

# Systems Reference Library

# IBM 7040/7044 Operating System (16/32K)

# Systems Programmer's Guide

This publication contains information useful to systems programmers who require a thorough understanding of the IBM 7040/7044 Operating System, Program Number 7040-PR-150. This system consists of the following:

System Monitor	(#7040-SV-951)
Input/ Output Control System	(#7040-IO-952)
Generalized Sorting System	(#7040-SM-953)
Monitored Utility Programs	(#7040-UT-975)
Processor	(#7040-PR-954)
Monitor	(#7040-SV-811)
Loader	(#7040-SV-812)
Library	(#7040-LM-813)
Macro Assembly Program	(#7040-SP-814)
FORTRAN IV Compiler	(#7040-FO-815)
COBOL Compiler	(#7040-CB-816)
Debugging Processor	(#7040-TA-817)
Update Program	(#7040-UT-955)

The publication includes descriptions of the format of the System Library, the system assembly parameters, the System Editor control cards used for system maintenance, and the contents of the Nucleus. It also contains detailed descriptions of the Processor components.

Separate publications describe the MAP, FORTRAN IV, COBOL, and Debugging languages, the Input/Output Control System, and the Generalized Sorting System. Other related publications contain information needed by applications programmers, instructions for machine operators, and information on the contents of the Subroutine Library, which is a Processor component.

Note: The IBM 1302 Disk Storage Unit is now designated the IBM 2302 Disk Storage Unit; there has been no change in the unit itself, in the applications for which the unit may be used, or in the programming parameters used to specify those applications. References in this publication to IBM 1302 Disk Storage Units should be understood to be references to the IBM 2302 Disk Storage Units.

#### **Preface**

This publication contains editing instructions and descriptive information useful to systems programmers who require a thorough understanding of the IBM 7040/7044 Operating System. It includes descriptions of the format of the System Library, the system assembly parameters, the System Editor control cards, and the contents of the Nucleus. It also includes detailed descriptions of the Processor components.

It is recommended that one person at each installation be assigned the responsibility of authorizing and documenting all changes made to the IBM 7040/7044 Operating System at the installation. This ensures adequate control over the contents of the System Library. This person and his assistants are the systems programmers for whom this publication is written.

The machine requirements of the IBM 7040/7044 Operating System are given in the publication IBM 7040/7044 Operating System (16/32K): Programmer's Guide, Form C28-6318.

It is assumed that the reader of this publication is familiar with the contents of the following IBM publications:

IBM 7040/7044 Principles of Operation, Form A22-6649

IBM 7040/7044 System Summary, Form A28-6289 IBM 7040/7044 Operating System (16/32K): Programmer's Guide, Form C28-6318

Major Revision (October 1965)

This publication, Form C28-6339-4, supersedes Form C28-6339-3 and technical newsletters N28-0520-0, N28-0525-0, and N28-0529-0. Significant changes are indicated by a dot (●) to the left of the heading on the Contents page.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

Address comments concerning the contents of this publication to:
IBM Corporation, Programming Systems Publications, Dept. D39, 1271 Avenue of the Americas, New York, N.Y. 10020

# Contents

		다. 오르스 다시 (1) 그렇게 모르스 (1) (1) (1) (1) (1) (1) (1) (1) (1) (1)	
System Library	5	System Monitor	. 36
FORMAT	5	THE Nucleus	
Structure		Words Allocated for Machine Use	
Table of Contents	6	Trap Words	. 36
Index		Engineering Words	. 36
System Programs	7	Load Area	. 36
		System Transfer Words	
System Assembly	9	System Data Areas	. 38
Assembly Parameters	9	Constants	
Machine Configuration		Variable Data	
Input/Output Device Assignment		Symbolic Units Table	
Input/Output Unit Availability	11	Control Blocks	
Assembly Options		Other Tables	
System Editor Assembly Options		Nucleus Routines	
Subroutine Library Assembly Options		System Loader Operator Interrupt Test	
Other Assembly Options		Dump Routine	44
Initial Editing Instructions		System Monitor Recall Routine	44
Initial System Unit Requirements Preparing the Backup Library	15	Return Routine	44
Maintaining the Symbolic Master File	10	Restart Routine	
Manitaning the Symbolic Master Pile	10	Installation Accounting Routine	
그 사람들은 얼마는 그들은 사람들이 되는 것이 가장 하는 것을 하셨다.		Change Communication Region Routine	. 45
System Editor		Supervisor	
SYSTEM EDITOR COMPONENTS		IMMON Routine	
Editor Monitor Input		IMAST Routine	
Editor Program		IMSYS Routine	
Phase 1		IMSRT Routine	
Phase 2		IMEXE Routine	
External Storage Unit Requirements		INJOB Routine	
EDITOR CONTROL CARDS		IMRCD Routine IMSCN Routine	. 40
• \$IBEDT Card	21	IMSCN Routine IMPAS Routine	
\$ENDEDIT Card	23	IMUQI Routine	
Editor Parameter Cards		IMINT Routine	
CALLS Card		IMLSU Routine	
ETC Card		IMDLY Routine	
INSERT Card		IMSWI Routine	. 47
REPLACE Card	25	IMATT Routine	. 47
MODIFY Card REMOVE Card	25	IMDET Routine	. 47
AFTER Card	26 26	IMCLO Routine	. 47
DUP Card		IMRES Routine	
REMARK Card		IMTIF Routine	
REWIND Card	28	IMPUN Routine	
LIBE Card		IMLIS Routine	48
LIBEND Card	28	IMUNL Routine	48 48
Using the System Editor	29	IMOID Routine IMPAU Routine	
OCT Card	29	IMSTO Routine	. 40
Preparation of Programs to be Edited	29	IMTIS Routine	
Symbolic Unit Assignment	30	IMTIM Routine	49
Examples of Edit Runs		IQRST Routine	49
Edit Run for Disk Inserting a Program into the Library		IQRSC Routine	. 49
Replacement Using Symbolic Editing		IMUNI Routine	. 49
Modification Using Symbolic Editing		IMCHA Routine	. 49
Inserting a Relocatable Subroutine	31	SCAN Routine	. 49
Adjustments After Reassembling the Nucleus	31	IMBDC Routine	. 49
Inserting a Source Language Subroutine		IMOPN Routine	. 49
Inserting a New Subroutine Library	32	INPUT/OUTPUT CONTROL SYSTEM	. 50
Inserting a Chain Program	32	● DUMP PROGRAM	. 50
Special Considerations	32	<u> </u>	
INTERFACE WITH THE PROCESSOR	33	Processor (IBJOB)	. 51
\$JEDIT Card	33	Processor Monitor	. 51
Special Conditions Messages  EDITOR PROGRAM OFF-LINE MESSAGES	33	Operation	. 51
TABLE TO BE THE TRANSPORT OF THE PARTY OF TH	4.4	DOUTING	E 1

rocessor Files	54	Intermediate Results
System Input File	54	Processor Limitations
System Output File		Input/Output Unit Allocation
System Punch File	. ວວ	Phase I, Part 1 –, Initial Edit
Alternate System Input File	55	Phase I, Part 2 — Qualification Reduction  Phase II, Part 1 — Syntax Analysis
Alternate System Output File	55	Phase II, Part 2 – Table Processing
System Load File	55 55	Phase III – Data Reduction
System Edit File	. 56	Phase IV – Procedure Generation
System Copy and Reload File Processor File Maintenance		Files Assigned to System Units
Itility Files		System Input Unit
abels	56	System Output Unit
Compiler and Assembler Work Units	56	System Peripheral Punch Unit
Available Work Units	57	COBOL Library Subroutines  MOVPAK Routine
하는 모든 이 사람은 사람은 하고 있는 그를 모든 사람들이 했다.		Compare Subroutine
oader (IBLDR)	59	Display Subroutine
Brief Description of the Loader Operation	59	Examine Subroutine
DESCRIPTION OF THE PROGRAM DECK	59	Add Subroutine
CD Card Format	59	Subtract Subroutine
Binary Card Format	59	Multiply Subroutine
Binary Control Dictionary	60	Divide Subroutine
Relocatable Binary Program Text	61	Exponentiation Subroutine Round or Truncate Subroutine
STORAGE ALLOCATION	64	Scale Subroutine Scale Subroutine
Format of Input	65	IF Alphabetic Subroutine
Operation	66	IF Numeric Subroutine
Macro Assembly Program A Brief Discussion of the Assembly Process Phase A	67	Subroutine Library FORMAT
Interface	68	Debugging Processor
Dictionary Reduction	69	Preprocessor
Phase B	69	Debug Assembly Parameters
Cross Reference Dictionary	69	Format of the DDICT
Error Editor	69	Object-Time Routines
그 마시 그 그 이 아이 있는 것이 되는 것이 되는 그 모르게 생활 경우 그 방에 그 이 전 그 전 있는 것이 되는 것이 되는 것이 있는 것이 없다.		Postprocessor
The FORTRAN Compiler	70	보는 사람들이 가는 사람이 하는 것이 살아 가는 사람들이 되었다. 그는 사람들이 살아 가는 사람들이 되었다.
A BRIEF DESCRIPTION OF THE FORTRAN COMPILER		Appendix A. System Input and Output Editors Input Editor
Definitions	70	Output Editor
Scan	70	
Storage Allocator	71	
Indexing Analyzer	71	Appendix B. 7040/7044 — 1401 Auxiliary
Instruction Generator and Index Generator	71	Programs Total Total
FORTRAN System Routines	71	REASSEMBLY OF THE 7040/7044 — 1401 AUXILIARY PROGRAMS
Format Conversion Routines	71	Input/Output Utility Program
Machine-Oriented Subroutines	73	MAP Symbolic Undate Program
System Communications Subroutines	74	원리하다 회문사는 문에 생활을 만나고 그러나는 원리에 사하는 그리스는 그는 그는 비문에 하는 이 경험이라고,
FORTRAN File Routines		
FORTRAN Routine Storage Requirements Subroutine Error Exits		Appendix C. Symbolic Unit Assignment
Input/Output Subroutine Error Exits	76	Subroutines S.SCHI Subroutine
Convert Subroutine Error Exits	76	• S.SCOD Subroutine
Machine-Oriented Subroutine Error Exits		S SCHI, Subroutine
		S.SVUN Subroutine
The COBOL Compiler	77	
Ine COBOL Compiler	77	있는 사람들 경기 보다들이 가입되어요? 사람들이 나는 사람들은 함께 그리면 살린 아무얼어나는 없는
Phase I, Language Reduction Phase II, Syntax Analysis		
Phase II, Syntax Analysis Phase III, Data Reduction	80	(IMCHA Routine)
Phase IV, Procedure Generation	80	
Special Operands	81	Index
Special Operands		

The IBM 7040/7044 Operating System consists of a collection of monitors, processors, and installation programs, which are organized into a System Library on an external storage medium. The System Library is composed of absolute programs and relocatable subroutines. In the distributed version, subroutines and tables are organized into the Subroutine Library (IBLIB), which is loaded by the Loader (IBLDR), used for relocatable object programs. The System Loader (S.SLDR), which is an absolute loader in the Nucleus, obtains system phases (for processing compilations, assemblies, sorts, etc.) from the System Library.

The System Library is generated by the System Editor on one or two system units.

#### **Format**

Every System Library generated by the System Editor is composed of absolute core storage loads, or phases. A phase of an edited program is composed of one or more physical records, or blocks, each chained to the next for loading by the System Loader. This chaining permits phases to be scatter-loaded into core storage, if necessary.

The blocks of a phase appear in a System Library in ascending order according to their load addresses. The records are ordered and chained in this way to minimize searching time during later modification to any part of the phase. A simple matching process between the records in the library and the alteration records is followed. Ordering and chaining blocks may also result in longer blocks for those programs that contain many overlaid portions.

The blocks are chained to each other by a word appended to the last word of each block. This additional word does not cause any portion of the program already loaded to be lost, as the System Loader saves and restores the locations so overlaid.

The chain word contains the load address and the word count of the next block. With this information, the System Loader can locate and save the word that is about to be overlaid by the following chain word. Bit 2 of the prefix of a chain word is set on when the next record contains a fill word. A fill word is appended to any absolute record that is only one word in length,

since all blocks must be a minimum of three words. The chain word of the last record of the phase differs from the chain words of the other records in that phase: its sign is minus, it has a word count of zero, and it contains the address of the entry point to the phase.

The System Loader finds the load address and word count of the first block in the Table of Contents.

The first block in each phase is prefixed by a one-word BCD name that is used to verify loading of the System Library and to facilitate editing. This BCD phase name of six or fewer left-justified characters is prefixed to the block during editing from the phase name appearing on the INSERT card, or it is taken from the first word of DUP input records. The addition of this prefix word does not cause the location below the load point of the record to be lost, since the System Loader saves and restores that location.

This type of System Library format permits use of a 7040/7044 Data Processing System with limited core storage to edit a library for a computer with a greater capacity. More dependable loading of the system phases can be expected with shorter records and predetermined input/output command words. Scatter-reading must be simulated for those programs not assembled in contiguous order.

Figure 1 illustrates the absolute format of a program phase in the System Library. The symbols used in that figure are defined as follows:

name

BCD phase name used to designate the core storage load; for example, FTC1, MAP1

addrn

Load address of the next block

numn

Number of words in the next block

entry

Entry point for the core storage load

The extent of a phase is determined by the input to the assembler and the Loader, and by the size of core storage; it is not influenced by the length of a block. The length of a block is determined by the length of the Editor's buffer, by the maximum length of a physical record on an external storage medium, or by the block size specified by the user on the SIBEDT card, whichever is the smallest.

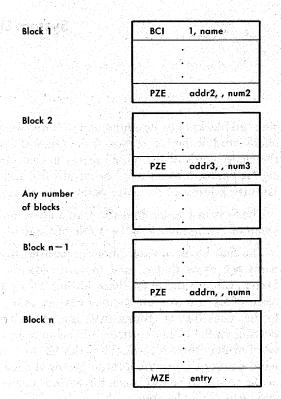


Figure 1. System Library Format

#### Structure

The minimum System Library generated by the System Editor has the following as its first six phases:

IBOOT

Device-oriented Bootstrap routine

**IBNUC** 

The system communication and data words, the Nucleus routines, and a portion of IOCS (IOEX and IOOP1)

IBSUP

The combined monitors, that is, the dependent links under IBSUP: IBJOB, which includes IBEDT, and MONITO, which includes IMSRT

IBTOC

The Index and Table of Contents for the System Library IBDMP1

Phase 1 of the Dump program

IBIOC

IOOP2, IOLS, and IOBS

In addition, the following phases are required for system operation:

IBDMP2

Phase 2 of the Dump program

IBDMP3

Phase 3 of the Dump program

A map of the contents of the System Library can be obtained during an edit run, using the System Editor.

Since the IBOOT phase that is edited into the System Library depends on the type of device that is being used to produce the new library, several forms of IBOOT

are contained within the System Editor. The distributed version of the System Editor includes incomprograms for magnetic tape and for disk and drum storage.

The IBOOT and IBNUC phases must be located on s.slbi. The other system phases may be on either s.slbi or s.slb2, but all the blocks that make up a single phase must be on the same library unit. The IBOOT, IBNUC, IBSUP, IBTOC, and IBIOC are loaded at initial start. Pressing LOAD brings in an IBOOT routine, which loads the Nucleus, including IOEX and IOOP1 with select and error recovery routines for s.slbi. Control is then transferred to the housekeeping routine of the Nucleus.

The decrement and address of the first word in the System Library contain the version number and the modification number, respectively, of the system currently in operation. These numbers are loaded into octal location 100, and control is transferred to octal location 101. The housekeeping routine later moves the contents of octal location 100 to location s.s.v.l.

The housekeeping routine transmits the trap words to lower core storage, overlaying the Bootstrap routine. The second section of the housekeeping routine sorts the system control blocks into ascending order, and the third section attaches the input/output devices, according to the specifications used in assembling the Nucleus. After initializing the Nucleus, the housekeeping routine transfers control to s.sret, and the System Loader loads the combined monitors and the higher levels of rocs, permanently overlaying the housekeeping routine.

During operation of the System, the Supervisor, the Table of Contents, and the Input/Output Control System phases (IBSUP, IBTOC, and IBIOC) are brought into core storage when required.

#### **Table of Contents**

The System Loader can pre-position the devices assigned as system library units so that they are correctly positioned when the System Loader is called to provide the next phase of a system program. This facility is particularly valuable when the System Library is distributed over more than one external storage device or when several system components share a common phase. The use of portions of the Macro Assembly Program by both the COBOL and FORTRAN compilers is an example of the latter.

To pre-position devices, the System Loader refers to the Abbreviated Table of Contents in the Nucleus, which indicates the phases that compose the pertinent section of the System Library. As each phase of a system program is called to be executed, the System Loader checks the Abbreviated Table of Contents before loading and transferring control to the phase. The Abbreviated Table of Contents is set up by each subsystem monitor from sections of the Table of Contents in the IBTOC logical record. The System Monitor sets up the Abbreviated Table of Contents for programs that are executed by a SEXECUTE card. The Loader, the Reload Program and the Restart Program create the Abbreviated Table of Contents for object programs.

An entry in the Table of Contents consists of three words; there is one entry for each phase of every system component edited into the System Library. Thus, a system component of n phases has n entries in the Table of Contents. These entries are arranged in the table in a sequence that reflects the logical flow of the component from phase to phase. The order of the entries is specified on the Calls card. (For a description of the Calls card, see the section "Editor Parameter Cards.")

Among the entries for a particular system component, there may be entries for some phases used in common with other system components. For example, some phases of the Macro Assembly Program are used for compiler input. Entries of this type are included if the names of the common phases appear on the CALLS card defining the logical flow for the system component.

The information in each Table of Contents entry is used to locate a particular logical record in the System Library. To position the system library unit, the unit number and the block number of the first block of the phase are given. To load the phase into core storage, the load address and the word count of the first block are provided.

The first three words of the Table of Contents are shown in Figure 2. In the figure, *loc* is the location of the first word of the Index or Table of Contents, and *length* is the number of words in the Index or Table of Contents.

The format of an entry in the Table of Contents is shown in Figure 3. The symbols used in that figure are defined as follows:

pfx

A prefix describing the unit on which the phase resides, as follows:

PZE Library unit; an entry created by the System Editor.

MZE Utility unit; an entry created by other programs to use the System Loader to load a phase in library format from a utility unit. (Entries of this type do not appear in the Table of Contents, but can be placed in the Abbreviated Table of Contents.)

pfn

A prefix containing the unit number. For a library unit, the prefix is either 1 or 2; for a utility unit, 0 through 7. If the prefix is 0, the unit number is in the decrement.

lognam

The BCD phase name.

Load address of the first block.

wdct

Number of words in the first block.

recno

Record number of the first block. The record number of the first block of the System Library is zero.

res

Reserved for use by the Editor.

unit

Unit number (0-99) if pfn is zero, otherwise reserved for use by the Editor.

١	Word 1 BCI	1,IBTOC
۱	Word 2 PZE	loc, , length of Index
l	Word 3 PZE	loc, , length of Table of Contents

Figure 2. The First Three Words of the Table of Contents

Γ	Word 1	BCI	1,lognam	
100	Word 2	pfx	la, res, wdct	
	Word 3	pfn	recno, , unit	

Figure 3. Format of an Entry in the Table of Contents

-1		the state of the s	 
	M/ 1 1 DOI		
٠,	Word 1 BCI	1, sysnam	
- 2		1/ Jyanuni	
-1			
п	Word 2 PZE		
	Word 2 PZE	reladd, , count	
		retada, , coom	and the second second
			the state of the s
	the state of the s	the state of the s	The second of the second

Figure 4. Format of an Entry in the Index

#### Index

The Index is provided to aid a monitor in finding the pertinent entries in the Table of Contents. It contains a two-word entry for each system component, as shown in Figure 4. Word 1 contains the BCD name of the system component. This is usually the name used on the card that causes control to be passed to a particular program, such as IBFTC for the FORTRAN Compiler. The address of Word 2 contains the location, relative to the beginning of the Table of Contents, of the first word of the entries for the system component. The decrement of Word 2 contains the total number of words occupied by the entries for the system component.

#### System Programs

Following the IBTOC phase are the phases of systems edited into the System Library, as well as any subroutine libraries, such as IBLIB, generated by the System Editor. The sequence of programs in the Library, the position of the end-of-file trailer label, and the

distribution of the System Library between the library units are controlled by parameters that are supplied as input to the System Editor. For example, Phases 1 and 2 of the Editor program can be located anywhere in the System Library.

If the System Library resides on one unit, the last record on the unit is the 120-character end-of-file (1EOF) trailer label. If the library is distributed between two units, the end-of-reel (1EOR) trailer label is the last record on the primary unit.

The Nucleus of the IBM 7040/7044 Operating System is assembled using the distributed version of the Nucleus and the Macro Assembly Program. This assembly is divided into two core storage loads or phases that are delimited by a TCD pseudo-operation in the MAP symbolic deck. This deck contains all of the installation options that determine the contents of the Nucleus.

All other system parts are assembled in relocatable form. Their absolute locations are determined by the Loader (IBLDR) after the Nucleus is established.

# **Assembly Parameters**

The options that permit an installation systems programmer to specify much of the content of the Nucleus during assembly of the system are described in the following paragraphs.

### **Machine Configuration**

It is necessary to define the configuration of the 7040/7044 Data Processing System for which the Operating System is being used, so that the proper amount of core storage can be allotted for unit control blocks and system control blocks. The MAP symbolic statements used to define the number of input/output devices present on each channel are shown in Figure 5. Each of these definitions must be included in the symbolic input deck for the Nucleus, even if its variable field is zero:

The letters xyz denote the number of devices attached to 1414 Input/Output Synchronizer interfaces, numbered 1, 2, and 3, respectively. The number of devices that may be attached to an interface is limited as follows:

0 or 1 for dtua, ptra, prta, rpua

0, 1, or 2 for RIUA

0, 1, 2, or 3 for TTIA, TTOA

The letter p is either 1 or 0, indicating the presence or absence of the device. The letter n may be any decimal number from 0 through 10, and q may be any digit from 0 through 6. Disk modules are defined as being present or absent by using 1 or 0, respectively. Thus, modules 0 through 4 are defined by a series of five digits, here represented as *abcde*. The series fghij represents the definition of modules 5 through 9.

Space is reserved in the Nucleus for control blocks for each device in the 7040/7044 Data Processing System, according to the following rules:

1. Nine words for each unit control block.

2. Four words for each system control block for

Symbol	Code	Variable	Remark
TTIA	SET	xyz	INPUT TELEGRAPH UNITS ON CHAN A
TTOA	SET	xyz	OUTPUT TELEGRAPH UNITS ON CHAN A
DTUA	SET	xyz	DATA TRANSMISSION UNITS ON CHAN A
RIUA	SET	хуz	REMOTE INQUIRY UNITS ON CHAN A
PTRA	SET	xyz	PAPER TAPE READERS ON CHAN
PRTA	SET	xyz	PRINTERS ON CHAN A
RPUA	SET	xyz	READER/PUNCHES ON CHAN A
TAPA	SET	n	TAPES ON CHAN A
PRTS	SET	P	PRINTER ON ON-LINE 1401
RPUS	SET	P	READER/PUNCH ON 1401
TAPS	SET	q	TAPES ON ON-LINE 1401
TAPB	SET	n	TAPES ON CHAN B
DF1B	SET	abcde	DISK ACCESS 0, MODULES 0-4
DF2B	SET	fghij	DISK ACCESS 0, MODULES 5-9
DF3B	SET	abcde	DISK ACCESS 1, MODULES 0-4
DF4B	SET	fghij	DISK ACCESS 1, MODULES 5-9
DDIB	SET	þ	DIRECT DATA CONNECTION ON CHAN B
ВМХВ	SET	i Postovije.	COMMUNICATION CONTROL SYSTEM ON CHAN B
TAPC	SET	n	TAPES ON CHAN C
DFIC	SET	abcde	DISK ACCESS 0, MODULES 0-4
DF2C	SET	fghij	DISK ACCESS 0, MODULES 5-9
DF3C	SET	abcde	DISK ACCESS 1, MODULES 0-4
DF4C	SET	fghij	DISK ACCESS 1, MODULES 5-9
DDIC	SET	P	DIRECT DATA CONNECTION ON CHAN C
вмхс	SET	p	COMMUNICATION CONTROL SYSTEM ON CHAN C
TAPD	SET	n	TAPES ON CHAN D
DFID	SET	abcde	DISK ACCESS 0, MODULES 0-4
DF2D	SET	fghij	DISK ACCESS 0, MODULES 5-9
DF3D	SET	abcde	DISK ACCESS 1, MODULES 0-4
DF4D	SET	fghij	DISK ACCESS 1, MODULES 5-9
DDID	SET	P	DIRECT DATA CONNECTION ON CHAN D
BMXD	SET.	<b></b>	COMMUNICATION CONTROL
TAPE	SET		TAPES ON CHAN E
DF1E	SET	- " 	그리트 아이들은 사람들이 살아 가는 것이 없었다.
		abcde	DISK ACCESS 0, MODULES 0-4
DF2E	SET	fghij	DISK ACCESS 0, MODULES 5-9
DF3E	SET	abcde	DISK ACCESS 1, MODULES 0-4
DF4E	SET	fghij	DISK ACCESS 1, MODULES 5-9
DDIE	SET	in Partosais Jangan	DIRECT DATA CONNECTION ON CHAN E
BMXE	SET	p	COMMUNICATION CONTROL SYSTEM ON CHAN E

Figure 5. Symbolic Definition of the Machine Configuration

magnetic tape units, unit record equipment, telecommunications devices attached through Model IV or V of the 1414 Input/Output Synchronizer, and 7740 Communication Control System input devices.

3. Eight words for each system control block for disk and drum storage units and 7740 Communication Control System output devices.

Each card reader/punch, 1009 Data Transmission Unit, 1014 Remote Inquiry Unit, 7740 Communication Control System, and telegraph unit is considered to be two devices requiring two unit control blocks - one for input, one for output. A disk or drum module is treated as a device that requires one unit control block. A device may require one or more system control blocks, depending on the logical unit assignments made with the ATTACH macro-instruction described later in the text.

#### Input/Output Device Assignment

All input/output devices and each system unit must be defined at assembly time by an ATTACH macroinstruction. This macro-instruction is used to construct the Symbolic Units Table. If changes must be made in this table, the ATTACH macro-instructions can be modified and the Nucleus reassembled, or the SDETACH card and the sattach card can be used.

Care should be taken in the assignment of system utility units so that no priority program will use a utility unit specified for use by a system program or an installation program.

In general, each system unit should refer to a unique physical unit. However, the system checkpoint unit (s.sck1) may be the same physical unit as a system utility unit (s.suxx). Note that this may cause conflicts in unit assignments during use of certain features of the Operating System (Sort, load-time debugging, the Chain feature, the Copy and Reload features).

The format of the ATTACH macro-instruction is as follows:

44.12	8		16				<u> </u>
	A	TTACH	S.Sxx	x, devic	e, chan,	number,	type
			[, dir	[, n [,	from, to	$\begin{bmatrix} , \\ 157 \\ 158 \end{bmatrix}$	}]

The arguments in the variable field are separated by commas. They are:

This must be one of the following mnemonics:

MNEMONIC	SYSTEM FUNCTION
S.SLB1	Library 1
S.SLB2	Library 2
S.SIN1	Input 1
S.SIN2	Input 2
S.SOU1	Output 1
S.SOU2	Output 2
S.SPP1	Punch 1

MNEMONIC SYSTEM FUNCTION	<b>)1</b> \
S.SPP2 Punch 2	
S.SCK1 Checkpoint	
S.SU00 Utility 00	. 19
S.SU01 Utility 01	
S.SUnn Utility nn	

device

ice	그리고 있으로 가는 사람들이 살아 있다면 하는데 되었다.
his ma	y be any of the following symbols:
Т	Magnetic tape
DT	Data transmission unit
PT	Paper tape reader
RI	Remote inquiry unit
D	Disk or drum storage
RD	Card reader
PU	Card punch
PR	Printer
TT	Telegraph unit
BX	Communication control system

chan

Channel or channel A interface to which the device is connected. It must be one of the following characters, consistent with device and system configuration: A, B, C, D, E, S for the appropriate channel; or in the case of devices attached to channel A (other than magnetic tape units), the appropriate 1, 2, or 3 interface to which the device is attached. The letter S indicates that the device is attached to channel A through an on-line 1401. For example, the 1403 Printer can be attached only to interface 1, 2, or 3, or to channel S.

The systems engineer should determine from the customer engineer the appropriate 1414 I/O Synchronizer interface number to which the unit record equipment has been attached. Generally, when unit record devices are attached, interface numbers are assigned in the following manner: first device of a type, interface number 3; second device, interface number 1; third device, interface number 2.

number

This is the device number. It must be one of the following numbers, consistent with device, channel, and system configuration: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, or 20. The numbers 0 through 9 are used for tape devices; the numbers 1 through 10 are used for 1301 Disk Storage, 1302 Disk Storage, and 7320 Drum Storage, the numbers 11 through 20 are used for 1302 Disk Storage.

The device number may not be larger than the quantity of these devices defined as available on the channel; however, device numbers 11 through 20 are treated in the same manner as device numbers 1 through 10. Since each access mechanism is treated as a device, "number," for example, is a 1 for access 0 module 0, 10 for access 0 module 9, 11 for access 1 module 0, etc. (see Figure 6). For magnetic tape units, this digit designates the number that is dialed on the tape unit.

This must be one of the following equipment type numbers, consistent with device, channel, and system configuration: 729, 1009, 1011, 1014, 1301, 1302, 1402, 1403, 1622, 7320, 7330, 7740, and telegraph.

dir

One of the following symbols: I for an input unit, O for an output unit, or blank for both. This option is required for devices that need separate input and output unit control blocks (1009, 1014, 7740, and telegraph).

The following arguments are omitted for sequential access devices, such as unit record equipment or magnetic tape units:

This argument is used to specify the method of using a random access device such as disk storage. It is one of the following:

R Random access.

FT Full track with record addresses.
CY Cylinder mode (optional feature).

x Single-record operation. The letter x represents the number (1-63) of records defined on one format track.

from

This is the logical starting point for the function within the device. It is the cylinder number of the first cylinder assigned to the function: 0-249 for disk storage; 0-9 for drum storage.

This is the logical ending point of the function within the device. It is the number of the last cylinder assigned to the function: 0-249 for disk storage; 0-9 for drum storage. This argument must be supplied for functions for which a logical starting point is specified.

157 or 158

This is the permanent reservation code (718 or 728) to be appended to the device being attached. All preceding fields, if null, must be indicated by commas.

If a logical starting point and a logical ending point are not specified for a random access device, the function is assumed to occupy the entire device. The arguments used in the ATTACH macro-instruction must appear in the order presented.

Access 0				111		
module 0 1 2 3 4	5	6	7	8	9	
number 1 2 3 4 5	6	7	8	9	10	
Access 1 module 0 1 2 3 4	5	6	7	8	9	
number 11 12 13 14 15	16	17	18	19	20	Service of

Figure 6. Disk and Drum Device Numbers

After the Nucleus is initialized at an initial start, each entry in the Symbolic Units Table has the form:

In this table, loc(ucb) is the location of several unit control words describing the device and its status, and loc(scb) is the location of several system control words that further describe the symbolic unit. S.Sxxx is as defined for the ATTACH macro-instruction.

An explanation of the prefix, pfx, may be found under "Symbolic Units Table." The prefix should be set by object programs not using the 10BS level of 10CS. If a particular symbolic unit has no device attached to it, the address and decrement of the appropriate entry are set to zero and the symbolic unit may not be used.

If the parameters of an ATTACH macro-instruction refer to a channel that has not been defined, the appropriate entry in the Symbolic Units Table is set to zero and the corresponding symbolic unit may not be used. The table entry is set at initial start when a \*RESTORE card is encountered, or when \*ATTACH cards are processed.

The following is an example of an ATTACH macro-instruction:

This card causes a 729 Magnetic Tape Unit with a dial setting of 1 on channel C to be assigned as the primary system library unit.

# Input/Output Unit Availability

Disk and drum units are considered available for use under the variable unit selection techniques only if their formats, as specified in the system control blocks, match one of the allowable formats specified in the ALLOW macro-instruction.

ALLOW is punched in columns 8-12 of the ALLOW macro-instruction. The variable field starts in column 16. The arguments in the variable field are separated by commas. They are:

FORMAT		CODE (OCTAL) IN SYSTEM
TYPE	MEANING	CONTROL BLOCK
R	Random	20
SR	Single record	21
FT	Full track	22
C1301	Cylinder 1301	$\frac{\overline{23}}{23}$
C7320	Cylinder 7320	$\frac{1}{24}$
C1302	Cylinder 1302	25

If an installation does not wish to have disk or drum units of any format type chosen as available units, an ALLOW macro-instruction with a blank variable field must be used.

The following is an example of an ALLOW macro-instruction:

This macro-instruction will cause the selection of only those disk and drum units that are attached as full track.

#### **Assembly Options**

The following parameters allow the system programmer to specify the conditions that apply at his installation. The values given for n are the equivalences in the distributed version of the system.

SYMBOL	CODE	VARIABLE	DESCRIPTION
IBNUC	BOOL	n = 135	The octal number n is the
			location of the first system
S.SORG	BOOL	n=3000	transfer word, S.SLDR.  The octal number is the
	DOOD.	n-0000	greater of the following:
i karaja kutij			1. The location of the
			first word available to ob-
			ject programs if the value
			of S.SORG is greater than
			IP1ND (the limit of the low-
o neto kaltura y di. Solonia di Selonia			est level of IOCS).
			2. The end of protected
S.SEND	DOOL	07777	storage.
3.3END	BOOL	n = 37777	The octal number n is the
			location of the last word available to the system.
IFRPM	SET	n=0	If the system is to be stor-
			age protected, n is 1; other-
			wise, n is 0.
IFCLK	SET	n=0	If the interval timer may
Raye Kar		er silver og e	be used by the system, n
	198	-446	is 1; otherwise, n is 0.

SYMBOL	CODE	VARIABLE	DESCRIPTION	SYMBOL	CODE	VARIABLE	DESCRIPTION
MXCLK	EQU	n=3	The decimal number n is the maximum number of minutes that a job may run if a \$TIME card is not used. This is not significant un- less IFCLK is 1.				unit. The decimal number n is the number of tracks that should remain unused on the unit at the time the end-of-medium return is taken.
IBPER	EQU	n=10	The decimal number n is the percentage of redun- dancies that, if exceeded, causes an on-line message to be typed when a \$STOP	IBLTZ	SET	n=41	The decimal number n is the number of entries in the Abbreviated Table of Contents.
er Politikalija erak Julious e <b>n</b> red			card is read. It may range from 0 through 1,000 in tenths of a percent.	IBRTZ	SET		The decimal number n is the number of entries in the Recognizable Control Card Table, S.SRCC.
RDNRT	EQU	n=100	The decimal number n is the number of times the input/output operation will be repeated in attempts to recover from redundancy errors.	IBRSL	SET	n=0	The decimal number n speci- fies the number of priority levels (1-8) that determine whether a routine can be interrupted to permit exe- cution of a special routine
RUPSW	EQU	n=1	The decimal number n, 1-6, is the sense switch to be used for operator interruption of a stack of jobs.		O de sei O seculos O seculos	i i propinsi serimba Propinsi serimba Propinsi serimba	of a higher priority. When IBRSL is 0, no special rou- tine can be executed. It gov- erns the length of the push-
CYLMO	SET	n=0	This option indicates whether the cylinder mode is present on the 7631 file control for the 1301 disk, 1302 disk or 7320 drum as follows:				down list in IOEX used to save the registers of the interrupted routines. This number is not identical to the number of priorities permitted.
			0—The cylinder mode is not present.  1—The cylinder mode is present.	IFSNS	SET	n=0	This option specifies how IOOP handles the IOSNS operation as follows:  0 – IOOP will ignore the
IFSHF	SET	n=0	This option indicates whether a 7631 file control is shared with another host			eranie anie in a Lieu dina delina Lieu dina delina	operation.  1 – IOOP will sense the specified device.
			computer.  0 — File control is not shared  1 — File control is shared	IFCHO	SET	<b>n=0</b>	If n is 0, IBJOB and IBEDT will choose work units without attempting to optimize channel usage. If n is 1,
LABELS	SET	n=1	This option specifies the use that an installation makes of labeled storage media as follows:	i i Berge (Aerica La Beneria Inc. La Beneria (		ik godeni (i di pojet den koje den godender	IBJOB and IBEDT will attempt to optimize channel usage in choosing work units. If n is 1, IFSNS should be set to 1.
	is Politica Politica		0—The IOLS creation and verification routines do not appear on the System Library.  1—The IOLS creation	IFLSC	SET	n=0	This option specifies whether the installation uses de- ferred label searching as follows:
			and verification routines appear on the System Library, but they are not used to verify the retention dates of output reels.  2—The IOLS creation	i i santa fisi i Vittables Vittables Vittables i sant Vittables i santa Vittables i santa			<ul> <li>0 – The installation does not use deferred label searching.</li> <li>1 – The installation uses deferred label searching.</li> </ul>
		보면 10명 (1997) 1일 - 1일 - 1일 (1997) 1일 - 1998 - 1998	and verification routines ap- pear on the System Li-	IF162	SET	n=0	Non-zero if the card reader is a 1622.
			brary and must be used to verify the retention dates of output reels.	CPRPU	EQU		Assembled combined system print/punch flag setting.
			The IOLS reel-handling routines are not affected by this parameter.	IFURA	SET	n=0	If n is 0, IOCS does not respond to unit record interrupts. If n is 1, IOCS
LABSW	EQU	n=6	The decimal number n is the sense switch tested by IOLS.				responds to unit record interrupts originating from a 1402 Card Read Punch
EOMTR	EQU	n=4	This option is used when a disk or drum storage area is assigned as a symbolic				or a 1403 Printer. If IF162 is 1, IFURA will be assumed to be 0.

	ga.	
SYMBOL	CODE VARIABLE	E DESCRIPTION
LISTSW	SET n=0	If a listing of all unassembled cards in IBNUC is to be provided, n is 1; otherwise, n is 0.
	SET n=0	This option specifies how IOCS will handle errors on magnetic tape. If n is 0, IOCS will distinguish between incomplete word errors and redundancy errors. If n is 1, IOCS will treat incomplete word errors as if they were redundancy errors.
TPOPN	SET n=0	If n is 0, IOCS discards a telecommunications message received prior to the opening of telecommunications input. If n is 1, a waiting message is accepted upon opening, i.e., the IOCS special routine is entered.
IFTYP	SET n=0	If n is non-zero, the type-writer routine, S.XPRT, is assembled to execute in single-character mode. If Tele-processing devices are attached to channel A, IFTYP will be assumed to be non-zero.
	SET n=0	If n is 1, devices defined in IBNUC as attached to a nonexistent channel will be detached by IBSUP. Further, if n is 1, disk or drum units which have been defined in IBNUC but which are not ready will also be detached by IBSUP. The parameter on the system tape is n=0 because the version of IBNUC distributed to the user defines disk units on channels B and C. Individual systems without a disk or a drum unit physically attached to channels B and C and without Control Adapter 1074 will hang up in IBNUC if n=1. If the input/output configuration defined in IBNUC matches the system configuration, n should be 1.
SHARE	SET n=1	If n is 1, the SHARE set of data record control characters is used. The character M is taken as an indication that the record is binary.  If n is 0, the alternate set of data record control characters is used

acters is used.

For additional informa-

tion, see the section on

record formats in the publication IBM 7040/7044

Operating System (16/32K):

Input/Output Control Sys-

tem, Form C28-6309.

Priority Processing: The capability for priority processing is controlled by the assembly option IBRSL. A specification of:

1	8	16	<u> </u>			
IBRSL	SET	0	1-3360	right stay	And the	- 17

prevents any priority programming from taking place. A main program cannot be interrupted by a real-time routine or by a random processing routine.

				4.4.2.3				
. Δ	an	00	:4:	201	in.	~ ` ~	٠Ľ.	
A	งม	CC.	ш	aı	Ю		)1:	

	경화하는 큰 사람들이 살아 하는 살아가 다.					* * *	
	1 8	16			100		
٠	The second secon	 -					
			 1000	-	111111111	 	
	IBRSL SET	1					
		 -	10 1 1 1 A 1				

permits all priority processing. However, if a special routine has been entered, no other special routine can be entered, no matter how high its priority may be, until the special routine in control has been completed. Eventually, all special routines are entered and completed and the main program is permitted to continue. This specification entails the addition of approximately 150 instructions in IOEX, as well as a pushdown list of 16 words.

Each additional nesting level that is specified requires an additional 16 words in the pushdown list. Each additional nesting level permits an additional level of priority processing; for instance, it is less likely that a higher priority program will find the pushdown list full and will have to wait until a lower priority routine has finished. The total length of time in special routines, until the main program is permitted to continue, is not affected by the nesting level.

Priorities 0 through 7 are available to object programs.

Unit Record Interrupts: Response to unit record interrupts is controlled by the assembly option IFURA. A specification of:

prevents response to unit record interrupts. If a unit record operation is scheduled, but the device is busy, and no other activity on any channel is in progress or can be scheduled, IOEX waits in a hold loop until the unit record device is no longer busy.

A specification of:

permits response to unit record interrupts. If a unit record operation is scheduled, but the device is busy, and no other activity on any channel is in progress or can be scheduled, IOEX returns to the calling program. A unit record interrupt trap occurs when the device is no longer busy.

This provides additional processing time following a call to s.xact or following an Initiate or Return Im-

mediate call to s.100r. The additional time is used if the call for the next operation on a unit record device is made immediately after the end-of-operation trap for the current operation on the same device. Any calculations performed between the two requests for unit record operations reduce the benefit of the unit record interrupt trap.

In general, a program is either process limited or limited to the sum of iocs trap-processing time and transmission time on the busiest channel. If the program is completely process limited, no benefit accrues from unit record interrupt traps. Programs that tend to fall into a set input/output pattern accrue limited benefit from unit record interrupt traps. Thus, this specification is most useful when ibrsl is not zero, since the intermittent processing on unit record devices tends to break the input/output pattern, and the quicker return from ioex assists the interrupted program.

## System Editor Assembly Options

The following symbol definitions appear in the System Editor symbolic decks.

SYMBOL	CODE	VARIABLE	DESCRIPTION
IER01	EQU	n=1000	The size of the largest buffer the System Editor may use.
IER03	EQU	n=330	Word length of physical system records on magnetic tape units on the 1401. IOCP and IBEDT allow the 1401 buffer size to be up to 370.
IELFG	EQU	n=16	The decimal number n is the position in the Nucleus, relative to S.SFLG, of the System Editor's flag words.
IELCT	EQU	n=20	The decimal number n is the number of cards per block in the IBLIB subrou- tines for the Processor.
IEWST	EQU	n=1	The decimal number n is the number of words re- quired by IOOP in each buffer.

#### Subroutine Library Assembly Options

The following symbol definitions appear in the symbolic input to the Subroutine Library:

SYMBOL	CODE	VARIABLE	DESCRIPTION
ERRIN	SET	n=0	ERRIN must be set to the same value in decks
			INSYFB and IASYFB. If
			S.SIN1 is attached to a
		Last Miles VAV	unit record device and a
			read error occurs, retry is permitted if n is 1.
ERROU	SET	n=0	ERROU must be set to
			the same value in decks
			OUSYFB and OASYFB. If
			S.SOU1 is attached to a
			unit record device and a

SYMBOL	CODE	VARIABLE	DESCRIPTION
ERRPP	SET	n=0	print error occurs, the error is ignored if n is 1. ERRPP appears in deck PPSYFB. If S.SPP1 is attached to a unit record device and a punch error occurs, the error is ignored if n is 1.
SUPRES	SET	n=1	This option specifies a feature in the ECV routine (see "FORTRAN System Subroutines").
EXTRA	SET	n=0	This option specifies a feature in the FCV routine (see "FORTRAN System Subroutines").
FPTLIM FPULIM ODDLIM MESLIM DPOPT	EQU	n=10 n=10 n=1 n=20 n=1	These options specify features in the FPT subroutine (see "FORTRAN System Subroutines").

*Unit Record Error Returns:* These returns are controlled by the following cards:

1 8 16			***	*	
ERRIN SET 0 o	r 1				

A specification of 0 causes the system file block s.fbin (deck insyfb) to specify that the error return s.iner is to be assembled as equ 0. When any error occurs while reading, iobs performs its standard error procedure. If the system input unit is a unit record device and the error occurs on this device, no retry is permitted; the job is terminated.

A specification of 1 causes a control section (s.iner) to be assembled. This control section replaces the standard error recovery routine in 10Bs. If the system input unit is a unit record device and either an unrecoverable error occurs while reading or an incomplete word is typed, the computer pauses to allow the operator to reload the card reader, and the card is reread.

ERRIN must be set to the same value in decks IASYFB and INSYFB.

The format of the ERROU card is:

그렇게 하는 하는 장마는 사람들이 가는 그 사람이 되었다. 그리고 있는 것이 되었다.		A
1 8 16		
annorr one		
ERROU SET 0 or 1	and the second	

A specification of 0 causes the system file block s.fbou (deck ousyfb) to specify that the error return s.ouer is to be assembled as equ 0. When any error occurs while writing, iobs performs its standard error procedure. If the system output unit is a unit record device and the error occurs on this device, the job is terminated.

A specification of 1 causes a control section (s.ouer) to be assembled. This control section replaces the standard error recovery routine in 10BS. If the system output unit is a unit record device and an unrecover-

able error occurs while writing, then a message is written on the system output unit, the erroneous record is ignored, and the job continues.

ERROU must be set to the same value in decks OASYFB and OUSYFB.

The format of the ERRPP card is:

A specification of 0 causes the system file block s.fbpp (deck ppsyfb) to specify that the error return s.ppeb is to be assembled as equ 0. When any error occurs while writing, 10Bs performs its standard error procedure. If the system punch unit is a unit record device and the error occurs on this device, the job is terminated.

A specification of 1 causes a control section (s.pper) to be assembled. This control section replaces the standard error recovery routine in 10Bs. If the system punch unit is a unit record device and an unrecoverable error occurs while punching, then a message is typed, the computer pauses to allow the operator to mark the deck, the erroneously punched card is ignored, and the job continues.

#### Other Assembly Options

Assembly options related to specific subsystems are discussed elsewhere. Options in the Macro Assembly program, which apply to disk and drum orders, are described later, in the section "Phase A." Options in the Dump program are described in the publication IBM 7040/7044 Operating System (16/32K): Debugging Facilities, Form C28-6803. Assembly options in the Sort program are discussed in the publication IBM 7040/7044 Operating System (16/32K): Generalized Sorting System, Form C28-6337.

