA(T+10)
        ADD(R14)
        ENA(E21)
        QJP(L+1)
        QJP4(6G)
        SLJ(3T)
        SLJ(N)
        THS(100)
        THS(10)
        LDQ(R6)
        ENQ(O)
        SAL(T+1)
        LDA(T)
        SCL(77B)
        STA(T)
        IJP2(L-5)
        SLJ(L-10)
        SLJ4(7T)
        ENQ(O)
        STQ(T+10)
        ZRO(O)
        STA(E21)
        SLJ4(L+6)
        STA(E23)
        LDQ(24)
        LDQ(4K)
        SLJ4(3E)
        ZRO(O)
        ENI2(J)
        ENI2(1)
        ENI2(1)
        STQ(T)
        DVI2(R0)
        LIL1(T+1)
        ALS(6)
        ADD1(E0)
        LLS(48)
        LDA(T)
        ZRO(O)
        . EXIT/ENTRY
        . GET MINUTES
        . GET SECONDS
        . CONVERT NUMBER TO BCD
        . RETURN
02573
02574
02575
02576
02577
02600
02601
02602
02603
02604
02605
02606
02607
02610
02611
02612
02613
02614
02615
02616
02617
02620

```

* LIMIT ROUTINE

```

4T      SLJ(N)
5T      ZRO(O)
6T      ZRO(O)
        SLJ4(7T)
        LDA(5T)
        ADD(6T)
        STA(Z+57B)
        SLJ(L+3)
        ZRO(O)
        ZRO(O)
        ZRO(O)
        MUI(6J)
        ADD(Z+50B)
        SLJ(4T)
        . EXIT/ENTRY
        . MINUTES
        . SECONDS
        . COMPUTE AND SET LIMIT
        . EXIT
02621
02622
02623
02624
02625
02626
02627

```

* START CLOCK IF NOT RUNNING

```

7T      SLJ(N)
        ENI1(O)
        LDA(Z)
        EQS(Z)
        STQ(Z+50B)
        ENI(O)
        ENQ(O)
        ISK1(40000B)
        SLJ(7T)
        LDA(R12)
        . EXIT/ENTRY
        . WAIT
        . SENSE CLOCK RUNNING
        . INITIALIZE TIME
02630
02631
02632
02633
02634

```

STA(Z)	EXF(120B)	START CLOCK	02635
EXF(1000B)	SLJ(7T)	EXIT	02636

* REWIND SPECIFIED MEDIUM

1V
2V

SLJ(N)	SLJ(L+2)	EXIT/ENTRY	02637
ZRO(O)	ZRO(O)	MEDIUM	02640
ENI(0)	ISK1(3000B)	DELAY	02641
LDA(2V)	SLJ4(1G)	GET NUMERIC DESIGNATOR	02642
IJP1(L+2)	IJP2(L+13)	SENSE TAPE REFERENCE	02643
SLJ(1VA)	ZRO(O)	EXIT	02644
IJP1(L+1)	SLJ(1VA)	SET SELECT CODES	02645
LDA(U1)	ENI(O)		02646
IJP1(L+1)	LDA(UO)		02647
IJP1(1VA)	ARS(24)		02650
STA(T)	SAL(L+5)		02651
ENA2(O)	ALS(3)		02652
INA(2)	ADD(T)		02653
SAU(L+2)	INA(3)		02654
SAU(L+2)	SLJ4(1 FLAG)		02655
EXF(N)	EXF7(N)	REWIND TAPE	02656
EXF(N)	SLJ(1VA)	EXIT	02657
IJP2(L+1)	SLJ(1VA)	SENSE PAPER TAPE REFERENCE	02660
IJP2(L+10)	ENA(O)	PROCESS PAPER TAPE	02661
STA(T+1)	ENA(T+2)	PUNCH LEADER	02662
SAL(Z+2)	SLJ4(1 FLAG)	LOOP	02664
EXF(21210B)	ENI1(100)		02665
EXF2(T+1)	EXF7(2 1B)		02666
IJP1(L-1)	ENI(O)		02667
ENA(57B)	STA(T)	PUNCH CARRIAGE RETURN	02670
ENA(45B)	STA(T+1)	EXIT	02671
EXF2(T)	EXF7(2 1B)	SENSE MEMORY REFERENCE	02672
SLJ(1VA)	ZRO(O)	REWIND MEMORY FILE	02673
INI2(-3B)	IJP2(1VA)		02674
LDA(4S)	SAL(6S)		02675
EXF(62560B)	SLJ(1V)		

1VA

* REWIND TAPE WITH INTERLOCK

3V
4V

SLJ(N)	SLJ(L+2)	EXIT/ENTRY	02676
ZRO(O)	ZRO(O)	MEDIUM	02677
ENI(0)	ISK1(3000B)	DELAY	02700
LDA(4V)	SLJ4(1G)	GET NUMERIC DESIGNATOR	02701
IJP1(L+1)	SLJ(3V)	SENSE TAPE REFERENCE	02702
IJP1(L+1)	SLJ(3V)	SET SELECT CODES	02703
LDA(U1)	IJP1(L+2)		02704
LDA(UO)	SLJ4(1 FLAG)		02705

ARS(24)	STA(T)	.	02706
SAL(L+4)	ENA2(J)	.	02707
ALS(3)	INA(2)	.	02710
ADD(T)	SAU(L+2)	.	02711
INA(5)	SAU(L+2)	.	02712
EXF(N)	EXF7(N)	.	02713
EXF(N)	ENI(O)	.	02714
EXF(62560B)	SLJ(3V)	.	02715
3VV	SLJ4(3V)		
	SLJ(Z+33B)		
	ZRO(O)	.	02716
	ZRO(O)	.	02717

* WRITE ENDFILE MARKER ON SPECIFIED MEDIUM

5V	SLJ(N)	EXIT/ENTRY	02720
6V	ZRO(O)	MEDIUM	02721
	ENI(O)	DELAY	02722
	LDA(6V)	GET NUMERIC DESIGNATOR	02723
	IJP1(L+1)	SENSE TAPE REFERENCE	02724
	IJP1(L+1)		02725
	LDA(U1)	SET SELECT CODES	02726
	IJP1(L+1)		02727
	IJP1(5V)		02730
	SAU(L+6)		02731
	SAL(L+4)		02732
	SAL(T)		02733
	ALS(3)		02734
	ADD(T)		02735
	EXF(N)	WRITE TAPE ENDFILE	02736
	EXF(N)		02737
	EXF(62560B)		02740
	ENA2(-6)	EXIT	02741
	LIL1(5S)	SENSE MEMORY REFERENCE	02742
	LDQ(R4)	WRITE MEMORY ENDFILE	02743
	STQ7(6S)		02744
	RAD(6S)	EXIT	02745

* BACKSPACE SPECIFIED MEDIUM

7V	SLJ(N)	EXIT/ENTRY	02746
8V	ZRO(O)	MEDIUM	02747
	ENI(O)	DELAY	02750
	LDA(8V)	GET NUMERIC DESIGNATOR	02751
	IJP1(L+2)	SENSE TAPE REFERENCE	02752
	SLJ(7V)		02753
	IJP1(L+1)		02754
	LDA(U1)	SET SELECT CODES	02755

```

IJP1(L+1)
ALS(24)
IJP1(7V)
SAU(L+6)
SAL(L+4)
SAL(T)
ALS(3)
ADD(T)
EXF(N)
EXF(N)
EXF(62560B)
ENA2(-5)
LDA(5S)
SLJ(7V)

```

