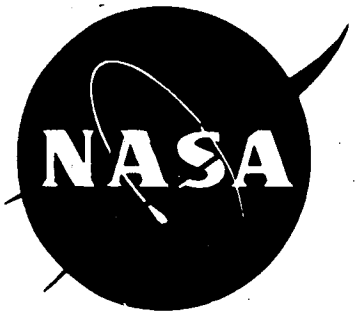
GODDARD SPACE FLIGHT CENTER



COMPUTER PROGRAMMER'S MANUAL

GODDARD SPACE FLIGHT CENTER

COMPUTER PROGRAMMER'S MANUAL

7 September 1965

Prepared for

DATA SYSTEMS DIVISION

under Contract NAS5-9817

by

SYSTEM DEVELOPMENT CORPORATION

FOREWORD

The Programmer's Manual applies to all users of the Data Systems Division electronic data processing equipment of the Goddard Space Flight Center, Greenbelt, Maryland. It is recognized that changes to this manual will be necessary as operational experience is gained and operational computer programs are revised. Recommendations for revision and/or changes are encouraged and will be submitted to the Data Systems Division.

Initial distribution of this manual and change pages will be made in accordance with established distribution lists. Proposed or recommended revisions and/or changes will not be implemented until such changes are published by the Data Systems Division. Changes will be issued as replacement pages.

The technical material contained in the Programmer's Manual was in large part derived from documents authored by IBM and UNIVAC and from the Programming Methods Section and the Advanced Orbit Programming Branch of the Data Systems Division.

CONTENTS

CONTENTS (Cont'd)

CONTENTS (Cont'd)

ILLUSTRATIONS

TABLES

INDICES

(1-ii blank)

CONTENTS

CHAPTER 1 PRELIMINARY INFORMATION

CHAPTER 1

PRELIMINARY INFORMATION

1.1 INTRODUCTION

This manual describes the several different programming support systems employed at the Goddard Space Flight Center, namely, 7094 FORTRAN, MYSTIC, EXEC II PROCESSOR-1107, and AUTOCODER-SPS. It is tailored to the experienced programmer as it presents in detail how to use the various support systems. Each Chapter in this manual is a separate entity, and as such can be removed without disturbing the contents of the other chapters. There is a detailed table of contents for each Chapter, as well as a general table of contents covering the manual as a whole.

1.2 CONTENT

Chapter 2 describes the 7094 FORTRAN operating system and its use. It contains a system description, machine configurations, and system tape maintenance and error reporting procedures. This Chapter includes several illustrations showing job deck composition for a number of typical runs; presents a description of the control cards and their use; and offers means by which to use the system effectively. There is a bibliography which provides the user an access to the list of documents describing the major components of the 7094 operating system. There is a section describing in brief the characterization of the systems contained on the combined IBSYS/FMS master tape employed on the 7094 A, B, C, E, and F computers. A final section describes utility routines that have been prepared for use by Goddard Space Flight Center programmers.

Chapter 3 describes the Computer-independent Abstract Machine-language Encoder and Operating system (CAMEO) and its use. It contains a system description, machine configurations, and system tape maintenance and error procedures. This Chapter includes several illustrations showing computer set-up for a number of typical runs; presents a description of control commands and their use; and offers means by which to use CAMEO effectively. There is a bibliography which provies a list of abstracts of documents on CAMEO. Also included are functional aids and functional subroutines along with the encoders that are embedded in the CAMEO system.

Chapter 4 describes the EXEC II System for the Univac 1107. It touches upon EXEC II 1107-1108 configuration differences; details the 19 control cards that exercise control over the EXEC II System; presents illustrations showing job deck composition for a number of typical runs, as well as describing programming procedures, operating procedures, utility routines; and has a bibliography to the list of documents describing the major components of the 1107 operating system.

Chapter 5 describes the similar symbolic programming systems, the Autocoder and the Symbolic Programming System (SPS) for IBM peripheral equipment. It describes system configuration, machine configuration, system tape maintenance and error reporting, processor-control operations, utility routines, as well as a bibliography with abstracts.

Chapter 7 is to be provided.

Chapter 8 is an addendum. The programmer uses this supplement to insert memorandums, various forms, telephone numbers, and other information to assist him in the performance of his everyday duties.

1.3 MISCELLANEOUS INFORMATION

This section will eventually include the forms pertaining to error reporting indicated in the subsequent Chapters of this manual.

CONTENTS

CHAPTER 2 7094 FORTRAN OPERATING SYSTEM

CONTENTS (Cont'd)

CONTENTS (Cont'd)

CONTENTS (Cont'd)

CONTENTS (Cont'd)

ILLUSTRATIONS

ILLUSTRATIONS (Cont'd)

TABLES

INDEX

CHAPTER 2

7094 FORTRAN OPERATING SYSTEM

2.1 7094 FORTRAN SYSTEM DESCRIPTION

This chapter describes the 7094 FORTRAN Operating System and its use. The 7094 FORTRAN system is available on all Goddard Space Flight Center Data Systems Division large scale computers. An integral part of the IBM 7094 Data Processing System, the 7094 FORTRAN system consists of a comprehensive set of programming aids operating under a master System Monitor as subsystems. The System Monitor consists of 1) The System Supervisor, 2) The System Core-Storage Dump Program, 3) The System Editor, 4) The System Nucleus, and 5) The Input/Output Executor.

The 7094 FORTRAN Operating System is designed to process sequentially a variety of unrelated jobs with little or no operator intervention. With less human participation, jobs are processed more rapidly and there is less likelihood of human error.

2.1.1 SYSTEM CONFIGURATION

The FORTRAN IV compiler and its associated assembly program, IBMAP, are embedded in the IBSYS/IBJOB operating system on the 7094 data processing equipment. This system is completely independent of the FORTRAN II monitor system, FMS. However, the two systems have been combined onto one tape which is now in use on the 7094 A, B, C, E, and F computers.

To make the two systems as compatible as possible, additional control cards are used to enable either system to call in the other. Thus, FORTRAN II and IV jobs may be batched on the same input tape. However, each job must operate wholly within one system or the other. Mixtures

within a job are not permitted. Furthermore, the relocatable decks produced by the two systems are not compatible, and standard subroutine linkages are different. If FORTRAN II jobs are to be converted to FORTRAN IV, the source program should be SIFTed and recompiled. (SIFT is a program to convert FORTRAN II language to FORTRAN IV.)

The flow diagram (Figure 2-1) illustrates the various components on the system tape and the paths of control between them. The notations on each line indicate the control card, statement, conditions, or action which causes that path to be taken. For the sake of clarity, this diagram is simplified and does not show all possible error returns, restart, and recovery procedures, etc. It also omits the inter-component logic of the FORTRAN II Version III, IBSFAP, CT, SORT, 90PAC, and IOCS processors which are also on the tape and are controlled by IBSYS.

Tape assignments within IBSYS, IBJOB, etc., now conform to the FMS assignments. Namely, logical units 1 through 10 correspond to A1 through A10; 11 through 20 to B1 through B10; and 21 through 30 to C1 through C10. As with FMS, logical 2 is the standard input tape and logical 3 is standard print and punch output. Any other tapes used by a FORTRAN IV program may be either in a BCD mode or a binary mode, but must never be in a mixed mode; i.e., BCD binary records on the same tape. Such a tape can be written, but it is difficult to read. Symbolic tape unit designation should also be avoided. Although permitted by the manual, the following construction is undesirable as it is likely to cause failures.

```
KTAPE = 7
WRITE (KTAPE, 100) A, B, C
```

All I/O activity in FORTRAN IV/IBMAP programs must be handled by IOCS. The FORTRAN IV compiler sets this up automatically. However, it is imperative that the IOCS manual, C28-6100-2, be consulted before attempting to perform I/O in a MAP language program.

For programming on FORTRAN IV, consult IBM 7090/7094 Programming Systems: FORTRAN IV Language, Form C28-6274-4.

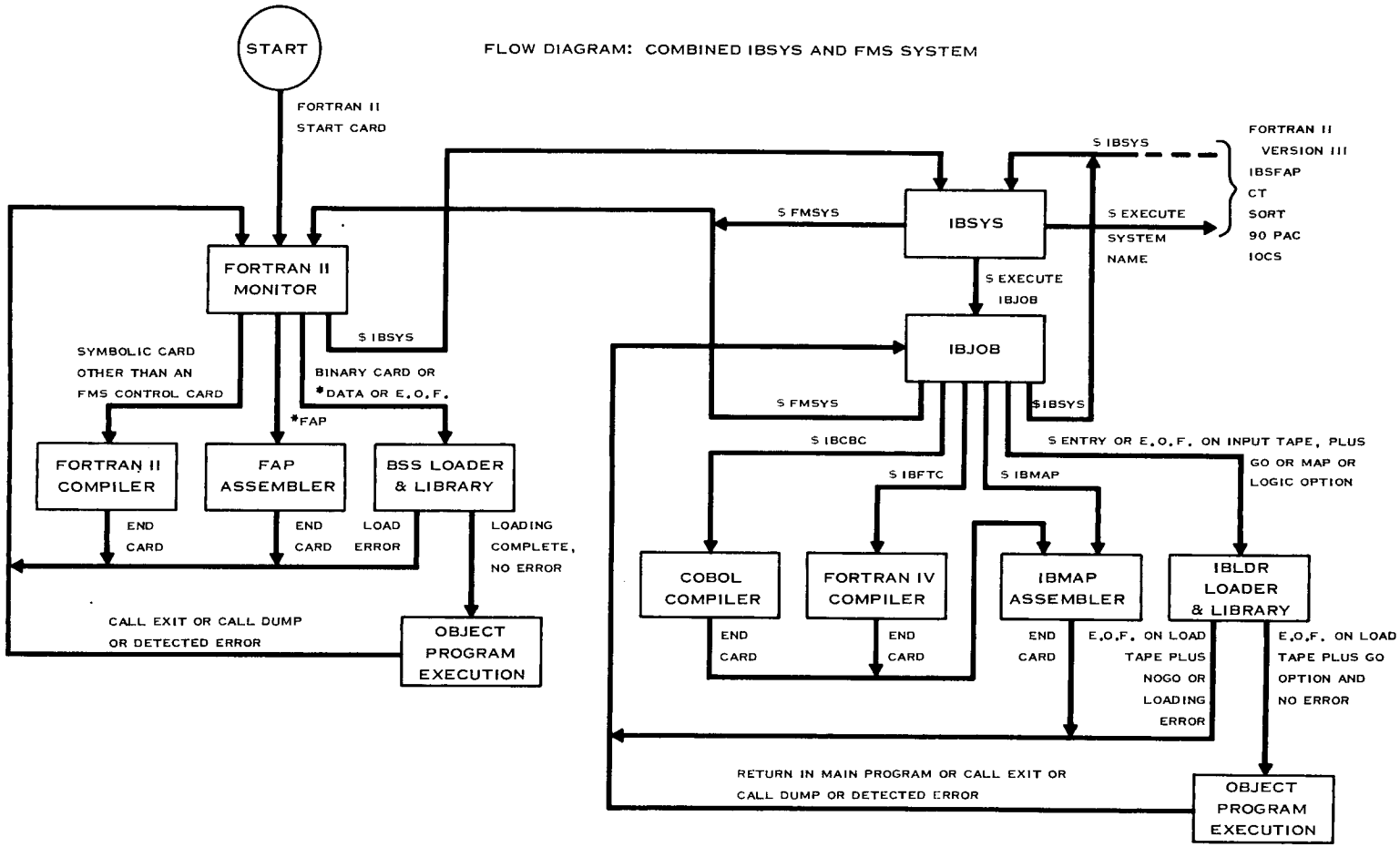


Figure 2-1. Flow Diagram of Combined IBSYS and FMS System Tape

2.1.2. MACHINE CONFIGURATION

The programmer using the Goddard Space Flight Center computer complex has a vast quantity of data processing equipment at his disposal. The IBSYS/FMS combined system operates on several of these large scale computers. A partial list of this equipment is given in Table 2-1. In addition to the major computer configurations, there exists an associated collection of peripheral equipment. A partial list of this equipment is given in Table 2-2.

Table 2-1. Major Computer Equipment

Bldg. Loc.	Computer Facility	Memory Size	Magnetic Tape Units	Line Printer	Card Reader	Card Punch	Disk	Data Channel	DCC
14	A-7094	65K	14-729-IV	716-I	7223-I		1301-II	3	Yes
14	B-7094	65K	14-729-IV	716-I	7223-I		1301-II	3	Yes
3	C-7094-II	65K	14-729-IV	716-I	7223-I	721-I	1301-II	3	Yes
1	E-7094-II	32K	12-729-IV 4-729-VI	716-I	711-I	721-I		2	No
3	F-7094	32K	12-729-IV	716-I	711-I			2	No

Table 2-2. Peripheral Equipment

Bldg. Loc.	Computer Facility	Memory Size	Magnetic Tape Units	Card Read/Punch	Line Printer
14	A-IBM-1401	1.4K	1-7330		1403-II
14	B-IBM-1401	8K	2-7330	1402-I	1403-II
14	C-IBM-1401	8K	3-7330	1402-I	1403-II
14	D-IBM-1401	4K	2-7330	1012-I* 1402-I	1403-II
14	E-IBM-1401	4K	2-729-II	1402-I	1403-II
14	F-IBM-1401	8K	2-729-II	1402-I	1403-II
1	IBM-1460	8K	4-729-VI*** 2-729-IV***	1402-I	1403-III
14	I-IBM-7010**	100K	8-729-IV	1402-I	1403-III
14	IBM-1401	8K	4-729-IV	1402-I	1403-II
14	CDC-3200**	16K	5-607	405 523	501
20	CDC-3200				
3	CDC-160A	4K	2-603	167*** 170***	166-2***
3	CDC-160A	4K	2-603	*** ***	***

*Paper tape reader/punch
**1301 disk
***Switchable units

2.1.3 SYSTEM TAPE MAINTENANCE

The Programming Methods Section (PMS) of the Data Systems Division has primary responsibility for maintaining the combined IBSYS/FMS master tape. Tape revisions or updating occur periodically as a result of one or more of the following conditions: 1) new IBSYS version or significant corrections released by IBM; 2) major changes originating from GSFC programmers; and 3) catastrophic errors requiring the immediate issuance of new tape.

In certain cases, when errors are of a minor consequence or unique to a particular application and immediate release of a new master tape is not warranted, the PMS provides binary decks to circumvent the error condition. With the release of new master tapes, the decks are subsequently discarded by the programmer.

2.1.4 ERROR REPORTING

The PMS has the responsibility of maintaining the combined IBSYS/FMS system. Any questions regarding system utilization and system discrepancies should be directed to PMS personnel. The current method of reporting system discrepancies verbally is expeditious. However, it is recommended that the Systems Discrepancy Report (see Form 1-1, Chapter 1) be used for submittal to the PMS coordinator. In this way, a current file of all discrepancies will be maintained along with the corrective actions taken. A copy of the Discrepancy Form will be available in the PMS coordinating office (Room 127, Bldg. 3) and in the dispatcher's office (Bldg. 1 and Bldg. 3). Programmers are requested to periodically check the Systems Status Report (see Form 1-2, Chapter 1) to insure satisfactory operational performance from the system used.

2.2 DETAILED PROCEDURES

This section includes several illustrations showing job deck composition for a number of typical runs; presents a description of the control cards and their use; and offers means by which to use the system effectively.

2.2.1 CONTROL CARDS FORMAT AND USAGE

This paragraph presents in detail a description of the control cards that IBSYS, IBJOB, IBLDR, and FMS recognize. The user controls and directs the processing of his job by inserting the proper control cards in the job deck, thereby directing the Operating System to perform any one of several operations. An illustration is provided depicting the placement and categorizing of the importance of the control cards.

2.2.1.1 IBSYS Control Card Description

A dollar sign ($) in column 1 accompanies all IBSYS control cards. Except for the comment card which has an asterisk, card function is punched beginning in column 2, with no embedded blanks. All systems on the tape recognize the $IBSYS card. This ensures control by IBSYS of its control cards. These control cards are placed in any order, but must follow an $IBSYS card. An $ID card is mandatory if the job is to perform any meaningful processing. Figure 2-2 shows the IBSYS control cards format.

SEE PAR. 2.2.1.1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(1)	$	I	B	S	Y	S															
(2)	$	I	D		L	D	D	L	D	D	D	D	L	L	L						
(3)	$	J	O	B			L	D	D	L		D	D	D	D	OPTIONS					
(4)	$	E	X	E	C	U	T	E								SYSTEM NAME					
(5)	$	D	A	T	E											M	M	D	D	Y	Y
(6)	$	*														COMMENT					
(7)	$	P	A	U	S	E										COMMENT					

NOTE: CONTROL CARDS (1), (2), AND (4) ARE ESSENTIAL, (3) MAY BE USED INTERCHANGEABLY WITH $ID CARD, (5) THROUGH (7) ARE USED FREQUENTLY.

Figure 2-2. IBSYS Control Cards Format

An individual description of IBSYS control cards is as follows:

(1) $IBSYS:

1	2	3	4	5	6	7 ... 72
$	I	B	S	Y	S	

This card reloads IBSYS and transfers control to its beginning. When IBSYS recognizes this card, it initializes the I/O configuration, and destroys the effect of any previous $ID or $DATE card.

(2) $ID Job Number Time Initials:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 ... 72
$	I	D		L	D	D	L	D	D	D	D	L	L	L	
				JOB NUMBER				RUN TIME MINUTES				INITIALS			

A job number, consisting of a letter, followed by two digits, and T (testing), P (production), or R (rerun) is a requirement. The other information is optional. Time is the estimated running time of the job in minutes and Initials are the initials of the person submitting the job, as recognized by the Dispatcher.

(3) $JOB Job Number Time Options:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 ... 72
$	J	O	B			L	D	D	L		D	D	D	D	OPTIONS

This card reloads IBSYS from the system tape and insures complete initialization of the system at the expense of some additional execution time. Thus, $JOB must precede any $DATE, $EXECUTE, or other control cards which have an initializing effect on the system. A job number, consisting of a letter, followed by two digits, and T, P, or R, is a requirement. Time is the estimated running time in minutes.

(4) $EXECUTE System Name:

1	2	3	4	5	6	7	8	9	10 11	12	13	14	15	16 72
$	E	X	E	C	U	T	E							SYSTEM NAME

This card causes the controlling element of the designated system to be loaded, and control is transferred to it. The IBJOB monitor controls FORTRAN IV, IBMAP, COBOL, and their loader, so that a FORTRAN IV run, for example, must contain an $EXECUTE IBJOB card. The permissible system names are:

SYSTEM NAME	SYSTEM DESIGNATION
IBJOB	for FORTRAN IV, IBMAP, COBOL
FORTRAN	for FORTRAN II version III
IBSFAP	for FAP under IBSYS using IOCS
CT	for Commercial Translator
SORT	for 7090 Sort/Merge Processor
90PAC	a commercial data processing language processor
IOCS	independent IOCS

NOTE: FORTRAN II version III is not the customary FORTRAN II in use at Goddard Space Flight Center. It is FORTRAN II operating under IBSYS, which imposes storage restrictions on the object code, and contains no local modifications. The usual FORTRAN II system is called by the $FMSYS control card.

(5) $DATE:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22 72
$	D	A	T	E											M	M	D	D	Y	Y	

Columns 16-21 contain six digits representing the date. This date is part of the output of the IBJOB system. The $DATE card is optional, but once encountered, it remains effective until replaced by an $IBSYS, $FMSYS or another $DATE card. If there is no $DATE card, the date of some preceding job (if IBSYS has not been loaded in the meantime) or the date of creation of the system tape is part of the output of the IBJOB system.

(6) $*:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 72
$	*														COMMENTS

The contents of this card are printed on-line for communication with the operator.

(7) $PAUSE:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 72
$	P	A	U	S	E										COMMENTS

This card causes the machine to halt. The operator presses the START button to continue the job. Since all control cards print on-line, columns 16-72 may contain a message to the operator.

2.2.1.2 IBJOB Control Card Description

The $IBJOB control card is the first card in an IBJOB deck (i.e., following an $EXECUTE IBJOB card). However, the $ID, $JOB, and/or $DATE card can precede the $IBJOB card if IBSYS is unable to recognize these cards. For example, several jobs can be run in succession within the IBJOB framework, without going back to IBSYS between jobs.

An $IBFTC card precedes each FORTRAN IV source deck before compilation. An $IBMAP card precedes each IBMAP symbolic or PREST deck to be assembled. An $IBLDR card precedes each relocatable binary deck (that is, for each individual subprogram).

FORTRAN IV compilations, IBMAP assemblies, and binary decks to be loaded may be arranged in any order within a given job, provided each subprogram is prefaced with an appropriate control card. The order of deck placement determines the order of their storage in memory, except as modified by $ORIGIN cards. The complete object program deck is placed behind the $IBJOB card.

If the $ENTRY card is used, it must be placed behind the object program deck. $DATA card must be placed behind the $ENTRY card, if present, or behind the object program deck if there is no $ENTRY card. Any data cards which the object program expects to find on logical 2, the standard input unit, are placed behind this $DATA card. If data cards are present, an end-of-file card must follow them.

$* comment and $PAUSE cards may be placed anywhere in the deck that IBJOB (or IBSYS) expects to find control cards. $IEDIT or $OEDIT cards may be placed anywhere in the deck that IBJOB expects to find control cards, provided they follow the $IBJOB card.

One or more *ALTER cards must follow any $IBMAP card which follows an $IEDIT card containing the ALTER option. An *ENDAL card must terminate the alter deck. The IBMAP symbolic or PREST deck must follow the *ENDAL card unless an alternate input unit was also specified on the $IEDIT card. In this case, a matching $IBMAP card and the symbolic or PREST deck must appear on the specified unit. Once again, the ordering of the cards for altering a subprogram is: $IEDIT with ALTER option, $IBMAP, a succession of *ALTER card + symbolic insertions, *ENDAL, symbolic or PREST deck.

Following an $IEDIT card specifying an alternate input unit, all FORTRAN source decks, IBMAP symbolic or PREST decks and relocatable binary decks for which there are $IBMAP, $IBFTC, or $IBLDR cards in the main input will be taken from specified alternate unit until another $IEDIT card, an $ENTRY card, $DATA, or an end-of-file is encountered on the normal input unit. Alter decks are always taken from the normal input unit. Each deck on the alternate unit must be

prefaced with an $IBFTC, $IBMAP, or $IBLDR card, as appropriate, on which the deck name matches that of the corresponding control card in the main input. They need not contain matching options. The options specified on the control cards on the normal input unit will be exercised; those on the alternate will be ignored. Figure 2-3 shows IBJOB control cards format.

> NOTE: All relocatable binary decks resulting from a compilation or assembly under IBJOB will contain at least two, and at most five, automatically produced control cards. These cards are considered part of the deck and should never be removed, nor should the deck be rearranged in any way. The first card of the deck should be either $TEXT, $DDICT, or $FDICT (in that order of decreasing likelihood), and the last card must always be a $DKEND. The system will also punch out an $IBJOB card for each job, and an $IBLDR card ahead of each relocatable binary deck. These may be used on subsequent runs or discarded according to the user's needs.

SEE PAR. 2.2.1.2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(1)	$	I	B	S	Y	S															
(2)	$	I	D		L	D	D	L	D	D	D	D	L	L	L	COMMENTS					
(3)	$	J	O	B			L	D	D	L		D	D	D	D	OPTIONS					
(4)	$	I	B	J	O	B										OPTIONS					
(5)	$	I	B	F	T	C		DECKNAME								OPTIONS					
(6)	$	I	B	M	A	P		DECKNAME								COUNT, OPTIONS					
(7)	$	I	B	L	D	R		DECKNAME								OPTIONS					
(8)	$	D	A	T	A																
(9)	$	N	A	M	E											OPTIONS					
(10)	$	E	N	T	R	Y										OPTIONS					
(11)	$	D	A	T	E											M	M	D	D	Y	Y
(12)	$	*														COMMENTS					
(13)	$	P	A	U	S	E										COMMENTS					
(14)	$	I	B	R	E	L															
(15)	$	I	E	D	I	T															
(16)	$	0	E	D	I	T															
(17)	M							*	A	L	T	E	R			N					
(18)	M							*	A	L	T	E	R			N	,	K			
(19)								*	E	N	D	A	L								

NOTE: CONTROL CARDS (1) THROUGH (7) ARE ESSENTIAL, (CARDS (2) AND (3) MAY BE USED INTERCHANGEABLY; HOWEVER (2) IS PREFERRED); (8) AND (10) THROUGH (13) ARE USED FREQUENTLY; (9) AND (14) THROUGH (19) ARE USED OCCASIONALLY.

Figure 2-3. IBJOB Control Card Format

An individual description of IBJOB control cards is as follows:

(1) $IBSYS:

1	2	3	4	5	6	7 ... 72
$	I	B	S	Y	S	

IBSYS is called in, replacing IBJOB, and control is transferred to it. The System Supervisor is called in.

(2) $ID Job Number Time Initials Etc.:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 ... 72
$	I	D		L	D	D	L	D	D	D	D	L	L	L	COMMENTS
				JOB				TIME (MINUTES)				INITIALS			

This card is described under IBSYS control cards. It is also recognized by IBJOB to obviate the necessity of returning to IBSYS after each job.

(3) $JOB Job Number Time Options:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 ... 72
$	J	O	B			L	D	D	L		D	D	D	D	OPTIONS

This card reloads IBSYS from the system tape and insures complete initialization of the system at the expense of some additional execution time. Thus, $JOB must precede any $DATE, $EXECUTE, or other control cards which have an initializing effect on the system. A job number, consisting of a letter, followed by two digits, and C, T, P, or R, is a requirement. Time is the estimated running time in minutes.

(4) $IBJOB:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 ... 72
$	I	B	J	O	B										OPTIONS

This card provides IBJOB with control information which governs the processing of object decks in this job. There are seven options: (The standard options are underlined.)

1. GO or NOGO--GO instructs the system to load and execute this job when an $ENTRY card or an end-of-file is encountered on the input unit. NOGO suppresses execution of the object code.

2. LOGIC or NOLOGIC--LOGIC causes the off-line printout of the origin and extent of each subprogram in the job (including library routines), all symbolic references between various routines and I/O buffer assignments. NOLOGIC suppresses this printout. The use of the LOGIC option does not imply that the GO option must also be used.

3. MAP or NOMAP--MAP provides a complete storage map of the object program, to be printed off-line. NOMAP suppresses this printout. The use of the MAP option does not imply that the GO option must also be used.

4. FILES or NOFILES--FILES causes the off-line printing of a file list showing all I/O unit assignments and tape mounting instructions applicable to this job. NOFILES suppresses this printout.

5. FIOCS, LABELS, BASIC, MINIMUM or IOEX--This option governs what portion of the IOCS package is to be made available to the object program at execution time. LABELS is the full IOCS package with labeling features. BASIC is the full IOCS package but without labeling features. MINIMUM is basic IOCS minus the routines for handling internal files; for stringing buffer pools together; for transferring input directly to output; for checkpoints and for the hardware functions of write end-of-file, backspace file, backspace record and rewind. IOEX is a trap supervisor. It does not handle the initiation of I/O operations. FORTRAN IV programs require minimum IOCS. The use of ENDFILE, BACKSPACE, or REWIND statements in a FORTRAN routine does not necessitate calling in basic IOCS.

6. SOURCE or NOSOURCE--SOURCE informs the system that this job contains at least one compilation or assembly. NOSOURCE tells the system that this job contains only binary decks to be loaded. This circumvents the system's need to transcribe binary decks onto a scratch tape to hold them while compilations or assemblies are in progress. NOSOURCE running under the SOURCE option will be processed correctly, but at the expense of some additional tape handling. NOSOURCE may not be used when the binary decks are obtained from an alternate unit by means of an $IEDIT card.

7. FLOW or NOFLOW--This option applies to overlay jobs. The loader detects inadmissable overlay structures, such as the existence of

CALL statements which cause themselves to be overlaid. The FLOW option causes error messages to be printed (off-line) and the execution to be deleted if such errors occur. NOFLOW permits execution to begin in spite of such errors, and suppresses the printing of error messages unless the LOGIC option is also used.

NOTE: GO, MAP, and LOGIC are independent options. Any one or any combination of these options will cause the object program to be loaded as if for execution. However, execution will take place only if the GO option is exercised. If NOGO, NOMAP, and NOLOGIC are all in effect, loading is suppressed.

(5) $IBFTC Deck Name Options:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	72
$	I	B	F	T	C				DECKNAME						OPTIONS	

Normally this card informs the system that the following deck is a FORTRAN IV source program to be compiled. The $IEDIT card may be used to inform the system that a FORTRAN IV source deck ready for compilation is located on a specified alternate input unit. There are six options: (The standard options are underlined.)

1. LIST, NOLIST, or FULIST--LIST indicates that an off-line listing of the generated IBMAP coding is desired. NOLIST suppresses this listing as well as assembly error message printout. Compiler errors are always printed. The LIST option produces a listing of generated IBMAP coding without octal equivalents, and such that the coding is read across the page, like a dump.

2. REF or NOREF--REF indicates that a symbol cross-reference table is wanted with the generated IBMAP listing. NOREF suppresses this print. REF is ineffective if the NOLIST option is used.

3. DECK or NODECK--DECK indicates that a relocatable binary object deck for this subprogram is desired. It will be punched off-line. NODECK suppresses this punch output.

4. M90, M94, or M94/2--M90 instructs the compiler not to generate any 7094 machine instructions in its compiled coding. M94 indicates that the compiler may generate 7094 coding. M94/2 generates 7094 coding and EVEN pseudo-operations are treated as commentary.

5. XR3, XR4, XR5, XR6, or XR7--This option dictates the number of index registers for which the compiler is allowed to generate coding in the compiled program. A minimum of three index registers must be allowed. Index registers will be selected in the order: 1, 2, 3, 4, 5, 6, 7.

6. NODD, DD, SDD--These options refer to the debugging dictionary. The standard option, NODD specifies no debugging dictionary is desired. For option DD, a full debugging dictionary is output. The contents of the full dictionary are all symbols used in an assembled program or for a FORTRAN IV program, all statement numbers, all programmer-specified symbols, and all symbols generated by IBFTC. The option, SDD, provides a short debugging dictionary. Only these symbols specified by the MAP pseudo-operation KEEP are output for assembled programs. Statement numbers and programmer-specified symbols are output for FORTRAN IV programs.

(6) $IBMAP Deck Name Count Options:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	72
$	I	B	M	A	P		DECKNAME								COUNT OPTIONS	

Normally this card informs the system that the deck to follow is an IBMAP symbolic or PREST deck to be assembled. The $IEDIT card may be used to inform the system that an IBMAP symbolic or PREST deck ready for compilation is located on a specified alternate input unit. A count of the approximate number of symbolic cards may be given, starting in column 16. This helps to speed the assembly. It may not exceed five digits. If a count is not given, a value of 2000 is assumed. In this case, the options (if any) must be punched beginning in column 16. There are eight options: (The standard options are underlined.)

1. LIST or NOLIST--LIST indicates that an off-line assembly listing is desired. NOLIST suppresses this listing as well as error message print. (The LIST/NOLIST option also appears on the $IBFTC card, but note that the standard option is reversed.)

2. REF or NOREF--REF indicates that a symbol cross-reference table is wanted with the assembly listing. NOREF suppresses this printout. REF is ineffective if the NOLIST option is used. (The REF/NOREF option also appears on the $IBFTC card, but note that the standard option is reversed.)

3. DECK or NODECK--DECK indicates that a relocatable binary object deck for this subprogram is desired. It is punched off-line. NODECK suppresses this punch output.

4. M90, M94, or M94/2--M90 instructs the assembler to treat 7094 instruction mnemonics as macros, the expansion of which are defined within the system. M94 indicates that 7094 op-codes are permissible. The M94/2 option must not be used, nor may the resulting binary decks be used on 7094 A or B machines. These machines are Model I's.

5. RELMOD, ABSMOD or SYSMOD--RELMOD indicates that a relocatable assembly is desired. ABSMOD indicates that this is to be an absolute assembly. SYSMOD indicates that this is a special relocatable assembly having absolute origins.

6. NO() or ()OK--NO() indicates that parentheses are to be treated as illegal characters in an IBMAP symbol. However, only a warning is produced, and the assembly is not aborted. ()OK indicates that parentheses are permissible characters in an IBMAP symbol.

7. MFTC or NOMFTC--MFTC indicates that the macro definitions corresponding to the FORTRAN built-in functions are to be supplied for this assembly. NOMFTC indicates that these macro definitions are not wanted.

8. NODD, DD or SDD--These options refer to the debugging dictionary. The standard option NODD specifies no debugging dictionary is desired. For option DD, a full dictionary is output containing all symbols used in the assembled program. Option SDD indicates only a short debugging dictionary is desired. It contains only those symbols specified by the MAP pseudo-operation KEEP.

(7) $IBLDR Deck Name Options:

1	2	3	4	5	6	7	8 9 10 11 12	13	14	15	16 72
$	I	B	L	D	R		DECKNAME				OPTIONS

Normally this card informs the system that a relocatable binary deck follows or is to be obtained from the Library. The $IEDIT card may be used to inform the system that a relocatable binary deck is located on a specified alternate input unit. There are two options: (The standard options are underlined.)

1. LIBE or NOLIBE--LIBE indicates that the deck named is obtainable from the system library. In this case, the entire object program must consist of routines supplied via $IBLDR cards with the LIBE option. A mixture of LIBE and NOLIBE within a job is not permitted. NOLIBE indicates that the deck is scheduled next on the standard input unit, or is obtainable from an alternate input unit, if preceded by an $IEDIT card.

2. TEXT or NOTEXT--TEXT indicates that the text section (i.e., the actual instructions) of this deck is to be loaded. NOTEXT indicates that the text section of this deck is to be ignored. This permits the system to use dictionary sections of the deck without consuming storage with its instructions.

(8) $DATA:

1	2	3	4	5	6 ... 72
$	D	A	T	A	

This control card indicates the beginning of a data file on the input unit.

(9) $NAME:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17 ... 72
$	N	A	M	E												OPTIONS

This control card may be used to change the name of a file or control section. Name changes may be required when: 1) the same name has been used in different decks for two or more distinct file or control sections, in which case one of them must be renamed with a distinct name, and 2) two different names are used to refer to the same file or control section, in which case one name is replaced by the other.

The variable option field consists of two alphameric names separated by an equal sign, i.e., ABC = XYZ. The name on the left may be a qualified external name which is to be replaced by the name on the right. Files to be renamed must have the name and qualifier, if specified, enclosed by quotation marks. A qualifier is defined as a deck name, i.e., DK1 (ABC) = XYZ. If not qualified, external names on the left are replaced by the name on the right whenever encountered. If a name is qualified by a deck name, it is replaced by the name on the right only in the deck named.

(10) $ENTRY:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 ... 72
$	E	N	T	R	Y										NAME

This card, which is optional, governs the point in the object code at which execution begins. If an $ENTRY card is not used, or if an $ENTRY card with a blank name field is supplied, the loader will seek the control section name ..... (six periods) and transfer to it. (NOTE: A FORTRAN IV main routine will have this name as its entry point.) If the name ..... cannot be found, the system will begin executing the object program at the first entry point of the first subprogram in the input deck. If execution is to begin at some other point, an $ENTRY card containing a name punched beginning in column 16 must be supplied. This name may be an external name (i.e., it

appears in some control section--an entry point) to which the initial transfer is made, or it may be a deck name, in which case, the initial transfer will be made to the standard entry point of that deck.

(11) $DATE: As described in Paragraph 2.2.1.1 (1).

(12) $*: As described in Paragraph 2.2.1.1 (6).

(13) $PAUSE: As described in Paragraph 2.2.1.1 (7).

(14) $IBREL:

1	2	3	4	5	6	7 ... 72
$	I	B	R	E	L	

This control card indicates that no more compilations or assemblies follow on the System Input Unit. The IBJOB Processor Monitor then reads in and transfers control to the Loader. The $IBREL card has the effect of negating the SOURCE option on the $IBJOB card because no further compilation or assemblies are performed after it is encountered.

(15) $IEDIT:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 ... 72
$	I	E	D	I	T										OPTIONS

This card instructs the system that a deviation from normal input procedures is forthcoming. With it, an alternate input tape may be brought into action and/or the presence of alter cards may be signalled. The effect of an $IEDIT card continues until another $IEDIT card with different options is encountered.

There are three options: (The standard options are underlined.)

1. SYSIN1, SYSCK1, SYSCK2, SYSLB2, SYSLB3, SYSLB4--This option designates the I/O unit on which the source, symbolic, PREST, or relocatable decks may be found for each subsequent $IBFTC, $IBMAP, or $IBLDR card encountered on SYSIN1. Unless SYSIN1 is the designated unit, all decks on the unit designated must be prefaced with an $IBFTC, $IBMAP, or $IBLDR card, as appropriate, on which the deck name matches that appearing on the $IBFTC, $IBMAP, or $IBLDR card on SYSIN1. (SYSIN1 is the standard input unit.)

2. SRCH or NOSRCH--SRCH indicates that the system must search the alternate input unit to find an $IBFTC, $IBMAP, or $IBLDR card with a matching deck name. (SYSIN1 must not be searched.) NOSRCH indicates that the designated alternate unit is correctly positioned to read the next deck.

3. ALTER or NOALTER: ALTER indicates that an alter deck will be found on SYSIN1 immediately following all subsequent $IBMAP cards until another $IEDIT card is encountered, or until the end of this job. NOALTER indicates that there are no alter cards.

(16) $OEDIT:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 72
$	0	E	D	I	T										OPTIONS

This card signals an impending change in the normal output operation. With it, an alternate unit may be selected for off-line print, and/or PREST decks may be obtained. There are three options: (The standard options are underlined.)

1. SYSOU1, SYSCK1, SYSCK2, SYSLB2, SYSLB3, SYSLB4--This option designates the I/O unit to be used for off-line print output for the remainder of the job, or until another $OEDIT card is encountered. SYSOU1 is the normal output unit.

2. PREST or NOPREST--PREST indicates that a PREST deck for all following symbolic input decks to the assembler is to be punched off-line. Relocatable binary decks may also be obtained or deleted in the usual manner. NOPREST indicates that a PREST deck is not wanted.

3. CPREST, NOCPR--CPREST indicates that a PREST deck for all following source input decks to the compiler is to be punched off-line. NOCPR indicates that a PREST deck is not wanted. If both PREST and CPREST are specified in the $OEDIT card that precedes a source deck, both compiler input and output are written off-line in PREST form.

NOTE: PREST is a BCD representative of a symbolic program as supplied or as compiled by FORTRAN. Insignificant blanks are deleted. PREST decks are punched out in column binary card format, but the information contained in each binary data word is to be interpreted as BCD. PREST decks are identified by 2, 3, 4, 5, 7, and 9 punches in column 1. Relocatable decks (IBJOB format--not FORTRAN II) will not have a 2 punch in column 1, and there will be various combinations of 3, 4, and 5 punches to identify various sections of the deck. If column 1 contains a 1 punch, it is not an IBJOB deck.

(17) m *ALTER n:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	72
M							*	A	L	T	E	R			N		

This is one form of alter cards that defines the exact point for making modifications in a symbolic program. (This card is not strictly a control card.) The alter point is found by combining m and n to form a string of up to eight characters. This is matched against the label punched into columns 73-80 of the symbolic deck being altered (ignoring any leading blanks) or if a PREST deck is being altered, against the instruction line label printed on the original listing. Any symbolic cards following the *ALTER card, up to but not including the next *ALTER or *ENDAL card, are inserted into the deck just preceding the machine word corresponding to the matched label.

(18) m *ALTER n,k:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	72
M							*	A	L	T	E	R			N	,	K		

This form of the *ALTER card matches labels mn and mk, and the original coding from the line labeled mn to the one labeled mk, inclusive, is deleted. If symbolic cards follow the *ALTER card (preceding the *ENDAL card), they are inserted into this hole.

(19) *ENDAL:

8	9	10	11	12	13
*	E	N	D	A	L

The use of this trailer card is required to terminate an alter deck.

2.2.1.3 IBLDR Control Card Description

All IBLDR control cards have a dollar sign ($) in column 1. Control card function is punched in column 2 with no embedded blanks. Deck name, if required, is punched beginning in column 8. Any additional information required is punched beginning in column 16, with no embedded blanks.

See Paragraph 2.2.1.2 for a discussion of $IBLDR and $ENTRY cards. Although recognized by IBJOB, these cards are also passed along to IBLDR for further processing. $LABEL cards may be placed anywhere in the deck that IBJOB expects to find control cards, providing they occur after the $IBJOB card. IBJOB recognizes $LABEL only to the extent of making it available to the loader. $ORIGIN and $INCLUDE cards may be placed immediately before any $IBFTC, $IBMAP, or $IBLDR cards in the deck. Figure 2-4 shows the IBLDR control cards format.

SEE PAR. 2.2.1.3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(1)	$	I	B	L	D	R		DECKNAME								OPTIONS					
(2)	$	E	N	T	R	Y										OPTIONS					
(3)	$	O	R	I	G	I	N									SYMBOLS			OPTIONS		
(4)	$	I	N	C	L	U	D	E								NAMES					
(5)	$	L	A	B	E	L										LABELLING					

NOTE: CONTROL CARD (1) IS ESSENTIAL, (2) IS USED FREQUENTLY; (3) THROUGH (5) ARE USED OCCASIONALLY.

Figure 2-4. IBLDR Control Cards Format

An individual description of the IBLDR control cards is as follows:

(1) $IBLDR Deck Name Options: See Paragraph 2.2.1.2(7).

(2) $ENTRY Name: See Paragraph 2.2.1.2(10).

(3) $ORIGIN Symbol Options:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 72
$	O	R	I	G	I	N									SYMBOLS, OPTIONS

This card is used to define the beginning of each link of an overlay job. Overlay is a feature which permits the execution of jobs which exceed memory capacity. Several subprograms or sets of subprograms may be relocated by the loader so as to occupy the same portion of core memory. Each such segment of the program (i.e., each link) is recorded on tape, and during execution various links are automatically called into memory as needed.

The symbol punched beginning in column 16 is a string of from one to six alphanumeric characters. The six special characters ()=,/. may not be used. A symbol is mandatory on every $ORIGIN card. In addition to this logical origin symbol, there are three options: (The standard options are underlined.)

1. Absolute Origin, Octal Location--To specify an absolute origin, punch from one to five numeric characters indicating the decimal memory location at which the following routine is to be originated. To specify octal location, punch an alphabetic O, followed by one to five digits representing the octal value.

2. SYSUT2, SYSUT3, SYSLB2, SYSLB3, SYSLB4, SYSCK1, SYSCK2, or UT2, UT3, LB2, LB3, LB4, CK1, CK2--This option specifies the I/O unit on which the following link is to be written. It is assumed that the unit is in ready status and that it is used only to contain overlay links during job execution.

3. REW or NOREW--REW indicates that the I/O unit containing this link is to be rewound after loading each time this link is called for during execution. NOREW suppresses these rewinds.

The operation of overlay is best illustrated by an example. At least one subprogram is required in the job which remains in memory at all times (it must not be overlaid). This subprogram is link 0 or the main link. The decks comprising link 0 are placed physically in front of all other decks in the job. Following link 0, an $ORIGIN card appears; and it contains a symbol which the loader associates with

the next available memory location (or with the absolute origin, if specified). In this situation, it is an $ORIGIN ALPHA card as shown in Figure 2-5. Following this $ORIGIN card, the decks comprising any one of the links issuing from this origin must appear. If two or more links issue from the link just described, an $ORIGIN card with a new origin symbol must appear next, and it must be followed by all the decks for any one of the links originating at that point. Thus, the correct positioning of $ORIGIN cards is important only for the first occurrence of each origin symbol. Parallel links headed by $ORIGIN cards containing symbols already defined may be placed anywhere in the deck so long as each complete link is kept together and the definition of new origin symbols is not disturbed. However, the ordering of decks will affect the order in which the links are written onto tape. The various links may all be written on the same tape, or they may be distributed among several tapes.