## Initial Editing Instructions

The binary system tape, distributed by the IBM Data Processing Division Program Information Department, can be used immediately at an installation as a 7040/7044 Operating System. Its first application should be the generation of an operating system adapted to the particular needs of that installation.

The configuration of the 7040/7044 Data Processing System for which the distributed system tape is assembled is shown in Figure 7 in the format of the ATTACH macro-instruction.

## **Initial System Unit Requirements**

The systems programmer determines the changes, if any, that must be made in the configuration of the

S.SLB1,T,C,1,729	
S.SIN1,T,C,2,729	
S.SOU1,T,C,3,729	
S.SPP1,T,B,4,729	
S.SCK1,T,B,1,729 .	
S.SU00,T,B,2,729	
S.SU01,T,B,3,729	그런 아를 막지겠다고 뭐니
S.SU02,T,C,4,729	
S.SU03,T,A,1,729	
S.SU04,T,A,2,729	
S.SU05,T,A,3,729	
S.SU06,T,A,4,729	
S.SU07,T,C,5,729	
S.SU08,T,A,5,729	
S.SU09,T,A,6,729	
S.SU10,T,A,7,729	
S.SU11,T,B,5,729	These entries
S.SU12,T,S,1,729	are in the
S.SU13,RD,3,1,1402	format of the
S.SU14,RD,S,1,1402	ATTACH macro-
S.SU15,PU,3,1,1402	instruction.
S.SU16,PU,S,1,1402	
S.SU17,PR,3,1,1403	
S.SU18,PR,S,1,1403	tina jestaliai ir v. gri
S.SU19,D,B,1,1301,FT,0,99	
S.SU20,D,B,1,1301,FT,100,149	
S.SU21,D,B,1,1301,FT,150,199	
S.SU22,D,B,1,1301,FT,200,249	
S.SU23,D,B,2,1301,FT	
S.SU24,D,C,1,1301,FT,0,99	
S.SU25,D,C,1,1301,FT,100,149	
S.SU26,D,C,1,1301,FT,150,199	
S.SU27,D,C,1,1301,FT,200,249	
S.SU28,D,C,2,1301,FT	
S.SU29,T,B,6,729	
S.SU30,T,B,7,729	

Figure 7. Machine Configuration for the Distributed System

7040/7044 Operating System at his installation to permit him to make the initial edit runs with the distributed system tape. He then prepares the necessary switch cards to effect these changes. If the 7040/7044 Data Processing System at his installation does not have an on-line card reader, he can switch units through the entry keys. The minimum configuration that he must have to prepare the backup library is:

SYMBOLIC UNIT	DEVICE	CHANNEL
S.SLB1	729 Tape	Α
S.SIN1	1402 Card Read Punch	Ā
S.SOU1	1403 Printer	Ä
S.SU00	729 Tape	Ā
S.SU01	729 Tape	Ā

Both utility units may be attached to different cylinders on the same module of disk storage.

## **Preparing the Backup Library**

This procedure is optional, but strongly recommended, as it provides system maps and a duplicate library for emergency use.

- 1. Place the following cards on the system input file:
  - \$DATE
  - \$IBEDT
  - \$ENDEDIT
  - \$IBSYS
  - \$STOP
- 2. Mount the system tape.
- 3. Follow the initial start procedures given in the publication *IBM* 7040/7044 Operating System (16/32K): Operator's Guide, Form C28-6338.

## Maintaining the Symbolic Master File

From the listing of symbolic cards that is obtained from the symbolic master file and from the assembly

parameters described in this publication, determine the serial numbers of the symbolic cards that must be replaced on the symbolic master file. Prepare the change cards, and insert the changed assembly parameters into the symbolic master file.

The first deck on reel 1 of the distributed symbolic master tape contains comments cards (CCARDS). This deck contains information regarding the contents of the symbolic tapes and describing the runs necessary to maintain the distributed tapes and to create a System Library. Specific deck setups that include the control cards for each deck in the Subroutine Library and each phase of the Operating System are also shown.

The System Editor (IBEDT) is used to maintain the System Library of the IBM 7040/7044 Operating System. With the proper control cards, the user can add, delete, or modify any absolute records of system programs or add or delete any relocatable records from the Subroutine Library (IBLIB). The System Editor itself is incorporated in the System Library as a monitored subsystem that is called through the System Monitor. The System Editor can be used with the Loader (IBLDR), or with a processor that an installation may edit into the System Library to generate and insert new library records. The user's programs, or any subroutines or data, are thus added to the System Library in the proper format.

The System Editor can also be used to place a program that was compiled and/or assembled by the Processor into the System Library in absolute machine language format or in the relocatable subroutine format. A program added in absolute format can be loaded and executed with the SEXECUTE card. In the case of subroutines added in relocatable format, the programmer directs the Loader (IBLDR) to load the subroutine by using the EXTERN pseudo-operation in his coding. In this way, the user can create a specific operating system for his own installation. If desired, an installation monitor could be placed into the combined monitor core storage loads for monitoring these special subsystems. It would be called by the \$xxxxx control card specified by the installation, where xxxxxx is the name given to the installation monitor.

# System Editor Components

The System Editor consists of two logical sections: the Editor Monitor and the Editor program, which is further divided into Phase 1 and Phase 2.

#### **Editor Monitor**

The Editor Monitor is part of the combined monitor core storage loads. It is entered directly from the System Monitor when a SIBEDT card is encountered. A description of the SIBEDT card is in the section "Editor Control Cards."

The Editor Monitor processes the Editor control cards and, in effect, the Processor control cards, if any, before the editing process begins. This monitor also determines which input/output units will be used and initializes them for the Editor program. If the LABEL

option is specified on the SIBEDT card, the Editor Monitor checks the labels of the old library unit(s) containing the System Library to be updated and/or the new library unit(s) on which the new System Library is to be created. If the assembly parameter LABELS was defined as LABELS SET 2, header labels on the output units are checked for retention dates.

The Editor Monitor, in effect, directs the use of the Processor Monitor (or any other processor that the installation has edited into the System Library) during edit runs involving source language input or relocatable input (see Figure 8).

#### Input

The System Editor can accept the following types of input:

- 1. MAP, FORTRAN, or COBOL source language coding
- 2. OCT instruction cards
- 3. Absolute column binary cards
- 4. Relocatable binary cards
- 5. Source language statements
- 6. Records in System Library format

Input to the Editor program on ocr instruction cards for patching, or in the absolute column binary format produced by an assembler such as the 7040/7044 Basic Assembly Program (see Figure 9), is passed directly to the Editor program.

The Editor program verifies check sums except in the following cases:

- 1. When duplicating with the DUP card
- 2. When the check-sum word is zero
- 3. When bit 2 of the load control word is punched

The first instruction in the card is loaded in the core storage word whose actual address is specified in bit positions 21-35 of word 1 of the card; other instructions in the card are loaded in words with ascending sequential addresses. The decrement and address fields within the instructions cannot be modified during loading.

This absolute column binary format should not be confused with the ABSMOD output of the Macro Assembly Program, which has an absolute origin but is essentially in relocatable format.

Under the control of the Editor Monitor, the Loader (IBLDR) may be used to convert relocatable binary output from the Macro Assembly Program into the System Library binary format acceptable to the Editor program. These converted records

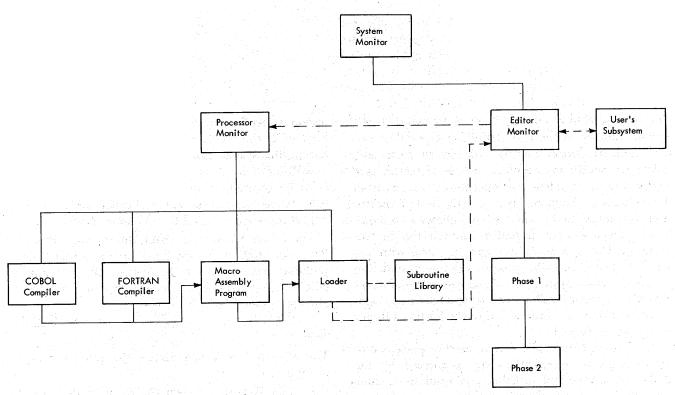
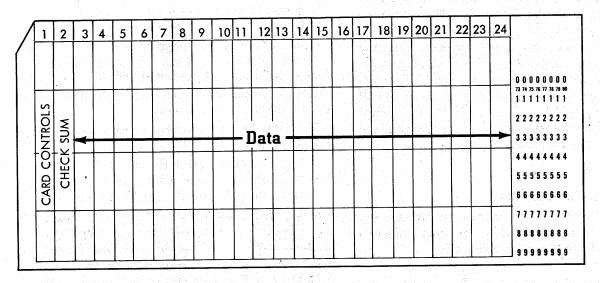


Figure 8. Use of the Processor by the System Editor



Word Bit Positions	Contents	Word Bit Positions	Contents
	lust be blank.	12-17	Count of words in the card, excluding words 1 and 2.
2 If	Nust be blank.  punched, the System Editor does not compute a check sum from card data for	21-35	Absolute loading address — the actual address of the core storage word where the first instruction of this card is stored.
	omparison with the card's prepunched heck sum.	\$ 35	Check sum — the logical sum (Add and Carry Logical Word) of all words in this
	lust be punched. lust be punched.	3-24	card except word 2.  Absolute binary instructions.

Figure 9. Absolute Column Binary Format Acceptable to the Editor Monitor

are stored on a utility file, called the edit file, for sorting during Phase 1. The Editor Monitor chooses an available unit for the edit file. Relocatable subroutine decks destined for the Subroutine Library (IBLIB) must remain in relocatable format. Consequently, they are not passed to the Loader, but are placed in the edit file and later put into the proper format by the Editor program.

The characteristics of the edit file are given in the section "Processor Files," which is included in the discussion of the Processor (IBJOB).

The capability of the System Editor to use the Processor permits source language (fortran IV, cobol, and MAP) editing. This means that programs to be edited into the System Library may be in any source language that can be translated by the Processor. If translation and/or conversion is necessary, the Editor Monitor passes control to the Processor and ultimately receives an absolute machine language version of the symbolic coding for the particular phase on the edit file following the parameter cards. The edit file is rewound after all translation is complete. All the input to Phase 1 is on the edit file; Phase 1 treats the edit file as though it were on s.sin1. The same capability can be extended to any processor that the installation has edited onto the System Library.

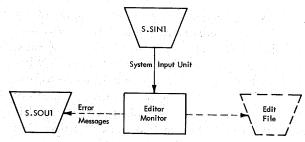
#### **Editor Program**

The Editor Program is called into core storage by the Editor Monitor when the editing is to begin. Phase 1 performs all necessary sorting of input data and analyzes the Editor parameter cards, placing them on an intermediate storage unit followed by the linked programs. Phase 2 uses this information to create the new version of the System Library on the designated output unit. Figure 10 shows the flow of control during an edit run.

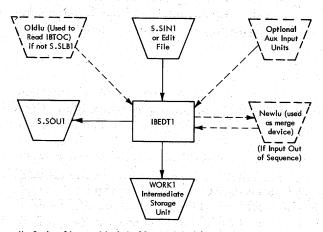
#### Phase 1

Phase 1 of the Editor program passes over the input units, blocking the card image (or IBLDR) input and sorting all the blocks of a phase in ascending order by the load addresses. The sorted blocks are concatenated, and the load address and the word count of each block are placed in the first word of the block. The input unit, which may be the system input unit, the edit file, or auxiliary units, is specified by the Editor Monitor and by certain Editor parameter cards. Phase 1 also performs a full check of the parameter cards and passes encoded information about them to Phase 2.

If necessary, Phase 1 sorts the input to agree with the sequence in which the altered phases appear in the Subroutine Library.



 Initialization by the Editor Monitor. Dotted lines indicate a device that may not always be used.



II. Storing of Input and Analysis of Parameter Cards by IBEDT1.

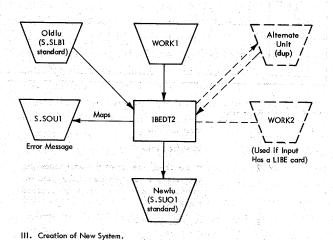


Figure 10. Flow of Control During an Edit Run

The Table of Contents is created or updated by Phase 1. For some edit runs, Phase 1 is able to pass a final Table of Contents to Phase 2. For other runs, Phase 2 may have to alter the Table of Contents. If the edit deck includes a modify parameter, a dup parameter, or a libe parameter, or if the edit run involves devices with different maximum block sizes, Phase 1 is unable to determine the exact number of blocks that will result from the operations.

If it is later determined during Phase 2 that these operations did alter the block count, and if the new library unit is a sequential device, a second pass will be made to insert the updated Table of Contents into the new library. (If the new library unit is a random access device, there is no need for a duplication pass because the Editor program, when writing the new System Library, reserves enough tracks to accommodate the phase containing the Index and the Table of Contents. This phase, IBTOC, is inserted into the proper place upon completion of the edit run.)

#### Phase 2

Phase 2 merges the update records on the intermediate storage unit with the old master System Library. Upon completion of Phase 2, an updated version of the System Library is ready for use.

The first function of Phase 2 is the selection of the Bootstrap program appropriate for the new library unit. Then the update records are merged.

As the blocks are written, a chain word is set up in the last word of each block. The load address and the word count of the first block of each phase are placed in the Table of Contents. The last word in the last block of a phase has an MZE prefix code and contains the entry point to the phase in its address portion.

Then, Phase 2 writes the 1EOF trailer label, rewinds the units, and returns control to the Supervisor via s.sret.

The new library may be on one or two units. The new library will be on two units if a 1EOR trailer label is encountered while copying the old library unit or if the edit deck included an INSERT EOR parameter card. Units are switched and writing continues until editing is completed and a 1EOF trailer label is written. All trailer labels are updated to the current date. The version and modification level is changed only if it so specified on the SIBEDT card. A discussion of the LABEL option on the SIBEDT card is presented later in the text.

In Figure 11, a sample sequential map is shown that depicts the physical order of all phases written onto the new library unit, with an indication of the Editor parameters, if any, that affected specific phases. Appearing beside each phase name is a string of entries that indicate the block number and word count of each record that makes up the phase. If the edit run includes

		SEQUE	040/4 NT I A L	4 IBSYS	EDIT PHASE	s				
IBOOT IBNUC IBSUP	0/ 1/ 5/	48 1000, 1000,		1000,	3/ 1	000, 548,	4/	748 13		
IBTOC	9/	291	6/	1000,		5401				
IBDMP1 IBIOC	10/	1000,	12/	939				1000	176	1000
MONITO	13/ 18/	7, 506,	14/	1000,	20/	568		1000,		1000,
IBJOB LDRPRE	21/	7, 1000,	22/	1000,	28/	1000,	29/	1000,	25/ 30/	434,
RELOAD	31/	103, 11	32/	1000,	33/	1000,	34/	1000,	35/	615,
CBC010	37/	1000,	38/	1000, 37,	39/	1000,	40/	1000,	41/	1000,
CBC012	45/	1000,	43/	918	177	100				
CBC014 CBC016	47/ 50/	7, 7,	48/ 51/	927,	49/ 52/	964 233				
BC020	53/ 58/	633, 25,	54/ 59/	1000, 52,	55/	1000, 475,	56/	551,	57/	107,
BC030	62/	1000,	63/	1000,	64/	1000,	70/	969,	71/	1000,
	72/	659.	73/	1000,	76/		of the	1000,	78/	519,
BC040	74/ 79/	40. 5					4. 9			
BC050	80/ 85/	265,	81/	1000,	82/	1000,	. 4.	1000,		1000,
BC060	87/ 92/	1000,	88/	1000,	89/	1000,	90/	1000,	91/	986.
BC066	93/	64,	94/	1000,	97/	1000,	98/	1000.	99/	1000,
CBC070	95/ 100/	1000,	101/	1000,	102/	1000,		1000,		1000,
CBC074	105/ 106/	651 301								
FTC010	107/	1000,	108/	1000,	109/	159,	110/	604,	111/	1000,
FTC020	117/	1000,	118/	1000,		1000,	120/	29,	121/	61,
FTC030	123/	1000,	124/	1000,		1000,	126/	1000,	127/	372,
FTC040	128/	3, 1000,	129/	3, 640,	130/	725,	134/	28,	135/	6,
FTC050	136/	1000,	137/	1000, 229,	143/	1000, 3,	139/	1000,	140/	1000,
4AP010	145/	1000,	146/	1000,	147/	1000,	148/	1000,	149/	776,
MAP020	150/ 152/	204, 456,	153/	11		1000	1574	1000	158/	1000.
MAP030	154/ 159/	1000,	155/	1000, 519,	161/	1000,	162/	1000, 233	128/	1000
1AP035	163/ 165/	7, 7,	164/	538 192					ing set Tuga seta	
MAP050	167/ 172/	1000,	168/	1000, 421,	169/ 174/	1000,	170/	1000,	171/	1000,
MA P060	175/	1000,	176/	1000,	177/	1000	178/	1000,	179/	1000,
MAPO70	180/ 184/	427, 782,	181/	1000, 342,	182/ 186/	106,	1837	10		
MAP072 Map074	187/	819,	188/	7 189,	191/	4				
MAP080 MAP082	192/	383, 7,	193/	10 1000,	196/	119.	197/	1000,	198/	1000,
5-4-1 (S.A.)	199/	1000,	200/	1000,	201/	483	205/	992	1,2,30	
MAPO86 LDRO	202/	7, 697,	203/	456,	204/	1000.	2057	992		
LDRI LDR1	209/	7, 7,	210/		213/	1000,	214/	1000,	215/	426,
IBLIB	216/	1000,	218/							
SRNAD IBREL	219/	492,	220/	547.	221/	393,	222/	449, 505,	223/ 228/	505, 505,
	224/	393, 505,	225/	449,	226/	449, 505,	232/	505,	233/	561,
	234/	463,	235/		236/	407,	237/	505, 435,	238/ 243/	505, 449,
	244/	449,	245/	449,	246/ 251/	449,	247/	449, 561,	248/ 253/	421, 505,
	249/ 254/	407, 561,	255/	505,	256/	561, 491,	257/	477, 505,	258/ 263/	379, 505,
	259/ 264/	407,	260 A	491,	261/	463.	267/	435.	268/	407,
	269/ 274/	491, 421,	270/	407	271/	505, 491,	272/	393, 449,	273/ 278/	421, 449,
	279/ 284/	561, 449,	280	561,	281/	449,	282/	449,	283/	449,
	2897	435,	290	407,	291/	463	292/		293/	
LDR2	294/ 296/	477 <b>,</b> 7,	295 297	/ 1000,	298/	1000,	299/	1000,	300/	626,
LDRL LDR3	301/ 304/	7, 7,	302		303/	1000,	307/	491	1.50	
LDR4	308/	7, 761,	309 311	234	312/	1000,	313/	910		te a
IBDMP2 IBDMP3	314/	1000,	315	1000.	316/	1000,	317/			
IBOLVA IBDLVB	318/ 321/	7,	319	1000,	320/ 323/	13 1000,		1000,	325/	1000,
IBDLVC	326/	1000,	327	/ 1000, / 1000,	328/	1000,	329/	1000,	334/	610,
DEFA IBDEF	335/ 337/	821,	336			1000,		1000,	341/	
IEDBG	342/	3		100	3377	20001				
SRNDB 1BDBG	345/	1000, 520,	344 346	/ 561,	347/	561,	348/	561,	349/	561,
IBOPPE	350/ 352/	561, 1000,	351 353	/ 519	354/	181,	355/		356/	7,
IBTRAN	357/ 362/	1000,	358	/ 1000, / 1000,	359/	1000	360/		361/	428,
IBEDT1	363/	1000,	364	/ 36,	365/	1000,	3667	1000,	367/	1000,
IBEDT2		1000,	370	/ 1000,	371/	1000,	372	758		
SORT EDPHAZ	373/ 375/	1000,	374	/ 313	377/	1000,	378/	1000,	379/	1000,
DMPYEX	380/	1000.	381	/ 1000,	382/	859,	383/	861		
PH1FIX	385/	1000,	386							
PH2FIX PH3FIX	387/ 389/	522,	388 390	/ 769 / 1000,	391/	154,	392	173		
LOAD PH1VAR	393/ 394/	572	395	/ 1000,	396/	370		115	1	
PH2VAR PH3VAR	397	742,		/ 765	401/	234,	402	/ 187		
IBUTL	403	/ 1000,	404	/ 128		-271	.321			. 41 %
IBUF IBUL	405, 407,	/ 915	406	/ 81	9					
IBUD IBUR	408	601		1.36.3	医胸部			4		
*DOL	410	/ 1000,	411	/ 831				ağat v		
IBUU	412		414	/ 1000,	415/	1000,	416	/ 663,	417	/ 1000,
IBUU IBUC UPDATE	413.	/ 1000.								
IBUC		/ 1000,		/ 1000,	420/	195,	421	/ 19		

Figure 11. Sequential Map of the System Library

a LIBE parameter, a sequential map of the relocatable Subroutine Library is also written.

In Figure 12, a sample logical map, made from the Index and the Table of Contents, depicts the grouping of the phases of each system in the System Library.

The headings for these maps contain page numbers, the data stored in S.SDAT, and the contents of the first REMARK card in the edit deck. The BCD trailer label is printed at the end of the sequential map.

A complete copy of the System Library may be obtained by performing an edit run, using only the SIBEDT card and the SENDEDIT card. A map of the Subroutine Library may be obtained during this run by adding the LIBE and LIBEND parameter cards.

## **External Storage Unit Requirements**

In addition to the usual complement of system units (such as the system library unit, system input unit, and system output unit), the System Editor requires a minimum of two utility units, selected for the Editor phases by the Editor Monitor. The first is used as an intermediate storage unit (work1); the second is the new system library unit (newlu). The minimum configuration is limited to edit runs that involve only absolute column binary cards or the special oct patch cards described later in the text. Another utility unit, work2, must be available for an edit run that alters the IBREL section of the Subroutine Library (IBLIB). If an edit run causes the creation of an edit file, an additional unit is required. This unit must be a device that is not on channel S.

#### **Editor Control Cards**

Editor control cards, as distinguished from Editor parameter cards, have the standard System Monitor control card format with a \$ in column 1 and the control card name in columns 2-8. They are interpreted by the Editor Monitor and govern the use that the System Monitor makes of its Editor program and any Processor components.

## \$IBEDT Card

The format of the SIBEDT card is:

1	16		
\$IBEDT	op	otions	

This is the basic Editor control card, appearing as the first card of every edit deck. Options that regulate the edit run are specified in the variable field, which starts in column 16. If the number of options specified is too great for all of them to be contained on the SIBEDT card, a SETC card may follow the SIBEDT card. In this case, a comma must follow the last option on the SIBEDT card.

SYSNAM	7040/44 IBSYS E LOGNAM	DIT RECNO
BXLOAD	BXLOAD	422
DEBUG1	#BDLVA IBDLVB IBDLVC DEFA IBDEF IBDBG	318 321 330 335 337 345
DEBUG2	IBDPPE IBTRAN	352 357
DEBUG	IEDBG SRNDB IBDBG	342 343 345
IBCBC	CBC010 CBC012 CBC014 CBC016 CBC020 CBC030 CBC040 CBC050 CBC060 CBC060 CBC070 CBC074 MAP010	37 45 47 50 53 62 74 80 87 93 95 106
IBDMP	IBDMP1 IBDMP2 IBDMP3	10 310 314
IBEDT	IBEDT1 IBEDT2	363 369
IBFTC	FTC010 FTC020 FTC030 FTC040 FTC050 MAP010	107 117 123 131 136 145
IBLOR	LDRO LDRI LDRI IBLIB SRNAD IBREL LDR2 LDRL LDR3 LDR4	206 209 211 216 217 219 296 301 304 308
IBMAP	MAP020 MAP035 MAP035 MAP040 MAP050 MAP060 MAP070 MAP072 MAP074 MAP080 MAP082 MAP086	152 154 163 165 167 175 184 187 189 192 194
IBSRT	SORT EDPHAZ DMPYEX PHIFIX PH2FIX PH3FIX LOAD PHIVAR PH2VAR PH3VAR	373 375 384 385 387 389 393 394 397 399
IBSYS	IBOOT IBNUC IBSUP IBTOC IBIOC MONITO IBJOB	0 1 5 9 11 13 21
IBUTL	IBUTL IBUF IBUL IBUD IBUR IBUU IBUU	403 405 407 408 409 410 412
LDRPRE	LDRPRE	26
RELOAD UPDATE	RELOAD UPDATE	.31 413

Figure 12. Logical Map of the System Library

The options that can be specified in the variable field are:

$$\begin{bmatrix}
\frac{S.SLB1}{S.SUxx[} = Iyy] \\
Iyy[R] \\
NONE
\end{bmatrix}
\begin{bmatrix}
(S.SLB2) \\
(S.SUxx[ = Iyy]) \\
(Iyy[R])
\end{bmatrix}$$

This option is used to specify the unit or units that contain the old (master) System Library. Both a primary unit and a secondary unit may be specified. The primary unit specification appears first; the secondary unit specification follows immediately and is enclosed in parentheses.

S.SLB1 indicates that the primary system library unit is to

be used.

S.SUxx indicates that utility unit xx is to be used. If the =Iyy specification is included, the unit is to be assigned the intersystem reservation code yy.

Iyy indicates that the unit already reserved by intersystem code yy is to be used. If R is added, the intersystem code is released and the reservation is canceled upon completion of the edit run.

NONE indicates that no units are needed for the old System Library. The Editor is to create a new System Library, using only the contents of the edit deck for input. The IBOOT phase is inserted automatically. IBTOC is created in core storage but must be inserted into the new library through the use of an INSERT parameter card (described below).

S.SLB2 indicates that the secondary system library unit is to be used. This option may be specified only when the primary

unit specification is S.SLB1.

If the primary unit for the old System Library is not specified (that is, if column 16 contains a blank or a comma), the primary

system library unit (S.SLB1) is used.

If the old System Library is contained on more than one unit, the last record on the first unit is a 1EOR trailer label. When the Editor encounters this record, it switches to the specified secondary unit. In the event no secondary unit is specified, the Editor switches to S.SLB2 if S.SLB1 was used as the primary unit. Otherwise, the edit run is terminated.

$$\left[ \begin{array}{l} \left\{ \begin{array}{l} S.SUxx[=Iyy] \\ Iyy[R] \\ Iyy_1 R=Iyy_2 \end{array} \right\} \left[ \begin{array}{l} \left\{ \begin{array}{l} \left( S.SUxx[=Iyy] \right) \\ \left( Iyy[R] \right) \\ \left( Iyy_1 R=Iyy_2 \right) \end{array} \right\} \end{array} \right] \right]$$

This option is used to specify the unit or units that will contain the new System Library. Both a primary unit and a secondary unit may be specified. The primary unit specification appears first; the secondary unit specification follows immediately and is enclosed in parentheses. The secondary unit must be the same type of device as the primary unit.

S.SUxx indicates that utility unit xx is to be used. If the lyy specification is included, the unit is to be assigned the

intersystem reservation code yy.

Iyy indicates that the unit already reserved by intersystem code yy is to be used. If R is added, the intersystem code is released and the reservation is canceled upon completion of the edit run.

Iyy<sub>1</sub> R=Iyy<sub>2</sub> indicates that the unit reserved by intersystem code yy<sub>1</sub> is to be used. Upon completion of the edit run, this code is to be released and the unit is to be reserved by the intersystem code yy<sub>2</sub>.

If the primary unit for the new System Library is not specified, S.SU01 is used. If S.SU01 is not available, an available unit is chosen. If no unit is available, the edit run is terminated.

If a secondary unit for the new System Library is not specified, but a secondary unit is required, the edit run is terminated.

#### [,LABEL (nn,mm,q,p)]

This option is specified if checking of the version and modification numbers of the old library unit(s) or label checking of the new library unit(s) is desired to preclude improper use of these units.

The four subfields of the LABEL option have the following interpretation:

nn = the version number of the old System Library.

mm = the modification number of the old System Library.

q = the number of old library units (1 or 2).

p =the number of new library units (1 or 2).

The use of these subfields is explained in the following paragraphs.

The version and modification numbers apply to the entire Operating System and should appear in the first word on each master library unit. This is the word that the System Monitor, at an initial start, moves from location 100s to location S.SLVL in the Nucleus. If these subfields are used, the Editor Monitor checks the first word on the master library unit(s) against the specified nn and mm values to ensure that the proper System Library is being updated. An unequal comparison results in a code 31501 error message, and an operator-action pause permits acceptance or termination of the edit run. If the nn and mm values are verified or accepted, the modification number (mm) is incremented by one, and the new System Library is given this updated value.

The q and p subfields specify the number of library units that require checking before the edit run begins. If q is omitted, one old library unit is assumed. If p is omitted, one new library unit is assumed; at the installations that use a labeled system, header labels on all intermediate units and one new library unit are checked by IOLS, unless p specifies a number greater than one. If label checking detects an invalid label, an error message is typed and an operation-action pause permits acceptance or rejection of the unit. If the installation does not use labels (indicated by the system assembly parameter LABELS SET 0 or 1), no header label verification is made for output units.

The following combinations of the LABEL option subfields are valid:

OPTION	MEANING
LABEL(,,,1)	Check the label on one new library unit.
LABEL(,,,2)	Check the labels on two new library units.
LABEL(nn, mm)	Check the version and modifi-
LABEL(nn, mm, 1)	cation numbers on one old
LABEL(nn, mm, , 1)	library unit against the values
LABEL(nn, mm, 1, 1)	nn and mm, and check the label on one new library unit.
LABEL(nn, mm, , 2)	Same as the above, except that
LABEL (nn, mm, 1, 2)	the labels on the two new library units are to be checked.
LABEL(nn, mm, 2)	Check the version and modifi-
LABEL (nn, mm, 2, 1)	cation numbers on two old
	library units against the values nn and mm, and check the label on one new library unit.
LABEL(nn, mm, 2, 2)	Same as the above, except that the labels on the two new library units are to be checked.

To reset the version number, the preceding version number nn must be specified and mm must be omitted.

The following examples show several LABEL options and the version and modification level of the new System Library resulting from each option:

LABEL(9,) Version 10, Level 0

LABEL(9, 1, 1) Version 10, Level 0

LABEL(9, 0, 1, 1) Version 9, Level 1

# SOURCE NOSOURCE

This subfield indicates to the Editor Monitor the format of the edit run input. SOURCE indicates that some decks in the edit deck are in the source language of FORTRAN, GOBOL, or MAP, whereas NOSOURCE indicates that some decks are in the MAP binary output format and must be processed by IBLDR. The SOURCE option logically includes the NOSOURCE option. Thus, the former should appear on the \$IBEDT card, if the edit deck includes source language decks as well as binary decks in MAP output format. The absence of either SOURCE or NOSOURCE indicates that the edit deck (including any LIBE data) consists only of cards in a format that may be processed directly by the Editor program, i.e., absolute column binary cards, OCT instruction cards, or relocatable cards for a subroutine library. The list of options can be extended by assembly modification of the Editor Monitor and the Editor program.

[, MXBLK(nnnn)]

This subfield specifies the maximum block size for the new system library unit. It may be any decimal number that is less than, or equal to, the assembly parameter IER01. If the decimal number is greater than the parameter IER01, the block size of IER01 is used.

#### [, NOMAP]

This subfield specifies that the two maps, including the Subroutine Library section list, will not be listed during the edit run. However, all parameter cards and error messages are listed.

[, EDTFIL(nn)]

This subfield indicates that a previously generated edit file (with utility number nn) is to be accepted by the Editor, instead of creating another edit file. The number nn must not refer to a utility unit on channel S. The use of this option is meaningful only when a prior edit attempt was terminated (after the message 11500 BEGIN EDIT was typed) because of a machine or program malfunction.

## [,CORE(nnnnn)]

This subfield indicates to the Editor Monitor that the value of S.SEND will have changed as a result of this edit run. The use of this subfield does not itself effect the change, but it enables the Editor to relocate the IBTOC record of the new system, using the new value of S.SEND. The new value (nnnnn) is expressed as a decimal number between 16383 and 32767.

This subfield should be used only in a total relocatable edit when it is essential that the IBTOC record be relocated; that is, when either:

- 1. The value of S.SEND will be decreased.
- 2. The system has been expanded such that IBSUP on the new system will overlay a part or all of the IBTOC record, if the location of the IBTOC record is determined by the Editor using the current value of S.SEND.

#### [, MIN]

This subfield indicates that the physical unit on which the edit file is located is to be used for the intermediate storage unit WORK2. This reduces by one the number of units required for a NOSOURCE edit run that both alters the Subroutine Library and causes the creation of an edit file (e.g., a total system edit). The MIN option may not be specified when SQURCE is specified on the \$IBEDT card.

Note that when the MIN option is specified, the edit file is destroyed during the edit and, therefore, cannot be saved for subsequent runs that specify EDTFIL.

Examples of typical \*IBEDT cards follow:

This card causes the Editor Monitor to check the decrement and the address of the first word on the single master library unit (s.sl.bl) for the values 4 and

19 (that is, version 4 and modification 19). The modification number 20 will appear on the new System Library. If the installation uses a labeled system, the Input/Output Label System checks the retention dates on the new library units as well as on the editor work tape.

This card causes the old System Library on s.su03 to be edited onto s.su04. No check of the version and modification numbers is made, and the new System Library will have the same modification number as the master library. The source option causes the edit file to be stacked with the SIBEDT card, the appropriate parameter control cards, and an absolute machine language version of the source language program.

#### **\$ENDEDIT Card**

The format of the SENDEDIT card is:

This is the last card in the edit deck. It indicates to the Editor Monitor, during the stacking of the edit file, that the end of the deck has been reached.

During Phase 1, the SENDEDIT card indicates the end of the input deck to the Editor program. If the SOURCE OF NOSOURCE option was taken, this card indicates the end of the edit file.

Any text may appear in columns 16-72.

Upon completion of an edit run, the message edit complete is typed on line. If there is a nonterminal error during editing, the message error encountered appears after the edit complete message. If there is a terminal error during editing, the message edit terminal error was due to an unposition the system input unit immediately after the sendedit card. If the terminal error was due to an unrecognizable card on the system input unit, the Editor saves the unrecognizable card record and directs the Supervisor to inspect it.

An example of a typical sendedit card follows:

This card causes the contents of the master library to be copied onto the new library through the 1EOF record.

Any Editor parameter cards on the system input unit after the \*ENDEDIT card (and not preceded by a new \*IBEDT card) are treated as erroneous control cards.

## Editor Parameter Cards

Editor parameter cards direct the Editor program in the creation of an updated System Library. Note that these cards do not have a \$ in column 1. The card name begins in column 8, and the variable field begins in column 16. In some cases, a variable appears in columns 1-6. Serial numbers may be punched in columns 73-80.

These BCD cards are interpreted by the Editor program. The Editor Monitor simply stacks them on the edit file if the SOURCE OF NOSOURCE options are taken.

#### **CALLS Card**

The format of the CALLS card is:

svsnam	CALLS	phase1, phase2,	phasen [serial]
1	8	16	73

This card defines the normal sequence taken during execution of the phases of a subsystem. The Editor program requires this information to construct the Index and the Table of Contents.

The CALLS parameter need only be used for the edit run during which a subsystem program is initially introduced into the System Library. The Index and Table of Contents entries are then available for subsequent edit runs involving that subsystem. However, this parameter may be used again, if it is desired to change a phase name, add a new phase, or alter the flow of a subsystem.

The removal of a phase from the System Library does not require a new CALLS card. The name of the removed phase is deleted from the Table of Contents by using a REMOVE card (described later in this text), and the flow is assumed to pass from the phase prior to the removed phase to that following it.

The variable field of the CALLS card can be extended by the ETC card (described later in this section).

The content of columns 1-6 is:

sysnan

This is the name by which a particular subsystem is identified. It will be stored in the first word of the Index entry for this subsystem.

The content of columns 16-72 is:

phase<sub>1</sub>, phase<sub>2</sub>, . . . , phase<sub>n</sub>

This is a list of the BCD phase names of the subsystem identified by sysnam in the order in which they are called by the System Loader. The phase name is stored in the first word of the Table of Contents entry for the phase.

This list does not necessarily describe the actual sequence of the phases on the library unit. Accordingly, it is not necessary to have duplicate copies in the System Library of a phase common to several subsystems. If the name of a phase that is already part of the System Library appears on a CALLS card, the Editor program creates, within that portion of the Table of Contents used by the new subsystem, an entry defining the location of the common phase.

The CALLS card and all ETC cards used to extend the list of phase names must be grouped together in the edit deck ahead of all other Editor parameter cards, except the REMARK cards, if any.

An example of a typical CALLS card follows:

A card of this form results in the creation of one Index entry and four Table of Contents entries for the subsystem identified by INAB. This example shows how any phase in the System Library, xxzo20 in this case, may be designated as part of the logical flow of any subsystem component merely by placing it in the phase name string of the subsystem. This assumes, of course, that the phases can be executed sequentially, and that the common phase already exists or is being inserted into the System Library.

## **ETC Card**

The format of the ETC card is:

This card may be used to extend the list of phase names in the variable field of the CALLS card, if this list overflows the parameter card.

#### **INSERT Card**

The format of the INSERT card is:

8 16 73

INSERT 
$$\begin{cases} lognam \\ EOR \end{cases} \left[ , \left\{ S.SUxx \\ Iyy[R] \right\} \right]$$
 [serial]

The INSERT card causes the specified phase, previously defined in a CALLS card, to be placed on the new library unit at its current position. If EOR is specified, it causes a standard 1EOR trailer label to be written immediately on the new library unit. The previous positioning of the library units is accomplished by other Editor parameter cards. Insertions may be made from either the system input unit, where it is now positioned, or any utility unit designated. If the program being inserted was assembled by the 7040/7044 Basic Assembly Program, the last card of the absolute column binary deck must be a transfer card indicating the entry point to the program.

The content of the variable field is:

{ lognam } { EOR }

The lognam is the BCD phase name (which must begin with a character other than zero) that the Editor program prefixes to the first block of the phase being inserted. This phase name is used by the System Loader to verify the loading of the phase and by the Editor program for altering the Table of Contents and for later editing of the System Library. The Editor program ignores any attempt to insert a phase with the same name as one that is already in the System Library and that is not re-

moved during the edit run, prior to the insertion of the new phase. The identical binary deck may be inserted at various points in the System Library by using different phase names on the INSERT card.

The EOR option is available to cause reel switching to the next new library unit while writing on the new library unit. A 1EOR trailer label is placed on the current new library unit, preceded and followed by a file mark, at the point at which it is currently positioned.

S.SUxx is the alternate utility unit containing the program to be inserted. Iyy specifies that the device to be used was previously assigned intersystem reservation code yy. If R is appended to the Iyy code, the reservation will be removed after the device is used by the Editor program.

This specification directs the Editor program. A subsystem must be directed by its own control cards, e.g., \$IEDIT.

An example of a typical INSERT card follows:

This card can be used to insert the phase FTC010 from the system input unit into the System Library at the current position of the new library unit.

## **REPLACE Card**

The format of the REPLACE card is:

This card is used to replace a phase in the System Library.

The REPLACE card causes the master System Library to be copied onto the new library unit(s) up to the phase (logical record) identified by lognam. The specified phase is skipped over on the master library, and a new one is written in its place on the new library from the deck following the REPLACE card. The new phase has the same name as the deleted phase, and the Table of Contents is updated to reflect the new load address and word count of the replacing phase's first block.

The content of the variable field is:

lognam

This is the BCD phase name. See the definition of lognam given in the description of the INSERT card.

$$\left[, \left\{ \begin{array}{l} S.SUxx \\ Iyy[R] \end{array} \right\} \right]$$

S.SUxx is the alternate utility unit containing the deck of alteration cards or card images if it is not on the system input unit. Iyy specifies that the device to be used was previously assigned intersystem reservation code yy. If R is appended to the Iyy code, the reservation will be removed after the device is used by the Editor program.

This specification directs the Editor program. A subsystem must be directed by its own control cards, e.g., \$IEDIT.

An example of a typical REPLACE card follows:

This card causes the blocks in the phase FTC010 to be replaced in the System Library by the data following this REPLACE card on S.SIN1.

#### **MODIFY Card**

The format of the MODIFY card is:

The Modify card provides the normal means of patching a program that already resides within the System Library in absolute form. Corrections within or beyond the original program length may be made with this parameter.

The old System Library is copied onto the new library unit(s) up to the phase lognam. The word(s) to be changed within the program is overlaid by the patch word(s) while the program phase specified by lognam is being copied onto the new library unit. In cases where the modification is not within the length of the original program, new blocks may have to be created or the present ones expanded (the section "System Library Format" gives the description of a block).

The modification cards, containing the load address and the patch, may be in the format of the special ocr patch card described later in the text, or they may be in any other card format acceptable to the Editor Monitor.

The content of the variable field is:

This is the BCD phase name. (A definition of lognam is given in the description of the INSERT card.)

$$\left\lceil, \left\{ \begin{array}{l} S.SUxx \\ Iyy[R] \end{array} \right\} \right\rceil$$

This option is used to specify the alternate unit holding the modification cards if they do not follow the MODIFY card on the system input unit. When the modification cards are on the system input unit, the Editor program can determine the end of the correction cards by the appearance of the next Editor parameter card. If an alternate unit is specified by this option, the final modification card should be a transfer card. If the transfer address is zero, the existing entry point is not modified. This specification directs the Editor program. A subsystem must be directed by its own control cards, e.g., \$IEDIT.

S.SUxx specifies that utility unit xx is to be used as the alternate unit. Iyy specifies that the device to be used was previously assigned intersystem reservation code yy. If R is appended to the Iyy code, the reservation will be removed after the device is used by the Editor program.

FTC010,S.SU10

#### MODIFY

This card causes the patches on the modification cards on utility unit 10 to be included in the phase FTC010 as it is being written on the new library unit.

#### **REMOVE Card**

The format of the REMOVE card is:

1	8	16			73
	ta est e	<b>F</b> (1	) ¬		
[sysnam]	REMOV	$\mathbf{E} \begin{bmatrix} \begin{cases} \log \\ \mathbf{E} \end{cases} \end{bmatrix}$	nam ( )	in the	[serial]

This card is used to remove a phase from the System Library. If a subsystem name appears in the location field (card columns 1-6), the entire subsystem will be deleted.

Upon recognizing the REMOVE card, the Editor program copies the old System Library onto the new library unit up to the designated phase or up to the 1EOR trailer label. The phase is spaced over on the old library unit, its name is removed from the Table of Contents, and editing resumes with the next Editor parameter card.

When a phase has been removed from the Table of Contents by means of a lognam or a sysnam reference (as explained below) that phase may not be used in another Editor parameter card until the phase is reinserted.

The content of columns 1-6 is:

[sysnam]

This is the name by which the subsystem to be removed is identified. It appears in the first word of the Index entry for this subsystem. In this use of the REMOVE parameter, the variable field (card columns 16-72) is left blank. Phases common to other subsystems and the subsystem being removed are not removed from the System Library. Only their appearance in the Table of Contents string of entries for the removed subsystem

The content of columns 16-72 is:

lognam ( EOR

The lognam is the BCD phase name that identifies the phase to be removed. It appears as the first word of the Table of Contents entry for the phase.

If the EOR option is specified, the contents of the System Library up to, but not including, the 1EOR trailer label are copied into the new library. This is a method of condensing the System Library onto one system unit. While unit switching occurs on the master library unit, the new library unit remains unchanged.

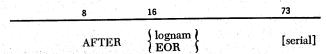
An example of a typical REMOVE card follows:

**IBCBC** REMOVE

This card would be used to remove the COBOL Compiler from the Operating System.

## **AFTER Card**

The format of the AFTER card is:



This card causes the copying from the old library to the new library of all phases up to and including either the phases specified or the 1EOR trailer label. In either case, unit switching of both the old and the new library units is automatic when the end-of-reel record is read.

The content of the variable field is:

∫ lognam ( EOR

The lognam is the BCD phase name of the last phase to be copied. If EOR is specified, the master System Library will be copied through the 1EOR trailer label, with automatic unit switching.

An example of a typical AFTER card follows:

The use of this card causes the master System Library to be copied from its current position through phase FTC010.

#### **DUP Card**

The format of the DUP card is:

9 (2) (n. 1961) (n. 1864) (n. 1964) (n. 1964)	16	73
DUP	unit1, unit2, n, inlabel, cdate, oulabel, rdays	[serial]

This parameter permits a specified number of phases to be copied from one symbolic unit to another. Phases may be duplicated from a master library unit onto a utility unit, from a utility unit onto a new library unit, or from one utility unit onto another utility unit.

The use of the DUP card with the REWIND card (described in the next section) simplifies the reordering of various phases or complete systems within the System Library. This parameter is also useful for placing specially edited phases, such as special data blocks, library subroutines, test cases, etc., on the new library unit, provided they are in the System Library format.

If the DUP card is used to rearrange the system, do not copy the Table of Contents (IBTOC), since it reflects the order of the old system.

The content of the variable field is:

unitl

This is the unit from which the phase will be duplicated. The unit1 specification may be either S.Sxxx or Iyy[R]. S.Sxxx specifies the symbolic unit to be used. Iyy specifies that the unit previously assigned intersystem reservation code yy is to be used. If R is appended to the Iyy code, the reservation will be removed after the unit is used by the Editor program.

unit2

This is the unit onto which the phases are copied. An attempt to copy records onto a master library unit or onto a system library unit, system input unit, system output unit, or system punch unit is ignored. The unit2 specification may be either of those described above for unit1.

n

The number of phases to be duplicated is specified by n. One phase is duplicated if n is blank.

inlabel

This specifies up to ten characters, which will be left-justified with trailing zeros. These characters are used to verify the input label.

cdate

The five-digit date in the form yyddd on which the input label was created

oulabel

Up to ten characters that will be left-justified with trailing zeros and used to create the output label

rdays

A four-digit number that specifies the number of days the file is to be retained

All arguments appearing in the variable field must be separated by commas and must be in the order presented.

The fields inlabel, cdate, oulabel, and rdays are optional and are used to effect label verification or creation for DUP input or output units. It is the user's responsibility to include these fields on DUP cards at the point where label verification or creation is required. The Editor assumes all units to be properly positioned in front of the data to be processed next. The REWIND parameter should be used, if required, before label verification or creation.

Examples of typical DUP cards follow:

B 16

DUP S.SU03,S.SU04,3,DUPLICIN, 63031,DUPLICIN,10

This card causes three phases to be copied from s.su03 onto s.su04. It will also verify the header label on s.su03, created on January 31, 1963 with file identification DUPLICIN, and will create on s.su04 a label that has the same file identification and that is to be retained at least ten days.