```

LDA(U3)
ENI(0)
INA(6)
SCL(777B)
SAL(L+5)
ENA2(3)
INA(2)
SAU(L+1)
EXF7(N)
EXF7(N)
SLJ(7V)
AJP1(7V)
RSB(6S)
ZRO(0)

```

BACKSPACE TAPE

EXIT
SENSE MEMORY REFERENCE
BACKSPACE MEMORY
EXIT

02756
02757
02760
02761
02762
02763
02764
02765
02766
02767
02770
02771
02772
02773

* BCD WRITE ROUTINE

```

1W
SLJ(N)
IJP1(L+6)
SLJ(1W)
IJP2(L+1)
IJP2(L+1)
ENA2(-3)
SLJ(5W)
IJP1(10W)
ENA2(-2)
SLJ(5W)
SLJ(3W)
ENA(W+24)
EXF(24004B)
EXF2(W)
EXF(62560B)
SLJ(N)
ENA(1)
STA(T)
ENQ(77B)
STA(T+1)
LDA2(GO)
INI2(-1)
SSH(T+1)
LDA(T)
ENI(0)
ENQ(77B)
STA(T+1)
LDA2(GO)
INI2(-1)

```

```

SLJ4(2G)
IJP2(L+2)
ZRO(0)
SLJ(6W)
SLJ(7W)
AJP(9W)
ZRO(0)
ENI(0)
AJP(2W)
ZRO(0)
ZRO(0)
SLJ4(1FLAG)
SAL(Z+2)
EXF7(14004B)
SLJ(1W)
ENI(0)
ALS(7)
ENI(1)
LDL1(W)
LIL2(T+1)
ALS(6)
STA(T+1)
ENI(0)
SLJ(L+2)
RAD2(W)
IJP2(L-3)
LDL1(W+40)
LIL2(T+1)
ALS(6)
STA(T+1)
ENI(0)

```

EXIT/ENTRY
SENSE NO MEDIUM REFERENCE
EXIT
SENSE TYPEWRITER OUTPUT
SENSE PAPER TAPE OUTPUT
SENSE MEMORY OUTPUT
SENSE MAGNETIC TAPE OUTPUT
SENSE CARD OUTPUT
PROCESS CARD PUNCH

EXIT
FORM CARD IMAGE

TABLE LOOKUP

TAG CONTROL WORDS

LOOP

TABLE LOOKUP

02774
02775
02776
02777
03000
03001
03002
03003
03004
03005
03006
03007
03010
03011
03012
03013
03014
03015
03016
03017
03020
03021
03022
03023
03024
03025
03026
03027
03030
03031
03032

```

SSH(T+1)
LDA(T)
ENI(O)
LDA(T)
STA(T)
ISK1(39)
LDA1(W)
STA1(W)
ISK1(23)
SLJ(3W)
ENA(E12)
ENA(R5)
EXF(21110B)
ENI(2+1)
ENI(2119)
LDA2(W)
AJP1(L+1)
SIU2(L+13)
LDA2(W)
LIL1(T)
QLS(6)
AJP1(L+4)
EXF7(11141B)
ENA(57B)
SLJ(L+3)
EXF7(11140B)
ENA(47B)
EXF2(T)
QLS(6)
EXF2(T)
ISK2(N)
EXF7(11141B)
ENA(57B)
EXF2(T)
SLJ(1W)
ENI(2119)
ZRO(O)
ENA(T+1)
EXF(21210B)
LDA2(W)
AJP1(L+1)
SIU2(L+18)
LDA2(W)
INA(-20B)
ENA(4)
LIL1(T)
QLS(18)
AJP1(L+5)
LDA(L-12)

```

5W
6W

7W

```

SLJ(L+2)
RAD2(W+1)
IJP2(L-3)
ALS(1)
ENI(O)
SLJ(4W)
ARS(7)
ENI(O)
SLJ(L-2)
ZRO(O)
SLJ(1)
SLJ4(7G)
ENI(O)
SAL(Z+2)
ENI(O)
INA(-20B)
IJP2(L-1)
ENI(2)
STA(T)
LDQ1(KO)
LDL(77B)
ENI(O)
SLJ(L+6)
STA(T)
ZRO(O)
SLJ(L+3)
STA(T)
EXF7(21B)
STQ(T)
EXF7(21B)
SLJ(L-12)
SLJ(1W)
STA(T)
EXF7(21B)
ZRO(O)
SLJ(L+2)
ZRO(O)
SLJ4(1FLAG)
SAL(Z+2)
INA(-20B)
IJP2(L-1)
ENI(2)
STA(T)
AJP1(L+2)
SLJ(L+13)
LDQ1(KO)
LDL(77B)
ENI(O)
AJP(L+8)

```

```

TAG CONTROL WORDS
LOOP
SHIFT MARK
LOOP
POSITION RESULTS
LOOP
RETURN
PROCESS TYPEWRITER
SEARCH FOR TERMINAL BLANKS
STORE LENGTH OF ENTRY
TABLE LOOKUP
JUMP ON UPPER CASE
JUMP ON LOWER CASE
CARRIAGE SHIFT
PRINT CHARACTER
LOOP
SENSE CARRIAGE DOWN
SHIFT CARRIAGE DOWN
EXIT
PROCESS PAPER TAPE
CURRENT CASE
SEARCH FOR TERMINAL BLANKS
STORE LENGTH OF ENTRY
SENSE SPACE
TABLE LOOKUP
JUMP ON CARRIAGE UP
JUMP ON CARRIAGE DOWN

```

03033
03034
03035
03036
03037
03040
03041
03042
03043
03044
03045
03046
03047
03050
03051
03052
03053
03054
03055
03056
03057
03060
03061
03063
03064
03065
03066
03067
03070
03071
03072
03073
03074
03075
03076
03077
03100
03101
03102
03103
03104
03105
03106
03107
03110
03111
03112
03113


```

2X LDA(R6)
3X STA1(R+10)
4X ENA(N)
5X ENI2(4)
6X EXF(N)
7X EXF(N)
8X EXF(N)
9X SUB(R4)
10X INI1(-1)
11X INI1(1)
12X EXF7(N)
13X EXF7(N)
14X EXF7(N)
15X QJP1(1XA)
16X SLJ(1XA)
17X IJP2(L+2)
18X IJP2(L+1)
19X AJP(3X)
20X EXF(N)
21X LIL1(N)
22X ENA(T6)
23X ENA(T9)
24X ENA(T12)
25X ENA(TOX)
26X LIU1(2X)
27X SIL1(L-1)
28X SAU(6X)
29X SLJ(11X)
30X SLJ4(5G)
31X AJP(1XA)
32X SLJ(11X)
33X SLJ4(5G)
34X AJP(3X)
35X SLJ(11X)
36X EXF(62560B)

```

* TAPE WRITING ROUTINE

```

1Y SLJ(N)
   ENI(0)
   SAU(9Y)
   SAL(2Y)
   SAL(8Y)
   INA(3)
   SAU(11Y)
   SAU(6Y)
   INA(1)
   SAU(7Y)
   SLJ4(1FLAG)
   SAU(2Y)
   SCL(777B)
   SAL(4Y)
   SAL(10Y)
   SAU(5Y)
   INA(2)
   SAU(12Y)
   SAU(8Y)
   SAU(13Y)

```