In Figure 2-5, link 0 consists of two decks not preceded by an $ORIGIN card. Links 1, 4, 5, and 6 must each be preceded by an $ORIGIN ALPHA card. Links 2 and 3 must each be preceded by an $ORIGIN BETA card. Links 7 and 8 must each be preceded by an $ORIGIN GAMMA card. Links 9 and 10 must each be preceded by an $ORIGIN DELTA card. Link 0 must be followed by either link 1, 4, 5, or 6. Link 1, whenever it occurs in the input deck, must be followed by either link 2 or link 3. Link 6 must be followed by link 7 or 8. Link 8 must be followed by link 9 or 10.

The subdivision of the object program into overlay links does not require any special coding since the system automatically handles the loading of links as they are needed. However, there are certain obvious situations which cannot be handled and must be avoided by the user. For example, a routine in one link may not CALL a routine in another link since this might induce an overlay which will wipe out the CALL. Also, overlay is initiated only by CALL statements and function usage (or the FORTRAN function subprogram type). A data reference to a symbol contained in the control section of a routine within a dependent link is permissible only if a preceding CALL or function reference has guaranteed the loading of that link.

(4) $INCLUDE Name 1 Name 2 Name 3 ...:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 72
$	I	N	C	L	U	D	E								NAME1,NAME2, ETC.

This card specifies that the decks and/or the control sections named starting in column 16 are to be included in the link in which this $INCLUDE card appears, rather than in the link to which they would normally be assigned. The names specified may be either deck names

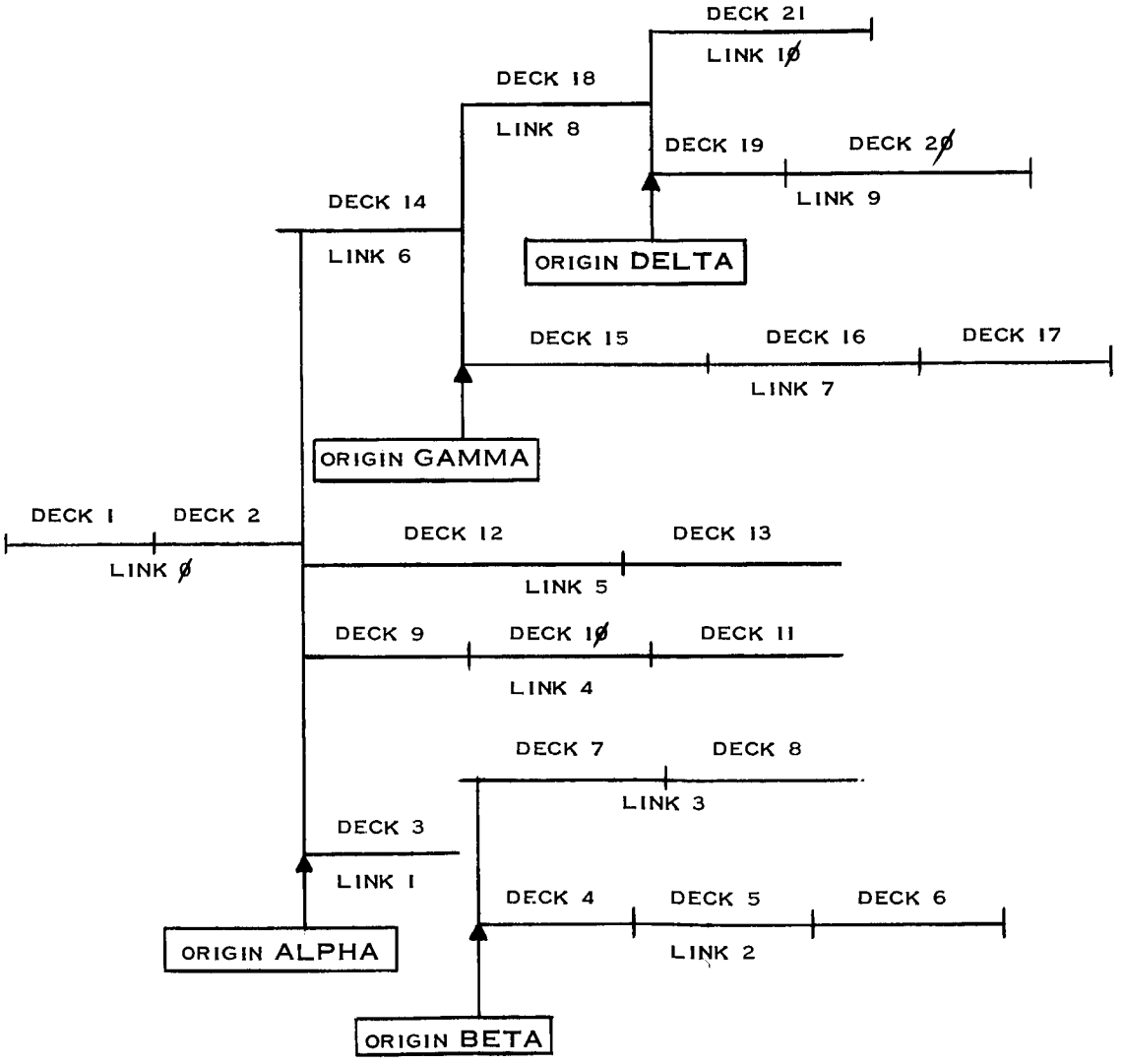


Figure 2-5. Operation of Overlay

(usually library subroutines) or real control sections names of non-zero length (usually a block of data or coding).

If a library subroutine is specified, its deck name, not one of its entry points, must be given. Normally, all library subroutines used by the job will automatically be loaded with the main link so that they are available to all subsequent links. A library subroutine may, however, be assigned to a dependent link by means of an $INCLUDE control card. A subroutine or control section may not be loaded in more than one link. If it is called from more than one link, it must be loaded in a higher level link that is available to all calling links. The library routines .FPTRP, .LXCON, and .LOVRY must be allowed to load with the main link.

(5) LABEL 'Filename' Serial Reel When Name:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 72
$	L	A	B	E	L										LABELLING

This card is used to provide labeling information to IOCS when the LABELS option of the $IBJOB card is exercised (see Paragraph 2.2.1.2 (5)5.). It is an exception to the usual control card format in that all information must be supplied in the exact order indicated. The three interior items--serial, reel, and when--are optional and may be deleted. However, the punctuation is essential. There must always be four commas, in addition to any that may appear in 'filename' or name. No blanks are permitted from column 16 through the fourth comma.

1. 'Filename'--This is the symbolic name of the file to which this information applies, enclosed between apostrophes. The first apostrophe must be punched in column 16. Maximum length of this item is 18 characters.

2. Serial--This is an alphanumeric field of five or less characters. Input labels are checked against this serial value, if present. Output labels will contain this serial value, provided the reel option is greater than one. Output serials, in the absence of this option, are normally taken from the label already present on the first output reel.

3. Reel--This is a numeric field of four or less characters. It indicates the reel sequence number of the first reel of a file. If omitted, the sequence number is assumed to be zero for input files, or one for output files. The actual reel sequence number is adjusted at object time to reflect reel switching, and it is checked in standard input labels.

4. When--This field is used for checking a standard input label. If it is omitted, the date is not checked. If it is present, either of two forms may be used. In the first form, the date is represented by y/d, where y is two digits representing the year, and d is the day of the year. Thus, December 6, 1963, would appear as 63/340.

In the second form, a numeric field of four or less digits represents the number of days a tape is to be retained from the date on which it was written. An attempt to write a labeled file on this tape before the end of the retention period will result in an on-line error message.

5. Name--This is a field consisting of 18 alphanumeric characters among which blanks are permitted. Input labels are checked against this name, and output files are labeled with this name. If completely blank, the label is not checked for input files, or the existing label is retained for output files.

2.2.1.4 FMS Control Card Description

The $IBSYS card has $IBSYS punched in columns 1 to 6. All other FMS control cards have an asterisk (*) in column 1. On the FMS identification cards, columns 2-72 are available for punching optional information. On all other FMS control cards, card function is punched beginning in column 7. Blanks are permitted. On the * DATE card, 2 slashes and 2 digits for the year are required. The month and day may each be represented by zero, one or two digits.

If an $IBSYS card appears in the deck, no other FMS control card may appear between it and a $FMSYS card. In an FMS job, * DATE, if used, must be the first card in the deck with an * identification card immediately following it. If the * DATE is not used, the * identification card must be first. An * identification card is mandatory for FMS runs. * XEQ and/or * FORMAP, if used, in either order, must immediately follow the * identification card.

The control cards * LIST, * LIST8, * CARDS COLUMN, * CARDS ROW, * ROW, * LABEL, * LIBE, * SYMBOL TABLE and * FAP apply to individual subprogram compilations or assemblies. The source or symbolic deck for each subprogram to be compiled or assembled must be preceded by any control cards of this group which the user may want to include. They may be in any order except the * FAP card which must immediately precede the symbolic deck to which it applies.

* LIST, * LIST8, * LIBE, and * SYMBOL TABLE are ineffective for FAP assemblies. Assembly listings are produced automatically without a control card. The debug facilities are inoperative with FAP routines. Source and symbolic decks with their control cards are placed in any sequence behind the * identification, * XEQ and/or * FORMAP cards, but before any debug, binary, or data cards which may appear in the job.

The * DEBUG card, if used, must be placed immediately behind the last source or symbolic deck, if any, or behind the * identification, * XEQ and/or * FORMAP cards if there are no compilations or assemblies. Any and all debug request cards are placed immediately behind the * DEBUG control card.

Any binary decks to be loaded must be placed, in any sequence, behind the debug request cards, if any, or behind the last source or symbolic card if there are no debug requests, or behind the * identification, * XEQ and/or * FORMAP if there are neither source nor symbolic nor debug cards.

Any data to be read by the program from logical unit 2 must be preceded by a * DATA card. The * DATA card and its accompanying data deck must

be placed behind all the above mentioned cards, i.e., following all source, symbolic, debug, and binary cards. The DATA card is not needed when there are non-execute runs, or when the object program does not read from logical 2. The deck must terminate with an end-of-file card (7 and 8 punches in column 1).

* comment and/or *PAUSE cards may be placed at any point in the deck at which the system expects control cards, provided they occur behind the * identification card and ahead of any and all * DEBUG, binary, and/or * DATA cards in the deck, as the case may be. For chain jobs, the source, symbolic, debug, and binary cards for each chain link are sequenced as described above, with the appropriate control cards for individual subprograms and debug decks. A * CHAIN card must be placed in front of the deck for each chain link. These decks are sequenced in the desired order and placed behind the * XEQ card and in front of the * DATA card, or end-of-file card if there is no * DATA card. Figure 2-6 shows the FMS control cards format.

SEE PAR. 2.2.1.4

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(1)	$	I	B	S	Y	S															
(2)	*	IDENTIFICATION																			
(3)	*						X	E	Q												
(4)	*						F	O	R	M	A	P									
(5)	*						L	I	S	T											
(6)	*						L	I	S	T	8										
(7)	*						C	A	R	D	S		C	O	L	U	M	N			
(8)	*						F	A	P												
(9)	*						D	A	T	A											
(10)	*						D	A	T	E	M	M	/	D	D	/	Y	Y			
(11)	*						S	Y	M	B	O	L		T	A	B	L	E			
(12)	*						D	E	B	U	G										
(13)	*	COMMENT																			
(14)	*						P	A	U	S	E										
(15)	*						L	A	B	E	L										
(16)	*						C	H	A	I	N										
(17)	*						L	I	B	E											
(18)	*						C	A	R	D	S		R	O	W						
(19)	*						R	O	W												

NOTE: CONTROL CARDS (1) THROUGH (9) ARE ESSENTIAL; (10) THROUGH (14) ARE USED FREQUENTLY; (15) THROUGH (19) ARE USED OCCASIONALLY.

Figure 2-6. FMS Control Cards Format

An individual description of FMS control cards is as follows:

(1) $IBSYS:

1	2	3	4	5	6	7 ... 72
$	I	B	S	Y	S	

IBSYS is called in, replacing FMS, and control is transferred to it.

(2) * Identification:

1	2 ... 72
*	IDENTIFICATION

This card is mandatory for any job which operates under FMS. Columns 2 to 72 may be filled with optional information which is reproduced by the system on the output listing for that job.

(3) * XEQ:

1	2	3	4	5	6	7	8	9	10 ... 72
*						X	E	Q	

This card informs FMS that the following job is to be loaded and executed after completing any compilations and/or assemblies for which there are source and/or symbolic decks.

(4) * FORMAP:

1	2	3	4	5	6	7	8	9	10	11	12	13 ... 72
*						F	O	R	M	A	P	

In an execution run, a memory map is printed off-line indicating names, locations, and entry points for all subprograms in the job (including library routines used), the extent of COMMON, and the amount of vacant storage.

(5) * LIST:

1	2	3	4	5	6	7	8	9	10	11 ... 72
*						L	I	S	T	

The system is instructed to print a symbolic FAP listing of the FORTRAN program following this card, after it has been compiled. The generated listing is printed three columns of instructions per page.

(6) * LIST8:

1	2	3	4	5	6	7	8	9	10	11	12	72
*						L	I	S	T	8		

Same as * LIST except that the listing is printed in two columns with the octal representation of each instruction.

(7) * CARDS COLUMN:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	72
*						C	A	R	D	S		C	O	L	U	M	N		

This card causes a column binary relocatable deck to be punched off-line for the source or symbolic program following this card, after it has been compiled or assembled.

(8) * FAP:

1	2	3	4	5	6	7	8	9	10	72
*						F	A	P		

This card identifies the symbolic deck that follows it to be assembled as a machine language program by FAP.

(9) * DATA:

1	2	3	4	5	6	7	8	9	10	11	72
*						D	A	T	A		

This card identifies those cards that follow as data cards to be read during object program execution. It is not needed for non-execute jobs, nor those requiring no data on logical 2.

(10) * DATE:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	72
*						D	A	T	E	M	M	/	D	D	/	Y	Y		

The date supplied on this card is printed on each page of output generated by the system for the job.

(11) * SYMBOL TABLE:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	72
*						S	Y	M	B	O	L		T	A	B	L	E		

This card causes the system to punch (off-line) a symbol table for the FORTRAN source program following it after it has been compiled. The symbol table is required by the debug package if debug requests are to be embedded in this particular subprogram.

(12) * DEBUG:

1	2	3	4	5	6	7	8	9	10	11	12	72
*						D	E	B	U	G		

This card identifies the cards that follow as debug requests. The interpretation of debug requests stops when a binary card, * DATA card, or end-of-file is encountered.

(13) * comment:

1	2	72
*	COMMENT	

The contents of this card are printed on-line and off-line.

(14) * PAUSE:

1	2	3	4	5	6	7	8	9	10	11	12	72
*						P	A	U	S	E		

This card causes the machine to halt. The job is continued by instructing the operator to press the START button. Since all control cards are printed on-line, columns 12-72 may contain a message to the operator.

(15) * LABEL:

1	2	3	4	5	6	7	8	9	10	11	12	72
*						L	A	B	E	L		

The relocatable binary cards produced as a result of the compilation or assembly of the deck following this card will be labeled and serialized in columns 73-80. The label punched in columns 73-78 for a FORTRAN compilation will be taken from columns 2-7 of the first card of the source deck (excluding other control cards) provided this card has a C in column 1. Otherwise, the subprogram name is used (000000 for a main routine). For FAP assemblies, columns 2-7 of the page title card will be used as a label. The cards will be serially numbered in columns 79 and 80, beginning with 00 and recycling when 99 is reached. However, if the label does not use all of columns 73-78, serial numbering will occupy all unused columns at the end of the label, recycling when 10^n-1 is reached (where n is the number of columns available). Symbol tables and all subroutines obtained with the deck (i.e., LIBE) will be labeled with their own names.

(16) * CHAIN (R or T)

1	2	3	4	5	6	7	8	9	10	11	12	72
*						C	H	A	I	N		

This card identifies all cards that follow until another * CHAIN, a * DATA, or end-of-file is encountered, as a chain link. R is an identifying label for the tape record which will contain this link, and must be an integer greater than 0 and less than 32,768. T is the actual unit designation of the tape on which this link is to be recorded. Units B2, B3, and A4 are the only ones which may be used. The argument T in the * CHAIN control card may be one of these physical unit designations, or one may use simply the digits 2, 3, or 4, respectively. Either notation is acceptable. The argument T in the CALL CHAIN statement within the program must use the external logical unit designations for these tapes, namely 12, 13, or 4, respectively.

(17) * LIBE:

1	2	3	4	5	6	7	8	9	10	11	72
*						L	I	B	E		

This card causes the Monitor to search the library following a compilation, and punch out relocatable decks for any Library subroutine required by the subprogram just compiled.

(18) * CARDS ROW:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	72
*						C	A	R	D	S		R	O	W		

Row binary cards are punched off-line for the compilation or assembly to which this control card is applied. If the routine compiled is a FORTRAN main routine, a nine card BSS loader will be punched out ahead of the deck, and a transfer card at the end. (A FORTRAN II transfer card has a 9 punch in column 1.)

(19) * ROW:

1	2	3	4	5	6	7	8	9	10	72
*						R	O	W		

Same as CARDS ROW.

2.2.2 DECK STRUCTURE

Deck order is directly related to performance order. The first function to be performed should have its cards before the second function to be performed, and so on. The following sample decks illustrate various operations to be performed under IBSYS or FMS control.

2.2.2.1 FORTRAN IV--Compilations, No Execution

Routine XYZ is to be compiled for the 7094 Data Processing System. A relocatable binary deck is output; nothing else. Since no options are specified in the $IBJOB and $IBFTC cards, this is equivalent to cards in which the standard options are punched; namely:

```
$IBJOB      NOGO,NOLOGIC,NOMAP,NOFILES,MINIMUM,SOURCE,FLOW
```

and

```
$IBFTC XYZ  NOLIST,NOREF,DECK,M94,XR7,NODD
```

Similarly, in all following examples, where a standard option is called for, it does not appear in the control card. The $DATE card is always optional, but its inclusion is recommended.

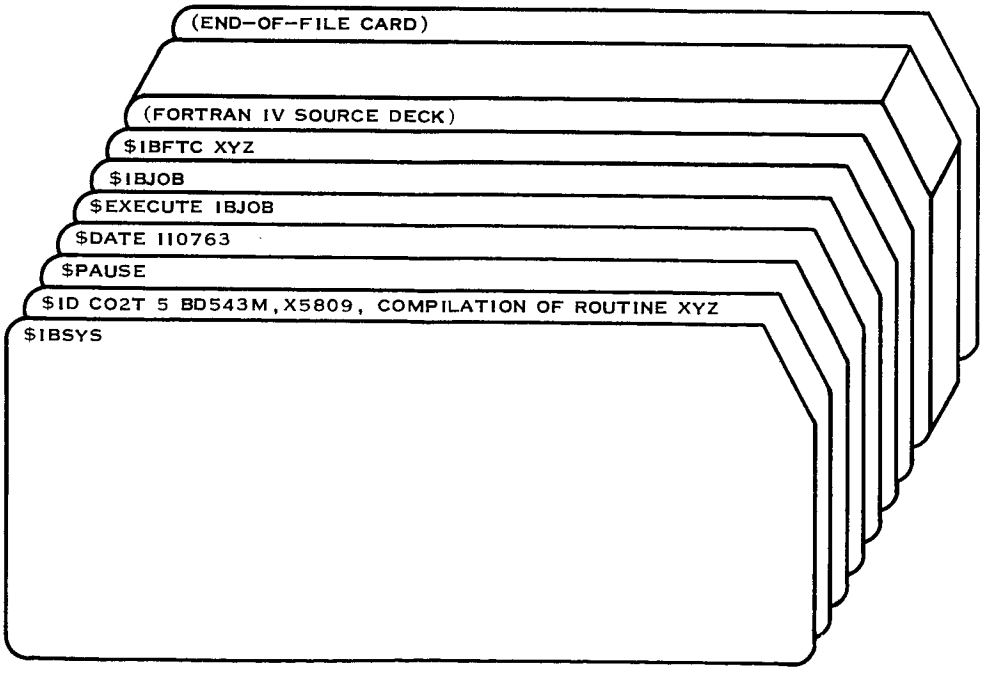


Figure 2-7. Sample FORTRAN IV Compilation, No Execution

2.2.2.2 FORTRAN IV--Execution Run from Binary Decks

Execution run from binary decks begins with the subprogram, MAIN. The programmer desires a storage map. His program reads data cards from logical unit 2.

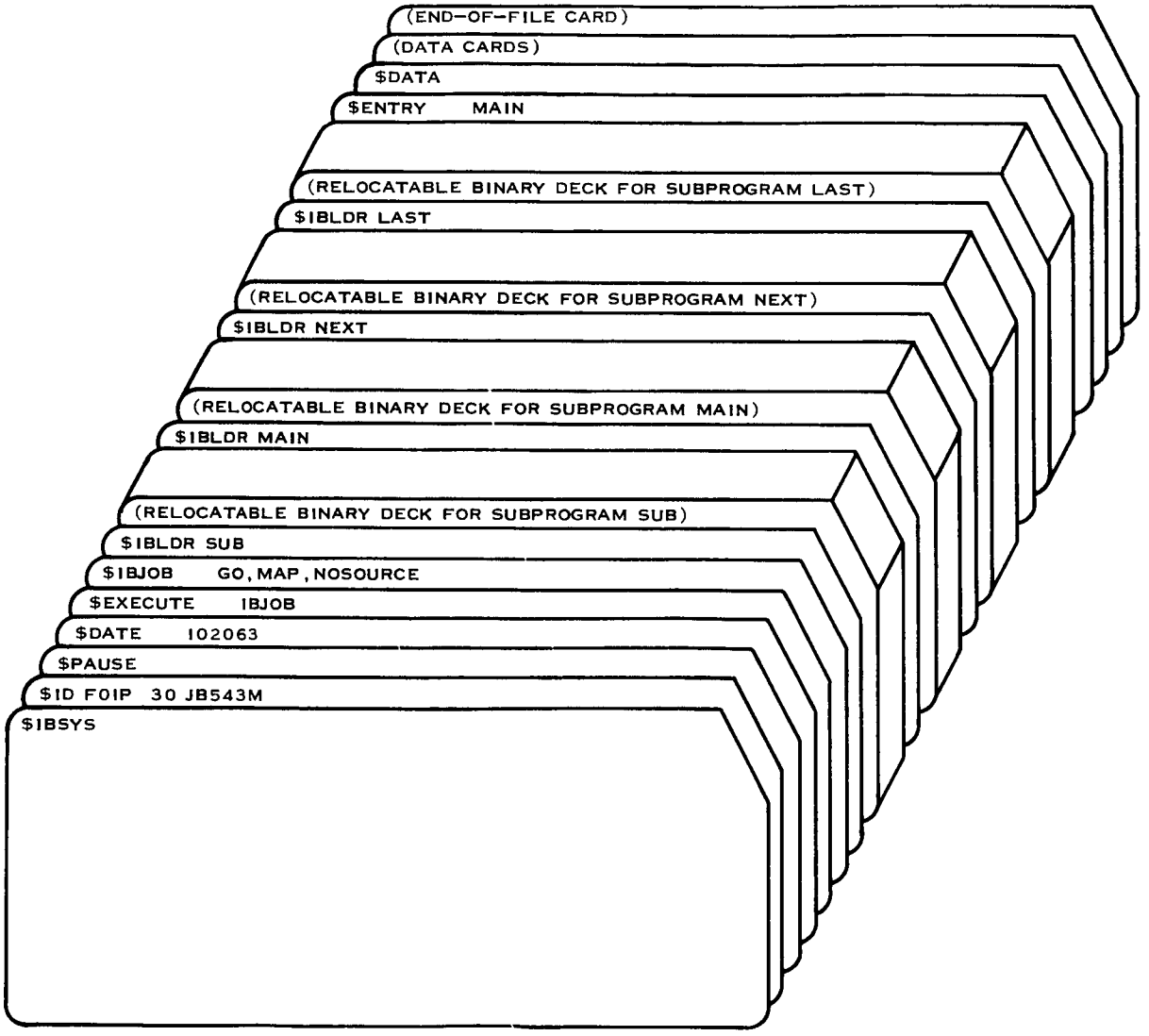


Figure 2-8. Sample FORTRAN IV Execution Run from Binary Decks

2.2.2.3 Multiple FORTRAN IV Compilations and IBMAP Assemblies

Execution begins with the main routine, if any; otherwise, it begins with SUB1. These routines are to be compiled/assembled for the 7094 DPS, using all seven index registers. There are approximately 550 cards to illustrate the IBMAP language routine which uses FORTRAN macros. Do not consider symbols in parentheses as errors. A listing of all subprograms except SUB1 is wanted.

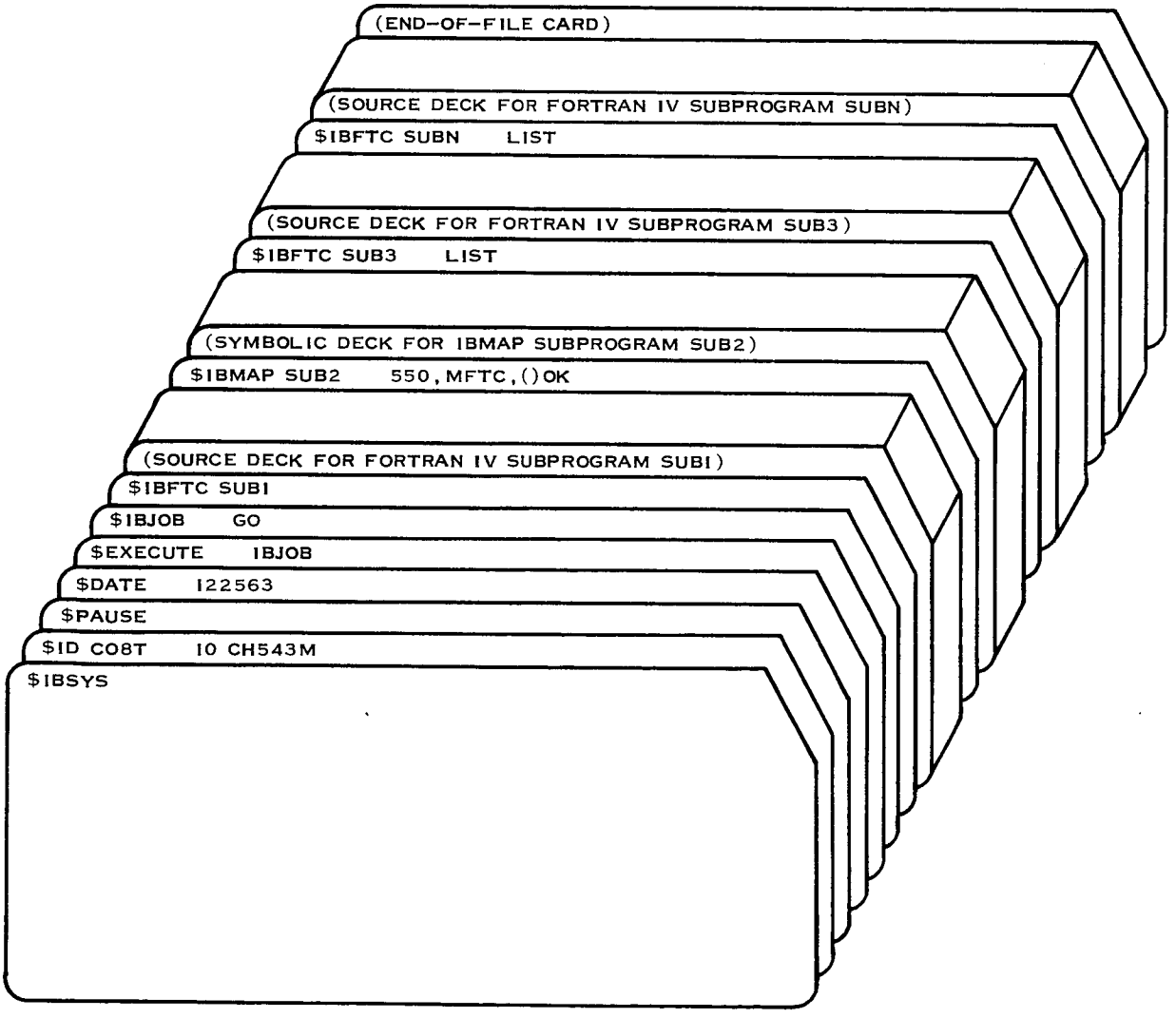


Figure 2-9. Sample Multiple FORTRAN IV Compilations and IBMAP Assemblies

2.2.2.4 FORTRAN IV--Compilation, IBMAP Assemblies, Load Binary Decks

This deck performs several functions, such as 1) FORTRAN IV compiling, 2) IBMAP assembling, and 3) loading relocatable binary decks. It executes the object program beginning with subprogram P4. Data from logical 2 are read during execution. A storage map and logic listing are desired, as well as a printout of the symbol cross-reference table for all compilations and assignments; and no punched output for routine P3 are desired. Only 3 index registers may be used when compiling (assembling) for the 7094 DPS. No FORTRAN macros are desired in IBMAP routine P5; they are desired for routine P6. A PREST deck is to be punched for P5, in addition to the relocatable binary deck.

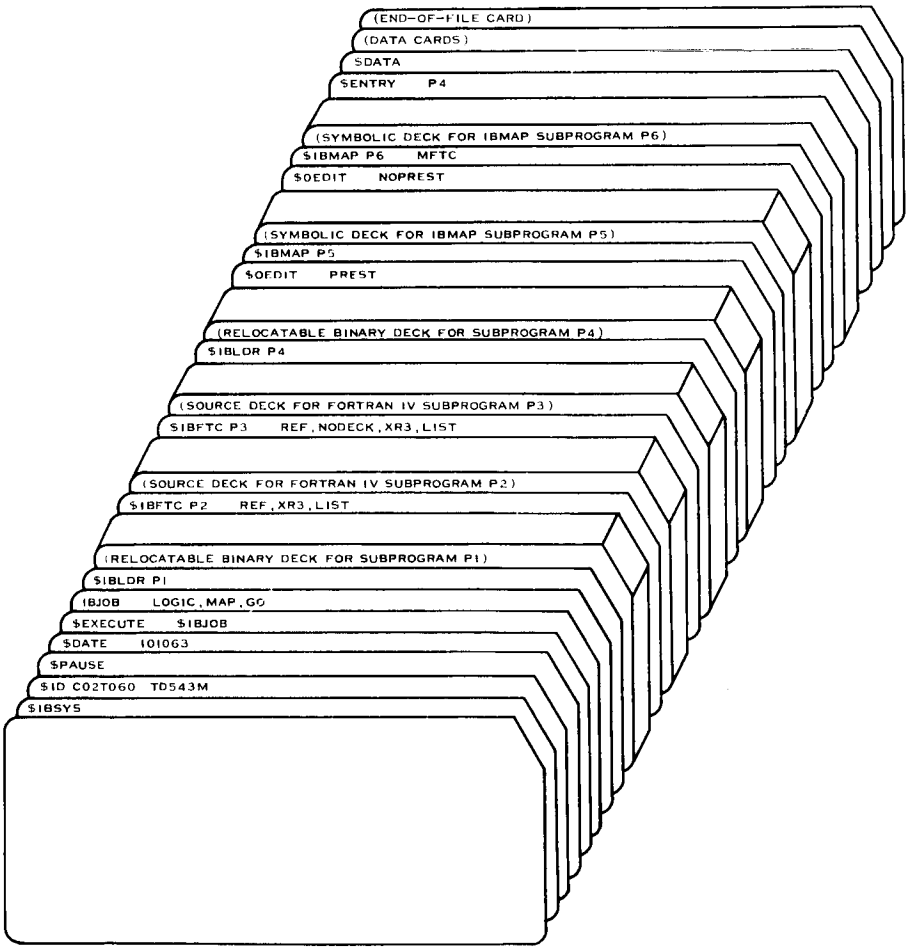


Figure 2-10. Sample FORTRAN IV Compilations, IBMAP Assemblies, Load Binary Decks

2.2.2.5 FORTRAN IV--Execute Programs from Binary Tape

A programmer has his tape containing many previously compiled or assembled subprograms, with a suitable control card preceding each routine. He may want to select five of these routines which are interdependent and execute them beginning with the one named STARTX. The decks for STARTX, ORBIT, and INTEG are known to be consecutive on this tape. Do not load routine DUMMY. However, it is called to define its COMMON definitions, etc., in memory ahead of all other routines in the job. Data cards are to be read from logical unit 2.

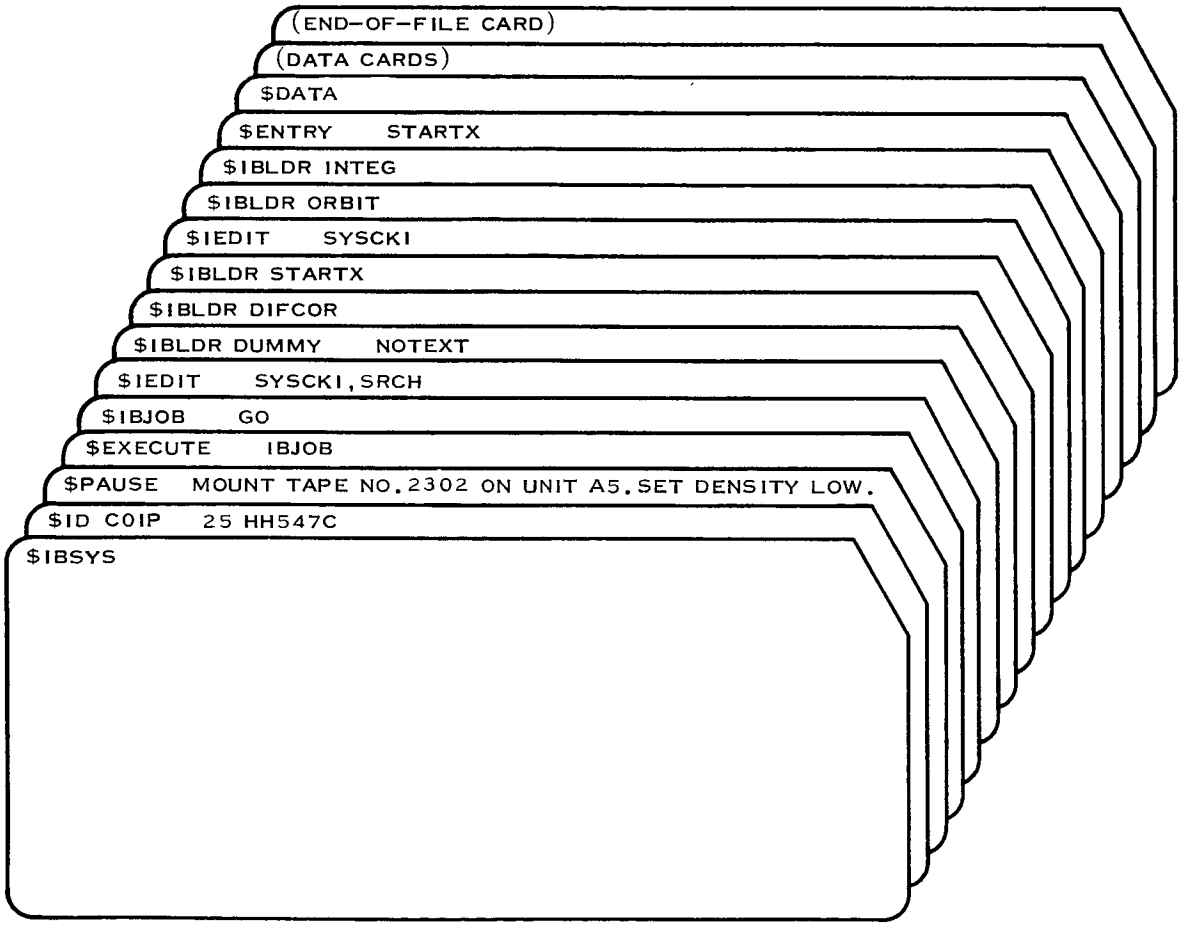


Figure 2-11. Sample FORTRAN IV Execute Programs from Binary Tape

2.2.2.6 FORTRAN IV--Modify PREST Decks and Execute

A programmer has a symbolic deck for subroutine FLAP and a PREST deck for subroutine CLAP on alternate tape No. 2300. In addition, a PREST deck for subroutine WHAP and a relocatable deck for subroutine SLAP are to be included with the input deck. There are ALTERs for FLAP and WHAP, and the routines are to be loaded in this order--WHAP, FLAP, CLAP, SLAP. No punch output is desired except a PREST deck for FLAP. Execution is to begin with the first entry point of routine WHAP. No input is read from logical 2 during execution.

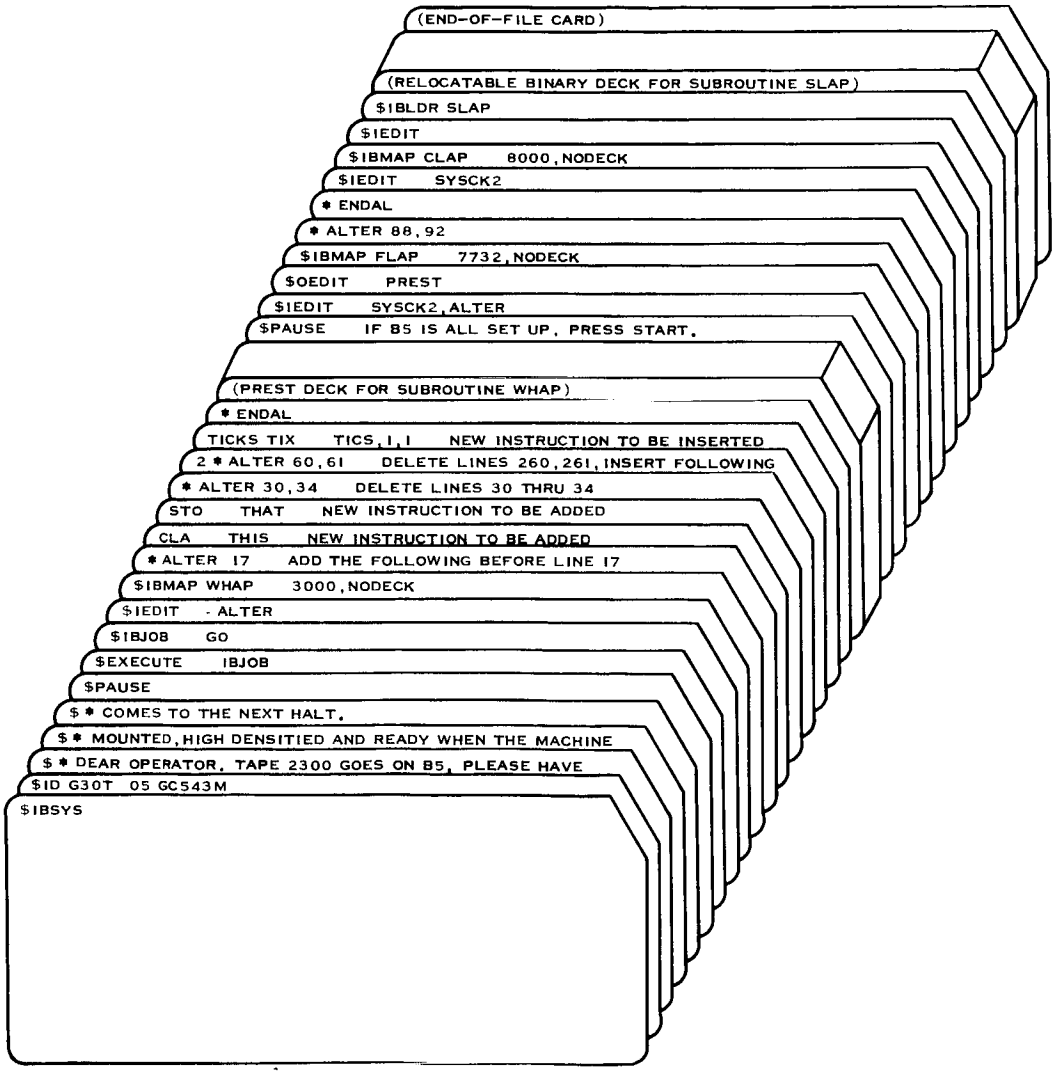


Figure 2-12. Sample FORTRAN IV Modify PREST Decks and Execute

2.2.2.7 FORTRAN IV--Overlay Job

For simplicity, it is assumed that relocatable binary decks are available for all subprograms. However, this is not a requirement. The main link (or link 0) consists of two decks, ZORBA and GLESPY. Each of two dependent links is to begin immediately after the main link. Of these, link 1 consists of two decks, CASEY and MAGGIE; and link 2 consists of one deck, KILDAR. Links 3 and 4 are each to occupy core immediately following KILDAR. Link 3 consists of one deck, SPOCK; and link 4 consists of two decks, JEKYLL and HYATT, plus the library FDMD, FLOG, and FATN. Links 1 and 2 are to be written on SYSUT2; links 3 and 4 on SYSUT3. During execution, SYSUT2 is to be rewound each time after link 2 is loaded, but not after link 1 (since the tape will be properly positioned to read link 2). Similarly for SYSUT2 and links 3 and 4.

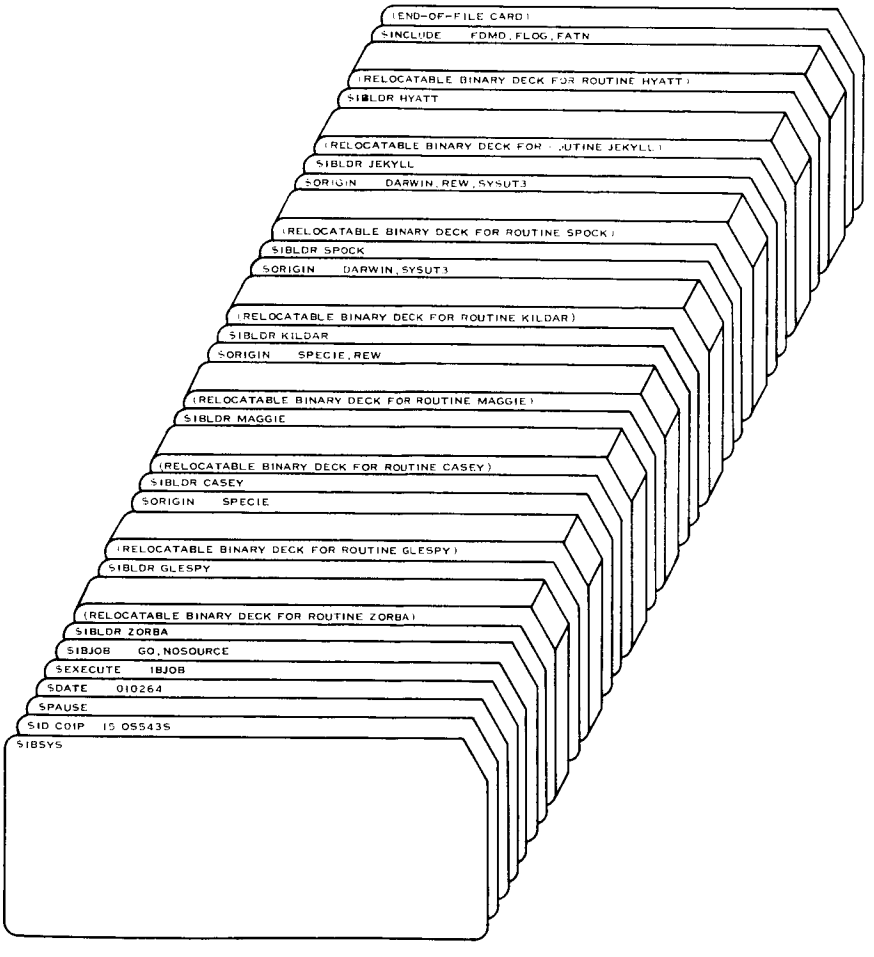


Figure 2-13. Sample FORTRAN IV Overlay Job

2.2.2.8 FORTRAN IV--Debug Execution

The deck structure specifies that the main program MAIN and the subroutine SUBR, written in FORTRAN IV source language, are to be compiled and executed. Further, the debugging option has been specified. Whenever statement 20 in the main program or statement 12 in the subroutine are executed, control is transferred to the debug package. At this time, BETA or GAMMA are dumped and control is returned to the calling routine.