8	16	
DUP	S.SLB1,S.SU04,1,,,DUF	OUTB,365

This card causes one phase to be copied from s.slb1 onto s.su04. It also causes a header label with file identification DUPOUTB and a retention period of 365 days to be written on s.su04. s.slb1 has no label.

8		16				
DUI	Р	S.SU04	ı,s.su	11,,LB	LNAM	,63032

This card causes one phase to be copied from s.su04 onto s.su11. It also causes the verification of the header label on s.su04. The label was created on February 1, 1963 and has the file identification LBLNAM. No label will be created on s.su11. s.su11 may not be positioned at the beginning of the file.

Format Requirements: Phases copied onto the new library unit must meet the following System Library format requirements:

- 1. Each block of the phase, except the last block, must end with a word whose sign is plus.
- 2. The last block of the phase must end with a word whose sign is minus.
- 3. The first word of the initial block must contain a BCD phase name that is also listed on a CALLS card.
- 4. There must exist a three-word header block specifying the load address and word count of the initial block of the phase to be duplicated, with the following format:

WORD1	BCI	1,phznam
	PZE	la,,wdct
	BĊI	1.*****

The following definitions apply to the symbols used above:

phznam

Phase name to be entered into the Table of Contents. It is left-justified, with trailing blanks, if any.

la

Load address of the first block.

wdet

Number of words in the first block.

This three-word block is used by the Editor program to update the Table of Contents entry for the specific phase. All phases duplicated by the Editor from the old library unit are prefixed by this header; it is not copied into the new System Library.

If the record being duplicated on the new library unit does not have a three-word header block, the Editor accepts the logical record, but the load address in Word 2 of the Table of Contents entry for the phase is set to zero. The System Loader may be used to position the library to this block, but it cannot be used to load the phase. This requirement for a header block does not apply when duplicating a phase from the old library unit indicated on the SIBEDT card, as the Table of Contents already contains the proper phase name, load address, and word count for such a phase.

#### REMARK Card

The format of the REMARK card is:

The REMARK card is written in full on the system output unit. Columns 16-36 of the first REMARK card in the edit deck are inserted into the title line of the System Library maps. Subsequent REMARK cards are typed and are followed by a machine pause for action by the operator during Phase 1. An asterisk in card column 7 will effect typing and pause during Phase 2 for DUP reel handling.

The variable field contains the message for the machine operator and/or programmer.

An example of a typical REMARK card follows.

If this is the first REMARK card in the deck, it will appear in the title line of the System Library maps on the system output unit.

#### **REWIND Card**

The format of the REWIND card is:

This parameter causes the specified symbolic unit to be rewound. Any attempt to rewind the new library unit or the System Editor intermediate unit (work1) is disregarded, and the next control card is read.

The content of the variable field is:

This is the symbolic unit to be rewound. Iyy specifies that the device to be used was previously assigned intersystem reservation code yy. If R is appended to the Iyy code, the reservation will be removed after the device is used by the Editor program.

An example of a typical REWIND card follows:

This card will cause utility unit 5 to be rewound.

#### **LIBE Card**

The format of the LIBE card is:

The LIBE parameter signals that relocatable subroutines are to be added, deleted, or replaced in the System Library. Each subroutine deck must be preceded by an INSERT, REPLACE, REMOVE, or AFTER parameter card to specify the type of editing to be performed. An alternate input unit may not be specified on any of these cards.

The relocatable subroutine deck that is used as direct input to the Editor program for editing into the Subroutine Library (IBLIB) contains a SIBLDR card, a SCDICT card, a binary control dictionary, a STEXT card, binary text, and a SDKEND card, in that order (see Figure 17, which appears later in the text). This is the relocatable binary deck normally produced by the Macro Assembly Program. The Editor program analyzes the control dictionaries of all the subroutines and generates the Control Section Name List, the Dependency Name List, and the subroutine deck blocks needed by the Loader (IBLDR). STEXT and binary text may be omitted, in order to include only the control dictionary information with the Subroutine Library.

The content of the variable field is:

#### [libenm]

The libenm is the phase name of the subroutine library to be edited. It specifies the phase up to which the master System Library is copied. If libenm is omitted, IBLIB is assumed. If libenm is IEDBG, a separate subroutine library will be edited, for use by the Debugging Processor.

#### [.format]

This option defines the relocation scheme; that is, it indicates the format of the relocatable card decks. For example, relocatable binary output from the Macro Assembly Program would be designated by IBMAP. If this option is omitted, Macro Assembly Program output format is assumed.

An example of a typical LIBE card follows:

This card causes the System Library to be copied up to the Subroutine Library. It must be followed by an insert, replace, remove, after, or libend card. A map relocatable output deck or a Processor deck, including a source language deck, may follow next. In the latter case, the Editor Monitor calls the Processor to produce the relocatable deck.

#### LIBEND Card

The format of the LIBEND card is:

The LIBEND card must be the last card on a Subroutine Library alteration deck. It signals the Editor program that the end of the alteration cards for the Subroutine Library specified on a prior LIBE card has been reached.

An example of a typical LIBEND card follows:

## LIBEND

This card would be used to indicate the end of the last relocatable deck. If the LIBE card is followed by a LIBEND card, the Subroutine Library (IBLIB) is reprocessed. That is, a new Control Section Name List and a new Control Section Dependency List are generated, and a map of the Subroutine Library is printed.

# Using the System Editor

The System Editor can accept any of the following types of input:

- 1. MAP, FORTRAN IV, or COBOL source language coding
- 2. Source language coding for any subsystem that an installation has added to the System Library whose output is in an acceptable format
  - 3. Absolute 22-word column binary card images
- 4. Macro Assembly Program binary output card images
  - 5. Records in System Library format
  - 6. oct card images

In addition, the System Editor can be expanded by an installation to include other formats for which a processor exists on the System Library.

The first five types of input were described previously in the discussion of the Editor Monitor. A description of the oct instruction or patch card follows.

#### **OCT Card**

The format of the oct card is as follows:

The our card is convenient for changing existing programs in absolute format without reassembly.

The content of columns 1-6 is:

loada

This is the octal location at which the first data word will be overlaid or inserted.

Column 7 contains an asterisk if the oct card and the name of the phase it affects are to be listed by Phase 1 of the Editor program.

The content of columns 16-72 is:

patch<sub>1</sub>, patch<sub>2</sub>, . . . , patch<sub>n</sub>

These indicate the patches for n consecutive data words, starting with the data word at location loada. Each patch is separated from the others by commas and may consist of from 0 through 12 octal digits. If there are fewer than 12 octal digits in a patch, the specified digits are inserted into the rightmost positions of the data word; the leftmost positions are filled with zeros. Consecutive commas (a patch consisting of 0 octal digits) result in a patch word that contains only zeros. Thus, up to 58 patches may appear on one OCT card.

Serialization, if present, is checked on the parameter cards as well as on the binary change-cards. A blank serial field terminates one serial check and begins another. Cards that are out of sequence will be noted on the system output unit during Phase 1; however, they will be used.

An example of a typical ост card follows:

1	د 8	16
11325	OCT	050000410351,007400410102,
		77777,,0,400000

This oct card would be used to insert a CLA instruction, a TSX instruction, and four parameters into six consecutive data words, starting with octal location 11325. The following shows the octal location and the contents of each patched word:

11325	050000410351
11326	007400410102
11327	000000077777
11330	00000000000
11331	000000000000
11332	000000400000

# Preparation of Programs to be Edited

Programs that are not assembled by the Macro Assembly Program must be pre-assembled in absolute column binary format. The System Editor does not consider relocation schemes. The relocation of Macro Assembly Program output is performed solely by the Loader, except in the case of subroutines to be placed in the Subroutine Library (IBLIB).

Programs must have a means of indicating the point to which control should be transferred after they are loaded by the System Loader. This may be accomplished by the transfer card produced by the Basic Assembly Program when the symbolic name of the entry point appears in the variable field of the END card. Determination of the entry point of a Macro Assembly Program output deck is accomplished by the Loader through inspection of the control dictionary for that program or inspection of the sentry control card. The Loader passes this information to the Editor program. The sentry card is omitted in input to the Subroutine Library.

An object program that is not in a format acceptable to the Editor Monitor, but which must be included in the System Library, can only be added by use of the DUP card. Blocks are copied from the DUP input unit onto the new library unit without alteration in format.

If the programs to be edited onto the new library unit are in a source language that can be translated by the Processor, in the Macro Assembly Program binary output card format, or in a format that can be treated by a processor that the installation has added to the System Library, the programmer must provide the control cards normally required by any of these.

For example, he must prepare \$18JOB, \$1EDIT, \$1BFTC, \$FILE, and \$ENTRY cards. In the case of the Macro Assembly Program output, he must ensure that the \$1BLDR, \$CDICT, \$TEXT, and \$DKEND cards (and possibly the \$ENTRY card) are present in the deck.

If a sodict card is encountered during any edit run, this card and the subsequent debugging dictionary will be ignored by the Editor; they will not appear in the phase on the new System Library.

#### Symbolic Unit Assignment

If symbolic channel assignment or symbolic unit reference is used in an object program that is to be edited into the System Library, the following points should be noted:

- 1. If a symbolic unit reference is specified, the absolute address of the Symbolic Units Table entry for the unit is taken from the Nucleus CDICT in the Subroutine Library at load time.
- 2. If a symbolic channel reference, intersystem reservation code, or "any unit" reference is specified, a symbolic unit is equated to the reference at load time and the absolute address for the unit is determined as in point 1 above. However, a unit that is available at load time may not be available at execution time.
- 3. If label searching is specified and the mounting option is not DEFER, the unit chosen is the one for which the label on the device is suitable at load time. The absolute address for the unit is determined as in point 1 above.
- 4. If deferred label searching is specified, the unit chosen is the one for which the label is suitable at execution time.
- 5. If NONE is specified, the Symbolic Units Table addresses of the primary and secondary units must be placed in the appropriate fields of the file control blocks before the file is opened. Subroutines are included in the Subroutine Library to search for appropriate units. One of these subroutines, s.scop, accepts as input any of the unit specifications that are allowed on the spile card or in the variable field of the file pseudo-operation. This subroutine converts the unit specification into codes that can then be used as input to other routines. A description of these subroutines is given in Appendix C.

### **Examples of Edit Runs**

#### **Edit Run for Disk**

Figure 13 shows the method of editing the System Library onto disk storage. It includes the control cards needed to format a 1301 Disk Storage unit before editing, as well as those needed to edit the system onto the disk. The following points are assumed:

- 1. The Nucleus of the master library has been reassembled so that s.sl.bl is attached as a disk unit.
- 2. The system was initially loaded from s.su19, a tape unit, using the special initial start procedure for loading the system from a unit other than s.slb1 (see the publication *IBM 7040/7044 Operating System* (16/32K): Operator's Guide, Form C28-6338). This will cause an automatic switch of s.slb1 and s.su19.

```
1 8 16
$DATE 01/29/65
$EXECUTE IBUTL
IBUFBW $.$U19/0-99/01/465
IBUFHW Z/0-3999
$185YS
$IBEDT $.$LB1,$.$U19
$ENDEDIT
$IBSYS
$STOP
```

Figure 13. Editing the System onto Disk

## Inserting a Program Into the Library

Figure 14 shows the control cards needed to edit the system sysx, composed of the phases LOG1, LOG2, and COMP, into the System Library. The new System Library is written on utility unit 6. After the system has been edited into the System Library, it may be called by means of the SEXECUTE card.

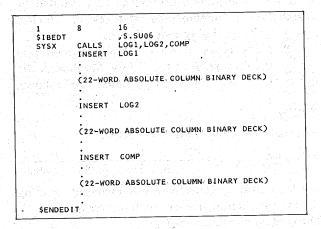


Figure 14. Inserting a Program into the Library

#### Replacement Using Symbolic Editing

Figure 15 shows the method of editing a source language program, in this case a MAP language program, into the System Library. Since the phase PHZ98 is replacing a previously existent phase, there is no need for another CALLS parameter. The LABEL option indicates to the Editor Monitor that it must check the first word of the master System Library for the values corresponding to version 3, modification 19. The 2 in the third subfield of the LABEL argument indicates that the master System Library is on two devices. The modification number 20 will appear on the new System Library. The Input/Output Label System checks the

retention date in the label on the unit used for the new System Library.

Although two master library units are indicated, the new System Library will be created on only one unit. A REMOVE EOR parameter is included to cause the remaining contents of the two units to be combined on one new library unit.

The Editor Monitor, finding the SOURCE option on the SIBEDT card, causes the edit file to be stacked initially with the REPLACE parameter. Control is released to the Processor to prepare a machine language version of the source program PHZ98. The Processor Monitor calls the Macro Assembly Program to assemble the program, directs the Loader (IBLDR) to place an absolute machine language version of the program on the edit file, and returns control to the Editor Monitor.

```
1 8 16
$IBEDT LABEL(3,19,2,1),SOURCE
REPLACE PHZ98
$IBJOB PHZ98
$FILE PHZ98 'FILEA',U00
.
.
. (SOURCE LANGUAGE PROGRAM IN THE MAP LANGUAGE)
.
$ENTRY PHZ98
REMOVE EOR
$ENDEDIT
```

Figure 15. Replacement Using Symbolic Editing

## **Modification Using Symbolic Editing**

Figure 16 shows the absolute source method of editing for system modification. The example illustrates how a routine for initial start processing can be inserted after the Nucleus.

```
$ I BEDT
                  SOURCE
$IBJOB
SIBMAP
        TRHOUR
                 ABSMOD
IBXIT
                           ABSOLUTE LOCATION OF IBXIT
        BOOL
                  7157
        ORG
                  IBXIT
                  IBTIM
                           OVERLAYS INST. SWT RUPSW
        ORG
                           LAST LOCATION + 1 OF HOUS
KEEPING IS S.SCBL + IBS15
IBTIM
        EQU
     (ROUTINE IN MAP LANGUAGE)
        SWT
                 RUPSW
                          RESTORE OVERLAID INST.
                 IBXIT+1
        TRA
                 IBXIT+2
IBMVE
       BOOL
                           START OF HOUSEKEEPING
                 IBMVF
        END
SENTRY
$ENDEDIT
```

Figure 16. Modification Using Symbolic Editing

The Assembler produces a standard entry point for every deck it assembles. Since the deck that is being assembled, in most cases, will not have the same entry point as the system record that is being modified, the entry point of the system record is designated by placing a symbol in the variable field of the END card.

The origin of the modification (specified in the variable field of the orc pseudo-operation) must be the first location at which the modification is to be placed.

#### Inserting a Relocatable Subroutine

Figure 17 illustrates the insertion of a program into the Subroutine Library (IBLIB) and the removal of a program already in IBLIB. The LIBE parameter specifies the library name and the format of the cards. The sibedt card does not contain the nosource option because the Editor program processes the MAP binary decks and control cards itself when a run affects only the Subroutine Library. In this example, the master System Library is on s.suo7, and the new one is chosen by the Editor since it is unspecified.

```
16
SIBFDT
                5.5007
                IBLIB, IBMAP
       LIBE
        INSERT
                DECKXY
$IBLDR
$CDICT
       DECKXY
        (CONTROL DICTIONARY FOR DECKXY)
$TEXT
       (BINARY TEXT FOR DECKXY)
$DKEND DECKXY
       REMOVE
                DECKPO
       LIBEND
$ENDEDIT
```

Figure 17. Inserting a Relocatable Subroutine

## Adjustments After Reassembling the Nucleus

If the Nucleus or a relocatable program used by the Processor, such as the Input Editor or the Output Editor, is reassembled and system symbols are redefined, all subsystems may require reloading to adjust their references to the changed program. The subroutines with their revised references must be inserted into the Subroutine Library (IBLIB), and the Control Section Name List and the Dependency Name List must be regenerated. After the library has been revised, it is possible for the Loader to reload the subsystem programs that are affected by changes, e.g., to the Nucleus, with the altered locations of entry points, and data, properly adjusted.

A pause may be made with the spause card, if not enough input/output devices are available to allow a device to replace the one containing the original System Library. It is possible in two (uninterrupted) runs to obtain a properly reloaded System Library. A

similar situation arises when a revision to the input/ output editors necessitates an edit run to reload certain system programs.

The sswitch card is used to switch the library function with a utility function; the only changes on the first newly written System Library are the record counts of various Table of Contents entries and the IBLIB, SRNAD, and IBREL logical records. Upon determining that the s.slb1 function is being switched, the Supervisor reads in a new Index and Table of Contents, a new IBSUP phase, and resets the contents of location s.sdex. The Loader on the new System Library relocates those systems programs within the second edit deck, against a revised IBLIB. At the completion of the edit run for the second deck, a new System Library, containing a new Nucleus and a properly adjusted subsystem program, is ready for testing.

## Inserting a Source Language Subroutine

Figure 18 shows the input deck used to insert the relocatable subroutine ARCTAN into the Subroutine Library. The Editor Monitor passes control to the Processor to translate the FORTRAN source language subroutine and stack it on the edit file in relocatable format. After this edit run, ARCTAN will be the first subroutine in the Subroutine Library. Since the Loader (IBLDR) is not used on decks to be placed in the Subroutine Library, the SENTRY card is not used. Instead, the SIEDIT card returns control to the Editor Monitor.

```
12/31/63
$DATE
$JOB
SIBEDT
         REMARK
                 SYMBOLIC LIBRARY UPDATE
        INSERT
                 ARCTAN
                  NOGO
$ I BJOB
         ARCTAN
$ I BFTC
         (FORTRAN STATEMENTS FOR ARCTAN SUBROUTINE)
$JEDIT
         LIBEND
$ENDEDIT
SIBSYS
                  ...NEXT JOB...
$JOB
```

Figure 18. Inserting a Source Language Subroutine

# Inserting a New Subroutine Library

Figure 19 shows the input deck used to insert the new Subroutine Library iedbg for the Debugging Processor. The Calls cards causes the three names iedbg, srndb, and ibdg to be added to the Table of Contents. These names are counterparts of iblib, srnad, and ibrel in the Subroutine Library iblib. After this edit run, iedbg

will appear after phase IBDEF of the Debugging Processor.

```
8
                 16
SDATE
                 12/4/64
SJOB
SIBEDT
                 IEDBG , SRNDB , IBDBG
DEBUG
        CALLS
        AFTER
        LIBE
                 I EDBG
        (DECK)
        LIBEND
SENDEDIT
SIBSYS
SSTOP
```

Figure 19. Inserting New Subroutine Library

#### Inserting a Chain Program

Figure 20 shows the procedure to be followed to insert a chain program into the System Library.

```
SDATE
                  INSERT MAINAM ONTO SYSTEM
$JOB
$IBEDT
                  NOSOURCE
                  CALLS MAINAM, LINK1, LINK2,
        MAINAM
                  AFTER SUBR
                  INSERT MAINAM
                  NOSOURCE
$IBJOB PROGR
                  MAINAM
$CHAIN
$ENTRY
                  LINK1
$LINK
$ENTRY
                  LINKN
$LINK
SENTRY
SENDCH
SENDEDIT
$IBSYS
```

Figure 20. Inserting a Chain Program

The program should then be called as follows:

```
$DATE
$JOB
$EXECUTE MAINAM
$IBSYS
$STOP
```

#### **Special Considerations**

Because of the simulation of scatter-loading of System Library programs, the Editor program must end a block whenever there is a break in contiguity within the edited program. Such breaks occur because of BSS areas, new origins, etc. Thus, to avoid unnecessary interrecord gaps on a tape library unit, or poor usage of disk track space, depending on the programs edited into the System Library, it might be practicable to replace BSS areas within the program data length by words of zero. For BSS areas appearing after the data length of the program, no problem arises.

It might be desirable to patch an existing, previously loaded program by inserting a one-word overlay. If

this one-word overlay is contiguous with data in the program, no problem arises. However, if the one-word overlay is not contiguous with other data, a new block will be created for this single word. Since all blocks must be three words or longer, a zero word will be appended to the data word by the System Editor. Such a block would cause a word in the overlaid program to be lost, since the System Loader saves only the single word overlaid by the chain word.

To avoid this problem, such a one-word overlay should always be lengthened to two words by also overlaying the word following the originally desired overlay address. This would bring the data block, plus the chain word, up to the standard block length, and no zero word need be appended.

# Interface with the Processor

Upon encountering SOURCE or NOSOURCE on the SIBEDT card, the Editor Monitor prepares to pass control to the Processor to have the input to the Editor program preprocessed, that is, compiled, assembled, and, if necessary, relocated. The Editor Monitor defines the edit stack file as described in the section "Editor Monitor," turns the edit flag bit on in location s.sflc, and initializes the Nucleus communication word s.edun.

Then, the monitor copies all cards from s.sin1 onto the edit stack file, except Processor control cards such as those processed by the Loader or by the FORTRAN and COBOL compilers. The Processor control cards are indicated by the presence of a sibjob card immediately after the Editor parameter card naming the phase to be replaced, inserted, or modified.

Upon encountering the sibjob card, the Editor Monitor allows the Processor to gain control of the system input unit, s.sin1. The Processor then causes the material affecting the specified phase to be converted and placed on the edit stack file, immediately after the Editor parameter card. Following this stacking by the Processor, control is returned to the Supervisor by means of a control card, sjedit, which is placed in s.save by the Loader (ibld). This card causes the Supervisor to return control to the Editor Monitor, which recognizes that an edit stack file is being prepared.

The Editor Monitor then sets a switch in the Supervisor to cause the card-reading routine, IMRET, to reenter the Editor Monitor when a binary card or a BCD card that is not a \$ card is read. Such cards are stacked on the edit stack file. Control is given either to the Processor, because a \$IBJOB card follows an Editor parameter card, or to the Editor program, because a \$ENDEDIT card is found. When the last control card is encountered, the edit stack file is closed and rewound, and control is given to the Editor program through

s.sldr. Phase 1 of the Editor program now reads the edit stack file instead of s.sin1, and editing proceeds as usual.

# \$JEDIT Card

This card is used to terminate each source insertion or replacement that affects the Subroutine Library. This card is also generated by the Loader after the Loader has stacked all material for a phase to be inserted or replaced in the System Library. Upon reading this card, the System Monitor transfers control to the Editor Monitor which processes the Editor control or parameter card.

## **Special Conditions**

To cause the Editor Monitor to enter a monitor other than the Processor Monitor, the \$ control card for the particular monitor must be entered in the Recognizable Control Card Table for the Supervisor. During an edit run, the presence of the \$ control card after an Editor parameter card causes control to be given over to the specified monitor.

Further modification may be effected by causing the Editor Monitor to recognize options other than source or nosource on the sibed card and by transferring to the necessary routines to operate with the new monitor. The list limited by symbols ired and ired should be expanded to include the new option and the entry point to the new subroutine. If the new monitor accepts the opening and stacking by the Editor Monitor of the edit stack file, and the setting of bit 24 of s.sflg, the only change required is the addition of the new monitor's control card and entry point in the Recognizable Control Card Table for the Supervisor. In this case, the appearance of nosource on the sibed card is sufficient to accomplish proper preprocessing for the edit run.

# Editor Program Off-Line Messages

The following messages are listed off-line on s.sou1. Any one of the three action messages below may be appended to the message text.

- 1. Condition ignored indicates that the error (check sum error, etc.) is accepted.
- 2. Condition cannot be ignored indicates that editing proceeds by deleting the erroneous data (invalid parameter card, etc.).
- 3. Edit terminated indicates that the edit run is stopped and that control is returned to IBSUP.

If no action message appears, the message comes from the Editor Monitor.

#### 11500 BEGIN EDIT

This message is typed and control is transferred to Phase 1 of the Editor following opening of the input/output devices

and interpretation of the \$IBEDT card. This message also appears following possible use of processor programs for special preprocessing of edit input.

11501 BEGIN EDIT PREPROCESSING, EDIT FILE

This message is typed when the Editor Monitor temporarily releases control to the Supervisor for reading and routing of a System Monitor control card, such as a \$IBJOB card, because of the SOURCE-NO SOURCE option (or a similar option) on the \$IBEDT card. For example, the \$IBJOB card causes control to pass to a Processor component to compile, assemble, and/or load the phases to be edited. So that it may be saved for future use, the edit file is identified by means of the EDTFIL (nn) option of the \$IBEDT card. Message number 11500 is typed when the \$ENDEDIT card is read, and control is transferred to the Editor program.

11502 PREPROCESSOR DETECTED ERROR

One or more errors that would cause deletion of execution in a non-edit run have been detected by IBLDR and/or IBMAP during edit preprocessing. The specific error(s) is listed on the system output unit. This message will always be followed by message 11541.

11503 S.SU01 NOT AVAILABLE; NEWLU IS xxxxxx

The new library unit was not specified in the \$IBEDT card.

Since S.SU01 is reserved, xxxxxx is the unit chosen for the new library unit.

11504 XXXXXX FIELD IN ERROR, OPTION IGNORED

This message is typed when an option on the \$IBEDT card is not valid. For example, this message would be used if the LABEL option specifies six new library units or if the specified new library unit duplicates the edit stack file.

11505 ILLEGAL PARAMETER xxxxxx

xxxxxx is the contents of columns 8-13 of an input card that is not a valid Editor parameter card. The edit run is terminated.

11506 DUPLICATE PHASE NAME, NEW PHASE OF 'XXXXXX' SKIPPED, CONDITION CANNOT BE IGNORED

The specified phase name, xxxxxx, on an INSERT card already exists within the System Library and cannot be re-entered. The phase already in the library is retained; the input phase is ignored.

11507 CHECK SUM ERROR, CARD NO. xxxxxxxxx, CONDITION IGNORED

11507 CHECK SUM ERROR IN PHASE XXXXXX, LOAD ADDR=XXXXX, CONDITION IGNORED

11508 CARD ORDER ERROR, XXXXXXXX, CONDITION IGNORED

Card serial numbers in columns 73-80 are not in sequence.

11509 PARAMETER CARD SEQUENCE ERROR, OPER-ATION 'xxxxxx' ON 'yyyyyy' RESEQUENCED, CONDITION IGNORED

The order of phase alterations in the edit run input should be in the same order that the phases appear in the System Library. If the order is different, the Editor will resequence the input phase alterations to agree with the phase order in the System Library, as shown in IBTOC. To accomplish the resequencing, Phase 1 of the Editor will start a sort sequence, thereby slowing the speed of the edit run. Parameters affecting the Subroutine Library must be in the same order as the subroutines within the library.

11510 XXXXXX DOES NOT EXIST, CONDITION CANNOT BE IGNORED

A REPLACE, REMOVE, or MODIFY parameter cannot affect a phase name not already within the System Library.

11511 ILLEGAL DUP OPERATION 'xxx...xxx', CON-DITION CANNOT BE IGNORED

Certain DUP requests (e.g., 'S.SLB1, S.SLB2, 4') cannot be processed; they are deleted from the edit run.

11512 INVALID LOAD ADDRESS FOR OCT 'xxxxxx', CONDITION CANNOT BE IGNORED

An OCT patch card has an invalid character in the load-address portion. The complete card is deleted from the edit run.

11513 INVALID CHARACTER FOR OCTAL, 'xxxxxxxxxx' SET ZERO, CONDITION CANNOT BE IGNORED

An OCT patch card has an invalid character in its data-words field (columns 16-72). That data word is changed to zeros.

11514 ILLEGAL REWIND OF S.Sxxx, CONDITION CANNOT BE IGNORED

A REWIND operation (e.g., on the new library unit) cannot be accomplished; the operation is deleted from the edit run.

11515 BLANK CARD READ CONDITION IGNORED

11516 IBTOC CANNOT BE LOCATED ON OLDLU= 'S.Sxxx', EDIT TERMINATED

When an old library unit, other than S.SLB1, is specified on the \$IBEDT card, it must be searched for the Table of Contents. The inability to find the Table of Contents, because of a permanent input/output error or reading a file mark, terminates the edit run.

11517 ILLEGAL PARAMETER FOR LIBE EDITING 'xxxxxx', CONDITION CANNOT BE IGNORED

The parameter cards allowed between the LIBE and LIBEND parameters do not include MODIFY, sysnam RE-MOVE, DUP, or REWIND. The parameter card is deleted from the edit run.

11518 ENTRY POINT FOR PHASE 'xxxxxx' MISSING, ZERO ASSUMED CONDITION IGNORED

A transfer card or an MZE record (from IBLDR) was not read before the reading of a parameter card. The Editor supplies a phase on the new library unit whose entry point is zero. As exception is made for the MODIFY parameter, this may be altered by a later run using the MODIFY parameter, which contains only a transfer card.

11519 UNEXPECTED PARAMETER CARD READ, CONDITION CANNOT BE IGNORED (or EDIT TERMINATED)

If Phase 1 reads, for example, two REPLACE cards with no data between them, the first REPLACE card is deleted, and the phrase CONDITION CANNOT BE IGNORED is appended to the message. Phase 2 terminates the edit run, as this condition is probably caused by a machine error. The IODER routine of IOOP is used to indicate the probability of input/output error conditions.

11520 UNEXPECTED BINARY RECORD READ, CONDITION CANNOT BE IGNORED (or EDIT TERMINATED)

Phase 1 or 2 expects a parameter card, but reads a binary card or load record (from IBLDR). If this occurs in Phase 2, the edit run is terminated, as it is probably a machine error. The IODER routine types an error message.

11521 UNEXPECTED EOF READ, CONDITION CANNOT BE IGNORED (or EDIT TERMINATED)

Phase 1 or 2 expects data, but reads a file mark. IODER also types an error message specifying the unit in error. If this occurs in Phase 2, the edit run is terminated.

11522 'xxx ... xxx' WAS READ, 1EOR/1EOF EX-PECTED, EDIT TERMINATED

Phase 2 expects a trailer label after the file mark on the old library unit, but reads something else.

11523 'xxxxxx' NOT IN TABLE OF CONTENTS, CONDITION CANNOT BE IGNORED (or EDIT TERMINATED)

All phases to be edited into the System Library must appear on a CALLS parameter to be entered in the Table of Contents. For example, any INSERT parameter that specifies a phase that is not on a preceding CALLS card or in the Table of Contents is deleted from the edit run. A phase being duplicated onto the new library unit may also cause this error.

#### 11524 ILLEGAL DEVICE TYPE, S.Sxxx, EDIT TER-MINATED

The unit-type indicator in the system control block for the unit S.Sxxx shows that the unit requested for editing may not be used. For example, the card reader cannot be the new library unit. The IODER routine types an error message similar to this off-line message.

#### 11525 PROBABLE MACHINE ERROR AT LOC. XXXXX, EDIT TERMINATED

See listing at octal location xxxxx for the condition that caused this message.

#### 11526 OLDLU PHASE READ XXXXXX, EXPECTED yyyyy, EDIT TERMINATED

The Subroutine Library Edit routine expects one of its three phases (IBLIB, SRNAD, or IBREL), but reads another phase.

#### 11527 XXXXXX DECK SEQUENCE ERROR, CONDITION CANNOT BE IGNORED

The Subroutine Library Edit routine requires the relocatable MAP decks to be in sequence number order; also, the \$ control cards must be in the order: \$IBLDR, \$CDICT, \$TEXT, and \$DKEND.

#### XXXXXX DUPLICATE DECKNAME, CONDITION 11528 CANNOT BE IGNORED

The Subroutine Library Edit routine cannot insert the name of the duplicate relocatable subroutine; the input deck is deleted.

#### UNEXPECTED \$ CONTROL CARD ENCOUN-11529 TERED, EDIT TERMINATED

The Editor has read a \$ card which it cannot process. This \$ card is placed in S.SAVE for interpretation by the System Monitor, and the Editor transfers to the System Dump program, which skips to the next job.

#### CARD GENERATED FOR LIBRARY ROUTINE. 11530 CONDITION IGNORED

The Editor has found the \$DKEND or LIBEND card to be missing from the edit run input and has attempted to supply the proper card in order to continue editing.

Note: Further meaningless error messages may result from this attempt.

#### ILLEGAL XXXXXX EOR REQUEST, CONDITION 11532 CANNOT BE IGNORED

The symbol xxxxxx is either REMOVE or INSERT. No more than the assembled limits of S.SLBx units may be created by INSERT EOR; no more EOR trailer labels can be removed than already exist. Pauly Caist.

#### IMPROPER LIBE INPUT - REMAINING DECKS 11533 IGNORED. CONDITION CANNOT BE IGNORED.

An error is present in the Editor input that follows the LIBE card. The error may be either of the following:

1. The order in which subroutines are specified in the input deck does not correspond to the order of the subroutines in the library.

2. A subroutine specified on a REPLACE, REMOVE, or AFTER card is not contained in the library.

#### 11534 SECONDARY XXXXX NEEDED BUT UNSPECI-FIED, EDIT TERMINATED

xxxxx is either NEWLU or OLDLU. This message is produced when the input to an edit run requires a secondary unit but the secondary unit has not been specified.

#### IBTOC ENTRY REMOVED FROM TABLE OF CONTENTS, CONDITION CANNOT BE IG-NORED .

The IBTOC entry in the Table of Contents may not be removed either by a phase REMOVE card or a subsystem REMOVE card. The edit is terminated.

#### XXXXXXXXXXX ERROR ON S.SXXX nnnnn EDIT 11540 TERMINATED

The twelve x's indicate the type of input/output error. nnnnn is the device address.

#### UNRECOGNIZABLE INPUT TYPE ENCOUN-11543 TERED, PHASE XXXXXX NOT PROCESSED, CON-DITION CANNOT BE IGNORED

The binary data following an Editor parameter card is not of a form directly acceptable to the Editor. The data and the parameter card are deleted from the edit run.

#### 11544 NO MERGE DEVICE AVAILABLE, CONDITION CANNOT BE IGNORED (or EDIT TER-MINATED)

In certain cases, Phase 1 of the Editor must sort and merge the input for a phase, or must attempt to resequence the input phase order. If no merge device (new library unit) is available (since it might be the edit file at the moment), a merge requirement will terminate the edit, while a resequence condition will result in merely dropping the out-of-sequence phases(s).

## S.SUxx CANNOT BE USED FOR MERGING

Phase 2 of the Editor must perform a duplication pass to insert an updated Table of Contents into the new System Library. S.SUxx, the work unit to be used for this pass, cannot accept the records from the new library because their block size is too great. The edit run is terminated.

EDIT COMPLETE - NEWLU xxxxxx, AND yyyyyy The edit run has been completed, xxxxxx is the primary NEWLU, and the secondary NEWLU, if applicable, is indicated by yyyyy.

# System Monitor

The System Monitor provides for continuous machine processing of jobs that require subsystems of the IBM 7040/7044 Operating System. It accomplishes this by providing a constant area of core storage, the Nucleus, for communication of information from subsystem to subsystem. A supervisory routine, the Supervisor, processes System Monitor control cards to change the core storage environment, symbolic unit assignments, and the input/output device configuration and to pass control to the subsystem or program required by the next application. An Input/Output Control System and several routines in the System Monitor are also provided for the user.

#### The Nucleus

The Nucleus contains the data and tables that must be passed from subsystem to subsystem and routines that may be useful to any object program. The contents of the Nucleus may be divided into the following functional sections:

- 1. Words allocated for machine use
- 2. System transfer points
- 3. System data areas
- 4. Control blocks
- 5. Other tables
- 6. Nucleus routines
- 7. Lower levels of the Input/Output Control System Figure 21 shows the approximate core storage allocated to the Nucleus.

The locations of the ends of iols and iobs may be greater than indicated if core storage protection is used by the System Monitor. Neither iols nor iobs is storage protected; the erasable portion of iols and all of iobs must follow the end of storage protection.

Lower storage is reserved for communication words and system routines of the Nucleus, 10ex, and 100p1. 100p2, 10ls, and 10bs may be overlaid by a subsystem or an object program. The System Monitor restores these whenever control is returned to it. The entire Nucleus is defined at system assembly time and is assembled in absolute format.

#### **Words Allocated for Machine Use**

The first 93 words of core storage are allocated to machine functions.

#### Trap Words

Locations 0-33  $(0-41_8)$  are trap words initialized by the System Monitor at initial start. They are used to transfer control to trap routines in IOEX.

ក្នុងប្រធាន ក្នុងស្ថេនដែល ខណៈនៃការប្រកាសដែល។ សិក្សាសិក្សា សេក្សានិ <b>Section</b> (សេក្សាសិក្សាសិក្សា)	didi. Santa	Approximate Size
BNUC	1777	
Machine Functions	93	
Pointers and Data Words	120	<b>通信的证据</b> 第三人称
*Symbolic Units Table	1.4	ord/symbolic unit
*Unit Control Blocks	9 v	ords/device
*System Control Blocks	.4 w	ords/symbolic unit
(Unit record, Tele-processing,		The Court of the
magnetic tape, or 7740		
input)		Sales service at the sale
*System Control Blocks	8 v	vords/symbolic unit
(7740 output or		
1301/1302/7320)		o materio Gartalo e de Cili. Como escabe e de
Nucleus Routines	600	n Norway of State (1997) Anna State (1997)
IOEX	. 20 te e. Het ave et	an in a markan persentan ing Santan pelalah dianggan persentan
Basic Functions	510	
*Basic Functions Channel Tables	14 v	vords/channel
1301/1302/7320 Functions	140	ANALY STATES
**1301/1302/7320 Functions	4 \	words/channel (approx.
Channel Tables		1.5 words/module)
Interrupt Scheduler	150	
*Interrupt Scheduler Channel	1,	word/channel
Pushdown List	16 1	words/level (level=IBRS
Storage Protect	40	
IOOP		
greenes, nated is god valvough no made a line of the color of the colo	320	
Magnetic Tape	350	
**1301/1302/7320	500	
***Unit Record	170	
Telecommunications	349	
Common	120	
IOLS		and the man of the second of the second
Reel Processing	290	
Labels	360	
System	70	
(18일 : 15년 1일 : 15일 : 15일     TOBS	940	
		<u> 18 17 18 18 18 18 18 18 18 18 18 18 18 18 18 </u>

\*\*\* P2ND is located at the end of the Unit Record section. This section varies, depending on the devices (e.g., 1402, 1403) specified during system assembly.

Figure 21. Approximate Core Storage Allocation

A description of the content of each word is given in Figure 22.

#### **Engineering Words**

Locations 34-63 ( $42-77_8$ ) are reserved for engineering use.

#### Load Area

Locations 64-92 ( $100-134_8$ ) are reserved as a read-in area for loading.

Octal Location	Label	Initialize OP	d to Address	Description
0	IBTFP	PZE	0	Address word for floatin
elb side	ew polici	afodri		point overflow and ur
la běá:	ver sé		grafia ab	derflow traps and ST
1		VFD	12/IXTPR	trap Used to reset the tra
	erinereteki Vertu jirk	ETC	12/0	words When Interval Timer no
			12/0	available
			or	
Tell 19		ETC	12/IXTPI	When Interval Timer avai
		ETC	12/9	<b>uble</b>
2		TRA	IXTPR	Transfer word for STR tra
3,	IBTDD	PZE	0	Direct Data trap words
4	arpen et	TRA	34	
5	IBCLK	PZE	0	Interval Timer Count
6	IBTTI	PZE	0	Interval Timer Overflow
7.		TRA	IXTPI	trap words When Interval Timer avail
		IKA	IAIFI	able
			or	uble
		TRT*	IBTTI	When Interval Timer no
10	dayan .	TOA		available
		TRA	11	Transfer word for floating point overflow and un
		fortel.	4. 2	derflow traps
11	ing salah salah Kancasan salah	TRT*	IBTFP	Transfer word for floating
				point overflow and un derflow traps
12	IBTCA	PZE	0	Channel A trap words
13		TSL	IXT00	
14	IBTCB	PZE	0	Channel B trap words
15		TSL	IXT00	
16 17	IBTCC	PZE	0	Channel C trap words
20	IBTCD	TSL	IXT00	
21		PZE TSL	0 IXTOO	Channel D trap words
22	IBTCE	PZE	0	Channel E trap words
23	, inverse	TSL	IXTOO	Chamier E frap words
24-30	i de la ci	ja is or		Assembly options and date
31	er en eks Gwelen e	rojation. Valenti		of assembly
1975 Bright	IDTCD			Not used
32	IBTSP	PZE	0	Storage Protection trap word
33		TRT*	IBTSP	When Storage Protection
		, la sit. Halifa iyo	or	not available
	j kroby	TRA	IXTPS	When Storage Protection
34	Proposity in The Control of the Cont	RCT		available Restores trapping for
25	Argueleta Alba			Direct Data trap
35	2014 1814 2 <u>122</u> 184	TRT*	IBTDD	
36	IBTIR	PZE	0	Interval Timer Reset trap words
37		TRT*	IBTIR	words When Interval Timer not
		and Mar		available
		TRA	or IXTPL	(W) 12 ( 12 1 4 2
		INA	TATES	When Interval Timer avail- able
40	IBTPA	PZE	0	Storage Parity trap words
41		TRA	IXTPY	

Figure 22. Trap Words

### **System Transfer Words**

The system transfer words begin at location 93 (135<sub>8</sub>). They contain transfer instructions to entry points in the Nucleus routines, 10ex routines, 10er, 10

Label	OP	Address	Routine
Total La	ORG	IBNUC	
S.SLDR	TRA	IBLOD	System Loader
S.SRPT	SWT	RUPSW	Operator interrupt test
S.SDMP	BRA	IBDMP, ,	Dump routine
		IBDMP	
S.SRUP	TRA	IBRUP	System Monitor Recall routine
S.SRET	BRA	IBRET, ,	Return routine (if disk is not
		IXTRL	assembled, the decrement is
S.SRST	TRA	IBRST	Restart routine
S.SCCR	TRA	IBCCR	Change Communication Region
S.SIDR	TRA	2.4	Installation Accounting routine
S.XACT	TRA	IXACT	IOEX Active routine
S.XDAC	BRA	IXDAC,,	IOEX Deactivate routine
Funda Gild		IXCON +	
	na referenció Maño de la Se	IXCDX	
S.XPRT	TRA	IXPRT	IOEX Print routine
S.XPSE	TRA	IXOPP	IOEX Pause routine
S.XOVA	TRA	IXOCV+1	
S.XOVD	TRA	a Birthari	Binary to octal, address
1.00 34 3.15		IXOCV	Binary to octal, decrement
S.XDVA	TRA	IXDCV+1	Binary to decimal, address
S.XDVD	TRA	IXDCV	Binary to decimal, decrement
S.XUCV	TRA	IODEV	Change UCUNI to display format
S.SCKT	TRA	3,4	Checkpoint routine
			(Must be initialized by the
, en	erskira sakira		object program.)
S.IOOP	TRA	IOP00	IOOP Unit Synchronizer
S.TPBF	TRA	ITPBF	Tele-processing Buffer
CHAD DAMS			System (assembled only if
	mitur Gu		Tele-processing devices are
Take Market			attached)
S.IOLS	TRA	IL\$00	IOLS Verification and Creation routines
S.OPEN	TRA	ISN00	IOBS Open routine
S.OPNL	TRA	ISN01	IOBS Open List routine
S.GETL	TSL	ISGPE	IOBS Get Logical Record
UZ ALTAVE	over the state		routine
S.GETB	TSL	ISGPE	IOBS Get Physical Record
male (A)		To Aliga Shirth	routine
S.PUTL	TSL	ISGPE	IOBS Put Logical Record routine
S.PUTB	TSL	ISGPE	IOBS Put Physical Record routine
S.PLOC	TSL	ISGPE	IOBS Put Locate routine
S.CLSE	TRA	ISC00	IOBS Close routine
S.CLSL	TRA	ISC01	IOBS Close List routine
S.BSR	TRA	ISBSB	IOBS Backspace Record routine
S.WEF	TRA	ISWEF	IOBS Write File Mark routine
S.REW	TRA	ISREW	IOBS Rewind routine
S.FEOR	TRA	ISFER	IOBS Force End-of-Reel routine
S.CKPT	TRA	ISCPT	IOBS Checkpoint routine
S.BDMP	EQU	ISDME	IOBS Error Routine

Figure 23. System Transfer Words

### System Data Areas

System data areas are classified as follows:

CONTENTS

- 1. Constants set from system assembly parameters
- 2. Variable data set by subsystem components for communication to other components
- 3. Tables describing the logical and physical organization of the system

### Constants a light of the same state of the design of the same state of

SYMBOL

The following data provided by assembly parameters is stored in this region:

	and the Victorial Control
PZE	S.SEND,, IBORG+1
	e legis de la composition della composition dell
	ILEND,, ISEND
VFD ETC ETC ETC	1/IFSNS 1/SHARE 16/LABELS 18/IP2ND+1
	PZE PZE VFD ETC ETC ETC

la malaysis i May May selbig it y May a 196

S.SLVL PZE nn,, mm

This word contains the version number, mm, and the level number, nn, of this system.

These are the limits of core storage available to object program.

IBORG+1 equals
IP1ND or S.SORG,

value. ILEND is I plus the last location of IOLS. ISEND is I plus the last location of IOBS. The sign bit of this word indicates the setting of IFSNS in the Nucleus as follows:

whichever is the greater

+ IFSNS is 0 - IFSNS is 1

The value of SHARE indicates which set of data control characters is being used.

LABELS indicates whether or not the installation uses labeled storage media as follows:

0-Label creation and verification routines do not exist on the System Library.

1-Label creation and verification routines are on the System Library, but blank reel header labels are not verified.

2-Label creation and verification routines exist on the System Library and all output reels must be checked for purge dates.

IP2ND+1 is the last location of IOOP2. MXCLK is the value to which the interval timer should be set. The value is the maximum number of minutes that a job may run.

NTRES contains the reservation code of the interrupt unit for input during the job interrupt procedure.

Percent of redundancies that, if exceeded, SYMBOL CONTENTS

causes an on-line message to be typed when a \$STOP card is read.

The next group of words provides information concerning the location and length of the tables within the Nucleus. These symbolic locations can be considered to be pointers to these tables.

SYMBOL	CONTENTS
S. SUBC PZE	E IBUCB,, CHES

Edgen State Tomber of Teachers

5 This word points to a table delimiting the unit control blocks. This table contains one word per channel and has the following format:

PZE lucbx,,lgx where lgx is the number of words required by the unit control blocks for a specific channel, x, and lucbx is the location of the first of these blocks. IBUCB is the location of this table; CHE5 is the number of channels.

S. SSBC PZE IBSCB,, CHE5

IBSCB is a table in the following format that delimits the system control blocks:

PZE lscbx,, lgx where lgx is the number of words required by the system control blocks for a specific channel, x, and lscbx is the location of the first of these blocks. IBSCB is the location of this table; CHE5 is the number of channels.

S. SUNI PZE S.SU00,, n

S.SU00 is the first of n number of utility units. This information is used to assign a utility unit to a program.