```

ENI(0)
IJP1(L)
SAL(N)
LIL1(N)
EXF7(N)
EXF7(N)
LDA(R)
LILS(48)
LDA1(Z)
SUB(R4)
SLJ(15X)
SLJ(L+4)
SLJ(13X)
AJP(3X)
ZRO(0)
SLJ(14X)
SLJ(13X)
QJP(3X)
EXF7(N)
SLJ(4X)
SLJ(17X)
SLJ(17X)
SLJ(13X)
SAL(9X)
ENA(R+10)
SAL7(2X)
SAL(13R)
ZRO(0)
ZRO(0)
AJP3(3X)
ZRO(0)
ZRO(0)
AJP3(3X)
ZRO(0)
SLJ(1X)

```

SELECT UNIT
ACTIVATE CHANNEL

SENSE ENDFILE
SENSE PARITY ERROR
SENSE LENGTH ERROR
SENSE FOR BAD RECORD
EXIT FIVE PARITY ERRORS
SENSE FIVE LENGTH ERRORS
SENSE FOR BAD RECORD
BACKSPACE
RETURN TO REREAD
SET TO INDICATE TROUBLE

SET TO REREAD NEW LENGTH

RETURN TO REREAD
TROUBLE EXIT
SENSE ACTION TO TAKE
TROUBLE EXIT
SENSE ACTION TO TAKE

EXIT/ENTRY
INITIALIZE

03171
03172
03173
03174
03175
03176
03177
03200
03201
03202
03203
03204
03205
03206
03207
03210
03211
03212
03213
03214
03215
03216
03217
03220
03221
03222
03223
03224
03225
03226
03227
03230
03231
03232
03233

03234
03235
03236
03237
03240
03241
03242
03243
03244
03245

1SENS	EXF7(32505B)	SLJ(3ST00)	160-3/4 INTERRUPT	03317
	EXF7(52505B)	SLJ(5ST00)	160-5/6 INTERRUPT	03320
	EXF7(00111B)	SLJ(1CLR)	DIVIDE FAULT	03321
	EXF7(00121B)	SLJ(1CLR)	SHIFT FAULT	03322
	EXF7(00131B)	SLJ(1A)	OVERFLOW FAULT	03323
	EXF7(00141B)	SLJ(1CLR)	EXPONENT FAULT	03324
	SLJ(L+1)	SLJ(N)	CR INTERRUPT	03325
	SLJ(L+1)	SLJ(N)	CR 3 READY READ	03326
	SLJ(L+1)	SLJ(N)	CH 5 READY READ	03327
	SLJ(L+1)	SLJ(N)	CH 4 READY WRITE	03330
	SLJ(L+1)	SLJ(N)	CH 6 READY WRITE	03331
	SLJ(L+1)	SLJ(N)	CARD READER/PUNCH READY	03332
	SLJ(L+1)	SLJ(N)	PAPER TAPE READER END OF TAPE	03333
	SLJ(L+1)	SLJ(N)	DD65 DISPLAY INTERRUPT	03334
	SLJ(L+2)	LIJ1(Z+7)	ERROR INTERRUPT	03335
	SIL1(L+1)	ENI(O)	LOCK-OUT	03336
	ENI1(N)	SLJ(N)	RETURN TO PROGRAM	03337
1CLR	EXF(70B)	SLJ(Z+7)	CLEAR ARITHMETIC FAULT	03340
2CLR	EXF(11100B)	SLJ(Z+7)	CLEAR CR INTERRUPT	03341

* CHANNEL 3-4 SATELLITE INTERRUPT ROUTINE

3ST00	STA(3TEMP)	STQ(3TMP)	SAVE A/Q REGISTERS	03342
	SIU3(3PX22)	EXF(32505B)	SAVE B3 / REL INTERRUPT	03343
	LDA(3IBC)	STA(3TBF)	SET 1 WORD SWITCH	03344
	ENA(3STCC)	SAL(3STBB)	SET LOOP SWITCH	03345
3STAA	EXF(32503B)	EXF7(32000B)	SEL 160-1604 / WAIT READY	03346
	SIU1(3PXIT)	EXF7(42000B)	WAIT WRITE READY	03347
	LDA(3STBF)	SAL(Z+3)	SET TERMINAL ADDRESS	03350
	ARS(30B)	SAU(L+2)	SET INITIAL ADDRESS	03351
	EXF(32001B)	EXF(L+2)	READ BINARY / 160 ACTION REQ.	03352
	EXF3(N)	EXF7(318)	ACTIVATE / WAIT CH INACTIVE	03353
	ENI(O)	EXF7(32000B)	PASS / WAIT READY TO READ	03354
	ENA(3EXIT)	EXF(42500B)	SET FOR LOOP / REL DIRECT	03355
3STBB	SIL2(3PXIT)	SLJ(N)	SWITCH	03356
3STCC	EXF7(42521B)	SLJ(3STD)	EXIT NO F2 / GO CASE S.R. OUTPUT	03357
	SSK(3STBF)	SLJ(L+2)	INPUT OR OUTPUT / GO OUTPUT	03360
	SAL(3STBB)	SLJ(3TAA)	SWITCH TO EXIT / GO INPUT	03361
3STCD	EXF(42503B)	EXF7(42000B)	SEL 1604-160 / WAIT READY	03362
	ENI(O)	EXF7(32000B)	WAIT READY READ	03363
	LDA(3STBF)	SAL(Z+4)	SET TERMINAL ADDRESS	03364
	ARS(30B)	SAU(L+2)	SET INITIAL ADDRESS	03365
	EXF(42001B)	EXF(L+2)	WRITE BINARY / 160 ACTION REQ.	03366
	EXF4(N)	EXF7(418)	ACTIVATE / WAIT CH INACTIVE	03367
	ENI(O)	EXF7(42000B)	PASS / WAIT READY TO WRITE	03370
3EXIT	EXF(42500B)	EXF(42520B)	REL DIRECT / REL F2	03371
	EXF(42502B)	EXF(42502B)	TAKE READ AND WRITE CONTROL	03372


```

3PXIT ENI1(N)
3PX22 ENI3(N)
      LDQ(3QTEMP)
3STD  LDA(3QTBFF)
      SAL(3STED)
      SSK(3STBF)
      ALS(3B)
      INA(-1B)
      SAU(L+2)
      AJP(3STEE)
      ENA1(N)
      ISK1(N)
      ALS(3B)
3STEE ENI(O)
3STED SLJ(3EXIT)
      ZRO(3STBF)
3IBC  ZRO(O)
3STBF ZRO(O)
3TEMP ZRO(O)
3QTMP ZRO(O)

```

```

ENI2(V)
LDA(3TEMP)
SLJ(Z+7)
SAL(L+7)
ENI1(1)
SLJ(3STEE)
ARS(33B)
ENI(O)
ARS(17B)
SAU(L+2)
STA1(V)
SLJ(L-1)
ARS(33B)
SLJ4(V)
ZRO(O)
ZRO(3TEMP)
ZRO(O)
ZRO(O)
ZRO(O)

```

```

• RESTORE B1 AND B2
• RESTORE B3 / A REG.
• RESTORE Q REG / EXIT VIA 7
• PRIME FOR ARG INPUT JUMP
• AND S.R. RETURN JUMP
• CHECK ARG MODE / GO LDA ARG.
• PACK
• THE
• ARGUMENT
• ADDRESSES
• INTO
• SUBROUTINE
• POSITION A REGISTER ARGUMENT
• GO EXECUTE SUBROUTINE
• GO EXIT FROM INTERRUPT
• INITIAL BUFFER CONTROL
• BUFFER CELL
• TEMP STORAGE
• TEMP STORAGE FOR Q REGISTER

```