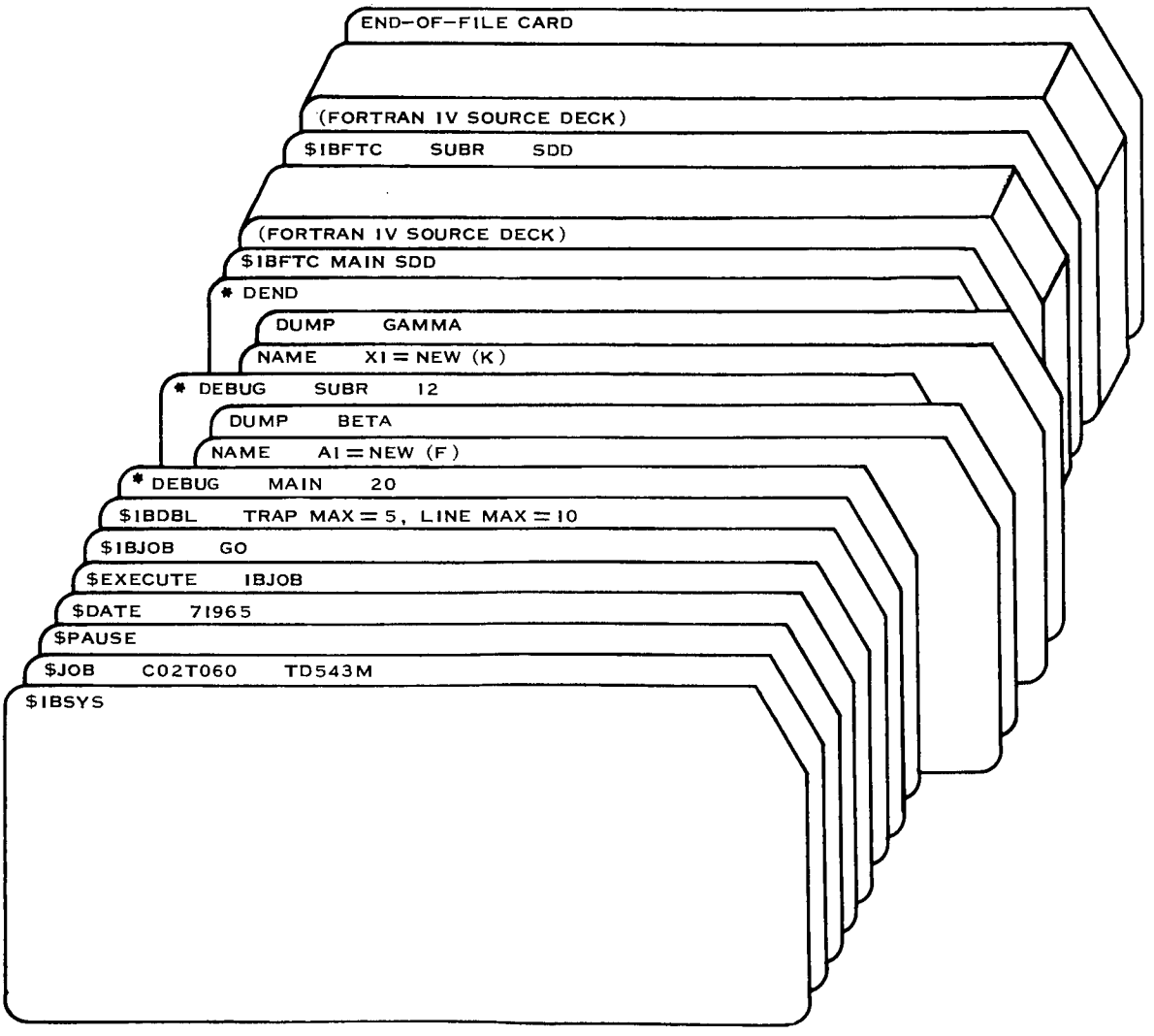


Figure 2-14. Sample FORTRAN IV Debug Execution

2.2.2.9 FORTRAN II--Execute

The decks of binary cards are loaded and the program executed. The PAUSE card is required by operations. It causes a halt to occur, during which time the programming may inform the computer operator of any necessary action he must take.

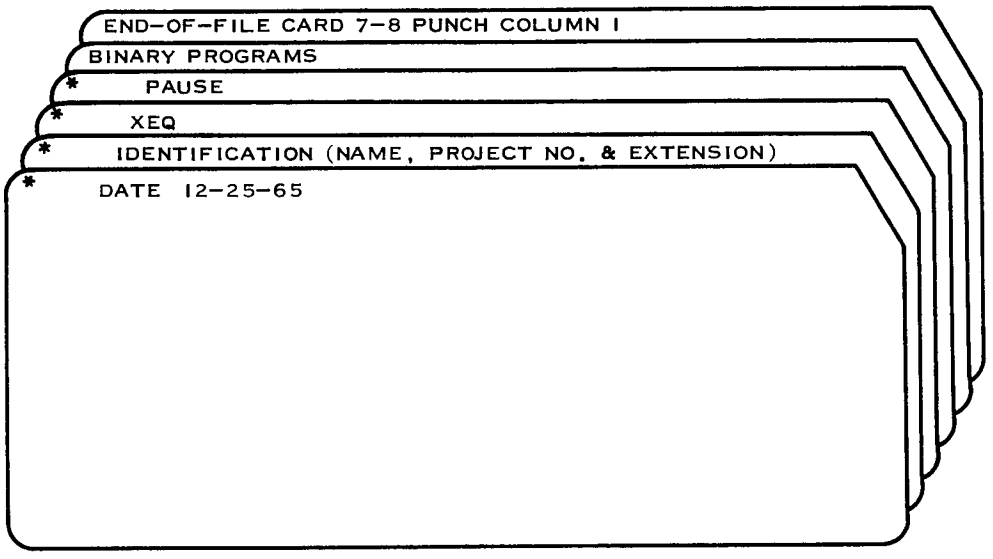


Figure 2-15. Sample FORTRAN II Execute

2.2.2.10 FORTRAN II--Compile

The FORTRAN II program is compiled. The output requested is a relocatable column binary deck to be punched off-line and a FAP listing of the program along with the octal representation of each instruction.

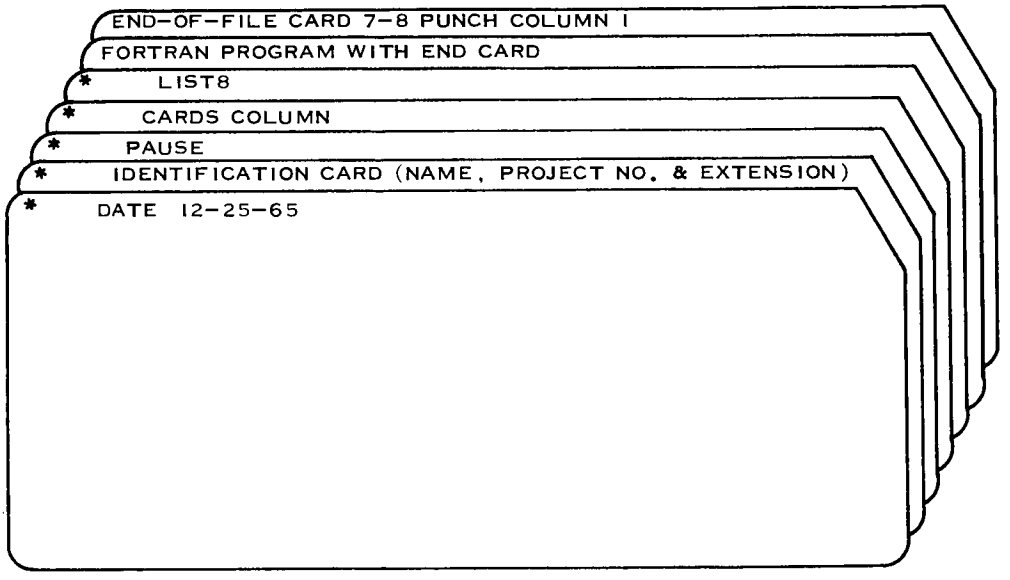


Figure 2-16. Sample FORTRAN II Compile

2.2.2.11 FORTRAN II--Compile and Execute

The FORTRAN II source language program is to be compiled and executed. A symbolic FAP listing of the program is required. Row binary for the compiled FORTRAN main routine along with a nine-card BSS loader deck are to be punched out off-line. During the execution of the object program, information contained on the data cards is to be read in.

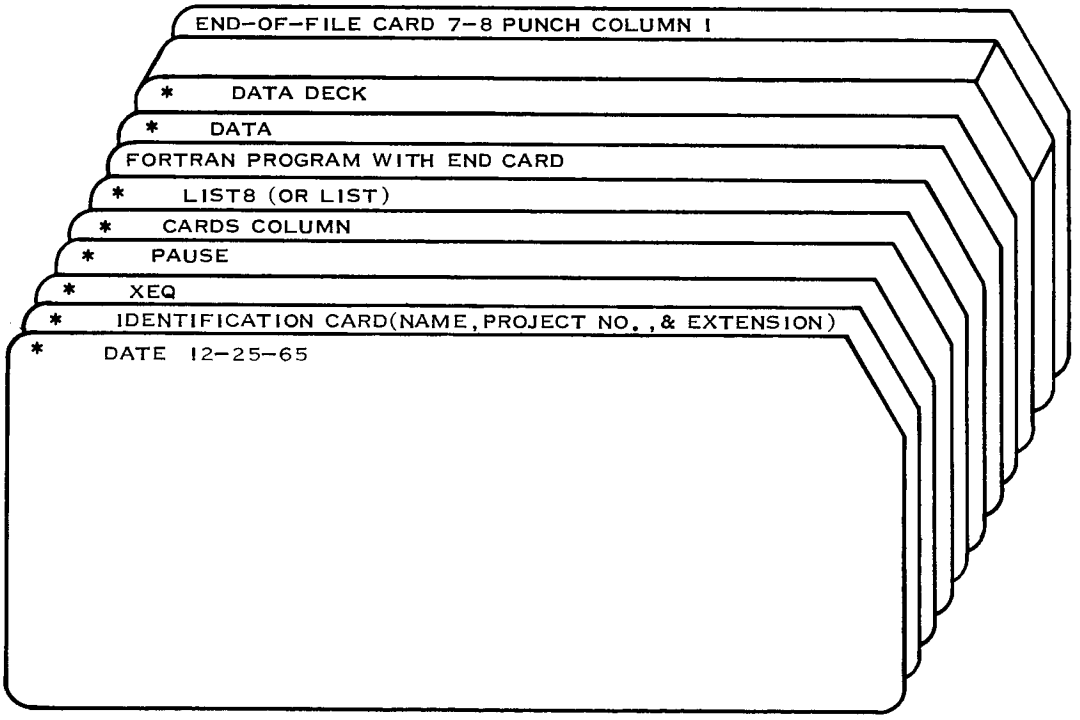


Figure 2-17. Sample FORTRAN II Compile and Execute

2.2.2.12 FORTRAN II--Compile and Execute with Binary Subroutines

The FORTRAN program is to be compiled and the deck(s) of binary subroutines are to be loaded before execution of the program. A FAP listing is requested and a relocatable column binary deck of the program is to be punched off-line. The binary subroutines must be in column binary form.

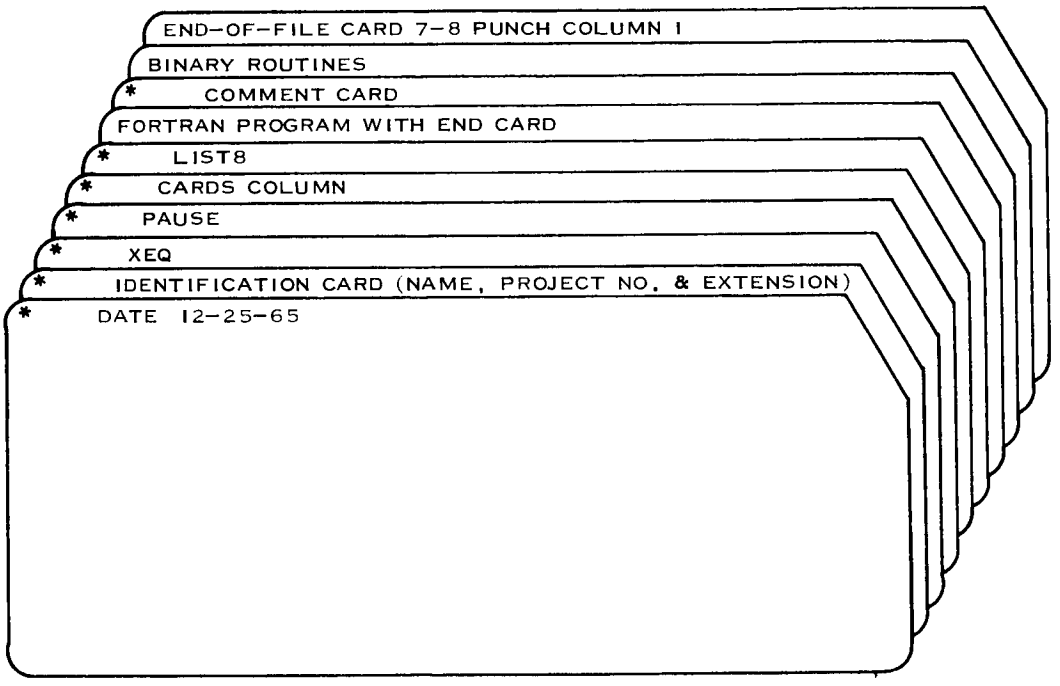


Figure 2-18. Sample FORTRAN II Compile and Execute with Binary Subroutines

2.2.2.13 FORTRAN II--Compile, Assemble, and Execute

The FORTRAN II source language program is to be compiled. The following FAP language program is to be assembled. The complete object program is to be executed during which time the data cards are loaded. A FAP listing and a relocatable column binary deck are requested.

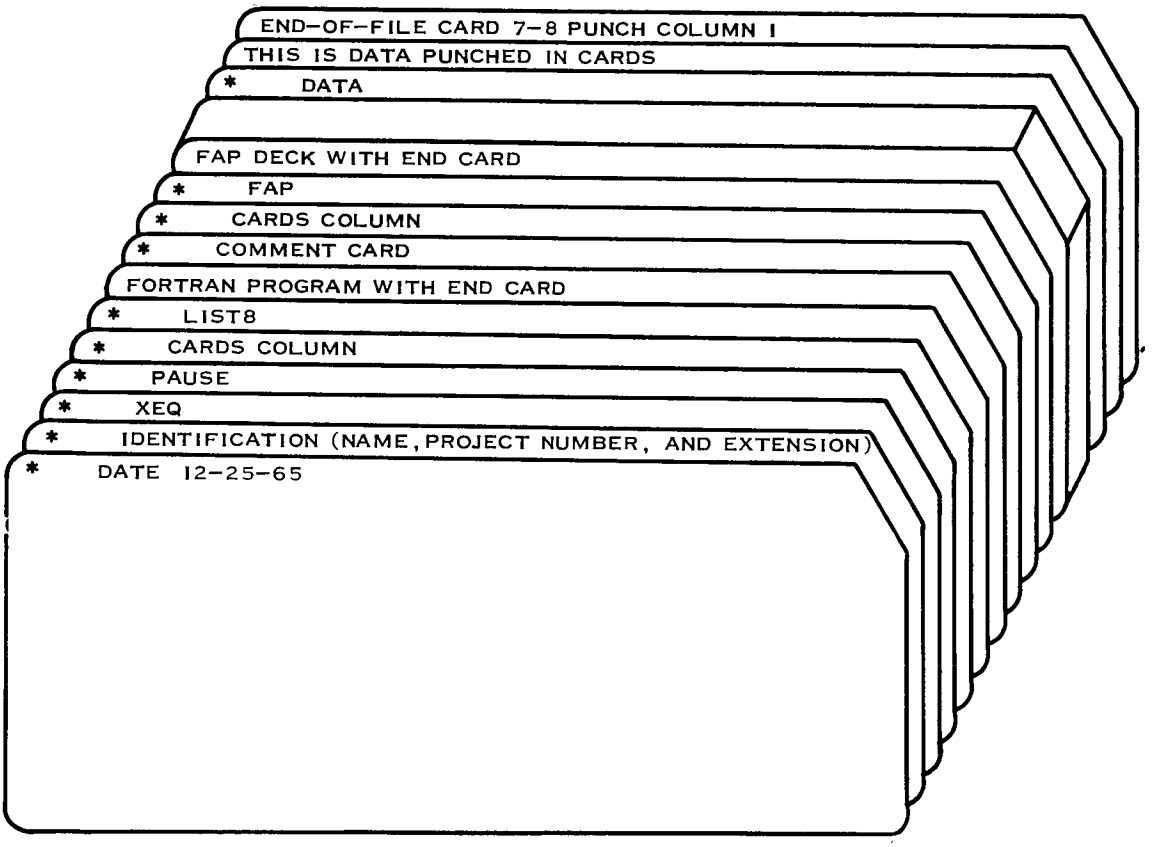


Figure 2-19. Sample FORTRAN II Compile, Assemble, and Execute

2.2.2.14 FORTRAN II--Compile, Execute, and Debug

The FORTRAN II program is to be compiled and executed. A symbol table, required for debugging, and a memory map are requested. The debugging option is specified and will occur at statement 60 in the FORTRAN program. The variable MA and MB are to be dumped the first time statement 60 is executed up to a maximum of four times.

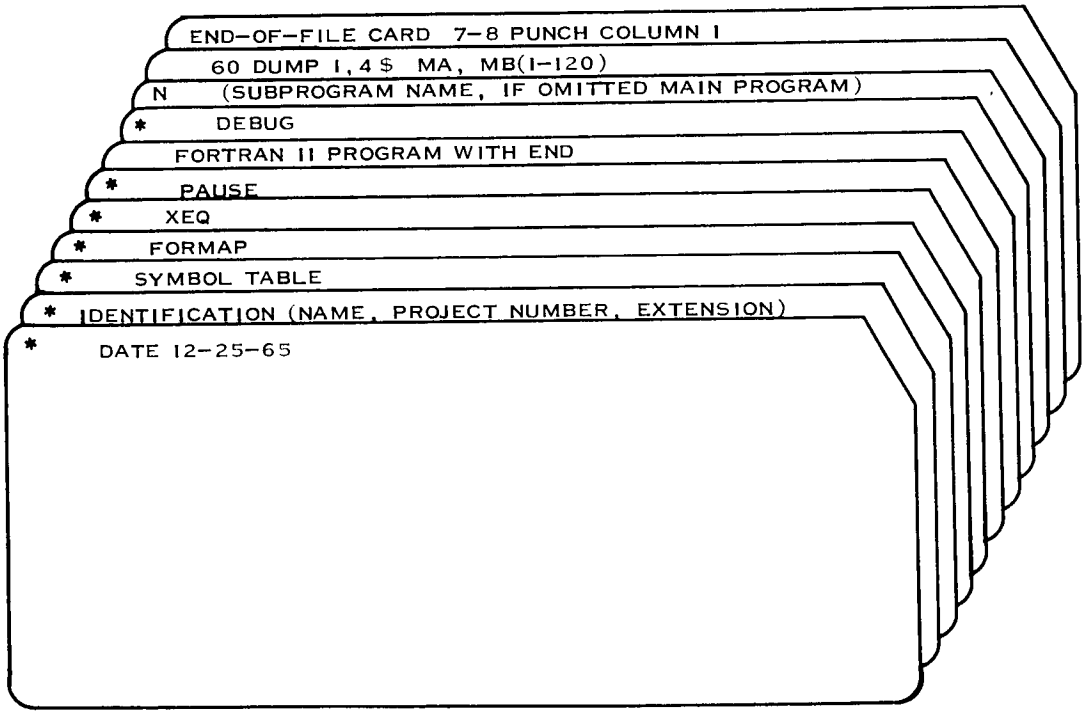


Figure 2-20. Sample FORTRAN II Compile, Execute, and Debug

2.2.3 PROGRAMMING AIDS

This paragraph presents to the programmer tips and techniques as well as precautions to be considered while performing programming functions. These programming aids should be of help to the programmer in his utilization of the 7094 FORTRAN system.

2.2.3.1 FORTRAN II

(1) FORTRAN II Diagnostic: The FORTRAN II double-precision square root produces a diagnostic when given a negative argument and execution is terminated. The reaction to double-precision division by zero has been modified such that a diagnostic is printed off-line, but the program is continued. The quotient will be set equal to the largest DP value the 7094 can represent (i.e., MSH = 377 777777777_8, LSH = 344 777777777_8).

(2) FORTRAN II Dump Routines: The FORTRAN II dump routine attempts a recovery if a tape check occurs during the dumping process. This applies to PDUMP, as well as DUMP. It also will produce dumps of B Core for programs which use 65K. The programmer may call both B Core (or a portion thereof) using CALL DUMP or CALL PDUMP by specifying format code 5, 6, 7, or 8 instead of 0, 1, 2, or 3, respectively. Multiple sets of arguments may be used within the CALL, but all requests from A Core should be given first. Once a reference to B Core is encountered in executing any such call (scanning left to right), all requests to the right will be taken from B Core regardless of the format code given. B Core dumps may also be initiated from the console by depressing Key 19.

(3) IODINE Disk Routine: A disk routine, IODINE, is available from the FORTRAN II system library. There is a writeup outlining its usage. The explicit programming standards for the use of disk exist. Users of IODINE (or any other disk I/O routine) must be aware of these standards. (See DSD Memorandum, "1301 Disc Standards", dated 4/27/64.)

(4) UMPLOT: A version of UMPLOT compatible with FORTRAN IV and IOCS exists on the system library under IBJOB. It is used for both FORTRAN II and IV. UMPLOT has been in existence on the FORTRAN II library. (See Paragraph 2.5.1 for complete description.)

2.2.3.2 FORTRAN IV

(1) Accuracy of FORTRAN IV Library Mathematical Subroutines: There is available a set of accuracy estimates for the mathematical subroutines in the FORTRAN IV library (IBLIB). These estimates are derived from test results using in many cases a large number of selected points in the range of each subroutine. The following contains these estimates for most of the single precision, double precision and complex routines in the library.

1. Single Precision

a. XPI/Exponential-FXPT Base-FXPT Exponent--Tests run for 2^X where X = 1, 2, ..., 27 gave a relative error $\approx 1 \times 10^{-8}$. Tests run for 10^X where X = 1, 2, ..., 10 showed no error

b. XP2/Exponential-FLPT Base-FXPT Exponent--Tests run for 10^X where X = $\pm$ 1, $\pm$ 2, ..., $\pm$ 33 gave a maximum relative error = 4×10^{-8}

c. FXP3/Exponential-FLPT Base-FLPT Exponent--Tests run for 2^X where X = $\pm$ 1, $\pm$ 2, ..., $\pm$ 39 gave a maximum relative error = 3×10^{-7}

d. FXPF/Floating Point Natural Exponential--Tests run for a larger number of selected points gave a relative error:

$\approx 2 \times 10^{-8}$	$\lvert X \rvert < 10$
$\approx 5 \times 10^{-8}$	$10 \leq \lvert X \rvert < 65$
$\approx 5 \times 10^{-7}$	$65 \leq \lvert X \rvert < 88$

e. FLOG/Floating Point Logarithm--Tests run for a large number of selected points in the range $10^{-38} < x < 10^{38}$ yielded a relative error = 1×10^{-8}, except in the neighborhood of X = 1, where the absolute error was = 1×10^{-8}

f. FATN/Floating Point Arctangent--ATAN(X), Tests run for a larger number of selected points in the range $|X| < 10000$ gave a relative error $\approx 1 \times 10^{-8}$
For values of X of the order 10^{-4}, ATAN(X) = X
ATAN2$\left(\frac{Y}{Z}\right)$, Tests run for Y = 2M - 1, M = $\pm$ 1, $\pm$ 2, ..., $\pm$ 10
Z = $\pm$ 1, $\pm$ 2, ..., $\pm$ 60 and
Tests run for Y = 2M, M = $\pm$ 26, $\pm$ 27, ..., $\pm$ 54
Z = $\pm$ 61, $\pm$ 62, ..., $\pm$ 100 gave a relative error $\approx 1 \times 10^{-8}$

g. FSCN/Floating Point Sine and Cosine--Tests run for SIN(X) and COS(X) for a large number of selected points with $0 \leq X \leq \pi/2$ gave an absolute error $\approx 1 \times 10^{-8}$ for SIN(X) and an absolute error $\approx 5 \times 10^{-8}$ for COS(X)

h. FSQR/Floating Point Square Root--Tests run for a large number of selected points in the range $10^{-38} < x < 10^{38}$ gave a relative error $\approx 1 \times 10^{-8}$

2. Double Precision

a. FDX1/Double Precision and Complex Exponential Functions--FDX1, Tests run for 10^X X = $\pm$ 1, $\pm$ 2, ..., $\pm$ 33 gave a maximal relative error $\approx 6 \times 10^{-16}$
CXP1, Tests run for $(10 + 10i)^X$ where X = $\pm$ 1, $\pm$ 2, ..., $\pm$ 33, gave a relative error $\approx 1 \times 10^{-8}$ for both the real and imaginary parts of the result.

b. FDX2/Double Precision Exponential Function--Tests run for a large number of selected points with bases 2, 10, e, gave a relative error $\approx 9 \times 10^{-16}$ for $|X| < 10$ and a relative error $\approx 7 \times 10^{-15}$ for $10 \leq |X| < 65$

c. FDXP/Double Precision Natural Exponential Function--Tests run for a large number of selected points gave:
relative error $\approx 7 \times 10^{-16}$ for $|X| < 10$
relative error $\approx 5 \times 10^{-15}$ for $10 \leq |X| < 65$
relative error $\approx 5 \times 10^{-14}$ for $-70 \leq X \leq 88$

d. FDSQ/Double Precision Square Root Function--Tests run for a large number of selected points in the range $10^{-29} < X < 10^{37}$ gave a relative error $\approx 1 \times 10^{-16}$

e. FDLG/Double Precison Logarithmic Function--Tests run for a large number of selected points in the range $10^{-29} < X < 10^{38}$ gave a relative error 5×10^{-16} except in the neighborhood of $X = 1$, where the absolute error was $\approx 3 \times 10^{-16}$

f. FDSC/Double Precision Sine and Cosine Functions--Tests run for values of $X = (30n)^\circ$ $n = 0, 1, 2, \ldots, 12$ gave a maximal absolute error = 5×10^{-15}

3. Complex

a. FCAB/Complex Absolute Value--Tests run for a large number of selected points gave a relative error $\approx 3 \times 10^{-8}$ whenever the relative error in $\frac{X}{Y}$ or $\frac{Y}{X}$ was $\leq 1 \times 10^{-8}$

b. FCLG/Complex Natural Logarithm--Tests run for a large number of selected points in the range $1 \leq X, Y \leq 1000$ gave a relative error $\approx 4 \times 10^{-8}$ in the real part and $\approx 1 \times 10^{-8}$ in the imaginary part of the result (provided the relative error in $\frac{X}{Y}$ or $\frac{Y}{X}$ was $\leq 1 \times 10^{-8}$)

c. FCSQ/Complex Square Root--Tests run for a large number of selected points in the range $10^{-39} \leq X, Y \leq 10^{38}$ gave a relative error $\approx 3 \times 10^{-8}$ whenever the relative error in $\frac{Y}{2R}$ was $\leq 1 \times 10^{-8}$ and no overflow or underflow occurred upon squaring of X and/or Y.

(2) Binary Mode Tape Operations: In FORTRAN IV, where the statements REWIND i and/or BACKSPACE i are used to manipulate binary mode tapes, it was necessary to add 30_{10} to the logical unit number. For example, to WRITE (17) ... , REWIND 47 was required when using Version 8. This is not required under Version 12.

(3) Punch Statement--FORTRAN IV: FORTRAN IV punch statements will produce card decks off-line. That is, card images will be recorded on the output tape and will be recognized as such by the 1401. The 1401 must backspace and reread each time a mode change is encountered. In the interests of off-line efficiency, group punching operations together whenever practical.

(4) DUMP/PDUMP Routine: The DUMP/PDUMP routine under IBJOB has an additional entry point to permit the programmer to obtain a full core dump (32K) within FORTRAN IV programs. This is accomplished by executing the statement:

```
CALL FPDUMP (A, B, I)
```

where A and B are dummy arguments and may be any variable names defined within that subprogram. I is the format code (0, 1, 2, or 3).

(5) IBSYS Debug: Version 12 provides the FORTRAN IV feature of input-output without explicit LIST and FORMAT, as described in IBM manual C28-6377. Version 12 also incorporates the debugging facilities described in the IBJOB manual C28-6247-4, and in the supplementary debugging manual C28-6362-1. An additional tape, B10, is required when debugging features are used. The programmer should indicate B10 on his job request card. This unit designation may be reassigned if necessary (e.g., for using debug in programs which may be using logical 20, or in which there are no free drives on channel B). Note, however, that debug refers to this unit as SYSCK2, which is also an optional overlay tape. Conflicts here can be resolved by changing the choice of the overlay tape.

(6) UMPLOT: A version of UMPLOT compatible with FORTRAN IV and IOCS exists on the system library under IBJOB. UMPLOT has been in existence on the FORTRAN II library. (See Paragraph 2.5.1 for complete description.)

2.2.3.3 65K Dump Routine--Operating Instructions

Operating instructions for the FORTRAN system tape dump routines are as follows:

a. format and limits in keys of console
b. lower limit in decrement
c. upper limit in address
d. format in keys 1 and 2 as follows:

Operation	Key 1	Key 2
octal	up	up
octal with mnemonics	down	up
floating point	up	down
decimal integer	down	down

Key 20 down indicates that another dump request is to follow. The 7094 will stop with HTR 67344 to allow the keys to be reset. When key 20 is up, B CORE is dumped. The dump from B CORE will be in the same format as the last dump from A CORE.

A zero in both the address and the decrement causes A CORE and B CORE to be dumped. To dump only B CORE, put key 19 down (without key 20 down) when the last portion of A CORE is dumped.

2.2.3.4 7094 Machine Language I/O

All programming shortcuts on the 7090/94 Mod I are not applicable on the 7094 Mod II. On the 7090/94 Mod I, the non-data-select instructions (BSR, BSF, REQ, RUN, and WEF) require only that the channel and unit be specified. On the 7094 Mod II, the full address is required. For example, to manipulate unit B6 with an instruction on the 7090/94 Mod I, the octal configuration 2006 in the address is sufficient. However, on the 7094 Mod II, the full address (2206 or 2226 octal) must be given. Otherwise, the machine will hang up.

Normally, the correct address is generated by the assembly program (e.g., if REWB 6 is written). If the programmer wants to compute a variable unit address to store in an instruction written, for example, REW**, the program must supply the full address in order for it to run properly on the 7094 Mod II.

2.2.3.5 Channel Tape Assignments

Under IBSYS, B(1), the first available unit on channel B is B5. A(1) is A5 and C(1) is C1. The system symbolic unit references are listed below. The B10 unit is required when the debug package is specified.

A1 SYSLB1; A2 SYSIN1; A3 SYSOU1; A4 SYSUT1
B1 SYSUT2; B2 SYSUT3; B3 SYSUT4; B4 SYSPP1; B10 SYSCK2

2.3 BIBLIOGRAPHY

This section provides the user access to the list of documents describing the major components of the IBM 7090/7094 FORTRAN Operating System. For each document there is provided an abstract along with a table listing each document (Table 2-3) and a form-number index of each document (Table 2-4) for cross-referencing purposes.

Table 2-3. 7094 FORTRAN Documentation Listing

Table 2-3. 7094 FORTRAN Documentation Listing (Cont'd)

PARAGRAPH		FORM NO.	PAGE NO.
2.3.5.2	IBM 7090/7094 Programming Systems: FORTRAN II Operations	C28-6606-6	2-64
2.3.5.3	IBM 7090/7094 FORTRAN IV Compiler (IBFTC) Replacement: Specifications and Language Additions	C28-6376	2-64
2.3.5.4	FORTRAN	F28-8074-3	2-64
2.3.5.5	IBM 7090/7094 Programming Systems: FORTRAN IV Language	C28-6274-4	2-65
2.3.5.6	IBM 7090/7094 IBSYS Operating System: IBJOB Processor	C28-6275-4	2-65
2.3.5.7	IBM 7090/7094 IBSYS Operating System: Specifications for IBJOB Processor Debugging Package	C28-6362-1	2-65
2.3.5.8	7090/7094 PROGRAMMING SYSTEMS: IBJOB Processor, Overlay Feature of IBLDR	C28-6331	2-65
2.3.5.9	IBM 7090/7094 IBSYS Operating System: Input/Output Control System	C28-6345-2	2-65
2.3.5.10	IBM 7090/7094 IBSYS Operating System: Utilities	C28-6364-3 with N28-0125-D	2-66
2.3.5.11	IBM 7090/7094 Generalized Sorting System: 7090/7094 Sort	C28-6307	2-66
2.3.5.12	IBM 7090/7094 IBSYS Operating System: System Monitor (IBSYS)	C28-6248-2	2-66
2.3.5.13	IBM 7090/7094 IBSYS Operating System: Operator's Guide	C28-6355	2-67
2.3.5.14	IBM 7090/7094 IBSYS Operating System: Symbolic Update Program--Preliminary Specifications	C28-6386-3	2-67
2.3.5.15	IBM 7090/7094 FORTRAN IV Language: Input/Output without Explicit List and Format	C28-6377	2-67
2.3.6	INSTALLATION SUPPLIES		2-68
2.3.6.1	7094 Reference Card	X22-6691	2-68
2.3.6.2	COBOL Program Sheet	X28-1464	2-68
2.3.6.3	COBOL Reference Card	X28-1520	2-68
2.3.6.4	IBM 7040/44-7090/94 Symbolic Language-Coding Sheet	X28-6333	2-68
2.3.6.5	FORTRAN Coding Form	X28-7327	2-68

2.3.1 IBM 7090/7094 SYSTEMS REFERENCE LIBRARY

2.3.1.1 7094 Data Processing System Configurator Form A22-6689

This is a schematic drawing depicting the overall 7094 Data Processing System. Whenever new features are incorporated for the 7094 system, these changes will be reflected in the drawing via publication change notices.

2.3.1.2 IBM 7094 Model II Configurator Form A22-6764

This is a schematic drawing depicting the overall 7094 Model II Data Processing System. Whenever new features are incorporated for the 7094-II system, these changes will be reflected in the drawing via publication change notices.

2.3.2 MACHINE SYSTEM

2.3.2.1 7094 Data Processing System--Principles of Operation Form A22-6703

This manual explains in detail the computer instructions, commands, and orders required for the operation of the 7094 system. In addition, it provides information about the units associated with the 7094 system. It expounds upon the use of the IBM 1301 Disk Storage, the IBM 1414-6 Input/Output Synchronizer, and the IBM 7340 Hypertape Drive; discusses the IBM 7909 Data Channel Interrupt features; and presents operating techniques for the Data Channel and Operator consoles.

2.3.2.2 IBM 7094 Model II Data Processing System (Bulletin) Form A22-6760

This bulletin describes the main features of the 7094 Model II Data processing System. It contains a sample instruction sequence to illustrate the reduction of core storage cycles through extended sequence overlap operations. It also includes a listing of instruction cycle changes and a table of instructions that can be overlapped.

2.3.2.3 IBM 729, 7330, and 727 Magnetic Tape Units--Principles of Operation Form A22-6589

This manual presents a complete description on the use of the various magnetic tape units. It covers the procedures for tape unit load and unload, tape error recovery and handling of tape during system usage; principles of writing and reading coded data on magnetic tape; operation of keys and lights; organization of tape records and reels, tape labeling and tape library records; and use of equipment associated with magnetic tapes.

2.3.2.4 IBM 1301 and 1302 Disk Storage: Sequential Data Organization Form A22-6784

This manual presents a different approach to the storing and the retrieving of data on the IBM 1301 and 1302 Disk Storage. It explains how all data files in disk storage can use a common set of programs and techniques. Even though data are loaded sequentially, retrieval can be either in a random or sequential manner. An index, created as the data file is loaded, associates data identifiers with actual track addresses. Conversion is accomplished from data handling methods presently in use.

2.3.2.5 IBM 1301 and 1302 Disk Storage, Models 1 and 2, with the 7090, 7094, and 7094 Model II Data Processing Systems
Form A22-6785

This manual contains a general description of the IBM Disk Storage Units 1301 and 1302, Models 1 and 2, in their association with the IBM 7090, 7094, and 7094 II Data Processing Systems. The reader should be versed in these data processing systems.

2.3.3 PROGRAMMING SYSTEMS

2.3.3.1 Catalog of Programs for IBM Data Processing Systems-- KWIC Index Form C20-8090

This catalog is divided into four main sections: 1) This section tells how to order programs from the IBM Program Information Department and from the Program Distribution Center. 2) This section has an index in both Keyword-in-Content (KWIC) format and classification code format. 3 & 4) These sections provide abstracts describing each program available from the IBM Program Information Department and from the Program Distribution Center, respectively.

2.3.3.2 IBM 7090/7094 Programming Systems: FORTRAN II Assembly Program (FAP) Form C28-6235-3

This publication provides enough information to the programmer concerning the 7090/7094 FORTRAN II Assembly Program (FAP) so that it enables him to code in the FAP language. FAP is a machine-oriented symbolic language. The major part of the program can be written in either FORTRAN or FAP. When programming in FORTRAN, FAP subroutines are used when FORTRAN is unsuitable. When programming in FAP, FORTRAN subroutines are useful for certain computational and input/output operations. FAP and FORTRAN can be used with the IBM FORTRAN Monitor or the IBM 7090/7094 IBSYS System Monitor.

2.3.3.3 IBM 7090/7094 Programming Systems: Macro Assembly Program (MAP) Language Form C28-6311-3

This publication has three main parts: 1) This part provides detailed information on the 7090/7094 Macro Assembly Program (MAP) language. The MAP language provides the user with an extensive set of pseudo-operations along with all of the 7090 and 7094 machine operations. 2) This part describes the functions of the pseudo-operations and cites examples of their formats and uses. 3) This part describes macro operations and macro-related pseudo-operations and expounds upon their use in programs. Since the Macro Assembly Program (IBMAP) is a component of the 7090/7094 IBJOB Processor, it operates under the IBJOB Processor.

2.3.4 COBOL

2.3.4.1 COBOL-- General Information Manual Form F28-8053

This manual describes the COBOL language (Common Business Oriented Language) as developed under the auspices of the Department of Defense and other Federal Government agencies, the Conference of Data Systems Languages (CODASYL) and computer manufacturers.

2.3.4.2 IBM 7090/7094 Programming Systems: IBJOB Processor Part 5: COBOL Compiler (IBCBC) Form J28-6260

This bulletin describes in full those elements of the COBOL language that appear in the initial version of the 7090/7094 COBOL Compiler (IBCBC), a component of the IBJOB Processor. This is the fifth of several bulletins that comprise the IBJOB Processor manual.

2.3.5 FORTRAN

2.3.5.1 IBM 7090/7094 Programming Systems: FORTRAN II Programming Form C28-6054

This publication undertakes to tell users of what sources are available to translate FORTRAN II statements into machine language statements. This is accomplished by using either the FORTRAN II Processor operating under the System Monitor of the 7090/7094 IBSYS Operating System or the independent FORTRAN Monitor System. The IBM Formula Translating System, 7090/7094 FORTRAN, is an automatic coding system for the IBM 7090/7094 Data Processing System.

2.3.5.2 IBM 7090/7094 Programming Systems: FORTRAN II Operations Form C28-6066-6

The purpose of this publication is to provide instructions for the operation and use of the IBM 7090/7094 FORTRAN II System. It also tells how the FORTRAN II System tape is produced and maintained. Included in the FORTRAN System are the FAP (FORTRAN Assembly Program) Assembler, the FORTRAN II Monitor, and the FORTRAN II Compiler. The compilation, assembly, and execution of FORTRAN and FAP programs are coordinated by the FORTRAN II Monitor.

2.3.5.3 IBM 7090/7094 FORTRAN IV Compiler (IBFTC) Replacement: Specifications and Language Additions Form C28-6376

This publication provides enough information for the programmer to enable him to make the transition from the present FORTRAN IV Compiler to the new 7090/7094 FORTRAN IV Compiler (IBFTC). The new compiler will operate practically within the same manner in respect to the present IBJOB Processor. Four new language features are provided for the FORTRAN IV language. They include 1) up to seven dimensions for arrays, 2) nonstandard returns from subroutines, 3) multiple entry points to a subprogram, and 4) input/output without an explicit input/output list and format statement.

2.3.5.4 FORTRAN Form F28-8074-3

This manual presents a description of FORTRAN. FORTRAN is a high-level problem-oriented computer language that is available for most IBM Data Processing Systems.

2.3.5.5 IBM 7090/7094 Programming Systems: FORTRAN IV Language
Form C28-6274-4

This publication undertakes to describe the FORTRAN IV language and provides a description of the format and effect of arithmetic, control, input/output, and specification statements. It also cites examples as to how these statements can be used, as well as describing methods of using available mathematical subroutines. The FORTRAN IV Compiler, a component of the IBJOB Processor, processes the FORTRAN IV language.

2.3.5.6 IBM 7090/7094 IBSYS Operating System: IBJOB Processor
Form C28-6275-4

This two-part publication presents a comprehensive description of the 7090/7094 IBJOB Processor. It also discusses the various components associated with the IBJOB Processor. Part one contains information for the applications programmer. Part two is for the systems programmer. The IBJOB Processor, a group of programs used to translate programming languages, consists of: The FORTRAN IV (IBFTC) and the COBOL (IBCBC) compilers; the Loader (IBLDR); the IBJOB Debugging Processor (IBLIB); and if applicable, the Subroutine Library (IBLIB). The latter is a library of preassembled subroutines that can be used by the object program.

2.3.5.7 IBM 7090/7094 IBSYS Operating System: Specifications for IBJOB Processor Debugging Package
Form C28-6362-1

This publication describes the debugging packages available for COBOL programs at compilation time and for FORTRAN IV and MAP programs at load time. The IBJOB Processor Debugging Package is an aid to programmers in that it obtains dynamic dumps of specified areas during program execution.

2.3.5.8 7090/7094 PROGRAMMING SYSTEMS: IBJOB Processor, Overlay Feature of IBLDR
Form C28-6331

This publication provides enough information to the experienced programmer to enable him to use the Overlay Feature of the Loader, IBLDR. With the Overlay Feature, it is possible to exceed the capacity of a single core storage load. Since IBLDR is a component of the IBJOB Processor, the reader should have prior knowledge of the IBJOB Processor.

2.3.5.9 IBM 7090/7094 IBSYS Operating System: Input/Output Control System
Form C28-6345-2

This publication undertakes to tell programmers of the 7090/7094

Input/Output Control System. There is a discussion of basic concepts, an explanation of the use of IOCS commands and routines, and information on the techniques of sequential processing as well as a separate section covering random processing. IOCS is responsible for making data readily available for processing by automatically controlling the transmission of data to and from recording devices. This publication further expounds upon the three forms of IOCS. The programmer has access to both the Library IOCS and the full IOCS (7090-IO-919). The former, as used with the Macro Assembly Program (MAP), is contained in the IBJOB Subroutine Library. The latter, as used with the IBM 7090/7094 FORTRAN II Assembly Program (IBSFAP), is an independent system. The differences in the two systems are noted. The third form, FORTRAN IOCS, is used by FORTRAN IV object programs.