S. SLTC PZE IBLTB,, IBLTZ\*3 The address of the word contains the location of the first word in the Abbreviated Table of Contents; the decrement gives the length of this table.

S. SRCC PZE IBRTB,, IBRTZ\*2

The address of this word contains the location of the first word in the Recognizable Control Card Table; the decrement gives the length of this table.

Pointer to IOBS label

S.BSLA PZE ISLBL

### area.

Variable Data

The following variable data is used for communication among the subsystem components:

SYMBOL CONTENTS
S. SDAT BCI 2,mmddyy0yyddd The

DESCRIPTION
The current date is stored here from the \$DATE card or the

S.SCMX PZE MXCLK, NTRES

S. SPER PZE IBPER

SYMBOL	CONTENTS	DESCRIPTION	SYMBOL	CONTENTS	DESCRIPTION
		entry keys. It may be			Up to 15 codes are
		used by any part of the system.			available to an installa-
S. SCLK PZ	E **	This word contains the			tion. This code is set by each subsystem moni-
		time of day at the time when the interval timer		에 살려 있는 사이 활동하는 것 4. 이번 사이 아이들 것이다.	tor as it receives con-
		was last set in binary	S. SFLG	DEC ,,,,,,,	Those on As
		format. It is set by the		DEC ,,,,,,,,	These 20 flag words contain information
		Supervisor before con- trol is passed to any			that must be trans-
		subsystem. This infor-			mitted from one part of a processor to another.
		mation may be used by the object program.			(Figure 24 illustrates
S. SCIS PZ	E **	This is the last value to		ani di Maraki Aliyosi da k	the usage by the Processor.)
		which the clock was set by the IMTIS routine.	S.SAVE	DEC,,,,,,,,,,,,	, These words contain
		This routine is called	Marke ( ) . Certi		the next control card to be processed by the
		by every monitor be- fore any exit to a sub-			system when one of the
c cnev c		system.			combined monitors is not in control. This
S. SDEX pfx pfn		This information is used by the Supervisor, the			card was read by some
		Editor Monitor, and the	and the second of the second s	n de Capada pelasina La Barana da Santan	system part and was stored in this location
		Sort Monitor to read the Table of Contents from	adecrati		for use by the S.SRET
		the System Library. It	S.SCDI	pfx ,,**	routine. The sign bit of this
		is provided by the Supervisor at an initial			word indicates the pres-
		start. la is the load ad-			ence or absence of a card in S.SAVE as fol-
		dress of the first block of the Table of Con-			lows:
		tents phase. In is the			<ul><li>Card present.</li><li>Card absent.</li></ul>
		number of words in that block, pfn is the	Kada Keda I		S.SCDI is set to minus
		number of the library			by the INSAVE rou- tine of the (Processor)
		unit, and re is the block number.			Input Editor, by the
S. SCUR BC	I 1,	The name of the cur-			close function of the System Input Editor,
		rent subsystem in con- trol is stored in this			or by a routine of
		location. S.SCUR is set			similar function in the assembler or the com-
		to IBSYS by the Super- visor at system assem-			pilers. The S.SRET rou-
		bly. It is reset by a sub-			tine resets it to plus when the card is used.
		system monitor when it receives control. When			The decrement con-
		control is returned to	gods (1 gi) - fil (i		tains one of the follow- ing codes indicating the
		the Supervisor, it is reset to IBSYS. It is			reason for recalling the
		also reset to IBSYS			System Monitor: 1-Return without
		when control is given to an object program.		The state of the s	the next card
		S.SCUR is used by the System Dump program.			2—Return with the next card
S. SFAZ BCI	1,	This word is set by		and the second of the second o	3—Initial start 4—Return for inter-
		the System Loader (S.SLDR) to the name			rupt
		of the last phase that	Maria Salah		This information is used by the S.SRET rou-
0.00		was loaded. It is used by the System Dump.			tine and by the Sys- tem Monitor.
S. SSWI PZE	,, <b>*</b> *	S.SSWI contains one of	and the second s	pfx **	This word contains the
		numbers, identifying		ng in merengawan ditrak Najar menengan penganan ditrak	listing page count. The sign bit of this word is
		the subsystem in con- trol to the Supervisor.		e de la companya de l	the list switch as fol- lows:
		1—The System Moni- tor, the Utility			+the list switch is
		Monitor, the Up-			on.  —the list switch is
		date program, and the object pro-		a de la companya de La companya de la co	off. The switch is assem-
		gram	C CITION .		bled as —.
		2—The Processor 4—The System Editor	S.SHDR 1	BCI 5,	These words contain the heading text from
		8—The Sort program			the \$JOB card.
					and the second s

ord_	Bit	Name	Set by	Used by	ŀ	Word		Name	Set by	Used by
1.,	S	SOURCE flag	IBJOB	IBJOB, IBMAP, IBLDR, DEBUG		4	S	Indicator for \$POOL chain	LDRPRE	IBLDR
	1	DECK flag	IBJOB	IBMAP, IBJOB			1-2	T, COL CIIGIII	Unspecified	
	2		IBJOB	IBJOB, IBLDR			3-17	Number of	LDRPRE	IBLDR *
	- <del>4</del> - /	Storage Map	IBJOB	IBJOB, IDEBK			0.,,	\$POOL cards	Larabatan i	Nervana i dei I
		flag	IBLDR	IBMAP			18-20		Unspecified	ARREST RESPONDENCE
		Harris and Artifect	Reset for no	100000			21-35	JBPCI	LDRPRE	IBLDR (this field
			load by IBMAP						etilli a mai	is the relative
			IBCBC		235					location of the
		gasification ray	IBFTC		鐵石		5.78	1 to	english in Agent Alle.	first \$POOL
	3	GO flag	IBJOB	IBJOB, IBMAP,				1 1 3 4	८९५ क्षिक्त छ।	chain entry in
	•	GO Hug	10000	IBLDR					es bloghod i	CISB)
		그 경우 원생 원생 경	Reset for	IDEDK		5	S	Indicator for	LDRPRE	IBLDR, DEBUG
		DUDAN BRANCH DIRECTOR	NOGO		- 1			\$NAME chain	gjen tij ten da de de eg	
		Carl Market	by IBMAP		- 1		1-20		Unspecified	ag da gañea
		er i de la companya d	IBFTC		- 1		21-35	JBNCI	LDRPRE	IBLDR, DEBUG
	100		IBCBC		- 1					(this field is the
		ng ng apagu	IBLDR							relative location
	4	LOGIC flag	IBJOB	IBJOB, IBLDR,				4.4	GOTAL FOREST	the first \$NAME
	4	LOGIC mag	18308	IBMAP	500.4	110	DAM L	200	early duote.	chain entry in
			IBLDR	IDMAI	1.0				and the	CISB)
		Contact to the	19071			. 6	S	Indicator for	LDRPRE	IBLDR, DEBUG
		1 45, 54, 24, 24 to	Reset for no		ا	Ĭ	•	\$USE, \$OMIT		
		COPOL A	load by IBMAP	IRIOR			1-20	,	Unspecified	
	5	COBOL flag	IBJOB	IBJOB IBLDR			21-35	JBUOI	LDRPRE	IBLDR, DEBUG
	6-7	IOCS flag	IBJOB IBJOB, IBLDR					7771 - 1 N		(this field is the
	8	FILES flag	277	IBLDR						relative location
	9	Load file flag	IBJOB	IBJOB, IBLDR					en et gleatier	of the first \$USE
	10-12	化二酚二醇原子	Reserved		1.5		1		1. 1. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.	or \$OMIT chain
	13	IEDIT flag	IBJOB	JOBIN, IBJOB						entry in CISB)
		and the same	IBLDR	IBLDR		7	S	Indicator for	LDRPRE	IBLDR
	14	SEARCH flag	IBJOB	JOBIN		1 .	•	\$LABEL	The state of the	
	15	OEDIT flag	IBJOB	JOBOU			1-20	APLIANT	Unspecified	a Balaka Kabupaten da
			IBLDR			110	21-35	JBLCI	LDRPRE	IBLDR, S.LABL
	16	COPY flag	IBJOB	IBLDR	. 1	9.7	- 0	Jule,		(this field is the
		COI I IIUg	Reset for no					49.5	6 464 354.	relative location
	2.34.0		load by IBMAP							of the first \$LABI
	17	Punch file open	IBSUP, JOBPP	IBSUP, PCLOSE	1 1					chain entry in
	18	RELOC flag	IBJOB	IBJOB, IBLDR					e best wit	CISB)
			IBLDR	IBJOB, IBLOK	1 1	8	s	Replace (+)	IBEDT	IBLDR
	19	Loader called	IBLUK	DJOB	1	ľ°		or insert (-)	IBLUI	IDLDK
		flag	10100	INION				on edit		
	20	Preprocessor	IBJOB	IBJOB		.1	1-2	on ean	Unspecified	
	21	Interrupt unit					3-17	Reservation	IBEDT	IBEDT
		for S.FBIN					3-17		וסבטו	IDEDI
	22	CHAIN flag	IBJOB	IBLDR		100		code for	역 그는 역사를	eritin jaron gengan da
	23	Jackson (1984)	Reserved				10.00	primary newlu	Unamadicad	
	24	IBEDIT flag	IBEDT	IBJOB, IBLDR			18-20 21-35	Reservation	Unspecified IBEDT	IBEDT
			Reset by				21-33	code for	IBEDI	IDEDI
		<ul> <li>DATEMAN</li> </ul>	IBEDT					secondary		
			IBCBC		1			newlu	선 되었습니다	reference of the contract of t
3.1			IBLDR		١.	۱,	S-35	newid	Unspecified	er Albert Bertie
			IBFTC			10	S-35		IBDMP1	IBDMP2
			IBMAP	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		11	S-35		Unspecified	IDDINI Z
	25	Object time files	LDRPRE	IBLDR	1	12		Match \$ Field	IBJOB	JOBIN
	26	Unload COPY	IBJOB	IBLDR		'2	3-33	Maich & Field	INCLOS	3000
		file				٠,٠	S-35	Match	IBJOB	JOBIN
	27	Output file open	JOBOU	JOBPP, JOBOU	1	13	3-33			JOHN
		Trade Artis	JOBPP		I	١.,		deckname	INCLOS	
	28	IBMAP T.O.C.	IBJOB	IBJOB	1	14			Unspecified	grani se trak ili
	100	to Nucleus				15			Unspecified	ชน์ . ปลอ โดยวัน
	29	IBFTC T.O.C. to	IBJOB	IBJOB		16		Mumber	Unspecified	IDENT <sup>3</sup>
	, the c	Nucleus	, while the		1	17	S-2	Number of	IBEDT	IBEDT1
	30	IBCBC T.O.C. to	IBJOB	IBJOB	1	1		OLDEO 3	(BEDY	IDENT
		Nucleus	19 TT TT 1		1	1	3-17	Address of	IBEDT	IBEDT1
	31	IBLDR T.O.C. to	IBJOB	IBJOB		1		OLDLU	1.18 2.14 Y St.	
		Nucleus		<del></del>	1		18-20	Number of	IBEDT	(BEDT)
	32	LIBEDIT	IBEDT	IBMAP		1	2.5	NEWLU's	Service Cons	Carrellada (1981)
	33	S.FBIA	IBJOB	JOBIN	1	1	21-35	Address of	IBEDT	IBEDT1
		opened with		300	1	1		NEWLU		
		openea with labels and				18		MAP flag	IBEDT	IBEDT2
		rewind				1	. 1	Discontinue	IBMAP	IBEDT
	0.4		INCLOS	JOBIN	1	1		edit flag	IBLDR	jahan sidili
	34	IEDIT Match S.SAVE to	INCLUS	JOHN	1	1 .	2		Unspecified	Legis D. Lore
٠.					1		3-17	Version number	IBEDT	IBEDT2
	9-	S.SFLG	IRIO	IBLDR	1		18-20		Unspecified	1815 Y (1)
	35	DLOGIC flag	IBJOB		1	1	21-35	Level number	IBEDT	IBEDT2
2	S	No separate	IBJOB	IBCBC		19			Unspecified	
		LOADUN	0.5.00		1		3-17	Core size	IBEDT	IBEDT2
	1-20		Unspecified	1 4 6	1			(if specified)	(58) (10) (57) (17) (43) (17) (17) (18) (18) (18) (18) (18) (18) (18) (18	
	21-35	Compiler/	IBCBC	IBMAP (contains	14.4		18-20		Unspecified	apatang nitrotoni.
	eri .	Map Unit	IBFTC	the S.SUNI ref-	1	1	21-35	Maximum block	IBEDT	IBEDT1, IBEDT2
	100			erence for the	1			size of NEWLU	a jedili kali a kode	ejskjestes ner
		보통된 함께서		internal text file)	1	20	S-2		Unspecified	Salar Joh
•		Indiantas fas	LDRPRE	IBLDR, DEBUG		1 -	3-17	Address of sec-	IBEDT	(BEDT1
3	\$	Indicator for	LURFKE	IDEDR, DEDUG		1		ondary oldlu		Casal Children
		CISB	lla-ma-te-u		1		18-20	7 y olaid	Unspecified	r se Te
	1-20		Unspecified LDRPRE	IBLDR, DEBUG	1	1	21-35	Address of sec-	IBEDT	JBEDT1
	21-35						41-33	Warings Of Sec.	1	

Note: Words 2-20 of S.SFLG are overlaid by IBLDR (in a GO situation) and Reload.

Figure 24. Communication Flags

The next two words, containing information regarding input/output operation, are set by 100P for use by a calling program. The publication IBM 7040/7044 Operating System (16/32K): Input/Output Control System, Form C28-6309, contains additional information.

SYMBOL CONTENTS DESCRIPTION S.SSCH pfx \*\*,, \*\* This word contains the results of the last operation on an input/output unit. The address portion contains 1 plus the last location of the data transmitted. The prefix and decrement contain flags indicating the status at the end of the operation. S.SSNS PZE \*\*,, \*\* This word contains the results of the last operation on an input/output unit other than telecommunications input devices.

The following words, containing information regarding input/output operations, are set by the Input/Output Executor (IOEX) for use by select plus and select minus routines.

SYMBOL CONTENTS DESCRIPTION S.XTDT PZE \*\*,, \*\* The decrement of S.XTDT contains trap condition indicators derived from the channel trap cell. The address contains 1 plus the location of the interrupted program. S.XSNS BCI 2, These words contain the sense data obtained by IOEX before each entry to a select plus routine or to a select minus routine for an operation that resulted in an unusual end condition. S.XLTP PZE S.XLTP contains the position of the device prior to the activity that ended in a trap. The address of S.XSCH S.XSCH PZE \*\*,, \*\* is equal to 1 plus the location of the last word that was read or written. The decrement contains the number of words remaining. S.XTPS pfx The sign of this word is set and interpreted by the Input/Output

Executor as follows:

+Trap is not in prog-

not inhibited.

-Trap is in progress; trapping is

inhibited.

S.XTPS is interrogated

by any routine that

ress; trapping is

SYMBOL CONTENTS

DESCRIPTION
must temporarily inhibit and restore channel traps.

The following words contain miscellaneous information:

SXCPS pfx

S.NAPT pfx ,, NTRPT

fida (sagra myselve) with

S.SFBL PZE a,, lm

 $\widetilde{PZE} \stackrel{d}{\rightarrow} , , l_n^{lm}$ 

The sign bit of this word is set and interpreted by the Checkpoint routine, as follows:

DESCRIPTION

+Checkpoint is not in progress.

-Checkpoint is in progress.

The prefix of S.NAPT indicates whether or not a snapshot has been taken as follows:

+No snapshot has been taken.

-Snapshots exist on S.SCK1.

The prefix of S.NAPT is set by the Snapshot routine for use by the System Monitor and the Dump program. It is reset by the Dump program and the Checkpoint routine.

NTRPT is a pointer to the interrupt unit used for input during the job interrupt pro-

cedure.

These words contain the following information pertaining to the file control blocks:

a: Location of the first word of the area occupied by contiguous file control blocks

lm: Total length of the file control blocks

d: Displacement of label information from the start of each file control block

In: Total length of each control block

This information is set by initialization routines for all subsystem components and by the Loader for object programs. It is used by the Checkpoint routine.

Johnam is one of the following:

1. Deck name from columns 8-13 of the

\$IBJOB card
2. Main name from columns 8-13 of the \$CHAIN card in a Chain application

3. Program name from the variable field of the \$RELOAD card

S.JNAM BCI 1,jobnam

5.JNAM BCI 1,Jobnan

tut kalleta u Va. Inga Tijitatik alif ya SYMBOL CONTENTS

DESCRIPTION

- 4. Phase name from columns 16-21 of the \$EXECUTE card

In items 1-3, jobnam is stored by the Processor Monitor for use by the Loader, the Reload Program, and the CHAIN subroutine.

In item 4, jobnam is stored by the System Monitor for use by the CHAIN subroutine.

These unit reference words are extracted from the unit references in the appropriate file control block immediately after any of these files are released; that is, closed without rewinding. The Processor Input and Output Editors maintain them for the alternate input, output, and punch files. The Loader and the postexecution routines maintain them for object programs.

The following words are used to save information about the Processor's files.

SYMBOL CONTENTS S.IAUN PZE \*\*,, \*\*

#### DESCRIPTION

S.IAUN is initialized by the Processor Monitor whenever a \$IEDIT card specifying an alternate input file is encountered.

S.OAUN is initialized by the Processor Monitor whenever a \$OEDIT card specifying an alternate output file is

encountered.

S.EDUN is initialized by the Editor Monitor to the proper unit to be used as the edit file. It is set by the Processor Monitor to the proper unit to be used as the copy or reload file.

The first word indicates the contents of the Symbolic Units Table entry for S.SIN1 when there is an alternate unit for an interrupt job.

The second word indicates the contents of the Symbolic Units Table entry for the interrupt unit used for input during the job interrupt procedure. The prefix bits of this word indicate the settings of the assembly parameters IFCHO, IFCHT, and IF162. The address field, CPRPU, indicates the setting of the system print/punch flag, which may be modified by using the \$OPEN and

\$CLOSE control cards.

The following words contain information necessary fo aydagabilgasi i ser

or symbol	lic channel assignment	Kyovie iš
SYMBOL	CONTENTS	ja å e <b>r</b>
S.SYCV	For any of the five sym-	These
S.SYCW	bols, the contents may	defining
S.SYCX	be:	bolie ch
S.SYCY	VFD 6/chan,	fields a
S.SYCZ	ETC 6/am1	System
Extig Prints	ETC 6/am2	a \$CH
1975/1915	ETC 6/am3	recogni
100000000000000000000000000000000000000	ETC 6/am4	cleared
	ETC 6/am5	card is
nvelgd v		Chan
		channel
	超级管理经验 為城市 人名英格兰	as follo
		00 = 1
。 激动感染物的		10=0

S.SYFS VFD 6/code6 or 0

6/code5 or 0

6/code4 or 0

6/code3 or 0

6/code2 or 0

6/code1 or 0

ETC

ETC

ETC

ETC

ETC

S.SDBG pfx dwu,, blksiz

Anno Mon Ballone (1997)

otie erder Lotus Adviso

DESCRIPTION words are the g words for symnannels V-Z. The are set by the Monitor when ANNEL card is ized, and they are when a \$JOB recognized. is the absolute el code in octal ws:

No channel

Channel A

20=Channel B 30=Channel C

40=Channel D 50 = Channel E

am1-am5 is symbolic module n and contains: 00-11 (octal) indicating arm1 module 0-9, 12-23 (octal) indicating arm2 module 0-9, 77 (octal) undefined.

Codes 1 through 6 represent the allowable disk and/or drum format types for selection as available units within an installation. These codes are defined by the use of the ALLOW macro-instruction. In the released version, code 1 is full track and codes 2 through 6 are not used. This word contains information used by the Debugging Processor, the Loader, the Processor Monitor, and the System Monitor. The sign bit is on (-) if debugging is requested. Bit I is on if there is output on the debug work unit. Bit 2 is on if IBDMP has terminated execution and called the postprocessor. In the address field, dwu is the location of the entry in the Sym-

bolic Units Table for

the debug work unit;

in the decrement field,

blksiz is the block size

for the debug work

unit.

S.OAUN PZE \*\*,, \*\*

S.EDUN PZE \*\*,, \*\*

\*\*\* \*\* \*\* \*\*\* \*\* \*\*\* \*\* S.SRUS

พรายใหม่ใหม่ใหม่พระที่ไม่ใช้สัมใหม่ พูลส์ ประพ

S.SPRP VFD 1/IFCHO, ETC 1/IFCHT, ETC 1/IF162, ETC 18/0, ETC 15/CPRPU

### Symbolic Units Table

Service of the Service

The Symbolic Units Table, which occupies the remaining system data area, describes the logical units available to the system. At an initial start, an entry for each logical unit available to the system is initial-

ized by the System Monitor housekeeping routine from the ATTACH macro-instructions used at system assembly time. A list of units not physically available to the system will be typed. The Supervisor maintains this table and modifies it when processing a sswitch, SATTACH, SDETACH, or SRESTORE card. The table is fixed in length up to s.suo2.

The Symbolic Units Table is referred to by any part of the system that uses an input/output unit in processing, by the calling sequence to s.100P, and by all file control blocks.

The content of the Symbolic Units Table is:

	1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1	
SYMBOL	OPC VAR FIELD	SYSTEM UNIT
S.SLB1	***(Grade and *); (**) decide	LIBRARY 1
S.SLB2	***	LIBRARY 2
S.SIN1	***	INPUT 1
S.SIN2	*** **	INPUT 2
S.SOU1	*** **	OUTPUT 1
S.SOU2	*** ** **	OUTPUT 2
S.SPP1	*** ** **	PUNCH 1
S.SPP2	****** A TO A TO A TO A TO A TO A TO A T	PUNCH 2
S.SCK1	****	CHECKPOINT 1
S.SU00	***	UTILITY 0
S.SU01	****	UTILITY 1
S.SU02	***: 150 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	UTILITY 2
•		skiner energy being
•		
•	Campan to paragraph of the site of the	
S.SU99	***	UTILITY 99

Each entry in the table requires one word; the number of entries is determined by the ATTACH macroinstructions at system assembly time. The prefix bits of each word are interpreted as follows:

- Bit 0 = 0Unit may or may not be unloaded after being rewound.
  - Unit must be unloaded after being rewound. This bit should be set for S.SIN1, S.SIN2, S.SOU1, S.SOU2, S.SPP1, and S.SPP2 for data protection.
- Bit 1 = 0Unit is not in use by current program.
  - Unit is in use by current program. This bit should be set by any object program that does not use the IOBS level of IOCS, so that checkpoint and restart procedures can be performed. Not used.

The address and decrement portion of these words are:

Bit 2

The location of the unit control block for the device assigned to this logical unit Decrement

The location of the system control block for the part of the device assigned to this logical unit

### **Control Blocks**

The unit control blocks and the system control blocks, which occupy the next section of the Nucleus, are described in the publication IBM 7040/7044 Operating System (16/32K): Input/Output Control System, Form C28-6309. 

#### Other Tables

Two other tables are included in the final portion of the Nucleus along with the Nucleus routines that use them.

Abbreviated Table of Contents: The Abbreviated Table of Contents, IBLTB, consists of three-word entries identical to the Table of Contents entries for the subsystem in control. Since it contains the name and position in the library of each phase of a subsystem, it allows the s.sret routine to use the System Loader to call subsystem parts without reloading the Supervisor and the subsystem monitors. The Abbreviated Table of Contents, also referred to as a phase dictionary, is set up by all monitors through the IMPAS routine.

An entry in the Abbreviated Table of Contents has the same format as an entry in the Table of Contents (see the section "Table of Contents").

Recognizable Control Card Table: The Recognizable Control Card Table, IBRTB, is a list of control cards that can be recognized by the s.sret routine and used by the System Loader to call a specified subsystem component. The Recognizable Control Card List contains two words per entry in the following format:

SYMBOL CONTENTS IBRTB BCI 1, \$lgcrd PZE p

DESCRIPTION The first word contains a recognizable control card name, \$lgred, from columns 1-6 of any subsystem control card. p is the location of the Abbreviated Table of Contents entry for the subsystem component associated with the preceding control card name.

### **Nucleus Routines**

The following routines remain in storage at all times to maintain the continuity of the system:

- 1. System Loader (s. sldr)
- 2. Interrupt Test (s.srpt)
- 3. Dump routine (s.spmp)
- 4. System Monitor Recall routine (s.srup)
- 5. Return routine (s.srer)
- 6. Restart routine (s.srst)
- 7. Change Communication Region routine (s.sccr)

In addition, the Installation Accounting Routine (s.SIDR) might be added to the Nucleus to remain in core storage at all times. System Loader

The System Loader is an absolute program loader that positions the proper system library unit for loading, loads a system phase, and verifies that phase using information supplied to it from the Abbreviated Table of Contents. It performs the following functions:

- 1. Pre-positions the device specified in the Abbreviated Table of Contents, loads a phase from the device, and verifies the accuracy of the positioning by comparing the phase name to the first word of the Abbreviated Table of Contents entry.
  - 2. Initiates post-positioning of the device.

## 3. Transfers control to the phase just loaded.

Functions 1, 2, and 3 are optional, as specified in the calling sequence to the System Loader, which is:

TSX S.SLDR,4 pfx ptr

where the prefix codes have the following meanings: Sign Bit

1–Do not post-position.

0-Post-position.

1-Do not load.

0-Load.

Bit 2

nt 2
1—Do not transfer control.
0—Transfer control.

and ptr has the following interpretation:

= 0: Use the pointer as is to reference the Abbreviated Table of Contents in the established sequence.

≠ 0: Readjust the reference to the Abbreviated Table of Contents to location ptr.

Note: Calling s.SLDR with a prefix code of PTH will position the device in front of the phase referred to

The System Loader uses the following subroutines:

Severiteral Existing

1. s.100P

2. S.XPRT

3. S.XPSE

Two error conditions are possible. In the event of a permanent error when reading, a code 20529 error message is typed and processing is suspended. If the phase name cannot be verified, a code 20527 error message is typed and processing is suspended. For both cases, pressing START causes the device to be rewound and the loading to be restarted.

### Operator Interrupt Test

The machine operator can interrupt the flow of processing to process a priority job by setting the entry keys and then executing the operator interrupt procedure. To test for an operator interrupt request, any program may, at a convenient point, include the following instructions:

XEC

This instruction is executed if there is no interrupt request.

This instruction is executed if there is an interrupt request.

\*\*\* is any operation code. Index register 4 may be altered by this procedure. If an interrupt request exists, the programmer should complete all operations currently in progress, take a checkpoint if applicable, and transfer control to s.srup.

The instruction in location s.srpr, in the distributed version, is:

S.SRPT SWT • RUPSW

The parameter RUPSW in the distributed version is: RUPSW EQU 1

By modifying RUPSW, the installation may assign the test to any sense switch. By modifying s.SRPT, the installation may modify the interrupt test procedure. Since it is specified that index register 4 may be altered by this procedure, the installation is free to assemble

routn.4 TSX S.SRPT

where routn is the entry point to an interrupt analysis routine that the installation has included in the Nucleus.

### **Dump Routine**

The Dump routine uses 100P to write an area of core storage on the system checkpoint unit, if that unit is attached. It then calls the System Loader to load the Dump program into that area. The calling sequence to the Dump routine is:

SXA \*+3, 4 (or the equivalent) TSX S.SDMP, 4 pfx return, t, errno **PZE** 

where return, t is the location to which the Dump program will transfer control after dumping. If the effective address (return, t) is zero, control is returned to the Supervisor. The symbol errno is the five-digit error number. The prefix pfx is interpreted as follows:

Sign Bit = 1 Pause before returning Bit 1 = 1 Dump system panel

Bit 2 = 1 Traceback

The error message 20298 is typed if s.sck1 cannot be written without error.

### System Monitor Recall Routine

This routine uses the System Loader to bring the Supervisor into core storage. It is called with a TRA s.srup instruction.

#### **Return Routine**

This routine examines columns 1-6 of the next control card that is in location s.save. If the card is recognized, the System Loader loads the required subsystem component and passes control to it. If the next card has not been saved, or if it is not recognized, the System Loader loads the Supervisor record containing the combined monitors.

s.sret uses the contents of s.save and s.scot to obtain the next control card and calls the System Loader to bring in the proper phase.

#### **Restart Routine**

This routine reads in the record of core storage from the checkpoint device and transfers control to the program being restarted. The Restart routine is called with a TRA S.SRST instruction.

As input, the Restart routine requires a work area describing the checkpoint device, the core storage to be restored, and the return point to the program for which the checkpoint was taken.

The Restart routine uses 100P to read from the checkpoint device, under the assumption that it is properly positioned.

The error message 20298 is typed if the checkpoint device cannot be read without error.

### Installation Accounting Routine

This routine is defined by each installation. It has the following calling sequence:

TSX S.SIDR,4 PZE loc,, n

where loc has the following meanings:

≠ 0: loc is the location of the \$ID, \$JOB, or \$STOP card.

at america to give

= 0: No \$ID, \$JOB, or \$STOP card exists.

and n has the following meanings if loc = 0:

= 0: Entry is from the Supervisor.

1: Entry is from IBMAP.

2: Entry is from IBFTC.

3: Entry is from IBCBC.

4: Entry is from the beginning of IBLDR.

5: Entry is from the end of IBLDR.

6: Entry is from S.JXIT.

9: Entry is from IBSRT.

12: Entry is from the beginning of RELOAD.

13: Entry is from the end of RELOAD.

One transfer to the accounting routine is made for the first sjob card that is encountered after initial start; two transfers are made for each subsequent sjob card.

In the distributed version of the system, location s.SIDR is a TRA 2,4 instruction which should be changed when the installation defines its accounting routine.

### **Change Communication Region Routine**

The Change Communication Region routine permits the programmer to release storage protection, execute one instruction that changes the contents of the Nucleus, and restore storage protection. The calling sequence is:

TSX S.SCCR, 4
\*\*\* Instruction that affects the Nucleus
Return

### Supervisor

The Supervisor routines process the System Monitor control cards. They may be called by any subsystem that an installation assembles into the combined monitors.

The functions of the Supervisor are performed by the following:

1. IMMON Routine

4. IMSRT Routine

2. IMAST Routine

5. IMEXE Routine

3. IMSYS Routine

6. INJOB Routine

7. IMRCD Routine 21. IMLIS Routine 8. IMSCN Routine 22. IMUNL Routine 23. IMOID Routine 9. IMPAS Routine 10. IMUQI Routine 24. IMPAU Routine 11. IMINT Routine 25. IMSTO Routine 12. IMLSU Routine 26. IMTIS Routine 27. IMTIM Routine 13. IMDLY Routine 14. IMSWI Routine 28. IORST Routine 15. IMATT Routine 29. IQRSC Routine 16. IMDET Routine 30. IMUNI Routine 17. IMCLO Routine 31. IMCHA Routine 18. IMRES Routine 32. SCAN Routine 19. IMTIF Routine 33. IMBDC Routine 20. IMPUN Routine 34. IMOPN Routine

### IMMON Routine

Saltuck Mikist

Purpose: The main flow of control through the Supervisor begins with this routine. The routine checks to see if IBJOB is in control and, if it is not, releases all units with 70 and 62 reservation codes. The routine also examines the decrement of location s.scor to determine the reason that the Supervisor was entered. Processing is initialized if this is an initial start.

### **IMAST Routine**

Purpose: To type the contents of a \*\* card on-line. Calling Sequence:

TRA

The Supervisor enters this routine after recognizing a \*\* card.

Input: s\* card.

Output: None.

Subroutine Used: IMTIF

Error Conditions: None.

#### **IMSYS** Routine

Purpose: To insert the name of the subsystem currently in control into location s.scun and to set s.sswi.

Calling Sequence:

TRA IMSYS

The Supervisor enters this routine after recognizing a SIBSYS card.

Input: sibsys card.

Output: None.

Subroutine Used: s.sccr

Error Conditions: None.

### **IMSRT** Routine

Purpose: To process the SIBSRT card. This routine places IBSRT into location s.scur, sets s.sswi and transfers control to the IMEXE routine.

Calling Sequence:

TRA\* IMSRT

The Supervisor enters this routine after recognizing a SIBSRT card.

Input: sibsrt card.

Output: None.

Subroutine Used: s.sccr

Error Conditions: None

### **IMEXE** Routine

Purpose: To process a sexecute card, which provides convenient access to a program in the System Library.

adiša bita beso Portend najbus

This routine scans the Index for the program name starting in column 16 of the SEXECUTE card. If this name is found, the library-unit position(s) and the load address(es) of the program are placed in the Abbreviated Table of Contents in the Nucleus, In addition, the program name is stored in s.JNAM and control is passed to the System Loader. If the name is not found, a code 10515 error message is typed.

Calling Sequence:

TRA

**IMEXE** 

The Supervisor enters this routine after recognizing a SEXECUTE card. XECUTE CAICI.

Input: SEXECUTE card.

Output: None

Output: None.

Subroutines Used: s.sldr, s.sccr, imscn, and s.xprt. Error Conditions: The name of the program to be executed is not in the Table of Contents. The SEXECUTE card is ignored and the next control card is read.

### **INJOB** Routine

Purpose: To process a sjob card. All intersystem reservation codes, system work unit codes, and object program unit codes in the reservation fields of the system control blocks are set to zero, columns 16-45 of the sjob card are saved in s.Shdr, the installation accounting routine is called, the interval timer is reset to the value assembled as MXCLK in IBNUC, and the control card is typed. Word 1 of s.sflc is reinitialized except for the punch file open and output file open bits. Entries to special input/output routines are cleared if those entries pointed to locations within s.scon limits. All 100P special routine entries that fall between IBORG and s.send will be deleted.

Calling Sequence:

INIOB TRA

The Supervisor enters this routine after recognizing a sjob card.

Input: s10B card.

Output: None.

Subroutines Used: s.sccr, s.sidr, imtis, and imtif. Error Conditions: None.

#### **IMRCD** Routine

Purpose: To move the next card into a fixed area and initialize the IMSCN routine. The IMRCD routine reads a control card and checks it for a \$ in column 1.

Calling Sequence:

IMRCD, 4 TSX Return 1 Return 2

Return 1 is taken if the next card is a binary card or if it is not a \$ control card. Return 2 is taken if the next card is a \$ control card.

Input: None.

Output: None.

Subroutine Used: JOBIN Error Conditions: None.

#### **IMSCN** Routine

Purpose: To pick up twelve characters at a time indeping från dan eft kvægstende fed Græss end fill formstold frånde dann from a control card.

Calling Sequence:

TSX

IMSCN, 4 ในสูตราสที่เกาไว้สินในสายคุณ เพลิโด เมื่อกเพียงกลับก

Input: None.

Output: Up to twelve low-order BCD characters in the accumulator and multiplier-quotient register. If there are fewer than twelve characters, the high-order positions are filled with leading zeros.

Subroutines Used: None. Error Conditions: None.

### IMPAS Routine

Purpose: To pass a portion of the Table of Contents to the Nucleus and construct a table of recognizable - Principal and Statistics of Carlos (A) control cards.

Calling Sequence:

IMPAS, 4 TSX pfx a,,b BCI 1, c

where the prefix, pfx, is plus for a new Table of Contents and minus for an addition to the existing Table of Contents. The letter a is the length of the portion of the Table of Contents, b is the location of the first word in the portion, and c is the contents of columns 1-6 of the control card.

Input: None.

Output: Abbreviated Table of Contents.

Subroutine Used: S.SCCR

Error Conditions: Table of Contents or control card list overflow.

# kida Papalipa dali skolovitali ilikula inik kin tudina vi

IMUQI Routine Purpose: To compare the utility units portion of the Symbolic Units Table with the entire Symbolic Units Table, checking for uniqueness. If one of the two symbolic units composed is s.sck1, this is not considered a non-unique entry.

Calling Sequence:

IMUQI, 4 TSX

Return if each entry is unique. Return if any entry is non-unique.

Input: None.

Output: Address of lowest non-unique entry in the accumulator if any is non-unique.

Subroutines Used: None. Error Conditions: None.

### **IMINT** Routine

Purpose: To change unit assignments and/or process a special job when the operator interrupt procedure is followed. The entry keys are examined for the type of action to be taken.

Calling Sequence:

TSL IMINT

Input: Entry keys.

Output: None.

Subroutines Used: s.xprt and s.xpse.

Error Conditions: Unable to determine request.

### **IMLSU** Routine

Purpose: To validate a symbolic unit name. Calling Sequence:

TSX

IMLSU, 4

Return if the symbolic unit name is invalid.

Return if the symbolic name is valid.

Input: The symbolic unit name in the MQ.

Output: The relative location in the Symbolic Unit Table, in index register 2.

Subroutines Used: None. Error Conditions: None.

### IMDLY Routine

Purpose: To delay until all input/output activity is completed.

Calling Sequence:

TSL IMDLY

Input: None. Output: None.

Subroutine Used: S.100P

Error Conditions: Since no error return is provided, an 100P error results in a dump.

### **IMSWI** Routine

Purpose: To process a sswitch card, which interchanges the devices assigned to the symbolic units that are either specified or indicated by intersystem reservation codes.

Calling Sequence:

TRA IMSWI

The Supervisor enters this routine after a sswitch card is recognized.

Input: \$SWITCH card.

Output: None

Subroutines Used: IMSCN, S.XPSE, IMDLY, S.XPRT, and S.SCHI.

Error Conditions: Two error conditions apply:

- 1. An attempt to switch s.sin1, s.sou1, or s.slb1 with a symbolic unit that has no device attached results in an error message and a pause. The card is ignored.
- 2. Iyy specified was illegal or not defined. The card is ignored.

### **IMATT** Routine

Purpose: To attach a device to the system and assign it to the symbolic unit specified. This routine sets an attachment indicator in the unit control block and places the location of the unit and system control blocks in the Symbolic Units Table.

Calling Sequence:

TRA IMATT

The Supervisor enters this routine after recognizing a \$ATTACH card.

Input: sattach card.

Output: None. Subroutines Used: IMSCN, S.XPSE, IMDLY, and S.XPRT. Error Conditions: The following error conditions apply:

1. Attempt to attach an attached unit.

- 2. Attempt to attach a device not defined at assembly time.
- 3. Attempt to attach more symbolic units to a device than were attached at assembly time.
- 4. Attempt to attach a device not physically avail-
- 5. Attempt to assign a reservation code other than 57 or 58 to the device.

These conditions result in an error message and a pause. The card is ignored.

### IMDET Routine

Purpose: To detach a device and its unit. Calling Sequence:

> TRA IMDET

The Supervisor enters this routine after recognizing a SDETACH card.

Input: SDETACH card.

Output: None.

Subroutines Used: IMSCN, IMDLY, S.XPSE, and S.XPRT. Error Conditions: Two error conditions apply:

- 1. Attempt to detach a device for which there is no unit control block will result in an error message and the card will be ignored.
- 2. Attempt to detach a device reserved by the programmer or an attempt to detach a detached device results in an error message and a pause. The card is ignored.

### **IMCLO** Routine

Purpose: One of the following, depending upon the unit specification on the sclose card:

- 1. If s.spp1 is specified, the combined print/punch status bit is set to combined. If either MARK, REMOVE, or REWIND is specified, the punch unit is closed accordingly and the punch-file open bit is set to off.
- 2. If s.suxx is specified, the unit is released, the associated reservation code is reset to zero, and the other options are processed.
- 3. If IyyR is specified, the unit to which the intersystem reservation code yy is appended is treated as in point 2 above.

Calling Sequence:

TRA IMCLO

The Supervisor enters this routine after recognizing a \*CLOSE card. Sand Baril Representati

Input: sclose card.

Output: None.

Subroutines Used: IMSCN, S.XPRT, S.XPSE, S.IOOP, IMDLY, and s.schi.

Error Conditions: The following error conditions apply:

TO SECURE ALEXANDER

- 1. An attempt to close s.sin1, s.sou1, or s.slb1 results in an error message and a pause. The control card is ignored.
  - 2. Iyy is undefined or illegal. The card is ignored.

### IMRES Routine

Purpose: To restore the unit assignment scheme to its initial conditions. All intersystem reservation codes are released and a list of all units not physically avail-Touris alderenis to be historical a section able is typed. Calling Sequence:

TRA

**IMRES** 

The Supervisor enters this routine after recognizing a \*RESTORE card.

Input: srestore card.

Output: None.

Subroutines Used: S.SLDR, S.IOOP, S.XPRT, and S.XPSE.

Error Conditions: Unable to restore results in an error message. The card is ignored.

### IMTIF Routine

Purpose: To type the control card unless it is already typed.

Calling Sequence: TSL

Input: Control card in IMCRD.

Output: None.

Subroutine Used: S.XPRT.

Error Conditions: None Error Conditions: None.

## IMPUN Routine is a standard standard or of the standard

Purpose: To type a card containing an invalid field and an error message.

Calling Sequence:

TRA

Input: Control card in IMCRD.

Output: None.

Subroutines Used: s.xprt and imtif.

Error Conditions: None. Purpose: To set the list switch so that all \$ control cards, processed by the combined monitors, are typed. Calling Sequence:

TRA

The Supervisor enters this routine after recognizing a \*LIST card. Input: \$LIST card.

Output: s.scor set plus.

Subroutine Used: IMTIF.

Error Conditions: None.

IMUNL Routine

Purnose: To turn off skeller Purpose: To turn off the list switch.

Calling Sequence:

TRA

IMUNL

The Supervisor enters this routine after recognizing a sunlist card. Input: sunlist card.

Output: s.scot set minus.

Subroutine Used: None.

Error Conditions: None.

#### IMOID Routine

Purpose: To enter the installation accounting routine,

Calling Sequence:

TRA

The Supervisor enters this routine after recognizing a sid card.

Input: sid card.

Output: None.

Subroutines Used: IMTIF and S.SIDR.

Error Conditions: None.

#### **IMPAU** Routine

Purpose: To pause or suspend machine processing. Calling Sequence:

> TRA IMPAU

The Supervisor enters this routine after recognizing a \*PAUSE card. Input: SPAUSE card.

Output: None. .

Subroutines Used, IMTIF and S.XPSE.

Error Conditions: None.

### **IMSTO** Routine

Purpose: To terminate an interrupt job or the series of jobs on s.sin1. The installation accounting routine is called and the interval timer is reset to zeros, and all intersystem reservation codes are released.

Calling Sequence:

TRA

IMSTO

The Supervisor enters this routine after recognizing a \$STOP card. Input: \$STOP card.

Output: None.

Subroutines Used: IMTIF, S.XPRT, S.XPSE, S.XDVA, S.XDVD, IMRES (IMRES is not used in an interrupt job), s.SIDR, and IMTIS.

Error Conditions: None.

### **IMTIS** Routine

Purpose: To set the interval timer prior to entering an object program.

Calling Sequence:

TSL IMTIS

Input: Length of job in IMTIC.

Output: Interval timer (IBCLK) is set.

Subroutine Used: s.sccr.

Error Conditions: None

#### **IMTIM Routine**

Purpose: Sets the interval timer to the time specified on the STIME card.

Calling Sequence:

TRA

**IMTIM** 

The Supervisor enters this routine after recognizing a STIME card.

Input: STIME card.

Output: Length of job in IMTIC. Subroutines Used: IMSCN and IMTIF.

Error Conditions: None.

### **IQRST** Routine

Purpose: To accomplish a restart.

Calling Sequence:

TRA

**IQRST** 

Input: Key setting. Output: None.

Subroutines Used: IMSCN and IMINT.

Error Conditions:

1. Specified checkpoint device not attached.

2. Specified checkpoint record cannot be located. An error message is typed and a halt for operator action occurs.

#### **IQRSC** Routine

Purpose: To accomplish a restart.

Calling Sequence:

TRA

IQRSC

Input: srestart card.

Output: None.

Subroutines Used: See "IQRST Routine" above. Error Conditions: See "IQRST Routine" above.

### **IMUNI** Routine

*Purpose*: To type the current unit assignments and reservation codes.

Calling Sequence:

TRA

**IMUNI** 

The Supervisor enters this routine after recognizing a sunits card.

Input: sunits card.

Output: The typewritten paragraph.

Subroutine Used: S.XPRT. Error Conditions: None.

### **IMCHA** Routine

Purpose: To set up equivalences between the real channels and the symbolic channels specified in the schannel card. Also, to set up equivalences between real disk access mechanisms (or drum modules) and any symbolic access mechanisms specified. See Appendix D for a detailed description of the IMCHA routine.

Calling Sequence:

TRA

**IMCHA** 

The Supervisor enters this routine when a \*CHANNEL card is recognized.

Input: schannel card.

Output: None.

Subroutines Used: s.sccr, s.xprt, imtif, imscn, scan, and imbdc.

Error Conditions: Any irregularities encountered while the schannel card is being processed cause the following to be done:

1. The \*CHANNEL card is typed.

2. A message is typed:

10502 \$CHANNEL CARD IN ERROR

- 3. A message is written on the system output unit: SKIPPING TO NEXT JOB
- 4. If the requirements specified by the \*CHANNEL card cannot be fulfilled, the contents of the \*CHANNEL card are typed and a second message is written:

10518 \$CHANNEL REQUIREMENTS CANNOT BE SATISFIED

#### **SCAN Routine**

Purpose: To scan one or more characters of IMCRD (fixed area)

Calling Sequence:

AXT n,4
TSL SCAN
TRA Return 1
TRA Return 2

n indicates the number of characters (1-6).

Return 1 is for a valid control card.

Return 2 is for an invalid control card.

Input: None.

Output: The n characters scanned, in the accumulator, right-justified, high-order zeros.

Subroutine Used: IMRCD. Error Conditions: None.

#### **IMBDC** Routine

Purpose: To convert BCD to binary.

Calling Sequence:

TSL IMBDC

Input: BCD word in accumulator.

Output: Converted binary data in accumulator.

Subroutines Used: None. Error Conditions: None.

### **IMOPN** Routine

Purpose: To process the sopen card as follows:

- 1. If the card specifies s.spp1, the combined print/punch status bit is set to uncombined.
- 2. If the card specifies S.SUxx=Iyy, reservation code yy is placed in the reserved status field of the system control block for S.SUxx.

- 3. If the card specifies T = Iyy, D = Iyy, or U = Iyy, reservation code yy is placed in the reserved status field of the system control block for an available tape unit (T), an available disk or drum unit (D), or any available unit (U).
  - 4. If REWIND is specified, the unit will be rewound. Calling Sequence:

TRA

**IMOPN** 

Input: sopen card.

Output: None.

Subroutines Used: IMSCN, S.SCCR, S.SCHI, and SCAN.

Error Conditions:

1. S.SUxx was already reserved.

2. Iyy is the reservation code for another unit.

3. Illegal variable field.

4. No units of the type requested were available.

All of the error conditions listed above cause the sopen card to be ignored and a message to be written.