```

03373
03374
03375
03376
03377
03400
03401
03402
03403
03404
03405
03406
03407
03410
03411
03412
03413
03414
03415

```

* CHANNEL 5-6 SATELLITE INTERRUPT ROUTINE

```

5ST00 STA(5TEMP)
      SIU3(5PX22)
      LDA(5IBC)
      ENA(5STCC)
      EXF(52503B)
5STAA SIU1(5PXIT)
      LDA(5STBF)
      ARS(3OB)
      EXF(52001B)
      EXF5(N)
      ENI(O)
      ENA(5EXIT)
      SIL2(5PXIT)
      EXF7(62521B)
5STCC SSK(5STBF)
      SAL(5STBR)
      EXF(62503B)
5STCD ENI(O)
      LDA(5STBF)
      ARS(3OB)
      EXF(62001B)
      EXF6(N)
      ENI(O)
      EXF(62500B)
5EXIT EXF(62502B)
5PXIT ENI1(N)

```

```

STQ(5QTMP)
EXF(52505B)
STA(5STBF)
SAL(5STBR)
EXF7(52000B)
EXF7(62000B)
SAL(Z+5)
SAU(L+2)
EXF(62504B)
EXF7(51B)
EXF7(52000B)
EXF(62500B)
SLJ(N)
SLJ(5STDD)
SLJ(L+2)
SLJ(5STAA)
EXF7(52000B)
EXF7(52000B)
SAL(Z+6)
SAU(L+2)
EXF(62504B)
EXF7(61B)
EXF7(52000B)
EXF(62502B)
ENI2(V)

```

```

• SAVE A/Q REGISTERS
• SAVE B3 / REL. INTERRUPT
• SET 1 WORD SWITCH
• SET LOOP SWITCH
• WAIT 160-1604 / WAIT READY
• WAIT READY WRITE
• SET TERMINAL ADDRESS
• SET INITIAL ADDRESS
• READ BINARY / 160 ACTION REQ.
• ACTIVATE / WAIT CH INACTIVE
• PASS / WAIT CH INACTIVE
• SET FOR LOOP / REL DIRECT
• SWITCH
• EXIT NO F2 / GO CASE S.R.
• INPUT OR OUTPUT / GO OUTPUT
• SWITCH TO EXIT / GO INPUT
• SET 1604-160 / WAIT READY
• WAIT READY READ
• SET TERMINAL ADDRESS
• SET INITIAL ADDRESS
• WRITE BINARY / 160 ACTION REQ.
• ACTIVATE / WAIT CH INACTIVE
• PASS / WAIT / READY TO WRITE
• REL DIRECT / REL F2
• TAKE READ AND WRITE CONTROL
• RESTORE B1 AND B2

```

```

03416
03417
03420
03421
03422
03423
03424
03425
03426
03427
03430
03431
03432
03433
03434
03435
03436
03437
03440
03441
03442
03443
03444
03445
03446
03447

```



```

5PX22 ENI3(N) LDA(STEMP)
5STD0 LQ(5QTMP) SAL(Z+7)
      LDA(5STBF) ENI1(1)
      SAL(5STBF) SLJ(5STEE)
      ALS(3B) ARS(33B)
      INA(-1B) ENI(O)
      SAU(L+2) ARS(17B)
      AJP(5STEE) SAU(L+2)
      ENA1(N) STA1(V)
      ISK1(N) SLJ(L-1)
      ALS(3B) ARS(33B)
5STED ENI(O) SLJ4(V)
      SLJ(5EXIT) ZRO(O)
      ZRO(5STBF) ZRO(STEMP)
5IBC ZRO(O) ZRO(O)
5STBF ZRO(O) ZRO(O)
5TEMP ZRO(O) ZRO(O)
5QTMP ZRO(O) ZRO(O)

```

```

• RESTORE B3 / A REG.
• RESTORE Q REG / EXIT VIA 7
• PRIME FOR ARG INTRN JUMP
• AND S.R. RETURN JUMP
• CHECK ARG MODE / GO LDA ARG.
• PACK THE
• ARGUMENT
• ADDRESSES
• INTO
• SUBROUTINE
• POSITION A REGISTER ARGUMENT
• GO EXECUTE SUBROUTINE
• GO EXIT FROM INTERRUPT
• INITIAL BUFFER CONTROL
• BUFFER CELL
• TEMP STORAGE
• TEMP STORAGE FOR Q REGISTER

```

```

03450
03451
03452
03453
03454
03455
03456
03457
03460
03461
03462
03463
03464
03465
03466
03467
03470
03471

```

* 160 SATELLITE LIBRARY CALL ROUTINE HANDLER

```

160CL STA(2NAME) EXF(63511B)
      LDA(BLOCKS) STA(150TP)
      LDA(2X) STA(2XSTR)
      LDA(5X) STA(5XSTR)
      LDA(6X) STA(6XSTR)
      LDA(2NAME) ENI1(10B)
      EQS1(IDENT) SLJ(1LAST)
      LDA1(SIZE) SAU(L+2)
      SSK(1SWT) SLJ(1LAST)
      ENI3(N) LDA(2V)
      STA(2VSTR) ENA(1B)
      SAL(2V) SLJ4(1V)
      LDA(2VSTR) STA(2V)
      ENI(O) SLJ4(160IN)
      RSO(160TP) AJP(1LAST)
      SSK(2VSTR) SLJ(1SRCH)
      LDA1(SATBUF) STA(2VAME)
      IJP3(160UT) SLJ(1LAST)
      ISK1(65B) SLJ(L-2)
      SLJ(160LA) ZRO(O)
      SLJ4(160WD) ZRO(O)
      SLJ(160LC) ZRO(O)
      SLJ(N) ENA(2XSTR)
      EXF(62503B) EXF7(52000B)
      SAL(Z+6) EXF7(52000B)
      EXF6(2NAME) EXF(62504B)
      EXF7(51B)

```

```

• STORE NAME / TURN OFF LIGHT 1
• SET BLOCK READ LIMIT
• PROTECT RESIDENT
• BINARY READ
• ROUTINE
• SEARCH IDENTIFY PROGRAM
• TO IDENTIFY WORDS TO BE READ
• SET NR OF WORDS TO BE READ
• EXIT IF BINARY READ LOCKOUT
• SAVE REWIND ARGUMENT
• REWIND LIBRARY TAPE
• RESTORE REWIND ARG.
• GO READ LIB BLOCK
• CHECK FOR LIMIT
• CHECK HIT FLAG
• SET WORD IN OUTPUT BUFFER
• CHECK IF LAST WORD
• LOOP UNTIL BLOCK IS EXHAUSTED
• GO READ NEXT BLOCK
• GO WRITE OUT WORD
• RETURN TO READ NEXT WORD
• WRITE OUT WORD ROUTINE

```

```

03472
03473
03474
03475
03476
03477
03500
03501
03502
03503
03504
03505
03506
03507
03510
03511
03512
03513
03514
03515
03516
03517
03520
03521
03522
03523
03524

```

[illegible]

* SATELLITE MONITOR JOB ROUTINE HANDLER

```

1SWTH      SLJ(N)
            SAL(8V)
            ENI(O)
            ENI(O)
            LDA(2M)
            LDA(3M)
            LDA(4K)
            LENA(2INTM)
            ENA(66B)
            ENA(1INTM)
            LDL(OSWT)
            AJP(L+1)
            STQ(2M)
            STA(7RA)
            STA(7WA)
            STA(7WC)
            SAU(Z+30B)
            INA(-2B)
            SLJ(1SWTH)
            SLJ4(1STRD)
            SLJ4(1STWT)