2.3.5.10 IBM 7090/7094 IBSYS Operating System Utilities
Form C28-6364-3, with N28-0125-D

This publication undertakes to describe the following seven utility routines: 1) Restore Disk/Drum, 2) Load Disk/Drum, 3) Tape Dump, 4) Format-Track Generation, 5) Home-Address and Record-Address Generation, 6) Dump Disk/Drum, and 7) Clear Disk/Drum. These utility routines are available to users of the IBM 7090/7094 IBSYS Operating System equipped with IBM 729 Magnetic Tape, 1301 Disk Storage, 7320 Drum Storage, or the 7340 Hypertape Units. There is also a description of the Utility Monitor.

2.3.5.11 IBM 7090/7094 Generalized Sorting System: 7090/7094 Sort
Form C28-6307

This publication presents a description of the 7090/7094 Generalized Sorting System as it operates under the System Monitor. It explains how the sort program sorts fixed-length or variable-length records by using either the commercial or scientific collating sequences. These records are written in either signed or unsigned binary or BCD mode. Part one of this publication discusses the organization and structure of the sort program. A description of the sorting and merging techniques is provided. Part two describes in detail the program usage. It covers such areas as tape record format and file structure, control card formats, general specifications, and user modification procedures.

2.3.5.12 IBM 7090/7094 IBSYS Operating System: System Monitor (IBSYS)
Form C28-6248-2

This publication covers the role of the System Monitor in its overall control and direction of advanced business and scientific programming aids of the IBSYS Operating System. With practically no human intervention, the System Monitor is able to process a variety of unrelated

jobs sequentially. Of particular interest to the applications programmer are the introduction and sections describing the System Supervisor and the System Core-Storage Dump Program. The systems programmer should direct his attention to sections dealing with the System Nucleus, the System Editor, and the Input/Output Executor.

2.3.5.13 IBM 7090/7094 IBSYS Operating System: Operator's Guide
Form C28-6355

This publication provides enough information to the operator and machine room personnel to enable them to operate the 7090/7094 IBSYS Operating System. There are descriptions along with explanations of System Monitor control cards, system halts, initial start procedures, and the on-line messages.

2.3.5.14 IBM 7090/7094 IBSYS Operating System: Symbolic Update Program--Preliminary Specifications
Form C28-6386-3

This publication describes how the Symbolic Update Program provides the IBSYS user with the means to modify serialized symbolic tapes, as well as those tapes for the operating system itself. It defines the requirements for the Program's implementation and pseudo-instructions.

2.3.5.15 IBM 7090/7094 FORTRAN IV Language: Input/Output without Explicit List and Format
Form C28-6377

This publication provides enough information to the programmer to make him knowledgeable of the FORTRAN IV language addition scheduled for implementation in a future version of the FORTRAN IV Compiler. Discussed is the input/output and conversion system without an explicit input/output list and a FORMAT statement.

2.3.6 INSTALLATION SUPPLIES

2.3.6.1 7094 Reference Card Form X22-6691

All 7094 instructions in both alphabetic and numeric sequence are listed in the 7094 Reference Card. The card also provides data channel commands, and disk storage and Hypertape orders.

2.3.6.2 COBOL Program Sheet Form X28-1464

This sheet has provisions for the four divisions of the COBOL program; namely, Identification, Environment, Data and Procedure. See "COBOL General Information Manual," Form F28-8053.

2.3.6.3 COBOL Reference Card Form X28-1520

This card provides the COBOL programmer with material that he frequently uses. It is a handy reference card showing a COBOL word list and a basic COBOL format along with tables concerning special characters used in COBOL; arithmetic and conditional expressions (sequence of symbols); arithmetic and relational operators; and data-items pictorial characters.

2.3.6.4 IBM 7040/44-7090/94 Symbolic Language-Coding Sheet Form X28-6333

This form is used for coding any of the symbolic languages accepted by the assembly programs in the 7090/7094 Programming Systems and the 7040/7044 Programming Systems. For ease in both programming and card punching, columns and lines are ruled and numbered.

2.3.6.5 FORTRAN Coding Form Form X28-7327

When programming in the FORTRAN language, the FORTRAN Coding Form is used. For ease of both programming and card punching, columns and lines are ruled and numbered.

Table 2-4. 7094 FORTRAN Documentation Form-Number Index

FORM NO.	TITLE	PARAGRAPH	PAGE NO.
A22-6589	IBM 729, 7330, and 727 Magnetic Tape Units--Principles of Operation	2.3.2.3	2-60
A22-6689	7094 Data Processing System Configurator	2.3.1.1	2-59
A22-6703	7094 Data Processing System	2.3.2.1	2-60
A22-6760	7094 Model II Data Processing System (Bulletin)	2.3.2.2	2-60
A22-6764	IBM 7094 Model II Configurator	2.3.1.2	2-59
A22-6784	IBM 1301 and 1302 Disk Storage: Sequential Data Organization	2.3.2.4	2-60
A22-6785	IBM 1301 and 1302 Disk Storage, Models 1 and 2, with 7090, 7094, and 7094 Model II Data Processing Systems	2.3.2.5	2-61
C20-8090	Catalog of Programs for IBM Data Processing Systems--KWIC Index	2.3.3.1	2-62
C28-6235-3	IBM 7090/7094 Programming Systems: FORTRAN II Assembly Program (FAP)	2.3.3.2	2-62
C28-6054	IBM 7090/7094 Programming Systems: FORTRAN II Programming	2.3.5.1	2-64
C28-6066-6	IBM 7090/7094 Programming Systems: FORTRAN II Operations	2.3.5.2	2-64
C28-6248-2	IBM 7090/7094 IBSYS Operating System: System Monitor (IBSYS)	2.3.5.12	2-66
C28-6274-4	IBM 7090/7094 Programming Systems: FORTRAN IV Language	2.3.5.5	2-65
C28-6275-4	IBM 7090/7094 IBSYS Operating System: IBJOB Processor	2.3.5.6	2-65
C28-6307	IBM 7090/7094 Generalized Sorting System: 7090/7094 Sort	2.3.5.11	2-66
C28-6311-3	IBM 7090/7094 Programming Systems: Macro Assembly Program (MAP) Language	2.3.3.3	2-62
C28-6331	7090/7094 Programming Systems: IBJOB Processor, Overlay Feature of IBLDR	2.3.5.8	2-65
C28-6345-2	IBM 7090/7094 IBSYS Operating System: Input/Output Control System	2.3.5.9	2-65
C28-6355	IBM 7090/7094 IBSYS Operating System: Operator's Guide	2.3.5.13	2-67
C28-6362-1	IBM 7090/7094 IBSYS Operating System: Specifications for IBJOB Processor Debugging Package	2.3.5.7	2-65
C28-6364-3	IBM 7090/7094 IBSYS Operating System Utilities	2.3.5.10	2-66

Table 2-4. 7094 FORTRAN Documentation Form-Number Index (Cont'd)

FORM NO.	TITLE	PARAGRAPH	PAGE NO.
C28-6376	IBM 7090/7094 FORTRAN IV Compiler (IBFTC) Replacement: Specifications and Language Additions	2.3.5.3	2-64
C28-6377	IBM 7090/7094 FORTRAN IV Language: Input/ Output without Explicit List and Format	2.3.5.15	2-67
C28-6386-3	IBM 7090/7094 IBSYS Operating System: Symbolic Update Program--Preliminary Specifications	2.3.5.14	2-67
F28-8053	COBOL--General Information Manual	2.3.4	2-63
F28-8074-3	FORTRAN	2.3.5.4	2-64
J28-6260	IBM 7090/7094 Programming Systems: IBJOB Processor Part 5: COBOL Compiler (IBCBC)	2.3.4.2	2-63
X22-6691	7094 Reference Card	2.3.6.1	2-68
X28-1464	COBOL Program Sheet	2.3.6.2	2-68
X28-1520	COBOL Reference Card	2.3.6.3	2-68
X28-6333	IBM 7040/44-7090/94 Symbolic Language-Coding Sheet	2.3.6.4	2-68
X28-7327	FORTRAN Coding Form	2.3.6.5	2-68

2.4 SYSTEM TAPE CONTENTS

This section describes in brief the characterization of the systems contained on the combined IBSYS/FMS master tape employed on the 7094 A, B, C and E computers.

2.4.1 FORTRAN MONITOR SYSTEM (FMS)

The FORTRAN Monitor System is a collection of programs that enable a programmer to compile a FORTRAN II source language program, assemble a FAP language program, and execute an object program. In addition, the FMS system contains a library of input/output and exponential subroutines utilized by the FORTRAN compiler and a library of mathematical subroutines. The processor provides the capability of combining programs written in FORTRAN II and FAP languages with previously assembled program segments to form a single executable object program. Facilities are available for changing core storage loads so that executed portions of a program can be overlaid with program portions yet to be executed. Each of the subprograms of the system are briefly described in 2.4.1.1, 2.4.1.2, and 2.4.1.3.

2.4.1.1 FORTRAN II Compiler

The FORTRAN II Compiler translates programs written in the FORTRAN II language and produces input to the assembler. The assembler processes the input, and the resulting binary object program may be loaded for execution. The object program, which is a result of compiling, assembling, and loading, consists of the generated instructions and the subroutines from the subroutine library.

Additional information on the FORTRAN II language and operation is contained in the following publications: IBM FORTRAN II General Information Manual, Form C28-8074 and IBM 7090/7094 Programming Systems: FORTRAN II Operations, Form C28-6066.

2.4.1.2 FORTRAN II Assembly Program (FAP)

The FORTRAN Assembly Program processes FAP language programs and produces binary formalized object programs. The assembled object program may then be loaded for execution. A detailed description of the assembler is contained in the IBM 7090/7094 Programming Systems: FORTRAN II Operations, Form C28-6060; FORTRAN II Assembly Programs, Form C28-6235; and in the 7094 Data Processing System-Principles of Operation, Form A22-6703.

2.4.1.3 Binary Symbolic Subroutine Loader (BSS)

The Binary Symbolic Subroutine Loader is punched out by FORTRAN as the first nine cards of each main program. Additionally, a program card is punched out for each main program deck and for each subprogram deck. The program card specifies the number of locations to be occupied by the routine; this number is used as an increment for relocating an immediately subsequent routine. The loader uses the information contained on the program cards to allocate core storage for the main program routine and any associated subprogram routine. Additional information on the BSS loader is contained in the IBM 7090/7094 Programming System: FORTRAN II Operations, Form C28-6060.

2.4.1.4 FORTRAN II Library

The FORTRAN II library is a collection of subroutines in relocatable binary form. It consists of the input/output and exponential subroutines utilized by the FORTRAN compiler and the FORTRAN library mathematical subroutines. The entry point names, card labels, and descriptions of the subroutines contained in the library follows.

(1) Special DSD Library Subroutines: A brief description of additional routines which are unique to the DSD system library is as follows:

a. Mathematical

Name	Source	Description
ASIND	Y = ASIN(A)	Computes the principal value of the arc sine in degrees. Output is a normalized floating-point number in the AC.
ACOSD	Y = ACOS(A)	Computes the principal value of the arc cosine in degrees. Output is a normalized floating-point number in the AC.
ATANGD	Y = ATANGD(A,B)	Computes the properly quadranted arc tangent in degrees of the quotient of the two signal inputs. Output is a normalized floating-point number in the AC.

(1) Special DSD Library Subroutines (Cont'd)

a. Mathematical (Cont'd)

Name	Source	Description
ACOSR	Y = ACOSR(A)	Computes the arc cosine in radians. Output is a normalized floating-point number in the AC.
ASINR	Y = ASINR(A)	Computes the arc sine in radians. Output is a normalized floating-point number in the AC.
ATANGR	Y = ATANGR(A,B)	Computes the arc tangent in radians. Output is a normalized floating-point number in the AC.
SIND	Y = SIND(A)	Computes the sine of an angle A expressed in degrees. Output is a normalized floating-point number in the AC.
COSD	Y = COSD(A)	Computes the cosine of an angle A expressed in degrees. Output is a normalized floating-point number in the AC.

b. UMPLOT

Name	Source	Description
UMPLOT	See Paragraph 2.5.1.1	Rapid plotting of numerical information for use with FORTRAN calling programs. The resulting graph is copied onto any decimal output tape for off-line printing. (See Paragraph 2.5.1.1 for complete description.)

(1) Special DSD Library Subroutines (Cont'd)

c. CalComp 570 Plotter Routines

Name	Source	Description
CCPLOT	CALL CCPLOT (X, Y, IC)	CCPLOT is analogous to the current PLOT routine. It is used to perform pen movements on the CalComp 570 Plotter. (See Paragraph 2.5.1.4 for detailed instructions in its use.)
CPLOTS	CALL CPLOTS (BUFFER, IDT, INDIC8)	CPLOTS is analogous to the current PLOTS routine. It is used to set up a tape buffering area. (See Paragraph 2.5.1.5 for detailed instructions in its use.)
SYMBOL	CALL SYMBOL (X, Y, HEIGHT, BCD, THETA, N)	SYMBOL is analogous to the current SYMBL4 routine. It is used to generate symbols and alphameric characters for the Cal Comp 570 plotter. (See Paragraph 2.5.1.6 for detailed instruction in its use.)

d. Utility Routines

Name	Source	Description
CLOCKS	CALL MIN ON(I) CALL OFF MIN CALL MINITS(M) CALL MSEC ON(X) CALL OFF MS CALL MSECS(X)	This is a subroutine to operate the millisecond and/or minute trap subchannels on the DCC on channel F. For each clock, there are entry points to turn the clock on, turn it off, and to read out the current value of elapsed time. (See DSD Memorandum dated 11/14/63 by R. Danek for additional information.)

(1) Special DSD Library Subroutines (Cont'd)

d. Utility Routines (Cont'd)

Name	Source	Description
DUMP	CALL DUMP (A,B,C) CALL PDUMP (A,B,C)	Dumps core according to the specification in the arguments A, B and C. DUMP initiates next job while PDUMP returns control to calling program. (See Paragraph 2.5.1.2 for description of modified 65K dump routine.)

(2) Mathematical Library Subroutines:

a. Single-Precision Subroutines

Name	Source	Description
EXP(1	I**J	Exponential expression--given a fixed-point base and a fixed-point exponent. Output is a fixed-point number in the AC.
EXP(2	A**J	Exponential expression--given a floating-point base and a fixed-point exponent. Output is a normalized floating-point number in the AC.
EXP(3	A**B	Exponential expression--given a floating-point base and a floating-point exponent. Output is a normalized floating-point number in the AC.
EXP	Y = EXPF(A)	Computes e^A, the natural antilogarithm of the number A. Output is a normalized floating-point number in the AC.

(2) Mathematical Library Subroutines (Cont'd)

a. Single-Precision Subroutines (Cont'd)

Name	Source	Description
SQRT	Y = SQRTF(A)	Computes the positive square root for a single-precision, real number A. Output is a normalized floating-point number in the A.C.
SIN,COS	Y = SINF(A) Y = COSF(A)	Computes the sine or cosine of an angle A expressed in radians. Output is a normalized floating-point number in the AC.
ATAN	Y = ATANF(A)	Computes the arctangent in radians of the argument A. Output is a normalized floating-point number in the AC.
LOG, LOG10	Y = LOGF(A) Y = LOG10F(A)	Computes the natural logarithm or the common logarithm of the number A. Output is a normalized floating-point number in the AC.
TANH	Y = TANHF(A)	Computes the hyperbolic tangent of the argument A. Output is a normalized floating-point number in the AC.

b. Double-Precision Subroutine

Double-Precision arithmetic is a technique for carrying out floating-point calculations with twice the normal number of significant decimal places. Only single-precision floating-point numbers may be input/output; output data may be more accurate as a result of using double-precision operations internally. The standard FORTRAN II library has been modified by the addition of double-precision subroutines that utilize the 7094 double-precision hardware. These routines may be employed by placing an E in column 1 as described herein. When the E double-precision routines are specified, the I/O subroutines (ESL1) and (ESL0) provide control for

(2) Mathematical Library Subroutines (Cont'd)

b. Double-Precision Subroutine (Cont'd)

the input and output of lists containing nonsubscripted array names. For additional information, see the Programming Methods Section Memorandum dated 5/18/65.

Name	Source	Description
DMOD EMOD	D Y = MODF(D,E) E Y = MODF(D,F)	Computes D modulo E (defined as D (D/E) * E) where only the integer portion of (D/E) is used in evolving the equation.
DINT	D Y = INTF(D)	Obtains the integer part of a double-precision number.
DEXP(2 EEXP(2	D Y = D**E E Y = D**E	Exponential expression for double-precision.
DLOG DLOG10 ELOG ELOG10	D Y = LOGF(D) D Y = LOG10F(D) E Y = LOGF(D) E Y = LOG10F(D)	Computes the natural logarithm or the common logarithm of the argument D.
DSIN DCOS ESIN ECOS	D Y = SINF(D) D Y = COSF(D) E Y = SINF(D) E Y = COSF(D)	Computes the sine or cosine of an angle expressed in radians.
DATAN DATAN2 EATAN EATAN2	D Y = ATANF(D) D Y = ATAN2F(D,E) E Y = ATANF(D) E Y = ATAN2F(D,E)	Computes the arctangent, in radians of one or two arguments.
DFAD DFSB DFMP DFDP	D A = B + C D A = B - C D A = B * C D A = B / C	Performs the basic double-precision arithmetic operations (addition, subtraction, multiplication, and division).
DEXP(3 EEXP(3	D Y = D**E E Y = D**E	One of two routines is compiled depending on whether a floating-point base, fixed-point exponent, or a floating-point exponent is specified.

(2) Mathematical Library Subroutines (Cont'd)

b. Double-Precision Subroutine (Cont'd)

Name	Source	Description
ETANH	E Y = TANHF(D)	Computes in double-precision floating-point the hyperbolic arctangent of the argument D.
DSQRT ESQRT	D Y = SQRTF(D) E Y = SQRTF(D)	Computes in double-precision floating-point the positive square root of the value D.
DEXP EEXP	D Y = D**E E Y = D**E	Exponential subroutine for double-precision numbers.
EMAXI	E Y = MAXIF (A,B,C...)	Selects the largest of the specified arguments.
EMINI	E Y = MINIF (A,B,C...)	Selects the smallest of the specified arguments.
ESIGN	E Y = SIGNF(A,B)	Changes the sign of the second argument B to the sign of the first argument A.

c. Complex Arithmetic

Complex Arithmetic is a technique for carrying out floating-point calculations with the real and imaginary parts of complex numbers. No provision is made for the input/output of complex numbers; however, since each part is requested internally as a separate single-precision floating-point number, each part may be input/output separately.

Name	Source	Description
IABS	I A = ABSF(C)	Computes the absolute value of the argument C, where $C^2 = X^2 + Y^2$ for the complex number X + iY.

(2) Mathematical Library Subroutines (Cont'd)

c. Complex Arithmetic (Cont'd)

Name	Source	Description
IEXP	I A = EXPF(C)	Computes the natural antilogarithm of the argument C; if C = X + iY then IEXP computes e^C
ILOG	I A = LOGF(C)	Computes the natural logarithm of the argument C; if C = X + iY then ILOG computes $\log_e C$.
ISQRT	I A = SQRTF(C)	Computes the principal square root of the argument C; if C = X + iY then ISQRT computes $C^{1/2}$.
ISIN ICOS	I A = SINF(C) I A = COSF(C)	Computes the sine or cosine of the argument C.
IFMP IFDP	I C = A*B I C = A/B	Performs complex multiplication and complex division, respectively.
IEXP(2	I Y = D**E	Exponential expression for complex numbers.

(3) FORTRAN II I/O: The FORTRAN input/output library contains the necessary routines to insure the correct operation of source language I/O statements. A brief characterization of each routine on the DSD library is given below.

Name	Card Label	Description
(STH), (STHM), (STHD)	9STH	Writes BCD tape record(s) from storage.
(SCH)	9SCH	Punches alphameric card(s) from storage.
(SPH)	9SPH	Prints information from storage on the on-line printer.

(3) FORTRAN II I/O (Cont'd)

Name	Card Label	Description
(BST)	9BST	Backspaces the designated tape one record.
(EFT)	9EFT	Writes an EOF mark on the designated tape.
(RWT)	9RWT	Rewinds the designated tape.
(SLI)	9SLI	Controls input of lists containing nonsubscripted array names.
(SLO)	9SLO	Controls output of lists containing nonsubscripted array names.
(RER), (RDC)	9RER	Error routines for tape reading.
(WER), (WTC)	9WER	Error routines for tape writing.
(IOB), (EXB), (BUF), (SET)	9IOB	Controls I/O of binary data.
(SRD), (DRS)	9DRM	Writes or reads data on the designated drum. (SRD) is for writing and (DRS) is for reading.
(IOU)	9IOU	DSU channel-unit table. Local modifications have been made to rearrange logical unit definitions.
(IOH), (FIL), (RTN)	9IOH	Controls input/output and conversion of alphameric data.
(IOS), (RDS), (WRS), (BSR), (WEF), (REW), (ETT), (RCH), (TEF), (TCO), (TRC)	9IOS	Supervisory control of channel-unit designation during input/output.
(TSB), (RLR)	9TSB	Reads binary tape record(s) into storage.

(3) FORTRAN II I/O (Cont'd)

Name	Card Label	Description
(STB), (WLR)	9STB	Writes binary tape record(s) from storage.
(CSH)	9CSH	Reads alphameric card(s) into storage and converts their contents to BCD.
(TSH), (TSHM)	9TSH	Reads BCD tape record(s) into storage.

(4) FORTRAN II Utility Subroutines:

Name	Card Label	Description
XLOC	9XLO	The source statement, L = XLOCF(N), returns the relocated location of its argument (variable N) to the accumulator as a FORTRAN fixed-point constant.
(EXE)	9EXE	Controls the object program error procedure when execution is not under Monitor control. Modified so traps are disabled.
EXIT	9XIT	Positions the system tape at the sign-out record and restores 1-CS to begin the next job. Routine is modified to permit call-out of multiple tag mode and traps disabled.
(EXEM)	9EXEM	Controls the object program error procedure when execution is under Monitor control. Routine is modified to permit call-out of multiple tag mode and traps disabled.
(TES)	9TES	The instruction XEC * $ (TES) may be used in a FAP-coded subroutine to make sure that the execution of any previous FORTRAN WRITE statement is complete and checked.
CHAIN	9CHN	Locates chain links, loads the chain executive loader into lower storage, and transfers control to it.

2.4.2 SYSTEM MONITOR (IBSYS)

The System Monitor provides the execution, control, and coordination that enables a series of unrelated jobs to be processed with little or no operator intervention. Operating under the control of the System Monitor are several subsystems that provide the programmer with a variety of programming tools. The capabilities of these subsystems may be used singly or in combination to process a particular job. The System Monitor consists of the:

System Supervisor--This controls and coordinates the processing of jobs.

System Nucleus--This provides facilities for intercommunications among the subsystems.

Input/Output Executor--This coordinates and controls input/output.

Core-Storage Dump Program--This facilitates testing and analysis of programs.

System Editor--This is used in modifying and maintaining the System Monitor and subsystems.

2.4.2.1 IBJOB Processor

The IBJOB Processor consists of a group of programs used to translate programming languages and to permit the loading and execution of the compiled and assembled programs. Contained within the IBJOB Processor are the following programs:

Processor Monitor (IBJOB)

FORTRAN IV Compiler (IBFTC)

COBOL Compiler (IBCBC)

Macro Assembly Program (IBMAP)

Relocating Loader (IBLDR)

Subroutine Library (IBLIB)

Dubugging Package (IBDBL & IBDBC)

The Processor Monitor reads control cards that specify the action to be performed. The action can consist of one or more compilations, assemblies, or the loading of relocatable programs that were assembled previously. In this way, the monitor controls the interaction among the several IBJOB Processor components. These include:

(1) The FORTRAN IV Compiler (IBFTC): The IBFTC compiler translates programs written in the FORTRAN IV source language and produces input to the assembler. The assembler and, if required, the Loader process the input. The processing and loading to be performed by the Loader is specified on the $IBJOB control card via the functions called GO, LOGIC, DLOGIC, or MAP (see Section 2.2). The object program is composed of generated instructions and subroutines from the subroutine Library as a result of compiling, assembling, and loading. The Processor Monitor calls the FORTRAN IV compiler into core storage when it reads an $IBFTC control card (see Section 2.2).

(2) The Macro Assembly Program (IBMAP): The Macro Assembly Program processes two types of programs: Those written in the MAP language and those generated MAP programs that are output from FORTRAN IV and COBOL compiler. There may be two types of outputs from the assembler: relocatable or absolute binary. The output from the assembling and loading functions is an object program composed of machine instructions generated by the assembler and coinciding with the MAP mnemonics. In addition the object program may contain input/output routines that are part of the subroutine Library and possibly FORTRAN IV mathematical subroutines from the subroutine Library. The assembler is called into operation when the Processor Monitor reads an $IBMAP control card (see Section 2.2).

(3) The Subroutine Library (IBLIB): The subroutine Library consists of a number of relocatable subroutines for system and programmer use. It is composed of system subroutines and FORTRAN IV subroutines. (COBOL subroutines have been removed from Data Systems Division systems.) The FORTRAN section is divided into three groups: mathematics, input/output, and utility. These subroutines are made available to the programmer via the Loader, which incorporates them, as required, into the object program at load time.

1. Special Library Subroutines

A brief description of additional Library subroutines which are unique to the Data Systems Division system Library is as follows:

a. Mathematical

Name	Source	Description
ASIND	Y = ASIN(A)	Computes the principal value of the arc sine in degrees. Output is a normalized floating-point number in the AC.
ACOSD	Y = ACOS(A)	Computes the principal value of the arc cosine in degrees. Output is a normalized floating-point number in the AC.
ATANGD	Y = ATANGD(A,B)	Computes the properly quandranted arctangent in degrees of the quotient of the two signal inputs. Output is a normalized floating-point number in the AC.
ACOSR	Y = ACOSR(A)	Computes the arc cosine in radians. Output is a normalized floating-point number in the AC.
ASINR	Y = ASINR(A)	Computes the arc sine in radians. Output is a normalized floating-point number in the AC.
ATANGR	Y = ATANGR(A,B)	Computes the arctangent in radians. Output is a normalized floating-point number in the AC.
SIND	Y = SIND(A)	Computes the sine of an angle A expressed in degrees. Output is a normalized floating-point number in the AC.
COSD	Y = COSD(A)	Computes the cosine of an angle A expressed in degrees. Output is a normalized floating-point number in the AC.

1. Special Library Subroutines (Cont'd)

b. UMPLOT

Name	Source	Description
UMPLOT	See Section 2.5	Rapid plotting of numerical information for use with FORTRAN calling programs. The resulting graph is copied onto any decimal output tape for off-line printing. (See Section 2.5 for complete description.)

c. Utility Routines

Name	Map Call	Description
FDMP	CALL DUMP(A,B,I) CALL PDUMP(A,B,I) CALL FPDUMP(A,B,I)	Modified 65K Core Dump Routine. (See Section 2.2 for additional information.)

2. Mathematical Library Subroutines

a. Single-Precision Subroutines

Name	Source	Map Call	Description
FXP1	I**J	CALL .XP1.(I,J)	Exponential expression--given a fixed-point base and a fixed-point exponent. Output is a fixed-point number in AC.
FXP2	A**J	CALL .XP2.(A,J)	Exponential expression--given a floating-point base and a fixed-point exponent. Output is a normalized floating-point number in AC.

2. Mathematical Library Subroutines (Cont'd)

a. Single-Precision Subroutines (Cont'd)

Name	Source	Map Call	Description
FXPF	Y = EXP(A)	CALL EXP(A)	Computes e^A, the natural antilogarithm of the number A. Output is a normalized floating-point number in AC.
FLOG	Y = ALOG(A) Y = ALOG10(A)	CALL ALOG(A) CALL ALOG10(A)	Computes the natural logarithm or the common logarithm of the number A. Output is a normalized floating-point number in AC.
FATN	Y = ATAN(A) Y = ATAN2(A)	CALL ATAN(A) CALL ATAN2(A)	Computes the arctangent, in radians, of the arguments A or A/B, respectively. Output is a normalized floating-point number in AC.
FSCN	Y = SIN(A) Y = COS(A)	CALL SIN(A) CALL COS(A)	Computes the sine or cosine of an angle A expressed in radians. Output is a normalized floating-point number in AC.
FTNH	Y = TANH(A)	CALL TANH(A)	Computes the hyperbolic tangent of the argument A. Output is a normalized floating-point number in AC.
FSQR	Y = SQRT(A)	CALL SQRT(A)	Computes the positive square root of the number A. Output is a normalized floating-point number in AC.

2. Mathematical Library Subroutines (Cont'd)

b. Double-Precision Subroutines

Name	Source	Map Call	Description
FDMD	Y = DMOD(D,E)	CALL DMOD(D,E)	Computes D modulo E (defined as D-(D/E)*E, where only the interger portion of (D/E)is used in evaluating the equation). Output is a normalized double-precision floating-point number in the AC and MQ.
FDX1	1) D**I 2) C**I	CALL .DXP1.(D,I) CALL .CXP1.(C,I)	1) Exponential expression--given a double-precision floating-point base and a fixed-point exponent. Output is a normalized double-precision floating-point number in the AC and MQ. 2) Exponential expression--given a complex base and a fixed-point exponent. Output is a complex number with the real portion in the AC and the imaginary portion in the MQ.
FDX2	1) D**A 2) D**E	CALL .DXP2.(D,A) CALL .DXP2.(D,E)	1) Exponential expression--given a double-precision floating-point base and a single-precision floating-point exponent. Output is a double-precision normalized floating-point number in the AC and MQ.

2. Mathematical Library Subroutines (Cont'd)

b. Double-Precision Subroutines (Cont'd)

Name	Source	Map Call	Description
			2) Exponential expression--given a double-precision floating-point base and a double-precision floating-point exponent. Output is a normalized double-precision floating-point number in the AC and MQ.
FDXP	Y = DEXP(D)	CALL DEXP(D)	Computes the natural antilogarithm of the double-precision number D. Output is a normalized double-precision floating-point number in the AC and MQ.
FDLG	Y = DLOG(D) Y = DLOG10(D)	CALL DLOG(D) CALL DLOG10(D)	Computes the natural logarithm or common logarithm of the double-precision argument D. Output is a normalized double-precision floating-point number in the AC and MQ.
FDSQ	Y = DSRQT(D)	CALL DSRQT(D)	Computes the positive square root of the double-precision argument D. Output is a normalized double-precision floating-point number in the AC and MQ.
FDSC	Y = DSIN(D)	CALL DSIN(D)	Computes the sine or cosine of an angle expressed in radians. Output is a normalized double-precision floating-point number in the AC and MQ.

2. Mathematical Library Subroutines (Cont'd)

b. Double-Precision Subroutines (Cont'd)

Name	Source	Map Call	Description
FDAT	Y = DATAN(D) Y = DATAN2(D,E)	CALL DATAN(D) CALL DATAN2(D,E)	Computes the arctangent in radians, of one or two double-precision arguments. Output is a normalized double-precision floating-point number in the AC and MQ.

c. Complex Subroutines

Name	Source	Map Call	Description
FCAS	C*F C/F	CALL .CFMP.(C,F) CALL .CFDP.(C,F)	Performs complex multiplication and complex division, respectively. Output is a complex number in the AC and MQ.
FCAB	Y = CABS(C)	CALL CABS(C)	Computes the absolute value of the argument C. Output is a complex number in the AC and MQ.
FCXP	Y = CEXP(C)	CALL CEXP(C)	Computes the natural antilogarithm of the argument C. Output is a complex number in the AC and MQ.
FCLG	Y = CLOG(C)	CALL CLOG(C)	Computes the natural logarithm of the argument C. Output is a complex number in the AC and MQ.

2. Mathematical Library Subroutines (Cont'd)

c. Complex Subroutines (Cont'd)

Name	Source	Map Call	Description
FCSQ	Y = FCSQ(C)	CALL CSQRT(C)	Computes the principal square root of the argument C. Output is a complex number in the AC and MQ.
FCSC	Y = CSIN(C) Y = CCOS(C)	CALL CSIN(C) CALL CCOS(C)	Computes the sine or cosine of the argument C. Output is a complex number in the AC and MQ.

3. Machine Test Indicator Subroutines

These routines are used to test indicators by CALL statements in the FORTRAN language.

Name	Map Call	Description
FSLITE	1) CALL SLITE(I) 2) CALL SLITE(I,J)	1) For I = 0 all sense lites are set off. If I = 1, 2, 3, 4 the corresponding sense lite is set on. 2) The sense lite I = 1, 2, 3, 4 is tested and set OFF. For sense lite ON the variable J is set to 1, for OFF conditions J is set to 2.
FSSWTH	CALL SSWTCH(I,J)	The sense switch I = 1, 2, 3, 4, 5, 6 is tested and if switch is DOWN the variable J is set to 1. For switch UP, J is set to 2.
FOVERF	CALL OVERFL(J)	A test is performed on overflow condition. When overflow exists the variable J is set to 1; no overflow results in J set to 2. Machine is left in non-overflow condition after execution.

3. Machine Test Indicator Subroutines (Cont'd)

Name	Map Call	Description
FDVCHK	CALL DVCHK(J)	The variable J is set to 1 when Divide Check Indicator is ON and J is set to 2 when DCI is OFF. The DCI is set OFF after test.

4. FORTRAN IV I/O

The FORTRAN input/output library contains the necessary routines to insure the correct operation of source language I/O Statements. A brief characterization of each routine on the DSD Library is given below.

Name	Description
FOUT	Writes blocked records on the system output unit.
FRWT	Rewinds designated unit.
FSLDI	Controls processing of lists containing nonsubscripted BCD array names for input.
FSLBI	Controls processing of lists containing nonsubscripted binary array names for input.
FSLI	Sets up indexing for input of nonsubscripted arrays.
FSLDO	Controls processing of lists containing nonsubscripted BCD array names for input.
FSLBO	Controls processing of lists containing nonsubscripted array names for output.
FSLO	Sets up indexing for output of nonsubscripted arrays.
FVIO	Establishes identification between a variable logical unit and the corresponding FORTRAN file.
FRCD	Controls reading of cards on-line and conversion of alphameric card code to BCD.

4. FORTRAN IV I/O (Cont'd)

Name	Description
FEFT	Writes a file mark on the designated unit.
FBST	Backspaces the designated unit one record.
UNITXX	I/O routines used by loader for the initialization of IOCS files used by the object program.
FWRD	Controls BCD write operation.
FWRB	Controls binary write operation.
FRDD	Controls BCD read operation.
FRDB	Controls binary read operation.
FPRN	Controls print operation (on-line).
FPUN	Controls punch operation (off-line).
FCNV	Effects the necessary conversion for input or output list items.
FIOB	Processes list items for binary transmission.
FIOS	Initializes all I/O library IOCS calling sequences for binary and BCD transmission.
FIOH	Scans FORMAT Statements and links to the object program to begin conversion of data.
FSEL	Performs all IOCS selects.
FWRO	Controls processing of list of variables and arrays associated with BCD output.
FWRU	Controls writing of BCD records.
FRDU	Controls reading of BCD records.
FIOU	Controls processing of lists of variables and arrays associated with a NAMELIST name for BCD input.

5. System Subroutines

Given below is a brief characterization of the system subroutines contained on the DSD Library. The marked routines (*) are called automatically for each object program

Name	Description
.IBSYS	Defines the indexes of the system units.*
.IOEX	Defines the location of the I/O Executor entry points.*
.JBCON	Relocates Processor Monitor communication words to an area immediately above IOEX during object program execution.*
.LXCON	Normally entered at termination of object program execution; however, it is also entered at a system stop or on a STR instruction. Object program files are closed and all I/O activity is stopped.*
.RAND	Provides for processing of random records on 1301 disk storage.
.FPTRP	Floating-point trap subroutine; determines the cause of trap and write message on system output unit indicating cause of trap and the octal location at which it occurred.
.IODEF	Contains primary I/O system communication region. General entry and exit routines used by IOCS are contained in this subroutine.
.IOCSF	Initializes the communication region required by FORTRAN IOCS and contains text for the special IOCS.
.IOCS	Contains the text for all levels of relocatable IOCS.
.IOCSM	Initializes the communication region required by MINIMUM IOCS.
.IOCSB	Initializes the communication region required by BASIC IOCS.
.IOCSL	Initializes the communication region required by LABEL IOCS--modified to delete switch tests.

5. System Subroutines (Cont'd)

Name	Description
.LOVRY	Loads overlay links--required for all object programs using overlay feature.
.LXSL	Read/write select routine.
.IBDBI	Required for DEBUG requests. Interprets the request and executes the operation.
.DSTRN	Required for all nonoverlay DEBUG requests--searches a table for DEBUG request points.
.DSTRO	Used for overlay DEBUG requests--searches table for request points.

6. FORTRAN Utility Subroutines

Name	Description
.ERAS	Erasable words used by object program.
FPARST	Used by program SIFT to determine, for FORTRAN IV programs, address of desired part of double-precision or complex pair, as specified in FORTRAN II program.
FXEM	Controls object program error procedure--modified for diagnostics to logical 3 and to delete messages.
XIT	Returns control to subroutine .LXCON.

(4) The Loader (IBLDR): The Loader processes relocatable binary program decks generated by the assembler and combines any required subroutines from the subroutine Library to form one executable object program. It assigns absolute core storage locations to the relocatable binary text of the program and resolves cross-references. Additionally, it allocates core storage for pools of input/output buffers and attaches files to the buffer pools. The Loader performs these functions automatically; however, a programmer can modify the procedure by using control cards.

(5) The COBOL Compiler (IBCBC): THE IBCBC compiler translates programs written in COBOL language and produces input to the assembler. The assembler and, if required, the Loader process the input. The processing and loading to be performed by the Loader is specified on the $IBJOB control card via the function calls GO, LOGIC, DLOGIC, or MAP (see Paragraph 2.2.1.2(4)). The object program is composed of generated instructions and subroutines from the subroutine Library as a result of compiling, assembling, and loading. The Processor Monitor calls the COBOL compiler into core storage when it reads an $IBFTC control card (see Paragraph 2.2.1.2(5)).

2.4.2.2 The Commercial Translator Processor (CT)

This is available for compiling, assembling, loading, and executing programs written in the IBM Commercial Translator Language. A complete description of the CT is contained in the publication, IBM 709/7090 Commercial Translator Processor, Form J28-6169.

2.4.2.3 The 90PAC Processor (90PAC)

This is used to establish and maintain data files and to generate reports on the data in the files. Additional information on the 90PAC Processor is available in the following IBM publications:

7090 Programming System, Share 7090 90 PAC
Part 1 of Introduction and General Principles, Form J28-6166
Part 2 of The File Processor, Form J28-6167
Part 3 of The Report Generator, Form J28-6168

2.4.2.4 The Input/Output Control System (IOCS)

This is used by programs assembled by the FORTRAN II Processor. IOCS automatically controls the blocking and unblocking of data records; the overlapping of processing with input and output; and the preparation and checking of labels. The required portions of IOCS are loaded with the assembled object program, and this relieves the programmer of the task of writing complex I/O routines. The publication, IBM 7090/7094 IBSYS Operating System: Input/Output Control System, Form C28-6345, provides a detailed description of IOCS.

2.4.2.5 The IBSFAP

This mode of the FORTRAN processor can be used to assemble programs written in the FAP language. However, it may not be used to load and execute the assembled program. The system programmer has used

IBSFAP primarily for updating symbolic tapes by changing, deleting, or adding instructions. An IBSFAP assembled program can be loaded and executed under control of the IOCS system or the FORTRAN mode of the FORTRAN II Processor. Additional information on IBSFAP is available in the publication, IBM 7090/7094 Programming Systems: FORTRAN II Assembly Program (FAP), Form C28-6235.

2.4.2.6 The FORTRAN II Processor (Version III)

This operates under control of IBSYS and can be used to compile, assemble, load, and execute programs written in FORTRAN II language. This version of FORTRAN is available on the system tape; however, it is not currently maintained. Those programmers requiring the use of this version of FORTRAN II should contact the Programming Methods Section for additional information.

2.4.2.7 The Utilities (DK9OUT)

These utilities, available under the control of IBSYS monitor, consist of 1) a tape dump routine for 729 Magnetic Tape Units and 7340 Hypertape Drives and 2) several 1301 Disk Storage and 7320 Drum Storage routines. These routines consist of format track generation, home address and record address identification, load disk/drum, dump disk/drum, restore disk/drum, and clear disk/drum. A complete description of these routines and their use is contained in the publication, IBM 7090/7094 IBSYS Operating System Utilities, Form C28-6364.

2.4.2.8 The RESTART Program

This is used exclusively by the operator of the system. It is designed to enable the operator to restart an interrupted program using a checkpoint record recorded by IOCS before the interruption occured. A description of RESTART is contained in the publication, IBM 7090/7094 IBSYS Operating System: Operator's Guide, Form C28-6355.

2.5 UTILITY ROUTINES

This section describes those utility routines that have been prepared for use by Goddard Space Flight Center programmers. These routines are contained on C1 utility tape, in binary decks, and on system library.

2.5.1 FORTRAN II

2.5.1.1 UMPLOT Plotting Subroutine

(1) Purpose: The purpose of this subroutine is for rapid plotting of numeric information for use with FORTRAN Calling programs. The resulting graph is copied onto any decimal output tape for off-line printing.

(2) Method: The philosophy used in writing this subroutine was to treat a region of core storage (called the image) much as a piece of graph paper when plotting data manually.

First the image region is blanked out and a grid formed of I's and -'s (with + at the intersections) is placed in the image. Given the maximum and minimum values of the two variables, say X and Y, the routine can place any specified BCD character at the appropriate position in the image for a given pair of values (X_i, Y_i).

Each point (X_i, Y_i) is plotted individually and independently. A character falling on a grid line replaces the grid character in that position. A character falling on a previously plotted character will replace that character. Points falling outside the grid limits will be ignored. When all desired points have been plotted, the image is copied onto the specified decimal output tape for off-line printing.

(3) Use: The subroutine has four main entries which perform the following functions:

PLOT 1 sets up the grid spacing and the total width and length of the graph image. It determines the location of the decimal points and the multiplying factors (powers of 10) for values of the ordinate and the abscissa to be printed at the grid lines.

PLOT 2 prepares the grid, examines the maximum and minimum values of the coordinates, and establishes internally a formula for computing the location in the image corresponding to the point (X_i, Y_i).

PLOT 3 places a specified BCD character in the appropriate position(s) corresponding to the given (X_i, Y_i).

FPLOT 4 writes the image of the completed graph on the output tape for off-line printing. A label for the ordinate is printed vertically at the left edge of the page. Values of the abscissa and ordinate are printed at the grid lines outside the bottom and left edges of the graph.

(4) Calling Sequence: The calling sequences are as follows:

```
CALL PLOT1  (NSCALE,NHL,NSBH,NSBV)
CALL PLOT2  (IMAGE, XMAX,XMIN,YMAX,YMIN)
CALL PLOT3  (BCD, X,Y,NDATA)
CALL FPLOT4 (NCHAR,NHABCDEF ... )
```

1. Description of Arguments--The following is a description of the arguments:

a. NSCALE This is an array in the users program having one or five locations. If the user wishes to use the standard scale factors and decimal point positions, NSCALE should equal zero. To alter the standard scale factors, NSCALE must have five locations containing:

Location	Contents	Function
NSCALE(1)	any nonzero value	to alter standard factors
NSCALE(2)	I	printed values of the ordinate (Y) are 10^I times the actual value.
NSCALE(3)	J	printed values of the ordinate (Y) have J digits following the decimal point ($J \leq 8$)
NSCALE(4)	K	printed values of the abscissa (X) are 10^K times actual values
NSCALE(5)	M	printed values of the abscissa (X) have M digits following the decimal point ($M \leq 9$)

When standard scale factors are used, effective values of I, J, K and M are 0,3,0 and 3 respectively.

b. NHL This is the number of horizontal grid lines in the graph image.

c. NSBH This is the number of spaces between horizontal grid lines.

d. NVL This is the number of vertical grid lines in the graph image.

e. NSBV This is the number of spaces between vertical grid lines.

f. IMAGE This is a dimensioned array in the users program consisting of N locations where: N=P*(NSBH*NHL+1) P=(NSBV*NVL+1)/6 rounded to nearest integee.

g. XMAX This is the value of the abscissa at the rightmost grid line.

h. XMIN This is the value of the abscissa at the leftmost grid line.

i. YMAX This is the value of the ordinate at the uppermost grid line.

j. YMIN This is the value of the ordinate at the lowermost grid line.

k. BCD This is the Hollerith plotting character.

l. X This is a single location (or array name) containing the $\overline{X}$ coordinate(s) of the point(s) (X_i, Y_i).

m. Y This is a single location (or array name) containing the $\overline{Y}$ coordinate(s) of the point(s) (X_i, Y_i).

n. NDATA This is the number of data points (X_i, Y_i) associated with the arrays X and Y. With NDATA equal to 1, a single point will be plotted for a single execution of PLOT3. With NDATA equal to Q, Q points (X_i, Y_i) taken in sequence will be plotted for a single execution of PLOT3.

o. NCHAR This is the number of BCD (Hollerith) characters, including blanks, in the label array.