### Input/Output Control System

The system requires that the following levels of the Input/Output Control System (see Figure 25, which appears later in the text) remain in core storage:

- 1. IOEX.
- 2. 100P synchronizer.
- 3. 100P1 select and error recovery routines.

The subsystem requirements of the Input/Output Control System are:

- 1. IOLS and IOBS are required by the Supervisor, the Processor, and the Update Program.
- 2. IOLS is required by Sort, Dump, the System Editor, and the monitored utility programs.

IOOP2, IOLS, and IOBS are optionally available to object programs processed by the Processor.

### **Dump Program**

The IBM 7040/7044 Operating System contains a Dump program that provides error messages, error dumps, and debugging dumps during operation of the system. The Dump program uses the Input/Output Operations (IOOP) level of the Input/Output Control System (IOCS). Error returns for all conditions are provided. The Dump program also uses the system Output Editor for writing on the system output unit.

The Dump program is discussed fully in the publication IBM 7040/7044 Operating System (16/32K): Debugging Facilities, Form C28-6803.

igh orage	Blank COMMON	Ť	able of Availa	ble Work Uni	ts		Modification Programs	Table of Available Work Units			
orago	Debugging Subroutines						Common Parameter Area				
	Object Program	COBOL Compiler	FORTRAN IV Compiler	Loader	Macro Assembly Program	Dump	Sort	Editor	Combined Monitors	Debugging Preprocessor	Debugging Postprocesso
							DMPYEX				
			l 10	BS	inguis.		Sort Monitor		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	IOBS	and and and
			10	LS							
			IO	OP2					io en 144. 12 - 1114.		(1981) 188 <u>(1881) - 1888</u>
		ID CARD PROGRAM, IF ANY					jakî tekstine. <u>Nave li Alfi</u>				
ow orage				NUC DEX AND 10	OP1						

Figure 25. Storage Allocation for All Operating System Components

This section contains detailed information on the Processor components for the systems programmer.

### **Processor Monitor**

The Processor Monitor, one of the combined monitors, controls Processor applications.

#### Operation

The Supervisor transfers control to the Processor Monitor when a sibjob card is read. The Processor Monitor has the following functions:

- 1. It decodes the sівјов card and saves this information.
- 2. It sets up the input/output unit configuration and ensures availability of the units.
- 3. It creates the phase dictionary from the Table of Contents, using the SIBJOB card parameters.
- 4. It creates a list of control cards that are valid for this Processor application.
- 5. It calls the Preprocessor (LDRPRE), which interprets the Preprocessor cards, converts the information

to binary coding, and writes the information on a utility unit. The Preprocessor is called into core storage only one time for each Processor application.

- 6. It processes the \*IBFTC, \*IBCBC, \*IBMAP, \*IBLDR, and \*RELOAD cards. That is, it processes the first such card in a Processor application and processes subsequent cards when they are not recognized by s.sret.
- 7. It transfers to the s.sret routine, which examines columns 1-6 of the card in s.save. If the card is valid for this application, s.sldr loads the system part required and passes control to it. The s.sret routine is in core storage at all times and if, subsequently, s.sret examines a card in s.save that is invalid for this Processor application, it calls in the combined monitors and transfers control to the Supervisor.

Figure 26 shows the Processor Monitor operation.

#### **Routines**

The following is a list of Processor Monitor routines that indicates the function and calling sequence for each routine:

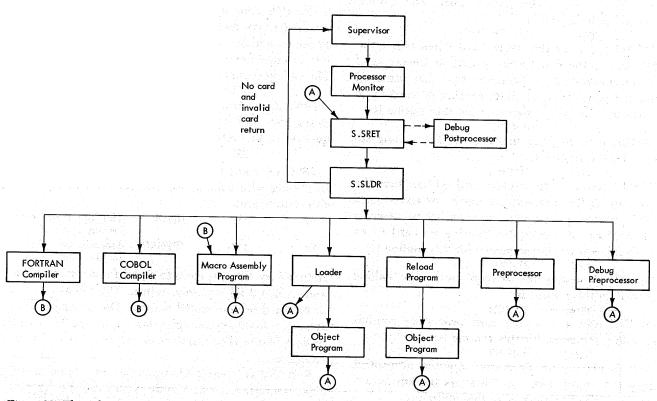


Figure 26. Flow of Control During a Processor Application

AVLUNI tests the availability of units for work1, work2, and work3 and places the Symbolic Units Table addresses of the units into the compiler units table. The calling sequence for this routine is:

TSL AVLUNI

CBAVL tests for the availability of the extra COBOL unit WORK4. If the unit is not available, the load file is used as the unit for WORK4. The Symbolic Units Table address of the unit is placed in the compiler units table. The calling sequence for this routine is:

ΓSL CBAVL

FTCENT obtains the address of the Table of Contents entry for a Processor section. Prior to executing the calling sequence, the accumulator must be set to BCI 1, XXXXXX, where XXXXXX is the appropriate control card name. The calling sequence for this routine is:

TSL FTCENT

The routine returns with the accumulator set to PZE A1,,L1, where A1 is the control address of the Table of Contents entry for the specified control card, and L1 is three times the number of phases in the Table of Contents entry.

GERP writes error messages. The calling sequence for this routine is:

TSX GERP, 4 PZE A, m, n

where:

n = the number of parameter words at A.

= the error severity code.

IMCHN processes the schain card. The calling sequence for this routine is:

TRA IMCHN

IMCBC processes the sibcbc card when either the sibcbc card is the first sibcbc, sibftc, or sibmap card after the sibjob card, or when an error has occurred in the first phase of a compiler, causing no card to be read ahead. The calling sequence for this routine is:

TRA IMCBC

IMDBL processes the \$IBDBL card. The calling sequence for this routine is:

TRA IMDBL

IMFTC processes the SIBFTC card when either the SIBFTC card is the first SIBMAP, SIBFTC, or SIBCBC card after the SIBJOB card, or when an error has occurred in the first phase of a compiler, causing no card to be read ahead. The calling sequence for this routine is:

TRA IMFTC

IMECH processes the SENDCH card. The calling sequence for this routine is:

rra imech

IMENT calls the SENTRY card processing routine. The calling sequence for this routine is:

TRA IMENT

IMJOB places the Processor Monitor in control, processes a sibjob card, and initializes the job parameters. The calling sequence for this routine is:

TRA IMJOB

IMLDR processes the SIBLDR card. The calling sequence for this routine is:

TRA IMLDR

IMLNK processes the SLINK card. The calling sequence for this routine is:

ΓRA IMLNK

IMMAP processes the SIBMAP card when either the SIBMAP card is the first SIBMAP, SIBFTC, or SIBCBC card after the SIBJOB card, or when an error has occurred in the first phase of a compiler, causing no card to be read ahead. The calling sequence for this routine is:

TRA IMMAP

IMPRE calls the Preprocessor, which is a separate phase, using the System Loader. The calling sequence for this routine is:

TRA IMPRE

IMREL processes the SIBREL card. The calling sequence for this routine is:

TRA IMREL

IMRLD calls the RELOAD program. The calling sequence for this routine is:

TRA IMRLD

INCLOS closes the system input file or its alternate without rewinding or label checking. The calling sequence for this routine is:

CALL INCLOS

*INCLSM* is used by the monitor to close the system input file without rewinding or label checking. The calling sequence for this routine is:

CALL INCLSM

INSAVE saves a card from the system input file in the area s.save and closes the system input file or its alternate without rewinding or label checking. The system input file must be open before insave is called. The calling sequence for this routine is:

CALL INSAVE

JBKILL prints a message that the job is terminated, closes files, and returns to the caller. The calling sequence for this routine is:

TSL JBKILL

JBSCAN scans the variable field of a control card, beginning with column 16. The calling sequences for this routine are:

X<sub>1</sub> TSL JBSCI (for the initial entry for any one card).
X<sub>2</sub> TSL JBSCAN (for subsequent entries for the same card).

Return from this routine is to  $X_n+1$  for an error return, and to  $X_n+2$  for a normal return. The initial location of the card to be scanned is CARD. The collected characters are in CHARS through CHARS+2, left-justified with trailing blanks. The terminating character is in the accumulator.

JOBERR is entered when a utility unit is not available; the return is made to the s.sret routine. The interrupt switch will be tested when the combined

monitors are loaded. The calling sequence for this routine is:

TSX JOBERR, 4
PZE \*\*

JOBIN is used for input from the system input unit or its alternate by all Processor components except the monitor. The calling sequence for this routine is:

CALL JOBIN

A logical record is located in the input file buffer. If the record is acceptable, the routine returns with the accumulator set, as follows:

pfx la,, r

where:

= PZE if the record is BCD and the control character in the logical control word was 4 or 5.

= PON if the record is BCD with a \$ in column 1 and the control character in the logical control word was 4 or 5.

= MZE if the record is binary and the control character in the logical control word was 6 or 7.

la

= the initial address of the record in the buffer.

= the number of words located: 14 if the words are BCD; 27 (from unit record equipment) or 28 (from nonunit record equipment) if the words are binary.

The following is a list of conditions that cause the record to be unacceptable:

- A redundancy on the system input file or its alternate.
  - 2. A file mark on the system input file.
- 3. An unsuccessful search for a matching \$ card on the alternate input file.

JOBINM is used by the monitor to open the system input unit or its alternate on the first entrance or to get a single logical record of input from the system input file. The calling sequence for this routine is:

CALL JOBINM

The routine returns with the accumulator set as indicated previously under "jobin."

JOBOU is used for output to the system output unit or its alternate by all Processor components except the monitor. The calling sequences for this routine are:

> CALL JOBOU(list) CALL JOBOUL(list)

where list is the initial address of a set of parameters that have the following form:

PZE n n Al, Tl, Ml

 $\begin{array}{ccc}
\cdot \\
\text{pfx} & A_n, T_n, M_n
\end{array}$ 

where:

= the number of words in the list following the first word. pfx

PZE if a new line is to be started after a single space.
PON if a new line is to be started after a skip to a new

PON it a new line is to be started after a skip to a new page.

PTW if a new line is to be started after a double space.
PTH if a new line is to be started after a space that is determined by the first character of the output line.

= MZE if the current line is to be continued.

NOTE: When the preceding prefixes are used, line over-flow is ignored. When PTH is used, the user has assumed responsibility for spacing, page overflow, and page count. The user can maintain the line count by decrementing the counter L.PGLN by the number of lines that are used for output. L.PGLN is initially set to 57 (octal 71) and is reset when it reaches zero.

= the location of the first word of the text.

 $T_i$ = 0 if the initial byte = 0.  $\neq$  0 if the initial byte =  $T_i - 1$ .

M<sub>j</sub>

= the word count if  $T_i = 0$ .

= the character count if  $T_1 \neq 0$ . The symbol j may be  $1 \dots n$ .

JOBOUL is used for writing complete lines. The first character of the line must be a blank. When JOBOUL is used, MZE is invalid, T<sub>j</sub> must be 0, and M<sub>j</sub> must be the word count.

Page heading and page numbering are automatic and are printed at the top of every new page.

Page Heading: A field of 17 words (PACHD) and a field of 14 words (SUBHD) are provided as entry points within Jobou for page heading information. They must be initialized by the object program. The contents of S.SHDR are inserted into the first five words of PACHD on the first entry to JOBOU.

Page Numbering: A one-word page number counter, PCNUM, is provided within jobou for page numbering. The field is entirely numeric and the page number is right-justified within it. The field is set to the address field of s.PCCT on the first entry to jobou; the routine L.UPPG increments the field by one before a page heading is written. The calling sequence for L.UPPG is:

TSL L.UPPG

PGNUM may be modified at any time by the user's program.

Data Record Control Character: 10Bs Type 3 data records produced by the System Output Editor (JOBOU) in a combined print/punch installation do not have the specified look-ahead control characters on a mode change. The 7040/7044-1401 Input/Output Utility program, as distributed, correctly processes mixed-mode records.

JOBOUM is used by the monitor for output to the system output unit or its alternate. The calling sequence for this routine is:

CALL JOBOUM(list)

where list is the initial address of a set of parameters that are described under "JOBOU."

JOBPP is used for output to the system punch file. The calling sequence for this routine is:

CALL JOBPP(list)

The symbol list is the location of one parameter having the following form:

list pfx card where:

pfx

PZE if the record is BCD.MZE if the record is binary.

card

= the initial address of a block of 14 words if the record is BCD or a block of 24 words if the record is binary. The block contains the information to be placed in the buffer. The last two words of a binary record (that is, card columns 73-80) will be taken from the field PPLBL.

Card sequencing of binary cards is provided automatically from the PPLBL field. The symbol PPLBL is an entry point within JOBPP that may be initialized by the user. It is an eight-character card label (BCD) and is left-justified. Its initial setting is:

BCI 2, 00000001

The first four bytes are assumed to be alphabetic, the next four numeric, and the last four blank. The Punch Editor will increase the numeric bytes by one, after each binary record is placed into the buffer.

JOBPP opens the system punch file with repositioning and label check if both the punch file open bit (17 in s.sflc) and the combined status bit (35 in s.sflp) are off. If the punch file open bit is on and the combined status bit is off, jobpp opens the punch file without repositioning or label checking. If the combined status bit is on, jobpp does not open the system punch file.

LDAVL tests for the need and availability of a load file and opens the load file, if necessary. The calling sequence for this routine is:

TSL LDAVL

LDRSTK stacks the load file, or skips the deck on the input file. The calling sequence for this routine is:

TSL LDRSTK

OCLOS closes the system output file or its alternate without rewinding or label checking. The calling sequence for this routine is:

CALL OCLOS

OCLOSM is used by the Processor Monitor to close the system output file without rewinding or label checking. The calling sequence for this routine is:

CALL OCLOSM

*PCLOSE* closes the peripheral punch file. The calling sequence for this routine is:

CALL PCLOSE

IMJIE processes the SIEDIT card. The calling sequence for this routine is:

TRA IMJIE

PIMENT processes the SENTRY card. The calling sequence for this routine is:

TSL PIMENT

PIREL processes the SIBREL card. The calling sequence for this routine is:

TSL PIREL

**POEDIT** processes the SOEDIT card. The calling sequence for this routine is:

TSL POEDIT

*PPAVL* checks the availability of the system punch unit. The calling sequence for this routine is:

TSL PPAVL

PVARJ decodes the variable field of the SIBJOB card. The calling sequence for this routine is:

TSL PVARJ

SYSTUS tests availability status and reserve status of a given unit. The calling sequence for this routine is:

TSL SYSTUS pfx name

where:

pfx

= PZE if name is S.SUxx

= MZE if name is a file name

name

= a symbolic unit (S.SUxx) or a file name

Any of the following file names may be used in the calling sequence:

S.FBIA

S.FBOA

S.FBPP

S.FBLD

Each of these file names is the initial address of a standard 19-word file control block, as described in the publication *IBM 7040/7044 Operating System* (16/32K): Input/Output Control System, Form C28-6309.

At the completion of the routine, the accumulator is set to + if the unit is available; it is set to - if the unit is not available.

TESTIO decodes rocs options on the SIBJOB card. The calling sequence for this routine is:

TSL TESTIO

### Processor Files

The Processor files have the following file control blocks (note that the file name becomes the name of the file control blocks):

NAME	DESCRIPTION	ASSIGNED TO
S.FBIN	System input file	S.SIN1
S.FBOU	System output file	S.SOU1
S.FBPP	System punch file	S.SPP1 or S.SOU1 if print and punch out- put are combined
S.FBIA	Alternate system input	S.SUxx, as indicated by
D.I DIII	file	a \$IEDIT card
S.FBOA	Alternate system output file	S.SUxx, as indicated by a \$OEDIT card
S.FBLD	System load file	S.SUxx, as determined
		by the Editor or Proc- essor Monitor
S.FBED	System edit file	S.SUxx, as determined by the Edit Monitor
S.FBCP	System copy and reload	S.SUxx, as indicated on
	file	the \$IBJOB and \$RELOAD cards

### **System Input File**

This file contains the input to the system and may be used by object programs. It has the following characteristics:

- 1. It may be labeled.
- 2. It is multireel. Unit switching is automatic.
- 3. There must not be file marks within the body of the file.
- 4. Its file control block, s.fbin, is in the Subroutine Library. It specifies an error return, s.iner, which permits retry only on card equipment errors.
- 5. Its file description is fixed by system requirements and installation option. Its error return is fixed by installation option.
  - 6. Its buffer areas are included in the using program.
- 7. All \$ control cards are singly blocked on this file; all other cards are blocked according to installation option.
  - 8. The file may be created on the 1401.
  - 9. It is a mixed-mode file.

### System Output File

This file contains the print output for the system and may be used by object programs. It has the following characteristics:

- 1. It may be labeled.
- 2. It is multireel. Unit switching is automatic.
- 3. There must not be file marks within the body of the file.
- 4. Its file control block, s.fbou, is in the Subroutine Library.
- 5. Its file description is fixed by system requirements and installation option.
  - 6. Its buffer areas are included in the using program.
- 7. It is a BCD file, unless it is a combined print/punch file.
- 8. If print and punch output are combined, then both files (s.fbou and s.fbpp) are assigned to s.sou1, buffer areas are shared in the program, and the combined files form one mixed-mode file.

### System Punch File

This file contains the punch output for the system and may be used by object programs. It has the following characteristics:

- 1. It may be labeled.
- 2. It is multireel. Unit switching is automatic.
- 3. There must not be file marks within the body of the file.
- 4. Its file control block, s.fbpp, is in the Subroutine Library.
- 5. Its file description is fixed by system requirements and installation option. Its error return is fixed by installation option.
  - 6. Its buffer areas are included in the using program.
  - 7. It is a mixed-mode file.

### **Alternate System Input File**

This file (s.fbia) serves as an alternate source of symbolic input to the compilers and relocatable input to

the Loader. It is used by inserting a SEDIT card in the system input file. It has the same characteristics as the system input file, except that it cannot be multireel and that it only contains source or relocatable decks.

### **Alternate System Output File**

This file serves as an alternate for compiler print output only. It is used by inserting a SOEDIT card in the system input file. It has the same characteristics as the system output file except that it cannot be multireel.

#### System Load File

This file contains stacked relocatable input to the Loader. It has the following characteristics:

- 1. It may be labeled.
- 2. It is single-reel.
- 3. There may be no file marks within the body of the file.
- 4. Its file control block, s.fbl.d, is in the Subroutine Library.
- 5. Its file description is fixed by system requirements and installation option.
  - 6. Its buffer areas are included in the using program.
  - 7. It is a binary file.
- 8. If IFCHO is set to 1, it is preferably on the same channel as s.sin1. It is preferably not on channel A or on channel S.
- 9. If only four work units are available for a COBOL compile and load application, it is on a unit that is one of the compiler work units.
- 10. It is a tape file or a disk or drum file in a standard format other than random access.

### System Edit File

This file contains control cards and absolute input to the System Editor. It has the following characteristics:

- 1. It may be labeled.
- 2. It is single-reel.
- 3. It may not contain file marks within the body of the file.
- 4. Its file control block, s.fbed, is in the Subroutine Library.
- 5. Its file description is fixed by system requirements and installation option.
- 6. Its buffer areas are included in the using programs.
  - 7. It is a binary file.
- 8. If NOSOURCE is specified on the SIBEDT card, the edit file will preferably not be on the same channel as S.SIN1. If SOURCE is specified, the edit file will be on the same channel as S.SIN1. In any case, the edit file will preferably not be on channel A and will never be on channel S. (These rules apply only if the system is assembled with IFCHO SET 1.)

9. It is a tape file or a disk or drum file in a standard format other than random access.

#### System Copy and Reload File

This file contains absolute programs, in System Library format, to be used by the Reload Program. It has the following characteristics:

- 1. It may be labeled.
- 2. It is single-reel.
- 3. It will contain file marks within the body of the file.
- 4. Its file control block, s.fbcp, is in the Subroutine Library.
- Its file description is fixed by system requirements and installation option.
- 6. Its buffer areas are included in the Reload Program, but they are created by the Loader only if copy is specified.
  - 7. It is a binary file.
- 8. Its block size is the same as that of the system edit file. It is assembled as zero.

#### **Processor File Maintenance**

Beginning- and end-of-file procedures include rewinding and, if the file is labeled, verification or creation of header or trailer labels.

The following are cases in which files are opened with beginning-of-file procedures:

- 1. The s.sin1 and s.sou1 files are opened with a beginning-of-file procedure by the Supervisor.
- 2. The s.spp1 file is opened with beginning-of-file procedures by JOBPP. Bit 17 of location s.splc is set to 1 to indicate the file is open.
- 3. The alternate input and the alternate output files are opened with a beginning-of-file procedure by the Processor Monitor when it encounters a SIEDIT OF SOEDIT card specifying an alternate unit.
- 4. The load file is opened with beginning-of-file procedures, first by the Processor Monitor when stacking of relocatable decks is required by the Loader, and later by the Loader itself.
- 5. The edit file is opened with a beginning-of-file procedure by the System Editor.

The following are cases in which files are closed with end-of-file procedures:

- 1. The s.sin1, s.sou1, and s.spp1 files are closed with end-of-file procedures by the Supervisor.
- 2. The alternate input and alternate output files are closed with end-of-file procedures by the Processor Monitor when the next \*IEDIT or \*OEDIT card is encountered or at the end of a job.
- 3. The load file is closed with end-of-file procedures by the compilers or by the Processor Monitor.
- 4. The edit file is closed with end-of-file procedures by the Editor Monitor and later by the Loader.

A file is reopened when it is opened without rewinding or label processing. Any system component that uses a system file must reopen it. Only a monitor is permitted to open the files. Before the alternate input, alternate output, peripheral punch, load, or edit files are opened, a unit assignment must be obtained from the Nucleus and placed in the file control block.

A file is released when it is closed without rewinding or label processing. Any system component that uses a system file must release it.

The Input and Output Editors (JOBIN, JOBOU, JOBPP) perform reopening and releasing automatically for all system files except the load file and the edit file. The pre- and post-execution routines of the Processor Monitor perform these procedures for object programs.

The copy and reload file is opened and closed with end-of-file procedures and label processing for each program that is stacked and loaded. A dummy file, created by the Loader, indicates the end of all stacked programs.

### **Utility Files**

All utility files can be opened and closed at the user's discretion. All of them must be closed when processing is completed by the routine. Object program files will be closed by the post-execution routines of the Processor Monitor. Figure 27 shows the use of input/output units by the Processor components.

#### Labels

The file block for each of these files may include the following label information:

WORD	CONTENTS	
FCFSN	PZE	
FCRSN	PZE 1	
FCRET	PZE	
FCID1	BCI 2, S.S.	XXX
FCID2	BCI 1,0000	000

### **Compiler and Assembler Work Units**

The Processor Monitor chooses available units for the compilers, assembler, and Loader. It places the Symbolic Units Table entry for each of these units into a table in common storage. If possible, units are chosen according to the specifications that follow:

- 1. WORK1 (and WORK4 if COBOL is specified) should be on the same channel as s.sin1, but should not be on channel A or on channel S.
- 2. WORK2 and WORK3 should not be on the same channel as s.SIN1, nor should they be on either channel A or channel S.
- 3. In an edit run using the NOSOURCE option, WORK3 is the unit used for the system edit file.

4. The units are tape, disk, or drum units. Disk or drum units must be in a standard format other than random access.

Note: If If If to is set to 0, no consideration is given to channel qualifications when choosing units (rules 1 and 2 above do not apply).

### **Available Work Units**

A unit is available if the following conditions exist:

- 1. The unit is attached.
- 2. The unit is in ready status. (A test for ready status is performed only if the system is assembled with IFSNS SET 1.)
  - 3. The reservation code assigned to the unit is 00.
  - 4. The unit is not a unit record device.
- 5. The format of a disk or drum unit corresponds to a standard format of the installation.
- 6. If the system is assembled with LABELS SET 2, the label associated with the unit must be standard and must have an expired retention date.

The following procedures will be followed when there is a shortage of available work units:

- 1. When a job specifies the COBOL, NOGO, NOMAP, and NOLOGIC options, the job will be terminated if there are not at least four available units.
- 2. When a job specifies the COBOL, and GO, MAP, LOGIC, or DLOGIC options, the job will run normally if there are five available units (four work units plus the load unit). If there are four available units, no Preprocessor cards (\$NAME, \$USE, \$POOL, \$OMIT, \$LABEL, \$FILE) will be allowed. The job will run if there is only one COBOL deck and it appears first; otherwise, the job will be terminated. The job will also be terminated if less than four units are available.
- 3. When a job specifies the NOCOBOL and SOURCE, GO, MAP, LOGIC, or DLOGIC options, four units must be available (three plus the load file) or the job will be terminated.
- 4. When a job specifies the NOSOURCE option and less than three units are available, the job will run until one more than the number of available units is required.

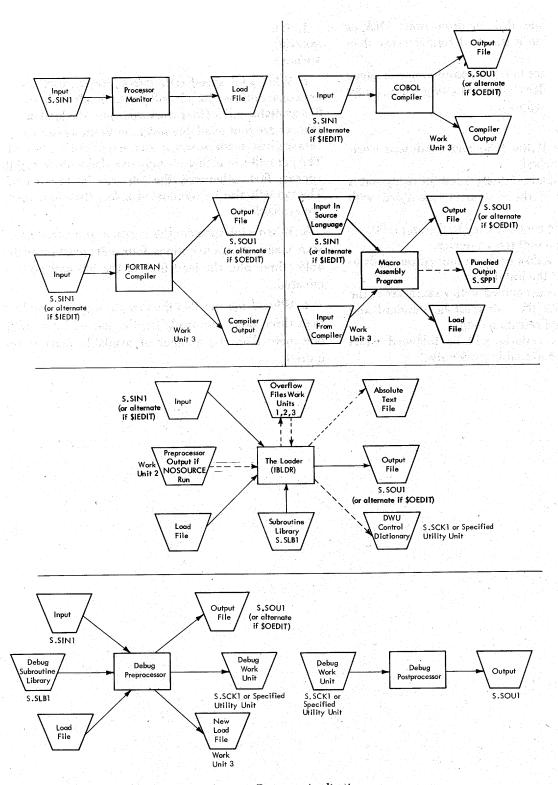


Figure 27. Flow of Input/Output Data During a Processor Application

## A Brief Description of the Loader Operation

The Loader operation is divided into phases or sections.

Section 1 performs the following functions:

1. It reads the relocatable input decks.

- 2. It forms a text-processing control dictionary and program name table from the input control dictionaries.
- 3. It separates the file text from the data and procedure text and stores them for later processing.
- 4. It processes the information retained by the Preprocessor from the sname, suse, and somit cards.
- 5. It determines which control dictionary entries refer to virtual names.
- 6. It searches the Subroutine Name Table for virtual names and forms a list of the required subroutines.
- 7. It reads the subroutine input decks, completes the control dictionary processing, and saves the subroutine file, data, and procedure texts.
- 8. It processes the information saved by the Preprocessor from the spool cards.
  - 9. It determines the size of blank COMMON.

Section 2 performs the following functions:

- 1. It assigns an absolute location to s.sloc and assigns absolute locations to control dictionary file entries.
- 2. It assigns absolute locations to control dictionary data and procedure entries.
- 3. It processes the file text saved by Section 1, forming working file blocks.
- 4. It processes the working file blocks; it performs unit assignment, buffer assignment, and data and procedure address field assignment.

Section 3 performs the following functions:

- 1. It places the file blocks in absolute text.
- 2. It processes the data and procedure text to form absolute text.
- 3. It places absolute text in core storage and places any overflow on an external storage unit in a form acceptable to the System Loader.
  - 4. It creates a copy file if copy is specified.

Section 4 performs the following functions:

- 1. It clears those parts of storage not used by the object program.
- 2. It loads the text from core storage and uses the System Loader to load from the overflow unit, if necessary.
  - 3. It transfers control to the object program.

Section O and Section I are two small sections that control Loader processing and perform initialization, respectively.

Section L produces a MAP, LOGIC, DLOGIC, and/or FILES output listing, if specified on the sibjob card.

### Description of the Program Deck

The program deck that is produced by the Macro Assembly Program for processing by the Loader consists of: BCD control cards; a binary control dictionary; binary text; and, optionally, a debugging dictionary.

The first card of each program deck is a SIBLDR card. This is followed by a SCDICT card, which signifies that the binary card(s) that follows contains the control dictionary for the deck. If a DDICT (either full or abbreviated) has been requested on either a SIBMAP or SIBFTC card, the next alphameric card is a SDDICT card followed by the binary card(s) containing the debugging dictionary. Otherwise, the next alphameric card is a STEXT card indicating that the subsequent binary card(s) contains the program text for this deck. The last card of every deck is a SDKEND card.

The STEXT card and the binary text cards following it are optional in a program deck that is to be loaded; however, they are usually included. All other previously mentioned cards must be present for each deck to be loaded and executed. At least one deck, in a group of decks to be loaded and executed together, must contain the STEXT card and the associated binary text cards.

### **BCD Card Format**

COL	UMN	INDICATES
1	8	in the state of th
\$IBLDR	decknm	Beginning of the program deck.
\$CDICT	decknm	Beginning of the binary control dictionary.
\$DDICT	decknm	Beginning of the binary debugging dictionary.
\$TEXT	decknm	Beginning of the relocatable binary text.
\$DKEND	decknm	End of the program deck

The decknm in columns 8-13 is not checked by the Loader. The Loader uses the deckname entry in the binary control dictionary to identify the deck. The date, which is inserted by the Assembler, starts in column 16 of the SIBLDR card.

### **Binary Card Format**

There are three binary sections of the Loader program deck: the binary control dictionary, the binary debugging dictionary (optional), and program text.

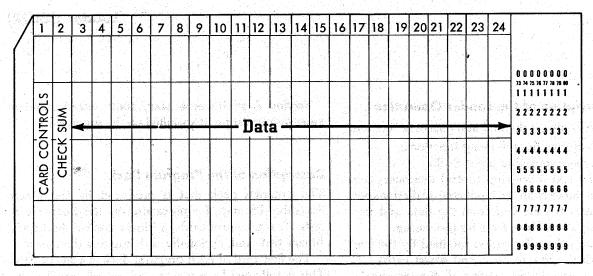


Figure 28. Column Binary-Card Format

The 24-word column binary card form is used as shown in Figure 28.

Each 36-bit binary word is punched in three columns of the card as shown in Figure 29.

The first word of the column binary card is a control word that describes the information on the rest of the card as follows:

BITS	CONTENT	MEANING
S, 1	11	Relocatable program deck indicator.
2	0	Verify check sum.
	1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1	Do not verify check sum.
3	0	IBJOB relocatable deck.
		Deck unacceptable to the Loader.
4	0	
5-7	001	The information on the rest of the card is relocatable program text.
atij tid jik	010	The information on the rest of the card is control dictionary text.
	100	The information on the rest of the card is debugging dictionary text.
8-12	01010	Column binary card indicators (9-7 punch).
13-17	xxxxx	Word count of meaningful words on card.
18-20	000	'보고등이 보고하고 모든 현실을입니.
21-35	oy still be w	Card sequence number.

The second word on the card is the logical sum of all words on the card except the check sum word. If this word is zero or if bit 2 of word 1 is 1, the Loader will not check the logical sum.

These two words are common to the control dictionary, relocatable program text, and the debugging dictionary portions of the deck (if present). Each of the three sections is sequenced independently (word 1, bits 21-35). Sequencing starts at zero; all cards must be in sequence.

The description of the other words on the card (words 3-24) will depend on whether the card is a control dictionary card, a relocatable program text card, or a debugging dictionary card. Since the debugging dictionary is an optional section, words 3-24 of

the DDICT card are discussed in detail in the section "Debugging Processor."

#### **Binary Control Dictionary**

Each card of the binary control dictionary has bits 5-7 of word 1 punched 010. The data words start in word three of the card and continue until either there is no more room on the card or the control dictionary is complete. The word count in the first word specifies the number of control dictionary words on each card.

The control dictionary is made up of two-word entries. There are seven possible kinds of entries in the control dictionary of each program deck to be loaded: Preface, Deck Name, Control Section, Reference Point, External Reference, Even, and File.

The order of these entries in the control dictionary is:

- 1. The Preface entry
- 2. The Deck Name entry
- 3. All File entries
- 4. All External Reference entries

12 [	S	12	24	
и [	1	13	25	
0	2	14	26	
1	3	15	27	
2 ير	4	16	28	
Punch Positions	5	17	29	
& 4	6	18	30	
[ 5	7	19	31	
ີ 6	8	20	32	
7	9	21	33	
8	10	22	34	
9	11	23	35	
	16,	17	18	
	Car	d Columns		

Figure 29. Placement of a 36-Bit (S-35) Binary Word in Three Columns of a Column Binary Card

5. Control Section, Reference Point, and Even entries in ascending order according to address

The sign bit of the last word of the control dictionary is minus.

Preface Entry: The first entry in the control dictionary indicates whether the deck is a relative or absolute deck, gives the location of the entry point for the deck, and indicates the number of bits needed to express a reference to the control dictionary. The contents of the two words in the Preface entry are:

where:

pfx

=PZE if the deck is a relative deck (produced by a RELMOD assembly)

=PTW if the deck is a relative deck and contains a (TCD

pseudo-operation) break.

=MON if the deck is an absolute deck (produced by an ABSMOD assembly).

=MTH if the deck is an absolute deck and contains a (TCD pseudo-operation) break.

= the relative or absolute location of the entry point for this

=the highest relative location assigned to the relocatable binary text, if the deck is a relative deck.

=the number of control dictionary entries, if the deck is an absolute deck.

=the maximum number of bits required to express a reference to any entry in the control dictionary. The value of n is such that the number of entries in the control dictionary is less than or equal to 2<sup>n</sup> but greater than 2<sup>n-1</sup>. (For example, if the control dictionary has 15 entries, n=4.)

Deck Name Entry: This is the second entry in the control dictionary. The contents of the two words in the Deck Name entry are:

where:

deckname

= the deck name as punched in columns 8-13 of the \$IBMAP, \$IBFTC, or \$IBCBC card for the deck.

= the length of the program for the deck.

File Entries: These entries are for file areas within the program deck that may be replaced, deleted, or referred to by other program decks. The format of each File entry is:

where filnam (a real name) is the name of the file area.

External Reference Entries: These entries are for references within a program deck to names in other program decks. The Macro Assembly Program produces External Reference entries for a symbolic deck containing:

- 1. EXTERN pseudo-operations. Every name in the variable field of an EXTERN pseudo-operation is included in the control dictionary.
- 2. System symbols, of the form S.xxxx, that are not defined within the symbolic deck.

The contents of the two words in each External Reference entry are:

where exname is the name that is external to the deck. Exname is a virtual name.

Control Section Entries: These entries are for procedure and data areas within the program deck that may be replaced, deleted, or referred to by other program decks. The contents of the two words in each Control Section entry are:

BCI	1, ctlnam	
PON	loc, , lngt	

where:

ctlnam

= the name of the control section. Ctlnam is a real name.

=the relative or absolute location of the beginning of the control section. lngth

=the length of the control section.

Reference Point Entries: These entries are for reference points within this program deck. These reference points are not necessarily entry points to the deck. References can be made by other program decks to be loaded with this program deck. Entries of this type are produced by an entry pseudo-operation. The contents of the two words in each Reference Point entry are:

BCI	1, refnam
PON	loc, 0
TOIL	100, 0

where:

refnam

=the name of the reference point. Refnam is a real name.

= the relative or absolute location of the reference point.

Even Entries: These entries provide information to the Loader in order to ensure an even load address for a certain instruction or word of data. Even entries are produced by an even pseudo-operation in the program deck. The contents of the two words in each Even entry are:

where loc is the location of the data or instruction, relative to the start of the deck.

### Relocatable Binary Program Text

There are two sections of relocatable binary text:

- 1. File text
- 2. Data and procedure text

File text always appears first, if it exists. Data and procedure text follows file text and always starts on a new card. Either or both of these parts of the relocatable binary program text may be omitted. Words 1 and 2 are control and check sum words, as previously noted. Bits 5-7 of the first word are 001 for relocatable binary program text cards. Words 3, 4, and 5 contain nineteen 5-bit control groups that give relocation information for the nineteen data words on the card (words 6-24). Words 3 and 4 each contain seven control groups; word 5 contains the remaining five control groups (see Figure 30). The sign of these words is plus, with one exception: word 3 of the first card containing data and procedure text has a sign of minus.

Words 6 through 24 of the card contain relocatable binary program text.

The five-bit control groups indicate to the Loader how the associated data word is to be processed. The first bit of the five bits indicates that the data word which follows is either a standard entry or a special entry.

Standard Entry: Bit 1 of the five-bit control group is 1 for a standard entry. The remaining four bits describe the manner in which the address and decrement of the associated data word are to be processed. It is important to realize that several data words on a card may form a single absolute text word. Only the address and decrement fields of the standard entry data word are relocatable. The prefix and tag fields are not changed by the Loader; they are determined by the Macro Assembly Program and passed on to the Loader.

Standard entry control groups have the following format:

#### 1 ddaa

where the bits dd and aa specify the type of relocation for the decrement and address, respectively, of the data word. The following bit settings are possible for dd or aa:

00 means the value of the field is constant.

- 01 means the value of the field is determined in relation to the start of the deck.
- 10 means the field is a reference to the control dictionary (this setting is possible for aa only).
- 11 means the field is part of a complex expression that may extend into the subsequent data words.

Constant Addresses and Decrements: The code 00 indicates that the address or decrement of the associated data word that is passed to the Loader by the Macro Assembly Program is not changed by the Loader. No processing is done by the Loader; the field has been completely processed by the Macro Assembly Program.

Relative Addresses and Decrements: The code 01 indicates that the address or decrement of the associated data word is a value that is determined in relation to the start of the program deck. The Control Break Table, which is internal to the Loader and accounts for deletions or insertions made in a program deck, is scanned for the relative value in the associated data field (address or decrement). If the relative value is within a deleted section of this deck, the reference is considered invalid and no assignment will be made. If the value is within a retained section of the deck,

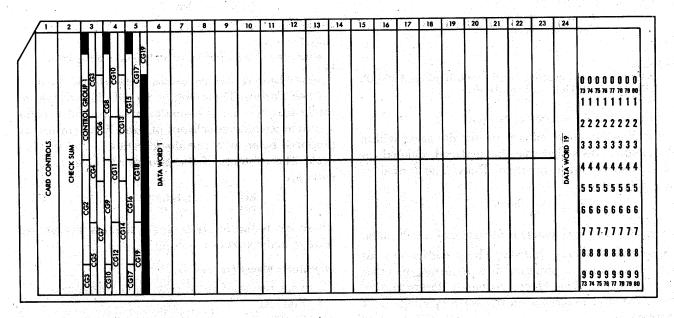


Figure 30. Relocatable Binary Text Format

an absolute base that has been adjusted for all previous insertions and deletions is added to the relative value.

Dictionary Reference Addresses: The code 10 indicates that the value in the address field is a dictionary reference. This code may be used for only the address field of a data word within a long form complex. A dictionary reference always results in a complex expression.

Note: Any reference within a symbolic deck to a point within a control section is expressed to the Loader in the form b + c, where b is a pointer to the control dictionary and c is a displacement from the start of the control section. At load time, the Loader will adjust this reference to allow for insertions within the control section that result from even pseudo-operations. External references are expressed in the form b + c, with c equal to 0.

Complex Field Decrements and Addresses: The decrement or address, or both, of a data word can be represented as a complex expression requiring evaluation at load time. Addresses or decrements requiring such evaluation are given a relocation code of 11. The expression to be evaluated may then be expressed in one of two ways:

1. The associated data field (decrement/address) equals zero (sometimes called Long Form Complex). The long form complex is used when an expression (e.g., A/4+3/4) is longer than can be expressed in the associated data field or when an expression uses multiplication or division. The long form complex consists of a string of data words, each with a corresponding control group.

Each word of a long form complex has the form:

where:

pfx

=PZE if addition (+) is indicated

=PON if subtraction (-) is indicated

=PTW if multiplication (\*) is indicated

=PTH if division (/) is indicated

The control groups describing subsequent words in the strings for an expression have the same format as the control groups already described, except that the meaning of the 11 code is changed to mean a result storage cell. Seven such result storage cells (designated by placing b equal to 0, 1, ..., 6) can be used for intermediate results during the evaluation of a complex expression.

A word in a complex expression is interpreted as:

where b is the result storage cell into which the result

is to be placed. The complex expression is terminated by a word for which b = 7.

2. The associated data field (decrement/address) does not equal zero (sometimes called Short Form Complex). This form may be used to express complex fields of the form, name+c, where name is the external name of a control section. The 15-bit field is formed as follows:

Bit 1 = 0.

Bits 2 through n + 1

indicate the relative location of name in this deck's control dictionary. n is the number of bits required to express the total length of the control dictionary (for example, if CDICT contains 15 entries, then 4 bits, required to express 15 as a binary number, would be used, i.e., bits 2-5).

Bits n + 2 through 15

contain a constant (15 - n - 1) bits long) to be added to the location assigned to name. Note that the long form complex may have to be used if more than 14 bits are required to express the length of the control dictionary and the addend.

Special Entry: Bit 1 of the five-bit control group is 0 for a special entry. Special entry control groups have the following format:

0 ssss

where the four bits, ssss, specify the type of special entry.

0 0000: End of card (no data word is associated with this entry). No more data words appear on this card.

0 0001: Location counter control. The associated data word expresses a change in location counter. The format of the data word is:

where:

pfx

- =PZE if x is an absolute origin.1
- =PON if x is a relative origin.
- = MZE if a complex origin within this deck is defined in the associated expression that contains a dictionary reference to a real section.
- = MON if a complex origin within this deck is defined in the associated expression that contains a dictionary reference to a virtual section.
- =MTW if a complex origin may or may not be within this deck. It is used in a FORTRAN blocked data program.
- =PTW if this is a BSS of length x.
- =PTH if this is an EVEN pseudo-operation; x is zero.

0 0011: File text follows. The address portion of the associated data word contains the relative location of the file name entry in the control dictionary. The decrement portion contains the number of file information words that follow. (This number is dependent on whether label information is present or not. If label information is present, the sign is minus.) The subse-

Origins are an integral part of text and each card does not carry its relative load address.

quent data words and their associated control groups contain information for generating file control blocks. They are formed from FILE pseudo-operations.

The eight words of file text, which follow the first word, correspond respectively to words fccur, fclng, fcrct, fcuni, fccon, fcfcn, fcbsn, and fcexi of the file control block described in the publication IBM 7040/7044 Operating System (16/32K): Input/Output Control System, Form C28-6309. Control groups of constant address and decrement are associated with them except where an external name is specified (as in err, eor, eof, addled, and nslbl options), in which case the appropriate portion of the control group is complex and the data field is short form complex. If there is labeling information, there are five more words of file text. These have control groups indicating constant addresses and decrements.

0 01XX: End of current logical record (TCD pseudooperation) break, where xx describes the address portion of the data word as follows:

- 00 Constant
- 01 Relative
- 11 Complex

The address formed will be the location of the entry point for the storage load that has just been constructed. (All Subroutine Library subroutines called by a program using the TCD pseudo-operation will be loaded with the final logical record; all file control blocks and pool control words will be loaded with the first logical record.)

0 10VV: vpp expression, where vv describes the manner in which the data field is to be processed by the Loader. (The meaning of the vv codes is given later in the text.) The Loader will assemble a string of bits or words that contain constants, relative locations, dictionary references, and complex expressions that do not necessarily fall within the normal boundaries of address or decrement.

Each data word that corresponds to a VFD control group specifies one field of the VFD expression and specifies whether a word should be terminated. The general format of a data word is:

		and the second s		10/10/11/11/11		- W. A.	A STATE OF THE STA
	t	.ct			lata field		
٠.	S 1		5 6				35
	whore		a season in Cityral	The state of the	2. 6		Builder Store J.

=0 if the assembly continues in the same machine word or words (across machine word boundaries).

=1 if the current machine word is terminated and filled with zeros in the unused rightmost positions.

=the number of bits from the data field that are to be inserted into the generated string. The rightmost ct bits are inserted. (ct≤30)

The vv codes that may appear in the control group for a VFD expression are as follows:

=00 if the data field is constant.

=01 if the address portion (last 15 bits) of the data field contains a relative address (described under "Standard Entry"). It is relocated in the same manner as any other relative field. The rightmost ct bits are inserted into the VFD string.

=10 if the address portion of the data field is a dictionary reference (described under "Standard Entry"). The address is computed and the rightmost ct bits are inserted into the

VFD string.

=11 if the address portion of the data field conforms to the rules for complex expressions (described under "Standard Entry"). After evaluation, the rightmost ct bits are inserted into the VFD string.

o 1100. Object-time reference to the object program entry point (used by the Debugging Processor). The associated data word can be any constant that has zero in the address portion of the word. The address portion will be set to the program entry point by IBLDR.

0 1111: End of text indicator. No data word is associated with this control group.

### Storage Allocation

The Loader allocates storage to the object program, as follows:

1. If s.s.oc is referenced in the program, it is assigned to the first non-storage-protected location after the level of rocs specified on the sibjob card, and a five-word block is placed in the succeeding locations.

The five words will contain the following:

S.SLOC +1 fblorg, t, totlng

+2 7, fbllng

+3 tabavl +4 dporg,, dpend

+5 blcorg

where:

fblorg

= the location of the first word of the first file control block.

=6 if the IOBS level of IOCS was specified on the \$IBJOB card.

=4 if the IOLS level of IOCS was specified on the \$IBJOB card.

=2 if the IOOP2 level of IOCS was specified on the \$IBJOB card.

=0 if the IOOP1 level of IOCS was specified on the \$IBJOB card.

toting

= the total length in words of the file control blocks.

fbllng

= the length in words of a file control block (19).

=the first location available for tables. It is one of the following:

 The first location for the /DEBUG deck if load-time debugging facilities are used.

2. The first location in the Pool Control Table.

The first location in blank COMMON if there are no buffers.

4. S.SEND if there are neither buffers nor blank COMMON.

dporg

=the first location used by data or procedure text in the decks loaded for this program.

dpend

=the last location used by data or procedure text in the decks loaded for this program.

blcorg

= the first location used in blank COMMON.

NOTE: If the appropriate items do not exist for a program, fblorg, fbllng, and blcorg are zero. Unspecified bits do not necessarily contain zeros.

Figure 34 shows the relative location of these words in core storage. When a dependent chain link is loaded into core storage, dpend is adjusted.