2SWTH
3SWTH      LDA(2M)
            SLJ4(7V)
            ENQ(7777B)
            STA(2MSTR)
            STA(3MSTR)
            STA(4CSTR)
            SAL(Z+25B)
            STA(4C)
            SAL(11MA)
            ENQ(65B)
            STQ(3M)
            LDA(2SWTH)
            LDA(3SWTH)
            STA(7WB)
            ENA(2RCVR)
            ENA(11MA)
            SAU(1SWTH)
            ZRG(O)
            ZRO(O)
            ZRO(O)

```

```
LIB BLOCK READ ROUTINE
SEARCH TAPE FOR NAME
LOOP UNTIL HIT
SET HIT FLAG
GO TRANSMIT LIBRARY ROUTINE
GO WRITE OUT LAST WORD
RESTORE
TAPE READ
ROUTINE
TURN ON LIGHT 1 / REL COMM 1
EXIT, LOCK-OUT SET
REWIND LIB / NORMAL EXIT
```

SAVE ON-LINE	035556
MONITOR JOB	035557
DESCRIPTION	035560
	035561
SAVE MONITOR	035563
INPUT/OUTPUT	035564
	035565
SET FOR ARGUMENTS	035567
SET SAT FLEX ERROR	035570
SET SWITCH DIVERTER	035571
CHECK IF DESIRE PUNCH	035572
OUTPUT	035573
IF SO, SET OUTPUT ARG	035574
SET FLEX INPUT	035575
PRIME FLEX READ	035576
PRIME FLEX WRITE	035577
	036001
SET TROUBLE RECOVERY	035578
SET TO READ SAT JOB	035579
RETURN	035580
JUMP TO READ FLEX	035581
JUMP TO WRITE FLEX	035582

2MSTR	ZRO(0)	ZRO(0)	...	03602
3MSTR	ZRO(0)	ZRO(0)	...	03603
1INTM	SLJ(N)	ENAL(11MA)	...	03604
	ENAL(-0)	ENAL(Z+25B)	...	03605
2INTM	SLJ(N)	SLJ(Z+25B)	...	03606
	SLJ(N)	SLJ(N)	...	03607
1STRD	STA(5STBF)	STA(5STBF)	...	03610
	ALS(24)	ALS(24)	...	03611
	SAL(3STRD)	SAL(3STRD)	...	03612
	SAU(Z+7)	SAU(Z+7)	...	03613
	SAL(5STBB)	SAL(5STBB)	...	03614
	SIL2(5PXIT)	SIL2(5PXIT)	...	03615
	STQ(5QTMP)	STQ(5QTMP)	...	03616
	EXF(62540B)	EXF(62540B)	...	03617
2STRD	ZRO(T)	ZRO(T)	...	03620
3STRD	EXF(1000B)	EXF(1000B)	...	03621
1STWT	SLJ(N)	SLJ(N)	...	03622
	STA(5STBF)	STA(5STBF)	...	03623
	ALS(24)	ALS(24)	...	03624
	SAL(3STWT)	SAL(3STWT)	...	03625
	SAU(Z+7)	SAU(Z+7)	...	03626
	SIL2(5PXIT)	SIL2(5PXIT)	...	03627
	STQ(5QTMP)	STQ(5QTMP)	...	03630
	SLJ(5STCD)	SLJ(5STCD)	...	03631
3STWT	EXF(1000B)	EXF(1000B)	...	03632
1RCVR	SLJ(N)	SLJ(N)	...	03633
	STA(T)	STA(T)	...	03634
	EXF(21240B)	EXF(21240B)	...	03635
	LDA(2MSTR)	LDA(2MSTR)	...	03636
	LDA(3MSTR)	LDA(3MSTR)	...	03637
	ENAL(2A)	ENAL(2A)	...	03640
	LDA(3RCVR)	LDA(3RCVR)	...	03641
	LDA(4RCVR)	LDA(4RCVR)	...	03642
	STA(7WB)	STA(7WB)	...	03643
	LDA(4KSTR)	LDA(4KSTR)	...	03644
	EXF(52502B)	EXF(52502B)	...	03645
	ENAL(11A)	ENAL(11A)	...	03646
	ENAL(1SWTH)	ENAL(1SWTH)	...	03647
	STQ(R)	STQ(R)	...	03650
	STQ(OSWT)	STQ(OSWT)	...	03651
3RCVR	EXF1(T)	EXF1(T)	...	03652
4RCVR	EXF2(T)	EXF2(T)	...	03653
			...	03654
			...	03655

* DD65 SCOPE SIMULATOR FOR AN ON-LINE PRINTER

65DPY SLJ1(4A) EXF(77010B) . SELECT 1604 TO DD65 03656

40A744E4,	547,	30402,	20413
-I4 -+8,	11,	110,	704
TUK -7+2,	50,	533,	62176
+9/I +,	641,	10424,	25071
H+ C64	417,	22252,	6105
	147,	54075,	31351
	732,	15061,	66003
	710,	11575,	0
1 23232S,	14073,	76016	
QIM	100,	1	
-ZH1 XC4,	220,	65030,	12070
-Q-RA5 Z,	432,	72230,	2057
	244,	40002,	42311
- S2HAU/,	403,	45457,	26606
B5S2HAXQ,	74,	22027,	26211
+ S2I2,1,	14477,	37101	
E+7776+5,	630,	140702,	50401
	423,	12001,	2225
P C4P AV,	602,	45645,	56033
FORTBIN	3,	14000,	0
MAPTBIN	3,	1200,	0
GRAPH	4,	4770,	0
GRAPH2	17,	44536,	0
ROOTS	31,	7720,	0
ROOTS2	11,	6414,	0
SDRT	6,	2013,	0
FORTMAP	4,	12200,	0
MANLIB	7,	15000,	0
FORTRUN	5,	14000,	0
FORTLIB	6,	14000,	0
MTTEST	0,	62045,	0
COMPSYM	4,	62751,	0
CONPAC	2,	353,	0
REMCON	2,	355,	0

ENCLOSURES

A. IDENTIFICATION

TITLE: 160 Grafplot Routine

PROGRAMMERS: R.L. HOGG / D.C. GLOVER

ORGANIZATION: U.S. Naval Postgraduate School
Monterey, California

DATE: 1 April 1963

B. PURPOSE

To plot multiple curved graphs on the CalComp 165 plotter. Graph titling and axes annotation are provided. The input to the routine is a specially prepared magnetic tape, see the CDC 160-A program of the same name for the input tape format. This routine is a truncated version of the 160-A program, interpolation has been deleted in order to operate in the reduced memory of the 160 computer.

C. USAGE

1. Normal Operation

- a. Load the specially prepared magnetic tape input (See 160-A write-up) on unit #1. Select unit for Binary parity.
- b. Load bi-octal grafplot program at address 0000.
- c. Position the plotter pen at position where the lower left corner of the graph is to be plotted.

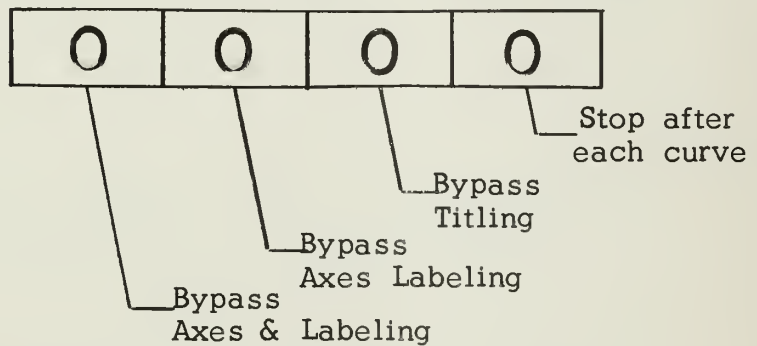
d. Starting Addresses

<u>Location</u>	<u>Contents of "A"</u>	<u>Results</u>
0000	# of graphs to be plotted	Reads tape and plots # of graphs called for. If A=0, all graphs will be processed.
0001		Skip to next graph.
0002		Backspace to beginning of current graph.
0003		Back-up one graph.
0004		Same as a 0000 start.
0010		Rewind tape to Load Pt.
0040	Graph # in A	Skips forward N-1 graphs, into position to plot graph #N.

e. Optional Running Modes

Four modes of operation are provided. To select the mode desired, execute a start at cell 0020 with the following "A" register arguments set. This primes the routine for the mode selected. Remaining graphs will be plotted in this mode, operations as previously described.

"A" REGISTER FORMAT for MODE SELECTION



NOTE: Entry of any non-zero octal number, constitutes setting an argument.

f. Normal Stops

<u>Location</u>	<u>Situation</u>
0004	Normal halt after operation is completed
1157	Optional halt after each curve. Run to continue.

g. Error Stops

0130	*Parity - Title location data
0135	**
0173	*Parity - Title character
0200	**
0373	*Parity - Axed data
0400	**
1125	*Parity - Curve data
1163	**

* Re-Select Start to continue questionable plot.

** Unable to Continue / Skip to next graph to proceed.

A. IDENTIFICATION

TITLE: 160-A Graph Plot Program

DATA CENTERS IDENT: *B001.

PROGRAMMERS: Maurice D. Weir, Milton Spritzer, D. W. McIlhenny,
Harry Farnsworth

ORGANIZATION: Fleet Numerical Weather Facility, Monterey, California

DATE: 15 August 1962

REVISED: 4 February 1963, by R. Hogg and D. Glover

B. PURPOSE

To plot N curves on the CONTROL DATA 165 Plotter, complete with titling information and annotated axes, from a specially prepared magnetic tape. SEE TAPE FORMAT.

C. USAGE**1. Computer Operational Procedure****a. Normal operation:**

- 1) Specially prepared magnetic tape on Unit #1. (SEE TAPE FORMAT)
- 2) Load bi-octal tape at location 0, bank 0. Set direct bank equal to zero.
- 3) Position the plotter pen such that the cross-hairs line up in the following INITIAL POSITION: LEFT most VERTICAL LINE and one of the HALF-INCH HORIZONTAL LINES on the graph paper.

CAUTION: User should indicate this POSITION in relation to a previously plotted graph. This prevents the forthcoming graph from being written over a previously plotted one.

4) Starting Address

<u>Location</u>	<u>Contents of A</u>	<u>Result</u>
0000	No. of graphs to be plotted.	Reads tape and plots number of graphs called for. If A = 0, tape will be completely processed.