2. Restrictions on Arguments--The following are the restrictions on arguments:

NHL	>	0
NSBH	>	0
NVL	>	0
NSBV	>	0
NSBV*NVL	≤	101
XMAX	>	XMIN
YMAX	>	YMIN
BCD	>	LEFT ADJUSTED BCD CHARACTER; i.e., 1H*, 1HA, 1H1, ETC.

The image array must be dimensioned at least 867 (Decimal) locations. LABEL and BCD must contain Hollerith information only. The arguments which deal directly with data values (XMAX, XMIN, YMAX, YMIN, X, Y,) must be in floating-point mode.

(5) Deleting the Printing of Certain Portions of the Graph: There is a provision for deleting printout of certain items. These are:

1. Numeric values of the abscissa at the grid lines.
2. Numeric values of the ordinate at the grid lines.
3. Items 1. and 2.
4. The complete bottom horizontal grid line.
5. Items 1. and 2.
6. Items 2. and 4.
7. Items 1., 2., and 4.

This is accomplished by entry OMIT anytime before execution of FPLOT4.

CALL OMIT (ARG): where ARG is a positive number corresponding to one of the above seven items.

To restore printing of any of the seven items, OMIT can be called with ARG a negative number corresponding to the number of the item(s) to be restored.

(6) Changing the Decimal Output Tape During Execution: An entry PLTAPE is available for changing the output tape during execution in cases where the desired tape does not coincide with the standard output tape.

For FORTRAN IV, the calling sequence is: CALL PLTAPE. The next statement must be WRITE (A, FORMAT), where A is the number of the output tape and FORMAT is the number of a FORMAT statement.

For FORTRAN II, the calling sequence is: CALL PLTAPE (N, CHAN), where CHAN is 1 if channel A, and CHAN is 2 if channel B. Example:

> CALL PLTAPE (7,1) causes all plotting output to be written on A7 until another CALL PLTAPE (N, CHAN) is executed.

2.5.1.2 FORTRAN Subroutines for Using 65K

The 7094 computers A, B, and C are equipped with 65K memory (i.e., 65,536 words, rather than the standard 32,768 words). The additional memory is upper memory or B bank, while the machine's normal memory unit is the lower memory or A bank. Due to hardware design considerations, B bank is usually treated as an alternate memory, except that: 1) certain machine features (e.g., trapping) always revert to A bank and 2) others (e.g., I/O activity) show a distinct preference for A bank. For these reasons, upper memory has heretofore been inaccessible to FORTRAN programs.

The FAP assembler (as well as SOS) will recognize the additional instructions needed to operate 65K. Either of two procedures may be adopted, depending upon the requirements of the problem being coded. Upper memory may be used as a data storage area, or it may be used to contain an executable FORTRAN/FAP program. If used as a data storage area, it is not possible for a FORTRAN program in A bank to refer directly to quantities stored in B bank; but routines are provided to transfer data back and forth. All 32,768 words of upper memory may be used. If used to contain an executable FORTRAN/FAP program, the two banks of memory will be occupied by more or less independent FORTRAN/FAP programs; but provision is made for transferring control back and forth between them and for the exchange of COMMON data. It is not intended that both procedures be used in the same job. The following subroutines have been designed to facilitate the use of 65K memory by FORTRAN programmers.

(1) CALL STASH (X, I, J): This moves the array X of length I from A bank to locations J through J + I - 1 of B bank. I and J must be integer variables or constants, with $I \geq 1$ and $0 \leq J \leq 32767$. The mode of X is immaterial. The argument I designates the number of words to be transferred, so that if X is a double-precision or complex array, I will be double the array dimension; or if the entire array is not to be transferred, two or more calls are required.

(2) CALL BACK (X, I, J): This moves I words from locations J through J + I - 1 of B bank to fill the FORTRAN array X in A bank. I and J must be interger variables or constants, with $I \geq 1$ and $0 \leq J \leq 32767$. The mode of X is immaterial. The above comments on double-precision and complex arrays apply here also.

(3) CALL COPY: The contents of lower memory from the normal FORTRAN loading address (144_8) through the top of COMMON (77461_8) are copied into upper memory. The routine does not return to the calling routine, but instead exits to the monitor.

(4) CALL B PROG: This statement may be executed anywhere within a FORTRAN or FAP program in lower memory. It will transfer control to upper memory at the first executable statement or instruction following the CALL COPY which placed that program in B bank.

(5) CALL A PROG: This statement may be executed anywhere with a FORTRAN or FAP program in upper memory. It will transfer control to lower memory at the first executable statement or instruction following the last CALL B PROG which had been executed.

(6) CALL FROM A (Y, K): This statement may be executed by a program occupying upper memory. Y is assumed to be a variable in COMMON as defined by this program. K words beginning with Y are filed with contents of the corresponding COMMON cells of lower memory. Thus, if the program in B bank defines: COMMON P, Q, R, S, T, ..., then the statement CALL FROM A (B, 3) will place the current values of Q, R, and S into B, C, and D, respectively. K must be an integer variable or constant greater than zero. The mode of Y is immaterial.

(7) CALL FROM B (Z, L): Analogous to CALL FROM A, except that this statement is to be executed by a program in A bank, and COMMON data are transferred from upper to lower memory.

(8) Example for Using 65K: An example will help to clarify the use of these routines. Let us assume that the programmer has a FORTRAN/FAP program which is about to grow beyond single core memory capacity. He must select certain portions of the job which can be detached, subroutinized, and relegated to upper memory. Preferably, these should be segments of the job which will not require the transfer of large amounts of data between core banks. The main portion of the computation will occupy lower memory and will be called the A program. The detached segments will occupy upper memory and will be called the B program. Each must be a complete job in the FORTRAN sense; each must have a main routine, and there must be no missing subroutines. COMMON must be defined in such a way that variables which are evaluated in one program, and used in the other, occupy the same position.

The two programs are loaded into the machine as separate, consecutive jobs, each with I.D. and XEQ control cards. Either or both may be compile and execute runs. The B program must be loaded first. All data cards must follow the second, or A program (preceded by a *DATA card).

The first executable statement of the B program should be CALL COPY. As that job is loaded, its only effect is to place the program in upper memory. This routine exits to the monitor which then proceeds to process the A program.

When the A program has been loaded, it begins to execute in the normal fashion. At various points in the program it will be necessary to perform some computation which resides in upper memory. To do this, the A program assigns a predetermined code value to a control variable in COMMON, and then executes the statement, CALL B PROG. Control is thus transferred to the second executable statement of B program (i.e., the statement following CALL COPY) which should be a CALL FROM A (n, 1) in order to obtain the control variable.

B program may then use the control variable in an IF or computed GO TO to determine what computation is to be performed. In carrying out this computation, B program may obtain any necessary data from A program with one or more CALL FROM A statements; and when finished, it may return to A program by means of the statement, CALL A PROG. Control then reverts to the statement in A program immediately following the CALL B PROG which caused this diversion. The A program may obtain any results produced by B program by giving one or more CALL FROM B statements. Either program may terminate the execution by means of a CALL EXIT statement, and error diagnostics and floating-point traps are handled correctly from either program. DUMP and PDUMP will not output correctly from B program, and the debugging package cannot be used in B program. I/O statements may not be used in the B program unless the program is accompanied by a special I/O package, obtainable in the form of a binary deck from the Programming Methods Section.

There are console switches which must be set for the use of 65K, so it is advisable to note such usage on the job request card in big letters, and suitable comments followed by a PAUSE card among the control cards preceding B program may be helpful.

Subroutine decks are available.

2.5.1.3 CalComp Subroutines for IBM 7094

FORTRAN II utility routines developed for the CalComp 570 plotter are available on the system library tape. Since these routines vary significantly from the old routines, they require some explanation.

The current routines (PLOT, PLOTS, SYMBL4, TRW, TRWS) remain in the FORTRAN II library without change in calling sequence. The new routines (CCPLOT, CPLOTS, SYMBOL, and their associated routines) are recommended for any future programming for the CalComp plotter, since the FORTRAN IV routines are similar. A brief description of the new routines follows.

2.5.1.4 CALL CCPLOT (X, Y, IC)

CCPLOT is analogous to the current PLOT routine. It examines the third argument to decide whether to lift or lower the pen, then moves the pen to the position (X, Y). X and Y are both floating-point numbers, representing the distance (in inches) of the point from the origin. IC is a fixed-point number: ± 2 lower the pen; ± 3 raise the pen. IC is a signed variable: Normally IC will be positive, indicating that the plot is not yet finished. A negative IC instructs the plot routine to establish a new origin at the coordinates (relative to the current origin) given.

2.5.1.5 CALL CPLOTS (BUFFER, IDT, INDIC8)

Before attempting to do any plotting, CPLOTS (analogous to the current PLOTS) should be called. This sets up a tape buffering area with the dimensioned variable BUFFER. The first member of the dimensioned array should be used. For example, if 512 locations have been set aside for DATA, the following statements would be required before any plotting was attempted:

```
DIMENSION DATA (512)
CALL CPLOTS (DATA (I), 512, INDIC8)
```

IDT is the dimension of the tape buffering area, and INDIC8 is an end-of-tape indicator. It is originally set to zero by the routine CPLOTS. If an end-of-tape is detected while writing the plot tape (A6 or A7), all tape writing in the CalComp routines are bypassed, and INDIC8 is set nonzero. It is the programmer's responsibility to reset INDIC8 to zero so that tape writing may resume (on A7 or A6). The programmer should take the steps outlined in Section I of the CalComp Manual whenever an end-of-tape is encountered.

2.5.1.6 CALL SYMBOL (X, Y, HEIGHT, BCD, THETA, N)

SYMBOL (analogous to the current SYMBL4) is used to generate plotting symbols and alphanumeric characters.

Plotting Symbols: (X, Y) is the center of the desired symbol. HEIGHT is the height in floating-point inches. BCD contains a fixed-point integer (0-16) to specify symbol tape. THETA is a floating-point angle in degrees. A (-1) value of N specifies the pen is to be lifted before moving to location (X, Y). A (-2) value allows pen to remain lowered when moving to (X, Y).

Alphanumeric characters: (X, Y) is the lower left corner of first character. HEIGHT is the height in floating-point inches. Spacing

of characters is 6/7 (HEIGHT). BCD is location of the first word of Hollerith information. THETA is angle in floating-point degrees. (THETA is positive counterclockwise; THETA = 0 indicates a character perpendicular to X-axis.) N is the number of characters to be drawn. A negative HEIGHT may be used to pick up words stored forward in core.

2.5.2 FORTRAN IV

2.5.2.1 UMPLOT Plotting Subroutine

See Paragraph 2.5.1.1

2.5.2.2 FORTRAN Subroutines for Using 65K

See Paragraph 2.5.1.2

2.5.3 SUPPORT

2.5.3.1 FORTRAN Preprocessor

The FORTRAN Preprocessor is a 1401 program that is used to scan a FORTRAN source program for errors. This helps the programmer to eliminate those errors prior to the program being submitted to the 7094 for compilation.

The complete program writeup and listing is available in the Programming Methods Section. This is contained in the DSD Memorandum, "FORTRAN Preprocessor", dated 7/2/62.

2.5.3.2 Routines on the C1 Utility Tape

The C1 utility tape contains routines that are most frequently used by GSFC programmers. The programs on the tape are located with CALL CARDS which position the C1 tape to the record containing the desired program. There are two ways to load the C1 program: 1) C1 Program Control Card--Each routine on the C1 tape has its own CALL CARD that loads the desired program when a LOAD CARDS button action is taken. All key settings, tape setups, etc., must be done beforehand. 2) General Call Card--The proper program is read from the C1 tape by inserting the octal record number of the desired program in the address of the keys, and by pressing the LOAD CARDS button. The control program halts at location 26_8, at which time key settings, tape setups, etc., are performed. The program is executed by pressing the START button. The record numbers indicated in the following descriptions are in octal.

2.5.3.3 WDOMFP-Octal Mnemonic/Floating Point Core Dump (Record No. 1)

This prints on the on-line printer or writes on tape A2 the contents of a selected part of core memory, in octal or floating decimal. Octal Dumps may be taken with or without mnemonics. The contents of the console registers are recorded each time WDOMFP is loaded from C1. Several dumps may be taken in succession. Starting and ending addresses plus control information may be supplied either by control cards (see writeup) or via the keys. Tape B2 is always required as a scratch tape. This routine cannot be loaded with the General C1 Call Card.

2.5.3.4 MXMRGE-Merge Mods with SQUOZE (Record No. 2)

This merges control cards and modification packet with SQUOZE tape on B8. The control cards and mods may be read in on-line or from tape A5. Output is a job tape, produced on A3, to be used for an SOS execution run.

2.5.3.5 IBTD-Tape Dump (Record No. 3)

This prints or writes on tape A2 the contents of selected records or files of any tape on any channel (except A2). Input tape may be dumped from its present position or may be rewound prior to dumping. Input tape is not repositioned after dumping.

2.5.3.6 PPTDAC-Tape Duplicate and Compare (Record No. 5)

This reads any tape on channel A and copies onto or compares with any tape on channel B. The number of records or files to be so treated may be specified on control cards (see writeup) via the keys. Codes are also provided to space either tape backward or forward any number of records or files, or to rewind either tape, or to write an end-of-file on the B tape. When duplicating or comparing records, an end-of-file mark is counted as a record. Miscomparison will produce an on-line print giving the file and record number from the starting point of that operation. If comparison is OK, no print occurs.

2.5.3.7 MXHSPR-Print High Speed from Log Tape (Record No. 12)

MXHSPR reads a B6 log tape and writes a duplicate on C6 while searching for a time of lift-off in the logged data. Upon finding the time of lift-off or using a time of lift-off of zero if no lift-off was found, MXHSPR will search the B6 log tape for all data pertinent to the first display. This data are unpacked, scaled and written on the output tape. B6 and C6 are alternately searched for the various displays until all data has been recorded on the output.

2.5.3.8 MXPRLG-Select TTY Data from Log Tape (Record No. 13)

This is accomplished by pressing the console entry keys corresponding to subchannel numbers whose messages are to be printed. MXPRLG will read the B6 log tape for complete messages corresponding to these subchannels. The messages are written on the output tape and MXPRLG continues to search the log tape and the intermediate tapes for additional messages. The sense lites are used as a binary counter to count the number of channels to be processed.

2.5.3.9 MXCHER-Print Selected Subchannels from Mercury Log (Record No. 14)

After reading a Mercury log tape on unit B6, select and print (off-line) all 17-word blocks of information which are identified with a subchannel number which has been selected for output. An option is provided to permit the user to search the tape for several subchannels in one pass or to perform a separate search for each subchannel requested. In the first mode, entries are printed out in order in which they occur on the tape, while in the second mode, all entries for a given subchannel are printed together. The log tape is tested for density mode and is checked for readability.

2.5.3.10 HSIN7-Decode and Print High Speed from Log Tape (Record No. 15)

This decodes and prints the high speed input data (B/GE, IP 7090, and Bermuda High Speed Radar) from a Mercury log tape.

2.5.3.11 MXPOCL-Print Mercury Log Tape in Octal (Record No. 16)

This reads a Mercury log tape on unit B6 and prints (off-line) the information it contains in octal. B6 is set to the proper density mode and is checked for readability. An option is provided to permit the user to print the contents of the whole tape, or to begin printing just prior to the lift-off indication, or to print those entries time tagged within a selected time interval.

2.5.3.12 MSHSPL-Log Tape Plotting Program (Record No. 20)

This reads a Mercury log tape on unit B6, extracts information designated as DCC Subchannel 3 and plots, via the DCC and local plotboards, that information relating to Cape Kennedy plot-boards 1, 2, 3, 4 or Bermuda. The selection is made by key entry. Program stops occur between various phases of the Mercury run.

2.5.3.13 GFCHEK-Checksum Corrector (Record No. 25)

This reads binary cards on-line and repunches the same information, but with the checksum recomputed. Sense switch options indicate to

the program whether input cards are row or column binary, and whether they are two-word cards with the checksum in 9R (or Columns 4-6 in the case of column binary) or 23-word cards with the folded checksum in Columns 25-39 or 9L (or Column 3 in the case of column binary cards). Output cards will be of the same format as of input cards.

2.5.3.14 OHC01-Hollerith to OCT Pseudo-Op Card Image (Record No. 27)

This reads Hollerith cards on-line, and for each input card punches on-line a set of OCT pseudo-op cards which, if assembled with the user's program, will generate a card image corresponding to the input card. A blank card will separate each card image set of OCT's. A sense switch option permits reading input cards in pairs so that the contents of Columns 1-6 of the first card will appear as a location symbol in the first OCT produced in that set, and the second card will be treated as described above.

2.5.3.15 WDCTS-Card-to-Tape Simulator (Record No. 32)

WDCTS will transcribe cards from the card reader to any desired tape in either BCD or column binary as designated by the standard control punching in the cards. The format of the tape record is exactly like that written by an off-line 1401, except that the information in Columns 73-80 is replaced by blanks for Hollerith cards and by zeros for column binary cards. Special control cards may be used to cause the program to write an end-of-file mark or to cause the program to clear itself out of memory and simulate the action of the Load Cards key. Each Hollerith card is checked for illegal punches.

2.5.3.16 SUMMARY-Summarize SOS SQUOZE Tape Statistics (Record No. 36)

This locates and reads preface record from an A5 SQUOZE tape resulting from an SOS compilation or PS run and prints, on-line, various information affecting the permissible size and mod deck which the examined program will accept.

2.5.3.17 COL8ER-Update Symbolic Tape, Produce Symbolic from Listing Tape (Record No. 41)

This routine is used to update symbolic tapes by adding or deleting sections of a program by using the alter number generated by SOS.

2.5.3.18 MXILCO-Print Real-Time CORE Output (Record No. 57)

This reads a tape on unit B6 containing real-time CORE (RTCOR) output, and prints (off-line) the information it contains in the formats specified by codes on the tape. Output is produced on unit A3.

2.5.3.19 SHARE Library Index

A cross-reference index to the program library is maintained by the Data Systems Division Programming Methods Section. It consists of a Key Word in Context (KWIC) index to the library routines on file, plus the title and descriptive listings sorted in various orders.

The subject codes are those defined by SHARE, but extended by several new categories of a more specialized nature and applicable to the type of programs used at the Goddard Space Flight Center. The SHARE routines are identified by a serial number (in addition to the customary symbolic name) consisting of the letters SDA (SHARE Distribution Agency), followed by a four-digit number. It is planned to include programs other than those distributed by SHARE in the program library. These programs will be identified by a serial number which begins with letters other than SDA.

More complete information on any of the routines in the cross-reference index is available in either the SHARE library abstract listing or in Building 3, Room 127.

There is also available a catalog of abstracts which lists programs and subprograms contained in the reissued SHARE library, and which is supplemented by routines from both Goddard Space Flight Center personnel and other personnel. Those desired programs unattainable in the reissued SHARE library may be located in its predecessor dated 10/30/63; however, this old SHARE library catalog is not being maintained.

The programs are categorized according to two character classification codes and are listed in alphanumeric order within each code. SHARE codes consist of a letter followed by a digit. These codes have been enlarged as a result of several locally defined categories of a more specialized nature. Consequently. these codes consist of a period which is followed by two characters.

INDEX TO CHAPTER 2

INDEX TO CHAPTER 2 (Cont'd)

C (Cont'd)

D

INDEX TO CHAPTER 2 (Cont'd)

D (Cont'd)

E

INDEX TO CHAPTER 2 (Cont'd)

E (Cont'd)

F

INDEX TO CHAPTER 2 (Cont'd)

F (Cont'd)

INDEX TO CHAPTER 2 (Cont'd)

F (Cont'd)

G

INDEX TO CHAPTER 2 (Cont'd)

H

I

INDEX TO CHAPTER 2 (Cont'd)

INDEX TO CHAPTER 2 (Cont'd)

L (Cont'd)

M

N

O

INDEX TO CHAPTER 2 (Cont'd)

O (Cont'd)

P

Q

R

INDEX TO CHAPTER 2 (Cont'd)

R (Cont'd)

S

INDEX TO CHAPTER 2 (Cont'd)

S (Cont'd)

T

U

INDEX TO CHAPTER 2 (Cont'd)

CONTENTS

CHAPTER 3 CAMEO

CONTENTS (Cont'd)

CONTENTS (Cont'd)

ILLUSTRATIONS

TABLES

CHAPTER 3

CAMEO

3.1 SYSTEM DESCRIPTION

This chapter describes the Computer-independent Abstract Machine-language Encoder and Operating-system and its use. The CAMEO system is available on all Goddard Space Flight Center Data Systems Division large scale computers.

CAMEO is designed to process a variety of related or unrelated jobs sequentially or individually with complete operator convenience and control. With complete control in the hands of the operator, jobs are processed more responsibly and intelligently with less attention to strictly operational requirements demanded of the programmer.

3.1.1 SYSTEM STRUCTURE

There is a CAMEO for the 7094 data processing equipment and a CAMEO for the 1107 data processing system. On both systems, CAMEO is on a self-loading magnetic tape, a copy of which is stored at the console of the A, B, C, E, F, G, H, and J computers.

The flow diagram (Figure 3-1) illustrates the various functional components of CAMEO and the paths of control between them. The notations on each line indicate the command, condition, or action which causes that path to be taken.

Tape assignments within CAMEO are entirely at the option of the programmer and, for production work, at the option of the operator as well. Any tape may be used in a BCD mode or a binary mode or in a mixed mode. For instruction in the use of CAMEO, consult Programming in Mystic: A Primer on the Use of CAMEO, X542-64-393, December 1964.

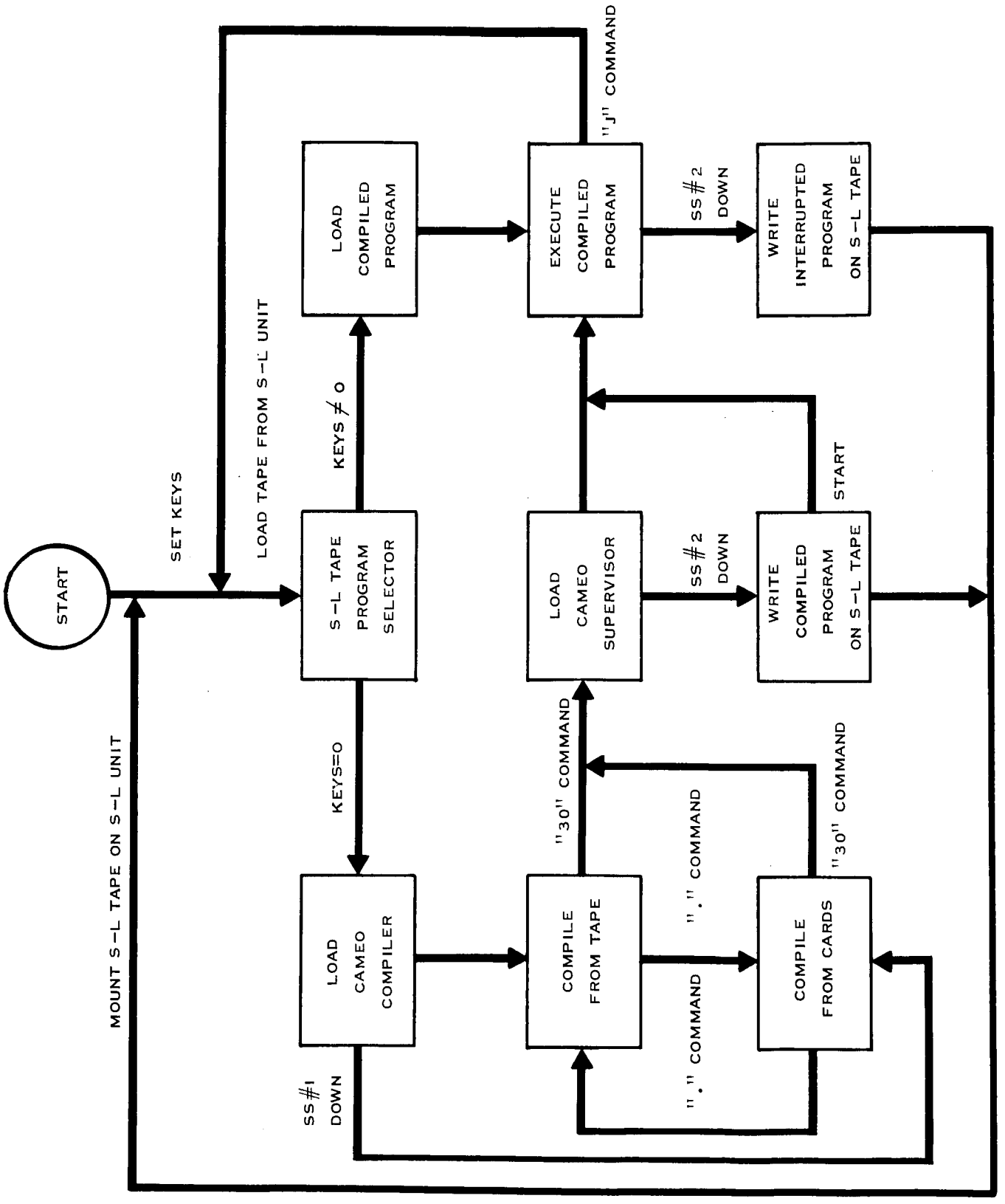


FIGURE 3-1 FLOW DIAGRAM OF CAMEO OPERATION

3.1.2 MACHINE CONFIGURATION

The programmer using the Goddard Space Flight Center computer complex has a vast quantity of data processing equipment at his disposal. The CAMEO system operates on these large scale computers. A partial list of this equipment is given in Table 3-1. In addition, there exists an associated collection of peripheral equipment for off-line support. A partial list of this equipment is given in Table 3-2.

Table 3-1. Major Computer Equipment

Bldg. Loc.	Computer Facility	Memory Size	Magnetic Tape Units	Line Printer	Card Reader	Card Punch	Disk	Data Channel	DDC
14	A-7094	65K	14-729-IV	716-I	7223-I		1301-II	3	Yes
14	B-7094	65K	14-729-IV	716-I	7223-I		1301-II	3	Yes
3	C-7094-II	65K	14-729-IV	716-I	7223-I	721-I	1301-II	3	Yes
1	E-7094-II	32K	12-729-IV 4-729-VI	716-I	711-I	721-I		2	No
3	F-7094	32K	12-729-IV	716-I	711-I			2	No
14	1107G*	65K	24	1	1	1	2 Drums	3	No
14	1107H*	65K	16	1	1	1	2 Drums	3	No

*Paper tape reader/punch

Table 3-2. Peripheral Equipment

Bldg. Loc.	Computer Facility	Memory Size	Magnetic Tape Units	Card Read/Punch	Line Printer
14	A-IBM-1401	1.4K	1-7330		1403-II
14	D-IBM-1401	4K	2-7330	1012-I* 1402-I	1403-II
14	E-IBM-1401	4K	2-729-II	1402-I	1403-II
14	F-IBM-1401	8K	2-729-II	1402-I	1403-II
1	IBM-1460	8K	4-729-VI*** 2-729-IV***	1402-I	1403-III
14	I-IBM-7010**	100K	8-729-IV	1402-I	1403-III
14	IBM-1401	8K	4-729-IV	1402-I	1403-II

* Paper tape reader/punch
** 1301 disk
*** Switchable units

3.1.3 SYSTEM TAPE MAINTENANCE

The Advanced Orbital Programming Branch (AOPB) of the Data Systems Division has primary responsibility for maintaining the CAMEO system tape. Tape revisions or updating occur periodically as a consequence of one or more of the following conditions:

(1) A requirement to translate valuable additions to GSFC hardware into software features.

(2) Spin-off from a continuing research and development effort in computer programming.

3.1.4 ERROR REPORTING

The Advanced Orbital Programming Branch (AOPB) of the Data Systems Division has the responsibility of maintaining the CAMEO system. Any question regarding system utilization and system discrepancies should be directed to AOPB personnel. The current method of reporting system discrepancies is verbal.

3.2 DETAILED PROCEDURES

This section includes several illustrations showing computer set-up for a number of typical runs; presents a description of the control commands and their use; and offers means by which to use CAMEO effectively.

3.2.1 CAMEO CONTROL COMMANDS

This paragraph presents in detail a description of the control commands recognized by CAMEO. The user controls and directs the processing of his job through the operator by requesting settings of console switches thereby directing CAMEO to perform any one of several functions.

3.2.1.1 Encoding Control

(1) Initial Input Mode Switch: Setting this switch causes CAMEO to begin encoding by reading from the card reader. This switch is Sense Switch #1 on the 7094 and Jump Switch #15 on the 1107. The input mode is controlled internally by the dot (.) command after encoding starts.

(2) Program Save Switch: Setting this switch causes the encoded program to be recorded on a self-loading magnetic tape for later reloading. This switch is Sense Switch #2 on the 7094 and Jump Switch #14 on the 1107.

(3) Unit Program Switch: Setting this switch causes the tape unit containing the source program to rewind following CAMEO encoding since it is not part of a batch run. This switch is Sense Switch #3 on the 7094 and Jump Switch #13 on the 1107.

(4) Error Bypass Switch: Setting this switch causes the CAMEO encoder to bypass any source program errors detected by simply deleting the erroneous commands and continuing instead of halting for operator correction. This switch is Sense Switch #4 on the 7094 and Jump Switch #12 on the 1107.

(5) Cumulative Encoding Switch: Setting this switch causes CAMEO when loaded to leave core memory undisturbed and to encode the coming program in with the contents of core as found. This switch is Sense Switch #5 on the 7094 and Jump Switch #11 on the 1107.

3.2.1.2 Execution Control

Interrupt-unload Switch: Setting this switch causes the running program to be interrupted at the next flow connector execution and to be recorded on a self-loading magnetic tape for later restart. This switch is Sense Switch #2 on the 7094 and Jump Switch #14 on the 1107. (See 3.2.1.1(2)).

3.2.2 CAMEO CONSOLE SETTINGS

CAMEO processes a job not under the control of a control deck but rather under the control of switches set at the computer console by a person responsible for successful completion of the job. There follow some examples of typical machine set-ups and console settings to illustrate the various operations performed under CAMEO.

3.2.2.1 Encode and Execute

Machine Set-up

CAMEO Element	7094 unit	1107 unit
Encoder ready	on tape A1	on Tape 2-0
Source deck ready	at Card Reader	at Card Reader (12-0)
Source tape ready	on Tape A2	on Tape 2-1

Console Switch Settings

CAMEO Switch	7094 Switch	1107 Switch
Initial Input Mode	Sense Switch 1	Jump Switch 15
Unit Program	Sense Switch 3	Jump Switch 13

Operator Action

CAMEO Action	7094 Action	1107 Action
Start	Press Load Tape	Bootstrap from 2-0

Encoding will begin with source cards. A dot (.) card in the deck will cause encoding to continue from that point with the tape. On the tape, a dot (.) record may be used to cause to return to encoding from the card reader.

3.2.2.2 Encode and Save on Self-Loading Tape

Machine Set-up

CAMEO element	7094 unit	1107 unit
Encoder ready	on Tape A1	on Tape 4-0
Source deck ready	at Card Reader	at Card Reader (12-0)
Source tape ready	on Tape A2	on Tape 4-1
Save tape ready	on Tape A3	on Tape 4-2

The Operator Action is the same as in 3.2.2.1

Console Switch Settings

CAMEO Switch	7094 Switch	1107 Switch
Initial Input Mode	Sense Switch 1	Jump Switch 15
Unit Program	Sense Switch 3	Jump Switch 13
Program save	Sense Switch 2	Jump Switch 14

When encoding is complete the encoded program is written on a self-loading tape for later use.

3.2.2.3 Load and Execute

Machine Set-up

CAMEO Element	7094 unit	1107 unit
Program ready	on Tape A1	on Tape 4-0

No switches are set at the console and the Operator Action required is the same as in 3.2.2.1.

3.2.2.4 Postmortem Decimal Dump

Machine Set-up

CAMEO Element	7094 unit	1107 unit
Encoder ready	on Tape A1	on Tape 4-0
Dump ready	on Tape A2	on Tape 4-1

CAMEO Switch	7094 Switch	1107 Switch
Unit Program Switch	Sense Switch 3	Jump Switch 13
Cummulative Encoding Switch	Sense Switch 5	Jump Switch 11

The Operator Action required is the same as in 3.2.2.1.

3.3 CAMEO BIBLIOGRAPHY

This section provides a list with abstracts of documents on CAMEO.

3.3.1 Programming in MYSTIC: A Primer on the Use of CAMEO X-542-64-393--Goddard Space Flight Center

The aim of this document is to guide the beginner in learning the use of CAMEO. It provides a comprehensive description of the MYSTIC command repertoire and presents to the reader a set of representative exercises.

3.3.2 CAMEO System Description X-542-64-148--Goddard Space Flight Center

This document presents a system description of CAMEO and of the MYSTIC language that is the input to the CAMEO system. It develops the preparation of a problem for solution by presenting a typical problem. There is also included a brief description of the system's performance since its inception in mid-1955. As an introduction, this document offers a simplified analysis of the complete program production process as it is accomplished by a composite system made up of men and a machine.

3.3.3 CAMEO: Univac 1107 Usage X-542-64-248--Goddard Space Flight Center

This document is concerned with the information necessary for the operation of CAMEO on the Univac 1107. It describes tape, card, and console set-ups; how halts are handled; and shows the typewriter messages generated at the time of normal stops or error halts. A complete listing of the Sleuth II program which does the encoding and a flow diagram of that program are attached to provide answers to any questions that may arise about error recovery, I/O operations and floating point adjustments. It is not necessary to be familiar with the Univac 1107 to use this system, but the listing and diagram are provided for those who are interested in the details of the encoding system.

3.3.4 CAMEO: IBM 7094 Usage X-542-64-363--Goddard Space Flight Center

This document covers the necessary steps in operation of the CAMEO system on the IBM 7094. It describes the input to the CAMEO encoder, tape assignments, console operations, sense switch settings, output, and CAMEO halts, as well as supplementary operations which are used to control encoding in the 7094 CAMEO. It supplies information concerning the duplicating of the machine language program tape and generating of the CAMEO tape. The 7094 flow diagram and the CAMEO FAP listing are included for completeness.

3.3.5 MYSTIC Dictionary Routine
X-542-65-17--Goddard Space Flight Center

This document describes how the MYSTIC Dictionary Routine is used to provide rapid communication between programming systems by permitting the translation of MYSTIC programs to computers for which no CAMEO compiler exists. There are reproductions of the MYSTIC Memory Maps of the Executive Routine and the subroutines written for us by this program; flow diagrams of the program; and a listing of the source program.

3.3.6 R15 CAMEO Quick Diagnostic Function Program Description: AOPB Systems Manual
X-542-65-119--Goddard Space Flight Center

This document describes how the CAMEO quick diagnostic function is used to analyze a MYSTIC program to provide an index of instruction line numbers versus input program addresses. The index will distinguish those instructions which alter the contents of an address from those which use the contents of an address. Also, for each address it provides a list of all Op Codes operating on it.

3.3.7 R143 Tape Modification Routine Program Description: AOPB Systems Manual
X-542-65-215--Goddard Space Flight Center

This document describes how the utility program R143 is used for modifying (or updating) MYSTIC and other BCD tapes. It may be used to delete or insert records or files, and provides a fast and easy-to-use means of program maintenance, thereby reducing the need to retain, manually update, and reconvert card decks. The document further contains a listing of a sample modify deck and the on-line output produced during the run, along with a description of the on-line output.

3.4 PROGRAMMING SUPPORT PACKAGES

This section collects descriptions of the use of many of the pre-programmed packages available to programmers of the Advanced Orbital Programming Branch.

3.4.1 UTILITY PACKAGE

This package is used by the CAMEO programmer to checkout MYSTIC programs.

3.4.1.1 Print-Out Memory (F004)

This subroutine prints on line or writes on tape the contents of operand memory in floating point decimal five locations per line.

3.4.1.2 Interval Core Dump (F062)

This subroutine is used to dump intervals of core memory between specified core locations at specified points during a program run. Core intervals are dumped prior to the execution of a specified Begin Command.

3.4.1.3 Change of Code (F177)

This subroutine is used to make temporary changes in a MYSTIC program at execution time in MYSTIC but without re-encoding.

3.4.2 ARITHMETIC PACKAGES

These packages are used by the CAMEO programmer to simplify his work by employing the notation of the problem in the programming of scientific problems.

3.4.2.1 Vector Arithmetic Package

This package allows vector operations to be performed on the contents of three consecutive addresses in the manner of a two address machine. Operations permitted are: move, take magnitude, take direction (unitize), add, subtract, dot product, cross product, scalar multiply.

3.4.2.2 Matrix Arithmetic Package

This package allows matrix operations to be performed on the contents of specified addresses in the manner of a two address machine. Operations permitted are: multiply, invert, transpose.

3.4.2.3 Fourier Series Arithmetic Package

This package allows Fourier series operations to be performed on the contents of specified addresses in the manner of a two-address machine. Operations permitted are: move add, subtract, multiply, term extract, differentiate, scalar multiply, integrate.

3.4.2.4 Power Series Arithmetic Package

This package allows power series operations to be performed on the contents of specified addresses in the manner of a two-address machine. Operations permitted are: move, add, subtract, multiply.

3.4.2.5 High-speed Elementary Functions Package

This package provides the high-speed computation of elementary functions by way of optimized machine language programs to compute: reduced angle, arc sine/arc cosine, arc tangent, exponential, natural logarithm, sine/cosine, square root.

3.4.3 SPECIAL PURPOSE PACKAGES

These packages are designed to make available to the CAMEO programmer some of the special-purpose modifications of GSFC equipment for use on specific problems which would benefit from them.

3.4.3.1 Real-Time Adapter IBM 7094

This program allows use of Mystic routines in the real-time mode. This adapter was used to support a real-time fly-by of a SYNCOM satellite over Wallops Island. The one adapter introduces the entire CAMEO system to real-time applications.

3.4.3.2 FACIL (Fortran Assembly Compatible Interface Linkage)

This subroutine provides linkage between programs coded in Fortran IV and subroutines initially coded in Mystic and converted in IBMAP.

3.4.3.3 MYSTIC DICTIONARY ROUTINE (See GSFC Document, X-542-65-17)

This program is used to provide rapid communication between programming systems and expedite the translation of Mystic programs to computers for which no CAMEO compiler exists. Since Mystic commands are very similar to Macros, they can be defined prior to any processing. A change in the definition of the Mystic commands produces a new set of operations and thus allows a variety of output

languages. The Mystic program to be converted must be on tape in BCD. The Memory Map and dictionary are on cards. There are two options in this program that allow the user to choose between naming symbolic references in the program or allowing the routine to compute the references.

3.4.3.4 Drum Storage Adapter (1107)

This program allows the Mystic programmer to use the 1107 magnetic drum storage and an indirectly addressed large-scale memory.

3.4.4 PERIPHERAL EQUIPMENT UTILITY ROUTINES

These routines are used to facilitate program development, system development, and program and system documentation.

3.4.4.1 Table of Contents

Using a set of control cards and the AOPB Functions Tapes as input to the IBM 1401, this routine produces a Table of Contents of the AOPB Function Tapes. This Table of Contents includes each routine name, description, and where it may be found (i.e., volume number and page number) in the AOPB Functions List. The Table of Contents is keyed to the Mystic list.

3.4.4.2 Tape Modifications Program

This IBM 1401 program is used to modify BCD tapes (usually BCD program tapes). The input consists of the tape to be modified and a card deck containing control cards and any desired modifications. The output is the modified tape and a listing of the modified tape, preceded by a table indicating each modification. The listing of the modified tape may be either an unedited, single-spaced listing, or an edited Mystic Listing (see 3.4.4.3).

3.4.4.3 Mystic List

This routine is used to produce edited listings of CAMEO routines and programs. The listings are used to facilitate program development and for program documentation. The IBM 1401 version accepts as input either cards or tape and produces, at the end of the listing, an index of subroutines by page number and an index of subroutines by K-card. The Univac 1004 version accepts only card input and does not produce the subroutine indices.

3.5 ENCODER TAPES

This section describes in brief the characterization of the several encoders that are embedded in the CAMEO system. These Mystic compilers operate under the same principle and are available on the Univac 1107 and the IBM 7094 32K and 65K.

3.5.1 ENCODER FOR UNIVAC 1107

The Mystic encoder written in the 1107 SLEUTH II assembly language is available for encoding on Mystic programs to be run on the 1107 computer. The programmer should indicate that his program is to be run on the 1107 when submitting his job request card.

3.5.2 ENCODER FOR IBM 7094 (32K)

The Mystic encoder written in the FORTRAN ASSEMBLY PROGRAM (FAP) language is available for encoding Mystic programs to be run on the 7094 32K computer. The programmer should indicate that his program is to be run on the 7094 32K when filling out his job request card.

3.5.3 ENCODER FOR IBM 7094 (65K)

The Mystic encoder written in the FORTRAN ASSEMBLY PROGRAM (FAP) language is available for encoding Mystic programs to be run on the 7094 65K computer. The programmer should indicate that his program is to be run on the 7094 65K when submitting his job request card.

3.5.4 ENCODER FOR IBM 7094 (DOUBLE PRECISION)

A Mystic encoder written in the FORTRAN ASSEMBLY PROGRAM (FAP) language is available for encoding 16 digit Mystic programs to be run in double precision on the 7094. Double-precision is a technique for carrying out floating-point calculations with twice the normal number of significant places. Programmers desiring double-precision should so indicate by specifying the 7094 double-precision encoder on the job request card.

3.6 FUNCTIONAL AIDS AND CODING SHEETS

3.6.1 MYSTIC STORAGE MAP

The Mystic Storage Map (see Figure 3-2) is designed to aid in allocation of storage. Each page of a map can be used to specify the contents of a block of one hundred locations.

3.6.2 CAMEO CODING SHEETS

The CAMEO Coding Sheets (see Figure 3-3) are so designed that the formats of the various instructions are apparent. This coding paper facilitates both the writing and the keypunching of programs.

3.7 JOB PROCEDURE

3.7.1 DEFINITION

1) Define the problem, limits of variables, input, desired output.
2) Analyze the proposed technique of solution.
3) Organize and flow chart the steps necessary for solution.

3.7.2 DEVELOPMENT

1) Acquire necessary standard functions from AOPB functions library. If program is long, put these functions on tape.
2) Acquire Print-Out Memory, Interval Core Dump, One Word Load subroutines and include them in the program.
3) Write routines specific to the problem using CAMEO Coding Forms and CAMEO Memory Maps.
4) Obtain Mystic Listing of the program and use this to check the program.

3.7.3 TESTING

1) Make any necessary or desirable changes in the program.
2) Obtain Mystic Listing of the program for checking.
3) Test the program.
4) Check results. If results are correct, proceed to step 5). If results are not correct, consider use of Interval Core Dump for analysis of errors and proceed to step 1).
5) Make and check self-loading binary systems tape.