- 2. File text for each file is formed into a 19-word file control block. All file control blocks are located either immediately after the five-word block following s.s.oc or at the first non-storage-protected location following the area used by the level of iocs specified on the sibjob card. (File text from the Subroutine Library appears with the rest of the file text.)
- 3. The data and procedure text are formed into absolute text starting at the first location after the last word assigned to the file blocks.
- 4. Subroutine Library data and procedure text follow the data and procedure text formed from the input decks.
- 5. Blank common is assigned storage immediately below the highest location available to the system (s.send).
- 6. Buffers are assigned immediately below blank common.
- 7. Pool control words are created and are assigned locations immediately below the buffers.

Note: Programs assembled in Absmod are loaded without regard to the points above. The user must, therefore, exercise care so that absolute text does not conflict with locations assigned for relocatable text, file control blocks, etc. Locations assigned to absolute text are not taken into account when relocatable text (including library routines) is loaded.

### **Reload Program**

The Reload Program is a subsystem under the Processor Monitor. It is designed to search for, load, and execute absolute object programs produced by the Loader when copy is specified on the SIBJOB card.

The Reload Program eliminates the necessity of using the Loader each time the programs are loaded, thereby reducing load time of debugged programs.

### Format of Input

The input unit to the Reload Program contains one or more files in the following format:

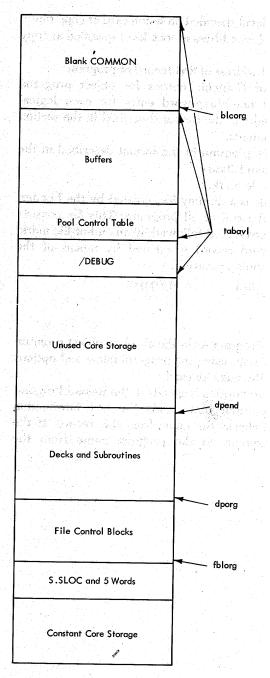


Figure 31. Storage Allocation

- 1. Information record about the object program including:
  - a. Name
  - b. File block information
  - c. Entry point
  - d. Buffer origin
  - e. // common origin
  - f. s.send at copy time

- g. 10Cs level specified on \$1BJOB card at copy time
- h. Absolute address of rocs level specified at copy time
- i. Load address of first record of program
- 2. Table of Contents entries for object program, consisting of one three-word entry for each logical record (or link) in the format described in the section "Table of Contents."
- 3. Absolute program in the format described in the section "System Library."
  - 4. End-of-file mark.

The last file is a dummy file, created by the Loader, to indicate the end of all programs. This file consists of a three-word record followed by an end-of-file mark. The three-word record is created by means of the following pseudo-operation:

BCI 3, /DUMMY

### **Operation**

The Reload Program seeks the absolute object program on the input unit, using the program name and options specified in the SRELOAD card.

If the NOSRCH option is specified, the Reload Program rewinds the input unit, reads the first information record, and checks the name from this record. If the name corresponds to the program name from the

sreload card, the rest of the information record is processed and checking is done to see if the program can be loaded correctly. If the name does not correspond, processing is terminated and the job is skipped.

If the SRCH option is specified, the Reload Program positions the input to the next information record and checks the program name for this record. If the name corresponds to the program name from the SRELOAD card, the rest of the information record is processed and checking is done to see whether the program can be loaded correctly. If the name in the next information record does not correspond to the one in the SRELOAD card, the Reload Program repeats the above process until the specified program is located, or until all files on the input unit have been checked. If the program cannot be located, processing is terminated and the job is skipped.

The Table of Contents record is then read and adjusted to correspond to the unit that contains the object program. It is then transmitted to the Abbreviated Table of Contents area in the Nucleus.

Core storage is then cleared, the System Loader is used to load the object program, a BEGIN message is typed, and control is given to the object program. If the System Loader finds that the device is incorrectly positioned, a permanent error occurs since the device cannot be correctly positioned.

## A Brief Discussion of the Assembly Process

The information in this section is provided to give the reader an understanding of the basic structure of the Macro Assembly Program.

The Macro Assembly Program is divided into six sections: (1) Phase A (for source input only) performs a pass over source input to expand macroinstructions and to form input to the Dictionary Reduction section and to Phase B; (2) Interface (for output from the compilers only) performs a pass over output from the compilers to expand macro-instructions and generate card images for the listing; (3) the Dictionary Reduction section assigns values to symbols for use in Phase B and produces the control dictionary for use by the Loader; (4) Phase B assembles the instruction text for the Loader and produces the assembly listing for the deck; (5) the Cross Reference Dictionary

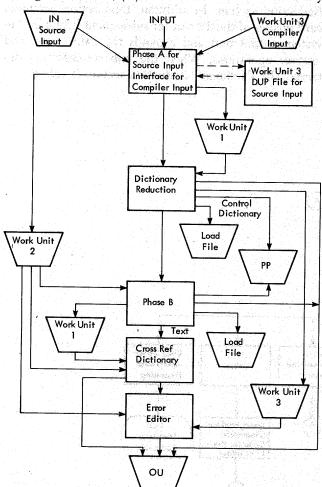


Figure 32. Block Diagram for Source and Compiler Input

section produces the cross reference dictionary if REF is specified, and (6) the Error Editor writes out the diagnostic error messages for the deck.

The sequence of sections for source input is:

Phase A

Dictionary Reduction

Phase B

网络拉拉斯克斯 河北 化特别克拉 潜水电影 Cross Reference Dictionary

Error Editor

The sequence of sections for compiler input is:

Interface

Dictionary Reduction

Phase B

**Cross Reference Dictionary** 

Error Editor

Figure 32 shows the sequences in a generalized block diagram form.

tika dan dikabilik dikabili balan kalendar dikabili dikabili dikabili dikabili dikabili dikabili dikabili dikab

The input/output units are as follows:

IN contains the source input, if applicable, and may be S.SIN1 or any utility unit specified by a \$IEDIT card.

Work Unit 3 contains compiler input, if applicable, and later contains diagnostic data from Dictionary Reduction and Phase B, for the Error Editor. Work Unit 3 is also the Phase A DUP file, if applicable to source input.

Work Unit 1 contains the dictionary data between Phase A and the Dictionary Reduction phase. It also contains intermediate data between Phase B and the Cross Reference phase, if REF is specified.

Work Unit 2 contains the internal text (T2) and, if LIST is specified, card images between Phase A and Phase B. It also contains a symbol dictionary for the Cross Reference Dictionary and Error Editor.

OU contains the assembly output, including the listing and diagnostic messages. It may be S.SOUT or any utility unit specified by a \$OEDIT

PP contains the punched output, and is S.SPP1.

Load File is used when either GO, MAP, FILES and/or LOGIC or DLOGIC options are specified to contain the relocatable binary decks for the program.

The Input, Output, and Punch Editors are used by the Macro Assembly Program. The actual output of the assembler is dependent upon the SIBCBC, SIBFTC, SIBMAP, or SIBJOB card options.

### Phase Asserting to the special section will be applied by

This section obtains source input from the system input unit (or the unit indicated by a SIEDIT card) and produces two sets of data: the definition file for the Dictionary Reduction section and the internal binary text (T2) to be processed by Phase B.

Phase A contains an initialization section which: (1) interprets the SIBMAP card and sets the appropriate switches, and (2) enters, by hashing, the operation codes into an External Reference Table.

When the initialization process is completed, each source input statement is read in and classified as either a macro definition, a macro usage, a pseudo-operation, a one-for-one instruction, a comments (\*) card, or an end statement. Each source input statement is then processed by the appropriate section of Phase A to produce for that statement the internal text (T2), which is written onto works. If the statement has a location symbol or is a pseudo-operation, an entry is also written on works for the Dictionary Reduction section. When the end statement is detected, the following processing is completed before the Dic-

tionary Reduction section is called: literals are processed, EXTERNS for undefined system symbols (e.g., s.send) are written on the definition file for the Dictionary Reduction section, and the END statement is written in internal text format on work1 and work2.

Disk and Drum Orders: The IBM 1301 Disk, 1302 Disk, and 7320 Drum Orders are an optional feature of the 7040/7044 Macro Assembly Program. To assemble these orders, the user must update and reassemble Phases A and B, and then edit the binary decks onto the current system tape. The symbolic instruction DISKSW SET 0 must be changed to DISKSW SET 1 in MP3000, MP3300, and MP6000. Since Phase A (MP2600, MP3000, MP3300, and MP4000) is chained, all of it must be reassembled and edited together.

Figure 33 shows an overall flowchart for Phase A.

### Interface

This section obtains input from the compiler input file and routes the T2 text to the Phase B file. If the statement has a location symbol or is a pseudo-operation, an entry is also written on works for the Dictionary Reduction section. In addition, this section produces card images of the T2 text for subsequent listing on the system output unit and expands the CALL, SAVE, and RETURN operations. When the END statement is detected, the Dictionary Reduction section is called.

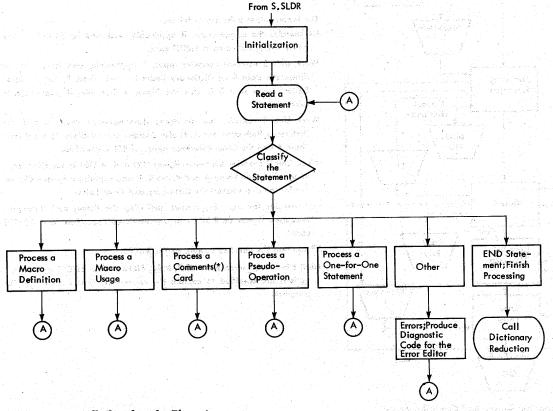


Figure 33. Overall Flowchart for Phase A

### **Dictionary Reduction**

The function of this section is assigning values to all symbols and generating the control dictionary for use by the Loader.

The Dictionary Reduction section builds an internal dictionary that contains values for all symbols in the program, as computed by the Dictionary Reduction section. Pseudo-operations are evaluated, if possible, and the location counter is incremented as required. The ENTRY, EVEN, and CONTROL pseudo-operations are evaluated; these, together with FILE and EXTERN pseudooperations, are used to form the control dictionary, which is punched on s.spp1 and/or written on the load file as it is formed. If a pseudo-operation cannot be evaluated initially, it is placed into the pseudo-operation dictionary for later evaluation. Each entry in the internal dictionary is chained to the preceding entry to form a program chain; entries in the pseudo-operation dictionary are made part of the program chain. When the END statement is read, the pseudo-operation dictionary is consulted to see if there are any unevaluated pseudo-operations, which are then evaluated. The internal text is then reordered into the format required by Phase B. If errors are detected, error indications are passed to the Error Editor.

The control dictionary will be listed on s.sou1, if LIST is specified.

At the completion of the Dictionary Reduction section, the internal dictionary is left in storage and Phase B is called.

### Phase B

The primary function of this phase is to produce an output deck in relocatable format. Direct input to this section is produced by both Phase A and Dictionary Reduction. The output consists of sequential coding of the program in the relocatable binary format acceptable to the Loader (IBLDR).

Phase B reads the text from either Phase A or the Interface and, using the internal dictionary produced by Dictionary Reduction, inserts the appropriate (relocatable) values. If an error message is required for a statement, a code is written on the error unit for the Error Editor. The statement is processed by determining the values of any symbols in the statement from the internal dictionary and producing an assembled instruction in relocatable binary format. The assembled statement is written on the system peri-

pheral punch file if DECK is specified on the \$ card for the deck; it is written on the system output file with an assigned statement number if LIST is specified on the \$ card for the deck; and it is written on the load file if CO, MAP, and/or LOGIC or DLOGIC are specified on the \$IBJOB card for the Processor application. When Phase B encounters the END statement, it calls the Cross Reference Dictionary if REF has been specified, and, if necessary, calls the Error Editor. Otherwise, it returns to the System Return routine (s.SRET).

### **Cross Reference Dictionary**

The function of the Cross Reference Dictionary section is to produce the cross reference dictionary that appears in the listing if the REF option applies for an assembly.

The cross reference dictionary in the listing may contain four types of information:

- 1. Statement number references to defined symbols
- 2. Multiply-defined symbols
- 3. Undefined symbols
- 4. Starting and ending statement numbers for location counters

Internally, the Cross Reference Dictionary section has three major parts. The first part reads the external dictionary produced by Phase A or Interface. The external dictionary contains the BCD symbol names in the same order as the entries giving the definition of these names in the internal dictionary. Using the external and the internal dictionaries, this part classifies all names as defined, multiply-defined, undefined, or as names associated with the location counters. The first part also alphabetically sorts the names within each of these four groups.

The second part reads a file that is produced by Phase B. This file contains one-word entries (or two-word entries, if within a macro) giving the location in the internal dictionary where the name is defined, and the statement number in which the reference to it is found. Using this information, this part associates the statement number with the BCD name to which it refers. The third part writes the output on s.sou1.

#### **Error Editor**

This section interprets the internal coding of error indications from the Macro Assembly Program, the FORTRAN Compiler, and the COBOL Compiler and produces the appropriate diagnostic error messages for the listing.

### The FORTRAN Compiler

### A Brief Description of the FORTRAN Compiler

The five sections of the FORTRAN Compiler are:

The Scan, which classifies source statements, produces diagnostic messages, forms dictionaries of symbols, and translates statements into internal text.

The Storage Allocator, which generates constants, translates FORMAT statements, builds alphameric tables, generates equivalences, reserves storage, and generates EXTERN pseudo-operations for external symbols.

The Arithmetic and Logical Translator, which expands input/output lists, builds the indexing table, and reorders arithmetic and logical statements into computational order.

The Indexing Analyzer, which forms the input to the Indexing Generator, optimizes the use of subscripts, and allocates index registers.

The Instruction Generator and Indexing Generator, which completes the generation of the T2 text for input to the Macro Assembly Program.

Figure 34 shows the relation of these sections, input/output unit allocation, interphase data flow, and the order of execution of these sections.

#### **Definitions**

TI Text: An internal text produced by the Scan section and modified by the Arithmetic and Logical Translator

T2 Text: The text that is passed to the Macro Assembly Program.

# Scan

This phase initially receives control from s.SLDR, as a result of a SIBFTC card being recognized. This phase then reads all statements from s.SIN1 and lists them on s.SOU1. Statements controlling data allocation (COMMON and EQUIVALENCE statements) are converted to internal reference text, and written on a utility unit.

Other information is tabled in the internal and external dictionaries. The imperative statements are written in the T1 text. During this process, undeclared names are added to the external dictionary and are classified. When errors are detected in the source program, the messages are placed in T2 text form in COMMON-EQUIVALENCE text. At the end of the pass, the now complete external dictionary, containing BCD names, is written on the T2 text file.

### Storage Allocator

This phase performs storage allocation based upon COMMON and EQUIVALENCE statements, and other information contained within the internal dictionary. Storage allocation instructions are added to the T2 text file. As soon as the COMMON and EQUIVALENCE information is read at the beginning of this phase, the unit is repositioned for use by the first section of the Arithmetic and Logical Translator Phase. In addition,

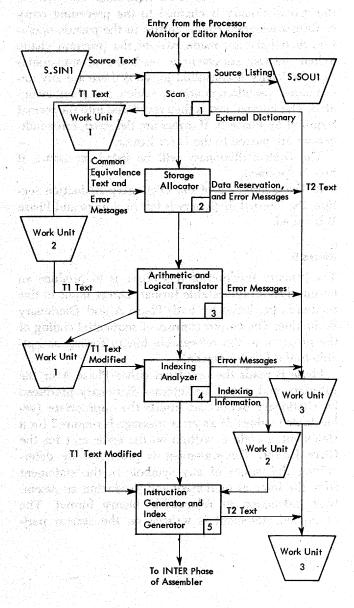


Figure 34. Flowchart Showing Unit Allocation, Interphase Data Flow, and the Order of Execution of the Sections

HOUSE THE RESEARCH

FORMAT statements are translated into calling sequences to the proper FORTRAN library conversion routines. Alphameric arguments in CALL statements are also written out in T2 form, as are any other literals referenced by the source program. Scan error messages are copied in T2 form onto the T2 text file.

### **Arithmetic and Logical Translator**

This phase reads statements in T1 text form, reorders and optimizes the arithmetic and the logical expressions read, and then writes the modified T1 text. The Arithmetic and Logical Translator creates the Dofon table for input to the Indexing Analyzer and expands input/output statements into Do loops, where this is implied in the input/output list.

### **Indexing Analyzer**

This phase reads the altered T1 text prepared by the Arithmetic and Logical Translator. Simultaneously, further indexing information and "regional" context records are written.

### Instruction Generator and Index Generator

This phase reads T1 text and indexing information and compiles generated instructions (in T2 format), adding these instructions to the T2 text file. The T2 text file then contains the entire input to the assembly phase of the 7040/7044 Macro Assembly Program. Control is transferred to the Macro Assembly Program, using the System Loader (s.sldr), for completion of the assembly and preparation of the object listing on s.sou1 and

for writing out the relocatable binary deck on the system peripheral punch unit (s.spp1).

Figure 35 shows the use of symbolic units by the FORTRAN Compiler.

### **FORTRAN System Routines**

#### **Format Conversion Routines**

The routines that provide for the editing and conversion of input and output records under the control of FORMAT statements are:

The ACV Routine performs alphameric conversion for BCD data.

IOHAC. is the entry point for the FORMAT A specification.

The MAP calling sequence generated by the compiler for this routine is:

AXT	field count, 2 (if any)
TSX	IOHAC., 4
PZE	width

The DCV Routine performs floating-point conversion for double-precision numerical data.

IOHDC. is the entry point for the FORMAT D specification.

The MAP calling sequence generated by the compiler for this routine is:

AXT	field count, 2 (if any)
TSX	IOHDC., 4
PZE	width., number of decimal

The ECV Routine performs floating-point conversion for numerical data.

Editing of true zero values is available for D and E

Compiler	UNIT					
Section	S. SIN1	S. SOU1	Work Unit 1	Work Unit 2	Work Unit 3	
Scan	Read Source Statements	Write Source Statements	Write Common- Equivalence Text and Error Messages	Write TI Text	et sûneeê ranî e dir i et si e di si etrasayî ê	
			Repositioning	Repositioning	Write External Dictionary	
Storage Allocation			Read Common– Equivalence Text and Copy Error Messages	Repositioning	Write T2 Text Write Error Messages	
Arithmetic		in the second	Repositioning	Repositioning		
Translator			Write 11 Text	Read T1 Text	Write Error Messages	
Index Analysis			Repositioning	Repositioning		
Andrysis			Read T1 Text	Write Merge Text	Write Error Messages	
Instruction Generation		Maraya Nasar Maraya Nasar	Repositioning	Repositioning		
Ceneration			Read T1 Text	Read Merge Text	Write T2 Text Write Error Messages	

Figure 35. Input/Output Unit Assignment

conversion of output. If the editing feature is included, a zero value is printed as a single digit, followed by a decimal point and, if the zero value is negative, preceded by a minus sign, as follows:

0. Positive zero value
-0. Negative zero value

The remainder of the field (the positions following the decimal point) contains blanks.

Without the editing feature, all zero digits and the exponent are printed, as in the following example:

#### -0.0000000E-38

The use of the editing feature is controlled by the following assembly parameter in the ECV routine:

SUPRES SET x

where:

X

=1 if the editing feature is to be included =0 if the editing feature is not to be included

The distributed version of the ECV routine provides the editing feature.

IOHEC. is the entry point for the FORMAT E specification

The MAP calling sequence generated by the compiler for this routine is:

AXT field count, 2 (if any)
TSX IOHEC., 4

PZE width, number of decimals

The FCV Routine performs fixed point conversion for numerical data.

For F conversion of output, the entire field is filled with asterisks when the specified width is insufficient to contain the number. For F conversion of positive numbers, limited relaxation of the field width requirement is provided by the extra digit feature. If this feature is used, the sign position of the field, which otherwise contains a blank, provides space for an additional significant digit.

The use of the extra digit feature is controlled by the following assembly parameter in the FCV routine:

EXTRA SET x

where:

=1 if the sign position may be used for a digit =0 if the sign position may not be used for a digit

The distributed version of the FCV routine does not provide the extra digit feature.

IOHFC. is the entry point for the FORMAT F specification.

The MAP calling sequence generated by the compiler for this routine is:

AXT field count, 2 (if any)
TSX IOHFC., 4

PZE width, , number of decimals

The HCV Routine inserts BCD characters from the input record into the FORMAT specification and inserts BCD characters from the FORMAT specification into the

output record.

IOHHC. is the entry point for the FORMAT H specification.

The MAP calling sequence generated by the compiler for this routine is:

TSX IOHHC., 4
PZE number of characters
BCI (number of characters+5)/6,

BCD text

The ICV Routine performs integer conversion for numerical data.

IOHIC. is the entry point for the FORMAT I specification.

The MAP calling sequence generated by the compiler for this routine is:

AXT field count, 2 (if any)
TSX IOHIC., 4
PZE width

The LCV Routine performs logical conversion for T and F characters.

IOHLC. is the entry point for the FORMAT L specification.

The MAP calling sequence generated by the compiler for this routine is:

AXT field count, 2 (if any)
TSX IOHLC., 4
PZE width

The OCV Routine performs octal conversion for numerical data.

юнос. is the entry point for the FORMAT o specification.

The MAP calling sequence generated by the compiler for this routine is:

AXT field count, 2 (if any)
TSX IOHOC., 4
PZE width

The RWD Routine controls the input and output of BCD records, and controls the conversion of alphameric data in accordance with FORMAT specifications.

IOHCM. is the initial entry point from conversion routines that use input/output list data items.

IOHCT. is the final entry point from conversion routines.

IOHLP, is the entry point for the FORMAT left parenthesis.

The MAP calling sequence generated by the compiler for this routine is:

AXT group count, 1 (if any)
TSL IOHLP.

IOHRP. is the entry point for the FORMAT right parenthesis.

The MAP calling sequence generated by the compiler for this routine is:

TSL IOHRP.

IOHEF. is the entry point for the end of a FORMAT statement.

The MAP calling sequence generated by the com-

piler for this routine is:

TRA IOHEF.

IOHIO. is the entry point for the FORMAT slash (it is also for all required BCD input/output).

The MAP calling sequence generated by the compiler for this routine is:

TSX IOHIO., 4

IOHSF. is the entry point for a FORMAT scale factor. The MAP calling sequence generated by the compiler for this routine is:

TSX IOHSF., 4 (signed scale factor)

Other entry points to this routine that are not concerned primarily with format conversion are listed in the publication *IBM* 7040/7044 Operating System (16/32K): Programmer's Guide, Form C28-6318.

The SCA Routine performs a scan and format translation for variable (or read-in) FORMAT statements at object time.

FMTSC. is the entry point from RWD upon ascertaining that the FORMAT statement is variable and, thus, has not been pretranslated by the compiler.

The XCV Routine skips characters in the input record and inserts blanks in the output record.

IOHXC. is the entry point for the FORMAT x specification.

The MAP calling sequence generated by the compiler for this routine is:

TSX IOHXC., 4
PZE number of characters

#### **Machine-Oriented Subroutines**

The routines that provide for machine indicator testing and traps at object time follow.

Note that, in these routines, I equals an integer indicating the number of the sense light or sense switch and J equals an integer variable to be set according to the results of the test.

The DCT Routine tests the divide-check indicator, sets J to 1 if it is on, and sets J to 2 if it is off.

DVCHK is the entry point for the following source program statement:

CALL DVCHK(J)

The FPT Routine: The AC and MQ are set according to the trap condition:

- 1. An AC and MQ overflow condition has occurred: the AC and MQ are set to all 1's.
- 2. An AC and MQ underflow condition has occurred: the AC and MQ are set to all 0's.
- 3. An MQ overflow condition has occurred: the AC is set equal to the old AC and the MQ is set to all 1's.
- 4. An MQ underflow condition has occurred: the AC is set equal to the old AC and the MQ is set to all 0's.
- 5. An AC overflow condition has occurred: the AC is set to all 1's and the MQ is set equal to the old MQ.
  - 6. An AC underflow condition has occurred: the AC

is set to all 0's and the MQ is set equal to the old MQ.

The number of floating-point traps allowed depends on three assembly parameters:

FPTLIM is the maximum number of overflow traps allowed.

ODDLIM is the maximum number of odd address traps allowed.

FPULIM is the maximum number of underflow traps allowed.

A separate count is kept for each type of trap condition. When the number of traps exceeds the maximum allowed, execution is terminated and the user is notified of excessive traps. Each of the assembly parameters, FPTLIM and FPULIM, is set equal to 10 in the distributed version. The parameter ODDLIM is set equal to 1.

The user has the option of having a message printed each time a trap occurs. The assembly parameter GETMES regulates this option. With GETMES SET 1, messages will be printed; with GETMES SET 0, messages will not be printed. The distributed version contains GETMES SET 1.

The following messages are possible:

1.	AC, MQ	OVERFLOW	AT	XXXXX
2.	AC, MQ	UNDERFLOW	AT	XXXXX
3.	MQ	OVERFLOW	AT	XXXXX
4.	MQ	UNDERFLOW	AT	XXXXX

5. AC OVERFLOW AT XXXXX

6. AC UNDERFLOW AT XXXXX
7. ODD ADDRESS TRAP AT XXXXX

The assembly parameter MESLIM indicates the maximum number of messages to be printed. In the distributed version, MESLIM equals 20.

SETFP. is the entry point for initialization of non-standard processing of floating-point traps.

The MAP calling sequence generated by the compiler for this routine is:

TSL SETFP

OVERF. is the storage cell for the last floating-point trap indicators; referenced by ovf.

FPT provides a routine to simulate double-precision operations. The inclusion of this routine is controlled by the following assembly parameter:

DPOPT SET x

where:

viiere:

=1 if the simulation routine is to be included =0 if the simulation routine is to be deleted

The distributed version of the FPT routine includes the simulation routine. If this simulation is unnecessary, the user may delete the routine by reassembling FPT with the proper setting of proper and editing the new FPT into the Subroutine Library.

The OVF Routine tests for floating-point overflow/ underflow, sets J to 1 if there is overflow, sets J to 3 if there is underflow, and sets J to 2 if neither condition exists.

OVERFL is the entry point for the following source program statement:

CALL OVERFL (J)

The SLN Routine turns on the FORTRAN sense light; it turns off all fortran sense lights if I = 0.

SLITE is the entry point for the following source program statement: reizant soll et enzents

SLITE (I) CALL

The SLT Routine tests FORTRAN Sense Light I, sets I to 1 if it is on, and sets I to 2 if it is OFF.

SLITET is the entry point for the following source program statement:

SLITET (I, J) CALL

The SWT Routine tests Sense Switch I and sets J to 1 if it is on, 2 if it is off.

SSWTCH is the entry point for the following source program statement: SSWTCH (I, J)

CALL

#### **System Communication Subroutines**

The following routines provide direct communication with the System Monitor and Processor Monitor:

The DMP Routine takes storage dumps according to specifications in arguments.

DUMP is the entry point for the following source program statement:

> CALL DUMP  $(a_1, b_1, f_1 \dots a_n, b_n, f_n)$

where:

a, b =variable data names indicating the limits of the core storage to be dumped.

=an integer indicating the dump format desired.

After dumping, control is returned to the System at the authorizable for helicity

PDUMP is the entry point for the following source program statement:

PDUMP  $(a_1, b_1, f_1 \dots a_n, b_n, f_n)$ CALL

where:

=variable data names indicating the limits of the core storage to be dumped.

=an integer indicating the dump format desired.

After dumping, core storage is restored and execu-

The PWS Routine types the contents of the address portion of the accumulator in octal and causes a pause during execution.

PAUSE. is the entry point called by the source program statement: PAUSE n

The MAP calling sequence generated by the compiler for this routine is:

\* \*\* TSL PAUSE.

The XEM Routine controls object program error procedure.

ERROU. is the entry point from an invalid Computed GO TO.

FEXEM. is the entry point from FORTRAN library subroutines upon detecting an object program error at execution time.

MATOP. is the programmed switch that allows bypassing the call to s.SDMP and resumption of execution in the following mathematical library subroutines:

XP1, XP2, XP3, XPN, ATN, LOG, SCN, ARSCN, SQR, FCXP, FCLG, FCSC, FDX1, FDX2, FDXP, FDLG, FDSC, FDSQ, FCA

The MATOP. switch is a control section.

sysop, is the programmed switch that allows bypassing the call to s.spmp and resumption of execution in the following system library subroutines:

IOS, RWB, RWD, ICV, LCV, DCV, SCA, SLN, SLT, SWT, UTV, BST, EFT, RWT, FPT

The sysop, switch is a control section.

If the optional return to execution has been chosen, the following message is written in the system output

ERROR CONDITION AT XXXXX FORTRAN ERROR XX IGNORED, RETURN TO EXECUTION

FORTRAN error messages that are written in the system output file indicate the location of the error in the object program. The following message appears in the system output file:

ERROR CONDITION AT xxxxx

where xxxxx is the address, in octal, of the core storage location that contains the detected error.

The XIT Routine returns control to the Processor

EXIT is the entry point for the following terminal source program statements:

PROGRAM IN WHICH THE STATEMENT STATEMENT OCCURS Main or subprogram STOP CALL EXIT Main or subprogram RETURN Main program

### **FORTRAN** File Routines

The following routines are used to generate the file control blocks for FORTRAN files. These routines are located in the Subroutine Library in the relocatable binary format.

ROUTINE DESCRIPTION F00 Contains the file text used by any source program input/output statement that refers to logical unit 0. FTC00. FILE U00, U00, BLOCK = 0257, SINGLE, TYPE3, LRL=256, ETC RCT = 001, ERR = RERRX., EOF = REOFX.,<sup>1</sup> ETC ETC ETC EOR = REORX., REEL FIL00.2 PZE FTC00.

<sup>1</sup>RERRX, and REORX, and REOFX, are control sections in the IOS routine. They are defined by an EXTERN in each file control block routine, <sup>2</sup>This is the entry point to which reference is made in the instruction sequences generated by the compiler. The entry point is defined by an ENTRY pseudo-operation.

	[일본 회사 골집 기계 전 기계 사고 있으면 기계 등 등 사람들은 기계
ROUTINE	
F01	Contains the file text used by any source program
17975318	ETCOL FILE HOLLIOLDI COTE COME
	FTC01, FILE U01, U01, BLOCK = 0257, ETC SINGLE TYPE3 I BI - 256
	ETC SINGLE, TYPE3, LRL=256, ETC RCT = 001, ERR = RERRX.,1 ETC EOF = REOFX.,1
	ETC EOF = REOFX., <sup>1</sup>
	EIC EOR = REORX, REEL
Пов	FIL01.º PZE FTC01.
	Contains the file text used by any source program input/output statement that refers to logical unit 2.
	FTC02. FILE U02, U02, BLOCK = 0257, ETC SINGLE, TYPE3, LRL=256,
	ETC SINGLE, TYPE3, LRL=256, ETC RCT = 001, ERR = RERRX., 1
	ETC EOF = $REOFX^{1}$
	ETC EOR = REORX., REEL
	FIL02.2 PZE FTC02.
F03	Contains the file text used by any source program
1.07	input/output statement that refers to logical unit 3. FTC03. FILE U03, U03, BLOCK = 0257,
	ETC SINGLE, TYPE3, LRI.=256
	$ETC EOF = REOFX.^{1}$
i dibakaran. Telebahan	DIO LON - NEORA., REEL
F04	FILO3. <sup>2</sup> PZE FTC03.
FU4	Contains the file text used by any source program input/output statement that refers to logical unit 4.
The Washing	FTC04. FILE U04 U04 BLOCK - 0957
	ETC SINGLE TYPE3 LBL = 256
	ETC RCT = $001$ ERR = RERRY 1
	ETC EOF = REOFX., <sup>1</sup> ETC EOR = REORX., <sup>1</sup> REEL
	FIL04.2 PZE FTC04.
F05	This routine is used to refer to the system input
	file control block. It is used by any source pro-
	gram input/output statement that refers to log
To the All	ical unit 5.
EOC	FIL05.º PZE S.FBIN
rub	This routine is used to refer to the system output file control block. It is used by any source
	program input/output statement that refers to
	logical unit 6
	FIL06. <sup>2</sup> PZE S.FBOU
F07	This routine is used to refer to the system periph-
	eral punch file control block. It is used by any
ili estad gle	source program input/output statement that refers to logical unit 7.
States	FILO7. <sup>2,3</sup> MZE S.FBPP, , S.FBOU
FRD	This routine is used to refer to the system input
	file control block. It is used by any source program
and a policy	input/output statement that refers to logical unit
3,660,48	READ.
er en sid (	FILRD. <sup>2</sup> PZE S.FBIN
FPR	
	file control block. It is used by any source program input/output statement that refers to logical unit
	PRINT.
	FILPR. <sup>2</sup> PZE S.FBOU
FPC	This routine is used to refer to the system periph-
	eral punch file control block. It is used by any
	source program input/output statement that refers
	to logical unit PUNCH. FILPC. <sup>2,8</sup> MZE S.FBPP, , S.FBOU
	S.F.BPP, , S.F.BOU
AND THE STREET, SALES	그 살아왔다. 그 그는 그는 그는 그는 그는 그는 살 수 있다. 이는 그 활동 원리를 가지 말해 보았다는데, 그는 그는

RERRX. and REORX. and REOFX are control sections in the IOS routine. They are defined by an EXTERN in each file control block routine. This is the entry point to which reference is made in the instruction sequences generated by the compiler. The entry point is defined by an ENTRY pseudo-operation.

The punch file pointers are indicated with the MZE prefix. This will indicate the use of S.FBOU (the system output file) when combined print/punch is involved.

**FORTRAN Routine Storage Requirements** 

The following chart shows octal and decimal storage requirements and dependencies for FORTRAN routines. Parentheses are used to indicate a non-fortran routine.

- w. O. M. 1100		RAGE	cate a non-Fortran routine.
	for the state of t	REMENTS	DEPENDENCIES—ROUTINES
ROUTINE	OCTAL	DECIMAL	
ACV	30	24	RWD, IOS
ARSCN	117	79	XEM, SQR, ATN
ATN	230	152	XEM
BST	126	86	XEM, XIT, (IOBS), IOS
CNSTNT	7	7	[[프로프로마 스타스 프랑스 스타스
DCT	26	22	
DCV	241	161	RWD, ECV, FFC, INTJ
DMP	1670	952	XIT, IOS
ECV	207	135	RWD, INTJ, FFC, IOS
EFT	33	27	IOS, XEM, (IOBS)
ERF F00	136	94	
F01	24 24	20	One 257-word buffer
F02	24 24	20 20	One 257-word buffer
F03	24	20 20	One 257-word buffer
F04	24 24	20 20	One 257-word buffer One 257-word buffer
F05	ar 📅 🖖	1	(System input file control
			block)
F06	1	1	(System output file control
			block)
F07	1	13	(System peripheral punch file
	Shurry (A		control block)
FRD	1	1	(System input file control
			block)
FPR	1	1	(System output file control
San Francis		11. Per 17 P.	block)
FPC	1	1	(System peripheral punch file
	en e		control block)
FCA	131	89	XEM
FCAB	44	36	SOR
FCLG	45	37	FCAB, ATN, LOG, XEM
FCSC	110	72	SCN, SCNH, XEM
FCSQ	55	45	SQR, FCAB
FCV	71	57	RWD, INTJ, FFC, IOS
FCXP	63	51	SCN, XPN, XEM
FDAT	231	153	XEM
FDLG	166	118	XEM
FDMD	60	48	
FDSC	215	141	XEM
FDSQ FDXP	75	61	XEM
FDX1	174 210	124	XEM
FDX2	72	136 58	XEM
FFC	367	247	FDXP, FDLG, XEM
FPT	350	232	INTJ, RWD, IOS
GAMA	245	165	XEM, XIT, (IOEX) XEM, XPN, LOG
GCV	23	. 19	ECV, FCV, RWD, FFC
HCV	103		RWD, IOS
IOS	267	183	XEM, XIT, (IOBS)
ICV	20	16	RWD, INTJ, IOS
INTJ	314	204	RWD, IOS
LCV	35	29	XEM, RWD, IOS
LOG	145	101	XEM .
MTN	20	16	
OCV	60	48	XEM, RWD, IOS
OVF	42	34	FPT
PWS	13	11	(IOEX)
RWB	411	265	IOS, XEM, (IOBS)
RWD	467	311	IOS, XEM. (IOBS)
RWT	17	15	IOS, XEM, (IOBS)
SCA	266	182	XEM, ACV, DCV, ECV,
		人名英格兰 医皮肤	FCV, GCV, HCV, FCV,
CONT	186	160 (34) 20	LCV, OCV, XCV, RWD
SCN	157	111	
SCNH	100	64	XEM, XPN

	STO	RAGE	ASSESSED CONTRACT FOR MITTHE		
Artematical De	REQUIR	EMENTS	DEPENDENCIES—ROUTINES		
ROUTINE	OCTAL	DECIMAL	REFERRED TO BY THIS ROUTINE		
SLI	25	21	XEM		
SLN	45	37	XEM		
SLO	25	21	XEM		
SLT	51	41	XEM		
SOR	100	64	XEM		
SWT	41	33	XEM		
TNCT	171	121	XEM		
TNH	76	62	XPN		
UTV	33	27	XEM, XIT, F00, F01, F02,		
			F03, F04, F05, F06, F07		
XCV	16	14	RWD, IOS		
XEM	223	147	(IBDMP), (IOEX), (IOBS)		
XIT	2	2	(IBJOB)		
XP1	73	59	XEM		
XP2	67	55	XEM		
XP3	51	41	XEM, XPN, LOG		
XPN	106	70	XEM		

#### Subroutine Error Exits

Error conditions in the object program that are discovered by the FORTRAN library will normally result in a call to IBDMP, which will write error messages on- and off-line and terminate execution. [On-line messages are listed, together with an explanation, in the publication IBM 7040/7044 Operating System (16/32K): Operator's Guide, Form C28-6338.] However, two core storage locations, used as switches, are provided to allow the IBDMP call to be bypassed and execution to be resumed in some fortran subroutines. These words, MATOP. and SYSOP., are assembled as control sections in the XEM routine. Bits 1-29 of MATOP. control error conditions 01-29 [described in the publication IBM 7040/7044 Operating System (16/32K): Subroutine Library (FORTRAN IV Mathematical Subroutines), Form C28-6806], while bits 1-29 of sysop. control error conditions 31-59. If the optional return for any of these error conditions is desired, the corresponding bit should be set to 1, either temporarily by replacing the control section with the desired bit configuration or permanently by reassembling the XEM routine.

# Input/Output Subroutine Error Exits

ROUTINE IN WHICH THE			
ERROR IS EN-	ERROR	The state of the s	OPTIONAL RETURN
UTV	32	Variable unit not defined.	To the XIT routine.
BST	33	Attempt to back- space past begin- ning-of-file.	To the XIT routine.
	59	Attempt to back- space the system output or punch file.	Ignore the operation.
EFT	34	Attempt to write file mark on the system input, out- put, or punch unit.	Ignore the operation.
RWT	35		Ignore the operation.

ROUTINE IN WHICH THE ERROR IS EN- COUNTERED	ERROR CODE		ERROR CONDITION	OPTIONAL RETURN
ios	36	1.	Attempt to write on the system input unit.	To the XIT routine.
	37	2.	Attempt to read the system out- put file.	To the XIT routine.
englik ka <mark>no</mark> Palitikan	38	3.	File mark reading.	Read the next file.
	49		Block sequence error.	Process the record read.
	50		Check sum error.	Process the record read.
	51	5.	Block sequence and check sum	Process the record read.
	52	e	errors. Permanent read	Process the
	40	Ų,	redundancy.	record read.
	53		Attempt to write on unopened file.	Return to IOBS for job termination.
	54	8.	Buffer overflow.	Write as much of the record as will fit in buffer.
	-55	9.	Error in IOBS Type 2 or 3	Set the word count to actual
			record control word.	number of words remaining in buffer.
	56	10.	Unexpected mode change.	Process the rec- ord that was read.
RWB	39	1.		Process the record read.
	40		Input list ex- ceeds FORTRAN record length.	Set the remaining list items to zero.
RWD		Ou	tput line has over- wed 1401 limit.	Continue to write on next line.

#### **Convert Subroutine Error Exits**

ROUTINE IN	441	v Bud au Georgi iz Seri	zeran unită oji ji e etta
WHICH THE	化 相归为	Specific process of the di	aprili (4) a
ERROR IS EN-	ERROR	医乳腺性 医电影 医二十二烷 医电影电影 经产品 化二甲烷二甲烷 经工作	gift in was some in the second
COUNTERED	CODE	CONDITION	OPTIONAL RETURN
ICV	42	Invalid input	Treat the charac-
		character.	ter as zero.
LCV	43	Invalid input	Treat the charac-
rational graphs w	s war f	character.	ter as blank.
OCV	44	Invalid input	Treat the charac-
		character.	ter as zero.
SCA	45	Invalid character	Treat the charac-
		in variable	ter as end of
	A Series	FORMAT.	format.

#### **Machine-Oriented Subroutine Error Exits**

ROUTINE IN WHICH THE	ing d	ing and the state of the state	
ERROR IS EN-	ERRO	R ERROR	ricegi (del ) Olimpia del del como
COUNTERED	CODE	CONDITION	OPTIONAL RETURN
FPT	31	Excessive floating	To the XIT
		point traps <sup>1</sup>	routine.
SLN	46	I>41	Take no action.
SLT	47	$I=0, I>4^1$	Set $J=2$ .
SWT	48	$I=0, I>6^1$	Set $J=2$ .
XEM	60	Illegal GO TO	Take first branch.

<sup>&</sup>lt;sup>1</sup>The maximum value is determined by an assembly parameter in the corresponding routine.

The cobol Compiler consists of four phases, as follows: Phase I, Language Reduction, reads the source program and reduces it to a concise form that can be handled easily by subsequent phases of the compiler.

Phase II, Syntax Analysis, reads the reduced—i.e., encoded—source program, and further reduces the statements in the Procedure Division to a simpler form suitable for processing by Phase IV. In addition, it provides special handling of PERFORM statements, and collects information concerning files and external names from portions of the Environment, Data, and Procedure Divisions.

Phase III, Data Reduction, has a twofold function. It generates in assembler-interface (T2) text, the necessary area-reservation and data-generation instructions for the entries in the Data Division, and creates the out-of-line 10cs linkages needed for each file described in the File Section. Also, it attaches to each data-name in the Procedure Division the characteristics described for it in the Data Division.

Phase IV, Procedure Generation, reads the Procedure Division statements, in reduced, internal-text form, and generates the appropriate instructions in T2 text for the Macro Assembly Program.

The first three phases are each divided into two parts, each part consituting a separate core storage load. Thus, the COBOL Compiler comprises seven distinct core storage loads. Further, part 1 of Phase I consists of three dependent links chained to a main link. Phase IV consists of two dependent links chained to a main link.

The compiler requires four tape units for compilation in addition to the minimum of three required concurrently by the Processor (s.slb1, s.sln1, and s.sou1). The assembler-interface input is written by Phase IV on one output tape.

Figure 36 shows the relationship of the phases, input/output unit allocation, and interphase data flow. Figure 37 shows the input/output unit allocation for compilations that involve recursive passes. A more detailed description of each of the phases follows.

## Phase I, Language Reduction

Phase I consists of two parts, each a separate core storage load: part 1, the Initial Edit; and part 2, Qualification Reduction. Phase I, part 1 initially receives control from s.sldr, as a result of a sibcbc card being recognized. It initializes all cobol system words, and then reads the source program, one card at a time. It

recognizes each of the four divisions of a COBOL program and interprets the contents of each appropriately, partially encoding all input statements. This includes a recognition of COBOL words, data-names, procedurenames, special-names, and file description. All source names at point-of-definition are encoded and placed in an external dictionary for use by Phase I, part 2. In addition to building the external dictionary and partially encoding the source program, part 1 produces the source-program listing on s.sou1.

Phase I, part 2 reads the partially encoded program on the main file and compares it with the external dictionary, assigning internal identifiers to the source names at point-of-usage. This may require more than one pass of the main file if the external dictionary created by part 1 was too large to be retained in core storage. In addition to the completely encoded main program, part 2 produces, on a separate file, an internal dictionary that contains a unique identifier and encoded data characteristics for every name in the Data Division. A glossary of source-program names with corresponding generated names is produced on s.sou1.

#### Phase II, Syntax Analysis

Phase II consists of two parts: (1) Syntax Analysis, and (2) Table Processing. Phase II, part 1 reads the main file written by Phase 1, part 2 and converts the internally coded Procedure Division statements into sequences of elementary operations in a form suitable for the individual verb analyzers of Phase IV. In addition, part 1 extracts, from the main file, information about each of the specified source-program files, all external names specified in control and file-refer-ENCE statements in the Environment Division, and external names following ENTRY statements in the Procedure Division. Also, information is collected from PERFORM statements for return linkages following procedures executed by these statements. This information is placed in a table that is retained in storage for analysis by Phase II, part 2.

All other main-file input read by Phase II, part 1 is placed, unchanged, into the main output file.