<u>Location</u>	<u>Contents of A</u>	<u>Result</u>
0001		Skip to next file.
0002		Backspace to beginning of current graph.
0003		Back up one complete graph.
0010		Rewind tape to load point.
0040	Graph No. N	Skip N-1 graphs, and stop

NOTE: Normal halt for above operations is location 4.

- 5) Plotter pen will position itself, draw and annotate the horizontal and vertical axes. Pen will finish by driving back such that the cross-hairs are now lined up with the INTERSECTION of the HORIZONTAL and VERTICAL AXES just drawn.
- 6) Plotter will plot N curves with their identification (if any). Pen will finish by driving back to the INTERSECTION of the HORIZONTAL and VERTICAL AXES.

7) Normal Stop

(P) = 0004 - Normal halt for all options.

8) Error Stops

- a) The following error stops, which occur during the read from the magnetic tape, have two options. These options are:

- i) RUN for questionable plot, or
- ii) SET selective jump switch #2 and RUN. This will result in by-passing that particular section of the program (e.g., titling, annotation of axes, or curve).

The error stops are:

- P = 0135 - Parity error in title locations
- P = 0176 - Parity error in titling data
- P = 0401 - Parity error in axes data
- P = 1144 - Parity error in curve data

- b) The following error stops, which occur during the read from the magnetic tape, have no options. If any occur, operator should check to make certain magnetic tape unit is on unit #1, tape is at load point, tape is ready, 163-2 units are on. Then set (P) according to step 4).

The error stops in this category are:

P = 0142 - titling location

P = 0203 - titling data

P = 0406 - Axes data

P = 1200 - curve data

P = 1103 - error in number of interpolations. RUN.

b. Operational variations:

The Graph Plot program contains some special operational procedures which allow for variations in the graph by altering the Normal operation.

1) Option to by-pass titling:

After the plotter pen has been positioned in the previously defined INITIAL POSITION, SET selective jump switch #2. Set (P) according to step a.4) and RUN. Titles will be read from magnetic tape but not plotted. Program will make NORMAL STOP P = 0361, and pen will not have moved.

2) Option to by-pass axes annotation:

With plotter pen at AXES INTERSECTION and at SPECIAL STOP P = 0361, SET selective jump switch #2 and RUN (do not clear). Axes data will be read from magnetic tape but not plotted. Program will make NORMAL STOP P = 1051, and pen will not have moved.

3) Option to PLOT - one-curve-STOP:

With plotter pen at AXES INTERSECTION and at SPECIAL STOP P = 1051, SET selective stop switch #1 and RUN. After curve is plotted, pen will drive back to AXES INTERSECTION, and program will make SPECIAL STOP P = 1174 before the next curve record is read in. Pen-ink may now be changed for next curve, etc. RUN to read in and plot next curve.

4) Option to by-pass one curve at a time:

At SPECIAL STOP P = 1051, SET selective jump switch #2 and RUN. Curve data will be read in, and program will make SPECIAL STOP P = ~~1065~~ 1136. If selective jump switch # 2 remains set when RUN is set, next curve will be read in, and program will again make SPECIAL STOP P = ~~1065~~. If selective jump switch # 2 is not set when RUN is set, next curve will be read in and plotted, etc.

2. Space Requirements

- a. Program: 0 - 4137 in Bank zero.
- b. Curve data and interpolation: 100 - 7777 in Bank one.

3. Temporary Storage

Locations 50 - 77 in Bank zero.

4. Tape Mountings

Magnetic tape must be read from unit # 1.

5. Caution to Users

Do not step through program. Read write-up carefully before using the Graph Plot Program.

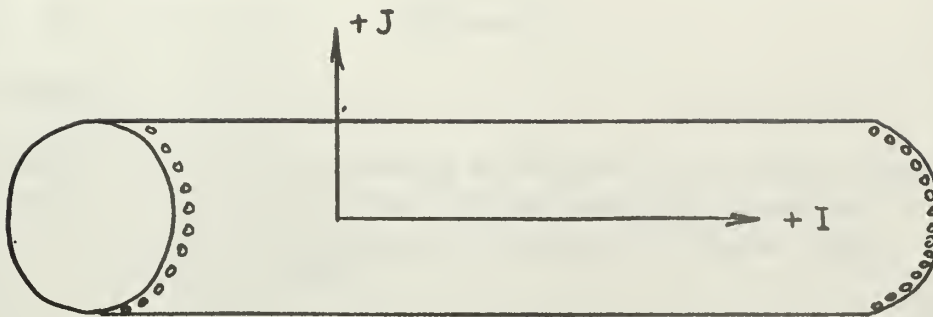
D. TAPE FORMAT

Each magnetic tape to be used by this program must have one or more records (as indicated) in each of the following categories:

- I Titling data - multiple records
- II Axes data - one record
- III Curve data - multiple records
- IV End of File.

These records must be on the magnetic tape in the order defined above and in the format to be specified below.

AXES DEFINED AS FOLLOWS:



I = horizontal axis

J = vertical axis

I Titling data

1. Record 0

- a. 12 bit binary mode
- b. 4 160-A words (i.e., 1 1604 word)
- c. Format:

word 0
(I_T)

I distance of initial pt. of first title
from the INITIAL POSITION, given as
a signed ΔI movement in inches times
 100_d .

word 1
(J_T)

J distance of initial pt. of first title
from the INITIAL POSITION, given as a
signed ΔJ movement in inches times
 100_d .

word 2

Orientation of titles defined as follows:

0004

. FNWF

0001

. FNWF

0002

. FNWF

0003

. FNWF

word 3

Number of lines in title given as a 12
bit integer. (OCTAL)

2. Records 1 to T

- a. BCD characters in 6-bit binary mode.
- b. Maximum length of 120_d characters where 119_d is number
of characters to be plotted.
- c. Format:

character 0

Size of letters in title line. Size X is
defined to be "X times one-tenth inch".
The width of a letter is approximately
equal to its height.

character 1
:
:
character 119

Title line in BCD. Not necessary to
fill out this line

II Axes data

a. 1 record in 12 bit binary mode

b. 20_d 160-A words

c. Format:

word 0
(I_H)

I distance of initial point of horizontal axis
from the INITIAL POSITION, given as a signed
 ΔI movement in inches times 100_d .

word 1
(J_H)

J distance of initial point of horizontal axis
from the INITIAL POSITION, given as a signed
 ΔJ movement in inches times 100_d .

word 2
(L_H)

Length of the horizontal axis given as a signed
 ΔI movement from where the axis begins in inches
times 100_d .

word 3
(I_{HL})

I distance of initial point of first horizontal axis
label from the INITIAL POSITION, given as a
signed ΔI movement in inches times 100_d .

word 4
(J_{HL})

J distance of initial point of first horizontal
axis label from the INITIAL POSITION, given as
a signed ΔJ movement in inches times 100_d .

word 5
(I_{HL})

I distance between horizontal labels, given
as a signed ΔI movement in inches times 100_d .

word 6
(V_H)

First value which is to be plotted for the hori-
zontal axis labeling. This is a signed 12 bit
integer which will be converted to BCD. Its
maximum value cannot exceed $+999_d$.

word 7
(V_H)

Value of the ΔI distance between horizontal
axis labels given as a signed 12 bit integer.

word 8
(N_H)

Number of horizontal axis labels to be plotted.
This is always a positive 12 bit integer.

word 9 (S _H) (O _H)	<u>Upper six bits:</u> Size of letters in horizontal axis labeling. <u>Lower six bits:</u> Orientation of horizontal axis labels. Orientation previously defined.
word 10 (I _V)	I distance of initial point of vertical axis from the INITIAL POSITION, given as a signed ΔI movement in inches times 100_d .
word 11 (J _V)	Vertical axis analogy to word 1.