3.7.4 DOCUMENTATION

1) Review and, where necessary, modify documentation developed in Paragraphs 3.7.1 and 3.7.2.
2) Include the following:
   a. Mystic list of the program
   b. Memory Map of the program
   c. Sample input
   d. Sample output
   e. Flow chart

MYSTIC STORAGE MAP

MYSTIC PROGRAM NO. _____ PAGE ____ OF ____

DESCRIPTION: _____

_____

PROGRAMMER: _____

00	00	01	02	03	04
05	05	06	07	08	09
10	10	11	12	13	14
15	15	16	17	18	19
20	20	21	22	23	24
25	25	26	27	28	29
30	30	31	32	33	34
35	35	36	37	38	39
40	40	41	42	43	44
45	45	46	47	48	49
50	50	51	52	53	54
55	55	56	57	58	59
60	60	61	62	63	64
65	65	66	67	68	69
70	70	71	72	73	74
75	75	76	77	78	79
80	80	81	82	83	84
85	85	86	87	88	89
90	90	91	92	93	94
95	95	96	97	98	99

NOTES:

Figure 3-2. Mystic Storage Map

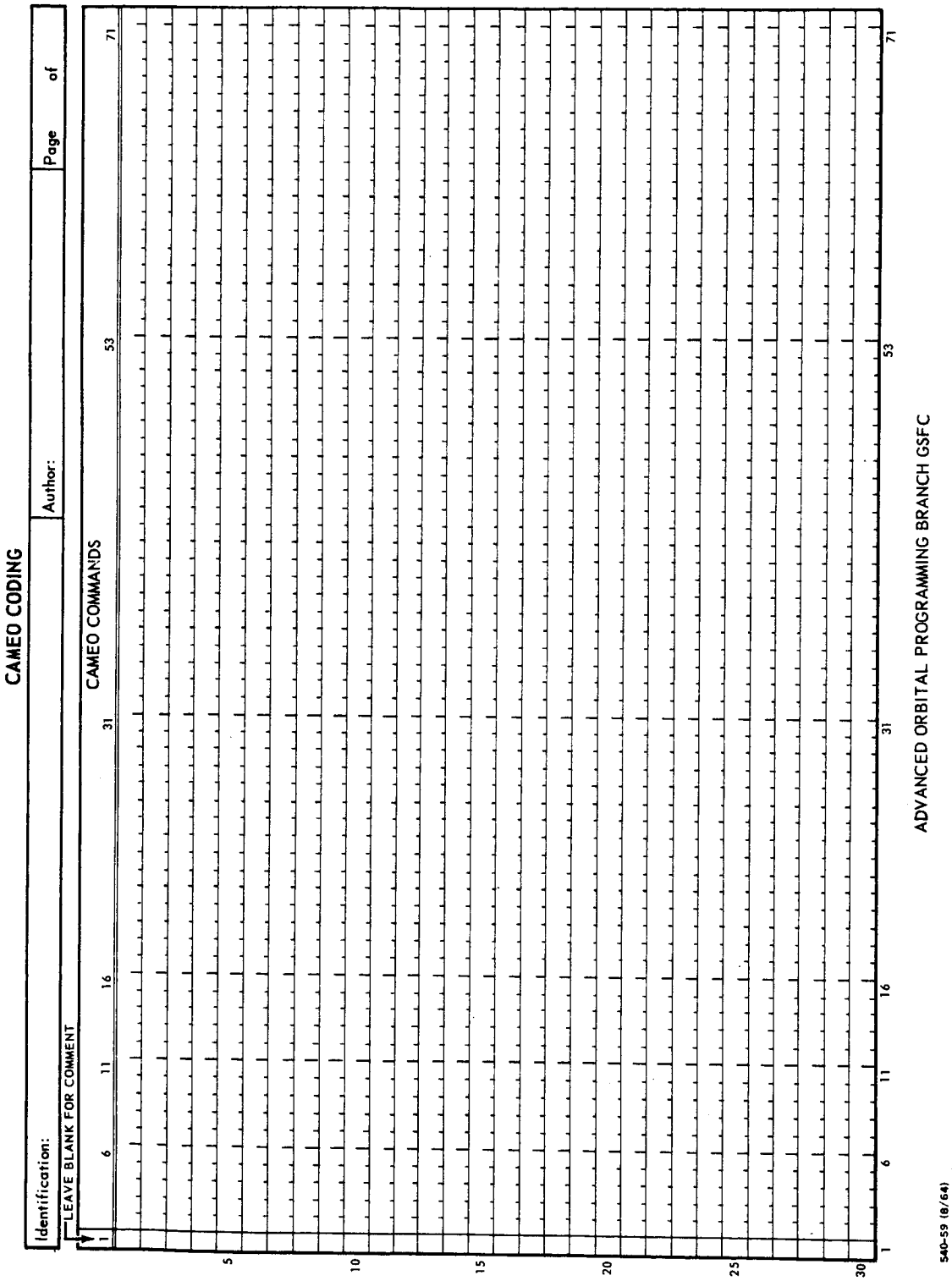


Figure 3-3. CAMEO Coding Sheet

3.8 AOPB FUNCTIONAL SUBROUTINES

The Advanced Orbital Programming Branch (AOPB) has a library of commonly used subroutines such as functions for generating the sine or cosine of an angle, determining the square root or n^{th} root of a number, etc. These routines are written in MYSTIC and are as follows:

AOPB Number and Date	Function Name	Function Objective and Remarks
FF001--611015	Sine-Cosine	Computes sine at entrance K + 1. Computes cosine at sine entrance +4. Argument must be in radians. Requires 30 storage locations.
F002--611015	Arc-sine, Arc-Cosine, Square Root	Computes angle in first or fourth quadrant if entered with sine. Computes arc sine at entrance K + 1. Computes arc sine entrance +10. Resulting angle is in radians in first or fourth quadrant for arc sine and in first or third quadrant for arc cosine. Extracts square root at arc sine entrance +40. Requires 60 storage locations.
F003--611015	Square Root	Extracts square root. Requires 20 storage locations.
F004-611015	Memory Print, Output Scale	Prints on-line the contents of memory in five locations per line in floating point decimal. Normally located at K09727 output converter, location K09748, it takes floating-point number and exponent from one location and places base in specified location followed by exponent in subsequent location. Requires 150 storage locations.

AOPB Number and Date	Function Name	Function Objective and Remarks
F005--611015	N^{th} Root	Extracts n^{th} root of number. Argument is followed by N in subsequent location. Requires 40 storage locations.
F006--611015	Position in Ellipse Orbit Generator (PE)	NOTE: Routines F006 through F009 are available from AOPB.
F007--611015	Brouwer Satellite Theory Orbit Generator (Brouwer)	
F008--611015	Hansen Satellite Theory Orbit Generator (H ST)	
F009--611015	Gill Method Integration Orbit Generator (MCOI)	
F010--611015	Arc-Tangent X	Computes arc tangent or argument. Resulting angle is in radians in first or fourth quadrant. Requires 25 storage locations.
F011--611015	Arc-Tangent	Computes arc tangent with proper quadrant allocation. Arguments are sine and cosine in consecutive locations. Resulting angle is in radians. Requires 30 storage spaces.
F012--611102	Vector Package	For vector relocation enter K + 1. Compute vector magnitude at package entrance +10. Computer vector direction at relocation entrance +20. Vector subtraction at relocation entrance +40. Dot product at relocation entrance +50. Cross product at relocation entrance +60. Scalar by product relocation at +75. Argument for this product is scalar. Arguments

AOPB Number and Date	Function Name	Function Objective and Remarks
F012--611102 (Cont'd)	Vector Package (Cont'd)	for all others call for vector in these successive locations. Entire package requires storage locations.
F013--611102	Quadrant Determination	Computes angle with proper quadrant allocation. Arguments are sine and cosine in successive locations. Resulting angle is in radians. Requires 25 storage locations.
F014--611102	Matrix Multiplication	Multiplies first matrix by second. Arguments are in consecutive locations. Elements of product matrix are stored in same manner. Requires 35 storage locations.
F015--611102	Integer Converter	Converts floating-point number to integer and fractional parts in consecutive locations. Requires 20 storage locations.
F016--611102	Natural Log	Computes the natural logarithm of the absolute value of P. Requires 40 storage locations.
F017--611102	Exponential	Computes E to the X for value of X. Requires 30 storage locations.
F018--611102	Degrees, Minutes, Seconds to Radians	Converts degrees, minutes, and seconds to radians. Arguments must be positive in three consecutive locations. Requires 15 storage locations.
F019--611102	Alphabetic Sign, Degrees, Minutes, Seconds to Radians	Converts alpha sign, degrees, minutes, seconds, fractions of seconds to radians. Will convert positive or negative degrees. Argument must be in three consecutive locations, that is, first location a sign

AOPB Number and Date	Function Name	Function Objective and Remarks
F019--611102 (Cont'd)	Alphabetic Sign, Degrees, Minutes, Seconds to Radians (Cont'd)	sign and the second in degrees; third in minutes, and seconds, and fractions of seconds (capable of handling three decimals). Requires 25 storage locations.
F020--611102	Hours, Minutes, Seconds to Radians	Converts hours, minutes, seconds to radians. Argument in three consecutive locations. Requires 15 storage locations.
F021--611102	Hours, Minutes, Seconds to Seconds	Converts hour, minutes, seconds to seconds. Argument in three consecutive locations. Requires 10 storage locations.
F022--611102	Day Count	Converts year, month, and day to number of days from Jan. 1 of given year through given date. Arguments in three consecutive locations. Requires 35 storage locations.
F023--611102	Date	Converts year and day count (number of days from Jan. 1 of given year through given date) to year, month, and day. Arguments are in two consecutive locations. Requires 50 storage locations.
F024--611102	Observation Day	Converts year of reference, day count (number of days from Jan. 1 of given year through day of reference), observation year, month, day to number of days from reference date through observation date. Arguments must be in five consecutive locations. Requires 20 storage locations.

AOPB Number and Date	Function Name	Function Objective and Remarks
F025-611102	Julian Days-- Seconds to Vanguard Units of Time	Converts Julian days and seconds to Vanguard units of time. Arguments must be in two successive locations. Vanguard units of time are based on 1 VUT = 806.832 seconds. Requires 15 storage locations.
F026--611102	Julian Days-- Seconds to Julian Days, Hours, Minutes, Seconds	Converts Julian days and seconds to Julian days, hours, minutes, and seconds. Arguments are in three successive locations; the third location contains the rounding factor. Requires 20 storage locations.
F027--611102	Azimuth Elevation to L, M, N	Converts azimuth and elevation to direction cosines L, M, and H. Argument in radians are in successive locations. Requires 15 storage locations.
F028-611102	Hours Angle and Declination to L, M, N	Converts hour angle, declination and Phi to direction cosines L, M, H. Arguments in Radians are in three successive locations. Requires 25 storage locations.
F029-611102	Right Ascension, Declination to L, M, N	Converts universal time, right ascension, declination, right ascension mean sun, stations longitude, and station latitude to direction cosines L, M, N. Arguments in radians in six consecutive locations. Requires 25 storage locations.
F030--611102	Header Load	Loads header data from card of argument equals zero, or from tape argument not equal to zero into 14 specified locations. Requires storage locations.

AOPB Number and Date	Function Name	Function Objective and Remarks
F031--611102	Satellite Identification Load	Loads satellite identification data from card if argument equals zero and from tape if argument does not equal to zero into nine consecutive storage locations. Requires 30 storage locations.
F032--611102	Satellite Identification Load and Print	Loads and prints on-line satellite identification data. If initial argument equals zero, it loads a card; if argument does not equal to zero, it loads from tape. If second argument is not equal to zero, it punches a card; if equal to zero, no card is output. If fourth argument is not equal to zero, it writes on tape B1; if equal to zero, no tape is output. Data is always printed on-line. Requires 25 storage locations.
F033--611102	Run Identification Load and Print	Loads run identification data into specified locations and prints on-line run identification information. Requires 75 storage locations.
F034--611102	Station Data Load	Loads station data into specified locations (allow six locations for each station data item). Requires 40 storage locations.
F035--611102	Station Data Search	Searches table for given station name and extracts data. Argument Z equals initial location of table. First two output locations contain station names (must be supplied); allow four additional consecutive locations for extracted data. If station name being searched for is not in table, zeroes will be placed in four locations. Requires 20 storage locations.

AOPB Number and Date	Function Name	Function Objective and Remarks
F036--611102	Ionosphere Data Load	Loads ionosphere data into specified locations (allow 15 locations for each item). If argument equals zero, it loads cards. If argument does not equal to zero, it reads from tape. A blank card or record signals end of data. One word of 99999999 is stored to signal end of table. Requires 45 storage locations.
F037--611102	Ionosphere Data Search	Searches table for station number and extracts data. Argument Z indicates initial location of table. First output location contains station number (must be supplied). Allow 20 additional consecutive locations for extracted information. Requires 30 storage locations.
F038--611102	Optical Station Number and Corresponding Code Load	Loads Optical station numbers and corresponding codes (names) into table. If argument equals zero, load cards. If argument does not equal to zero, read from tape. Blanks indicate end data and zeroes are stored in table. Requires 50 storage locations.
F039--611102	Optical Data Load	Loads optical data into table. If argument equals zero, loads cards; and if argument does not equal to zero, it reads from tape. Blank indicates end of data and zeroes are placed at end of table. Requires 55 storage locations.

AOPB Number and Date	Function Name	Function Objective and Remarks
F040--611102	Refined Optical Data Load	Loads refined optical data into table. If argument equals zero, it loads cards. If argument does not equal zero, it reads from tape. Blank indicates end of data and zeroes are placed at end of table. Requires 55 storage locations.
F041--611102	Right Ascension Mean Sun Data	Loads table with right angle mean sun data. If argument is zero, it loads cards; and if argument is not zero, it reads from tape. Blank indicates end of data and zeroes are placed at end of table. Requires 40 storage locations.
F042--611102	Right Ascension Mean Sun Data Search	Searches table for desired right angle mean sun. Argument Z equals initial location of table, and first three locations of output, year, month, day of desired R.A.M.S. rams will be in radians. If search is successful, it places word 99999999 in output location. Requires 20 storage locations.
F043--611102	Weight = (F, A, B, SD, SD*)	NOTE: Routines F043 and F044 are available from AOPB.
F044--611102	Heun Method Integration	
F045--611102	Angle Compatibility Package	Consists of angle relocation, subtraction and reducer, as well as scalar multiplication.
F047--611102	Angle Reducer	Places angle between 0 and 2 PI.
F048--611102	Angle Reducer	Places angle between - 2 PI and + 2 PI.

AOPB Number and Date	Function Name	Function Objective and Remarks
F049--611102	Range Rate	NOTE: Routines F049 through F052 are available from AOPB.
F050--611102	Absolute Value	
F051--611102	Drag Data	
F052--611102	Observation Load	
F053--611102	Ionosphere Refraction	
F054--620308	Matrix Inversion	Performs matrix inversion. Argument consists of number of rows in matrix to be inverted, number of columns, and elements of matrix by rows (in consecutive locations). Inverse is stored by rows in locations occupied by original matrix. Requires storage locations.
F055--620308	Word Load	Substitutes new value for value currently in storage location. One card for each value to be superseded. Blank card indicates end.
F056	Numerical Integration Position Partial Derivatives	NOTE: Routines F056 through F061 are available from AOPB.
F057--620724	Kepler (Revised)	
F058--620724	Geodetic Latitude and Height to Geocentric Latitude and Radius Vector	
F059--611015	Input Converter	
F060--630215	Output Constants for D.C.	
F061--630215	Output Constants for S.D.	

AOPB Number and Date	Function Name	Function Objective and Remarks
F062--630315	Interval Core Dump	NOTE: Routines F062 through F074 are available from AOPB.
F063--630315	Interval Core Dump Print	
F064--601015	Square Root Matrix Solution	
F065--601015	Fitting Function Partial	
F066--601015	Matrix Normalizer	
F067--601015	Matrix Clear	
F068--621015	Sub-Satellite Point and Height	
F069--611015	Round and Scale	
F070--600615	Lunear Equations Solutions	
F071--630703	BCD Output Plot	
F072--611015	Sunlight Determination	
F073--610130	Element Load (Conversion of Elements)	
F074--611015	GSFC Elements Print	

CONTENTS

CHAPTER 4 EXEC II PROCESSOR--1107

CONTENTS (Cont'd)

CONTENTS (Cont'd)

ILLUSTRATIONS

TABLE

CHAPTER 4

EXEC II PROCESSOR--1107

4.1 SYSTEM DESCRIPTION

This chapter describes the EXEC II System for the Univac 1107. The EXEC II System uses the SLEUTH II (Symbolic LanguagE for the Univac 1107 THin Film Computer) assembler and the FORTRAN compiler.

The Univac 1107 is the solid-state successor to the vacuum-tube 1105 and 1103 scientific systems. There is no program compatibility between the 1107 and its predecessors. However, programs written for the 1107 computer can be used interchangeably with its successor, the 1108 computer. See Paragraph 4.1.2 for configuration differences.

Although straightforward programming of the Univac 1107 is not unusually complex, it does take a seasoned programmer to be able to take full advantage of the powerful optional elements offered in most instructions.

The EXEC II System is an operating system designed to monitor the compilation and execution of programs, maximize utilization of available hardware, and minimize operator intervention. The system utilizes an FH-880 Magnetic Drum as a high capacity buffer store to keep the card readers, punches, and printers fully occupied and as a fast access auxiliary store for program segments. An integrated set of diagnostic aids and library maintenance facilities is included.

4.1.1 SYSTEM STRUCTURE

The EXEC II System processes jobs by means of control cards. In the simplest case, the input to the system consists of a RUN control card, a program deck, and source data cards. Various other control cards are used to punctuate this input. In general, a single run can construct programs from one or more source language processors (e.g., FORTRAN), previously compiled subprograms, and library retrievals; execute these programs (with data input cards if required); and produce a diagnostic output for debugging purposes. An inverted capital Greek delta ∇ in column 1 of the card (which consists of a 7-8 punch) identifies a control card. See Paragraph 4.2.1 for detailed specifications as to the form and content of these control cards.

User programs in the 1107 Monitor System are controlled primarily by means of a card deck. The primary control exercised by the operator over the Monitor System is by means of unsolicited key-ins (keys on the operator console). These key-ins are used to terminate the program, to cause the system to continue after having been delayed, to force a wait condition, etc. Symbionts, multiprogrammed routines, receive all of their control through the operator's keyboard.

The flow diagram (Figure 4-1) illustrates the various functional components of SLEUTH II and the paths of control between them. The first loading of program instructions is accomplished by using the initial load bootstrap routine (EXEC II Load Tape (COSM)). The first 224 addresses in core storage are reserved for this bootstrap routine which serves to bring in the remainder of the resident and various other parts of the system. The bootstrap routine also provides a simple card load routine, a panic dump (simple dump routine), and a method of patching the resident system prior to writing it to drum.

At each bootstrap from tape, a short routine is executed which makes a simple check of the hardware.

Tape assignments within SLEUTH II are entirely at the option of the programmer and, for production work, the selection of the tape unit to be used for a particular reel is left to the discretion of the operator. The assignment of multi-reel files is accomplished through the use of an operational label.

4.1.2 EXEC II 1107-1108 CONFIGURATION DIFFERENCES

Univac 1108 jobs are sent to the more available computer unless the programmer states a particular machine is to be used. The 1108 EXEC II is quite similar to 1107 EXEC II, and most programs should run under both systems with no change. There are, however, several differences between EXEC II on the 1107 and 1108. These are discussed below. If any others are detected, programmers should contact one of the system programmers. The configuration of the 1108 is as follows:

1) Channel 0--6 FH432 drums; Channel 1--one FASTRAND II unit; Channels 2, 3, and 12--VIIIC tape drives, 4 to a channel; Channel 13--1004 card reader and printer; Channel 14--1004 card reader, punch, and printer; Channel 15--console.

2) As on the 1107, character-count errors on tape input may be ignored by use of the TACE$ routine. However, the last word of such records is handled differently. On the 1107, the remainder of the last word is filled with the number of characters read. On the 1108, the remainder of the word is zero filled, and the character count is not available. For example, if we are reading a record of 8 characters, the second word of the buffer would appear as xx2222 on the 1107 and xxØØØØ on the 1108 (where xx are the last 2 characters of the record).

3) The mnemonic dump capability has had to be deleted, at least temporarily. Any requests for mnemonic dumps will be honored with octal dumps.

4) The resident of the 1108 EXEC II System takes up 12K, whereas for the 1107 it takes up only 8K. Since the total available core space on both machines is the same, this means that some large programs may have to be pared down to run on the current 1108 EXEC II System. EXEC VIII will be a 12K system, also.

5) On the VIIIC tape units, end-of-files will be read into the first word of the input buffer. The end-of-file will appear as an octal 17 in the leftmost six bits. The end-of-file will be recognized as such, and control will be transferred to whatever end-of-file routines are used by the programmer. The remainder of the input buffer will not be destroyed.

6) The VIIIC tape drives have three densities: 200 (L), 556 (H), and 800 (X) bits per inch.

7) The blank specifications field on the PMD card will not be honored. The programs or areas of core desired to be dumped must be stated explicitly. This modification will probably be added to the 1107 at a later date.

8) Any parity error on a 1- or 2-word tape record will be treated as noise during a read and will be automatically disregarded.

9) The Trace Mode does not exist on the 1108, thus any routines using the Trace Mode will not execute. Users of the TRACE routine will be returned to their programs immediately upon attempted execution of any of the TRACE entry points (TRC, TRC$, etc.).

Anyone expecting to continue coding on the 1108 should remember, however, that the EXEC II is only a temporary expedient; and it will eventually be replaced by EXEC VIII, which will be considerably different.

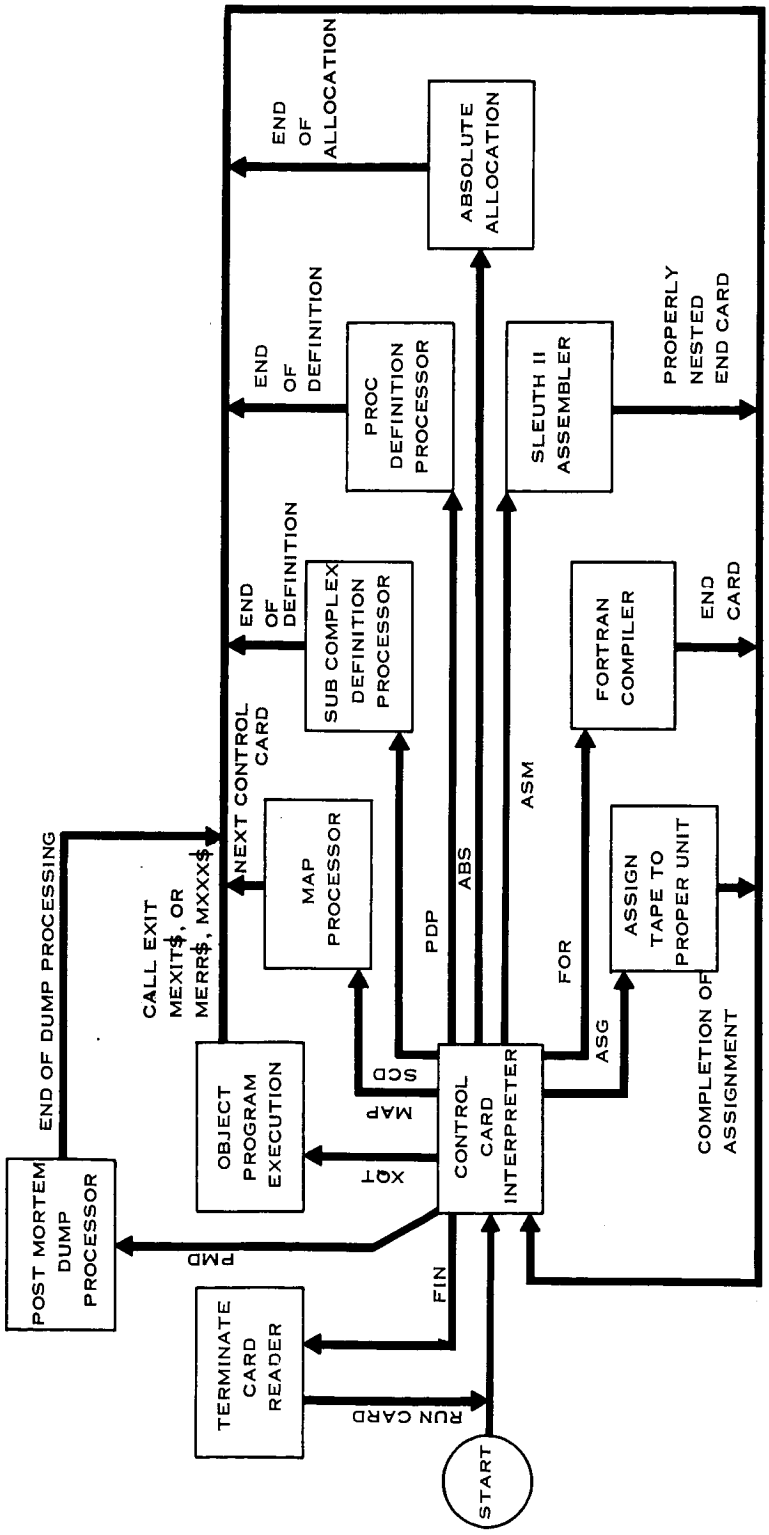


Figure 4-1. Flow Diagram of 1107 SLEUTH II

4.1.3 MACHINE CONFIGURATION

The programmer using the Goddard Space Flight Center computer complex has a vast quantity of data processing equipment at his disposal. The EXEC II System operates on two of these large scale computers. A list of this equipment is given in Table 4-1.

Table 4-1. Major Computer Equipment

Bldg. Loc.	Computer Facility	Memory Size	Magnetic Tape Units	Line Printer	Drum	No. of Data Channel	Card Reader	Card Punch
14	1107G	65K	24	1	2	16	1	1
14	1108H*	65K (36-bit words)	12	2	6	16	2	1

*1108 also has available a FASTRAND II random access mass storage device.

4.1.4 SYSTEM TAPE MAINTENANCE

The Programming Methods Section (PMS) of the Data Systems Division has primary responsibility for maintaining the EXEC II System master tape. Tape revisions or updating occur periodically as a result of one or more of the following conditions: 1) new version or significant corrections issued by Univac; 2) major changes originating from GSFC programmers; and 3) catastrophic errors requiring the immediate issuance of new tape.

In certain cases, when errors are of a minor consequence or unique to a particular application and immediate release of a new master tape is not warranted, the PMS provides binary decks to circumvent the error condition. With the release of new master tapes, the decks are subsequently discarded by the programmer.

4.1.5 ERROR REPORTING

The PMS has the responsibility of maintaining the EXEC II System. Any questions regarding system utilization and system discrepancies should be directed to PMS personnel. The current method of reporting system discrepancies verbally is expeditious. However, it is recommended that the Systems Discrepancy Report (see Form 1-1., Chapter 1) be used for submittal to the PMS coordinator. In this way, a current file of all discrepancies will be maintained along with the corrective actions taken. A copy of the Discrepancy Form will be available in the PMS coordinating office (Room 127, Building 3) and in the dispatcher's office (Buildings 3 and 14). Programmers are required to periodically check the System Status Report (see Form 1-2., Chapter 1) to insure satisfactory operational performance from the system used.

4.2 DETAILED PROCEDURES

This section includes several illustrations showing job deck composition for a number of typical runs; presents a description of the control cards and their use; and offers means by which to use the system effectively.

The Master Space Character, ∇, (which consists of a 7-8 punch) in Column 1 of the card identifies a control card. (This should not be confused with the blank space.) Options, if desired, are noted by a string of letters and are punched beginning in Column 2 of the control card. A string of option letters is terminated by a blank Column. Column 2 must be blank if no options are specified. The control card mnemonic appears next on the card. These mnemonics consist of three letters, and they must be one of the set of 17 as described in this paragraph. A string of one or more blanks must precede the mnemonic. In some cases, the mnemonic may be followed by a comma and then a single character. If this is so, a blank must follow that character; otherwise, a blank must follow the mnemonic.

The remainder of the control card contains specifications for the system routine involved. They are again preceded by a string of one or more blanks. Since the specification portion varies considerably from one control card to another, a general discussion is appropriate.

In summary, a control card has the form:

∇options xxx, λ specifications

where options represent the option letter string; xxx, the control card mnemonic; λ, a single character; and specifications, the specification portion.

The EOF or FIN control cards used by the card input system are interpreted at interrupt time, rendering the free form described above impractical. These two cards must be of the form:

Column 1	∇
Column 2	blank
Column 3, 4, 5	EOF or FIN
Column 6	blank

If a malformed control card is encountered, the card itself is printed, followed by the message:

ABOVE CONTROL CARD IN ERROR--IGNORED

In the case of premature termination of a run, due to an error, any control card other than a PMD (see Paragraph 4.2.1.1 (4)) will produce the meassage:

REMAINING CONTROL CARDS IGNORED

If the system is expecting a control card, but encounters one or more cards not containing a ∇ in Column 1, the following message is printed:

DATA CARDS ENCOUNTERED BY SYSTEM--IGNORED.

4.2.1 CONTROL CARDS FORMAT AND USAGE

This paragraph presents in detail the description of the control cards that exercise control over the EXEC II System. The user controls and directs the processsing of his job by inserting the proper control cards in the job deck, thereby directing the operating system to perform any one of several operations.

There are 19 control cards that fall into four categories, namely, System Control, Processor Control, Allocator Control, and Card Input Control. These control cards are detailed in Paragraphs 4.2.1.1 through 4.2.1.4.

4.2.1.1 System Control Card Description

The system control cards are summarized in Figure 4-2. A detailed description of the control cards and their use is provided.

SEE PAR. 4.2.1.1	MNEMONIC	APPLICATION	OPTIONS
(1)	RUN	To initiate each computer run.	(priority)
(2)	ASG	To cause assignment of magnetic tapes	ACEFHKLORX
(3)	MSG	To type message to operator	HN
(4)	PMD	To cause memory printout after execution	ABCDEIQX
(5)	ELT	To introduce an element into the program complex file from cards	None
(6)	HDG	To give heading information for printer output from the run	NP
(7)	TPR	To specify which I/O device--magnetic drum or tape--is to be used to store intermediate Fieldata images for the printer symbiont.	S
(8)	DPR	To route printer output to drum storage.	None

Figure 4-2. System Control Cards Summary

An individual description of the System Control Cards is as follows:

(1) RUN Control Card: (Variable Field Format)

1		80
▽		RUN identification, account, running time, print output, print output channel, punch output channel

A RUN control card precedes each job to be performed. If the system encounters a control card other than the expected run control card, it prints the following message:

RUN CARD MISSING--DECK NOT ACCEPTED

NOTE: Priority which may appear on a run card should not be used by the programmer. Priority will be assigned by the dispatcher.

The identification field should contain the installation's conventional run identification. Both it and the account field may consist of up to six characters taken from the set A ... Z, 1 ... 9, +, =, ., $ Blanks are illegal.

The running time field is an estimation of the running time in minutes for the problem. Should this time be exceeded, the operator will be informed and the run continued. It is the operator's responsibility to terminate the run. The print output field is an estimation of the print output in pages required for the problem. Should this estimation be exceeded, the operator should be informed and the run completed. It is the operator's responsibility to terminate the run when estimated running time or print out is exceeded. The running time field may be omitted, in which case, the system assumes a maximum time of five minutes; likewise, the output field may be omitted and a maximum of 50 pages will be assumed.

(2) ASG Control Card: (Variable Field Format)

1		80
▽		ASG assignment, assignment, ...

The ASG control card will associate a logical unit designation with a scratch tape or with an operational label. The option filed may contain the following letters (in any order):

A--accepts character count errors (1108 only)
H--density set high (556 bpi)
L--density set low (200 bpi)
X--density set to 556 bpi (1107) or 800 bpi (1108)

If no density setting is indicated, it will be assumed to be high
O--parity set odd
E--parity set even
If no parity setting is indicated, it will be assumed to be odd
R--rewind
C,K--causes automatic Fieldata to BCD conversion on output and BCD to Fieldata conversion on input (C is software, K is hardware)
F--turns BCD conversion off

Any option letter appearing on an ASG control card will apply to all assignments made on that card. An attempt to assign the same logical designation more than once will produce a message on the printer, but will accept the last assignment made. No harm results from assigning more than one logical designation to the same operational label. Any number of ASG control cards may appear in a program; their effect is cumulative.

It is sometimes desirable for the user to specify absolute tape assignments channel/unit (c/u). This feature eliminates the need for the A key-in and "MOUNT" message but requires the operator to mount the tapes where and when specified by the user. When absolute assignments are made, the ASG card takes the forms

∇	ASG	c/u,c/u,...		(Type I)
∇OPTIONS	ASG	assignment		(Type II)
∇OPTIONS	ASG	c/u,c/u,...	assignment	(Type III)

A card of Type I immediately followed by a card of Type II produces the same results as a card of Type III. Only one "assignment" field is allowed per card. A maximum of 10 cards of Type I collectively, specifying a maximum of 10 channel/unit pairs, can be grouped together. Cards of Type I must immediately precede one card of Type II. The "options" field is meaningful only on those cards containing an "assignment" field. A space must follow the last unit number on cards of Types I and III.

All tapes at the end of a job are automatically rewound, except when a dollar sign ($) is encountered as the last character of a tape label.

(3) MSG Control Card: (Variable Field Format)

1			80
∇	H	MSG	message to be typed

The MSG control card is used to type a message to the operator. The

typing will be prefaced by the line MSG and will commence with the first non-blank character of the specifications field. If the options filed contain an H (as in the example), the operator will be given the opportunity to execute or scratch the run by causing a wait loop. Typing may be suppressed (resulting in the printing of the MSG control card on the printer only) by an N option. In this case, the H option is not effective.

(4) PMD Control Card: (Variable Field Format)

1		80
▽	C	PMD specifications

The PMD (Post-Mortem Dump) control card may be used to dump core memory following the execution of an object program. Dumps may be made of segments, elements, or specified parts of elements, as long as they were currently in the memory at the time the routine was terminated. Several options are available for output formatting, and core areas to be dumped. The option C (in the example) will cause a dump of the words that were changed during the execution of the allocated program for the area of core prescribed by the PMD card. In the event no DUMP is available, the message

NO PROGRAM EXECUTED--NO DUMP TAKE

is printed and the PMD card is ignored.

The option field may contain the following letters:

E--causes the PMD card to be processed only when the previous routine terminated at systems error exit MERR$.

C--causes a dump of the words that were changed during the execution of the allocated program for the area of core prescribed by the PMD card.

B--causes, after processing the rest of the PMD card, an octal dump of all of blank common storage area. If used with the C option, it will be ignored. (Changed word dumps of blank common may be taken when requested as a location counter under the blank option (See UNIVAC 1107 LIBRARY II, Section 8.2.2).

Q--used for absolute dumps of arbitrary areas of core.

M--obtains a mnemonic Post Mortem Dump. This option will be overridden if any format other than 8014 is specified or if a Change Word Dump is requested.

(5) ELT Control Card: (Variable Field Format)

1	2 ... 80
▽	ELT name/version (flag), type, date time

The ELT control card introduces an element into the current program complex file from punched cards. The ELT is always followed by the cards containing the element. Name/version is the name or name and version to be associated with the element. The field type gives the type number of the element. If omitted, Type 1 (source language is assumed. The date and time fields identify when the element was created or last altered. The date is punched as a six-digit decimal number of the form yymmdd, and the time is punched as the decimal number of seconds from midnight. If these fields are omitted when an ELT is read, the current date and time will be entered into the element table. When an element is punched by a processor or by CUR, it is always preceded by a suitable ELT control card. Such decks can simply become part of the input to subsequent runs. The complex utility routine (see UNIVAC 1107 EXEC II Programmer's Guide U-3671) is called into play when an ELT card is encountered. No option letters are associated with an ELT control card.

(6) HDG Control Card: (Variable Field Format)

1	2	80
▽	N	HDG heading line (columns 13-72)

The HDG control card is used to give heading information for printer output from the run. The Heading control card must precede any Processor control cards in the run. A page number and current date appears to the right of the printed heading. Options are:

N--Turn off page heading on print output
P--Reset page count

(7) TPR Control Card: (Variable Field Format)

1	2	80
▽	S	TPR label

The TPR control cards may be used by the programmer to specify which I/O device--printer or magnetic tape--is to be used as the standard output device. Since these cards override Print Cooperative Key-Ins, they must be used with caution.

The option may be an S (as in the example) to permit switching to the next tape in the string of assigned .PR tapes at the beginning of the run in which the TPR card is encountered. The label field may contain up to six alphanumeric characters, if desired. The first word of each tape block contains this label. The TPR card must immediately follow the RUN card. .PR tapes may be printed on the 1107 and 1401.

See Paragraph 4.5.2.2 for further information on off-line print routines.

(8) DPR Control Card: (Variable Field Format)

1		80
∇		DPR

The DPR control card is used to route printer output to drum storage. There are no options or labels.

4.2.1.2 Card Control Control-Card Description

The card-control control cards are summarized in Figure 4-3. A detailed description of the control cards and their use is provided.

SEE PAR. 4.2.1.2	MNEMONIC	APPLICATION	OPTIONS
(1)	EOF	To punctuate a data check.	None
(2)	FIN	To mark the end of a card stream. Not normally required by the programmer	None

Figure 4-3. Card Control Control-Card Summary

An individual description of the Control Card control-cards is as follows:

(1) EOF Control Card: (Fixed Field Format)

1	2	3	4	5	6	7 ... 80
▽		E	O	F		

The EOF control card is used to punctuate a data check. Column 7 may contain any character, and Columns 8 through 80 are ignored. There are no options. On encountering an EOF card, the subroutine exits to the abnormal return with the character in Column 7 located in A0.

(2) FIN Control Card: (Fixed Field Format)

1	2	3	4	5	6	7 ... 80
▽		F	I	N		

The FIN control card marks the end of a card stream. It is not normally required by the programmer. There are no options.

4.2.1.3 Processor Call Control-Card Description

The Processor Call Control Cards are summarized in Figure 4-4. A detailed description of the control cards and their use is provided.

SEE PAR. 4.2.1.3	MNEMONIC	APPLICATION	OPTIONS
(1)	ASM	To call out Assembler	AILNPSWXZ
(2)	FOR	To call the FORTRAN Compiler	ADILNPSTWXZ
(3)	MAP	To call out the memory allocation processor	AINPSWX
(4)	PDP	To call out the procedure definition processor	AXLIS

Figure 4-4. Processor Call Control-Card Summary

An individual description of the Processor Call control cards is as follows:

(1) ASM Control Card

1	2	3	4 80
∇			ASM loc n_1/v_1, n_2/v_2, n_3/v_3 (flag)

The ASM control card calls out SLEUTH II assembler. There are nine options for Column 2. (See UNIVAC 1107 EXEC II, Section V.) Loc gives the location of the input: omitted, input from cards; *, input from the complex on drum; an alphabetic character, input from corresponding logical tape unit. The specification n_1/v_1 is the name or name/version source language element, if one exists; n_2/v_2 is the name or name/version of the updated source language element, if one exists. (If this field is omitted no updated source language is placed in the complex.) The n_3/v_3 is the name and version to be applied to the relocatable element code resulting from this processing (if omitted, the relocatable element will have the same name as that of the updated source language element, and a version name CODE). Flag is a string of alphabetic characters enclosed in parentheses, giving the flags to be associated with the newly created relocatable code (if this is omitted, the flag associated with the relocatable code is taken to be all zeros).

(2) FOR Control Card

The FOR control card calls out the FORTRAN compiler. (Same general rules as in (1).)

(3) MAP Control Card

The MAP control card calls out the memory allocation processor. (Same general rules as in (1).)

(4) PDP Control Card

1	2	3 4 80
∇		PDP, loc n_1/v_1, n_3/v_3

The PDP control card calls out the procedure definition processor. The PDP accepts source language defining SLEUTH II procedures and builds an element to be included in the program complex file. There are five options for Column 2. (See UNIVAC 1107 EXEC II, Section V.) The specification loc indicates whether the element is to be taken from cards (loc is omitted); from drum (*); or from tape (alphabetic character referencing the proper tape unit). The n_1/v_1 is the name and version of the source input element, and may be either a source language element (type 1) or a procedure element (type 7). The

n_2/v_2 is the name and version to be applied to the procedure element code resulting from this processing (if omitted, the procedure element produced by PDP will be entitled n_1/CODE).

In the options field of an ASM or FOR card, one or more of the following letters may appear: A, I, L, N, P, S, X, Z. Each of these letters represents a single option:

A--accept the results of the processing as correct, even though errors were detected.

I--single space listing without relocation information.

L--produce a complete printed listing.

N--suppress all printing by the processor.

P--punch the resulting relocatable element into cards.

S--punch the (updated) symbolic language in compressed form. If both P and S are specified, the compressed symbolic deck will appear first.

X--abort the remainder of the compilation if any errors are detected by the processor.

Z--suppress the formation of information to be given the diagnostic system.

Ordinarily, the L and P options will be used. If no P option appears, no punched deck will be output. If the L option is dropped, a listing of the symbolic input deck will be output without the corresponding assembly. If the N option is used, no listing will be output unless errors are detected in the program.

An additional option letter W will list all correction cards, if any. In FORTRAN, the option letter T will return the compilation time.

4.2.1.4 Allocator Control Control-Card Description

The Allocator control cards are summarized in Figure 4-5. A detailed description of the control cards and their use is provided.

The allocator is a system routine which collects subprograms and interconnects them. Cross-reference between these subprograms are resolved and relative locations are assigned. Depending on how the allocator is called, it may go on to produce an absolute program which is put away on drum. Finally, this absolute program may be loaded into core and run.

The allocator will also, unless requested otherwise, construct a group of tables which serves as one of the inputs to the diagnostic system. If desired, the allocator can work with the output of the memory allocation processor in order to provide a flexible and sophisticated segmentation ability. In the absence of a map, the allocator assumes that all subprograms and common blocks will occupy core simultaneously.

Three separate control cards, XQT, ABS, and SCD, result in calling the allocator. These cards cause, respectively, the allocator to construct (if necessary) an absolute program and execute it; to construct an absolute program and put it into the user's drum PCF; and to construct a relocatable program and put it into the user's drum PCF.

SEE PAR. 4.2.1.4

	MNEMONIC	APPLICATION	OPTIONS
(1)	XQT	To execute a program (including allocation, if required)	ACLNXZ
(2)	ABS	To produce an absolute program	ACLNXZ
(3)	SCD	To define a subcomplex	ACLNPX

Figure 4-5. Allocator Control Card Summary

An individual description of the Allocator control cards is as follows:

(1) XQT Control Card: (Variable Field Format)

1			80
▽		XQT, λ	name/version,i

The XQT control card is used to execute a program (including allocation if required). The λ is the letter to be used in selecting elements for the basis of their flags. There are six options for Column 2. (See UNIVAC 1107, EXEC II, Section V.) The name/version is the name or name and version of the program to be executed. The i, when present, is a decimal integer indicating how many times to go through the FORTRAN error routine, NERR$, before terminating.

(2) ABS Control Card: (Variable Field Format)

1			80
▽		ABS, λ	name/version, name/version

The ABS control card is used to produce an absolute program, as it does for the XQT control card. In this instance, however, the resulting program will not be executed, but instead will be entered into the program complex file as an absolute element. The second name/version field becomes the name of the element.

(3) SCD Control Card: (Variable Field Format)

1			80
▽		SCD, λ	name/version, name/version

The SCD control card is used to define a subcomplex. This card behaves much as does the ABS control card. A relocatable element is inserted into the complex, rather than an absolute element. To be useful, a SCD control card will usually require a map. For details, see UNIVAC 1107 EXEC II, Section VI.