Phase II, part 2 processes the tables left in storage by part 1. The output of this part is placed on the miscellaneous file (an input to Phase III), and consists of the following types of records:

- 1. FILE pseudo-operation records
- 2. LABEL pseudo-operation records
- 3. PERFORM insertion records
- 4. CONTROL insertion records

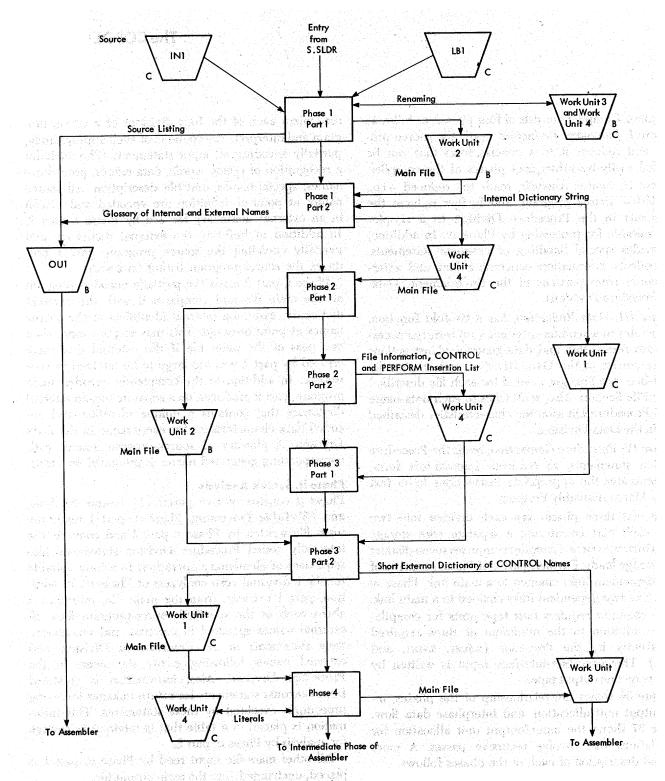


Figure 36. Input/Output Unit Allocation for COBOL Compiler

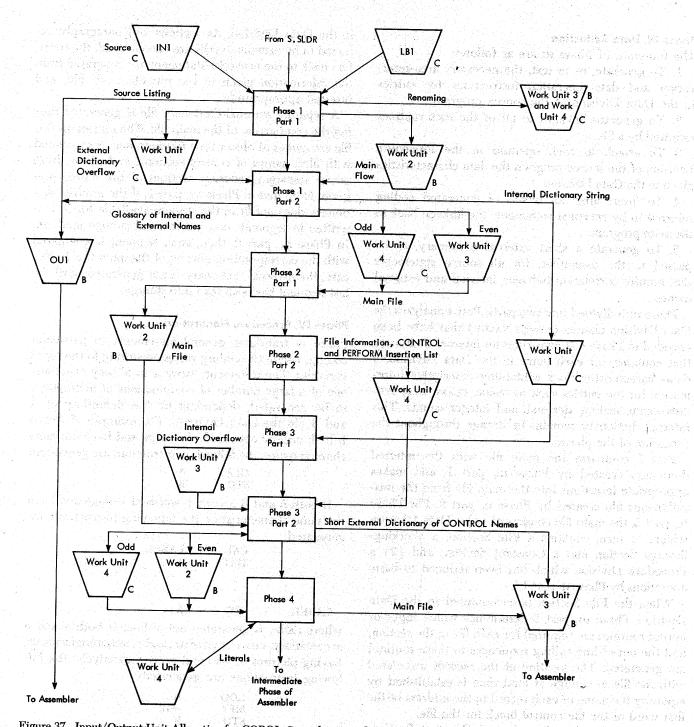


Figure 37. Input/Output Unit Allocation for COBOL Compilation with Recursive Passes

carried by definitions of how arms, left less, and respective

satification and anical relieved bodies and a supposed anical relieved anical relieved and a supposed anical relieved anical r

with the review transfer of the

They are estimated the continues of

arthur wer term of the for hornous divine will be received

The COBOL Compiler

### Phase III, Data Reduction

The functions of Phase III are as follows:

- 1. To generate, in T2 text, the necessary area-reservation and data-generation instructions for entries in the Data Division of the source program.
- 2. To generate linkages to all of the IOCs routines required by a file.
- 3. To attach to each operand in the Procedure Division of the source program the data characteristics given in the Data Division.
- 4. To insert after all blocks of generated coding referred to by PERFORM statements the linkage back to the main program.
- 5. To generate a short external dictionary, to be passed to the assembler, for all source statements that require correlation between internal and external names.

Phase III is divided into two parts. Part 1 analyzes the Data Division clauses (except VALUE) that have been encoded in Phase I. It constructs an internal dictionary that contains, for each entry in the Data Division, a data-characteristic block containing descriptive information for the entries such as USAGE, CLASS, SIZE, SYNCHRONIZED, scaling, decimal, and integer counts. This internal dictionary remains in storage throughout the execution of the phase.

Part 2 compares the main file with the internal dictionary created by Phase III, part 1, and makes appropriate insertions into the main file from the miscellaneous file created by Phase II, part 2. For Phase III, part 2, the main file consists of (1) a Data Division which, in turn, contains a File Section, a Working-Storage Section and a Constant Section; and (2) a Procedure Division which has been reduced to basic operations by Phase II, part 1.

When the File Section is encountered in the Data Division, Phase III, part 2 determines which input or output routines are required for each file in the section, and the out-of-line calling sequences to these routines are generated. The location of the records associated with the file in storage at load time is established by equating the name of each record to the address of the first word of the file control block for the file.

When the Working-Storage and Constant Sections are encountered in the Data Division, Phase III, part 2 determines, for each record, whether or not any VALUE clauses have been specified. If not, a single BSS, labeled with the record name, is generated having an operand computed from the total number of characters stated in the data-characteristic block. If any VALUE clauses are present, our pseudo-operations are generated to incorporate the constants into the area generated for the entry.

When the Procedure Division is encountered, each data-name has the characteristics that are defined for it

in the Data Division. As sections and paragraphs referred to by PERFORM verbs are encountered, the transfers back to the PERFORM statement are generated from the information given in the miscellaneous file, and inserted appropriately.

A separate external-dictionary file is generated during the comparison of the main file. The entries on this file are names of object-time routines, names associated with files, names of control sections, and names from ENTER ASSEMBLY-PROGRAM sections of the source program. Note that in Phase III, part 1, if the internal dictionary does not fit in the storage available for it, it is written in segments onto an external storage medium. In Phase III, part 2, then, each segment is compared with the corresponding portion of the main file. In this case, the external dictionary is not generated until the last segment has been read into storage.

### Phase IV, Procedure Generation

Phase IV translates general statements to particular coding, where the coding varies according to the input variables. The statement "MOVE A TO B" may cause any one of a large number of combinations of instructions to be generated, depending on the definitions of A and B (in the Data Division). For example, if A and B both occupy one word of storage and have the same characteristics, the following instructions are generated:

If both A and B occupy n words of storage and have the same characteristics, the following instructions are generated:

where LABEL is generated out-of-line. If both A and B are NUMERIC, COMPUTATIONAL, and SYNCHRONIZED RIGHT, having pictures of 99v9 and 99v99 respectively, the following instructions are generated:

The determination of the particular series of instructions to be generated is accomplished by an "analyzer." There is one analyzer for each type of statement possible (MOVE, ADD, IF, GO TO, etc.).

Another function of Phase IV is the optimization of generated coding. This involves keeping track of the contents of the index registers, arithmetic registers, temporary storage area, and subscript address words by associating the data-name with the status words for the register or area. For example, if the instruction LXA FIELD, 4 is generated, FIELD is associated with the

status words for index register 4. If an instruction is then generated that can change the contents of FIELD, such as sto field, then the status words for index register 4 would be cleared. All status words are cleared if a source procedure-name or compiler-generated procedure-name is encountered.

The use of the status words is illustrated by the following example. A NUMERIC DISPLAY field is moved to a NUMERIC COMPUTATIONAL field. Phase IV generates coding to convert the source field, saves the converted field in a temporary storage area, and moves it to the target field; in addition, it initiates the status words for the temporary storage area. If, subsequent to the move but prior to any procedure-name or coding that would change the contents of the NUMERIC DISPLAY field, that field is moved to another COMPUTATIONAL field, the generated coding will pick up the converted field from the temporary storage area. This eliminates a duplicate conversion. The relevant status words indicate that the converted field is in the temporary storage area.

The optimizing procedure cannot detect that the contents of a field are being changed when the contents are referred to by a different name, a situation that arises when REDEFINES is used. Consequently, caution should be exercised in the use of REDEFINES when numeric items are involved.

# **Special Operands**

There are two special types of operands which require separate discussions; these are subscript symbols and intermediate results.

## Subscript Symbols

A subscript symbol is used in order that a subscript may be extracted from the statement in which it appears and made into a separate statement. This method facilitates processing in both Phase II and Phase IV. The translation works as follows: consider the source statement "MOVE A(I, J) TO B." Phase II converts this to the two statements, (1) subs A, I, J, S. and (2) MOVE s, B. The SUBS analyzer processes the first statement and assigns data characteristics to the subscript symbol S. The recognition of the subscript symbol in the MOVE analyzer causes the characteristics assigned to it to be used in the resultant MOVE generation. year out that endough and their

### Intermediate Results

A complex statement is broken down to a series of simple statements. For example, COMPUTE A = B + C \* Dbecomes:

> MULT C BY D GIVING IR1 ADD IR1, B GIVING IR2 MOVE IR2 TO A

In the preceding example, IR1 and IR2 represent intermediate results. An intermediate result is a value in the AC, MQ, or AC-MQ, or in temporary storage.

The general format of an arithmetic operation is:

operation, operand 1, operand 2, operand 3

The operation is addition, subtraction, multiplication, division, or exponentiation; operands 1 and 2 are data names, literals, or previously defined intermediate results, and operand 3 is always an intermediate result. For the purpose of this discussion, the definitions of the operands are generalized as:

> operand 19(m) V 9(n)operand 29(r) V9(s) operand 3 9(x) V 9(y)

where x and y are a function of the operation and of m, n, r, s, and d (d is the maximum number of decimals used in the analyzed source statement up to and including the current operation).

The number of decimal places (y) in operand 3 is calculated as follows:

1. For addition or subtraction (operand 1 ± operand 2);

$$y = (n, s)_{max}$$

- 2. For multiplication (operand 1 \* operand 2):
- 3. For division within a compute statement or an if statement (operand 1/operand 2):

4. For division resulting from a DIVIDE statement:

$$y = t$$
 (if  $s + t - n \ge 0$ )  
 $y = n - s$  (if  $s + t - n < 0$ )

where t is the number of decimal places in the field to receive the quotient.

5. For exponentiation (operand 1 \*\* operand 2):  $y \neq n + 2$ 

The number of integer places (x) in operand 3 for double-precision operations (that is, operand 1 and/or operand 2 is double-precision) is calculated as follows:

1. For addition or subtraction:

$$\mathbf{x} = (\mathbf{m}, \mathbf{r})_{\text{max}} + 1$$

2. For multiplication:

$$-\hat{\mathbf{x}} = \mathbf{m} + \mathbf{r}$$

3. For division:

$$\mathbf{x} = \mathbf{m} + \mathbf{s}$$

4. For exponentiation with a literal exponent: x = p \* m

where p is the number of places in the exponent after it has been rounded to the next highest integer.

5. For exponentiation with a nonliteral exponent: x = r \* m

If x + y is greater than 20 and d is greater than y, then x is equated to 20 - v.

If x + y is greater than 20 and d is less than y, then y is equated to d. If x + d is greater than 20, then x is equated

The number of integer places (x) in operand 3 for single-precision operations (that is, both operand 1 and operand 2 are single-precision) is determined by calculating the largest value that the intermediate result can attain. This value is calculated by picking values for operands 1 and 2 and performing the relevant operation. The values are picked to maximize the result with the exception of an operand that is an intermediate result; the value calculated for an intermediate result when it was defined is used. The number of bytes required for the (binary) value is calculated and the maximum number of digits that can be expressed in that number of bytes minus the number of decimals is the number of integers.

#### **Processor Limitations**

The Processor limitations are as follows:

- 1. The size of all data fields (group or elementary) may not exceed 32,766 characters.
- 2. The number of files defined in a COBOL program may not exceed 50.
- 3. The number of PERFORM statements and/or external dictionary entries may not exceed 300.
- 4. A single OPEN statement may not contain more than 20 file-names.
- 5. A single GO TO . . . DEPENDING ON statement may not contain more than 95 procedure-names.
- 6. The maximum number of nested IF's is 102. However, when any of these ir's contain a logical operator (AND, OR), the maximum depends on a number of variables. These include:
  - a. The number of ELSE's
  - b. The number of AND's, not followed by or's
  - c. The number of on's, not followed by AND's
- 7. Error checking of the consistency between the names in the DATA-RECORDS clause and in the record descriptions under an FD entry takes place only if there are 50 or fewer data records for that FD entry.
- 8. Within the COMPUTE, IF, and the UNTIL portion of a PERFORM statement, 360 is the maximum number of elements that can be passed over before an operation within a formula can be evaluated. Due to the hierarchy of operations involved, the pattern in the first formula below would cause overflow at the 51st level of parenthesization and the pattern in the second formula would cause overflow at the 180th level:

When the number of elements exceeds 360, all subsequent elements will be deleted until any of the following are found:

- a. The end of the statement
- b. The words AND or OR, within the IF and the UNTIL portion of a PERFORM statement
- e. The word after, within the until portion of a PERFORM statement

NOTE: Within the IF and the UNTIL portion of the PERFORM statement, the levels of parenthesization are limited to 50. Within the COMPUTE statement there is no limit on parentheses.

9. Within a formula to be evaluated by the сом-PUTE, IF, and the UNTIL portion of a PERFORM statement, an additional limitation exists. If the elements to be passed over, as explained under Limitation 8, include literals or subscripted variables, overflow occurs when remarks as a firm a mach as the council paper

$$n=A$$
 $\sum_{n=1}^{\infty} (2 + Cn/6) + 2N + NN + 3S + D > 900$ 

where A = the number of alphanumeric literals

C = the number of characters in each alphanumeric literal N = the number of numeric literals, single or double-

precision, used as subscripts or not NN = the number of double-precision numeric literals

S = the number of subscripted variables

D = the number of data-names used as subscripts

When such overflow occurs, subsequent literals and subscripted variables will be deleted. Normal processing will be resumed when the hierarchy of operation allows some portion of the formula containing subscripts or literals to be evaluated.

- 10. The maximum number of times an area may be redefined, when the redefining entry is a group item, is 100.
- 11. The maximum number of redefinitions on level 01 and/or level 77 is 100.
- 12. The maximum number of Data Division entries per data record is about 1,110. The maximum depends on a number of variables, such as the number and length of error messages and report field pictures, and the number of redefinitions and data items subscripted.
- 13. The area of core storage allocated for tables sets the following limitations on SELECT entries with the da mid hijibi RENAMING option:
  - a. A minimum of 10 select statements with RENAMING. This limit assumes that both the renamed and renaming file-names are of maximum source length, and that each sense switch available to the user has had both its on and off status symbolically defined.
  - b. A maximum of 45 select statements with RENAMING. This limit assumes that both the renamed and renaming file-names are of minimum source length, one to six bytes. This limit also assumes that the user does not use the sense switches at the source level.
- 14. The maximum number of elements allowed for a Data Division entry is 54. The following example has four distinct elements:

02A	PICTU	RE :	IS	9(6)
	$\sim$		~	~~
1	2:	304	3	4

The number of elements is arrived at by counting the source name and its level number as one element, each of the cobol words as one element, and the predicate of the PICTURE clause as one element. Literals also count as one element.

15. The maximum number of temporary storage areas that may be in use at one time is 50. Moreover, if a field that is to be saved is larger than any of the fields already in the temporary storage area and there are more than 25 fields in this area, the compilation will terminate with halt cBC070. Temporary storage areas are required by the Processor as follows:

- a. For saving partial results of a complex computation.
- b. For saving the converted form of a field when conversion is required.
- c. For saving registers that contain a field that is an operand of an operation, where one operand is neither computational nor syn-CHRONIZED RIGHT, and has been scaled; or the other operand is neither COMPUTATIONAL nor synchronized right, and requires scaling.

16. The maximum number of invalid cobol words or names in a Data Division entry is approximately 25. The maximum depends on several variables such as the number and length of the error messages generated, the appearance of data-names interspersed with the invalid cobol words, and the number of correct clauses already processed for the entry.

17. Due to the manner in which file-names and endof-file labels are constructed, a deck name consisting of three letters followed by three numbers (e.g., ABC001) may produce duplicate control sections. The value of the numeric portion of the deck name must be greater than three times the number of files plus the number of use after error procedure statements.

18. The size of the COBOL Compiler restricts the use of optional features of the 7040/7044 Operating System for a machine with 16K core storage. After editing the system, the user should examine the storage map of the COBOL phases for unused core to determine whether there is enough space to allow the addition of optional features.

#### Input/Output Unit Allocation

A minimum of seven input/output units is required by the Processor during a cobol compilation. Three of these are the system units, s.slbi, s.sini, and s.soui, required by the Processor Monitor. The other four are utility units used internally by the compiler, one of these is used for the Phase IV output file, which contains the T2-text input to the Macro Assembly Program. (In an application that includes compilation, loading, and execution, the Processor uses an additional utility unit

for the load file required by the Loader.) The over-all allocation of the required seven units is shown in Figure 36 and is described in the following paragraphs. Figure 37 shows the unit allocation for compilation of a large source program requiring recursive passes of the compiler phases.

Phase I, Part 1 — Initial Edit The source program is read from s.sin1 and a source program listing is produced on s.sou1. The partially processed source program (main file) is written on WORK2. An external dictionary is created in storage; a dictionary segment is written on work1 at the end of the pass and each time the dictionary exceeds storage. Nothing is written on works if the external dictionary does not exceed storage. works and works may be used internally for RENAMING processing.

# Phase I, Part 2 — Qualification Reduction

The main file is read from works and, if the external dictionary overflowed in part 1, a segment of the external dictionary is read from works. The main file is compared with the external dictionary and the output is written on work4. If more than one segment of the external dictionary is involved, this pass will recur, the main file being read from and written onto works and WORK4 alternately for each segment. During the last pass (the only pass if only one segment of the external dictionary was involved) an internal dictionary string is written on works. A glossary of equivalences of internal and external names is written on s.sou1.

# Phase II, Part 1 — Syntax Analysis

The main file is read from work4. Processed output is written on works. Tables are created for PERFORM, CONTROL, and FD information.

#### Phase II, Part 2 — Table Processing

The FD information and the PERFORM and CONTROL tables from part 1 are processed. Output is written on WORK4.

# Phase III — Data Reduction

Part 1: The internal dictionary is read from WORK1 and an internal table of these entries is built. If the internal dictionary table area is exceeded, segments of the internal dictionary are written onto works until the complete internal-dictionary input string on work1 is processed.

Part 2: If the table area is not exceeded in part 1, the main file is read from works and a new main file is produced on works. Updated FD information from WORK4 constitutes the first records in the new main file. Approximate insertions from works are made in the new main file for PERFORM and CONTROL statements. In addition, a new external dictionary is created on

WORK3, consisting of any external names of CONTROL entries that may have been included in the source program.

If overflow occurred in part 1, the main file is read from work2 and compared with the first internal dictionary segment, the new main file being written on work1. Subsequent passes will cause the main file to be read from and written onto work1 and work2 alternately for each internal dictionary segment. During the last pass, the external dictionary is written on work3.

#### Phase IV — Procedure Generation

The main file is read from WORK1 and WORK2, depending upon the number of recursive passes, if any, in Phase III. A new main file is written on WORK3 so that it follows the external dictionary. The new main file becomes the input to the Interface section of the Macro Assembly Program. WORK4 may be used internally for long-literal processing.

# Files Assigned to System Units

If the system input unit, system output unit, or system peripheral punch unit (IN, OU, or PP) is specified for a file, in an Assign to clause in the Environment Division, the Input, Output, or Punch Editor, respectively, is automatically loaded from the library tape, at load time, together with the object program. All references to these files by input/output verbs are made through the appropriate editor. It is the user's responsibility to ensure that the records described and the operations specified for one of these files do not contradict the system and installation file specifications. These specications determine whether the file is input or output and determine its blocking, logical-record length, mode, opening/closing characteristic, and labeling. The compiler effectively ignores any conflicting specifications in the FD entry for the file in the Data Division, any references to the file in the Environment Division (other than the Assign to clause), and any input/ output statements in the Procedure Division which are contrary to the file's specifications.

The files created on or for the system units are always Type 3 when the system units are not assigned to unitrecord equipment.

# System Input Unit

Records read from the system input unit (s.sin1) are available in the buffer provided by the Input Editor. All references to records in this file are to the record in this buffer area. The user must determine whether the current record is BCD or binary.

The object program will execute the AT END portion of the READ statement for s.sin1 whenever \$ appears as the first character of a physical record on this unit.

Normally, this signifies the appearance of a system control card on the system input file. If, however, it is possible for the user's file to legitimately contain records with \$ as the first character of the record, the user must determine, by interrogating other fields in the record, whether this is one of his records, or a system control card. In the former case, more READ statements may be executed for the file. In the latter, a CLOSE statement must be executed. This will cause the last card to be the next card available to the system.

For a file on s.sin1, processing of the file should be terminated on the basis of an AT END indication, rather than on the basis of having processed the last input record.

# System Output Unit

Records for a file assigned to the system output unit (s.sou1) are processed through the Output Editor. The compiler generates a single 22-word work area for all records of this file. All references to fields within these records result in references to locations in this work area. A write statement causes the Output Editor to move the record from the work area to the output buffer. The record is never available in the buffer. Records for a file on s.sou1 must be BCD and are assumed to be intended for listing.

# System Peripheral Punch Unit

Records for a file assigned to the system peripheral punch unit (s.spp1) are processed through the Punch Editor. The compiler generates a single 28-word work area for all records of this file. All references to fields within these records result in references to locations in the work area. A write statement causes the Punch Editor to move the record from the work area to an output buffer. The record is never available in the output buffer.

It is possible to intermix BCD and binary records on a s.SPP1 file. An indication is given to the Punch Editor that a logical record is binary if at least one USACE IS COMPUTATIONAL clause appears in the description of the record under the FD entry for the file. In the absence of a COMPUTATIONAL (i.e., binary) field in the record, it is assumed that the record is to be punched in BCD mode.

#### **COBOL Library Subroutines**

The 7040/7044 cobol Compiler is designed to minimize storage requirements by using a number of common subroutines for character handling and parameter analysis. In all subroutine calling sequences that include an address reference, a standard cobol address parameter is used. This address parameter takes one of the following forms:

motion of the Mark to Michigan

PZE location,, byte

The above form is used when the data item is in working storage.

# MZE bl,,sp+nnn

The above form is used when the data item is located by a base locator (base=first word of the record in the buffer). bl is the address of the base locator (first word of the file control block). sp+nnn is the address of the word containing the item's relative location from the base (word displacement in the address, byte displacement in the decrement).

#### MON pi

The above form is used when the data item is located by a positional indicator, which contains the effective address of a subscripted variable. pi is the reference to the particular indicator. reformation to the San Re.

PON no. of characters,, character

The above form is used in MOVE ALL "character" to field. The actual character to be moved is in byte 2 of the parameter.

#### PON character

The above form is used to compare each ALL "character." The actual character is in byte 5 of the address parameter.

The evaluation of the effective address for an item located by a base locator or a positional indicator is done by the locator analyzing subroutine, CBLAN. This routine is entered by a TSL CBLAN with the address parameter in the AC. When CBLAN returns to the calling subroutine, the AC contains the effective address and byte of the data item.

The common character-handling subroutines are PCSTAB, CCSTAB, and SACTAB. These are used, for example, by the compare, move, and examine subroutines, and assume initialization of a work area and an index register to point to the character to be handled. PCSTAB assumes that a work area, CAFLD, contains the address of the word containing the character to be obtained and that index register 1 contains the byte number. CCSTAB and SACTAB assume that CBFLD and index register 2 contain the corresponding information. All three routines contain instructions to increment the word address and reinitialize the index register when necessary. In order to fetch a character, assuming all initialization has been done, the instruction used is:

#### XEC PCSTAB,1

CAFLD and CBFLD are among a group of one-word work areas defined as entry points within a separate deck in the library. These entries are referred to by many of the subroutines and, in addition, may be referred to by in-line coding.

Other items defined within the deck are:

C.AC an arithmetic subroutine source area C.MQ an arithmetic subroutine source area CACHLD an arithmetic subroutine work area CMOHLD an arithmetic subroutine work area TALLY

special register that may be altered by the EXAMINE verb or referred to by other source

language statements

WORK used as a switch by MOVPAK or C.COMP

the COBOL size-error switch C.OFLO COFLOW

a subroutine to set C.OFLO on and return to

the in-line routine

a subroutine to call S.SDMP if execution of a program containing a serious compilation

(source) error has been attempted.

The group of arithmetic areas above and the C.OFLO switch are referred to by most of the arithmetic subroutines. The c.oflo switch is tested to determine if a SIZE ERROR option is to be executed. It is set on for all arithmetic overflows and for floating-point errors (in exponentiation only).

#### **MOVPAK Routine**

C.ERR

MOVPAK is a generalized routine, or actually a group of subroutines, which is called upon by the object program for movement, conversion, and editing of data. The routine has four major entry points and numerous minor entry points. A TSX to one of the major entry points is always followed by one or more TXI's to minor entry points, except when moving figurative constants.

The following is a description of the calling sequences for the four major entry points in MOVPAK:

> C.MOVE.4 source address reference target address reference (begin specific move call)

The C.MOVE entry uses the source and target address reference information to set the contents of the CAFLD and CBFLD locations. Control is then transferred to the subsequent string of instructions to perform the move.

TSX C.MOV1,4 target address reference (begin specific move call)

The C.MOV1 entry is used when no source address is necessary, or when the source field is in an arithmetic register.

TSX C.MOV2.4 source address reference (begin specific move call)

The C.MOV2 entry is used when the resultant field is to be left in an arithmetic register.

> TSX **C.MOV3, 4** (begin specific move call)

The C.MOV3 entry is used when both the source and target addresses are set up in-line.

Following the TSX instruction and associated address parameter(s), the specific move call consists of one or more TXI instructions, with one exception, where only a control word is used (see "CHAN" below). Some of the calls are fixed length; other calls are terminated by a TXI instruction which transfers control to a particular location.

AN	Alphanumeric Field (ALPHABETIC, ALPHANUMERIC, non-report, or group item)
RP	Report Field (the PICTURE clause contains editing characters)
XD	External Decimal (NUMERIC DISPLAY)
<b>ID</b>	Internal Decimal (NUMERIC COMPUTATIONAL and SYNCHRONIZED RIGHT)
IN	Internal Decimal (not SYNCHRONIZED RIGHT)

The following are figurative constants:

SP	SPA	CES		Miya Sa si	with the	56
		ROS				
		racters (th				
	QU	OTE, HIC	H-VALU	JÉ, and I	LOW-V	/ALUÉ
مرابريا الألأور	con	stants)	A. Fana	d vice Alive. Se	in a great and	raliana.

Literals are classed as AN or ID, whichever is appropriate.

The following MOVPAK subroutine calls are given in alphabetical order by the abbreviated representations for the type of source and target fields. The order is AN, CH, ID, IN, RP, SP, XD, and ZE. For example, the move from Internal Decimal to External Decimal is designated by the letters mxp. Combinations other than those given are not permitted.

ANAN: Moves of an fields are handled by generated in-line instructions if the moves are simple, but more complex cases are handled by one of the following calls:

Move without trailing spaces —

TXI C.AN1, 1, number of characters to move

Move with trailing spaces—

TXI C.AN2, 1, number of characters to move

TXI C.AN3, 1, number of spaces to insert

ANID, ANIN, ANRP, ANXD: Substitute XD for AN; then see the equivalent section.

CHAN: Only the following control word is used for this operation:

PON number of characters, , character The actual character to be inserted is in byte 2 of the control word. erylyn Silling ol orders eryblini Cadiffe

IDAN: See "iDXD." IDID: MOVPAK is not used. In-line decimal-alignment instructions are generated as appropriate.

IDIN: (See "IDID" first.)

C.IDIN, 1, character length of

The source field is in the accumulator or in the AC-MQ.

Character length of target is the smallest multiple of six bits that is sufficient to contain the defined Internal Decimal field and its sign.

IDRP: (See "IDID" first.)

C.IDRP, 1, number of digits to

This instruction is followed by one or more instructions from the Report Field TXI instruction set described below. The particular instructions used reflect the characters constituting the field's PICTURE clause.

en vila hale da self-bye

The members of the Report Field instruction set are:

TXI	C.R999, 1, number of consecutive 9 occurrences
TXI	C.RZZZ, 1, number of consecutive Z occurrences
TXI	C.RAAA, 1, number of consecutive * occurrences
TXI	C.ROOO, 1, number of consecutive O occurrences
TXI	C.RBBB, 1, number of consecutive B occurrences
TXI	C.RSIN, 1, c1+64*c2

The character cl is inserted if the sign of the field is plus, and the character c2 is inserted if the sign of the field is minus.

# TXI C.RSIG, 1, c3+64\*c4

The character c4 is inserted if no preceding significant digit has been inserted, and the character c3 is inserted if a preceding significant digit has been inserted. If a TXI C.RFLs instruction (described next) has been executed and the floating sign has not yet been inserted, the character actually inserted is c4, c5, or c6.

#### TXI C.RFLS, 1, c5+64\*c6

c5 is the floating-sign character that is ultimately inserted if the sign of the field is plus, and c6 is ultimately inserted if the sign of the field is minus. If the first digit value is zero, a blank or the appropriate choice of c5 or c6 is immediately inserted as a result of this TXI instruction. Note that the first floating-sign position is traversed by this TXI instruction.

TXI C.RFFF, 1, number of consecutive floating-sign ocpalation of the contract of the currences

The above coding is used when other floating-sign positions are to follow.

TXI C.RFFQ, 1, number of consecutive floating-sign ocgrad file made of the browner

The above coding is used when no other floating-sign positions are to follow.

TXI C.REFC, 1, number of consecutive floating-sign occurrences

The above coding is used when no other floating-sign positions are to follow, but there is a comma before the next digit.

Values that c1 and c2 can assume as character pairs are:

cl: blank c2:

TXI

Values that c3 and c4 can assume as character pairs are:

c3: , , , , , c4: , blank \*

Values that c5 and c6 can assume as character pairs are:

c5: + blank \$ c6: - - \$

The report image TXI string is terminated by:

TXI C.RQIT, 1, value

where value = 0, unless the field is to be blank when zero, in which case value is equal to the total length of the target field.

Example 1: PICTURE IS \$\$\$,\$\$\$.99 is handled by:

TXI	C.RFLS, 1, c5=c6=\$ 43+64*43
TXI	C.RFFF, 1, 2
TXI	C.RSIG, 1, c3=, 59+64*48 c4=blank
TXI	C.RFFQ, 1, 3
TXI	C.RSIN, 1, $c1=c2=$ . 27+64*27
TXI	C.R999, 1, 2
TXI	C.RQIT, 1, 0

Example 2: PICTURE IS ZZZ,ZZZ.ZZ+ is handled by:

TXI	C.RZZZ, 1, 3	a sample will be
TXI	C.RSIG, 1,	c3=,
and the second second	59+64*48	c4=blank
TXI	C.RZZZ, 1, 3	
TXI	C.RSIN, 1,	c1 = c2 = .
	27+64*27	
TXI	C.R999, 1, 2	note choice of
		C.R999
TXI	C.RSIN, 1,	cl=+
1.4	16+64*32	c2=-
TXI	C.RQIT, 1, 11	note blank
	various billionistis.	when zero
	Committee of the state of	option

IDXD: (See "IDID" first.) The Internal Decimal contents of the accumulator or the AC-MQ are converted to External Decimal by one of the following calls:

TXI C.IDX1, 1, number of characters to develop

TXI C.IDX3, 1, number of characters to develop

where C.IDX3 is used if the target field is signed.

INAN: See "INID," then apply "IDAN."

INID: (Apply "non" after the following.)

TXI C.INID, 1, character length of source

The results are left in the AC or AC-MQ. Character length of source is the smallest multiple of six bits sufficient to contain the defined Internal Decimal field and its sign.

ININ: See "INID," then apply "IDIN." INRP: See "INID," then apply "IDRP."

INXD: See "INID," then apply "IDXD."

RPAN: See "ANAN."

SPAN: See "CHAN."

SPXD: See "CHAN."

XDAN: See "xDXD."

XDID: (Apply "IDID" after the following.)

TXI C.XDID, 1, number of characters to convert

This subroutine converts data from External Decimal to Internal Decimal and leaves the results in the accumulator or in the AC-MQ. The sign of the source field is assumed to be over the low-order digit. The absence of a sign is treated as denoting plus. Leading spaces appearing in the source field are treated as zero.

XDIN: See "XDID," then apply "IDIN." XDRP:

### TXI C.XDRP, 1, 0

This instruction is followed by one or more instructions from the External Decimal TXI instruction set (see "XDXD" for a definition of this set).

TXI C.XDRQ, 1, number of digits developed for target field

This instruction is preceded by the above-mentioned External Decimal TXI string and is followed by one or more instructions from the Report Field TXI instruction set (see "IDRP" for a definition of this set). As with IDRP, the calling sequence is terminated by TXI C. RQIT, 1, value.

XDXD:

TXI C.XDXD, 1, target sign con-

Target sign convention is 0 if the target field has no sign provision; it is 2 if the target field always has a sign over the low-order digit.

This instruction is followed by one or more instructions from the External Decimal TXI instruction set. The particular instructions used represent a procedural method for construction of the proper string of digits for the target field. The members of the set are as follows:

TXI	C. MOV, 1, number of digits to move	)
TXI	C.NZT, 1, number of digits to test for non-zero	)
TXI	C.BYP, 1, number of digits to bypass	
TXI	C.INZ, 1, number of digits to insert	
TXI	CXBND 1 0	

If non-zero significance is encountered under control of C.NZT, it causes C.OFLO to be set to non-zero.

C.XRND is used when rounding is desired at the current position.

Three alternate subroutine entry points, CXMVS, C.NZS, and C.BYS, correspond to C.MOV, C.NZT, and C.BYP. They are used instead when there may be a sign over the last digit treated by the instruction.

The instruction string is terminated for XDXD moves by:

TXI C.XDXO, 1, number of digits developed for tar-

Example 1: The statement COMPUTE A ROUNDED = B, ON SIZE ERROR... (where A'S PICTURE IS S999V999 and B'S PICTURE IS S9V9) results in:

TXI C.XDXD,	1, 2
TXI C.NZT, 1	
TXI C.MOV,	
TXI C.XRND, TXI C.BYS, 1	
-144	C.BYS for sign
	switch
TXI C.XDXQ	.1,2

Example 2: The statement MOVE A TO B (where A's PICTURE IS S9V99 and B'S PICTURE IS V9999) results in:

TXI	C.XDXD, 1, 0
TXI	C.BYS, 1, 1
TXI	C.MVS, 1, 2
TXI	C.INZ, 1, 2
TXI	C.XDXQ, 1, 4

ZEAN: See "CHAN."

ZEID: See "CHAN," then apply "INID."

ZEIN: See "CHAN."

ZERP: A sufficient number of zero digits is provided by generated in-line instructions which place zeros in one or more temporary storage words. The move is then performed as if it were XDRP.

ZEXD: See "CHAN."

# Compare Subroutine policy of the first of the confidence was a few

The linkage is:

C.COMP, 4 COBOL address parameter (first operand) COBOL address parameter (second operand) Length parameter Transfer parameter andresen and fridition

The length parameter takes the form:

bit 0-On if COLLATE MACHINE-SEQUENCE specified bit 1-On if first operand longer than second operand bit 2-On if second operand longer than first operand bits 3-17-Difference in length between first and second operand bits 21-35-Length of shorter operand

The transfer parameter takes the form:

transfer address pfx

where:

= PTW if testing for first equal to second operand

= MZE if testing for first greater than second operand

= PON if testing for first less than second operand

= MON if testing for first not equal to second operand

= PTH if testing for first not greater than second operand

= MTW if testing for first not less than second operand

#### Display Subroutine

Except for debugging sections, the linkage is:

TSX , C.TYPE, 4 COBOL address parameter Length parameter

If more than one area is to be typed, an additional pair of address and length parameters is used for each additional area.

The length parameter is of the form:

 $\mathbf{pfx}$ 

where:

pfx

= PZE if the item is not the last one to be typed

= MZE if the item is the last one to be typed

= the length of the item in characters

Within debugging sections, the linkage is:

C.DUMP, 4 Dump parameter COBOL address parameter Length parameter OUT. MOAD

The form of the length parameter is the same as described for the C.TYPE linkage.

The dump parameter is of the form:

pfx dump, , marker

where:

pfx

= MZE if a marker is to be written on a specified output file

= PZE if no marker is to be written

= 0 if the displayed information is to be written on the system output unit (S.SOUx)

≠ 0 if the displayed information is to be written on an alternate output unit (S.SUxx)

marker

= 0 if the requested marker is to be written on the system

 $\neq 0$  if the requested marker is to be written on an alternate output file

#### **Examine Subroutine**

The linkage is:

C. EXAM, 4 COBOL address parameter Examine parameter

The examine parameter is of the form:

bit 0 — On if TALLYING requested
bits 3-17 — The length in bytes of the field to be examined

bits 18-23 — The character to examine for

bits 24-29 - The character to replace with (if not replacing, the same as bits 18-23)

-On if a signed numeric display field is being exbit 32 amined for a numeric character (0-9)

bits 33-35 - An indication of the condition controlling the examination

=1s if REPLACING FIRST

=2s if LEADING

=3s if ALL

= 4s if UNTIL FIRST without REPLACING

=58 if UNTIL FIRST with REPLACING

This routine scans a field, counting or replacing characters as necessary.

#### Add Subroutine

# The linkages are:

TSX	C.ADD1,4	Add C.AC-C. MO
		to AC-MQ
TSX	C.ADD2,4	Add C.AC-C.MQ
		to AC
TSX	C.ADD3,4	
		AC-MQ
TSX	C.ADD4,4	Add C.MQ to AC
TSX	C.ADD5,4	Add double-preci-
		sion number to
		AC-MQ
COROL 3	ddress naramet	Ar .

L address parameter

C.ADD6,4 Add single-precision number to AC-MQ

COBOL address parameter

C.ADD7,4 Add double-precision number to AC COBOL address parameter

C.ADD8,4 Add single-precision number to AC

COBOL address parameter

The resulting sum is always placed in the AC-MQ, with the low-order portion in the MO.

#### Subtract Subroutine

#### The linkages are:

·		
TSX	C.SUB1,4	Subtract C.AC-C. MQ from AC-MO
TSX	C.SUB2,4	Subtract C.AC-C. MQ from AC
TSX	C.SUB3,4	Subtract C.MQ from AC
TSX	C.SUB4,4	Subtract C.MQ from AC
TSX	C.SUB5,4	Subtract double- precision number from AC-MO
COBOL a	address parame	ter
TSX		Subtract single-

precision number from AC-MO

COBOL address parameter

TSX C.SUB7.4 Subtract doubleprecision number

from AC COBOL address parameter

TSX C.SUB8,4 Subtract singleprecision number from AC

COBOL address parameter

The resulting difference is always placed in the AC-MQ with the low-order portion in the MQ.

### **Multiply Subroutine**

The linkage is:

TSX C.MPYn COBOL address parameter

where:

= 1 if the multiplicand is in the MQ and the multiplier is double precision

= 2 if the multiplicand is in the AC-MQ and the multiplier is double precision

= 3 if the multiplicand is in the AC-MQ and the multiplier is single precision

If the resultant product exceeds a double-precision number, it is saved in a four-word work area, C.MRES, and a switch, c.ofl2, is set. However, if c.ofl2 had been previously set (and not restored), the normal overflow switch, c.oflo, is set. If the resultant product is a single- or double-precision number, it is left in the AC-MQ.

#### **Divide Subroutine**

The linkage is:

TSX	C.DIV1,4 if AC-MQ/C.AC-C.MQ
TSX	C.DIV2,4 if AC-MQ/C.AC
TSX	C.DIV3,4 if MQ/C.AC-C.MQ

The resultant quotient only is returned in the AC-MQ, and the remainder is not retained.

# **Exponentiation Subroutine**

The linkage is:

C.XPab VFD 6/c, 6/d, 6/e, 18/0

where:

= N if the object computer has floating-point capabilities = F if the object computer lacks floating-point capabilities

= 1 if C.MQ \*\* AC= 2 if C.MO \*\* AC-MO = 3 if C.AC-C.MQ \*\* AC= 4 if C.AC-C.MQ \*\* AC-MQ

= the number of decimal positions desired in the result.

= the number of decimal positions desired in the base.

= the number of decimal positions desired in the exponent.

The exponentiation subroutine first scales the source fields, depending on the number of decimals, and then converts the input into double-precision floating-point form. The result is converted to fixed-point form and scaled to give the desired number of decimal positions. It is returned in the AC-MQ. In all cases, a maximum of 15 digits will have mathematical significance.

## **Round or Truncate Subroutine**

The linkage is:

AXT n,4 name where:

n

= the power of ten by which the AC-MQ or a four-word area, C.MRES, is to be divided (1<n<20).

name

= C.TRUN if the remainder of the division is to be ignored.

= C.RND if the remainder of the division is to be tested to determine if the quotient should be increased.

= C.TRN2 if the quadruple-precision result of a previous multiplication in C.MRES is to be scaled down.

The round or truncate subroutine is used to adjust the number of decimal positions in a number. The result is left in the AC-MQ.

ere freezement in the Leville of the State of

#### Scale Subroutine

The linkage is:

AXT n,4 TSL C,SCLx

where:

n

= the power of ten by which the MQ or AC-MQ is to be multiplied (1<n<20).

= 1 if the value is in the AC-MQ = 2 if the value is in the MQ only

The primary use of the scale subroutine is in adjusting the number of decimal positions in a number before using the number in another computation or as output. The result is left in the AC-MQ.

#### IF Alphabetic Subroutine

The linkage is:

TSX C.IFAL,4
COBOL address parameter
pfx transfer address,,n

where

pfx

= PZE for transfer if field is alphabetic

= MZE for transfer if field is not alphabetic

= the length, in bytes, of the field to be tested

This subroutine tests whether all characters of a field are alphabetic (including blanks).

#### IF Numeric Subroutine

The linkage is:

TSX C.IFNM,4
COBOL address parameter
pfx transfer address,,n

where:

υfx

= PZE for transfer if field is numeric

= MZE for transfer if field is not numeric

= the length, in bytes, of the field to be tested

This subroutine tests whether all characters of a field are numeric. If the field has a plus or minus sign on the last character, it is considered to be numeric.

#### **Format**

The Subroutine Library consists of two sections in the System Library located, in the distributed version, between Sections 1 and 2 of the Loader (IBLDR). The System Editor may be used to place one or more subroutines elsewhere in the System Library, such as on a separate library unit. The first section contains two lists:

The Subroutine Name Table (SRNT): This table is a list of all real control section names appearing in the Subroutine Library. Each control section name appearing in the list must be unique. Associated with each entry in the list is a number (the subroutine index) of the subroutine in which this control section appears and, if other routines are dependent upon this control section, a position in a second list (the Subroutine Dependence Table). The SRNT is in alphabetical order.

The Subroutine Dependence Table (SRDT): This table contains, for each entry in the Subroutine Name Table that has a dependency list, a list showing the control sections that must be loaded for execution of this control section. Therefore, a given control section is said to be dependent upon those control sections whose names appear in the corresponding portion of the dependence list.

The second section of the Subroutine Library contains the program decks for all library subroutines.

Figure 38 shows the format of the Subroutine Library. In the figure, IBLIB is the name of the header record, SZSRNT is the size of the Subroutine Name Table (SRNT), SZSRDT is the size of the Subroutine Dependence Table, SRNAD is the name of the SRNT/SRDT record, and IBREL is the name of the relocatable deck record.

The logical record of SRNT/SRDT contains the SRNT and SRDT tables.

In the SRNT, there is one entry for every real control section file and entry point in the Subroutine Library. The table is sorted alphabetically. Each entry consists of two words:

•		
BCI	1 .	
	1,1	ame
pfx	pt,	, srn

where:

name

=the name of the control section.

- =PZE if a dependency list is associated with this name and starts in the decrement.
- =PON if a dependency list is associated with this name and starts in the address.
- =MZE if no dependency list is associated with this name.

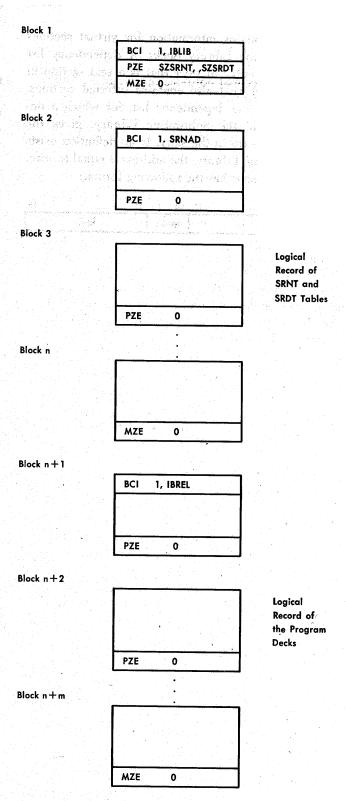


Figure 38. Format of the Subroutine Library

pt
= an address of a word in SRDT (relative to the start of that table) that contains the first word of the dependency list associated with this name.

srr

=the number of the subroutine in which the named section is a real control section.

The SRDT contains information for virtual sections in the Subroutine Library decks. A dependency list exists for each entry in SRNT that is a real section in a program deck that also contains external sections. Every entry in the dependency list, for which a definition exists in the Subroutine Library, gives the address of an entry in the SRNT. If no definition exists in the Subroutine Library, the address is equal to zero. Every word in SRDT has the following format:

S 2	3 17,	18 20 21	35
opi	k <sub>i</sub>	opi+1	ki+1

where:

op

- = PZE if this is the start of a dependency list.
- =PTW if the dependency list is continued.
- =MZE if this is the end of dependency list.

k

- $\neq 0$  if it is the location +1 (relative to the start of SRNT) that contains the definition of this section.
- = 0 if no definition exists in the Subroutine Library.

The logical record of program decks for the Subroutine Library consists of card image records of all decks in the Subroutine Library. Each deck starts with a SIBLDR record that contains the deck name in columns 8-13 and ends with a SDKEND record also containing the deck name. The blocking factor for the Subroutine Library is 10 records (card images) per block; the logical record size is 28 column binary words for the text and the dictionary of 14 alphameric code words for control cards. Each block, except for the last, has a final PZE word. The last block has a standard MZE flag word.

The Debugging Processor, which handles requests for load-time debugging, consists of three major parts: the preprocessor, the object-time routines, and the postprocessor. The preprocessor scans the SIBDBL card, compiles all of the debug requests, and creates the /DEBUG deck. The /DEBUG deck is loaded into upper core storage at object time to perform the debug requests. The postprocessor edits and translates the debugging dumps into a meaningful listing. Figure 39 shows the sequences in a generalized block diagram

# both the control of the best of the control of Preprocessor

The preprocessor is divided into two sections. Section 1 consists of the phases IBDLVA, IBDLVB, and IBDLVC. Section 2 consists of the phases DEFA and IBDEF.

भी करते हैं है है अपने अपने अने हैं है जिसके हैं है है

After all source decks in the IBJOB application have been processed, the SIBDBL card causes control to be transferred to the preprocessor. Input to section 1

(debug compiler) of the preprocessor comes from the system input unit. Output from section 1 consists of data to be used as input to the second section (creation of /DEBUG deck) and data to be written on the debug work unit (DWU) for use by the postprocessor.

The initialization section assigns table space and processes all of the information from the SIBDBL card. The main workload is handled by the debug compiler, which scans the debug source language statements, builds various tables to be left in core storage for section 2, and creates the debug request-point table. All counters and constants are initialized during this process. The spend card terminates this procedure.

At this point, a special ddict for the /DEBUG deck, a list of subroutines not necessary for this application, debugging text, and constants are written onto scri. The DDICT is created from symbols defined by =NEW in NAME statements. The messages encountered within quotation marks in LIST and DUMP statements are written onto the DWU. Control is then transferred to section 2.