word 12 (L _V)	Vertical axis analogy to word 2.
word 13 (I _{VL})	Vertical axis analogy to word 3.
word 14 (J _{VL})	Vertical axis analogy to word 4
word 15 (J _{VL})	Vertical axis analogy to word 5
word 16 (V _V)	Vertical axis analogy to word 6.
word 17 (V _V)	Vertical axis analogy to word 7.
word 18 (N _V)	Vertical axis analogy to word 8.
word 19 (S _V) (O _V)	<u>Upper six bits:</u> Size of letters in vertical axis labeling. <u>Lower six bits:</u> Orientation of vertical axis labels.

III Curve data

- a. Multiple records in 12 bit binary mode.
- b. m words long where m has maximum value of 3677_8 .
- c. Each I, J must be given as a distance from the INITIAL POSITION, as a signed $\Delta I, \Delta J$ movement in inches times 100_d .

d. Format:

word 0	0000
word 1	0000
word 2	0000
word 3	number of interpolations (00 - 07)
word 4	I ₁
word 5	J ₁
word 6	I ₂
word 7	J ₂
:	
:	
word m-5	3777 end flag
word m-4	3777 end flag
word m-3	curve name. 4 BCD characters,
word m-2	packed two to a word.
word m-1	<u>upper six bits:</u> letter size of curve name
	<u>lower six bits:</u> orientation of curve name

IV End of File

E. TIMING

Each graph takes 2 - 3 minutes to plot, including titling, annotated axes, and interpolation.

F. METHOD

1. General

All location points on the graph are interpreted as being relative to the (0, 0) INITIAL POSITION previously defined. All lengths, intervals, etc. must have the appropriate sign according to the definition of the(I, J) axes. Information is plotted as follows: titles, horizontal axis, horizontal axis annotation, vertical axis, vertical axis annotation, and finally the curve(s) (with interpolations, if any). The program uses TAP for reading the magnetic tape input, OVDR for driving to a position with the pen up, LINE for joining given points with the pen down, and LABEL for titling.

2. Titles

The initial points from the magnetic tape input are used to position the pen in the desired location on the graph for the first title. The

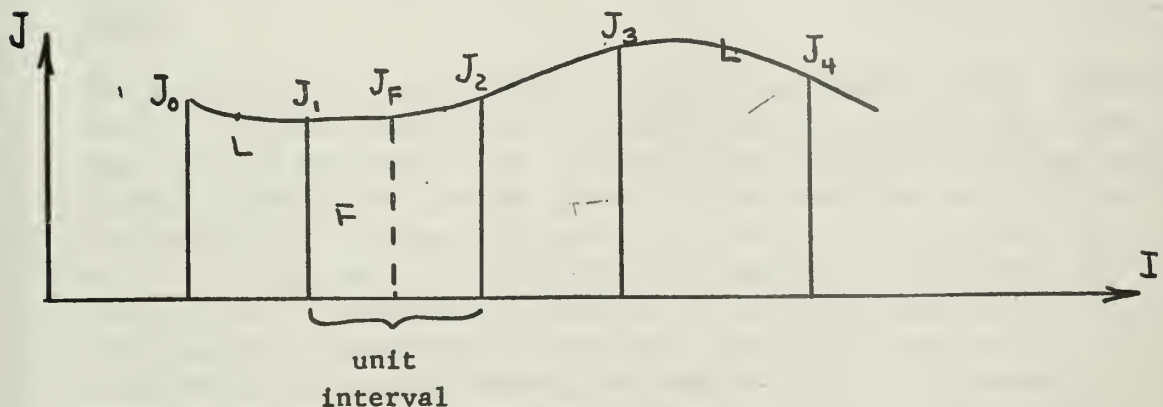
first title is plotted, and if there are additional titles, the program calculates the appropriate spacing between lines according to the orientation and the letter size of the incoming title. The program keeps track of all pen movements so that when the last title is plotted, pen drives back to the (0, 0) INITIAL POSITION.

3. Annotation of Axes

The initial points from the magnetic tape input are used to position the pen in the desired location on the graph for the start of the horizontal axis. The axis is drawn, and then the pen moves into position to begin the first horizontal axis label. This first label is plotted according to its size and orientation; the remaining labels being computed and then plotted at the desired intervals, based upon the initial value, the scaling, and the number of labels desired. At the completion of the horizontal axis annotation, the pen positions itself at the start of the vertical axis. The axis is drawn and labelled as in the horizontal axis annotation. The program keeps track of all pen movements so that when the vertical axis annotation has been completed, pen drives back to the INTERSECTION of the HORIZONTAL and VERTICAL AXES.

4. Interpolation

This is a four-point interpolation method based on the Bessel Interpolation Equations.



The interpolation scheme fits the four points to a quadratic curve, computing the interpolated point J_F by the following equation:

$$J_F = J_1 + F (J_2 - J_1) + \frac{F (F - 1)}{4} (J_0 + J_3 - J_2 - J_1)$$

The program uses $F = 1/2$.

This interpolation method depends upon equal I intervals. Since it cannot perform an interpolation between the first and last pairs of J values, a linear interpolation is performed at these two positions (marked "L" in the diagram). This linear interpolation occurs at the half-interval of both end pairs, which allows the user to make more than one interpolation by maintaining equality of I intervals.

The number of interpolations X desired must comply with the following inequality:

$$\frac{\Delta I}{2^x} > \frac{1}{100}$$

The program has set the maximum number of interpolations as 07. If the number of interpolations indicated (see TAPE FORMAT, "curve data") exceeds this maximum, program will interpolate 07 times only, and indicate an error in the number of interpolations indicated.

Caution: Because of the necessary storage space for the curve data points, the maximum number of (I, J) points that can be plotted after interpolation is 980_d.

WARNING: Since the interpolation depends upon equal spacing of the I intervals, any curve which does not have equally spaced points will not be properly interpolated. This applies to multiple-valued functions and asymptotic values.

5. Curves

The first (I, J) curve data point from the magnetic tape input is used to position the pen in the desired location on the graph for the start of the curve plot. Curve is then plotted by joining successive points by a straight line. At the completion of plotting the curve (with the desired number of interpolations, if any), program labels curve (if desired) with indicated orientation and letter size. Program keeps track of all pen movements so that when the curve labelling is completed, pen drives back to the INTERSECTION of the HORIZONTAL and VERTICAL AXES. If another curve is to be plotted, program proceeds in the same manner as for the first curve. When the final curve is plotted, pen will be positioned at the INTERSECTION of the HORIZONTAL and VERTICAL AXES.

Caution: User may not desire a curve label, in which case blanks are indicated on the magnetic tape for the four BCD characters.

(see TAPE FORMAT, "curve data.") Nevertheless, an orientation and letter size must still be indicated since they are required by the LABEL subroutine.

FINIS

A. IDENTIFICATION

TITLE: 1604 Graph Plotting Subroutine

DATA CENTERS IDENT: *A002.

PROGRAMMERS: R. L. Hogg and D. C. Glover

ORGANIZATION: Naval Postgraduate School, Monterey, California

DESCRIPTION WRITE-UP: E. A. Campbell, CDC

DATE: 5 March 1963

B. PURPOSE

To write data on a magnetic tape for off-line plotting on the CONTROL DATA 165 plotter. The tape format is compatible with the 160-A Graph Plot Program (Data Centers Ident *B001.) which is used to drive the 165 plotter.

C. USAGE

1. Calling Sequence

a. FORTRAN-62

CALL GRAFPLOT(MODCURV, NUMPTS, X, Y, LABEL, INTERPS, SFX, SFY, IXOFFS, IYOFFS, IWIDE, IHIGH, MODE, TITLESIZ, TITLE1, TITLE2,, TITLEN)

CALL AUTOGRAF(MODCURV, NUMPTS, X, Y, TITLE1, TITLE2,, TITLEN)

b. CODAP1

Either:	ENA	(MODCURV)
	ENQ	(NUMPTS)
	RTJ	GRAFPLOT
+	ZRO	X
	ZRO	Y
	ZRO	LABEL
	ZRO	INTERPS
	ZRO	SFX
	ZRO	SFY
	ZRO	IXOFFS
	ZRO	IYOFFS
	ZRO	IWIDE
	ZRO	IHIGH
	ZRO	MODE
	ZRO	TITLESIZ
	ZRO	TITLE1
	ZRO	TITLE2

:

ZRO TITLEN