In the options field, one or more of the following letters may appear: A, C, L, N, X, Z. Each of these represents a single option:

A--accept the results of processing as suitable for execution even though errors were detected.

C--make the following patches.

L--Produce a complete allocation listing and map.

N--Produce no allocation listing.

X--abort. Do not execute if errors detected.

Z--suppress the formation of information to be given the diagnostic system.

The A option will be overridden if the errors are sufficiently serious to prevent execution. If neither the L nor N option is specified, a partial listing will be produced. The N option will be overridden if diagnostics are produced.

4.2.1.5 Programming Procedures

This paragraph will describe basic programming procedures for the Univac 1107 computer. For detailed information, consult the UNIVAC 1107 EXEC II Programmer's Guide, U-3671, and the Library II Programmer's Guide, U-3672.

To communicate with the 1107 monitor system, the following deck set-up is required:

```
∇ RUN identification, account, running time, output
∇options FOR program name
 FORTRAN IV source program
∇options ASM program name
 SLEUTH II source program
∇options XQT program name
 Data Cards
∇ EOF
 Data Cards
∇options PMD specification field
∇ FIN
```

The program name on a FOR or ASM card need not be the same as the actual name of the routine but is limited to six characters. The program name on the XQT card is the name of the first program to be executed, and this name must appear on a FOR, ASM, or ELT card. If only a compilation is desired, the XQT card is omitted.

The PMD card is discussed under Debugging in Paragraph 4.2.3.2.

4.2.2 PROGRAMMING AIDS

This paragraph presents to the programmer tips and techniques, as well as precautions to be considered while performing programming functions. These programming aids should be of help to the programmer in his utilization of the EXEC II System.

4.2.2.1 Debugging

For programs written in SLEUTH II, several procedures and programs are available which will aid the programmer in debugging.

a. In most of the dump procedures, a format must be specified. The standard formats are:

'M'		standard octal format plus mnemonics*
'F'	(8F14.8)	fixed decimal
'E'	(8E14.8)	floating decimal
'I'	(8I14)	integer
'A'	(16A6)	alphanumeric
'@'	(8@14)	octal

*There is no mnemonic dump on the 1108. An octal dump without mnemonics will be given instead.

Any message up to 120 characters may be produced on the Printer by use of the X$MESG procedure:

```
X$MESG    k
'diagnostic message'
```

where k is the number of words in the diagnostic message.

b. A dump of thin film may be produced by the X$FILM procedure:

```
X$FILM    start, length, format
```

which results in length film locations beginning at film location start being edited and printed according to format as described above. Start and length locations may be either symbolic, decimal, and/or octal addresses.

c. The X$CORE procedure dumps core memory:

```
X$CORE    start, length, format
```

which results in length core locations beginning at core location start being edited and printed according to format as described above. Start and length locations may be either symbolic, decimal, and/or octal addresses.

d. The X$DUMP procedure dumps thin film and core memory:

```
X$DUMP    start, length, format, registers
```

which results in a panel dump of the current state of the machine, B, A, and/or R registers and a printout of core memory where registers, printed in octal, are specified by the letters B, A, or R, or any combination. "length" core locations are dumped beginning at core location "start"and edited and printed according to the format as described above. If registers are not desired, the field should be left blank. Start and length locations may be either symbolic, decimal, and/or octal addresses.

e. A dump may be specified at the end of the program. This is specified by the PMD control card:

∇options PMD specifications

where ∇ is the 7-8 multipunch.

Options include:

C--changed-word dump

E--dump executed only if program exits through error

B--will cause octal dump of Blank Common

A--dump of Bank 2 named in spec. list*

I--dump of Bank 1 named in spec. list*

*Only one of these may be specified in a single PMD card.

X--if A, D, or I specified,dumps everything except that which is named in the spec. list.

M--standard mnemonic dump. (Does not work with C option.)**

**There is no mnemonic dump on the 1108. An octal dump without mnemonics will be given instead.

Specifications include:

a. If no A, D, or I are specified, the spec. list must have the form:

name, start, length, format

name is the name of an element; start is of the form n$m, where n is an address relative to location counter m; length is the number of words to be dumped in format.

b. If A, D, or I are specified, the spec. list must have the form:

name 1, name 2, name 3 (etc.)

where name X is an element or segment name.

c. An unconditional dump governed by the specification is given if there are no options. If there are no options or specifications, no dump will be taken.

f. PDUMP, FPDUMP, and DUMP take storage dumps as specified by arguments (A,B,I), restore the condition of the machine, and return to the program which called them.

The general form of the PDUMP statement is:

CALL PDUMP (A_1, B_1, I_1, ... A_n, B_n, I_n)

where A and B are variable data names indicating limits of core storage to be dumped. Either A_i or B_i may represent upper or lower limits. I_i is a FORTRAN integer indicating the format desired, as follows:

I = 0	
I = 1	dump in floating point
I = 2	interpret as decimal integer
I = 3	dump in octal with mnemonics

If a PDUMP CALL is made with no arguments, the entire user's program core area is dumped in octal.

If the last format in a string of arguments is omitted, the area of core between the two specified dump locations will be dumped in octal.

When PDUMP is executed, the machine is restored to its condition upon entry, and control is returned to the next executable statement. The storage dumps appear on the printer with other dump output from the job.

4.2.2.2 1107 Item Advance Routines

The item I/O routines relieve the programmer of the task of item handling chores when designing a SLEUTH II program. Routines are designed by procedure calls which generate object coding in the user's program.

(1) Input Procedure Files:

Function AnOPEN--Open input file n. Routine reads list block, transfers first item (contains label) to work-storage (item-ws). Initiates a read of the next block (data) into the input area (block-storage). (Reads two blocks of data for buffered option.)

Function AnREAD--Read input item from file An. Transfers current input item to input working storage (item-ws). Initiates refill of the input area (block-storage) after transferring last item.

(2) Output Procedure Files:

Function BnOPEN--Open output file Bn. Routine transfers one item (label) from work storage (item-ws) to the output area (block-storage) and initiates a write of this block.

Function BnWRIT--Write output item of file Bn. Transfers current item from working storage (item-ws) to the output area (block-storage). Initiates a write of one block when the output area is full.

Function BnCLSE--Close output file Bn. Fills any remaining output area (block-storage) with any specified sentinel, and writes this block followed by one EOF mark. CALL this routine twice in succession to insure two consecutive EOF marks.

For more information, contact Mrs. Pat Barnes of the Programming Methods Section, Advanced Projects Branch, Building 3, Room 127, Extension 6796.

4.2.2.3 1107 Analyzer

The programmer should find the object program analyzer very useful in debugging operations. This routine is in the standard systems library, and it will analyze a selected area for all cross references (XREF) within a group of selected areas. Output is an HSP listing, giving the address and instruction of XREF's. Constants that can be recognized as such are edited in octal. Some constants, however, are indistinguishable from instructions and will be so edited. For more information, contact Mrs. Pat Barnes as noted in Paragraph 4.2.2.2.

4.2.2.4 1107 Editing Routine

This library routine alleviates the programmer of attending to tedious editing considerations when printing on-line. By describing a printed page via a parameter list, the main program need only communicate with the routine when a line of data is to be printed. All responsibilities for spacing, headers (single or multi-line), page count, and paper-feeding (PLINE$) are assumed by the editing routine. For more information concerning this, contact Mrs. Pat Barnes as noted in Paragraph 4.2.2.2.

4.2.2.5 Save and Restore Routine

The routines to save and restore all A and B registers have been incorporated into the system library. For more information, contact Mrs. Pat Barnes as noted in Paragraph 4.2.2.2.

4.2.2.6 Data Generator Routine

This routine can be used to alter data information in core to assist in realistic program checkout. The data image is assumed to be in core initially. The data may be dynamically altered by the programmer between calls. For more information, contact Mrs. Pat Barnes as noted in Paragraph 4.2.2.2.

4.2.2.7 Diagnostic Trace (SNOOPY)

SNOOPY is a diagnostic library routine designed to assist in program checkout. SNOOPY will printout film information (as per options) whenever a successful skip or jump instruction occurs. Calling SNOOPY results in the location begin address being preserved and a link to SNOOPY being inserted. When this instruction is executed, SNOOPY takes control, restores the link, and snoops until end add, where it returns control.

The programmer may take snapshot dumps at any instruction by coding a list of dump parameters. The dump will be taken following the execution of the instruction. The number of such parameter pairs is unrestricted and is specified by # dumps. For information, contact Mrs. Pat Barnes as noted in Paragraph 4.2.2.2.

4.2.2.8 Label Check Routine

This routine is developed to provide the programmer with a standard label checking ability. For more information, contact Mrs. Pat Barnes as noted in Paragraph 4.2.2.2.

4.2.2.9 Trigonometric Functions

For compatibility with the FORTRAN system in use on the IBM 7094's within the Data Systems Division, the following function subprograms have been added to NASA UNIVAC 1107 EXEC II Library:

Function ACOSD (y)--computes principal value of Arc Cosine y in degrees.

Function ASIND (y)--computes principal value of Arc Sine y, in degrees.

Function ATANGD--(y,x)--for signed inputs y, x computes angle between 0° and 360° whose tangent is y divided by x.

Function COSD (a)--gives Cosine of input angle expressed in degrees

Function SIND (a)--gives Sine of input angle expressed in degrees

Function ATANGR (y,x)--for signed inputs y,x computes radian angle between 0 and 2π whose tangent is y divided by x.

Function ACOSR (y)--computes principal value of Arc Cosine y, in radians.

Function ASINR (y)--computes principal value of Arc Sine y, in radians.

Anyone desiring more information on these routines should be directed to Mrs. Pat Barnes of the Programming Methods Section, Advanced Projects Branch, Building 3, Room 127, Extension 6796.

4.2.2.10 Trace Routines

The trace routine is a blend between a software and hardware trace. Software components are necessary to allow for restoring register R3 after a trace interrupt occurs. They also permit establishing the locations of all executed JMGI/MOJP (which are not provided by a trace interrupt) and of all instructions performing a skip. The hardware components of this routine are used solely for detecting that a jump instruction was executed, for obtaining the address to which control was to be transferred, and for returning control to the trace routine. See An Improved Approach to Trace Routines X-545-65-115, Goddard Space Flight Center. (No trace routines on 1108.)

4.2.2.11 SETEOF

A FORTRAN programmer may now return to his program upon reaching an EOF in formatted input. Prior to instituting his read, the programmer should

CALL SETEOF ($n)

where n is a statement number in the calling program to which the programmer wishes to go upon reaching an EOF.

4.2.2.12 S-C 4020 Package

The Lockheed S-C 4020 Package is on the system library. There are a number of GSFC modifications to this package; details are obtainable from Mrs. Pat Barnes, Programming Methods Section, Advanced Projects Branch, Building 3, Room 125, Extension 6796.

4.2.2.13 MOVER

This routine extends the re-read feature of FORTRAN IV. Its calling sequence is

CALL MOVER (X)

where X is the location of the image which the programmer would like to process on his next reference to the re-read unit, which is FORTRAN logical unit 0 in the standard FORTRAN I/O table. This allows a programmer to re-read a given set of data at any time in his program, rather than only immediately after it is read. It should be noted that any read statement will reset the reference, so that no-read statements should lie between the CALL to MOVER and the re-read statement.

4.2.2.14 TACE

This routine allows FORTRAN programmers to ignore the character-count-error interrupt (interrupt code 70_8) on input tapes. Its calling sequence is

CALL TACE (ITAPE)

where ITAPE is the FORTRAN logical unit on which the interrupt is to be ignored.

4.2.2.15 JFACTO

The following procedure may be called at or near the beginning of the user's program:

JFACTO name(1) name(2) name (3)

Specifying a name(1) will determine which set of mnemonics is defined internal to the calling program. These are summarized in Table I of the 1107 Programmer Note 3, Data Processing Branch, Information Processing Division, T&DS.

4.2.2.16 Tape Transfer Subroutine

If a SLEUTH II program wished to reference tape drives by some logical unit letter designatorα, but the tape drives had been previously assigned to $\beta(\alpha \neq \beta)$, those tapes were heretofore unavailable. This subroutine allows all tape drives associated with α to be referenced by β, and inversely.

The calling sequence

```
LMJ     11, TXFR$
        +'αβ'
```

(where α and β are any valid logical unit designators) will attempt to interchange tapes associated with α and β. If transfer is permissible, register 12 (A$\emptyset$) is positive upon return to user; if not, register 12 is negative. Transfer is permissible if all of the following are met:

(1) logical unit designators are valid
(2) no tapes affected by the transfer are assigned to EXEC II proper or the parasites (NOTE: unassigned tapes and tapes assigned to the user are treated identically)
(3) the current tape operation has been completed (NOTE: if an operation is in process, TXFR$ will wait for completion and then perform the interchange)

4.2.2.17 TUTIL

This program provides the user with various tape operations which are executable at arbitrary points in his run deck. The utility program is called by the card

```
VN   XQT   TUTIL
```

Following this card is a stream of cards specifying the tape operations to be performed. This operations card stream is terminated by any non-EOF control card. For details, see 1107 Programmer Note 6, obtainable from Mrs. Pat Barnes, Building 3, Room 125.

4.2.2.18 PSWTCH

There are two methods the programmer may use to determine the state of an indicated program switch. The first is by the subroutine call

```
CALL PSWTCH (n, variable name)
```

where n is a program switch ($1 \leq n \leq 36$) and variable name is a variable which is to receive the integer value 1 if switch n is off, or 2 if n is on. The second is by the function call

```
PSWTCH (n)
```

where n is a program switch ($1 \leq n \leq 36$) and the function will receive the integer 1 if switch n is off, or 2 if n is on. In both cases, if n has been previously specified or is out of range, the standard FORTRAN error exit will be taken. See 1107 Programmer Note 4, obtainable from Mrs. Pat Barnes, Building 3, Room 125.

4.2.2.19 CalComp Routines

(1) CPLOTS

CPLOTS must be called before any of the other CalComp routines are used. It need be called only once. The main purpose of this routine is to set up a buffer area in which to store the CalComp commands.

(2) CCPLOT

This subroutine will write a tape that is readable by the CalComp 570 Plotter, draw graphs, maps, plots, etc.

(3) SYMBOL

The purpose of this routine is to create commands that will produce alphabetic characters, numerals, and other selected symbols on the CalComp 570 Plotter.

(4) PACRAT, PAKRAT, SHIFTY

The purpose of these three routines is to generate input data for the SYMBOL routine. However, they may be found useful for other purposes. PACRAT packs Fieldata characters stored one to a word, left-justified, into consecutive locations, such that there are 6 Fieldata characters per word. PAKRAT stores FORTRAN integers one to a word, left-justified, into consecutive locations, such that there are six Fieldata characters per word. SHIFTY shifts a FORTRAN integer so that the least significant six bits are stored in the left-most six bits of another word.

4.2.2.20 Standard FORTRAN I/O Table for Univac 1107/1108 EXEC II System

FORTRAN programs using I/O devices will automatically call on the standard I/O table, provided the user has not specified one of his own. The standard I/O table is listed in Table 4-2.

Table 4-2. Standard FORTRAN I/O Table for Univac 1107/1108 EXEC II

FORTRAN Logical Unit Number	Specific I/O Device
0	Reread Unit
1	B
2	C
3	D
4	E
5	F
6	G
7	H
8	I
9	J
10	K
11	L
12	M
13	N
14	O
15	P
16	Q
17	R
18	S
19	T
20	U
21	V
22	W
23	X
24	Y
25	Z
26	)
27	-
28	A
29	Not Used
30	Not Used
31	Card Reader
32	Printer
33	Card Punch
34	Console
35	Drum File 1*
36	Drum File 2*

Tape Units

*The standard drum files are both 16000000_8 locations long and Drum File 2 sequentially follows Drum File 1. On the 1107, Drum File 1 begins at location 26000000_8; on the 1108, Drum File 1 begins at location 31000000_8.

NOTE: The Card Reader, Printer, and Card Punch are defined as units -1, -2, and -3, respectively. Programmers writing their own I/O table should so note. The entry point for the I/O table is defined as NTAB$. (NTAB$ is logical unit 0, NTAB$+1 is logical unit 1, etc.)

4.3 OPERATING PROCEDURES

The operating procedures are developed here for quick reference. For details concerning operating instructions which are directed toward the EXEC II System, see Section 8 of the UNIVAC EXEC II Programmer's Guide.

4.3.1 TAPE BOOTSTRAP ROUTINE

The EXEC II Load Tape (COSM) has as its second block* a 224-word Bootstrap Routine which serves to bring in the remainder of the resident and various other parts of the system. The Bootstrap Routine also provides a simple card load routine, a panic dump, and a method of patching the resident system prior to writing it to drum.

4.3.2 DRUM BOOTSTRAP ROUTINE

Once the system has been set up, it may be reinitiated by performing a manual bootstrap operation from appropriate drum channel. No console jump switches are effectual from a drum bootstrap. Whenever the system has been bootstrapped from either drum or tape, date and time key-ins should be performed and the real-time clock turned on.** In addition, the printer forms should be adjusted to print on physical line one.

4.3.3 BASIC SYSTEM KEY-INS (OPERATOR ACTIONS)

The primary control exercised by the operator over the Monitor System is by means of system key-ins. A system key-in is one made that is not in response to a specific request by the running program. The first character of key-ins always determines the routine which is to process the information entered into the computer. What follows this first character depends on the specific type of key-in.

4.3.3.1 E and X Key-Ins

System key-ins of E and X are used to terminate the program currently being run. The E key-in is normally used in situations such as the program entering a closed loop. An X key-in is usually used when the program reaches some impasse, such as a request for a non-existent tape reel, etc.

---

*Block 1 is a hardware test. On the 1108, Block 1 is the boot block and is 2000_8 words in length.

**Time Key-In is not necessary on the 1108. May be booted without turning off RTC.

4.3.3.2 D and T Key-Ins

The D key-in is used to enter the current date into the system. The T key-in is used to set the system clock to the current time of day. The date and time key-ins must be used each time the system is brought from tape or drum.*

4.3.3.3 W Key-In

The W key-in is used only to cause a delay in the user's program; any parasite operation continues.

4.3.3.4 S Key-In

The S key-in is used primarily to cause the system to continue after having delayed. Reasons for delays may be caused by 1) the program running, 2) an MSG control card with an H option, 3) the occurrence of a new run when the system is in the stop between jobs mode, or 4) a system key-in of W (see Paragraph 4.3.3.3).

4.3.3.5 LF Key-In

The LF key-in provides for approximately four inches of paper from the typewriter to be spaced to allow paper to be conveniently torn off.

4.3.3.6 A Key-In

The A key-in is used for making magnetic tape assignments. There are four options:

A, AR--assign and rewind

AI--assign, and rewind with interlock

AN--assign without rewind

4.3.4 SYSTEM KEY-INS FOR INPUT/OUTPUT CONTROL

The system key-ins C (continue), R (recover), and F (fault) are used to exercise control over input/output operations which did not function normally. They are used to respond to type-outs produced by the system. The proper response depends on the kind of input/output device involved and the particular message produced.

*Time Key-In is not necessary on the 1108. May be booted without turning off RTC or Day Clock.

4.3.4.1 The Card Reader

The messages produced by the card reader are:

(1) CARD INTLK nn

This card reader has filled a stacker or the last card detected is not a properly punched FIN card.

(2) VERIFY ERROR nn

The card reader failed to read a card properly.

(3) CODE ERROR nn

An invalid character code was read.

(4) CARD FAULT nn

The card reader is inoperative.

4.3.4.2 The Card Punch

The messages produced by the card punch are:

(1) PUNCH INTLK nn

The card punch has exhausted its supply of blank cards, filled a stacker, or filled its chip box.

(2) PUNCH FAULT nn

The card punch is inoperative.

4.3.4.3 Magnetic Tape

Messages concerning magnetic tape are:

4.3.5 SYSTEM KEY-INS FOR SYMBIONT CONTROL

Symbionts must receive all of their control through the operator's keyboard. Key-ins used for control of symbionts are thoroughly described in the UNIVAC 1107 EXEC II Programmer's Guide and for additions and modifications, Goddard's Computer Operator's Manual.

4.3.6 TYPEWRITER MESSAGES PRODUCED BY THE MONITOR

During execution of programs under the EXEC II System, there are various messages produced on the typewriter. Consult the UNIVAC 1107 guide.

4.4 BIBLIOGRAPHY

This section provides the user access to the list of documents describing the major components of the EXEC II System. For each document, there is provided an abstract.

4.4.1 UNIVAC 1107 EXEC 2 User's Manual, NASA EXEC 2 Processor Modified Version of the UNIVAC System X-543-64-212, Goddard Space Flight Center

This document is intended as a working guide for the programmer using the NASA EXEC II System on the Univac 1107 computer. The EXEC II System has been modified and conventions for its use differ from one installation to another. This Manual is an attempt to assemble documentation of this nature into a single source. Besides the addition of two additional control cards, this Manual presents programming procedures, aids for debugging, and describes the NASA Auxiliary Library Tape as well as utility routines. This document is essentially obsolete.

4.4.2 UNIVAC 1107 EXEC II Programmer's Guide, General Manual U-3671

This Manual describes the EXEC II System and its supporting routines. It presents detailed specifications as to the form and content of control cards that exercise control over the EXEC II System; explains the input/output facilities, the program complex file, the memory allocation processor, the auxiliary processors; as well as commenting on operating instructions and the role of the symbiont (a small multiprogrammed routine).

4.4.3 UNIVAC 1107 Library II Programmer's Guide, General Reference Manual U-3672

This Manual presents to the programmer a description of the I/O packages for the communication with magnetic tape, drum, console, and paper tape, as well as explaining the EXEC II diagnostic system. In addition, the buffering routines for block and item level handling are described.

4.4.4 UNIVAC 1107 SLEUTH II Programmer's Guide, General Manual UP-3670--Rev. 1

This Manual provides a basic introduction to the SLEUTH II assembler language, describes its directives, and explains their use, and further presents a brief programmer's guide to the SLEUTH II language.

4.4.5 UNIVAC 1107 Central Computer, General Manual
UP-2463--Rev. 2

This hardware-oriented Manual provides a description of the Univac 1107 Computer and its principal components and an explanation of the instruction word form and of the Univac 1107 instructions, with examples of their use. Input/output operations are presented only briefly.

4.4.6 Decimal to Octal Conversion Table for Decimal Values 0 to 65,999
X-542-64-118, Goddard Space Flight Center

This document consists of tables of octal and decimal numbers.

4.4.7 S-C 4020 Programmer's Manual

This manual includes the modified Lockheed S-C 4020 Package which is on the system library.

4.4.8 UNIVAC 1108 EXEC II, Programmer's Reference Manual
UP-4058

This Manual is intended to be a guide for programmers using the EXEC II System on the 1108. There is sufficient information on fundamental concepts of the internal structure and operational qualities of the system. Background information is also provided to permit even those who are unfamiliar with EXEC II to learn the system. There are four major sections dealing with 1) structure of the EXEC II System as it exists in core and on drum; 2) controls--both the control of the executive over the machine environment and the user control over the executive; 3) the references required by the programmer to build and test a worker program; and 5) job set-up, which essentially presents in capsule form the material covered in the previous sections.

4.5 AUXILIARY TAPE CONTENTS

The primary purpose for creating the NASA Auxiliary Library Tape is to make available to the programmer library routines that cannot be put on the program complex file because of the limited drum space allocated for the system. There are two files generated.

4.5.1 AUXILIARY LIBRARY DECK SET

Figure 4-6. shows the deck setup that must precede the program in order to use routines in the Auxiliary Library. If all the routines in the auxiliary library are not to be used, the IN λ card should be replaced with the following:

```
FIND    λ, YYYYYY/ZZZZZZ
TRD     λ
```

for each routine desired. λ is the logical designation of the UNISERVO upon which the auxiliary library tape is mounted; AUXLIB is the label assigned to λ; YYYYYY is the name of the routine desired; and ZZZZZZ is the version name. The IN will read a file and place the elements it contains in the program complex until an end-of-file is reached. The FIND searches the tape until either the element is located or an end-of-file is reached. TRD will read one element from the specified unit and place the element in the program complex.

NOTE: When using the Auxiliary Library Tape, be sure that the tape has been positioned at the beginning of the file which contains the desired routines. For efficiency, the routines should be called in the order of their appearance on the tape.

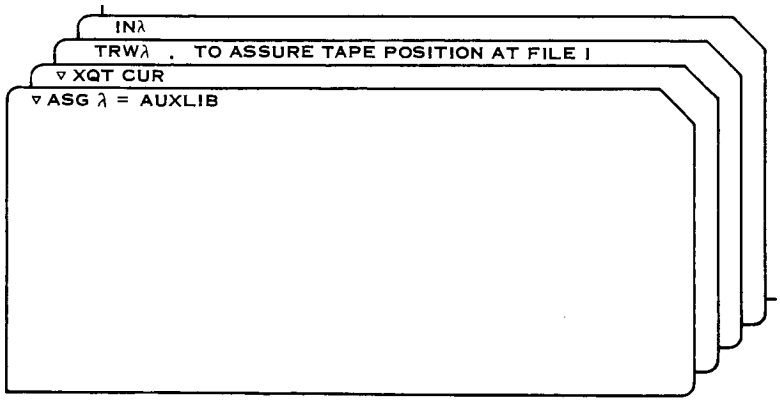


Figure 4-6. Sample Auxiliary Library Tape Setup

4.5.2 FILE 2 (Independent Executable Programs)

File 2 of the auxiliary library tape consists of four programs.

4.5.2.1 CULL

CULL provides a reference-map of a SLEUTH II program. CULL associates each occurrence of a name (alphabetic or numeric) in a program with the line number and name of the program in which it occurs. Each name occurrence (which is a label with an asterisk) is tagged. The completed map is in alphanumeric order. In the deck setup, λ is the tape unit assigned to the auxiliary library tape; yyyyyy is the label of the auxiliary library tape; xxxxxx is the label of the blank tape; PROGn/VERSn are in the name/version of the routines to be culled; and PROGRAM is the name of the program to be executed.

NOTE: G is used as the CULL input tape. First, the entire program deck is loaded into PCF. Before execution of the program, those routines which are to be CULLed are written out on Tape G. The program is then executed. Following execution, CULL is read into PCF and executed. CULL is part of the second file on the Auxiliary Library tape and is entered into the user PCF by the complex utility routine.

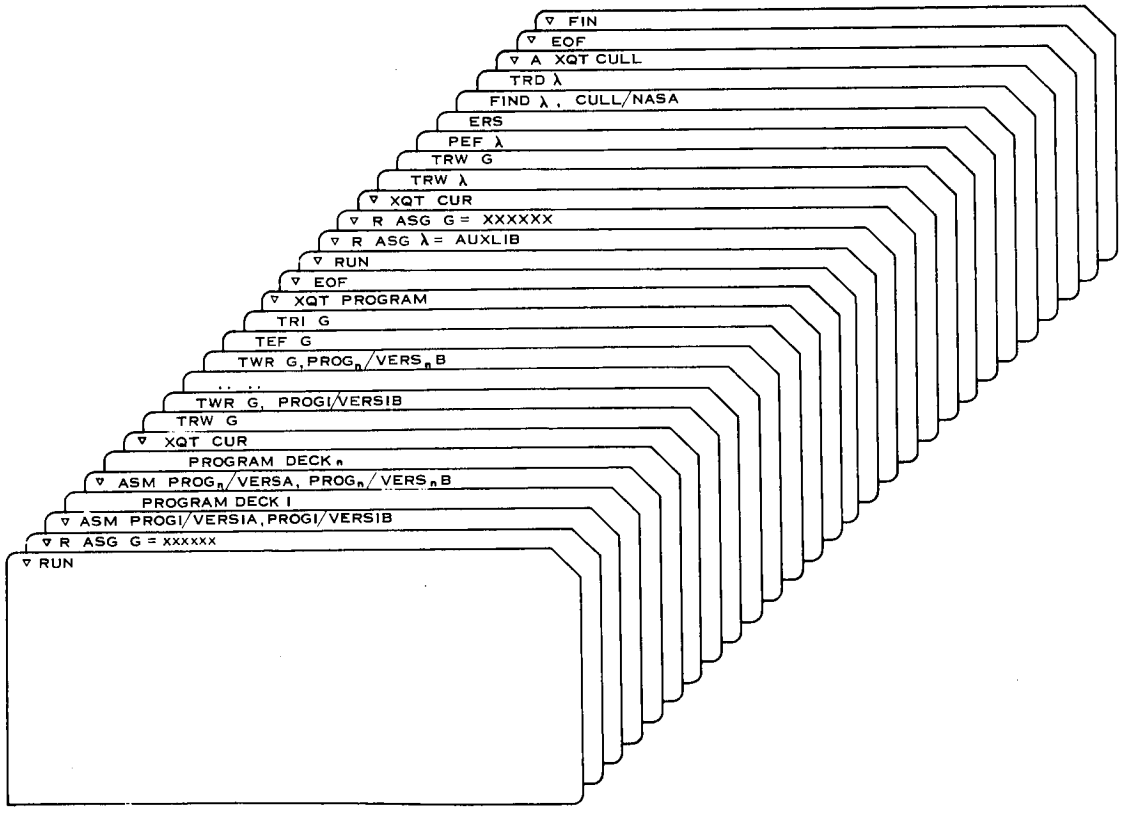


Figure 4-7. Sample CULL

4.5.2.2 FORTRAN II to FORTRAN IV--Translator (LIFT)

The translator is a program which automatically translates a FORTRAN II source program to a FORTRAN IV source program. The Translator operates under control of the 1107 EXEC II Monitor System. The Translator is also part of the second file on the Auxiliary Library Tape and is entered into the user's PCF by the complex utility routine. If in the deck setup, λ is the logical designation of the UNISERVO upon which the Auxiliary Library tape is mounted, AUXLIB is the label of the Auxiliary Library Tape. The programmer should consult the UNIVAC 1107 FORTRAN II to IV Translator (LIFT) Programmer's Procedure Manual, UP-3863.

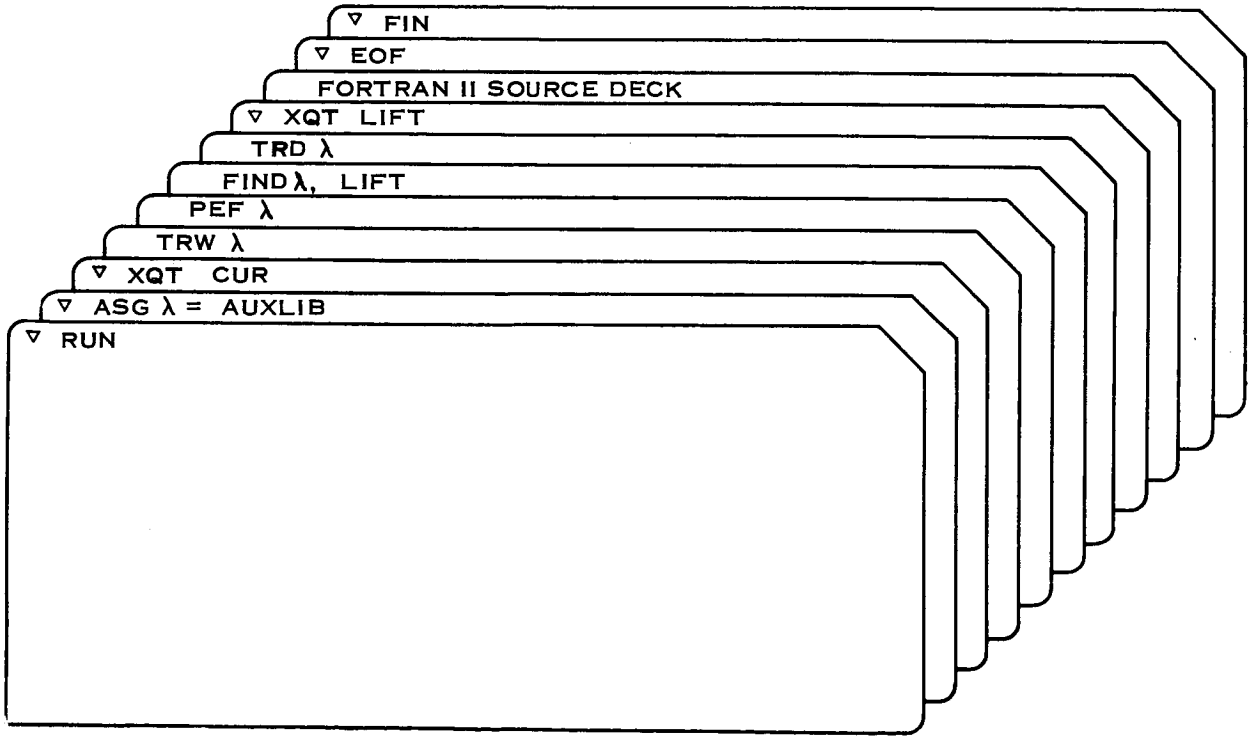


Figure 4-8. Sample FORTRAN II to FORTRAN IV--Translator (LIFT)

4.5.2.3 1107 FAP Translator

The FAP Translator is a program which automatically translates a FORTRAN Assembly Program (FAP) to a SLEUTH II source program. The program operates under control of the 1107 EXEC II Monitor System.

The programmer should consult the 1107 FAP Translator Programmer's Guide (UP-2593.17) for detailed information.

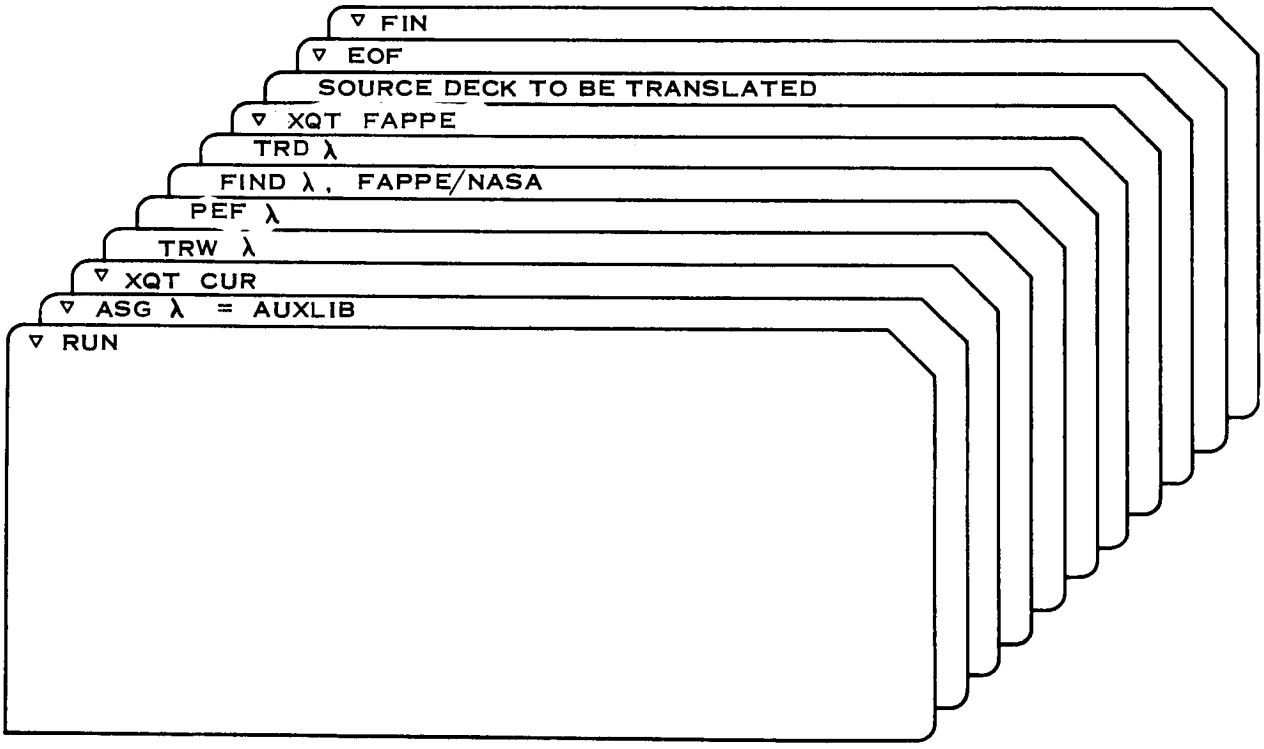


Figure 4-9. Sample 1107 FAP Translator

4.5.2.4 Linear Programming

Linear programming is a mathematical technique for finding the best or optimum solution to a particular problem. The optimum solution may be in terms of the lowest cost, least effort, shortest time, or minimum configuration of equipment to achieve a specific goal. With linear programming, it is also possible to determine the courses of action which will provide the greatest monetary return, the largest number of units of a product, or the biggest level of efficiency. In the deck setup, λ is the logical designation of the UNISERVO upon which the Auxiliary Library Tape is mounted. AUXLIB is the label to the Auxiliary Library Tape. The programmer should consult the 1107 Linear Programmer User's Manual Preliminary UP-3897 for detailed information.

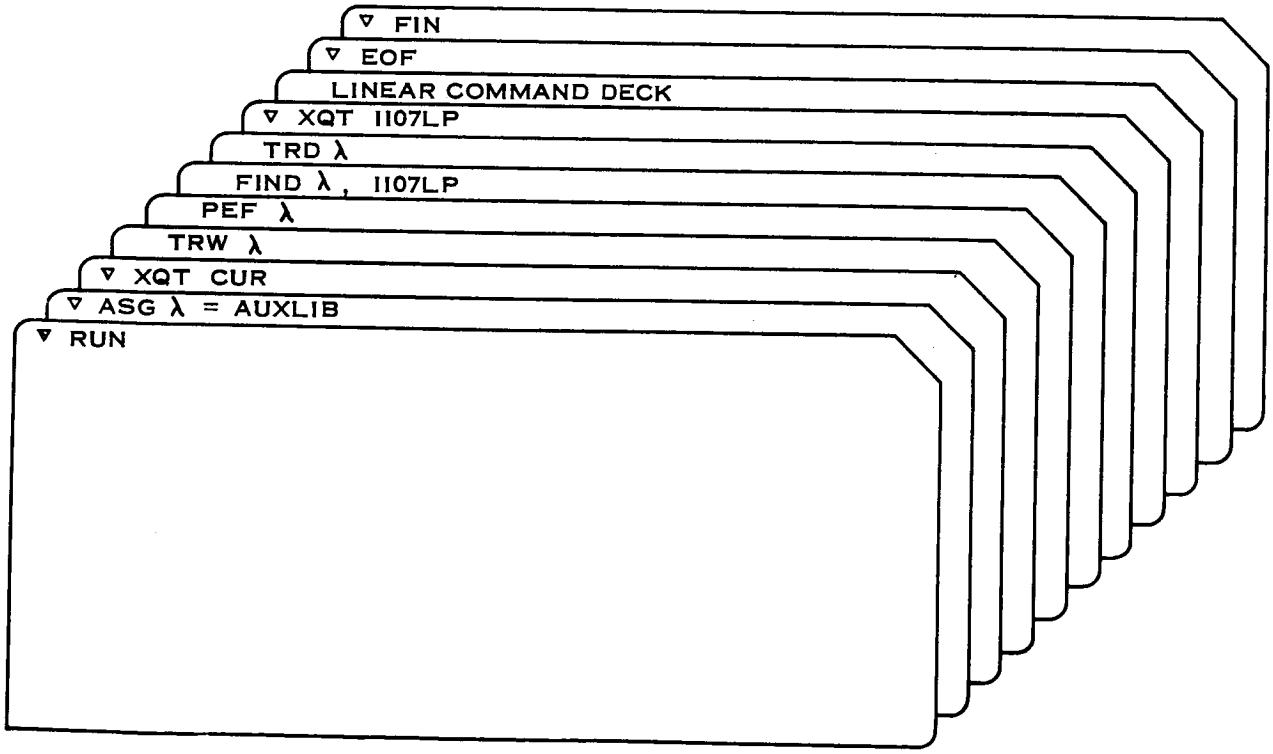


Figure 4-10. Sample Linear Programming

4.6 UTILITY ROUTINES

This section describes those utility routines that have been prepared for use by Goddard Space Flight Center programmers.

4.6.1 1107 UTILITY PROGRAMS

4.6.1.1 Tape Copy

This utility routine will accept assignments for up to eight input tapes and eight output tapes. In the copy process, density and parity may be varied by console key-in parameters. Also, data representation can be changed from Fieldata to BCD, or the converse, or may be left in its original form.

By appropriate manipulation of the tape copy parameters, the user may perform such tasks as duplicating a tape, taking files from an input tape and scattering them onto separate output tapes, and gathering files from several input tapes onto a single output.

The tape copy routine may operate in any one of three environments: 1) a subprogram to a user's routine, 2) a standard operator's utility program, and 3) a subroutine to the higher level Tape Utility Program. Use of TCOPY$ is virtually the same from any environment.

To request TCOPY$, the programmer fills out the request card (1107 Instruction Card) indicating to the operator the tape(s) desired. The request card must be filled out in the following manner:

On one side of the request card is a large, almost empty space, labeled "Special Instructions." In this space, write the characters: PPDDFTNN

a. Under the first P, write the letter O or E, depending on whether the input tape is of odd or even parity.

b. Under the second P, write O or E for the output tape.

c. Under the first D, write L or H, depending on whether the input tape is of low or high density.

d. Under the second D, write L or H for the output tape.

e. Under the F, write one of the digits 1 through 99 to specify the number of consecutive end-of-files at which the current copy should terminate, or write a 0 to mean that TCOPY should ignore this parameter.

f. Under the T, write one of the digits 0 (no conversion), 1 (BCD to FD, card base), 2 (FD to BCD, card base), 3 (BCD to FD, print base), or 4 (FD to BCD, print base).

g. Under the NN, write one of the numbers 01 through 99, to specify the number of serial end-of-files at which the current copy should terminate.

EXAMPLE: PPDDFTNN
EEHH3Ø58

ANOTHER EXAMPLE: PPDDFTNN
OELLØ2Ø6

Once control has been given to the TCOPY$ routine, requests for input parameters are made. These concern parity and density settings, format conversions, and copy termination characteristics. Any parameter specification normally comes from an input card; but, by substituting an EOF control card in its logical place, requests may be made from the console.

Since the TCOPY$ routine occupies much of core, it is often necessary for the user to define an overlay structure with the MAP processor. The TCOPY$ routine has been written so that the Autoload feature of MAP is permissible. Linkage to the tape copy routine is

```
LMJ     11, TCOPY$
+  'XY'
```

where X is logical unit designator of the file of input tapes, and Y is the logical unit designator of the file of output tapes. For details, see 1107 Programmer Notes 6 and 7, obtainable from Mrs. Pat Barnes, Building 3, Room 125.

4.6.1.2 Tape Print

This utility routine will accept assignments for up to eight input tapes. Console key-in parameters allow for setting parity, density, print termination characteristics, and data formation in a manner analogous to the tape copy utility routine. In addition, up to 99 files may be skipped before printing begins. Also, an identification field is provided to allow a unique heading on the printer hard copy.

To request TPRNT$, the programmer fills out the job request card (1107 Instruction Card) indicating to the operator the tape(s) desired.

The request card must be filled out in the following mannner:

On one side of the request card is a large, almost empty space, labeled "Special Instructions". In this space, write the characters: PDFTNNSS IDENT

a. Under the PD, write the letter O or E, depending on whether the input tape is of odd or even parity.

b. Under the F, write one of the digits 1 through 9 to specify the number of consecutive end-of-files at which the current copy should terminate, or write a 0 to mean that TPRINT should ignore this parameter.

c. Under T, write one of the letters F (Fieldata), B (BCD to FD), O (octal), D (decimal), E (exponential floating point), X (floating point to fixed point), or U (user's general format, if any).

d. Under NN, write one of the numbers 01 through 99 to specify the number of serial end-of-files at which the current copy should terminate. 00 or ∇∇ means that this field is ignored.

e. Under SS, write one of the numbers 01 through 00 to specify the number of end-of-file marks to be skipped at the beginning of the current print.

f. Under IDENT, write any six characters or less, to request any identifying key-in for the current print. This field is completely arbitrary and need not be provided.