Containing reference to their most engine for the

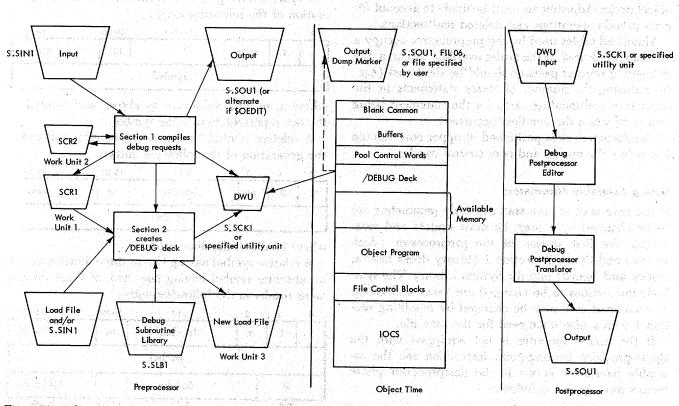


Figure 39. Debugging Processor

The function of section 2 of the preprocessor is to prepare a new load file and to prepare additional information for use by the postprocessor. Section 2 completes the tables built by section 1. It fills in information about names and values from the DDICTS, reading first from scri if there were symbols defined by =NEW, next from the load file, and then from any remaining binary decks that follow the debugging requests and precede a sentry card (or sendch if this is a chain job) on the system input unit. The poicts and end-of-link indicator records are copied onto the pwu. The object decks, followed by newly created /DEBUG decks (one for each link), are written in the new load file.

The /DEBUG deck is created from information left on scri and tables left in core storage by section 1 and from information contained in IBDINZ, which must be the first deck in the Debug Subroutine Library (IEDBG). IBDINZ contains all the object-time routines, each described as a control section. Any unnecessary sections are ignored; only those sections pertinent to the current run are used. via encarence bus stand frauncidati

Tables contain virtual references (plus adjustments) to deck name entries and real sections in object decks. This permits references within the debugging routines to decks that are being debugged. (The evaluation of virtual references to deck name entries made in the /DEBUG deck differs from that of similar references in object decks. Adjustments must be made to account for EVEN pseudo-operations and deleted real sections.)

Almost all tables used by the preprocessor occupy a common table area. If the tables overflow this area, the debugging request package should be shortened (e.g., by reducing the number of NAME statements or the number of continuation cards for the statement being processed when the overflow occurred).

The /DEBUG deck is positioned in upper core storage just below the buffers and pool control words.

# **Debug Assembly Parameters**

If the TRAP MAX or LINE MAX assembly parameters are to be changed by a user, he must update and reassemble the first section of the preprocessor (deck IBDLVA) and then edit section 1 (binary decks IBDLVA, IBDLVB, and IBDLVC) into the System Library. The symbolic instructions to be changed are TMAX and LMAX. The assumed DWU may be changed by re-editing section 1 with a new SFILE card for the DWU file.

If the user's computer is not equipped with the single-precision floating-point instruction set, the assembly parameter fltsw in the postprocessor phase IBTRAN must be set as follows: Appliford Salato

The phase must then be reassembled and edited into the System Library.

#### Format of the DDICT

The debugging dictionary, requested by either DD or SDD on the appropriate \$ card, is always headed by a SDDICT card. The rest of the dictionary consists of binary cards having two words for control and checksum information (see "Binary Card Format") and 22 words for dictionary entries. A dictionary entry may be contained on more than one card. The debugging dictionary follows the control dictionary in IBMAP output.

Each entry in DDICT is from one through three words in length. Mode changes that occur at a location for which there is no associated symbol will cause a oneword entry in the following form to be placed in DDICT:

0 x m	0 m	value
0 1 2	3-17 18-2	0 21-35

The x will be 0 if the value is relative, 1 if it is absolute. The m is the four-bit mode designator. The m, in octal, may be one of the following:

ense careful ad he share a motor

00 ≠ octal

01 = symbolic

04 = logical

05 = fixed-point

06 = alphameric

07 = floating point 16 = double-precision

17 = complex

A symbol having no dimensions will cause the generation of the following entry:

8 v	0 1	2 3-17	18-20	21-35
i.	1 x	m 0	m	value
	, and	symbol		

where x, m, and value are as above, and symbol is the BCD representation of the symbol.

A relative symbol having one dimension will cause the generation of the following entry:

	0	1	2 🐣	3-17		18-20	Consultation	21-35
	1	0	m	1st dim		m		value
1				symbo	ol			

where 1st dim is the dimension.

A relative symbol having two or three dimensions, or an absolute symbol having one, two, or three dimensions results in the following entry:

0 1	2	3-17	18-20	21-35
1 1	m	1st dim	m	value
	A sind to see	symbol		
0		2nd dim	x	3rd dim
0-2		3-17	18-20	21-35

where x is 0 if the value is relative, 1 if absolute. The 3rd (and 2nd) dimension will be zero if not applicable.

# **Object-Time Routines**

The object-time routines, along with tables and work areas generated by the preprocessor, make up the /DEBUG deck. The routines necessary for object-time debugging are an initialization routine, the STR supervisor that determines whether or not an STR is the result of a debugging request, the interpretive routines that process the compiled debugging requests and perform the desired action, and a routine that executes the instruction overlaid by the STR.

Control is transferred to the /DEBUG initialization section immediately after the loading of each link. Besides initializing the /DEBUG deck, this section saves the instructions at debug request points, inserts strainstructions, and writes information on the second file of the DWU. Control is transferred to the object program.

Upon executing an STR instruction, the STR supervisor is entered. It saves all registers and machine conditions that may be destroyed by the interpretive routines and determines whether or not debugging has been requested at this location. If debugging has been requested, control is transferred to the debug request interpreter; otherwise, a terminal dump is taken.

The debug request interpreter interprets computational, testing, and dumping instructions as supplied to it by the debug compiler in the preprocessor. Since a wider range of functions may be performed by the interpreter than is required for a particular execution, the interpretive functions are coded in a modular fashion. All functions are represented by control sections, and unneeded sections have been deleted by the preprocessor.

Output information is written on the debug work unit using one of several logical records to describe the dump(s). When the debug request(s) has been satisfied, control is passed to a routine that restores registers and machine conditions, executes the instruction overlaid by the STR, and returns to the object program.

where the first and the following the below the second the first first file.

A terminal dump is taken if the instruction overlaid by the STR was an STR or if it was an XEC referring to an STR.

# **Postprocessor**

The postprocessor, divided into the editing and translating sections, produces listings of object-time dumps from information previously written on the DWU by the preprocessor, by IBLDR, and by the object-time routines.

The editing phase transforms information from the first file of the DWU to appropriate internal tabular formats. The tables contain comment strings and assigned information. The debugging dictionaries are matched against the absolute assignment form of the control dictionaries to produce final absolute values. Unnecessary entries are deleted to conserve space. The generated tables are left in core storage for use by the translator phase. If table overflow occurs during post-processing, the size and/or number of DDICTS should be reduced.

The translator sequentially processes the table and object-time dump records from the DWU to create the full debugging listing that appears on the system output file.

These dumps appear in the same order as the STR instructions executed at debugging request points. The dumps are identified by a number that corresponds to the object-time marker written on the output unit (if specified on the SIBDEL card). Each dump has an identification line indicating the dump number, the symbolic request point, the deck name, and the relative and absolute locations of the request point. Following this, the dump is listed in the format specified for it. If no mode has been specified (in the object-time output or in a debugging dictionary) for the locations dumped, they appear in octal mode.

At the end of the translator phase, control is returned to ibsup by way of s.sret.

# Appendix A. System Input And Output Editors

The system Input and Output Editors are used by several system programs to read from the system input unit (s.sinx) and to write on the system output unit (s.soux) and the system peripheral punch unit (s.sppx). Both of these editors are used by the System Editor (IBEDT); the system Output Editor is used by the Dump program. The routines are assembled into each system program that uses them, they are not available to the user in the Subroutine Library.

The system Input and Output Editors call 100P for input and output operations and 10Ls for end-of-reel procedures. Thus, the system programs incorporating the editors may overlay 10BS.

These editors can read or write Type 1 records on unit record equipment and Type 3 records on any device on which a blocked record can be written.

an into the task party for them the Big

# **Input Editor**

The calling sequence for the Input Editor is:

TSX SYSIN,4

(return)

On the first call to the sysin routine, the routine initializes itself to read Type 1 records from unit record equipment or Type 3 records from devices that can be blocked.

Upon return from the sysin routine, the accumulator contains:

pfx

fword, , wdcnt 

where:

=PZE if a BCD card with other than a \$ in column 1 was read

=PON if a BCD card with a \$ in column 1 was read

=MZE if a binary card was read

is the location of the first word of the card.

wdcnt (the word count)

is the number of words in the card.

The calling sequence used to close the Input Editor is:

TSL SYS52 BRN SYSFR **MSP** SYSIN+3

If a \$ control card was read ahead, it is placed in s.save, and s.scoi is set on.

If a file mark is detected while reading from unit record equipment, the unit record equipment is immediately reselected. If a file mark is detected while reading a file that can be labeled, the Input/Output Labeling System is called for trailer label verification. If a 1EOF trailer label is read, the following message is typed:

10544 1EOF S.SIN

A SSTOP card is then simulated, and control is transferred to s.sret.

Block Length: The block length for input files is preassembled as:

140 SYBRL EOU

Up to five column binary cards, or nine BCD cards may appear in one block.

The first block control character in each block is examined to determine the mode for the next block. Subsequent block control characters are ignored.

Error Conditions: If an error occurs while reading from s.sin1, the following message is typed together with the first six words of the block that was read in error:

S.SIN ERROR, (six words)

The operator may press start to continue processing, ignoring the error. No error indication is given to the calling program.

If a device is not attached to s.sini, the following message is typed:

10545 S.SIN NOT ATTACHED

A sstop card is then simulated, and control is transferred to s.sner.

# Output Editor

A privovala

The calling sequence for the Output Editor is:

SYSOU, 4 TSX pfx fword, wdent (return)

where:

=PZE if a line is to be printed, preceded by a single space (block control character 2)

=PON if a line is to be printed, preceded by a skip to a new page (block control character 2)

=PTW if a line is to be printed, preceded by a double space (block control character 6)

=PTH if a line is to be printed, preceded by a double space and followed by a double space

=MZE if a BCD card is to be punched (block control character 2)

=MON if a binary card is to be punched

=MTW if a new subheading line is to be used (block control character 2)

=MTH if the system Output Editor is to be closed (block control character 4)

is the location of the first word to be written out. wdent

is the word count. If wdent is 0, only spacing or deletion of the subheading line will occur; no printing or punching will take place.

On the first call to the sysou routine, the routine initializes itself to write Type 1 records on unit record equipment or Type 3 records on devices that can be

blocked. The page count and the line count are initialized from s.pcct. The calling program should initialize the page heading, which may be up to 22 words beginning at the location syship. The routine examines locations s.soul and s.sppl to determine if the print and punch output are combined.

If the end of medium is detected while writing on s.sou1 or s.spp1, the Input/Output Labeling System is called to switch units.

When the Output Editor is closed, the last block on s.sou1 and s.spp1 are truncated with the BCD look-ahead block control characters. The page count and the line count are saved in s.pccr.

Paging: The listing page length is assembled as:
SYPCT EQU 58

Pages are ejected and page numbers incremented on each call to eject a page, on each call to use a new subheading line, and each time a page overflow occurs, Page numbers and line counts appear in the address field of locations sys22+1 and sys14, respectively. These may be modified by the calling program.

Block Length: The block length for output files is preassembled as:

SYBLL EQU 150 SYBCL EQU 150

Up to seven print lines may be written on s.sou1; seven column binary card images or 13 BCD card images may be written on s.spp1. If the print and punch output are combined, up to five column binary card images or 11 BCD card images may be written on s.sou1.

Note that the first block on s.soul or s.sppl is always in BCD. If the first call for s.soul or s.sppl is for a column binary card image, a special three-word, Type 3, P-(ignore) mode switch block is written.

Error Conditions: If an error occurs while writing on s.soul or s.sppl, one of the following messages is typed together with the first six words of the block that follows the block written in error:

20543 S.SOU ERROR, (six words) 20543 S.SPP ERROR, (six words)

The operator may press START to continue. No error indication is given to the calling program.

If a device is not attached to s.soui, the following message is typed:

10545 S.SOU NOT ATTACHED

A strop card is then simulated, and control is transferred to s.sret.

# Appendix B. 7040/7044 - 1401 Auxiliary Programs

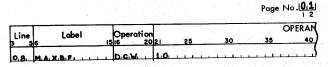
# Reassembly of the 7040/7044 – 1401 Auxiliary Programs

The 7040/7044 – 1401 auxiliary programs, as distributed, are assembled for an IBM 1401 Data Processing System with 4,000 positions of core storage. These programs may be easily reassembled for a 1401 with greater storage capacity, if such is available, to allow a larger maximum blocking factor for the tape files accepted or produced by the programs. Depending upon the blocking specified for the associated 7040/7044 system input/output files, this increased maximum blocking factor can result in greater operating efficiency for a particular 7040/7044 installation.

The reassembly for each of the programs involves an expansion of one or more input/output buffer areas and a change in the maximum blocking-factor constants. Specific procedures for each program follow.

# Input/Output Utility Program

The 1401 Autocoder statements which are to be changed in order to reassemble the 7040/7044 – 1401 Input/Output Utility Program for a larger-than-4K 1401 are specified in the distribution version of the program as follows:



The operand in columns 21 and 22 of this statement should be changed to a number which is the maximum BCD blocking factor allowed.

				Page No.IUI
Line	Label 6	Operation		OPERAN 30 35 40
_		DA.	1,x,9,00,, x,1,	

The size factor (900) in columns 23-25 of this statement defines a tape buffer area which will accommo-

date ten BCD card records (90 characters each), six BCD print records (138 characters each), or five binary card records (174 characters each). This factor can be increased to allow for as large a tape record as desired, provided that the reassembled object program does not exceed the storage capacity of the object machine.

# **MAP Symbolic Update Program**

The 1401 Autocoder statements that must be changed in order to reassemble the 7040/7044 – 1401 MAP Symbolic Update Program for a larger-than-4K 1401 are specified in the distributed version of the program as follows:

The state of the control of the cont	J
	Ĩ.,
OPERAN COPERAN	
Line Lobel Operation 5 15 16 20 21 25 30 35 40	)
	1
5.0, OUTPUT, DA 1x45.0	4
	ř.,

					Page No. 0.7
1		Label	Operation		OPERAN
	Line 3 5	6	1516 20	21 25 30	35 40
,	55.	M.I.N.P.U.T.	. D.A	1.x.4.5.0.	استنت

The size factor (450) in columns 23-25 of these statements defines a tape buffer area that can contain 5 BCD card records (90 characters each). This factor can be increased according to the size of the available 1401. Of course, this limit must not exceed the buffer size provided by the associated 7040/7044 Operating System (16/32K).

								Po	ge No. <b>[0,8</b> ]
-	Line	Label	15	Operation		25	30	35	OPERAN 40/
		H.K.C.O.N.F.			0.5			ببب	لبب

The operand in columns 21 and 22 of this statement should be changed to a number that is the maximum blocking factor allowed.

# **Appendix C. Symbolic Unit Assignment Subroutines**

# S.SCHI Subroutine

Calling Sequence:

CALL S.SCHI (p1, p2) CALL S.SCHA

Purpose: To search for the unit to which a given reservation code (in pl) is appended. The code in pl is to be expressed as a binary number. It should be used to find available units and units having intersystem reservation codes.

Output: p2 will contain, in bits 21-35, the machine address of the Symbolic Units Table entry corresponding to p1. Bits 3-17 of p2 will contain the address of the first word of the system control block for that unit.

If p1 contains zero, then p2 will point to the first available unit found.

A call to s.scha (without parameters) must not be the initial call to this routine. s.scha will search for the next entry in the Symbolic Units Table corresponding to p1. It will begin searching where the previous search left off and will exit with p2 set as above.

Not Found Condition: If an entry is not found with a code corresponding to that in p1, or if p1 is zero and an available unit is not found, p2 and the accumulator will contain zeros.

### **S.SCOD Subroutine**

Calling Sequence:

# CALL S.SCOD (p1)

Purpose: To encode a valid BCD unit specification so that it can be contained in an 18-bit field and used as input to subsequent subroutines.

Input: p1 and p1+1 (if necessary) contain the BCD unit specification, left-justified, with trailing blanks. p1+1 is not examined if p1 contains a blank.

Output: Bits 18-35 of the accumulator contain the encoded form. The bits are set as follows:

BITS	CONTENTS	MEANING
18-20	6	Symbolic unit
	4	Miscellaneous unit
	5	Variable unit

If bits 18-20 are set to 6 to specify a symbolic unit, bits 21-29 are as follows:

BITS	CONTENTS MEANING	
21-29	LBI	
	2 LB2	
	3 IN	
	<u>5</u>	

BITS	CONTENTS	MEANING
	11	CK1
	$100 + xx_8$	Uxx
	330+m	PRAm
	310+m	PUAm
	320+m	RDAm
and the second was to	360+m	PRSm
	340+m	PUSm
	350+m	RDSm

If bits 18-20 are set to 4 to specify a miscellaneous unit, bits 21-29 are as follows:

BITS	CONTENTS	MEANING
21-23	0	NONE
	1.47.1	* (in secondary only)
3 1 3 m	3	Iyy[R]
	4	LIN
	5	LOU
24-29	ENERGY PROPERTY	Not used if bits 21-23 specify NONE
shadished.	igenst specifieds	gor the state of the same in the same of the same
	уу <sub>8</sub>	lyy
	$24 + yy_8$	lyyn
	m+n	LIN or LOU
\$13. Egg (17)		m=0 for tape, disk, or drum
10 A 10 A	ar de de maid	m=20 for tape
		m=40 for disk or drum
		n=0 for any channel
		n=1 for channel A
The Debridges	Land St. Commission	n=2 for channel B n=3 for channel C
		n=3 for channel C n=4 for channel D
	di demonik sa	n=5 for channel E
e francisco Santonia		n=10 for channel V
	and the second	n=11 for channel W
		n=12 for channel X
	ashida islam	n=13 for channel Y
150 B. C. A	despiral visuas	n=14 for channel Z
		the state of the s

If bits 18-20 are set to 5 to specify a variable unit, bits 21-29 are as follows:

MEANING
Any unit
Tape
Disk or drum
Disk or drum arm or module m
(m=1=5)
Any channel
Channel V, index 1-10
Channel W, index 1-10
Channel X, index 1-10
Channel Y, index 1-10
Channel Z, index 1-10

The remaining bits indicate whether the unit is to be assigned a new intersystem reservation code. Bits 30-35 are set as follows:

BITS CONTENTS	YWH.	ME	ANING
30-35 0	No	intersystem	code
<b>yy</b> s	=1	уу	

Error Conditions: If p1 contains an unrecognizable set of characters, the AC will be zero upon return.

Note: It is the caller's responsibility to determine whether the field is valid in his case (e.g., the character \* is a valid specification to s.scod, but it may not be acceptable in the context of the caller's \*FILE card or FILE statement).

#### S.SCHL Subroutine

Calling Sequence:

CALL S.SCHL (r, lc, u) erret CALL S.SCHM (r, lh, v, lf) erret

The following applies when the calling sequence CALL S.SCHL is used.

Purpose: To search for an available input or output unit of the type and channel specified; and for input, having a label that matches a specified label control block.

Input: r contains, in bits 21-35, an encoded variable unit reference as output from s.scop. Note that this routine will assign intersystem reservation codes.

lc contains, in bits 21-35, the location of a nineword label control block that has labeling information for the unit being searched for. A label control block is required for both input and output units.

Output: u will contain, in bits 21-35, the address of the Symbolic Units Table entry for the first unit that satisfies the search specifications. If bits 30-35 of r are nonzero, and contain a value from 01-24, that value will be stored in the reservation code field of the system control block for which a match was found, provided that no other unit already has that reservation code.

Not Found Conditions: If there are no units available of the type and channel desired, u will contain zeros in bits 21-35, and its sign will be made minus.

If there are available units but none can be found that satisfy all the label search specifications, the operator is notified of the label being sought and of an available unit on which the file can be mounted. The operator has the choice of mounting the file on the specified unit, indicating an alternate unit to use, or terminating the job.

Error Conditions: Return will be made to erret if any errors are detected in the calling sequence, a reservation code greater than 24 (octal) is to be assigned, or a duplicate reservation code is to be assigned.

General: S.SCHL uses S.SVUN and S.SVUA to provide it with Symbolic Units Table entries. If labels are to be checked and IFSNS is set to 1, all units are initially rewound. After the label on each unit is checked, the unit is rewound again. (For output label searching, labels are checked only in a LABELS SET 2 SYSTEM.) S.SCHI is used to ensure that no duplicate reservation codes are assigned.

IOLS must be in core storage when s.SCHL is called.

The following applies when the calling sequence CALL S.SCHM is used.

Purpose: To search for an available input or output unit of the type and channel specified, and for input having a label that matches any one of a given list of label control blocks.

Input: r contains, in bits 21-35, an encoded variable unit reference of the type described as output from s.scop. It is equivalent to the input to s.schl.

lh is a list header word of the form PZE lt,, n where lt is a list of n label control block pointers:

4 4	100		Take The Supp
lt PZE	lcb1		9년(1월 년 년 1일) - 1일:
PZE			
PZE	lcb3	slddw.	de locali s
neit ent e	, history	精彩 惊点	State of the Control
P7E			

The label control blocks are required for both input and output units.

Output: u will contain in 21-35 the address of the Symbolic Units Table entry for the unit found, or zeros if no unit is available or if no matching label can be found.

If no units are available, the prefix will be MZE.

If no matching label is found, the prefix will be PZE.

If a unit is found, and bits 30-35 of r are 01-24, that value will be stored in the system control block for which a match was found, provided that no other unit already has that reservation code. If will contain, in positions 21-25, the location in the lt list of the pointer to the label control block for which the match was found.

Error Conditions: Return will be made to erret if any errors are detected in the calling sequence, a reservation code greater than 24 (octal) is to be assigned, or a duplicate reservation code is to be assigned.

General: When a label is read from a unit, the label will be matched against all the label control blocks specified by the list lh.

If the prefix of a list pointer is MZE, rather than PZE, that entry will be skipped. When a match is found, the pointer to the label control block giving the match will be placed in If. A label control block can be eliminated from subsequent searches by setting the pointer prefix to MZE.

IOLS must be in core storage when this routine is called.

Note that, unlike s.schl., no operator intervention is provided.

Subroutines Used: s.svun, s.svua, and s.schi are used by s.schm.

Output from S.SCHL and S.SCHM: Figure 40 illustrates the output of these calling sequences.

LABELS	S.SCHL	S.SCHM
=0 INPUT	operator option on first	none available
OUTPUT	first available	first available
=1 INPUT OUTPUT	first available that matches* first available	first available that matches first available
=2 INPUT	first available that matches* first available that has expired*	first available that matches first available that has expired

<sup>\*</sup>indicates operator option if no match or if label has not expired. erret is taken if one of the following conditions occurs:

reservatively of the resembled substituted adds that such add to

HOUSE HE SHE WITH SHEET TO SHEET

- 1. Encoding of r is in error.
- 2. A reservation code greater than 248 is to be assigned.
- 3. A duplicate reservation code is to be assigned.

Figure 40. Output from S.SCHL and S.SCHM

# S.SVUN Subroutine

Calling Sequence:

Santon Salt Japan Bolk Dala Hylano CALL S.SVUN (p1, p2) erret CALL S.SVUA, erret

Purpose: To find an available unit of the type, channel, and number specified.

Input: p1 contains, in bits 18-35, an encoded variable unit reference of the type generated as output by s.scop. Note that this routine will not assign an intersystem reservation code (it ignores the six low-order bits).

Output: p2 will contain, in bits 21-35, the machine address of the Symbolic Units Table entry for the unit.

Errors: If pl is encoded incorrectly, return is to erret. If no unit is available, p2 will contain zeros and a normal routine is taken.

Note: A call to s.svua will find the next available unit (after that found by s.svun) that meets the same specifications as those given for the previous unit. Output is the same as for s.svun and is placed in p2. This entry cannot be used if the previous unit specified was a symbolic arm or module, or if the previous call resulted in an error. han alministic all linear points

## S.SLDS Subroutine

This subroutine is loaded from the Subroutine Library by IBLDR when the mounting option is DEFER and label search is specified in the units field of the FILE statement or SFILE card. The location of the subroutine is stored in bits 3-17 of word FCFCN of the file control block. The subroutine is entered from iols when a file is opened or unit switched, and the primary or secondary unit field specifies that a label search is required.

The user cannot use s.slds, but he can cause iols to use it by putting its location in bits 3-17 of word LCFCN, and setting the appropriate codes in word LCUNI.

# Appendix D. Processing the \$CHANNEL Card (IMCHA Routine)

Using the information in the schannel card, the IMCHA routine establishes equivalences between the specified symbolic channels and the real channels. The IMCHA routine also establishes equivalences between any specified symbolic modules and the real disk or drum modules. The equivalences established must satisfy all requirements in the schannel card except the maximum tape or disk requirements; if such requirements are given, the equivalences must be the best possible set in terms of satisfying the maximum requirements. If the maximum tape or disk requirements cannot be met, the number of units available must at least be equal to the sum of all the units specified for all the symbolic channels.

Tables are used in choosing the best set of equivalences. The most important are the Totals Table, the Arm Table, the Array Table, and the Main Table.

The Totals Table contains entries indicating the total number of disk areas, drum areas and tape units available; the table also indicates the number of such units and areas requested.

The Arm Table contains an entry for each disk or drum module and indicates the number of areas available for each. Each entry in the Arm Table also contains a five-bit segment that is used in developing symbolic module equivalences.

The Array Table is a 20-word table in which symbolic module equivalences are developed for each possible set of symbolic channel equivalences. At the end of the processing done by the IMCHA routine, the appropriate words are moved from the Array Table into the symbolic channel table in IBNUC.

The Main Table is a matrix containing a row for each symbolic channel and a column for each real channel. Each element of the matrix is set to 63; the elements are then reduced by varying amounts as the corresponding equivalences between real and symbolic channels are found to be less than optimal. An element is set to zero only if the corresponding equivalence is impossible to establish.

At the beginning of the IMCHA routine, the hardware tables in IBNUC are scanned and the Totals Table and Arm Table are set up. Then the \*CHANNEL card is scanned and the various parameters are converted to entries in the tables.

If a given number of tape or disks units is required

for a particular symbolic channel, the Totals Table is scanned to determine which real channels have the required number of units available. If any real channel does not have the required number of units available, the element in the Main Table corresponding to that real channel and the given symbolic channel is set to zero.

If a maximum number of tape (or disk) units is specified for a given symbolic channel, the same check of the Totals Table is made. However, if a real channel is found that does not have the maximum number of units available, then the difference between the maximum specified and the number available is multiplied by the device bias factor, and the result is subtracted from the appropriate element in the Main Table. (This operation will never reduce an element to less than one.) If it is specified that a given symbolic channel must not be made equivalent to a particular real channel, the element of the Main Table corresponding to this equivalence is set to zero. If it is specified that a given symbolic channel must be equivalent to a particular real channel, the routine will not set the symbolic channel equivalent to any other real channel, and will not set any other symbolic channel equivalent to this real channel. Therefore, every element except the corresponding one in the Main Table is set to zero. If a system unit code or intersystem reservation code is used, the corresponding real channel is found and treated as above.

If one or more symbolic modules are specified for a given symbolic channel, the five-bit segments of the Arm Table entries are used as a five-by-n matrix, with the five columns corresponding to the five symbolic modules and the n rows corresponding to the n real modules. Essentially the elements of this matrix are given the same treatment as the elements of the Main Table, with respect to the specifications given for each symbolic module. The matrix is then divided into submatrixes, one per channel, and each is examined to see if a solution can be found. If a solution is found, it is transcribed into the Array Table for use if the corresponding equivalence is used. If no solution is found, the corresponding element of the Main Table is set to zero. When this process has been repeated for each of the submatrixes, there will be a mapping of symbolic modules to real modules for every equivalence of the current symbolic channel to a real channel, except those for which the corresponding elements in the Main Table are zero.

When the entire schannel card has been processed, the Main Table will contain nonzero elements only for usable equivalences. The table is then examined for

possible solutions. A solution must have a nonzero element for each specified symbolic channel (row), no two of which are on the same real channel (column). Each solution has a value consisting of the sum of its elements. The solution with the greatest value is used.

# Index

	16	ດ
Abbreviated Table of Contents	CYLMO	
IMEXE routine	DCT routine	3
location and length	DCV routine	1
accounting routine, installation	debug assembly parameters 9	14
ACV routine 71	Debug Subroutine Library	)4
add subroutine (COBOL)	debugging dictionary 9	4
AFTER card	Debugging Processor	13
ALLOW macro-instruction	deck name entry	1
alternate system input file	dependency list	1
alternate system input file	detach routine (IMDET)	17
assembly, system	device assignment	0
machine configuration 9	device assignment 1	0
machine configuration 10	device type numbers	0
input/output device assignment		.0
options	dictionary reduction	30
parameters	reduction	יטנ פנ
priority processing	reference addresses and decrements	).)   1
unit record interrupts	disk and drum formats	L.L
Subroutine Library options	disk and drum orders	80
System Editor options	display subroutine (COBOL)	38
ATTACH macro-instruction	divide subroutine (COBOL)	39
attaching input/output devices (IMATT)	DMP routine	74
auxiliary programs (1401)	DPOPT 7	73
available units	DUMP 7	74
AVLUNI 52	and Demonstrate the second	
backup library	program	50
backup library	routine	44
BCD card format	DUP card	26
binary card format	ECV routine	79
control dictionary		
program text, relocatable	edit file	==
blank COMMON	file	วก
Bootstrap routine	run for disk	30
CALLS card	runs, examples of	
CBAVL 52	editing instructions, initial	, r
chain program, inserting	instructions, initial	15
Change Communication Region routine	instructions, symbolic	30
\$CHANNEL card	Editor	
channel assignment, symbolic	components, system	17
checkpoint (S.NAPT and S.SFBL)	control cards	21
check sum	input	17
COBOL	messages	33
Compiler	Monitor	17
files assigned to system units	parameter cards	24
input/output unit allocation 83	program19,	28
library subroutines 84	EDTFIL option	23
processor limitations 82	\$ENDEDIT card	23
processor limitations	engineering words	36
special operands	EOMTR	12
communication subroutines	ERRIN	14
compare subroutine (COBOL)	Error Editor	
complex field decrements and addresses	error exits, subroutine (FORTRAN)	76
constant addresses and decrements	ERROU	14
constants (system data area)	ERRPP	
control	ETC card	24
blocks	even entries	61
cards, Editor	examine subroutines (COBOL)	88
cards, Editor parameter	exponentiation subroutine (COBOL)	20
dictionary, binary	exponentiation subroutine (CODOL)	Q1
group	external reference entries	70
control section entries	EXTRA	
convert subroutine error exits	FCV routine	72
copy and reload file, system	file	
copying (DUP Card)	control blocks	54
CORE option 23	entries	61
CPRPU 12	maintenance	56
Cross Reference Dictionary 69	text	62
Closs Reference Dictionary		

Control Addition and Sillie College of appropriate

flags, communication	ixed-point conversion 72	IMOID routine
Boating-point conversion   71, 72, MPAS routine	lags, communication 40	IMOPN routine
Format conversion routines	oating-point conversion 71.79	IMPAS routine
FORTRAN	ormat conversion routines 71	IMPAIL routine
Compiler   170   MFUN routine   Rile routines   174   MRCD routine   Rile routines   175   MREL   Subroutine error exists   176   MRES routine   177   MRED   Subroutine   177   RMRLD   Subroutine   177   RMRL	FORTRAN	IMPRE
file routines or yet requirements 75 bMREL subroutine error exists 76 bMREL subroutine error exists 76 bMREL subroutine error exists 76 bMRES routine yet with the subroutine error exists 76 bMRES routine 97 bMRLD subroutine error exists 76 bMRES routine 97 bMRCD routine 98 bMRC	Compiler 70	
Toutine storage requirements   75	file routines	
Submoutine error exis   76	routine storage requirements	IMPET
System routines		IMPEC
FPT routine		TART D
FFTLIM	PT routing	IMPLED 5
FPULIM	PTI IM	Mison routine 4
FTCENT   52   MSWI routine	PIH IM	TATOTO - 1
MSYS routine   CETMES   73   MTIT routine   MTIM routine   MTIM routine   MTIM routine   MTIM routine   MTIM routine   MUNI	TODINI	IMS 10 routine4
MTIF routine   MTIS		IMSW1 routine4
GETMES   73   IMTIM routine   12   IMTIS routine   12   IMTIS routine   13   IMTIS routine   13   IMTIM routine   14   IMTIS routine   15   IMTIS routine   16   IMTIS routine   17   IMTIS routine   18   IMTIS routine	FRP	IMSYS routine
IRC	ETMES 72	IM ITF routine
IBDINZ		IMTIM routine 4
IBDINZ	ICV routine 79	IMTIS routine 4
SIBEDT card	。1911年 - 1911年 - 1911	IMUNI routine
BIOB	BDINZ	IMUNL routine 4
BIOB	IBEDT card 21	IMUQI routine
IBLDR	BJOB51	INCLOS
IBLITZ	BLDR 50	INCLSM 5
Index	BLTZ	INCWD 1
BOOT		Index
IBPER	BOOT	Indexing Analyzer 7
IBRSL	BPER	initial editing instructions
IBRTAN	BRSL 10 12	initial system unit requirements
BTRAN	BRTZ 12,10	INJOB routine
Imput   Input   Inpu	BTRAN	Input Editor
EDBG	CV routine	input file, system
ELCT	EDBC 20 04	Input/Output Control System
ELFG	FI CT	Input/Output Subroutine Error Evits 7
INSAVE   INSERT card   Inserting a relocatable subroutine   installation accounting routine	FI FC	Input/Output Utility Program
INSERT card   INSTRUCTION Calculated installation accounting routine   Instruction Concerator and Index Generator   Instruction Concerator   Instruction   In	5D01	INCAVE
Inserting a relocatable subroutine   F Alphabetic subroutine   90	EDOO	INCERT
F. Alphabetic subroutine	2.NU3	Constitute a malacatalla colonia
F. Numeric subroutine   90	5W51	issisting a relocatable subroutine
Fife2	Alphabetic subroutine	Instanation accounting routine 4
FCHO	FIGO	integer conversion (ICV)
FCHT	FOUO 12	Interface
FCLK         11, 12         interrupts, unit record           FLSC         12         IQRSC routine           FRPM         11         IQRST routine           FSHF         12         JBKILL           FSNS         12         JBSCAN           FTYP         13         JOBERR           FURA         12, 13         JOBIN           MAST routine         45         JOBINM           MATT routine         47         JOBOUL           MCDC routine         47         JOBOUL           MCBC         52         JOBOUM           MCHA routine         49, 102         JOBPP           MCLO routine         47         LABEL (edit option)           MCHA routine         47         LABELS (assembly parameter)           MDLY routine         47         LABSW           MECH         52         LCV routine           MENT         52         LDAVL           MEXE routine         45         LDRSTK           MFTC         52         LIBE card           MINT routine         46         LIBEND card           MINT routine         46         LIBEND card           MILOR         52         LMAX	FOUT	interrupt test energies 67,6
FLSC 12 IQRSC routine FSPM 11 IQRST routine FSHF 12 JBKILL FSNS 12 JBSCAN FTYP 13 JOBERR FURA 12, 13 JOBERR FURA 12, 13 JOBIN MAST routine 45 JOBINM MAST routine 47 JOBOU MEDC routine 49 JOBOUL MCBC 52 JOBOUM MCHA routine 49, 102 JOBPP MCHA routine 49, 102 JOBPP MCHO routine 47 LABEL (edit option) MDBL 52 LABELS (assembly parameter) MDBT routine 47 LABSW MECH 52 LCV routine MENT 52 LCV routine MENT 52 LDAVL MEXE routine 45 LDRSTK MFTC 55 LIBE card MINT routine 46 LIBEND card MIDR 52 LMAX MLDR 52 LOAGER MLSV routine 48 LOAGER MLSV routine 48 LOAGER MLSV routine 48 LOAGER MLSV routine 48 LOAGER MLSV routine 47 IBLDR MAAP 55 System (S.S.I.DR)	FOLK	interrupte unit record
FRPM	TI CO	TORSC routing
TSHF   12	TDDM	TOPCT4
TTYP	CRPM 11	IQNOT routine
TTYP	'SHF 12	TRKILI
TYP	FSNS19	IBSCAN 55
MAST routine	TYP	IORERR
MAST routine         45         JOBINM           MATT routine         47         JOBOU           MBDC routine         49         JOBOUL           MCBC         52         JOBOUM           MCHA routine         49, 102         JOBPP           MCHN         52         LABEL (edit option)           MCLO routine         47         LABELS (assembly parameter)           MDET routine         47         Labels           MDLY routine         47         LABSW           MECH         47         LABSW           MECH         52         LCV routine           MENT         52         LDAVL           MEXE routine         45         LDRSTK           MFTC         52         LIBE card           MINT routine         46         LIBEND card           MILS routine         46         LISTSW           MIDR         52         LMAX           MLIS routine         48         load file, system           MLIS routine         47         IBLDR           MAP         52         System (S.SLDB)	CURA 12.13	TOPIN 5
MATT routine         47         JOBOU           MBDC routine         49         JOBOUL           MCBC         52         JOBOUM           MCHA routine         49, 102         JOBPP           MCHN         52         LABEL (edit option)           MCLO routine         47         LABELS (assembly parameter)           MDET routine         47         labels           MDLY routine         47         LABSW           MECH         52         LCV routine           MENT         52         LDAVL           MEXE routine         45         LDRSTK           MFTC         52         LIBE card           MINT routine         46         LIBEND card           MINT routine         46         LIBTSW           MJOB         52         LMAX           MLDR         52         load area           MLIS routine         48         load file, system           MLNK         52         Loader           MLNK         52         Loader           MLNY         1BLDR         System (S.SLDR)	MAST routine	TODINA
MBDC routine         49         JOBOUL           MCBC         52         JOBOUM           MCHA routine         49, 102         JOBPP           MCHON         52         LABEL (edit option)           MCLO routine         47         LABELS (assembly parameter)           MDET routine         47         labels           MDLY routine         47         LABSW           MECH         52         LCV routine           MEXE routine         45         LDRSTK           MFTC         52         LIBE card           MINT routine         46         LIBEND card           MILE         54         LISTSW           MJOB         52         LMAX           MLIS routine         48         load area           MLIS routine         48         load file, system           MLNK         52         Loader           MAP         52         System (S.SLDR)	MATT routine	TOPOU
MCBC	ABDC routine	JOBOU
MCHA routine   49, 102	MCBC 49	JOBOUL 5.
MCHN	ACHA routine	JOBOUM 5
MCLO routine         47         LABEL (edit option)           MDBL         52         LABELS (assembly parameter)           MDET routine         47         labels           MDLY routine         47         LABSW           MECH         52         LCV routine           MENT         52         LDAVL           MEXE routine         45         LDRSTK           MINT routine         46         LIBEND card           MINT routine         46         LIBEND card           MIDIS         54         LISTSW           MILDR         52         LMAX           MILIS routine         48         load file, system           MLNK         52         Loader           MLSU routine         47         IBLDR           MAP         52         System (S.SLDR)	49, 102	JOBPP 5
IDBL         52         LABELS (assembly parameter)           IDET routine         47         labels           IDLY routine         47         LABSW           IECH         52         LCV routine           IENT         52         LDAVL           IEXE routine         45         LDRSTK           IFTC         52         LIBE card           IINT routine         46         LIBEND card           IJIE         54         LISTSW           IJOB         52         LMAX           ILDR         52         LMAX           ILIS routine         48         load area           ILNK         52         Loader           ILSU routine         47         IBLDR           IMAP         52         System (S.SLDR)	ACLO routing	
ABELS (assembly parameter)	MDRI	LABEL (edit option)
IDET routine	IDEL	LABELS (assembly parameter)
### ADEI routine	ADI V routing	labels 5
#ECH	ADLI routine	LABSW
IENT	IEUH 52	LCV routine 7
#EXE routine 45 LDRSTK #FTC 52 LIBE card #INT routine 46 LIBEND card #IJIE 54 LISTSW #IJOB 52 LMAX #ILDR 52 load area #ILIS routine 48 load file, system #ILIS routine 48 Loader #ILSU routine 47 LIBLDR #IBLDR #MAP 52 System (S.SLDR)	1EIN 1 52	LDAVL
IFTC         52         LIBE card           IINT routine         46         LIBEND card           IJIE         54         LISTSW           IJOB         52         LMAX           ILDR         52         load area           ILIS routine         48         load file, system           ILNK         52         Loader           ILSU routine         47         IBLDR           IMAP         52         System (S.SLDR)	1EXE routine	- LDRSTK
AINT routine         46         LIBEND card           AJIE         54         LISTSW           AJOB         52         LMAX           ALDR         52         load area           ALIS routine         48         load file, system           ALNK         52         Loader           ALSU routine         47         IBLDR           AMAP         52         System (S.SL.DR)	MFTC	LIBE card
MJIE       54       LISTSW         MJOB       52       LMAX         MLDR       52       load area         MLIS routine       48       load file, system         MLNK       52       Loader         MLSU routine       47       IBLDR         MMAP       52       System (S.SLDR)	AINT routine	LIBEND card 28
MJOB       52       LMAX         MLDR       52       load area         MLIS routine       48       load file, system         MLNK       52       Loader         MLSU routine       47       IBLDR         MMAP       52       System (S.SLDR)	<i>I</i> JIE54	LISTSW
MLDR         52         load area           MLIS routine         48         load file, system           MLNK         52         Loader           MLSU routine         47         IBLDR           MMAP         52         System (S.SLDR)	AJOB	EMAX
MLIS routine       48       load file, system         MLNK       52       Loader         MLSU routine       47       IBLDR         MMAP       52       System (S.SLDR)	ALDR	load area
MLNK         52         Loader           MLSU routine         47         IBLDR           MMAP         52         System (S.SL.DR)	ALIS routine	load file system
MLSU routine         47         IBLDR           MMAP         52         System (S.SLDR)	ALNK	load file, system
MMAP 52 System (S.SLDR)	ALSU routine	
NEW TITE OF THE CONTROL OF THE CONTR	MMAP 47	Section (CCIDD)
MMON routine	ANALYSI	System (S.SLDR)
MMON routine 45 long form complex		long form complex 63

Return routine		S.SPND
Restart routine	45	S.SPER
replacement using symbolic editing		S.SORG
REMOVE card REPLACE card		S.SLTC S.SLVL
REMARK card	28	S.SLOC
binary program text subroutine, inserting		S.SIDR S.SLDR
elocatable	THE DEVINE THE	S.SHDR
Reload Program	65	S.SFLG
relative addresses and decrementsreload file, system copy and	62	S.SFAZ S.SFBL
eference point entries	61	S.SEND 11
Recognizable Control Card Table	38, 43	S.SDMP
Recall routine, System Monitor	44	S.SDEX
RDNRT eassembly Nucleus		S.SDAT S.SDBG
PWS routine		S.SCUR
PVARJ	54	S.SCSN
ounch, file, system		S.SCMX S.SCOR
Monitor program deck		S.SCLK S.SCMY
files		S.SCKT
Processor (IBJOB)		S.SCIS
priority processing		S.SCCR S.SCDI
preface entry		S.SAVE
PPAVL		S.REW
postprocessor		S.PUTB S.PUTL
PIREL POEDIT		S.PLOC
PIMENT	54	S.PGCT
Phase B (MAP)	69	SOURCE option
Phase A (MAP)		SOURCE option
PDUMP PGNUM		S.OPEN
PCLOSE		S.OAUN
paging	97	S.NAPT snapshot
numbering PAGHD		SLT routine
heading		SLN routine
page		S.IOOP S.JNAM
OVF routine	74	S,IOLS S,IOOP
output file, system		S.IAUN
orders, disk and drum Output Editor		short form complex
operator interrupt test		S.GETL SHARE
ODDLIM		SCETI
OCV routine		S.FEOR
OCLOSM OCT card		S.FBPP
OCLOS	J#	S.FBOA S.FBOU
object-time routines	95	S.FBLD
routines	43	S.FBIN
Nucleus reassembly adjustments		S.FBED S.FBIA
NOSOURCE option	22, 23	S.FBCP
NOMAP option		SEDUN
MXCLK Option		S.CLSE S.CLSL
Multiply subroutine (COBOL) MXBLK option		S.CKPT
MOVPAK routine (COBOL)		SCAN routine (COBOL)
MODIFY card		SCA routine scale subroutine (COBOL)
MESLIM MIN option		S.BSR
MAP Symbolic Update Program		S.BSLA
Macro Assembly Program	67	S.BDMP
error exits		DWD waiting
machine-oriented subroutines		round or truncate subroutine (COBOL) RUPSW

그리는 경우다 그런 시간 전에 살 말한 사람이, 난 살이 있어요? 관련하다면 바로 화를 다 살려.	
S.SPRP	42 S.XLTP 41
S.SRCC	38 S.XOVA
S.SRET	37 S.XOVD
S.SRPT	37 S.XPRT 37
S.SRST	37 S.XPSE 37
S.SRVP	37 S.XSCH
S.SRUS	40 C VCNC
S.SSBC	20 C. V.T.D.T.
2 2 2 2 1	56 S.XTD1 41 41 S.XTPS 41
C CCNC	41 S.XUCV
e ceurt	39 symbolic 37
C CLID C	38 channel assignment
CCITAII	38 editing replacement using
CCVCV	
	42 master file
CCVAV	42 unit reference
C CVCV	42 Units Table 42
S SYCZ	42 Update Program 98
	42 SYSHD 97
standard entry	42 System Editor
standard entry S.TPBF	62 System Library 5
S.TPBF storage	37 System Monitor
	51STUS 54
Allocaton	
Allocator	Table of Contents
requirements, FORTRAN subroutine subroutine	75 TESTIO
	time of day
Dependence Table	91 TMAX
insertion	はまた ・・ 2000 元 700 こうさいけい サイト・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
Library	transfer words
Name Table	at trap words
SUBHD	71 text
subtract subroutine (COBOL)	39 T2 text
Supervisor	15 Unit
SUPRES	79
S.WEF	99
Sw 1 routine	74 Undata Promise C. 1 1
5.AAG1	o <del>rd</del> er i de la companya della companya de la companya de la companya della compa
S.XCPS	Work unite
S.XDAC	
S.ADVA	70 Touring
SXIIVII	AEM routine
	74 XIT routine