```

Or:      ENA      (MODCURV)
          ENQ      (NUMPTS)
          RTJ      AUTOGRAF
+        ZRO      X
          ZRO      Y
          ZRO      TITLE1
          :
          :
          ZRO      TITLEN
+        RTJ      ERROR
  
```

2. Input Parameters

<u>Parameter Name</u>	<u>Value</u>	<u>Description</u>
*MODCURV	0	First and last curve for this graph
	1	First curve of many
	2	Intermediate curve
	3	Final curve for this graph
NUMPTS	1-900	Number of points
X		Location of X coordinates
Y		Location of Y coordinates
LABEL	4 BCD	
	Char.	BCD label for curve
	0	No label
	-0	Automatic sequencing of curves for No. 1-8.
INTERPS	0	No interpolation between coordinate points
	1-7	One to 7 interpolations between coordinate points
SFX	0	Program will find best scale factor
	F1. Pt. Number	Scale factor for X **
SFY	0	Program will find best scale factor
	F1. Pt. Number	Scale factor for Y **
IXOFFS	1-10	Number of inches to offset X axis
	0	No offset
IYOFFS	1-10	Number of inches to offset Y axis
	0	No offset
IWIDE	1-10	Length of X axis - inches
IHIGH	1-120	Length of Y axis - inches

*If MODCURV is 2 or 3, then only the first 6 parameters need to be repeated.

**Scale factor = 100/units per inch on plotted output.

<u>Parameter Name</u>	<u>Value</u>	<u>Description</u>
MODE	0	Automatically calculate best axes offset for origin at lower left
	1	Locate origin at center
	2	Locate origin at lower left
	3	Locate origin at upper left
	4	Locate origin at upper right
	5	Locate origin at lower right
TITLESIZ	1-24	Multiplier for lettering size. Basic size is .1 inches.
TITLE1		Location of first title. Title array must be at least 10 words of BCD characters. Fill unused characters with blanks.
:		
TITLEN		Location of Nth title array

3. Space Requirements

1120 decimal locations

6. Error Returns

No error return is used, however, an error return point is skipped in order to be compatible with FORTRAN subroutines.

8. Input/Output Tape Mountings

Plot data output on logical 48 through CO-OP Monitor.

9. Input/Output Tape Formats

See 160-A Graph Plot Program write-up for details. The basic form is titling information, axes annotation, and curve data with an end of file separating graphs. No special end of tape mark is used to terminate the tape.

13. Cautions to Users

- a. If scale factors are defined, then the appropriate axes offsets should also be defined.
- b. The normal width of the plotter paper is 10 inches. For best results, do not use a horizontal axis length greater than 9 inches.

- c. The number of interpolations N required must comply with the following inequality: $\Delta I / 2^n > 1/100$ where ΔI is the uniform (equal) increment along the horizontal axis. See reference for further details.

14. Equipment Configuration

CO-OP Monitor configuration with at least one output tape available.

15. References

This program uses 160-A Graph Plot Program (Data Centers Ident No. *B001) as a subset to drive the 165 plotter.

thesH679

Control system programming remote comput



3 2768 002 06875 1

DUDLEY KNOX LIBRARY