Since the TPRNT$ routine occupies much of core, it is occasionally necessary for the user to define an overlay structure with the MAP processor. The tape print routine is written so that the Autoload of MAP is permissible. The calling sequence is:

```
        LMJ     11, TPRNT$
F       FORM    18, 12, 6
          F     LABEL, BLKLIM, LU
```

where LABEL is the entry point name of the user's own formatting subprogram or the value ∅ if none is defined; BLKLIM is the integer value of the number of blocks to print before terminating the current print job. This count will be nullified for the current file if an end-of-file mark is read before the designated number of blocks. The value ∅ (zero) means an indefinite number of blocks is to be read. LU is any logical unit designation for the input file (such as "A").

For more details on Tape Print, see 1107 Programmer Notes 6, 7, and 8, obtainable from Mrs. Pat Barnes, Building 3, Room 125.

4.6.2 1401 UTILITY PROGRAMS

4.6.2.1 Univac 1107 Fieldata Code Conversion in IBM/BCD No. 55

The Univac 1107 Fieldata code to IBM Binary Coded Decimal utility program provides the ability to convert a Univac 1107 Fieldata tape to IBM Binary Coded Decimal tape equivalent. Field Fieldata-to-BCD Dump processing of any combination of Fieldata tape-to-printer and Fieldata tape-to-BCD tape operations.

Programming information on the Fieldata-to-BCD Dump may be obtained from Mrs. Pat Barnes, Building 3, Room 125, Extension X-6796. When submitting runs, simply specify Fieldata Dump on your job request card. This program is available in the 1401 machine room in Building 14.

Any difficulties or discrepancies encountered in the use of this utility program should be directed to the Programming Methods Section, Advanced Projects Branch, Extension 6796.

4.6.2.2 1401 Program (No. 56) to Interpret and Print 1107, .PR Tapes

The 1401 program to print and interpret 1107 .PR tapes is used in conjunction with the 1107 control card VS TPR. This has been done to increase utilization of the 1107, to speed up execution of the programs processed by that large scale computer, to more efficiently use the drums, and to exactly position the margins. Additionally, output may be saved and/or reprinted.

4.7 ASSEMBLY AND EXECUTION FROM TAPE

Programs and subroutines may be assembled and executed from tape rather than from cards through the use of the Complex Utility Routine (CUR). (See page 5-15 of the EXEC II Manual, U-3671.)

The program must have been previously been loaded onto tape through the 1107 or 1108 via CUR. This may be done by a simple OUT instruction, which will put the entire drum PCF onto the specified tape, or specific routines may be loaded onto tape by use of the TWR instruction. The OUT instruction may be used to specify one or more of the seven different types of elements. When the programmer has the desired routines on tape, an end-of-file is put on the tape, using the TEF instruction.

In execution from tape, it is sufficient to read in the entire file of the input tape via the CUR instruction, IN. The entire file will be read into the drum PCF. Any additional routines may then be read in from cards or another tape. (If two routines are read into the drum PCF having the same name and version, the last one read in will delete the previous one. Therefore, if one wishes to substitute a new routine for one currently on tape, it is only necessary to read in the new routine after the old one has been loaded into the drum PCF.) Once the entire PCF has been loaded, an XQT card will allocate the routines needed and enter the desired execution. Several different main programs may be held on the drum PCF at the same time, so that several executions may follow one another without erasure and rewriting of the drum PCF.

More often than not, it is desired to reassemble or recompile a program that is already loaded onto a program tape. It is not necessary to delete the program by a new card deck, as indicated above, if the program is loaded onto the program tape in symbolic form. (Obviously, one cannot reassemble or recompile a relocatable routine.) It is only necessary to reassemble from drum with correction cards. To specify assembly or compilation from drum, the three-letter mnemonic for the processor should be followed with "*". For example, if we wish to replace the fourth card of a FORTRAN IV program named TEST, the following deck will suffice:

```
∇ IWL     FOR,*     TEST
-4,4

      GO  TO 128
```

In the above example, we will get a single-spaced listing of the routine named TEST, with assembly listing; the correction cards will be listed prior to the listing, and the fourth card will be replaced by the FORTRAN instruction GO TO 128. The relocatable element formed by this assembly will have the version name CODE.

The programmer may reassemble or recompile as many routines as necessary. The W option is not necessary for assembly with corrections; it merely lists the correction cards for each reference. The programmer should be careful that he has the proper listing for the symbolics from which he is updating--with a large number of updates, it is easy to forget which routine is on tape (the voice of experience). He should give the relocatable the same name and version as the previous relocatable element, unless the previous relocatable is deleted by the DEL instruction or selection is made by use of flags.

Further information on correction cards is found on page 5-10A of the EXEC II Manual (U-3671), and information on flags is found on pages 5-5, 5-9, 5-10, 5-19, 5-27, 5-28, 5-30, and 5-31 of the same Manual.

4.8 CODING SHEETS

A special printed coding form (see Figure 4-11.) is provided for writing programs in SLEUTH II language. Coding is in free form. This coding form facilitates both the writing and the keypunching of programs. These forms are obtainable from Mrs. Pat Barnes, Building 3, Room 125.

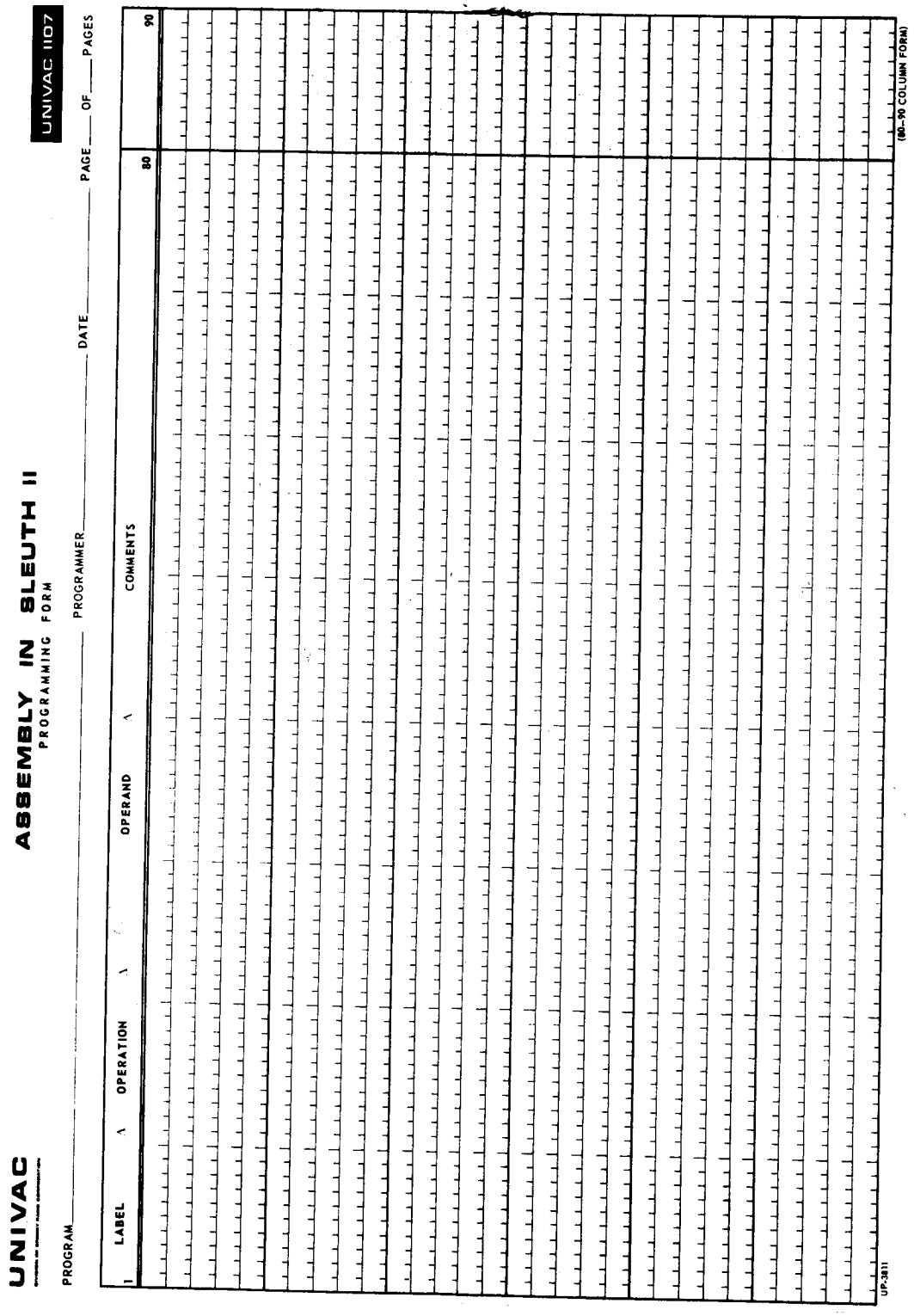


Figure 4-11. Assembly in SLEUTH II Programming Form

CONTENTS

CHAPTER 5 AUTOCODER-SPS

CONTENTS (Cont'd)

ILLUSTRATIONS

TABLES

CHAPTER 5

AUTOCODER-SPS

5.1 SYSTEM DESCRIPTION

This chapter describes two similar symbolic programming systems, the Autocoder and the Symbolic Programming System (SPS) for IBM peripheral equipment. (See Table 5-1.) The symbolic programming systems facilitate logical, efficient programming, with a minimum of actual coding effort. The programmer may use symbolic addresses in place of numerical addresses, use mnemonic operation codes rather than machine language codes and by use of symbolic language he can control the locations of record and work areas if he so chooses, or he can leave this job to the processor program.

5.1.1 SYSTEM CONFIGURATION

The Autocoder language is not directly compatible with SPS, but the Autocoder translator can translate source programs coded in either language or in a combination of the two. The Autocoder is a more powerful machine-oriented language than SPS and the major differences between them is that the Autocoder:

1. Provides macro instructions whereby one instruction in the source program is translated into many actual machine instructions. (SPS does not use macros.)

2. Uses literals whereby the Autocoder processor assigns a location to the constant and fills in the assigned address in the instruction. (SPS does not use literals.)

3. Uses a free-form coding sheet whereby the programmer uses as much space as required for each operand and separates the operands by commas if there is more than one. (SPS coding is of the fixed-form type.)

4. Provides library routines. (SPS would require tape units.)

The Autocoder requires at least four magnetic tape units and 4,000 positions of core storage. SPS is usable on cards only. SPS-1 can assemble programs for any size object machine from 1,400 to 4,000 positions of core storage employing a 1,400-character machine. SPS-2 can assemble programs for any size object machine from 1,400 to 16,000 positions of core storage employing a 4,000-character machine. There is a strict one-to-one correspondence of SPS statements. The SPS translator requires 4 card passes (or 5 if a condensed object program deck is desired). The 1401 Program Library contains several user-developed revisions of the SPS translator utilizing tape passes, rather than card passes. Although these programs will effectively speed up the time required for translation, they do require from one to three magnetic tape units (depending on the program used) in addition to the minimum configuation requirements.

There is a 1401 Input/Output Control System as a supplement to the Autocoder which handles all of the normal input and output programming considerations with a minimum of programmer effort. It consists of additional control and macro operations that handle reading and writing, tape blocking and unblocking, file labeling, and error checking.

Figure 5-1 shows the Autocoder flow diagram. Figure 5-2 shows the SPS flow diagram.

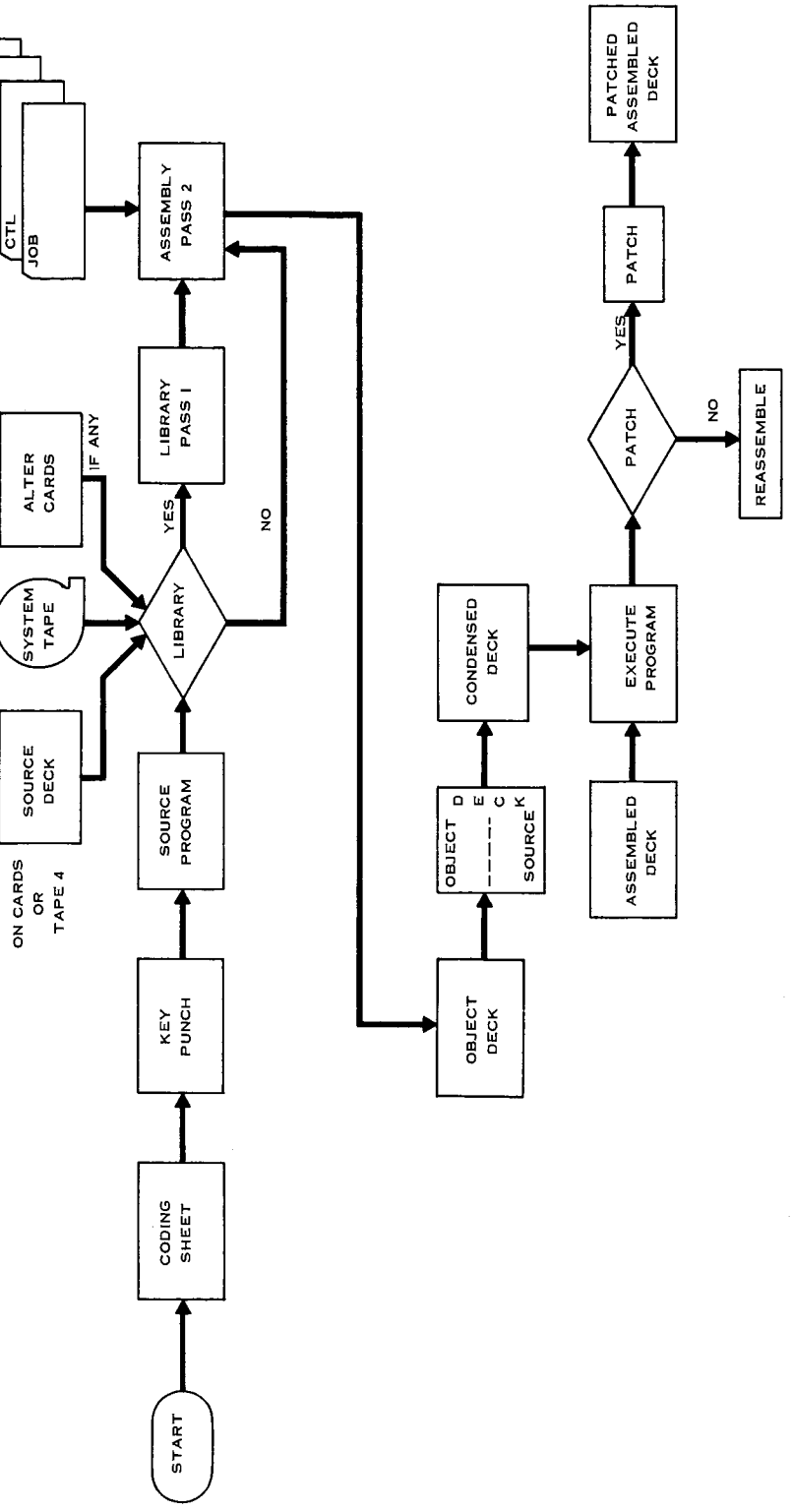


Figure 5-1. Autocoder Flow Diagram

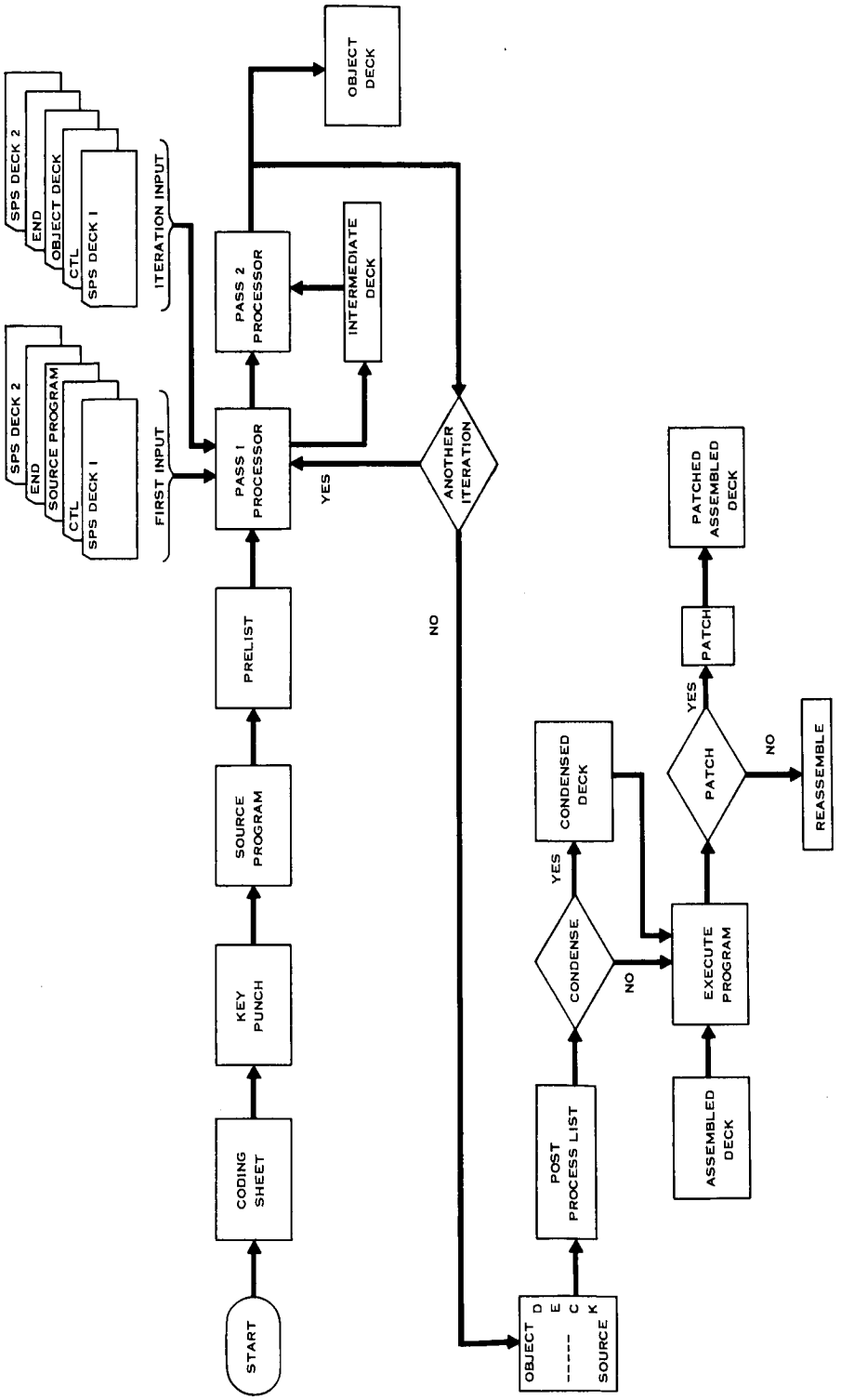


Figure 5-2. SPS Flow Diagram

5.1.2 MACHINE CONFIGURATION

The programmer using the Goddard Space Flight Center Computer Complex has a vast quantity of data processing equipment at his disposal. The Autocoder-SPS systems operate on several of these computers. A list of this equipment is given in Table 5-1.

Table 5-1. Computer Equipment

Bldg. Loc.	Computer Facility	Memory Size	Magnetic Tape Units	Card Read/Punch	Line Printer
14	A-IBM-1401	1.4K	1-7330		1403-II
14	B-IBM-1401	8K	2-7330***	1402-I	1403-II
14	C-IBM-1401	8K	3-7330***	1402-I	1403-II
14	D-IBM-1401	4K	2-7330	1012-I* 1402-I	1403-II
3	E-IBM-1401	4K	2-729-II	1402-I	1403-II
14	F-IBM-1401	8K	2-729-II	1402-I	1403-II
21	IBM-1460	8K	4-729-VI*** 2-729-IV***	1402-I	1403-III
14	I-IBM-7010**	100K	8-729-IV**** 1-729-VI	1402-I	1403-III
14	IBM-1401	8K	2-729-IV	1402-I	1403-II

* Paper tape reader/punch
** 1301 disk
*** Switchable units
**** Two of these are switchable on the 1401.

5.1.3 SYSTEM TAPE MAINTENANCE

The Programming Methods Section (PMS) of the Data Systems Division has primary responsibility for maintaining the Autocoder master tape. Tape revisions or updating occur periodically as a result of one or more of the following conditions: 1) new Autocoder version or significant corrections released by IBM; 2) major changes originating from GSFC programmers; and 3) catastrophic errors requiring the immediate issurance of new tape.

5.1.4 ERROR REPORTING

The PMS has the responsibility of maintaining the Autocoder and SPS systems. Any questions regarding system utilization and system discrepancies should be directed to PMS personnel. The current method of reporting system discrepancies verbally is expeditious. However, it is recommended that the System Discrepancy Report (see Form 1-1, Chapter 1) be utilized for submittal to the PMS coordinator. In this way, a current file of all discrepancies will be maintained along with the corrective actions taken. A copy of the Discrepancy Form will be available in the PMS coordinating office (Room 127, Building 3) and in the dispatcher's office (Building 1, Building 14, and Building 3). Programmers are required to periodically check the System Status Report (see Form 1-2, Chapter 1) to insure satisfactory operational performance from the system used.

5.2 DETAILED PROCEDURES

This section describes the Autocoder processor-control operations and the SPS processor-control operations.

5.2.1 AUTOCODER PROCESSOR-CONTROL OPERATIONS

Autocoder has several control operations that enable the user to exercise some control over the assembly process. They are developed here for quick reference. For details, see the IBM 1401 and 1460 Autocoder (on Tape) Language Document, Form C24-3319-0.

Table 5-2. Autocoder Processor-Control Operations

Op Code	Purpose	Description
JOB	Job Card	This is the first card in the user's source program deck. It is used to print a heading line on each page of the output listing from the assembly process and to identify the self-loading program deck or tape. It allows the programmer to identify a job or parts of a job in the output listing.
CTL	Control Card	The control statement is the second entry in the source program deck. It is used to specify size of the processing and object machine, type of output, and the presence or absence of the Modify-Address feature. The MA instruction is standard in the 1460 systems and in 1401 systems with 8-, 12-, and 16-thousand positions of core storage. For an object machine not equipped with the MA feature, the Autocoder processor automatically assembles a routine to simulate the MA instruction, types out the object program, and indicates the presence of the fifth tape and of the Read-Punch Release Feature.
ORG	Origin	Used to specify a storage address at which the processor should begin assigning locations to instructions, constants, and work areas. Allows programmer to chose area(s) of storage where object program will be located.

Table 5-2. Autocoder Processor-Control Operations (Cont'd)

Op Code	Purpose	Description
LTORG	Literal Origin	Coded in same way as ORG statements, the LTORG statements direct the processor to assign storage locations to previously encountered literals and closed library routines, beginning with the address written in the operand field of the LTORG statement. LTORG statements can appear anywhere in the source program.
EX	Execute	Used during the loading of the assembled machine language program to discontinue the loading process temporarily to execute a portion of the program just loaded. Allows use of several program sections if total program exceeds the limits of available storage capacity. If inputs to the program are on magnetic tape and the program is also on tape, one tape unit can be assigned to the program and another can be assigned to the input data.
XFR	Transfer	Same function as an EX statement except that literals, closed library routines, and address constants are not stored. An XFR statement transfers to and executes instructions that have been previously loaded.
END	End	As the last card in the source deck, it is used to signal the processor that all of the source program entries have been read, and to provide the processor with the information necessary to create a bootstrap card. This bootstrap card causes a transfer to the first instruction in the object program after it has been loaded into the machine at program load time. Thus, program execution begins automatically.

Table 5-2. Autocoder Processor-Control Operations (Cont'd)

Op Code	Purpose	Description
SFX	Suffix	Directs the processor to put a suffix code in the sixth position of all labels in the symbolic program that have five, or fewer chapters, until another SFX statement is encountered. In this way, the programmer can use the same label in different sections of the complete program. A suffix statement with a blank operand can be used to stop the assignment of a suffix code.
ENT	Enter New Coding Mode	Although the 1401 and 1460 Autocoder processor accepts source programs coded in either free-form Autocoder language or in fixed-form SPS language, it is possible to assemble a single program coded in a combination of the two languages. With ENT the processor is informed that a change in coding form follows. Allows programmer programs prepared wholly or partially in SPS format to be reassembled by the processor.
ALTER	Alter	Used to add, delete, or substitute instructions in the object program after the original assembly has been completed. By saving Tape 4, which, at the end of assembly, contains a source program, it is possible to reassemble the program easily by processing ALTER cards. During each assembly, each statement can be altered by an ALTER entry if assigned a sequence number. This number is listed in the first column of the output listing. These numbers are used in the ALTER entries to reference statements to be changed during the reassembly.

5.2.2 SPS PROCESSOR-CONTROL OPERATIONS

SPS has several control operations that enable the user to exercise some control over the assembly process. They are developed here for quick reference. For details see the IBM 1401 SPS document Form C24-1480-1.

Table 5-3. SPS Processor-Control Operations

Op Code	Purpose	Description
CTL	Control	This control card is placed at the beginning of the source deck, so that the SPS processor is able to distinguish the storage sizes of the processing machine and the object machine.
ORG	Origin	An ORG statement causes the processor's storage assignment counter to assign addresses beginning at a particular location specified by the programmer. If it is entered as the first card of the source program, an ORG card can cause the initial assignment of addresses to be at a location other than 333. An ORG statement may be included at any desired point in the source program. This will cause the counter to be reset and cause all future entries to be assigned addresses beginning at the particular location designated by the programmer. Character adjustment and fixing are not valid in an ORG statement.
EX	Execute	Used during the loading of a machine-language program to discontinue the loading process temporarily to execute a portion of the program just loaded. Allows programmer to also divide his program into several program sections if his total program exceeds the limit of available capacity.
END	End	Signals the processor that the last card in the source program has been processed. If the programmer specified in the (A) operand the actual or symbolic address at which the object program is to begin execution, and END statement will produce an instruction that will start program execution immediately after loading. If the (A) operand is blank, the 1401 will halt when the last instruction has been loaded.

5.3 BIBLIOGRAPHY

This section provides a list with abstracts of documents on Autocoder-SPS.

5.3.1 IBM 1401 and 1460 Autocoder (on Tape) Language Specifications and Operating Procedures Form C24-3319-0

This publication contains the language specifications and operating procedures for the Autocoder (on Tape) programming system. The IBM 1401 Autocoder processor can assemble programs for all 1401 and 1460 systems. The language specifications cover two sections: 1) specifications of the symbolic language (mnemonics, labels, address types, and control operations) and the rules for writing the source program; and 2) descriptions of macro operations and macro instructions. The operating instructions describe the procedures to be performed by the operator when assembling an Autocoder program on an IBM 1401 or 1460 tape system. There is also a description of the phases of the Autocoder processor as well as an explanation on system halts and restarts.

5.3.2 IBM 1401 Symbolic Programming Systems Form C24-1480-0

The reader should have a basic knowledge of 1401 machine language programming in order to understand this manual which provides programmers with the information necessary to code a 1401 program in SPS language and assemble a machine-language object program. Described are symbolic programming principles and concepts as well as detailed specifications of the 1401 Symbolic Programming Systems, SPS-1 and SPS-2. Operating instructions for processing the SPS source program are expounded upon. For the beginning SPS programmer, a sample program is included. There are also shown input and output forms, a block diagram of the program procedure, the symbolic program, and SPS output listings of the symbolic and machine-language programs.

5.3.3 IBM 1401 Data Processing System Reference Manual Form A24-3067

This manual presents the physical features of the 1401 Data Processing System, enumerates the processing concept, discusses magnetic tape, presents a description of address modification, and expounds upon the operating features and timing, as well as providing appendices showing forms, flow charts, operation codes, and other charts.

5.3.4 System Operation Reference Manual, IBM 1401 and IBM 1460 Data Processing Systems
Form A24-3067-1

This manual is the first of five providing the complete instruction set for the IBM 1401 and 1460. The operation code for each instruction is provided in actual and mnemonic form, along with examples of each. The formula for calculating the execution time of each instruction is also included. A general knowledge of the IBM 1401 or 1460 Data Processssng Systems is assumed.

5.3.5 IBM 1447 Console
Form A24-3031-3

This reference publication describes the specific models of the IBM 1447 Console that can be attached to the 1401 and 1460 Data Processing Systems. It presents detailed descriptions of indicator lights, keys, dials and switches.

5.3.6 Miscellaneous Input/Output Instructions, IBM 1401 and IBM 1460 Data Processing Systems
Form A24-3068-0

This publication presents a description of the instructions used by the 1401 or 1460 to operate miscellaneous input/output units. Also included is timing information for each unit attached to a 1401 or 1460 Data Processing System.

5.3.7 Tape Input-Output Instructions
IBM 1401, 1440, and 1460
Form A24-3069-1

This publication not only contains a description of the instructions used by the data processing system to operate the tape units attached to it but also includes timing information on the 729, 7330, and 7335 tape units.

5.3.8 Special Features Instructions, IBM 1401 and IBM 1460 Data Processing Systems
Form A24-3071-2

This manual describes the special features available for the 1401 and/or 1460 Data Processing Systems. Each feature is described and identified for the system to which it can be applied. Included are instructions for the special features on the IBM 1402, 1403, and 1009 when these units are used with the 1401 or 1460 Data Processing Units.

5.3.9 IBM 1402 Card Read-Punch
Form A24-3072-1

This manual describes the 1402 Card Read-Punch as it pertains to the 1401, 1410, and 1460 data processing systems. It covers the major mechanical units, their functions and operating controls, and special features that can be installed to expand the capabilities of the basic machine.

5.3.10 IBM 1403 Printer
Form A24-3073

This manual describes the 1403 Printer as it pertains to the 1401, 1410, and 1460 data processing systems.

5.3.11 Input/Output Control System (on Tape), Specifications and Operating Procedures, IBM 1401 and 1460
Form C24-1462-2

This publication presents the programming required to use the IOCS to control the input/output of data from card reader, card punch, printer, and tape files. There is a detailed description of the IOCS entries (DIOCS and DTF) and the macro instructions. Of especially useful to experienced programmers are the sections dealing with Summaries (briefly lists storage-area considerations, macro instructions and processing-overlap considerations and Program Operation (describes IOCS library routines, labels, halts, and error indications. IOCS is a supplement to Autocoder.

5.3.12 Utility Programs for IBM 1401 Tape System: Preliminary Specifications
Form J24-1411-1

This bulletin describes the card-to-tape, tape-to-card and tape-to printer programs for the 1401. This publication is a major revision and obsoletes the previous bulletins, J24-1411-0 and J29-1411-0. The listed programs can, within limitations, accommodate magnetic tapes and card decks prepared on any IBM system.

5.3.13 IBM 1410/7010 Operating System (1410-PR-155) System Monitor
Form C28-0319-3

This publication provides programmers and systems analysts on the use of the System Monitor to control the 1410/7010 Operating System. Autocoder is an element of this operating system.

5.3.14 IBM 1410-7010 Operating System (1410-PR-155) Autocoder
Form C28-0326-2

This publication deals with the Autocoder language concerning this 1410/7010 Operating System. It describes the basic concepts and functions of Autocoder, as well as types of operand entries and operation codes, and further presents the macro system.

5.3.15 IBM 1410/7010 Operating System (1410-PR-155), Basic Input/Output Control System
Form C28-0322-3

This publication presents to 1410 and 7010 programmers the information needed to write efficient programs incorporating the Basic Input/Output Control System. This system can schedule, implement, and control the transfer of data to and from core storage. It can also perform functions relating to the transfer of data, such as error detection and correction.

5.4 1401 UTILITY ROUTINES

These routines are used to facilitate program development, system development, and program and system documentation.

5.4.1 FORTRAN PREPROCESSOR

The FORTRAN Preprocessor is a 1401 program that is used to scan a FORTRAN source program for errors. This helps the programmer to eliminate those errors prior to the program being submitted to the 7094 for compilation.

The object deck (L2010000-L2010340) is used on the 1401 to write the FORTRAN PREPROCESSOR on the Library Tape 1. To write this tape, READY Tape 1, place the FORTRAN Preprocessor deck in the card reader and press the LOAD button. When the reader stops for the last card, press START. To preprocess, place the FORTRAN source deck(s) in the card reader, mount a blank tape on Unit 2, and the FORTRAN Preprocessor tape on Unit 1. Tape 1 must be positioned ready to read Record 1 before pressing or simulating the LOADTAPE button.

The complete program writeup and listing is available in the Programming Methods Section.

5.4.2 TWO-TAPE AUTOCODER ASSEMBLY SYSTEM

The Two-Tape Autocoder Assembly System is used for the purpose of bringing to the user of a smaller 1401 system the benefits of the Autocoder Programming Language. It is designed to meet as many of the specifications of the regular Autocoder system as is possible within the limitations imposed by the availability of only two tape units.

The complete program writeup and listing is available in the Programming Methods Section.

5.4.3 MAST (Minneapolis Assembly of SPS Two)

The MAST program is a modification of the 1401 SPS II Assembly Program to use magnetic tape to store the partly assembled output of PASS 1 rather than on punched cards.

The complete program writeup and listing is available in the Programming Methods Section.

5.4.4 GODDARD MULTIPURPOSE UTILITY PROGRAM

The Goddard Multipurpose Utility Program is designed to enable Goddard Space Flight Center to perform present day 7090 peripheral tape-to-printer, card-to-tape, and tape-to-card processing in any combination of operations on the 1401 computer. The functions to be performed are specified only by the use of sense switches, located on the 1401 console; no control cards are necessary.

The complete program writeup and listing is available in the Programming Methods Section.

5.4.5 TAPE DUPE (SNOOPY)

This program is used to duplicate tapes written in BCD and/or binary mode. It reads tape from Tape Unit 1 and copies onto the tape on Tape Unit 2. The program will dupe any length record up to a maximum of 6689 characters. A check is made for minimum record length of three 7094 words. Ten tries are made at reading a record before the program halts. If the record is ignored (dropped) a line is printed giving the record number and file, and a count of these records is printed at the end of the file. An input tape can be read without writing an output. This is used to scan a tape since the statistics are printed at the end of the file.

The complete program writeup and listing is available in the Programming Methods Section.

5.4.6 PRE-PROCESS LISTING ROUTINE

This SPS Pre-Process Listing Routine makes it possible for the programmer to detect many coding or keypunching errors in the program deck before assembly. Besides restoring the printer carriage, this routine prints the card image and messages in eleven fields on the printed page. It also prints at the end of the source program, the number of significant labels in the entire object program as well as causes the highest storage address assigned by the processor (exclusive of the actual address assigned by the programmer) to be printed.

The complete program writeup and listing is available in the Programming Methods Section.

5.4.7 1401/1460 COMBINED UTILITY ROUTINE

This utility routine is used for routine card-to-tape and tape-to-print/punch operations. The combined functions permit tape loading concurrent with printing and punching. Simultaneous operations cause printing to slow down to card reader speed. The 1447 IBM console typerwriter on the 1460 will print the reason for each programmed halt.

The complete program writeup and listing is available in the Programming Methods Section.

5.4.8 1.4K 1401 PROGRAM

This is a print program for the 1.4K 1401 available from the Programming Methods Section.

5.4.9 1401 CARD TO TAPE PROGRAM FOR 7090 IBSYS SYSTEMS

This 1401 card-to-tape program for the 7090 IBSYS systems is available from the Programming Methods Section.

5.4.10 UNIVAC 1107 FIELDATA CODE CONVERSION IN IBM/BCD

The UNIVAC 1107 Fieldata code to IBM Binary Coded Decimal utility program provides the ability to convert a UNIVAC Fieldata tape to IBM Binary Coded Decimal tape equivalent. Fieldata-to-BCD Dump utility program for the IBM 1401 Tape System performs concurrent processing of any combination of Fieldata tape-to-printer and Fieldata tape-to-BCD tape operations.

5.4.11 1401 PROGRAM TO INTERPRET AND PRINT 1107 .PR TAPES

The 1401 program to print and interpret 1107 .PR tapes is used in conjunction with the 1107 control card ▽ TPR or G Keyin. This has been done to increase utilization of the 1107, to speed up execution of the programs processed by that large scale computer, to more efficiently use the drums, and to exactly position the margins. Additionally, output may be saved and/or reprinted.

5.4.12 TABLE OF CONTENTS

Using a set of control cards and the Advanced Orbital Programming Branch (AOPB) Functions Tapes as input to IBM 1401, this routine

produces a Table of Contents of the AOPB Function Tapes. This Table of Contents includes each routine name, description, and where it may be found (i.e., volume number and page number) in the AOPB Function List. The Table of Contents is keyed to the Mystic list.

5.4.13 TAPE MODIFICATIONS PROGRAM

This IBM 1401 program is used to modify BCD tapes (usually BCD program tapes). The input consists of the tape to be modified and a card deck containing control cards and any desired modifications. The output is the modified tape and a listing of the modified tape, preceded by a table indicating each modification. The listing of the modified tape may be either an unedited, single-spaced listing, or an edited Mystic Listing.

5.4.14 MYSTIC LIST

This routine is used to produce edited listings of CAMEO routines and programs. The listings are used to facilitate program development and for program documentation. The IBM 1401 version accepts as input either cards or tape and produces, at the end the listing, an index of subroutines by page number and an index of suboutines by K-card. The Univac 1004 version accepts only card input and does not produce the suboutine indices.

5.4.15 IBTD TAPE DUMP ROUTINE

This routine is for the 4K 1401 (or larger) machines. It prints records up to 2347 characters. If the tape record exceeds this limit it will not be read-in nor will message be printed to so indicate.

5.4.16 FORTRAN II LANGUAGE CONVERSION PROGRAM (FORTRAN LCP)

The FORTRAN LCP is used to reduce the time and effort required to convert existing FORTRAN II programs into System/360 FORTRAN IV programs. Those statements that cannot be converted are appropriately flagged. The FORTRAN LCP can be executed on an IBM 1401 Data Processing System (8,000 positions of core storage). For details on the FORTRAN LCP, consult the IBM System/360 Transition Aids: FORTRAN II Language Conversion Program for the IBM 1401, Form C28-6560-0.

5.4.17 DOCUMENTATION AIDS SYSTEM

The Documentation Aids (AD) System is designed as an aid to documenting an existing program written in an assembly language. The DA System provides machine-generated documentation aids to those users who are programming in current IBM-supported assembly languages. The system processes, on a 1401 or 1460, source programs written in SPS, Autocoder, MAP, FAP, or Symbolic Flowchart Language. For details on the DA System, consult the Documentation Aids System (1401-SE-12X) Program Reference Manual, H20-0177-0.

5.5 CODING SHEETS

The Autocoder coding sheet (See Figure 5-3) is free-form, thus allowing the programmer greater coding flexibility. The SPS coding sheet (See Figure 5-4) is fixed-form whereby the operand portion of each line is divided into specific fields.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

AUTOCODER CODING SHEET

IBM 1240 - 1401 - 1410 - 1440 - 1460

Form X24-1350-3
Printed in U. S. A.

Program _____
Programmed by _____
Date _____

Identification 76 77 78 79 80

Sheet ___ of ___

Pg	Line	Label	Operation	Operand
1 2	3 4 5	6 7 8 9 10 11 12 13 14 15	16 17 18 19 20	21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
	0 1 0			
	0 2 0			
	0 3 0			
	0 4 0			
	0 5 0			
	0 6 0			
	0 7 0			
	0 8 0			
	0 9 0			
	1 0 0			
	1 1 0			
	1 2 0			
	1 3 0			
	1 4 0			
	1 5 0			
	1 6 0			
	1 7 0			
	1 8 0			
	1 9 0			
	2 0 0			
1 2	3 4 5	6 7 8 9 10 11 12 13 14 15	16 17 18 19 20	21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72

R0365200MPM

Figure 5-3. Autocoder Coding Sheet

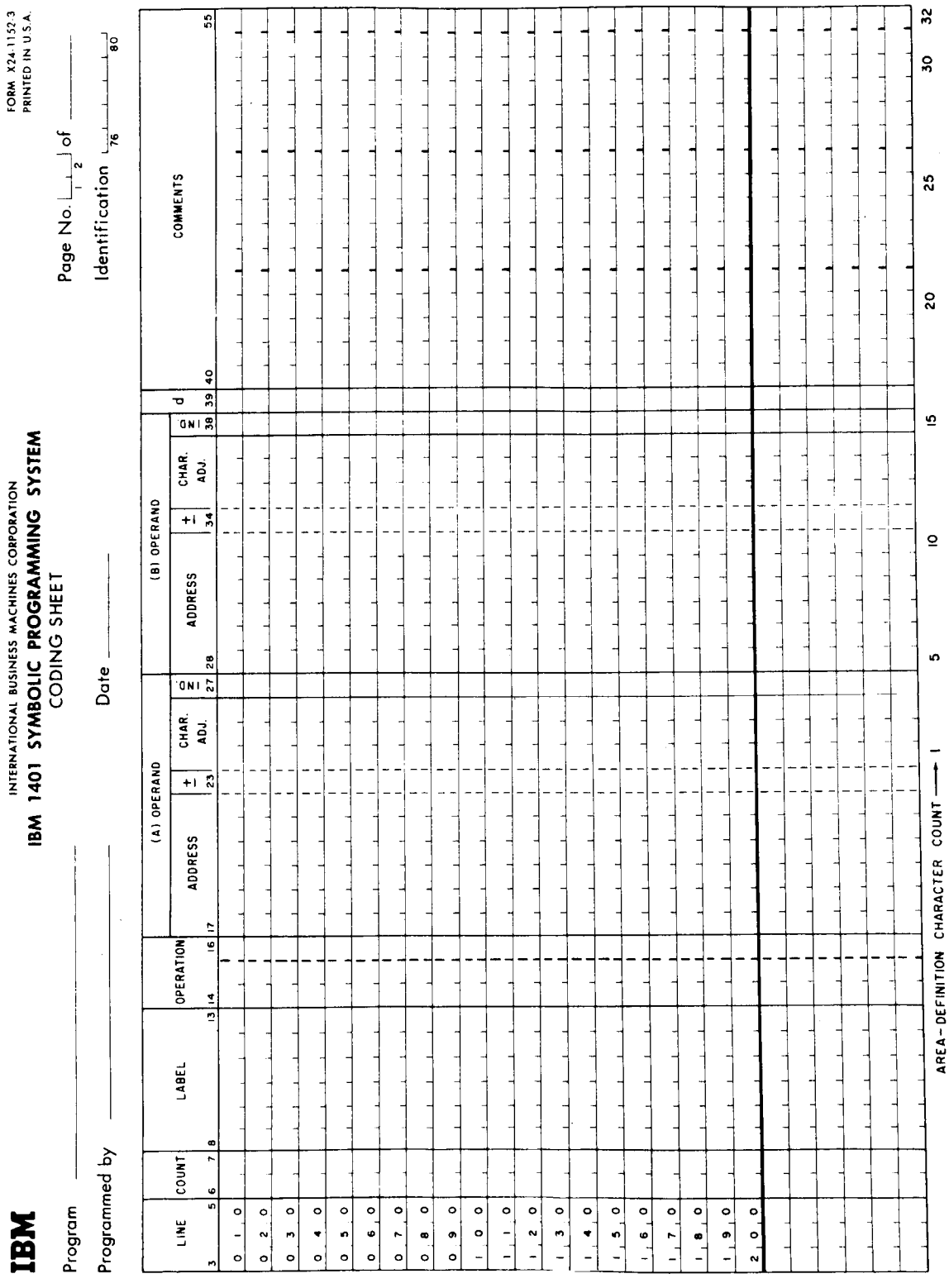


Figure 5-4. SPS Coding Sheet

CHAPTER 6

SHARE OPERATING SYSTEM

NOTE

This chapter to be provided at a later date.

CHAPTER 7

360 OPERATING SYSTEM

NOTE

This chapter to be provided at a later date.